Vladimir Britanak · K. R. Rao

Cosine-/Sine-Modulated Filter Banks

General Properties, Fast Algorithms and Integer Approximations

Springer

Cosine-/Sine-Modulated Filter Banks

Vladimir Britanak • K.R. Rao

Cosine-/Sine-Modulated Filter Banks

General Properties, Fast Algorithms and Integer Approximations

Springer

Vladimir Britanak
Institute of Informatics
Slovak Academy of Sciences
Bratislava, Slovakia

K.R. Rao
The University of Texas at Arlington
Arlington, TX, USA

ISBN 978-3-319-61078-8 ISBN 978-3-319-61080-1 (eBook)
DOI 10.1007/978-3-319-61080-1

Library of Congress Control Number: 2017943966

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Analysis and synthesis cosine-/sine-modulated filter banks are perfect reconstruction filter banks that are used for the time-to-frequency transformation of an audio data block, and vice versa, many sub-band/transform-based schemes for high-quality lossy/lossless compression of digital audio signals. These include the modified discrete cosine and sine transforms, the modulated lapped transforms, the extended lapped transforms, the low delay modified discrete cosine transform, the modulated complex lapped transform, and various forms of (low delay) complex exponential-modulated and real-valued cosine-modulated quadrature mirror filter banks. The perfect reconstruction cosine-/sine-modulated filter banks are fundamental processing components in many state-of-the-art international audio coding standards and in proprietary audio compression algorithms, broadcasting/speech/data communication codecs, and open-source royalty free audio/speech codecs. In general, the computation of the complete perfect reconstruction analysis and synthesis filter banks is the most time-consuming operation in audio/speech coding schemes, and therefore, the fast algorithms for their efficient real-time hardware and software implementation are very important.

Although many excellent (text) books have been published up to now (see references in the introductory chapter), generally they are almost all devoted to the theory and design of near-perfect and perfect reconstruction quadrature mirror filter and modulated filter banks, to the theory of orthogonal lapped transforms, to the detailed description of audio coding methods/algorithms/technologies (psychoacoustics principles and models, quantization, and perceptual audio coding strategies), and to the detailed description of audio coding standards and proprietary audio compression algorithms and their audio coding applications. However, in these books, frequently the discussion about fast algorithms for their efficient implementation is often limited, or they are discussed only marginally. On the other hand, the research interest and activities in the efficient implementations of perfect reconstruction cosine-/sine-modulated filter banks have much increased recently, mainly owing to the existence of their many audio coding applications appearing in

the consumer and professional electronics market (portable players, mobile phones, and digital multimedia communication systems). This was the main motivation to prepare the edition of this book, and the authors strongly believe that it will fill up this gap.

This book is devoted essentially and exclusively to the theory and design of fast algorithms for the efficient implementation of perfect reconstruction cosine-/sine-modulated filter banks as well as to the theory of algorithm complexity. It summarizes the research results achieved by the research community over three decades in this hot research topic. The book covers various algorithmic developments in the cosine-/sine-modulated filter banks including their general mathematical properties in the time and frequency domains, their (block) matrix representations, their fast algorithms employed in modern transform-based coding technologies, and various local and global methods to their integer approximation (integer-approximate cosine-/sine-modulated filter banks), being recently innovative transform-based technologies for lossless audio coding.

The book *Cosine-/Sine-Modulated Filter Banks: General Properties, Fast Algorithms and Integer Approximations* is aimed at students, engineers, researchers, and scientists at research institutes, universities, and companies who are interested in theoretical aspects (origin and general mathematical properties) and practical aspects (fast algorithms and methods to integer approximation). The fervent hopes and aspirations of the authors are that the book will serve both as an excellent reference to perfect reconstruction cosine-/sine-modulated filter banks and as an incentive/inspiration for further advanced research.

Bratislava, Slovakia — Vladimir Britanak
Arlington, TX, USA — K.R. Rao
April 2017

Acknowledgments

Many many hours and efforts have been spent behind the computer in preparing this manuscript, in analytical derivation of fast algorithms and various sparse block matrix factorizations, in drawing the figures of signal flow graphs, and in verifying the fast algorithms by computer programs. This book is the result of the long-term association of two authors, V. Britanak and K.R. Rao. Special thanks go to their respective families for their support and understanding. We appreciate also the continued encouragement and many helpful suggestions of our friends and colleagues in the departments of the institute and universities.

The second author likes to acknowledge the support provided in various forms by Dr. Peter Crouch, Dean, COE; Dr. Jean-Pierre Bardet, Former Dean, College of Engineering (COE); Dr. J.W. Bredow, Chair, Department of Electrical Engineering; and all colleagues in UTA. The leading author wishes to thank his daughters, Zuzka and Katka, and son Tomas for the understanding and encouragement during the years of the preparation of this book. Many thanks belong to a colleague and friend Dr. Jan Glasa from the Institute of Informatics, Slovak Academy of Sciences, for his patience, encouragement, and support. Finally, it is also appropriate to acknowledge here the financial support provided by the Slovak Scientific Agency VEGA, project No. 2/0184/14 and No. 2/0165/17.

The authors have been honored to have worked with Springer Science + Business Media B.V. on this project. The encouragement, support, and understanding provided by the publishing editorial staff at the Applied Sciences Department, and in particular, Senior Editor Mary E. James and Editorial Assistant Zoe Kennedy are greatly appreciated.

Contents

List of Acronyms

1-D	One-dimensional
2-D	Two-dimensional
AAC	Advanced Audio Coding
AAC-ELD	Enhanced Low Delay AAC
AAC-LC	Low Complexity AAC
AAC-LD	Low Delay AAC
AACPRO	SBR-Enhanced version of MPEG-AAC
AAZ	Advanced Audio Zip
AC-3	Dolby Digital AC-3 audio coding
ACELP	Algebraic code-excited linear prediction (technology)
AHT	Adaptive hybrid transform
AMR-WB+	Extended AMR-WB (speech codec)
ARM	ARM processor architecture
ASIC	Application-specific integrated circuit
ATRAC	Adaptive TRansform Acoustic Coding
ATSC	Advanced Television Systems Committee
CELT	Constrained energy lapped transform
CDFT	Complex DFT
CLDFB	Complex Low Delay Filter Bank
DAB	Digital audio broadcasting
DAB+	Advanced digital audio broadcasting
DCT	Discrete cosine transform
DFT	Discrete Fourier transform
DHT	Discrete Hartley transform
DIF	Decimation-in-frequency
DIT	Decimation-in-time
DLU	DLU matrix factorization or DLU computational structure
DMB	Digital multimedia broadcasting
DRM	Digital Radio Mondiale
DSC HDTV	Digital spectrum compatible HDTV
DSP	Digital signal processor

DST	Discrete sine transform
DTT	Discrete trigonometric transform
DVB	Digital video broadcasting
DVD	Digital video disc
E-AC-3	Enhanced Dolby Digital AC-3 audio coding system
ELD	Enhanced Low Delay
ELT	Extended lapped transform
EPAC	Enhanced PAC
eSBR	Enhanced SBR
EVRC-WB	Enhanced Variable Rate Codec–Wideband (speech codec)
EV-VBR	Embedded variable bit rate (speech codec)
FBMC	Filter bank multicarrier (transmission system)
FFT	Fast Fourier transform
FHT	Fast Hartley Transform
FPGA	Field-programmable gate array
GDCT	Generalized discrete cosine transform
GDFT	Generalized discrete Fourier transform
GDHT	Generalized discrete Hartley transform
GDST	Generalized discrete sine transform
GenLOT	Generalized lapped orthogonal transform
HD-AAC	High-definition AAC
HDTV	High-definition television
HE-AAC	High-efficiency AAC
HQ-LD	High-quality low delay
HQ-SBR	High-quality SBR
IEC	International Electrotechnical Commission
INTDCT	Integer discrete cosine transform
INTFFT	Integer fast Fourier transform
INTMDCT	Integer-modified discrete cosine transform
INTMLT	Integer-modulated lapped transform
IP	Internet Protocol
ISO	International Standards Organization
ITU-T	International Telecommunication Union–Telecommunications sector
KBD	Kaiser-Bessel-derived windowing function
LC	Low complexity
LD	Low delay
LD-MDCT	Low Delay MDCT
LD-SBR	Low Delay SBR
LDU	(Block) LDU matrix factorization
LOT	Lapped orthogonal transform
LP	Linear prediction
LP-LD	Low power low delay
LP-SBR	Low power SBR
LT	Lapped transform

LTAC	Lossless transform-based audio coding
LU	(Block) LU matrix factorization
LUL	LUL matrix factorization or LUL computational structure
MCLT	Modulated complex lapped transform
MDCS	Multi-dimensional computational structure
MDCT	Modified Discrete Cosine Transform
MDST	Modified discrete sine transform
MLBT	Modulated lapped biorthogonal transform
MLT	Modulated lapped transform
MP3	MPEG-1/2 Layer 3
MP3PRO	SBR-enhanced version of MP3
MPAC	Multichannel PAC
MPEG	Moving Picture Experts Group
MPEG AAC	MPEG Advanced Audio Coding
MPEG NBS/AAC	MPEG Non-backward Compatible AAC
MPEG-4 ALS	MPEG-4 Audio Lossless Coding
MPEG-4 SLS	MPEG-4 Scalable to Lossless Audio Coding
MSE	Minimum Square Error
NBC/AAC	Non-backward Compatible AAC
NMCLT	Nonuniform modulated complex lapped transform
NMLBT	Nonuniform modulated lapped biorthogonal transform
O^2-DFT	Odd-time odd-frequency DFT
OFDM	Orthogonal frequency division multiplexing (transmission system)
PAC	Perceptual audio coder
PFA	Prime factor (decomposition) algorithm
PLEAC	Progressive to lossless embedded audio coder
PLUS	(Block) PLUS matrix factorization
PQF	Polyphase quadrature filter
PQMF	Polyphase QMF
Pseudo-LOT	Pseudolapped orthogonal transform
Pseudo-QMF	Pseudo quadrature mirror filter
QMF	Quadrature mirror filter
QR	(Block) QR matrix factorization
RDFT	Real-valued DFT
RISC	Reduced instruction set computing
RMSE	Root-mean-square error
RVFFT	Real-valued FFT
SBR	Spectral Band Replication compression technology
SDCT	Scaled DCT
SDDS	Sony Dynamic Digital Sound
SLS	Scalable to lossless coding solution
SOC	Systems on a Chip
SOPOT	Sum-Of-Powers-Of-Two
SSB	Single-sideband

SSR	Scalable Sampling Rate
SVD	Singular value decomposition
TDA	Time domain aliasing
TDAC	Time domain aliasing cancellation
TW-MDCT	Time-warped MDCT
ULU ULU	Matrix factorization or ULU computational structure
USAC	Unified Speech and Audio Coding
VCD	Video CD
VLSI	Very-large-scale integration
VoiceIP	Voice over Internet
WFTA	Winograd-Fourier transform algorithm

Chapter 1
Cosine/Sine-Modulated Analysis/Synthesis Filter Banks

1.1 Introduction

One of the topics in multi-rate digital signal processing is the theory and design of M-band (or M-channel) analysis and synthesis quadrature mirror filter (QMF) banks for sub-band signal decomposition and coding [3–5, 9–11]. They are also called M-band maximally decimated critically sampled QMF banks. The analysis QMF bank consists of M uniform and equally spaced channel filters to decompose the input signal into M sub-band signals. The synthesis QMF bank consists of channel filters to reconstruct the original signal exactly from sub-band signals, or to recover a signal which is nearly perfect approximation of the original signal. Historically, discovering the 2-band QMF banks [15] in 1976 stimulated and started research activities leading to extending the theory of near-perfect and perfect reconstruction QMF banks for arbitrary number of sub-bands, to developing a family of near-perfect modulated filter banks (or pseudo-QMF banks) and perfect reconstruction modulated filter banks based on the concept of time domain aliasing cancellation [16–144].

Among studied pseudo-OMF banks and perfect reconstruction modulated filter banks, the cosine-modulated filter banks gained widespread attention. It is not surprising, they have many attractive features: a simple formulation and structure, analysis and synthesis filters are of equal length, and in particular, they have a fast implementation with low computational complexity. Indeed, in M-band pseudo-OMF or perfect reconstruction cosine-modulated filter banks, the analysis and synthesis filters are equally spaced, and they are cosine modulated versions of a low-pass prototype filter. In a simplest common form, the set of analysis and the set of synthesis filters are, respectively, expressed as

$$h_{k,n} = w_n\, c_{k,n},$$
$$g_{k,n} = w_n\, c_{k,n}, \qquad k = 0, 1, \ldots, M-1, \qquad n = 0, 1, \ldots, L-1,$$

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1_1

where w_n is the low-pass prototype filter or windowing function of the length $L\,(L > M)$, M is the number of sub-bands, and $c_{k,n}$ is the cosine modulation. Obviously, the synthesis filters $h_{k,n}$ are related to the analysis filters $g_{k,n}$ by

$$g_{k,n} = h_{k,L-1-n},$$

i.e., the synthesis filters are time reversed versions of the analysis filters. If the windowing function satisfies certain constraints, the length $L = 2M$, and the cosine modulation is chosen to be, for example, $c_{k,n} = \cos\left[\frac{\pi}{M}(k+\frac{1}{2})(n+\frac{M}{2}+\frac{1}{2})\right]$, then M-band cosine-modulated filter bank achieves the perfect reconstruction. Importantly, such cosine-modulated filter bank has an efficient implementation.

This book is devoted essentially and exclusively to the theory and design of fast algorithms for the efficient implementation of perfect reconstruction cosine/sine-modulated filter banks used/employed in modern transform-based audio coding technologies, and to the theory of algorithm complexity. It covers and summarizes the research results achieved by the research community over three decades in this hot research topic.

1.2 Additional References

An extensive list of references [1–144] has been appended to this introductory chapter (almost not all are cited in subsequent chapters). No claim for completeness of this list is made. Besides books devoted to multi-rate digital signal processing [1–11], books on discrete cosine/sine (DCT/DST) transforms [12–14], the appended references [15–144] reflect the retrospective research efforts and developments in the theory and design of QMF filter banks.

1.3 Organization of the Book

The book is organized in terms of chapters starting with this introductory chapter. Each chapter begins with Abstract and contains its own list of references.

In *Chap. 2* perceptual transform-based audio coding schemes developed up to now are briefly reviewed including the family of ISO/IEC MPEG audio coding standards, proprietary audio compression algorithms, broadcasting/speech/data communication codecs, and open-free, patent royalty-free audio/speech codecs. The discussion is concentrated especially on adopted near-perfect QMF (pseudo-QMF) and perfect reconstruction cosine/sine-modulated filter banks, processing methods, and specified transform block sizes.

The evenly and oddly stacked modified discrete cosine transform (MDCT) and the corresponding modified discrete sine transform (MDST), the modulated lapped transform (MLT), the extended lapped transforms (ELTs), and their biorthogonal

versions are real-valued cosine/sine-modulated filter banks satisfying the perfect reconstruction property. The modulated complex lapped transform (MCLT) is the complex-valued filter bank whose real part is the MLT or equivalently, the oddly stacked MDCT, and the imaginary part is the oddly stacked MDST. In *Chap. 3*, definitions, general properties in the time and frequency domain, and matrix representations of the MDCT/MDST, MLT, ELT, and MCLT filter banks are presented. In order to an analysis/synthesis filter bank be perfect reconstruction, the necessary and sufficient conditions imposed on the analysis and synthesis windowing functions play an important role. Therefore, additionally the windowing procedure and perfect reconstruction (biorthogonal) conditions in the case of identical and (nonidentical) analysis and synthesis windowing functions are discussed. Further, design of a windowing function including definitions of commonly windowing functions used in audio coding applications, adaptive switching of transform block sizes and windowing functions, and general perfect reconstruction conditions for the ELT filter bank with multiple overlapping factor both for the orthogonal and biorthogonal cases are derived and/or discussed in detail.

The MDCT/MDST, MLT, ELT, and MCLT are fundamental processing components for the time-to-frequency transformation of an audio data block in many audio coding schemes for high quality audio compression. Since the computations of cosine/sine-modulated filter banks are the most time-consuming operations in audio coding schemes, the crucial aspect for their applicability is the existence of fast algorithms that allow their efficient software/hardware implementation compared to the direct implementation via their corresponding analytical forms. In *Chap. 4*, radix-2, even-length and mixed-radix fast algorithms for the efficient implementation of the forward/backward evenly stacked MDCT/MDST, oddly stacked MDCT/MDST, MLT, ELT, and MCLT block transforms are presented. The emphasis is imposed particularly on basic steps, various tricks (trigonometric and algebraic) and approaches leading to the derivation of final formulae of a fast algorithm. For each fast algorithm complete formulae or a sparse block matrix factorization of transform matrix, a corresponding generalized signal flow graph, the total computational complexity, and a possible structural simplification of the algorithm are presented.

The MPEG-1/2 audio coding standard for the time-to-frequency transformation of an audio signal and vice versa, in layers I and II has adopted the pseudo-QMF banks. In layer III (known as MP3) it has additionally adopted the MLT or MDCT associated with the sine windowing function. *Chapter 5* describes and compares various efficient implementations of the forward and backward MLT (MDCT) tailored directly on MP3 audio including the efficient implementation of pseudo-QMF banks for completeness. The efficient MLT (MDCT) implementations are discussed in the context of complete (fast) analysis/synthesis MLT (MDCT) filter banks in the MP3 encoder and decoder. In general, for each efficient forward/backward MLT (MDCT) block transforms implementation are presented: Complete formulae or sparse (block) matrix factorizations, the corresponding signal flow graph for short audio block and the total arithmetic complexity as well as the useful comments related to improving the arithmetic complexity and a possible

structural simplification of the algorithm. Finally, the fast analysis and synthesis MLT (MDCT) filter banks for MP3 encoder and decoder are discussed in detail.

The Dolby Digital (AC-3) and the Dolby Digital Plus or Enhanced AC-3 (E-AC-3) audio coding standards developed by the Dolby Labs are currently the key enabling technologies for high-quality compression of digital audio signals. For the time/frequency transformation of an audio data block, and vice versa, the AC-3 and E-AC-3 have adopted the oddly stacked MDCT. The AC-3 besides the MDCT defines additional two variants of cosine-modulated filter banks called the first and second short transforms. Moreover, the current AC-3 and E-AC-3 codecs for better spectral estimation and for phase angle adjustment have adopted the oddly stacked MDST which together with the MDCT forms a complex MCLT filter bank. *Chapter 6* is devoted to the perfect reconstruction cosine/sine-modulated filter banks used in the Dolby AC-3 and E-AC-3 codecs. The definitions of the analysis/synthesis AC-3 filter banks, their general symmetry properties both in the time and frequency domains, and their efficient unified implementations are presented. Matrix representations of AC-3 filter banks, their properties and relations among transform (sub-)matrices provide the basis to derive relations between the frequency coefficients and the time domain aliasing data sequences of AC-3 transforms, and in particular, the basis for derivation of a fast algorithm for conversion of frequency coefficients of AC-3 transforms directly in the frequency domain. Finally, conversion methods of the MDCT to MDST frequency coefficients directly in the frequency domain are discussed.

Spectral Band Replication (SBR) is an enhancement compression technology which significantly improves the compression efficiency of perceptual audio and speech coding schemes. Central to the operation of standard SBR and low delay version of SBR are dedicated complex exponential-modulated and real-valued cosine-modulated QMF banks as the basic mathematical tools to analyze and synthesize audio signals. *Chapter 7* presents the complete unified efficient implementations of complex exponential-modulated and real-valued cosine-modulated QMF banks used both in the standard SBR and low delay SBR encoder and decoder. For each QMF bank, definition in its equivalent block transform with a common parameter M representing the number of sub-bands, its general symmetry property in the frequency or time domain, and the derivation of a fast algorithm for its efficient implementation are presented. All the fast algorithms are analyzed in detail in terms of the arithmetic complexity, regularity, and structural simplicity for a potential real-time low-cost implementation in hardware or software.

In order to achieve low algorithmic delay for bidirectional communication systems, the MPEG-4 Advanced Audio Coding—Enhanced Low Delay audio coding standard has adopted a perfect reconstruction analysis and synthesis low delay MDCT (LD-MDCT) filter banks. In *Chap. 8*, definitions of the analysis and synthesis LD-MDCT filter banks, their general symmetry properties in the time and frequency domains, relations between the LD-MDCT and the oddly stacked MDCT

both in the analytical forms and the equivalent matrix representations, and efficient implementations of the even-length analysis/synthesis low delay MDCT filter banks are discussed in detail. For each fast LD-MDCT algorithm the complete formulae are derived. All the fast even-length LD-MDCT algorithms are investigated and compared in terms of arithmetic complexity and structural simplicity.

Enabling technology for transform-based lossless audio coding is the integer transform. Integer transform is a transform which maps integers to integers by a reversible (invertible) way so that it preserves all mathematical properties of the original real-valued transform, such as perfect reconstruction, energy compaction property, and fast algorithm. Indeed, the IntMDCT or IntMLT enabled to design and implement this innovative coding technology for scalable lossy to lossless audio coding. In *Chap. 9*, the local and global methods to integer approximation of perfect reconstruction cosine/sine-modulated filter banks and cosine-modulated QMF banks are discussed in detail. They are based on computational methods of linear algebra, matrix theory and matrix computations, and in particular, on the (block) matrix decompositions. In fact, the scalar and block matrix decompositions are powerful mathematical tools to construct the reversible (invertible) integer transforms.

All chapters end with a summary, problems/exercises, and references. Problems/exercises reflect the contents of the corresponding chapters and are intended for the reader in terms of refresh/review/reinforce their contents. Extensive definitions, principles, properties, signal flow graphs, derivations, and examples are provided throughout the book for proper understanding of the strengths and shortcomings of the spectrum of perfect reconstruction cosine/sine-modulated filter banks.

1.4 Appendices

Appendices A through G review and present the important mathematical basics from matrix theory and linear algebra (Appendix A), definition and symmetry properties of odd-time odd-frequency DFT (Appendix B), fast DCT/DST computational structures (Appendix C), optimized efficient short odd-length complex DFT, real-valued DFT and (S)DCT modules (Appendix D), optimized efficient short-length forward/backward MDCT modules (Appendix E), efficient implementations of Givens-Jacobi rotations (Appendix F), and finally, definitions of symmetric/anti-symmetric and periodic/anti-periodic sequences (Appendix G).

In general, Appendices provide both theoretical basics necessary for the derivation of fast algorithms and also practical tools, fast DCT/DST computational structures efficient and optimized efficient short-length computational modules, and tools necessary for completing efficient forward/backward MDCT/MDST, MLT, ELT, and MCLT implementations.

1.5 References

To retain the connectivity among the chapters of the book as much as possible, each chapter in the book includes its own list of references related to the discussed subject. Therefore, some references may appear in the lists of references of chapters more than once.

References

1. A.N. Akansu, R.A. Haddad, *Multi-Resolution Signal Decomposition: Transforms, Subbands, Wavelets*, Chaps. 3 and 4. 2nd edn. (Academic, San Diego, CA, 2001)
2. M. Bosi, R.E. Goldberg, *Introduction to Digital Audio Coding and Standards* (Springer Science+Business Media, New York, 2003)
3. M. Bellanger, *Digital Processing of Signals: Theory and Practice*, 2nd edn. (Wiley, Chichester, 1989)
4. R.E. Crochiere, L.R. Rabiner, *Multirate Digital Signal Processing* (Prentice-Hall, Englewood Cliffs, NJ, 1983)
5. N.J. Fliege, *Multirate Digital Signal Processing: Multirate Systems, Filter Banks, Wavelets* (Wiley, Chichester, 1994)
6. H.S. Malvar, *Signal Processing with Lapped Transforms* (Artech House, Norwood, MA, 1992)
7. V.K. Madisetti, D.B. Williams (eds.), *The Digital Signal Processing Handbook* (CRC, IEEE, Boca Raton, FL, 1998)
8. A. Spanias, T. Painter, V. Atti, *Audio Signal Processing and Coding* (Wiley-Interscience, NJ, 2007)
9. G. Strang, T. Nguyen, *Wavelets and Filter Banks*, Revised edn. (Wellesley, Cambridge Press, Wellesley, MA, 1997)
10. P.P. Vaidyanathan, *Multirate Systems and Filter Banks* (Prentice-Hall, Englewood Cliffs, NJ, 1993)
11. M. Vetterli, J. Kovačević, *Wavelets and Subband Coding* (Prentice-Hall, Englewood Cliffs, NJ, 1995)

(Text)Books on DCTs and DSTs

12. G. Bi, Y. Zeng, *Transforms and Fast Algorithms for Signal Analysis and Representations* (Birkhäuser, Boston, 2004)
13. V. Britanak, P.C. Yip, K.R. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations* (Academic, Elsevier, Amsterdam, 2007)
14. K.R. Rao, P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications* (Academic, Boston, 1990)

Theory and Design of Two-Channel QMF Banks

15. M. Belanger, G. Bonnerot, M. Coudreuse, Digital filtering by polyphase network: application to sample rate alteration and filter banks. IEEE Trans. Acoust. Speech Signal Process. **ASSP-24**(2), 109–114 (1976)

16. S.O. Aase, T.A. Ramstad, Parallel FIR filter banks for robust subband image coding, in *Proceedings of the IEEE ICASSP'93*, vol. V, Minneapolis, MN, April 1993, pp. 566–569
17. J.L. Arrwood, M.J.T. Smith, Exact reconstruction analysis/synthesis filter banks with time-varying filters, in *Proceedings of the IEEE ICASSP'93*, vol. III, Minneapolis, MN, April 1993, pp. 233–236
18. T.P. Barnwell, M.J.T. Smith, Filter banks for analysis-reconstruction systems: a tutorial, in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'90)*, New Orleans, LA, April–May 1990, pp. 1999–2003
19. S.C. Chan, The generalized lapped transform (GLT) for sub-band coding applications, in *Proceedings of the IEEE ICASSP'95*, Detroit, MI, April 1995, pp. 1508–1511
20. C.-K. Chen, J.-H. Lee, Design of quadrature mirror filters with linear phase in the frequency domain. IEEE Trans. Circuits Syst. II Analog Digit. Signal Process. **39**(9), 593–605 (1992)
21. S.C. Chan, Quadrature modulated filter banks, in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'94)*, vol. 2, London, May 1994, pp. 501–504
22. S. Cheung, J.S. Lim, Incorporation of biorthogonality into lapped transforms for audio compression, in *Proceedings of the IEEE ICASSP'95*, Detroit, MI, April 1995, pp. 3079–3082
23. P.L. Chu, Quadrature mirror filter design for an arbitrary number of equal bandwidth channels. IEEE Trans. Acoust. Speech Signal Process. **ASSP-33**(1), 203–218 (1985)
24. C.D. Creusere, S.K. Mitra, A simple method for designing high-quality prototype filters for M-band pseudo QMF banks. IEEE Trans. Signal Process. **43**(4), (1995), 1005–1007
25. R.V. Cox, The design of uniformly and nonuniformly spaced pseudoquadrature mirror filters. IEEE Trans. Acoust. Speech Signal Process. **ASSP-34**(5), 1090–1096 (1986)
26. A. Croisier, D. Esteban, C. Galand, Perfect channel splitting by use of interpolation/decimation/tree decomposition techniques, in *Proceedings of the International Conference on Information Science, Circuits and Systems*, Patras, August 1976, pp. 443–446
27. F. Cruz-Roldán, P. Amo-López, P. Martín-Martín, F. López-Ferreras, Alternating analysis and synthesis filters: a new pseudo-QMF bank. Digital Signal Process. **11**(4), 329–345 (2001)
28. F. Cruz-Roldán, F. López-Ferreras, P. Amo-López, J.D. Osés del Campo, Arbitrary-length spectral factor applied to the design of pseudo-QMF cosine-modulated filter banks. IEEE Trans. Circuits Syst. II Analog Digit. Signal Process. **48**(3), 321–325 (2001)
29. R.L. de Queiroz, K.R. Rao, Time-varying lapped transforms and wavelet packets. IEEE Trans. Signal Process. **41**(12), 3293–3305 (1993)
30. R.L. de Queiroz, T.Q. Nguyen, K.R. Rao, Generalized lapped orthogonal transforms. Electron. Lett. **30**(2), 107–108 (1994)
31. R.L. de Queiroz, T.Q. Nguyen, K.R. Rao, The GenLOT: generalized linear-phase lapped orthogonal transform. IEEE Trans. Signal Process. **44**(3), 497–507 (1996)
32. Z. Doganata, P.P. Vaidyanathan, T.Q. Nguyen, General synthesis procedures for FIR lossless transfer matrices, for perfect reconstruction multirate filter banks applications. IEEE Trans. Acoust. Speech Signal Process. **36**(10), 1561–1574 (1988)
33. D. Esteban, C. Galand, Application of quadrature mirror filters to split band voice coding schemes, in *Proceedings of the IEEE ICASSP'77*, Hartford, CT, May 1977, pp. 191–195
34. C.R. Galand, H.J. Nussbaumer, New quadrature mirror filter structures. IEEE Trans. Acoust. Speech Signal Process. **ASSP-32**(3), 522–531 (1984)
35. C.R. Galand, H.J. Nussbaumer, Quadrature mirror filters with perfect reconstruction and reduced computational complexity, in *Proceedings of the IEEE ICASSP'85*, Tampa, FL, March 1985, pp. 525–529
36. X.G. Gao, X.D. Wang, Z.Y. He, Cosine-modulated FIR filter banks with linear phase and paraunitary properties. Electron. Lett. **32**(8), 723–724 (1996)
37. R. Gluth, A unified approach to transform-based FIR filter-banks with special regard to perfect reconstruction systems, in *Proceedings of the IEEE ICASSP'93*, vol. III, Minneapolis, MN, April 1993, pp. 157–160

38. R. Gluth, U. Heute, Analysis/synthesis filter banks based on generalized sinusoidal transforms with an application to speech coding, in *Proceedings of the 6th European Signal Processing Conference (EUSIPCO'92)*, vol. 1, Brussels, August 1992, pp. 215–218
39. R.A. Gopinath, C.S. Burrus, Some results in the theory of modulated filter banks and modulated wavelet tight frames. Appl. Comput. Harmon. Anal. **2**(4), 303–326 (1995)
40. R.A. Gopinath, Factorization approach to time-varying filter banks and wavelets, in *Proceedings of the IEEE ICASSP'94*, vol. III, Adelaide, April 1994, pp. 109–112
41. R.A. Gopinath, C.S. Burrus, Theory of modulated filter banks and modulated wavelet tight frames, in *Proceedings of the IEEE ICASSP'93*, vol. III, Minneapolis, MN, April 1993, pp. 169–172
42. R.A. Gopinath, Modulated filter banks and wavelets: a general unified theory, in *Proceedings of the IEEE ICASSP'96*, vol. III, Atlanta, GA, May 1996, pp. 1586–1589
43. R.A. Gopinath, C.S. Burrus, Factorization approach to unitary time-varying filter bank trees and wavelets. IEEE Trans. Signal Process. **43**(3), 666–680 (1995)
44. P.N. Heller, T. Karp, T.Q. Nguyen, A general formulation of modulated filter banks, IEEE Trans. Signal Process. **47**(4), 986–1002 (1999)
45. C. Herley, M. Vetterli, Orthogonal time-varying filter banks and wavelet packets. IEEE Trans. Signal Process. **42**(10), 2650–2663 (1994)
46. C. Herley, Boundary filters for finite-length signals and time-varying filter banks. IEEE Trans. Circuits Syst. II Analog Digit. Signal Process. **42**(2), 102–114 (1995)
47. P.H. Hoang, P.P. Vaidyanathan, Nonuniform multirate filter banks: theory and design, in *Proceedings of the IEEE ICASSP'89*, Glasgow, May 1989, pp. 371–374
48. V.K. Jain, R.E. Chrochiere, A novel approach to the design of analysis/synthesis filter banks, in *Proceedings of the IEEE ICASSP'83*, Boston, MA, April 1983, pp. 228–231
49. V.K. Jain, R.E. Chrochiere, Quadrature mirror filter design in the time domain. IEEE Trans. Acoust. Speech Signal Process. **ASSP-32**(2), 353–361 (1984)
50. A. Jain, R. Saxena, S.C. Saxena, An improved and simplified design of cosine-modulated pseudo-QMF filterbanks. Digital Signal Process. **16**(3), 225–232 (2006)
51. J.D. Johnston, A filter family designed for use in quadrature mirror filter banks, in *Proceedings of the IEEE ICASSP'80*, Denver, CO, April 1980, pp. 291–294
52. T. Karp, A. Mertins, G. Schuller, Efficient biorthogonal cosine-modulated filter banks. Signal Process. **81**(5), 997–1016 (2001)
53. R.D. Koilpillai, P.P. Vaidyanathan, A new approach to the design of FIR perfect reconstruction QMF banks, in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'90)*, New Orleans, LA, April–May 1990, pp. 125–128
54. R.D. Koilpillai, P.P. Vaidyanathan, Cosine-modulated FIR filter banks satisfying perfect reconstruction. IEEE Trans. Signal Process. **40**(4), 770–783 (1992)
55. R.D. Koilpillai, P.P. Vaidyanathan, A spectral factorization approach to pseudo-QMF design. IEEE Trans. Signal Process. **41**(1), 82–92 (1993)
56. J. Kovačević, M. Vetterli, Time-varying modulated lapped transforms, in *Proceedings of the 27th Asilomar Conference on Signals, Systems and Computers*, vol. 1, Pacific Grove, CA, November 1993, pp. 481–485
57. R.D. Koilpillai, P.P. Vaidyanathan, New results on cosine-modulated FIR filter banks satisfying perfect reconstruction, in *Proceedings of the IEEE ICASSP'91*, Toronto, May 1991, pp. 1793–1796
58. Y.P. Lin, P.P. Vaidyanathan, Linear phase cosine modulated maximally decimated filter banks with perfect reconstruction. IEEE Trans. Signal Process. **43**(11), 2525–2539 (1995)
59. H.S. Malvar, Reduction of blocking effects in image coding with a lapped orthogonal transform, in *Proceedings of the IEEE ICASSP'88*, Glasgow, April 1988, pp. 781–784
60. H.S. Malvar, The LOT: a link between block transform coding and multirate filter banks, in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'88)*, Espoo, June 1988, pp. 835–838
61. H.S. Malvar, Pseudolapped orthogonal transform. Electron. Lett. **25**(5), 312–314 (1989)

62. H.S. Malvar, Lapped transforms for efficient transform/sub-band coding. IEEE Trans. Acoust. Speech Signal Process. **ASSP-38**(6), 969–978 (1990)
63. H.S. Malvar, Modulated QMF filter banks with perfect reconstruction. Electron. Lett. **26**(13), 906–907 (1990)
64. H.S. Malvar, Lapped biorthogonal transforms for transform coding with reduced blocking and ringing artifacts, in *Proceedings of the IEEE ICASSP'97*, vol. 3, Munich, April 1997, pp. 2421–2424
65. H.S. Malvar, Biorthogonal and nonuniform lapped transforms for transform coding with reduced blocking and ringing artifacts. IEEE Trans. Signal Process. **46**(4), 1043–1053 (1998)
66. H. Malvar, A modulated complex lapped transform and its applications to audio processing, in *Proceedings of the IEEE ICASSP'99*, Phoenix, AR, May 1999, pp. 1421–1424
67. H.S. Malvar, D.H. Staelin, The LOT: transform coding without blocking effects. IEEE Trans. Acoust. Speech Signal Process. **37**(4), 553–559 (1989)
68. J. Masson, Z. Picel, Flexible design of computationally efficient near perfect QMF filter banks, in *Proceedings of the IEEE ICASSP'85*, Tampa, FL, March 1985, pp. 541–544
69. J. Mau, Computationally efficient pseudo QMF filter bank for a multi-compatible HDTV codec, in *Proceedings of the IEEE ICASSP'91*, Toronto, May 1991, pp. 2849–2852
70. J. Mau, Perfect reconstruction modulated filter banks, in *Proceedings of the IEEE ICASSP'92*, vol. IV, San Francisco, CA, April 1992, pp. 273–276
71. J. Mau, Perfect reconstruction modulated filter banks: fast algorithms and attractive new properties, in *Proceedings of the IEEE ICASSP'93*, vol. III, Minneapolis, MN, April 1993, pp. 225–228
72. J. Mau, Regular M-band modulated orthogonal transforms, in *Proceedings of the IEEE ICASSP'94*, vol. III, Adelaide, April 1994, pp. 125–128
73. J. Mau, J. Valot, D. Minaud, Time-varying orthogonal filter banks without transient filters, in *Proceedings of the IEEE ICASSP'95*, Detroit, MI, April 1995, pp. 1328–1331
74. H.S. Malvar, Extended lapped transforms: Properties, applications, and fast algorithms. IEEE Trans. Signal Process. **40**(11), 2703–2714 (1992)
75. F. Mintzer, Filters for distortion-free two-band multirate filter banks. IEEE Trans. Acoust. Speech Signal Process. **ASSP-33**(3), 626–630 (1985)
76. K. Nayebi, T.P. Barnwell, M.J.T. Smith, Low delay FIR filter banks: design and evaluation. IEEE Trans. Signal Process. **42**(1), 24–31 (1994)
77. K. Nayebi, T.P. Barnwell, M.J.T. Smith, Time domain filter bank analysis: a new design theory. IEEE Trans. Signal Process. **40**(6), 1412–1429 (1992)
78. K. Nayebi, T.P. Barnwell, M.J.T. Smith, Time domain conditions for exact reconstruction in analysis/synthesis systems based on maximally decimated filter banks, in *Proceedings of the 9th Annual Southeastern Symposium on System Theory*, Clemson, SC, March 1987, pp. 498–503
79. T.Q. Nguyen, P.N. Heller, Biorthogonal cosine-modulated filter bank, in *Proceedings of the IEEE ICASSP'96*, vol. III, Atlanta, GA, May 1996, pp. 1471–1474
80. K. Nayebi, M.J.T. Smith, T.P. Barnwell, Analysis-synthesis systems based on time-varying filter banks structures, in *Proceedings of the IEEE ICASSP'92*, vol. IV, San Francisco, CA, April 1992, pp. 617–620
81. K. Nayebi, T.P. Barnwell, M.J.T. Smith, Design and implementation of computationally efficient modulated filter banks, in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'91)*, vol. 1, Singapore, June 1991, pp. 650–653
82. K. Nayebi, T.P. Barnwell, M.J.T. Smith, On the design of FIR analysis-synthesis filter banks with high computational efficiency. IEEE Trans. Signal Process. **42**(4), 825–834 (1994)
83. K. Nayebi, T.P. Barnwell, M.J.T. Smith, A general time domain analysis and design framework for exact reconstructing FIR analysis/synthesis filter banks, in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'90)*, New Orleans, LA, April–May 1990, pp. 2022–2025

84. K. Nayebi, T.P. Barnwell, M.J.T. Smith, The time domain analysis and design of exactly reconstructing FIR analysis/synthesis filter banks, in *Proceedings of the IEEE ICASSP'90*, vol. 3, Alburquerque, NM, April 1990, pp. 1735–1738
85. T.Q. Nguyen, R.D. Koilpillai, Theory and design of arbitrary-length cosine-modulated filter banks and wavelets satisfying perfect reconstruction. IEEE Trans. Signal Process. **44**(3), 473–483 (1996)
86. T.Q. Nguyen, Near-perfect-reconstruction pseudo-QMF Banks. IEEE Trans. Acoust. Speech Signal Process. **ASSP-42**(1), 65–76 (1994)
87. T.Q. Nguyen, A class of generalized cosine-modulated filter banks, in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'92)*, vol. 2, San Diego, CA, May 1992, pp. 943–946
88. T.Q. Nguyen, P.P. Vaidyanathan, Two-channel perfect reconstruction FIR QMF structures which yield linear phase FIR analysis and synthesis filters. IEEE Trans. Acoust. Speech Signal Process. **ASSP-37**(2), 676–690 (1989)
89. T.Q. Nguyen, P.P. Vaidyanathan, Structures for M-channel perfect reconstruction FIR QMF banks which yield linear-phase analysis filters. IEEE Trans. Acoust. Speech Signal Process. **ASSP-38**(3), 433–446 (1990)
90. H.J. Nussbaumer, Pseudo QMF filter banks. IBM Tech. Discl. Bull. **24**(6), 3081–3087 (1981)
91. H.J. Nussbaumer, M. Vetterli, Computationally efficient QMF filter banks, in *Proceedings of the IEEE ICASSP'84*, San Diego, CA, March 1984, pp. 11.3.1–11.3.4
92. H.J. Nussbaumer, M. Vetterli, Pseudo quadrature mirror filters, in *Proceedings of the International Conference on Digital Signal Processing*, Florence, September 1984, pp. 8–12
93. M. Padmanabhan, K. Martin, Some further results on modulated/extended lapped transforms, in *Proceedings of the IEEE ICASSP'92*, vol. IV, San Francisco, CA, April 1992, pp. 265–268
94. S.M. Phoong, P.P. Vaidyanathan, Time-varying filters and filter banks: some basic principles. IEEE Trans. Signal Process. **44**(12), 2971–2988 (1996)
95. S.M. Phoong, P.P. Vaidyanathan, Factorability of lossless time-varying filters and filter banks. IEEE Trans. Signal Process. **45**(8), 1971–1986 (1997)
96. S.M. Phoong, P.P. Vaidyanathan, A polyphase approach to time-varying filter banks, in *Proceedings of the IEEE ICASSP'96*, vol. III, Atlanta, GA, May 1996, pp. 1554–1557
97. J.P. Princen, A.B. Bradley, Analysis/synthesis filter bank design based on time domain aliasing cancellation. IEEE Trans. Acoust. Speech Signal Process. **ASSP-34**(5), 1153–1161 (1986)
98. J.P. Princen, A.W. Johnson, A.B. Bradley, Subband/transform coding using filter bank designs based on time domain aliasing cancellation, in *Proceedings of the IEEE ICASSP'87*, Dallas, TX, April 1987, pp. 2161–2164
99. M. Poize, M. Rnaudin, P. Venier, A general time domain approach for the design of perfect reconstruction modulated filter banks, in *Proceedings of the IEEE ICASSP'93*, vol. III, Minneapolis, MN, April 1993, pp. 221–224
100. R.L. Queiroz, K.R. Rao, Variable-block-size lapped transforms. IEEE Trans. Signal Process. **44**(12), 3139–3142 (1996)
101. T.A. Ramstad, Analysis/synthesis filter banks with critical sampling, in *Proceedings of the International Conference on Digital Signal Processing*, Florence, September 1984, pp. 130–134
102. T.R. Ramstad, J.P. Tanem, Cosine-modulated analysis-synthesis filter bank with critical sampling and perfect reconstruction, in *Proceedings of the IEEE ICASSP'91*, Toronto, May 1991, pp. 1789–1792
103. J.H. Rothweiler, Polyphase quadrature filters – A new subband coding technique, in *Proceedings of the IEEE ICASSP'83*, Boston, MA, April 1983, pp. 1280–1283
104. T. Saramäki, A general class of cosine-modulated filter banks, in *Proceedings of the 1st International Workshop on Transforms and Filter Banks*, vol. 1, Tampere, February 1998, pp. 336–365

105. M. Schnell, R. Geiger, M. Schmidt, M. Multrus, M. Mellar, J. Herre, G. Schuller, Low delay filter banks for enhanced low delay audio coding, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, October 2007, pp. 235–238
106. G. Schuller, T. Karp, Modulated filter banks with arbitrary system delay: efficient implementations and time-varying case. IEEE Trans. Signal Process. **48**(3), 737–748 (2000)
107. G.D.T. Schuller, M.J.T. Smith, New framework for modulated perfect reconstruction filter banks. IEEE Trans. Signal Process. **44**(8), 1941–1954 (1996)
108. G. Schuller, A new factorization and structure for cosine modulated filter banks with variable system delay, in *Proceedings of the 30th Asilomar Conference on Signals, Systems and Computers*, vol. 2, Pacific Grove, CA, November 1996, pp. 1310–1314
109. G. Schuller, Time-varying filter banks with low delay for audio coding, in *105th AES Convention*, San Francisco, CA, September 1998. Preprint #4809
110. G. Schuller, Time-varying filter banks with variable system delay, in *Proceedings of the IEEE ICASSP'97* **3**, Munich, April 1997, pp. 2469–2472
111. G. Smart, A.B. Bradley, Filter bank design based on time domain aliasing cancellation with non-identical windows, in *Proceedings of the IEEE ICASSP'94*, vol. III, Adelaide, April 1994, pp. 181–184
112. M.J.T. Smith, T.P. Barnwell, A procedure for designing exact reconstruction filter banks for tree-structured sub-band coders, in *Proceedings of the IEEE ICASSP'84*, San Diego, CA, March 1984, pp. 27.1.1–27.1.4
113. M.J.T. Smith, T.P. Barnwell, A unifying framework for analysis/synthesis systems based on maximally decimated filter banks, in *Proceedings of the IEEE ICASSP'85*, Tampa, FL, March 1985, pp. 518–521
114. M.J.T. Smith, T.P. Barnwell, Exact reconstruction for tree-structured sub-band coders. IEEE Trans. Acoust. Speech Signal Process. **ASSP-34**(3), 431–441 (1986)
115. M.J.T. Smith, T.P. Barnwell, A new filter bank theory for time-frequency representation. IEEE Trans. Acoust. Speech Signal Process. **ASSP-35**(3), 314–327 (1987)
116. I. Sogadar, K. Nayebi, T.P. Barnwell, M.J.T. Smith, Time-varying analysis-synthesis systems based on filter banks and post filtering. IEEE Trans. Signal Process. **43**(11), 2512–2524 (1995)
117. A.K. Soman, P.P. Vaidyanathan, T.Q. Nguyen, Linear-phase orthonormal filter banks, in *Proceedings of the IEEE ICASSP'93*, vol. III, Minneapolis, MN, April 1993, pp. 209–212
118. A.K. Soman, P.P. Vaidyanathan, T.Q. Nguyen, Linear phase paraunitary filter banks: Theory, factorizations and design, IEEE Transactions on Signal Processing **41**(12), December 1993, pp. 3480–3496
119. I. Sodagar, K. Nayebi, T.P. Barnwell, Time-varying filter banks and wavelets, IEEE Trans. Signal Process. **42**(11), 2983–2996 (1994)
120. T.D. Tran, R. de Queiroz, T.Q. Nguyen, The generalized lapped biorthogonal transform, in *Proceedings of the IEEE ICASSP'98*, Seattle, WA, May 1998, pp. 1441–1444
121. P.P. Vaidyanathan, P.-Q. Hoang, The perfect reconstruction QMF bank: new architectures, solutions, and optimization strategies, in *Proceedings of the IEEE ICASSP'87*, Dallas, TX, April 1987, pp. 2169–2172
122. P.P. Vaidyanathan, Theory and design of M-channel maximally decimated quadrature mirror filters with arbitrary M, having the perfect reconstruction property. IEEE Trans. Acoust. Speech Signal Process., **ASSP-35**(4), 476–492 (1987)
123. P.P. Vaidyanathan, Quadrature mirror filter banks, M-band extensions and perfect reconstruction techniques. IEEE ASSP Mag. **4**(3), 4–20 (1987)
124. P.P. Vaidyanathan, Multirate digital filters, filter banks, polyphase networks, and applications: a tutorial. Proc. IEEE **78**(1), 56–93 (1990)
125. P.P. Vaidyanthan, K. Swaminathan, Alias-free, real-coefficients M-band QMF banks for arbitrary M. IEEE Trans. Circuits Syst. **CAS-34**(12), 1485–1496 (1987)
126. P.P. Vaidyanathan, Z. Doganata, T.Q. Nguyen, More results on the perfect reconstruction problem in M-band parallel QMF banks, in *Proceedings of the IEEE International Symposium on Circuits an Systems (ISCAS'87)*, Philadephia, PA, May 1987, pp. 847–850

127. P.P. Vaidyanathan, P.Q. Hoang, Lattice structures for optimal design and robust implementation of two-channel perfect-reconstruction QMF banks. IEEE Trans. Acoust. Speech Signal Process. **ASSP-36**(1), 81–94 (1988)
128. P.P. Vaidyanathan, T.Q. Nguyen, Z. Doganata, T. Saramäki, Improved technique for design of perfect reconstruction FIR QMF banks with lossless polyphase matrices. IEEE Trans. Acoust. Speech Signal Process. **ASSP-37**(7), 1042–1056 (1989)
129. P. Vary, G. Wackersreuther, A unified approach to digital polyphase filter banks. AEU Int. J. Electron. Commun. **37**(1/2), 29–34 (1983)
130. M. Vetterli, Filter banks allowing perfect reconstruction. Signal Process. **10**(3), 219–244 (1986)
131. M. Vetterli, A theory of multirate filter banks. IEEE Trans. Acoust. Speech Signal Process. **ASSP-35**(3), 356–372 (1987)
132. M. Vetterli, D.J. Le Gall, Perfect reconstruction FIR filter banks: lapped transforms, pseudo QMF's and paraunitary matrices, in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'88)*, Espoo, June 1988, pp. 2249–2253
133. M. Vetterli, D.J. Le Gall, Perfect reconstruction FIR filter banks: some properties and factorizations. IEEE Trans. Acoust. Speech Signal Process. **ASSP-37**(7), 1057–1071 (1989)
134. A. Viholainen, J. Alhava, M. Renfors, Efficient implementation of complex modulated filter banks using cosine and sine modulated filter banks. EURASIP J. Appl. Signal Process. Article ID 58564, 1–10 (2006)
135. A. Viholainen, J. Alhava, M. Renfors, Efficient implementation of 2× oversampled exponentially modulated filter banks. IEEE Trans. Circuits Syst. Express Briefs **53**(10), 1138–1142 (2006)
136. G. Wackersreuther, A novel approach to the design of filters for filter banks, in *Proceedings of the IEEE ICASSP'85*, Tampa, FL, March 1985, pp. 73–76
137. G. Wackersreuther, On the design of filters for ideal QMF and polyphase filter banks. AEU Int. J. Electron. Commun. **39**(2), 123–130 (1985)
138. G. Wackersreuther, Some new aspects of filters for filter banks, IEEE Trans. Acoust. Speech Signal Process. **ASSP-34**, 1182–1200 (1986)
139. G. Wang, The most general time-varying filter bank and time-varying lapped transforms. IEEE Trans. Signal Process. **54**(10), 3775–3789 (2006)
140. G. Wang, Analysis of M-channel time-varying filter banks. Digital Signal Process. **18**(2), 127–147 (2008)
141. G. Wang, Time-varying discrete-time signal expansions as time-varying filter banks. IET Signal Process. **3**(5), 353–367 (2009)
142. G. Wang, U. Heute, Time-varying MMSE modulated lapped transform and its applications to transform coding for speech and audio signals. Signal Process. **82**(9), 1283–1304 (2002)
143. G. Wang, Time-varying cosine-modulated filter banks. Digital Signal Process. **15**(3), 237–254 (2005)
144. H. Xu, W.S. Lu, A. Antoniou, Efficient iterative design method for cosine-modulated QMF banks. IEEE Trans. Signal Process. **44**(7), 1657–1667 (1996)

Chapter 2
Audio Coding Standards, (Proprietary) Audio Compression Algorithms, and Broadcasting/Speech/Data Communication Codecs: Overview of Adopted Filter Banks

2.1 Introduction

In general, audio coding or audio compression algorithms are used to obtain compact digital representation of high-quality audio signals for their efficient transmission and storage. The central objective in audio coding is to represent the signal with a minimum number of bits while achieving its transparent reproduction. Motivated by these demands considerable research activities have been spent toward formulation of audio compression/coding schemes to satisfy simultaneously requirements of high compression ratios and transparent reproduction quality of audio signals. As a result, a number of audio coding schemes were developed/standardized for the high-quality audio coding [1, 2, 6, 11].

Besides speech coding schemes based on linear prediction methods which are especially tailored for efficient speech compression, the developed perceptual transform-based audio coding schemes gained a greater attention, particularly for applications in consumer electronics. Typically, any transform-based audio coding scheme utilizes a near-perfect quadrature mirror filter (QMF) and/or perfect reconstruction cosine-modulated filter bank to obtain a block-wise representation of the audio signal in the frequency domain. The obtained spectral coefficients after quantization are then efficiently encoded into a compact form.

In this chapter, perceptual transform-based audio coding schemes developed up to now are briefly reviewed including the family of ISO/IEC MPEG audio coding standards, proprietary audio compression algorithms, broadcasting/speech/data communication codecs, as well as open-free, patent royalty-free audio/speech codecs. The discussion is concentrated especially on adopted near-perfect QMF and perfect reconstruction cosine-modulated filter banks, processing methods, and specified transform block sizes. For more details about specific audio coding standard/compression algorithm, an interested reader can find in the appropriate official MPEG document(s), available web sites, or books [1, 2, 9, 11, 15].

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1_2

2.2 Family of ISO/IEC MPEG Audio Coding Standards

During more than 25 years, the MPEG committee in collaborative work with many companies, universities, and research institutes worldwide developed and standardized several audio codecs for high-quality compression of audio/speech coding. From the viewpoint of adopted filter banks, processing methods, and specified transform block sizes they are briefly reviewed in the following subsections.

2.2.1 MPEG-1/2 Audio Coding Standards

The MPEG-1 audio compression algorithm [6, 9, 11, 12], finalized in 1992, is the first established international coding standard for high-quality compression of digital audio signals. The MPEG-2 audio coding standard [6, 9, 11, 13], finalized in 1994, extends the multichannel capabilities not offered by MPEG-1 audio. MPEG-1/2 algorithms involve three distinct layers for compression. Layer I forms the most basic compression algorithm, while layers II and III are enhancements that use some elements of layer I. Each successive layer improves the compression performance but at the cost of greater encoder and decoder complexity [1, 9]. Essentially, layer III of MPEG-1/2, known as MP3 standard, has become at that time key technology to realize audio decoders for music distribution via Internet and consumer electronics (portable MP3 players and multimedia systems).

For the time-to-frequency transformation of digital audio signals the MP3 standard [13] employs the hybrid filter bank consisting of a near-perfect reconstruction cosine-modulated QMF bank referred also to as the pseudo-QMF bank [1, 9], and the adaptive modulated lapped transform (MLT) [3, 4], or equivalently, the adaptive oddly stacked modified discrete cosine transform (MDCT) [8] associated with the sine windowing function [4]. The pseudo-QMF bank, common to all three layers of MPEG-1/2 audio, decomposes the input audio signal into 32 equally spaced frequency sub-bands, i.e., $N = 64$. Each sub-band is then coded either to single groups of 12-sample blocks (in layer I) or to groups of three successive 12-sample blocks (in layer II). For 32 sub-bands this results in two data frames of $32 \times 12 = 384$ and $32 \times 36 = 1152$ samples [1, 9].

In MP3 standard [13], the outputs of pseudo-QMF bank are further processed by the MLT (MDCT) filter bank operating on the block of $N = 12$ samples (the short block) or the block of $N = 36$ samples (the long block). The long block allows greater frequency resolution for signals with stationary characteristics, while the short block provides better time resolution for transient signals. Basic windowing operation is defined for the long block and short block. During transient signals, the long block is replaced by a series of three overlapped short blocks, thus maintaining the same total number of samples as for the long block. Each of the three short blocks is then windowed separately. Switching between long and short blocks is not instantaneous. In order to ensure smooth transition between long and short blocks

and vice versa, transient blocks (long-to-short and short-to-long blocks having the same size as the long block) are specifically defined and windowed [1, 9]. Both short and long block sizes are not powers of two. Actually they are composite lengths of the form $2^m \times q$, where q is an odd integer. Specifically, $12 = 4 \times 3 = 2^2 \times 3$ and $36 = 4 \times 9 = 2^2 \times 9$.

In Chap. 5, a number of various efficient implementations of the pseudo-QMF and MLT (MDCT) filter banks tailored directly to the MP3 audio coding standard are discussed in detail.

2.2.2 *MPEG-2/4 Advanced Audio Coding (AAC) Audio Coding Standards*

MPEG-1/2 standards discussed in the previous subsection involve practical audio compression algorithms for high-quality coding of monophonic and stereophonic material. By the early 1990s, the demand for high-quality coding of multichannel audio at reduced bit rates had increased significantly. Therefore, MPEG group started in 1994 standardization activities for developing a higher quality multichannel non-backwards compatible advanced audio coding system [1, 11]. This effort with collaborative effort among worldwide companies, universities, and research institutes (Dolby Laboratories, Sony Corporation, AT&T Bell Laboratories or Lucent Technologies, Fraunhofer Institute and University of Hannover) led to the adoption of the MPEG-2 Non-Backwards Compatible/Advanced Audio Coding (NBC/AAC) standard [14]. The MPEG-2 NBC/AAC was later renamed MPEG-2 AAC and finalized in 1997. The AAC technology made use of all the advanced audio coding methods available at the time of its development [1, 11, 15].

The MPEG-2 AAC standard is organized as a set of modular coding tools. Based on a trade-off among desired quality coding, channel resources, and memory/power processing requirements, the MPEG-2 AAC system allows to select from three complexity profiles: Main Profile, Low Complexity (LC) Profile, and Scalable Sampling Rate (SSR) Profile. Each profile recommends a specific combination of coding tools. In the Main Profile configuration, the MPEG-2 AAC provides the best audio quality at any given data rate. Memory and processing power requirements in the Main Profile configuration are higher than those in the LC Profile configuration. In the SSR Profile configuration, the gain control tool is used. It consists of a Polyphase QMF (PQMF) filter bank of order 96, gain detectors and gain modifiers. The PQMF filter bank splits each audio channel input signal into four frequency bands of equal width. Then each PQMF filter bank output is processed by the oddly stacked MDCT [8] to produce 256 spectral coefficients (i.e., $N = 512$), for a total of 1024 coefficients. The gain control can be applied to each of four bands independently [1, 14, 15].

The MPEG-2 AAC coding algorithms constitute the kernel of the MPEG-4 AAC audio coding standard (version 1 finalized in 1999 and version 2 finalized in 2000) [16]. These MPEG-4 AAC versions include several additional functionalities such as scalability, error resilience, technology for coding general audio (speech and synthetic audio), and some additional spectral processing tools. The MPEG-4 AAC was targeted for a wide number of applications including wired, wireless, streaming, digital broadcasting, interactive multimedia, telephony and mobile communication, and high-quality audio/video [1, 11].

Fundamental component of MPEG-2/4 AAC encoder is the conversion of time domain signals into frequency representation by applying of the time-variant oddly stacked MDCT filter bank [8]. With the adaptive block size switching procedure, quasi-stationary audio segments are analyzed/synthetized with 2048-sample long data block ($N = 2048$), while transient signals are analyzed/synthetized with a series of eight 256-sample short data blocks ($N = 256$) to reduce pre-echo effects. Since the windowing function has a significant impact on the MDCT filter bank frequency response, the MPEG-2/4 AAC allows a dynamical switching between two distinct windowing functions (adaptive switching procedure of windowing function) to best adapt to signal characteristics. The sine windowing function [4] or a parametric Kaiser–Bessel Derived (KBD) windowing function [45, 49] is used. We note that the adaptive switching of windowing function is employed on the 2048-sample long data blocks only [1, 14].

2.2.3 MPEG-4 AAC-Low Delay (AAC-LD) Audio Coding Standard

Although, in general, MPEG-2/4 AAC perceptual audio codecs provided high sound quality even at low bit rates for broad range of signals, the total delay of encoder/decoder chain was still considerably high, than can be acceptable for upcoming high-quality interactive bidirectional communication applications such as telephony, Voice over Internet (VoiceIP), and teleconferencing. Therefore, it was concluded that a novel coding scheme has to be designed/introduced combining the advantages of perceptual audio coding with the low delay operation required for interactive bidirectional communication [17, 19].

As the first step, a general overview over the structure of existing perceptual audio coding schemes (MPEG-2/4 AAC, MP3) has to be performed followed by an analysis to identify the primary sources of algorithmic delay inherent in the encoding/decoding chain of such schemes. It is noted that the algorithmic delay is defined as the theoretical minimum delay allowed by an algorithm. Then, the total algorithmic delay can be derived as the sum of delay contributions of coding algorithms [17]. Based on the detailed analysis [18], the following main sources contributing to the total algorithmic delay have been identified [19]:

- Block-based processing associated with data block size—due to the use of a block transform, a certain amount of time is needed to collect all samples belonging to one block.
- Filter bank algorithmic delay—due to the overlap/add procedure of the filter bank with 50% overlap to previous and subsequent data blocks, a delay of one data block is caused by the filter bank.
- Look-ahead time for data block size switching procedure—in transition from the long to short data blocks and vice versa, the so-called transition blocks (labeled as "start" and "stop") have to be constructed to preserve the perfect reconstruction property of filter bank in the overlapped part.

As a result of this analysis, the so-called MPEG-4 AAC Low Delay (AAC-LD) was derived from MPEG-4 AAC general audio object type (MPEG-2 AAC-LC plus some additional coding tools) and optimized for very low delay operation [17, 19].

The following modifications on the standard MPEG-4 AAC algorithm have been performed in order to achieve the low delay operation in MPEG-4 AAC-LD [17, 19]:

- Data block size has been reduced from $N = 2048$ to $N = 1024$ or $N = 960$ samples. This leads to a delay of 512 or 480 samples. The MDCT analysis/synthesis filter banks cause a further delay of the same size. Note that the data block size $N = 960$ is a composite integer of the form $2^m \times q$, where q is an odd integer. Specifically, $960 = 2^6 \times 15$.
- Block size switching procedure has been deactivated.
- The AAC-LD has allowed the use of two different windowing functions for windowing procedure depending on signal characteristics. Besides the sine windowing function applied to the stationary signals, the AAC-LD uses a low overlap windowing function between subsequent data blocks in the case of transient signals (see Figs. 8 and 9 in [17]). This dynamic adaptation of windowing function does not imply any additional delay.

Thus, the MPEG-4 AAC-LD represents the modified MPEG-4 AAC codec fulfilling requirements of the total low algorithmic delay for full-duplex communication applications. Moreover, it filled the gap between existing low delay speech coding schemes (such as ITU-T G.722.1 Annex C [60, 68]) and perceptual high-quality audio coding schemes, as well as provided a baseline for development of the low delay coding in MPEG-4 codec family [19].

2.2.4 MPEG-4 High-Efficiency AAC (HE-AAC) Audio Coding Standard

The rapid development of digital communication has opened numerous opportunities for new multimedia services such as terrestrial- and satellite-based digital audio broadcasting as well as wireless music downloads to cellular phones [21]. In

2001, MPEG committee identified two areas for improved audio coding technology and issued a call for proposals [20] for: Improved compression efficiency of audio or speech signals by a bandwidth extension method, which is forward and backward compatible with existing MPEG-4 technology, and improved compression efficiency of high-quality audio signals by a parametric stereo coding method. Based on the MPEG-2/4 AAC architecture, the work item led to standardization in the form of amendments, at first in 2003 of the MPEG-4 High-Efficiency AAC (HE-AAC) audio coding standard (known also as AACPlus), and subsequently in 2004 to standardization of the MPEG-4 HE-AAC version 2 (known also as AACPlus v2). MPEG-4 HE-AAC is the combination MPEG-2/4 AAC and integrated bandwidth extension method, the so-called Spectral Band Replication (SBR) compression technology, which enables to reconstruct high-frequency band based on low-frequency band data. The MPEG-4 HE-AAC v2 is the combination of MPEG-4 HE-AAC and Parametric Stereo (PS) coding tool which achieves a significantly increased coding efficiency for stereo signals when compared to HE-AAC operating in normal stereo mode [20, 21]. The MPEG-4 HE-AAC (v2) standards are targeted on low bit rate applications with higher coding efficiency such as mobile music and TV, digital radio and TV broadcasting, Internet streaming, and consumer electronics.

The SBR compression technology significantly improves the compression efficiency of perceptual audio and speech coding schemes. SBR always operates in conjunction with a conventional codec, a core codec (it is not a stand-alone coder). The SBR acts as pre-processing at the encoder, and as post-processing at the decoder [20, 21]. SBR is based on the fact that in most cases there is strong correlation between the characteristics of lower and higher frequency content of an audio signal. Consequently, the high frequency part can be reconstructed from the low frequency part, or in other words, the SBR is able to recreate the missing high frequency components of a decoded audio signal in a perceptually accurate way by re-using signal information from the decoded low frequency part, thus allowing a much higher audio quality at low data rates. Therefore, transmission of the high frequency part is not necessary, only the low frequency part and a small set of control data need to be carried in the bit stream to guarantee an optimal reconstruction of high frequencies. Essentially, SBR-enhanced codecs have the major advantage of being backward and forward compatible to the core codec. This fact permits to integrate the SBR technology to existing systems, thus enabling a smooth transition from a conventional audio coder to its more efficient SBR-enhanced version. In general, the SBR can be combined with any conventional (even not necessarily perceptual) audio/speech codec [23].

Central to the operation of standard SBR compression technology used in the MPEG-4 HE-AAC [22] are dedicated complex exponential-modulated and real-valued cosine-modulated QMF banks as the basic mathematical tools to analyze and synthesize audio signals. Standard SBR for the encoder defines only one complex exponential-modulated analysis QMF bank, while standard SBR for the decoder defines two types of analysis/synthesis QMF banks depending on

the application. Specifically, complex exponential-modulated QMF banks forming the high quality SBR (HQ-SBR), and real-valued cosine-modulated QMF banks forming the low power SBR (LP-SBR). The main difference between HQ-SBR and LP-SBR is how the data is represented during the SBR process. In the HQ-SBR all subsequent calculations are realized in complex arithmetic. The LP-SBR operates with real-valued cosine-modulated analysis and synthesis QMF banks, and hence in real-valued arithmetic to reduce the computational complexity. In situations, where a lower sampling rate is sufficient, for example in portable devices, the SBR can run in a down-sampled mode and down-sampled versions of complex and real-valued synthesis QMF banks can be employed. The complex exponential-modulated QMF banks are intended for use in applications requiring the best possible audio quality at a given bit rate, while the real-valued QMF banks are intended to be lower complexity versions that still produce acceptable results in terms of audio quality and bit rate. The modulation stages of QMF banks in the ISO/IEC MPEG document [22] are defined by matrix-vector products with the number of sub-bands being 64 or 32, i.e., audio data blocks of sizes $N = 128$ or 64, and with fixed values of time shift factors in the transform kernels. The symmetric prototype filter is of order 640.

Complex exponential-modulated and real-valued cosine-modulated analysis and synthesis QMF banks used in the standard SBR technology of MPEG-4 HE-AAC, their general symmetry properties in the frequency or time domain, and their efficient implementations are discussed in detail in Chap. 7.

2.2.5 MPEG-4 AAC-Enhanced Low Delay (AAC-ELD) Audio Coding Standard

The MPEG committee in 2008 has completed the development and standardization process of an audio communication codec, the MPEG-4 Advanced Audio Coding—Enhanced Low Delay (AAC-ELD), targeted towards the high-quality real-time (interactive) bidirectional communication applications such as audio and video conferencing [24]. Essentially, the MPEG-4 AAC-LC, HE-AAC, and AAC-LD codecs form the basis of AAC-ELD. In order to achieve the high coding efficiency and low algorithmic delay, the AAC-ELD combines a low delay-optimized SBR compression technology [24, 26, 26] known from the HE-AAC [22], and a perfect reconstruction low delay cosine-modulated filter bank [10, 26, 27]. In 2011 the MPEG completed the standardization of a Low Delay MPEG Surround as a parametric stereo coding tool for enhancing the AAC-ELD codec [25]. The combination of both technologies, the Low Delay MPEG Surround and AAC-ELD, is also known as the AAC-ELD v2. The applications of AAC-ELD v2 involve broadcasting and mobile videoconferencing.

The low delay version of SBR (LD-SBR) compression technology was integrated into AAC-ELD [24]. Since the LD-SBR is derived from the standard SBR [22] (see also Sect. 2.2.4) with some modifications, similarly, the LD-SBR for the encoder defines also only one low delay complex exponential-modulated analysis QMF bank and two types of low delay analysis and synthesis QMF banks for the decoder forming the high quality LD-SBR (HQ-LD-SBR) and low power LD-SBR (LP-LD-SBR). Similarly, in situations where a lower sampling rate is sufficient, the LD-SBR can operate in a down-sampled mode using down-sampled versions of low delay complex and real-valued synthesis QMF banks. The modulation stages of low delay QMF banks in the ISO/IEC MPEG document [24] are defined with the number of sub-bands being 64 or 32, i.e., audio data blocks are of sizes $N = 128$ or 64.

Low delay complex exponential-modulated and real-valued cosine-modulated analysis and synthesis QMF banks used in the LD-SBR compression technology, their general symmetry properties in the frequency or time domain, and their efficient implementations are discussed in detail in Chap. 7.

On the other hand, now it is well known that the MPEG audio coding standards, such as MPEG-4 AAC-LC, HE-AAC, and AAC-LD, utilize for the time-to-frequency transformation of an audio data block and vice versa, the oddly stacked MDCT which is based on the concept of time domain aliasing cancellation (TDAC) [8]. However, the AAC-ELD has adopted a perfect reconstruction low delay filter bank, called the Low Delay MDCT (LD-MDCT) [10]. The purpose of the LD-MDCT is to reduce the reconstruction delay independent of the prototype filter length, while still maintaining the perfect reconstruction property. The LD-MDCT has a similar cosine modulation kernel as TDAC MDCT, but substantial delay reduction is achieved by utilizing an asymmetric analysis windowing function with a low reconstruction delay and with multiple overlap (four succeeding data block are overlapped). This cannot be accomplished with TDAC MDCT which employs a symmetric windowing function and thus has an algorithmic delay identical to the block size minus one [26, 27]. The AAC-ELD defines for the LD-MDCT the data block size to be $2N$ with $N = 1024$ or 960 [24]. Note that $960 = 2^6 \times 15$ is the composite length. The low delay analysis windowing functions for $N = 1024$ and 960 are tabulated in the explicit form in ISO/IEC MPEG document [24]. Note that the first $\frac{N}{8}$ values of windowing functions are implicitly equal to zero. Since the low delay analysis windowing function is asymmetric, the low delay synthesis windowing function is the time-reversed version of the corresponding analysis windowing function [10, 26, 27].

Definitions of the analysis and synthesis LD-MDCT filter banks used in the AAC-ELD, their general symmetry properties in the time and frequency domains, relations between the LD-MDCT and TDAC MDCT, as well as efficient LD-MDCT implementations are discussed in detail in Chap. 8.

2.2.6 *MPEG-4 Scalable Lossless Audio Coding (SLS) and High-Definition AAC/SLS (HD-AAC/SLS) Audio Coding Standards*

Almost all perceptual audio coding schemes discussed in this chapter are based on the transform-based approach, i.e., they employ sub-band filter banks to obtain a block-wise representation of the audio signal in the frequency domain. They operate in floating-point arithmetic, and therefore are lossy in nature [1]. Due to increasing demand of delivery of high sampling rate and high resolution digital audio at lossless quality for high-quality applications, such as audio archiving systems, the lossy compression became inappropriate since every bit in the original audio signal has to be preserved. On the other hand, the transition from lossy to lossless coding by a scalable way would facilitate the digital audio services to interchange compressed audio across various application domains using scalable lossy to lossless compressed formats. Thus, a scalable to lossless audio coding technology that will support both lossy and lossless audio compression simultaneously was desirable [28]. Responding to these demands, the MPEG audio standardization group decided to start a new work item to explore a new relevant innovative technology for lossless and near-lossless coding of audio signals in 2002. For the extension to lossless operation, the lossy MPEG-4 AAC codec has been used as a core codec [28, 31].

Research and standardization efforts of MPEG audio group led to the specification of SLS (scalable lossless coding solution) technology in the form of amendment to the MPEG-4 audio standard [28, 30–32]. As the extension of MPEG-4 AAC perceptual audio codec, the MPEG-4 SLS codec includes a scalable lossless audio coding solution that integrates the functionalities of high-compression, lossless audio coding, perceptual audio coding, and fine granular scalable audio coding into a single coder, while simultaneously provides the backward compatibility to existing MPEG-4 AAC codec at the bit stream level [31]. An enabling technology for the scalability in the frequency domain is the Integer MDCT (IntMDCT), being an integer approximation of the oddly stacked MDCT filter bank [8] or equivalently, the integer MLT (IntMLT) with the sine windowing function. In order to achieve backward compatibility to existing MPEG-4 AAC codec, the MPEG-4 SLS adopts two-layer structure to code IntMDCT spectral coefficients: the AAC core layer, and a lossless enhancement layer working on the top of AAC architecture. These two layers in the encoder generate the core layer bit stream which is MPEG-4 AAC compliant, and the lossless enhancement layer bit stream providing the scalability from lossy to lossless coding. AAC compliant bit stream is embedded in the final bit stream. Bit-exact reconstruction of the input original audio signal in the decoder is independent to the implementation accuracy of the AAC core codec. Since MPEG-4 SLS provides the fine granular bit rate scalability from lossy to lossless coding, it becomes a universal compression system for digital audio applications which up to now required different audio coding technologies. Moreover, the need for any transcoding is completely eliminated [31]. Similarly, other AAC coding tools such

as Mid/Side (M/S) stereo coding and Temporal Noise Shaping are considered and implemented in an invertible integer way on the IntMDCT spectral coefficients. An interested reader can find an overview of the MPEG-4 SLS standard, its application scenarios, structure, and description of coding tools in [31].

In 2007 the MPEG audio group has successfully concluded the standardization process on enhanced SLS technology for lossless coding of high-definition (HD) audio signals—ISO/IEC MPEG-4 High-Definition Scalable Advanced Audio Coding (MPEG-4 HD-AAC/SLS) [29]. HD-AAC/SLS audio coding technology provides a fine grain scalable lossless extension of the MPEG-4 AAC perceptual audio coder up to fully lossless reconstruction at word lengths and sampling rates typically used for HD audio. In the context of HD audio applications it is frequently to achieve lossless signal reconstruction at higher sampling rates as the MPEG-4 AAC operates. In this case, the MPEG-4 HD-AAC/SLS can operate in the so-called "oversampling factor" mode by using longer IntMDCT transform sizes such as $N = 4096$ or even $N = 8192$ samples. Using the longer IntMDCT provides a better lossless performance for stationary signals than the transform size $N = 2048$. Enhanced HD-AAC/SLS technology generates a universal digital audio format for a variety of (HD) applications including digital audio archiving, network audio streaming, portable audio players, digital VCD and DVD media, consumer electronics, and digital broadcasting [29].

The MPEG-4 SLS and MPEG-4 HD-AAC/SLS audio coding standards combine local and global methods to construct the integer approximate MDCT (MLT) filter banks. They are discussed in detail in Chap. 9.

2.2.7 MPEG-D Unified Speech and Audio Coding (USAC)

All previously discussed MPEG audio coding standards, such as MPEG-4 HE-AAC v2, achieve high subjective sound reproduction quality at low bit rates for music signals. However, psychoacoustic models in the spectral domain used in such audio coding schemes do not perform well on speech signals at low bit rates. On the other hand, existing speech coding schemes, such as extended Adaptive Multi-Rate Wide-Band (AMR-WB+) [63, 65], use the time domain source filter to closely model speech process and consequently, perform very well for speech signals at low bit rates, but they show poor quality for music signals. This is main reason why the speech and music signals have been encoded separately using different coding schemes to achieve their high-quality reproduction [35, 37, 40]. Moreover, in many applications areas, such as broadcasting, audio books, and audio video playback, the content is highly variable and is not restricted to speech or music only [33]. Motivated by these facts, the MPEG initiated standardization process for a new codec with consistent high quality coding of speech, music and mixed content over broad range of bit rates, or other words, a single coding scheme which can encode both speech and music signals without any degradation of quality. MPEG standardization process with the working title of "MPEG-D Unified Speech and

Audio Coding (USAC)" started in 2007. After issued Call for Proposals on USAC as a new technology, a first reference model architecture (RM0) was developed and presented in 2009 [38, 39] which already combined all advantages of state-of-the-art speech and general audio coders. In the subsequent collaborative phase of Fraunhofer Institute Erlangen, further enhancements were integrated into the system from companies worldwide: VoiceAge Canada, Dolby Sweden, Philips, Sony, Panasonic, Samsung, NTT DOCOMO Japan, Audio Research Labs USA, and International Audio Laboratories Erlangen. The MPEG-D USAC standard was finalized in early 2012. Main applications of the MPEG-D USAC are multimedia downloads to mobile devices, digital radio, mobile TV, and audio books and in general, applications dealing with a mixed content of speech and music signals [36, 40, 41].

The MPEG-D USAC architecture combines improved/enhanced/refined coding methods and algorithms of MPEG-4 HE-AAC v2 codec [22], and the speech AMR-WB+ codec with Algebraic Code Excited Linear Prediction (ACELP) speech compression technology [63, 65] by means of switching between the core coders of two standards. High performance of USAC arises from the intelligent interaction between two coding schemes controlled by a signal classification module. Two coding schemes share common innovative technologies: a parametric enhanced SBR (eSBR) compression technology and a parametric stereo coding based on MPEG Surround technology [40, 41].

The USAC transform-based core coder supports a larger set of the oddly stacked MDCT transform sizes. Specifically, additional transforms sizes $N = 1024$ and 512 complement the AAC $N = 2048$ and 256-sample data blocks, thus providing a more suitable time-frequency decomposition for variety of signals [35, 36]. Further flexibility is achieved by allowing 768-sample windowed data block. In this mode all above mentioned transform sizes are reduced to $\frac{3}{4}$th of their original ones, i.e., $N = 2048$ is reduced to $N = 1536 = 2^9 \times 3$, $N = 1024$ is reduced to $N = 768 = 2^8 \times 3$, $N = 512$ is reduced to $N = 384 = 2^7 \times 3$, and $N = 256$ is reduced to $N = 192 = 2^6 \times 3$.

In USAC transform-based core coder the MDCT filter bank is designed to be more flexible [40, 41]. It is well known that the MDCT has a good energy compaction property especially for harmonic tones with constant fundamental frequencies. However, when the fundamental frequency is time-varying, typically for voiced speech, the energy is spread over several spectral coefficients, and the voiced speech portions are not represented accurately at low bit rates. This fact leads to a loss of coding efficiency. In order to overcome this problem, the USAC introduced a new filter bank, the so-called time-warped MDCT (TW-MDCT) [33, 40]. By means of a continuous fundamental frequency estimation, a time varying resampling (time warping) is applied locally within every audio data block prior to the MDCT. The windowing functions have also need to be adapted accordingly to preserve the perfect reconstruction property of the MDCT. The resampling (time warping) ideally leads to a constant fundamental frequency within audio data block, and thus the TW-MDCT filter bank can adapt its spectral representation for a better energy compaction of voiced speech signals

compared to the conventional MDCT. When the fundamental frequency in USAC transform-based core coder is set to zero, the TW-MDCT becomes the conventional MDCT [33].

Since USAC core codec is switching between the AAC transform-based and linear prediction (LP) speech coder, a special attention was paid for a fast adaptation to either speech, music, or mixed content without blocking effects, and a smooth transition between the signal types without additional overhead leading to the design of new transition windowing functions in transition regions for AAC coding mode [39]. These "start" (transition from the MDCT domain to LP time domain) and "stop" (transition from the LP time domain to MDCT domain) windowing functions are similar to regular AAC transition ones, with either the KBD or sine windowing function on the appropriate half side of AAC transformed signal. They consist of a flat top region of 448 or 576 samples equal to 1, respectively, followed by the sine windowing function of size 64 or 128 samples, respectively, and consecutive number of zero samples. In particular, due to a requirement on constant framing, for the transition from LP speech to transform-based coder the right side of the windowing function is completed with 64 zero samples. Consequently, the transform block size is enlarged from $N = 2048$ to $N = 2304 = 2^8 \times 9$. For more details see transition schemes shown in Figs. 2 and 3 of Neuendorf et al. [39].

The eSBR is derived from the standard SBR compression technology (see Sect. 2.2.4) with many incorporated new functionalities and tools, such the harmonic transposer, predictive vector coding, and inter-sub-band-sample temporal envelope shaping [37, 40]. The standard SBR was initially designed as a 2:1 system, i.e., at first 1024 low frequency coefficients from the core coder are fed into 32-band analysis QMF bank ($N = 64$). After reconstructing the high frequency content, the signal is transformed back to the time domain using a 64-band synthesis QMF bank ($N = 128$) resulting in 2048 time domain samples. For USAC, the standard SBR was extended by two additional operating modes. For lower sampling rates the first mode 4:1 uses 16-band analysis QMF bank (i.e., $N = 32$) instead of 32-band analysis QMF bank. In the second mode the eSBR is capable of operating in an 8:3 mode. In this case, a 24-band analysis QMF bank (i.e., $N = 48 = 2^4 \times 3$) is used [40]. Complex exponential-modulated and real-valued cosine-modulated analysis and synthesis QMF banks with a common parameter representing the number of sub-bands which are used in the standard SBR, their general symmetry properties in the frequency or time domain, and their efficient implementations are discussed in detail in Chap. 7.

MPEG Surround and unified stereo coding in USAC codec employ complex exponential-modulated QMF banks which are shared with the eSBR compression technology [40]. In order to improve the signal compaction property of stereo (mid/side) coded channel spectra, the USAC provides a complex-valued stereo prediction tool operating directly in the MDCT domain of encoder/decoder [34]. Left and right channels of stereo signal represented in MDCT domain are first converted to mid and side (sum and differences) spectra, and required complex-valued down-mix spectrum is obtained via a real-to-imaginary transform

(in USAC labeled as R2I), whose real part is the MDCT and imaginary part is the corresponding modified discrete sine transform (MDST). R2I transform actually constructs the modulated complex lapped transform (MCLT) [5] directly in the frequency domain from given MDCT spectra. Exact and approximation conversion methods to construct the MCLT directly in the frequency domain for arbitrary symmetric windowing function are discussed in detail in Chap. 6.

The MPEG-D USAC is the first codec that merges the speech and audio coding into unified form and it represents the new state-of-the-art coding technology for speech, music, and mixed content signals at low bit rates. This makes USAC the most efficient codec for all signal categories and moreover, it can be considered as the 4th generation MPEG audio codec [40, 41].

2.3 Proprietary Audio Compression Algorithms

Besides the family of MPEG audio coding standards, the international companies such as Sony, AT&T Bell Laboratories (Lucent Technologies), and Dolby Labs developed their own audio compression algorithms which are their property. From the viewpoint of adopted filter banks, processing methods, and specified transform block sizes they are briefly reviewed in the following subsections.

2.3.1 Family of Sony® ATRAC/SDDS/ATRAC2/ATRAC3/ ATRAC3plus and ATRAC Advanced Lossless Digital Audio Compression Systems

The ATRAC (Adaptive TRansform Acoustics Coding) digital audio compression system developed by Sony [53] was originally intended for a low-cost, battery-powered consumer electronics equipment, its rewritable MiniDisc portable player [54]. The ATRAC combines sub-band and transform coding methods to achieve nearly CD quality audio coding. Using a cascaded two-stage QMF analysis bank, the ATRAC encoding process first splits the input signal into three sub-bands: low-, mid-, and high-frequency. Then, each sub-band is transformed into the frequency domain by the signal adaptive oddly stacked MDCT analysis filter bank [8] with a symmetric identical analysis and synthesis windowing functions [2]. The adaptive switching of transform block sizes is employed based on the signal characteristics in each sub-band. There are two block-size modes, the long mode and short mode. The transform block size switching procedure works as follows. During stationary (steady-state) periods, the high resolution MDCT analysis filter bank is attained using long 512-sample data blocks, i.e., $N = 512$. During transient periods, short

blocks are used, specifically, $N = 256$ for the high-frequency band, and $N = 128$ for the low- and mid-frequency bands to cancel pre-echo artifacts [6, 11]. The ATRAC mapping structure can be found in [2]. It is noted that Sony during the ATRAC development was active in MPEG-2 AAC research and standardization process [14].

The ATRAC digital audio compression algorithm has been adopted as a core of Sony's digital cinematic sound system, the so-called Sony Dynamic Digital Sound (SDDS). The SDDS integrates 8 independent ATRAC modules to carry information for each left, right, left center, center, right center, subwoofer, left surround, and right surround channels typically present in a modern theater [2, 6, 11].

ATRAC2 digital audio compression algorithm is an enhanced version of the ATRAC containing two new coding tools: First, based on a time-frequency analysis in the encoding process, the ATRAC2 extracts psychoacoustically important tone components from the input signal spectrum which are efficiently encoded separately from less important spectral data. Secondly, ATRAC2 prevents pre-echoes adaptively by using the so-called gain modification coding tool. ATRAC2 performs a signal analysis using combination of a polyphase quadrature filter (PQF) bank and fixed-length MDCT with different analysis and synthesis windowing functions. The PQF bank splits the input signal into four sub-bands. Each sub-band is then transformed by the MDCT with a frequency resolution being twice that of ATRAC, i.e., the block size is $N = 1024$. The ATRAC2 mapping structure can be found in [2].

The structure of ATRAC3 is very similar to that of ATRAC2. ATRAC3 uses the cascaded two-stage QMF bank, similarly as ATRAC, but the input signal is split into four sub-bands. The main reason for employing the QMF banks instead of PQF bank is to facilitate direct transformation between the bit stream of ATRAC and that of ATRAC3 [2]. ATRAC3plus supports the multi-channel coding (max. 64) and provides twice the coding efficiency of ATRAC3 by using the following technologies: The PQF divides stereo input signal, left and right channel, each is split into 16 sub-bands which is four times the number of sub-bands in ATRAC3. The MDCT transform size is twice that of ATRAC3, i.e., $N = 2048$. The ATRAC3plus mapping structure can be found in [2].

Finally, for the lossless audio coding applications, Sony developed a scalable lossy to lossless encoder/decoder, the ATRAC Advanced Lossless codec [2]. The ATRAC Advanced Lossless codec consists of a base layer and an enhancement layer. In the base layer, the ATRAC3 or ATRAC3plus encoder encodes the input audio signal into the base-layer bit stream. Subsequently, the base-layer bit stream is decoded by the deterministic ATRAC3 or ATRAC3plus decoder to restore a large portion of the original audio signal. The residual signals which are the difference signals between the original and restored signals of the deterministic decoder, are encoded by the enhancement layer encoder. Both the base-layer and enhancement-layer bit streams are transmitted in the lossless enhancement bit stream [2].

2.3.2 Lucent Technologies PAC/EPAC/MPAC Audio Coders

Audio compression algorithm, the Perceptual Audio Coder (PAC), was originally developed by AT&T Bell Laboratories [50, 51]. Historically, AT&T and Lucent Technologies separated after the Multichannel PAC (MPAC) algorithm was evaluated for MPEG-2 NBC/AAC testing, and the PAC algorithm subsequently became proprietary to Lucent Technologies. AT&T, meanwhile, has become active in MPEG-2 AAC research and standardization process. The low-complexity profile of AAC, MPEG-2 AAC-LC has become the AT&T coding standard. Lucent PAC algorithm is flexible in that it supports monophonic, stereophonic, and multiple channel modes [2, 6, 11].

The original PAC is an adaptive audio compression algorithm. For the time-to-frequency transformation of an audio signal the PAC has adopted the signal adaptive oddly stacked MDCT filter bank [50, 51]. A long data block of 2048 samples ($N = 2048$) is used during stationary segments. In the presence of transient segments, a series of short 256-sample data blocks ($N = 256$) is used to eliminate pre-echo effects. In contrast, for example, to the ATRAC system, the original PAC relies on the MDCT alone rather than incorporating MDCT analysis into a hybrid filter bank structure [6, 11].

One of the major enhancements in the Enhanced PAC (EPAC) algorithm was improving the quality at lower bit rates of signal with transients [52]. In EPAC, a signal adaptive switched filter bank is used which switches between the high spectral resolution MDCT and a nonuniform (tree structured) wavelet filter bank based on time-varying characteristics of the signal. Stationary audio segments are processed by the MDCT, while transient segments by the wavelet filter bank. Finally, the MPAC algorithm extends the capabilities of the stereo PAC algorithm to the coding of multiple audio channels. A more comprehensive description of Lucent PAC/EPAC/MPAC audio compression algorithms can be found in [2, 6, 11].

2.3.3 AC-2 (AC-2A), Dolby® Digital (AC-3) and Digital Plus (E-AC-3) Audio Compression Systems

Since the late 1980s up to now, the Dolby Labs developed three generations of digital audio compression systems, specifically, a family of AC-2 (AC-2A), the Dolby® Digital (AC-3), and the Dolby® Digital Plus or Enhanced AC-3 (E-AC-3). Sony and AT&T Bell Laboratories (Lucent Technologies), the Dolby Labs also actively participated in MPEG-2 AAC research and standardization process.

2.3.3.1 AC-2 (AC-2A)

In 1991 Dolby Labs introduced the family of AC-2 (AC-2A) digital audio compression algorithms [42, 43, 47, 48]. AC-2 (AC-2A) codecs were designed as single-channel coding systems with complete channel independence when used in two-channel configurations. Four AC-2 variants were available. The first two variants were designed for low-complexity and low-delay applications, while the other two ones for higher quality at the expense of increased complexity and delay. In all AC-2 variants, the input audio signal was mapped into the frequency domain by the evenly stacked MDCT filter bank [7] with a parametric KBD windowing function [6, 11].

In variant 1, fixed 128-sample MDCT filter bank was used, i.e., $N = 128$. Variant 2 used the same filter bank, but it exploited the time redundancy across block pairs. Variant 3 used fixed 512-sample filter bank, i.e., $N = 512$, to improve the coding gain for stationary audio segments. Finally, variant 4 (the AC-2A algorithm) [42, 47] employed an adaptive switched 512/128-sample MDCT filter bank ($N = 512$ or 128) to improve coding quality for transient signals.

2.3.3.2 Dolby® Digital (AC-3)

Based on the design experience of AC-2 (AC-2A) core technology, the Dolby Labs developed a digital audio compression system of next generation, the Dolby® Digital (AC-3) multichannel audio compression algorithm [44, 45, 48] designed for digital media delivery to consumer electronic products. AC-3 is capable of delivering one to 5.1 discrete audio channels for simultaneous presentation, and it is the first multichannel surround sound codec offered to the broadcast market. First released in 1991 for cinema industry needs and standardized in 1995 by ATSC (Advanced Television Systems Committee) [45], the AC-3 went through many stages of refinements, improvements, and fine-tuning. The resulting algorithm is currently in use in a number of standard applications in consumer electronics including the North American HDTV, the DVD-Video, Digital Video Broadcasting (DVB), and Blue-ray Disc standards [1, 2, 6, 11]. The time-to-frequency transformation of audio blocks is realized as follows.

Compared to AC-2 (AC-2A), the AC-3 for the time/frequency transformation of an audio data block has adopted the oddly stacked MDCT filter bank [8] with the parametric KBD windowing function. In general, the AC-3 defines the analysis and synthesis filter banks with a variable parameter α [45]. Besides a long transform being the MDCT, AC-3 defines additional two variants of cosine-modulated filter banks called the first and second short transforms. They are actually real-valued polyphase filter banks, where all channels are shifted versions of the same prototype low-pass filter (windowing function), and these filter banks are derived directly from the type-IV discrete cosine/sine transform (DCT-IV/DST-IV) kernels [1, 11].

Unlike the MPEG-2 AAC approach [14], the AC-3 maintains the perfect reconstruction of filter banks while avoiding transitional blocks. In the AC-3 transform

block-size switching procedure [45], a long block of $N = 512$ samples or two short blocks each of $N = 256$ samples can be employed. The windowed long block is transformed when the spectrum remains stationary, or varies only slowly with time resulting in 256 unique nonzero frequency coefficients. During transients, when the signal changes rapidly in time, shorter blocks are constructed to reduce pre-echo effects by taking windowed long 512-sample block and splitting it into two adjacent half segments each containing 256 samples. The first half of long block is transformed separately from the second half of that block. Each half-block produces 128 unique nonzero frequency coefficients. This is identical to the number of frequency coefficients produced by a single long block, but with two times improved temporal resolution. Frequency coefficients from those two half-blocks are interleaved together on a coefficient-by-coefficient basis to form a single audio block of 256 coefficients being processed identically.

In the current architecture of AC-3 [44, 49] blocks of frequency coefficients are grouped into continuous frames. The AC-3 frame length is fixed at 1536 frequency coefficients per input channel corresponding to six 256-coefficients blocks. Transformed blocks are then quantized and transmitted as the so-called spectral envelope with the associated side information. A similar, mirror image procedure is applied in the decoder during signal reconstruction. The current AC-3 encoder obtains better spectral power estimation in terms of improving the fidelity through the power energy summation of the MDCT frequency coefficients and frequency coefficients of the corresponding MDST [44, 49]. The MDCT as the real part and MDST as the imaginary part compose a complex MCLT filter bank [5].

2.3.3.3 Dolby® Digital Plus (E-AC-3)

The Dolby® Digital Plus or E-AC-3 is essentially the advanced version of AC-3 providing increased coding efficiency, flexibility, and wider range of supported bit rates, expanded channel formats (up to 15.1 channels) and reproduction circumstances while preserving a high level of compatibility and interoperability with existing AC-3 system [2, 44, 46, 49]. The E-AC-3 preserves frame structure of six 256-coefficients blocks while also allows for shorter frames composed of one, two, or three frequency coefficients blocks. This feature enables to transport audio at data rates in formats limiting the amount of data per frame, such as DVD.

The E-AC-3 utilizes new powerful coding tools such as an improved filter bank, improved quantization, enhanced channel coupling with phase preservation, spectral bandwidth extension, and transient pre-noise processing. The improved filter bank is an adaptive hybrid transform (AHT) composed of two linear transforms connected in cascade [49]. The first transform is identical to that of employed in AC-3: the windowed long (MDCT) transform producing 256 unique nonzero frequency coefficients. For frames containing audio signals which are stationary, a second linear transform can optionally be applied by E-AC-3 encoder, and inverted by the decoder. It is a non-windowed, non-overlapped type-II discrete cosine transform (DCT-II). When the DCT-II is applied, six 256-coefficients blocks are converted to

a single 1536-coefficients block thereby increasing the frequency resolution and resulting in the significantly improved coding efficiency and perceptual coding performance for stationary audio signals [49]. Enhanced channel coupling process in the encoder and decoder requires the phase information for angle adjustment and therefore, besides the MDCT the corresponding MDST, i.e., complex MCLT filter bank, is also generated. The transient detector is similar but more sensitive to that of employed in the standard AC-3 encoder. However, although the E-AC-3 in the presence of a transient can be switched into AC-3 block switching mode, the E-AC-3 decoder processes transient segments in a different way by the transient pre-noise coding tool [49]. The E-AC-3 bit streams are similar in nature to AC-3 bit streams, but are not backwards compatible, i.e., they are not decodable by AC-3 decoders. Annex E of [46] specifies E-AC-3 the bit stream syntax for decoding process.

Excellent overviews of the current AC-3 and E-AC-3 codecs are presented in [1, 2, 44, 49]. In particular, in Chap. 6 the perfect reconstruction cosine/sine-modulated filter banks used in the Dolby® Digital (Plus) digital audio compression systems are discussed in detail including their definitions, general properties in the time and frequency domains, their efficient implementations, their matrix representations, relationships between them, and the latest achieved research results.

2.4 Broadcasting/Speech/Data Communication Codecs

Digital Audio Broadcasting (DAB) is a digital radio standard which has been developed in 1990s by the Eureka 147/DAB project [59]. The DAB has been designed as a universal multimedia broadcast system with the aim to replace the existing AM and FM audio broadcast services. The original DAB system is based on the MPEG-1/2 layer II audio coding standard, where the input signal is transformed from the time to frequency domain by a 32-band pseudo-QMF bank, i.e., $N = 64$ [12, 13]. The advanced DAB system (DAB+) [59] has adopted a newer audio coding standard as a source coder, the MPEG-4 HE-AAC v2 [22, 23] being the combination of MPEG-4 AAC, the SBR compression technology and Parametric Stereo coding tool. Recall that the standard MPEG-4 HE-AAC v2 or aacPlus standard employs the oddly stacked MDCT filter bank on block sizes $N = 2048$ (long block) or $N = 256$ (short block) with the sine or parametric KBD windowing functions. However, in the DAB+ system the block sizes have been changed to $N = 1920$ or $N = 240$, which are not powers of two. Actually both block sizes are composite lengths of the form $2^m \times q$, where q is an odd number. Specifically, $1920 = 2^7 \times 15$ and $240 = 2^4 \times 15$.

The MPEG-4 HE-AAC v2 as a source coder has been adopted by the Digital Radio Mondiale (DRM), universal openly standardized digital broadcasting system [58], as well as by the XM Satellite Radio broadcasting system [70] being one of two satellite-based digital radio services (XM Satellite Radio and Sirius Satellite Radio [66]) used in United States and Canada. In July 2008, XM Satellite Radio and Sirius Satellite Radio merged forming Sirius XM Radio. Further, the MPEG-4

HE-AAC v2 as the source coder has also been adopted by the extended AMR-WB+ speech coder [63, 65].

On the other hand, issued ITU-T G.722.1 [60, 68], ITU-T G.722.1C [68], G.719 [69], G.718 and G.729.1 [62, 64] speech codecs, 3GPP2 EVRC-WB [55] vocoder, and ITU-T G.EV-VBR standard [61, 67] have adopted the oddly stacked MDCT filter bank or equivalently, the MLT of the length $N = 640$ with the sine windowing function [4]. Compared to ITU-T G.722.1, in the ITU-T G.722.1C speech codec the transform size is doubled to $N = 1280$. Similarly, the block sizes are composite integers, i.e., $640 = 2^7 \times 5$, and $1280 = 2^8 \times 5$.

Recently, filter bank multicarrier (FBMC) [56] and lapped-OFDM (Orthogonal Frequency Division Multiplexing) [57] multicarrier transmission schemes for the efficient modulation, transmission, and asynchronous access in future wireless digital data communication systems such as the mobile telephony and cognitive radio have been proposed. For the time-to-frequency transformation of an input signal the approach in the FBMC scheme [56] is based on the complex MCLT filter bank with the sine windowing function [5], whereas the approach in the lapped-OFDM scheme [57] is based on the MLT [4]. In the performed simulations the transform sizes have been set to $N = 2M$, where $M = 256$ [56, 57].

2.5 Open-Source and Patent/Royalty-Free Audio/Speech Codecs

At present time, there exist several professional general purpose perceptual transform-based audio/speech codecs developed by the Xiph.Org Foundation which are completely open-source and patent/royalty-free distributed.

Perhaps the most famous is the Ogg Vorbis (or Vorbis I) multichannel audio codec [77, 78] intended for the high-quality audio and music compression. For the time-to-frequency transformation of an input audio signal, and vice versa, the Vorbis codec has adopted the oddly stacked MDCT filter bank [8] with own originally introduced the Vorbis windowing function [77, 78]. Based on signal characteristics, the Vorbis codec employs the transform block-size switching procedure. A long block of $N = 2048$ samples is used when the signal spectrum is stationary. During transients, when the signal changes rapidly in time, short blocks of the size $N = 512$ are constructed to reduce pre-echo effects. In general, in the Vorbis codec the legal block sizes to be allowed are also sizes being powers of two in range of 64–8192 samples [77].

Typically, the algorithmic delay in existing perceptual transform-based audio codecs is equal to the transform block size. Constrained-Energy Lapped Transform (CELT) audio codec [72, 73, 75, 76] provides high-quality coding, while maintaining a very low delay. Thus, it is suitable for communication applications where both high-quality and low delay are desired like the real-time interactive teleconferencing, VoiceIP, and remote live stereo music performances across the

Internet [72]. In order to minimize algorithmic delay, the CELT combines the oddly stacked MDCT filter bank [8] applied to short data blocks of size $N = 256$ with a low overlap windowing function (the so-called power complementary windowing function with reduced overlap). The windowing function with reduced overlap is constructed from the basic 512-sample Vorbis windowing function [77, 78] by zero-padding 64 values on each side, and inserting ones in the middle (128 samples) such that the resulting windowing function still satisfies the power complementarity (see configuration shown in Fig. 1 of Valin et al. [73] or in Fig. 3 of Valin et al. [75]). For this windowing and overlapping configuration the data block size to be processed is 256 samples with only 128 overlap and 64 look-ahead samples resulting in the total algorithmic delay of 384 samples. To avoid pre-echo effects, after detecting the transient the (long) data block is split into two smaller blocks and the MDCT is applied to each smaller block. The obtained spectral coefficients of two smaller MDCTs are interleaved and coded as if only one (long) MDCT was used. The last version of CELT codec provides also an optional support for other sampling rates as well as the block size of 128-sample blocks with 64 look-ahead samples [72]. In general, the CELT originally supported all even block sizes from 64 to 512, although powers of two were recommended and most CELT development was done using the block size $N = 256$. The original stand-alone CELT codec has been integrated into one layer of the OPUS codec [71, 74], and therefore it is important to note that this is now obsolete [72].

The OPUS interactive speech and audio codec [71, 74] is intended for the real-time interactive speech and music transmission over the Internet, and also for storage and streaming applications. In order to achieve good compression of both speech and music, the OPUS is a hybrid codec. One layer is based on linear prediction coding methods (especially for speech coding), while the second layer is based on the CELT compression algorithm (for music coding). Since OPUS codec incorporates Skype's SILK audio compression format, it is a modern standard for wide-band voice applications such as Skype.

2.6 Summary

Perceptual transform-based audio coding schemes developed up to now including the family of ISO/IEC MPEG audio coding standards, proprietary audio compression algorithms, broadcasting/speech/data communication codecs, as well as open-free, patent royalty-free audio/speech codecs have been briefly reviewed. In discussion, the emphasis has been imposed particularly on the adopted near-perfect QMF and perfect reconstruction cosine-modulated filter banks, processing methods, and specified transform block sizes. One can see that the employed cosine-modulated QMF banks (complex-valued or real-valued), and perfect reconstruction cosine-modulated filter banks such as the MDCT, MDST, MLT, MCLT, and LD-MDCT are applied to audio data blocks with various sizes being either a power of two (32, 64, 128, 256, 512, 1024, 2048, 4096 and even 8192), or they are

mixed-radix or composite lengths of the form $2^m \times q$, where $m > 0$ and q is an odd integer, specifically $q = 3,\ 5,\ 9$ and 15. Importantly, for a real-time implementation of each employed cosine-modulated QMF bank and perfect reconstruction cosine-modulated filter bank fast algorithm(s) are required.

References

1. M. Bosi, R.E. Goldberg, *Introduction to Digital Audio Coding and Standards*, Part II: Audio Coding Standards (Springer Science+Business Media, New York, 2003), pp. 265–430
2. V.K. Madisetti (ed.), *The Digital Signal Processing Handbook: Video, Speech, and Audio Signal Processing and Associated Standards*, 2nd edn. (CRC, Boca Raton, FL, 2010)
3. H.S. Malvar, Extended lapped transforms: properties, applications, and fast algorithms. IEEE Trans. Signal Process. **40**(11), 2703–2714 (1992)
4. H.S. Malvar, *Signal Processing with Lapped Transforms* (Artech House, Norwood, MA, 1992)
5. H. Malvar, A modulated complex lapped transform and its applications to audio processing, in *Proceedings of the IEEE ICASSP'99*, Phoenix, AR, May 1999, pp. 1421–1424
6. T. Painter, A. Spanias, Perceptual coding of digital audio. Proc. IEEE **88**(4), 451–513 (2000)
7. J.P. Princen, A.B. Bradley, Analysis/synthesis filter bank design based on time domain aliasing cancellation. IEEE Trans. Acoust. Speech Signal Process. **ASSP-34**(5), 1153–1161 (1986)
8. J.P. Princen, A.W. Johnson, A.B. Bradley, Sub-band/transform coding using filter bank designs based on time domain aliasing cancellation, in *Proceedings of IEEE ICASSP'87*, Dallas, TX, April 1987, pp. 2161–2164
9. K.R. Rao, J.J. Hwang, MPEG-1 audiovisual coder for digital storage media (Chapter 10), in *Techniques and Standards for Image, Video, Audio Coding* (Prentice-Hall, Upper Saddle River, NJ, 1996), pp. 242–265
10. M. Schnell et al., Low delay filter banks for enhanced low delay audio coding, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, October 2007, pp. 235–238
11. A. Spanias, T. Painter, V. Atti, Audio coding standards and algorithms (Chapter 10), in *Audio Signal Processing and Coding* (Wiley-Interscience, Hoboken, NJ, 2007), pp. 263–342

MPEG-1/2 Audio Coding Standards

12. Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s. Part 3: Audio, ISO/IEC JTC1/SC29/WG11 MPEG, International Standard 11172-3 (MPEG-1) (1992)
13. Information Technology – Generic Coding of Moving Pictures and Associated Audio, Part 3: Audio, ISO/IEC JTC1/SC29/WG11 MPEG, International Standard 13818-3 (MPEG-2) (1994)

MPEG–2/4 AAC Audio Coding Standards

14. M. Bosi et al., ISO/IEC MPEG-2 advanced audio coding, in *101st AES Convention*, Los Angeles, CA, November 1996. Preprint #4382. Also published in J. Audio Eng. Soc. **45**(10), 789–813 (1997)

15. Information Technology – Generic Coding of Moving Pictures and Associated Audio Information, Subpart 7: Advanced Audio Coding (AAC), ISO/IEC JTC1/SC29/WG11 MPEG, International Standard 13818-7 (MPEG-2 AAC) (1997)
16. Information Technology – Coding of Audio-Visual Objects, Part 3: Audio, ISO/IEC JTC1/SC29/WG11 MPEG, International Standard 14496-3 (MPEG-4 Audio) (1999)

MPEG-4 AAC-LD Audio Coding Standard

17. E. Allamanche, R. Geiger, J. Herre, T. Sporer, MPEG-4 low delay audio coding based on the AAC codec, in *106th AES Convention*, Munich, May 1999. Preprint #4929
18. M. Lutzky, G. Schuller, M. Gayer, U. Krämer, S. Wabnik, A guideline to codec delay, in *116th AES Convention*, Berlin, May 2004. Preprint #6062
19. M. Lutzky, M. Schnell, M. Schmidt, R. Geiger, Structural analysis of low latency audio coding schemes, in *119th AES Convention*, New York, NY, October 2005. Preprint #6601

MPEG-4 HE-AAC Audio Coding Standard

20. A.C. den Brinker et al., An overview of the coding standard MPEG-4 audio Amendments 1 and 2: HE-AAC, SSC and HE-AAC v2. EURASIP J. Audio Speech Music Process. Article ID 468971, 21 (2009)
21. J. Herre, M. Dietz, MPEG-4 High-Efficiency AAC coding. IEEE Signal Process. Mag. **25**(3), 137–142 (2008)
22. Information Technology – Coding of Audio-Visual Objects – Part 3: Audio, Subpart 4: General Audio Coding (GA)-AAC, TwinVQ, BSAC. ISO/IEC 14496–3:2005(E) (2005)
23. M. Wolters, K. Kjörling, D. Homm, H. Purnhagen, A closer look into MPEG-4 High Efficiency AAC, in *115th AES Convention*, New York, NY, October 2003. Preprint #5871

MPEG-4 AAC-ELD Audio Coding Standard

24. Information Technology – Coding of Audio-Visual Objects – Part 3: Audio, Amendment 9: Enhanced Low Delay AAC. ISO/IEC 14496–3:2005/FDAM 9:2007(E), N9499, Shenzhen, October 2007
25. M. Lutzky, M.L. Valero, M. Schnell, J. Hilpert, AAC-ELD v2 – The new state of the art in high quality communication audio coding, in *131st AES Convention*, New York, NY, October 2011. Preprint #8516
26. M. Schnell et al., Enhanced MPEG-4 low delay AAC – Low bitrate high quality communication, in *122nd AES Convention*, Vienna, May 2007. Preprint #6998
27. M. Schnell et al., MPEG-4 enhanced low delay AAC – A new standard for high quality communication, in *125th AES Convention*, San Francisco, CA, October 2008. Preprint #7503

MPEG-4 SLS and HD-AAC/SLS Scalable Lossless Audio Coding Standards

28. R. Geiger, G. Schuller, J. Herre, R. Sperschneider, T. Sporer, Scalable perceptual and lossless audio coding based on MPEG-4 AAC, in *115th AES Convention*, New York, NY, October 2003. Preprint #5868

29. R. Geiger, R. Yu, J. Herre, S. Rahardja, S.-W. Kim, X. Lin, M. Schmidt, ISO/IEC MPEG-4 high-definition scalable advanced audio coding. J. Audio Eng. Soc. **55**(1)/2, 27–43 (2007)
30. ISO/IEC 14496-3:2005/Amd.3:2006, Coding of Audio-Visual Objects – Part 3: Audio, Amendment 3: Scalable Lossless Coding (SLS). International Standards Organization, Geneva (2006)
31. R. Yu, R. Geiger, S. Rahardja, J. Herre, X. Lin, H. Huang, MPEG-4 scalable to lossless audio coding, in *117th AES Convention*, San Francisco, CA, October 2004. Preprint #6183
32. R. Yu, S. Rahardja, X. Lin, C.C. Ko, A fine granular scalable to lossless audio coding. IEEE Trans. Audio Speech Lang. Process. **14**(4), 1352–1363 (2006)

MPEG-D USAC: Unified Speech and Audio Coding

33. B. Edler, S. Disch, S. Bayer, G. Guillaume, R. Geiger, A time-warped MDCT approach to speech transform coding, in *126th AES Convention*, Munich, May 2009. Preprint #7710
34. C.R. Helmrich et al., Efficient transform coding of two-channel audio signals by means of complex-valued stereo prediction, in *Proceedings of the IEEE ICASSP'2011*, Prague, May 2011, pp. 497–500
35. A. Heuerberger, G. Elst, R. Hanke (eds.), MPEG unified speech and audio coding – Bridging the gap, in *Microelectronic Systems: Circuits, Systems and Applications* (Springer, Berlin, 2011), pp. 343–353
36. ISO/IEC 23003—3:2012, MPEG audio technologies, Part 3: Unified Speech and Audio Coding, Geneva, January 2012
37. K. Kikuri, N. Naka, MPEG Unified speech and audio coding enabling efficient coding of both speech and music. NTT DOCOMO Tech. J. **13**(3), 17–22 (2011)
38. M. Neuendorf et al., A novel scheme for low bit rate Unified Speech and Audio Coding – MPEG RM0, in *126th AES Convention*, Munich, May 2009. Preprint #7713
39. M. Neuendorf et al., Unified speech and audio coding scheme for high quality at low bitrates, in *Proceedings of the IEEE ICASSP'2009*, Taipei, April 2009, pp. 1–4
40. M. Neuendorf et al., The ISO/MPEG Unified Speech and Audio Coding standard – Consistent high quality for all content types and at all bit rates, in *132nd AES Convention*, Budapest, April 2012. Preprint #8654. Also published in J. Audio Eng. Soc. **61**(12), 956–977 (2013)
41. S. Quackenbush, MPEG unified speech and audio coding. IEEE MultiMedia **20**(2), 72–78 (2013)

Proprietary Audio Compression Algorithms

42. M. Bosi, G.A. Davidson, High-quality, low-rate audio transform coding for transmission and multimedia applications, in *93rd AES Convention*, San Francisco, CA, December 1992. Preprint# 3365
43. G.A. Davidson, L.D. Fielder, M. Antill, Low-complexity transform coder for satellite link applications, in *89th AES Convention*, New York, NY, September 1990. Preprint# 2966
44. G.A. Davidson, M.A. Isnardi, L.D. Fielder, M.S. Goldman, C.C. Todd, ATSC video and audio coding. Proc. IEEE **94**(1), 60–76 (2006)
45. Digital Audio Compression (AC-3) ATSC Standard, Document A/52/10 of Advanced Television Systems Committee (ATSC), Audio Specialist Group T3/S7, Washington, DC, December 1995
46. Digital Audio Compression Standard (AC-3, E-AC-3), Revision B, Document A/52B of Advanced Television Systems Committee (ATSC), Washington DC, December 2012

47. L.D. Fielder, G.A. Davidson, AC-2: a family of low complexity transform-based music coders, in *Proceedings of the 10th International AES Conference: Images of Audio*, London, September 1991, pp. 55–70
48. L.D. Fielder, D.P. Robinson, AC-2 and AC-3: the technology and its applications, in *5th Australian Regional Convention*, Sydney, April 1995. Preprint #4022
49. L.D. Fielder et al., Introduction to Dolby Digital Plus, an enhancement to the Dolby digital coding system, in *117th AES Convention*, San Francisco, CA, October 2004. Preprint #6196
50. J.D. Johnson, A.J. Ferreira, Sum-difference stereo transform coding, in *Proceedings of the IEEE ICASSP'92*, vol. II, San Francisco, CA, March 1992, pp. 569–572
51. J. Johnson et al., AT&T perceptual audio coder (PAC), in *Collected Papers on Digital Audio Bit-Rate Reduction*, ed. by N. Gilchrist, C. Grewin (Audio Engineering Society, New York, 1996), pp. 73–81
52. D. Sinha, J.D. Johnson, Audio compression at low bit rates using a signal adaptive switched filterbank, in *Proceedings of the IEEE ICASSP'96*, Atlanta, GA, May 1996, pp. 1053–1056
53. K. Tsustsui at al., ATRAC: adaptive transform acoustics coding for MiniDisc, in *93rd AES Convention*, San Francisco, CA, October 1992. Preprint #3456
54. T. Yoshida, The rewritable MiniDisc system. Proc. IEEE **82**(10), 1492–1500 (1994)

Broadcasting/Speech/Data Communication Codecs

55. 3GGP2 C.S0014–C v1.0, Enhanced variable rate codec, speech service Option 3, 68 and 70 for wide-band spread spectrum digital systems (2007)
56. M. Bellanger, D. Matera, M. Tanda, A filter bank multicarrier scheme running at symbol rate for future wireless systems, in *Proceedings of the IEEE Wireless Telecommunications Symposium (WTS'2015)*, New York, NY, April 2015, pp. 1–5
57. M. Bellanger, D. Matera, M. Tanda, Lapped-OFDM as an alternative to CP-OFDM for 5G asynchronous access and cognitive radio, in Proceedings of the IEEE 81st Vehicular Technology Conference (VTC Spring), Glasgow, May 2015, pp. 1–5
58. Digital Radio Mondiale (DRM): System Specification, ETSI ES 201 980 v3.1.1 (2009–08), ETSI Standard, August 2009 (available on web site http://www.drm.org)
59. W. Hoeg, T. Lauterbach (eds.), Audio services and applications (Chapter 3), in *Digital Audio Broadcasting: Principles and Applications of DAB, DAB+ and DMB*, 3rd edn. (Wiley, Chichester, 2009), pp. 93–165
60. ITU-T Recommendation G.722.1 Annex C, Low-complexity coding at 24 and 32 kbit/s for hands-free operation in systems with low frame loss. Annex C: 14 kHz Mode at 24, 32, and 48 kbit/s, May 2005
61. ITU-T SG16 Q9 – Contribution 199: extended high-level description of the Q9 EV-VBR baseline codec (2007)
62. L. Laaksonen et al., Super wide-band extension of G.718 and G.729.1 speech codec, in *Proceedings of 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010*, Makuhari, September 2010
63. J. Mäkinen et al., AMR-WB+: a new audio coding standard for 3rd generation mobile audio services, in *Proceedings of the IEEE ICASSP'2005*, vol. II, Philadelphia, PA, March 2005, pp. 1109–1112
64. S. Ragot et al., ITU-T G.729.1: an 8–32 kbit/s scalable coder interoperable with G.729 for wideband telephony and voice IP, in *Proceedings of the IEEE ICASSP'2007*, Honolulu, HI, April 2007, pp. 529–532
65. R. Salami et al., Extended AMR-WB for high-quality audio on mobile devices. IEEE Commun. Mag. **44**(5), 90–97 (2006)
66. Sirius Satellite Radio, Available on web site: http://www.siriusradio.com

67. T. Vaillancourt et al., ITU-T EV-VBR: a robust 8–32kbit/s scalable coder for error prone telecommunication channels, in *Proceedings of the 16th European Signal Processing Conference*, Lausanne, August 2008
68. M. Xie, D. Lindbergh, P. Chu, From ITU-T G.722.1 to ITU-T G.722.1 Annex C: a new low-complexity 14kHz bandwidth audio coding standard, in *Proceedings of the IEEE ICASSP'2006*, vol. 5, Toulouse, May 2006, pp. 173–176. Also published in J. Multimedia **2**(2), 65–76 (2007)
69. M. Xie, P. Chu, A. Taleb, M. Briand, ITU-T G.719: a new low-complexity full-band (20kHz) audio coding standard for high quality conversational applications, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA'2009)*, New Paltz, NY, October 2009, pp. 265–268
70. XM Satellite Radio, Available on web site: http://www.xmradio.com

Open-Source and royalty-Free Audio/Speech Codecs

71. OPUS interactive audio/speech codec, 2016. Available on web sites: www.vorbis.com or www.opus-codec.org
72. The CELT ultra-low delay audio codec, February 2011. Available on web sites: www.vorbis.com or www.celt-codec.org
73. J.-M. Valin, T.B. Terriberry, G. Maxwell, A full-bandwidth audio codec with low complexity and very low delay, in *Proceedings of the 17th European Signal Processing Conference (EUSIPCO'2009)*, Glasgow, August 2009, pp. 1254–1258
74. J.M. Valin, K. Vos, T.B. Terriberry, Definition of the OPUS audio codec, Internet Engineering Task Force (IETF). RFC 6716 Standard Specification, September 2012. Available on web site: www.vorbis.com
75. J.-M. Valin, T.B. Terriberry, C. Montgomery, G. Maxwell, A high-quality speech and audio codec with less than 10 ms delay. IEEE Trans. Audio Speech Lang. Process. **18**(1), 58–67 (2010)
76. J.-M. Valin, G. Maxwell, T.B. Terriberry, C. Montgomery, K. Vos, High-quality, low-delay music coding in the Opus codec, in *135th AES Convention*, New York, NY, October 2013. Preprint #8942
77. Vorbis I specification, Xiph.Org Foundation (2015). Available on web site: www.vorbis.com
78. K. Wright, Notes on Ogg Vorbis and the MDCT, Draft document available on web site: www.free-comp-shop.com/vorbis.html (2003), 7 pp.

Chapter 3
MDCT/MDST, MLT, ELT, and MCLT Filter Banks: Definitions, General Properties, and Matrix Representations

3.1 Introduction

Multirate analysis/synthesis critically sampled, maximally decimated filter banks involving near-perfect (pseudo) and perfect reconstruction quadrature mirror filter (QMF) banks [75, 79], have received widespread attention in speech and audio sub-band coding. In general, the analysis filter bank splits the input signal into a number of sub-bands via the analysis filters, while the synthesis filter bank recombines the sub-band signals via the synthesis filters to (approximately or exactly) reconstruct the original signal in the overlapped part of adjacent data blocks. The theory and design methods of critically sampled, maximally decimated filter banks are covered in [79].

Among the QMF filter banks, the cosine/sine-modulated filter banks [1–14], [79] and the cosine-modulated pseudo-QMF banks belonging to the class of modulated filter banks [5, 6, 8, 14] have been studied extensively. Due to their attractive features (simple structure, analysis and synthesis filters are of equal length, low computational complexity), they have received a great interest in audio coding applications. In fact, the cosine-modulated pseudo QMF and cosine/sine-modulated filter banks with perfect reconstruction property are employed in the international speech and audio coding standards and proprietary audio compression algorithms [32, 38, 41]. The modulated filter banks achieving the perfect reconstruction property can be obtained by cosine or sine modulation of a linear-phase prototype filter or equivalently, of a windowing function, when certain constraints are imposed on the prototype filter [6]. On the other hand, by properly choosing the phases of modulation cosine/sine functions in a pseudo-QMF bank and assuming that the length of analysis/synthesis filters is to be twice the number of sub-bands, whereby analysis filters are time-reverse versions of the synthesis filters, the pseudo-QMF bank will achieve the perfect reconstruction property [6, 63]. It is worthwhile here to get into the reader attention for further inspiration the unified or generalized forms of cosine-modulated filter banks [3, 4], a family of extended unique filter

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1_3

banks containing all known M-band modulated filter banks [9, 10], as well as a universal approach to the description of modulated filter banks, specifically, the real-valued cosine- and sine-based versions defined via the generalized discrete cosine transform (GDCT) and the corresponding generalized discrete sine transform (GDST) [1, 2].

The oddly and evenly stacked modified discrete cosine transform (MDCT) and the corresponding modified discrete sine transform (MDST), the modulated lapped transform (MLT), the extended lapped transforms (ELTs), and their biorthogonal versions are real-valued cosine/sine-modulated filter banks satisfying the perfect reconstruction property. The modulated complex lapped transform (MCLT) is the complex-valued filter bank whose real part is the MLT or oddly stacked MDCT, and the imaginary part is the oddly stacked MDST. All these filter banks, except the evenly stacked MDCT/MDST, belong to the class of lapped transforms, and can also be viewed as block transforms, in which the basis functions overlap the adjacent blocks by 50%. In this chapter, definitions, general mathematical properties, and matrix representations of the MDCT/MDST, MLT, ELT, and MCLT filter banks are presented. Principally, for each analysis/synthesis filter bank the chapter presents:

- Its original derivation and definitions in the form of block transforms.
- General and special mathematical properties of forward/backward block transforms both in the time and frequency domains including relationships between/among them.
- (Block) matrix representations, which are powerful and rigorous mathematical tools to investigate structure, properties, and efficient implementations, generally, of any filter bank.

In order to obtain the perfect reconstruction property of an analysis/synthesis filter bank, the necessary and sufficient conditions imposed on the analysis and synthesis windowing functions play an important role. Therefore, additionally the following are derived and/or discussed in detail:

- Windowing procedure and perfect reconstruction conditions in the case of identical analysis and synthesis windowing functions.
- Design of a windowing function including definitions of commonly used windowing functions in audio coding applications.
- Biorthogonal conditions for perfect reconstruction in the case of nonidentical analysis and synthesis windowing functions.
- Adaptive switching of transform block sizes and windowing functions.
- General perfect reconstruction conditions for the ELT filter bank with multiple overlapping factor both for the orthogonal and biorthogonal cases.

Thus, this chapter provides the complete information for designing and constructing the cosine/sine-modulated analysis/synthesis filter bank satisfying perfect reconstruction for audio coding applications.

3.2 MDCT and MDST Filter Banks

The MDCT and corresponding MDST are perfect reconstruction cosine/sine-modulated analysis/synthesis filter banks based on the concept of time domain aliasing cancellation (TDAC). Therefore, they are frequently called the TDAC transforms, or alternatively, the critically sampled single-sideband (SSB) analysis/synthesis filter banks providing perfect reconstruction of a signal from a set of critically sampled analysis signals by a weighted overlap/add method [18, 24].

Princen, Bradley, and Johnson defined two types of the MDCT, specifically for an evenly stacked [17, 18] and for an oddly stacked analysis/synthesis system [23, 24]. Both systems are very similar. The main difference lies in transform operations. In general, the MDCT/MDST for the evenly stacked system is defined by a transform kernel

$$\cos\left[\frac{\pi}{2}\, m - \omega_k\,(n + n_0)\right], \qquad \omega_k = \frac{2\pi k}{N}, \tag{3.1}$$

while the MDCT for the oddly stacked system is defined by a transform kernel

$$\cos\left[\omega_k\,(n + n_0)\right], \qquad \omega_k = \frac{2\pi}{N}\left(k + \frac{1}{2}\right), \tag{3.2}$$

where N is the length of a data block, n is the time index, k is the frequency index, m is the index of the data block, and ω_k denotes the center frequencies of channels. For both systems cosine modulation functions in (3.1) and (3.2) include a time phase factor n_0 having a value of $\frac{N}{4} + \frac{1}{2}$ which is necessary for perfect reconstruction. The terms "evenly" and "oddly" stacked systems come simply from the fact that the center frequencies ω_k of the system (3.1) indicate the even channel stacking arrangement, while the center frequencies ω_k of the system (3.2) indicate the odd channel stacking arrangement.

The evenly and oddly stacked MDCT/MDST filter banks can also be viewed as block transforms, in which the basis functions overlap the adjacent data blocks by 50%. The oddly stacked system has some advantages compared with the evenly stacked system. It is uniform, i.e., it has equally spaced bands of equal bandwidth, and its computational structure though more complex, requires the application of the MDCT only, while the evenly stacked system is nonuniform (it has half width bands at normalized frequencies of 0 and π), and moreover, alternate applications of the MDCT and corresponding MDST are required. For these reasons, the oddly stacked system is preferred in the audio coding applications [23, 24].

3.2.1 Evenly Stacked MDCT/MDST Filter Banks

The evenly stacked, critically sampled SSB analysis filter bank has been originally derived as a block transform operation in the following form [17, 18]

$$X_k^{(m)} = (-1)^{mk} \cos\left(\frac{\pi m}{2}\right) \sum_{r=0}^{P-1} x_{mM+r}^{(m)} \, h_{P-1-r} \cos\left[\frac{2\pi k}{N}(r+n_0)\right] + (-1)^{mk} \sin\left(\frac{\pi m}{2}\right) \sum_{r=0}^{P-1} x_{mM+r}^{(m)} \, h_{P-1-r} \sin\left[\frac{2\pi k}{N}(r+n_0)\right],$$
$$k = 0, 1, \ldots, N-1, \qquad M = \frac{N}{2}, \qquad \frac{N}{2} \le P \le N, \tag{3.3}$$

where $\{X_k^{(m)}\}$ are channel signals, and $\{x_k^{(m)}\}$ are samples of the mth data block. $\{h_{P-1-r}\}$ is a time reversed finite impulse response filter of the length P, or the analysis windowing function. The value of P in (3.3) indicates that an overlap occurs between adjacent data blocks ($P = N$ implies the maximum overlap). Cosine and sine modulation functions $\cos\left[\frac{2\pi k}{N}(r+n_0)\right]$ and $\sin\left[\frac{2\pi k}{N}(r+n_0)\right]$ in (3.3) are referred to as the evenly stacked MDCT and MDST transform kernels, respectively. It can be seen that for a given value of the block index m only one of the terms in (3.3) is nonzero. When m is even, the channel signals are defined by the MDCT of windowed data sequence and modified by $(-1)^{mk}$. When m is odd, the channel signals are defined by the MDST of windowed data sequence and modified by $(-1)^{mk}$ [18].

The evenly stacked, critically sampled SSB synthesis filter bank has been originally derived as a block transform operation in the following form [17, 18]

$$\hat{x}_r^{(m)} = \cos\left(\frac{\pi m}{2}\right) \frac{1}{N} \sum_{r=0}^{N-1} (-1)^{mk} \, X_k^{(m)} \cos\left[\frac{2\pi k}{N}(r+n_0)\right] + \sin\left(\frac{\pi m}{2}\right) \frac{1}{N} \sum_{r=0}^{N-1} (-1)^{mk} \, X_k^{(m)} \sin\left[\frac{2\pi k}{N}(r+n_0)\right],$$
$$r = 0, 1, \ldots, N-1, \tag{3.4}$$

where $\{\hat{x}_r^{(m)}\}$ are time domain aliased samples of the mth data block. From (3.4) and with respect to (3.5), one can see that when m is even, then $\{\hat{x}_r^{(m)}\}$ is the windowed backward MDCT of the modified channel signals, while when m is odd, it is the windowed backward MDST of the modified channel signals.

After applying the analysis MDCT/MDST filter bank followed by the synthesis MDCT/MDST filter bank, the time domain aliasing distortion is introduced. Assuming that the perfect reconstruction conditions hold (see Sect. 3.2.4), then for

the phase factor $n_0 = \frac{N}{4} + \frac{1}{2}$, the time domain aliasing is canceled and the original data sequence $\{x_r^{(m)}\}$ is recovered (i.e., the TDAC is achieved) from two adjacent time domain aliased data sequences $\{\hat{x}_r^{(m)}\}$ and $\{\hat{x}_r^{(m+1)}\}$ by the so-called windowing and overlap/add procedure as [18]

$$x_r^{(m+1)} = x_{\frac{N}{2}+r}^{(m)} = g_{\frac{N}{2}+r}\,\hat{x}_{\frac{N}{2}+r}^{(m)} + g_r\,\hat{x}_r^{(m+1)}, \qquad r = 0, 1, \ldots, \frac{N}{2} - 1, \tag{3.5}$$

where $\{g_r\}$ is the synthesis windowing function. For simplicity, in applications the modifications by $(-1)^{mk}$ in the analysis and synthesis MDCT and MDST filter banks are removed. More technical details about the evenly stacked system can be found in [18].

The windowing procedure and perfect reconstruction conditions imposed on the analysis and synthesis windowing functions $\{h_r\}$ and $\{g_r\}$ are discussed in detail in Sect. 3.2.4. The design of windowing functions is discussed in Sect. 3.2.5.

3.2.1.1 MDCT and MDST Block Transforms

In general, when investigating general mathematical properties of the analysis and synthesis filter banks both in time and frequency domains, or when developing fast computational structures for their efficient implementation, they are frequently considered as the block transforms applied to a single windowed data block. Consequently, without loss of generality the superscript m denoting the data block index is omitted. In the following the evenly stacked forward and backward MDCT and MDST block transforms are defined, and their basic symmetry properties both in time and frequency domains are investigated.

Let $\{x_n\}$, $n = 0, 1, \ldots, N-1$ represent a windowed input data sequence, and N is assumed to be an even integer. Based on (3.3) and (3.4), the forward and backward MDCT block transforms are, respectively, defined as [18]

$$c_k^E = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(2n + 1 + \frac{N}{2}\right)k\right], \quad k = 0, 1, \ldots, N-1, \quad c_{\frac{N}{2}}^E = 0, \tag{3.6}$$

$$\hat{x}_n^{E-\text{MDCT}} = \frac{1}{N}\sum_{k=0}^{N-1} c_k^E \cos\left[\frac{\pi}{N}\left(2n + 1 + \frac{N}{2}\right)k\right], \quad n = 0, 1, \ldots, N-1, \tag{3.7}$$

while the corresponding forward and backward MDST block transforms are, respectively, defined as [18]

$$s_k^E = \sum_{n=0}^{N-1} x_n \sin\left[\frac{\pi}{N}\left(2n + 1 + \frac{N}{2}\right)k\right], \quad k = 0, 1, \ldots, N-1, \quad s_0^E = 0, \tag{3.8}$$

$$\hat{x}_n^{E-\text{MDST}} = \frac{1}{N}\sum_{k=0}^{N-1} \overset{E}{s}_k \sin\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)k\right], \quad n=0,1,\ldots,N-1, \tag{3.9}$$

where $\{\overset{E}{c}_k\}/\{\overset{E}{s}_k\}$ are MDCT/MDST transform coefficients. The notations $\hat{x}_n^{E-\text{MDCT}}$ in (3.7) and $\hat{x}_n^{E-\text{MDST}}$ in (3.9) emphasize the fact that the data sequences recovered by appropriate backward transforms do not correspond to the original data sequence. In the context of TDAC analysis/synthesis filter banks the distorted sequences $\{\hat{x}_n^{E-\text{MDCT}}\}$ and $\{\hat{x}_n^{E-\text{MDST}}\}$ are said to be time domain aliased [18].

We recall that in evenly stacked system alternate MDCT/MDST computations are required. The MDCT can be recognized as the transform operation when the time index m of a data block is even, while the MDST can be recognized as the transform operation when the time index m of a data block is odd.

3.2.1.2 Symmetry Properties of the MDCT and MDST Block Transforms

In order to investigate the symmetry properties of evenly stacked MDCT and MDST block transforms, let us substitute $N-k$ for k into (3.6) and we get

$$\overset{E}{c}_{N-k} = \sum_{n=0}^{N-1} x_n \cos\left[\pi\left(2n+1+\frac{N}{2}\right)\right]\cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)k\right],$$
$$k=1,2,\ldots,\frac{N}{2}-1. \tag{3.10}$$

The expression $(2n+1+\frac{N}{2})$ in $\cos\left[\pi(2n+1+\frac{N}{2})\right]$ under the sum of (3.10) may be odd or even depending on a value of $\frac{N}{2}$ being even or odd, respectively. Thus, we have

$$\overset{E}{c}_{N-k} = \begin{cases} -\overset{E}{c}_k, & \text{if } (2n+1+\frac{N}{2}) \text{ is odd}, \quad k=1,2,\ldots,\frac{N}{2}-1, \\ \overset{E}{c}_k, & \text{if } (2n+1+\frac{N}{2}) \text{ is even}, \quad k=1,2,\ldots,\frac{N}{2}-1, \end{cases} \tag{3.11}$$

implying that the MDCT coefficients have either the odd anti-symmetry property when $\frac{N}{2}$ is even, or the odd symmetry property when $\frac{N}{2}$ is odd. The odd symmetric and anti-symmetric sequences are defined in Appendix G.1.

On the other hand, substituting $N-k$ for k into (3.8), we get

$$\overset{E}{s}_{N-k} = -\sum_{n=0}^{N-1} x_n \cos\left[\pi\left(2n+1+\frac{N}{2}\right)\right]\sin\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)k\right],$$
$$k=1,2,\ldots,\frac{N}{2}. \tag{3.12}$$

Similarly, the expression $(2n+1+\frac{N}{2})$ in $\cos\left[\pi(2n+1+\frac{N}{2})\right]$ under the sum of (3.12) may be odd or even depending on a value of $\frac{N}{2}$ being even or odd, respectively. Thus, we have

$$s^{E}_{N-k} = \begin{cases} s^{E}_{k}, & \text{if } (2n+1+\frac{N}{2}) \text{ is odd}, \quad k = 1,2,\ldots,\frac{N}{2}-1, \\ -s^{E}_{k}, & \text{if } (2n+1+\frac{N}{2}) \text{ is even}, \quad k = 1,2,\ldots,\frac{N}{2}-1, \end{cases} \tag{3.13}$$

implying that the MDST coefficients have either the odd symmetry property when $\frac{N}{2}$ is even, or the odd anti-symmetry property when $\frac{N}{2}$ is odd. Knowing a priori that $c^{E}_{\frac{N}{2}} = s^{E}_{0} = 0$, then from (3.11) and (3.13) it follows that only $\frac{N}{2}+1$ MDCT/MDST coefficients are unique, whereby $\frac{N}{2}$ MDCT/MDST coefficients are nonzero.

The time domain aliased data sequences $\{\hat{x}^{E-\text{MDCT}}_{n}\}$ and $\{\hat{x}^{E-\text{MDST}}_{n}\}$ recovered by the backward MDCT/MDST have the following local symmetries

$$\hat{x}^{E-\text{MDCT}}_{n} = \hat{x}^{E-\text{MDCT}}_{\frac{N}{2}-1-n}, \qquad \hat{x}^{E-\text{MDCT}}_{\frac{N}{2}+n} = \hat{x}^{E-\text{MDCT}}_{N-1-n}, \tag{3.14}$$

$$\hat{x}^{E-\text{MDST}}_{n} = -\hat{x}^{E-\text{MDST}}_{\frac{N}{2}-1-n}, \qquad \hat{x}^{E-\text{MDST}}_{\frac{N}{2}+n} = -\hat{x}^{E-\text{MDST}}_{N-1-n},$$

$$n = 0,1,\ldots,\frac{N}{4}-1. \tag{3.15}$$

The symmetry properties (3.14) and (3.15) can be simply verified by proper substitution into (3.7) and (3.9). It can be seen that $\{\hat{x}^{E-\text{MDCT}}_{n}\}$ exhibits the even symmetries both in the first and second half, whereas $\{\hat{x}^{E-\text{MDST}}_{n}\}$ exhibits the odd symmetries both in the first and second half. From an algorithmic point of view this means that it is sufficient to compute only the time domain aliased samples $\{\hat{x}^{E-\text{MDCT}}_{n}\}$, $\{\hat{x}^{E-\text{MDCT}}_{\frac{N}{2}+n}\}$ and $\{\hat{x}^{E-\text{MDST}}_{n}\}$, $\{\hat{x}^{E-\text{MDST}}_{\frac{N}{2}+n}\}$ for $n = 0,1,\ldots,\frac{N}{4}-1$ by the backward MDCT/MDST. Note that N must be integer divisible by 4.

For a correct matrix representation taking into account properties (3.11) and (3.13), the evenly stacked MDCT and MDST block transforms are redefined by introducing scaling factors ϵ_k and τ_k into (3.7) and (3.9), respectively. The forward and backward MDCT block transforms are, respectively, redefined as [15]

$$c^{E}_{k} = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)k\right], \qquad k = 0,1,\ldots,\frac{N}{2}-1, \quad c^{E}_{\frac{N}{2}} = 0, \tag{3.16}$$

$$\hat{x}^{E-\text{MDCT}}_{n} = \frac{1}{N}\sum_{k=0}^{\frac{N}{2}-1} \epsilon_k\, c^{E}_{k} \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)k\right], \qquad n = 0,1,\ldots,N-1, \tag{3.17}$$

where $\epsilon_0 = 1$, and $\epsilon_k = 2$ for $k = 1, 2, \ldots, \frac{N}{2} - 1$. The corresponding forward and backward MDST block transforms are, respectively, redefined as [15]

$$\overset{E}{s}_k = \sum_{n=0}^{N-1} x_n \ \sin\left[\frac{\pi}{N}\left(2n + 1 + \frac{N}{2}\right) k\right], \qquad k = 1, 2, \ldots, \frac{N}{2}, \quad \overset{E}{s}_0 = 0, \tag{3.18}$$

$$\hat{x}_n^{E-\text{MDST}} = \frac{1}{N} \sum_{k=1}^{\frac{N}{2}} \tau_k \ \overset{E}{s}_k \ \sin\left[\frac{\pi}{N}\left(2n + 1 + \frac{N}{2}\right) k\right], \qquad n = 0, 1, \ldots, N-1, \tag{3.19}$$

where $\tau_{\frac{N}{2}} = 1$, and $\tau_k = 2$ for $k = 1, 2, \ldots, \frac{N}{2} - 1$.

3.2.1.3 Relation Between the MDCT and MDST Block Transforms

In order to obtain the relation between the evenly stacked MDCT and MDST block transforms, let us substitute $\frac{N}{2} - k$ for k into (3.16), and we get [16, 19]

$$\overset{E}{c}_{\frac{N}{2}-k} = (-1)^{\frac{N}{4}} \sum_{n=0}^{N-1} (-1)^n \ x_n \ \sin\left[\frac{\pi}{N}\left(2n + 1 + \frac{N}{2}\right) k\right], \quad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{3.20}$$

Equation (3.20) implies that the MDST coefficients $\{\overset{E}{s}_k\}$ can be simply obtained from the MDCT ones only by sign changes applied to odd-indexed samples in the original data sequence, and after the MDCT computation, the MDST coefficients are in reverse order. The relation between the MDCT and MDST block transforms results in a simple method to compute alternating MDCT/MDST using only one fast algorithm for the MDCT computation.

Similarly, substituting $\frac{N}{2} - k$ for k into (3.18), we get

$$\overset{E}{s}_{\frac{N}{2}-k} = (-1)^{\frac{N}{4}} \sum_{n=0}^{N-1} (-1)^n \ x_n \ \cos\left[\frac{\pi}{N}\left(2n + 1 + \frac{N}{2}\right) k\right], \quad k = 1, 2, \ldots, \frac{N}{2}. \tag{3.21}$$

Note that for correct relation between the MDCT and MDST block transforms in (3.20) and (3.21) N must be integer divisible by 4.

3.2.1.4 Relation Between the MDCT/MDST and DFT

The evenly stacked N-point MDCT and MDST block transforms are closely related to N-point complex DFT, or its fast implementation, FFT. The forward and backward MDCT/MDST transform kernels can be expressed as a DFT with appropriate pre- and post-twiddle (rotation) operations. Exploiting the fact that FFT operates on an array of complex components, by packing one windowed data block in the real part of the FFT and the other in the imaginary part, two adjacent overlapped data blocks can be transformed simultaneously as [16, 17]

$$f_k = e^{-i\,(\frac{\pi}{2}+\frac{\pi}{N})k} \sum_{n=0}^{N-1} (x_n + i\,x_{n+\frac{N}{2}})\; e^{-i\,\frac{2\pi nk}{N}}, \quad i = \sqrt{-1}, \quad k = 0, 1, \ldots, N-1. \tag{3.22}$$

After the FFT computation and post-twiddle operations, the MDCT and MDST coefficients are given by

$$\begin{aligned} c_k^E &= -\frac{1}{2}\,(\Re e\,\{f_{N-k}\} - \Re e\,\{f_k\}), \qquad c_0^E = f_0, \\ s_k^E &= \frac{1}{2}\,(\Re e\,\{f_k\} + \Re e\,\{f_{N-k}\}), \qquad s_0^E = 0, \quad k = 1, \ldots, \frac{N}{2}. \end{aligned} \tag{3.23}$$

Equations (3.22) and (3.23) immediately also define an algorithm for the implementation of simultaneous computation of the MDCT and MDST block transforms via the complex FFT.

3.2.1.5 Matrix Representations of the MDCT and MDST Block Transforms

An alternative way to represent the MDCT and MDST block transforms is in matrix-vector form. Matrix representations are very powerful tools to analyze MDCT/MDST characteristics of the single windowed data block both in time and frequency domains. In particular, on the basis of matrix representations the concept of TDAC is better understood.

Consider the evenly stacked MDCT and MDST block transforms defined by (3.16) and (3.18), respectively. The symmetry properties (3.11) and (3.13) imply that only $\frac{N}{2}$ rows of MDCT and MDST matrices are linear independent. Therefore, let $\boldsymbol{C}^E_{\frac{N}{2}\times N}$ and $\boldsymbol{S}^E_{\frac{N}{2}\times N}$, be the $\frac{N}{2} \times N$ MDCT and MDST matrices, respectively. Then $[\boldsymbol{C}^E_{\frac{N}{2}\times N}]^T = \boldsymbol{C}^E_{N\times\frac{N}{2}}$ and $[\boldsymbol{S}^E_{\frac{N}{2}\times N}]^T = \boldsymbol{S}^E_{N\times\frac{N}{2}}$, where T denotes transposition. Next, let $\mathbf{x} = [x_0, x_1, \ldots, x_{N-1}]$, $\mathbf{c}^E = [c_0^E, c_1^E, \ldots, c^E_{\frac{N}{2}-1}]$, and $\mathbf{s}^E = [s_1^E, s_2^E, \ldots, s^E_{\frac{N}{2}}]$ be row vectors, and $\mathbf{x}^T$, $[\mathbf{c}^E]^T$ and $[\mathbf{s}^E]^T$ their corresponding column representations. Then

Eqs. (3.16) and (3.18) for the forward MDCT and MDST block transforms can be, respectively, written in the equivalent matrix-vector form as

$$[\mathbf{c}^{E}]^{T} = \boldsymbol{C}^{E}_{\frac{N}{2}\times N}\,\mathbf{x}^{T}, \qquad [\mathbf{s}^{E}]^{T} = \boldsymbol{S}^{E}_{\frac{N}{2}\times N}\,\mathbf{x}^{T}, \tag{3.24}$$

while Eqs. (3.17) and (3.19) for the backward MDCT and MDST block transforms without the normalization factor $\frac{1}{N}$ can be, respectively, written as

$$[\hat{\mathbf{x}}^{E-\mathrm{MDCT}}]^{T} = \epsilon_k\,\boldsymbol{C}^{E}_{N\times\frac{N}{2}}\,[\mathbf{c}^{E}]^{T}, \qquad [\hat{\mathbf{x}}^{E-\mathrm{MDST}}]^{T} = \tau_k\,\boldsymbol{S}^{E}_{N\times\frac{N}{2}}\,[\mathbf{s}^{E}]^{T}. \tag{3.25}$$

For the products of MDCT/MDST matrices and their transposed versions the following relations hold [15, 19]

$$\boldsymbol{C}^{E}_{\frac{N}{2}\times N}\,\epsilon_k\,[\boldsymbol{C}^{E}_{\frac{N}{2}\times N}]^{T} = \boldsymbol{S}^{E}_{\frac{N}{2}\times N}\,\tau_k\,[\boldsymbol{S}^{E}_{\frac{N}{2}\times N}]^{T} = N\,\boldsymbol{I}_{\frac{N}{2}}, \tag{3.26}$$

$$\epsilon_k\,[\boldsymbol{C}^{E}_{\frac{N}{2}\times N}]^{T}\,\boldsymbol{C}^{E}_{\frac{N}{2}\times N} = \frac{N}{2}\begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{J}_{\frac{N}{4}} & \mathbf{0} & \mathbf{0} \\ \boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{J}_{\frac{N}{4}} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} = \frac{N}{2}\begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} + \boldsymbol{J}_{\frac{N}{2}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_{\frac{N}{2}} + \boldsymbol{J}_{\frac{N}{2}} \end{pmatrix}, \tag{3.27}$$

$$\tau_k\,[\boldsymbol{S}^{E}_{\frac{N}{2}\times N}]^{T}\,\boldsymbol{S}^{E}_{\frac{N}{2}\times N} = \frac{N}{2}\begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & -\boldsymbol{J}_{\frac{N}{4}} & \mathbf{0} & \mathbf{0} \\ -\boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I}_{\frac{N}{4}} & -\boldsymbol{J}_{\frac{N}{4}} \\ \mathbf{0} & \mathbf{0} & -\boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} = \frac{N}{2}\begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} - \boldsymbol{J}_{\frac{N}{2}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_{\frac{N}{2}} - \boldsymbol{J}_{\frac{N}{2}} \end{pmatrix}, \tag{3.28}$$

where $\mathbf{0}$ is null matrix, $\boldsymbol{I}_{\frac{N}{2}}$ is the identity matrix, and $\boldsymbol{J}_{\frac{N}{2}}$ is the opposite diagonal identity matrix, all of order $\frac{N}{2}$. It is clear that the MDCT and MDST matrices in (3.26)–(3.28) have the same full rank (see Appendix A.1), i.e.,

$$\mathrm{rank}\,(\boldsymbol{C}^{E}_{\frac{N}{2}\times N}) = \mathrm{rank}\,(\boldsymbol{S}^{E}_{\frac{N}{2}\times N}) = \frac{N}{2}.$$

Let us denote their transposed matrices by $[\boldsymbol{C}^{E}]^{+}$ and $[\boldsymbol{S}^{E}]^{+}$. Using the relations (3.26)–(3.28) it can be verified that they satisfy Moore–Penrose conditions, and therefore, they are called to be generalized inverses or pseudoinverses of their corresponding matrices [15] (generalized inverse or pseudoinverse matrix is defined

in Appendix A.1). Hence, the matrices representing the forward and backward MDCT/MDST block transforms are actually pseudoinverse pairs.

The pseudoinverse matrix and its properties provide an elegant mathematical tool to characterize the MDCT/MDST block transforms in a matrix representation. There is the following interpretation of equations in (3.25). If we consider the forward MDCT/MDST in matrix representation given by (3.24) to be systems of linear equations, then the time domain aliased data sequences $\{\hat{\mathbf{x}}^{E-\mathrm{MDCT}}\}$ and $\{\hat{\mathbf{x}}^{E-\mathrm{MDST}}\}$ in (3.25) for given MDCT/MDST coefficients can be interpreted as least squares solutions, i.e., solutions with minimum norm [76–78]. Moreover, we can derive the time domain aliased data sequences for the backward MDCT and MDST block transforms explicitly in terms of the original data samples [15]. Indeed, substituting Eq. (3.24) into (3.25), i.e., performing the forward and backward MDCT/MDST block transforms and using relations (3.27) and (3.28) we have

$$[\hat{\mathbf{x}}^{E-\mathrm{MDCT}}]^T = \epsilon_k \, \boldsymbol{C}^E_{N\times\frac{N}{2}} \, [\mathbf{c}^{\mathbf{E}}]^T = \epsilon_k \, \boldsymbol{C}^E_{N\times\frac{N}{2}} \, \boldsymbol{C}^E_{\frac{N}{2}\times N} \, \mathbf{x}^T = \frac{N}{2} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} + \boldsymbol{J}_{\frac{N}{2}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_{\frac{N}{2}} + \boldsymbol{J}_{\frac{N}{2}} \end{pmatrix} \mathbf{x}^T, \tag{3.29}$$

$$[\hat{\mathbf{x}}^{E-\mathrm{MDST}}]^T = \tau_k \, \boldsymbol{S}^E_{N\times\frac{N}{2}} \, [\mathbf{s}^{E}]^T = \tau_k \, \boldsymbol{S}^E_{N\times\frac{N}{2}} \, \boldsymbol{S}^E_{\frac{N}{2}\times N} \, \mathbf{x}^T = \frac{N}{2} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} - \boldsymbol{J}_{\frac{N}{2}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_{\frac{N}{2}} - \boldsymbol{J}_{\frac{N}{2}} \end{pmatrix} \mathbf{x}^T. \tag{3.30}$$

From Eqs. (3.29) and (3.30) it follows that the time domain aliased data samples $\{\hat{x}_n^{E-\mathrm{MDCT}}\}$ can be derived explicitly in terms of the original data samples $\{x_n\}$ as

$$\begin{aligned} \hat{x}_n^{E-\mathrm{MDCT}} &= x_n + x_{\frac{N}{2}-1-n}, \qquad \hat{x}_{\frac{N}{2}-1-n}^{E-\mathrm{MDCT}} = \hat{x}_n^{E-\mathrm{MDCT}}, \\ \hat{x}_{\frac{N}{2}+n}^{E-\mathrm{MDCT}} &= x_{\frac{N}{2}+n} + x_{N-1-n}, \quad \hat{x}_{N-1-n}^{E-\mathrm{MDCT}} = \hat{x}_{\frac{N}{2}+n}^{E-\mathrm{MDCT}}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1, \end{aligned} \tag{3.31}$$

while the time domain aliased data sequence $\{\hat{x}_n^{E-\mathrm{MDST}}\}$ can be derived explicitly as

$$\begin{aligned} \hat{x}_n^{E-\mathrm{MDST}} &= x_n - x_{\frac{N}{2}-1-n}, \qquad \hat{x}_{\frac{N}{2}-1-n}^{E-\mathrm{MDST}} = -\hat{x}_n^{E-\mathrm{MDST}}, \\ \hat{x}_{\frac{N}{2}+n}^{E-\mathrm{MDST}} &= x_{\frac{N}{2}+n} - x_{N-1-n}, \quad \hat{x}_{N-1-n}^{E-\mathrm{MDST}} = -\hat{x}_{\frac{N}{2}+n}^{E-\mathrm{MDST}}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{3.32}$$

In Eqs. (3.31) and (3.32) we can clearly observe the recovered time domain aliased data sequences including their forms both in terms of the original data samples and their symmetry properties given by (3.14) and (3.15). With Eqs. (3.31) and (3.32)

in mind we can compute exactly the results of applying the forward and backward MDCT and MDST block transforms.

Finally, consider the relations (3.20) and (3.21) between the evenly stacked MDCT and MDST block transforms. Then, they can be represented in the matrix products, respectively, as

$$\boldsymbol{S}^{E}_{\frac{N}{2}\times N} = (-1)^{\frac{N}{4}}\, \boldsymbol{J}_{\frac{N}{2}}\, \boldsymbol{C}^{E}_{\frac{N}{2}\times N}\, \boldsymbol{D}_N, \qquad \boldsymbol{C}^{E}_{\frac{N}{2}\times N} = (-1)^{\frac{N}{4}}\, \boldsymbol{J}_{\frac{N}{2}}\, \boldsymbol{S}^{E}_{\frac{N}{2}\times N}\, \boldsymbol{D}_N, \tag{3.33}$$

where $\boldsymbol{J}_N$ is the opposite diagonal identity matrix, and $\boldsymbol{D}_N = \text{diag}\,\{1, -1, 1, \ldots, -1\}$ is the diagonal odd sign changing diagonal matrix, both of order N.

3.2.2 Oddly Stacked MDCT Filter Bank

The development of oddly stacked system follows closely that of the evenly stacked system. The oddly stacked, critically sampled SSB analysis filter bank based on TDAC has been originally derived in the form of a block transform operation as [23, 24]

$$X_k^{(m)} = \cos(\pi\, mk) \sum_{r=0}^{N-1} x^{(m)}_{m\frac{N}{2}+r}\, h_{N-1-r}\, \cos\left[\frac{2\pi}{N}\left(k+\frac{1}{2}\right)(r+n_0)\right],$$
$$k = 0, 1, \ldots, N-1, \tag{3.34}$$

where $\{X_k^{(m)}\}$ are channel signals, and $\{x_k^{(m)}\}$ are samples of the mth data block. $\{h_{N-1-r}\}$ is a time reversed low pass prototype filter of the length N, or equivalently, the analysis windowing function. The cosine modulation function $\cos\left[\frac{2\pi}{N}(k+\frac{1}{2})(r+n_0)\right]$ under the sum of (3.34) is referred to as the oddly stacked MDCT kernel. From (3.34) it can be seen that the channel signals are defined by the MDCT of windowed sequence and modified by $\cos(\pi\, mk)$. Compared to the evenly stacked system requiring the alternate applications of the MDCT and corresponding MDST, the oddly stacked system requires application of the MDCT only.

The oddly stacked, critically sampled SSB synthesis filter bank has been originally derived as a block transform operation in the following form [24]

$$\hat{x}_r^{(m)} = \frac{2}{N} \sum_{k=0}^{N-1} \cos(\pi\, mk)\, X_k^{(m)}\, \cos\left[\frac{2\pi}{N}\left(k+\frac{1}{2}\right)(r+n_0)\right], \quad r = 0, 1, \ldots, N-1, \tag{3.35}$$

where $\{\hat{x}_r^{(m)}\}$ are time domain aliased samples of the mth data block. From (3.35) and with respect to (3.5), the data sequence $\{\hat{x}_r^{(m)}\}$ is actually the windowed backward MDCT of the modified channel signals. The modifications by $\cos(\pi\, mk)$ in (3.34) and (3.35) can usually be ignored since they result in odd channels being inverted in frequency.

Similarly, after applying the analysis MDCT filter bank followed by the synthesis MDCT filter bank the time domain aliasing distortion is introduced. However, if the perfect reconstruction conditions hold (see Sect. 3.2.4), then for the phase factor $n_0 = \frac{N}{4} + \frac{1}{2}$, the time domain aliasing is canceled, and the original data sequence $\{x_r^{(m)}\}$ is recovered from two adjacent time domain aliased data sequences $\{\hat{x}_r^{(m)}\}$ and $\{\hat{x}_r^{(m+1)}\}$ by the windowing and overlap/add procedure defined by (3.5). More technical details about the oddly stacked system can be found in [24].

Similarly as for the evenly stacked system, the windowing procedure and perfect reconstruction conditions imposed on the analysis and synthesis windowing functions $\{h_r\}$ and $\{g_r\}$ are discussed in detail in Sect. 3.2.4, while the design of windowing functions is discussed in Sect. 3.2.5.

Note 1: Based of the SSB DFT filter bank [75], the oddly stacked, critically sampled SSB analysis and synthesis filter banks can be derived, respectively, in alternative forms as [22]

$$X_k^{(m)} = (-1)^{mk}\, 2\, \Re e \left\{ \sum_{r=0}^{N-1} x_{m\frac{N}{2}+r}^{(m)}\, h_{N-1-r}\, e^{-i\,\frac{2\pi}{N}(k+\frac{1}{2})(r+n_0)} \right\}, \quad i = \sqrt{-1},$$
$$k = 0, 1, \ldots, N-1, \tag{3.36}$$

and

$$\hat{x}_r^{(m)} = g_r\, \frac{1}{N}\, \Re e \left\{ \sum_{k=0}^{N-1} (-1)^{mk}\, X_k^{(m)}\, e^{i\,\frac{2\pi}{N}(k+\frac{1}{2})(r+n_0)} \right\}, \quad r = 0, 1, \ldots, N-1. \tag{3.37}$$

Rearranging Eq. (3.36) and omitting factor $(-1)^{mk}$ compensated in the synthesis filter bank we get [22]

$$X_k^{(m)} = 2\, \Re e \left\{ e^{-i\,\frac{2\pi}{N}(k+\frac{1}{2})(n_0-\frac{1}{2})} \sum_{r=0}^{N-1} x_{m\frac{N}{2}+r}^{(m)}\, h_{N-1-r}\, e^{-i\,\frac{2\pi}{N}(k+\frac{1}{2})(r+\frac{1}{2})} \right\},$$
$$k = 0, 1, \ldots, N-1. \tag{3.38}$$

The transform kernel under the sum on the right-hand side of (3.38) is recognized as the odd-time odd-frequency DFT (O^2DFT). The definition of O^2DFT and its properties are presented in Appendix B.1.

3.2.2.1 MDCT and MDST Block Transforms

Since the oddly stacked MDCT is preferred in audio coding applications, the forward and backward MDCT block transforms are defined and their properties

in the frequency and time domains are investigated in more detail. Additionally, the corresponding forward and backward MDST block transforms are defined and similarly, their properties are investigated. Without loss of generality the superscript m denoting the data block index is omitted.

Let $\{x_n\},\ n = 0, 1, \ldots, N-1$ represent a windowed input data sequence before its transformation, and N is assumed to be an even integer. Based on (3.34) and (3.35), the forward and backward MDCT block transforms are, respectively, defined as [24]

$$c_k^o = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \quad k = 0, 1, \ldots, N-1, \tag{3.39}$$

and

$$\hat{x}_n^{O-\text{MDCT}} = \frac{4}{N}\sum_{k=0}^{N-1} c_k^o \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \quad n = 0, 1, \ldots, N-1. \tag{3.40}$$

The corresponding forward and backward MDST block transforms are, respectively, defined as [21]

$$s_k^o = \sum_{n=0}^{N-1} x_n \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \quad k = 0, 1, \ldots, N-1, \tag{3.41}$$

and

$$\hat{x}_n^{O-\text{MDST}} = \frac{4}{N}\sum_{k=0}^{\frac{N}{2}-1} s_k^o \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \quad n = 0, 1, \ldots, N-1, \tag{3.42}$$

where $\{c_k^o\}/\{s_k^o\}$ are MDCT/MDST coefficients, and $\{\hat{x}_n^{O-\text{MDCT}}\}/\{\hat{x}_n^{O-\text{MDST}}\}$ are time domain aliased data sequences recovered by the backward MDCT/MDST. The data sequences $\{\hat{x}_n^{O-\text{MDCT}}\}$ in (3.40) and $\{\hat{x}_n^{O-\text{MDST}}\}$ in (3.42) recovered by appropriate backward transforms do not correspond to the original data sequence. In the context of TDAC analysis/synthesis filter banks the distorted sequences $\{\hat{x}_n^{O-\text{MDCT}}\}$ and $\{\hat{x}_n^{O-\text{MDST}}\}$ are said to be time domain aliased.

3.2.2.2 Symmetry Properties of the MDCT and MDST Block Transforms

In order to investigate the symmetry properties of oddly stacked MDCT and MDST block transforms, let us substitute $N-k-1$ for k into (3.39), and we get

$$\overset{o}{c}_{N-k-1} = \sum_{n=0}^{N-1} x_n \cos\left[\pi\left(2n+1+\frac{N}{2}\right)\right] \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$
$$k = 0, 1, \ldots, N-1. \tag{3.43}$$

The expression $(2n+1+\frac{N}{2})$ in $\cos\left[\pi(2n+1+\frac{N}{2})\right]$ under the sum of (3.43) may be odd or even depending on a value of $\frac{N}{2}$ being even or odd, respectively. Thus, we have

$$\overset{o}{c}_{N-k-1} = \begin{cases} -\overset{o}{c}_k, & \text{if } (2n+1+\frac{N}{2}) \text{ is odd}, \quad k = 0, 1, \ldots, \frac{N}{2}-1, \\ \overset{o}{c}_k, & \text{if } (2n+1+\frac{N}{2}) \text{ is even}, \quad k = 0, 1, \ldots, \frac{N}{2}-1, \end{cases} \tag{3.44}$$

implying that the MDCT coefficients have either the even anti-symmetry property when $\frac{N}{2}$ is even, or the even symmetry property when $\frac{N}{2}$ is odd. The even symmetric and anti-symmetric sequences are defined in Appendix G.1.

On the other hand, substituting $N-k-1$ for k into (3.41), we get

$$\overset{o}{s}_{N-k-1} = -\sum_{n=0}^{N-1} x_n \cos\left[\pi\left(2n+1+\frac{N}{2}\right)\right] \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$
$$k = 0, 1, \ldots, N-1. \tag{3.45}$$

Similarly, the expression $(2n+1+\frac{N}{2})$ in $\cos\left[\pi(2n+1+\frac{N}{2})\right]$ under the sum of (3.45) may be odd or even depending on a value of $\frac{N}{2}$ being even or odd, respectively. Thus, we have

$$\overset{o}{s}_{N-k-1} = \begin{cases} \overset{o}{s}_k, & \text{if } (2n+1+\frac{N}{2}) \text{ is odd}, \quad k = 0, 1, \ldots, \frac{N}{2}-1, \\ -\overset{o}{s}_k, & \text{if } (2n+1+\frac{N}{2}) \text{ is even}, \quad k = 0, 1, \ldots, \frac{N}{2}-1, \end{cases} \tag{3.46}$$

implying that the MDST coefficients have either the even symmetry property when $\frac{N}{2}$ is even, or the even anti-symmetry property when $\frac{N}{2}$ is odd. From (3.44) and (3.46) it follows that only $\frac{N}{2}$ MDCT/MDST coefficients are unique.

The time domain aliased data sequences $\{\hat{x}_n^{O-\text{MDCT}}\}$ and $\{\hat{x}_n^{O-\text{MDST}}\}$ recovered by the backward MDCT/MDST have the following local symmetries

$$\hat{x}_n^{O-\text{MDCT}} = -\hat{x}_{\frac{N}{2}-1-n}^{O-\text{MDCT}}, \qquad \hat{x}_{\frac{N}{2}+n}^{O-\text{MDCT}} = \hat{x}_{N-1-n}^{O-\text{MDCT}}, \tag{3.47}$$

$$\hat{x}_n^{O-\text{MDST}} = \hat{x}_{\frac{N}{2}-1-n}^{O-\text{MDST}}, \qquad \hat{x}_{\frac{N}{2}+n}^{O-\text{MDST}} = -\hat{x}_{N-1-n}^{O-\text{MDST}}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{3.48}$$

The symmetry properties (3.47) and (3.48) can simply be verified by proper substitution into Eqs. (3.40) and (3.42). It can be seen that $\{\hat{x}_n^{O-\text{MDCT}}\}$ exhibits odd/even symmetries in the first/second half, whereas $\{\hat{x}_n^{O-\text{MDST}}\}$ exhibits even/odd symmetries in the first/second half. From an algorithmic point of view this means that it is sufficient to compute only the time domain aliased samples $\{\hat{x}_n^{O-\text{MDCT}}\}$, $\{\hat{x}_{\frac{N}{2}+n}^{O-\text{MDST}}\}$ and $\{\hat{x}_n^{O-\text{MDST}}\}$, $\{\hat{x}_{\frac{N}{2}+n}^{O-\text{MDST}}\}$ for $n = 0, 1, \ldots, \frac{N}{4} - 1$, by the backward MDCT/MDST. Note that N must be integer divisible by 4.

Exploiting the symmetry properties (3.44) and (3.46), the oddly stacked forward and backward MDCT block transforms are, respectively, expressed as [20]

$$c_k^o = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)(2k+1)\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{3.49}$$

and

$$\hat{x}_n^{O-\text{MDCT}} = \frac{4}{N} \sum_{k=0}^{\frac{N}{2}-1} c_k^o \cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)(2k+1)\right], \quad n = 0, 1, \ldots, N-1, \tag{3.50}$$

while the oddly stacked forward and backward MDST block transforms are, respectively, expressed as [20]

$$s_k^o = \sum_{n=0}^{N-1} x_n \sin\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)(2k+1)\right], \quad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{3.51}$$

and

$$\hat{x}_n^{O-\text{MDST}} = \frac{4}{N} \sum_{k=0}^{\frac{N}{2}-1} s_k^o \cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)(2k+1)\right], \quad n = 0, 1, \ldots, N-1. \tag{3.52}$$

It is noted that the normalization factor $\frac{4}{N}$ in the backward MDCT and MDST defined by (3.50) and (3.52), respectively, is very frequently uniformly distributed between the forward and backward MDCT/MDST block transforms in the form of $\sqrt{\frac{4}{N}}$.

3.2.2.3 Periodicity Properties of the MDCT/MDST Transform Kernels

Periodic and anti-periodic data sequences are defined in Appendix G.2. Denoting the elements of oddly stacked MDCT and MDST kernels in (3.49) and (3.51), respectively, as [20]

$$t_{k,n}^{(c)} = \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$
$$t_{k,n}^{(s)} = \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \tag{3.53}$$

and substituting $n+N$, and then $n+2N$ for n into (3.53) we obtain

$$t_{k,n}^{(c)} = -\,t_{k,n+N}^{(c)}, \qquad t_{k,n}^{(c)} = t_{k,n+2N}^{(c)},$$
$$t_{k,n}^{(s)} = -\,t_{k,n+N}^{(s)}, \qquad t_{k,n}^{(s)} = t_{k,n+2N}^{(s)}, \qquad \forall\ k. \tag{3.54}$$

Equation (3.54) implies that the MDCT and MDST transform kernels are anti-periodic sequences with period N, and periodic sequences with period $2N$.

3.2.2.4 Symmetry Properties of the MDCT/MDST Basis Vectors

For a given N, consider the MDCT and MDST transform kernels given by (3.53). One can observe that the MDCT and MDST basis vectors exhibit the following local symmetries [20]:

$$t_{k,\frac{N}{2}-1-n}^{(c)} = -\,t_{k,n}^{(c)}, \qquad t_{k,\frac{N}{2}+n}^{(c)} = t_{k,N-1-n}^{(c)},$$
$$t_{k,\frac{N}{2}-1-n}^{(s)} = t_{k,n}^{(s)}, \qquad t_{k,\frac{N}{2}+n}^{(s)} = -\,t_{k,N-1-n}^{(s)}, \qquad \forall\ k, \qquad n = 0, 1, \ldots, \frac{N}{4}-1. \tag{3.55}$$

The symmetry properties of the MDCT and MDST basis vectors (3.55) can be simply verified by proper substitution into Eq. (3.53). Note that they are quite similar to those of the time domain aliased data sequences $\{\hat{x}_n^{O-\mathrm{MDCT}}\}$ and $\{\hat{x}_n^{O-\mathrm{MDST}}\}$ given by (3.47) and (3.48), respectively.

3.2.2.5 Special Properties of the MDCT/MDST Block Transforms

Studies of the MDCT block transform and its implications to audio coding and error concealment have been presented in [27–29]. In these, characteristics of the MDCT of a single data block both in time and frequency domains are analyzed (symmetry and non-orthogonal properties, energy-compaction capability, and the

concept of TDAC), and their impact on audio coding performance are discussed. In particular, based on Fourier frequency analysis the authors formulate conditions in which MDCT coefficients become zero even with nonzero time domain samples in the single data block. Also, conditions are derived in which the original time domain samples can be perfectly reconstructed by performing forward and backward MDCT block transforms without even applying an overlap/add procedure. These conditions constitute special or the so-called peculiar properties of the MDCT block transform.

Claim 1 If the input data sequence $\{x_n\}$ exhibits a local symmetry such that

$$x_n = x_{\frac{N}{2}-1-n}, \qquad x_{\frac{N}{2}+n} = -x_{N-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{3.56}$$

then the MDCT coefficients are degenerated to zero, i.e., $c_k^{o} = 0$ for $k = 0, 1, \ldots, \frac{N}{2} - 1$.

In order to prove this property, let us split the forward N-point MDCT block transform given by (3.49) into two sums as

$$\begin{aligned} c_k^{o} &= \sum_{n=0}^{\frac{N}{2}-1} x_n \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right] \\ &\quad + \sum_{n=\frac{N}{2}}^{N-1} x_n \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right] \\ &= \sum_{n=0}^{\frac{N}{2}-1} x_n \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right] \\ &\quad + \sum_{n=0}^{\frac{N}{2}-1} x_{\frac{N}{2}+n} \cos\left[\frac{\pi}{2N}\left(2n+1+3\frac{N}{2}\right)(2k+1)\right], \end{aligned}$$

and substituting $\frac{N}{2} - 1 - n$ for n into both sums we get

$$\begin{aligned} c_k^{o} &= \sum_{n=0}^{\frac{N}{4}-1} (x_n - x_{\frac{N}{2}-1-n}) \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right] \\ &\quad + \sum_{n=0}^{\frac{N}{4}-1} (x_{\frac{N}{2}+n} + x_{N-1-n}) \cos\left[\frac{\pi}{2N}\left(2n+1+3\frac{N}{2}\right)(2k+1)\right] \\ &= \sum_{n=0}^{\frac{N}{4}-1} (x_n - x_{\frac{N}{2}-1-n}) \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right] \\ &\quad + \sum_{n=0}^{\frac{N}{4}-1} -(x_{\frac{N}{2}+n} + x_{N-1-n}) \cos\left[\frac{\pi}{2N}\left(2n+1-\frac{N}{2}\right)(2k+1)\right]. \end{aligned} \tag{3.57}$$

Now, the special property stated by **Claim 1** immediately follows from Eq. (3.57). The property also illustrates that the MDCT block transform does not fulfill Parseval's theorem [29]. Note that the N-point MDCT block transform has been decomposed into two $\frac{N}{4}$-point MDCT block transforms.

Claim 2 If the input data sequence $\{x_n\}$ exhibits a local symmetry such that

$$x_n = -x_{\frac{N}{2}-1-n}, \qquad x_{\frac{N}{2}+n} = x_{N-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{3.58}$$

then by the forward and backward MDCT block transforms, the original input data sequence will be perfectly reconstructed without the need for an overlap/add procedure, i.e., $\hat{x}_n^{O-\text{MDCT}} = x_n$ for $n = 0, 1, \ldots, N-1$.

This property immediately follows from (3.47) and (3.57). The properties stated by **Claims 1** and **2** are very special theoretical cases which rarely occur in real audio coding applications, especially after the proper windowing operation. Nevertheless, in both cases the time domain aliased samples can still be perfectly reconstructed by the overlap/add procedure. However, based on **Claim 1** if the input signal is close to (3.56) then the MDCT spectrum will show an unwanted and unrecoverable behavior. For completeness, additional properties are presented in the following text as partial cases of **Claim 2**.

Claim 3 If the input data sequence $\{x_n\}$ exhibits a local symmetry such that

$$x_n = x_{\frac{N}{2}-1-n}, \qquad x_{\frac{N}{2}+n} = x_{N-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{3.59}$$

then after applying the forward and backward MDCT block transforms, the first half of the recovered data sequence will be equal to zero, i.e., $\hat{x}_n^{O-\text{MDCT}} = 0$, $n = 0, 1, \ldots, \frac{N}{2} - 1$, and the second half of the original data sequence $\{x_{\frac{N}{2}+n}\}$ will be perfectly reconstructed without an overlap/add procedure, i.e., $x_{\frac{N}{2}+n} = \hat{x}_{\frac{N}{2}+n}^{O-\text{MDCT}}$, $n = 0, 1, \ldots, \frac{N}{2} - 1$.

Claim 4 If the input data sequence $\{x_n\}$ exhibits a local symmetry such that

$$x_n = -x_{\frac{N}{2}-1-n}, \qquad x_{\frac{N}{2}+n} = -x_{N-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{3.60}$$

then after applying the forward and backward MDCT block transforms, the first half of the original input data sequence $\{x_n\}$ will be perfectly reconstructed without the need for an overlap/add procedure, i.e., $x_n = \hat{x}_n^{O-\text{MDCT}}$, $n = 0, 1, \ldots, \frac{N}{2} - 1$, and the second half of the recovered data sequence will be equal to zero, i.e., $\hat{x}_{\frac{N}{2}+n}^{O-\text{MDCT}} = 0$, $n = 0, 1, \ldots, \frac{N}{2} - 1$.

Properties stated by **Claims 3** and **4** also follow from (3.47) and (3.57).

Note 2: The special properties of the MDST can be derived in a similar way.

3.2.2.6 Relation Between the MDCT and MDST Block Transforms

In order to obtain the relation between the oddly stacked MDCT and MDST block transforms, let us substitute $\frac{N}{2}-k-1$ for k in (3.49) and (3.51), and we, respectively, get [20, 21]

$$c^{o}_{\frac{N}{2}-k-1} = (-1)^{\frac{N}{4}} \sum_{n=0}^{N-1}(-1)^n x_n \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2}-1. \tag{3.61}$$

and

$$s^{o}_{\frac{N}{2}-k-1} = (-1)^{\frac{N}{4}} \sum_{n=0}^{N-1}(-1)^n x_n \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2}-1, \tag{3.62}$$

Equations (3.61) [and (3.62)] imply that the MDST coefficients $\{s^{o}_{k}\}$ (MDCT coefficients $\{c^{o}_{k}\}$) can be simply obtained from the MDCT (MDST) ones only by sign changes applied to odd-indexed samples in the original data sequence, and after the MDCT (MDST) computation, the MDST (MDCT) coefficients are in reverse order.

The relations between the MDCT and MDST block transforms result in a simple method to compute MDST (MDCT) using only one fast algorithm for the MDCT (MDST) computation.

3.2.2.7 Relation Between the MDCT/MDST and O^2DFT

Based on (3.38), the oddly stacked MDCT and MDST block transforms can be expressed in terms of the O^2DFT as follows [22, 26]:

$$\begin{aligned} X_k &= e^{-i\frac{\pi}{4}(2k+1)} \sum_{n=0}^{N-1} x_n\, e^{-i\frac{2\pi}{4N}(2k+1)(2n+1)} \\ &= \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right] \\ &\quad -i\sum_{n=0}^{N-1} x_n \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \\ k &= 0, 1, \ldots, N-1. \end{aligned} \tag{3.63}$$

From (3.63) one can see that the MDCT is the real component of O^2DFT while the MDST is the imaginary component of O^2DFT. The definition of O^2DFT and its properties are presented in Appendix B.1, and its fast algorithm in Appendix B.2. It is noted that the arguments of MDCT/MDST kernels can alternatively be written as

$$\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)=\frac{\pi}{2N}(2n+1)(2k+1)+\frac{\pi}{4}(2k+1). \tag{3.64}$$

3.2.2.8 Relation Between the MDCT/MDST and DFT

To derive convolution (linear filtering) algorithms in the MDCT/MDST domain, the oddly stacked MDCT and MDST block transforms have been expressed in terms of the DFT as follows [25, 26].

Consider an N-point windowed data sequence $\{x_n\}$. Then, its unnormalized N-point DFT and the corresponding real part are, respectively, given by

$$f_k^N=\sum_{n=0}^{N-1}x_n\, e^{-i\,\frac{2\pi nk}{N}},\quad \Re e\left\{f_k^N\right\}=\sum_{n=0}^{N-1}x_n\,\cos\left[\frac{2\pi nk}{N}\right],\quad k=0,1,\ldots,N-1. \tag{3.65}$$

If in the real part of (3.65) we replace n with $2n+1+\frac{N}{2}$, k with $2k+1$ and N with $2N$, we obtain the expression for MDCT. Replacing n with $2n+1+\frac{N}{2}$ in x_n is equivalent to up-sampling $\{x_n\}$ by a factor 2 followed by shifting $\frac{N}{2}$ samples. Similarly, replacing k with $2k+1$ is equivalent to subsampling by a factor 2 in the frequency domain. Changing N to $2N$ results in $4N$-point DFT instead of N-point DFT. It means that the MDCT of $\{x_n\}$ is equal to the real part of the subsampled $4N$-point DFT of $\{y_n\}$, $n=0,1,\ldots,4N-1$, where $\{y_n\}$ is obtained from $\{x_n\}$ by up-sampling and shifting [25], i.e.,

$$y_{2n+1+\frac{N}{2}}=\begin{cases}x_n, & n=0,1,\ldots,N-1,\\ 0, & \text{otherwise.}\end{cases} \tag{3.66}$$

Then, the MDCT coefficients are expressed in terms of the DFT ones as [26]

$$c_k^o=\Re e\left\{f_{2k+1}^{4N}\right\},\qquad k=0,1,\ldots,\frac{N}{2}-1, \tag{3.67}$$

where $\{f_{2k+1}^{4N}\}$ is the $4N$-point DFT of $\{y_n\}$ given by (3.66). The first $2N$ values of the $4N$-point DFT provide the full spectrum of the N-point signal. By the symmetry of the DFT, the first $2N$ values are equal to the second $2N$ values, but sign changed [25].

Following the similar steps, the MDST coefficients are expressed in terms of the DFT ones as [26]

$$s_k^{O} = -\,\Im m\left\{f_{2k+1}^{4N}\right\}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{3.68}$$

Note 3: A method for the DFT spectrum estimation from the MDCT and MDST spectra has been developed in [31]. Specifically, based on matrix analysis, a sparse matrix representation for the conversion to and from the DFT has been derived.

3.2.2.9 Matrix Representations of the MDCT and MDST Block Transforms

Consider the oddly stacked MDCT and MDST block transforms defined by (3.49) and (3.51), respectively. The symmetry properties (3.44) and (3.46) imply that only $\frac{N}{2}$ rows of MDCT and MDST matrices are linear independent. Therefore, let $\boldsymbol{C}^{O}_{\frac{N}{2}\times N}$ and $\boldsymbol{S}^{O}_{\frac{N}{2}\times N}$, be the $\frac{N}{2} \times N$ MDCT and MDST matrices, respectively. Then $[\boldsymbol{C}^{O}_{\frac{N}{2}\times N}]^T = \boldsymbol{C}^{O}_{N\times\frac{N}{2}}$ and $[\boldsymbol{S}^{O}_{\frac{N}{2}\times N}]^T = \boldsymbol{S}^{O}_{N\times\frac{N}{2}}$, where T denotes transposition. Next, let $\mathbf{x} = [x_0, x_1, \ldots, x_{N-1}]$, $\mathbf{c}^{O} = [c_0^{O}, c_1^{O}, \ldots, c_{\frac{N}{2}-1}^{O}]$, and $\mathbf{s}^{O} = [s_0^{O}, s_1^{O}, \ldots, s_{\frac{N}{2}-1}^{O}]$ be row vectors, and $\mathbf{x}^T$, $[\mathbf{c}^{O}]^T$, and $[\mathbf{s}^{O}]^T$ their corresponding column representations. Then (3.49) and (3.51) for the forward MDCT and MDST block transforms can be, respectively, written in the equivalent matrix-vector form as

$$[\mathbf{c}^{O}]^T = \boldsymbol{C}^{O}_{\frac{N}{2}\times N}\,\mathbf{x}^T, \qquad [\mathbf{s}^{O}]^T = \boldsymbol{S}^{O}_{\frac{N}{2}\times N}\,\mathbf{x}^T, \tag{3.69}$$

while (3.50) and (3.52) for the backward MDCT and MDST block transforms without the normalization factor $\frac{4}{N}$ can be, respectively, written as

$$[\hat{\mathbf{x}}^{O-\mathrm{MDCT}}]^T = \boldsymbol{C}^{O}_{N\times\frac{N}{2}}\,[\mathbf{c}^{O}]^T, \qquad [\hat{\mathbf{x}}^{O-\mathrm{MDST}}]^T = \boldsymbol{S}^{O}_{N\times\frac{N}{2}}\,[\mathbf{s}^{O}]^T. \tag{3.70}$$

For the products of MDCT/MDST matrices and their transposed versions, the following relations hold [15, 19]

$$\boldsymbol{C}^{O}_{\frac{N}{2}\times N}\,[\boldsymbol{C}^{O}_{\frac{N}{2}\times N}]^T = \boldsymbol{S}^{O}_{\frac{N}{2}\times N}\,[\boldsymbol{S}^{O}_{\frac{N}{2}\times N}]^T = \frac{N}{2}\,\boldsymbol{I}_{\frac{N}{2}}, \tag{3.71}$$

$$[\boldsymbol{C}^{O}_{\frac{N}{2}\times N}]^T\,\boldsymbol{C}^{O}_{\frac{N}{2}\times N} = \frac{N}{4}\begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & -\boldsymbol{J}_{\frac{N}{4}} & \mathbf{0} & \mathbf{0} \\ -\boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{J}_{\frac{N}{4}} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} = \frac{N}{4}\begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} - \boldsymbol{J}_{\frac{N}{2}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_{\frac{N}{2}} + \boldsymbol{J}_{\frac{N}{2}} \end{pmatrix} \tag{3.72}$$

$$[\boldsymbol{S}^{O}_{\frac{N}{2}\times N}]^T\,\boldsymbol{S}^{O}_{\frac{N}{2}\times N} = \frac{N}{4}\begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{\frac{N}{4}} & -\boldsymbol{J}_{\frac{N}{4}} \\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} = \frac{N}{4}\begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} + \boldsymbol{J}_{\frac{N}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{\frac{N}{2}} - \boldsymbol{J}_{\frac{N}{2}} \end{pmatrix} \tag{3.73}$$

where $\boldsymbol{0}$ is null matrix, $\boldsymbol{I}_{\frac{N}{2}}$ is the identity matrix, and $\boldsymbol{J}_{\frac{N}{2}}$ is the opposite diagonal identity matrix, all of order $\frac{N}{2}$. Now it is clear that the MDCT and MDST matrices in (3.71)–(3.73) have the same full rank (see Appendix A.1), i.e.,

$$\text{rank}\,(\boldsymbol{C}^{O}_{\frac{N}{2}\times N}) = \text{rank}\,(\boldsymbol{S}^{O}_{\frac{N}{2}\times N}) = \frac{N}{2}.$$

Let us denote their transposed matrices by $[\boldsymbol{C}^{O}]^{+}$ and $[\boldsymbol{S}^{O}]^{+}$. Using the relations (3.71)–(3.73) it can be verified that they satisfy Moore–Penrose conditions, and therefore, they are called to be generalized inverses or pseudoinverses of their corresponding matrices [15] (generalized inverse or pseudoinverse matrix is defined in Appendix A.1). Hence, the matrices representing the forward and backward MDCT/MDST block transforms are actually pseudoinverse pairs.

Similarly as for the evenly stacked system, we can derive the time domain aliased data sequences for the oddly stacked backward MDCT and MDST block transforms explicitly in terms of the original data samples [15]. Indeed, substituting Eq. (3.69) into (3.70), i.e., performing the forward and backward MDCT/MDST block transforms and using relations (3.72) and (3.73) we have

$$[\hat{\mathbf{x}}^{O-\text{MDCT}}]^T = \boldsymbol{C}^{O}_{N\times\frac{N}{2}}\,[\mathbf{c}^{O}]^T = \boldsymbol{C}^{O}_{N\times\frac{N}{2}}\,\boldsymbol{C}^{O}_{\frac{N}{2}\times N}\,\mathbf{x}^T = \frac{N}{4}\begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} - \boldsymbol{J}_{\frac{N}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{\frac{N}{2}} + \boldsymbol{J}_{\frac{N}{2}} \end{pmatrix}\mathbf{x}^T, \tag{3.74}$$

$$[\hat{\mathbf{x}}^{O-\text{MDST}}]^T = \boldsymbol{S}^{O}_{N\times\frac{N}{2}}\,[\mathbf{s}^{O}]^T = \boldsymbol{S}^{O}_{N\times\frac{N}{2}}\,\boldsymbol{S}^{O}_{\frac{N}{2}\times N}\,\mathbf{x}^T = \frac{N}{4}\begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} + \boldsymbol{J}_{\frac{N}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{\frac{N}{2}} - \boldsymbol{J}_{\frac{N}{2}} \end{pmatrix}\mathbf{x}^T. \tag{3.75}$$

From Eqs. (3.74) and (3.75) it follows that the time domain aliased data samples $\{\hat{x}_n^{O-\text{MDCT}}\}$ can be derived explicitly in terms of the original data samples $\{x_n\}$ as

$$\begin{aligned} &\hat{x}_n^{O-\text{MDCT}} = x_n - x_{\frac{N}{2}-1-n}, \qquad \hat{x}_{\frac{N}{2}-1-n}^{O-\text{MDCT}} = -\,\hat{x}_n^{O-\text{MDCT}}, \\ &\hat{x}_{\frac{N}{2}+n}^{O-\text{MDCT}} = x_{\frac{N}{2}+n} + x_{N-1-n}, \quad \hat{x}_{N-1-n}^{O-\text{MDCT}} = \hat{x}_{\frac{N}{2}+n}^{O-\text{MDCT}}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1, \end{aligned} \tag{3.76}$$

while the time domain aliased data sequence $\{\hat{x}_n^{O-\text{MDST}}\}$ can be derived explicitly as

$$\begin{aligned}
&\hat{x}_n^{O-\text{MDST}} = x_n + x_{\frac{N}{2}-1-n}, \qquad \hat{x}_{\frac{N}{2}-1-n}^{O-\text{MDST}} = \hat{x}_n^{O-\text{MDST}}, \\
&\hat{x}_{\frac{N}{2}+n}^{O-\text{MDST}} = x_{\frac{N}{2}+n} - x_{N-1-n}, \quad \hat{x}_{N-1-n}^{O-\text{MDST}} = -\hat{x}_{\frac{N}{2}+n}^{O-\text{MDST}}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1.
\end{aligned} \tag{3.77}$$

In Eqs. (3.76) and (3.77) we can clearly observe the recovered time domain aliased data sequences including their forms both in terms of the original data samples and their symmetry properties given by (3.47) and (3.48). With Eqs. (3.76) and (3.77) in mind we can compute exactly the results of applying the forward and backward MDCT and MDST block transforms.

Finally, consider the relations (3.61) and (3.62) between the oddly stacked MDCT and MDST block transforms. Then, they can be represented in the matrix products, respectively, as

$$\boldsymbol{S}_{\frac{N}{2}\times N}^{O} = (-1)^{\frac{N}{4}} \boldsymbol{J}_{\frac{N}{2}} \boldsymbol{C}_{\frac{N}{2}\times N}^{O} \boldsymbol{D}_N, \qquad \boldsymbol{C}_{\frac{N}{2}\times N}^{O} = (-1)^{\frac{N}{4}} \boldsymbol{J}_{\frac{N}{2}} \boldsymbol{S}_{\frac{N}{2}\times N}^{O} \boldsymbol{D}_N, \tag{3.78}$$

where $\boldsymbol{J}_N$ is the opposite diagonal identity matrix, and $\boldsymbol{D}_N = diag\,\{1, -1, 1, \ldots, -1\}$ is the diagonal odd sign changing diagonal matrix, both of order N.

3.2.2.10 Consequences of MDCT/MDST Matrix Representations

From a viewpoint of terminology used in the literature the matrix representations of the MDCT and MDST block transforms imply two very important facts:

1. Contrary to the discrete trigonometric transforms (various versions of DCT and DST transforms) [74] which are represented by square invertible orthogonal/orthonormal matrices of order N (therefore forward and inverse transform), the MDCT and MDST block transforms are represented by non-square $\frac{N}{2} \times N$ matrices for which matrix inversions are not defined (actually, there exist their pseudoinverses [15]). Consequently, the notion of "inverse MDCT/MDST," frequently used in the literature, is vague from a standpoint of matrix theory. The more comprehensive notion to be used is "backward MDCT/MDST" (preferred) or "reverse MDCT/MDST." The second argument supporting such a conclusion is that compared to discrete unitary transforms, the MDCT/MDST do not fulfill Parseval's theorem, i.e., its time domain energy is not equal to its frequency domain energy. There exist several published papers pointing out and discussing this topic [27–30].
2. Further, the discrete trigonometric transforms [74] for input data sequences of length N generate N unique frequency coefficients (therefore N-point forward and inverse transform). Hence, the notion of "N-point" is naturally associated with the length of a input data sequence. In the case of the

MDCT/MDST, the forward MDCT/MDST block transforms for input time domain sequences of lengths N generate $\frac{N}{2}$ unique frequency coefficients while the backward MDCT/MDST for $\frac{N}{2}$ input frequency domain coefficients generate N time domain aliased samples. Therefore, we would use the notions N-point forward MDCT/MDST and $\frac{N}{2}$-point backward MDCT/MDST block transforms.

3.2.2.11 Relations and Products Among MDCT/MDST Block Sub-matrices and Their Properties

Consider the matrices $\boldsymbol{C}^{o}_{\frac{N}{2}\times N}$ and $\boldsymbol{S}^{o}_{\frac{N}{2}\times N}$ in (3.69) representing the forward MDCT and MDST block transforms, respectively. Since $\boldsymbol{C}^{o}_{\frac{N}{2}\times N}$ and $\boldsymbol{S}^{o}_{\frac{N}{2}\times N}$ are non-square matrices, we know that their inverse matrices do not exist. However, together with their corresponding transposed versions provide pseudoinverse pairs (see Appendix A.1).

Let the matrices $\boldsymbol{C}^{o}_{\frac{N}{2}\times N}$ and $\boldsymbol{S}^{o}_{\frac{N}{2}\times N}$ be split into the following two square blocks sub-matrices:

$$\boldsymbol{C}^{o}_{\frac{N}{2}\times N} = \left(\boldsymbol{K}_{\frac{N}{2}}\ \boldsymbol{L}_{\frac{N}{2}}\right), \qquad \boldsymbol{S}^{o}_{\frac{N}{2}\times N} = \left(\boldsymbol{P}_{\frac{N}{2}}\ \boldsymbol{Q}_{\frac{N}{2}}\right), \tag{3.79}$$

where $\boldsymbol{K}_{\frac{N}{2}}$, $\boldsymbol{L}_{\frac{N}{2}}$ and $\boldsymbol{P}_{\frac{N}{2}}$, $\boldsymbol{Q}_{\frac{N}{2}}$ are square nonsingular sub-matrices of order $\frac{N}{2}$. Splitting the matrices $\boldsymbol{C}^{o}_{\frac{N}{2}\times N}$ and $\boldsymbol{S}^{o}_{\frac{N}{2}\times N}$ into two block square sub-matrices is very useful in a solution of many problems represented in the block matrix form, and plays a key role in the derivation of a relation between MDCT and MDST coefficients directly in the frequency domain (see Chap. 6). Based on the relation (3.78), the sub-matrix pairs $\boldsymbol{K}_{\frac{N}{2}}, \boldsymbol{P}_{\frac{N}{2}}$ and $\boldsymbol{L}_{\frac{N}{2}}, \boldsymbol{Q}_{\frac{N}{2}}$ are closely related as

$$\boldsymbol{P}_{\frac{N}{2}} = (-1)^{\frac{N}{4}}\boldsymbol{J}_{\frac{N}{2}}\boldsymbol{K}_{\frac{N}{2}}\boldsymbol{D}_{\frac{N}{2}}, \qquad \boldsymbol{Q}_{\frac{N}{2}} = (-1)^{\frac{N}{4}}\boldsymbol{J}_{\frac{N}{2}}\boldsymbol{L}_{\frac{N}{2}}\boldsymbol{D}_{\frac{N}{2}}, \tag{3.80}$$

where the elements of $\boldsymbol{K}_{\frac{N}{2}}$ and $\boldsymbol{L}_{\frac{N}{2}}$ are, respectively, given by

$$\begin{aligned}
\{\boldsymbol{K}_{\frac{N}{2}}\}_{k,n} &= \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],\\
\{\boldsymbol{L}_{\frac{N}{2}}\}_{k,n} &= \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{3N}{2}\right)(2k+1)\right],\\
k,\ n &= 0,1,\ldots,\frac{N}{2}-1.
\end{aligned} \tag{3.81}$$

According to (3.80) the elements of $\boldsymbol{P}_{\frac{N}{2}}$ and $\boldsymbol{Q}_{\frac{N}{2}}$ are, respectively, given by

$$\{\boldsymbol{P}_{\frac{N}{2}}\}_{\frac{N}{2}-1-k,n} = (-1)^{n+\frac{N}{4}} \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$
$$\{\boldsymbol{Q}_{\frac{N}{2}}\}_{\frac{N}{2}-1-k,n} = (-1)^{n+\frac{N}{4}} \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{3N}{2}\right)(2k+1)\right],$$
$$k,\ n = 0, 1, \ldots, \frac{N}{2}-1. \tag{3.82}$$

Then, the transposed matrices $[\boldsymbol{C}^{o}_{\frac{N}{2}\times N}]^T = \boldsymbol{C}_{N\times\frac{N}{2}}$ and $[\boldsymbol{S}^{o}_{\frac{N}{2}\times N}]^T = \boldsymbol{S}_{N\times\frac{N}{2}}$ in the block form are, respectively, given by

$$\boldsymbol{C}^{o}_{N\times\frac{N}{2}} = \begin{pmatrix} \boldsymbol{K}^T_{\frac{N}{2}} \\ \boldsymbol{L}^T_{\frac{N}{2}} \end{pmatrix}, \qquad \boldsymbol{S}^{o}_{N\times\frac{N}{2}} = \begin{pmatrix} \boldsymbol{P}^T_{\frac{N}{2}} \\ \boldsymbol{Q}^T_{\frac{N}{2}} \end{pmatrix}, \tag{3.83}$$

and the row vectors of $\boldsymbol{K}_{\frac{N}{2}}/\boldsymbol{L}_{\frac{N}{2}}$ and $\boldsymbol{P}_{\frac{N}{2}}/\boldsymbol{Q}_{\frac{N}{2}}$ become the column vectors of $\boldsymbol{K}^T_{\frac{N}{2}}/\boldsymbol{L}^T_{\frac{N}{2}}$ and $\boldsymbol{P}^T_{\frac{N}{2}}/\boldsymbol{Q}^T_{\frac{N}{2}}$. For clarity, the sub-matrices $\boldsymbol{K}_{\frac{N}{2}}$ and $\boldsymbol{L}_{\frac{N}{2}}$ defined in (3.79) in explicit form for $N = 8$ are given by

$$\boldsymbol{K}_4 = \begin{pmatrix} \cos\frac{5\pi}{16} & \cos\frac{7\pi}{16} & -\cos\frac{7\pi}{16} & -\cos\frac{5\pi}{16} \\ -\cos\frac{\pi}{16} & -\cos\frac{5\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{\pi}{16} \\ \cos\frac{7\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{7\pi}{16} \\ \cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \cos\frac{\pi}{16} & -\cos\frac{3\pi}{16} \end{pmatrix}, \boldsymbol{L}_4 = \begin{pmatrix} -\cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\cos\frac{3\pi}{16} \\ \cos\frac{7\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & \cos\frac{7\pi}{16} \\ \cos\frac{\pi}{16} & -\cos\frac{5\pi}{16} & -\cos\frac{5\pi}{16} & \cos\frac{\pi}{16} \\ \cos\frac{5\pi}{16} & -\cos\frac{7\pi}{16} & -\cos\frac{7\pi}{16} & \cos\frac{5\pi}{16} \end{pmatrix},$$

and the sub-matrices $\boldsymbol{P}_{\frac{N}{2}}$ and $\boldsymbol{Q}_{\frac{N}{2}}$ in explicit form for $N = 8$ are given by

$$\boldsymbol{P}_4 = \begin{pmatrix} \cos\frac{3\pi}{16} & \cos\frac{\pi}{16} & \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} \\ \cos\frac{7\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & \cos\frac{7\pi}{16} \\ -\cos\frac{\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{5\pi}{16} & -\cos\frac{\pi}{16} \\ \cos\frac{5\pi}{16} & -\cos\frac{7\pi}{16} & -\cos\frac{7\pi}{16} & \cos\frac{5\pi}{16} \end{pmatrix}, \quad \boldsymbol{Q}_4 = \begin{pmatrix} \cos\frac{5\pi}{16} & \cos\frac{7\pi}{16} & -\cos\frac{7\pi}{16} & -\cos\frac{5\pi}{16} \\ \cos\frac{\pi}{16} & \cos\frac{5\pi}{16} & -\cos\frac{5\pi}{16} & -\cos\frac{\pi}{16} \\ \cos\frac{7\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{7\pi}{16} \\ -\cos\frac{3\pi}{16} & \cos\frac{\pi}{16} & -\cos\frac{\pi}{16} & \cos\frac{3\pi}{16} \end{pmatrix}.$$

From the explicit forms of sub-matrices immediately we can observe between pairs $\boldsymbol{K}_4$, $\boldsymbol{Q}_4$ and $\boldsymbol{L}_4$, $\boldsymbol{P}_4$ the following basic relations:

- Even-indexed row vectors of $\boldsymbol{K}_4$ are the same as even-indexed row vectors of $\boldsymbol{Q}_4$, while odd-indexed row vectors of $\boldsymbol{K}_4$ are equal sign changed odd-indexed row vectors of $\boldsymbol{Q}_4$.

- Odd-indexed row vectors of $\boldsymbol{L}_4$ are the same as odd-indexed row vectors of $\boldsymbol{P}_4$, while even-indexed row vectors of $\boldsymbol{L}_4$ are equal to sign changed even-indexed row vectors of $\boldsymbol{P}_4$.

Indeed, the above-mentioned basic relations can be expressed in the matrix forms as

$$\begin{aligned}
&\boldsymbol{K}_{\frac{N}{2}} = \boldsymbol{D}_{\frac{N}{2}}\, \boldsymbol{Q}_{\frac{N}{2}}, \quad \boldsymbol{K}_{\frac{N}{2}}^T = \boldsymbol{Q}_{\frac{N}{2}}^T\, \boldsymbol{D}_{\frac{N}{2}}, \quad \boldsymbol{Q}_{\frac{N}{2}} = \boldsymbol{D}_{\frac{N}{2}}\, \boldsymbol{K}_{\frac{N}{2}}, \quad \boldsymbol{Q}_{\frac{N}{2}}^T = \boldsymbol{K}_{\frac{N}{2}}^T\, \boldsymbol{D}_{\frac{N}{2}}, \\
&\boldsymbol{L}_{\frac{N}{2}} = -\boldsymbol{D}_{\frac{N}{2}}\, \boldsymbol{P}_{\frac{N}{2}}, \quad \boldsymbol{L}_{\frac{N}{2}}^T = -\boldsymbol{P}_{\frac{N}{2}}^T\, \boldsymbol{D}_{\frac{N}{2}}, \quad \boldsymbol{P}_{\frac{N}{2}} = -\boldsymbol{D}_{\frac{N}{2}}\, \boldsymbol{L}_{\frac{N}{2}}, \quad \boldsymbol{P}_{\frac{N}{2}}^T = -\boldsymbol{L}_{\frac{N}{2}}^T\, \boldsymbol{D}_{\frac{N}{2}}.
\end{aligned} \tag{3.84}$$

In general, investigating the symmetry properties of row vectors of $\boldsymbol{K}_{\frac{N}{2}}, \boldsymbol{Q}_{\frac{N}{2}}, \boldsymbol{P}_{\frac{N}{2}}$, and $\boldsymbol{Q}_{\frac{N}{2}}$ we obtain matrix forms given by

$$\begin{aligned}
&\boldsymbol{K}_{\frac{N}{2}}\, \boldsymbol{J}_{\frac{N}{2}} = -\boldsymbol{K}_{\frac{N}{2}}, \quad \boldsymbol{L}_{\frac{N}{2}}\, \boldsymbol{J}_{\frac{N}{2}} = \boldsymbol{L}_{\frac{N}{2}}, \qquad \boldsymbol{J}_{\frac{N}{2}}\, \boldsymbol{K}_{\frac{N}{2}}^T = \boldsymbol{K}_{\frac{N}{2}}^T, \quad \boldsymbol{J}_{\frac{N}{2}}\, \boldsymbol{L}_{\frac{N}{2}}^T = \boldsymbol{L}_{\frac{N}{2}}^T, \\
&\boldsymbol{P}_{\frac{N}{2}}\, \boldsymbol{J}_{\frac{N}{2}} = \boldsymbol{P}_{\frac{N}{2}}, \quad \boldsymbol{Q}_{\frac{N}{2}}\, \boldsymbol{J}_{\frac{N}{2}} = -\boldsymbol{Q}_{\frac{N}{2}}, \qquad \boldsymbol{J}_{\frac{N}{2}}\, \boldsymbol{P}_{\frac{N}{2}}^T = \boldsymbol{P}_{\frac{N}{2}}^T, \quad \boldsymbol{J}_{\frac{N}{2}}\, \boldsymbol{Q}_{\frac{N}{2}}^T = -\boldsymbol{Q}_{\frac{N}{2}}^T,
\end{aligned} \tag{3.85}$$

implying that row vectors of $\boldsymbol{K}_{\frac{N}{2}}$ and $\boldsymbol{Q}_{\frac{N}{2}}$ have the even anti-symmetry property, while row vectors of $\boldsymbol{L}_{\frac{N}{2}}$ and $\boldsymbol{P}_{\frac{N}{2}}$ have the even symmetry property (see Appendix G.1).

From (3.71) to (3.73), using block forms (3.79) and (3.83) we obtain properties of (the sum) products of $\boldsymbol{K}_{\frac{N}{2}}, \boldsymbol{L}_{\frac{N}{2}}$ and $\boldsymbol{P}_{\frac{N}{2}}, \boldsymbol{Q}_{\frac{N}{2}}$. Specifically, from (3.71) we get

$$\boldsymbol{K}_{\frac{N}{2}}\, \boldsymbol{K}_{\frac{N}{2}}^T + \boldsymbol{L}_{\frac{N}{2}}\, \boldsymbol{L}_{\frac{N}{2}}^T = \boldsymbol{P}_{\frac{N}{2}}\, \boldsymbol{P}_{\frac{N}{2}}^T + \boldsymbol{Q}_{\frac{N}{2}}\, \boldsymbol{Q}_{\frac{N}{2}}^T = \frac{N}{2}\, \boldsymbol{I}_{\frac{N}{2}}, \tag{3.86}$$

indicating that the matrix products $\boldsymbol{K}_{\frac{N}{2}}\boldsymbol{K}_{\frac{N}{2}}^T, \boldsymbol{L}_{\frac{N}{2}}\boldsymbol{L}_{\frac{N}{2}}^T, \boldsymbol{P}_{\frac{N}{2}}\boldsymbol{P}_{\frac{N}{2}}^T$, and $\boldsymbol{Q}_{\frac{N}{2}}\boldsymbol{Q}_{\frac{N}{2}}^T$ are symmetric matrices. Further, from (3.72) and (3.73) we get

$$\begin{aligned}
\boldsymbol{K}_{\frac{N}{2}}^T\, \boldsymbol{K}_{\frac{N}{2}} &= \boldsymbol{Q}_{\frac{N}{2}}^T\, \boldsymbol{Q}_{\frac{N}{2}} = \frac{N}{4}\left(\boldsymbol{I}_{\frac{N}{2}} - \boldsymbol{J}_{\frac{N}{2}}\right), \\
\boldsymbol{L}_{\frac{N}{2}}^T\, \boldsymbol{L}_{\frac{N}{2}} &= \boldsymbol{P}_{\frac{N}{2}}^T\, \boldsymbol{P}_{\frac{N}{2}} = \frac{N}{4}\left(\boldsymbol{I}_{\frac{N}{2}} + \boldsymbol{J}_{\frac{N}{2}}\right),
\end{aligned} \tag{3.87}$$

and

$$\begin{aligned}
&\boldsymbol{K}_{\frac{N}{2}}\, \boldsymbol{L}_{\frac{N}{2}}^T = \boldsymbol{L}_{\frac{N}{2}}\, \boldsymbol{K}_{\frac{N}{2}}^T = \boldsymbol{0}_{\frac{N}{2}}, \qquad \boldsymbol{K}_{\frac{N}{2}}^T\, \boldsymbol{L}_{\frac{N}{2}} = \boldsymbol{L}_{\frac{N}{2}}^T\, \boldsymbol{K}_{\frac{N}{2}} = \boldsymbol{0}_{\frac{N}{2}}, \\
&\boldsymbol{P}_{\frac{N}{2}}\, \boldsymbol{Q}_{\frac{N}{2}}^T = \boldsymbol{Q}_{\frac{N}{2}}\, \boldsymbol{P}_{\frac{N}{2}}^T = \boldsymbol{0}_{\frac{N}{2}}, \qquad \boldsymbol{P}_{\frac{N}{2}}^T\, \boldsymbol{Q}_{\frac{N}{2}} = \boldsymbol{Q}_{\frac{N}{2}}^T\, \boldsymbol{P}_{\frac{N}{2}} = \boldsymbol{0}_{\frac{N}{2}}.
\end{aligned} \tag{3.88}$$

Matrix products in (3.88) are based on the fact that the scalar products of row/column even anti-symmetric/symmetric or even symmetric/anti-symmetric nonzero vectors are always equal to zero. Finally, knowing the symmetry properties of row vectors given by (3.85) the same properties hold for the matrix products:

$$\begin{aligned} \boldsymbol{K}_{\frac{N}{2}}\,\boldsymbol{P}_{\frac{N}{2}}^{T} &= \boldsymbol{P}_{\frac{N}{2}}\,\boldsymbol{K}_{\frac{N}{2}}^{T} = \boldsymbol{0}_{\frac{N}{2}}, \\ \boldsymbol{L}_{\frac{N}{2}}\,\boldsymbol{Q}_{\frac{N}{2}}^{T} &= \boldsymbol{Q}_{\frac{N}{2}}\,\boldsymbol{L}_{\frac{N}{2}}^{T} = \boldsymbol{0}_{\frac{N}{2}}. \end{aligned} \tag{3.89}$$

3.2.3 Relation Between the Evenly and Oddly Stacked MDCT

The relations between the MDCT and corresponding MDST block transforms both for the evenly and oddly stacked systems given by (3.20) and (3.61) allow us to concentrate on the MDCT block transforms only.

Consider the evenly stacked MDCT block transform defined by (3.16), and the oddly stacked MDCT block transform defined by (3.49). The oddly stacked MDCT coefficients $\{c_k^{o}\}$ satisfy the following relation [19]

$$\begin{aligned} c_k^{o} + c_{k-1}^{o} &= c_k^{E} \quad \text{for} \quad k > 0, \\ 2c_0^{o} &= c_0^{E}, \end{aligned} \tag{3.90}$$

where $\{c_k^{E}\}$ are evenly stacked MDCT coefficients. In fact, using the trigonometric identity $\cos(\alpha+\beta) = 2\cos(\alpha)\cos(\beta) - \cos(\alpha-\beta)$ and setting $\alpha = \frac{\pi}{N}(2n+1+\frac{N}{2})k$, $\beta = \frac{\pi}{2N}(2n+1+\frac{N}{2})$, the oddly stacked MDCT given by (3.49) can be written in the form

$$\begin{aligned} c_k^{o} &= \sum_{n=0}^{N-1} \left(2\, x_n \, \cos\left[\frac{\pi}{2N} \left(2n+1+\frac{N}{2} \right) \right] \right) \cos\left[\frac{\pi}{N} \left(2n+1+\frac{N}{2} \right) k \right] - c_{k-1}^{o}, \\ k &= 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{3.91}$$

The transform kernel under the sum of (3.91) is recognized as an N-point evenly stacked MDCT of the modified input data sequence. Hence, the N-point oddly stacked MDCT is converted to the evenly stacked MDCT of the same size with additional N pre-multipliers and $\frac{N}{2}-1$ recursive post-additions. The relation between oddly stacked MDCT and evenly stacked MDCT is quite similar to that of between the DCT-IV and DCT-II (see Appendix C.3). From (3.91) we have

$$c_k^{o} = \sum_{n=0}^{N-1} y_n \quad \cos\left[\frac{\pi}{N} \left(2n+1+\frac{N}{2} \right) k \right] \quad - \quad c_{k-1}^{o}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{3.92}$$

where

$$y_n = 2x_n \cos\left[\frac{\pi}{2N}(2n+1+\frac{N}{2})\right], \qquad n = 0, 1, \ldots, N-1, \tag{3.93}$$

and

$$\begin{aligned} 2c_0^{\,o} &= c_0^{\,E}, \quad \text{if } k = 0, \\ c_k^{\,o} &= c_k^{\,E} - c_{k-1}^{\,o} \quad \text{for } k > 0, \end{aligned} \tag{3.94}$$

The relation between the oddly stacked MDCT and evenly stacked MDCT given by (3.92) can be equivalently represented by the matrix product

$$\boldsymbol{C}^{\,o}_{\frac{N}{2}\times N} = \boldsymbol{B}_{\frac{N}{2}} \ \boldsymbol{C}^{\,E}_{\frac{N}{2}\times N} \ \text{diag}\left\{2\cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)\right]\right\}_N, \tag{3.95}$$

where $\boldsymbol{B}_{\frac{N}{2}}$ is a lower triangular matrix of order $\frac{N}{2}$ given by

$$\boldsymbol{B}_{\frac{N}{2}} = \begin{pmatrix} \frac{1}{2} & 0 & 0\,0 & \cdots & 0 \\ -\frac{1}{2} & 1 & 0\,0 & \cdots & 0 \\ \frac{1}{2} & -1 & 1\,0 & \cdots & 0 \\ -\frac{1}{2} & 1 & -1\,1 & \cdots & 0 \\ \vdots & \vdots & \vdots\,\vdots & \ddots & \vdots \\ -\frac{1}{2} & 1 & -1\,1 & \cdots & 1 \end{pmatrix}. \tag{3.96}$$

On the other hand, the matrix $\boldsymbol{B}_{\frac{N}{2}}$ and diagonal matrix on the right-hand side of (3.95) are nonsingular and therefore, their inverse matrices exist. Performing matrix multiplications on both sides of (3.95), i.e., left-multiplying by $\boldsymbol{B}^{-1}_{\frac{N}{2}}$, then right-multiplying by inverse diagonal matrix, we get

$$\boldsymbol{C}^{\,E}_{\frac{N}{2}\times N} = \boldsymbol{B}^{-1}_{\frac{N}{2}} \ \boldsymbol{C}^{\,o}_{\frac{N}{2}\times N} \ \text{diag}\left\{\frac{1}{2\,\cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)\right]}\right\}_N, \tag{3.97}$$

where $\boldsymbol{B}^{-1}_{\frac{N}{2}}$ is the lower bidiagonal matrix of order $\frac{N}{2}$ given by

$$\boldsymbol{B}^{-1}_{\frac{N}{2}} = \begin{pmatrix} 2 & 0 & 0 & & \cdots & 0 \\ 1 & 1 & 0 & & \cdots & 0 \\ 0 & 1 & 1 & & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 1\ 1 & 0 \\ 0 & 0 & 0 & \cdots & 0\ 1 & 1 \end{pmatrix}, \tag{3.98}$$

and the computation of evenly stacked MDCT can be realized via an existing fast oddly stacked MDCT algorithm only by pre- and post-processing of input and output data sequences.

3.2.4 Windowing Procedure and Perfect Reconstruction Conditions

We discussed so far both the evenly and oddly stacked systems in the forms of block transforms or alternatively in matrix-vector representations and we investigated their general properties in time and frequency domains. But the complete TDAC evenly and oddly stacked systems include the windowing procedure to eliminate the blocking artifacts. On the other hand, to achieve the TDAC, i.e., the original data sequence will be perfectly reconstructed by the synthesis filter bank, the analysis and synthesis windowing functions have to satisfy the so-called perfect reconstruction conditions [18, 24].

Let $\{x_n^{(m)}\}$ and $\{x_n^{(m+1)}\}$ be two adjacent overlapped data blocks. Transform these data blocks by the forward MDCT followed by the backward MDCT block transform. Assuming that no changes are made in data blocks during the transformation, we find that the original signal in the overlapped part is perfectly reconstructed even without windowing. Indeed, according to the overlap/add procedure given by (3.5) and using (3.31), (3.32) for the evenly stacked system or (3.76) for the oddly stacked system, whereby in the overlapped part the following relations hold: $x_n^{(m+1)} = x_{\frac{N}{2}+n}^{(m)}$ and $x_{\frac{N}{2}-1-n}^{(m+1)} = x_{N-1-n}^{(m)}$ for $n = 0, 1, \ldots, \frac{N}{2} - 1$, and we have

$$\hat{x}_{\frac{N}{2}+n}^{(m)} + \hat{x}_n^{(m+1)} = (x_{\frac{N}{2}+n}^{(m)} + x_{N-1-n}^{(m)}) + (x_n^{(m+1)} - x_{\frac{N}{2}-1-n}^{(m+1)}) = 2x_{\frac{N}{2}+n}^{(m)},\ n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{3.99}$$

and the second half of the original data sequence $\{2x_{N-1-n}^{(m)}\}$ in the overlapped part can be obtained from the symmetry properties of proper $\{\hat{x}_{\frac{N}{2}+n}^{(m)}\}$ and $\{\hat{x}_n^{(m+1)}\}$.

The main objective in audio compression applications is to represent the transformed audio signal by fewer bits, while keeping the audio quality at an acceptable level. In modern encoders, signal parts that are not audible are removed resulting in a loss of information. Consequently, after coding and decoding the transformed signal is changed slightly and we can no longer expect a perfect reconstruction compared with the original audio signal. Of course, the remaining information that is passed should be analyzed and synthesized without errors. A serious problem here is that when data blocks are obtained using rectangular windows, sudden signal changes ("discontinuities") at the block boundaries are to be expected, that will affect the coded data in an unrecoverable way. To eliminate these so-called

"blocking artifacts," each data block is multiplied by a windowing function such that the data block ends smoothly at both boundaries, while keeping overall gain constant during the block transition. In order to accomplish this while keeping the perfect reconstruction property for the analysis/synthesis process, the windowing functions are applied to both the input and output of the transform procedure as follows.

Let $h_n^{(0)}$ and $h_n^{(1)}$ be the windowing functions applied to the input data blocks m and $m+1$, respectively. Assume the windowed data blocks $\{h_n^{(0)}\, x_n^{(m)}\}$, $\{h_n^{(1)}\, x_n^{(m+1)}\}$, $n = 0, 1, \ldots, N-1$, are transformed by the forward MDCT block transform. Let $g_n^{(0)}$ and $g_n^{(1)}$ be the windowing functions applied to the outputs after performing the backward MDCT block transform. Then from (3.99) we have

$$\begin{aligned} & g^{(0)}_{\frac{N}{2}+n}\,(h^{(0)}_{\frac{N}{2}+n}\, x^{(m)}_{\frac{N}{2}+n} + h^{(0)}_{N-1-n}\, x^{(m)}_{N-1-n}) \;+\; g^{(1)}_n\,(h^{(1)}_n\, x^{(m)}_{\frac{N}{2}+n} - h^{(1)}_{\frac{N}{2}-1-n}\, x^{(m)}_{N-1-n}) \\ & = (g^{(0)}_{\frac{N}{2}+n}\, h^{(0)}_{\frac{N}{2}+n} + g^{(1)}_n\, h^{(1)}_n)\, x^{(m)}_{\frac{N}{2}+n} \;+\; (g^{(0)}_{\frac{N}{2}+n}\, h^{(0)}_{N-1-n} - g^{(1)}_n\, h^{(1)}_{\frac{N}{2}-1-n})\, x^{(m)}_{N-1-n}. \end{aligned}$$

In order to recover the original data sequence $\{x^{(m)}_{n+\frac{N}{2}}\}$, $n = 0, 1, \ldots, \frac{N}{2}-1$, we need the coefficient of $x^{(m)}_{\frac{N}{2}+n}$ to be one and the coefficient of $x^{(m)}_{N-1-n}$ to be zero. In other words, the perfect-reconstruction conditions for windowing functions have to be satisfied as follows

$$\begin{aligned} & g^{(0)}_{\frac{N}{2}+n}\, h^{(0)}_{\frac{N}{2}+n} \;+\; g^{(1)}_n\, h^{(1)}_n = 1, \\ & g^{(0)}_{\frac{N}{2}+n}\, h^{(0)}_{N-1-n} \;-\; g^{(1)}_n\, h^{(1)}_{\frac{N}{2}-1-n} = 0, \qquad n = 0, 1, \ldots, \frac{N}{2}-1. \end{aligned} \tag{3.100}$$

Usually, the same windowing functions are used for all data blocks, so we can drop the superscripts, i.e., $h_n = h_n^{(0)} = h_n^{(1)}$ and $g_n = g_n^{(0)} = g_n^{(1)}$. If $h_n = g_n$ is identical both for the analysis and synthesis MDCT filter banks, then from (3.100) the perfect-reconstruction conditions in the overlapped part are given by

$$\begin{aligned} & h_n^2 + h^2_{\frac{N}{2}+n} = 1, \\ & h_n = h_{N-1-n}, \quad \text{or} \quad h_{\frac{N}{2}+n} = h_{\frac{N}{2}-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{2}-1. \end{aligned} \tag{3.101}$$

Equation (3.101) implies that the windowing function $\{h_n\}$ has to be symmetric. The reader is referred to a draft document [30] where mathematical properties of the oddly stacked MDCT are proved using basic trigonometry only.

3.2.4.1 Matrix Representation of the Windowing Procedure

Denote by $\{w_n\}$, $n = 0, 1, \ldots, N-1$, a symmetric windowing function identical for the analysis and synthesis filter bank. Let $\boldsymbol{W}_N$ be a diagonal matrix with elements $\{w_n\}$ on the main diagonal defined as

$$\boldsymbol{W}_N = \begin{pmatrix} \boldsymbol{W}_{\frac{N}{2}}^{(1)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_{\frac{N}{2}}^{(2)} \end{pmatrix}, \tag{3.102}$$

where $\boldsymbol{W}_{\frac{N}{2}}^{(1)}$ and $\boldsymbol{W}_{\frac{N}{2}}^{(2)}$ represent the first and the second half of the windowing function, respectively. In order to ensure TDAC, windowing functions of two succeeding data blocks have to satisfy the perfect reconstruction conditions (3.101) which are defined in the matrix form by Britanak and Arriëns [20]

$$\boldsymbol{W}_{\frac{N}{2}}^{(1)}\, \boldsymbol{W}_{\frac{N}{2}}^{(1)} + \boldsymbol{W}_{\frac{N}{2}}^{(2)}\, \boldsymbol{W}_{\frac{N}{2}}^{(2)} = \boldsymbol{I}_{\frac{N}{2}}, \qquad \boldsymbol{W}_{\frac{N}{2}}^{(2)} = \boldsymbol{J}_{\frac{N}{2}}\, \boldsymbol{W}_{\frac{N}{2}}^{(1)}\, \boldsymbol{J}_{\frac{N}{2}}, \tag{3.103}$$

where $\boldsymbol{I}_{\frac{N}{2}}$ is the identity matrix, and $\boldsymbol{J}_{\frac{N}{2}}$ is the opposite diagonal identity matrix, both of order $\frac{N}{2}$.

3.2.5 *Design of a Windowing Function*

Since a windowing function has a significant impact on the MDCT filter bank frequency response [32, 33] motivated by results described in [35], an important problem in MDCT-based perceptual audio codecs has been the design of an additional windowing function (or prototype finite impulse response low-pass filter) satisfying the perfect reconstruction conditions.

Smoothing windowing functions, in general, are characterized in the frequency domain by the normalized DFT spectrum [35, 36, 42]. The frequency log-magnitude response of a windowing function consists of the main-lobe and sidelobes with increasing frequency. The reduced level of monotonically decreasing sidelobe magnitudes is related to the reduction of audible blocking artifacts (leakage attenuation or stop-band attenuation), while the width of main-lobe is related to the accurate frequency identification (frequency resolution) of the windowing function. However, a trade-off exists between the level of side-lobe magnitudes and main-lobe width. As the magnitude of side-lobes is decreased, the main-lobe width is increased. This is consistent with the reciprocal relationship between the frequency and spatial domains, i.e., narrow frequency bandwidths correspond to wide spatial functions [32, 42]. On the other hand, it was shown that the shape of windowing function determines the degree of frequency separation of the MDCT filter bank [35]. Thus, some major factors in the design of filter banks for audio coding are the ability to maximize their frequency separation and the ability to minimize spectral

leakage. Two parameters of the windowing function are directly linked to these properties: selected size of the data block and the shape of windowing function. Generally, the best choice of windowing function is application-dependent [32].

3.2.5.1 Commonly Used Windowing Functions in Audio Coding

Frequency selectivity of the MDCT filter bank is dependent on a windowing function [32, 33]. In the following, commonly used symmetric windowing functions in audio coding applications satisfying perfect reconstruction conditions (3.101) are presented.

- The rectangular windowing function defined as [32]

$$w_n = \frac{1}{\sqrt{2}}, \qquad n = 0, 1, \ldots, N-1. \tag{3.104}$$

  This windowing function is not often used in audio codecs because it produces an MDCT filter bank with poor frequency resolution.
- The sine windowing function is defined as [32, 38, 41, 63]

$$w_n = \sin\left[\frac{\pi(2n+1)}{2N}\right], \qquad n = 0, 1, \ldots, N-1. \tag{3.105}$$

  It produces good frequency separation (in particular, DC energy is concentrated in a single coefficient) and it is asymptotically optimal in terms of coding gain (efficiency). Therefore, its properties are typically referred as the performance benchmark when a new windowing function is proposed. The sine windowing function is used in the MP3 [32] and MPEG-2 AAC [32, 33] audio coding standards. A plot of the sine windowing function for $N = 2048$ is shown in Fig. 3.1.
- The coding efficiency of MDCT filter bank can be improved replacing the sine windowing function by a windowing function having better leakage attenuation property for some class of signals. Such a parametric, the so-called Kaiser–Bessel Derived (KBD) windowing function, was constructed by Dolby Labs [32, 38, 41]. The parametrized KBD windowing function is derived by a numerical procedure from the parametrized symmetric modified zeroth-order Kaiser–Bessel function [32] (see also Chap. 6). In general, a parametric windowing function leaves much flexibility in controlling its shape and hence, desired frequency resolution (mainlobe width) and leakage attenuation property (level of sidelobe magnitudes) by adjusting its free parameter. The KBD windowing function is employed in the Dolby Digital (Plus) AC-3 audio compression algorithms with parameter $\alpha = 5$, while in the MPEG-4 AAC with $\alpha = 4$ (long windowing function) for stationary signals, and with $\alpha = 6$ (short windowing function) for transient signals [33]. Plot of the KBD windowing function for $N = 512$ and $\alpha = 5$ is shown in Chap. 6 (see Fig. 6.1).

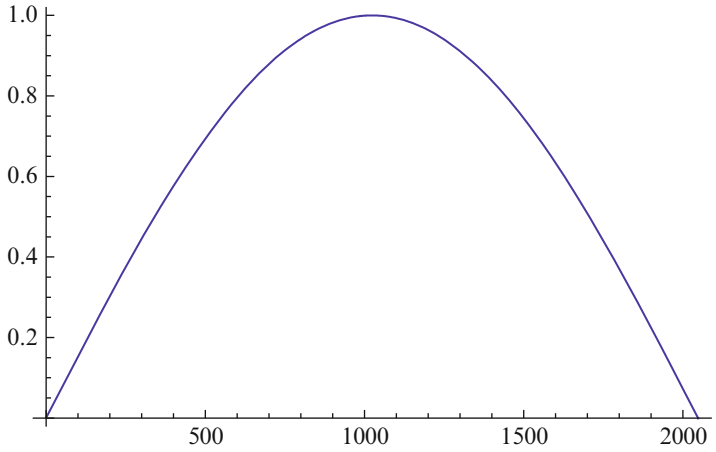

Fig. 3.1 Plot of the sine windowing function for $N = 2048$

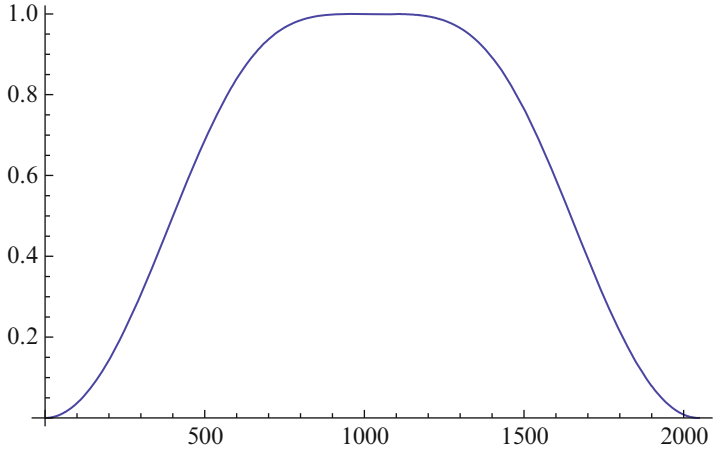

Fig. 3.2 Plot of the Ogg Vorbis windowing function for $N = 2048$

- The Ogg Vorbis audio compression algorithm (fully open, nonproprietary, and patent royalty-free distributed) uses the windowing function defined by Wright [30]

$$w_n = \sin\left[\frac{\pi}{2}\,\sin^2\left(\frac{\pi(2n+1)}{2N}\right)\right], \qquad n = 0, 1, \ldots, N-1. \tag{3.106}$$

A plot of the Ogg Vorbis windowing function for $N = 2048$ is shown in Fig. 3.2.

The Ogg Vorbis windowing function, but defined in a low overlap form (the so-called power complementary windowing function with reduced overlap), is used

in the royalty-free Constrained-Energy Lapped Transform (CELT) audio codec [46, 47] with especially low algorithmic delay intended for applications, where high-quality and low delay are desired. The original stand-alone CELT codec has been integrated as a one layer into the open, royalty-free hybrid OPUS interactive speech and audio codec [45, 48].

Note 4: In general, any customized windowing function for the MDCT filter bank satisfying perfect reconstruction conditions (3.101) can be derived from a parametrized initial symmetric nonnegative kernel function of the length $M + 1$. The resulting windowing function $\{w_n\}$ is generated by applying a transformation of the form [32, 41]

$$w_n = \sqrt{\frac{\sum_{j=0}^{n} v_j}{\sum_{j=0}^{M} v_j}}, \qquad n = 0, 1, \ldots, M-1, \tag{3.107}$$

where $\{v_j\}$ represents the symmetric nonnegative kernel. The resulting identical analysis and synthesis windowing function $\{w_n\}$ is of the length M and symmetric. The simple parametrized and symmetric Landau nonnegative kernel function is reported in [39]. It is defined in the analytical form as

$$v(x) = (1 - x^2)^{\beta}, \qquad x \in < -1, 1 >, \tag{3.108}$$

where β is the free parameter. Taking a discrete version of (3.108) as an initial symmetric nonnegative kernel given by

$$v_n = \left[1 - \left(\frac{n - \frac{N}{4}}{\frac{N}{4}}\right)^2\right]^{\beta}, \qquad n = 0, 1, \ldots, \frac{N}{2}, \tag{3.109}$$

and by applying the normalization procedure defined by (3.107) we obtain a customized, the so-called Landau Designed (LD) windowing function.

3.2.5.2 Low (Reduced) Overlap Windowing Functions

The primary sources of algorithmic delay inherent in the encoding/decoding chain of MDCT-based audio coding schemes is the block-based processing associated with data block size and filter bank algorithmic delay due to the overlap/add procedure of the filter bank with 50% overlap of previous and subsequent data blocks. We note that the algorithmic delay is defined as the theoretical minimum delay allowed by an algorithm [43, 44].

In order to minimize the total algorithmic delay of MDCT-based audio coding schemes, the transform block size is reduced and it is combined with a low (reduced) overlap windowing function. As an example, the low (reduced) overlap symmetric windowing function [46–48]. is constructed as follows. The 512-sample MDCT

data block is divided into 256-sample frames, with each block composed of two frames. To reduce the delay, the overlap is only 128 samples, with a 128-sample constant region in the center (samples are equal to 1), and 64 zero samples on each side. For the overlap region the Ogg Vorbis windowing function is constructed [see Eq. (3.106) for $N = 128$]. Thus, the effective overlap is reduced with the zeros on each side, while still maintaining the perfect reconstruction property. This results in an algorithmic delay of 384 samples for 256-sample frame size with 128 samples look-ahead (see Fig. 1 in [46], or Fig. 3 in [47]). The same windowing function is used for the analysis process and overlap/add synthesis process.

3.2.5.3 Biorthogonal Conditions for Nonidentical Windowing Functions

Although in the most audio coding applications the identical windowing functions both for the analysis and synthesis TDAC filter banks are used, this assumption is not necessary to maintain the perfect reconstruction property. The perfect reconstruction can be maintained also in the case, when the analysis and synthesis windowing functions are nonidentical [40]. Relaxing the constraint on windowing function shape allows an additional degree of freedom for controlling the shape of windowing function and frequency domain selectivity of a filter bank. Such TDAC filter banks are called biorthogonal. In the following the biorthogonal conditions are derived which enable us to generate the so-called biorthogonal TDAC filter banks [58, 65].

Let $\{h_n\}$ and $\{g_n\}$ be the analysis and synthesis windowing functions, respectively. Consider (3.100) defining the perfect reconstruction conditions for nonidentical windowing functions, where $h_n = h_n^{(0)} = h_n^{(1)}$ and $g_n = g_n^{(0)} = g_n^{(1)}$. Assuming a symmetric analysis windowing function $\{h_n\}$, i.e., $h_n = h_{N-1-n}$ or $h_{\frac{N}{2}+n} = h_{\frac{N}{2}-1-n}$, $n = 0, 1, \ldots, \frac{N}{2} - 1$, the second equation in (3.100) can be rearranged to give

$$g_{\frac{N}{2}+n}\, h_n - g_n\, h_{\frac{N}{2}+n} = 0, \tag{3.110}$$

and from (3.110) we have

$$g_{\frac{N}{2}+n} = \frac{g_n\, h_{\frac{N}{2}+n}}{h_n}. \tag{3.111}$$

Substituting (3.111) into the first equation in (3.100) we get

$$\frac{g_n\, h_{\frac{N}{2}+n}^2}{h_n} + \frac{g_n\, h_n^2}{h_n} = 1, \tag{3.112}$$

which is simplified to

$$g_n\, (h_{\frac{N}{2}+n}^2 + h_n^2) = h_n. \tag{3.113}$$

Finally, from (3.113) we obtain the complete description of the synthesis windowing function with respect to the analysis windowing function defined as [40]

$$g_n = \frac{h_n}{h^2_{\frac{N}{2}+n} + h^2_n}, \quad \text{where} \quad h^2_{\frac{N}{2}+n} + h^2_n \neq 0, \quad n = 0, 1, \ldots, \frac{N}{2} - 1, \tag{3.114}$$

where the generated synthesis windowing function $\{g_n\}$ is also symmetric. Equation (3.114) defines the biorthogonal conditions which guarantee the perfect reconstruction of TDAC filter banks with nonidentical analysis and synthesis windowing functions. The incorporation of biorthogonality into TDAC filter banks allows for greater flexibility in the design of analysis and synthesis windowing functions by increasing the number of degrees of freedom. Essentially, this fact has an impact on the proper implementation of psychoacoustics modeling [58].

As an example, however, with exchanged roles of analysis and synthesis windowing functions, the synthesis windowing function for biorthogonal filter banks has been proposed in [65], and it is defined by

$$g_n = \frac{1 - \cos\left[\pi \left(\frac{n+1}{2N}\right)^{\alpha}\right] + \beta}{2 + \beta}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1, \tag{3.115}$$

where the analysis windowing function is defined by (3.114). The parameter α controls the width of the windowing function, whereas β controls its end values. Plots of the analysis and synthesis windowing functions can be found in [65].

3.2.6 *Adaptive Switching of Transform Block Sizes and Windowing Functions*

Some of the major factors that play a key role in the design of filter banks for audio coding is the ability to minimize the effects of audible blocking artifacts and to maximize the frequency separation of the filter bank. Two parameters of the data block are directly related to these properties, specifically, the selected transform block size and shape of windowing function. The perfect reconstruction conditions involve the overlapped parts between two adjacent data blocks. If we use a single windowing function and constant block size, the perfect reconstruction conditions actually refer to the right and left half sides of consecutive data blocks [32].

In the following subsections the methods for adaptive switching of transform block sizes and adaptive switching of windowing functions including the corresponding perfect reconstruction conditions in the overlapped part are discussed.

3.2.6.1 Adaptive Switching of Transform Block Sizes

Transform block size switching is an effective tool for adapting the time/frequency resolution of the filter bank during the signal coding without losing the perfect reconstruction property. Such filter banks are also called the time-varying filter banks. The windowed long block is transformed when the signal spectrum remains stationary, or varies only slowly with time. During transients, when the signal changes rapidly in time, shorter blocks are more suitable to reduce pre-echo effects. In transition from the long to short block(s) and vice versa, the so-called transition blocks (labeled as "start" and "stop") have to be constructed to preserve the perfect reconstruction property in the overlapped part [34, 37].

Let N^{long} and N^{short} be the lengths of long and short data blocks, respectively. Assume a ("start") transition block from the long to short block. Then, the right half side of long block is first divided into three regions and the analysis/synthesis windowing function is redefined in each region as follows [34]: The $\frac{N^{\text{long}}}{4} - \frac{N^{\text{short}}}{4}$ values of the windowing function in the first region of transition block (which do not overlap with the short block) are set to one. $\frac{N^{\text{short}}}{2}$ values at the second region correspond to the right half side of short windowing function. The last $\frac{N^{\text{long}}}{4} - \frac{N^{\text{short}}}{4}$ values in the third region of transition block are set to zero. The ("stop") transition block from short to long block is defined as the time reversed version of the "start" transition block. Structure of the transition block from long to short block is shown in [32, 34]. The construction of transition blocks results in asymmetric windowing functions for the long blocks.

As an example, the basic windowing function utilized by MP3 audio coding standard is the sine windowing function given by (3.105). The MP3 specifies two different block sizes: the long block ($N = 36$) and the short block ($N = 12$). The windowing function for transition from the long to short block ("start" transition block) is defined as [32]

$$w_n = \begin{cases} \sin\left[\frac{\pi}{36}(n+\frac{1}{2})\right], & n = 0, 1, \ldots, 17, \\ 1, & n = 18, 19, \ldots, 23, \\ \sin\left[\frac{\pi}{12}(n-18+\frac{1}{2})\right], & n = 24, 25, \ldots, 29, \\ 0, & n = 30, 31, \ldots, 35, \end{cases}$$

while the windowing function for transition from the short to long block ("stop" transition block) is defined as [32]

$$w_n = \begin{cases} 0, & n = 0, 1, \ldots, 5, \\ \sin\left[\frac{\pi}{12}(n - 6 + \frac{1}{2})\right], & n = 6, 7, \ldots, 11, \\ 1, & n = 12, 13, \ldots, 17, \\ \sin\left[\frac{\pi}{36}(n + \frac{1}{2})\right], & n = 18, 19 \ldots, 35. \end{cases}$$

On the other hand, in the MPEG-2 AAC audio coding standard the transform block size can be set to either $N = 2048$ (the long block) or to $N = 256$ (the short block) [32, 33]. We note that the long block corresponds to eight short blocks.

3.2.6.2 Adaptive Switching of Windowing Functions

We recall that the windowing function has the significant impact on the filter bank frequency response. Therefore, no single windowing function is optimal for all signals. Given a certain block size for the input data to the filter bank, the selection of shape of windowing function determines the degree of spectral separation of the filter bank. Thus, depending on the characteristics of the input audio signal changing the windowing function may provide better frequency resolution for the signal representation [32]. The conditions for perfect reconstruction in the overlapped part are derived as follows.

Let $\{h_n\}$ and $\{g_n\}$ be two different symmetric windowing functions applied to two adjacent data blocks m and $m + 1$, respectively. Then, based on overlap/add procedure given by (3.99) incorporating the windowing procedure we have

$$\begin{aligned} h_{\frac{N}{2}+n}\, \hat{x}^{(m)}_{\frac{N}{2}+n} + g_n\, \hat{x}^{(m+1)}_n &= h_{\frac{N}{2}+n}\, (h_{\frac{N}{2}+n}\, x^{(m)}_{\frac{N}{2}+n} + h_{N-1-n}\, x^{(m)}_{N-1-n}) \\ &\quad + g_n (g_n\, x^{(m+1)}_n - g_{\frac{N}{2}-1-n}\, x^{(m+1)}_{\frac{N}{2}-1-n}) \\ &= h_{\frac{N}{2}+n}\, (h_{\frac{N}{2}+n}\, x^{(m)}_{\frac{N}{2}+n} + h_{N-1-n}\, x^{(m)}_{N-1-n}) \\ &\quad + g_n (g_n\, x^{(m)}_{\frac{N}{2}+n} - g_{\frac{N}{2}-1-n}\, x^{(m)}_{N-1-n}). \end{aligned} \tag{3.116}$$

By algebraic rearranging the expression on the right-hand of (3.116) we get

$$(h^2_{\frac{N}{2}+n} + g^2_n)\, x^{(m)}_{\frac{N}{2}+n} + (h_{\frac{N}{2}+n}\, h_{N-1-n} - g_n\, g_{\frac{N}{2}-1-n})\, x^{(m)}_{N-1-n}. \tag{3.117}$$

In order to recover the original data sequence $\{x^{(m)}_{n+\frac{N}{2}}\}$, $n = 0, 1, \ldots, \frac{N}{2} - 1$, we need the coefficient of $x^{(m)}_{\frac{N}{2}+n}$ to be one and the coefficient of $x^{(m)}_{N-1-n}$ to be zero in (3.117), i.e., the perfect-reconstruction conditions for windowing functions in the overlapped part are satisfied when it holds [34]

$$\begin{aligned} &h^2_{\frac{N}{2}+n} + g^2_n = 1, \\ &h_{\frac{N}{2}+n}\, h_{N-1-n} = g_n\, g_{\frac{N}{2}-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{3.118}$$

Equation (3.118) defines the general perfect reconstruction conditions for the adaptive switching of windowing functions in the overlapped part of two adjacent data blocks. Switching of the windowing function is accomplished by designing a pair of "transition" windowing functions for which one side of each overlaps correctly with the previous windowing function, while the other side of each overlaps correctly with the following windowing function. It means that the right half side of windowing function for the data block m is mirror symmetric to the left half side of subsequent windowing function for the data block $m + 1$ in the overlapped part. We note that the windowing function applied to the data block m becomes asymmetric.

As an example, in the MPEG-2 AAC audio coding standard the windowing function can be varied dynamically (sine or KBD windowing function) as a function of the signal characteristics. Thus, the change of windowing function shape allows the MDCT filter bank to efficiently separate spectral components for a wider variety of input signals, and at the same time maximizing coding efficiency. In fact, the sine windowing function ensures better close-selectivity because the main-lobe is narrower, while the KBD windowing function has better stop-band attenuation because it has lower level of side-lobe magnitudes. The adaptive switching of windowing function in the MPEG-2 AAC is used for the long blocks only, and is applied to the second half of windowing function, since the first half is constrained by the shape of windowing function from the preceding data block [32, 33].

The adaptive switching of transform block sizes and windowing functions allows us to change from the series of long transform blocks into a series of shorter transform blocks and vice versa, or from a series of one windowing function to a series of different windowing functions provided that we appropriately handle the perfect reconstruction conditions for each overlapped part. In summary, by adapting the transform block sizes and the windowing functions to the characteristics of input signal, the performance of an audio coding scheme is significantly improved. Illustrative figures of just discussed methods can be found in [32–34, 37].

3.3 Lapped Transforms

A class of lapped transforms including the lapped orthogonal transform (LOT) [49, 52, 53, 55–57, 63], a modified version of LOT, the pseudolapped orthogonal transform (PseudoLOT) [54], the modulated lapped transform [59, 63, 66], biorthogonal and nonuniform lapped transforms [58, 65], and the extended lapped transform [60–64, 66] has been applied in transform/sub-band coding and studied independently with respect to the oddly stacked MDCT and MDST filter banks discussed in previous sections. We recall that the evenly stacked MDCT/MDST system is not a lapped transform [63].

3.3.1 Lapped (Orthogonal) Transform

The main motivation in the original developing the LOT was the reduction of blocking artifacts in transform coding, i.e., the reduction of discontinuities in the reconstructed signal at the block boundaries. The idea was to extend the basis functions beyond the block boundaries, creating an overlap, in order to eliminate these blocking artifacts [52, 55]. Thus, the basis functions of a lapped transform (LT) overlap consecutive data blocks.

With the LT, L-sample data block is mapped into M transform coefficients, where L is the length of basis functions, whereby $L > M$. In order to keep the total sampling rate, M new transform coefficients are computed for every new M input samples, i.e., there exists an overlap of $L - M$ samples in the computation of transform coefficients of consecutive data blocks. Compared to standard block transforms, like the type-II DCT (DCT-II), where the orthogonality guarantees perfect reconstruction, with LT however, the orthogonality of basis functions is not sufficient due to their overlapping. There are additional orthogonality constraints among overlapping portions of the basis functions formulated in the time domain [56, 57, 63] (see matrix representation of LTs). Historically, the term LT is a general lapped transform with orthogonal basis functions, whereas the LOT is a special case of an LT with $L = 2M$, i.e., two adjacent data blocks are overlapped by M samples, and the basis functions have even/odd symmetry. The theory of lapped transforms and their properties are discussed in detail in [53, 63].

3.3.1.1 Matrix Representation of Lapped Transforms

An elegant matrix representation of the forward and backward LT with a common overlapping factor (or the number of overlapping data blocks) have been presented in [31, 56, 66], and it is worthwhile to review here.

In general, the LT is a linear projection from R^{KM} onto R^{M}, where $K > 1$ is the overlapping factor being a positive integer, $L = KM$, and M is the block length. To realize LT, the input signal is first partitioned into M-point data blocks as

$$\mathbf{x}_i = (x_{iM+0} \;\; x_{iM+1} \;\; \ldots \;\; x_{iM+M-1})^T. \tag{3.119}$$

Then, K consecutive data blocks are grouped into $KM \times 1$ vector as

$$\mathbf{x}_i^g = (\mathbf{x}_{i+0} \;\; \mathbf{x}_{i+1} \;\; \ldots \;\; \mathbf{x}_{i+K-1})^T, \tag{3.120}$$

which is linearly mapped into a spectrum of M transform coefficients

$$\mathbf{X}_i = \left(\boldsymbol{P}_M^{(0)} \; \boldsymbol{P}_M^{(1)} \; \ldots \; \boldsymbol{P}_M^{(K-1)}\right) \mathbf{x}_i^g = \sum_{l=0}^{K-1} \boldsymbol{P}_M^{(l)} \, \mathbf{x}_{i+l}, \tag{3.121}$$

where $\boldsymbol{P}_M^{(l)}$, $l = 0, 1, \ldots, K-1$ are block square sub-matrices of order M. The inverse transform of (3.121) consists of matrix multiplication and overlap/add procedures defined by

$$\hat{\mathbf{x}}_i = [\boldsymbol{P}_M^{(0)}]^T \, \mathbf{X}_i + [\boldsymbol{P}_M^{(1)}]^T \, \mathbf{X}_{i-1} + \ldots + [\boldsymbol{P}_M^{(K-1)}]^T \, \mathbf{X}_{i-K+1}, \tag{3.122}$$

and for the whole group of K blocks

$$\hat{\mathbf{x}}_i^g = \begin{pmatrix} \mathbf{0} & \ldots & \mathbf{0} & [\boldsymbol{P}_M^{(0)}]^T & \ldots & [\boldsymbol{P}_M^{(K-2)}]^T & [\boldsymbol{P}_M^{(K-1)}]^T \\ \mathbf{0} & \ldots & [\boldsymbol{P}_M^{(0)}]^T & [\boldsymbol{P}_M^{(1)}]^T & \ldots & [\boldsymbol{P}_M^{(K-1)}]^T & \mathbf{0} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \left[\boldsymbol{P}_M^{(0)}\right]^T & \ldots & [\boldsymbol{P}_M^{(K-2)}]^T & [\boldsymbol{P}_M^{(K-1)}]^T & \ldots & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{X}_{i-K+1} \\ \mathbf{X}_{i-K+2} \\ \vdots \\ \mathbf{X}_{i+K-1} \end{pmatrix}, \tag{3.123}$$

where $\mathbf{0}$ is null matrix. The necessary and sufficient condition to be $\hat{\mathbf{x}}_i = \mathbf{x}_i$ or $\hat{\mathbf{x}}_i^g = \mathbf{x}_i^g$ is [31]

$$\sum_{m=0}^{K-1-l} \boldsymbol{P}_M^{(m)} \, [\boldsymbol{P}_M^{(m+l)}]^T = \sum_{m=0}^{K-1-l} \boldsymbol{P}_M^{(m+l)} \, [\boldsymbol{P}_M^{(m)}]^T = \delta_m \, \boldsymbol{I}_M, \qquad l = 0, 1, \ldots, K-1, \tag{3.124}$$

where δ_m is the Kronecker delta function and $\boldsymbol{I}_M$ is the identity matrix of order M. Basis vectors of the matrix $\boldsymbol{P}_{M\times KM} = \left(\boldsymbol{P}_M^{(0)} \; \boldsymbol{P}_M^{(1)} \; \ldots \; \boldsymbol{P}_M^{(K-1)}\right)$ are constructed either from the DCT-II and type-IV DCT/DST (DCT-IV/DST-IV) basis vectors for the LOT [55, 59] and for PseudoLOT [54] (when $K = 2$), or they are generated by a recursive optimization procedure in order to maximize the coding gain [63].

The lapped transforms defined by (3.121) and (3.122) correspond to an M-channel KM-tap critically sampled perfect reconstruction filter bank, whose kth channel filter response is given by

$$h_{k,r+lm} = \{\boldsymbol{P}^{(l)}\}_{k,r}, \qquad r = 0, 1, \ldots, M-1, \qquad l = 0, 1, \ldots, K-1. \tag{3.125}$$

Hence, LTs and multirate filter banks are actually the same, although they have been studied independently in the past. In fact, the LT is a filter bank in which the impulse responses of the synthesis filters are the LT basis functions, and the impulse responses of the analysis filters are the time-reversed basis functions. The equivalence between LTs and multirate filter banks is discussed in [53, 63].

Note 5: The lapped transforms defined by (3.121) and (3.122) can also be viewed as a class of lapped orthogonal transforms with extended overlap, the so-called generalized linear-phase lapped orthogonal transforms (GenLOT) [50, 51, 56, 57]. The basis vectors of matrices $\boldsymbol{P}_M^{(l)}$, $l = 0, 1, \ldots, K-1$ are constructed by a recursive procedure from the DCT-II basis vectors which have even/odd symmetry property [74]. Thus, in this formulation the orthogonal DCT-II transform is the order-1 GenLOT, hence when $K = 1$, the LOT is the order-2 GenLOT when $K = 2$, etc., for any filter length that is an integer multiple of the block size.

3.3.2 Modulated Lapped Transforms

The modulated lapped transform (MLT) [59, 63] and the extended lapped transform (ELT) [60–64] are perfect reconstruction cosine/sine-modulated filter banks developed on the concept of a modulated filter bank. Modulated filter banks are a class of QMF banks, where all impulse responses of filters $p_{k,n}$ are obtained from modulated versions of a single low-pass filter prototype, or equivalently, of a single windowing function. The perfect reconstruction is actually achieved by proper choices of phases of the cosine and sine modulation and the low-pass filter prototype.

3.3.2.1 MLT Filter Bank

The first original MLT definition was based on the LOT using the following consideration: Another way to obtain a good set of basis functions for a lapped transform is to define an appropriate low-pass filter prototype in a modulated filter bank structure. If the length of the low-pass filter prototype is chosen to be equal to $2M$, it is possible to achieve not only aliasing cancellation, but also the perfect reconstruction with identical analysis and synthesis filters [59].

With respect to (3.125), the impulse response of kth synthesis filter $p_{k,n}$ of the modulated filter bank is based on the construction [59]

$$p_{k,n} = w_n \sqrt{\frac{2}{M}} \cos\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(n-\frac{L-1}{2}\right)+\frac{\pi}{4}\right], \qquad \text{if } k \text{ is even,}$$

$$p_{k,n} = w_n \sqrt{\frac{2}{M}} \sin\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(n-\frac{L-1}{2}\right)+\frac{\pi}{4}\right], \qquad \text{if } k \text{ is odd,}$$

$$k = 0, 1, \ldots, M-1, \qquad n = 0, 1, \ldots, 2M-1, \tag{3.126}$$

where L is the length of the low-pass filter prototype $\{w_n\}$, and M is the number of sub-bands or transform coefficients. The normalization factor $\sqrt{\frac{2}{M}}$ is necessary to generate an orthogonal transform implementation [59]. In order to keep the orthogonality of matrix $\boldsymbol{P}_{M\times 2M}$ generated by (3.126), the L-length low-pass filter prototype $\{w_n\}$ must satisfy the perfect reconstruction conditions given by (3.101). The filter bank defined by (3.126) with $L = 2M$ is called the MLT since it belongs to the class of lapped transforms. For the MLT, the low-pass filter prototype $\{w_n\}$ in (3.126) is the sine windowing function given by (3.105). It is important to note that the filter impulse responses of the MLT do not have even/odd symmetry, like the LOT. With the synthesis filters of the MLT defined by (3.126), the outputs of the analysis filter bank can be written as a block transform [59]

$$c_k^{(m)} = \sum_{n=0}^{2M-1} x_{mM+n}\, p_{k,2M-1-n}, \qquad k = 0, 1, \ldots, M-1, \tag{3.127}$$

because the impulse responses of the analysis filters are equal to the time-reversed impulse responses of the synthesis filters. The term m in (3.127) denotes the data block index. Equation (3.127) is equivalent to [59]

$$c_k^{(m)} = \sqrt{\frac{2}{M}} \sum_{n=0}^{2M-1} w_n\, x_n^{(m)} \cos\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(M-n-\frac{1}{2}\right)+\frac{\pi}{4}\right], \qquad \text{if } k \text{ is even,}$$

$$c_k^{(m)} = \sqrt{\frac{2}{M}} \sum_{n=0}^{2M-1} w_n\, x_n^{(m)} \sin\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(M-n-\frac{1}{2}\right)+\frac{\pi}{4}\right], \qquad \text{if } k \text{ is even,}$$

$$k = 0, 1, \ldots, M-1. \tag{3.128}$$

Finally, after some algebraic manipulations, (3.128) can be expressed in a common block transform form as [59]

$$c_k^{(m)} = \gamma_k \sqrt{\frac{2}{M}} \sum_{n=0}^{2M-1} w_n \, x_n^{(m)} \cos\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(n+\frac{M+1}{2}\right)\right],$$
$$k = 0, 1, \ldots, M-1, \tag{3.129}$$

where

$$\gamma_k = \begin{cases} (-1)^{\frac{k+2}{2}}, & k \quad \text{is even}, \\ \\ (-1)^{\frac{k-1}{2}}, & k \quad \text{is odd}. \end{cases} \tag{3.130}$$

Equations (3.129) and (3.130) constitute the first original definition of the analysis MLT filter bank. The synthesis MLT filter bank is defined as

$$\hat{x}_n^{(m)} = \sqrt{\frac{2}{M}} \, w_n \sum_{k=0}^{M-1} \gamma_k \, c_k^{(m)} \cos\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(n+\frac{M+1}{2}\right)\right],$$
$$n = 0, 1, \ldots, 2M-1, \tag{3.131}$$

where $\{\hat{x}_n^{(m)}\}$ is the time domain aliased data sequence.

An alternative derivation of the MLT is based on the near-perfect reconstruction M-band QMF bank, or equivalently, the pseudo-QMF bank with the synthesis filters $f_{k,n}$ defined by Malvar [63]

$$f_{k,n} = w_n \cos\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(n-\frac{L-1}{2}\right)+\phi_k\right],$$
$$k = 0, 1, \ldots, M-1, \qquad n = 0, 1, \ldots, L-1, \tag{3.132}$$

where the parameters ϕ_k control the relative phases of the modulating cosines. The analysis filters $h_{k,n}$ are obtained by time-reversing the synthesis filters in the form

$$h_{k,n} = f_{k,L-1-n}. \tag{3.133}$$

Assuming that $L = KM$, the phases of modulation cosine functions leading to the perfect reconstruction are given by

$$\phi_k = \left(k+\frac{1}{2}\right)(K+1)\,\frac{\pi}{2}. \tag{3.134}$$

Denoting $p_{k,n} = f_{k,n}$ with the phases ϕ_k given by (3.134), and introducing the normalization factor $\sqrt{\frac{2}{M}}$ to normalize the basis functions, the MLT basis functions can be written in the form [63]

$$p_{k,n} = w_n \sqrt{\frac{2}{M}} \cos\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(n+\frac{M+1}{2}\right)\right],$$
$$k = 0, 1, \ldots, M-1, \qquad n = 0, 1, \ldots, 2M-1. \tag{3.135}$$

Again, the low-pass filter prototype $\{w_n\}$ in (3.135) is the sine windowing function given by (3.105). Then, the analysis MLT filter bank as a block transform is defined as

$$c_k^{(m)} = \sqrt{\frac{2}{M}} \sum_{n=0}^{2M-1} w_n \, x_n^{(m)} \cos\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(n+\frac{M+1}{2}\right)\right],$$
$$k = 0, 1, \ldots, M-1, \tag{3.136}$$

while the synthesis MLT filter bank as a block transform is defined as

$$\hat{x}_n^{(m)} = \sqrt{\frac{2}{M}} \, w_n \sum_{k=0}^{M-1} c_k^{(m)} \cos\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(n+\frac{M+1}{2}\right)\right],$$
$$n = 0, 1, \ldots, 2M-1. \tag{3.137}$$

One can see that the factor γ_k in definitions of the analysis and synthesis MLT filter banks given by (3.129) and (3.131), respectively, has been eliminated in those of (3.136) and (3.137). The MLT is similar to the LOT in the sense that the basis functions have length $L = 2M$, i.e., the overlapping factor $K = 2$, where M is the number of sub-bands. On the other hand, historically, the MLT is a special case of the PseudoQMF bank with the filter length $L = 2M$ [32, 63].

By substituting $N = 2M$ into (3.49) and comparing the MLT given by (3.136) with the oddly stacked TDAC MDCT given by (3.49) after substitution, we find that they are identical. Thus, the MLT is the oddly stacked MDCT being associated with the sine windowing function given by (3.105). Consequently, general mathematical properties in the time and frequency domains including special properties, relationships, matrix representations, and the perfect reconstruction conditions imposed on a windowing function for the oddly stacked TDAC MDCT are also valid for the MLT. On the other hand, the oddly stacked TDAC MDCT filter bank actually corresponds to a lapped transform, since all perfect reconstruction filter banks with identical analysis and synthesis filters (within time reversal) are lapped transforms (the evenly stacked TDAC filter bank is not a lapped transform) [63].

Biorthogonal versions of the MLT, the modulated lapped biorthogonal transform (MLBT) and the nonuniform modulated lapped biorthogonal transform (NMLBT) [65], are obtained simply by applying the biorthogonality conditions (3.114) for nonidentical analysis and synthesis windowing functions $\{h_n\}$ and $\{g_n\}$, respectively, with the symmetric analysis or synthesis windowing functions. Such a windowing function is defined by (3.115).

3.3.2.2 ELT Filter Bank

As we have seen, the MLT is a perfect reconstruction M-channel multirate filter bank with the analysis filters being cosine-modulated versions of a low-pass filter prototype of length $L = KM$, where $K = 2$, and with the synthesis filters being the time reverse of the analysis filters. The ELT was developed as a generalization of the MLT with longer basis functions, i.e., $L > 2M$, in such a way that the perfect reconstruction is preserved [60–63]. The basis functions of the ELT are defined by the same cosine modulation function as in the MLT, but they are arbitrarily longer defined as [61, 63]

$$p_{k,n} = w_n \sqrt{\frac{2}{M}} \cos\left[\frac{\pi}{M}\left(k + \frac{1}{2}\right)\left(n + \frac{M+1}{2}\right)\right],$$
$$k = 0, 1, \ldots, M-1, \qquad n = 0, 1, \ldots, L-1, \tag{3.138}$$

where $L = KM$, $K > 2$. It is clear that the basis functions in (3.138) correspond to the general form of (3.132) with the modulation phases given by (3.134). The perfect reconstruction requires that the phases of the modulating cosines be related by

$$\phi_{k+1} - \phi_k = (2r+1)\ \frac{\pi}{2}, \tag{3.139}$$

where r is an integer. Comparing (3.139) with (3.134) we get $K = 2r$ and $L = 2rM$, where r is the overlapping factor. Thus, the length of basis functions must be an even multiple of the number of sub-bands. The overlapping factor r actually specifies how many pairs of input data blocks are used to compute the spectral coefficients of the ELT. Similarly as for MLT, the windowing function $\{w_n\}$ in (3.138) is assumed to be symmetric, i.e., $w_{L-1-n} = w_n$. Now a problem is to find a windowing function $\{w_n\}$ with the length $L = 2rM$ that leads to the perfect reconstruction [61, 63].

3.3.2.3 General Perfect Reconstruction Conditions: Orthogonal Case

Assume that the analysis and synthesis filters have lengths equal to $L = KM$, $K = 2r$, where r is the overlapping factor. Thus, the original MLT is actually an ELT with $r = 1$.

With respect to (3.121), let us define the block matrix $\left(\boldsymbol{P}_M^{(0)}\ \boldsymbol{P}_M^{(1)}\ \ldots\ \boldsymbol{P}_M^{(K-1)}\right)$, where sub-matrices $\boldsymbol{P}_M^{(l)}$ of order M represent the ELT basis vectors given by

$$\{\boldsymbol{P}_M^{(l)}\}_{k,n} = \sqrt{\frac{2}{M}}\ \cos\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(n+lM+\frac{M+1}{2}\right)\right],$$
$$k,\ n = 0,1,\ldots,M-1, \qquad l = 0,1,\ldots,K-1. \tag{3.140}$$

Further, let the quasi-diagonal block matrix $\boldsymbol{W}_{KM} = diag\ \{\boldsymbol{W}_M^{(0)}\ \boldsymbol{W}_M^{(1)}\ \ldots\ \boldsymbol{W}_M^{(K-1)}\}$ of order KM represent the windowing function $\{w_n\}$, $n = 0,1,\ldots,KM-1$, where

$$\boldsymbol{W}_M^{(l)} = \text{diag}\ \{w_{lM}, w_{lM+1}, \ldots, w_{lM+M-1}\}. \tag{3.141}$$

Then, the ELT matrix $\boldsymbol{P}_{M\times KM}$ representing (3.138) can be written in the block matrix form as [62]

$$\boldsymbol{P}_{M\times KM} = \left(\boldsymbol{W}_M^{(0)}\boldsymbol{P}_M^{(0)}\ \ \boldsymbol{W}_M^{(1)}\boldsymbol{P}_M^{(1)}\ldots\boldsymbol{W}_M^{(K-1)}\boldsymbol{P}_M^{(K-1)}\right) \tag{3.142}$$

A fundamental property of sub-matrices $\boldsymbol{P}_M^{(l)}$, $l = 0,1,\ldots,K-1$ is [62]

$$\boldsymbol{P}_M^{(p)}\ [\boldsymbol{P}_M^{(p+2m)}]^T = (-1)^m\ [\boldsymbol{I}_M + (-1)^p\ \boldsymbol{J}_M],$$
$$\boldsymbol{P}_M^{(p)}\ [\boldsymbol{P}_M^{(p+2m+1)}]^T = \boldsymbol{0}, \tag{3.143}$$

where $\boldsymbol{I}_M$ is the identity matrix and $\boldsymbol{J}_M$ is opposite diagonal identity matrix, both of order M. $\boldsymbol{0}$ is null matrix. Equation (3.143) shows that the row vectors of $\boldsymbol{P}_M^{(p)}$ are orthogonal to all rows of $\boldsymbol{P}_M^{(p+2m+1)}$. Using (3.143), after some algebraic manipulations, we get the following relation:

$$\sum_{m=0}^{K-1-l} \boldsymbol{P}_M^{(m)}\ [\boldsymbol{P}_M^{(m+l)}]^T = \cos\left(\frac{l\pi}{2}\right) \sum_{p=0}^{K-1-l} \boldsymbol{W}_M^{(p)}\ [\boldsymbol{I}_M + (-1)^p\ \boldsymbol{J}_M]\ \boldsymbol{W}_M^{(p+l)}. \tag{3.144}$$

Since the ELT matrix $\boldsymbol{P}_{M\times KM}$ will be a lapped transform only if (3.124) is satisfied, applying (3.124) to (3.144) we get [62]

$$\sum_{p=0}^{K-2m-1} \boldsymbol{W}_M^{(p)}\ \boldsymbol{W}_M^{(p+2m)} = \delta_m\ \boldsymbol{I}_M, \qquad m = 0,1,\ldots,r-1, \tag{3.145}$$

where δ_m is the Kronecker delta function. The scalar form of (3.145) is given by Malvar [62]

$$\sum_{p=0}^{K-2m-1} w_{pM+n}\ w_{(p+2m)M+n} = \delta_m, \quad m = 0,1,\ldots,r-1, \quad n = 0,1,\ldots,\frac{M}{2}-1. \tag{3.146}$$

The set of nonlinear equations in (3.146) represents the necessary and sufficient conditions on the windowing function $\{w_n\}$ for the generation of an ELT. When $r = 1,\ K = 2$, the ELT becomes an MLT and perfect reconstruction conditions imposed on $\{w_n\}$ are identical to those of (3.101). When $r = 2,\ K = 4$ we get the following perfect reconstruction conditions:

$$\begin{aligned} w_{4M-1-n} &= w_n, \qquad n = 0, 1, \ldots, 2M-1, \\ w_n^2 + w_{M+n}^2 + w_{2M+n}^2 + w_{3M+n}^2 &= 1, \quad n = 0, 1, \ldots, \frac{M}{2} - 1, \\ w_n\, w_{2M+n} + w_{M+n}\, w_{3M+n} &= 0, \quad n = 0, 1, \ldots, \frac{M}{2} - 1. \end{aligned}$$

A smooth windowing function identical both for the analysis and synthesis ELT filter banks satisfying the above conditions is defined as [63]

$$w_n = -\frac{1}{2\sqrt{2}} + \frac{1}{2}\cos\left[\frac{\pi}{M}\left(n + \frac{1}{2}\right)\right]. \tag{3.147}$$

It is important to note that based on a linear algebraic interpretation of the modulated filter bank, the necessary and sufficient conditions for perfect reconstruction have been derived in [66] for $K > 1$ being arbitrary positive integer. It is a more general case compared to [62, 63] that constrained K to be even integer.

3.3.2.4 ELT Block Transform

The analysis ELT filter bank for the mth data block as a block transform is defined as [61, 63]

$$\begin{aligned} c_k^{(m)} &= \sqrt{\frac{2}{M}} \sum_{n=0}^{KM-1} w_n\, x_n^{(m)} \cos\left[\frac{\pi}{M}\left(k + \frac{1}{2}\right)\left(n + \frac{M+1}{2}\right)\right], \\ k &= 0, 1, \ldots, M-1, \end{aligned} \tag{3.148}$$

while the synthesis ELT filter bank as a block transform is defined as

$$\begin{aligned} \hat{x}_n^{(m)} &= \sqrt{\frac{2}{M}}\, w_n \sum_{k=0}^{M-1} c_k^{(m)} \cos\left[\frac{\pi}{M}\left(k + \frac{1}{2}\right)\left(n + \frac{M+1}{2}\right)\right], \\ n &= 0, 1, \ldots, KM-1, \end{aligned} \tag{3.149}$$

where $\{\hat{x}_n^{(m)}\}$ is the time domain aliased data sequence and $\{w_n\}$ is given by (3.147). The theory, properties, and design methods of the modulated lapped transforms can be found in [63].

3.3.2.5 General Perfect Reconstruction Conditions: Biorthogonal Case

Signal representations used in the current audio coding algorithms can be improved by the incorporation of biorthogonality into the ELT. Relaxing the requirement on analysis and synthesis windowing functions to be identical leads to the loss of orthogonality; however, the filter bank can still achieve the perfect reconstruction. The resulting filter bank is referred to as biorthogonal. Biorthogonality allows more flexibility in the design of the analysis and synthesis windowing functions by increasing the number of degrees freedom [58].

Let $\{h_n\}$ and $\{g_n\}$ be the analysis and synthesis windowing functions, respectively, and let $K = 2r$, where r is the overlapping factor. Then, general perfect reconstruction conditions imposed on $\{h_n\}$ and $\{g_n\}$ are defined as [58]

$$\begin{aligned}
&\sum_{m=0}^{K-1-2s} g_{mM+n}\, h_{(m+2s)M+n} = \delta_s, \\
&\sum_{m=0}^{K-1-2s} (-1)^m\, g_{mM+n}\, h_{(m+2s)M+(M-1-n)} = 0, \\
&\qquad s = 0, 1, \ldots, r-1, \quad n = 0, 1, \ldots, M-1,
\end{aligned} \tag{3.150}$$

where δ_s is the Kronecker delta function. When the overlapping factor $r = 1$, then $K = 2$, only adjacent data blocks overlap, and we have the MLT. For this special case the prefect reconstruction conditions in (3.150) reduce to [58]

$$\begin{aligned}
&g_n\, h_n + g_{M+n}\, h_{M+n} = 1, \\
&g_n\, h_{M+n} - g_{M+n}\, h_n = 0, \qquad n = 0, 1, \ldots, \frac{M}{2} - 1.
\end{aligned} \tag{3.151}$$

For comparison see also Eq. (3.100). Assuming the analysis windowing function $\{h_n\}$ to be symmetric, i.e., $h_n = h_{2M-1-n}$, $n = 0, 1, \ldots, M-1$, and using (3.114), the synthesis windowing function $\{g_n\}$ is obtained as

$$g_n = \frac{h_n}{h_{M+n}^2 + h_n^2}, \qquad \text{where} \quad h_{M+n}^2 + h_n^2 \neq 0, \qquad n = 0, 1, \ldots, M-1. \tag{3.152}$$

When the overlapping factor $r = 2$, then $K = 4$, and we have the ELT. The perfect reconstruction conditions imposed on $\{h_n\}$ and $\{g_n\}$ are reduced to the following set of equations [58]

$$\begin{aligned}
&g_n\, h_n + g_{M+n}\, h_{M+n} + g_{2M+n}\, h_{2M+n} + g_{3M+n}\, h_{3M+n} = 1, \quad n = 0, 1, \ldots, \frac{M}{2} - 1, \\
&g_n\, h_{3M+n} - g_{M+n}\, h_{2M+n} + g_{2M+n}\, h_{M+n} - g_{3M+n}\, h_n = 0, \quad n = 0, 1, \ldots, \frac{M}{2} - 1, \\
&g_n\, h_{2M+n} + g_{M+n}\, h_{3M+n} = 0, \quad n = 0, 1, \ldots, M-1,
\end{aligned} \tag{3.153}$$

and

$$h_n\, h_{2M+n} + h_{M+n}\, h_{3M+n} = 0, \quad n = 0, 1, \ldots, \frac{M}{2} - 1. \tag{3.154}$$

One way of designing the windowing function satisfying the above conditions is to construct first the analysis windowing function $\{h_n\}$ under the constraint in (3.154). Once $\{h_n\}$ is constructed, equations in (3.153) form a set of linear equations than can be solved to obtain the synthesis windowing function $\{g_n\}$ [58]. Alternatively, an algorithm for the design of an ELT windowing function can be found in [63].

3.4 Complex MCLT Filter Bank

The modulated complex lapped transform (MCLT) is a complex filter bank mapping overlapped data blocks of real-valued signal into blocks of complex-valued transform coefficients [71, 73]. Originally, the MCLT has been introduced into transform/sub-band coding as a complex extension of the MLT. It is a particular kind of a generalized DFT filter bank oversampled by a factor of two, whose real part corresponds to the MLT. One disadvantage of the MLT for some audio coding applications is that its transform coefficients are real-valued, and therefore they do not explicitly carry phase information, only magnitude. However, the MCLT supports both the magnitude and phase representation of the signal in frequency domain which are useful measures in many perceptual audio coders for spectral analysis (see Chap. 6). In most applications, the MCLT can be used in place of the DFT filter bank [73].

In general, the basis functions of the MCLT are defined by the cosine and sine modulation of the analysis and synthesis windowing function $\{w_n\}$ in the form [73]

$$p_{k,n} = p^{(c)}_{k,n} - i\, p^{(s)}_{k,n}, \quad i = \sqrt{-1}, \tag{3.155}$$

where

$$p^{(c)}_{k,n} = w_n \sqrt{\frac{2}{M}} \cos\left[\frac{\pi}{M}\left(k + \frac{1}{2}\right)\left(n + \frac{M+1}{2}\right)\right],$$

$$p^{(s)}_{k,n} = w_n \sqrt{\frac{2}{M}} \sin\left[\frac{\pi}{M}\left(k + \frac{1}{2}\right)\left(n + \frac{M+1}{2}\right)\right],$$

$$k = 0, 1, \ldots, M-1, \qquad n = 0, 1, \ldots, 2M-1. \tag{3.156}$$

It is clear that the cosine-modulated functions in (3.156) correspond to the MLT basis functions. There are two main interpretations of this MCLT construction.

First, the additional sine-modulated functions in (3.156) can be viewed as a 2× oversampling in the frequency domain, because for every data block of M real-valued samples, the MCLT results in M complex-valued frequency coefficients. In other words, the MCLT basis functions form an overcomplete basis [73]. The second interpretation is that the MCLT is in fact a 2× oversampled DFT filter bank (using the O^2DFT) instead the conventional DFT. Note that unlike in DFT filter banks, the DC frequency sub-band of the MCLT is complex-valued [71].

Based on (3.156), the analysis MCLT filter bank of an input audio signal $\{x_n\}$, $n = 0, 1, \ldots, 2M-1$, as a block transform is defined as [73]

$$p_k = c_k - i\, s_k, \qquad k = 0, 1, \ldots, M-1, \tag{3.157}$$

where

$$c_k = \sqrt{\frac{2}{M}} \sum_{n=0}^{2M-1} w_n\, x_n\, \cos\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(n+\frac{M+1}{2}\right)\right],$$
$$k = 0, 1, \ldots, M-1, \tag{3.158}$$

and

$$s_k = \sqrt{\frac{2}{M}} \sum_{n=0}^{2M-1} w_n\, x_n\, \sin\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(n+\frac{M+1}{2}\right)\right],$$
$$k = 0, 1, \ldots, M-1, \tag{3.159}$$

where $\{w_n\}$ is the sine windowing function given by (3.105) for $N = 2M$, and $\{c_k\}$ are frequency coefficients of the MLT. Since the MLT is equivalent to the oddly stacked MDCT filter bank associated with the sine windowing function for $N = 2M$, the sine-modulated filter bank (3.159) defines the corresponding oddly stacked MDST filter bank, and $\{s_k\}$ are its transform coefficients. Consequently, general mathematical properties in the time and frequency domains, relationships, and matrix representations for the oddly stacked TDAC MDCT/MDST are also valid for the MCLT. Thus, the MLT or oddly stacked MDCT is the real part of the MCLT, while the oddly stacked MDST is the imaginary part of the MCLT. We recall that the oddly stacked MDCT and MDST are, respectively, the real and imaginary components of the O^2DFT [see Eq. (3.63)].

On the other hand, a reconstruction formula of the synthesis MCLT filter bank is not unique. Specifically, there is a reconstruction formula from the real part only, as well as one from the imaginary part only [71, 73]. However, the best reconstruction formula is the average of those from real and imaginary parts (preferred) defined as [73]

$$y_n = \frac{1}{2}\left(\hat{x}_n^{(c)} + \hat{x}_n^{(s)}\right), \qquad n = 0, 1, \ldots, 2M-1, \tag{3.160}$$

where $\{\hat{x}_n^{(c)}\}$ and $\{\hat{x}_n^{(s)}\}$ are time domain aliased data sequences recovered by the synthesis MLT (MDCT) and MDST filter banks, respectively, defined as

$$\hat{x}_n^{(c)} = w_n \sqrt{\frac{2}{M}} \sum_{k=0}^{M-1} c_k \cos\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(n+\frac{M+1}{2}\right)\right],$$
$$n = 0, 1, \ldots, 2M-1, \tag{3.161}$$

$$\hat{x}_n^{(s)} = w_n \sqrt{\frac{2}{M}} \sum_{k=0}^{M-1} s_k \sin\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(n+\frac{M+1}{2}\right)\right],$$
$$n = 0, 1, \ldots, 2M-1. \tag{3.162}$$

Using both the synthesis MLT (MDCT) and MDST filter banks for the reconstruction removes the time domain aliasing. It is important to emphasize that if we use the reconstruction formula (3.160), then the original data sequence $\{x_n\}$ is perfectly reconstructed even without the windowing, overlap/add procedure. It means that the windowing procedure in the MCLT computation is redundant. In fact, using the matrix representation of oddly stacked MDCT and MDST filter banks it can be shown that (3.160) leads to $y_n = w_n^2\, x_n$, $n = 0, 1, \ldots, 2M-1$ (see below), i.e., there is no time domain aliasing.

Denoting the MLT (MDCT) and MDST matrices, respectively, by $\boldsymbol{C}_{M\times 2M}$ and $\boldsymbol{S}_{M\times 2M}$, their transposed versions by $\boldsymbol{C}_{2M\times M}$ and $\boldsymbol{S}_{2M\times M}$, and representing the symmetric windowing function $\{w_n\}$ by a diagonal matrix $\boldsymbol{W}_{2M}$, firstly using (3.160)–(3.162), then (3.158), (3.159) in the matrix representation, and relations (3.74), (3.75), the following relation holds in the matrix-vector form [68]:

$$\begin{aligned}
\mathbf{y}^T &= \frac{1}{2}\left([\hat{\mathbf{x}}^{(c)}]^T + [\hat{\mathbf{x}}^{(s)}]^T\right) \\
&= \frac{1}{2}\left(\boldsymbol{W}_{2M}\,\boldsymbol{C}_{2M\times M}\,\mathbf{c}^T + \boldsymbol{W}_{2M}\,\boldsymbol{S}_{2M\times M}\,\mathbf{s}^T\right) \\
&= \frac{1}{2}\,\boldsymbol{W}_{2M}\,\boldsymbol{C}_{2M\times M}\,\frac{2}{M}\,\boldsymbol{C}_{M\times 2M}\,\boldsymbol{W}_{2M}\,\mathbf{x}^T + \frac{1}{2}\,\boldsymbol{W}_{2M}\,\boldsymbol{S}_{2M\times M}\,\frac{2}{M}\,\boldsymbol{S}_{M\times 2M}\,\boldsymbol{W}_{2M}\,\mathbf{x}^T \\
&= \frac{1}{2}\,\boldsymbol{W}_{2M}\begin{pmatrix}\boldsymbol{I}_M - \boldsymbol{J}_M & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_M + \boldsymbol{J}_M\end{pmatrix}\boldsymbol{W}_{2M}\,\mathbf{x}^T \\
&\quad + \frac{1}{2}\,\boldsymbol{W}_{2M}\begin{pmatrix}\boldsymbol{I}_M + \boldsymbol{J}_M & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_M - \boldsymbol{J}_M\end{pmatrix}\boldsymbol{W}_{2M}\,\mathbf{x}^T = \boldsymbol{W}_{2M}^2\,\mathbf{x}^T,
\end{aligned} \tag{3.163}$$

where $\mathbf{0}$ is null matrix, $\boldsymbol{I}_M$ is the identity matrix, and $\boldsymbol{J}_M$ is the opposite diagonal identity matrix all of order M.

A biorthogonal version of the MCLT can be constructed by considering non-identical analysis and synthesis windowing functions $\{h_n\}$ and $\{g_n\}$, respectively. Specifically, $w_n = h_n$ in (3.158) and (3.159) and $w_n = g_n$ in (3.161) and (3.162). The perfect reconstruction property is preserved by assuming the analysis windowing

function $\{h_n\}$ to be symmetric, i.e., $h_n = h_{2M-1-n}$, $n = 0, 1, \ldots, M-1$, and the synthesis windowing function $\{g_n\}$ is obtained by using (3.152). Note that for biorthogonal MCLT (3.160) leads to $y_n = h_n g_n x_n$, $n = 0, 1, \ldots, 2M-1$.

On the other hand, the nonuniform MCLT (NMCLT) being a nonuniform oversampled filter bank has been introduced in [69, 72] as a two-stage extension of the MCLT by cascading a MCLT with shorter size MCLTs applied to high-frequency sub-bands. Each shorter size MCLT plays the role of merging sub-bands. Thus, the NMCLT allows a better combination of the frequency resolution and time localization.

Note 6: A real-valued variant of the MCLT, called the lapped directional transform (LDT), has been defined in [67, 70]. The central idea behind the LDT is to perform two MLT filter banks with different complementary basis functions, i.e., with cosine and corresponding sine basis functions defined by (3.156), and to combine the resulting spectra in the form of the sum and difference of transform coefficients. Thus, the LDT allows for detection of spatial orientation by using the corresponding sine modulation.

3.5 Summary

The original derivations, definitions, general mathematical properties in the time and frequency domains, and (block) matrix representations of the evenly and oddly stacked MDCT and MDST, MLT, ELTs, MCLT, and their biorthogonal versions have been presented. They are cosine/sine-modulated filter banks satisfying the perfect reconstruction property. Since necessary and sufficient conditions imposed on the analysis and synthesis windowing functions play an important role to obtain the perfect reconstruction property, additionally they have been derived and/or discussed in detail: Windowing procedure and perfect reconstruction (biorthogonal) conditions in the case of identical and (nonidentical) analysis and synthesis windowing functions, design of a windowing function including definitions of commonly windowing functions used in audio coding applications, adaptive switching of transform block sizes and windowing functions, and general perfect reconstruction conditions for the ELT filter bank with multiple overlapping factor both for the orthogonal and biorthogonal cases are discussed in detail.

Problems and Exercises

1. Consider the evenly stacked backward MDCT and MDST block transforms (3.7) and (3.9), respectively. Verify the local symmetries of the corresponding time domain aliased data sequences $\{\hat{x}_n^{E-\mathrm{MDCT}}\}$ and $\{\hat{x}_n^{E-\mathrm{MDST}}\}$ given by (3.14) and (3.15), respectively.
2. Verify relations (3.20) and (3.21) between the evenly stacked MDCT and MDST block transforms.

3. Verify relations (3.22) and (3.23) between the evenly stacked MDCT/MDST block transforms and the DFT.
4. Consider the matrices $\boldsymbol{C}^{E}_{\frac{N}{2}\times N}$ and $\boldsymbol{S}^{E}_{\frac{N}{2}\times N}$ representing the evenly stacked MDCT and MDST, respectively. Show for $N = 8$ that their transposed versions are pseudoinverses of their corresponding matrices, i.e., they satisfy four Moore–Penrose conditions (see Appendix A.1).
5. Verify the periodicity properties of the oddly stacked MDCT and MDST transform kernels given by (3.53) and (3.54), respectively, and symmetry properties of oddly stacked MDCT/MDST basis vectors given by (3.55).
6. Repeat Problem 5 for the evenly stacked MDCT and MDST transform kernels.
7. Verify relations (3.61) and (3.62) between the oddly stacked MDCT and MDST block transforms.
8. Implement by a computer program the relation between the N-point oddly stacked MDCT, MDST, and the $4N$-point DFT using (3.66), and verify the validity of (3.67) and (3.68).
9. Similarly, consider the matrices $\boldsymbol{C}^{O}_{\frac{N}{2}\times N}$ and $\boldsymbol{S}^{O}_{\frac{N}{2}\times N}$ representing the oddly stacked MDCT and MDST, respectively. Show for $N = 8$ that their transposed versions are pseudoinverses of their corresponding matrices, i.e., they satisfy four Moore–Penrose conditions (see Appendix A.1).
10. Consider the matrices $\boldsymbol{C}^{O}_{\frac{N}{2}\times N}$ and $\boldsymbol{S}^{O}_{\frac{N}{2}\times N}$ representing the oddly stacked MDCT and MDST, respectively, each split into two block sub-matrices of order $\frac{N}{2}$ according to (3.79). Verify Eqs. (3.84)–(3.89) for $N = 8$.
11. Repeat Problem 10 for the evenly stacked MDCT and MDST matrices $\boldsymbol{C}^{E}_{\frac{N}{2}\times N}$ and $\boldsymbol{S}^{E}_{\frac{N}{2}\times N}$.
12. The symmetric Landau nonnegative kernel function is defined by (3.108), and its discrete version by (3.109). For $N = 512$ using the transformation (3.107), generate by a computer program the LD windowing function. Then show its plot and compare it with the KBD windowing function. Additionally, investigate its properties in the frequency domain by the normalized DFT spectrum (the frequency log-magnitude response).
13. Using the matrix representation of lapped transforms given by (3.119)–(3.124), derive the lapped transform for the overlapping factor $K = 2$. Assume that the basis vectors of block square sub-matrices $\boldsymbol{P}^{(l)}_{M}$, $l = 0, 1$, are constructed from the oddly stacked MDCT (MDST) basis vectors. Verify the properties (3.124).
14. Consider (3.128) representing the analysis MLT filter bank. Derive its common block transform form given by (3.129) and (3.130).
15. Construct the biorthogonal version of analysis/synthesis MLT filter banks using a symmetric synthesis windowing function given by (3.115). Then, the analysis windowing function is given by (3.114). Implement the direct form of biorthogonal MLT by a computer program, and in particular, show plots of the analysis and synthesis windowing functions for some value of M (see also [65]).

References

1. R. Gluth, A unified approach to transform-based FIR filter banks with special regard to perfect reconstruction, in *Proceedings of the IEEE ICASSP'93*, Minneapolis, MN, vol. III, April 1993, pp. 157–160
2. R. Gluth, U. Heute, Analysis/synthesis filter banks based on generalized sinusiodal transforms with an application to speech coding, in *Proceedings of the 6th European Signal Processing Conference (EUSIPCO'92)*, vol. 1, Brussels, August 1992, pp. 215–218
3. R.A. Gopinath, Modulated filter banks and wavelets – a general unified theory, in *Proceedings of the IEEE ICASSP'96*, Atlanta, GA, May 1996, pp. 1585–1588
4. R.A. Gopinath, C.S. Burrus, Theory of modulated filter banks and modulated wavelet tight frames, in *Proceedings of the IEEE ICASSP'93*, Minneapolis, MN, vol. III, April 1993, pp. 169–172
5. R.D. Koilpillai, P.P. Vaidyanathan, New results on cosine-modulated FIR filter banks satisfying perfect reconstruction, in *Proceedings of the IEEE ICASSP'91*, Toronto, May 1991, pp. 1793–1796
6. R.D. Koilpillai, P.P. Vaidyanathan, Cosine-modulated FIR filter banks satisfying perfect reconstruction. IEEE Trans. Signal Process. **40**(4), 770–783 (1992)
7. Y.-P. Lin, P.P. Vaidyanathan, Linear phase cosine modulated maximally decimated filter banks with perfect reconstruction. IEEE Trans. Signal Process. **42**(11), 2525–2539 (1995)
8. J. Mau, Perfect reconstruction modulated filter banks, in *Proceedings of the IEEE ICASSP'92*, vol. IV, San Francisco, CA, April 1992, pp. 273–276
9. J. Mau, Perfect reconstruction modulated filter banks: fast algorithms and attractive new properties, in *Proceedings of the IEEE ICASSP'93*, Minneapolis, MN, vol. III, April 1993, pp. 225–228
10. J. Mau, Regular M-band modulated orthogonal transforms, in *Proceedings of the IEEE ICASSP'93*, vol. III, Adelaide, April 1994, pp. 125–128
11. T.Q. Nguyen, R.D. Koilpillai, The theory and design of arbitrary-length cosine-modulated filter banks and wavelets, satisfying perfect reconstruction. IEEE Trans. Signal Process. **44**(3), 473–483 (1996)
12. T.A. Ramstad, J.P. Tanem, Cosine-modulated analysis-synthesis filterbank with critical sampling and perfect reconstruction, in *Proceedings of the IEEE ICASSP'91*, Toronto, May 1991, pp. 1789–1792
13. G.D. Schuller, M.J. Smith, New framework for modulated perfect reconstruction filter banks. IEEE Trans. Signal Process. **44**(8), 1941–1954 (1996)
14. M. Vetterli, D. Le Gall, Perfect Reconstruction FIR filter banks: some properties and factorizations. IEEE Trans. Acoust. Speech Signal Process. **37**(7), 1057–1071 (1989)

Evenly Stacked MDCT/MDST Analysis and Synthesis Filter Banks

15. V. Britanak, A note on the MDCT/MDST and pseudoinverse matrix. Comput. Inform. **23**(3), 205–214 (2004)
16. V. Britanak, K.R. Rao, A unified fast MDCT/MDST computation in the evenly stacked analysis/synthesis system. Circuits, Syst. Signal Process. **21**(4), 415–426 (2002)
17. T.D. Lookabaugh, M.G. Perkins, Application of the Princen–Bradley filter bank to speech and image compression. IEEE Trans. Acoust. Speech Signal Process. **ASSP-38**(11), 1914–1926 (1990)
18. J.P. Princen, A.B. Bradley, Analysis/synthesis filter bank design based on time domain aliasing cancellation. IEEE Trans. Acoust. Speech Signal Process. **ASSP-34**(5), 1153–1161 (1986)

Oddly Stacked MDCT/MDST Analysis and Synthesis Filter Banks

19. V. Britanak, An efficient computing of oddly stacked MDCT/MDST computation via evenly stacked MDCT/MDST and vice versa. Signal Process. **85**(7), 1353–1374 (2005)
20. V. Britanak, H.J.L. Arriëns, Fast computational structures for an efficient implementation of the complete TDAC analysis/synthesis MDCT/MDST filter banks. Signal Process. **89**(7), 1379–1394 (2009)
21. V. Britanak, K.R. Rao, A new fast algorithm for the unified forward and inverse MDCT/MDST computation. Signal Process. **82**(3), 433–459 (2002)
22. S. Cramer, R. Gluth, Computationally efficient real-valued filter banks based on a modified O^2 DFT, in *Proceedings of EUSIPCO'90, Signal Processing V: Theories and Applications* (Elsevier Science Publishers B.V., Barcelona, 1990), pp. 585–588
23. A.W. Johnson, A.B. Bradley, Adaptive transform coding incorporating time domain aliasing cancellation. Speech Commun. **6**(4), 299–308 (1987)
24. J.P. Princen, A.W. Johnson, A.B. Bradley, Sub-band/transform coding using filter bank designs based on time domain aliasing cancellation, in *Proceedings of the IEEE ICASSP'87*, Dallas, TX, April 1987, pp. 2161–2164
25. K. Suresh, T.V. Sreenivas, Direct MDCT domain psychoacoustic modeling, in *Proceedings of the IEEE International Symposium on Signal Processing and Information Technology (ISSPIT'2007)*, Cairo, December 2007, pp. 742–747
26. K. Suresh, T.V. Sreenivas, Linear filtering in DCT-IV/DST-IV and MDCT/MDST domain. Signal Process. **89**(6), 1081–1089 (2009)
27. Y. Wang, M. Vilermo, Modified discrete cosine transform – its implications for audio coding and error concealment. J. Audio Eng. Soc. **51**(1/2), 52–61 (2003)
28. Y. Wang, L. Yaroslavsky, M. Vilermo, On the relationship between MDCT, SDFT and DFT, in *Proceedings of the 5th International Conference on Signal Processing (ICSP'2000)*, Beijing, August 2000, pp. 44–47
29. Y. Wang, L. Yaroslavsky, M. Vilermo, M. Väänänen, Some peculiar properties of the MDCT, in *Proceedings of the 5th International Conference on Signal Processing (ICSP'2000)*, Beijing, August 2000, pp. 61–64
30. K. Wright, Notes on Ogg Vorbis and the MDCT. Draft document on web site: `www.free-comp-shop.com/vorbis.html`, May 2003, 7 pp.
31. S. Zhang, W. Dou, H. Yang, DFT spectrum estimation from critically sampled lapped transforms. Signal Process. **91**(2), 300–310 (2011)

Windowing Procedure, Perfect Reconstruction Conditions, and Design of Windowing Function

32. M. Bosi, R.E. Golberg, *Introduction to Digital Audio Coding and Standards* (Springer Science+Business Media, New York, NY, 2003)
33. M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, M. Dietz, J. Herre, G. Davidson, Y. Oikawa, ISO/IEC MPEG-2 advanced audio coding, in *101st AES Convention*, Los Angeles, CA, November 1996. Preprint #4382. Also published in J. Audio Eng. Soc. **45**(10), 789–813 (1997)
34. B. Edler, Coding of audio signals with overlapping block transform and adaptive window functions. Frequenz **43**(9), 252–256 (1989) (in German)
35. A.J. Ferreira, Convolutional effects in transform coding with TDAC: an optimal window. IEEE Trans. Speech Audio Process. **4**(2), 104–114 (1996)

36. F.J. Harris, On the use of windows for harmonic analysis with the discrete Fourier transform. Proc. IEEE **66**(1), 51–83 (1978)
37. T. Mochizuki, Perfect reconstruction conditions for adaptive blocksize MDCT. IEICE Trans. Fundam. **E77-A**(5), 894–899 (1994)
38. T. Painter, A. Spanias, Perceptual coding of digital audio. Proc. IEEE **88**(4), 451–513 (2000)
39. J.R. Rice, *The Approximation of Functions*, vol. I (Addison-Wesley, Reading, MA, 1964), pp. 124–131
40. G. Smart, A.B. Bradley, Filter bank design based on time domain aliasing cancellation with non-identical windows, in *Proceedings of the IEEE ICASSP'94*, vol. III, Adelaide, April 1994, pp. 185–188
41. A. Spanias, T. Painter, V. Atti, *Audio Signal Processing and Coding*, chap. 6 (Wiley, Hoboken, NJ, 2007), pp. 145–210
42. G. Wolberg, *Digital Image Warping* (IEEE Computer Press, Los Alamitos, CA, 1990), pp. 137–144

Low (Reduced) Overlap Windowing Functions

43. E. Allamanche, R. Geiger, J. Herre, T. Sporer, MPEG-4 low delay audio coding based on the AAC codec, in *106th AES Convention*, Munich, May 1999. Preprint #4929
44. M. Lutzky, M. Schnell, M. Schmidt, R. Geiger, Structural analysis of low latency audio coding schemes, in *119th AES Convention*, New York, NY, October 2005. Preprint #6601
45. OPUS interactive speech and audio codec, Available on web site: `www.opus-codec.org` (2016)
46. J.-M. Valin, T.B. Terriberry, G. Maxwell, A full-band audio codec with low complexity and very low delay, in *Proceedings of the 17th European Signal Processing Conference (EUSIPCO'2009)*, Glasgow, August 2009, pp. 1254–1258
47. J.-M. Valin, T.B. Terriberry, C. Montgomery, G. Maxwell, A high-quality speech and audio codec with less than 10 ms delay. IEEE Trans. Audio Speech Lang. Process. **18**(1), 58–67 (2010)
48. J.-M. Valin, G. Maxwell, T.B. Terriberry, C. Montgomery, K. Vos, High-quality, low-delay music coding in the Opus codec, in *135th AES Convention*, New York, NY, October 2013. Preprint #8942

(Generalized) Lapped Orthogonal Transforms

49. A.N. Akansu, F.E. Wadas, On lapped orthogonal transform. IEEE Trans. Signal Process. **40**(2), 439–443 (1992)
50. R.L. de Queiroz, T.Q. Nguyen, K.R. Rao, Generalized lapped orthogonal transforms. Electron. Lett. **30**(2), 107–108 (1994)
51. R.L. de Queiroz, T.Q. Nguyen, K.R. Rao, The GenLOT: generalized linear-phase lapped orthogonal transform. IEEE Trans. Signal Process. **44**(3), 497–507 (1996)
52. H.S. Malvar, Reduction of blocking effects in image coding with a lapped orthogonal transform, in *Proceedings of the IEEE ICASSP'88*, New York, NY, April 1988, pp. 781–784
53. H.S. Malvar, The LOT: a link between block transform coding and multirate filter banks, in *Proceedings of International Symposium on Circuits and Systems*, Espoo, June 1988, pp. 835–838
54. H.S. Malvar, Pseudolapped orthogonal transforms. Electron. Lett. **25**(5), 312–314 (1989)
55. H.S. Malvar, D.H. Staelin, The LOT: transform coding without blocking effects. IEEE Trans. Audio Speech Lang. Process. **37**(4), 553–559 (1989)

56. V.K. Madisetti, D.B. Williams (eds.), *The Digital Signal Processing Handbook*, Part 38, Lapped Transforms (CRC & IEEE, Boca Raton, FL, 1998), pp. 1–7
57. K.R. Rao, P.C. Yip (eds.), Lapped transforms for image compression (Chapter 5), in *The Transform and Data Compression Handbook* (CRC, Boca Raton, FL, 2001), pp. 197–265

MLT and ELT (Biorthogonal, Nonuniform) Analysis and Synthesis Filter Banks

58. S. Cheung, J.S. Lim, Incorporation of biorthogonality into lapped transforms for audio compression, in *Proceedings of the IEEE ICASSP'95*, Detroit, MI, April 1995, pp. 3079–3082
59. H.S. Malvar, Lapped transforms for efficient transform/sub-band coding. IEEE Trans. Acoust. Speech Signal Process. **38**(6), 969–978 (1990)
60. H.S. Malvar, Modulated QMF filter banks with perfect reconstruction. Electron. Lett. **26**(13), 906–910 (1990)
61. H.S. Malvar, Extended lapped transforms: fast algorithms and applications, in *Proceedings of the IEEE ICASSP'91*, Toronto, May 1991, pp. 1797–1800
62. H.S. Malvar, Extended lapped transforms: properties, applications, and fast algorithms. IEEE Trans. Signal Process. **40**(11), 2703–2714 (1992)
63. H.S. Malvar, *Signal Processing with Lapped Transforms* (Artech House, Norwood, MA, 1992). Chapter 4: Lapped Orthogonal Transforms, pp. 143–173 and Chapter 5: Modulated Lapped Transforms, pp. 175–218
64. H.S. Malvar, Extended cosine bases and applications to audio coding. Comput. Appl. Math. **15**(2), 111–123 (1996)
65. H.S. Malvar, Biorthogonal and nonuniform lapped transforms for transform coding with reduced blocking and ringing artifacts. IEEE Trans. Signal Process. **46**(4), 1043–1053 (1998)
66. M. Padmanabhan, K. Martin, Some further results on modulated/extended lapped transforms, in *Proceedings of the IEEE ICASSP'92*, vol. IV, San Francisco, CA, April 1992, pp. 265–268

MCLT Analysis and Synthesis Filter Banks

67. T. Aach, D. Kunz, A lapped directional transform for spectral image analysis and its application to restoration and enhancement. Signal Process. **80**(11), 2347–2364 (2000)
68. V. Britanak, New recursive fast radix-2 algorithm for the modulated complex lapped transform. IEEE Trans. Signal Process. **60**(12), 6703–6708 (2012)
69. S. Cheng, Z. Xiong, Audio coding and image denoising based on the nonuniform modulated complex lapped transform. IEEE Trans. Multimedia **7**(10), 817–827 (2005)
70. D. Kunz, T. Aach, Lapped directional transform: a new transform for spectral image analysis, in *Proceedings of the IEEE ICASSP'99*, Phoenix, AR, May 1999, pp. 3433–3436
71. H. Malvar, A modulated complex lapped transform and its applications to audio processing, in *Proceedings of the IEEE ICASSP'99*, Phoenix, AR, May 1999, pp. 1421–1424
72. Z. Xiong, H.S. Malvar, A nonuniform modulated complex lapped transform. IEEE Signal Process. Lett. **8**(9), 257–260 (2001)
73. B.-J. Yoon, H.S. Malvar, Coding over-complete representations of audio using the MCLT, in *Proceedings of the IEEE Data Compression Conference*, Snowbird, UT, March 2008, pp. 152–161

Supporting Literature

74. V. Britanak, P. Yip, K.R. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations* (Academic, Elsevier Science, Amsterdam, 2007)
75. R.E. Crochiere, L.R. Rabiner, Multirate techniques in filter banks and spectrum analyzers and synthesizers (Chapter 7), in *Multirate Digital Signal Processing* (Prentice-Hall, Englewood Cliffs, NJ, 1983), pp. 289–400
76. M. Fiedler, *Special Matrices and their using in Numerical Mathematics* (SNTL, Prague, 1981) (in Czech)
77. F.R. Gantmacher, *The Theory of Matrices*, 2nd edn. (Nauka, Moscow, 1966) (in Russian). English translation: Vol. 1 and 2, Chelsea, New York, 1959
78. G.H. Golub, C.F. Van Loan, *Matrix Computations*, 3rd edn. (Johns Hopkins University Press, Baltimore, MD, 1996)
79. P.P. Vaidyanathan, *Multirate Systems and Filter Banks* (Prentice-Hall, Englewood Cliffs, NJ, 1992)

Chapter 4
Fast MDCT/MDST, MLT, ELT, and MCLT Algorithms

4.1 Introduction

The perfect reconstruction cosine/sine-modulated analysis/synthesis filter banks such as the MDCT, MLT, MDST, ELT, and MCLT are fundamental processing components for the time-to-frequency transformation of an audio data block in many international audio coding standards, proprietary audio compression algorithms, broadcasting/speech/data communication codecs, as well as open-source royalty free audio/speech codecs for high quality audio/speech compression. Since the computations of cosine/sine-modulated filter banks are the most time-consuming operations in audio coding schemes, the crucial aspect for their applicability is the existence of fast algorithms that allow their efficient software/hardware implementation compared to the direct implementation via their corresponding analytical forms. Over more than two decades a number of fast algorithms for the efficient implementation of MDCT/MDST, MLT, ELT, and MCLT have been developed. In general, they are based on the indirect computation via a discrete sinusoidal unitary transform of a reduced size, or by a recursive generating a higher order transform from two lower order transforms. Fast algorithms are generally classified as radix-2 (or 2^n-length), even-length, mixed-radix (combination of radix-2 and radix-q algorithms, where q is an odd integer) or equivalently, composite length algorithms.

Any complete analysis cosine/sine-modulated filter bank involves the windowing procedure applied to an input data block followed by its transformation to the frequency domain via corresponding forward block transform. On the other hand, any complete synthesis cosine/sine-modulated filter bank involves transformation of spectral representation back to the time domain via corresponding backward block transform followed by the windowing/overlap/add procedure to recover the original data block. In developing fast algorithms for the efficient implementation of cosine/sine-modulated analysis/synthesis filter banks, for simplicity, they are considered frequently as block transforms applied to a single data block. Principally,

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1_4

by eliminating the time shift factor from cosine/sine transform kernel, the block transform can always be converted to a discrete sinusoidal unitary transform such as the DFT, type-II discrete cosine/sine transform (DCT-II/DST-II), or type-IV discrete cosine/sine transform (DCT-IV/DST-IV). Consequently, from the algorithmic point of view the cosine/sine-modulated filter banks and the DFT, DCT-II/DST-II, and DCT-IV/DST-IV are closely interrelated.

In this chapter, fast algorithms for the efficient implementation of the forward/backward evenly stacked MDCT/MDST, oddly stacked MDCT/MDST, MLT, ELT, and MCLT block transforms are presented. The emphasis is imposed particularly on basic steps, various tricks (trigonometric and algebraic), and approaches leading to the derivation of final formulae of a fast algorithm. Therefore, the chapter has also an educational value, thus giving the reader guidelines about how to derive a fast algorithm for a specific cosine/sine-modulated filter bank.

Almost all fast evenly stacked MDCT/MDST, oddly stacked MDCT/MDST, MLT, and MCLT algorithms are classified into the following categories:

- DFT/FFT-based algorithms,
- DCT-II-based algorithms,
- DCT-IV-based algorithms,
- DCT-IV/(scaled)DCT-II-based algorithms,
- Mixed-radix algorithms,
- Algorithms based on recursive/regressive filter structures,

including the unified evenly and oddly stacked MDCT/MDST computation.

For each fast algorithm complete formulae or a sparse block matrix factorization, a corresponding generalized signal flow graph, the total computational complexity, and a possible structural simplification of the algorithm are presented. Appendices provide all the necessary supporting efficient fast computational structures including short-length modules for completing forward/backward MDCT/MDST, MLT, ELT, and MCLT efficient implementations.

4.2 Fast Algorithms for the MDCT/MDST Computation in the Evenly Stacked System

The evenly stacked MDCT/MDST [4] have been adopted in the Dolby Labs AC-2 audio compression system [9–12]. AC-2 digital audio system is also used in DigiCipher and DSC (Digital Spectrum Compatible) HDTV audio coders [84]. In the following, at first definitions of the evenly stacked MDCT and MDST in the form of block transforms are presented including a relation between the MDCT and MDST. Then fast algorithms for their efficient implementation are discussed in detail.

4.2.1 Definitions of MDCT and MDST Block Transforms

Let $\{x_n\}$, $n = 0, 1, \ldots, N-1$ represent a windowed input data sequence, and N being the length of a data block is assumed to be an even integer. With respect to Chap. 3, the forward and backward MDCT block transforms are, respectively, defined as [4]

$$c_k^E = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)k\right],$$

$$k = 0, 1, \ldots, \frac{N}{2}-1, \quad c_{\frac{N}{2}}^E = 0, \tag{4.1}$$

$$\hat{x}_n^{E-\text{MDCT}} = \frac{1}{N}\sum_{k=0}^{\frac{N}{2}-1} \epsilon_k \, c_k^E \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)k\right],$$

$$n = 0, 1, \ldots, N-1, \tag{4.2}$$

where $\epsilon_0 = 1$, and $\epsilon_k = 2$ for $k = 1, 2, \ldots, \frac{N}{2}-1$. The corresponding forward and backward MDST block transforms are, respectively, defined as [4]

$$s_k^E = \sum_{n=0}^{N-1} x_n \sin\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)k\right],$$

$$k = 1, 2, \ldots, \frac{N}{2}, \quad s_0^E = 0, \tag{4.3}$$

$$\hat{x}_n^{E-\text{MDST}} = \frac{1}{N}\sum_{k=1}^{\frac{N}{2}} \tau_k \, s_k^E \sin\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)k\right],$$

$$n = 0, 1, \ldots, N-1, \tag{4.4}$$

where $\tau_{\frac{N}{2}} = 1$, and $\tau_k = 2$ for $k = 1, 2, \ldots, \frac{N}{2}-1$. $\{c_k^E\}$ and $\{s_k^E\}$ are, respectively, MDCT and MDST coefficients, and $\{\hat{x}_n^{E-\text{MDCT}}\}$ in (4.2) and $\{\hat{x}_n^{E-\text{MDST}}\}$ in (4.4) are time domain aliased data sequences. We recall that in the evenly stacked system alternate MDCT and MDST computations are required, i.e., the MDCT of $\{x_n\}$ and the MDST of $\{x_{\frac{N}{2}+n}\}$, where $\{x_n\}$ and $\{x_{\frac{N}{2}+n}\}$ are successive adjacent overlapped data blocks. The MDCT is recognized as the transform operation when the time index of data block is even, while the MDST is recognized as the transform operation when the

time index of data block is odd. We note that the relation between the MDCT and the MDST block transforms given by Britanak and Rao [8]

$$s^{E}_{\frac{N}{2}-k} = (-1)^{\frac{N}{4}} \sum_{n=0}^{N-1} (-1)^n \, x_n \, \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)k\right], \quad k = 1, 2, \ldots, \frac{N}{2}, \tag{4.5}$$

results in a simple method to compute the MDST (but of $\{x_{\frac{N}{2}+n}\}$) using only one fast algorithm for the MDCT computation. For simplicity, in the following discussion the normalization factors $\frac{1}{N}$ in (4.2) and (4.4) are neglected.

4.2.2 DFT/FFT-Based Fast Algorithms

The first efficient algorithm for the alternate MDCT and MDST computation in the evenly stacked system is based on the complex-valued N-point DFT/FFT [13]. It has been subsequently improved replacing the complex FFT by two N-point real-valued FFTs (RVFFTs) [14]. In any case, using the symmetry property of the DFT and defining the MDCT and MDST to be real and imaginary components of a single complex FFT [73], two windowed and overlapped adjacent data blocks may be transformed simultaneously by one complex FFT in order to obtain alternating MDCT/MDST coefficients. Indeed, the forward MDCT and MDST and their backward versions can be expressed as an DFT/FFT with appropriate post- and pre-twiddle operations, respectively.

Consider the forward MDCT given by (4.1) and the forward MDST given by (4.3). Exploiting the fact that FFT operates on an array of complex components, the forward MDCT/MDST computation can be realized by packing one windowed data block in the real part of the forward complex FFT, the other in the imaginary part, and both adjacent overlapping data blocks can be transformed simultaneously as [9, 10, 13]

$$f_k = e^{-i\,(\frac{\pi}{2}+\frac{\pi}{N})k} \sum_{n=0}^{N-1} (x_n + i\, x_{\frac{N}{2}+n}) \; e^{-i\,\frac{2\pi nk}{N}}, \quad i = \sqrt{-1}, \qquad k = 0, 1, \ldots, N-1, \tag{4.6}$$

where $\{f_k\}$ are DFT coefficients. After the complex FFT computation and post-twiddle operations, the MDCT and MDST coefficients are obtained as [8]

$$\begin{aligned} c^{E}_{k} &= -\frac{1}{2}\,(\Re e\,\{f_{N-k}\} - \Re e\,\{f_k\}), \qquad c^{E}_{0} = f_0, \\ s^{E}_{k} &= \frac{1}{2}\,(\Re e\,\{f_k\} + \Re e\,\{f_{N-k}\}), \qquad s^{E}_{0} = 0, \quad k = 1, \ldots, \frac{N}{2}. \end{aligned} \tag{4.7}$$

We note that the complex multiplications by the factors $e^{i\,(\frac{\pi}{2}+\frac{\pi}{N})k}$ in (4.6) can be realized as the blocks of Givens–Jacobi rotations (see Appendix F.4).

The number of arithmetic operations can be reduced using two RVFFTs, one for the MDCT and one for the MDST computation [14]. Let $\{g_k^{(1)}\}$ be the RVFFT of windowed $\{x_n\}$, and $\{g_k^{(2)}\}$ be the RVFFT of windowed $\{x_{\frac{N}{2}+n}\}$. Then MDCT and MDST coefficients are given by Britanak and Rao [8]

$$\begin{aligned}
&\overset{E}{c}_k = g_k^{(1)} \cos\phi_k - g_{N-k}^{(1)} \sin\phi_k, \overset{E}{c}_0 = g_0^{(1)},\\
&\overset{E}{s}_k = g_k^{(2)} \sin\phi_k + g_{N-k}^{(2)} \cos\phi_k,\\
&\overset{E}{s}_0 = 0, \quad \overset{E}{s}_{\frac{N}{2}} = g_{\frac{N}{2}}^{(2)},\\
&k = 1, \ldots, \frac{N}{2} - 1,
\end{aligned} \tag{4.8}$$

where $\phi_k = (\frac{\pi}{2} + \frac{\pi}{N})k$.

Note 1 For the real-time PC-based implementation of AC-2 audio compression in [12] two methods for the alternate MDCT/MDST computation are discussed: One method based on complex $\frac{N}{2}$-point FFT, and the other based on two $\frac{N}{4}$-point FFTs. The required computational complexity is not mentioned. However, it was concluded that although both of these methods allow better than 50% savings in time, the former method is preferred in the trade-off between data manipulation and processing time reduction.

Now, consider the backward MDCT given by (4.2) and the backward MDST given by (4.4). The backward MDCT can be mapped into a complex N-point inverse DFT/FFT as [11]

$$\hat{x}_n^{E-\mathrm{MDCT}} = \Re e \left\{ \sum_{k=0}^{\frac{N}{2}-1} \left[\epsilon_k \overset{E}{c}_k e^{i\,(\frac{\pi}{2}+\frac{\pi}{N})k} \right] e^{i\,\frac{2\pi nk}{N}} \right\}, \qquad n = 0, 1, \ldots, N-1, \tag{4.9}$$

while the backward MDST can be mapped as [11]

$$\hat{x}_n^{E-\mathrm{MDST}} = \Im m \left\{ \sum_{k=1}^{\frac{N}{2}} \left[\tau_k \overset{E}{s}_k e^{i\,(\frac{\pi}{2}+\frac{\pi}{N})k} \right] e^{i\,\frac{2\pi nk}{N}} \right\}, \qquad n = 0, 1, \ldots, N-1. \tag{4.10}$$

By packing the real-valued data sequence $\{\epsilon_k \overset{E}{c}_k\}$ or $\{\tau_k \overset{E}{s}_k\}$ in the real part of an inverse FFT after multiplying by the twiddle factors $e^{i\,(\frac{\pi}{2}+\frac{\pi}{N})k}$, both they become complex-valued. Similarly, the complex multiplications by the factors $e^{i\,(\frac{\pi}{2}+\frac{\pi}{N})k}$ in (4.9) and (4.10) can be realized as the blocks of Givens–Jacobi rotations (see Appendix F.4). Equations (4.9) and (4.10) imply that the time domain aliased data sequence $\{\hat{x}_n^{E-\mathrm{MDCT}}\}$ is obtained as the real part of the N-point inverse complex

FFT, while the time domain aliased data sequence $\{\hat{x}_n^{E-\mathrm{MDST}}\}$ is obtained as the imaginary part of the N-point inverse complex FFT. Thus, for the computation of $\{\hat{x}_n^{E-\mathrm{MDCT}}\}$ and $\{\hat{x}_n^{E-\mathrm{MDST}}\}$ we need two separate N-point inverse complex FFTs [11] (see Exercise 1).

Note 2 There exists a simple method how to use a forward complex FFT for the inverse complex FFT computation [80]. Specifically, at first it is necessary to exchange the real and imaginary parts in the input complex-valued data sequence, then performing the forward complex FFT and finally, again exchanging the real and imaginary parts of the result.

4.2.3 DCT-II-Based Fast Algorithms

Although the DFT/FFT-based fast algorithms are structurally very simple, they require complex arithmetic. However, the DCT-II-based algorithms require the real arithmetic only, and they are more efficient in terms of the computational complexity [7, 8, 67]. The required DCT-II fast algorithm or fast DCT-II computational structure is presented in Appendix C.1.

4.2.3.1 DCT-II-Based Fast Algorithm [8] and Its Refined Version [7]

The evenly stacked N-point MDCT given by (4.1) can be decomposed into the form [8]

$$\begin{aligned} c_k^E &= \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi(2n+1)k}{N} + \frac{\pi}{2}k\right] \\ &= \cos\frac{\pi}{2}\,k \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi(2n+1)k}{N}\right] - \sin\frac{\pi}{2}\,k \sum_{n=0}^{N-1} x_n \sin\left[\frac{\pi(2n+1)k}{N}\right], \\ k &= 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{4.11}$$

From (4.11) it can be easily seen that the MDCT coefficients are obtained from the alternate computation of two sums. For even-indexed MDCT coefficients (k is even) the first sum is computed, while for odd-indexed MDCT coefficients (k is odd) the second sum is computed, and hence (4.11) is equivalent to

$$c_{2k}^E = (-1)^k \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi(2n+1)2k}{N}\right],$$

$$c_{2k+1}^{E} = (-1)^{k+1} \sum_{n=0}^{N-1} x_n \sin\left[\frac{\pi(2n+1)(2k+1)}{N}\right], \qquad k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.12}$$

The symmetry of cosine and sine transform kernels on the right-hand side of (4.12) provides the first reduction. Indeed, substituting $N-1-n$ for $n = 0, 1, \ldots, \frac{N}{2}-1$ into both sums of (4.12), and using the trigonometric identities, they can be written as

$$\begin{aligned}
c_{2k}^{E} &= (-1)^{k} \sum_{n=0}^{\frac{N}{2}-1} (x_n + x_{N-1-n}) \cos\left[\frac{\pi(2n+1)k}{N/2}\right] \\
&= (-1)^{k} \sum_{n=0}^{\frac{N}{2}-1} x_n'' \cos\left[\frac{\pi(2n+1)k}{N/2}\right], \\
c_{2k+1}^{E} &= (-1)^{k+1} \sum_{n=0}^{\frac{N}{2}-1} (x_n - x_{N-1-n}) \sin\left[\frac{\pi(2n+1)(2k+1)}{N}\right] \\
&= (-1)^{k+1} \sum_{n=0}^{\frac{N}{2}-1} x_n' \sin\left[\frac{\pi(2n+1)(2k+1)}{N}\right], \\
k &= 0, 1, \ldots, \frac{N}{4} - 1.
\end{aligned} \tag{4.13}$$

The last step includes again using the symmetry of cosine and sine transform kernels and by substitution $\frac{N}{2}-1-n$ for $n = 0, 1, \ldots, \frac{N}{4}-1$ in (4.13), the complete formulae constituting the fast algorithm for the alternate N-point MDCT/MDST computation in the evenly stacked system (N is integer divisible by 4) are obtained as [8]

$$\begin{aligned}
c_{2k}^{E} &= (-1)^{k} \sum_{n=0}^{\frac{N}{4}-1} (x_n'' + x_{\frac{N}{2}-1-n}'') \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right] \\
c_{2k+1}^{E} &= (-1)^{k+1} \sum_{n=0}^{\frac{N}{4}-1} (x_n' + x_{\frac{N}{2}-1-n}') \sin\left[\frac{\pi(2n+1)(2k+1)}{4(N/4)}\right], \\
k &= 0, 1, \ldots, \frac{N}{4} - 1,
\end{aligned} \tag{4.14}$$

where

$$x_n' = x_n - x_{N-1-n}, \qquad x_n'' = x_n + x_{N-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.15}$$

The cosine and sine transform kernels in (4.14) are recognized as an unnormalized $\frac{N}{4}$-point DCT-II of $\{x_n^{''} + x_{\frac{N}{2}-1-n}^{''}\}$ and an unnormalized $\frac{N}{4}$-point DST-IV of $\{x_n^{'} + x_{\frac{N}{2}-1-n}^{'}\}$, respectively. Thus, the even-indexed MDCT coefficients are converted to the DCT-II while the odd-indexed MDCT coefficients are converted to the DST-IV.

The corresponding generalized signal flow graph for the alternate forward and backward MDCT/MDST computation is shown in Fig. 4.1 for $N = 16$. Based on the relation between the MDCT and MDST block transforms given by (4.5), the symbols in brackets correspond to the MDST computation. After the MDCT computation, the MDST coefficients are in reverse order. From (4.14) it can be easily seen that the fast MDCT algorithm is generally valid for any integer N divisible by 4. Therefore, its computational complexity depends on the adopted fast DCT-II and DST-IV algorithms (see Appendices C.1 and C.2, respectively). $\frac{N}{4}$ may be even or odd, thus requiring efficient even-length or odd-length DCT-II and DST-IV algorithms.

The backward MDCT/MDST computation can be realized by reversing the generalized signal flow graph for the forward MDCT computation and performing inverse operations. However, there is a minor change. The backward MDCT given by (4.2) can be decomposed into the form

$$\hat{x}_n^{E-\text{MDCT}} = \sum_{k=0}^{\frac{N}{2}-1} \epsilon_k \, c_k^E \left(\cos\frac{\pi k}{2} \cos\left[\frac{\pi(2n+1)k}{N}\right] - \sin\frac{\pi k}{2} \sin\left[\frac{\pi(2n+1)k}{N}\right]\right),$$
$$n = 0, 1, \ldots, N-1. \tag{4.16}$$

Using the odd anti-symmetry property of MDCT coefficients (when $\frac{N}{2}$ is even, see Chap. 3) we have

$$\hat{x}_n^{E-\text{MDCT}} = c_0^E + \sum_{k=1}^{\frac{N}{2}-1} \cos\frac{\pi k}{2} \, 2 \, c_k^E \cos\left[n\frac{\pi(2n+1)k}{N}\right]$$
$$- \sum_{k=1}^{\frac{N}{2}-1} \sin\frac{\pi k}{2} \, 2 \, c_k^E \sin\left[\frac{\pi(2n+1)k}{N}\right],$$
$$n = 0, 1, \ldots, N-1, \tag{4.17}$$

which can be finally written as

$$\hat{x}_n^{E-\text{MDCT}} = c_0^E + \sum_{k=1}^{\frac{N}{4}-1} (-1)^k \, 2 \, c_{2k}^E \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right] +$$

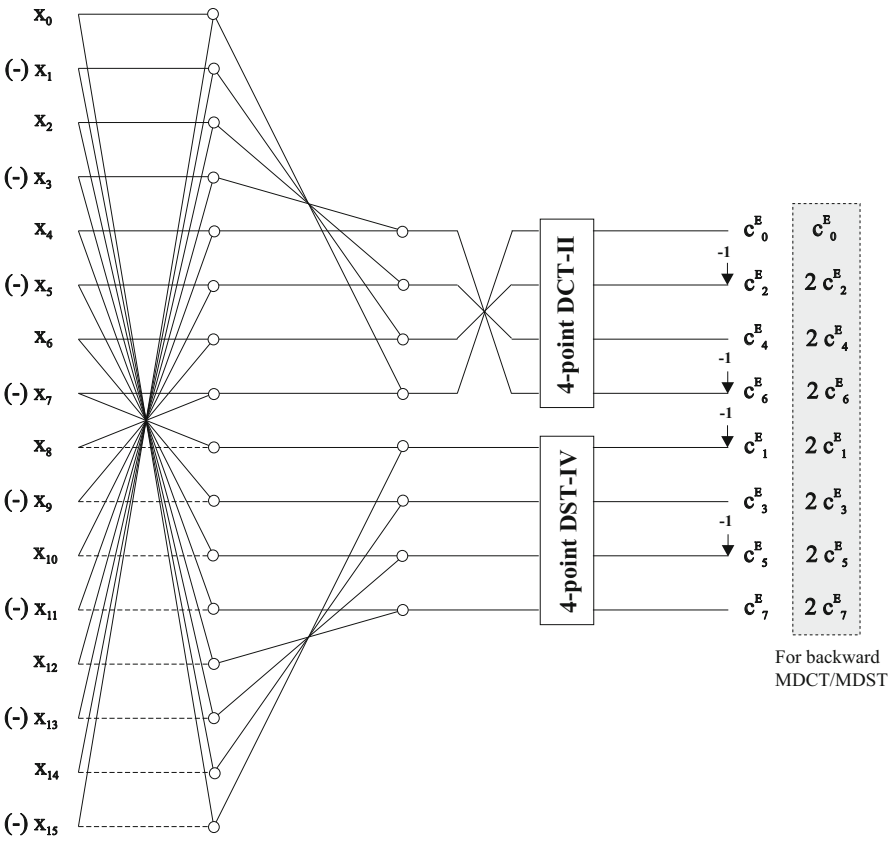

Fig. 4.1 Generalized signal flow graph for the alternate forward and backward MDCT/MDST computation in the evenly stacked system for $N = 16$

$$+ \sum_{k=0}^{\frac{N}{4}-1} (-1)^{k+1}\, 2\, c^{E}_{2k+1} \sin\left[\frac{\pi(2n+1)(2k+1)}{4(N/4)}\right],$$

$$n = 0, 1, \ldots, N-1. \tag{4.18}$$

It means that for the backward MDCT/MDST computation, the MDCT coefficients $\{c^{E}_{k}\}$, $k = 1, 2, \ldots, \frac{N}{2} - 1$, have to be scaled by the factor of 2 except for c^{E}_{0} according to (4.2). This is shown in the generalized signal flow graph in Fig. 4.1. Thus, the generalized signal flow graph represents the unified forward and backward MDCT/MDST computation in the evenly stacked system for any integer N divisible by 4.

The fast algorithm defined by (4.14) and (4.15) can be refined to involve two identical unnormalized $\frac{N}{4}$-point DCTs-II as follows. At first, substituting $\frac{N}{4}-1-k$ for k into the second sum of (4.14) we have

$$\begin{aligned} c^{E}_{\frac{N}{2}-1-2k} &= (-1)^{\frac{N}{4}-k} \sum_{n=0}^{\frac{N}{4}-1} (x'_n + x'_{\frac{N}{2}-1-n}) \sin\left[\frac{\pi(2n+1)\left(\frac{N}{2}-1-2k\right)}{4(N/4)}\right] \\ &= (-1)^{\frac{N}{4}-k} \sum_{n=0}^{\frac{N}{4}-1} (-1)^n (x'_n + x'_{\frac{N}{2}-1-n}) \cos\left[\frac{\pi(2n+1)(2k+1)}{4(N/4)}\right] \\ &= (-1)^{\frac{N}{4}-k} \sum_{n=0}^{\frac{N}{4}-1} z_n \cos\left[\frac{\pi(2n+1)(2k+1)}{4(N/4)}\right], \qquad (4.19) \\ k &= 0, 1, \ldots, \frac{N}{4}-1. \end{aligned}$$

Based on the relation between the DST-IV and the corresponding DCT-IV [75] (see also Appendix C.2), the DST-IV transform kernel is converted to that of the corresponding DCT-IV. Finally, using the trigonometric identity $\cos(\alpha+\beta) = 2\cos(\alpha)\cos(\beta) - \cos(\alpha-\beta)$, setting $\alpha = \frac{\pi}{N/2}(2n+1)k$, $\beta = \frac{\pi}{N}(2n+1)$, and exploiting the fact that the MDCT coefficients have odd anti-symmetry property, i.e., $c^{E}_{\frac{N}{2}+2k-1} = -c^{E}_{\frac{N}{2}-2k+1}$ when $\frac{N}{2}$ is even (see Chap. 3), we get the complete formulae of the refined fast algorithm for the forward N-point MDCT computation defined as [7]

$$\begin{aligned} c^{E}_{2k} &= (-1)^k \sum_{n=0}^{\frac{N}{4}-1} (x''_n + x''_{\frac{N}{2}-1-n}) \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right], \\ c^{E}_{\frac{N}{2}-1-2k} &= (-1)^{\frac{N}{4}-k} \sum_{n=0}^{\frac{N}{4}-1} \left(2\, z_n \cos\left[\frac{\pi}{N}(2n+1)\right]\right) \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right] + c^{E}_{\frac{N}{2}-2k+1}, \\ k &= 0, 1, \ldots, \frac{N}{4}-1, \qquad (4.20) \end{aligned}$$

whereby in the second sum of (4.20), the MDCT coefficients are related to the DCT-II coefficients $\{c^{\text{II}}_k\}$ by

$$\begin{aligned} c^{E}_{\frac{N}{2}-1} &= (-1)^{\frac{N}{4}}\, 2\, c^{\text{II}}_{\frac{N}{4}-1}, \qquad \text{if} \quad k=0, \\ c^{E}_{\frac{N}{2}-1-2k} &= (-1)^{\frac{N}{4}-k}\, c^{\text{II}}_{\frac{N}{4}-1-k} + c^{E}_{\frac{N}{2}-2k+1}, \qquad \text{for} \quad k>0, \qquad (4.21) \end{aligned}$$

and

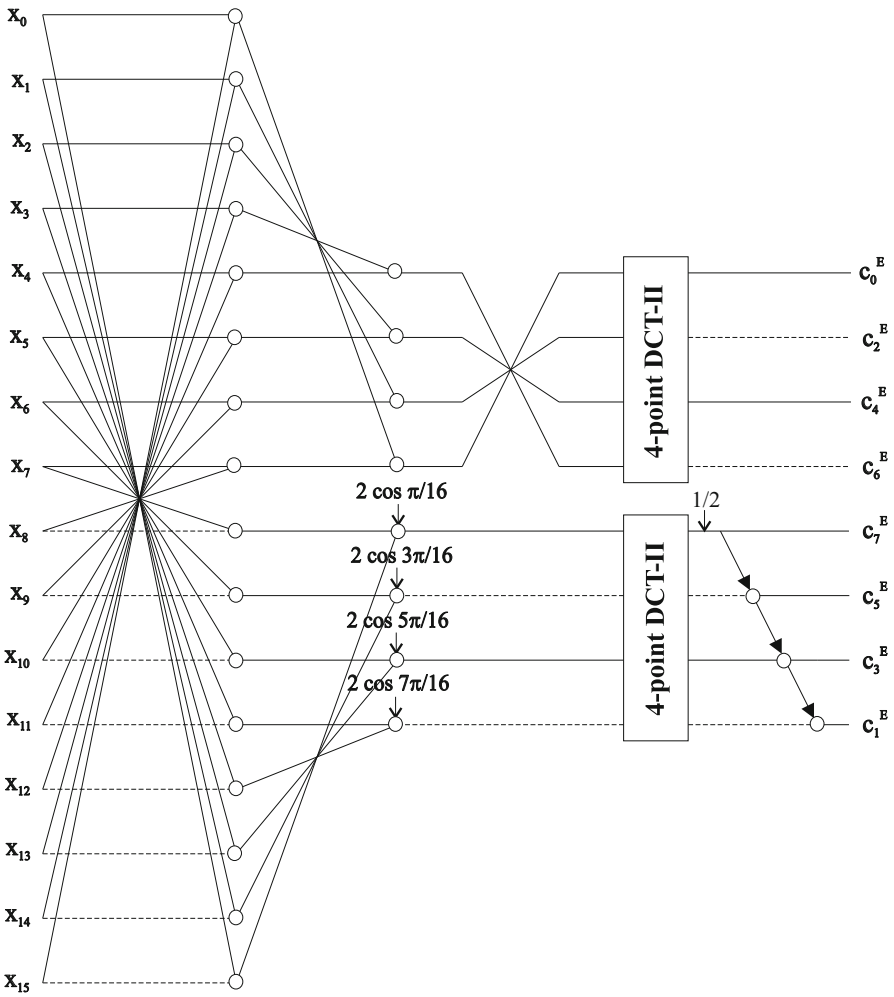

Fig. 4.2 Refined generalized signal flow graph for the alternate forward and backward MDCT/MDST computation in the evenly stacked system for $N = 16$

$$z_n = (-1)^n \, (x_n^{'} + x_{\frac{N}{2}-1-n}^{'}), \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.22}$$

The corresponding refined generalized signal flow graph for the alternate fast MDCT/MDST computation is shown in Fig. 4.2 for $N = 16$. The backward MDCT/MDST computation can be realized by reversing the generalized signal flow graph for the forward MDCT and performing inverse operations. The MDCT coefficients have to be scaled according to (4.2) (see Fig. 4.1).

The total computational complexity of the refined fast MDCT/MDST algorithm [7] in the general case, when N is integer divisible by 4 is given by $\frac{N}{4}$ multiplications, $7\frac{N}{4}-1$ additions, 1 shift plus the arithmetic complexity of two $\frac{N}{4}$-point DCT-IIs. The backward MDCT/MDST requires exactly $\frac{N}{2}$ less number of additions than that of the forward MDCT.

4.2.3.2 DCT-II-Based Fast Algorithm [7]

Substituting $N-1-n$ for n into (4.1) we have

$$c_k^E = \sum_{n=0}^{\frac{N}{2}-1} (x_n + (-1)^k\, x_{N-1-n})\, \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)k\right],$$
$$k = 0, 1, \ldots, \frac{N}{2}-1. \tag{4.23}$$

Now consider separately the even-indexed and odd-indexed MDCT coefficients in (4.23). Then, we have [7]

$$\begin{aligned} c_{2k}^E &= \sum_{n=0}^{\frac{N}{2}-1} (x_n + (-1)^{2k}\, x_{N-1-n})\, \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)2k\right] \\ &= \sum_{n=0}^{\frac{N}{2}-1} (x_n + x_{N-1-n})\, \cos\left[\frac{\pi}{N/2}\left(2n+1+\frac{N}{2}\right)k\right], \\ c_{2k+1}^E &= \sum_{n=0}^{\frac{N}{2}-1} (x_n + (-1)^{2k+1}\, x_{N-1-n})\, \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right] \\ &= \sum_{n=0}^{\frac{N}{2}-1} (x_n - x_{N-1-n})\, \cos\left[\frac{\pi}{2(N/2)}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \end{aligned} \tag{4.24}$$
$$k = 0, 1, \ldots, \frac{N}{4}-1.$$

The transform kernels for even-indexed and odd-indexed MDCT coefficients in (4.24) are recognized to be $\frac{N}{2}$-point MDCT and $\frac{N}{2}$-point oddly stacked MDCT, respectively. Again, using the trigonometric identity $\cos(\alpha+\beta) = 2\cos(\alpha)\cos(\beta) - \cos(\alpha-\beta)$, setting $\alpha = \frac{\pi}{N}(2n+1+\frac{N}{2})$, $\beta = \frac{\pi}{N/2}(2n+1+\frac{N}{2})k$, the second sum in (4.24) can be written in the form

$$c_{2k+1}^{E} = \sum_{n=0}^{\frac{N}{2}-1} \left(2\,(x_n - x_{N-1-n})\,\cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)\right]\right) \times \cos\left[\frac{\pi}{N/2}\left(2n+1+\frac{N}{2}\right)k\right] - c_{2k-1}^{E}, \tag{4.25}$$

and hence, we have the following expressions

$$c_{2k}^{E} = \sum_{n=0}^{\frac{N}{2}-1} x_n^{''}\,\cos\left[\frac{\pi}{N/2}\left(2n+1+\frac{N}{2}\right)k\right],$$

$$c_{2k+1}^{E} = \sum_{n=0}^{\frac{N}{2}-1} x_n^{'}\,\cos\left[\frac{\pi}{N/2}\left(2n+1+\frac{N}{2}\right)k\right] - c_{2k-1}^{E},$$

$$k = 0, 1, \ldots, \frac{N}{4}-1, \tag{4.26}$$

where

$$x_n^{'} = 2\,(x_n - x_{N-1-n})\,\cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)\right],$$

$$x_n^{''} = x_n + x_{N-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{2}-1. \tag{4.27}$$

Compared to (4.15), the data sequence $\{x_n^{'}\}$ is multiplied by $2\cos\left[\frac{\pi}{N}(2n+1+\frac{N}{2})\right]$. Since the transform kernels in both sums in (4.26) have the identical form, let us substitute $\frac{N}{2}-1-n$ for n, and we get

$$c_{2k}^{E} = \sum_{n=0}^{\frac{N}{4}-1} (x_n^{''} + x_{\frac{N}{2}-1-n}^{''})\,\cos\left[\frac{\pi}{N/2}\left(2n+1+\frac{N}{2}\right)k\right],$$

$$c_{2k+1}^{E} = \sum_{n=0}^{\frac{N}{4}-1} (x_n^{'} + x_{\frac{N}{2}-1-n}^{'})\,\cos\left[\frac{\pi}{N/2}\left(2n+1+\frac{N}{2}\right)k\right] - c_{2k-1}^{E},$$

$$k = 0, 1, \ldots, \frac{N}{4}-1, \tag{4.28}$$

The last step involves converting the MDCT kernel in both sums to that of the unnormalized DCT-II according to

$$\cos\left[\frac{\pi}{N/2}\left(2n+1+\frac{N}{2}\right)k\right]=(-1)^k\ \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right], \tag{4.29}$$

and the complete formulae for the fast N-point MDCT computation are given by Britanak [7]

$$\begin{aligned}
c^{E}_{2k} &= (-1)^k \sum_{n=0}^{\frac{N}{4}-1}(x''_n + x''_{\frac{N}{2}-1-n})\ \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right],\\
c^{E}_{2k+1} &= (-1)^k \sum_{n=0}^{\frac{N}{4}-1}(x'_n + x'_{\frac{N}{2}-1-n})\ \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right] - c^{E}_{2k-1},\\
k &= 0,1,\ldots,\frac{N}{4}-1,
\end{aligned} \tag{4.30}$$

whereby in the second sum of (4.30) the MDCT coefficients are related to the DCT-II coefficients $\{c^{\text{II}}_k\}$ by

$$\begin{aligned}
c^{E}_{1} &= 2\,c^{\text{II}}_{0}, \quad \text{if} \quad k=0,\\
c^{E}_{2k+1} &= (-1)^k\,c^{\text{II}}_{k} - c^{E}_{2k-1} \quad \text{for} \quad k>0.
\end{aligned} \tag{4.31}$$

The corresponding generalized signal flow graph for the forward MDCT computation is shown in Fig. 4.3 for $N = 16$. Obviously, the backward MDCT/MDST computation can be realized by reversing the generalized signal flow graph for the forward MDCT and performing inverse operations, and the MDCT coefficients have to be scaled according to (4.2) (see Fig. 4.1).

The total computational complexity of the fast MDCT algorithm [7] in the general case, when N is integer divisible by 4 is given by $\frac{N}{2}$ multiplications, $7\frac{N}{4}-1$ additions, 1 shift plus the arithmetic complexity of two $\frac{N}{4}$-point DCT-IIs. The backward MDCT/MDST requires exactly $\frac{N}{2}$ less number of additions than that of the forward MDCT.

4.2.3.3 DCT-II-Based Fast Algorithm [67]

Perhaps the most elegant fast algorithm for the MDCT and MDST computation in the evenly stacked system in terms of the computational complexity, regularity, and structural simplicity was developed in [67] as a part of the fast algorithm developed for the complex MCLT filter bank. It was shown that the evenly stacked N-point MDCT and MDST may be computed with two $\frac{N}{2}$-point DCTs-II of appropriately folded input sequences.

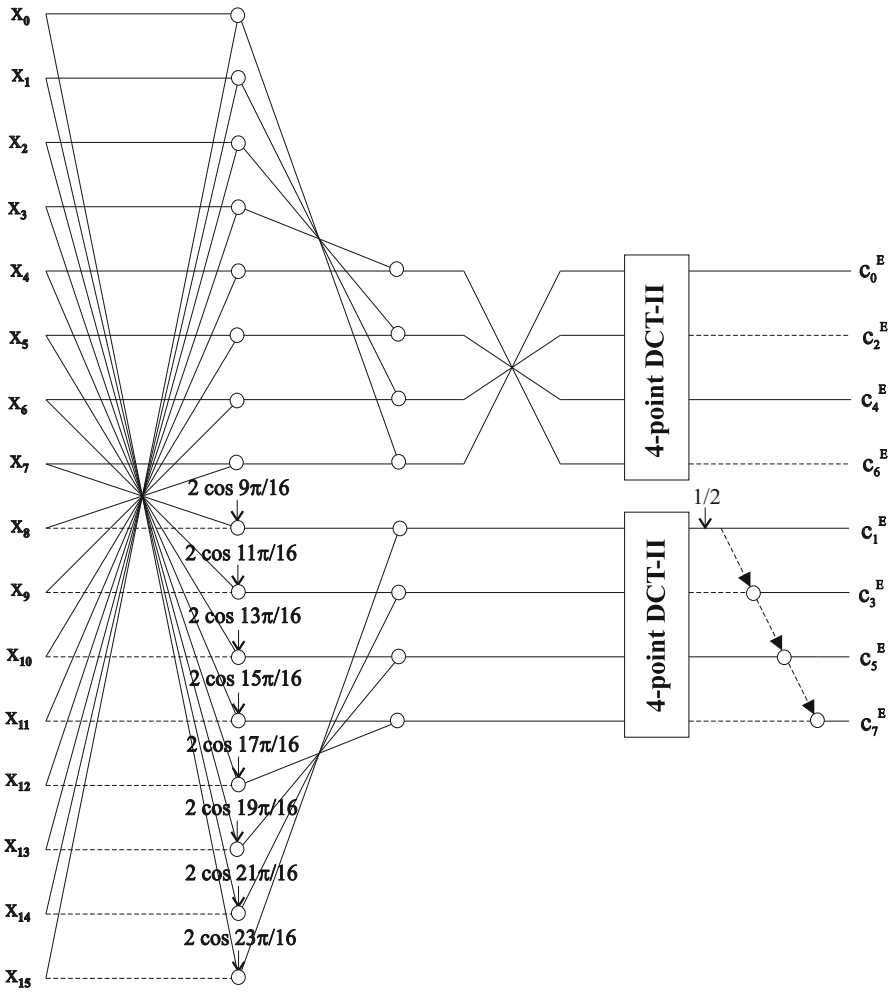

Fig. 4.3 Generalized signal flow graph for the alternate forward and backward MDCT/MDST computation in the evenly stacked system for $N = 16$

Consider the forward MDCT and MDST given by (4.1) and (4.3), respectively. In order to eliminate the time shift factors $\frac{N}{2}$ in the MDCT and MDST transform kernels, apply a permutation to the data sequence $\{x_n\}$ defined as

$$y_n = \begin{cases} x_{\frac{3N}{4}+n}, & n = 0, 1, \ldots, \frac{N}{4} - 1, \\ \\ x_{n-\frac{N}{4}}, & n = \frac{N}{4}, \frac{N}{4} + 1, \ldots, N - 1, \end{cases} \tag{4.32}$$

The permutation defined by (4.32) can be interpreted as a circular shift of the original data sequence $\{x_n\}$ to the right in the period N by $\frac{N}{4}$ samples. Subsequently, after the elimination of time shift factors $\frac{N}{2}$, using the symmetry property of cosine and sine transform kernels, the forward N-point MDCT is reduced to [67]

$$c_k^E = \sum_{n=0}^{N-1} y_n \cos\left[\frac{\pi(2n+1)k}{N}\right] = \sum_{n=0}^{\frac{N}{2}-1} (y_n + y_{N-1-n}) \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right],$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.33}$$

while the forward N-point MDST is reduced to [67]

$$s_k^E = \sum_{n=0}^{N-1} y_n \sin\left[\frac{\pi(2n+1)k}{N}\right] = \sum_{n=0}^{\frac{N}{2}-1} (y_n - y_{N-1-n}) \sin\left[\frac{\pi(2n+1)k}{2(N/2)}\right],$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.34}$$

The transform kernel in (4.33) is recognized as an unnormalized $\frac{N}{2}$-point DCT-II of $\{y_n + y_{N-1-n}\}$, and the transform kernel in (4.34) as the unnormalized corresponding $\frac{N}{2}$-point DST-II of $\{y_n - y_{N-1-n}\}$. Finally, combining (4.32) and (4.33), the fast algorithm for the forward N-point MDCT computation (N is integer divisible by 4) is defined as [67]

$$c_k^E = \sum_{n=0}^{\frac{N}{2}-1} u_n \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.35}$$

where

$$u_n = \begin{cases} x_{\frac{3N}{4}+n} + x_{\frac{3N}{4}-1-n}, & n = 0, 1, \ldots, \frac{N}{4} - 1, \\ x_{n-\frac{N}{4}} + x_{\frac{3N}{4}-1-n}, & n = \frac{N}{4}, \frac{N}{4} + 1, \ldots, \frac{N}{2} - 1. \end{cases} \tag{4.36}$$

On the other hand, combining (4.32) and (4.34) and using the relation between the DCT-II and the corresponding DST-II [75], the fast algorithm for the forward N-point MDST computation (N is integer divisible by 4) is defined as [67]

$$s_{N-k}^E = \sum_{n=0}^{\frac{N}{2}-1} (-1)^n v_n \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.37}$$

where

$$v_n = \begin{cases} x_{\frac{3N}{4}+n} - x_{\frac{3N}{4}-1-n}, & n = 0, 1, \ldots, \frac{N}{4} - 1, \\ x_{n-\frac{N}{4}} - x_{\frac{3N}{4}-1-n}, & n = \frac{N}{4}, \frac{N}{4} + 1, \ldots, \frac{N}{2} - 1. \end{cases} \tag{4.38}$$

The generalized signal flow graph for the forward MDCT computation in the evenly stacked system for $N = 16$ is shown in Fig. 4.4. The corresponding generalized signal flow graph for the forward MDST computation can be easily deduced taking into account (4.38) and sign changes applied to odd-indexed samples of $\{v_n\}$ according to (4.37). The MDST coefficients will be in reverse order.

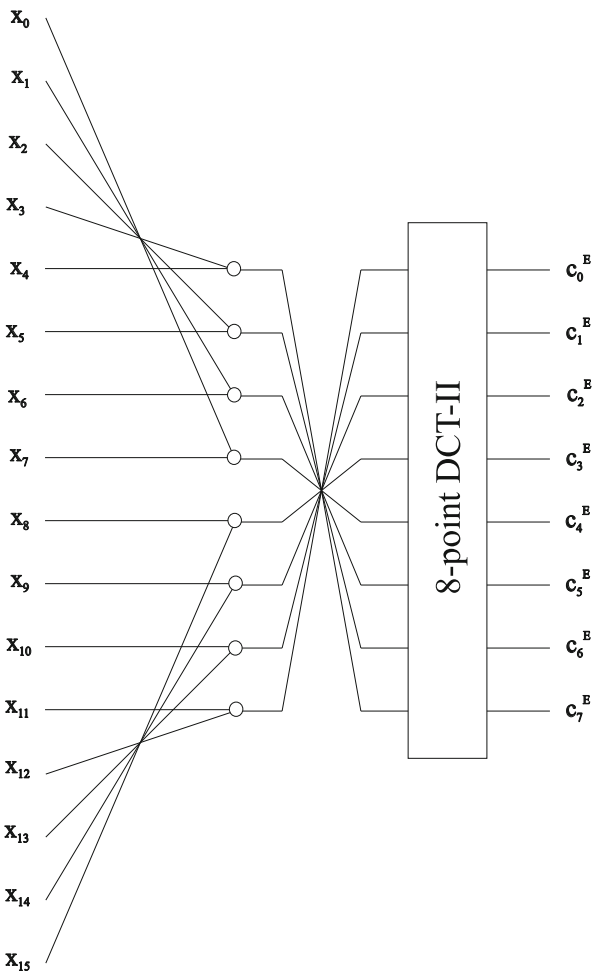

Fig. 4.4 Generalized signal flow graph for the forward and backward MDCT computation in the evenly stacked system for $N = 16$

Obviously, the backward MDCT or MDST computation can be realized by reversing the corresponding generalized signal flow graph for the forward MDCT and MDST and performing inverse operations. Before the backward MDCT and MDST computation, the MDCT coefficients have to be scaled according to (4.2), while the MDST coefficients according to (4.4). For this fast algorithm based on the relation between the MDCT and the MDST block transforms given by (4.5), the forward and backward MDST can be computed alternatively using the fast MDCT algorithm only.

The total computational complexity of the fast MDCT algorithm [67] in the general case when N is integer divisible by 4 is given by $\frac{N}{2}$ additions plus the arithmetic complexity of the $\frac{N}{2}$-point DCT-II. The backward MDCT/MDST requires exactly $\frac{N}{2}$ less number of additions than that of the forward MDCT.

Note 3 In the theory of fast algorithms for discrete unitary (orthogonal) transforms a generalized signal flow graph defines a sparse (block) matrix factorization (recursive or non-recursive) of a given transform matrix [75]. Therefore, generalized signal flow graphs for the MDCT/MDST computation in the evenly stacked system shown in Figs. 4.1, 4.2, 4.3, and 4.4 correspond to sparse matrix factorizations of the MDCT matrix [7]. Based on the relation between the MDCT and MDST block transforms given by (4.5) we can obtain sparse (block) matrix factorizations of the corresponding MDST matrix.

4.2.4 *Comparison of Evenly Stacked Fast MDCT/MDST Algorithms*

Comparison of the discussed fast algorithms for the alternate (simultaneous) MDCT/MDST computation in the evenly stacked system in terms of arithmetic complexity for $N = 2^n$, $n > 2$ is summarized in Table 4.1. For the DFT/FFT-based algorithms it is assumed that the best 2^n-point complex split-radix FFT [2, 79, 85] and RVFFT algorithms [2, 86] are employed. The computational complexity of DFT/FFT-based algorithms can be further improved by the FFT algorithm [82] being actually the modified split-radix FFT algorithm with fewer total number of

Table 4.1 Comparison of fast algorithms for the alternate (simultaneous) MDCT/MDST computation in the evenly stacked system in terms of arithmetic complexity for $N = 2^n$, $n > 2$

Fast algorithm	# of real mults	# of real adds	# of shifts
N-point complex FFT [10, 13]	$Nn + 1$	$N(3n + 1) + 1$	N
Two N-point RVFFTs [14]	$N(n - 1)$	$N(3n - 4) + 6$	
Refined DCT-II-based [7]	$\frac{N}{2}(n - 1)$	$\frac{N}{2}(3n - 1) + 2$	2
DCT-II-based [7]	$\frac{N}{2}n$	$\frac{N}{2}(3n - 1) + 2$	2
DCT-II-based [67]	$\frac{N}{2}(n - 1)$	$\frac{N}{2}(3n - 3) + 2$	2

real arithmetic operations. Specifically, for the $N = 2^n$-length FFT it reduces the computational complexity asymptotically from $4Nn + \mathcal{O}(N)$ to $\frac{34}{9} Nn + \mathcal{O}(N)$ real arithmetic operations [37].

For the DCT-II-based fast algorithms in general case, when N is a composite integer, i.e., it is of the form $N = 2^n \times q$, where q is an odd positive integer (or the mixed-radix length being the combination of radix-2 and radix-q lengths), the computational complexity of the N-point DCT-II is given by (C.8) in Appendix C.

4.3 Fast Algorithms for the MDCT (MLT)/MDST Computation in the Oddly Stacked System

Compared to the evenly stacked MDCT [4], the oddly stacked MDCT [5] or equivalently, the MLT [2], became preferring for the time-to-frequency transformation of an audio data block in many international audio coding standards as well as in many proprietary audio compression algorithms [1, 6]. In the following, at first definitions of the oddly stacked MDCT and the corresponding MDST block transforms are presented including a relation between the MDCT and MDST. Then, a relation between the MDCT and MLT, and fast algorithms for their efficient implementation are discussed in detail.

4.3.1 Definitions of MDCT (MLT) and MDST Block Transforms

Let $\{x_n\}$, $n = 0, 1, \ldots, N-1$ represent a windowed input data sequence, and N being the length of a data block is assumed to be an even integer. With respect to Chap. 3, the forward and backward MDCT block transforms are, respectively, defined as [5]

$$c_k^o = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)(2k+1)\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.39}$$

and

$$\hat{x}_n^{o-\text{MDCT}} = \frac{4}{N} \sum_{k=0}^{\frac{N}{2}-1} c_k^o \cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)(2k+1)\right], \quad n = 0, 1, \ldots, N-1. \tag{4.40}$$

The corresponding forward and backward MDST block transforms are, respectively, defined as [18]

$$s_k^O = \sum_{n=0}^{N-1} x_n \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \qquad k = 0, 1, \ldots, \frac{N}{2}-1, \tag{4.41}$$

and

$$\hat{x}_n^{O-\text{MDST}} = \frac{4}{N}\sum_{k=0}^{\frac{N}{2}-1} s_k^O \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \quad n = 0, 1, \ldots, N-1, \tag{4.42}$$

where $\{c_k^O\}/\{s_k^O\}$ are MDCT/MDST coefficients, and $\{\hat{x}_n^{O-\text{MDCT}}\}/\{\hat{x}_n^{O-\text{MDST}}\}$ are time domain aliased data sequences recovered by the backward MDCT/MDST. We note that the relation between the MDCT and the MDST block transforms given by Britanak and Rao [18]

$$s_{\frac{N}{2}-k-1}^O = (-1)^{\frac{N}{4}} \sum_{n=0}^{N-1}(-1)^n x_n \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2}-1, \tag{4.43}$$

results in a simple method to compute the MDST using only one fast algorithm for the MDCT computation.

The original definitions of the forward and backward MLT block transforms [2, 47] are simply obtained by substituting $N = 2M$ into (4.39) and (4.40). On the other hand, the MLT is always implicitly associated with the sine windowing function. Therefore, from algorithmic point of view the fast MDCT and MLT algorithms are equivalent. For simplicity, in the following discussion the normalization factors $\frac{4}{N}$ in (4.40) and (4.42) are neglected.

4.3.2 DFT/FFT-Based Fast Algorithms

Traditionally, after introducing any new sinusoidal unitary transform or any cosine-modulated filter bank to digital signal processing, in developing fast algorithms for their efficient computation the DFT and its fast implementation, FFT, frequently have been used in the first place. Indeed, similarly as for the evenly stacked MDCT, several fast DFT/FFT-based algorithms for the oddly stacked MDCT computation [15, 23–25, 28, 32, 34, 36] and the MLT computation [45, 46] have been developed. They are derived and discussed in the following subsections. The required fast DFT/FFT algorithms are presented in [2].

4.3.2.1 DFT/FFT-Based Fast Algorithm [23]

The first computationally efficient DFT/FFT-based MDCT algorithm originally proposed for the realization of real-valued perfect reconstruction single sideband analysis/synthesis filter banks [78] has been reported in [23]. Due to the same basic symmetry property of MDCT coefficients (see Chap. 3) and of the odd-time odd-frequency DFT (O^2DFT) coefficients (see Appendix B.1), the fast forward and backward MDCT computation is based on the fast O^2DFT algorithm derived for odd/even symmetric real-valued data sequences (see Appendix B.2).

The forward MDCT block transform given by (4.39) can be expressed in terms of the O^2DFT as [23]

$$c_k^{o} = 2\,\Re e\left\{ e^{-i\,\frac{\pi}{4}(2k+1)} \sum_{n=0}^{N-1} x_n\, e^{-i\,\frac{2\pi(2k+1)(2n+1)}{4N}} \right\},$$
$$i = \sqrt{-1}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.44}$$

As the first step, the original windowed data sequence $\{x_n\}$ is split into its real-valued odd and even symmetric parts, respectively, as

$$u_n = \frac{1}{2}\,(x_n - x_{N-1-n}), \qquad v_n = \frac{1}{2}\,(x_n + x_{N-1-n}), \qquad n = 0, 1, \ldots, N-1. \tag{4.45}$$

For the real-valued odd symmetric part $\{u_n\}$, the real-valued even symmetric part $\{v_n\}$ and their corresponding O^2DFT, the following symmetry relations hold [23]

$$\begin{aligned} u_n &= -u_{N-1-n} \;\Leftrightarrow\; \mathcal{U}_k = -\mathcal{U}_{N-1-k}, \\ v_n &= \;\; v_{N-1-n} \;\Leftrightarrow\; i\,\mathcal{V}_k = i\,\mathcal{V}_{N-1-k}, \qquad n,\, k = 0, 1, \ldots, N-1. \end{aligned} \tag{4.46}$$

Relations given by (4.46) imply that for $\{u_n\}$ being the real-valued odd symmetric data sequence, the transformed sequence $\{\mathcal{U}_k\}$ is purely real and odd symmetric, while for $\{v_n\}$ being the real-valued even symmetric data sequence, the transformed sequence $\{\mathcal{V}_k\}$ is purely imaginary and even symmetric [23]. Thus, the odd and even symmetric parts given by (4.46) may be transformed separately using the fast O^2DFT algorithm (see Appendix B.2), and then added afterwards [23]

$$x_n = u_n + v_n \;\Leftrightarrow\; \mathcal{U}_k + i\,\mathcal{V}_k. \tag{4.47}$$

This is already a reduction in computational complexity, because only two FFTs of the length $\frac{N}{4}$ are required compared to the direct realization of (4.44) requiring an FFT of the length N. However, exploiting the symmetry properties of O^2DFT the number of FFTs can be reduced to only one FFT of the length $\frac{N}{4}$, whereby N is an integer divisible by 4 [23].

Using (4.46) and (4.47), (4.44) may be expressed in the equivalent form as

$$c_k^o = 2\,\Re e\left\{e^{-i\,\frac{\pi}{4}(2k+1)}\,(\mathcal{U}_k - i\,\mathcal{V}_k)\right\}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.48}$$

Taking the real part of (4.48), after some algebraic manipulations we get

$$\begin{aligned} c_k^o &= \sqrt{2}\,\cos\frac{\pi}{2}\,k\,(\mathcal{U}_k - i\,\mathcal{V}_k) - \sqrt{2}\,\sin\frac{\pi}{2}\,k\,(\mathcal{U}_k + i\,\mathcal{V}_k), \\ k &= 0, 1, \ldots, \frac{N}{2} - 1, \end{aligned} \tag{4.49}$$

where the cosine and sine terms are defined on cyclic sets

$$\begin{aligned} \mathcal{C} &= \{1, 0, -1, 0\}, \qquad \cos\frac{\pi}{2}\,k \;\in\; \mathcal{C}, \quad \forall\, k \bmod 4, \\ \mathcal{S} &= \{0, 1, 0, -1\}, \qquad \sin\frac{\pi}{2}\,k \;\in\; \mathcal{S}, \quad \forall\, k \bmod 4. \end{aligned} \tag{4.50}$$

Now, split the MDCT coefficients $\{c_k^o\}$ into sequences containing the even- and odd-indexed values only, and from (4.49) we get

$$\begin{aligned} c_{2k}^o &= \;\;\sqrt{2}\,(-1)^k\,(\mathcal{U}_{2k} - i\,\mathcal{V}_{2k}), \\ c_{2k+1}^o &= -\sqrt{2}\,(-1)^k\,(\mathcal{U}_{2k+1} + i\,\mathcal{V}_{2k+1}), \\ k &= 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{4.51}$$

Using the symmetry relations given by (4.46) for values with odd indices we have

$$\mathcal{U}_{2k+1} = -\mathcal{U}_{N-2-2k} \qquad \text{and} \qquad \mathcal{V}_{2k+1} = \mathcal{V}_{N-2-2k}, \tag{4.52}$$

and the odd-indexed MDCT coefficients can be expressed as a function of the even-indexed values as

$$c_{2k+1}^o = \sqrt{2}\,(-1)^k\,(\mathcal{U}_{N-2-2k} - i\,\mathcal{V}_{N-2-2k}), \qquad k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.53}$$

Substituting $\frac{N}{4} + k$ for k into the first equation of (4.51) we get

$$c_{\frac{N}{2}+2k}^o = \sqrt{2}\,(-1)^{\frac{N}{4}+k+1}\,(\mathcal{U}_{\frac{N}{2}+2k} - i\,\mathcal{V}_{\frac{N}{2}+2k}), \qquad k = 0, 1, \ldots, \frac{N}{4} - 1, \tag{4.54}$$

and finally, using (4.51) and (4.54), after applying the fast O^2DFT algorithm (see Appendix B.2) we have

$$
\begin{aligned}
&\sqrt{2}\,(-1)^k\, c^{O}_{2k} + i\,\sqrt{2}\,(-1)^{\frac{N}{4}+k+1}\, c^{O}_{\frac{N}{2}+2k} \\
&= \Big[(\mathcal{U}_{2k} - i\,\mathcal{V}_{2k}) - i\,(\mathcal{U}_{\frac{N}{2}+2k} - i\,\mathcal{V}_{\frac{N}{2}+2k})\Big] \\
&= \Big[(\mathcal{U}_{2k} - \mathcal{V}_{\frac{N}{2}+2k}) - i\,(\mathcal{U}_{\frac{N}{2}+2k} + \mathcal{V}_{2k})\Big] \\
&= e^{-i\,\frac{\pi(8k+1)}{4N}} \sum_{n=0}^{\frac{N}{4}-1} \Big[y_n\, e^{-i\,\frac{\pi(8n+1)}{4N}}\Big]\, e^{-i\,\frac{2\pi kn}{N/4}}, \\
&k = 0, 1, \ldots, \frac{N}{4} - 1,
\end{aligned}
\tag{4.55}
$$

where

$$
y_n = \Big[(u_{2n} - v_{\frac{N}{2}+2n}) - i\,(u_{\frac{N}{2}+2n} + v_{2n})\Big], \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.56}
$$

From definitions of odd and even parts given by (4.45), the expressions for real and imaginary parts on the right-hand side of (4.56) in terms of the original data sequence $\{x_n\}$ correspond to

$$
\begin{aligned}
u_{2n} - v_{\frac{N}{2}+2n} &= \frac{1}{2}\,\Big[(x_{2n} - x_{\frac{N}{2}-1-2n}) - (x_{N-1-2n} + x_{\frac{N}{2}+2n})\Big], \\
u_{\frac{N}{2}+2n} + v_{2n} &= \frac{1}{2}\,\Big[(x_{2n} - x_{\frac{N}{2}-1-2n}) + (x_{N-1-2n} + x_{\frac{N}{2}+2n})\Big], \\
n &= 0, 1, \ldots, \frac{N}{4} - 1.
\end{aligned}
\tag{4.57}
$$

Now we are able to formulate the complete formulae of the fast DFT/FFT-based algorithm for the forward MDCT computation as follows [23]:

$$
\begin{aligned}
f_k = f_{2k} + i\, f_{\frac{N}{2}+2k} &= \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{4}-1} [(a_n - b_n) - i\,(a_n + b_n)]\; e^{-i\,\frac{\pi(4n+1)(4k+1)}{2N}} \\
&= \frac{\sqrt{2}}{2}\, e^{-i\,\frac{\pi(8k+1)}{4N}} \\
&\quad \sum_{n=0}^{\frac{N}{4}-1} \Big[[(a_n - b_n) - i\,(a_n + b_n)]\, e^{-i\,\frac{\pi(8n+1)}{4N}}\Big]\, e^{-i\,\frac{2\pi kn}{N/4}}, \\
&k = 0, 1, \ldots, \frac{N}{4} - 1,
\end{aligned}
\tag{4.58}
$$

where

$$a_n = x_{2n} - x_{\frac{N}{2}-1-2n}, \qquad b_n = x_{N-1-2n} + x_{\frac{N}{2}+2n},$$
$$n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.59}$$

The final MDCT coefficients are obtained as

$$\overset{o}{c}_{2k} = (-1)^k \, \Re e \, \{f_k\}, \qquad \overset{o}{c}_{2k+1} = (-1)^{\frac{N}{4}+k+1} \, \Im m \, \{f_{\frac{N}{4}-1-k}\},$$
$$k = 0, 1, \ldots, \frac{N}{4} - 1, \tag{4.60}$$

or alternatively, as

$$\overset{o}{c}_{2k} = (-1)^k \, \Re e \, \{f_k\}, \qquad \overset{o}{c}_{\frac{N}{2}-1-k} = (-1)^{\frac{N}{4}+k+1} \, \Im m \, \{f_k\},$$
$$k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.61}$$

The transform kernel on the right-hand side of (4.58) is the forward $\frac{N}{4}$-point complex FFT. The generalized signal flow graph of the fast DFT/FFT-based algorithm for the forward MDCT computation [23] is shown in Fig. 4.5. Complex multiplications by $e^{-i\,\frac{\pi(8n+1)}{4N}}$ and $e^{-i\,\frac{\pi(8k+1)}{4N}}$ correspond, respectively, to two identical blocks of $\frac{N}{4}$ Givens–Jacobi pre- and post-rotations (see Appendix F.4).

Scaling factors $\frac{\sqrt{2}}{2}$ in (4.58) can be incorporated into the block of Givens–Jacobi pre-rotations. Then, for the efficient implementation of pre-rotations, the bilinear computational structure has to be used (see Appendix F.3). Further, if $\frac{N}{4}$ is odd, then $(-1)^{\frac{N}{4}+k+1}$ in (4.60) and (4.61) is reduced to $(-1)^k$. For 2^n-lengths, is reduced to $(-1)^{k+1}$.

Now consider the backward MDCT block transform given by (4.40). Let $\{y_n\}$ represent an arbitrary real-valued data sequence. Then, with respect to (4.58) for the inverse O^2DFT computation the following relation holds (see Appendix B.2):

$$y_n = O^2\text{DFT}^{-1}\{f_k\} = O^2\text{DFT}\{f_k^*\}, \tag{4.62}$$

where $*$ denotes the complex conjugate. Consequently, the inverse O^2DFT computation can be realized by the forward O^2DFT algorithm. Thus, the fast DFT/FFT-based algorithm for the backward MDCT computation is defined as [23]

$$y_n = y_{2n} + i\, y_{\frac{N}{2}+2n}$$
$$= \frac{\sqrt{2}}{2} \sum_{k=0}^{\frac{N}{4}-1} \left[(-1)^k \, \overset{o}{c}_{2k} + i \, (-1)^{\frac{N}{4}+k+1} \, \overset{o}{c}_{\frac{N}{2}-1-2k} \right]^* e^{-i\,\frac{\pi(4n+1)(4k+1)}{2N}}$$

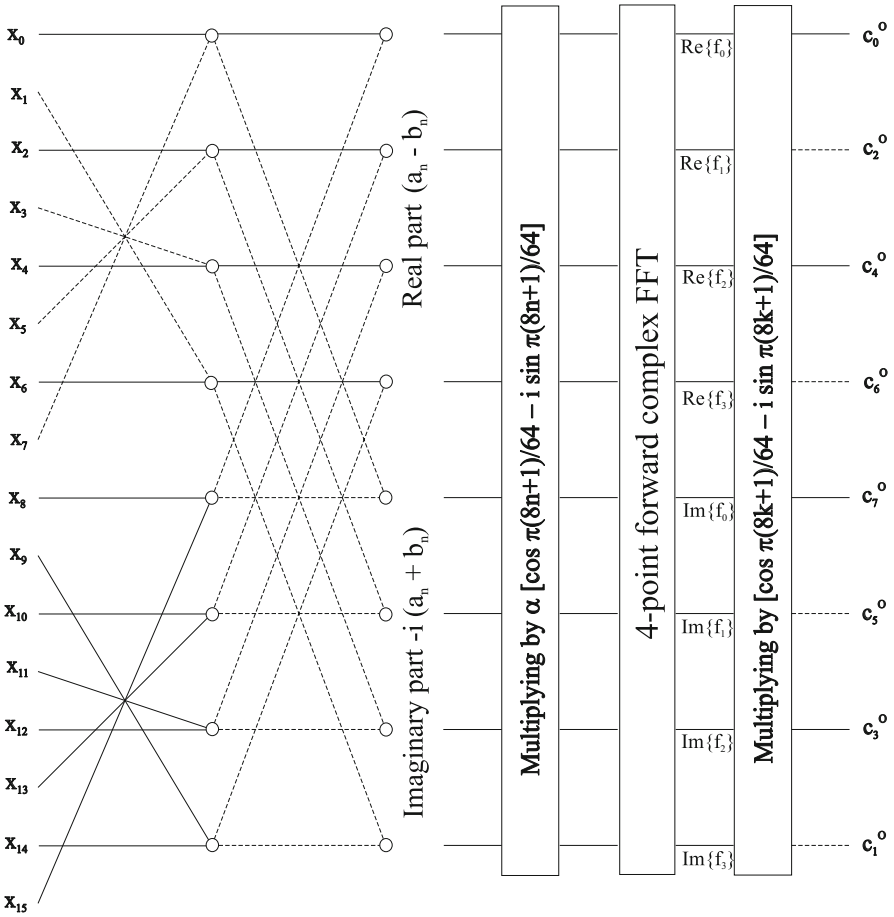

Fig. 4.5 Generalized signal flow graph for the fast DFT/FFT-based forward MDCT computation [23] in the oddly stacked system for $N = 16$, and $\alpha = \frac{\sqrt{2}}{2}$

$$= \frac{\sqrt{2}}{2}\, e^{-i\,\frac{\pi(8n+1)}{4N}} \sum_{k=0}^{\frac{N}{4}-1} \left[\left[(-1)^k\, c_{2k}^{o} - i\,(-1)^{\frac{N}{4}+k+1}\, c_{\frac{N}{2}-1-2k}^{o} \right] e^{-i\,\frac{\pi(8k+1)}{4N}} \right] e^{-i\,\frac{2\pi kn}{N/4}},$$

$$n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.63}$$

Exploiting the symmetry properties of time domain aliased data sequence $\{\hat{x}_n^{O-\mathrm{MDCT}}\}$ (see Chap. 3), it is recovered as [23]

$$\hat{x}_{2n}^{O-\text{MDCT}} = \Re e\,\{y_n\} + \Im m\,\{y_n\}, \qquad \hat{x}_{\frac{N}{2}-1-2n}^{O-\text{MDCT}} = -\hat{x}_{2n}^{O-\text{MDCT}},$$
$$\hat{x}_{\frac{N}{2}+2n}^{O-\text{MDCT}} = -\Re e\,\{y_n\} + \Im m\,\{y_n\}, \qquad \hat{x}_{N-1-2n}^{O-\text{MDCT}} = \hat{x}_{\frac{N}{2}+2n}^{O-\text{MDCT}},$$
$$n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.64}$$

The total computational complexity of the fast DFT/FFT-based algorithm for the forward N-point MDCT computation [23] is given by $\frac{3N}{2}$ real multiplications, $\frac{5N}{2}$ real additions plus the complexity of $\frac{N}{4}$-point forward complex FFT. The backward MDCT computation requires exactly $\frac{N}{2}$ real additions less than that of the forward MDCT.

4.3.2.2 DFT/FFT-Based Fast Algorithm [24, 34, 36]

Another fast DFT/FFT-based MDCT algorithm was developed in [24, 34], and its improved implementation was described in [36]. The algorithm is very similar to that discussed in previous subsection although it was derived by a quite different procedure. It is based on a $\frac{N}{4}$-point inverse complex-valued DFT/FFT.

The forward MDCT block transform given by (4.39) can be written in the equivalent as [24]

$$c_k^{O} = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}(2n+1)(2k+1) + \frac{\pi}{4}(2k+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.65}$$

The even anti-symmetry of MDCT coefficients (see Chap. 3) allows to restrict the computation to only half of coefficients. For the derivation of algorithm, the even-indexed coefficients are chosen. The odd-indexed coefficients can be deduced from the even anti-symmetry property, i.e., $c_{2k+1}^{O} = -c_{N-2-2k}^{O}$ for $k = 0, 1, \ldots \frac{N}{4} - 1$. Then (4.65) replacing c_{2k}^{O} by z_{2k} is equivalent to

$$z_{2k} = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}(2n+1)(4k+1) + \frac{\pi}{4}(4k+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.66}$$

which using the trigonometric identity for cosine transform kernel can be rewritten as

$$z_{2k} = (-1)^k \frac{\sqrt{2}}{2} \sum_{n=0}^{N-1} x_n \left(\cos\left[\frac{\pi}{2N}(2n+1)(4k+1) \right] - \sin\left[\frac{\pi}{2N}(2n+1)(4k+1) \right] \right),$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.67}$$

The symmetric cosine and sine transform kernels in (4.67) with respect to the time and frequency indices n and k, respectively, can be obtained by introducing the following permutation [24]

$$x_n^{'} = x_{2n}, \qquad x_n^{''} = x_{N-1-2n}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.68}$$

and from (4.67) we get

$$\begin{aligned} z_{2k} = (-1)^k \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \left(\cos\left[\frac{\pi}{2N}(4n+1)(4k+1) \right] \right. \\ \left. - \sin\left[\frac{\pi}{2N}(4n+1)(4k+1) \right] \right) \\ -(-1)^k \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{2}-1} x_{N-1-2n} \left(\cos\left[\frac{\pi}{2N}(4n+1)(4k+1) \right] \right. \\ \left. + \sin\left[\frac{\pi}{2N}(4n+1)(4k+1) \right] \right), \quad k = 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{4.69}$$

Now, for simplicity in the subsequent derivation of algorithm denote the cosine and sine transform kernels in (4.69) as

$$cs = \cos\left[\frac{\pi}{2N}(4n+1)(4k+1) \right], \qquad sn = \sin\left[\frac{\pi}{2N}(4n+1)(4k+1) \right].$$

Substituting $\frac{N}{4} + k$ for k into (4.69) we have

$$\begin{aligned} z_{2k} &= (-1)^k \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \, (cs - sn) - x_{N-1-2n} \, (cs + sn), \\ z_{\frac{N}{2}+2k} &= (-1)^{\frac{N}{4}+k+1} \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{2}-1} x_{N-1-2n} \, (cs - sn) + x_{2n} \, (cs + sn), \\ k &= 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{4.70}$$

Finally, substitute $\frac{N}{4}+n$ for n in (4.70). After some algebraic manipulations we get

$$z_{2k} = (-1)^k \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{4}-1} (x_{2n} - x_{\frac{N}{2}-1-2n})\,(cs - sn) - (x_{N-1-2n} + x_{\frac{N}{2}+2n})\,(cs + sn),$$

$$z_{\frac{N}{2}+2k} = (-1)^{\frac{N}{4}+k} \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{4}-1} (x_{N-1-2n} + x_{\frac{N}{2}+2n})\,(cs - sn) + (x_{2n} - x_{\frac{N}{2}-1-2n})\,(cs + sn),$$

$$k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.71}$$

The last step includes mapping (4.71) into the inverse $\frac{N}{4}$-point complex DFT/FFT (N is integer divisible by 4) as [24]

$$z_k = (-1)^k z_{2k} + i\,(-1)^{\frac{N}{4}+k} z_{\frac{N}{2}+2k} = \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{4}-1} [(a_n - b_n) + i\,(a_n + b_n)]\, e^{i\,\frac{2\pi(4n+1)(4k+1)}{4N}}$$

$$= \frac{\sqrt{2}}{2}\, e^{i\,\frac{2\pi k}{N}} \sum_{n=0}^{\frac{N}{4}-1} \left[[(a_n - b_n) + i\,(a_n + b_n)]\, e^{i\,\frac{\pi(4n+1)}{2N}} \right] e^{i\,\frac{2\pi kn}{N/4}},$$

$$k = 0, 1, \ldots, \frac{N}{4} - 1, \tag{4.72}$$

where $\{a_n\}$ and $\{b_n\}$ are given by (4.59). The final MDCT coefficients are obtained as [24]

$$c^{o}_{2k} = (-1)^k\, \Re e\, \{z_k\}, \qquad c^{o}_{\frac{N}{2}-1-2k} = (-1)^{\frac{N}{4}+k}\, \Im m\, \{z_k\},$$

$$k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.73}$$

The transform kernel on the right-hand side of (4.72) is the inverse $\frac{N}{4}$-point complex FFT. The generalized signal flow graph of the fast DFT/FFT-based algorithm for the forward MDCT computation [24] is shown in Fig. 4.6. Complex multiplications by $e^{i\,\frac{\pi(4n+1)}{2N}}$ and $e^{i\,\frac{2\pi k}{N}}$ correspond, respectively, to two blocks of $\frac{N}{4}$ Givens–Jacobi pre- and post-rotations (see Appendix F.4). Scaling factors $\frac{\sqrt{2}}{2}$ in (4.72) can be incorporated into the block of Givens–Jacobi pre-rotations. Then, for the efficient implementation of pre-rotations, the bilinear computational structure has to be used (see Appendix F.3). Further, if $\frac{N}{4}$ is even, then $(-1)^{\frac{N}{4}+k}$ in (4.72) and (4.73) is reduced to $(-1)^k$.

Now consider the backward MDCT block transform given by (4.40). Since the DFT is the unitary discrete transform, the approach in [80] allows for the forward complex FFT computation to use the inverse complex FFT. Specifically, it is necessary at first to exchange the real and imaginary parts in the input complex-valued data sequence, then performing the inverse complex FFT and finally, again

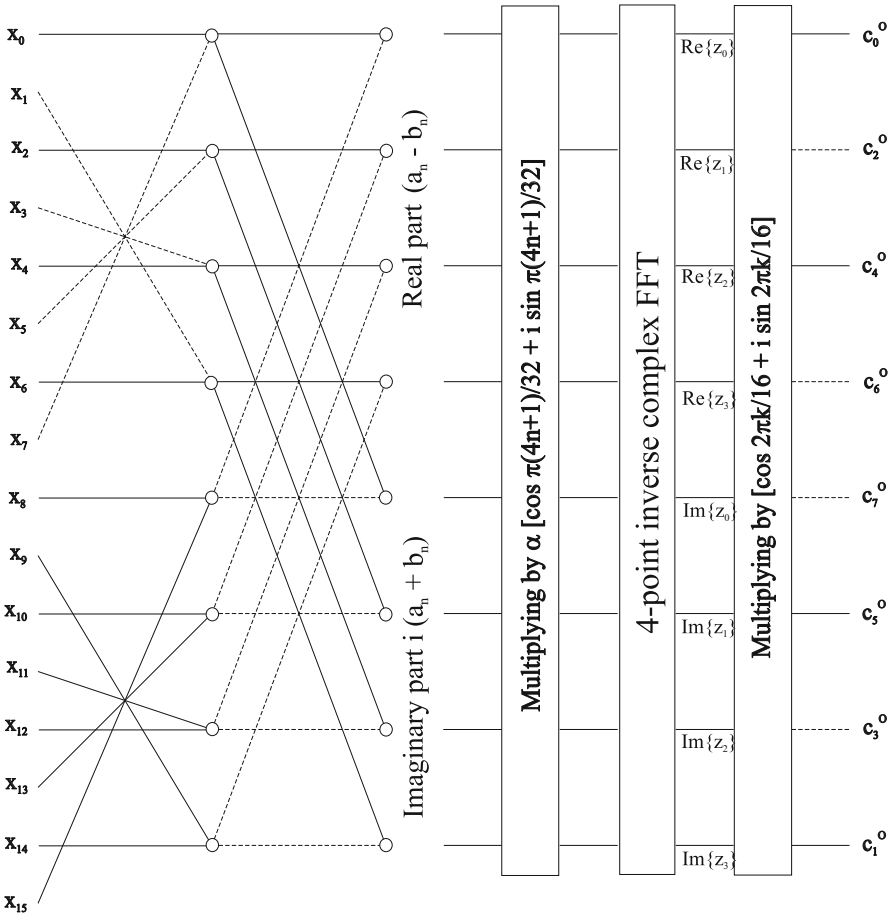

Fig. 4.6 Generalized signal flow graph for the fast DFT/FFT-based forward MDCT computation [24] in the oddly stacked system for $N = 16$, and $\alpha = \frac{\sqrt{2}}{2}$

exchanging the real and imaginary parts of the result. Let $\{y_n\}$ represent an arbitrary real-valued data sequence. Thus, the fast DFT/FFT-based algorithm for the backward MDCT computation is defined as [24]

$$y_n = y_{2n} + i\, y_{\frac{N}{2}+2n} = \frac{\sqrt{2}}{2} \sum_{k=0}^{\frac{N}{4}-1} \left[(-1)^{\frac{N}{4}+k}\, c^{o}_{\frac{N}{2}-1-2k} + i\,(-1)^{k}\, c^{o}_{2k} \right] e^{\,i\,\frac{2\pi(4n+1)(4k+1)}{4N}}$$

$$= \frac{\sqrt{2}}{2}\, e^{\,i\,\frac{2\pi n}{N}} \sum_{k=0}^{\frac{N}{4}-1} \left[[(-1)^{\frac{N}{4}+k}\, c^{o}_{\frac{N}{2}-1-2k} + i\,(-1)^{k}\, c^{o}_{2k}]\, e^{\,i\,\frac{\pi(4k+1)}{2N}} \right] e^{\,i\,\frac{2\pi kn}{N/4}},$$

$$n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.74}$$

Exploiting the symmetry properties of time domain aliased data sequence $\{\hat{x}_n^{O-\mathrm{MDCT}}\}$ (see Chap. 3), it is recovered as [24]

$$\hat{x}_{2n}^{O-\mathrm{MDCT}} = \Re e\,\{y_n\} + \Im m\,\{y_n\}, \quad \hat{x}_{\frac{N}{2}-1-2n}^{O-\mathrm{MDCT}} = -\hat{x}_{2n}^{O-\mathrm{MDCT}},$$
$$\hat{x}_{\frac{N}{2}+2n}^{O-\mathrm{MDCT}} = \Re e\,\{y_n\} - \Im m\,\{y_n\}, \quad \hat{x}_{N-1-2n}^{O-\mathrm{MDCT}} = \hat{x}_{\frac{N}{2}+2n}^{O-\mathrm{MDCT}}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.75}$$

The total computational complexity of the fast DFT/FFT-based algorithm for the forward N-point MDCT computation [24] is given by $\frac{3N}{2}$ real multiplications, $\frac{5N}{2}$ real additions plus the complexity of $\frac{N}{4}$-point forward complex FFT. The backward MDCT computation requires exactly $\frac{N}{2}$ real additions less than that of the forward MDCT.

Note 4 The windowing procedure by the sine windowing function can directly be incorporated into the fast algorithm without increasing the computational complexity [24]. An improved implementation of the windowing procedure by the sine windowing function both for the analysis and synthesis MLT (MDCT) filter bank is discussed in [36].

4.3.2.3 DFT/FFT-Based Fast Algorithm [25, 32]

The main idea of a fast DFT/FFT-based MDCT algorithm outlined in [32] is based on the simple fact that the forward MDCT block transform given by (4.39) can be written in the following equivalent form:

$$c_k^o = \Re e\left\{\sum_{n=0}^{N-1} x_{n-\frac{N}{4}}\, e^{-i\,\frac{2\pi(2n+1)(2k+1)}{4N}}\right\} = \sum_{n=0}^{N-1} x_{n-\frac{N}{4}}\, \cos\left[\frac{\pi}{2N}(2n+1)(2k+1)\right],$$
$$i = \sqrt{-1}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.76}$$

The transform kernels on the right-hand side of (4.76) are recognized, respectively, as the forward O^2DFT and the real-valued polyphase filter bank derived from the DCT-IV kernel [25]. The term $x_{n-\frac{N}{4}}$ is interpreted as a shifting of the original data sequence $\{x_n\}$ by $\frac{N}{4}$ samples with respect to the cosine transform kernel. In fact, exploiting the anti-periodicity property of the cosine transform kernel with the period N, i.e., substituting $N + n$ for n we obtain

$$\cos\left[\frac{\pi}{2N}(2n+1+2N)(2k+1)\right] = -\cos\left[\frac{\pi}{2N}(2n+1)(2k+1)\right].$$

It can be shown that the original data sequence $\{x_n\}$ has to be circularly shifted to the right in the period N by $\frac{N}{4}$ samples followed by sign changes of $\frac{N}{4}$ circularly shifted samples. This operation actually corresponds to a permutation applied to the original data sequence $\{x_n\}$ defined as [28]

$$y_n = \begin{cases} -x_{\frac{3N}{4}+n}, & n = 0, 1, \ldots, \frac{N}{4} - 1, \\ \\ x_{n-\frac{N}{4}}, & n = \frac{N}{4}, \frac{N}{4} + 1, \ldots, N - 1. \end{cases} \tag{4.77}$$

Then, applying the permutation (4.77) and using the symmetry property of cosine transform kernel, (4.76) can be rewritten as

$$\begin{aligned} c_k^o &= \sum_{n=0}^{N-1} y_n \cos\left[\frac{\pi}{2N}(2n+1)(2k+1)\right] \\ &= \sum_{n=0}^{\frac{N}{2}-1} (y_n - y_{N-1-n}) \cos\left[\frac{\pi}{4(N/2)}(2n+1)(2k+1)\right], \\ k &= 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{4.78}$$

The transform kernel on the right-hand side of (4.78) is recognized as an unnormalized $\frac{N}{2}$-point DCT-IV of $\{y_n - y_{N-1-n}\}$. We note that the permutation given by (4.77) was introduced first time in [28], but the $\frac{N}{2}$-point DCT-IV was mapped into an $\frac{N}{2}$-point complex DFT. Finally, combining (4.77) and (4.78) we get

$$c_k^o = \sum_{n=0}^{\frac{N}{2}-1} y_n \cos\left[\frac{\pi}{4(N/2)}(2n+1)(2k+1)\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.79}$$

where

$$y_n = \begin{cases} -x_{\frac{3N}{4}+n} - x_{\frac{3N}{4}-1-n}, & n = 0, 1, \ldots, \frac{N}{4} - 1, \\ \\ x_{n-\frac{N}{4}} - x_{\frac{3N}{4}-1-n}, & n = \frac{N}{4}, \frac{N}{4} + 1, \ldots, \frac{N}{2} - 1. \end{cases} \tag{4.80}$$

Since the $\frac{N}{2}$-point DCT-IV given by (4.79) is closely related to the real part of forward O^2DFT for real-valued odd symmetric data sequences [25], the $\frac{N}{2}$-point DCT-IV of $\{y_n\}$ can be efficiently computed via the fast O^2DFT algorithm (see Appendix C.2.1). The complete formulae for the DFT/FFT-based forward MDCT computation (N is integer divisible by 4) are defined as [25, 32]

$$f_k = f_{2k} + i\ f_{\frac{N}{2}+2k} = \sum_{n=0}^{\frac{N}{4}-1} (y_{2n} + i\, y_{\frac{N}{2}-1-2n})\, e^{-i\,\frac{\pi(4n+1)(4k+1)}{2N}}$$

$$= e^{-i\,\frac{2\pi k}{N}} \sum_{n=0}^{\frac{N}{4}-1} \left[(y_{2n} + i\, y_{\frac{N}{2}-1-2n})\, e^{-i\,\frac{\pi(4n+1)}{2N}} \right] e^{-i\,\frac{2\pi kn}{N/4}}, \quad k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.81}$$

The final MDCT coefficients are obtained as

$$c^{O}_{2k} = \Re e\ \{f_k\}, \qquad c^{O}_{\frac{N}{2}-1-2k} = -\Im m\ \{f_k\}, \quad k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.82}$$

The transform kernel on the right-hand side of (4.81) is the forward $\frac{N}{4}$-point complex FFT. The generalized signal flow graph of the fast DFT/FFT-based algorithm for the forward MDCT computation [25, 32] is shown in Fig. 4.7. Complex multiplications by $e^{-i\,\frac{\pi(4n+1)}{2N}}$ and $e^{-i\,\frac{2\pi k}{N}}$ correspond, respectively, to two blocks of $\frac{N}{4}$ Givens–Jacobi pre- and post-rotations (see Appendix F.4). Note that compared to the fast algorithms described in previous subsections, in the fast DFT/FFT-based algorithm given by (4.80), (4.81), and (4.82), scaling factors $\frac{\sqrt{2}}{2}$ as well as sign changing factors are completely eliminated.

Now consider the backward MDCT block transform given by (4.40). With respect to Appendix C.2.1 the fast DFT/FFT-based algorithm for the backward MDCT computation is defined as [25, 32]

$$y_n = y_{2n} + i\ y_{\frac{N}{2}+2n} = \sum_{k=0}^{\frac{N}{4}-1} (c^{O}_{2k} + i\, c^{O}_{\frac{N}{2}-1-2k})\, e^{-i\,\frac{\pi(4n+1)(4k+1)}{2N}}$$

$$= e^{-i\,\frac{2\pi n}{N}} \sum_{k=0}^{\frac{N}{4}-1} \left[(c^{O}_{2k} + i\, c^{O}_{\frac{N}{2}-1-2k})\, e^{-i\,\frac{\pi(4k+1)}{2N}} \right] e^{-i\,\frac{2\pi kn}{N/4}}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.83}$$

The time domain aliased data sequence $\{\hat{x}_n^{O-\mathrm{MDCT}}\}$ is partially recovered as

$$\hat{x}^{O-\mathrm{MDCT}}_{\frac{N}{4}+2n} = \Im m\ \{y_n\}, \quad \hat{x}^{O-\mathrm{MDCT}}_{\frac{3N}{4}-1-2n} = -\Re e\ \{y_n\}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{4.84}$$

and remaining samples of $\{\hat{x}_n^{O-\mathrm{MDCT}}\}$ are obtained as

$$\hat{x}^{O-\mathrm{MDCT}}_{n} = -\hat{x}^{O-\mathrm{MDCT}}_{\frac{N}{2}-1-n}, \quad \hat{x}^{O-\mathrm{MDCT}}_{N-1-n} = \hat{x}^{O-\mathrm{MDCT}}_{\frac{N}{2}+n}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.85}$$

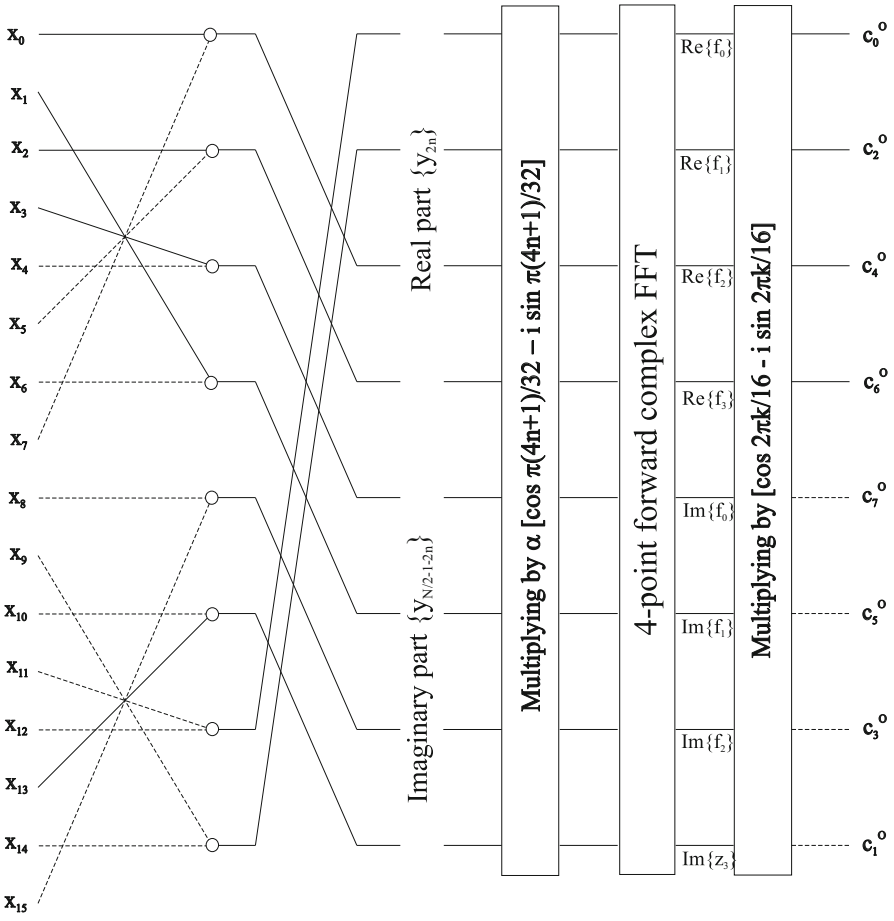

Fig. 4.7 Generalized signal flow graph for the fast DFT/FFT-based forward MDCT computation [25, 32] in the oddly stacked system for $N = 16$

The total computational complexity of the fast DFT/FFT-based algorithm for the forward N-point MDCT computation [25, 32] is given by $3(\frac{N}{2} - 1)$ real multiplications, $2N - 3$ real additions plus the complexity of $\frac{N}{4}$-point forward complex FFT. The backward MDCT computation requires exactly $\frac{N}{2}$ real additions less than that of the forward MDCT.

Note 5 Later, quite similar fast DFT/FFT-based algorithms for the MDCT computation were presented in [15, 16], and for the MLT computation in [45, 46]. In order to reduce further the computational complexity, in [15] the $\frac{N}{4}$-point complex FFT is alternatively decomposed into two $\frac{N}{8}$-point complex FFTs.

Note 6 The computational complexity of DFT/FFT-based algorithms can be further improved by the modified split-radix FFT algorithm [82] with fewer total number of real arithmetic operations. Specifically, for the $N = 2^n$-length FFT it reduces the computational complexity asymptotically from $4Nn + \mathcal{O}(N)$ to $\frac{34}{9} Nn + \mathcal{O}(N)$ real arithmetic operations [37].

4.3.3 DCT-II-Based Fast Algorithms

The fast DFT/FFT-based algorithms for the oddly stacked MDCT computation are structurally simple, but they require complex arithmetic. The fast MDCT algorithm reported in [18] and its improved version [22, 29, 35] are based on the DCT-II of reduced sizes, and use real arithmetic only. The required fast DCT-II algorithms/computational structures are presented in Appendix C.1.

4.3.3.1 DCT-II-Based Fast Algorithm [18]

Consider the forward MDCT block transform given by (4.39). The derivation of fast DCT-II-based algorithm [18] begins with Eq. (4.67), but further development is quite different compared to that of the fast DFT/FFT-based algorithm [24]. The symmetry of cosine and sine transform kernels in (4.67) provides the first reduction. Indeed, substituting $N-1-n$ for n into (4.67) and using the trigonometric identities it can be written in the form [18]

$$\begin{aligned} z_{2k} &= (-1)^k \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{2}-1} x_n^{'} \cos\left[\frac{\pi}{2N}(2n+1)(4k+1)\right] \\ &\quad -x_n^{''} \sin\left[\frac{\pi}{2N}(2n+1)(4k+1)\right], \\ k &= 0, 1, \ldots, \frac{N}{2} - 1, \end{aligned} \tag{4.86}$$

where

$$x_n^{'} = x_n - x_{N-1-n} \qquad x_n^{''} = x_n + x_{N-1-n}, \quad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.87}$$

Repeatedly using the symmetry of cosine and sine transform kernels and by substituting $\frac{N}{2} - 1 - n$ for n into (4.86), the second reduction is achieved as

$$z_{2k} = (-1)^k \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{4}-1} (x_n^{'} - x_{\frac{N}{2}-1-n}^{''}) \cos\left[\frac{\pi}{2N}(2n+1)(4k+1)\right]$$
$$- (x_n^{''} - x_{\frac{N}{2}-1-n}^{'}) \sin\left[\frac{\pi}{2N}(2n+1)(4k+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.88}$$

The last step includes expanding cosine and sine transform kernels as follows

$$\cos\left[\frac{\pi}{2N}(2n+1)(4k+1)\right] = \cos\frac{\pi(2n+1)}{2N}\cos\frac{\pi(2n+1)k}{N/2}$$
$$- \sin\frac{\pi(2n+1)}{2N}\sin\frac{\pi(2n+1)k}{N/2},$$
$$\sin\left[\frac{\pi}{2N}(2n+1)(4k+1)\right] = \sin\frac{\pi(2n+1)}{2N}\cos\frac{\pi(2n+1)k}{N/2}$$
$$+ \cos\frac{\pi(2n+1)}{2N}\sin\frac{\pi(2n+1)k}{N/2}, \tag{4.89}$$

and their substitution into (4.88). After some algebraic manipulations we get

$$z_{2k} = (-1)^k \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{4}-1} a_n \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right] - b_n \sin\left[\frac{\pi(2n+1)k}{2(N/4)}\right], \tag{4.90}$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1,$$

where data sequences $\{a_n\}$ and $\{b_n\}$ are defined as

$$a_n = (x_n^{'} - x_{\frac{N}{2}-1-n}^{''})\cos\frac{\pi(2n+1)}{2N} - (x_n^{''} - x_{\frac{N}{2}-1-n}^{'})\sin\frac{\pi(2n+1)}{2N},$$
$$b_n = (x_n^{'} - x_{\frac{N}{2}-1-n}^{''})\sin\frac{\pi(2n+1)}{2N} + (x_n^{''} - x_{\frac{N}{2}-1-n}^{'})\cos\frac{\pi(2n+1)}{2N},$$
$$n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.91}$$

The cosine and sine transform kernels in (4.90) are recognized as the unnormalized $\frac{N}{4}$-point DCT and DST of type II (DCT-II and DST-II) [75], respectively. Equation (4.91) defines the block of Givens–Jacobi rotations (see Appendix F.1). Finally, substituting $\frac{N}{4} + k$ for k into (4.90) and using the trigonometric identities

$$\cos\left[\frac{\pi}{2N}(2n+1)(4k+1+N)\right] = (-1)^{n+1}\sin\left[\frac{\pi}{2N}(2n+1)(4k+1)\right],$$
$$\sin\left[\frac{\pi}{2N}(2n+1)(4k+1+N)\right] = (-1)^{n}\quad\cos\left[\frac{\pi}{2N}(2n+1)(4k+1)\right],$$

after some algebraic manipulations the complete formulae of the fast DCT-II-based algorithm for the forward MDCT computation are obtained as (N is an integer divisible by 4) [18]

$$z_{2k} = (-1)^k\frac{\sqrt{2}}{2}\sum_{n=0}^{\frac{N}{4}-1}\left(a_n\cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right] - b_n\sin\left[\frac{\pi(2n+1)k}{2(N/4)}\right]\right),$$

$$z_{\frac{N}{2}+2k} = (-1)^{\frac{N}{4}+k}\frac{\sqrt{2}}{2}\sum_{n=0}^{\frac{N}{4}-1}(-1)^{n+1}\left(a_n\sin\left[\frac{\pi(2n+1)k}{2(N/4)}\right]\right.$$
$$\left.+b_n\cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right]\right),$$
$$k = 0, 1, \ldots, \frac{N}{4}-1. \tag{4.92}$$

The final MDCT coefficients using their even anti-symmetry property, i.e., $c^{o}_{2k+1} = -c^{o}_{N-2-2k}$, $k = 0, 1, \ldots \frac{N}{4}-1$, are given by

$$c^{o}_{2k} = z_{2k}$$
$$c^{o}_{2k+1} = -z_{N-2-2k}, \qquad k = 0, 1, \ldots, \frac{N}{4}-1. \tag{4.93}$$

From (4.92) it can be easily seen that the coefficients z_0 and $z_{\frac{N}{2}}$ for $k = 0$ are sums of $\{a_n\}$ and $\{b_n\}$, respectively. The fast DCT-II-based algorithm defined by (4.87), (4.91), (4.92), and (4.93) may be further refined as follows. Scaling factors $\frac{\sqrt{2}}{2}$ in (4.92) can be incorporated into the block of $\frac{N}{4}$ Givens–Jacobi rotations (4.91). Then, for the efficient implementation of Givens–Jacobi rotations the bilinear computational structure has to be used (see Appendix F.3). If $\frac{N}{4}$ is even, then $(-1)^{\frac{N}{4}+k}$ in (4.92) is reduced to $(-1)^k$. Further, since there exists a relation between the DCT-II and the corresponding DST-II [75], the $\frac{N}{4}$-point DST-II in (4.92) may be converted to the $\frac{N}{4}$-point DCT-II with proper preceding sign changes applied to odd-indexed samples. Then DST-II coefficients will be in reverse order. Thus, two identical $\frac{N}{4}$-point DCTs-II are used in the resulting fast computational structure. Based on the relation between the MDST and the MDCT given by (4.43), the MDST computation can be realized by the fast MDCT algorithm.

The generalized signal flow graph of the refined fast DCT-II-based algorithm for the unified forward and backward MDCT/MDST computation [18] is shown

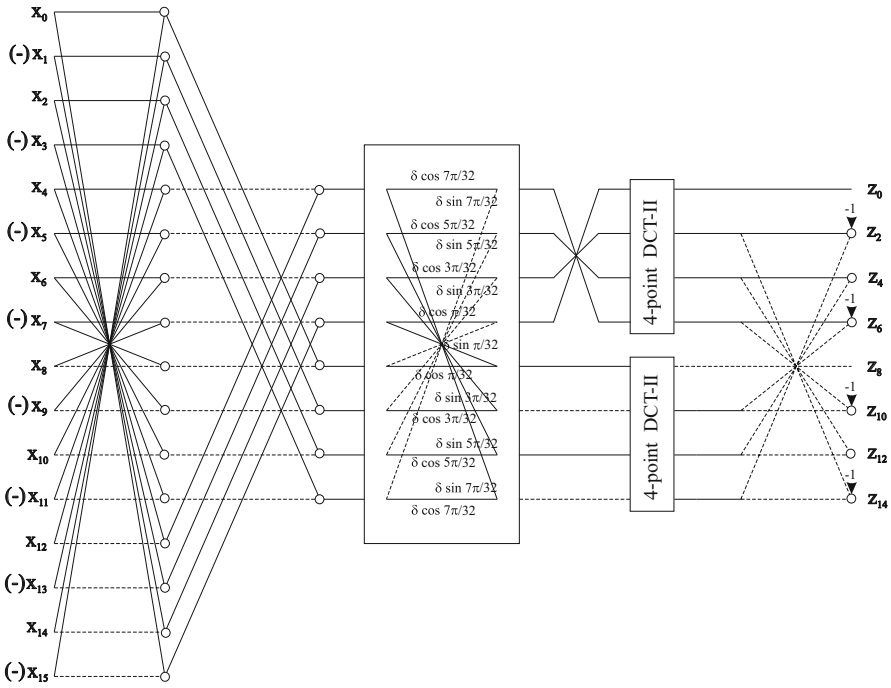

Fig. 4.8 Generalized signal flow graph of the refined fast DCT-II-based algorithm for the unified MDCT/MDST computation [18] in the oddly stacked system for $N = 16$, $\delta = \frac{\sqrt{2}}{2}$

in Fig. 4.8. The computation of backward MDCT/MDST is simply realized by reversing the signal flow graph for the forward MDCT computation and performing inverse operations. The symbols in round brackets in Fig. 4.8 correspond to the MDST computation. After the MDCT computation, the MDST coefficients are in reverse order.

The total computational complexity of the fast DCT-II-based algorithm for the forward N-point MDCT computation [18] is given by $\frac{3N}{4}$ multiplications, $\frac{11N}{4} - 2$ additions plus the complexity of two $\frac{N}{4}$-point DCTs-II. The backward MDCT computation requires exactly $\frac{N}{2}$ additions less than that of the forward MDCT.

The original fast DCT-II-based MDCT algorithm [18] was improved both in terms of the computational complexity and structural simplicity in [22, 29, 35].

4.3.3.2 Improved DCT-II-Based Fast Algorithm [22, 29, 35]

Recall that the even anti-symmetry of MDCT coefficients allows for computation of only the half number of coefficients. Therefore, the even-indexed coefficients are chosen. The odd-indexed coefficients can easily be deduced from the even anti-

symmetry property, i.e., $c^{o}_{2k+1} = -c^{o}_{N-2-2k}$ for $k = 0, 1, \ldots \frac{N}{4} - 1$. Then, the forward MDCT block transform given by (4.39) is equivalent to [35]

$$z_{2k} = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)(4k+1)\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.94}$$

In order to eliminate the time shift factor $\frac{N}{2}$ from the cosine transform kernel in (4.94), applying the permutation (4.77) we have

$$z_{2k} = \sum_{n=0}^{N-1} y_n \cos\left[\frac{\pi}{2N}(2n+1)(4k+1)\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.95}$$

Subsequently substituting $N - 1 - n$ for n into (4.95) and using the symmetry of cosine transform kernel we obtain

$$z_{2k} = \sum_{n=0}^{\frac{N}{2}-1} (y_n - y_{N-1-n}) \cos\left[\frac{\pi}{2N}(2n+1)(4k+1)\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.96}$$

Combining (4.77) and (4.96) we get

$$z_{2k} = \sum_{n=0}^{\frac{N}{2}-1} y_n \cos\left[\frac{\pi}{2N}(2n+1)(4k+1)\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.97}$$

where $\{y_n\}$ is given by (4.80). Further, substituting $\frac{N}{2} - 1 - n$ for n into (4.97), and again using the symmetry of cosine and sine transform kernels, the next reduction is achieved as

$$z_{2k} = \sum_{n=0}^{\frac{N}{4}-1} y_n \cos\left[\frac{\pi}{2N}(2n+1)(4k+1)\right] + y_{\frac{N}{2}-1-n} \sin\left[\frac{\pi}{2N}(2n+1)(4k+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.98}$$

The last step includes expanding cosine and sine transform kernels according to (4.89) followed by substituting $\frac{N}{2} + k$ for k, and the complete formulae of the improved fast DCT-II-based algorithm are defined as [22, 29, 35]

$$z_{2k} = \sum_{n=0}^{\frac{N}{4}-1} a_n \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right] + b_n \sin\left[\frac{\pi(2n+1)k}{2(N/4)}\right], \quad k = 0, \ldots, \frac{N}{4} - 1,$$

$$z_{N-2k} = \sum_{n=0}^{\frac{N}{4}-1} -a_n \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right] + b_n \sin\left[\frac{\pi(2n+1)k}{2(N/4)}\right], \quad k = 1, 2, \ldots, \frac{N}{4}, \tag{4.99}$$

where

$$\begin{aligned} a_n &= \quad y_n \cos\frac{\pi(2n+1)}{2N} + y_{\frac{N}{2}-1-n} \sin\frac{\pi(2n+1)}{2N}, \\ b_n &= -y_n \sin\frac{\pi(2n+1)}{2N} + y_{\frac{N}{2}-1-n} \cos\frac{\pi(2n+1)}{2N}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{4.100}$$

The data sequence $\{y_n\}$ is given by (4.80). Final MDCT coefficients are obtained from (4.93). The $\frac{N}{4}$-point DST-II in (4.99) can be converted to the $\frac{N}{4}$-point DCT-II with proper preceding sign changes applied to odd-indexed samples.

The modified generalized signal flow graph of the improved fast DCT-II-based algorithm [22, 29, 35] for the unified forward and backward MDCT/MDST computation is shown in Fig. 4.9. The computation of backward MDCT/MDST is simply realized by reversing the signal flow graph for the forward MDCT computation and performing inverse operations. Compared to the fast refined MDCT algorithm defined by (4.87), (4.91), (4.92), and (4.93), the improved fast MDCT algorithm eliminates: the first butterfly stage, thus saving N additions (see Fig. 4.8), further eliminates scaling factors $\frac{\sqrt{2}}{2}$ and final sign changes, thus simplifying the resulting fast computational structure. Comparing (4.91) and (4.100) one can note that they involve Givens–Jacobi rotations of opposite types.

The total computational complexity of the improved fast DCT-II-based algorithm for the forward N-point MDCT computation [22, 29, 35] is given by $\frac{3N}{4}$ multiplications, $\frac{7N}{4} - 2$ additions ($\frac{5N}{4} - 2$ additions for the backward MDCT) plus the complexity of two $\frac{N}{4}$-point DCTs-II.

4.3.4 DCT-IV-Based Fast Algorithms

Another class of fast MDCT algorithms [19, 33, 37, 39] and fast MLT algorithms [2, 47, 51] is based on the DCT-IV (or DST-IV) of reduced size. In fact, using a simple permutation applied to the input data sequence $\{x_n\}$, the MDCT or MLT can always be converted to the DCT-IV of reduced size. Then, it is sufficient only to specify a suitable fast DCT-IV algorithm/computational structure, in general, valid for N being an even integer. It is widely accepted that the fast DCT-IV-based MDCT or MLT algorithms are the most efficient both in terms of the

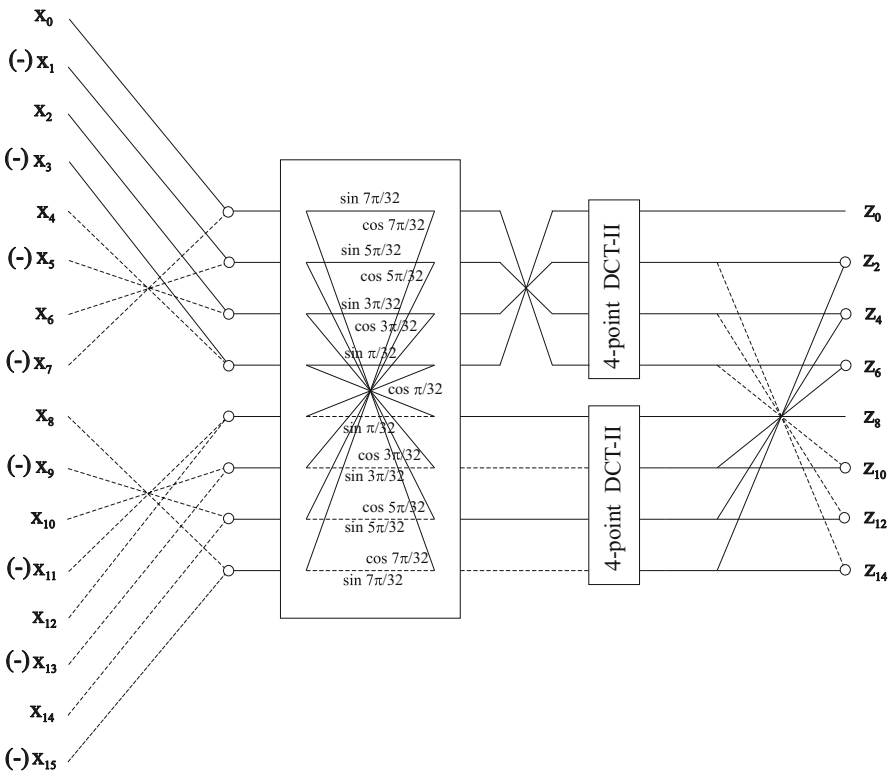

Fig. 4.9 Modified generalized signal flow graph of the improved fast DCT-II-based algorithm for the unified MDCT/MDST computation [22, 29, 35] in the oddly stacked system for $N = 16$

computational complexity and structural simplicity. The required DCT-IV (DST-IV) fast algorithms/computational structures are presented in Appendix C.2.

Essentially, applying the permutation defined by (4.80) to the input data sequence $\{x_n\}$, or a permutation originally introduced for the MLT defined as [2, 47, 51]

$$\begin{aligned} y_{\frac{N}{4}+n} &= x_n - x_{\frac{N}{2}-1-n}, \\ y_{\frac{N}{4}-1-n} &= -x_{\frac{N}{2}+n} - x_{N-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1, \end{aligned} \tag{4.101}$$

the N-point forward MDCT or MLT is converted into the $\frac{N}{2}$-point DCT-IV of $\{y_n\}$ [see Eq. (4.79)]. The backward MDCT or MLT is realized by the inverse $\frac{N}{2}$-point DCT-IV of $\{c_k^O\}$ using the same fast algorithm/computational structure. The time domain aliased data sequence $\{\hat{x}_n^{O-\text{MDCT}}\}$ can be recovered from $\{y_n\}$ applying an inverse permutation to that of (4.101) defined as

$$\begin{aligned}
\hat{x}_n^{O-\mathrm{MDCT}} &= y_{\frac{N}{4}+n}, & \hat{x}_{\frac{N}{2}-1-n}^{O-\mathrm{MDCT}} &= -y_{\frac{N}{4}+n}, \\
\hat{x}_{\frac{N}{2}+n}^{O-\mathrm{MDCT}} &= -y_{\frac{N}{4}-1-n}, & \hat{x}_{N-1-n}^{O-\mathrm{MDCT}} &= -y_{\frac{N}{4}-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1.
\end{aligned} \tag{4.102}$$

Although the permutations (4.80) and (4.101) seem to be different, they generate exactly the same time domain aliased data sequence $\{y_n\}$. Alternatively, applying the permutation (4.77) to the input data sequence $\{x_n\}$, the time domain aliased data sequence $\{\hat{x}_n^{O-\mathrm{MDCT}}\}$ is also recovered after the backward MDCT computation from $\{y_n\}$ as [19, 33]

$$\hat{x}_n^{O-\mathrm{MDCT}} = \begin{cases} y_{\frac{N}{4}+n}, & n = 0, 1, \ldots, \frac{3N}{4} - 1, \\ -y_{n-\frac{3N}{4}}, & n = \frac{3N}{4}, \frac{3N}{4} + 1, \ldots, N-1. \end{cases} \tag{4.103}$$

Since the DCT-IV matrix is symmetric and self-inverse, the $\frac{N}{2}$-point forward/inverse DCT-IV computation, and hence the N-point forward/backward MDCT (MLT) computation is realized by an identical fast computational structure with the properly appended permutations (4.101) and (4.102).

Taking into account permutation (4.80) or (4.101), the total computational complexity of fast DCT-IV-based algorithms [2, 47, 51] for the forward N-point MLT or MDCT/MDST computation is given by $\frac{N}{2}$ additions plus the complexity of $\frac{N}{2}$-point DCT-IV.

Note 7 Applying a permutation to the input data sequence $\{x_n\}$ defined as [3, 18]

$$\begin{aligned}
y_{\frac{N}{4}+n} &= x_n + x_{\frac{N}{2}-1-n}, \\
y_{\frac{N}{4}-1-n} &= x_{\frac{N}{2}+n} - x_{N-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1,
\end{aligned} \tag{4.104}$$

the N-point forward MDST is converted into the $\frac{N}{2}$-point DST-IV of $\{y_n\}$. The backward MDST is realized by the inverse $\frac{N}{2}$-point DST-IV of $\{s_k^O\}$ using the same fast algorithm or computational structure, and the time domain aliased data sequence $\{\hat{x}_n^{O-\mathrm{MDST}}\}$ can be recovered from $\{y_n\}$ applying an inverse permutation to that of (4.104) as

$$\begin{aligned}
\hat{x}_n^{O-\mathrm{MDST}} &= y_{\frac{N}{4}+n}, & \hat{x}_{\frac{N}{2}-1-n}^{O-\mathrm{MDST}} &= y_{\frac{N}{4}+n}, \\
\hat{x}_{\frac{N}{2}+n}^{O-\mathrm{MDST}} &= y_{\frac{N}{4}-1-n}, & \hat{x}_{N-1-n}^{O-\mathrm{MDST}} &= -y_{\frac{N}{4}-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1.
\end{aligned} \tag{4.105}$$

We note that based on a relation between the DST-IV and the DCT-IV (see Appendix C.2), the DST-IV can always be converted to the DCT-IV with proper preceding sign changes applied to odd-indexed samples.

Compact computational structures for the implementation of fast analysis and synthesis MDCT or MLT filter banks based on the DCT-IV including the windowing&overlap procedure in the analysis MDCT (MLT) filter bank and windowing&overlap&add procedure in the synthesis MDCT (MLT) filter bank are discussed in detail in Sect. 5.5 of Chap. 5.

4.3.5 DCT-IV/(Scaled)DCT-II-Based Fast Algorithms

Recall that applying the permutation (4.80) or (4.101) to the input data sequence $\{x_n\}$, the N-point forward MDCT is converted into the $\frac{N}{2}$-point DCT-IV of $\{y_n\}$ [see Eq. (4.79)]. The fast DCT-IV/DCT-II-based MDCT [19–21, 27, 31, 40, 41] or MLT algorithms [51] and the fast DCT-IV/scaled DCT-II (SDCT-II)-based MDCT algorithms [30, 43] utilize the following simple fact. Since there exists a simple relation between the DCT-IV and the DCT-II matrices [75] (see also Appendix C.3), the M-point DCT-IV may be converted to the DCT-II or SDCT-II of the same size at the cost of additional M pre-multiplications and $M-1$ recursive post-additions [83]. The required DCT-II fast algorithm/computational structure is presented in Appendix C.1.

Definitions of unnormalized M-point DCT-II and DCT-IV transforms are, respectively, presented in Appendices C.1 and C.2. Let $\{y_m\}$, $m = 0, 1, \ldots, M-1$, represent an input data sequence. An unnormalized M-point SDCT-II is defined as [30]

$$s_k^{\text{II}} = \epsilon_k \sum_{m=0}^{M-1} y_m \cos\left[\frac{\pi(2m+1)k}{2M}\right], \qquad k = 0, 1, \ldots, M-1, \tag{4.106}$$

where

$$\epsilon_k = \begin{cases} 1, & \text{if } k = 0, \\ 2, & \text{otherwise.} \end{cases} \tag{4.107}$$

Using the trigonometric identity $\cos(\alpha+\beta) = 2\cos\alpha\ \cos\beta - \cos(\alpha-\beta)$ to the DCT-IV kernel setting $\alpha = \frac{\pi(2m+1)k}{2M}$ and $\beta = \frac{\pi(2m+1)}{4M}$, the M-point DCT-IV is converted to the DCT-II of the same size as [83]

$$c_k^{\text{IV}} = \sum_{m=0}^{M-1} \left(2\, y_m \cos\frac{\pi(2m+1)}{4M}\right) \cos\left[\frac{\pi(2m+1)k}{2M}\right] - c_{k-1}^{\text{IV}}, \quad k = 0, 1, \ldots, M-1, \tag{4.108}$$

or to the SDCT-II of the same size as [30]

$$c_k^{\text{IV}} = \epsilon_k \sum_{m=0}^{M-1} \left(y_m \cos \frac{\pi(2m+1)}{4M} \right) \cos \left[\frac{\pi(2m+1)k}{2M} \right] - c_{k-1}^{\text{IV}},$$
$$k = 0, 1, \ldots, M-1, \tag{4.109}$$

where coefficients $\{c_k^{\text{IV}}\}$ and $\{c_k^{\text{II}}\}$, $(\{s_k^{\text{II}}\})$ satisfy the following conditions:

$$2\, c_0^{\text{IV}} = c_0^{\text{II}}, \quad (c_0^{\text{IV}} = s_0^{\text{II}}) \quad \text{if} \quad k = 0,$$
$$c_k^{\text{IV}} + c_{k-1}^{\text{IV}} = c_k^{\text{II}}, \quad (c_k^{\text{IV}} + c_{k-1}^{\text{IV}} = s_k^{\text{II}}) \quad \text{if} \quad k > 0. \tag{4.110}$$

Equations (4.108) or (4.109) and (4.110) for $M = \frac{N}{2}$ and $m = n$ define the first form of the fast DCT-IV/(S)DCT-II-based algorithm [19–21, 27, 31, 40, 41, 51]. Pre-multiplications by $\{2 \cos \frac{\pi(2m+1)}{4M}\}$ in (4.108) or by $\{\cos \frac{\pi(2m+1)}{4M}\}$ in (4.109) applied to the data sequence $\{y_m\}$ can be effectively absorbed into the windowing procedure in the complete analysis and synthesis MDCT or MLT filter banks.

However, an M-point DCT-II of $\{u_m\}$ in (4.108) or SDCT-II in of $\{u_m\}$ (4.109), whereby $u_m = 2\ y_m\ \cos \frac{\pi(2m+1)}{4M}$ for the DCT-II and $u_m = y_m\ \cos \frac{\pi(2m+1)}{4M}$ for the SDCT-II, can be recursively decomposed into $\frac{M}{2}$-point (S)DCT-II (even-indexed coefficients) and $\frac{M}{2}$-point DCT-IV (odd-indexed coefficients) as [30, 75]

$$c_{2k}^{\text{II}} = \sum_{m=0}^{\frac{M}{2}-1} (u_m + u_{M-1-m}) \cos \left[\frac{\pi(2m+1)k}{2(M/2)} \right],$$
$$c_{2k+1}^{\text{II}} = \sum_{m=0}^{\frac{M}{2}-1} (u_m - u_{M-1-m}) \cos \left[\frac{\pi}{2M}(2m+1)(2k+1) \right], \ k = 0, 1, \ldots, \frac{M}{2} - 1. \tag{4.111}$$

Using (C.33) from Appendix C.3, the $\frac{M}{2}$-point DCT-IV in (4.111) corresponding to odd-indexed coefficients $\{c_{2k+1}^{\text{II}}\}$ is converted to the $\frac{M}{2}$-point (S)DCT-II as [30]

$$c_{2k+1}^{\text{II}} = \sum_{m=0}^{\frac{M}{2}-1} \left(2\ (u_m - u_{M-1-m}) \cos \frac{\pi(2m+1)}{2M} \right) \cos \left[\frac{\pi(2m+1)k}{2(M/2)} \right] - c_{2k-1}^{\text{II}},$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{4.112}$$

where

$$2\, c_1^{\text{II}} = \hat{c}_0^{\text{II}}, \quad (c_1^{\text{II}} = \hat{s}_0^{\text{II}}) \quad \text{if} \quad k = 0,$$

$$c^{\text{II}}_{2k+1} + c^{\text{II}}_{2k-1} = \hat{c}^{\text{II}}_{k}, \quad (c^{\text{II}}_{2k+1} + c^{\text{II}}_{2k-1} = \hat{s}^{\text{II}}_{k}) \quad \text{if} \quad k > 0. \tag{4.113}$$

From (4.111) and (4.112) it can be seen that the M-point (S)DCT-II is decomposed into two identical $\frac{M}{2}$-point (S)DCTs-II. Equations (4.108) or (4.109), (4.110), (4.111), (4.112), and (4.113) for $M = \frac{N}{2}$ and $m = n$ define the second form of the fast DCT-IV/(S)DCT-II-based algorithm [30]. Since the DCT-IV matrix is symmetric and self-inverse, the forward/inverse DCT-IV computation via the (S)DCT-II (note that the DCT-II matrix is not symmetric), and hence the forward/backward MDCT or MLT may be realized by an identical fast computational structure. The time domain aliased data sequence $\{\hat{x}_n^{O-\text{MDCT}}\}$ is recovered after the backward MDCT computation according to (4.102).

Taking into account permutation (4.80) or (4.101), the total computational complexity of the first form of fast DCT-IV/DCT-II-based algorithm for the forward N-point MDCT or MLT computation is given by $N - 1$ additions, 1 shift plus the complexity of $\frac{N}{2}$-point DCT-II.

An interesting method of constructing a DCT-IV/SDCT-II-based algorithm for the forward/backward MDCT computation was reported in [43]. Consider the first form of algorithm defined by (4.108) and (4.110). Substituting $M = \frac{N}{2}$ and $m = n$ into (4.108), it can be rewritten into the equivalent form as

$$c^{\text{IV}}_{k} = \sum_{n=0}^{\frac{N}{2}-1} u_n \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right] - c^{\text{IV}}_{k-1}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.114}$$

where

$$u_n = 2\, y_n \cos\frac{\pi(2n+1)}{2N}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.115}$$

The data sequence $\{y_n\}$ in (4.115) is given by (4.80) or (4.101). Now using a simple trick, specifically, at first introducing the scaling factor $\sqrt{2}$ into the sum in (4.114) and immediately the scaling factor $\frac{1}{\sqrt{2}}$ into the right-hand side of (4.115), we obtain equivalent forms of (4.114) and (4.115), respectively, as [43]

$$c^{\text{IV}}_{k} = \sum_{n=0}^{\frac{N}{2}-1} u_n \sqrt{2} \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right] - c^{\text{IV}}_{k-1}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.116}$$

where

$$u_n = \frac{2}{\sqrt{2}}\, y_n \cos\frac{\pi(2n+1)}{2N} = \sqrt{2}\, y_n \cos\frac{\pi(2n+1)}{2N},$$
$$n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.117}$$

The cosine transform kernel in (4.116) is recognized as the $\frac{N}{2}$-point SDCT-II scaled by the factor $\sqrt{2}$. The just indicated simple trick is especially appreciated in the derivation of efficient MDCT implementation for MP3 audio coding standard (see Chap. 5). The total computational complexity of the algorithm is the same as that of the first form of algorithm defined by (4.108) and (4.110).

4.3.6 *Unified Evenly and Oddly Stacked MDCT/MDST Computation*

Another class of developed fast MDCT algorithms is based on a relation between the oddly stacked MDCT block transform given by (4.39) and the corresponding evenly stacked MDCT block transform given by (4.1). Although the oddly stacked and evenly stacked MDCT, at least according to their definitions, are quite different filter banks, they are closely related, and consequently, the efficient computation of oddly stacked MDCT can be realized via the evenly stacked MDCT and vice versa, only by simple pre- and post-processing of input and output data sequences [7]. This fact allows to handle evenly and oddly stacked MDCT block transform computations in a unified framework.

Consider the forward oddly stacked MDCT block transform given by (4.39). Using the trigonometric identity $\cos(\alpha+\beta) = 2\cos\alpha\ \cos\beta - \cos(\alpha-\beta)$ applied to the oddly stacked MDCT kernel in (4.39), setting $\alpha = \frac{\pi}{N}(2n+1+\frac{N}{2})k$ and $\beta = \frac{\pi}{2N}(2n+1+\frac{N}{2})$, we obtain the relation between the unnormalized oddly and evenly stacked MDCT block transforms defined as [7]

$$c_k^{o} = \sum_{n=0}^{N-1} y_n\ \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)\ k\right] - c_{k-1}^{o}, \qquad k = 0, 1, \ldots, \frac{N}{2}-1, \tag{4.118}$$

where

$$y_n = 2\ x_n\ \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)\right], \qquad n = 0, 1, \ldots, N-1. \tag{4.119}$$

The coefficients $\{c_k^{o}\}$ and $\{c_k^{E}\}$ satisfy the following conditions:

$$\begin{aligned} 2\ c_0^{o} &= c_0^{E}, \quad \text{if} \quad k = 0, \\ c_k^{o} + c_{k-1}^{o} &= c_k^{E}, \quad \text{if} \quad k > 0. \end{aligned} \tag{4.120}$$

The transform kernel on the right-hand side of (4.118) is recognized as an N-point evenly stacked MDCT. Hence, the N-point oddly stacked MDCT is converted

to the evenly stacked MDCT of the same size at the cost of additional N pre-multiplications and $N-1$ recursive post-additions. Note that this relation is quite similar to that of between the DCT-IV and the DCT-II given by (4.108).

Thus, the forward oddly stacked MDCT can be implemented via the forward evenly stacked MDCT algorithm only by simple pre- and post-processing of input and output data sequences, respectively. Any fast DCT-II-based MDCT algorithm for the evenly stacked system discussed in Sect. 4.2 can be adopted. Pre-multiplications by $\{2\cos\left[\frac{\pi}{2N}(2n+1+\frac{N}{2})\right]\}$ in (4.119) applied to the data sequence $\{y_n\}$ can be effectively absorbed into the windowing procedure in the complete analysis and synthesis MDCT or MLT filter banks, so saving N multiplications. Consequently, the total computational complexity of the oddly stacked MDCT is the same as that of the evenly stacked MDCT (see Table 4.1). We note that the number of multiplications and additions in Table 4.1 should be divided by 2.

The relation defined by (4.118), (4.119), and (4.120) may be represented equivalently in a matrix product [7] (see also Chap. 3). Inverting the matrix product results in the forward evenly stacked MDCT computation via the forward oddly stacked MDCT algorithm. Finally, the backward oddly/evenly stacked MDCT computation is realized by the forward evenly/oddly stacked MDCT algorithm but in reverse order performing inverse operations.

4.3.6.1 Comparison of Oddly Stacked Fast MDCT/MDST Algorithms

Comparison of the discussed fast DFT/FFT-, DCT-II-, DCT-IV-, DCT-IV/(S)DCT-II-, and evenly stacked MDCT-based algorithms for the MDCT and MDST computation in the oddly stacked system in terms of the arithmetic complexity for $N=2^n$, $n>2$ is summarized in Table 4.2.

For the DCT-II-based, DCT-IV-based and DCT-IV/(S)DCT-II-based fast and evenly stacked MDCT-based algorithms in general case, when N is a composite integer, i.e., it is of the form $N=2^n\times q$, where q is an odd positive integer (or

Table 4.2 Comparison of fast DFT/FFT-, DCT-II-, DCT-IV-, DCT-IV/(S)DCT-II-, and evenly stacked MDCT-based algorithms for the MDCT and MDST computation in the oddly stacked system in terms of the arithmetic complexity for $N=2^n$, $n>2$

Fast algorithm	# of mults	# of adds	# of shifts
DFT/FFT-based [23]	$\frac{N}{4}(n+1)+4$	$\frac{N}{4}(3n+1)+4$	
DFT/FFT-based [24, 34, 36]	$\frac{N}{4}(n+1)+4$	$\frac{N}{4}(3n+1)+4$	
DFT/FFT-based [25, 32]	$\frac{N}{4}(n+1)+1$	$\frac{N}{4}(3n-1)+1$	
Refined DCT-II-based [18]	$\frac{N}{4}(n+1)$	$\frac{N}{4}(3n+3)$	
Improved DCT-II-based [22, 29, 35]	$\frac{N}{4}(n+1)$	$\frac{N}{4}(3n-1)$	
DCT-IV-based	$\frac{N}{4}(n+1)$	$\frac{N}{4}(3n-1)$	
DCT-IV/(S)DCT-II-based	$\frac{N}{4}(n-1)$	$\frac{N}{4}(3n-1)$	1
Evenly stacked MDCT-based	$\frac{N}{4}(n-1)$	$\frac{N}{4}(3n-3)+2$	2

the mixed-radix length being the combination of radix-2 and radix-q lengths), the computational complexity of the N-point DCT-II is given by (C.8) and of N-point DCT-IV is given by (C.32) in Appendix C.

4.3.7 *Mixed-Radix Oddly Stacked Fast MDCT/MDST Algorithms*

Mixed-radix fast algorithms or fast algorithms for composite lengths are obtained by the combination of radix-q (q is an odd positive integer) and radix-2 fast algorithms. For a particular radix number q, the radix-q fast algorithms recursively divide the entire transform computation into q shorter $\frac{N}{q}$-point [72]. The computation of higher-order transforms is obtained by the recursive reusing the lower-order transforms. A generalized mixed-radix decomposition method applied to the DCT-II and DCT-III for the composite lengths $N = q^m \times 2^p$, $m,\ p > 0$, is described in [72]. It enables to optimize the performance of such algorithms in terms of the arithmetic complexity during the decomposition process for some special lengths.

All the fast MDCT algorithms described in the previous subsections convert the N-point MDCT, where N is integer divisible by 4, to a discrete unitary (orthogonal) transform such as the DFT or DCT, in general, of reduced odd-lengths. Consequently, radix-q fast algorithms can be applied to the q-length DFT or DCT [21]. It is important to note that the strictly radix-q fast MDCT algorithm cannot be constructed, since from the MDCT definition it follows that any odd q^m divided by 2 is not an integer. Therefore, the mixed-radix fast MDCT algorithms may be constructed only.

Several (recursive) mixed-radix fast algorithms for the MDCT computation were developed [17, 26, 38, 42, 44]. The first one [38, 44] is obtained by the decimation in frequency (DIF) method for composite lengths $N = 3^m \times 2, m > 0$, and subsequently it was improved and extended for composite lengths $N = 3^m \times 2^p, m > 0, p > 1$ [17]. The second one [42] is obtained by the decimation in time (DIT) method for composite lengths $N = 3^m \times 4, m > 0$. This DIT mixed-radix fast algorithms was extended to composite lengths $N = q^m \times 2^p, m,\ p > 1$, where $q = 3,\ 5$, and 9 [26]. Essentially, DIF and DIT fast MDCT algorithms can be combined with a recursive radix-2 DIF fast MDCT algorithm [42] to obtain in general fast algorithms for composite lengths $N = q^m \times 2^p, m > 0, p > 1$. All mixed-radix fast MDCT algorithms are discussed in detail in the following subsections.

4.3.7.1 DIF Mixed-Radix Fast Algorithm [38, 44]

Consider the forward MDCT given by (4.39) with the MDCT coefficients split into three sets, specifically, $\{c^O_{3k}\}$, $\{c^O_{3k+1}\}$, and $\{c^O_{3k+2}\}$ for $k = 0, 1, \ldots, \frac{N}{6} - 1$, where $N = 3^m \times 2, m > 0$. From (4.39) for the MDCT coefficients $\{c^O_{3k+1}\}$ we immediately have [38, 44]

$$
\begin{aligned}
\overset{o}{c}_{3k+1} &= \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(6k+3)\right] \\
&= \sum_{n=0}^{N-1} x_n \cos\left[\frac{3\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \\
k &= 0, 1, \ldots, \frac{N}{6}-1.
\end{aligned} \tag{4.121}
$$

Subsequently substituting $\frac{N}{3}-1-n$, $\frac{2N}{3}-1-n$, and then $N-1-n$ for n into (4.121) we get

$$
\begin{aligned}
\overset{o}{c}_{3k+1} &= \sum_{n=0}^{\frac{N}{3}-1} (x_{\frac{N}{3}-1-n} - x_{\frac{2N}{3}-1-n} + x_{N-1-n}) \\
&\quad \cos\left[\frac{\pi}{2(N/3)}\left(2n+1+\frac{N}{6}\right)(2k+1)\right], \\
k &= 0, 1, \ldots, \frac{N}{6}-1.
\end{aligned} \tag{4.122}
$$

Further, from (4.39) for the MDCT coefficients $\{\overset{o}{c}_{3k}\}$ and $\{\overset{o}{c}_{3k+2}\}$ we, respectively, have

$$
\begin{aligned}
\overset{o}{c}_{3k} &= \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(6k+1)\right], \\
\overset{o}{c}_{3k+2} &= \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(6k+5)\right], \qquad k = 0, 1, \ldots, \frac{N}{6}-1.
\end{aligned} \tag{4.123}
$$

Now, introduce the data sequences $\{a_k\}$ and $\{b_k\}$ defined as [38, 44]

$$
a_k = \overset{o}{c}_{3k} + \overset{o}{c}_{3k+2}, \qquad b_k = \overset{o}{c}_{3k} - \overset{o}{c}_{3k+2}, \qquad k = 0, 1, \ldots, \frac{N}{6}-1. \tag{4.124}
$$

Combining the MDCT transform kernels in (4.123), and using the trigonometric identities for the sums/differences of cosines, the data sequences $\{a_k\}$ and $\{b_k\}$ are given by

$$
a_k = -\sum_{n=0}^{N-1} \left(x_n\, 2\, \sin\frac{\pi(2n+1)}{N}\right) \cos\left[\frac{3\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],
$$

$$b_k = \sum_{n=0}^{N-1} \left(x_n \; 2 \; \cos \frac{\pi(2n+1)}{N} \right) \; \sin \left[\frac{3\pi}{2N} \left(2n + 1 + \frac{N}{2} \right) \; (2k+1) \right],$$

$$k = 0, 1, \ldots, \frac{N}{6} - 1. \tag{4.125}$$

From (4.124) it follows that the MDCT coefficients $\{c^{o}_{3k}\}$ and $\{c^{o}_{3k+2}\}$ are obtained as

$$c^{o}_{3k} = \frac{1}{2} \; (a_k + b_k), \qquad c^{o}_{3k+2} = \frac{1}{2} \; (a_k - b_k), \qquad k = 0, 1, \ldots, \frac{N}{6} - 1. \tag{4.126}$$

The last step involves subsequent substituting $\frac{N}{3} - 1 - n$, $\frac{2N}{3} - 1 - n$, and then $N - 1 - n$ for n into the first equation in (4.125), and we get

$$\begin{aligned} a_k = - \sum_{n=0}^{\frac{N}{3}-1} \Bigg[& \left(x_{\frac{N}{3}-1-n} + x_{\frac{2N}{3}-1-n} \right) \sqrt{3} \; \cos \frac{\pi(2n+1)}{N} \\ & + \left(x_{\frac{N}{3}-1-n} - x_{\frac{2N}{3}-1-n} - 2 \; x_{N-1-n} \right) \; \sin \frac{\pi(2n+1)}{N} \Bigg] \\ & \times \cos \left[\frac{\pi}{2(N/3)} \left(2n + 1 + \frac{N}{6} \right) \; (2k+1) \right], \end{aligned}$$

$$k = 0, 1, \ldots, \frac{N}{6} - 1. \tag{4.127}$$

Similarly, subsequent substituting $\frac{N}{3} + n$ and $\frac{2N}{3} + n$ for n into the second equation of (4.125), we finally get

$$\begin{aligned} b_k = (-1)^k \sum_{n=0}^{\frac{N}{3}-1} \Bigg[& \left(2 \; x_n + x_{\frac{N}{3}+n} - x_{\frac{2N}{3}+n} \right) \; \cos \frac{\pi(2n+1)}{N} \\ & + \left(x_{\frac{N}{3}+n} + x_{\frac{2N}{3}+n} \right) \sqrt{3} \; \sin \frac{\pi(2n+1)}{N} \Bigg] \\ & \times \cos \left[\frac{\pi}{2(N/3)} \left(2n + 1 + \frac{N}{6} \right) \; (2k+1) \right], \end{aligned}$$

$$k = 0, 1, \ldots, \frac{N}{6} - 1. \tag{4.128}$$

From (4.122), (4.127), and (4.128) it can be seen that the N-point forward MDCT is decomposed into three $\frac{N}{3}$-point MDCTs. Equations (4.122), (4.126), (4.127), and (4.128) define the DIF mixed-radix fast MDCT algorithm for composite lengths $N = 3^m \times 2$, $m > 0$. The factor $\frac{1}{2}$ from (4.126) may simply be absorbed in (4.127) and (4.128).

The total computational complexity of the DIF mixed-radix fast algorithm for the forward N-point MDCT or MLT computation [38, 44] without any optimization (M_N is the number of multiplications and A_N is the number of additions) is generally given by

$$M_N = 3\, M_{\frac{N}{3}} + \frac{4N}{3}, \qquad A_N = 3\, A_{\frac{N}{3}} + \frac{10N}{3}, \tag{4.129}$$

where $M_{\frac{N}{3}}$ and $A_{\frac{N}{3}}$ denote the number of multiplications and additions, respectively, for the computation of one $\frac{N}{3}$-point MDCT. Note that $\frac{N}{3}$ multiplications by 2 are implicitly taken as shift operations.

A detailed analysis of the computational structure of fast algorithm [38, 44] reveals that a corresponding generalized signal graph is partially regular. The reason why it is not completely regular, results from the following fact. Each algebraic expression between round brackets under the sum of (4.127) corresponding to $\{a_k\}$ is a proper linear combination of two algebraic expressions between round brackets under the sum of (4.128) corresponding to $\{b_k\}$, but for $n = \frac{N}{3} - 1 - n$, hence for algebraic expressions generated in reverse order. Indeed, for a given N taking two algebraic expressions from (4.128) for $n = \frac{N}{3} - 1 - n$, the following algebraic identities hold:

$$\begin{aligned}
2(x_{\frac{N}{3}-1-n} + x_{\frac{2N}{3}-1-n}) &= (2\, x_{\frac{N}{3}-1-n} + x_{\frac{2N}{3}-1-n} - x_{N-1-n}) \\
&\quad + (x_{\frac{2N}{3}-1-n} + x_{N-1-n}), \\
(x_{\frac{N}{3}-1-n} - x_{\frac{2N}{3}-1-n} - 2\, x_{N-1-n}) &= \frac{1}{2}\,(2\, x_{\frac{N}{3}-1-n} + x_{\frac{2N}{3}-1-n} - x_{N-1-n}) \\
&\quad - \frac{3}{2}\,(x_{\frac{2N}{3}-1-n} + x_{N-1-n}), \\
n &= 0, 1, \ldots, \frac{N}{3} - 1.
\end{aligned} \tag{4.130}$$

Consequently, the second $\frac{N}{3}$-point MDCT corresponding to $\{a_k\}$ needs to be constructed by a separate way using the indicated linear combinations of two algebraic expressions given by (4.130). Thus, the evaluation of (4.130) requires totally $\frac{N}{3}$ multiplications by 3 and $\frac{2N}{3}$ additions, thus saving $\frac{N}{3}$ additions in the original DIF mixed-radix fast MDCT algorithm [38, 44].

The original DIF mixed-radix fast MDCT algorithm [38, 44] was improved both in terms of the regularity and computational complexity and extended for composite lengths $N = 3^m \times 2^p$, $m > 0$, $p > 1$ in [17]. It is discussed in the following subsection.

4.3.7.2 Improved and Extended Recursive DIF Mixed-Radix Fast Algorithm [17]

From (4.122), (4.127), and (4.128) it can be seen that all algebraic expressions between round brackets combined with cosine/sine twiddle factors are quite different. The main idea for the derivation of improved mixed-radix DIF fast MDCT algorithm is based on (4.130). Defining each algebraic expression between round brackets under the first sum of (4.127) is derived by the linear combination of two algebraic expressions between round brackets under the second sum of (4.128). If we denote

$$u_n = 2x_n + x_{\frac{N}{3}+n} - x_{\frac{2N}{3}+n}, \qquad v_n = x_{\frac{N}{3}+n} + x_{\frac{2N}{3}+n}, \tag{4.131}$$

the following simple algebraic identities hold [17]:

$$\begin{aligned} x_{\frac{N}{3}-1-n} + x_{\frac{2N}{3}-1-n} &= \frac{1}{2}(u_{\frac{N}{3}-1-n} + v_{\frac{N}{3}-1-n}), \\ x_{\frac{N}{3}-1-n} - x_{\frac{2N}{3}-1-n} - 2x_{N-1-n} &= \frac{1}{2}(u_{\frac{N}{3}-1-n} - 3\, v_{\frac{N}{3}-1-n}), \\ n &= 0, 1 \ldots, \frac{N}{3} - 1. \end{aligned} \tag{4.132}$$

Substituting the algebraic identities (4.132) followed immediately by substituting $\frac{N}{3} - 1 - n$ for n into (4.127), and then substituting (4.131) into (4.128), after some algebraic manipulations we get a very regular form of the improved mixed-radix DIF fast MDCT algorithm defined as [17]

$$\begin{aligned} c_{3k+1} &= \sum_{n=0}^{\frac{N}{3}-1} \left[x_{\frac{N}{3}-1-n} - (x_{\frac{2N}{3}-1-n} - x_{N-1-n}) \right] \cos \phi_{n,k}, \\ k &= 0, 1, \ldots, \frac{N}{6} - 1, \\ \phi_{n,k} &= \frac{\pi}{2(N/3)} \left(2n + 1 + \frac{N}{6} \right) (2k + 1), \end{aligned} \tag{4.133}$$

with

$$a_k = -\sum_{n=0}^{\frac{N}{3}-1}\left[u_n \sin\frac{\pi(2n+1)}{N} - v_n\sqrt{3}\cos\frac{\pi(2n+1)}{N}\right]\cos\phi_{\frac{N}{3}-1-n,k},$$

$$b_k = (-1)^k\sum_{n=0}^{\frac{N}{3}-1}\left[u_n \cos\frac{\pi(2n+1)}{N} + v_n\sqrt{3}\sin\frac{\pi(2n+1)}{N}\right]\cos\phi_{n,k},$$

$$k = 0, 1, \ldots, \frac{N}{6}-1. \tag{4.134}$$

The complete set of MDCT coefficients is obtained from (4.126). Due to the algorithm regularity, the algebraic expression between round brackets in (4.122) is rewritten into the more convenient equivalent form in Eq. (4.133). Comparing (4.127) and (4.128) with (4.134) one can see that the terms u_n and v_n combined with the sine/cosine twiddle factors in (4.134) are the same. This fact enables us to investigate the computational structure of the algorithm in detail for all composite lengths $N = 3^m \times 2^p$, $m, p > 0$. The cosine transform kernel $\cos\phi_{\frac{N}{3}-1-n,k}$ in the first sum of (4.134) is recognized as the MDCT kernel but in reverse order, whereby $\cos\phi_{\frac{N}{3}-1-n,k} = (-1)^{k+1}\sin\phi_{n,k}$. For a given N the computation of a higher-order MDCT is obtained by recursively reusing three lower-order $\frac{N}{3}$-point MDCTs. The backward MDCT computation can be simply realized by reversing a fast computational structure for the forward MDCT and performing the inverse operations.

In general, the computational complexity of improved mixed-radix DIF fast MDCT algorithm [17] for composite lengths $N = 3^m \times 2^p$, $m, p > 0$, is given by the complexity of three $\frac{N}{3}$-point MDCTs associated with the value of p, and the complexity of (4.134). The best 2^p-point ($p > 1$) fast MDCT algorithm requires $\frac{N}{4}(p-1)$ multiplications and $\frac{N}{4}(3p-3)+2$ additions (see Table 4.2). The evaluation of algebraic expressions in (4.131) requires $\frac{4N}{3}$ additions, whereby $\frac{N}{3}$ multiplications by 2 are counted as additions (note that the multiplications by 2 can be implemented as shift operations). The evaluation of algebraic expressions between the square brackets in (4.133) requires only $\frac{N}{3}$ additions because the sub-expressions $(x_{\frac{2N}{3}-1-n} - x_{N-1-n})$ are precomputed in (4.131) for $u_{\frac{N}{3}-1-n}$. The computation of $\{a_k\}$ and $\{b_k\}$ in (4.134) requires $\frac{4N}{3}$ multiplications and $\frac{2N}{3}$ additions. Finally, to obtain the complete set of MDCT coefficients from (4.126) we need $\frac{N}{3}$ additions. Then, the total arithmetic complexity of the improved mixed-radix DIF fast MDCT algorithm for composite lengths $N = 3^m \times 2^p$, $m > 0$, $p > 2$, is given by Britanak [17]

$$M_N = 3 \times M_{\frac{N}{3}} + \frac{4N}{3}, \qquad A_N = 3 \times A_{\frac{N}{3}} + \frac{8N}{3}, \tag{4.135}$$

Compared to the total arithmetic complexity of the original algorithm [38, 44], $\frac{2N}{3}$ additions are saved in the improved algorithm [17]. In general, for composite lengths $N = 3^m \times 2^p$, the number of arithmetic operations for some special angles can be further reduced by an individual analysis. As an example, in the following the detailed analysis of computational complexity for composite lengths $N = 3^m \times 2$ is presented.

Let the factors $\frac{1}{2}$ in (4.126) be absorbed in (4.134). The evaluation of (4.134) for one n requires 4 multiplications and 2 additions. For $n = \frac{1}{2}(\frac{N}{6} - 1)$ the angle is $\frac{\pi}{6}$, then $\cos\frac{\pi}{6} = \frac{\sqrt{3}}{2}$, $\sin\frac{\pi}{6} = \frac{1}{2}$, and the expressions between square brackets in (4.134) are:

$$u_n \frac{1}{2}\sin\frac{\pi}{6} - v_n \frac{\sqrt{3}}{2}\cos\frac{\pi}{6} = \frac{1}{4}(u_n - 3v_n),$$

$$u_n \frac{1}{2}\cos\frac{\pi}{6} + v_n \frac{\sqrt{3}}{2}\sin\frac{\pi}{6} = \frac{\sqrt{3}}{4}(u_n + v_n),$$

requiring 1 multiplication, 4 additions, and 1 shift, where multiplication by 3 is realized by 2 additions. On the other hand, for $n = \frac{N}{6} + \frac{1}{2}(\frac{N}{6} - 1)$ the angle is $\frac{\pi}{2}$ and therefore, the expressions between square brackets in Eq. (4.134) are:

$$u_n \frac{1}{2}\sin\frac{\pi}{2} - v_n \frac{\sqrt{3}}{2}\cos\frac{\pi}{2} = \frac{1}{2}u_n,$$

$$u_n \frac{1}{2}\cos\frac{\pi}{2} + v_n \frac{\sqrt{3}}{2}\sin\frac{\pi}{2} = \frac{\sqrt{3}}{2}v_n,$$

requiring 1 multiplication and 1 shift, so totally saving of 6 multiplications. Then, the total arithmetic complexity for the composite lengths $N = 3^m \times 2,\ m > 0$, is given by Britanak [17]

$$M_N = 3 \times M_{\frac{N}{3}} + \frac{4N}{3} - 6, \qquad A_N = 3 \times A_{\frac{N}{3}} + \frac{8N}{3}, \tag{4.136}$$

where for $N = 3\times 2 = 6, M_6 = 2, A_6 = 16$ plus 2 shifts (compared to the arithmetic complexity $M_6 = 5, A_6 = 15$ in [38]). Exploiting the fact that the transform kernel of 2-point MDCT is $\cos\frac{\pi}{2}$ and $\cos\pi$ (and hence $M_2 = A_2 = 0$), the forward 6-point MDCT computation can be optimized in terms of the arithmetic complexity. The optimized signal flow graph for the forward 6-point MDCT computation is shown in Fig. 4.10. All redundant computations are indicated by the thicker lines in the signal flow graph in Fig. 4.10. Removing these redundant computations results in a forward 6-point MDCT module with the arithmetic complexity $M_6 = 1, A_6 = 11$ plus 1 shift (note that two multiplications by 2 are counted as 2 additions). Since the 6-point forward MDCT is recursively used for the computation of the higher-order forward MDCTs, the total arithmetic complexity is reduced significantly.

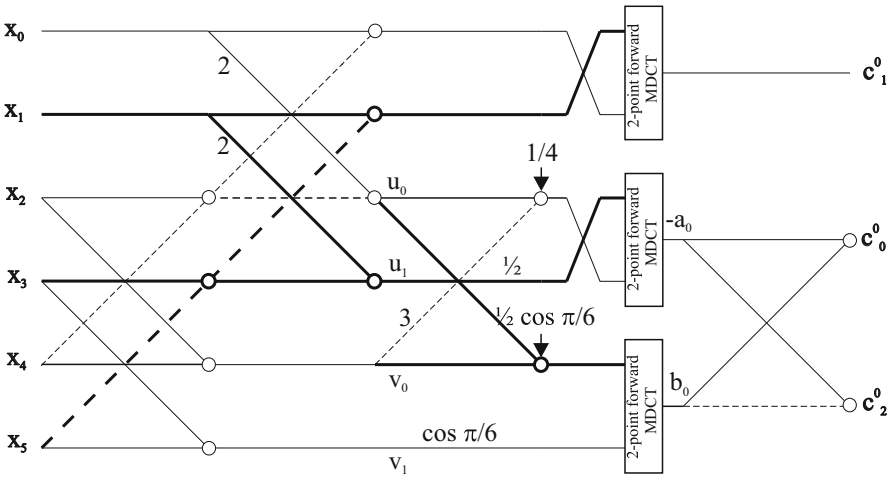

Fig. 4.10 Optimized signal flow graph for the forward 6-point MDCT computation

The analysis of computational complexity for the case of $N = 3^m \times 4$ can be found in [17] (see also Chap. 5).

4.3.7.3 Recursive DIT Mixed-Radix Fast Algorithms [26, 42]

Split the input data sequence $\{x_n\}$ into three sets of samples, specifically, $\{x_{3n}\}$, $\{x_{3n+1}\}$ and $\{x_{3n+2}\}$ for $n = 0, 1, \ldots, \frac{N}{3} - 1$, where $N = 3^m \times 4$, $m > 0$. Then, the forward MDCT given by (4.39) is decomposed as [42]

$$\begin{aligned} c_k^o = & \sum_{n=0}^{\frac{N}{3}-1} x_{3n+1} \cos\left[\frac{3\pi}{2N}\left(2n+1+\frac{N}{6}\right)(2k+1)\right] \\ & + \sum_{n=0}^{\frac{N}{3}-1} x_{3n} \cos\left[\frac{\pi}{2N}(6n+1+\frac{N}{2})(2k+1)\right] \\ & + \sum_{n=0}^{\frac{N}{3}-1} x_{3n+2} \cos\left[\frac{\pi}{2N}(6n+5+\frac{N}{2})(2k+1)\right], \quad k = 0, 1, \ldots, \frac{N}{2}-1. \end{aligned} \tag{4.137}$$

Combining the cosine transform kernels in the second and the third sum on the right-hand side of (4.137), and using the trigonometric identities for the sums/differences of cosines we get [42]

$$
\begin{aligned}
c_k^o &= \sum_{n=0}^{\frac{N}{3}-1} x_{3n+1} \cos\left[\frac{3\pi}{2N}\left(2n+1+\frac{N}{6}\right)(2k+1)\right] \\
&\quad + \sum_{n=0}^{\frac{N}{3}-1} x_{3n} \left\{\cos\alpha_n \cos\left[\frac{3\pi}{2N}\left(2n+1+\frac{N}{6}\right)(2k+1)\right]\right. \\
&\quad \left. + \sin\alpha_n \sin\left[\frac{3\pi}{2N}\left(2n+1+\frac{N}{6}\right)(2k+1)\right]\right\} \\
&\quad + \sum_{n=0}^{\frac{N}{3}-1} x_{3n+2} \left\{\cos\alpha_n \cos\left[\frac{3\pi}{2N}\left(2n+1+\frac{N}{6}\right)(2k+1)\right]\right. \\
&\quad \left. - \sin\alpha_n \sin\left[\frac{3\pi}{2N}\left(2n+1+\frac{N}{6}\right)(2k+1)\right]\right\}, \\
k &= 0,1,\ldots,\frac{N}{2}-1, \qquad \alpha_n = \frac{\pi(2n+1)}{N}.
\end{aligned} \tag{4.138}
$$

Now, for the further development rewrite (4.138) to more simplified form as

$$
c_k^o = e_k + f_k \cos\alpha_n + g_k \sin\alpha_n, \qquad k = 0,1,\ldots,\frac{N}{2}-1, \tag{4.139}
$$

where

$$
\begin{aligned}
e_k &= \sum_{n=0}^{\frac{N}{3}-1} x_{3n+1} \cos\left[\frac{\pi}{2(N/3)}\left(2n+1+\frac{N}{6}\right)(2k+1)\right], \\
f_k &= \sum_{n=0}^{\frac{N}{3}-1} (x_{3n}+x_{3n+2}) \cos\left[\frac{\pi}{2(N/3)}\left(2n+1+\frac{N}{6}\right)(2k+1)\right], \\
g_k &= \sum_{n=0}^{\frac{N}{3}-1} (x_{3n}-x_{3n+2}) \sin\left[\frac{\pi}{2(N/3)}\left(2n+1+\frac{N}{6}\right)(2k+1)\right], \\
k &= 0,1,\ldots,\frac{N}{6}-1.
\end{aligned} \tag{4.140}
$$

The MDST kernel in (4.140) corresponding to $\{g_k\}$, by substituting $\frac{N}{6}-1-k$ for k is converted to the MDCT kernel (note that N has to be integer divisible by 12), and we get

$$g_{\frac{N}{6}-1-k} = \sum_{n=0}^{\frac{N}{3}-1} (-1)^{n+\frac{N}{12}} \, (x_{3n} - x_{3n+2}) \, \cos\left[\frac{\pi}{2(N/3)}\left(2n+1+\frac{N}{6}\right)\,(2k+1)\right],$$

$$k = 0, 1, \ldots, \frac{N}{6} - 1. \tag{4.141}$$

From (4.140) and (4.141) it can be seen that the N-point forward MDCT is decomposed into three $\frac{N}{3}$-point MDCTs. Finally, substituting subsequently $\frac{N}{3}-1-k$ and $\frac{N}{3}+k$ into (4.139), after some algebraic manipulations we obtain the complete formulae of the DIT mixed-radix fast MDCT algorithm as [42]

$$\begin{aligned}
c_k^{o} &= e_k + (f_k \, \cos\alpha_n + g_k \, \sin\alpha_n)\,, \\
c_{\frac{N}{3}-1-k}^{o} &= -e_k + \frac{1}{2}\,(f_k \, \cos\alpha_n + g_k \, \sin\alpha_n) - \frac{\sqrt{3}}{2}\,(f_k \, \sin\alpha_n - g_k \, cos\alpha_n)\,, \\
c_{\frac{N}{3}+k}^{o} &= -e_k + \frac{1}{2}\,(f_k \, \cos\alpha_n + g_k \, \sin\alpha_n) + \frac{\sqrt{3}}{2}\,(f_k \, \sin\alpha_n - g_k \, \cos\alpha_n)\,, \\
k &= 0, 1, \ldots, \frac{N}{6} - 1,
\end{aligned} \tag{4.142}$$

where $\{e_k\}$, $\{f_k\}$ are defined by (4.140), and $\{g_{\frac{N}{6}-1-k}\}$ is defined by (4.141). For a given N the computation of a higher-order MDCT is obtained by recursively reusing three lower-order $\frac{N}{3}$-point MDCTs.

For composite lengths $N = 3^m \times 4$, $m > 0$, the multiplicative complexity of the DIT mixed-radix fast MDCT algorithm can be further reduced as follows. For the value of $n = \frac{1}{2}(\frac{N}{4} - 1)$ the angle is $\frac{\pi}{4}$, and from (4.142) we have

$$\begin{aligned}
\frac{1}{2}\,\left(f_k \, \cos\frac{\pi}{4} + g_k \, \sin\frac{\pi}{4}\right) &= \frac{1}{2}\cos\frac{\pi}{4}\,(f_k + g_k), \\
\frac{\sqrt{3}}{2}\,\left(f_k \, \sin\frac{\pi}{4} - g_k \, \cos\frac{\pi}{4}\right) &= \frac{\sqrt{3}}{2}\,\cos\frac{\pi}{4}\,(f_k - g_k),
\end{aligned}$$

requiring 2 multiplications and 2 additions (instead of 4 multiplications and 2 additions), so saving 2 multiplications. Then, the total computational complexity of the mixed-radix DIT fast MDCT algorithm is given by Wu et al. [42]

$$M_N = 3 \times M_{\frac{N}{3}} + \frac{2N}{3} - 2, \quad A_N = 3 \times A_{\frac{N}{3}} + \frac{5N}{3}, \tag{4.143}$$

plus $\frac{N}{6}$ shift operations. The regular signal flow graph for the forward MDCT computation for $N = 3 \times 4 = 12$ is shown in Fig. 5.20 (see Chap. 5). The backward MDCT computation is realized simply by reversing the signal flow graph for the forward MDCT computation and performing inverse operations.

Note 8 Following the decomposition method applied to the DCT-II and DCT-III [72], the DIT mixed-radix fast MDCT algorithm [42] was generalized and extended in [26] for composite lengths $N = q^m \times 2^p$, $m,\ p > 1$, where $q = 3,\ 5$ and 9.

4.3.7.4 Recursive Radix-2 DIF Fast Algorithm [42]

Essentially, the mixed-radix decomposition process can start with any radix number. However, a lower computational complexity is achieved, if the decomposition process starts with the smallest radix number, i.e., with the radix-2 [72]. A such radix-2 recursive DIF fast MDCT algorithm for any even-lengths N divisible by 4 was developed in [42], and it can be combined with any DIF or DIT mixed-radix fast algorithm for composite lengths $N = q^m \times 2^p$, $m > 0$, $p > 1$. The radix-2 recursive DIF fast algorithm decomposes the N-point MDCT into two $\frac{N}{2}$-point MDCTs.

Let us introduce the following two sub-sequences $\{a_k\}$ and $\{b_k\}$ defined as [42]

$$a_k = \overset{o}{c}_{2k} + \overset{o}{c}_{2k+1}, \qquad b_k = \overset{o}{c}_{2k} - \overset{o}{c}_{2k+1}, \qquad k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.144}$$

Then, from (4.144) and the definition of forward MDCT given by (4.39), the data sequences $\{a_k\}$ and $\{b_k\}$ by using the trigonometric identities for the sums/differences of cosines are, respectively, given by Wu et al. [42]

$$\begin{aligned} a_k &= \sum_{n=0}^{N-1} x_n \left\{ \cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)\ (4k+1)\right] \right. \\ &\quad \left. + \cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)\ (4k+3)\right]\right\} \\ &= \sum_{n=0}^{N-1} 2\ x_n\ \cos\left[\frac{\pi}{4} + \frac{\pi(2n+1)}{2N}\right] \\ &\quad \times \cos\left[\frac{\pi}{N}\left(2n + 1 + \frac{N}{4}\right)\ (2k+1) + \frac{\pi}{4}(2k+1)\right], \\ k &= 0, 1, \ldots, \frac{N}{4} - 1, \end{aligned} \tag{4.145}$$

and

$$\begin{aligned} b_k &= \sum_{n=0}^{N-1} x_n \left\{ \cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)\ (4k+1)\right] \right. \\ &\quad \left. - \cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)\ (4k+3)\right]\right\} \end{aligned}$$

$$= \sum_{n=0}^{N-1} 2\, x_n \, \sin\left[\frac{\pi}{4} + \frac{\pi(2n+1)}{2N}\right]$$
$$\times \sin\left[\frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)\,(2k+1) + \frac{\pi}{4}(2k+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.146}$$

Further, (4.144) implies that the even- and odd-indexed MDCT coefficients are, respectively, obtained as

$$\overset{o}{c}_{2k} = \frac{1}{2}\,(a_k + b_k), \qquad \overset{o}{c}_{2k+1} = \frac{1}{2}\,(a_k - b_k), \qquad k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.147}$$

At first consider (4.145). Extending the cosine transform kernel on the right-hand side of (4.145), i.e., the transform kernel $\cos\left[\frac{\pi}{N}(2n+1+\frac{N}{4})\,(2k+1) + \frac{\pi}{4}(2k+1)\right]$, we get

$$a_k = 2\, \cos\frac{\pi}{4}\,(2k+1) \sum_{n=0}^{N-1} x_n$$
$$\times \cos\left[\frac{\pi}{4} + \frac{\pi(2n+1)}{2N}\right] \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)\,(2k+1)\right] -$$
$$-2\, \sin\frac{\pi}{4}\,(2k+1) \sum_{n=0}^{N-1} x_n \, \cos\left[\frac{\pi}{4} + \frac{\pi(2n+1)}{2N}\right]$$
$$\times \sin\left[\frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)\,(2k+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.148}$$

Similarly, considering (4.146) and extending the sine transform kernel on the right-hand side of (4.146), i.e., the transform kernel $\sin\left[\frac{\pi}{N}(2n+1+\frac{N}{4})\,(2k+1) + \frac{\pi}{4}(2k+1)\right]$, we get

$$b_k = 2\, \sin\frac{\pi}{4}\,(2k+1) \sum_{n=0}^{N-1} x_n \, \sin\left[\frac{\pi}{4} + \frac{\pi(2n+1)}{2N}\right]$$
$$\times \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)\,(2k+1)\right]$$

$$-2\ \cos\frac{\pi}{4}\ (2k+1)\sum_{n=0}^{N-1}x_n\ \sin\left[\frac{\pi}{4}+\frac{\pi(2n+1)}{2N}\right]$$
$$\times\sin\left[\frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)\ (2k+1)\right],\quad k=0,1,\ldots,\frac{N}{4}-1. \tag{4.149}$$

Using the trigonometric identity

$$\cos\frac{\pi}{4}\ (2k+1)=(-1)^k\ \sin\frac{\pi}{4}\ (2k+1)=(-1)^{\lfloor\frac{k+1}{2}\rfloor}\ \frac{\sqrt{2}}{2}, \tag{4.150}$$

where $\lfloor\ .\ \rfloor$ denotes the lower integer part of argument, the terms $\{a_k\}$ given by (4.148) can be rewritten into simplified form as

$$a_k=\sqrt{2}\ (-1)^{\lfloor\frac{k+1}{2}\rfloor}\ (a_k^{(1)}-(-1)^k\ a_k^{(2)}), \tag{4.151}$$

where

$$a_k^{(1)}=\sum_{n=0}^{N-1}x_n\ \cos\left[\frac{\pi}{4}+\frac{\pi(2n+1)}{2N}\right]\ \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)\ (2k+1)\right],$$
$$a_k^{(2)}=\sum_{n=0}^{N-1}x_n\ \cos\left[\frac{\pi}{4}+\frac{\pi(2n+1)}{2N}\right]\ \sin\left[\frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)\ (2k+1)\right],$$
$$k=0,1,\ldots,\frac{N}{4}-1. \tag{4.152}$$

Similarly, using the trigonometric identity (4.150), and algebraic identity

$$(-1)^{\lfloor\frac{k+1}{2}\rfloor}=(-1)^{k+\lfloor\frac{k}{2}\rfloor}=(-1)^k\ (-1)^{\lfloor\frac{k}{2}\rfloor}, \tag{4.153}$$

the terms $\{b_k\}$ given by (4.149) can also be rewritten into simplified form as

$$b_k=\sqrt{2}\ (-1)^{\lfloor\frac{k}{2}\rfloor}\ (b_k^{(1)}+(-1)^k\ b_k^{(2)}), \tag{4.154}$$

where

$$b_k^{(1)}=\sum_{n=0}^{N-1}x_n\ \sin\left[\frac{\pi}{4}+\frac{\pi(2n+1)}{2N}\right]\ \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)\ (2k+1)\right],$$
$$b_k^{(2)}=\sum_{n=0}^{N-1}x_n\ \sin\left[\frac{\pi}{4}+\frac{\pi(2n+1)}{2N}\right]\ \sin\left[\frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)\ (2k+1)\right],$$
$$k=0,1,\ldots,\frac{N}{4}-1. \tag{4.155}$$

The further development involves decompositions of $\{a_k^{(1)}\}$, $\{a_k^{(2)}\}$ and $\{b_k^{(1)}\}$, $\{b_k^{(2)}\}$ as follows. $\{a_k^{(1)}\}$ is decomposed as [42]

$$\begin{aligned}
a_k^{(1)} &= \sum_{n=0}^{\frac{N}{2}-1} x_n \cos\left[\frac{\pi}{4} + \frac{\pi(2n+1)}{2N}\right] \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)(2k+1)\right] \\
&\quad + \sum_{n=\frac{N}{2}}^{N-1} x_n \cos\left[\frac{\pi}{4} + \frac{\pi(2n+1)}{2N}\right] \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)(2k+1)\right] \\
&= \sum_{n=0}^{\frac{N}{2}-1} x_n \cos\left[\frac{\pi}{4} + \frac{\pi(2n+1)}{2N}\right] \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)(2k+1)\right] \\
&\quad - \sum_{n=0}^{\frac{N}{2}-1} x_{\frac{N}{2}+n} \cos\left[\frac{3\pi}{4} + \frac{\pi(2n+1)}{2N}\right] \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)(2k+1)\right] \\
&= \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{2}-1} y_n^{(1)} \cos\left[\frac{\pi}{2(N/2)}\left(2n+1+\frac{N}{4}\right)(2k+1)\right], \\
k &= 0, 1, \ldots, \frac{N}{4}-1,
\end{aligned} \tag{4.156}$$

where

$$\begin{aligned}
y_n^{(1)} &= (x_n + x_{\frac{N}{2}+n}) \cos\frac{\pi(2n+1)}{2N} - \left(x_n - x_{\frac{N}{2}+n}\right) \sin\frac{\pi(2n+1)}{2N}, \\
n &= 0, 1, \ldots, \frac{N}{2}-1.
\end{aligned} \tag{4.157}$$

$\{a_k^{(2)}\}$ is decomposed as [42]

$$\begin{aligned}
a_k^{(2)} &= \sum_{n=0}^{\frac{N}{2}-1} x_n \cos\left[\frac{\pi}{4} + \frac{\pi(2n+1)}{2N}\right] \sin\left[\frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)(2k+1)\right] \\
&\quad + \sum_{n=\frac{N}{2}}^{N-1} x_n \cos\left[\frac{\pi}{4} + \frac{\pi(2n+1)}{2N}\right] \sin\left[\frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)(2k+1)\right] \\
&= \sum_{n=0}^{\frac{N}{2}-1} x_{\frac{N}{2}-1-n} \cos\left[\frac{3\pi}{4} - \frac{\pi(2n+1)}{2N}\right]
\end{aligned}$$

$$\times \sin\left[\frac{3\pi}{2}(2k+1) - \frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)(2k+1)\right]$$

$$+ \sum_{n=0}^{\frac{N}{2}-1} x_{N-1-n} \cos\left[\frac{5\pi}{4} - \frac{\pi(2n+1)}{2N}\right]$$

$$\times \sin\left[\frac{5\pi}{2}(2k+1) - \frac{\pi}{N}\left(2n+1+\frac{N}{4}\right)(2k+1)\right]$$

$$= \frac{\sqrt{2}}{2}(-1)^k \sum_{n=0}^{\frac{N}{2}-1} y_n^{(2)} \cos\left[\frac{\pi}{2(N/2)}\left(2n+1+\frac{N}{4}\right)(2k+1)\right],$$

$$k = 0, 1, \ldots, \frac{N}{4} - 1, \tag{4.158}$$

where

$$y_n^{(2)} = (x_{\frac{N}{2}-1-n} - x_{N-1-n}) \cos\frac{\pi(2n+1)}{2N} - (x_{\frac{N}{2}-1-n} + x_{N-1-n}) \sin\frac{\pi(2n+1)}{2N},$$

$$n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.159}$$

From (4.157) and (4.159), it can be easily verified that

$$y_n^{(2)} = -y_{\frac{N}{2}-1-n}^{(1)}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.160}$$

Substituting (4.156) and (4.158) in (4.151), and using (4.160) we get [42]

$$a_k = (-1)^{\lfloor \frac{k+1}{2} \rfloor} \sum_{n=0}^{\frac{N}{2}-1} (y_n^{(1)} - y_n^{(2)}) \cos\left[\frac{\pi}{2(N/2)}\left(2n+1+\frac{N}{4}\right)(2k+1)\right]$$

$$= (-1)^{\lfloor \frac{k+1}{2} \rfloor} \sum_{n=0}^{\frac{N}{2}-1} (y_n^{(1)} + y_{\frac{N}{2}-1-n}^{(1)}) \cos\left[\frac{\pi}{2(N/2)}\left(2n+1+\frac{N}{4}\right)(2k+1)\right],$$

$$= (-1)^{\lfloor \frac{k+1}{2} \rfloor} \sum_{n=0}^{\frac{N}{2}-1} y_n \cos\left[\frac{\pi}{2(N/2)}\left(2n+1+\frac{N}{4}\right)(2k+1)\right],$$

$$k = 0, 1, \ldots, \frac{N}{4} - 1, \tag{4.161}$$

where $\{y_n\}$ is explicitly given by

$$y_n = \left[(x_n - x_{\frac{N}{2}-1-n}) + (x_{\frac{N}{2}+n} + x_{N-1-n})\right] \cos\frac{\pi(2n+1)}{2N} - \left[(x_n - x_{\frac{N}{2}-1-n}) - (x_{\frac{N}{2}+n} + x_{N-1-n})\right] \sin\frac{\pi(2n+1)}{2N}, \quad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.162}$$

Moreover, the data sequence $\{y_n\}$ has an even symmetry property given by

$$y_n = y_{\frac{N}{2}-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.163}$$

Equation (4.161) corresponding to $\{a_k\}$ is recognized as the $\frac{N}{2}$-point MDCT of $\{y_n\}$.

Following the derivation procedure presented for $\{a_k^{(1)}\}$ given by (4.156) and $\{a_k^{(2)}\}$ given by (4.158), $\{b_k^{(1)}\}$ and $\{b_k^{(2)}\}$ defined by (4.155) after decomposition are, respectively, given by

$$b_k^{(1)} = \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{2}-1} z_n^{(1)} \cos\left[\frac{\pi}{2(N/2)}\left(2n + 1 + \frac{N}{4}\right)(2k+1)\right], \quad k = 0, 1, \ldots, \frac{N}{4} - 1, \tag{4.164}$$

where

$$z_n^{(1)} = (x_n + x_{\frac{N}{2}+n}) \sin\frac{\pi(2n+1)}{2N} + (x_n - x_{\frac{N}{2}+n}) \cos\frac{\pi(2n+1)}{2N}, \quad n = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.165}$$

and

$$b_k^{(2)} = \frac{\sqrt{2}}{2}(-1)^k \sum_{n=0}^{\frac{N}{2}-1} z_n^{(2)} \cos\left[\frac{\pi}{2(N/2)}\left(2n + 1 + \frac{N}{4}\right)(2k+1)\right], \quad k = 0, 1, \ldots, \frac{N}{4} - 1, \tag{4.166}$$

where

$$z_n^{(2)} = -(x_{\frac{N}{2}-1-n} - x_{N-1-n}) \sin\frac{\pi(2n+1)}{2N} - (x_{\frac{N}{2}-1-n} + x_{N-1-n}) \cos\frac{\pi(2n+1)}{2N}, \quad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.167}$$

Again, from (4.165) and (4.167) it can be easily verified that

$$z_n^{(2)} = -z_{\frac{N}{2}-1-n}^{(1)}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.168}$$

Substituting (4.164) and (4.166) into (4.154), and using (4.168) we get [42]

$$\begin{aligned} b_k &= (-1)^{\lfloor \frac{k}{2} \rfloor} \sum_{n=0}^{\frac{N}{2}-1} (z_n^{(1)} + z_n^{(2)}) \cos\left[\frac{\pi}{2(N/2)}\left(2n + 1 + \frac{N}{4}\right)(2k+1)\right] \\ &= (-1)^{\lfloor \frac{k}{2} \rfloor} \sum_{n=0}^{\frac{N}{2}-1} (z_n^{(1)} - z_{\frac{N}{2}-1-n}^{(1)}) \cos\left[\frac{\pi}{2(N/2)}\left(2n + 1 + \frac{N}{4}\right)(2k+1)\right], \\ &= (-1)^{\lfloor \frac{k}{2} \rfloor} \sum_{n=0}^{\frac{N}{2}-1} z_n \cos\left[\frac{\pi}{2(N/2)}\left(2n + 1 + \frac{N}{4}\right)(2k+1)\right], \\ k &= 0, 1, \ldots, \frac{N}{4} - 1, \end{aligned} \tag{4.169}$$

where $\{z_n\}$ is explicitly given by

$$\begin{aligned} z_n = &\left[(x_n - x_{\frac{N}{2}-1-n}) + (x_{\frac{N}{2}+n} + x_{N-1-n})\right] \sin\frac{\pi(2n+1)}{2N} \\ &+ \left[(x_n - x_{\frac{N}{2}-1-n}) - (x_{\frac{N}{2}+n} + x_{N-1-n})\right] \cos\frac{\pi(2n+1)}{2N}, \quad n = 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{4.170}$$

Moreover, the data sequence $\{z_n\}$ has an even anti-symmetry property given by

$$z_n = -z_{\frac{N}{2}-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.171}$$

Equation (4.169) corresponding to $\{b_k\}$ is recognized as the $\frac{N}{2}$-point MDCT of $\{z_n\}$.

The last step involves to derive the complete formulae defining the recursive radix-2 DIF fast MDCT algorithm. Based on (4.147), (4.161), (4.169) and using the algebraic identity (4.153), we have [42]

$$\begin{aligned} c_{2k}^{o} = &(-1)^{\lfloor \frac{k+1}{2} \rfloor} \sum_{n=0}^{\frac{N}{2}-1} \frac{1}{2}\left[y_n + (-1)^k z_n\right] \\ &\times \cos\left[\frac{\pi}{2(N/2)}\left(2n + 1 + \frac{N}{4}\right)(2k+1)\right], \end{aligned}$$

$$c_{2k+1}^{o} = (-1)^{\lfloor \frac{k+1}{2} \rfloor} \sum_{n=0}^{\frac{N}{2}-1} \frac{1}{2} \left[y_n - (-1)^k z_n \right] \times \cos\left[\frac{\pi}{2(N/2)} \left(2n + 1 + \frac{N}{4} \right) (2k+1) \right],$$
$$k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.172}$$

where the data sequence $\{y_n\}$ is given by (4.162) and data sequence $\{z_n\}$ is given by (4.170). Finally, directly calculating the algebraic expressions under both sums of (4.172) for k even and odd, we get [42]

$$c_{2k}^{o} = (-1)^{\lfloor \frac{k+1}{2} \rfloor} \sum_{n=0}^{\frac{N}{2}-1} x_n^{(k)} \cos\left[\frac{\pi}{2(N/2)} \left(2n + 1 + \frac{N}{4} \right) (2k+1) \right],$$
$$c_{2k+1}^{o} = (-1)^{\lfloor \frac{k+1}{2} \rfloor} \sum_{n=0}^{\frac{N}{2}-1} x_{\frac{N}{2}-1-n}^{(k)} \cos\left[\frac{\pi}{2(N/2)} \left(2n + 1 + \frac{N}{4} \right) (2k+1) \right],$$
$$k = 0, 1, \ldots, \frac{N}{4} - 1, \tag{4.173}$$

where

$$x_n^{(k)} = \begin{cases} (x_n - x_{\frac{N}{2}-1-n}) \cos \frac{\pi(2n+1)}{2N} \\ \quad + (x_{\frac{N}{2}+n} + x_{N-1-n}) \sin \frac{\pi(2n+1)}{2N}, & k \text{ is even}, \\ -(x_n - x_{\frac{N}{2}-1-n}) \sin \frac{\pi(2n+1)}{2N} \\ \quad + (x_{\frac{N}{2}+n} + x_{N-1-n}) \cos \frac{\pi(2n+1)}{2N}, & k \text{ is odd}, \end{cases}$$

$$x_{\frac{N}{2}-1-n}^{(k)} = \begin{cases} -(x_n - x_{\frac{N}{2}-1-n}) \sin \frac{\pi(2n+1)}{2N} \\ \quad + (x_{\frac{N}{2}+n} + x_{N-1-n}) \cos \frac{\pi(2n+1)}{2N}, & k \text{ is even}, \\ (x_n - x_{\frac{N}{2}-1-n}) \cos \frac{\pi(2n+1)}{2N} \\ \quad + (x_{\frac{N}{2}+n} + x_{N-1-n}) \sin \frac{\pi(2n+1)}{2N}, & k \text{ is odd}, \end{cases}$$
$$n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.174}$$

Since the data sequence $\{x_{\frac{N}{2}-1-n}^{(k)}\}$ is reversed to $\{x_n^{(k)}\}$, i.e.,

$$x^{(k)}_{\frac{N}{2}-1-n} = x^{(k)}_n, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.175}$$

consequently, it is satisfying to calculate $\{x^{(k)}_n\}$ and $\{x^{(k)}_{\frac{N}{2}-1-n}\}$ only for $n = 0, 1, \ldots, \frac{N}{4} - 1$.

The radix-2 recursive DIF fast MDCT algorithm defined by (4.173)–(4.175) decomposes the N-point MDCT into two $\frac{N}{2}$-point MDCTs. The decomposition process can recursively be repeated until the lower-order even-length (4-point or 6-point) MDCTs remain. Equation (4.174) defines the block of $\frac{N}{4}$ Givens–Jacobi rotations. The generalized signal flow graph of the recursive radix-2 DIF fast algorithm for the forward MDCT computation [42] for $N = 8$ is shown in Fig. 4.11. The backward MDCT computation can be realized simply by reversing the signal flow graph for the forward MDCT computation and performing inverse operations.

The total computational complexity of the recursive radix-2 DIF fast MDCT algorithm is given by Wu et al. [42]

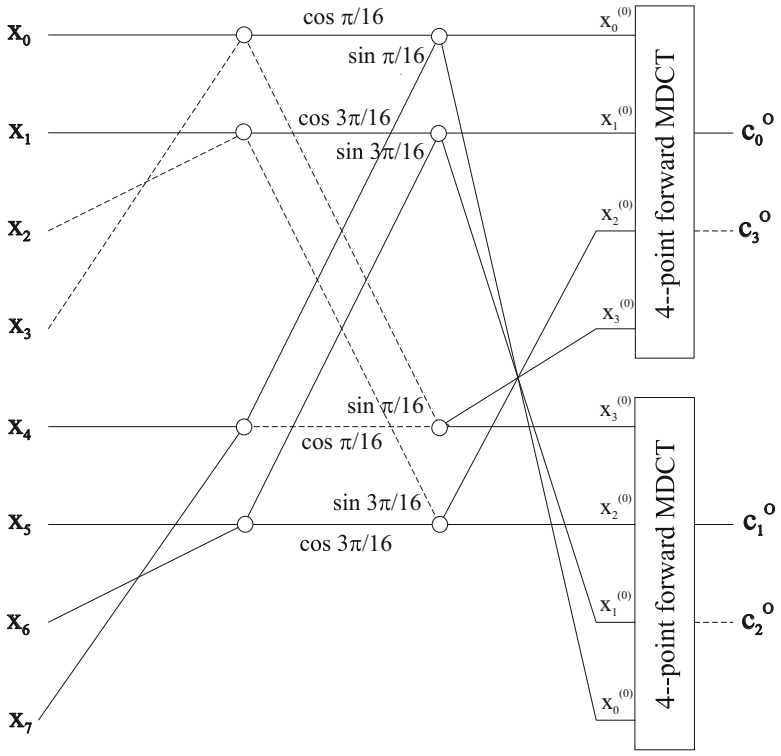

Fig. 4.11 Generalized signal flow graph of the recursive radix-2 DIF fast algorithm [42] for the forward MDCT computation for $N = 8$

$$M_N = 2 \times M_{\frac{N}{2}} + \frac{3N}{4}, \qquad A_N = 2 \times A_{\frac{N}{2}} + \frac{5N}{4}. \tag{4.176}$$

Note 9 Similarly as for the evenly stacked MDCT/MDST (see Sect. 4.2), generalized signal flow graphs for the MDCT/MDST computation in the oddly stacked system shown in Figs. 4.5, 4.6, 4.7, 4.8, and 4.9, DCT-IV-based, fast DCT-IV/(S)DCT-II-based, unified evenly/oddly stacked MDCT/MDCT algorithms as well as (recursive) fast DIF/DIT mixed-radix and radix-2 algorithms correspond to sparse (block) matrix factorizations of the MDCT matrix. Based on the relation between the MDCT and MDST block transforms given by (4.5) we can obtain sparse matrix factorizations of the corresponding MDST matrix.

4.3.8 *Oddly Stacked MDCT/MDST Implementations Based on the Recursive/Regressive Filter Structures*

Quite different class of algorithms for an efficient implementation of the forward/backward MDCT in the oddly stacked system or equivalently of the MLT are algorithms based on recursive or regressive filter structures [52–63]. Although these recursive algorithms are not so efficient in terms of the arithmetic complexity, they can be represented by simple regular regressive filter structures of the same type for the variable- or general-lengths forward/backward MDCT (MLT) computation. Thus, recursive algorithms provide efficient online schemes particularly suitable for parallel VLSI implementations.

Principally, the forward or backward MDCT (MLT) kernels are converted to recursive (or recurrent) equations based on:

- Recurrence formulae for Chebyshev polynomials of the second and third kinds [87]. A typical recursive forward/backward MDCT (MLT) implementation [53] follows closely the DCT-II-based approach [87].
- Sinusoidal recursive formulae exploiting the Goertzel recursive formula [81]. Such a typical recursive implementation [61, 63] follows closely the DCT-II-based approach [76, 77].
- Clenshaw's recurrence formula (used for upward or downward ordering) [71]. A typical recursive implementation [62] follows closely the DCT-II-based approach [71].
- Goertzel digital filters, where the forward and backward MDCT are expressed in the form of discrete time domain convolution sums [54].
- Recurrence formulae for the cosine and sine functions [55].

Now, we know that the N-point MDCT or MLT using the permutation (4.80) or (4.101) can be converted to the DCT-IV of half size which may be subsequently converted to the DCT-II of the same size at the cost of additional $\frac{N}{2}$ pre-multiplications and $\frac{N}{2} - 1$ recursive post-additions. Following this approach the recursive/regressive filter structures have been developed [52, 58–60]. On the other hand, the DCT-

IV-based recursive/regressive filter structures are presented in [56, 57]. Since the DCT-IV is self-inverse, both the forward and backward MDCT computations are implemented by an identical recursive/regressive filter structure.

In the following subsections typical representative recursive MDCT (MLT) implementations based on recursive/regressive filter structures are discussed in more detail. Recursive (recurrence) formulae are applied directly to the MDCT (MLT) kernel. We note that in the derivation of algorithms [55, 61, 63], the definitions of the forward and backward MLT are used. Therefore, we recall that the forward and backward MLT are, respectively, defined as [2]

$$c_k^{\text{MLT}} = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{M}\left(n+\frac{M+1}{2}\right)\left(k+\frac{1}{2}\right)\right], \qquad k = 0, 1, \ldots, M-1, \tag{4.177}$$

$$\hat{x}_n^{\text{MLT}} = \sum_{k=0}^{M-1} c_k^{\text{MLT}} \cos\left[\frac{\pi}{M}\left(n+\frac{M+1}{2}\right)\left(k+\frac{1}{2}\right)\right], \qquad n = 0, 1, \ldots, N-1, \tag{4.178}$$

where $\{x_n\}$ is the windowed input data sequence, $N = 2M$ and M is the number of MLT coefficients $\{c_k^{\text{MLT}}\}$. $\{\hat{x}_n^{\text{MLT}}\}$ is the time domain aliased data sequence.

4.3.8.1 Recursive MLT (MDCT) Algorithm [61, 63]

At first, we remind the recurrence formulae for the cosine and sine functions [52]

$$\begin{aligned} \cos[r\ \theta_k] &= 2\ \cos[(r-1)\ \theta_k]\ \cos\theta_k - \cos[(r-2)\ \theta_k], \\ \sin[r\ \theta_k] &= 2\ \sin[(r-1)\ \theta_k]\ \cos\theta_k - \sin[(r-2)\ \theta_k], \quad \theta_k = \frac{\pi\ k}{N}. \end{aligned} \tag{4.179}$$

With respect to the Goertzel recursive algorithm [81], defining a relation [76]

$$v_m\ \sin\theta_k = \sum_{n=m}^{N-1} x_n\ \sin(n-m+1)\ \theta_k, \tag{4.180}$$

and using the recurrence formula for the sine function (4.179) with $r = n - m + 1$, we have

$$v_m\ \sin\theta_k = x_m\ \sin\theta_k + \sum_{n=m+1}^{N-1} x_n\ \sin(n-m+1)\ \theta_k$$

$$
\begin{aligned}
&= x_m \ \sin\theta_k + \sum_{n=m+1}^{N-1} x_n \ [2 \ \cos\theta_k \ \sin(n-m) \ \theta_k - \sin(n-m-1) \ \theta_k] \\
&= x_m \ \sin\theta_k + 2 \ \cos\theta_k \sum_{n=m+1}^{N-1} x_m \ \sin(n-m) \ \theta_k \\
&\quad - \sum_{n=m+2}^{N-1} x_n \ \sin(n-m-1) \ \theta_k \\
&= x_m \ \sin\theta_k + 2 \ \cos\theta_k \ v_{m+1} \ \sin\theta_k - v_{m+2} \ \sin\theta_k, \\
m &= N-1, N-2, \ldots, 1, 0.
\end{aligned} \tag{4.181}
$$

From (4.181) we directly get the recursive equation defined as [76]

$$
v_m = x_m + 2 \ \cos\theta_k \ v_{m+1} - v_{m+2}, \quad m = N-1, N-2, \ldots, 1, 0, \tag{4.182}
$$

with the initial conditions: $v_{N-1} = x_{N-1}$ and $v_N = v_{N+1} = 0$ for $m = N-1$.

In order to obtain the recursive transfer function for the forward MLT given by (4.177), let us denote

$$
\theta_k = \frac{\pi}{M} \left(k + \frac{1}{2}\right), \qquad k = 0, 1, \ldots, M-1, \tag{4.183}
$$

and multiplying both sides of (4.177) by $\sin\theta_k$, we have [61, 63]

$$
\begin{aligned}
c_k^{\text{MLT}} \ \sin\theta_k &= \sum_{n=0}^{N-1} x_n \ \cos\left[\left(n + \frac{M+1}{2}\right) \ \theta_k\right] \ \sin\theta_k \\
&= x_0 \ \cos\left[(M+1)\frac{\theta_k}{2}\right] \ \sin\theta_k + \sum_{n=1}^{N-1} x_n \left\{\cos(n \ \theta_k) \ \cos\left[(M+1)\frac{\theta_k}{2}\right]\right. \\
&\quad \left. - \sin(n \ \theta_k) \ \sin\left[(M+1)\frac{\theta_k}{2}\right]\right\} \ \sin\theta_k \\
&= x_0 \ \cos\left[(M+1)\frac{\theta_k}{2}\right] \ \sin\theta_k + + \sum_{n=1}^{N-1} x_n \left\{\cos\theta_k \ \cos\left[(M+1)\frac{\theta_k}{2}\right]\right. \\
&\quad \left. - \sin\theta_k \ \sin\left[(M+1)\frac{\theta_k}{2}\right]\right\} \ \sin(n \ \theta_k) \\
&\quad - \sum_{n=1}^{N-1} x_n \ [\sin(n \ \theta_k) \ \cos\theta_k - \cos(n \ \theta_k) \ \sin\theta_k] \ \cos\left[(M+1)\frac{\theta_k}{2}\right]
\end{aligned}
$$

$$= x_0 \cos\left[(M+1)\frac{\theta_k}{2}\right] \sin\theta_k + \cos\left[(M+3)\frac{\theta_k}{2}\right] \sum_{n=1}^{N-1} x_n \sin(n\,\theta_k)$$

$$-\cos\left[(M+1)\frac{\theta_k}{2}\right] \sum_{n=2}^{N-1} x_n \sin[(n-1)\,\theta_k]$$

$$= x_0 \cos\left[(M+1)\frac{\theta_k}{2}\right] \sin\theta_k + v_1 \cos\left[(M+3)\frac{\theta_k}{2}\right] \sin\theta_k$$

$$-v_2 \cos\left[(M+1)\frac{\theta_k}{2}\right] \sin\theta_k. \tag{4.184}$$

From (4.184) we directly obtain the recursive transfer function for the forward MLT as [61, 63]

$$c_k^{\text{MLT}} = x_0 \cos\left[(M+1)\frac{\theta_k}{2}\right] + v_1 \cos\left[(M+3)\frac{\theta_k}{2}\right] - v_2 \cos\left[(M+1)\frac{\theta_k}{2}\right]$$

$$= (x_0 - v_2) \cos\left[(M+1)\frac{\theta_k}{2}\right] + v_1 \cos\left[(M+3)\frac{\theta_k}{2}\right],$$

$$k = 0, 1, \ldots, M-1. \tag{4.185}$$

From (4.182) for $m = 0$ we have

$$x_0 - v_2 = v_0 - v_1\, 2\, \cos\theta_k. \tag{4.186}$$

Substituting (4.186) in (4.185), then using the trigonometric identity $\cos(\alpha - \beta) = 2\cos\alpha\, \cos\beta - \cos(\alpha+\beta)$, where $\alpha = (M+1)\,\frac{\theta_k}{2}$ and $\beta = \theta_k$, after some algebraic manipulations we finally obtain [61, 63]

$$c_k^{\text{MLT}} = v_0 \cos\left[(M+1)\frac{\theta_k}{2}\right] - v_1 \cos\left[(M-1)\frac{\theta_k}{2}\right], \qquad k = 0, 1, \ldots, M-1. \tag{4.187}$$

Equations (4.182) and (4.187) define the recursive algorithm for the forward MLT (MDCT) computation. To compute one MLT (MDCT) coefficient c_k^{MLT}, we need to recursively generate $\{v_m\}$ according to (4.182). At Nth step, the kth MLT (MDCT) coefficient is evaluated according to (4.187). This requires $N+1$ multiplications and $2N-1$ additions.

The corresponding regressive filter structure for the forward MLT (MDCT) computation is shown in Fig. 5.3a in Chap. 5. The input data sequence $\{x_n\}$ is in reverse order.

Now, consider the backward MLT given by (4.178). Similarly, defining a relation [77]

$$v_m \ \sin\theta_n = \sum_{k=m}^{M-1} c_k^{\text{MLT}} \ \sin(k-m+1)\ \theta_n, \tag{4.188}$$

using the recurrence formula for the sine function (4.179) with $r = k - m + 1$, and following the derivation procedure (4.181), we get the recursive equation defined as [77]

$$v_m = c_m^{\text{MLT}} + 2\ \cos\theta_n\ v_{m+1} - v_{m+2}, \quad m = M-1, M-2, \ldots, 1, 0, \tag{4.189}$$

with initial conditions: $v_{M-1} = c_{M-1}^{\text{MLT}}$ and $v_M = v_{M+1} = 0$ for $m = M-1$.

In order to obtain the recursive transfer function for the backward MLT given by (4.178), let us denote

$$\theta_n = \frac{\pi}{M}\left(n + \frac{M+1}{2}\right), \qquad n = 0, 1, \ldots, N-1, \tag{4.190}$$

and multiplying both sides of (4.178) by $\sin\theta_n$, we have [61, 63]

$$\begin{aligned}
\hat{x}_n^{\text{MLT}} \ \sin\theta_n &= \sum_{k=0}^{M-1} c_k^{\text{MLT}} \ \cos\left[\left(k+\frac{1}{2}\right)\ \theta_n\right] \sin\theta_n \\
&= c_0^{\text{MLT}} \ \cos\frac{\theta_n}{2} \ \sin\theta_n + \sum_{k=1}^{M-1} c_k \ \left\{\cos(k\ \theta_n)\ \cos\frac{\theta_n}{2}\right. \\
&\quad \left. - \sin(k\ \theta_n)\ \sin\frac{\theta_n}{2}\right\} \ \sin\theta_n \\
&= c_0^{\text{MLT}} \ \cos\frac{\theta_n}{2} \ \sin\theta_n \\
&\quad + \sum_{k=1}^{M-1} c_k^{\text{MLT}} \ \left[\cos\theta_n \ \cos\frac{\theta_n}{2} - \sin\theta_n \ \sin\frac{\theta_n}{2}\right] \sin(k\ \theta_n) \\
&\quad - \sum_{k=1}^{M-1} c_k^{\text{MLT}} \ [\sin(k\ \theta_n)\ \cos\theta_n - \cos(k\ \theta_n)\ \sin\theta_n]\ \cos\frac{\theta_n}{2} \\
&= c_0^{\text{MLT}} \ \cos\frac{\theta_n}{2} \ \sin\theta_n + \cos\frac{3\theta_n}{2} \ \sum_{k=1}^{M-1} c_k^{\text{MLT}} \ \sin(k\ \theta_n) \\
&\quad - \cos\frac{\theta_n}{2} \ \sum_{k=2}^{M-1} c_k \ \sin[(k-1)\ \theta_n]
\end{aligned}$$

$$= c_0^{\text{MLT}} \cos\frac{\theta_n}{2} \sin\theta_n + v_1 \cos\frac{3\theta_n}{2} \sin\theta_n$$

$$-v_2 \cos\frac{\theta_n}{2} \sin\theta_n. \tag{4.191}$$

From (4.191) we directly obtain the recursive transfer function for the backward MLT as [61, 63]

$$\hat{x}_n^{\text{MLT}} = c_0^{\text{MLT}} \cos\frac{\theta_n}{2} + v_1 \cos\frac{3\theta_k}{2} - v_2 \cos\frac{\theta_n}{2}$$

$$= (c_0^{\text{MLT}} - v_2) \cos\frac{\theta_n}{2} + v_1 \cos\frac{3\theta_n}{2}, \qquad n = 0, 1, \ldots, N-1. \tag{4.192}$$

From (4.189) for $m = 0$ we have

$$c_0^{\text{MLT}} - v_2 = v_0 - v_1\, 2 \cos\theta_n. \tag{4.193}$$

Similarly, substituting (4.193) into (4.192), then using the trigonometric identity $\cos(\alpha - \beta) = 2\cos\alpha \cos\beta - \cos(\alpha + \beta)$, where $\alpha = \theta_n$ and $\beta = \frac{\theta_n}{2}$, we finally obtain [61, 63]

$$\hat{x}_n^{\text{MLT}} = (v_0 - v_1) \cos\frac{\theta_n}{2}, \qquad n = 0, 1, \ldots, N-1. \tag{4.194}$$

Equations (4.189) and (4.194) define the recursive algorithm for the backward MLT (MDCT) computation. To compute one time domain aliased sample $\hat{x}_n^{\text{MLT}}$, we need recursively generate $\{v_m\}$ according to (4.189). At Mth step, the time domain aliased sample is evaluated according to (4.194). This requires M multiplications and $2M-1$ additions. Exploiting the symmetry property of $\{\hat{x}_n^{\text{MLT}}\}$ (see Chap. 3), i.e., $\hat{x}_n^{\text{MLT}} = -\hat{x}_{\frac{N}{2}-1-n}^{\text{MLT}}$ and $\hat{x}_{\frac{N}{2}+n}^{\text{MLT}} = \hat{x}_{N-1-n}^{\text{MLT}}$, for $n = 0, 1, \ldots, \frac{N}{4} - 1$, it is sufficient to compute only $\frac{N}{2}$ samples.

The corresponding regressive filter structure for the backward MLT (MDCT) computation is shown in Fig. 5.3b in Chap. 5. The MLT (MDCT) coefficients are in reverse order.

4.3.8.2 Recursive MLT (MDCT) Algorithm [55]

Simple recursive algorithms [55] for the variable-length forward and backward MLT (MDCT) computation are based on a recursively generating the MLT (MDCT) kernel using the recurrence formulae (4.179).

Consider the forward MLT given by (4.177). Defining θ_k similarly as in (4.183), the MLT kernel is first rewritten into the equivalent form as [55]

$$f_n = \cos\left[\left(n + \frac{M+1}{2}\right)\ \theta_k\right], \qquad n = 0, 1, \ldots, N-1. \tag{4.195}$$

Applying the cosine recurrence formula (4.179) for $r = n$ to (4.195) we obtain the recurrence equation for generating the MLT kernel as

$$f_n = 2\ \cos\theta_k\, f_{n-1} - f_{n-2}, \qquad n = 1, 2, \ldots, N-1. \tag{4.196}$$

From (4.195) for $n = 0$ we obtain the initial conditions

$$f_0 = \cos\left[\frac{M+1}{2}\ \theta_k\right], \qquad f_{-1} = \cos\left[\frac{M-1}{2}\ \theta_k\right]. \tag{4.197}$$

Then, the computation of the forward MLT (MDCT) is defined as [55]

$$c_k^{\text{MLT}} = x_0\, f_0 + \sum_{n=1}^{N-1} x_n\, f_n, \qquad k = 0, 1, \ldots, M-1, \tag{4.198}$$

where the MLT kernel given by $\{f_n\}$, $n = 1, 2, \ldots, N-1$, is recursively generated according to (4.196) with initial conditions (4.197). The computation of one MLT (MDCT) coefficient c_k^{MLT} requires $2N-1$ multiplications and $2N-1$ additions.

Now, consider the backward MLT given by (4.178). Defining θ_n similarly as in (4.190), the MLT kernel is rewritten into the equivalent form as [55]

$$g_k = \cos\left[\left(k + \frac{1}{2}\right)\ \theta_n\right], \qquad k = 0, 1, \ldots, M-1. \tag{4.199}$$

Following the same derivation procedure as used for the forward MLT, the computation of the backward MLT (MDCT) is defined as [55]

$$\hat{x}_n^{\text{MLT}} = c_0^{\text{MLT}}\ g_0 + \sum_{k=1}^{M-1} c_k\ g_k, \qquad n = 0, 1, \ldots, N-1, \tag{4.200}$$

where

$$g_k = 2\ \cos\theta_n\ g_{k-1} - g_{k-2}, \qquad k = 1, 2, \ldots, M-1. \tag{4.201}$$

with initial conditions given by

$$g_0 = \cos\frac{\theta_n}{2}, \qquad g_{-1} = \cos\left(-\frac{\theta_n}{2}\right) = \cos\frac{\theta_n}{2}. \tag{4.202}$$

The computation of one time domain aliased sample $\hat{x}_n^{\text{MLT}}$ requires $2M-1$ multiplications and $2M-1$ additions.

For a potential hardware implementation, in [55] additionally the recursive algorithm [61, 63] and recursive algorithm [55] have been investigated in terms of the required number of bits for the fixed-point data representation during the backward MLT (MDCT) computation on data block sizes with $N = 1920$. Indeed, due to the nature of recursion, numerical values are accumulated very fast, and this fact has an impact for implementation on a hardware architecture with the limited number of bits for data representation. It was shown in [55] that the recursive algorithm [61, 63] frequently exceeded a maximum limit of 24 bits (actually it required up to 31 bits particularly when the magnitude $2\ \cos\theta_n$ reached the value of 2 in the inner loop of recursion), while the recursive algorithm [55] was able to process the same data with only 24 bits. Consequently, the computation of the MLT (MDCT) by the recursive algorithm [55] can be performed on a reduced bit-width architecture without a loss of accuracy.

Note 10 Although similar recursive algorithms have been also derived for the corresponding MDST in [55, 61, 63], using the relation between the MDST and the MDCT block transforms given by (4.43), the computation of MDST can be realized by the MDCT recursive algorithms.

4.4 Fast Algorithm for the ELT Computation

Essentially, the ELT is the MLT but with longer basis functions, or in general, with the basis functions of arbitrary length [2, 48–50]. We recall that the forward ELT as a block transform is defined as [2, 48, 50]

$$c_k^{\text{ELT}} = \sum_{n=0}^{2KM-1} x_n \cos\left[\frac{\pi}{M}\left(n + \frac{M+1}{2}\right)\left(k + \frac{1}{2}\right)\right], \qquad k = 0, 1, \ldots, M-1, \tag{4.203}$$

where $\{x_n\}$ is the windowed input data sequence, $K > 1$ is the overlapping factor being a positive integer. The overlapping factor K actually specifies how many pairs of overlapped input data blocks are used to compute the ELT coefficients. When $K = 1$, the ELT becomes the MLT.

On the other hand, the backward ELT as a block transform is defined as [2, 48, 50]

$$\hat{x}_n^{\text{ELT}} = \sum_{k=0}^{M-1} c_k^{\text{ELT}} \cos\left[\frac{\pi}{M}\left(n + \frac{M+1}{2}\right)\left(k + \frac{1}{2}\right)\right], \qquad n = 0, 1, \ldots, KM-1, \tag{4.204}$$

where $\{\hat{x}_n^{\text{ELT}}\}$ is the time domain aliased data sequence. We note that the normalization factors $\sqrt{\frac{2}{M}}$ in (4.203) and (4.204) for simplicity are omitted. For the efficient

forward and backward ELT computation the fast algorithms were developed in [2, 48–50], and they are discussed in the following text.

Consider the forward ELT given by (4.203). Using the change of variables $n = \frac{3M}{2} + s$, and defining

$$g_s \equiv x_{\frac{3M}{2}+s}, \tag{4.205}$$

we can write (4.203) in the form [2]

$$c_k^{\text{ELT}} = -\sum_{s=-\frac{3M}{2}}^{(4K-3)\frac{M}{2}-1} g_s \cos\left[\frac{\pi}{M}\left(s+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\right],$$

$$k = 0, 1, \ldots, M-1. \tag{4.206}$$

Assuming that $\{g_s\}$ defined by (4.205), but with $g_s = 0$ if $s < -\frac{3M}{2}$ or $s > (4K-3)\frac{M}{2}$, (4.206) may be written in the form

$$c_k^{\text{ELT}} = -\sum_{l=0}^{2K} \sum_{s=(l-2)M}^{(l-1)M-1} g_s \cos\left[\frac{\pi}{M}\left(s+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\right],$$

$$k = 0, 1, \ldots, M-1. \tag{4.207}$$

Further, splitting the first sum with $2K+1$ terms in (4.207) into two sums, specifically for $l = 2s$ and $l = 2s+1$, we get

$$c_k^{\text{ELT}} = -\sum_{r=0}^{K} \sum_{s=(2r-2)M}^{(2r-1)M-1} g_s \cos\left[\frac{\pi}{M}\left(s+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\right] -$$

$$-\sum_{r=0}^{K-1} \sum_{s=(2r-1)M}^{2rM-1} g_s \cos\left[\frac{\pi}{M}\left(s+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\right],$$

$$k = 0, 1, \ldots, M-1. \tag{4.208}$$

By substituting $s + 2(r-1)M$, and then $2rM - 1 - s$ for s in (4.208), we find that the modulating cosines satisfy the following relations [2, 50]:

$$\cos\left[\frac{\pi}{M}\left(s+2(r-1)M+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\right]$$

$$= (-1)^{r-1} \cos\left[\frac{\pi}{M}\left(s+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\right],$$

$$\cos\left[\frac{\pi}{M}\left(2rM-1-s+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\right]$$
$$=(-1)^r\cos\left[\frac{\pi}{M}\left(s+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\right]. \tag{4.209}$$

Finally, using (4.209), applying the change of variables $n=s-(2r-2)M$ in the first sum of (4.208), and the change of variables $n=2rM-1-s$ in the second sum, we obtain [2]

$$c_k^{\text{ELT}}=\sum_{n=0}^{M-1}u_n\cos\left[\frac{\pi}{M}\left(n+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\right],$$
$$k=0,1,\ldots,M-1, \tag{4.210}$$

where the data sequence $\{u_n\}$ is given by Malvar [2]

$$u_n=\sum_{r=0}^{K}(-1)^r\,g_{n+(2r-2)M}+\sum_{r=0}^{K-1}(-1)^{r-1}\,g_{2rM-1-n},$$
$$n=0,1,\ldots,M-1. \tag{4.211}$$

The cosine transform kernel in (4.210) is recognized as an M-point DCT-IV. Thus, using (4.205) and (4.211) the KM-point forward ELT of $\{x_n\}$ is converted to the M-point DCT-IV of $\{u_n\}$. Moreover, the windowing procedure for a given value of $K>1$ using (4.205) and (4.211) is decomposed to a product of K orthogonal butterfly matrices intermixed with delays. The ELT windowing function design method as well as C computer program to generate required butterfly angles can be found in [2, 50]. The backward ELT computation can be realized by reversing the fast forward ELT structure and performing the inverse operations. Generalized signal flow graphs both for the forward and backward ELT computation are presented in [2]. The required DCT-IV fast algorithms/computational structures are presented in Appendix C.2.

The total computational complexity of the fast forward/backward ELT, where $M=2^m$, is given [2]

$$M_{KM}=\frac{M}{2}(2K+m+3),\qquad A_{KM}=\frac{M}{2}(2K+m+1). \tag{4.212}$$

4.4.1 Fast ELT for K = 2

The forward ELT given by (4.203) for $K=2$ corresponds to

$$c_k^{\text{ELT}} = \sum_{n=0}^{4M-1} x_n \cos\left[\frac{\pi}{M}\left(n+\frac{M+1}{2}\right)\left(k+\frac{1}{2}\right)\right]$$
$$= \sum_{n=0}^{M-1} u_n \cos\left[\frac{\pi}{M}\left(n+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\right],$$
$$k = 0, 1, \ldots, M-1, \tag{4.213}$$

and assuming that $g_s = 0$ if $s \notin [-\frac{3M}{2}, \frac{5M}{2} - 1]$, the data sequence $\{u_n\}$ given by (4.211) can be written as [2]

$$u_i = \sum_{r=0}^{2}(-1)^r\, g_{i+(2r-2)M} + \sum_{r=0}^{1}(-1)^{r-1}\, g_{2rM-1-i}, \qquad i = 0, 1, \ldots, M-1.$$

Let us consider two halves of $\{u_i\}$ simultaneously, with indices $n = \frac{M}{2} + i$ and $n = \frac{M}{2} - 1 - i$. Then, we have

$$u_{\frac{M}{2}+i} = g_{i-\frac{3M}{2}} - g_{i+\frac{M}{2}} - g_{-\frac{M}{2}-1-i} + g_{\frac{3M}{2}-1-i},$$
$$u_{\frac{M}{2}-1-i} = -g_{\frac{M}{2}-1-i} + g_{\frac{5M}{2}-1-i} - g_{i-\frac{M}{2}} + g_{\frac{3M}{2}+i}, \qquad i = 0, 1, \ldots, \frac{M}{2} - 1.$$

Using (4.205), we finally obtain

$$u_{\frac{M}{2}+n} = x_n - x_{2M+n} - x_{M-1-n} + x_{3M-1-n},$$
$$u_{\frac{M}{2}-1-n} = -x_{2M-1-n} + x_{4M-1-n} - x_{M+n} + x_{3M+n},$$
$$n = 0, 1, \ldots, \frac{M}{2} - 1. \tag{4.214}$$

Computer programs in C language for the fast forward/backward ELT computation can be found in [2].

4.5 Fast Algorithms for the MCLT Computation

The MCLT [3] is the complex block transform mapping overlapped blocks of real-valued signal into blocks of complex-valued transform coefficients. Its real part is the MLT, or equivalently the oddly stacked MDCT, and its imaginary part is the corresponding oddly stacked MDST. Actually, the MDCT (MLT) and the MDST are, respectively, the real and imaginary components of the O^2DFT [23]. In general, based on the relation between the MDST and the MDCT given by (4.43), the MDST computation can be realized by any existing fast MDCT computational

structure with simple pre-processing of data sequences. Consequently, any fast MDCT algorithm discussed in Sect. 4.3 may be used for the MCLT computation. But, a fast MDCT computational structure should be applied sequentially to obtain the MCLT.

A number of fast algorithms were developed for the computation of the MCLT or more accurately, for the simultaneous (online) MDCT/MDST computation [3, 18, 64–70]. The conventional approach is to decompose the argument of MCLT kernel and map it into the complex DFT of the same size with complex pre- and post-multiplications [70]. However, this approach involves redundant arithmetic operations. To reduce the arithmetic complexity, the simultaneous MDCT/MDST computation is mapped into the real-valued DFT/FFT [68], type-IV generalized DFT (GDFT-IV) of real-valued data sequences or alternatively, into type-IV generalized discrete Hartley transform (GDHT-IV) [18], type-II GDHT (GDHT-II) [69], DCT-IV/DST-IV [3], DCT-II/DST-II [67], or it is realized by the DCT-IV/DCT-II combination [65, 66]. A direct recursive fast radix-2 MCLT algorithm [64] is based on the recursive radix-2 DIF MDCT algorithm [42]. In particular, exploiting the explicit form of the sine windowing function combined with the MCLT kernel, several fast MCLT algorithms with incorporated sine windowing function were derived: the real-valued DFT/FFT-based [68], DCT-II-based [67], and GDHT-II-based [69].

In the theory of fast MDCT/MDST algorithms it is well known that by a composition of simple permutations applied to the input data sequence, an N-point MDCT/MDST can always be converted to $\frac{N}{2}$-point DCT-IV/DST-IV (whereby DST-IV may always be converted back to DCT-IV with simple pre-processing of data sequences). Note that each $\frac{N}{2}$-point DCT-IV may be alternatively mapped into the $\frac{N}{4}$-point complex DFT with identical pre- and post-rotation stages [25].

4.5.1 Definitions of MCLT Block Transforms

Let $\{x_n\},\ n = 0, 1, \ldots, N-1$ represent an input data sequence, and N is assumed to be an even integer. Consider the forward MDCT given by (4.39) and the forward MDST given by (4.41) with $N = 2M$. Then, the forward MCLT as a block transform is defined as [3]

$$p_k = \overset{o}{c}_k - i\,\overset{o}{s}_k, \qquad i = \sqrt{-1}, \qquad k = 0, 1, \ldots, M-1, \tag{4.215}$$

where $\{\overset{o}{c}_k\}$ and $\{\overset{o}{s}_k\}$ are, respectively, the MDCT and MDST coefficients.

A reconstruction formula of the backward MCLT block transform is not unique. There is a reconstruction formula from the real part only, as well as one from the imaginary part only [3]. However, the best reconstruction formula is the average of those from real and imaginary parts (preferred) defined as [3]

$$y_n = \frac{1}{2}\left(\hat{x}_n^{O-\text{MDCT}} + \hat{x}_n^{O-\text{MDST}}\right), \qquad n = 0, 1, \ldots, 2M-1, \tag{4.216}$$

where $\{\hat{x}_n^{O-\text{MDCT}}\}$ and $\{\hat{x}_n^{O-\text{MDST}}\}$ are, respectively, the time domain aliased data sequences recovered by the backward MDCT given by (4.40) and the backward MDST given by (4.42). Therefore, for the backward MCLT computation we need to use a fast MDCT algorithm for the backward MDCT/MDST computation to obtain the time domain aliased data sequences $\{\hat{x}_n^{O-\text{MDCT}}\}$ and/or $\{\hat{x}_n^{O-\text{MDST}}\}$.

Note 11 It is important to emphasize that if we use the reconstruction formula (4.216), then the original data sequence $\{x_n\}$ is perfectly reconstructed even without the windowing, overlap, and add procedure, i.e., $y_n = x_n$ in (4.216). It means that the windowing procedure in the fast MCLT computation is redundant. However, when the windowing procedure nevertheless is used, then $y_n = w_n^2 x_n$, $n = 0, 1, \ldots, 2M-1$ (see Chap. 3).

4.5.2 (G)DFT/FFT-Based Fast Algorithms

The DFT/FFT-based fast MCLT algorithm [70] directly maps the $2M$-point MCLT kernel into $2M$-point complex DFT/FFT. Consequently, it is not so efficient in terms of the arithmetic complexity. In order to reduce the arithmetic complexity, the DFT/FFT-based fast MCLT algorithm developed in [68] is based on $2M$-point real-valued DFT/FFT with the incorporated sine windowing function. As an alternative, the fast MCLT algorithm [18] is based on the type-IV GDFT (DFT-IV) of real-valued data sequences [74]. Both algorithms are discussed in the following subsections.

4.5.2.1 Real-Valued DFT/FFT-Based Fast Algorithm [68]

At first, the sine windowing function can be written as a sum of two exponentials as [68]

$$\sin\left[\frac{\pi(2n+1)}{4M}\right] = \frac{i}{2}\left(e^{-i\frac{\pi(2n+1)}{4M}} - e^{-i\frac{\pi(-2n-1)}{4M}}\right), \qquad k = 0, 1, \ldots, 2M-1. \tag{4.217}$$

Then (4.215) with the normalization factor $\sqrt{\frac{2}{M}}$ is mapped into a real-valued DFT as

$$p_k = \sqrt{\frac{2}{M}} \sum_{n=0}^{2M-1} x_n \sin\left[\frac{\pi(2n+1)}{4M}\right] e^{-i\frac{\pi(2n+1+M)(2k+1)}{4M}}, \quad k = 0, 1, \ldots, M-1. \tag{4.218}$$

Combining (4.217) and (4.218), after some algebraic manipulations we get [68]

$$p_k = i\, d_k\, f_k + d_{k+1}\, f_{k+1}, \tag{4.219}$$

where

$$d_k = e^{-i\,\frac{\pi}{4}(2k+1)}\, e^{-i\,\frac{2\pi k}{4M}}, \tag{4.220}$$

and

$$f_k = \sqrt{\frac{1}{2M}} \sum_{n=0}^{2M-1} x_n\, e^{-i\,\frac{2\pi nk}{2M}}. \tag{4.221}$$

Equation (4.221) is a normalized $2M$-point DFT/FFT of the real-valued input data sequence $\{x_n\}$. From (4.219), (4.220) and (4.221) it follows that the MCLT coefficients $\{p_k\}$ are obtained by first computing the FFT of $\{x_n\}$, and then performing the additional operations with factors $\{d_k\}$ given by (4.220). Since factors $\{d_k\}$ satisfy $|d_k| = 1$, they actually correspond to Givens–Jacobi rotations (see Appendix F.4).

One of the advantages of the DFT/FFT-based fast MCLT algorithm [68] is that it does not require data shuffling. Another positive aspect is that the factors $\{d_k\}$ may be computed recursively. This fact can take advantage in a potential hardware implementation. Specifically, substituting $k-1$ for k in (4.220), after some manipulations we get [68]

$$d_k = d_{k-1}\, e^{-i\,\frac{2\pi(M+1)}{4M}}, \qquad k = 1, 2, \ldots, M. \tag{4.222}$$

The total computational complexity of the DFT/FFT-based fast MCLT algorithm [68] is given by the complexity of $2M$-point FFT of real-valued data sequences plus the complexity of (4.219) requiring $M+1$ complex multiplications and M complex additions. Each complex multiplication corresponds to the Givens–Jacobi rotation. Additionally, f_0 and f_M are real, factors d_0 and d_M have identical real and imaginary parts except for sign, and factor $c_{\frac{M}{2}} = e^{-i\,\frac{\pi}{2}(\frac{M}{2}+1)}$ is either purely real or purely imaginary for $M > 2$. Thus, (4.219) requires $3M-2$ real multiplications and $5M-6$ real additions. The required real-valued FFT algorithm is presented in [2].

4.5.2.2 GDFT-IV-Based Fast Algorithm [18]

The N-point forward and inverse GDFT-IV of a real-valued data sequence $\{x_n\}$, are, respectively, defined as [74]

$$f_k^{\mathrm{IV}} = \frac{1}{N} \sum_{n=0}^{N-1} x_n\, e^{-i\,\frac{2\pi(2n+1)(2k+1)}{4N}}, \qquad i = \sqrt{-1}, \qquad k = 0, 1, \ldots, N-1, \tag{4.223}$$

$$x_n = \sum_{k=0}^{N-1} f_k^{\text{IV}} \; e^{i\,\frac{2\pi(2n+1)(2k+1)}{4N}}, \qquad n = 0, 1, \ldots, N-1, \tag{4.224}$$

where $\{f_k^{\text{IV}}\}$ are GDFT-IV coefficients. The GDFT-IV coefficients $\{f_k^{\text{IV}}\}$ have the following symmetry property [18, 74]

$$f_k^{\text{IV}} = f_{N-1-k}^{*\,IV}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.225}$$

where $*$ denotes the complex conjugate. The forward GDFT-IV of real-valued data sequences may be decomposed as [18]

$$\begin{aligned} f_k^{\text{IV}} = \frac{1}{N} \sum_{n=0}^{\frac{N}{2}-1} & (x_n - x_{N-1-n}) \; \cos\left[\frac{\pi}{4(N/2)}(2n+1)(2k+1)\right] \\ & -i\,(x_n + x_{N-1-n}) \; \sin\left[\frac{\pi}{4(N/2)}(2n+1)(2k+1)\right], \\ k = 0, 1, \ldots, N-1. & \end{aligned} \tag{4.226}$$

The transform kernels on the right-hand side of (4.226) are recognized as the $\frac{N}{2}$-point DCT-IV of $\{x_n - x_{N-1-n}\}$ (real part), and the $\frac{N}{2}$-point DST-IV of $\{x_n + x_{N-1-n}\}$ (imaginary part). Based on the relation between the DST-IV and the DCT-IV (see Appendix C.2), the DST-IV in (4.226) may be converted to the DCT-IV with a simple pre-processing of data sequences. A sparse block matrix factorization of the GDFT-IV matrix defining a fast algorithm is presented in [18].

On the other hand, the forward MDCT in (4.215) for $N = 2M$ can be decomposed as [18]

$$\begin{aligned} c_k^o &= \sum_{n=0}^{2M-1} x_n \; \cos\left[\frac{\pi}{4M}(2n+1+M)(2k+1)\right] \\ &= \sum_{n=0}^{2M-1} x_n \; \cos\left[\frac{\pi}{4M}(2n+1)(2k+1) + \frac{\pi}{4}(2k+1)\right] \\ &= \cos\frac{\pi}{4}(2k+1) \sum_{n=0}^{M-1} (x_n - x_{2M-1-n}) \; \cos\left[\frac{\pi}{4M}(2n+1)(2k+1)\right] \\ &\quad - \sin\frac{\pi}{4}(2k+1) \sum_{n=0}^{M-1} (x_n + x_{2M-1-n}) \; \sin\left[\frac{\pi}{4M}(2n+1)(2k+1)\right], \\ k &= 0, 1, \ldots, M-1, \end{aligned} \tag{4.227}$$

From Sect. 4.3.4 we know that the MDCT and MDST are closely related to the DCT-IV and DST-IV of reduced sizes, respectively. Therefore, Eqs. (4.226) and (4.227) imply that the forward $2M$-point MCLT can be mapped into the $2M$-point GDFT-IV of real-valued data sequences utilizing its generalized signal flow graph with a simple post-processing of output data sequences. The terms $\cos\frac{\pi}{4}(2k+1)$ and $\sin\frac{\pi}{4}(2k+1)$ change signs on cyclic sets

$$\begin{aligned} \mathcal{C} &= \{+1,-1,-1,+1\}, \quad cc_k \in \mathcal{C} \quad \forall \ k \bmod 4, \\ \mathcal{S} &= \{+1,+1,-1,-1\}, \quad ss_k \in \mathcal{S} \quad \forall \ k \bmod 4. \end{aligned} \tag{4.228}$$

Then, the forward MDCT as the real part, and the forward MDST as the imaginary part of MCLT are related to the forward GDFT-IV by Britanak and Rao [18]

$$\begin{aligned} \overset{o}{c}_k &= \frac{\sqrt{2}}{2}\left(cc_{k \bmod 4}\, f_k^{\mathrm{IV}} + ss_{(2M-1-k) \bmod 4}\, f_{2M-1-k}^{\mathrm{IV}}\right), \\ \overset{o}{s}_{M-1-k} &= \frac{\sqrt{2}}{2}\left(ss_{(k) \bmod 4}\, f_k^{\mathrm{IV}} + cc_{(2M-1-k) \bmod 4}\, f_{2M-1-k}^{\mathrm{IV}}\right), \\ & k = 0,1,\ldots,M-1, cc_j \in \mathcal{C}, \quad ss_j \in \mathcal{S}, \quad j = 0,1,2,3. \end{aligned} \tag{4.229}$$

The GDFT-IV generalized signal flow modified for the simultaneous computation of the MDCT and MDST for $N = 2M = 16$ is shown in Fig. 4.12. The relation (4.229) forms the last simple butterfly stage.

Note 12 For the representation of terms $\cos\frac{\pi}{4}(2k+1)$ and $\sin\frac{\pi}{4}(2k+1)$ in (4.227), the trigonometric identity (4.150) and algebraic identity (4.153) alternatively may be used.

The backward MCLT computation is simply realized via the inverse GDFT-IV as follows. With respect to modified generalized GDFT-IV signal flow graph shown in Fig. 4.12, at first we need to compute two inverse DCTs-IV given by

$$\begin{aligned} z_n &= \text{DCT-IV of } \{\overset{o}{c}_k\}, \qquad n = 0,1,\ldots,M-1, \\ (-1)^n\, z_{2M-1-n} &= \text{DCT-IV of } \{\overset{o}{s}_{M-1-k}\}, \\ & n = M, M+1, \ldots, 2M-1, \end{aligned} \tag{4.230}$$

and the original data sequence $\{x_n\}$ is recovered from $\{z_n\}$ as

$$\begin{aligned} x_n &= z_n + z_{2M-1-n}, \qquad x_{2M-1-n} = -z_n + z_{2M-1-n}, \\ & n = 0,1,\ldots,M-1. \end{aligned} \tag{4.231}$$

With respect to the generalized signal flow graph shown in Fig. 4.12, the total computational complexity of the GDFT-IV-based fast MCLT algorithm [18] is given by $4M$ additions plus the complexity of two M-point DCTs-IV. The required DCT-IV fast algorithms/computational structures are presented in Appendix C.2.

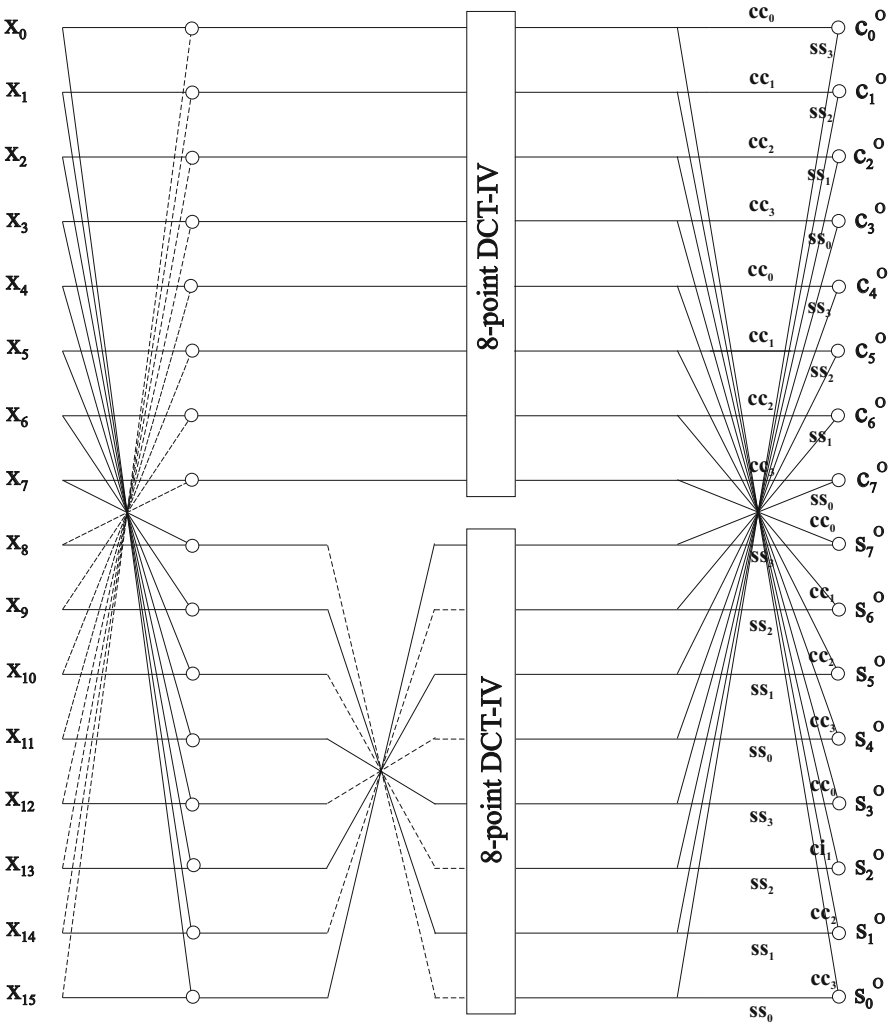

Fig. 4.12 Modified GDFT-IV generalized signal flow graph for the simultaneous computation of the MDCT and MDST for $N = 2M = 16$

4.5.3 GDHT-Based Fast Algorithms

For the efficient MCLT computation, GDHT-based algorithms were developed [18, 69], specifically, GDHT-IV-based fast MCLT algorithm [18] and GDHT-II-based fast MCLT algorithm with incorporated sine windowing function [69]. They are discussed in the following subsections.

4.5.3.1 GDHT-IV-Based Fast Algorithm [18]

N-point forward GDHT-IV of a real-valued data sequence $\{x_n\}$ is defined as [74]

$$h_k^{\text{IV}} = \frac{1}{N} \sum_{n=0}^{N-1} x_n \, \text{cas} \left[\frac{2\pi(2n+1)(2k+1)}{4N} \right], \qquad k = 0, 1, \ldots, N-1, \tag{4.232}$$

where $\{h_k^{\text{IV}}\}$ are GDHT-IV coefficients and cas $(.) = \cos(.) + \sin(.)$. A sparse block matrix factorization of the GDHT-IV matrix defining a fast algorithm is presented in [18].

Since the GDHT-IV is related to the GDFT-IV by Britanak and Rao [18]

$$\begin{aligned} h_k^{\text{IV}} &= f_k^{\text{IV}} + f_{N-1-k}^{\text{IV}}, \\ h_{N-1-k}^{\text{IV}} &= f_k^{\text{IV}} - f_{N-1-k}^{\text{IV}}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \end{aligned} \tag{4.233}$$

this fact implies that there also exists a relation among MDCT, MDST, and GDHT-IV. Indeed, the MDCT is related to the GDHT-IV by Britanak and Rao [18]

$$\begin{aligned} c_{2k}^{o} &= (-1)^{k+1} \sqrt{2} \, h_{N-1-2k}^{\text{IV}}, \\ c_{2k+1}^{o} &= (-1)^{k+1} \sqrt{2} \, h_{2k+1}^{\text{IV}}, \qquad k = 0, 1, \ldots, \frac{N}{4} - 1, \end{aligned} \tag{4.234}$$

while the MDST is related to the GDHT-IV by Britanak and Rao [18]

$$\begin{aligned} s_{2k}^{o} &= (-1)^{k} \sqrt{2} \, h_{2k}^{\text{IV}} \\ s_{2k+1}^{o} &= (-1)^{k+1} \sqrt{2} \, h_{N-2-2k}^{\text{IV}}, \qquad k = 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{4.235}$$

The modified GDHT-IV generalized signal flow graph for the simultaneous computation of the MDCT and MDST for $N = 2M = 16$ is shown in Fig. 4.13.

With respect to the generalized signal flow graph shown in Fig. 4.13, the total computational complexity of the GDHT-IV-based fast MCLT algorithm [18] is the same as for GDFT-IV-based fast MCLT algorithm.

4.5.3.2 GDHT-II-Based Fast Algorithm [69]

N-point forward GDHT-II of a real-valued data sequence $\{x_n\}$ is defined as [74]

$$h_k^{\text{II}} = \frac{1}{N} \sum_{n=0}^{N-1} x_n \, \text{cas} \left[\frac{2\pi(2n+1)k}{2N} \right], \qquad k = 0, 1, \ldots, N-1, \tag{4.236}$$

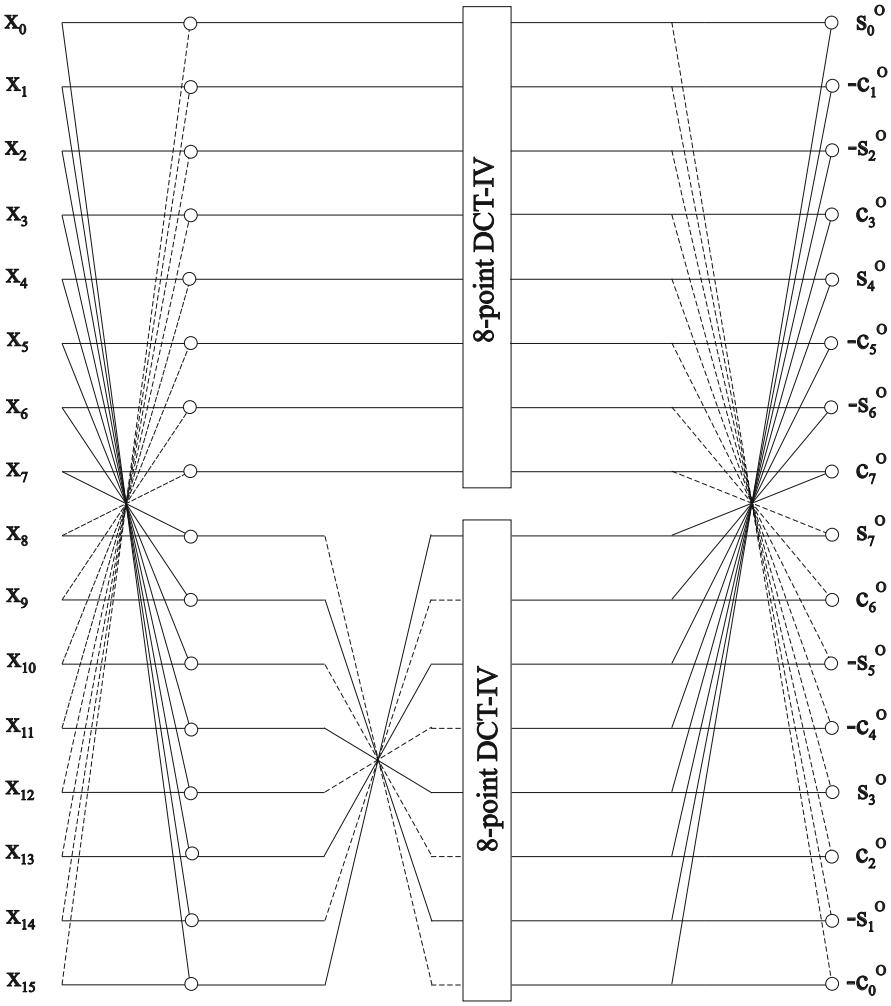

Fig. 4.13 Modified GDHT-IV generalized signal flow graph for the simultaneous computation of the MDCT and MDST for $N = 2M = 16$

where $\{h_k^{\text{II}}\}$ are GDHT-II coefficients. A sparse block matrix factorization of the GDHT-II matrix defining a fast algorithm is presented in [18].

With respect to (4.215), denote the basis functions of the MDCT and MDST with the normalization factor $\sqrt{\frac{2}{M}}$ and the incorporated negative sine windowing function, respectively, as

$$b_{k,n}^{(c)} = \sqrt{\frac{2}{M}} \sin\left[-\frac{\pi(2n+1)}{4M}\right] \cos\left[\frac{\pi}{4M}(2n+1+M)(2k+1)\right],$$

$$b_{k,n}^{(s)} = \sqrt{\frac{2}{M}}\ \sin\left[-\frac{\pi(2n+1)}{4M}\right]\ \sin\left[\frac{\pi}{4M}(2n+1+M)(2k+1)\right],$$
$$k = 0, 1, \ldots, M-1, \qquad n = 0, 1, \ldots, 2M-1. \tag{4.237}$$

It can be easily verified that the basis functions in (4.237) have the following symmetry properties [69]

$$b_{k,n}^{(c)} = (-1)^{M+1}\ b_{2M-1-k,n}^{(c)},$$
$$b_{k,n}^{(s)} = (-1)^{M}\ b_{2M-1-k,n}^{(s)}, \qquad k = 0, 1, \ldots, M-1, \tag{4.238}$$

Equation (4.238) implies that MDCT and MDST coefficients satisfy

$$c_{2M-1-k}^{o} = (-1)^{M+1}\ c_{k}^{o},$$
$$s_{2M-1-k}^{o} = (-1)^{M}\ s_{k}^{o}, \qquad k = 0, 1, \ldots, M-1, \tag{4.239}$$

demonstrating the even anti-symmetry and even symmetry property of $\{c_k^o\}$ and $\{s_k^o\}$, respectively, depending on the parity of M. Therefore, the computation of $\{c_k^o\}$ and $\{s_k^o\}$ can be realized by computing either via the even-indexed or odd-indexed coefficients. The even-indexed coefficients are chosen. Then, the basis functions $\{b_{k,n}^{(c)}\}$ become

$$b_{2k,n}^{(c)} = -\sqrt{\frac{2}{M}}\ \sin\left[\frac{\pi(2n+1)}{4M}\right]\ \cos\left[\frac{\pi}{4}(4k+1) + \frac{\pi}{4M}(2n+1)(4k+1)\right]$$
$$= -\sqrt{\frac{1}{M}}\ (-1)^{k}\ \sin\left[\frac{\pi(2n+1)}{4M}\right]\ \text{cas}\ \left[\frac{-\pi(2n+1)(4k+1)}{4M}\right],$$
$$k = 0, 1, \ldots, M-1. \tag{4.240}$$

Applying the trigonometric identity

$$-\sin(\alpha)\ \text{cas}\ (-\beta) = \frac{1}{2}\ [\text{cas}\ (\beta-\alpha) - \text{cas}\ (\alpha+\beta)], \tag{4.241}$$

to the sine windowing function, (4.240) finally becomes

$$b_{2k,n}^{(c)} = \sqrt{\frac{1}{M}}\ \frac{(-1)^{k}}{2}\ \left\{\text{cas}\ \left[\frac{\pi(2n+1)2k}{2M}\right] - \text{cas}\ \left[\frac{\pi(2n+1)(2k+1)}{2M}\right]\right\},$$
$$k = 0, 1, \ldots, M-1. \tag{4.242}$$

Similarly, for the basis functions $\{b_{k,n}^{(s)}\}$ we get

$$b_{2k,n}^{(s)} = -\sqrt{\frac{2}{M}}\ \sin\left[\frac{\pi(2n+1)}{4M}\right]\ \sin\left[\frac{\pi}{4}(4k+1)+\frac{\pi}{4M}(2n+1)(4k+1)\right]$$

$$= -\sqrt{\frac{1}{M}}\ (-1)^k\ \sin\left[\frac{\pi(2n+1)}{4M}\right]\ \text{cas}\ \left[\frac{\pi(2n+1)(4k+1)}{4M}\right]$$

$$= \sqrt{\frac{1}{M}}\ \frac{(-1)^k}{2}\ \left\{\text{cas}\ \left[\frac{-\pi(2n+1)(2k+1)}{2M}\right]-\text{cas}\ \left[\frac{-\pi(2n+1)2k}{2M}\right]\right\},$$

$$k = 0, 1, \ldots, M-1. \tag{4.243}$$

If we denote

$$h_k^{\text{II}} = \sqrt{\frac{1}{2M}}\ \sum_{n=0}^{2M-1} x_n\ \text{cas}\left[\frac{\pi(2n+1)k}{2M}\right],$$

$$\hat{h}_k^{\text{II}} = \sqrt{\frac{1}{2M}}\ \sum_{n=0}^{2M-1} x_n\ \text{cas}\left[\frac{-\pi(2n+1)k}{2M}\right], \quad k = 0, 1, \ldots, 2M-1, \tag{4.244}$$

it can be verified that

$$\hat{h}_{2M-k}^{\text{II}} = -h_k^{\text{II}}, \qquad k = 0, 1, \ldots, M-1. \tag{4.245}$$

Transform kernels in (4.244) are the normalized $2M$-point GDHT-II of $\{x_n\}$. According to (4.215) using (4.242), (4.243), (4.244), and (4.245), the MDCT and MDST coefficients $\{c_{2k}^o\}$ and $\{s_{2k}^o\}$ are, respectively, given by Shu et al. [69]

$$c_{2k}^o = (-1)^k\ \frac{\sqrt{2}}{2}\ \left(h_{2k}^{\text{II}} - h_{2k+1}^{\text{II}}\right),$$

$$s_{2k}^o = (-1)^k\ \frac{\sqrt{2}}{2}\ \left(\hat{h}_{2k+1}^{\text{II}} - \hat{h}_{2k}^{\text{II}}\right) = (-1)^k\ \frac{\sqrt{2}}{2}\ \left(h_{2M-2k}^{\text{II}} - h_{2M-1-2k}^{\text{II}}\right),$$

$$k = 0, 1, \ldots, M-1. \tag{4.246}$$

The odd-indexed coefficients $\{c_{2k+1}^o\}$ and $\{s_{2k+1}^o\}$ can be deduced from the even anti-symmetry and even symmetry properties (4.239), i.e.,

$$c_{2k+1}^o = (-1)^{M+1}\ c_{2M-2-2k}^o, \qquad s_{2k+1}^o = (-1)^M\ s_{2M-2-2k}^o,$$

$$k = 0, 1, \ldots, \frac{M}{2}-1. \tag{4.247}$$

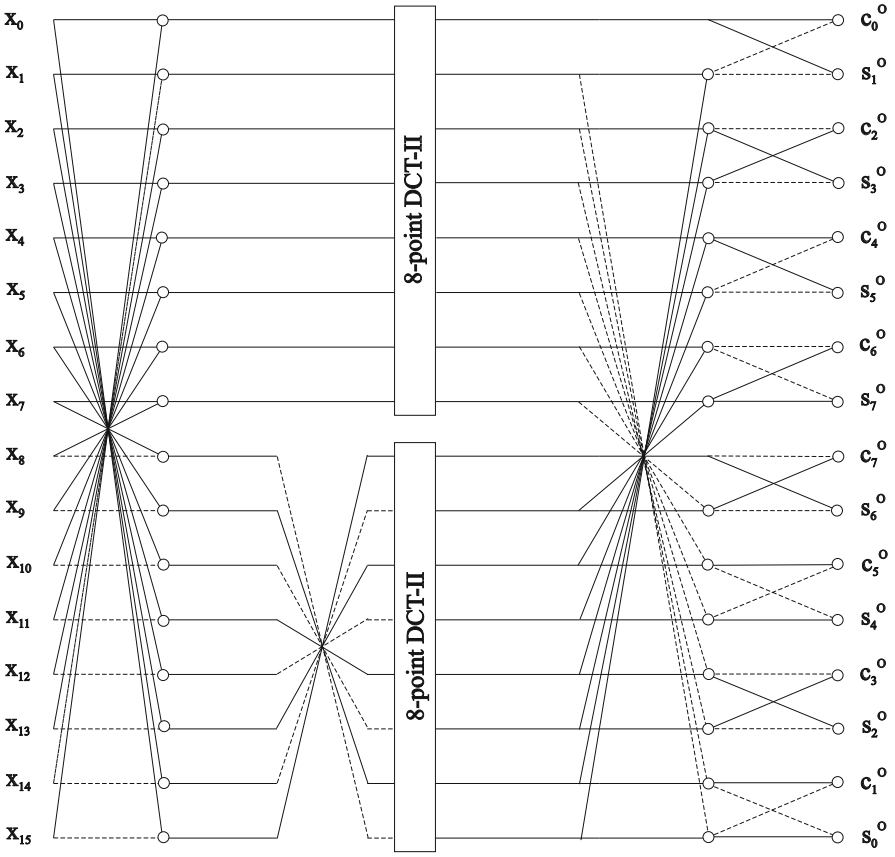

Fig. 4.14 Modified GDHT-II generalized signal flow graph for the simultaneous computation of the MDCT and MDST for $N = 2M = 16$

The GDHT-II generalized signal flow modified for the simultaneous computation of the MDCT and MDST for $N = 2M = 16$ is shown in Fig. 4.14. The relations (4.246) and (4.247) form the last simple butterfly stage.

With respect to the generalized signal flow graph shown in Fig. 4.14, the total computational complexity of the GDHT-II-based fast MCLT algorithm [69] is given by $6M - 2$ additions plus the complexity of two M-point DCTs-II. The required DCT-II fast algorithm/computational structure is presented in Appendix C.1.

4.5.4 DCT-II-Based Fast Algorithm [67]

A fast MCLT algorithm with the incorporated sine windowing function developed in [67] is based on the DCT-II-based evenly stacked fast MDCT/MDST algorithm

defined by (4.35), (4.36) and (4.37), (4.38) for $N = 2M$. In [67] it was shown that the real part (MDCT) and imaginary part (MDST) of the MCLT can be directly obtained from a combination of $\{\overset{E}{c}_k\}$ and $\{\overset{E}{s}_k\}$ coefficients. This fact in general allows for the fast MCLT computation to use any fast evenly stacked MDCT/MDST algorithm associated with the sine windowing function.

Consider the basis functions of the MDCT and MDST given by (4.237) with the incorporated negative sine windowing function. Using the trigonometric identity $\sin(\alpha)\cos(\beta) = \frac{1}{2}[\sin(\alpha-\beta)+\sin(\alpha+\beta)]$, the MDCT basis functions $\{b_{k,n}^{(c)}\}$ can be simplified to [67]

$$b_{k,n}^{(c)} = \sqrt{\frac{1}{2M}}\left(\sin\left[\frac{\pi}{2M}(2n+1+M)k+\frac{\pi}{4}\right]\right.$$
$$\left.-\sin\left[\frac{\pi}{2M}(2n+1+M)(k+1)-\frac{\pi}{4}\right]\right). \tag{4.248}$$

The transform kernels on the right-hand side of (4.248) correspond to the evenly stacked MDST. Further, applying the standard trigonometric identities to (4.248) we get [67]

$$b_{k,n}^{(c)} = \frac{1}{2\sqrt{M}}\left\{\cos\left[\frac{\pi}{2M}(2n+1+M)k\right]\right.$$
$$+\sin\left[\frac{\pi}{2M}(2n+1+M)k\right]$$
$$+\cos\left[\frac{\pi}{2M}(2n+1+M)(k+1)\right]$$
$$\left.-\sin\left[\frac{\pi}{2M}(2n+1+M)(k+1)\right]\right\}. \tag{4.249}$$

On the other hand, using the trigonometric identity $\sin(\alpha)\sin(\beta) = \frac{1}{2}[\cos(\alpha-\beta)-\cos(\alpha+\beta)]$, the MDST basis functions $\{b_{k,n}^{(s)}\}$ can be simplified to [67]

$$b_{k,n}^{(s)} = \sqrt{\frac{1}{2M}}\left(-\cos\left[\frac{\pi}{2M}(2n+1+M)k+\frac{\pi}{4}\right]\right.$$
$$\left.+\cos\left[\frac{\pi}{2M}(2n+1+M)(k+1)-\frac{\pi}{4}\right]\right). \tag{4.250}$$

The transform kernels on the right-hand side of (4.250) correspond to the evenly stacked MDCT. Similarly, applying the standard trigonometric identities to (4.250) we get [67]

$$b_{k,n}^{(s)} = \frac{1}{2\sqrt{M}} \left\{ -\cos\left[\frac{\pi}{2M}(2n+1+M)k\right] + \sin\left[\frac{\pi}{2M}(2n+1+M)k\right] + \cos\left[\frac{\pi}{2M}(2n+1+M)(k+1)\right] + \sin\left[\frac{\pi}{2M}(2n+1+M)(k+1)\right] \right\}. \tag{4.251}$$

From (4.249) and (4.251) it follows that the MDCT and MDST basis functions are decomposed into the sums/differences of evenly stacked MDCT and MDST transform kernels. Thus, according to (4.215) the MDCT coefficients being the real part of MCLT are obtained from (4.249) as

$$c_k^O = c_k^E + s_k^E + c_{k+1}^E - s_{k+1}^E, \qquad k = 0, 1, \ldots, M-1, \tag{4.252}$$

while the MDST coefficients being the imaginary part of MCLT are obtained from (4.251) as

$$s_k^O = -c_k^E + s_k^E + c_{k+1}^E + s_{k+1}^E, \qquad k = 0, 1, \ldots, M-1. \tag{4.253}$$

For the efficient implementation of (4.252) and (4.253), if we denote

$$u_k = c_{k+1}^E + s_k^E, \qquad v_k = s_{k+1}^E - c_k^E, \qquad k = 0, 1, \ldots, M-1, \tag{4.254}$$

then

$$c_k^O = u_k - v_k, \qquad s_k^O = u_k + v_k, \qquad k = 0, 1, \ldots, M-1. \tag{4.255}$$

The corresponding generalized signal flow graph of the DCT-II-based fast MCLT algorithm [67] for $N = 2M = 16$ is shown in Fig. 4.15.

Equations (4.36) and (4.38) for $N = 2M$ require $2M$ additions. Equations (4.254) and (4.255) ignoring the trivial additions by $c_M^E = s_0^E = 0$ require $4M - 2$ additions. Then, the total computational complexity of the DCT-II-based fast MCLT algorithm [67] is given by $6M - 2$ additions plus the complexity of two M-point DCTs-II. The required DCT-II fast algorithm/computational structure is presented in Appendix C.1.

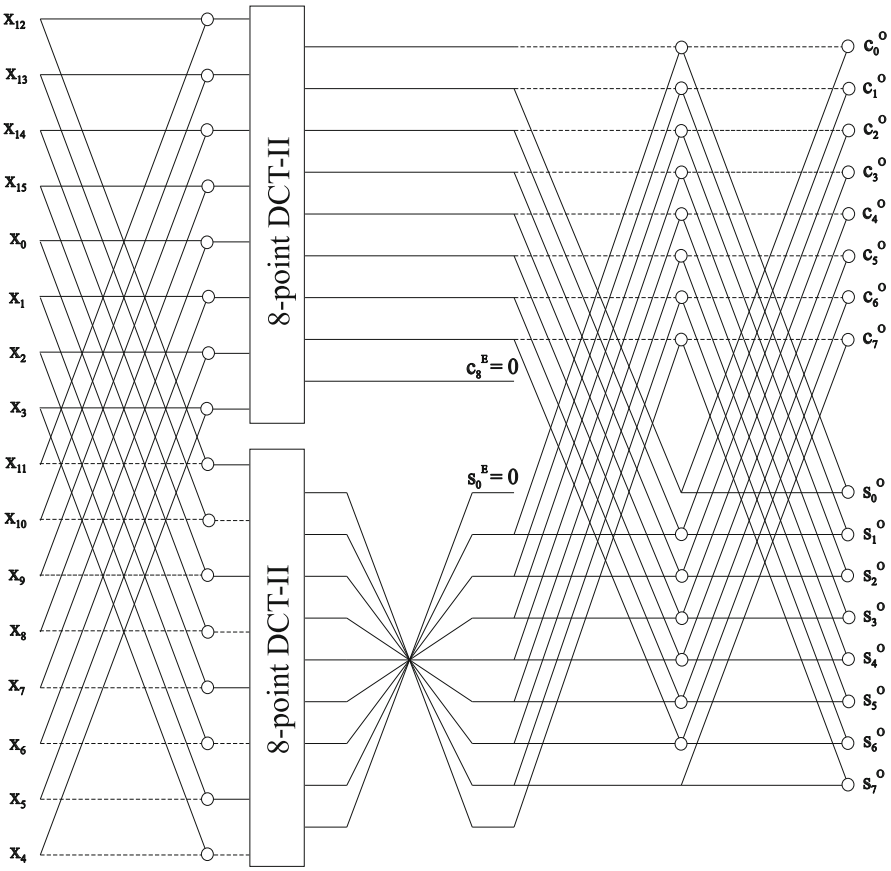

Fig. 4.15 Generalized signal flow graph of the DCT-II-based fast MCLT algorithm [67] for $N = 2M = 16$

4.5.5 DCT-IV-Based Fast Algorithms [3]

After introducing the MCLT immediately its DCT-IV-based fast algorithm was also proposed [3]. The real part of the MCLT, MLT, or equivalently MDCT is obtained via the DCT-IV of reduced size, while the imaginary part of the MCLT and MDST is obtained via the corresponding DST-IV of reduced size.

Indeed, for $N = 2M$ applying a permutation to the input data sequence $\{x_n\}$ defined by Malvar [3]

$$\begin{aligned} u_{\frac{M}{2}+n} &= \quad x_n \quad - x_{M-1-n}, \\ u_{\frac{M}{2}-1-n} &= -x_{M+n} - x_{2M-1-n}, \qquad n = 0, 1, \ldots, \frac{M}{2} - 1, \end{aligned} \tag{4.256}$$

the $2M$-point forward MLT or MDCT is reduced to the M-point DCT-IV of $\{u_n\}$ as

$$c_k^o = \sum_{n=0}^{M-1} u_n \cos\left[\frac{\pi}{4M}(2n+1)(2k+1)\right], \qquad k = 0, 1, \ldots, M-1. \tag{4.257}$$

On the other hand, applying a permutation to the input data sequence $\{x_n\}$ defined by Malvar [3]

$$\begin{aligned} v_{\frac{M}{2}+n} &= x_n \quad + x_{M-1-n}, \\ v_{\frac{M}{2}-1-n} &= x_{M+n} - x_{2M-1-n}, \qquad n = 0, 1, \ldots, \frac{M}{2} - 1, \end{aligned} \tag{4.258}$$

the $2M$-point forward MDST is reduced to the M-point DST-IV of $\{v_n\}$ as

$$s_k^o = \sum_{n=0}^{M-1} v_n \sin\left[\frac{\pi}{4M}(2n+1)(2k+1)\right], \qquad k = 0, 1, \ldots, M-1, \tag{4.259}$$

or equivalently, as

$$\begin{aligned} s_{M-1-k}^o &= \sum_{n=0}^{M-1} (-1)^n v_n \cos\left[\frac{\pi}{4M}(2n+1)(2k+1)\right], \\ k &= 0, 1, \ldots, M-1, \end{aligned} \tag{4.260}$$

The corresponding generalized signal flow graph of the DCT-IV-based fast MCLT algorithm [3] for $N = 2M = 16$ is shown in Fig. 4.16.

The total computational complexity of the DCT-IV-based fast MCLT algorithm [3] is given by $2M$ additions plus the complexity of two M-point DCTs-IV. The required DCT-IV fast algorithm/computational structures are presented in Appendix C.2.

4.5.6 *DCT-IV/DCT-II-Based Fast Algorithm [65, 66]*

An interesting and simple DCT-IV/DCT-II-based fast MCLT algorithm was developed in [65, 66]. Using trigonometric identities the basis functions of the MDCT and MDST are converted to the sum and difference of transform kernels corresponding to the DCT-IV of reduced sizes. Alternatively, based on the relation between the DCT-IV and the DCT-II (see Appendix C.3), the M-point DCT-IV can be converted to the DCT-II of the same size at the cost of M pre-multiplications and $M - 1$ recursive additions.

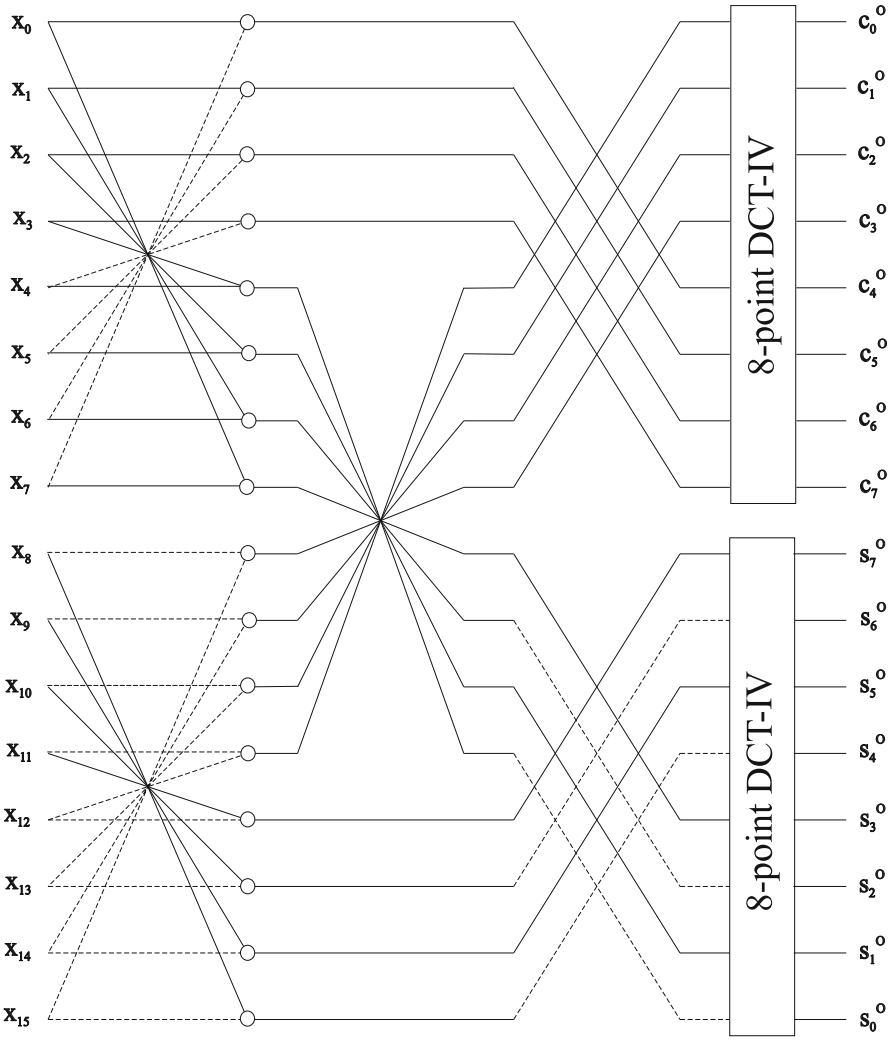

Fig. 4.16 Generalized signal flow graph of the DCT-IV-based fast MCLT algorithm [3] for N = 2M = 16

Consider the MDCT and MDST basis functions with the normalization factor $\sqrt{\frac{2}{M}}$, respectively, as

$$b_{k,n}^{(c)} = \sqrt{\frac{2}{M}} \cos\left[\frac{\pi}{4M}(2n+1+M)(2k+1)\right],$$

$$b_{k,n}^{(s)} = \sqrt{\frac{2}{M}} \sin\left[\frac{\pi}{4M}(2n+1+M)(2k+1)\right],$$

$$k = 0, 1, \ldots, M-1, \qquad n = 0, 1, \ldots, 2M-1. \tag{4.261}$$

Setting $\alpha_{k,n} = \frac{\pi}{4M}(2n+1+2M)(2k+1)$ and $\beta_{k,n} = \frac{\pi}{4M}(2n+1)(2k+1)$, and applying the trigonometric identities

$$\begin{aligned} \cos\,\alpha + \cos\,\beta &= \;\; 2\,\cos\,\frac{\alpha+\beta}{2}\,\cos\,\frac{\alpha-\beta}{2}, \\ \cos\,\alpha - \cos\,\beta &= -2\,\sin\,\frac{\alpha+\beta}{2}\,\sin\,\frac{\alpha-\beta}{2}, \end{aligned} \tag{4.262}$$

then using the trigonometric identity (4.150) and algebraic identity (4.153), the MDCT and MDST basis functions (4.261) can be written in the form [66]

$$\begin{aligned} b_{k,n}^{(c)} &= (-1)^{\lfloor \frac{k+1}{2} \rfloor} \sqrt{\frac{1}{2M}}\,(\cos\,\alpha_{k,n} + \cos\,\beta_{k,n})\,, \\ b_{k,n}^{(s)} &= -(-1)^{\lfloor \frac{k}{2} \rfloor} \sqrt{\frac{1}{2M}}\,(\cos\,\alpha_{k,n} - \cos\,\beta_{k,n})\,, \\ k &= 0, 1, \ldots, M-1, \qquad n = 0, 1, \ldots, 2M-1. \end{aligned} \tag{4.263}$$

Importantly, the cosines of $\alpha_{k,n}$ and $\beta_{k,n}$ satisfy the following relations [66]

$$\cos\,\alpha_{k,n} = \cos\,\beta_{k,M+n} = -\cos\,\beta_{k,M-1-n}, \qquad \cos\,\alpha_{k,M+n} = -\cos\,\beta_{k,n}. \tag{4.264}$$

According to (4.215) for the forward MDCT and MDST we have

$$\begin{aligned} c_k^o &= (-1)^{\lfloor \frac{k+1}{2} \rfloor} \sqrt{\frac{1}{2M}} \left\{ \sum_{n=0}^{2M-1} x_n\,\cos\left[\frac{\pi}{4M}(2n+1+2M)\,(2k+1)\right] \right. \\ &\quad \left. + \sum_{n=0}^{2M-1} x_n\,\cos\left[\frac{\pi}{4M}(2n+1)\,(2k+1)\right] \right\}, \\ s_k^o &= -(-1)^{\lfloor \frac{k}{2} \rfloor} \sqrt{\frac{1}{2M}} \left\{ \sum_{n=0}^{2M-1} x_n\,\cos\left[\frac{\pi}{4M}(2n+1+2M)\,(2k+1)\right] \right. \\ &\quad \left. - \sum_{n=0}^{2M-1} x_n\,\cos\left[\frac{\pi}{4M}(2n+1)\,(2k+1)\right], \right\}, \qquad k = 0, 1, \ldots, M-1. \end{aligned} \tag{4.265}$$

Now, investigate two unique sums on the right-hand sides of (4.265) separately. Using the relations (4.264) for the first sum we get

$$
\begin{aligned}
&\sum_{n=0}^{2M-1} x_n \cos\left[\frac{\pi}{4M}(2n+1+2M)\,(2k+1)\right] \\
&= \sum_{n=0}^{M-1} x_n \cos\left[\frac{\pi}{4M}(2n+1+2M)\,(2k+1)\right] \\
&\quad + \sum_{n=M}^{2M-1} x_n \cos\left[\frac{\pi}{4M}(2n+1+2M)\,(2k+1)\right] \\
&= \sum_{n=0}^{M-1} -x_{M-1-n} \cos\left[\frac{\pi}{4M}(2n+1)\,(2k+1)\right] \\
&\quad + \sum_{n=0}^{M-1} -x_{M+n} \cos\left[\frac{\pi}{4M}(2n+1)\,(2k+1)\right] \\
&= -\sum_{n=0}^{M-1} (x_{M-1-n} + x_{M+n}) \cos\left[\frac{\pi}{4M}(2n+1)\,(2k+1)\right]. \qquad (4.266)
\end{aligned}
$$

For the second sum we get

$$
\begin{aligned}
&\sum_{n=0}^{2M-1} x_n \cos\left[\frac{\pi}{4M}(2n+1)\,(2k+1)\right] \\
&= \sum_{n=0}^{M-1} x_n \cos\left[\frac{\pi}{4M}(2n+1)\,(2k+1)\right] \\
&\quad + \sum_{n=M}^{2M-1} x_n \cos\left[\frac{\pi}{4M}(2n+1)\,(2k+1)\right] \\
&= \sum_{n=0}^{M-1} x_n \cos\left[\frac{\pi}{4M}(2n+1)\,(2k+1)\right] \\
&\quad + \sum_{n=0}^{M-1} -x_{2M-1-n} \cos\left[\frac{\pi}{4M}(2n+1)\,(2k+1)\right] \\
&= \sum_{n=0}^{M-1} (x_n - x_{2M-1-n}) \cos\left[\frac{\pi}{4M}(2n+1)\,(2k+1)\right]. \qquad (4.267)
\end{aligned}
$$

Combining (4.265), (4.266), and (4.267) we finally obtain [66]

$$c_k^o = (-1)^{\lfloor \frac{k+1}{2} \rfloor} \sqrt{\frac{1}{2M}} \sum_{n=0}^{M-1} (u_n - v_n) \cos\left[\frac{\pi}{4M}(2n+1)(2k+1)\right],$$

$$s_k^o = (-1)^{\lfloor \frac{k}{2} \rfloor} \sqrt{\frac{1}{2M}} \sum_{n=0}^{M-1} (u_n + v_n) \cos\left[\frac{\pi}{4M}(2n+1)(2k+1)\right],$$

$$k = 0, 1, \ldots, M-1, \tag{4.268}$$

where

$$u_n = x_n - x_{2M-1-n}, \qquad v_n = x_{M-1-n} + x_{M+n}, \qquad n = 0, 1, \ldots, M-1. \tag{4.269}$$

Transform kernels on the right-hand side of (4.268) are M-point DCT-IV of $\{u_n - v_n\}$ and M-point DCT-IV of $\{u_n + v_n\}$. Alternatively, based on the relation between the DCT-IV and DCT-II, each M-point DCT-IV may be converted to the DCT-II of the same size at the cost of M pre-multiplications and $M-1$ recursive additions. Thus, we have two forms of the fast MCLT algorithm [66], the DCT-IV-based and DCT-IV/DCT-II-based form.

The generalized signal flow graph corresponding to the DCT-IV-based form of fast MCLT algorithm defined by (4.268) and (4.269) for $N = 2M = 16$ is shown in Fig. 4.17.

The total computational complexity of the DCT-IV-based form of fast MCLT algorithm [66] is given by $2M$ additions plus the complexity of two M-point DCTs-IV. The required DCT-IV fast algorithm/computational structures are presented in Appendix C.2.

On the other hand, the total computational complexity of the DCT-IV/DCT-II-based form of fast MCLT algorithm is given by $2M$ multiplications, $4M-2$ additions, and 2 shifts plus the complexity of two M-point DCTs-II. The required DCT-II fast algorithm/computational structure is presented in Appendix C.1.

4.5.7 Recursive Radix-2 DIF Fast Algorithm [64]

A recursive radix-2 DIF fast MCLT algorithm was developed in [64]. Essentially, it is based on the recursive radix-2 DIF fast MDCT algorithm [42] discussed in Sect. 4.3.7. The recursive radix-2 DIF fast MCLT algorithm [64] is constructed from an alternative recursive sparse block matrix factorization of the MDCT matrix without redundant computations, and a relation between the MDCT and the MDST matrices.

We recall that the recursive radix-2 DIF fast MDCT algorithm [42] for N being divisible by 4 is defined as [42]

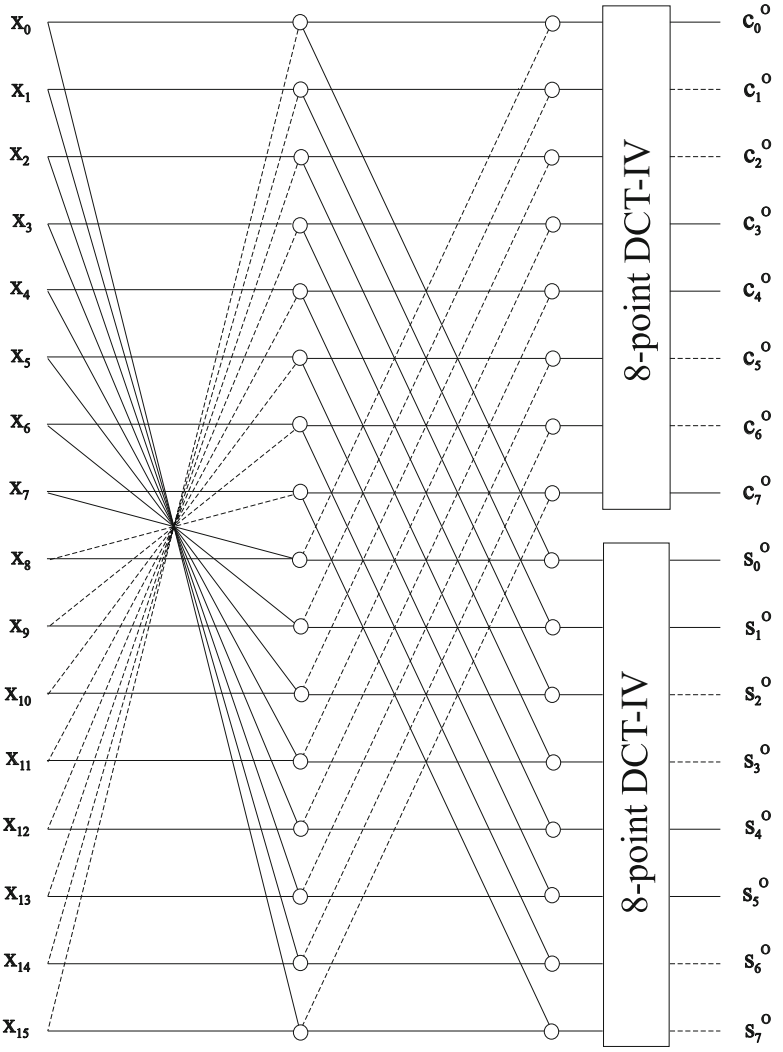

Fig. 4.17 Generalized signal flow graph corresponding to the DCT-IV-based form of fast MCLT algorithm [66] for $N = 2M = 16$

$$\begin{matrix} c^{o}_{2k}, & k \text{ even} \\ c^{o}_{2k+1}, & k \text{ odd} \end{matrix} \left\{ = (-1)^{\lfloor \frac{k+1}{2} \rfloor} \sum_{n=0}^{\frac{N}{2}-1} x_n^{(c)} \cos\left[\frac{\pi}{2(N/2)} \left(2n + 1 + \frac{N}{4} \right) (2k+1) \right], \right.$$

$$\begin{matrix} c^{o}_{2k+1}, & k \text{ even} \\ c^{o}_{2k}, & k \text{ odd} \end{matrix} \left\{ = (-1)^{\lfloor \frac{k+1}{2} \rfloor} \sum_{n=0}^{\frac{N}{2}-1} x_{\frac{N}{2}-1-n}^{(c)} \left[\frac{\pi}{2(N/2)} \left(2n + 1 + \frac{N}{4} \right) (2k+1) \right], \right.$$

$$k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.270}$$

The data sequence $\{x_n^{(c)}\}$ and its reversed version, $\{x_{\frac{N}{2}-1-n}^{(c)}\}$, are given by

$$\begin{aligned} x_n^{(c)} &= a_n \cos\frac{\pi(2n+1)}{2N} + b_n \sin\frac{\pi(2n+1)}{2N}, \\ x_{\frac{N}{2}-1-n}^{(c)} &= -a_n \sin\frac{\pi(2n+1)}{2N} + b_n \cos\frac{\pi(2n+1)}{2N}, \\ n &= 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{4.271}$$

where

$$\begin{aligned} a_n &= x_n - x_{\frac{N}{2}-1-n}, \qquad b_n = x_{\frac{N}{2}+n} + x_{N-1-n}, \\ n &= 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{4.272}$$

Equation (4.271) represents the block of $\frac{N}{4}$ Givens–Jacobi rotations. From (4.270) it can be seen that the computation of N-point MDCT is realized via the computation of two identical $\frac{N}{2}$-point MDCTs. The decomposition is applied recursively until trivial small even-length MDCT modules remain. However, the corresponding original recursive sparse block matrix factorization of the MDCT matrix presented in [42] involves redundant computations.

The key for the construction of the recursive fast radix-2 MCLT algorithm is the alternative recursive sparse block matrix factorization that involves no redundant computations. Denoting the MDCT matrix by $\boldsymbol{C}_{\frac{N}{2}\times N}$, it is defined as [64]

$$\boldsymbol{C}_{\frac{N}{2}\times N} = \boldsymbol{P}_{\frac{N}{2}} \begin{pmatrix} \boldsymbol{E}_{\frac{N}{4}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{E}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} \boldsymbol{C}_{\frac{N}{4}\times\frac{N}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_{\frac{N}{4}\times\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} \\ \boldsymbol{J}_{\frac{N}{2}} \end{pmatrix} \boldsymbol{G}_{\frac{N}{2}} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & -\boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix}, \tag{4.273}$$

where $\boldsymbol{I}_M$ is the identity matrix, $\boldsymbol{J}_M$ is the reverse ordered identity matrix, both of order M. $\boldsymbol{P}_{\frac{N}{2}}$ is the permutation matrix which reorders the MDCT coefficients $\{c_0, c_3, c_4, c_7, c_8, c_{11}, \ \ldots, \ c_1, c_2, c_5, c_6, c_9, c_{10}, \ \ldots\}$ to natural order. $\boldsymbol{0}'s$ are null matrices. The matrix $\boldsymbol{E}_{\frac{N}{4}} = \text{diag}\,\{(-1)^{\lfloor\frac{k+1}{2}\rfloor}\}$ is the diagonal sign-changing matrix, and $\boldsymbol{G}_{\frac{N}{2}}$ is the Givens–Jacobi rotation matrix defined by

$$\boldsymbol{G}_{\frac{N}{2}} = \begin{pmatrix} \cos\frac{\pi}{2N} & & & \sin\frac{\pi}{2N} \\ & \ddots & \iddots & \\ & \cos\frac{(\frac{N}{2}-1)\pi}{2N} & \sin\frac{(\frac{N}{2}-1)\pi}{2N} & \\ & -\sin\frac{(\frac{N}{2}-1)\pi}{2N} & \cos\frac{(\frac{N}{2}-1)\pi}{2N} & \\ & \iddots & \ddots & \\ -\sin\frac{\pi}{2N} & & & \cos\frac{\pi}{2N} \end{pmatrix}. \tag{4.274}$$

The recursive sparse MDCT matrix factorization given by (4.273) enables us to generate the higher-order MDCT matrix from two lower-order MDCT matrices.

There exits an intimate relation between the MDCT and MDST matrices. Actually, denoting the MDST matrix by $\boldsymbol{S}_{\frac{N}{2}\times N}$, the matrix $\boldsymbol{S}_{\frac{N}{2}\times N}$ is related to $\boldsymbol{C}_{\frac{N}{2}\times N}$ by Britanak [64]

$$\boldsymbol{S}_{\frac{N}{2}\times N} = (-1)^{\frac{N}{4}}\, \boldsymbol{J}_{\frac{N}{2}}\, \boldsymbol{C}_{\frac{N}{2}\times N}\, \boldsymbol{D}_N, \tag{4.275}$$

where $\boldsymbol{J}_{\frac{N}{2}}$ is the reverse ordered identity matrix of order $\frac{N}{2}$, and $\boldsymbol{D}_N$ is a diagonal odd sign-changing matrix of order N defined as $\boldsymbol{D}_N = \text{diag}\,\{1, -1, 1, \ldots, 1, -1\}$.

Using the alternative recursive sparse MDCT matrix factorization given by (4.273) and the relation between MDST and MDCT matrices given by (4.275), a recursive radix-2 DIF fast MDST algorithm is defined as [64]

$$\begin{aligned} &s^{o}_{\frac{N}{2}-1-2k},\ k \text{ even} \\ &s^{o}_{\frac{N}{2}-2-2k},\ k \text{ odd} \end{aligned} \Bigg\{ = (-1)^{\frac{N}{4}+\lfloor\frac{k+1}{2}\rfloor} \sum_{n=0}^{\frac{N}{2}-1} x_n^{(s)}\, \cos\left[\frac{\pi}{2(N/2)}\left(2n+1+\frac{N}{4}\right)(2k+1)\right], \\ \begin{aligned} &s^{o}_{\frac{N}{2}-2-2k},\ k \text{ even} \\ &s^{o}_{\frac{N}{2}-1-2k},\ k \text{ odd} \end{aligned} \Bigg\{ = (-1)^{\frac{N}{4}+\lfloor\frac{k+1}{2}\rfloor} \sum_{n=0}^{\frac{N}{2}-1} x^{(s)}_{\frac{N}{2}-1-n}\, \cos\left[\frac{\pi}{2(N/2)}\left(2n+1+\frac{N}{4}\right)(2k+1)\right], \\ k = 0, 1, \ldots, \frac{N}{4}-1. \end{aligned} \tag{4.276}$$

The data sequence $\{x_n^{(s)}\}$ and its reversed version, $\{x^{(s)}_{\frac{N}{2}-1-n}\}$, are given by

$$\begin{aligned} x_n^{(s)} &= c_n\, \cos\frac{\pi(2n+1)}{2N} + d_n\, \sin\frac{\pi(2n+1)}{2N}, \\ x^{(s)}_{\frac{N}{2}-1-n} &= -c_n\, \sin\frac{\pi(2n+1)}{2N} + d_n\, \cos\frac{\pi(2n+1)}{2N}, \qquad n = 0, 1, \ldots, \frac{N}{4}-1, \end{aligned} \tag{4.277}$$

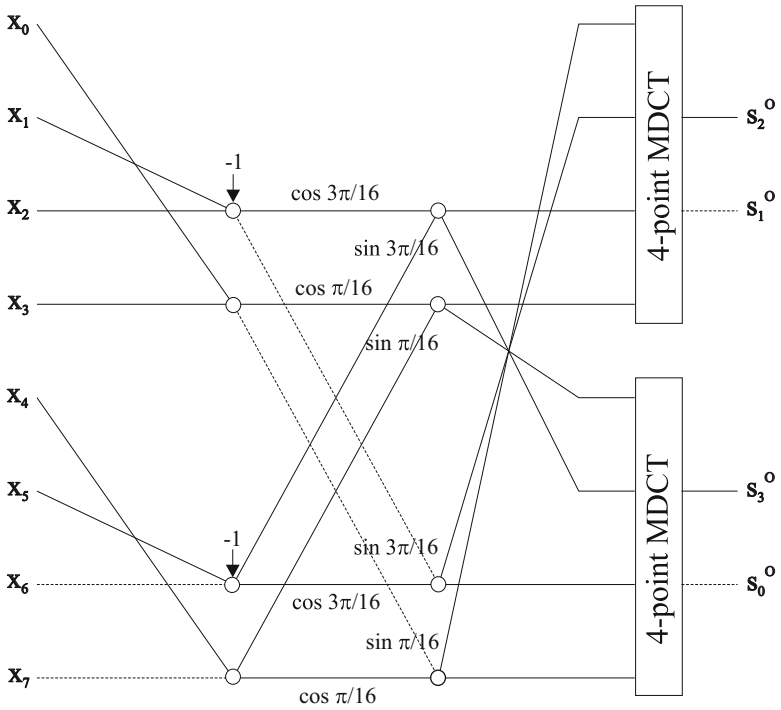

Fig. 4.18 Generalized signal flow graph of the recursive radix-2 DIF fast MDST algorithm for $N = 8$

where

$$c_n = (-1)^n(x_n + x_{\frac{N}{2}-1-n}), \quad d_n = (-1)^n(x_{\frac{N}{2}+n} - x_{N-1-n}),$$

$$n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{4.278}$$

Equation (4.277) again represents the block of $\frac{N}{4}$ Givens–Jacobi rotations. The generalized signal flow graph of the radix-2 DIF fast MDST algorithm for $N = 8$ is shown in Fig. 4.18.

One can easily see that both fast MDCT and MDST computational structures in Figs. 4.11 and 4.18, respectively, are complementary to each other. This fact enables us to construct a fast MCLT computational structure by a simple way. Specifically, the composition of the fast MDCT and MDST computational structures in Figs. 4.11 and 4.18 leads to the fast MCLT computational structure which is shown for $N = 8$ in Fig. 4.19.

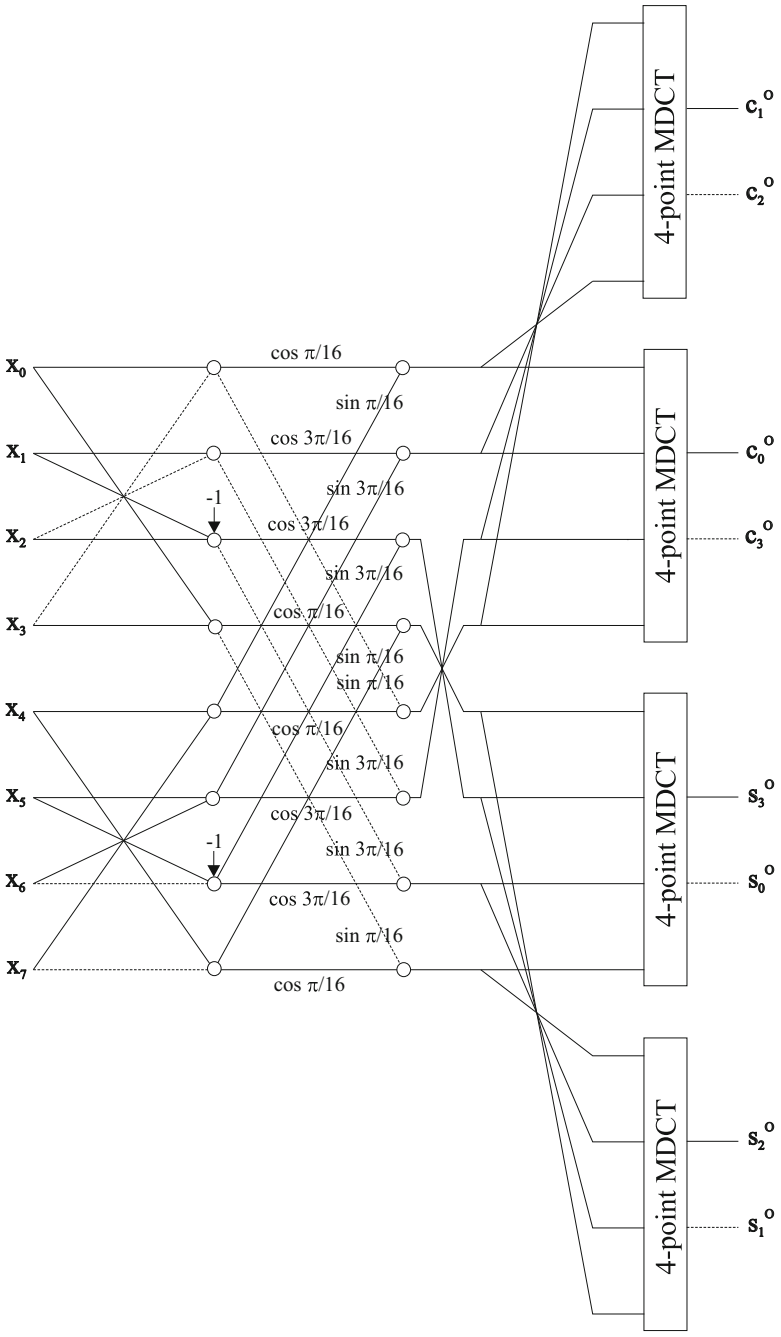

Fig. 4.19 Generalized signal flow graph of the recursive radix-2 DIF fast MCLT algorithm for $N = 8$ [64]

The total computational complexity of the radix-2 DIF fast MCLT algorithm [64] in general is given by

$$M_N = 4M_{\frac{N}{2}} + \frac{3N}{2}, \qquad A_N = 4A_{\frac{N}{2}} + \frac{5N}{2}. \tag{4.279}$$

Combining the recursive fast radix-2 DIF MCLT algorithm with the generalized fast mixed-radix MDCT algorithm [26] defined for the composite lengths $N = 2 \times q^m$, $m \geq 2$, where q is an odd positive integer, the MCLT can be computed for the composite lengths $N = 2^n \times q^m$, $n, m \geq 2$, thus supporting a wider range of transform sizes.

Note 13 The generalized signal flow graphs for the fast MCLT computation shown in Figs. 4.12, 4.13, 4.14, 4.15, 4.16, 4.17, and 4.19 correspond to sparse (block) matrix factorizations of the MCLT matrix.

4.5.8 Comparison of Fast MCLT Algorithms

Comparison of the fast MCLT algorithms for the simultaneous MDCT and MDST computation in terms of the arithmetic complexity for $N = 2M$, $M = 2^m$, $m > 2$, and used windowing function is summarized in Table 4.3.

For all discussed fast MCLT algorithms, in general, when N is a composite integer, i.e., it is of the form $N = 2^n \times q$, where q is an odd positive integer (or the mixed-radix length being the combination of radix-2 and radix-q lengths), the computational complexity of the N-point DCT-II is given by (C.8) and of N-point DCT-IV is given by (C.32) in Appendix C.

Table 4.3 Comparison of fast MCLT algorithms in terms of the arithmetic complexity for $N = 2M$, $M = 2^m$, $m > 2$, and used windowing function

Fast MCLT algorithm	# of real mults	# of real adds	Windowing function
Complex FFT-based [70]	$M(2m+3)$	$M(6m+3)+4$	Any
Real-valued FFT-based [68]	$M(m+1)$	$M(3m+3)-2$	Sine
GDFT-IV-based [18]	$M(m+2)$	$M(3m+4)$	Any
GDHT-IV-based [18]	$M(m+2)$	$M(3m+4)$	Any
GDHT-II-based [69]	Mm	$M(3m+4)$	Sine
DCT-II-based [67]	Mm	$M(3m+4)$	Sine
DCT-IV-based [3]	$M(m+2)$	$M(3m+2)$	Any
DCT-IV-based [65, 66]	$M(m+2)$	$M(3m+4)$	Any
DCT-IV/DCT-II-based [65, 66]	$M(m+2)$	$M(3m+4)$	Any

4.6 Summary

The fast algorithms (radix-2, even-length, mixed-radix being the combination of radix-2 and radix-q algorithms, where q is an odd integer) for the efficient implementation of the forward/backward evenly stacked MDCT/MDST, oddly stacked MDCT/MDST, MLT, ELT, and MCLT block transforms have been presented. The emphasis was imposed particularly on basic steps, various tricks (trigonometric and algebraic), and approaches leading to the derivation of final formulae of a fast algorithm. For each fast algorithm have been presented: Complete formulae or the sparse block matrix factorization, the corresponding generalized signal flow graph, the total computational complexity, and a possible structural simplification of the algorithm have been provided. Appendices provide all the necessary supporting efficient fast computational structures including short-length modules for completing forward/backward MDCT/MDST, MLT, ELT, and MCLT efficient implementations.

Problems and Exercises

1. The backward MDCT/MDST computation in the evenly stacked system requires two N-point inverse complex FFTs [see Eqs. (4.9) and (4.10)]. The real-valued data sequence $\{\epsilon_k \ \overset{E}{c}_k \ e^{i\,(\frac{\pi}{2}+\frac{\pi}{N})k}\}$, $k = 0, 1, \ldots, \frac{N}{2} - 1$ and $\{\tau_k \ \overset{E}{s}_k \ e^{i\,(\frac{\pi}{2}+\frac{\pi}{N})k}\}$, $k = 1, 2, \ldots, \frac{N}{2}$, are complex-valued, whereby $\overset{E}{c}_{\frac{N}{2}} = \overset{E}{s}_0 = 0$. In [11] it is claimed that two N-point inverse complex FFTs can be combined into a single N-point inverse complex FFT. Is it possible?
2. For a given N, implement by computer programs the fast DFT/FFT-based algorithms for the simultaneous forward and backward evenly stacked MDCT/MDST computation defined by (4.6) and (4.7), and then the fast algorithm via two RVFFTs defined by (4.8). Verify their computational complexities. The required split-radix complex FFT and split-radix RVFFT fast algorithms with specified computational complexity can be found in [2].
3. For a given N, implement by computer programs the fast DCT-II-based algorithms for the evenly stacked forward/backward MDCT/MDST computation discussed in Sect. 4.2.3. Verify their computational complexities. The required DCT-II fast algorithms with specified computational complexity are presented in Appendix C.1.
4. Based on generalized signal flow graphs for the MDCT/MDST computation in the evenly stacked system shown in Figs. 4.1, 4.2, 4.3, and 4.4 for a given N, derive sparse matrix factorizations of the MDCT matrix, and then of the MDST matrix.
5. Derive (4.49) from (4.48), and then derive (4.57) from (4.56) using (4.45).
6. From (4.69) at first derive (4.70), and then (4.71).
7. For a given N, implement by computer programs the DFT/FFT-based fast algorithms for the forward/backward oddly stacked MDCT/MDST compu-

tation discussed in Sect. 4.3.2. Verify their computational complexities. The required split-radix complex FFT fast algorithms with specified computational complexity can be found in [2].

8. Derive (4.90) from (4.88) by expanding indicated cosine and sine transform kernels.
9. Derive the second equation from the first equation in (4.92) using indicated trigonometric identities.
10. Derive (4.99) from (4.98) using indicated trigonometric identities.
11. For a given N, implement by computer programs the fast DCT-II-based algorithms for the forward/backward MDCT/MDST computation in the oddly stacked system discussed in Sect. 4.3.3. Verify their computational complexities. The required DCT-II fast algorithms with specified computational complexity are presented in Appendix C.1.
12. For a given N, implement by computer programs the fast DCT-IV-based algorithms for the forward/backward MDCT/MDST computation in the oddly stacked system discussed in Sect. 4.3.4. Verify their computational complexities. The required DCT-IV fast algorithms with specified computational complexity are presented in Appendix C.2.
13. For a given N, implement by computer programs the fast DCT-IV/DCT-II-based algorithms for the forward/backward MDCT/MDST computation in the oddly stacked system discussed in Sect. 4.3.5. Verify their computational complexities. The required DCT-II fast algorithms with specified computational complexity are presented in Appendix C.1.
14. Derive (4.122) from (4.121).
15. Based on (4.124), derive equations in (4.125) from (4.123). Then, derive (4.127) and (4.128) from (4.125).
16. Derive $\{a_k\}$ and $\{b_k\}$ in (4.134) from (4.127) and (4.128) using (4.131) and (4.132).
17. Following the derivation procedure for $\{a_k^{(1)}\}$ given by (4.156) and $\{a_k^{(2)}\}$ given by (4.158), decompose $\{b_k^{(1)}\}$ and $\{b_k^{(2)}\}$ given by (4.155).
18. Verify the even symmetry property (4.163) of data sequence $\{y_n\}$ given by (4.162).
19. Verify the even anti-symmetry property (4.171) of data sequence $\{z_n\}$ given by (4.170).
20. For a given N, implement by computer programs (recursive) DIF, DIT mixed-radix and DIF radix-2 fast algorithms the forward/backward MDCT computation in the oddly stacked system discussed in Sect. 4.3.7. Verify their computational complexities.
21. Based on generalized signal flow graphs and fast algorithms for the MDCT/MDST computation in the oddly stacked system discussed in Sect. 4.3 for a given N, derive sparse matrix factorizations of the MDCT matrix, and then of the MDST matrix.
22. For a given N divisible by 4, implement by computer programs the recursive algorithms for the forward/backward MLT (MDCT/MDST) computation discussed in Sect. 4.3.8.

23. Derive (4.219) combining (4.217) and (4.218).
24. Using (4.229) and (4.233) derive (4.234) and (4.235).
25. The MDCT and MDST basis functions with the incorporated sine windowing function are given by (4.237). Derive their simplified forms given by (4.248) and (4.250) using indicated trigonometric identities.
26. Verify relations (4.264).
27. For a given N, implement by computer programs the fast MCLT algorithms for the simultaneous MDCT and MDST computation discussed in Sect. 4.5. Verify their computational complexities.

References

1. M. Bosi, R.E. Goldberg, *Introduction to Digital Audio Coding and Standards* (Springer Science+Business Media, New York, 2003)
2. H.S. Malvar, *Signal Processing with Lapped Transforms* (Artech House, Norwood, MA, 1992)
3. H. Malvar, A modulated complex lapped transform and its applications to audio processing, in *Proceedings of the IEEE ICASSP'99*, Phoenix, AR, May 1999, pp. 1421–1424
4. J.P. Princen, A.B. Bradley, Analysis/synthesis filter bank design based on time domain aliasing cancellation. IEEE Trans. Acoust. Speech Signal Process. **ASSP-34**(5), 1153–1161 (1986)
5. J.P. Princen, A.W. Johnson, A.B. Bradley, Subband/transform coding using filter bank designs based on time domain aliasing cancellation, in *Proceedings of the IEEE ICASSP'87*, Dallas, TX, April 1987, pp. 2161–2164
6. A. Spanias, T. Painter, V. Atti, *Audio Signal Processing and Coding* (Wiley-Interscience, Hoboken, NJ, 2007)

Fast Algorithms for the Evenly Stacked MDCT/MDST

7. V. Britanak, An efficient computing of oddly stacked MDCT/MDST computation via evenly stacked MDCT/MDST and vice versa. Signal Process. **85**(7), 1353–1374 (2005)
8. V. Britanak, K.R. Rao, A unified fast MDCT/MDST computation in the evenly stacked analysis/synthesis system. Circuits Syst. Signal Process. **21**(4), 415–426 (2002)
9. G. Davidson, L. Fielder, M. Antil, High-quality audio transform coding at 128 kbits/s, in *Proceedings of the IEEE ICASSP'90*, Alberquerque, NM, April 1990, pp. 1117–1120
10. G. Davidson, L. Fielder, M. Antil, Low-complexity transform coder for satellite link applications, in *89th AES Convention*, Los Angeles, CA, September 1990. Preprint #2966
11. G. Davidson, W. Anderson, A. Lovrich, A low-cost adaptive transform decoder implementation for high-quality audio, in *Proceedings of the IEEE ICASSP'92* vol. II, San Francisco, CA, April 1992, pp. 93–196
12. A.G. Elder, S.G. Turner, A real-time PC based implementation of AC-2 digital audio compression, in *95th AES Convention*, New York, NY, October 1993. Preprint #3773
13. T.D. Lookabaugh, M.G. Perkins, Application of the Princen–Bradley filter bank to speech and image compression. IEEE Trans. Acoust. Speech Signal Process. **38**(11), 1914–1926 (1990)
14. D. Ševič, M. Popovič, Improved implementation of the Princen–Bradley filter bank. IEEE Trans. Signal Process. **42**(11), 3260–3261 (1994)

15. K.C. Anup, B.A. Kumar, A new efficient implementation of TDAC synthesis filterbanks based on radix-2 FFT, in *Proceedings of the 14th EUSIPCO Signal Processing Conference*, Florence, September 2006
16. M. Bosi-Goldberg, Analysis/synthesis filtering system with efficient oddly stacked single sideband filter bank using time domain aliasing cancellation. US Patent Application No. 5890106, Dolby Labs, San Francisco, CA, March 1999
17. V. Britanak, Improved and extended mixed-radix decimation in frequency fast MDCT algorithm. Comput. Inform. **29**(6), 1001–1012 (2010)
18. V. Britanak, K.R. Rao, A new fast algorithm for the unified forward and inverse MDCT/MDST computation. Signal Process. **82**(3), 433–459 (2002)
19. D.-Y. Chan, J.-F. Yang, S.-Y. Chen, Regular implementation algorithms of time domain aliasing cancellation. IEE Proc. Vision Image Signal Process. **143**(6), 387–392 (1996)
20. M.-H. Cheng, Y.-H. Hsu, Fast IMDCT and MDCT algorithms – a matrix approach. IEEE Trans. Signal Process. **51**(1), 221–229 (2003)
21. R.K. Chivukula, Y.A. Reznik, Efficient implementation of a class of MDCT/IMDCT filter banks for speech and audio coding applications, in *Proceedings of the IEEE ICASSP'2008*, Las Vegas, NV, March–April 2008, pp. 213–216
22. Y.-K. Cho, T.-H. Song, H.-S. Kim, An optimized algorithm for computing the modified discrete cosine transform and its inverse transform, in *Proceedings of the IEEE Region 10 Conference TENCON'2004*, Chiang Mai, November 2004, pp. 626–628
23. S. Cramer, R. Gluth, Computationally efficient real-valued filter banks based on a modified O^2 DFT, in *Proceedings of EUSIPCO'90, Signal Processing V: Theories and Applications*, Elsevier Science Publishers B.V., Barcelona, September 1990, pp. 585–588
24. P. Duhamel, Y. Mahieux, J.P. Petit, A fast algorithm for the implementation of filter banks based on time domain aliasing cancellation, in *Proceedings of the IEEE ICASSP'91*, Toronto, May 1991, pp. 2209–2212
25. R. Gluth, Regular FFT-related transform kernels for DCT/DST-based polyphase filter banks, in *Proceedings of the IEEE ICASSP'91*, Toronto, May 1991, pp. 2205–2208
26. Z.G. Gui, Y. Ge, D.Y. Zhang, J.S. Wu, Generalized fast mixed-radix algorithm for the computation of forward and inverse MDCTs. Signal Process. **92**(2), 363–373 (2012)
27. Y.-T. Hwang, S.-C. Lai, A novel MDCT/IMDCT computing kernel design, in *Proceedings of the IEEE Workshop on Signal Processing Systems Design and Implementation*, Athens, November 2005, pp. 526–531
28. M. Iwadare, A. Sugiyama, F. Hazu, A. Hirano, T. Nishitani, A 128 kb/s hi-fi CODEC based on adaptive transform coding with adaptive block size MDCT. IEEE J. Sel. Areas Commun. **10**(1), 138–144 (1992)
29. H.-S. Kim, Y.-K. Cho, W.-P. Lee, A new optimized algorithm for computation of MDCT and its inverse transform, in *Proceedings of the IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS'2004)*, Seoul, November 2004, pp. 528–530
30. S.-W. Lee, Improved algorithm for efficient computation of the forward and backward MDCT in MPEG audio coder. IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process. **48**(10), 990–994 (2001)
31. T. Li, R. Zhang, R. Yang, H. Huang, F. Lin, A unified computing kernel for MDCT/IMDCT in modern audio coding standards, in *Proceedings of the IEEE International Symposium on Communication and Information Technologies (ISCIT'2007)*, Sydney, October 2007, pp. 546–550
32. B. Lincoln, An experimental high fidelity perceptual audio coder, Project in MUS420 Win97, Center for Computer Research in Music and Acoustics, Stanford University, CA 94305-8180 (1998). Web site: http://ccrma-www.stanford.edu/~bosse/proj/proj.html
33. C.-M. Liu, W.-C. Lee, A unified fast algorithm for cosine modulated filter banks in current audio coding standards. J. Audio Eng. Soc. **47**(12), 1061–1075 (1999)

34. Y. Mahieux, J.P. Petit, High-quality audio transform coding at 64 kbps. IEEE Trans. Commun. **42**(11), 3010–3019 (1994)
35. V. Nikolajevič, G. Fettweis, Improved implementation of MDCT in MP3 audio coding, in *Proceedings of the IEEE 10th Asian–Pacific Conference on Communications and 5th International Symposium on Multi-Dimensional Mobile Communications (APCC/MDMC'2004)*, vol. 1, Tsinghua University, Beijing, August–September 2004, pp. 309–312
36. D. Ševič, M. Popovič, A new efficient implementation of the oddly stacked Princen–Bradley filter bank. IEEE Signal Process Lett. **1**(11), 166–168 (1994)
37. X. Shao, S.G. Johnson, Type-IV DCT, DST and MDCT algorithms with reduced number of arithmetic operations. Signal Process. **88**(6), 1313–1326 (2008)
38. H. Shu, X. Bao, C. Toumoulin, L. Luo, Radix-3 algorithm for the fast computation of forward and inverse MDCT. IEEE Signal Process. Lett. **14**(2), 93–96 (2007)
39. T. Sporer, K. Brandenburg, B. Edler, The use of multirate filter banks for coding of high quality digital audio, in *Proceedings of the 6th European Signal Processing Conference (EUSIPCO)*, vol. 1, Amsterdam, June 1992, pp. 211–214
40. T.K. Truong, P.D. Chen, T.C. Cheng, Fast algorithm for computing the forward and inverse MDCT in MPEG audio coding. Signal Process. **86**(5), 1055–1060 (2006)
41. P.-S. Wu, Y.-T. Hwang, Efficient IMDCT core designs for audio signal processing, in *Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS'2003)*, Seoul, August 2003, pp. 275–280
42. J.-S. Wu, H.-Z. Shu, L. Senhadji, L.-M. Luo, Mixed-radix algorithm for the computation of forward and inverse MDCTs. IEEE Trans. Circuits Syst. Regul. Pap. **56**(4), 784–794 (2009)
43. G. Wu, E.-H. Yang, A new efficient method of computing MDCT in MP3 audio coding, in *Proceedings of the IEEE International Conference on Multimedia Information Networking and Security (MINES'2010)*, Nanjing Yiangsu, November 2010, pp. 63–67
44. J. Wu, L. Wang, L. Senhadji, H. Shu, Improved radix-3 decimation-in-frequency algorithm for the fast computation of forward and inverse MDCT, in *Proceedings of the IEEE International Conference on Audio, Language and Image Processing (ICALIP'2010)*, Shanghai, November 2010, pp. 694–699

Fast MLT and ELT Algorithms

45. C.Y. Jing, H.-M. Tai, Fast algorithm for computing modulated lapped transform. Electron. Lett. **37**(12), 796–797 (2001)
46. C.Y. Jing, H.-M. Tai, Design and implementation of a fast algorithm for modulated lapped transform. IEE Proc. Image Signal Process. **149**(1), 27–32 (2002)
47. H.S. Malvar, Lapped transforms for efficient transform/sub-band coding. IEEE Trans. Acoust. Speech Signal Process. **38**(6), 969–978 (1990)
48. H.S. Malvar, Extended lapped transforms: fast algorithms and applications, in *Proceedings of the IEEE ICASSP'91*, Toronto, May 1991, pp. 1797–1800
49. H.S. Malvar, Fast algorithm for modulated lapped transform. Electron. Lett. **27**(9), 775–776 (1991)
50. H.S. Malvar, Extended lapped transforms: properties, applications, and fast algorithms. IEEE Trans. Signal Process. **40**(11), 2703–2714 (1992)
51. H.S. Malvar, Fast algorithms for orthogonal and biorthogonal modulated lapped transforms, in *Proceedings of IEEE Symposium on Advances in Digital Filtering and Signal Processing*, Victoria, June 1998, pp. 159–163

Recursive/Regressive MDCT/MDST Filter Structures (oddly Stacked System)

52. C.-H. Chen, B.-D. Liu, J.-F. Yang, Recursive architectures for realizing modified discrete cosine transform and its inverse. IEEE Trans. Circuits Syst. II Analog Digit. Signal Process. **50**(1), 38–45 (2003)
53. H.C. Chiang, J.C. Liu, Regressive implementations for the forward and inverse MDCT in MPEG audio coding. IEEE Signal Process. Lett. **3**(4), 116–118 (1996)
54. T.W. Fox, A. Carriera, Goertzel implementations of the forward and inverse modified discrete cosine transform, in *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE'2004)*, vol. 4, Niagara Falls, May 2004, pp. 2371–2374
55. R. Koenig, T. Stripf, J. Becker, A novel recursive algorithm for bit-efficient realization of arbitrary length inverse modified cosine transforms, in *Proceedings of the International Conference on Design, Automation and Test in Europe (DATE'2008)*, Munich, March 2008, pp. 604–609
56. S.-C. Lai, S.-F. Lei, C.-H. Luo, Common architecture design of novel recursive MDCT and IMDCT algorithms for application to AAC, AAC in DRM, and MP3 codecs. IEEE Trans. Circuits Syst. Express Briefs **56**(10), 793–797 (2009)
57. S.-C. Lai, Y.-P. Yeh W.-C. Tseng, S.-F. Lei, Low-cost and high-accuracy design of fast recursive MDCT/MDST/IMDCT/IMDST algorithms and their realization. IEEE Trans. Circuits Syst. Express Briefs **59**(1), 65–69 (2012)
58. S.-F. Lei, S.-C. Lai, Y.-T. Hwang, C.-H. Luo, A high-precision algorithm for the forward and inverse MDCT using the unified recursive architecture, in *Proceedings of the IEEE International Symposium on Consumer Electronics*, Algarve, April 2008, pp. 1–4
59. S.-F. Lei, S.-C. Lai, P.-Y. Cheng, C.-H. Luo, Low complexity and fast computation for recursive MDCT and IMDCT algorithms. IEEE Trans. Circuits Syst. Express Briefs **57**(7), 571–575 (2010)
60. H. Li, P. Li, Y. Wang, An efficient hardware accelerator architecture for implementing fast IMDCT computation. Signal Process. **90**(8), 2540–2545 (2010)
61. V. Nikolajevic, G. Fettweis, New recursive algorithms for the forward and inverse MDCT, in *Proceedings of the IEEE Workshop on Signal Processing Systems: Design and Implementation (SiPS'2001)*, Antwerp, September 2001, pp. 51–57
62. V. Nikolajevič, G. Fettweis, Computation of forward and inverse MDCT using Clenshaw's recurrence formula. IEEE Trans. Signal Process. **51**(5), 1439–1444 (2003)
63. V. Nikolajevič, G. Fettweis, New recursive algorithms for the unified forward and inverse MDCT/MDST. J. VLSI Sig. Proc. Systems Signal Image Video Technol. **34**(3), 203–208 (2003)

Fast MCLT Algorithms

64. V. Britanak, New recursive fast radix-2 algorithm for the modulated complex lapped transform. IEEE Trans. Signal Process. **60**(12), 6703–6708 (2012)
65. X. Chen, Q. Dai, A novel DCT-based algorithm for computing the modulated complex lapped transform. IEEE Trans. Signal Process. **54**(11), 4480–4484 (2006)
66. Q. Dai, X. Chen, New algorithm for modulated complex lapped transform with symmetrical window function. IEEE Signal Process. Lett. **11**(12), 925–928 (2004)
67. X. Dai, M.D. Wagh, Fast algorithm for modulated complex lapped transform. IEEE Signal Process Lett. **16**(1), 30–33 (2009)
68. H. Malvar, Fast algorithm for the modulated complex lapped transform. IEEE Signal Process. Lett. **10**(1), 8–10 (2003)

69. H. Shu, J. Wu, L. Senhadji, L. Luo, New fast algorithm for modulated complex lapped transform with sine windowing function. IEEE Signal Process Lett. **16**(2), 93–96 (2009)
70. H.-M. Tai, C. Jing, Design and efficient implementation of a modulated complex lapped transform using pipeline technique. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **E84-A**(5), 1280–1287 (2001)

Supporting Literature

71. M.F. Aburdene, J. Zheng, R.J. Kozick, Computation of discrete cosine transform using Clenshaw's recurrence formula. IEEE Signal Process Lett. **2**(8), 155–156 (1995)
72. G. Bi, Y. Zeng, *Transforms and Fast Algorithms for Signal Analysis and Representation*, chap. 6 (Birkhäuser, Boston, 2004), pp. 207–245
73. E.O. Brigham, FFT transform applications (Chapter 9), in *The Fast Fourier Transform and its Applications* (Prentice-Hall, Englewood Cliffs, NJ, 1988), pp. 188–191
74. V. Britanak, K.R. Rao, The fast generalized discrete Fourier transforms: a unified approach to the discrete sinusoidal transforms computation. Signal Process. **79**(12), 135–150 (1999)
75. V. Britanak, P. Yip, K.R. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations* (Academic, Elsevier Science, Amsterdam, 2007)
76. L.-P. Chau, W.-C. Siu, Recursive algorithm for the discrete cosine transform with general lengths. Electron. Lett. **30**(3), 197–198 (1994). Errata, Electron. Lett. **30**(7), 608 (1994)
77. L.-P. Chau, W.-C. Siu, Efficient recursive algorithm for the inverse discrete cosine transform. IEEE Signal Process Lett. **7**(10), 276–277 (2000)
78. R.E. Crochiere, L.E. Rabiner, Multirate techniques in filter banks and spectrum analyzers and synthesizers (Chapter 7), in *Multirate Digital Signal Pocessing* (Prentice-Hall, Englewood Cliffs, NJ, 1983), pp. 289–400
79. P. Duhamel, Implementation of 'Split-Radix' FFT algorithms for complex, real, and real-symmetric data. IEEE Trans. Acoust. Speech Signal Process. **ASSP-34**(2), 285–295 (1986)
80. P. Duhamel, B. Piron, J.M. Etcheto, On computing the inverse DFT. IEEE Trans. Acoust. Speech Signal Process. **ASSP-36**(2), 285–286 (1988)
81. G. Goertzel, An algorithm for the evaluation of finite trigonometric series. Am. Math. Mon. **65**, 34–35 (1958)
82. S.G. Johnson, M. Frigo, A modified split-radix FFT with fewer arithmetic operations. IEEE Trans. Signal Process. **55**(1), 111–119 (2007)
83. C.W. Kok, Fast algorithm for computing discrete cosine transform. IEEE Trans. Signal Process. **45**(3), 757–760 (1997)
84. K.R. Rao, J.J. Hwang, *Techniques and Standards for Image, Video, and Audio Coding* (Prentice Hall, Upper Saddle River, NJ, 1996)
85. H.V. Sorensen, M.T. Heideman, C.S. Burrus, On computing the split-radix FFT, IEEE Trans. Acoust. Speech Signal Process. **ASSP-34**(1), 152–156 (1986)
86. H.V. Sorensen, D.L. Jones, M.T. Heideman, Real-valued fast Fourier transform algorithms. IEEE Transactions on Acoustics, Speech and Signal Processing **ASSP-35**(6), 849–863 (1987)
87. Z. Wang, G.A. Julien, W.C. Miller, Recursive algorithm for the forward and inverse discrete cosine transform with arbitrary length. IEEE Signal Process. Lett. **1**(7), 101–102 (1994)

Chapter 5
Efficient Implementations of Cosine-Modulated Pseudo-QMF and MLT (MDCT) Filter Banks in MP3

5.1 Introduction

The MPEG-1 audio compression algorithm [2], finalized in 1992, is the first established international coding standard for high-quality compression and decompression of digital audio signals. The MPEG-2 audio standard [3], finalized in 1994, extends the multichannel capabilities not offered by MPEG-1 audio. MPEG-1/2 algorithms exploit perceptual audio coding principles and they involve three distinct layers for compression. Layer I forms the most basic compression algorithm, while layers II and III are enhancements that use some elements of layer I. Each successive layer improves the compression performance but at the cost of greater encoder and decoder complexity [1, 8, 16, 17]. Essentially, Layer III of MPEG-1/2, well known as MP3 standard, has become key technology to realize audio decoders for various computer platforms (music distribution via internet) and consumer electronics (portable MP3 players and multimedia systems).

For the time-to-frequency decomposition of digital audio signals the MP3 standard [3] employs the hybrid filter bank consisting of the near-perfect reconstruction cosine-modulated polyphase quadrature mirror filter (QMF) analysis/synthesis filter banks [1, 8] referred also to as the pseudo-QMF banks [1], and the perfect reconstruction modulated lapped transform (MLT) [5, 6] which is actually the modified discrete cosine transform (MDCT) based on the time domain aliased cancellation (TDAC) concept [4, 7] and associated with the sine windowing function. The pseudo-QMF bank, common to all three layers of MPEG-1/2 audio, decomposes the input audio signal into 32 equally spaced frequency sub-bands. Each sub-band is then coded either to single groups of 12-sample blocks (in layer I) or to groups of three successive 12-sample blocks (in layer II). For 32 sub-bands this results in two data frames of $32 \times 12 = 384$ and $32 \times 36 = 1152$ samples [1, 8, 16, 17]. In MP3 standard [3], the outputs of pseudo-QMF bank are further processed by the MLT (MDCT) filter bank. The MLT (MDCT) filter bank operates on the block of 12 samples (the short block) or the block of 36 samples (the long block). The

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1_5

long block allows greater frequency resolution for audio signals with stationary characteristics, while the short block provides better time resolution for transient audio signals. Basic windowing operation is defined for the long block and short block. During transient signals, the long block is replaced by a series of three overlapped short blocks thus maintaining the same total number of samples as for the long block. Each of the three short blocks is then windowed separately. Switching between long and short blocks is not instantaneous. In order to ensure smooth transition between long and short blocks and vice versa, transient blocks (long-to-short and short-to-long blocks having the same size as the long block) are specifically defined and windowed [1, 8]. In principle, the pseudo-QMF and MLT (MDCT) analysis/synthesis filter banks are the most time-consuming operations in the encoder/decoder, and therefore, their fast and efficient implementations with a simple and regular computational structure have played an important role for their real-time hardware/software implementations.

With the popularity of MP3 standard after its finalization in 1992 and 1994, much research has been devoted to develop the efficient implementations of analysis/synthesis pseudo-QMF and MLT (MDCT) filter banks. The efficient implementations of analysis/synthesis pseudo-QMF banks with additional possible optimizations are almost completely presented in [9–16, 18, 19, 53]. On the other hand, many fast MLT (MDCT) algorithms have been developed in the time period 1990–2012 which could/can be adopted for the efficient MLT (MDCT) implementation in MP3 audio [5, 6, 20–88]. Almost all existing fast MLT (MDCT) algorithms employ a discrete sinusoidal unitary transform of reduced size such as the discrete Fourier transform (DFT) and its fast implementation, FFT [20, 21, 37, 42, 43, 46, 47, 52, 57, 59, 61, 69], the discrete Hartley transform (DHT), and its fast implementation, FHT [62], type II discrete cosine/sine transform (DCT-II/DST-II) [33], type IV discrete cosine/sine transform (DCT-IV/DST-IV) [5, 6, 25, 27, 31, 49, 53, 63], or they are obtained in DCT-IV/DCT-II combination [34, 35, 45, 51, 54, 65, 69]. Other fast algorithms enabling an efficient MDCT implementation in MP3 are: mixed-radix (combined radix-2/3) MDCT algorithms [28, 44, 60, 67], algorithms based on the corresponding evenly stacked MDCT [26, 39, 64], and algorithms based on recursive/regressive filter structures [70–88]. It is widely accepted that the DCT-IV-based fast MLT (MDCT) algorithms are the most efficient both in terms of the computational complexity and structural simplicity [30, 31].

For the short and long audio blocks ($N = 12$ or 36) whose sizes are not powers of two (sizes are even integers divisible by 2, 3, 4, and 6), the original ISO source code [2, 3] recommends to implement the forward and backward MLT (MDCT) block transforms "as-is." This direct implementation requires 72 multiplications plus 66/60 additions for the short block, and 648 multiplications plus 630/612 additions for the long block. In the time period 1998–2012 several efficient MLT (MDCT) implementations tailored directly to the MP3 decoder only [41, 55, 58] or both the MP3 encoder and decoder [24–26, 28, 29, 31, 32, 36, 38, 40, 44, 48, 50, 56, 64, 66–68] have been developed. Although not surprising, it is less known that after the MPEG-1/2 standards

finalization, the proposed DFT/FFT-based MDCT algorithms [37, 42, 43] could be adopted for quite efficient MLT (MDCT) implementations in MP3 (required 3/9-point complex-valued DFT modules have been presented in [120, 127, 135, 139]), but these implementations did never appear in the literature. The first breakthrough in the efficient implementation of the backward MLT (MDCT) in MP3 decoder appeared in [58]. Using the symmetry of MDCT transform kernel the number of arithmetic operations was reduced by half compared to the direct implementation. Exploiting the approach [34, 53] a more efficient implementation of the backward MDCT in MP3 decoder has been reported in the unpublished work [55]. It required 13 multiplications and 27 additions for the short block, and 81 multiplications and 149 additions for the long block. The research interest in this area has dramatically increased since a regular and efficient forward/backward MLT (MDCT) implementation in MP3 based on the fast algorithm [33] has been proposed in [32]. After some refinements [24] it required 11 multiplications, 39/33 additions plus 2 shifts for the short block, and 43 multiplications, 165/147 additions plus 4 shifts for the long block. It has been sequentially improved/optimized in terms of the arithmetic complexity and structural simplicity [29, 31, 36, 48, 50, 56, 64]. As the best result, using identical computational structures, the forward/backward MLT (MDCT) computation required 11 multiplications, 27/21 additions, and 2 shifts for the short block, and 43 multiplications, 129/111 additions, and 4 shifts for the long block. In 2008, an MDCT hardware accelerator for MP3 audio has been presented in [38, 40] which is based on the most efficient DCT-IV algorithms in terms of the minimal multiplicative complexity known up to now. Specifically, the forward/backward MLT (MDCT) computation requires 9 multiplications, 27/21 additions, and 2 shifts for the short block (the same multiplicative complexity was achieved in [29]), and 36 multiplications, 148/130 additions, and 2 shifts for the long block. It is worth to note that in 2010, a new efficient method of computing the MDCT in MP3 has been proposed in [66]. Based on a recursive sparse block matrix factorization of scaled DCT-II matrix, the forward/backward MLT (MDCT) computation requires 10 multiplications, 28/22 additions, and 1 shift for the short block, and 37 multiplications, 135/117 additions, and 2 shifts for the long block.

Aforementioned efficient MLT (MDCT) implementations have led to low-complexity MP3 decoders realized by software [92, 102] and many real-time MP3 audio encoders/decoders realized on high-performance programmable DSP processors [89, 90, 95, 96, 104, 109, 112], universal RISC-based ARM processors [93, 97, 105, 106, 111], and implemented into VLSI full-custom ASIC [39, 40, 71, 94, 98–100, 107, 108, 110] or semi-custom circuits (FPGA) [76, 91, 98, 101, 103].

In this chapter, various efficient implementations of the forward and backward MLT (MDCT) block transforms in the MP3 audio coding standard developed in the time period 1990–2012 are described and compared, including the efficient implementation of pseudo-QMF banks for completeness. The efficient MLT (MDCT) implementations are discussed in the context of the complete (fast) analysis/synthesis MLT (MDCT) filter banks in the MP3 encoder and decoder. They are classified into the following categories:

- DFT/FFT-based efficient MLT (MDCT) implementations,
- DCT-II/DST-II-based efficient MLT (MDCT) implementations,
- DCT-IV-based efficient MLT (MDCT) implementations,
- DCT-IV/(scaled)DCT-II-based efficient MLT (MDCT) implementations,
- Efficient MLT (MDCT) implementations based on the corresponding evenly stacked MDCT [137],
- Mixed-radix (combined radix-2/3) efficient MLT (MDCT) implementations,
- Efficient MLT (MDCT) implementations based on recursive/regressive filter structures.

In general, for each efficient forward/backward MLT (MDCT) implementations are presented: Complete formulae or sparse (block) matrix factorizations, the corresponding signal flow graph for short audio block and the total arithmetic complexity as well as the useful comments related to improving the arithmetic complexity and a possible structural simplification of the algorithm are provided. Finally, the fast analysis and synthesis MLT (MDCT) filter banks for the MP3 encoder and decoder are discussed in detail. Appendices provide all the necessary supporting efficient optimized short-length computational modules and tools for completing efficient forward/backward MLT (MDCT) implementations.

5.2 Definitions and Properties of Filter Banks Used in MP3

5.2.1 Analysis/Synthesis Pseudo-QMF Banks

The standard specifications of analysis/synthesis pseudo-QMF banks for the encoder/decoder (also called the matrixing operations for sub-band filtering) are presented in [1–3, 16, 17]. In general, they can be reduced to the block transforms defined below.

Pseudo-QMF banks as the forward and backward block transforms are, respectively, defined as [1–3, 16, 17]

$$p_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n - \frac{N}{4}\right)(2k+1)\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{5.1}$$

$$\hat{x}_n = \sum_{k=0}^{\frac{N}{2}-1} p_k \cos\left[\frac{\pi}{N}\left(n + \frac{N}{4}\right)(2k+1)\right], \qquad n = 0, 1, \ldots, N-1, \tag{5.2}$$

where $\{x_n\}$ is the input data sequence multiplied by filter coefficients explicitly given in [1–3, 16, 17], $\{p_k\}$ are sub-band transform coefficients, and $\{\hat{x}_n\}$ is the time domain aliased data sequence which does not correspond to the original data

sequence. For MPEG-1/2 audio coding standards $N = 64$. Substituting $N - 1 - k$ for k into (5.1) we get

$$p_{N-1-k} = p_k, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{5.3}$$

demonstrating that the sequence of sub-band transform coefficients $\{p_k\}$ has the even symmetry property. From (5.3) it follows that only $\frac{N}{2}$ coefficients are unique in the sub-band transform sequence.

5.2.2 TDAC Analysis/Synthesis MDCT Filter Banks

The complete TDAC analysis and synthesis MDCT filter banks are, respectively, defined as [7]

$$c_k^{(t)} = \sqrt{\frac{4}{N}} \sum_{n=0}^{N-1} w_n \, x_n^{(t)} \cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)(2k+1)\right], \quad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{5.4}$$

$$\hat{x}_n^{(t)} = \sqrt{\frac{4}{N}} \, w_n \sum_{k=0}^{\frac{N}{2}-1} c_k^{(t)} \cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)(2k+1)\right], \quad n = 0, 1, \ldots, N-1, \tag{5.5}$$

where $\{w_n\}$ is a symmetric windowing function, and a superscript t denotes the data-block number. N, the length of the data block is assumed to be an even integer. In the analysis filter bank given by (5.4), for the tth data block, N windowed time domain samples $\{x_n^{(t)}\}$ are used to calculate $\frac{N}{2}$ unique transform coefficients $\{c_k^{(t)}\}$. Vice versa, the tth block of $\frac{N}{2}$ transform coefficients $\{c_k^{(t)}\}$ is used to calculate N windowed time domain aliased samples $\{\hat{x}_n^{(t)}\}$ with the synthesis filter bank given by (5.5). The complete TDAC analysis/synthesis MDCT filter bank provides critical sampling, overlapping of adjacent data blocks by $\frac{N}{2}$ samples, possesses energy-packing capability and allows perfect reconstruction. To achieve critical sampling in combination with overlapping data blocks, subsampling in the frequency domain is performed by the analysis MDCT filter bank. This subsampling introduces aliasing in the time domain, but this can be cancelled by overlapping and adding of two adjacent recovered data blocks by the synthesis filter bank. This procedure is known as TDAC [7]. It is important to note that the time domain aliased data sequence $\{\hat{x}_n^{(t)}\}$ does not correspond to the original data sequence $\{x_n^{(t)}\}$.

Two succeeding data blocks t and $t+1$ are overlapped by $\frac{N}{2}$ samples so that for each data block $\frac{N}{2}$ new time domain samples are processed. For a smooth block overlapping a windowing function $\{w_n\}$ is applied to $\{x_n^{(t)}\}$ and $\{x_n^{(t+1)}\}$ (the so-called windowing&overlap procedure). By applying analysis and synthesis MDCT filter banks, a time domain aliasing error is introduced which is independent for each half of the data block. This leads to the realization of adaptive block-size switching (processing with variable-block size) [125, 126, 133]. Flexible dynamic block-size switching is an important concept to reduce pre-echo effects in MDCT-based audio codecs [1, 125]. The aliasing error is cancelled (or perfect reconstruction is accomplished) by adding outputs of the synthesis MDCT filter bank of two succeeding windowed data blocks t and $t+1$ in the overlapped part (the so-called windowing&overlap&add procedure) as follows:

$$x_n^{(t+1)} = x_{\frac{N}{2}+n}^{(t)} = \hat{x}_{\frac{N}{2}+n}^{(t)} + \hat{x}_n^{(t+1)}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{5.6}$$

To ensure TDAC, the windowing functions of two succeeding data blocks have to satisfy the so-called perfect-reconstruction conditions in their overlapped part. A sufficient condition for TDAC is given by [4, 7]

$$\begin{aligned} &w_n^2 + w_{\frac{N}{2}+n}^2 = 1, \\ &w_n = w_{N-1-n}, \quad \text{or} \quad w_{\frac{N}{2}+n} = w_{\frac{N}{2}-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{5.7}$$

A symmetric windowing function used in MP3 audio coding standard [2, 3] both for the analysis and synthesis MDCT filter banks, satisfying perfect reconstruction conditions (5.7) and producing aliasing cancellation, is the sine windowing function defined by [5, 6]

$$w_n = \sin\left[\frac{\pi(2n+1)}{2N}\right], \quad n = 0, 1, \ldots N-1. \tag{5.8}$$

5.2.3 *The Forward and Backward MDCT and MLT Block Transforms*

When developing fast computational structures for an efficient implementation of any analysis/synthesis filter banks, they are frequently considered as block transforms applied to a single data block. Consequently, the data-block number t from (5.4) and (5.5) is omitted. The input data sequence $\{x_n\}$, $n = 0, 1, \ldots, N-1$ is assumed to be windowed by the sine windowing function given by (5.8) before its transformation. The forward and backward MDCT block transforms are, respectively, specified in simplified forms as

$$c_k = \sqrt{\frac{4}{N}} \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \quad k = 0, 1, \ldots, \frac{N}{2}-1, \tag{5.9}$$

$$\hat{x}_n = \sqrt{\frac{4}{N}} \sum_{k=0}^{\frac{N}{2}-1} c_k \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \quad n = 0, 1, \ldots, N-1. \tag{5.10}$$

The data sequence $\{\hat{x}_n\}$ recovered by the backward MDCT represents the time domain aliased data sequence. The forward and backward MDCT block transforms are actually a pseudoinverse pair [26]. General mathematical properties, matrix representations, and special (peculiar) properties of the MDCT are discussed in detail in [31] (see also Chap. 3). The forward/backward MDCT associated with the sine windowing function is equivalent to the forward/backward MLT [5, 6]. The original definition of MLT is obtained simply by substituting $N = 2M$ into Eqs. (5.9) and (5.10). The forward and backward MLT block transforms are, respectively, defined as [5, 6]

$$c_k = \sqrt{\frac{2}{M}} \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2M}\left(n+\frac{M+1}{2}\right)(2k+1)\right], \quad k = 0, 1, \ldots, M-1, \tag{5.11}$$

$$\hat{x}_n = \sqrt{\frac{2}{M}} \sum_{k=0}^{M-1} c_k \cos\left[\frac{\pi}{2M}\left(n+\frac{M+1}{2}\right)(2k+1)\right], \quad n = 0, 1, \ldots, N-1, \tag{5.12}$$

where M is the number of transform coefficients. Based on the aforementioned equivalence between the MDCT and MLT, these abbreviations in subsequent text are exchangeable.

When $\frac{N}{2}$ is even, the MDCT sequence $\{c_k\}$ has even anti-symmetry property given by [31, 33]

$$c_{N-k-1} = -c_k, \qquad k = 0, 1, \ldots, \frac{N}{2}-1, \tag{5.13}$$

while the time domain aliased data sequence $\{\hat{x}_n\}$ recovered by the backward MDCT has the following symmetries [31, 33]

$$\hat{x}_n = -\hat{x}_{\frac{N}{2}-1-n}, \qquad \hat{x}_{N-1-n} = \hat{x}_{\frac{N}{2}+n}, \qquad n = 0, 1, \ldots, \frac{N}{4}-1. \tag{5.14}$$

From (5.13) it follows that only $\frac{N}{2}$ coefficients are unique in the MDCT sequence. Further, from (5.14) it can be seen that the time domain aliased data sequence $\{\hat{x}_n\}$ exhibits two local symmetries (odd symmetry in the first half and even symmetry in the second half). From the algorithmic point of view this means that it is sufficient to compute only the time domain aliased samples $\hat{x}_n$ and $\hat{x}_{\frac{N}{2}+n}$ for $n = 0, 1, \ldots, \frac{N}{4} - 1$ by the backward MDCT.

5.3 Efficient Implementations of Pseudo-QMF Banks in MP3

The standard implementations of analysis/synthesis pseudo-QMF banks for the encoder/decoder (or matrixing operations for sub-band filtering) in the form of flow charts are specified in [1–3, 8]. The developed efficient implementations of analysis/synthesis pseudo-QMF banks based on the DCT-II, and its inverse, DCT-III, of reduced size can be found in [9, 12–16, 18, 19, 53].

Consider the forward pseudo-QMF bank given by (5.1). Applying the following permutation to the input data sequence $\{x_n\}$ [13, 53]

$$y_n = \begin{cases} x_{\frac{N}{4}+n}, & n = 0, 1, \ldots, \frac{3N}{4} - 1, \\ -x_{n-\frac{3N}{4}}, & n = \frac{3N}{4}, \frac{3N}{4} + 1, \ldots, N - 1, \end{cases} \tag{5.15}$$

where N is divisible by 4, we have

$$\begin{aligned} p_k &= \sum_{n=0}^{\frac{3N}{4}-1} x_{\frac{N}{4}+n} \cos\left[\frac{\pi(2k+1)n}{N}\right] + \sum_{n=\frac{3N}{4}}^{N-1} -x_{n-\frac{3N}{4}} \cos\left[\frac{\pi(2k+1)(n-N)}{N}\right] \\ &= y_0 + \sum_{n=1}^{N-1} y_n \cos\left[\frac{\pi(2k+1)n}{N}\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{5.16}$$

Substituting $N - n$ for $n = 1, 2, \ldots, \frac{N}{2} - 1$, into the sum on the right-hand side of (5.16), we finally get

$$\begin{aligned} p_k &= y_0 + \sum_{n=1}^{\frac{N}{2}-1} (y_n - y_{N-n}) \cos\left[\frac{\pi(2k+1)n}{2(N/2)}\right] = \sum_{n=0}^{\frac{N}{2}-1} y'_n \cos\left[\frac{\pi(2k+1)n}{2(N/2)}\right], \\ & k = 0, 1, \ldots, \frac{N}{2} - 1, \end{aligned} \tag{5.17}$$

where

$$y'_n = \begin{cases} y_0, & n = 0, \\ y_n - y_{N-n}, & n = 1, 2, \ldots, \frac{N}{2} - 1. \end{cases} \tag{5.18}$$

The cosine transform kernel in (5.17) is recognized as an unnormalized $\frac{N}{2}$-point DCT-III) of $\{y'_n\}$.

Now consider the backward pseudo-QMF bank given by (5.2). Since the DCT-III is an orthogonal transform, then based on (5.17) the sequence $\{y'_n\}$ can be recovered by the inverse DCT-III of $\{p_k\}$, i.e., by the DCT-II of $\{p_k\}$ as

$$y'_n = \sum_{k=0}^{\frac{N}{2}-1} p_k \cos\left[\frac{\pi(2k+1)n}{2(N/2)}\right], \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{5.19}$$

From (5.19) it can be seen that $y'_{\frac{N}{2}} = 0$ for $n = \frac{N}{2}$ and $y'_n = -y'_{N-n}$ for $n = \frac{N}{2} + 1, \ldots, N-1$. Let us introduce a new extended sequence $\{y_n\}$ defined as

$$y_n = \begin{cases} y'_n, & n = 0, 1, \ldots, \frac{N}{2} - 1, \\ 0, & n = \frac{N}{2}, \\ -y'_{N-n}, & n = \frac{N}{2} + 1, \ldots, N-1. \end{cases} \tag{5.20}$$

Note that the sequence $\{y_n\}$ exhibits the anti-symmetry property at the point $\frac{N}{2}$. From the definition of backward pseudo-QMF bank given by (5.2) and (5.19) we have

$$\begin{aligned} \hat{x}_n &= \sum_{k=0}^{\frac{N}{2}-1} p_k \cos\left[\frac{\pi}{N}\left(n + \frac{N}{4}\right)(2k+1)\right] = y_{\frac{N}{4}+n} \quad n = 0, 1, \ldots, \frac{3N}{4} - 1, \\ \hat{x}_n &= \sum_{k=0}^{\frac{N}{2}-1} p_k \cos\left[\frac{\pi}{N}\left(n - \frac{3N}{4} + N\right)(2k+1)\right] = -y_{n-\frac{3N}{4}} \quad n = \frac{3N}{4}, \ldots, N-1, \end{aligned} \tag{5.21}$$

and the time domain aliased data sequence $\{\hat{x}_n\}$ can be recovered from $\{y_n\}$ as

$$\hat{x}_n = \begin{cases} y_{\frac{N}{4}+n}, & n = 0, 1, \ldots, \frac{3N}{4} - 1, \\ -y_{n-\frac{3N}{4}}, & n = \frac{3N}{4}, \frac{3N}{4} + 1, \ldots, N-1. \end{cases} \tag{5.22}$$

The C source code of a fast recursive algorithm for the 2^n-length DCT-II computation and its inverse, DCT-III, can be found in [116]. To further reduce

the arithmetic complexity of sub-band synthesis windowing operation in the MP3 decoder, a recursive optimized algorithm is presented in [10, 11].

5.4 Efficient MDCT or MLT Implementations in MP3

Generally, for a given N the forward MDCT block transform given by (5.9) using the direct approach requires totally $\frac{N^2}{2}$ multiplications and $\frac{N}{2}(N-1)$ additions. The backward MDCT block transform given by (5.10) requires exactly $\frac{N}{2}$ less additions than that of the forward MDCT block transform. Thus, the forward/backward MDCT computation in MP3 using the direct approach requires for the short audio block 72 multiplications and 66/60 additions, while for the long audio block 648 multiplications and 630/612 additions are needed.

The description of each fast MDCT algorithm will include:

- General comments with reference to the bibliography.
- Complete formulae or sparse matrix factorizations for the efficient computation of both the forward and backward MDCT block transforms in MP3 audio coding standard. The detailed derivation of each fast MDCT algorithm can be found in the referred paper(s).
- The corresponding signal flow graphs are shown only for short audio blocks ($N = 12$). For the long audio blocks ($N = 36$), the signal flow graphs can be easily extrapolated from those of the short audio blocks, or they can be found in the referred paper(s). Note that the signal flow graphs define sparse matrix factorizations of the MDCT matrices.
- The total arithmetic complexity of the forward/backward MDCT computation as well as comments related to improving the arithmetic complexity and a possible structural simplification of the algorithm.

All necessary supporting efficient optimized modules (in the form of linear code) and tools for completing efficient MDCT implementations are provided in the Appendices. All the presented fast and efficient MDCT algorithms have been implemented and verified by computer programs in C and almost all can be also used for the 2^n-length data blocks.

5.4.1 DFT/FFT-Based Efficient MDCT Implementations

Traditionally, after introducing a new sinusoidal unitary transform in digital signal processing and in developing fast algorithms for its efficient computation, the DFT and its fast implementation, FFT, frequently has been used in the first place. Indeed, this is the case for MDCT. Several DFT/FFT-based algorithms for the fast MDCT

and MLT computation have been proposed [37, 42, 43, 46, 47, 52, 61] which at that time enabled the efficient implementation of the MDCT in MP3. They are discussed in the following subsections.

5.4.1.1 DFT/FFT-Based MDCT Implementation [37]

A computationally efficient DFT/FFT-based MDCT algorithm originally proposed for the realization of real-valued single sideband analysis/synthesis filter banks (with perfect reconstruction as well as with nearly or almost perfect reconstruction) has been reported in [37]. It is perhaps the first proposed fast algorithm which has enabled to efficiently implement the MDCT in MP3 standard. Due to the same basic symmetry property of the MDCT coefficients given by (5.13) and the odd-time odd-frequency DFT (O^2DFT) coefficients (see Appendix B.1), the fast forward and backward MDCT computation [37] is based on the fast O^2DFT algorithm [115] derived for odd/even symmetric real-valued data sequences (see Appendix B.2). Since it is valid for any N being an integer multiple of 4, consequently, it can be adopted for the efficient implementation of the MDCT in MP3 audio coding standard.

Complete formulae constituting the fast DFT/FFT-based algorithm for the efficient forward MDCT computation are given in [37] as

$$\begin{aligned} F_k = F_{2k} + i\, F_{\frac{N}{2}+2k} &= \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{4}-1} \left[(a_n - b_n) - i\,(a_n + b_n)\right]\, W_{4N}^{-(4n+1)(4k+1)} \\ &= \frac{\sqrt{2}}{2}\, W_{8N}^{-(8k+1)} \sum_{n=0}^{\frac{N}{4}-1} \left[\left[(a_n - b_n) - i\,(a_n + b_n)\right]\, W_{8N}^{-(8n+1)}\right] W_{\frac{N}{4}}^{-nk}, \\ k &= 0, 1, \ldots, \frac{N}{4} - 1, \end{aligned} \tag{5.23}$$

where $W_N^{-nk} = e^{-i\frac{2\pi nk}{N}}$ and $i = \sqrt{-1}$. Further, $F_{2k} = \Re e\,\{F_k\}$, $F_{\frac{N}{2}+2k} = \Im m\,\{F_k\}$, and data sequences $\{a_n\}$, $\{b_n\}$ are given by

$$a_n = x_{2n} - x_{\frac{N}{2}-1-2n}, \qquad b_n = x_{N-1-2n} + x_{\frac{N}{2}+2n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{5.24}$$

Finally, MDCT coefficients are obtained as

$$c_{2k} = (-1)^k\, \Re e\,\{F_k\}, \qquad c_{\frac{N}{2}-1-2k} = (-1)^{k+\frac{N}{4}+1}\, \Im m\,\{F_k\}, \quad k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{5.25}$$

The transform kernel $W_{4N}^{-(4n+1)(4k+1)}$ in (5.23) is uniformly split into three parts (whereby two of them are equal) as follows: $W_{4N}^{-(4n+1)(4k+1)} = W_{8N}^{-(8k+1)} W_{\frac{N}{4}}^{-nk} W_{8N}^{-(8n+1)}$, corresponding, respectively, to the block of $\frac{N}{4}$ Givens–Jacobi pre-rotations, an $\frac{N}{4}$-point forward complex-valued DFT (CDFT), and the identical block of $\frac{N}{4}$ Givens–Jacobi post-rotations (see Appendix F.1).

On the other hand, the fast DFT/FFT-based algorithm for the efficient backward MDCT computation is defined in [37] as

$$\begin{aligned} f_n = f_{2n} + i\, f_{\frac{N}{2}+2n} &= \frac{\sqrt{2}}{2} \sum_{k=0}^{\frac{N}{4}-1} \left[(-1)^k\, c_{2k} - (-1)^{k+\frac{N}{4}+1}\, i\, c_{\frac{N}{2}-1-2k} \right] W_{4N}^{-(4n+1)(4k+1)} \\ &= \frac{\sqrt{2}}{2}\, W_{8N}^{-(8n+1)} \sum_{k=0}^{\frac{N}{4}-1} \left[\left[(-1)^k\, c_{2k} \right.\right. \\ &\quad \left.\left. -(-1)^{k+\frac{N}{4}+1}\, i\, c_{\frac{N}{2}-1-2k} \right] W_{8N}^{-(8k+1)} \right] W_{\frac{N}{4}}^{-nk}, \\ n &= 0, 1, \ldots, \frac{N}{4} - 1, \end{aligned} \tag{5.26}$$

where $f_{2n} = \Re e\, \{f_n\}$ and $f_{\frac{N}{2}+2n} = \Im m\, \{f_n\}$. The time domain aliased data sequence $\{\hat{x}_n\}$ is recovered as

$$\begin{aligned} \hat{x}_{2n} &= \Re e\, \{f_n\} + \Im m\, \{f_n\}, \qquad \hat{x}_{\frac{N}{2}-1-2n} = -\hat{x}_{2n}, \\ \hat{x}_{\frac{N}{2}+2n} &= -\Re e\, \{f_n\} + \Im m\, \{f_n\}, \quad \hat{x}_{N-1-2n} = \hat{x}_{\frac{N}{2}+2n}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{5.27}$$

Note 1: If $\frac{N}{4}$ is odd (hence for the MDCT implementation in MP3), the factor $(-1)^{k+\frac{N}{4}+1}$ in (5.25) and (5.26) can be reduced to $(-1)^k$.

For the forward/backward MDCT computation in MP3 we need 3/9-point forward complex-valued DFT (CDFT) modules. The optimized efficient 3/9-point forward CDFT modules are presented in Appendix D.2. The regular signal flow graph for the computation of forward MDCT for $N = 12$ is shown in Fig. 5.1. Full lines represent transfer factors $+1$ while dashed lines represent transfer factors -1. Symbol $\circ$ represents addition. The common computing module identical both for the forward and backward MDCT is highlighted by shaded rectangle. Scaling factors $\frac{\sqrt{2}}{2}$ in (5.23) and (5.26) can be absorbed into Givens–Jacobi pre-rotation stage of the common computing module. Then, for the efficient implementation of Givens–Jacobi pre-rotations the bilinear computational structure has to be used (see Appendix F.3). The signal flow graph for the backward MDCT computation can be easily obtained taking into account (5.26) and (5.27).

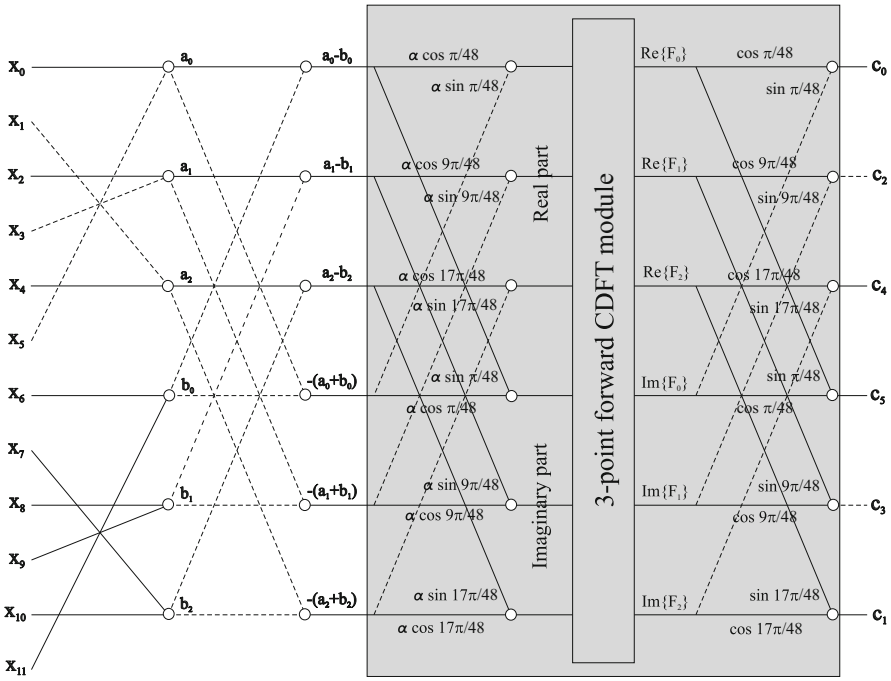

Fig. 5.1 Signal flow graph for the DFT/FFT-based forward MDCT computation for $N = 12$. $\alpha = \frac{\sqrt{2}}{2}$

In general, the total computational complexity of the forward and backward MDCT is $3\frac{N}{2}$ real multiplications and $5\frac{N}{2}$ real additions ($2N$ real additions for the backward MDCT) plus the arithmetic complexity of $\frac{N}{4}$-point forward CDFT. Thus, for the short block the forward/backward MDCT computation in MP3 requires 20 real multiplications, 42/36 real additions, and 2 shifts, while for the long block the forward/backward MDCT computation requires 70 real multiplications, 174/156 real additions, and 4 shifts.

Note 2: Alternatively, the transform kernel $W_{4N}^{-(4n+1)(4k+1)}$ in (5.23) and (5.26) can be split nonuniformly as follows: $W_{4N}^{-(4n+1)(4k+1)} = W_N^{-k} W_{\frac{N}{4}}^{-nk} W_{4N}^{-(4n+1)} = W_N^{-n} W_{\frac{N}{4}}^{-nk} W_{4N}^{-(4k+1)}$ (see next subsection). Since the terms W_N^{-k} and W_N^{-n} for $k,\ n = 0$ involve trivial rotations we can save three multiplications and three additions for the short and long blocks.

5.4.1.2 DFT/FFT-Based MDCT Implementation [42]

Another fast DFT/FFT-based MDCT algorithm has been reported in [42]. It is very similar to that described in previous subsection although it was derived by a quite

different procedure. Since the fast MDCT algorithm is also based on an $\frac{N}{4}$-point complex-valued DFT, it can be adopted for the efficient MDCT implementation in MP3 standard.

Complete formulae constituting the fast DFT/FFT-based algorithm for the efficient forward MDCT computation are given in [42] as

$$\begin{aligned} F_k = F_{2k} + i\, F_{\frac{N}{2}+2k} &= \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{4}-1} [(a_n - b_n) + i\,(a_n + b_n)]\; W_{4N}^{(4n+1)(4k+1)} \\ &= \frac{\sqrt{2}}{2}\, W_N^k \sum_{n=0}^{\frac{N}{4}-1} \left[[(a_n - b_n) + i\,(a_n + b_n)]\, W_{4N}^{4n+1} \right] W_{\frac{N}{4}}^{nk}, \\ k &= 0, 1, \ldots, \frac{N}{4} - 1, \end{aligned} \tag{5.28}$$

where $W_N^{nk} = e^{i\frac{2\pi nk}{N}}$ and $i = \sqrt{-1}$, $F_{2k} = \Re e\ \{F_k\}$, $F_{\frac{N}{2}+2k} = \Im m\ \{F_k\}$, and data sequences $\{a_n\}$, $\{b_n\}$ are given in (5.24). Finally, MDCT coefficients are obtained as

$$c_{2k} = (-1)^k\ \Re e\ \{F_k\}, \qquad c_{\frac{N}{2}-1-2k} = (-1)^{k+\frac{N}{4}}\ \Im m\ \{F_k\}, \qquad k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{5.29}$$

The transform kernel $W_{4N}^{(4n+1)(4k+1)}$ in (5.28) is nonuniformly split into three parts as follows: $W_{4N}^{(4n+1)(4k+1)} = W_N^k W_{\frac{N}{4}}^{nk} W_{4N}^{4n+1} = W_N^n W_{\frac{N}{4}}^{nk} W_{4N}^{4k+1}$ corresponding, respectively, to the block $\frac{N}{4} - 1$ Givens–Jacobi pre-rotations, an $\frac{N}{4}$-point inverse CDFT, and the block of $\frac{N}{4}$ Givens–Jacobi post-rotations (see Appendix F.1).

The fast DFT/FFT-based algorithm for the efficient backward MDCT computation is defined as [42]

$$\begin{aligned} f_n = f_{2n} + i\, f_{\frac{N}{2}+2n} &= \frac{\sqrt{2}}{2} \sum_{k=0}^{\frac{N}{4}-1} \left[(-1)^{k+\frac{N}{4}}\, c_{\frac{N}{2}-1-2k} + i\,(-1)^k\, c_{2k} \right] W_{4N}^{(4n+1)(4k+1)} \\ &= \frac{\sqrt{2}}{2}\, W_N^n \sum_{k=0}^{\frac{N}{4}-1} \left[\left[(-1)^{k+\frac{N}{4}}\, c_{\frac{N}{2}-1-2k} + i\,(-1)^k\, c_{2k} \right] W_{4N}^{4k+1} \right] W_{\frac{N}{4}}^{nk}, \\ n &= 0, 1, \ldots, \frac{N}{4} - 1, \end{aligned} \tag{5.30}$$

where $f_{2n} = \Re e\,\{f_n\}$ and $f_{\frac{N}{2}+2n} = \Im m\,\{f_n\}$. The time domain aliased data sequence $\{\hat{x}_n\}$ is recovered as

$$\begin{aligned} \hat{x}_{2n} &= \Re e\,\{f_n\} + \Im m\,\{f_n\}, \qquad \hat{x}_{\frac{N}{2}-1-2n} = -\hat{x}_{2n}, \\ \hat{x}_{\frac{N}{2}+2n} &= \Re e\,\{f_n\} - \Im m\,\{f_n\}, \qquad \hat{x}_{N-1-2n} = \hat{x}_{\frac{N}{2}+2n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{5.31}$$

Note 3: If $\frac{N}{4}$ is even, the factor $(-1)^{k+\frac{N}{4}}$ in (5.29) and (5.30) can be reduced to $(-1)^k$.

Compared to the DFT/FFT-based MDCT algorithm in MP3 described in the previous subsection, now we need 3/9-point inverse CDFT modules instead of the forward CDFT modules. The optimized efficient 3/9-point inverse CDFT modules can be easily obtained from the forward CDFT modules presented in Appendix D.2. The regular signal flow graph for the computation of forward MDCT for $N = 12$ is shown in Fig. 5.2. Again, the common computing module identical both for the forward and backward MDCT is highlighted by shaded rectangle. Scaling factors $\frac{\sqrt{2}}{2}$ in (5.28) and (5.30) can be absorbed into Givens–Jacobi pre-rotation stage of the common computing module. Then, for the efficient implementation of Givens–Jacobi pre-rotations the bilinear computational structure has to be used (see Appendix F.3). Note that the Givens–Jacobi post-rotations in the common computing module for $N = 12$ involve cosines and sines of rotation angles $\frac{\pi}{6}$ and $\frac{\pi}{3}$, where $\cos\frac{\pi}{6} = \sin\frac{\pi}{3} = \frac{\sqrt{3}}{2}$ and $\cos\frac{\pi}{3} = \sin\frac{\pi}{6} = \frac{1}{2}$. The signal flow graph for the backward MDCT computation can be easily obtained taking into account (5.30) and (5.31).

Note 4: It is important to note that the DFT/FFT-based fast MDCT algorithm [42] actually requires exactly about $\frac{N}{2} + 1$ real additions more than it has been reported in the original paper [42]. To obtain the correct result, an additional butterfly stage is necessary before the $\frac{N}{4}$-point inverse CDFT is applied (see the second butterfly stage in Fig. 5.2).

The total computational complexity of the forward and backward MDCT is $3(\frac{N}{2} - 1)$ real multiplications and $5\frac{N}{2} - 3$ real additions ($2N - 3$ real additions for the backward MDCT) plus the arithmetic complexity of $\frac{N}{4}$-point inverse CDFT. Thus, for the short block the forward MDCT computation in MP3 requires 17 real multiplications, 39/33 real additions, and 2 shifts, while for the long block the forward MDCT computation requires 67 real multiplications, 171/153 real additions, and 4 shifts.

Note 5: The windowing operation with the sine windowing function can be written in the form of Givens–Jacobi rotations applied to the input data sequence $\{x_n\}$, and it can be incorporated into the fast computational structure of the algorithm [42, 61].

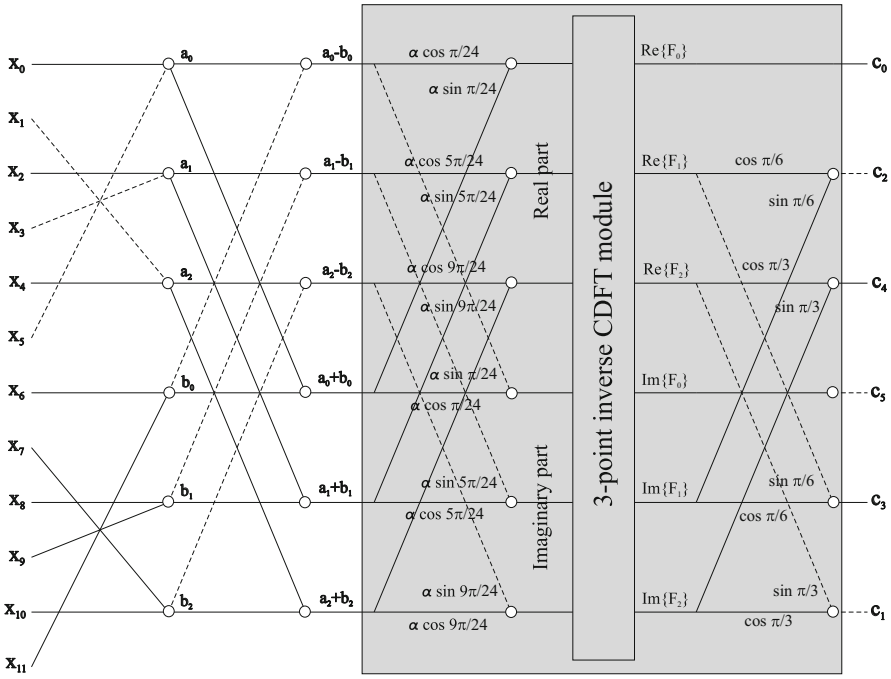

Fig. 5.2 Signal flow graph for the DFT/FFT-based forward MDCT computation for $N = 12$. $\alpha = \frac{\sqrt{2}}{2}$

5.4.1.3 DFT/FFT-Based MDCT Implementation [43, 52]

The main idea of a fast and efficient DFT/FFT-based MDCT algorithm outlined in [52] is based on the simple fact that the forward MDCT block transform given by (5.9) can be written in the following equivalent form:

$$c_k = \Re e \left\{ \sum_{n=0}^{N-1} x_{n-\frac{N}{4}} \; W_{4N}^{-(2n+1)(2k+1)} \right\} = \sum_{n=0}^{N-1} x_{n-\frac{N}{4}} \; \cos\left[\frac{\pi}{2N}(2n+1)(2k+1)\right],$$

$$k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{5.32}$$

where the transform kernels on the right-hand side of (5.32) are recognized, respectively, as the O^2DFT (see Appendix B.1) and the so-called real-valued polyphase filter bank derived from the DCT-IV transform kernel [43]. The term $x_{n-\frac{N}{4}}$ can be interpreted as a shifting of the original data sequence by $\frac{N}{4}$ samples with respect to the cosine transform kernel. In fact, exploiting the anti-periodicity

property of the cosine transform kernel with the period N, i.e., substituting $n+N$ for n we obtain: $\cos\left[\frac{\pi}{2N}(2n+1+2N)(2k+1)\right] = -\cos\left[\frac{\pi}{2N}(2n+1)(2k+1)\right]$, and it can be shown that the original data sequence has to be circularly shifted to the right in the period N by $\frac{N}{4}$ samples followed by sign changes of $\frac{N}{4}$ circularly shifted samples. This operation actually corresponds to a permutation applied to the original data sequence defined as [34, 53]

$$y_n = \begin{cases} -x_{\frac{3N}{4}+n}, & n = 0, 1, \ldots, \frac{N}{4}-1, \\ \\ x_{n-\frac{N}{4}}, & n = \frac{N}{4}, \frac{N}{4}+1, \ldots, N-1. \end{cases} \tag{5.33}$$

Then, applying the permutation described in (5.33) and using the symmetry property of the cosine transform kernel, (5.32) can be rewritten as

$$\begin{aligned} c_k &= \sum_{n=0}^{N-1} y_n \cos\left[\frac{\pi}{2N}(2n+1)(2k+1)\right] \\ &= \sum_{n=0}^{\frac{N}{2}-1} (y_n - y_{N-1-n}) \cos\left[\frac{\pi}{4(N/2)}(2n+1)(2k+1)\right], \\ k &= 0, 1, \ldots, \frac{N}{2}-1. \end{aligned} \tag{5.34}$$

The transform kernel in (5.34) is recognized as an $\frac{N}{2}$-point DCT-IV. Finally, combining (5.33) and (5.34) we get

$$c_k = \sum_{n=0}^{\frac{N}{2}-1} y_n \cos\left[\frac{\pi}{4(N/2)}(2n+1)(2k+1)\right], \qquad k = 0, 1, \ldots, \frac{N}{2}-1, \tag{5.35}$$

where

$$y_n = \begin{cases} -x_{\frac{3N}{4}+n} - x_{\frac{3N}{4}-1-n}, & n = 0, 1, \ldots, \frac{N}{4}-1, \\ \\ x_{n-\frac{N}{4}} - x_{\frac{3N}{4}-1-n}, & n = \frac{N}{4}, \frac{N}{4}+1, \ldots, \frac{N}{2}-1. \end{cases} \tag{5.36}$$

Since the $\frac{N}{2}$-point DCT-IV is closely related to the real part of N-point O^2DFT for real-valued odd symmetric data sequences, in order to efficiently compute the $\frac{N}{2}$-point DCT-IV of $\{y_n\}$, the fast O^2DFT algorithm is applied [6, 43, 138].

Complete formulae constituting the fast DFT/FFT-based algorithm for the efficient forward MDCT computation are given by [43]

$$F_k = F_{2k} + i\, F_{\frac{N}{2}+2k} = \sum_{n=0}^{\frac{N}{4}-1} \left(y_{2n} + i\, y_{\frac{N}{2}-1-2n}\right)\, W_{4N}^{-(4n+1)(4k+1)}$$

$$= W_N^{-k} \sum_{n=0}^{\frac{N}{4}-1} \left[\left(y_{2n} + i\, y_{\frac{N}{2}-1-2n}\right)\, W_{4N}^{-(4n+1)}\right] W_{\frac{N}{4}}^{-nk},$$

$$k = 0, 1, \ldots, \frac{N}{4} - 1, \tag{5.37}$$

where $W_N^{-nk} = e^{-i\frac{2\pi nk}{N}}$ and $i = \sqrt{-1}$, $F_{2k} = \Re e\ \{F_k\}$, $F_{\frac{N}{2}+2k} = \Im m\ \{F_k\}$. The data sequence $\{y_n\}$ is given by (5.36) or (5.47). Finally, MDCT coefficients are obtained as

$$c_{2k} = \Re e\ \{F_k\}, \qquad c_{\frac{N}{2}-1-2k} = -\Im m\ \{F_k\}, \qquad k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{5.38}$$

The transform kernel $W_{4N}^{-(4n+1)(4k+1)}$ in (5.28) is nonuniformly split into three parts as follows: $W_{4N}^{-(4n+1)(4k+1)} = W_N^{-k} W_{\frac{N}{4}}^{-nk} W_{4N}^{-(4n+1)} = W_N^{-n} W_{\frac{N}{4}}^{-nk} W_{4N}^{-(4k+1)}$ corresponding, respectively, to the block $\frac{N}{4} - 1$ Givens–Jacobi pre-rotations, an $\frac{N}{4}$-point forward CDFT, and the block of $\frac{N}{4}$ Givens–Jacobi post-rotations (see Appendix F.1).

The fast DFT/FFT-based algorithm for the efficient backward MDCT computation is defined as [43]

$$f_n = f_{2n} + i\, f_{\frac{N}{2}+2n} = \sum_{k=0}^{\frac{N}{4}-1} (c_{2k} + i\, c_{\frac{N}{2}-1-2k})\, W_{4N}^{-(4n+1)(4k+1)}$$

$$= W_N^{-n} \sum_{k=0}^{\frac{N}{4}-1} \left[(c_{2k} + i\, c_{\frac{N}{2}-1-2k})\, W_{4N}^{-(4k+1)}\right]\, W_{\frac{N}{4}}^{-nk},$$

$$n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{5.39}$$

where $f_{2n} = \Re e\ \{f_n\}$ and $f_{\frac{N}{2}+2n} = \Im m\ \{f_n\}$. The time domain aliased data sequence $\{\hat{x}_n\}$ is partially recovered as

$$\hat{x}_{\frac{N}{4}+2n} = \Im m\ \{f_n\}, \qquad \hat{x}_{\frac{3N}{4}-1-2n} = -\Re e\ \{f_n\}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{5.40}$$

and remaining samples of $\{\hat{x}_n\}$ can be easily deduced from the symmetry property given in (5.14).

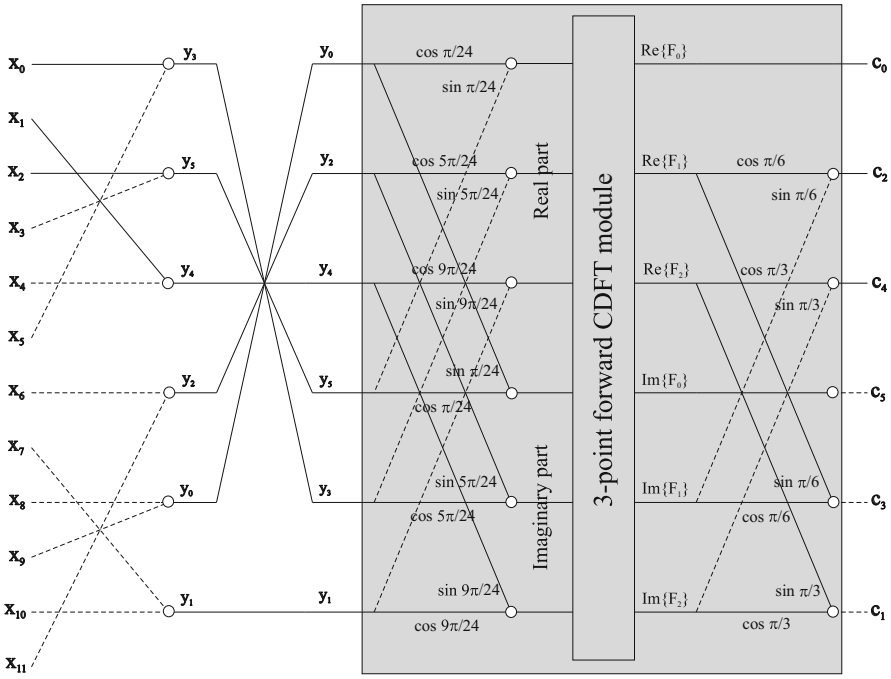

Fig. 5.3 Signal flow graph for the DFT/FFT-based forward MDCT computation for $N = 12$

For the forward/backward MDCT computation in MP3 we need 3/9-point forward CDFT modules. The optimized efficient 3/9-point forward CDFT modules are presented in Appendix D.2. The regular signal flow graph for the computation of forward MDCT for $N = 12$ is shown in Fig. 5.3. Again, the common computing module identical both for the forward and backward MDCT is highlighted by shaded rectangle. Compared to DFT/FFT-based MDCT algorithms [37, 42] (see two previous subsections), scaling factors $\frac{\sqrt{2}}{2}$ and required sign changes have been completely eliminated thus simplifying the algorithm structure. Efficient implementations of Givens–Jacobi rotations are presented in Appendices F.2 and F.3. Note that the Givens–Jacobi post-rotations in the common computing module for $N = 12$ involve cosines and sines of rotation angles $\frac{\pi}{6}$ and $\frac{\pi}{3}$, where $\cos\frac{\pi}{6} = \sin\frac{\pi}{3} = \frac{\sqrt{3}}{2}$ and $\cos\frac{\pi}{3} = \sin\frac{\pi}{6} = \frac{1}{2}$. The signal flow graph for the backward MDCT computation can be easily obtained taking into account (5.39) and (5.40).

The total computational complexity of the forward and backward MDCT is $3(\frac{N}{2} - 1)$ real multiplications and $2N - 3$ real additions, ($3(\frac{N}{2} - 1)$ real additions for the backward MDCT) plus the arithmetic complexity of $\frac{N}{4}$-point forward CDFT. Thus, for the short block the forward/backward MDCT computation in MP3 requires 17 real multiplications, 33/27 real additions, and 2 shifts, while for the long block the forward/backward MDCT computation requires 67 real multiplications, 153/135 real additions, and 4 shifts.

Note 6: An identical fast DFT/FFT-based algorithm for the efficient MLT computation has been presented in [46, 47]. Using the permutation given by (5.47), the N-point MLT is first converted to the $\frac{N}{2}$-point DCT-IV which is directly mapped into the $\frac{N}{4}$-point forward complex-valued DFT [6, 43].

5.4.2 DCT-II/DST-II-Based Efficient MDCT Implementations

The first published efficient MDCT implementation in MP3 [32] based on the fast MDCT algorithm [33] has been sequentially refined [24], improved [50, 56], and optimized [36, 48] both in terms of the arithmetic complexity and structural simplicity. The fast MDCT algorithm [33] uses the DCT-II and the corresponding DST-II of reduced sizes and it is valid for any N divisible by 4. Consequently, this fact enabled the efficient implementation of the forward/backward MDCT in MP3 audio coding standard.

In the following subsections the refined DCT-II/DST-II-based version [24] of the fast MDCT algorithm [32, 33] and its improved version [56] are discussed.

5.4.2.1 DCT-II/DST-II-Based MDCT Implementation [24, 32, 33]

Complete formulae constituting the fast DCT-II/DST-II-based algorithm for the forward/backward MDCT computation are given by [33]

$$z_{2k} = (-1)^k \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{4}-1} \left(a_n \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right] - b_n \sin\left[\frac{\pi(2n+1)k}{2(N/4)}\right]\right),$$

$$z_{2k+\frac{N}{2}} = (-1)^{k+\frac{N}{4}} \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{4}-1} (-1)^{n+1} \left(a_n \sin\left[\frac{\pi(2n+1)k}{2(N/4)}\right] + b_n \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right]\right),$$

$$k = 0, 1, \ldots, \frac{N}{4} - 1, \tag{5.41}$$

where

$$a_n = \left(x_n^{'} - x_{\frac{N}{2}-1-n}^{''}\right) \cos\frac{\pi(2n+1)}{2N} - \left(x_n^{''} - x_{\frac{N}{2}-1-n}^{'}\right) \sin\frac{\pi(2n+1)}{2N},$$

$$b_n = \left(x_n^{'} - x_{\frac{N}{2}-1-n}^{''}\right) \sin\frac{\pi(2n+1)}{2N} + \left(x_n^{''} - x_{\frac{N}{2}-1-n}^{'}\right) \cos\frac{\pi(2n+1)}{2N},$$

$$n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{5.42}$$

and

$$x_n' = x_n - x_{N-1-n}, \qquad x_n'' = x_n + x_{N-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \quad (5.43)$$

Final MDCT coefficients are obtained as

$$c_{2k} = z_{2k}, \qquad c_{2k+1} = -z_{N-2-2k}, \qquad k = 0, 1, \ldots, \frac{N}{4} - 1. \quad (5.44)$$

One can see from (5.41) to (5.44) that the N-point MDCT is decomposed into two pre-butterfly stages, the block of $\frac{N}{4}$ Givens–Jacobi rotations, an unnormalized $\frac{N}{4}$-point DCT-II and a corresponding unnormalized $\frac{N}{4}$-point DST-II, and the butterfly stage followed by proper sign changes. For $k = 0$, the coefficients z_0 and $z_{\frac{N}{2}}$ are sums of $\{a_n\}$ and $\{b_n\}$, respectively. The refined regular signal flow graph for the efficient forward/backward MDCT computation in MP3 for $N = 12$ is shown in Fig. 5.4. The backward MDCT computation can be simply realized by reversing the signal flow graph for the forward MDCT computation and performing inverse operations. The optimized efficient 3/9-point DCT-II and DCT-III modules are presented in Appendix D.4. The efficient MDCT implementation has been further refined as follows [24]. Since there exists a simple relation between the DCT-II and the DST-II [119], the $\frac{N}{4}$-point DST-II can be replaced by the $\frac{N}{4}$-point DCT-II with proper preceding sign changes, thus two identical $\frac{N}{4}$-point DCT-II modules are used in the resulting fast computational structure. Scaling factors $\frac{\sqrt{2}}{2}$ in expressions of (5.41) can be absorbed into Givens–Jacobi rotations. Then, for the efficient implementation of Givens–Jacobi rotations the bilinear computational structure has to be used (see Appendix F.3).

The total computational complexity of the forward MDCT is $3\frac{N}{4}$ multiplications and $11\frac{N}{4} - 2$ additions ($9\frac{N}{4} - 2$ additions for the backward MDCT) plus the arithmetic complexity of two identical $\frac{N}{4}$-point DCT-II/DCT-III modules. Thus, for the short block the forward/backward MDCT computation in MP3 requires 11 multiplications, 39/33 additions, and 2 shifts, while for the long block the forward/backward MDCT computation requires 43 multiplications, 165/147 additions, and 4 shifts.

5.4.2.2 Improved DCT-II/DST-II-Based MDCT Implementation [56]

Owing to the compatibility with the fast DCT-II/DST-II-based MDCT algorithm described in the previous subsection, the original form of improved fast DCT-II/DST-II-based MDCT algorithm [56] has been slightly modified. Complete formulae for the forward/backward MDCT computation are given by

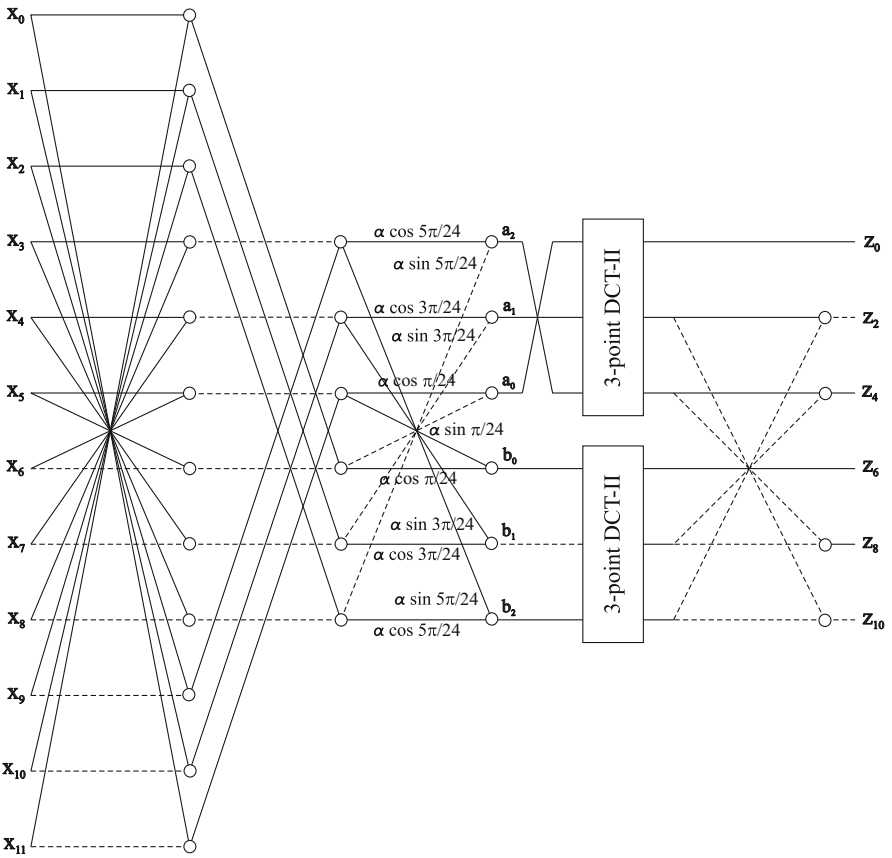

Fig. 5.4 Refined signal flow graph for the forward/backward MDCT computation for $N = 12$, $\alpha = \frac{\sqrt{2}}{2}$

$$z_{2k} = \sum_{n=0}^{\frac{N}{4}-1} \left(a_n \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right] + b_n \sin\left[\frac{\pi(2n+1)k}{2(N/4)}\right]\right),$$

$$z_{2k+\frac{N}{2}} = \sum_{n=0}^{\frac{N}{4}-1} (-1)^n \left(-a_n \sin\left[\frac{\pi(2n+1)k}{2(N/4)}\right] + b_n \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right]\right),$$

$$k = 0, 1, \ldots, \frac{N}{4} - 1, \tag{5.45}$$

where

$$a_n = y_n \cos\frac{\pi(2n+1)}{2N} + y_{\frac{N}{2}-n-1} \sin\frac{\pi(2n+1)}{2N},$$

$$b_n = -y_n \sin \frac{\pi(2n+1)}{2N} + y_{\frac{N}{2}-n-1} \cos \frac{\pi(2n+1)}{2N}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{5.46}$$

The data sequence $\{y_n\}$ is given by (5.36), and final MDCT coefficients are obtained from (5.44). The modified regular signal flow graph for the improved forward/backward MDCT computation in MP3 for $N = 12$ is shown in Fig. 5.5. The backward MDCT computation can be simply realized by reversing the signal flow graph for the forward MDCT computation and performing inverse operations. The optimized efficient 3/9-point DCT-II and DCT-III modules are presented in Appendix D.4. Compared to the fast MDCT algorithm given by (5.41)–(5.44), the improved fast MDCT algorithm given by (5.45) and (5.46) eliminates: the first butterfly stage thus saving N additions (see Fig. 5.4), further eliminates scaling factors $\frac{\sqrt{2}}{2}$ and final sign changes, thus simplifying the fast computational structure of the algorithm. On the other hand, comparing complete formulae of both fast MDCT algorithms one can note that they involve Givens–Jacobi rotations of opposite types. Efficient implementations of Givens–Jacobi rotations are presented in Appendices F.2 and F.3.

The total computational complexity of the forward MDCT is $3\frac{N}{4}$ multiplications and $7\frac{N}{4} - 2$ additions ($5\frac{N}{4} - 2$ additions for the backward MDCT) plus the arithmetic complexity of two identical $\frac{N}{4}$-point DCT-II/DCT-III modules. Thus, for the short block the forward/backward MDCT computation in MP3 requires 11 multiplications, 27/21 additions and 2 shifts, while for the long block the forward/backward MDCT computation requires 43 multiplications, 129/111 additions, and 4 shifts.

5.4.3 *DCT-IV-Based Efficient MDCT Implementations*

Another class of the fast MDCT algorithms developed up to now is based on the DCT-IV of half size [5, 6, 25, 27, 29, 31, 34, 36, 38, 40, 48, 53, 59, 63]. In fact, using a simple permutation applied to the input data sequence, the MDCT can always be converted to the DCT-IV of half size. Then, it is sufficient only to specify a suitable fast DCT-IV algorithm/computational structure which is valid for N being an even integer. It is widely accepted that the DCT-IV-based MDCT implementations are the most efficient both in terms of the arithmetic complexity and structural simplicity [30, 31]. Note that using the direct approach for computing the $\frac{N}{2}$-point DCT-IV immediately reduces the arithmetic complexity of the direct MDCT computation to $(\frac{N}{2})^2$ multiplications and $(\frac{N}{2})^2$ additions.

Principally, applying a permutation given by (5.36) to the input data sequence $\{x_n\}$ or a permutation defined as [6, 31]

$$\begin{aligned} y_{\frac{N}{4}+n} &= x_n - x_{\frac{N}{2}-1-n}, \\ y_{\frac{N}{4}-1-n} &= -x_{\frac{N}{2}+n} - x_{N-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1, \end{aligned} \tag{5.47}$$

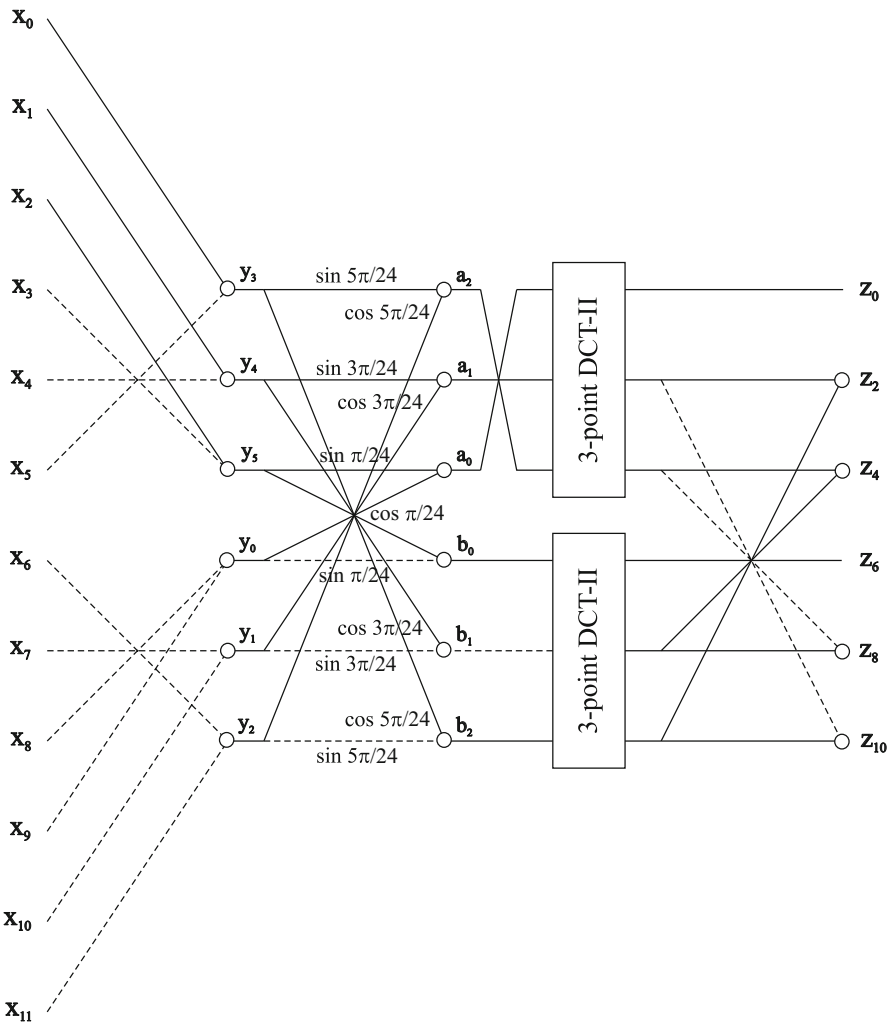

Fig. 5.5 Signal flow graph for the improved forward/backward MDCT computation for $N = 12$

the N-point forward MDCT or MLT is converted to an $\frac{N}{2}$-point DCT-IV of $\{y_n\}$. The backward MDCT or MLT is realized by the inverse $\frac{N}{2}$-point DCT-IV of $\{c_k\}$ using the same fast algorithm/computational structure, and the time domain aliased data sequence $\{\hat{x}_n\}$ can be recovered from $\{y_n\}$ applying an inverse permutation to that of (5.47) as [6, 31]

$$\begin{aligned} \hat{x}_n &= y_{\frac{N}{4}+n}, & \hat{x}_{\frac{N}{2}-1-n} &= -y_{\frac{N}{4}+n}, \\ \hat{x}_{\frac{N}{2}+n} &= -y_{\frac{N}{4}-1-n}, & \hat{x}_{N-1-n} &= -y_{\frac{N}{4}-1-n}, & n &= 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{5.48}$$

Note 7: Although the permutations (5.36) and (5.47) seem to be different, they generate exactly the same time domain aliased data sequence $\{y_n\}$ [31]. On the other hand, applying the permutation (5.33) to the input data sequence, the time domain aliased data sequence $\{\hat{x}_n\}$ can be recovered after the backward MDCT from $\{y_n\}$ as [34, 53]

$$\hat{x}_n = \begin{cases} y_{\frac{N}{4}+n}, & n = 0, 1, \ldots, \frac{3N}{4} - 1, \\ -y_{n-\frac{3N}{4}}, & n = \frac{3N}{4}, \frac{3N}{4} + 1, \ldots, N - 1. \end{cases} \tag{5.49}$$

Since the DCT-IV matrix is symmetric and self-inverse (the DCT-IV transform kernel is symmetric with respect to the time and frequency indices n and k), the forward/inverse DCT-IV computation, and hence as well as the forward/backward MDCT (or forward/backward MLT) is realized by an identical fast computational structure with the properly appended permutations (5.47) and (5.48). The $\frac{N}{2}$-point DCT-IV can be further decomposed either into two identical $\frac{N}{4}$-point DCTs-IV [114] or two identical $\frac{N}{4}$-point DCTs-II [25, 31, 136] with pre-butterfly/post-rotation or pre-rotation/post-butterfly stages. In order to efficiently implement the MDCT in MP3 we should specify a suitable fast DCT-IV algorithm/computational structure which is valid for N being an even integer, i.e., 6- and 18-point fast DCT-IV algorithms. In general, the even-length DCT-IV can be realized by:

- An indirect fast DCT-IV algorithm which maps the DCT-IV into a complex-valued DFT of half size [6, 43, 138]. Such an MDCT algorithm has been presented in Sect. 5.4.1.
- Recursive filter structures for the general-lengths DCT-IV [34, 76, 77, 80, 81, 84, 119] (see references on pp. 14–15).
- Combining radix-q fast DCT-IV algorithms (where q is an odd positive integer) [117, 130] with existing mixed-radix fast DCT-IV algorithms for the composite lengths $2^n \times q$, or with the even-length fast DCT-IV algorithms [114].
- Direct fast even-length recursive DCT-IV algorithms [121, 132].
- Fast even-length DCT-IV algorithms [25, 31] or based on orthogonal (recursive) sparse matrix factorizations of the DCT-IV matrix [119, 136].

Note 8: Recently, a new recursive algorithm for the 2^n-length DCT-IV computation has been presented [59] requiring fewer total real multiplications and real additions than algorithms published up to now. It is based on a new improved FFT algorithm being actually the modified split-radix FFT [131] with fewer arithmetic operations. Since the DCT-IV and MDCT are closely related, this improved indirect 2^n-length fast DCT-IV algorithm immediately implies an improved 2^n-length fast MDCT algorithm.

Note 9: Symmetry property of the DCT-IV matrix implies that by transposing its (orthogonal) sparse matrix factorization we can obtain the equivalent fast DCT-IV computational structure but in the reverse direction.

Note 10: The fast analysis and synthesis MDCT or MLT filter banks in MP3 based on the DCT-IV including the windowing&overlap procedure in the analysis MDCT (MLT) filter bank and windowing&overlap&add procedure in the synthesis MDCT (MLT) filter bank are presented in detail in Sect. 5.5.

Typical representative DCT-IV-based implementations adopted or directly tailored for the efficient MDCT computation in MP3 are discussed in the following subsections.

5.4.3.1 DCT-IV-Based MDCT Implementation (Simple Representative Version)

A simple representative efficient implementation of the MDCT in MP3 is obtained by combining the radix-3 and radix-9 fast DCT-IV algorithms [117, 130] with an even-length DCT-IV algorithm [114]. Let M be an even integer. Let $\{y_n\}$ be an input data sequence given by (5.36) or (5.47). A simple fast algorithm for the efficient M-point forward/inverse DCT-IV computation is defined as [114]

$$\begin{aligned} c_k^{IV} &= \cos\frac{\pi(2k+1)}{4M}\ a_k + \sin\frac{\pi(2k+1)}{4M}\ b_k, \\ c_{M-1-k}^{IV} &= -\sin\frac{\pi(2k+1)}{4M}\ a_k + \cos\frac{\pi(2k+1)}{4M}\ b_k, \qquad k = 0, 1, \ldots, \frac{M}{2} - 1, \end{aligned} \tag{5.50}$$

where

$$\begin{aligned} a_k &= \sum_{m=0}^{\frac{M}{2}-1} u_m\ \cos\left[\frac{\pi}{4(M/2)}(2m+1)(2k+1)\right], \\ b_{\frac{M}{2}-1-k} &= \sum_{m=0}^{\frac{M}{2}-1} (-1)^n\ v_m\ \cos\left[\frac{\pi}{4(M/2)}(2m+1)(2k+1)\right], \quad k=0, 1, \ldots, \frac{M}{2}-1. \end{aligned} \tag{5.51}$$

and

$$u_m = y_{2m} + y_{2m+1}, \qquad v_m = y_{2m} - y_{2m+1}, \qquad m = 0, 1, \ldots, \frac{M}{2} - 1. \tag{5.52}$$

Transposed version of this fast DCT-IV algorithm has been published in [81]. Taking $M = \frac{N}{2}$, (5.50), and (5.51) show that an $\frac{N}{2}$-point DCT-IV required for the efficient MDCT implementation in MP3 is decomposed into a butterfly stage, two identical unnormalized $\frac{N}{4}$-point DCTs-IV and the block of $\frac{N}{4}$ Givens–Jacobi rotations. For the efficient MDCT implementation in MP3 we need optimized efficient 3/9-point

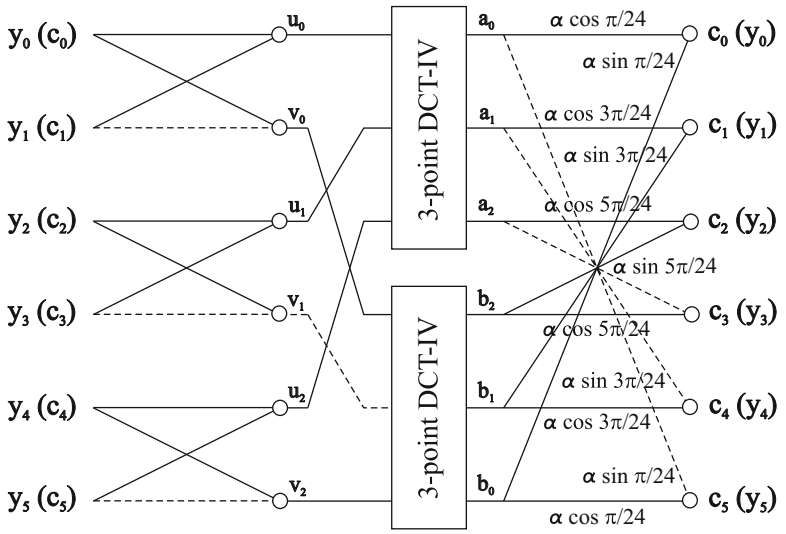

Fig. 5.6 Signal flow graph for the DCT-IV-based forward/backward MDCT computation for $N = 12, \alpha = \frac{\sqrt{2}}{2}$

scaled DCT-IV (SDCT-IV) modules. They are presented in Appendix D.6. The regular signal flow graph identical for the computation both of the forward and backward MDCT for $N = 12$ is shown in Fig. 5.6. Scaling factor $\frac{\sqrt{2}}{2}$ necessary for the 3/9-point DCT-IV can be absorbed into the block of $\frac{N}{4}$ Givens–Jacobi rotations. Then, for the efficient implementation of scaled Givens–Jacobi rotations the bilinear computational structure has to be used (see Appendix F.3). For the forward MDCT computation the data sequence $\{y_n\}$ is given by (5.36) or (5.47) while the time domain aliased data sequence $\{\hat{x}_n\}$ after the backward MDCT is recovered from $\{y_n\}$ according to (5.48) or (5.49).

The total computational complexity of the forward and backward MDCT is $3\frac{N}{4}$ multiplications and $7\frac{N}{4}$ additions ($5\frac{N}{4}$ additions for the backward MDCT) plus the arithmetic complexity of two $\frac{N}{4}$-point SDCT-IV modules. Thus, for the short block the forward/backward MDCT computation in MP3 requires 11 multiplications, 33/27 additions, and 2 shifts, while for the long block the forward/backward MDCT computation requires 61 multiplications, 169/151 additions, and 6 shifts.

5.4.3.2 DCT-IV-Based MDCT Implementation [25]

The fast DCT-IV/DST-IV algorithm [25] has been derived directly from the proposed fast MDCT algorithm [33] based on the following fact. The DCT-IV and DST-IV of double sizes are related to two variants of cosine-modulated filter banks (the so-called 1. and 2. short transforms) defined by the Dolby Digital (AC-3)

audio compression algorithm [1]. Since these two variants of cosine-modulated filter banks can be efficiently computed by the fast MDCT algorithm [33] (see also Chap. 6), the efficient DCT-IV/DST-IV computation can be realized via the MDCT of double size. The careful analysis of regular structure of the fast MDCT algorithm [33] (see also Sect. 5.4.2) allows to extract the even-length fast DCT-IV computational structure for the efficient implementation of the forward/backward MDCT in MP3.

Let $\boldsymbol{C}_M^{IV}$ be the orthogonal DCT-IV matrix of order M. Then, $\boldsymbol{C}_M^{IV}$ can be decomposed into the following sparse block matrix product [25]

$$\boldsymbol{C}_M^{IV} = \boldsymbol{Q}_M \begin{pmatrix} 1 & & & \\ & \boldsymbol{I}_{\frac{M}{2}-1} & & -\boldsymbol{J}_{\frac{M}{2}-1} \\ & & -1 & \\ & -\boldsymbol{J}_{\frac{M}{2}-1} & & -\boldsymbol{I}_{\frac{M}{2}-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{C}_{\frac{M}{2}}^{II}\boldsymbol{J}_{\frac{M}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_{\frac{M}{2}}^{II}\boldsymbol{D}_{\frac{M}{2}} \end{pmatrix} \boldsymbol{G}_M \begin{pmatrix} \boldsymbol{J}_{\frac{M}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{J}_{\frac{M}{2}} \end{pmatrix}, \tag{5.53}$$

where $\boldsymbol{I}_{\frac{M}{2}}$ is the identity matrix and $\boldsymbol{J}_{\frac{M}{2}}$ is reverse ordered identity matrix, both of order $\frac{M}{2}$, and $\boldsymbol{0}$'s are null matrices. The matrix product $\boldsymbol{C}_{\frac{M}{2}}^{II}\boldsymbol{J}_{\frac{M}{2}}$ denotes $\frac{M}{2}$-order DCT-II matrix with reversed order of its columns. The matrix product $\boldsymbol{C}_{\frac{M}{2}}^{II}\boldsymbol{D}_{\frac{M}{2}}$ denotes $\frac{M}{2}$-order DCT-II matrix with sign changed of its odd-indexed rows, whereby $\boldsymbol{D}_{\frac{M}{2}} = \text{diag}\,\{1, -1, 1, \ldots, (-1)^{\frac{M}{2}+1}\}$ is the diagonal matrix. $\boldsymbol{G}_M$ is the rotation matrix (Givens–Jacobi rotations) defined as

$$\boldsymbol{G}_M = \begin{pmatrix} \cos\frac{(M-1)\pi}{4M} & & 0 & 0 & & -\sin\frac{(M-1)\pi}{4M} \\ & \cos\frac{(M-3)\pi}{4M} & & & -\sin\frac{(M-3)\pi}{4M} & \\ & & \ddots & \ddots & & \\ & & \cos\frac{\pi}{4M} & -\sin\frac{\pi}{4M} & & \\ 0 & & & & & 0 \\ & & \sin\frac{\pi}{4M} & \cos\frac{\pi}{4M} & & \\ & & \ddots & \ddots & & \\ & \sin\frac{(M-3)\pi}{4M} & & & \cos\frac{(M-3)\pi}{4M} & \\ \sin\frac{(M-1)\pi}{4M} & & 0 & 0 & & \cos\frac{(M-1)\pi}{4M} \end{pmatrix}, \tag{5.54}$$

and $\boldsymbol{Q}_M$ is a permutation matrix corresponding to (5.44). The sparse matrix factorization of the DCT-IV matrix (5.53) is the direct consequence of a sparse matrix factorization derived for the MDCT matrix [33].

Taking $M = \frac{N}{2}$, from the factorization (5.53) it follows that an $\frac{N}{2}$-point DCT-IV is decomposed into the block of $\frac{N}{4}$ Givens–Jacobi rotations, two identical unnormalized $\frac{N}{4}$-point DCTs-II and the butterfly stage. For the efficient forward/backward MDCT implementation in MP3, the optimized efficient 3/9-point DCT-II modules are presented in Appendix D.4. The simple and regular signal flow graph identical for the computation both of the forward and backward MDCT for $N = 12$ is shown in Fig. 5.7. Efficient implementations of Givens–Jacobi rotations are presented in

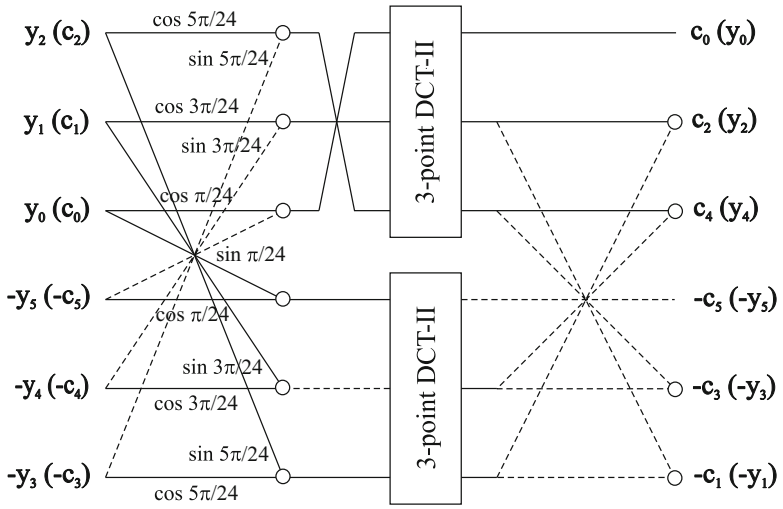

Fig. 5.7 Signal flow graph for the DCT-IV-based forward/backward MDCT computation for $N = 12$

Appendices F.2 and F.3. The input data sequence $\{y_n\}$ in the forward MDCT is given by (5.47) while the time domain aliased data sequence $\{\hat{x}_n\}$ is recovered after the backward MDCT from $\{y_n\}$ according to (5.48).

The total computational complexity of the forward MDCT is $3\frac{N}{4}$ multiplications and $7\frac{N}{4} - 2$ additions ($5\frac{N}{4} - 2$ additions for the backward MDCT) plus the arithmetic complexity of two $\frac{N}{4}$-point DCT-II modules. Thus, for the short block the forward/backward MDCT computation in MP3 requires 11 multiplications, 27/21 additions, and 2 shifts, while for the long block the forward/backward MDCT computation requires 43 multiplications, 129/111 additions, and 4 shifts.

5.4.3.3 DCT-IV-Based MDCT Implementation [31]

Complete formulae constituting the fast DCT-IV computational structure for the forward/backward MDCT computation are given by [31]

$$c_{2k} = \sum_{n=0}^{\frac{N}{4}-1} a_n \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right] + b_n \sin\left[\frac{\pi(2n+1)k}{2(N/4)}\right],$$

$$c_{2k-1} = \sum_{n=0}^{\frac{N}{4}-1} a_n \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right] - b_n \sin\left[\frac{\pi(2n+1)k}{2(N/4)}\right], \quad k = 1, 2, \ldots, \frac{N}{4} - 1, \tag{5.55}$$

where

$$a_n = y_n \cos\frac{\pi}{2N}(2n+1) + y_{\frac{N}{2}-1-n} \sin\frac{\pi}{2N}(2n+1),$$
$$b_n = -y_n \sin\frac{\pi}{2N}(2n+1) + y_{\frac{N}{2}-1-n} \cos\frac{\pi}{2N}(2n+1), \quad n = 0, 1, \ldots, \frac{N}{4}-1. \tag{5.56}$$

For $k = 0$ in the first sum and for $k = \frac{N}{4}$ in the second sum of (5.55), we, respectively, get

$$c_0 = \sum_{n=0}^{\frac{N}{4}-1} a_n, \qquad c_{\frac{N}{2}-1} = -\sum_{n=0}^{\frac{N}{4}-1} (-1)^n\, b_n. \tag{5.57}$$

It can be easily seen that the $\frac{N}{2}$-point DCT-IV is decomposed into the block of $\frac{N}{4}$ Givens–Jacobi rotations given by (5.56), an unnormalized $\frac{N}{4}$-point DCT-II and a corresponding unnormalized $\frac{N}{4}$-point DST-II. Denoting in (5.55) the $\frac{N}{4}$-point DCT-II of $\{a_n\}$ and the $\frac{N}{4}$-point DST-II of $\{b_n\}$, respectively, by

$$c_k^{II} = \sum_{n=0}^{\frac{N}{4}-1} a_n \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right], \qquad s_k^{II} = \sum_{n=0}^{\frac{N}{4}-1} b_n \sin\left[\frac{\pi(2n+1)k}{2(N/4)}\right],$$
$$k = 1, 2, \ldots, \frac{N}{4}-1,$$

and using a relation between the DCT-II and the DST-II [119], i.e., substituting $\frac{N}{4}-k$ for k into the above second sum, we get

$$s_{\frac{N}{4}-k}^{II} = \sum_{n=0}^{\frac{N}{4}-1} (-1)^n\, b_n \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right] = \sum_{n=0}^{\frac{N}{4}-1} b'_n \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right],$$
$$k = 1, 2, \ldots, \frac{N}{4}-1,$$

and the $\frac{N}{4}$-point DST-II of $\{b_n\}$ is converted to an $\frac{N}{4}$-point DCT-II of $\{b'_n\}$. Thus, (5.55) and (5.57) can be rewritten in a simplified equivalent form as

$$c_{2k} = c_k^{II} + s_k^{II}, \qquad c_0 = c_0^{II},$$
$$c_{2k-1} = c_k^{II} - s_k^{II}, \qquad c_{\frac{N}{2}-1} = -s_{\frac{N}{4}}^{II}, \qquad k = 1, 2, \ldots, \frac{N}{4}-1, \tag{5.58}$$

which corresponds to the butterfly stage.

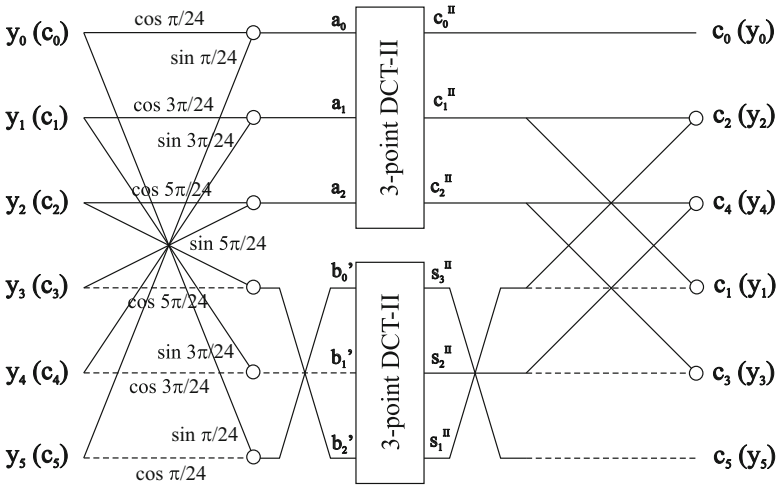

Fig. 5.8 Signal flow graph for the DCT-IV-based forward/backward MDCT computation for $N = 12$

The regular signal flow graph identical for the computation both of the forward and backward MDCT for $N = 12$ is shown in Fig. 5.8. For the efficient MDCT implementation in MP3 the optimized efficient 3/9-point DCT-II modules are presented in Appendix D.4. Efficient implementations of Givens–Jacobi rotations are presented in Appendices F.2 and F.3. The input data sequence $\{y_n\}$ in the forward MDCT is given by (5.47) while the time domain aliased data sequence $\{\hat{x}_n\}$ is recovered after the backward MDCT from $\{y_n\}$ according to (5.48).

The last butterfly stage in the signal flow graph shown in Fig. 5.8 can be further refined as follows. If we denote the output coefficients $\{c_0^{II}, c_1^{II}, \ldots, c_{\frac{N}{4}-1}^{II}, s_1^{II}, s_2^{II}, \ldots, s_{\frac{N}{4}}^{II}\}$ of the two unnormalized $\frac{N}{4}$-point DCTs-II by $\{z_k\}$, then the last butterfly stage can be reorganized and the final MDCT coefficients $\{c_k\}$ are obtained in natural order using the following mapping:

$$c_k = \begin{cases} z_{\frac{k}{2}} + z_{\frac{N}{4}-1+\frac{k}{2}}, & k = 2, 4, \ldots, \frac{N}{4} - 2, \quad \text{is even}, \quad c_0 = z_0, \\ z_{\frac{k+1}{2}} - z_{\frac{N}{4}-1+\frac{k+1}{2}}, & k = 1, 3, \ldots, \frac{N}{4} - 1, \quad \text{is odd}, \quad c_{\frac{N}{2}-1} = -z_{\frac{N}{2}-1}. \end{cases} \tag{5.59}$$

The corresponding signal flow graph with the reorganized last butterfly stage is shown in Fig. 5.9.

Note 11: Similar fast computational structures have been derived in [36, 48] with the aim to improve and optimize an efficient implementation of the MDCT in MP3 [25, 33].

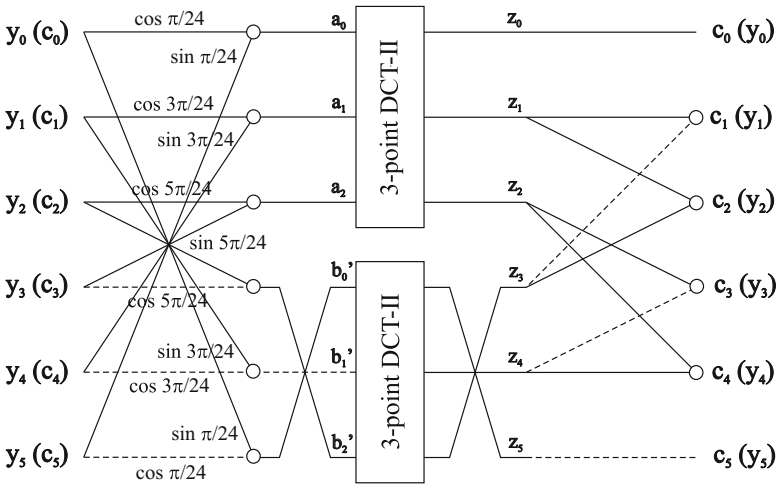

Fig. 5.9 Signal flow graph for the DCT-IV-based forward/backward MDCT computation for $N = 12$ with the reorganized last butterfly stage

The total computational complexity of the forward MDCT is $3\frac{N}{4}$ multiplications and $7\frac{N}{4} - 2$ additions ($5\frac{N}{4} - 2$ additions for the backward MDCT) plus the arithmetic complexity of two $\frac{N}{4}$-point DCT-II modules. Thus, for the short block the forward/backward MDCT computation in MP3 requires 11 multiplications, 27/21 additions, and 2 shifts, while for the long block the forward/backward MDCT computation requires 43 multiplications, 129/111 additions, and 4 shifts.

5.4.3.4 DCT-IV-Based MDCT Implementation [38, 40]

The most efficient MDCT implementations in MP3 in terms of the minimal multiplicative complexity known up to now [38, 40] are based on mixed-radix $3^m \times 2$, $m > 0$, fast DCT-IV algorithms developed specifically for 6-point and 18-point DCT-IV computations. Based on group theoretic partitioning of the DCT-IV transform kernel (or equivalently partitioning the DCT-IV matrix into sub-matrices), the unnormalized DCT-IV matrices of order 6 and 18 denoted, respectively, by $\boldsymbol{C}_6^{IV}$ and $\boldsymbol{C}_{18}^{IV}$, are decomposed into cyclic convolutions and Hankel matrix-vector products. Then, bilinear algorithms [134] are applied to each of the convolutions and 2-point Hankel matrix-vector products resulting in a single bilinear algorithm for the forward/backward MDCT computation used both for the short and long audio blocks.

Although the procedures in deriving the most efficient algorithms [38, 40] are based on rigorous mathematical theory of cyclic (Abelian) groups, cyclic

convolutions, and associated Hankel matrices [134, 140], we will try to explain the approach from the viewpoint of classical theory of matrices [128]. Since the approach in derivation of the most efficient algorithms is very sophisticated and elegant it merits to be presented in a detailed, but compact form as much as possible for the short and long audio blocks separately.

5.4.3.5 Case N = 12 (Short Data Block)

Consider the MDCT implementation for the short audio block represented in the matrix-vector form as

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{pmatrix} = \begin{pmatrix} \cos\frac{\pi}{24} & \cos\frac{3\pi}{24} & \cos\frac{5\pi}{24} & \cos\frac{7\pi}{24} & \cos\frac{9\pi}{24} & \cos\frac{11\pi}{24} \\ \cos\frac{3\pi}{24} & \cos\frac{9\pi}{24} & -\cos\frac{9\pi}{24} & -\cos\frac{3\pi}{24} & -\cos\frac{3\pi}{24} & -\cos\frac{9\pi}{24} \\ \cos\frac{5\pi}{24} & -\cos\frac{9\pi}{24} & -\cos\frac{\pi}{24} & -\cos\frac{11\pi}{24} & \cos\frac{3\pi}{24} & \cos\frac{7\pi}{24} \\ \cos\frac{7\pi}{24} & -\cos\frac{3\pi}{24} & -\cos\frac{11\pi}{24} & \cos\frac{\pi}{24} & -\cos\frac{9\pi}{24} & -\cos\frac{5\pi}{24} \\ \cos\frac{9\pi}{24} & -\cos\frac{3\pi}{24} & \cos\frac{3\pi}{24} & -\cos\frac{9\pi}{24} & -\cos\frac{9\pi}{24} & \cos\frac{3\pi}{24} \\ \cos\frac{11\pi}{24} & -\cos\frac{9\pi}{24} & \cos\frac{7\pi}{24} & -\cos\frac{5\pi}{24} & \cos\frac{3\pi}{24} & -\cos\frac{\pi}{24} \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}, \tag{5.60}$$

where the data sequence $\{y_n\}$ is given by (5.36) or (5.47), and $\{c_k\}$ are MDCT coefficients. Perhaps the most important fact, one can observe a special structure of some basis vectors of the symmetric matrix $\boldsymbol{C}_6^{IV}$ which does not appear in 2^n-order DCT-IV matrices [119]. Specifically, for the frequency/time indices $k/n = 0, 2, 3, 5$, the corresponding row/column basis vectors always consist of six different elements in magnitude, whereas for $k,\ n = 1, 4$, i.e., the first and fourth row/column basis vectors consist of only two different elements in magnitude. The proposed most efficient MDCT implementations [38, 40] just exploit this special structure of basis vectors of $\boldsymbol{C}_6^{IV}$ matrix.

At first, the time and frequency indices n and k, respectively, are partitioned into two sets. The set $S_1 = \{1, 4\}$ is made up of those values j for which $(2j + 1)$ is a multiple of 3, and the set $S_2 = \{0, 2, 3, 5\}$ otherwise. The computation of transform coefficients $\{c_k\}$ with respect to the time indices restricted to sets S_1 and S_2 is denoted by $c_k^{(1)}$ and $c_k^{(2)}$, respectively. Specifically, the DCT-IV matrix

$\boldsymbol{C}_6^{IV}$ is partitioned into four sub-matrices and the summation is carried out over the two index sets separately. Then, it is clear that the MDCT coefficients $\{c_k\}$ are obtained as

$$c_k = c_k^{(1)} + c_k^{(2)}, \qquad k = 0, 1, 2, 3, 4, 5. \tag{5.61}$$

Now, let us partition the matrix $\boldsymbol{C}_6^{IV}$ into four sub-matrices as follows. The transform components $\{c_k^{(1)}\}$, where $k \in S_1$ and $n \in S_1$ are given by

$$\begin{pmatrix} c_1^{(1)} \\ \\ c_4^{(1)} \end{pmatrix} = \begin{pmatrix} \cos\frac{9\pi}{24} - \cos\frac{3\pi}{24} \\ \\ -\cos\frac{3\pi}{24} - \cos\frac{9\pi}{24} \end{pmatrix} \begin{pmatrix} y_1 \\ \\ y_4 \end{pmatrix}, \quad \cos\frac{3\pi}{24} = \cos\frac{\pi}{8}, \quad \cos\frac{9\pi}{24} = \sin\frac{\pi}{8}. \tag{5.62}$$

Since the constant matrix in (5.62) is a Hankel matrix, this product can be obtained by using a corresponding bilinear algorithm for 2-point Hankel matrix-vector product (see Appendix A.2). Similarly, the transform components $\{c_k^{(2)}\}$, where $k \in S_1$ and $n \in S_2$ are given by

$$\begin{pmatrix} c_1^{(2)} \\ \\ c_4^{(2)} \end{pmatrix} = \begin{pmatrix} \cos\frac{3\pi}{24} & \cos\frac{9\pi}{24} \\ \\ \cos\frac{9\pi}{24} & -\cos\frac{3\pi}{24} \end{pmatrix} \begin{pmatrix} y_0 - y_3 \\ \\ -y_2 - y_5 \end{pmatrix}. \tag{5.63}$$

Equation (5.63) is based on the simple fact that some matrix elements are equal in magnitude (see the first and fourth rows of $\boldsymbol{C}_6^{IV}$). Again, since the constant matrix in (5.63) is a Hankel matrix this product can be obtained by using the corresponding bilinear algorithm for 2-point Hankel matrix-vector product (see Appendix A.2).

Further, the transform components $\{c_k^{(1)}\}$, where $k \in S_2$ and $n \in S_1$ are given by

$$\begin{pmatrix} c_0^{(1)} \\ -c_5^{(1)} \\ c_3^{(1)} \\ c_2^{(1)} \end{pmatrix} = \begin{pmatrix} \cos\frac{3\pi}{24} & \cos\frac{9\pi}{24} \\ -\cos\frac{9\pi}{24} & \cos\frac{3\pi}{24} \\ -\cos\frac{3\pi}{24} & -\cos\frac{9\pi}{24} \\ -\cos\frac{9\pi}{24} & \cos\frac{3\pi}{24} \end{pmatrix} \begin{pmatrix} y_1 \\ y_4 \end{pmatrix} = \begin{pmatrix} -c_4^{(1)} \\ -c_1^{(1)} \\ c_4^{(1)} \\ -c_1^{(1)} \end{pmatrix}. \tag{5.64}$$

Comparison of (5.64) with (5.62) shows that the transform components $c_0^{(1)}, -c_5^{(1)}, c_3^{(1)}$ and $c_2^{(1)}$ need not be computed separately.

Finally, the computation of transform components $\{c_k^{(2)}\}$ when $k, n \in S_2$, after proper row/column reordering and sign changes made in the remaining symmetric 4×4 sub-matrix and associated input/output data vectors (actually predicted by the structure of a cyclic group [38, 40]), it can be represented in the matrix-vector form as

$$\begin{pmatrix} c_0^{(2)} \\ -c_5^{(2)} \\ \hline c_3^{(2)} \\ c_2^{(2)} \end{pmatrix} = \left(\begin{array}{cc|cc} \cos\frac{\pi}{24} & -\cos\frac{11\pi}{24} & \cos\frac{7\pi}{24} & \cos\frac{5\pi}{24} \\ -\cos\frac{11\pi}{24} & -\cos\frac{\pi}{24} & \cos\frac{5\pi}{24} & -\cos\frac{7\pi}{24} \\ \hline \cos\frac{7\pi}{24} & \cos\frac{5\pi}{24} & \cos\frac{\pi}{24} & -\cos\frac{11\pi}{24} \\ \cos\frac{5\pi}{24} & -\cos\frac{7\pi}{24} & -\cos\frac{11\pi}{24} & -\cos\frac{\pi}{24} \end{array}\right) \begin{pmatrix} y_0 \\ -y_5 \\ \hline y_3 \\ y_2 \end{pmatrix}. \quad (5.65)$$

One can observe that the matrix in (5.65) is a block cyclic matrix with each block being a 2×2 Hankel matrix. In order to compute (5.65) the following procedure is applied. Denoting 2×2 Hankel matrices in (5.65) as

$$\boldsymbol{A}_2 = \begin{pmatrix} \cos\frac{\pi}{24} & -\cos\frac{11\pi}{24} \\ -\cos\frac{11\pi}{24} & -\cos\frac{\pi}{24} \end{pmatrix}, \qquad \boldsymbol{B}_2 = \begin{pmatrix} \cos\frac{7\pi}{24} & \cos\frac{5\pi}{24} \\ \cos\frac{5\pi}{24} & -\cos\frac{7\pi}{24} \end{pmatrix}, \quad (5.66)$$

and defining the two combined 2-point Hankel matrix-vector products we get the following important interrelations

$$\begin{aligned} \begin{pmatrix} e \\ f \end{pmatrix} &= \frac{1}{2}(\boldsymbol{A}_2 + \boldsymbol{B}_2) \begin{pmatrix} y_0 + y_3 \\ y_2 - y_5 \end{pmatrix} \\ &= \frac{1}{2} \left(\begin{array}{c|c} \cos\frac{\pi}{24} + \cos\frac{7\pi}{24} & \cos\frac{5\pi}{24} - \cos\frac{11\pi}{24} \\ \hline \cos\frac{5\pi}{24} - \cos\frac{11\pi}{24} & -\cos\frac{\pi}{24} - \cos\frac{7\pi}{24} \end{array}\right) \begin{pmatrix} y_0 + y_3 \\ y_2 - y_5 \end{pmatrix} \\ &= \frac{1}{2} \begin{pmatrix} c_0^{(2)} + c_3^{(2)} \\ -c_5^{(2)} + c_2^{(2)} \end{pmatrix}, \end{aligned}$$

$$\begin{pmatrix} g \\ \\ h \end{pmatrix} = \frac{1}{2}\,(\boldsymbol{A}_2 - \boldsymbol{B}_2) \begin{pmatrix} y_0 - y_3 \\ \\ -y_2 - y_5 \end{pmatrix}$$

$$= \frac{1}{2} \left(\begin{array}{c|c} \cos\frac{\pi}{24} - \cos\frac{7\pi}{24} & -\cos\frac{11\pi}{24} - \cos\frac{5\pi}{24} \\ \hline -\cos\frac{11\pi}{24} - \cos\frac{5\pi}{24} & -\cos\frac{\pi}{24} + \cos\frac{7\pi}{24} \end{array}\right) \begin{pmatrix} y_0 - y_3 \\ \\ -y_2 - y_5 \end{pmatrix}$$

$$= \frac{1}{2} \begin{pmatrix} c_0^{(2)} - c_3^{(2)} \\ \\ -c_5^{(2)} - c_2^{(2)} \end{pmatrix}. \tag{5.67}$$

Using trigonometric identities

$$\cos\frac{3\pi}{24} = \cos\frac{5\pi}{24} + \cos\frac{11\pi}{24}, \quad \cos\frac{9\pi}{24} = \cos\frac{\pi}{24} - \cos\frac{7\pi}{24},$$

$$\sqrt{3}\cos\frac{3\pi}{24} = \cos\frac{\pi}{24} + \cos\frac{7\pi}{24}, \quad \sqrt{3}\cos\frac{9\pi}{24} = \cos\frac{5\pi}{24} - \cos\frac{11\pi}{24},$$

the two expressions in (5.67) can be derived in the form of final 2-point Hankel matrix-vector products as

$$\begin{pmatrix} e \\ \\ f \end{pmatrix} = \frac{\sqrt{3}}{2} \begin{pmatrix} \cos\frac{3\pi}{24} & \cos\frac{9\pi}{24} \\ \\ \cos\frac{9\pi}{24} & -\cos\frac{3\pi}{24} \end{pmatrix} \begin{pmatrix} y_0 + y_3 \\ \\ y_2 - y_5 \end{pmatrix},$$

$$\begin{pmatrix} g \\ \\ h \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \cos\frac{9\pi}{24} - \cos\frac{3\pi}{24} \\ \\ -\cos\frac{3\pi}{24} - \cos\frac{9\pi}{24} \end{pmatrix} \begin{pmatrix} y_0 - y_3 \\ \\ -y_2 - y_5 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} c_4^{(2)} \\ \\ -c_1^{(2)} \end{pmatrix}. \tag{5.68}$$

Importantly, the second expression in (5.68) implies that the computation of (5.63) can be absorbed into that of (5.65). The constant matrices in (5.68) are Hankel matrices, and therefore, these products can be obtained by using the corresponding bilinear algorithm for 2-point Hankel matrix-vector product (see Appendix A.2). From (5.67) and (5.68) it follows that the complete set of transform components $\{c_k^{(2)}\}$ is given by

$$c_0^{(2)} = e + g,$$

$$
\begin{aligned}
-c_5^{(2)} &= f + h, \\
c_3^{(2)} &= e - g, \\
c_2^{(2)} &= f - h, \\
c_1^{(2)} &= -2h, \\
c_4^{(2)} &= \quad 2g.
\end{aligned} \tag{5.69}
$$

Now we are ready to derive the final MDCT coefficients $\{c_k\}$. According to (5.61) we get

$$
\begin{aligned}
c_0 &= c_0^{(1)} + c_0^{(2)} = -c_4^{(1)} + e + g = \quad e + (g - c_4^{(1)}), \\
c_1 &= c_1^{(1)} + c_1^{(2)} = \quad c_1^{(1)} - 2h, \\
c_2 &= c_2^{(1)} + c_2^{(2)} = -c_1^{(1)} + f - h = \quad f - (c_1^{(1)} + h), \\
c_3 &= c_3^{(1)} + c_3^{(2)} = \quad c_4^{(1)} + e - g = \quad e - (g - c_4^{(1)}), \\
c_4 &= c_4^{(1)} + c_4^{(2)} = \quad c_4^{(1)} + 2g, \\
c_5 &= c_5^{(1)} + c_5^{(2)} = -c_1^{(1)} - f - h = -f - (c_1^{(1)} + h).
\end{aligned} \tag{5.70}
$$

Revealing all the redundancies we need to evaluate only (5.62), (5.68), and (5.69) to compute 6-point DCT-IV. The regular signal flow graph for the 6-point DCT-IV computation, and hence, the identical fast computational structure both for the forward and backward MDCT for the short block is shown in Fig. 5.10. 2-point Hankel matrix-vector products given by (5.62) and (5.68) in the signal flow graph are alternatively converted to (scaled) Givens–Jacobi rotations which can be realized by a corresponding bilinear computational structure (see Appendix F.3). The time domain aliased data sequence $\{\hat{x}_n\}$ after the backward MDCT is recovered from $\{y_n\}$ according to (5.48) or (5.49).

It can be easily seen that the computational structure in Fig. 5.10 requires 9 multiplications, 21 additions, and 2 shifts. Then, taking into account the permutation (5.36) or (5.47), the forward/backward MDCT computation in MP3 for the short block requires 9 multiplications, 27/21 additions, and 2 shifts, and this MDCT efficient implementation achieves the minimal multiplicative complexity known up to now.

5.4.3.6 Case N = 36 (Long Data Block)

Now consider the MDCT implementation for the long audio block represented in the matrix-vector form as $\mathbf{c}^T = \boldsymbol{C}_{18}^{IV} \mathbf{y}^T$. Deriving the unnormalized symmetric DCT-IV matrix $\boldsymbol{C}_{18}^{IV}$ in the explicit form one can observe again a special structure of some of its basis vectors. For frequency and time indices

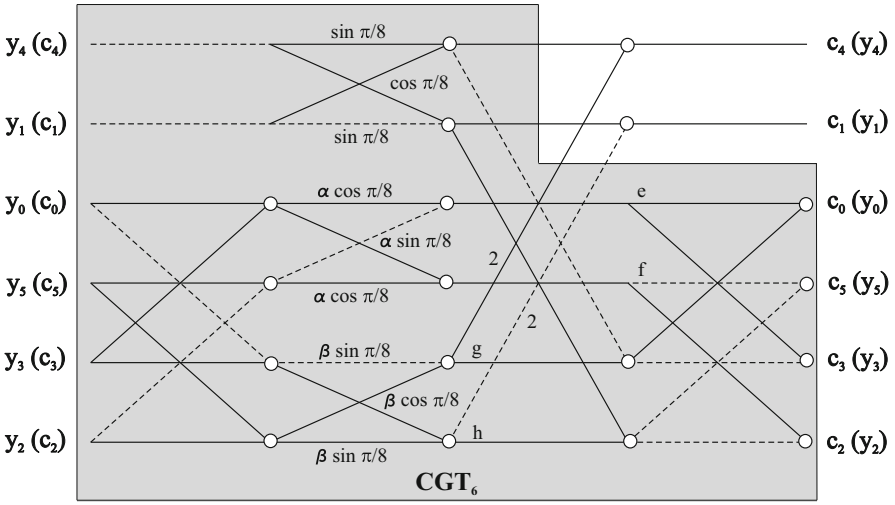

Fig. 5.10 Signal flow graph for the 6-point DCT-IV computation, $\alpha = \frac{\sqrt{3}}{2}$ and $\beta = \frac{1}{2}$

$k, n = 0, 2, 3, 5, 6, 8, 9, 11, 12, 14, 15, 17$, the corresponding row/column basis vectors always consist of 18 different elements in magnitude whereas for $k, n = 1, 4, 7, 10, 13, 16$ consist of only 6 different elements in magnitude. Particularly, writing explicitly the matrix-vector product $\boldsymbol{C}_{18}^{IV}\ \mathbf{y}^T$ we immediately realize that the MDCT coefficients $\{c_k\}$ for $k = 1, 4, 7, 10, 13, 16$ can be obtained directly from the 6-point DCT-IV of a folded data sequence defined as $\{y_j - y_{11-j} - y_{12+j}\}$, $j = 0, 1, 2, 3, 4, 5$, i.e., the MDCT coefficients $\{c_k\}$ for $k = 1, 4, 7, 10, 13, 16$, derived in the matrix-vector form are given by [38]

$$\begin{pmatrix} c_1 \\ c_4 \\ c_7 \\ c_{10} \\ c_{13} \\ c_{16} \end{pmatrix} = \boldsymbol{C}_6^{IV} \begin{pmatrix} y_0 - y_{11} - y_{12} \\ y_1 - y_{10} - y_{13} \\ y_2 - y_9 - y_{14} \\ y_3 - y_8 - y_{15} \\ y_4 - y_7 - y_{16} \\ y_5 - y_6 - y_{17} \end{pmatrix}. \tag{5.71}$$

This fact implies that the 6-point fast DCT-IV computational structure for the short block (see Fig. 5.10) can be reused for the long block to compute the MDCT coefficients $\{c_k\}$ for $k = 1, 4, 7, 10, 13, 16$. It requires 9 multiplications, 33 additions, and 2 shifts.

For remaining 12×18 sub-matrix of matrix $\boldsymbol{C}_{18}^{IV}$ the approach used for the short block is now extended to the long block. As before, the time and frequency indices

n and k, respectively, are partitioned into two sets. The set $S_1 = \{1, 4, 7, 10, 13, 16\}$ is made up of those values j for which $(2j + 1)$ is a multiple of 3, and the set $S_2 = \{0, 2, 3, 5, 6, 8, 9, 11, 12, 14, 15, 17\}$ otherwise. If the computation of transform coefficients $\{c_k\}$ with respect to the time indices restricted to sets S_1 and S_2 is denoted by $c_k^{(1)}$ and $c_k^{(2)}$, respectively, then remaining MDCT transform coefficients $\{c_k\}$ are obtained as

$$c_k = c_k^{(1)} + c_k^{(2)}, \qquad k \in S_2. \tag{5.72}$$

At first, the transform components $\{c_k^{(1)}\}$, where $k \in S_2$ and $n \in S_1$, are given by

$$\begin{pmatrix} c_0^{(1)} \\ c_2^{(1)} \\ c_3^{(1)} \\ c_5^{(1)} \\ c_6^{(1)} \\ c_8^{(1)} \\ c_9^{(1)} \\ c_{11}^{(1)} \\ c_{12}^{(1)} \\ c_{14}^{(1)} \\ c_{15}^{(1)} \\ c_{17}^{(1)} \end{pmatrix} = \begin{pmatrix} \cos\frac{\pi}{24} & \cos\frac{3\pi}{24} & \cos\frac{5\pi}{24} & \cos\frac{7\pi}{24} & \cos\frac{9\pi}{24} & \cos\frac{11\pi}{24} \\ \cos\frac{5\pi}{24} & -\cos\frac{9\pi}{24} & -\cos\frac{\pi}{24} & -\cos\frac{11\pi}{24} & \cos\frac{3\pi}{24} & \cos\frac{7\pi}{24} \\ \cos\frac{7\pi}{24} & -\cos\frac{3\pi}{24} & -\cos\frac{11\pi}{24} & \cos\frac{\pi}{24} & -\cos\frac{9\pi}{24} & -\cos\frac{5\pi}{24} \\ \cos\frac{11\pi}{24} & -\cos\frac{9\pi}{24} & \cos\frac{7\pi}{24} & -\cos\frac{5\pi}{24} & \cos\frac{3\pi}{24} & -\cos\frac{\pi}{24} \\ -\cos\frac{11\pi}{24} & \cos\frac{9\pi}{24} & -\cos\frac{7\pi}{24} & \cos\frac{5\pi}{24} & -\cos\frac{3\pi}{24} & \cos\frac{\pi}{24} \\ -\cos\frac{7\pi}{24} & \cos\frac{3\pi}{24} & \cos\frac{11\pi}{24} & -\cos\frac{\pi}{24} & \cos\frac{9\pi}{24} & \cos\frac{5\pi}{24} \\ -\cos\frac{5\pi}{24} & \cos\frac{9\pi}{24} & \cos\frac{\pi}{24} & \cos\frac{11\pi}{24} & -\cos\frac{3\pi}{24} & -\cos\frac{7\pi}{24} \\ -\cos\frac{\pi}{24} & -\cos\frac{3\pi}{24} & -\cos\frac{5\pi}{24} & -\cos\frac{7\pi}{24} & -\cos\frac{9\pi}{24} & -\cos\frac{11\pi}{24} \\ -\cos\frac{\pi}{24} & -\cos\frac{3\pi}{24} & -\cos\frac{5\pi}{24} & -\cos\frac{7\pi}{24} & -\cos\frac{9\pi}{24} & -\cos\frac{11\pi}{24} \\ -\cos\frac{5\pi}{24} & \cos\frac{9\pi}{24} & \cos\frac{\pi}{24} & \cos\frac{11\pi}{24} & -\cos\frac{3\pi}{24} & -\cos\frac{7\pi}{24} \\ -\cos\frac{7\pi}{24} & \cos\frac{3\pi}{24} & \cos\frac{11\pi}{24} & -\cos\frac{\pi}{24} & \cos\frac{9\pi}{24} & \cos\frac{5\pi}{24} \\ -\cos\frac{11\pi}{24} & \cos\frac{9\pi}{24} & -\cos\frac{7\pi}{24} & \cos\frac{5\pi}{24} & -\cos\frac{3\pi}{24} & \cos\frac{\pi}{24} \end{pmatrix} \begin{pmatrix} y_1 \\ y_4 \\ y_7 \\ y_{10} \\ y_{13} \\ y_{16} \end{pmatrix}. \tag{5.73}$$

From (5.73) it can be easily seen that the first four rows of sub-matrix are linearly independent (corresponding to transform components $c_0^{(1)}, c_2^{(1)}, c_3^{(1)}, c_5^{(1)}$), and therefore, it is sufficient to compute only the following matrix-vector product:

$$\begin{pmatrix} c_0^{(1)} \\ c_2^{(1)} \\ c_3^{(1)} \\ c_5^{(1)} \end{pmatrix} = \begin{pmatrix} \cos\frac{\pi}{24} & \cos\frac{3\pi}{24} & \cos\frac{5\pi}{24} & \cos\frac{7\pi}{24} & \cos\frac{9\pi}{24} & \cos\frac{11\pi}{24} \\ \cos\frac{5\pi}{24} & -\cos\frac{9\pi}{24} & -\cos\frac{\pi}{24} & -\cos\frac{11\pi}{24} & \cos\frac{3\pi}{24} & \cos\frac{7\pi}{24} \\ \cos\frac{7\pi}{24} & -\cos\frac{3\pi}{24} & -\cos\frac{11\pi}{24} & \cos\frac{\pi}{24} & -\cos\frac{9\pi}{24} & -\cos\frac{5\pi}{24} \\ \cos\frac{11\pi}{24} & -\cos\frac{9\pi}{24} & \cos\frac{7\pi}{24} & -\cos\frac{5\pi}{24} & \cos\frac{3\pi}{24} & -\cos\frac{\pi}{24} \end{pmatrix} \begin{pmatrix} y_1 \\ y_4 \\ y_7 \\ y_{10} \\ y_{13} \\ y_{16} \end{pmatrix}. \tag{5.74}$$

Further, from (5.73) it also follows that the transform components $c_6^{(1)}, c_8^{(1)}, c_9^{(1)}, c_{11}^{(1)}, c_{12}^{(1)}, c_{14}^{(1)}, c_{15}^{(1)}, c_{17}^{(1)}$ need not be computed separately, because after evaluating the matrix-vector product (5.74) they are directly given by

$$\begin{pmatrix} c_6^{(1)} \\ c_8^{(1)} \\ c_9^{(1)} \\ c_{11}^{(1)} \\ c_{12}^{(1)} \\ c_{14}^{(1)} \\ c_{15}^{(1)} \\ c_{17}^{(1)} \end{pmatrix} = \begin{pmatrix} -c_5^{(1)} \\ -c_3^{(1)} \\ -c_2^{(1)} \\ -c_0^{(1)} \\ -c_0^{(1)} \\ -c_2^{(1)} \\ -c_3^{(1)} \\ -c_5^{(1)} \end{pmatrix}. \tag{5.75}$$

In order to compute (5.74), we partition the sub-matrix in (5.74) into two parts. Defining a 2-point Hankel matrix-vector product as

$$\begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} \cos\frac{9\pi}{24} & -\cos\frac{3\pi}{24} \\ -\cos\frac{3\pi}{24} & -\cos\frac{9\pi}{24} \end{pmatrix} \begin{pmatrix} y_4 \\ y_{13} \end{pmatrix}, \tag{5.76}$$

for the first partitioned sub-matrix we have

$$\begin{pmatrix} \cos\frac{3\pi}{24} & \cos\frac{9\pi}{24} \\ -\cos\frac{9\pi}{24} & \cos\frac{3\pi}{24} \\ -\cos\frac{3\pi}{24} & -\cos\frac{9\pi}{24} \\ -\cos\frac{9\pi}{24} & \cos\frac{3\pi}{24} \end{pmatrix} \begin{pmatrix} y_4 \\ y_{13} \end{pmatrix} = \begin{pmatrix} -q \\ -p \\ q \\ -p \end{pmatrix}, \tag{5.77}$$

and after proper row/column reordering and sign changes made in the remaining symmetric 4×4 sub-matrix and associated input/output data vectors we have

$$\begin{pmatrix} c_0^{(1)} \\ -c_5^{(1)} \\ \hline c_3^{(1)} \\ c_2^{(1)} \end{pmatrix} = \left(\begin{array}{cc|cc} \cos\frac{\pi}{24} & -\cos\frac{11\pi}{24} & \cos\frac{7\pi}{24} & \cos\frac{5\pi}{24} \\ -\cos\frac{11\pi}{24} & -\cos\frac{\pi}{24} & \cos\frac{5\pi}{24} & -\cos\frac{7\pi}{24} \\ \hline \cos\frac{7\pi}{24} & \cos\frac{5\pi}{24} & \cos\frac{\pi}{24} & -\cos\frac{11\pi}{24} \\ \cos\frac{5\pi}{24} & -\cos\frac{7\pi}{24} & -\cos\frac{11\pi}{24} & -\cos\frac{\pi}{24} \end{array}\right) \begin{pmatrix} y_1 \\ -y_{16} \\ \hline y_{10} \\ y_7 \end{pmatrix}. \quad (5.78)$$

Equations (5.76), (5.77), and (5.78) are quite similar to (5.62), (5.64), and (5.65), respectively, and therefore, we can use the approach applied to the short block to compute (5.74). Similarly, defining two combined 2-point Hankel matrix-vector products we get

$$\begin{pmatrix} e \\ f \end{pmatrix} = \frac{1}{2}(\boldsymbol{A}_2 + \boldsymbol{B}_2) \begin{pmatrix} y_1 + y_{10} \\ y_7 - y_{16} \end{pmatrix} = \frac{\sqrt{3}}{2} \begin{pmatrix} \cos\frac{3\pi}{24} & \cos\frac{9\pi}{24} \\ \cos\frac{9\pi}{24} & -\cos\frac{3\pi}{24} \end{pmatrix} \begin{pmatrix} y_1 + y_{10} \\ y_7 - y_{16} \end{pmatrix},$$

$$\begin{pmatrix} g \\ h \end{pmatrix} = \frac{1}{2}(\boldsymbol{A}_2 - \boldsymbol{B}_2) \begin{pmatrix} y_1 - y_{10} \\ -y_7 - y_{16} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \cos\frac{9\pi}{24} & -\cos\frac{3\pi}{24} \\ -\cos\frac{3\pi}{24} & -\cos\frac{9\pi}{24} \end{pmatrix} \begin{pmatrix} y_1 - y_{10} \\ -y_7 - y_{16} \end{pmatrix}. \quad (5.79)$$

From (5.76), (5.77), and (5.79) it follows that the transform components $c_0^{(1)}, c_2^{(1)}, c_3^{(1)}, c_5^{(1)}$ are given by

$$\begin{aligned} c_0^{(1)} &= e + (g - q), \\ c_2^{(1)} &= f - (p + h), \\ c_3^{(1)} &= e - (g - q), \\ c_5^{(1)} &= -f - (p + h). \end{aligned} \quad (5.80)$$

Comparing (5.70) and (5.80) it is evident that the 6-point DCT-IV computational structure shown in Fig. 5.10 can be again reused (but incompletely) to compute transform components $c_0^{(1)}, c_2^{(1)}, c_3^{(1)}, c_5^{(1)}$. The corresponding computational structure originally called the 6-point cosine group transform (CGT_6) [38, 40] is a part of the 6-point DCT-IV computational structure. It is indicated in Fig. 5.10 by shaded polygon, where the associated reordered input data vector is given by

$[y_{13}, y_4, y_1, y_{16}, y_{10}, y_7]^T$. The computation of transform components $c_0^{(1)}, c_2^{(1)}, c_3^{(1)}, c_5^{(1)}$ requires 9 multiplications and 19 additions.

The computation of transform components $\{c_k^{(2)}\}$ when $k, n \in S_2$ for the remaining symmetric 12×12 sub-matrix is realized separately. Let s denote the element $\cos \frac{\pi s}{72}$ of the symmetric 12×12 sub-matrix. After its proper row/column reordering and sign changes (actually predicted by the structure of a cyclic group [38]) and associated input/output data vectors, the computation of transform components $\{c_k^{(2)}\}$, $k, n \in S_2$ can be represented in the matrix-vector form as

$$\begin{pmatrix} c_0^{(2)} \\ c_9^{(2)} \\ c_{11}^{(2)} \\ c_2^{(2)} \\ -c_{12}^{(2)} \\ c_{14}^{(2)} \\ c_8^{(2)} \\ c_{17}^{(2)} \\ c_{15}^{(2)} \\ c_6^{(2)} \\ c_3^{(2)} \\ c_5^{(2)} \end{pmatrix} = \begin{pmatrix} 1 & 19 & 23 & 5 & -25 & 29 & 17 & 35 & 31 & 13 & 7 & 11 \\ 19 & -1 & 5 & -23 & 29 & 25 & 35 & -17 & 13 & -31 & 11 & -7 \\ 23 & 5 & -25 & 29 & -1 & -19 & -31 & -13 & 7 & 11 & 17 & 35 \\ 5 & -23 & 29 & 25 & -19 & 1 & -13 & 31 & 11 & -7 & 35 & 17 \\ -25 & 29 & -1 & -19 & -23 & -5 & -7 & -11 & 17 & 35 & -31 & -13 \\ 29 & 25 & -19 & 1 & -5 & 23 & -11 & 7 & 35 & -17 & -13 & 31 \\ 17 & 35 & -31 & -13 & -7 & -11 & 1 & 19 & -23 & -5 & 25 & -29 \\ 35 & -17 & -13 & 31 & -11 & 7 & 19 & -1 & -5 & 23 & -29 & -25 \\ 31 & 13 & 7 & 11 & 17 & 35 & -23 & -5 & -25 & 29 & -1 & -19 \\ 13 & -31 & 11 & -7 & 35 & -17 & -5 & 23 & 29 & 25 & -19 & 1 \\ 7 & 11 & 17 & 35 & -31 & -13 & 25 & -29 & -1 & -19 & -23 & -5 \\ 11 & -7 & 35 & -17 & -13 & 31 & -29 & -25 & -19 & 1 & -5 & 23 \end{pmatrix}$$

$$= \begin{pmatrix} y_0 \\ y_9 \\ y_{11} \\ y_2 \\ -y_{12} \\ y_{14} \\ y_8 \\ y_{17} \\ y_{15} \\ y_6 \\ y_3 \\ y_5 \end{pmatrix}. \tag{5.81}$$

One can observe that the sub-matrix in (5.81) is a block cyclic matrix with each block being a 2×2 Hankel matrix. Extending the similar procedure applied to (5.65) or (5.78) , i.e., combining always two proper 2-point Hankel matrix-vector products, and using the following trigonometric identities

$$\cos\frac{\pi}{72}+\cos\frac{17\pi}{72}=2\,\cos\frac{\pi}{9}\,\cos\frac{\pi}{8}=2\,\cos\frac{\pi}{9}\,\cos\frac{3\pi}{24},$$
$$\cos\frac{\pi}{72}-\cos\frac{17\pi}{72}=2\,\sin\frac{\pi}{9}\,\sin\frac{\pi}{8}=2\,\sin\frac{\pi}{9}\,\cos\frac{9\pi}{24},$$

$$\cos\frac{19\pi}{72}+\cos\frac{35\pi}{72}=2\,\cos\frac{\pi}{9}\,\cos\frac{3\pi}{8}=2\,\cos\frac{\pi}{9}\,\sin\frac{\pi}{8},$$
$$\cos\frac{19\pi}{72}-\cos\frac{35\pi}{72}=2\,\sin\frac{\pi}{9}\,\sin\frac{3\pi}{8}=2\,\sin\frac{\pi}{9}\,\cos\frac{\pi}{8},$$

$$\cos\frac{23\pi}{72}+\cos\frac{31\pi}{72}=2\,\cos\frac{\pi}{18}\,\cos\frac{3\pi}{8}=2\,\sin\frac{4\pi}{9}\,\sin\frac{\pi}{8},\qquad \cos\frac{\pi}{18}=\sin\frac{4\pi}{9},$$
$$\cos\frac{23\pi}{72}-\cos\frac{31\pi}{72}=2\,\sin\frac{\pi}{18}\,\sin\frac{3\pi}{8}=2\,\cos\frac{4\pi}{9}\,\cos\frac{\pi}{8},\qquad \sin\frac{\pi}{18}=\cos\frac{4\pi}{9},$$

$$\cos\frac{5\pi}{72}+\cos\frac{13\pi}{72}=2\,\cos\frac{\pi}{18}\,\cos\frac{\pi}{8}=2\,\sin\frac{4\pi}{9}\,\cos\frac{3\pi}{24},$$

$$\cos\frac{5\pi}{72}-\cos\frac{13\pi}{72}=2\,\sin\frac{\pi}{18}\,\sin\frac{\pi}{8}=2\,\cos\frac{4\pi}{9}\,\cos\frac{9\pi}{24},$$

$$\cos\frac{25\pi}{72}+\cos\frac{7\pi}{72}=2\,\cos\frac{2\pi}{9}\,\cos\frac{\pi}{8}=2\,\cos\frac{2\pi}{9}\,\cos\frac{3\pi}{24},$$
$$\cos\frac{25\pi}{72}-\cos\frac{7\pi}{72}=-2\,\sin\frac{2\pi}{9}\,\sin\frac{\pi}{8}=-2\,\sin\frac{2\pi}{9}\,\cos\frac{9\pi}{24},$$

$$\cos\frac{29\pi}{72}+\cos\frac{11\pi}{72}=2\,\cos\frac{5\pi}{18}\,\cos\frac{\pi}{8}=2\,\sin\frac{2\pi}{9}\,\cos\frac{\pi}{8},\qquad \cos\frac{5\pi}{18}=\sin\frac{2\pi}{9},$$
$$\cos\frac{29\pi}{72}-\cos\frac{11\pi}{72}=-2\,\sin\frac{5\pi}{18}\,\sin\frac{\pi}{8}=-2\,\cos\frac{2\pi}{9}\,\sin\frac{\pi}{8},\qquad \sin\frac{5\pi}{18}=\cos\frac{2\pi}{9},$$

we subsequently cover the entire sub-matrix in (5.81), and after rigorous algebraic manipulations we get the following system of algebraic equations:

$$\begin{pmatrix} e_1 \\ f_1 \end{pmatrix} = \cos\frac{\pi}{9}\,\boldsymbol{R}_2\,\mathbf{u}_1+\cos\frac{4\pi}{9}\,\boldsymbol{R}_2\,\mathbf{u}_2-\cos\frac{2\pi}{9}\,\boldsymbol{R}_2\,\mathbf{u}_3 = \quad (a)+(b),$$
$$\begin{pmatrix} g_1 \\ h_1 \end{pmatrix} = \sin\frac{\pi}{9}\,\boldsymbol{Q}_2\,\mathbf{v}_1+\sin\frac{4\pi}{9}\,\boldsymbol{Q}_2\,\mathbf{v}_2+\sin\frac{2\pi}{9}\,\boldsymbol{Q}_2\,\mathbf{v}_3 = \quad (a')-(b'),$$

$$
\begin{aligned}
\begin{pmatrix} e_2 \\ f_2 \end{pmatrix} &= \cos\frac{4\pi}{9}\,\boldsymbol{R}_2\,\mathbf{u}_1 - \cos\frac{2\pi}{9}\,\boldsymbol{R}_2\,\mathbf{u}_2 - \cos\frac{\pi}{9}\,\boldsymbol{R}_2\,\mathbf{u}_3 = \quad (a)-(c),\\
\begin{pmatrix} g_2 \\ h_2 \end{pmatrix} &= \sin\frac{4\pi}{9}\,\boldsymbol{Q}_2\,\mathbf{v}_1 + \sin\frac{2\pi}{9}\,\boldsymbol{Q}_2\,\mathbf{v}_2 - \sin\frac{\pi}{9}\,\boldsymbol{Q}_2\,\mathbf{v}_3 = \quad (a')-(c'),\\
\begin{pmatrix} e_3 \\ f_3 \end{pmatrix} &= -\cos\frac{2\pi}{9}\,\boldsymbol{R}_2\,\mathbf{u}_1 - \cos\frac{\pi}{9}\,\boldsymbol{R}_2\,\mathbf{u}_2 - \cos\frac{4\pi}{9}\,\boldsymbol{R}_2\,\mathbf{u}_3 = -(b)-(c),\\
\begin{pmatrix} g_3 \\ h_3 \end{pmatrix} &= \sin\frac{2\pi}{9}\,\boldsymbol{Q}_2\,\mathbf{v}_1 - \sin\frac{\pi}{9}\,\boldsymbol{Q}_2\,\mathbf{v}_2 - \sin\frac{4\pi}{9}\,\boldsymbol{Q}_2\,\mathbf{v}_3 = \quad (b')-(c'),
\end{aligned}
\tag{5.82}
$$

where $\boldsymbol{R}_2$ and $\boldsymbol{Q}_2$ are, respectively, 2×2 Hankel matrices given by

$$
\boldsymbol{R}_2 = \begin{pmatrix} \cos\frac{3\pi}{24} & \cos\frac{9\pi}{24} \\ \cos\frac{9\pi}{24} & -\cos\frac{3\pi}{24} \end{pmatrix}, \qquad \boldsymbol{Q}_2 = \begin{pmatrix} \cos\frac{9\pi}{24} & \cos\frac{3\pi}{24} \\ \cos\frac{3\pi}{24} & -\cos\frac{9\pi}{24} \end{pmatrix}.
$$

$\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are, respectively, 2-point vectors given by

$$
\mathbf{u}_1 = \begin{pmatrix} y_0 + y_8 \\ y_9 + y_{17} \end{pmatrix}, \qquad \mathbf{u}_2 = \begin{pmatrix} y_{11} - y_{15} \\ y_2 - y_6 \end{pmatrix}, \qquad \mathbf{u}_3 = \begin{pmatrix} -y_{12} - y_3 \\ y_{14} - y_5 \end{pmatrix},
$$

$$
\mathbf{v}_1 = \begin{pmatrix} y_0 - y_8 \\ y_9 - y_{17} \end{pmatrix}, \qquad \mathbf{v}_2 = \begin{pmatrix} y_{11} + y_{15} \\ y_2 + y_6 \end{pmatrix}, \qquad \mathbf{v}_3 = \begin{pmatrix} -y_{12} + y_3 \\ y_{14} + y_5 \end{pmatrix}.
$$

For the transform components $\{c_k^{(2)}\}$, $k \in S_2$ the following relations hold:

$$
\begin{aligned}
c_0^{(2)} &= e_1 + g_1, \quad c_{11}^{(2)} = e_2 + g_2, \quad c_{12}^{(2)} = -e_3 - g_3,\\
c_8^{(2)} &= e_1 - g_1, \quad c_{15}^{(2)} = g_2 - e_2, \quad c_3^{(2)} = \quad g_3 - e_3,\\
c_9^{(2)} &= f_1 + h_1, \quad c_2^{(2)} = f_2 + h_2, \quad c_{14}^{(2)} = \quad f_3 + h_3,\\
c_{17}^{(2)} &= f_1 - h_1, \quad c_6^{(2)} = h_2 - f_2, \quad c_5^{(2)} = \quad h_3 - f_3.
\end{aligned}
\tag{5.83}
$$

The system of algebraic equations (5.82) requires to evaluate eighteen 2-point Hankel matrix-vector products. However, using the trigonometric identities

$$
\cos\frac{\pi}{9} = \cos\frac{4\pi}{9} + \cos\frac{2\pi}{9}, \quad \sin\frac{\pi}{9} = \sin\frac{4\pi}{9} - \sin\frac{2\pi}{9},
$$

the system of equations (5.82) can be evaluated by combining only six 2-point Hankel matrix-vector products which are defined as

$$\begin{aligned}
&(a) \quad \cos\tfrac{4\pi}{9}\, \boldsymbol{R}_2\,(\mathbf{u}_1+\mathbf{u}_2), && (a') \quad \sin\frac{4\pi}{9}\, \boldsymbol{Q}_2\,(\mathbf{v}_1+\mathbf{v}_2),\\
&(b) \quad \cos\tfrac{2\pi}{9}\, \boldsymbol{R}_2\,(\mathbf{u}_1-\mathbf{u}_3), && (b') \quad \sin\frac{2\pi}{9}\, \boldsymbol{Q}_2\,(\mathbf{v}_1-\mathbf{v}_3),\\
&(c) \quad \cos\tfrac{\pi}{9}\, \boldsymbol{R}_2\,(\mathbf{u}_2+\mathbf{u}_3), && (c') \quad \sin\frac{\pi}{9}\, \boldsymbol{Q}_2\,(\mathbf{v}_2+\mathbf{v}_3),
\end{aligned} \tag{5.84}$$

whereby $\mathbf{u}_1-\mathbf{u}_3=(\mathbf{u}_1+\mathbf{u}_2)-(\mathbf{u}_2+\mathbf{u}_3)$ and $\mathbf{v}_1-\mathbf{v}_3=(\mathbf{v}_1+\mathbf{v}_2)-(\mathbf{v}_2+\mathbf{v}_3)$. Combinations of expressions in (5.84) are explicitly indicated on the right-hand side of (5.82). Further, note on the right-hand side of (5.82), similarly the following relations hold:

$$\begin{aligned}
(a)-(c) &= [(a)+(b)]+[-(b)-(c)],\\
(a')-(c') &= [(a')-(b')]+[(b')-(c')].
\end{aligned}$$

The 2-point Hankel matrix-vector products can be realized by the corresponding bilinear algorithm (see Appendix A.2). The resulting regular fast computational structure for the computation of transform components $\{c_k^{(2)}\}$, $k \in S_2$ is shown in Fig. 5.11. It can be easily seen that the fast computational structure in Fig. 5.11 requires 18 multiplications and 66 additions. The remaining MDCT coefficients $\{c_k\}$, $k \in S_2$ are obtained according to (5.72) combining results of two fast computational structures taking into account (5.75) and (5.83).

If we take into account the permutation (5.36) or (5.47), the forward/backward MDCT computation in MP3 for the long block requires totally 36 multiplications, 148/130 additions and 2 shifts, and this MDCT efficient implementation again achieves the minimal multiplicative complexity known up to now. Considering the fast analysis/synthesis MDCT filter banks described in Sect. 5.5 together with efficient MDCT implementations [38, 40] we get perhaps the most efficient implementations of the complete analysis/synthesis MDCT or MLT filter banks in MP3 both in terms of the arithmetic complexity and structural simplicity.

Note 12: Open problem remains to prove that the most efficient MDCT implementations [38, 40] are optimal in terms of the multiplicative complexity.

5.4.3.7 DCT-IV-Based MDCT Implementation [29]

A universal rotation-based fast MDCT computational structure based on a novelty fast rotation-based DCT-IV computational structure has been proposed in [27]. Specifically, for $N'=\frac{N}{4}$ being odd, the fast rotation-based MDCT computational structure for k odd/even is defined as [29]

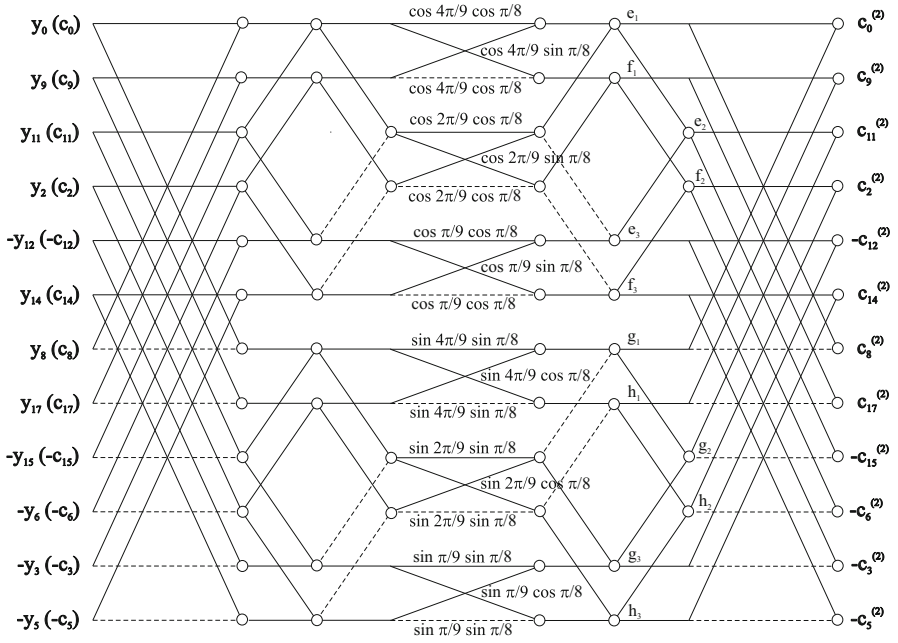

Fig. 5.11 Computational structure for the computation of transform components $\{c_k^{(2)}\}$, $k \in S_2$

$$c_{2k} = \sum_{n=0}^{\lfloor \frac{N'}{2} \rfloor - 1} \left[\left(a_n \mp a_{\frac{N}{4}-1-n} \right) \cos \varphi_{k,n} + \left(b_n \pm b_{\frac{N}{4}-1-n} \right) \sin \varphi_{k,n} \right],$$

$$c_{\frac{N}{2}-1-2k} = \sum_{n=0}^{\lfloor \frac{N'}{2} \rfloor - 1} (-1)^{n+1} \left[-\left(a_n \pm a_{\frac{N}{4}-1-n} \right) \sin \varphi_{k,n} + \left(b_n \mp b_{\frac{N}{4}-1-n} \right) \cos \varphi_{k,n} \right],$$

$$c_{\frac{N}{2}-2k} = \sum_{n=0}^{\lfloor \frac{N'}{2} \rfloor - 1} (-1)^{n} \left[\left(b_n \mp b_{\frac{N}{4}-1-n} \right) \cos \varphi_{k,n} + \left(a_n \pm a_{\frac{N}{4}-1-n} \right) \sin \varphi_{k,n} \right],$$

$$c_{2k-1} = \sum_{n=0}^{\lfloor \frac{N'}{2} \rfloor - 1} \left[-\left(b_n \pm b_{\frac{N}{4}-1-n} \right) \sin \varphi_{k,n} + \left(a_n \mp a_{\frac{N}{4}-1-n} \right) \cos \varphi_{k,n} \right],$$

$$\varphi_{k,n} = \frac{\pi}{2N'}(2n+1)k, \qquad k = 1, 2, \ldots, \left\lfloor \frac{N'}{2} \right\rfloor, \tag{5.85}$$

where $\lfloor\ .\ \rfloor$ denotes the lower integer part of an argument, and

$$\begin{aligned} a_n &= \quad y_n\ \cos\frac{\pi}{2N}(2n+1) + \ y_{\frac{N}{2}-1-n}\ \sin\frac{\pi}{2N}(2n+1), \\ b_n &= -\ y_n\ \sin\frac{\pi}{2N}(2n+1) + \ y_{\frac{N}{2}-1-n}\ \cos\frac{\pi}{2N}(2n+1), \quad n = 0, 1, \ldots, \frac{N}{4}-1, \end{aligned} \tag{5.86}$$

defines the block of $\frac{N}{4}$ Givens–Jacobi rotations. For $k = 0$ the coefficients c_0 and $c_{\frac{N}{2}-1}$ are given by

$$c_0 = a_{\frac{N'-1}{2}} + \sum_{n=0}^{\lfloor\frac{N'}{2}\rfloor-1} (a_n + a_{\frac{N}{4}-1-n}),$$

$$c_{\frac{N}{2}-1} = (-1)^{\frac{N'-1}{2}+1}\, b_{\frac{N'-1}{2}} + \sum_{n=0}^{\lfloor\frac{N'}{2}\rfloor-1} (-1)^{n+1}\, (b_n + b_{\frac{N}{4}-1-n}). \tag{5.87}$$

The data sequence $\{y_n\}$ in the forward MDCT computation is given by (5.47) whereas the time domain aliased data sequence $\{\hat{x}_n\}$ in the backward MDCT can be recovered from $\{y_n\}$ according to (5.48).

5.4.3.8 Case N = 12 (Short Data Block)

For the short block we have $N' = \frac{N}{4} = 3$, $\frac{N'-1}{2} = 1$ and $\lfloor\frac{N'}{2}\rfloor = 1$. For the forward MDCT computation the data sequence $\{y_n\}$ given by (5.47) requires 6 additions. According to (5.86) the block of three Givens–Jacobi rotations of vectors $[y_n, y_{5-n}]^T$ with rotation angles $\frac{\pi}{24}(2n+1)$, $n = 0, 1, 2$, requires 9 multiplications and 9 additions. Efficient implementations of Givens–Jacobi rotations are presented in Appendices F.2 and F.3.

Extending (5.87) for coefficients $c_0,\ c_5$, followed by extending (5.85) for $k = 1$ being odd, $n = 0$, and the angle $\varphi_{1,0} = \frac{\pi}{6}$ for the pairs of coefficients $c_2,\ c_1$ and $c_3,\ c_4$, after some algebraic manipulations we get [29]

$$\begin{aligned} c_0 &= a_1 + (a_0 + a_2), \\ c_5 &= b_1 - (b_0 + b_2), \\ c_2 &= \quad b_1 + (a_0 - a_2)\ \cos\frac{\pi}{6} + (b_0 + b_2)\ \sin\frac{\pi}{6}, \\ c_1 &= -b_1 + (a_0 - a_2)\ \cos\frac{\pi}{6} - (b_0 + b_2)\ \sin\frac{\pi}{6}, \\ c_3 &= -a_1 + (a_0 + a_2)\ \sin\frac{\pi}{6} - (b_0 - b_2)\ \cos\frac{\pi}{6}, \\ c_4 &= -a_1 + (a_0 + a_2)\ \sin\frac{\pi}{6} + (b_0 - b_2)\ \cos\frac{\pi}{6}. \end{aligned} \tag{5.88}$$

From (5.88) it can be easily seen that the pairs of coefficients c_2, c_1 and c_3, c_4 contain redundant algebraic expressions. Now we can straightforwardly generate an efficient implementation of (5.85) and (5.87) for the short block in the form of the following linear code [29]:

$$\begin{aligned} p_0 &= a_0 + a_2 & q_0 &= b_0 + b_2 \\ p_1 &= (a_0 - a_2)\ \cos\tfrac{\pi}{6} & q_1 &= b_1 + q_0\ \sin\tfrac{\pi}{6} \\ p_2 &= p_0\ \sin\tfrac{\pi}{6} - a_1 & q_2 &= (b_0 - b_2)\ \cos\tfrac{\pi}{6} \end{aligned}$$

$$\begin{aligned} c_0 &= a_1 + p_0 \\ c_1 &= p_1 - q_1 \\ c_2 &= p_1 + q_1 \\ c_3 &= p_2 - q_2 \\ c_4 &= p_2 + q_2 \\ c_5 &= b_1 - q_0 \end{aligned} \tag{5.89}$$

where $\cos\frac{\pi}{6} = \frac{\sqrt{3}}{2}$ and $\sin\frac{\pi}{6} = \frac{1}{2}$. Equation (5.89) requires 2 multiplications, 12 additions, and 2 shifts. For the short block (5.86) and (5.89) define the identical fast computational structure for the forward/backward MDCT computation which is represented by the signal flow graph shown in Fig. 5.12. Thus, the forward/backward MDCT computation requires totally 11 multiplications, 27/21 additions, and 2 shifts.

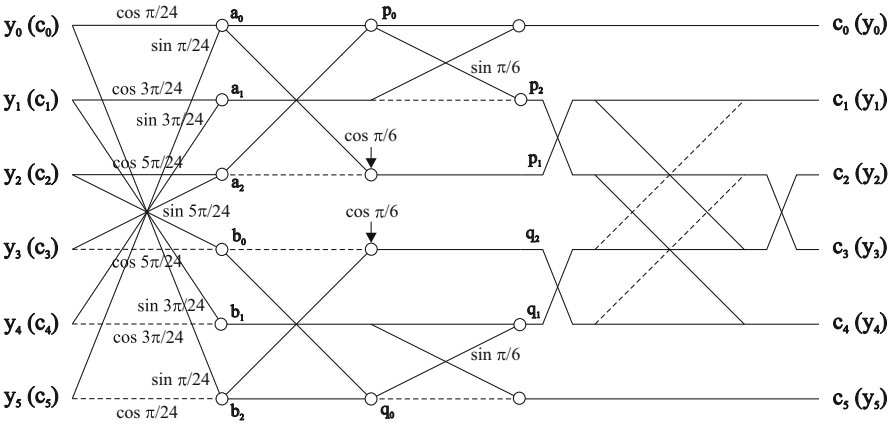

Fig. 5.12 The fast computational structure for the forward/backward MDCT computation when $N = 12$

Note 13: Investigating the fast computational structure in Fig. 5.12 for the short block and comparing with that of [38] (see Fig. 5.10 in the previous subsection), it can be seen that the implementation in Fig. 5.12 requires just about two multiplications more while the number of additions and shifts is the same. This fact indicates that there exist two redundant multiplications in the implementation shown in Fig. 5.12. Indeed, twiddle factors $\cos\frac{\pi}{6}$ are redundant. They can be absorbed (and similarly $\sin\frac{\pi}{6}$) into the following introduced block of three Givens–Jacobi rotations [29]:

$$
\begin{aligned}
p &= (-y_4)\cos\frac{\pi}{8} - (-y_1)\sin\frac{\pi}{8},\\
q &= (-y_4)\sin\frac{\pi}{8} + (-y_1)\cos\frac{\pi}{8},
\end{aligned}
$$

$$
\begin{aligned}
e &= (y_0 + y_3)\cos\frac{\pi}{6}\cos\frac{\pi}{8} - (y_5 - y_2)\cos\frac{\pi}{6}\sin\frac{\pi}{8},\\
f &= (y_0 + y_3)\cos\frac{\pi}{6}\sin\frac{\pi}{8} + (y_5 - y_2)\cos\frac{\pi}{6}\cos\frac{\pi}{8},
\end{aligned}
$$

$$
\begin{aligned}
g &= (y_2 + y_5)\sin\frac{\pi}{6}\cos\frac{\pi}{8} - (y_3 - y_0)\sin\frac{\pi}{6}\sin\frac{\pi}{8},\\
h &= (y_2 + y_5)\sin\frac{\pi}{6}\sin\frac{\pi}{8} + (y_3 - y_0)\sin\frac{\pi}{6}\cos\frac{\pi}{8}.
\end{aligned}
\tag{5.90}
$$

Then, after the computation of Givens–Jacobi rotations defined by (5.90) taking into account trigonometric identities, the MDCT coefficients are obtained by combining output values (*p, q, e, f, g, h*) of rotated vectors as follows:

$$
\begin{aligned}
c_0 &= \quad e + (g - q),\\
c_1 &= \quad p - 2h,\\
c_2 &= \quad f - (p + h),\\
c_3 &= \quad e - (g - q),\\
c_4 &= \quad q + 2g,\\
c_5 &= -f - (p + h).
\end{aligned}
\tag{5.91}
$$

Equations (5.47)/(5.48), (5.90), and (5.91) define the modified fast computational structure identical for the efficient forward/backward MDCT computation requiring 9 multiplications, 27/21 additions, and 2 shifts, thus achieving the same minimal multiplicative complexity as in [38, 40]. The modified fast computational structure is shown in Fig. 5.13. (Scaled) Givens–Jacobi rotations can be realized by a corresponding bilinear computational structure (see Appendix F.3).

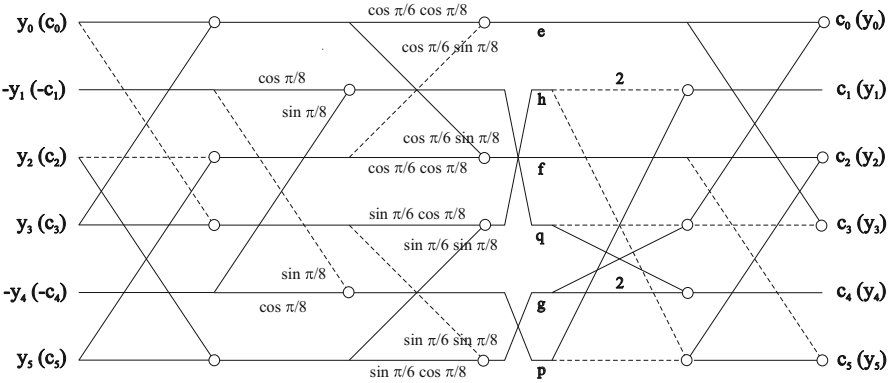

Fig. 5.13 The modified fast computational structure for the forward/backward MDCT when $N = 12$

5.4.3.9 Case N = 36 (Long Data Block)

For $N = 36$ we have $N' = \frac{N}{4} = 9$, $\frac{N'-1}{2} = 4$, and $\lfloor \frac{N'}{2} \rfloor = 4$. For the forward MDCT computation the data sequence $\{y_n\}$ given by (5.47) requires 18 additions. According to (5.86) the block of nine Givens–Jacobi rotations of vectors $[y_n, y_{17-n}]^T$ with rotation angles $\frac{\pi}{72}(2n+1)$, $n = 0, 1, 2, 3, 4, 5, 6, 7, 8$, requires 27 multiplications and 27 additions. Efficient implementations of Givens–Jacobi rotations are presented in Appendices F.2 and F.3.

Similarly, extending (5.87) for coefficients c_0, c_{17}, followed by extending (5.85) for $k = 1, 2, 3, 4$ and $n = 0, 1, 2, 3$ for the pairs of coefficients c_{2k-1}, c_{2k} and c_{17-2k}, c_{18-2k}, $k = 1, 2, 3, 4$, after rigorous algebraic manipulations and using trigonometric identities we obtain the following explicit algebraic expressions [29]

$$
\begin{aligned}
c_0 &= a_4 + (a_0 + a_8) + (a_1 + a_7) + (a_2 + a_6) + (a_3 + a_5), \\
c_{17} &= -b_4 - (b_0 + b_8) + (b_1 + b_7) - (b_2 + b_6) + (b_3 + b_5), \\
c_1 &= -b_4 + \left[(a_1 - a_7) \cos \frac{\pi}{6} \right] \\
&\quad + \left[(a_0 - a_8) \sin \frac{4\pi}{9} + (a_2 - a_6) \sin \frac{2\pi}{9} + (a_3 - a_5) \sin \frac{\pi}{9} \right] \\
&\quad - \left[(b_1 + b_7) \sin \frac{\pi}{6} \right] \\
&\quad + \left[(b_0 + b_8) \cos \frac{4\pi}{9} + (b_2 + b_6) \cos \frac{2\pi}{9} + (b_3 + b_5) \cos \frac{\pi}{9} \right],
\end{aligned}
$$

$$
\begin{aligned}
c_2 = \quad & b_4 + \left[(a_1 - a_7)\cos\frac{\pi}{6}\right] \\
& + \left[(a_0 - a_8)\sin\frac{4\pi}{9} + (a_2 - a_6)\sin\frac{2\pi}{9} + (a_3 - a_5)\sin\frac{\pi}{9}\right] \\
& + \left[(b_1 + b_7)\sin\frac{\pi}{6}\right] \\
& + \left[(b_0 + b_8)\cos\frac{4\pi}{9} + (b_2 + b_6)\cos\frac{2\pi}{9} + (b_3 + b_5)\cos\frac{\pi}{9}\right], \\
c_3 = & -a_4 + \left[(a_1 + a_7)\sin\frac{\pi}{6}\right] \\
& + \left[(a_0 + a_8)\cos\frac{\pi}{9} - (a_2 + a_6)\cos\frac{4\pi}{9} - (a_3 + a_5)\cos\frac{2\pi}{9}\right] \\
& - \left[(b_1 - b_7)\cos\frac{\pi}{6}\right] \\
& + \left[(b_0 - b_8)\sin\frac{\pi}{9} + (b_2 - b_6)\sin\frac{4\pi}{9} + (b_3 - b_5)\sin\frac{2\pi}{9}\right], \\
c_4 = & -a_4 + \left[(a_1 + a_7)\sin\frac{\pi}{6}\right] \\
& + \left[(a_0 + a_8)\cos\frac{\pi}{9} - (a_2 + a_6)\cos\frac{4\pi}{9} - (a_3 + a_5)\cos\frac{2\pi}{9}\right] \\
& + \left[(b_1 - b_7)\cos\frac{\pi}{6}\right] \\
& + \left[(b_0 - b_8)\sin\frac{\pi}{9} + (b_2 - b_6)\sin\frac{4\pi}{9} + (b_3 - b_5)\sin\frac{2\pi}{9}\right], \\
c_5 = & \, b_4 + [(a_0 - a_8) - (a_2 - a_6) - (a_3 - a_5)]\cos\frac{\pi}{6} \\
& - (b_1 + b_7) + [(b_0 + b_8) + (b_2 + b_6) - (b_3 + b_5)]\sin\frac{\pi}{6}, \\
c_6 = & -b_4 + [(a_0 - a_8) - (a_2 - a_6) - (a_3 - a_5)]\cos\frac{\pi}{6} \\
& + (b_1 + b_7) + [(b_0 + b_8) + (b_2 + b_6) - (b_3 + b_5)]\sin\frac{\pi}{6}, \\
c_7 = & \, a_4 - \left[(a_1 + a_7)\sin\frac{\pi}{6}\right] \\
& + \left[(a_0 + a_8)\cos\frac{2\pi}{9} - (a_2 + a_6)\cos\frac{\pi}{9} + (a_3 + a_5)\cos\frac{4\pi}{9}\right] \\
& - \left[(b_1 - b_7)\cos\frac{\pi}{6}\right]
\end{aligned}
$$

$$
\begin{aligned}
&+\left[(b_0-b_8)\sin\frac{2\pi}{9}-(b_2-b_6)\sin\frac{\pi}{9}-(b_3-b_5)\sin\frac{4\pi}{9}\right],\\
c_8 &= a_4-\left[(a_1+a_7)\sin\frac{\pi}{6}\right]\\
&+\left[(a_0+a_8)\cos\frac{2\pi}{9}-(a_2+a_6)\cos\frac{\pi}{9}+(a_3+a_5)\cos\frac{4\pi}{9}\right]\\
&+\left[(b_1-b_7)\cos\frac{\pi}{6}\right]\\
&+\left[(b_0-b_8)\sin\frac{2\pi}{9}-(b_2-b_6)\sin\frac{\pi}{9}-(b_3-b_5)\sin\frac{4\pi}{9}\right],\\
c_{15} &= a_4-\left[(a_1+a_7)\sin\frac{\pi}{6}\right]\\
&+\left[(a_0+a_8)\cos\frac{4\pi}{9}+(a_2+a_6)\cos\frac{2\pi}{9}-(a_3+a_5)\cos\frac{\pi}{9}\right]\\
&-\left[-(b_1-b_7)\cos\frac{\pi}{6}\right]\\
&+\left[(b_0-b_8)\sin\frac{4\pi}{9}+(b_2-b_6)\sin\frac{2\pi}{9}-(b_3-b_5)\sin\frac{\pi}{9}\right],\\
c_{16} &= a_4-\left[(a_1+a_7)\sin\frac{\pi}{6}\right]\\
&+\left[(a_0+a_8)\cos\frac{4\pi}{9}+(a_2+a_6)\cos\frac{2\pi}{9}-(a_3+a_5)\cos\frac{\pi}{9}\right]\\
&+\left[-(b_1-b_7)\cos\frac{\pi}{6}\right]\\
&+\left[(b_0-b_8)\sin\frac{4\pi}{9}+(b_2-b_6)\sin\frac{2\pi}{9}-(b_3-b_5)\sin\frac{\pi}{9}\right],\\
c_{13} &= b_4-\left[(a_1-a_7)\cos\frac{\pi}{6}\right]\\
&+\left[(a_0-a_8)\sin\frac{\pi}{9}+(a_2-a_6)\sin\frac{4\pi}{9}-(a_3-a_5)\sin\frac{2\pi}{9}\right]\\
&-\left[-(b_1+b_7)\sin\frac{\pi}{6}\right]\\
&+\left[(b_0+b_8)\cos\frac{\pi}{9}-(b_2+b_6)\cos\frac{4\pi}{9}+(b_3+b_5)\cos\frac{2\pi}{9}\right],\\
c_{14} &= -b_4-\left[(a_1-a_7)\cos\frac{\pi}{6}\right]\\
&+\left[(a_0-a_8)\sin\frac{\pi}{9}+(a_2-a_6)\sin\frac{4\pi}{9}-(a_3-a_5)\sin\frac{2\pi}{9}\right]
\end{aligned}
$$

$$
\begin{aligned}
&+\left[-(b_1+b_7)\sin\frac{\pi}{6}\right]\\
&+\left[(b_0+b_8)\cos\frac{\pi}{9}-(b_2+b_6)\cos\frac{4\pi}{9}+(b_3+b_5)\cos\frac{2\pi}{9}\right],\\
c_{11} &= -a_4-(a_1+a_7)+[(a_0+a_8)+(a_2+a_6)+(a_3+a_5)]\sin\frac{\pi}{6}\\
&-[(b_0-b_8)-(b_2-b_6)+(b_3-b_5)]\cos\frac{\pi}{6},\\
c_{12} &= -a_4-(a_1+a_7)+[(a_0+a_8)+(a_2+a_6)+(a_3+a_5)]\sin\frac{\pi}{6}\\
&+[(b_0-b_8)-(b_2-b_6)+(b_3-b_5)]\cos\frac{\pi}{6},\\
c_9 &= -b_4-\left[(a_1-a_7)\cos\frac{\pi}{6}\right]\\
&+\left[(a_0-a_8)\sin\frac{2\pi}{9}-(a_2-a_6)\sin\frac{\pi}{9}+(a_3-a_5)\sin\frac{4\pi}{9}\right]\\
&-\left[(b_1+b_7)\sin\frac{\pi}{6}\right]\\
&+\left[(b_0+b_8)\cos\frac{2\pi}{9}-(b_2+b_6)\cos\frac{\pi}{9}-(b_3+b_5)\cos\frac{4\pi}{9}\right],\\
c_{10} &= b_4-\left[(a_1-a_7)\cos\frac{\pi}{6}\right]\\
&+\left[(a_0-a_8)\sin\frac{2\pi}{9}-(a_2-a_6)\sin\frac{\pi}{9}+(a_3-a_5)\sin\frac{4\pi}{9}\right]\\
&+\left[(b_1+b_7)\sin\frac{\pi}{6}\right]\\
&+\left[(b_0+b_8)\cos\frac{2\pi}{9}-(b_2+b_6)\cos\frac{\pi}{9}-(b_3+b_5)\cos\frac{4\pi}{9}\right]. \qquad (5.92)
\end{aligned}
$$

From (5.92) it can be seen that each pair of coefficients c_{2k-1}, c_{2k} and c_{17-2k}, c_{18-2k}, $k = 1, 2, 3, 4$, is the result of addition and subtraction of the same unique algebraic expressions in brackets. This fact enables us to concentrate on the efficient computation of half the coefficients. In order to reduce the multiplicative complexity, further development is based on classical complexity theory and it is divided into four parts.

At first, consider a group of algebraic expressions extracted from (5.92) for coefficients c_{15}, c_4, and c_8, respectively, as

$$\begin{aligned}
&\left[(a_0+a_8)\cos\frac{4\pi}{9}+(a_2+a_6)\cos\frac{2\pi}{9}-(a_3+a_5)\cos\frac{\pi}{9}\right]=(a)+(c),\\
&\left[(a_0+a_8)\cos\frac{\pi}{9}-(a_2+a_6)\cos\frac{4\pi}{9}-(a_3+a_5)\cos\frac{2\pi}{9}\right]=(a)+(b),\\
&\left[(a_0+a_8)\cos\frac{2\pi}{9}-(a_2+a_6)\cos\frac{\pi}{9}+(a_3+a_5)\cos\frac{4\pi}{9}\right]=(b)-(c).
\end{aligned} \tag{5.93}$$

The direct approach to compute (5.93) requires nine multiplications and six additions provided the terms a_0+a_8, a_2+a_6, and a_3+a_5 are precomputed. The multiplicative complexity can be reduced as follows. Using the trigonometric identity:

$$\cos\frac{4\pi}{9}+\cos\frac{2\pi}{9}=\cos\frac{\pi}{9},$$

and decomposing $\cos\frac{2\pi}{9}$ in the first expression of (5.93), then $\cos\frac{\pi}{9}$ in the second expression, and finally, $\cos\frac{4\pi}{9}$ in the third expression, we get

$$\begin{aligned}
&\left[(a_0+a_8)\cos\frac{4\pi}{9}+(a_2+a_6)\left(\cos\frac{\pi}{9}-\cos\frac{4\pi}{9}\right)-(a_3+a_5)\cos\frac{\pi}{9}\right]\\
&\quad=[(\mathbf{a_0}+\mathbf{a_8})-(\mathbf{a_2}+\mathbf{a_6})]\cos\frac{\mathbf{4}\pi}{9}+[(a_2+a_6)-(a_3+a_5)]\cos\frac{\pi}{9},
\end{aligned}$$

$$\begin{aligned}
&\left[(a_0+a_8)\left(\cos\frac{4\pi}{9}+\cos\frac{2\pi}{9}\right)-(a_2+a_6)\cos\frac{4\pi}{9}-(a_3+a_5)\cos\frac{2\pi}{9}\right]\\
&\quad=[(a_0+a_8)-(a_2+a_6)]\cos\frac{4\pi}{9}+[(\mathbf{a_0}+\mathbf{a_8})-(\mathbf{a_3}+\mathbf{a_5})]\cos\frac{\mathbf{2}\pi}{9},
\end{aligned}$$

$$\begin{aligned}
&\left[(a_0+a_8)\cos\frac{2\pi}{9}-(a_2+a_6)\cos\frac{\pi}{9}+(a_3+a_5)\left(\cos\frac{\pi}{9}-\cos\frac{2\pi}{9}\right)\right]\\
&\quad=[(a_0+a_8)-(a_3+a_5)]\cos\frac{2\pi}{9}-[(\mathbf{a_2}+\mathbf{a_6})-(\mathbf{a_3}+\mathbf{a_5})]\cos\frac{\pi}{\mathbf{9}}.
\end{aligned} \tag{5.94}$$

One can easily see from (5.94) that for evaluation of (5.93) we need to combine three algebraic expressions which are highlighted by bold. Thus, introducing

$$
\begin{aligned}
&(a) \quad [(a_0 + a_8) - (a_2 + a_6)] \cos\frac{4\pi}{9},\\
&(b) \quad [(a_0 + a_8) - (a_3 + a_5)] \cos\frac{2\pi}{9},\\
&(c) \quad [(a_2 + a_6) - (a_3 + a_5)] \cos\frac{\pi}{9},
\end{aligned}
\tag{5.95}
$$

the efficient computation of (5.93) requires only three multiplications and six additions. Combinations of expressions in (5.95) are explicitly indicated on the right-hand side of (5.93).

Now, consider a group of algebraic expressions extracted from (5.92) for coefficients c_2, c_{13}, and c_9, respectively, as

$$
\begin{aligned}
&\left[(a_0 - a_8)\sin\frac{4\pi}{9} + (a_2 - a_6)\sin\frac{2\pi}{9} + (a_3 + a_5)\sin\frac{\pi}{9}\right] = (a) - (c),\\
&\left[(a_0 - a_8)\sin\frac{\pi}{9} + (a_2 - a_6)\sin\frac{4\pi}{9} - (a_3 + a_5)\sin\frac{2\pi}{9}\right] = (a) - (b),\\
&\left[(a_0 - a_8)\sin\frac{2\pi}{9} - (a_2 - a_6)\sin\frac{\pi}{9} + (a_3 + a_5)\sin\frac{4\pi}{9}\right] = (b) - (c).
\end{aligned}
\tag{5.96}
$$

Using the trigonometric identity:

$$
\sin\frac{4\pi}{9} - \sin\frac{2\pi}{9} = \sin\frac{\pi}{9},
$$

and decomposing $\sin\frac{2\pi}{9}$ in the first expression of (5.96), then $\sin\frac{\pi}{9}$ in the second expression, and finally, $\sin\frac{4\pi}{9}$ in the third expression, we obtain the following three algebraic expressions

$$
\begin{aligned}
&(a) \quad [(a_0 - a_8) + (a_2 - a_6)] \sin\frac{4\pi}{9},\\
&(b) \quad [(a_0 - a_8) + (a_3 - a_5)] \sin\frac{2\pi}{9},\\
&(c) \quad [(a_2 - a_6) - (a_3 - a_5)] \sin\frac{\pi}{9}.
\end{aligned}
\tag{5.97}
$$

Again, combinations of expressions in (5.97) are explicitly indicated on the right-hand side of (5.96). Following the same procedure for the remaining group of algebraic expressions for coefficients c_2, c_{13}, and c_9, respectively, as

$$\left[(b_0+b_8)\cos\frac{4\pi}{9}+(b_2+b_6)\cos\frac{2\pi}{9}+(b_3+b_5)\cos\frac{\pi}{9}\right]=(a)+(c),$$
$$\left[(b_0+b_8)\cos\frac{\pi}{9}-(b_2+b_6)\cos\frac{4\pi}{9}+(b_3+b_5)\cos\frac{2\pi}{9}\right]=(a)+(b),$$
$$\left[(b_0+b_8)\cos\frac{2\pi}{9}-(b_2+b_6)\cos\frac{\pi}{9}-(b_3+b_5)\cos\frac{4\pi}{9}\right]=(b)-(c), \tag{5.98}$$

we obtain

$$\begin{aligned}(a)\quad & [(b_0+b_8)-(b_2+b_6)]\ \cos\frac{4\pi}{9},\\ (b)\quad & [(b_0+b_8)+(b_3+b_5)]\ \cos\frac{2\pi}{9},\\ (c)\quad & [(b_2+b_6)+(b_3+b_5)]\ \cos\frac{\pi}{9},\end{aligned} \tag{5.99}$$

and finally, for a group of algebraic expressions for coefficients c_{15}, c_4 and c_8, respectively, as

$$\left[(b_0-b_8)\sin\frac{4\pi}{9}+(b_2-b_6)\sin\frac{2\pi}{9}-(b_3-b_5)\cos\frac{\pi}{9}\right]=(a)-(c),$$
$$\left[(b_0-b_8)\sin\frac{\pi}{9}+(b_2-b_6)\sin\frac{4\pi}{9}+(b_3-b_5)\cos\frac{2\pi}{9}\right]=(a)-(b),$$
$$\left[(b_0-b_8)\sin\frac{2\pi}{9}-(b_2-b_6)\sin\frac{\pi}{9}-(b_3-b_5)\cos\frac{4\pi}{9}\right]=(b)-(c), \tag{5.100}$$

we obtain

$$\begin{aligned}(a)\quad & [(b_0-b_8)+(b_2-b_6)]\ \sin\frac{4\pi}{9},\\ (b)\quad & [(b_0-b_8)-(b_3-b_5)]\ \sin\frac{2\pi}{9},\\ (c)\quad & [(b_2-b_6)+(b_3-b_5)]\ \sin\frac{\pi}{9}.\end{aligned} \tag{5.101}$$

Now we can straightforwardly generate an efficient implementation of (5.85) and (5.87) for the long block in the form of the following linear code [29]:

$$
\begin{array}{llll}
p_0 = a_0 + a_8 & q_0 = a_0 - a_8 & u_0 = b_0 + b_8 & v_0 = b_0 - b_8 \\
p_1 = a_1 + a_7 & q_1 = a_1 - a_7 & u_1 = b_1 + b_7 & v_1 = b_1 - b_7 \\
p_2 = a_2 + a_6 & q_2 = a_2 - a_6 & u_2 = b_2 + b_6 & v_2 = b_2 - b_6 \\
p_3 = a_3 + a_5 & q_3 = a_3 - a_5 & u_3 = b_3 + b_5 & v_3 = b_3 - b_5
\end{array}
$$

$$
\begin{array}{llll}
p_4 = p_0 + p_2 & q_4 = q_0 - q_2 & u_4 = u_0 + u_2 & v_4 = v_0 - v_2, \\
p_5 = (p_0 - p_2)\cos\frac{4\pi}{9} & q_5 = (q_0 + q_2)\sin\frac{4\pi}{9} & u_5 = (u_0 - u_2)\cos\frac{4\pi}{9} & v_5 = (v_0 + v_2)\sin\frac{4\pi}{9} \\
p_6 = (p_0 - p_3)\cos\frac{2\pi}{9} & q_6 = (q_0 + q_3)\sin\frac{2\pi}{9} & u_6 = (u_0 + u_3)\cos\frac{2\pi}{9} & v_6 = (v_0 - v_3)\sin\frac{2\pi}{9} \\
p_7 = (p_2 - p_3)\cos\frac{\pi}{9} & q_7 = (q_2 - q_3)\sin\frac{\pi}{9} & u_7 = (u_2 + u_3)\cos\frac{\pi}{9} & v_7 = (v_2 + v_3)\sin\frac{\pi}{9}
\end{array}
$$

$$
\begin{array}{llll}
p_8 = p_4 + p_3 & q_8 = q_1\frac{\sqrt{3}}{2} & u_8 = u_4 - u_3 & v_8 = v_1\frac{\sqrt{3}}{2} \\
p_9 = \frac{1}{2}p_8 & q_9 = (q_4 - q_3)\frac{\sqrt{3}}{2} & u_9 = \frac{1}{2}u_8 & v_9 = (v_4 + v_3)\frac{\sqrt{3}}{2} \\
p_{10} = a_4 - \frac{1}{2}p_1 & & u_{10} = b_4 + \frac{1}{2}u_1 & \\
p_{11} = a_4 + p_1 & & u_{11} = u_1 - b_4 & \\
p_{12} = p_5 + p_7 & q_{10} = q_5 - q_7 & u_{12} = u_5 + u_7 & v_{10} = v_5 - v_7 \\
p_{13} = p_5 + p_6 & q_{11} = q_5 - q_6 & u_{13} = u_5 + u_6 & v_{11} = v_5 - v_6 \\
p_{14} = p_6 - p_7 & q_{12} = q_6 - q_7 & u_{14} = u_6 - u_7 & v_{12} = v_6 - v_7
\end{array}
$$

$$
\begin{array}{llll}
p_{15} = p_{10} + p_{12} & q_{13} = q_8 + q_{10} & u_{15} = u_{10} + u_{12} & v_{13} = v_{10} - v_8 \\
p_{16} = p_{13} - p_{10} & q_{14} = q_{11} - q_8 & u_{16} = u_{13} - u_{10} & v_{14} = v_8 + v_{11} \\
p_{17} = p_9 - p_{11} & q_{15} = q_{12} - q_8 & u_{17} = u_{11} + u_9 & v_{15} = v_8 + v_{12} \\
p_{18} = p_{10} + p_{14} & & u_{18} = u_{10} + u_{14} &
\end{array}
$$

$$
\begin{array}{ll}
c_0 = p_8 + p_{11} & c_{17} = u_{11} - u_8 \\
c_1 = q_{13} - u_{15} & c_{15} = p_{15} - v_{13} \\
c_2 = q_{13} + u_{15} & c_{16} = p_{15} + v_{13} \\
c_3 = p_{16} - v_{14} & c_{13} = q_{14} - u_{16} \\
c_4 = p_{16} + v_{14} & c_{14} = q_{14} + u_{16} \\
c_5 = q_9 - u_{17} & c_{11} = p_{17} - v_9 \\
c_6 = q_9 + u_{17} & c_{12} = p_{17} + v_9 \\
c_7 = p_{18} - v_{15} & c_9 = q_{15} - u_{18} \\
c_8 = p_{18} + v_{15} & c_{10} = q_{15} + u_{18}
\end{array} \tag{5.102}
$$

requiring 16 multiplications, 84 additions, and 4 shifts. For the long block (5.86) and (5.102) define the identical fast computational structure both for the forward and backward MDCT computation. Thus, the forward/backward MDCT computation requires totally 43 multiplications, 129/111 additions, and 4 shifts. In contrast to the short block, the identical fast computational structure for the forward/backward MDCT computation when $N = 36$ cannot be represented by a regular signal flow graph.

Note 14: Therefore, the open problem remains to derive a modified 18-point fast DCT-IV computational structure with the minimal multiplicative complexity.

Note 15: The efficient MDCT implementations in MP3 audio coding standard in the form of linear code are particularly suitable for high-performance programmable DSP processors.

5.4.4 DCT-IV/(Scaled) DCT-II-Based Efficient MDCT Implementations

We recall that applying the permutation (5.36) or (5.47) to the input data sequence $\{x_n\}$, the N-point forward MDCT is converted to the $\frac{N}{2}$-point DCT-IV of $\{y_n\}$. Developed fast DCT-IV/DCT-II-based MDCT algorithms [34, 35, 45, 51, 54, 65, 66, 69] and DCT-IV/scaled DCT-II (SDCT-II)-based MDCT algorithms [50, 66] utilize the following simple fact. Since there exists a simple relation between the DCT-IV and the DCT-II matrices [119], the $\frac{N}{2}$-point DCT-IV may be converted to the DCT-II (SDCT-II) of the same size at the cost of additional $\frac{N}{2}$ pre-multiplications and $\frac{N}{2}-1$ recursive post-additions [121, 132].

Now let us summarize basic facts for developing DCT-IV/(S)DCT-II-based efficient MDCT implementations. Unnormalized M-point DCT-IV and DCT-II of an input data sequence $\{y_m\}$, $m = 0, 1, \ldots, M-1$ are, respectively, defined as [119]

$$\begin{aligned} c_k^{IV} &= \sum_{m=0}^{M-1} y_m \cos\left[\frac{\pi}{4M}(2m+1)(2k+1)\right], \\ c_k^{II} &= \sum_{m=0}^{M-1} y_m \cos\left[\frac{\pi(2m+1)k}{2M}\right], \qquad k = 0, 1, \ldots, M-1, \end{aligned} \tag{5.103}$$

whereas an unnormalized M-point scaled DCT-II (SDCT-II) is defined as [50]

$$s_k^{II} = \epsilon_k \sum_{m=0}^{M-1} y_m \cos\left[\frac{\pi(2m+1)k}{2M}\right], \qquad k = 0, 1, \ldots, M-1, \tag{5.104}$$

where

$$\epsilon_k = \begin{cases} 1, & \text{if } k = 0, \\ 2, & \text{otherwise.} \end{cases} \tag{5.105}$$

Using trigonometric identity $\cos(\alpha+\beta) = 2\cos\alpha\ \cos\beta - \cos(\alpha-\beta)$ to the DCT-IV transform kernel setting $\alpha = \frac{\pi(2m+1)k}{2M}$ and $\beta = \frac{\pi(2m+1)}{4M}$, the M-point DCT-IV can be converted to the DCT-II of the same size as [121, 132]

$$c_k^{IV} = \sum_{m=0}^{M-1} \left(2\, y_m \, \cos \frac{\pi(2m+1)}{4M} \right) \cos \left[\frac{\pi(2m+1)k}{2M} \right] - c_{k-1}^{IV}, \quad k = 0, 1, \ldots, M-1, \tag{5.106}$$

or to the SDCT-II of the same size as [50]

$$c_k^{IV} = \epsilon_k \sum_{m=0}^{M-1} \left(y_m \, \cos \frac{\pi(2m+1)}{4M} \right) \cos \left[\frac{\pi(2m+1)k}{2M} \right] - c_{k-1}^{IV}, \quad k = 0, 1, \ldots, M-1, \tag{5.107}$$

where transform coefficients $\{c_k^{IV}\}$ and $\{c_k^{II}\}$, $(\{s_k^{II}\})$ satisfy the following conditions:

$$\begin{aligned} 2\, c_0^{IV} &= c_0^{II}, \quad (c_0^{IV} = s_0^{II}) \quad \text{if} \quad k = 0, \\ c_k^{IV} + c_{k-1}^{IV} &= c_k^{II}, \quad (c_k^{IV} + c_{k-1}^{IV} = s_k^{II}) \quad \text{if} \quad k > 0. \end{aligned} \tag{5.108}$$

On the other hand, it is well known that an M-point (S)DCT-II of $\{u_m\}$, whereby $u_m = 2\, y_m \, \cos \frac{\pi(2m+1)}{4M}$ for the DCT-II and $u_m = y_m \, \cos \frac{\pi(2m+1)}{4M}$ for the SDCT-II, it can be recursively decomposed into $\frac{M}{2}$-point (S)DCT-II (even-indexed transform coefficients) and $\frac{M}{2}$-point DCT-IV (odd-indexed transform coefficients) as [50, 119, 121, 132]

$$\begin{aligned} c_{2k}^{II} &= \sum_{m=0}^{\frac{M}{2}-1} (u_m + u_{M-1-m}) \, \cos \left[\frac{\pi(2m+1)k}{2(M/2)} \right], \\ c_{2k+1}^{II} &= \sum_{m=0}^{\frac{M}{2}-1} (u_m - u_{M-1-m}) \, \cos \left[\frac{\pi}{2M}(2m+1)(2k+1) \right], \; k = 0, 1, \ldots, \frac{M}{2} - 1. \end{aligned} \tag{5.109}$$

Then, using (C.33) from Appendix C.3, the $\frac{M}{2}$-point DCT-IV in (5.109) corresponding to odd-indexed transform coefficients $\{c_{2k+1}^{II}\}$ can be converted to the $\frac{M}{2}$-point (S)DCT-II as

$$\begin{aligned} c_{2k+1}^{II} &= \sum_{m=0}^{\frac{M}{2}-1} \left(2\, (u_m - u_{M-1-m}) \, \cos \frac{\pi(2m+1)}{2M} \right) \cos \left[\frac{\pi(2m+1)k}{2(M/2)} \right] - c_{2k-1}^{II}, \\ & k = 0, 1, \ldots, \frac{M}{2} - 1, \end{aligned} \tag{5.110}$$

where

$$\begin{aligned} 2\, c_1^{II} &= \hat{c}_0^{II}, \quad (c_1^{II} = \hat{s}_0^{II}) \quad \text{if} \quad k = 0, \\ c_{2k+1}^{II} + c_{2k-1}^{II} &= \hat{c}_k^{II}, \quad (c_{2k+1}^{II} + c_{2k-1}^{II} = \hat{s}_k^{II}) \quad \text{if} \quad k > 0. \end{aligned} \tag{5.111}$$

Equations (5.109) and (5.110) show that the M-point (S)DCT-II is decomposed into two identical $\frac{M}{2}$-point (S)DCTs-II. Since the DCT-IV matrix is symmetric and self-inverse, the forward/inverse DCT-IV computation via the (S)DCT-II (note that the DCT-II matrix is not symmetric), and hence as well as the forward/backward MDCT can be realized by an identical fast computational structure. In order to efficiently implement the MDCT in MP3 we should specify a suitable fast DCT-II algorithm/computational structure which is valid for N being an even integer, i.e., 6- and 18-point DCT-II fast algorithms. In general, the even-length DCT-II can be realized by:

- An indirect fast DCT-II algorithm which maps the DCT-II into a complex-valued DFT of half size [6].
- Recursive filter structures for the DCT-II general lengths [113, 119, 123, 124, 141] (see references on pp. 14–15).
- Combining fast radix-q DCT-II algorithms (where q is an odd integer) with existing mixed-radix fast DCT-II algorithms for the composite lengths $2^n \times q$ [114, 119, 122] (see references on pp. 11–13), or with the direct fast even-length recursive DCT-II algorithms [121, 132].

5.4.4.1 Representative DCT-IV/DCT-II-Based MDCT Implementation

Substituting $M = \frac{N}{2}$ and $m = n$ into (5.106), (5.109), and (5.110), we get complete formulae defining the DCT-IV/DCT-II-based efficient MDCT implementation as

$$c_k^{IV} = \sum_{n=0}^{\frac{N}{2}-1} \left(2\, y_n \, \cos \frac{\pi(2n+1)}{2N} \right) \cos \left[\frac{\pi(2n+1)k}{2(N/2)} \right] - c_{k-1}^{IV}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{5.112}$$

where $\{y_n\}$ is given by (5.36) or (5.47), and

$$\begin{aligned} c_{2k}^{II} &= \sum_{n=0}^{\frac{N}{4}-1} (u_n + u_{\frac{N}{2}-1-n}) \, \cos \left[\frac{\pi(2n+1)k}{2(N/4)} \right], \\ c_{2k+1}^{II} &= \sum_{n=0}^{\frac{N}{4}-1} \left(2\, (u_n - u_{\frac{N}{2}-1-n}) \, \cos \frac{\pi(2n+1)}{N} \right) \cos \left[\frac{\pi(2n+1)k}{2(N/4)} \right] - c_{2k-1}^{II}, \\ & k = 0, 1, \ldots, \frac{N}{4} - 1, \end{aligned} \tag{5.113}$$

where $u_n = 2\, y_n \, \cos \frac{\pi(2n+1)}{2N}$, and transform coefficients $\{c_k^{IV}\}$ and $\{c_k^{II}\}$ are, respectively, given by (5.108) and (5.111). For the forward/backward MDCT computation the even-length fast recursive DCT-II algorithm [121, 132] is used. The regular signal flow graph identical for the computation both of the forward

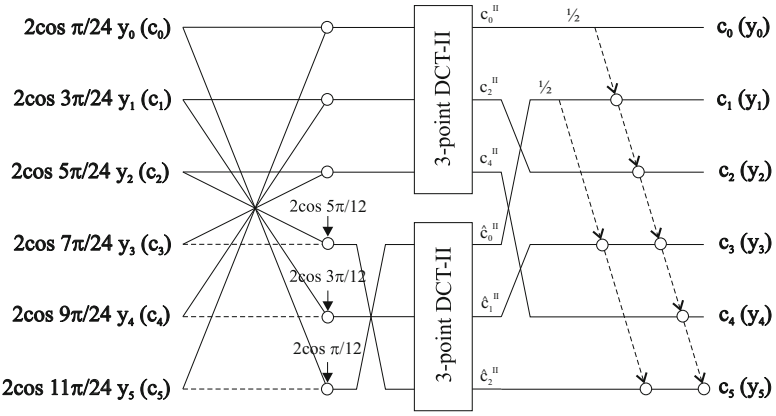

Fig. 5.14 Signal flow graph for the DCT-IV/DCT-II-based forward/backward MDCT computation for $N = 12$

and backward MDCT for $N = 12$ is shown in Fig. 5.14. The optimized efficient 3/9-point DCT-II modules are presented in Appendix D.4. The time domain aliased data sequence $\{\hat{x}_n\}$ is recovered after the backward MDCT from $\{y_n\}$ according to (5.48).

The total computational complexity of the forward MDCT is $3\frac{N}{4}$ multiplications and $7\frac{N}{4}-2$ additions ($5\frac{N}{4}-2$ additions for the backward MDCT) plus the arithmetic complexity of two identical $\frac{N}{4}$-point DCT-II modules. Thus, for the short block the forward/backward MDCT computation in MP3 requires 11 multiplications, 27/21 additions, and 4 shifts, while for the long block the forward/backward MDCT computation requires 43 multiplications, 129/111 additions, and 6 shifts.

Note 16: A similar efficient backward MDCT implementation in MP3 has been reported in [55], but $\frac{N}{2}$-point DCT-IV is converted to $\frac{N}{2}$-point DCT-III, being inverse of the DCT-II, which is further decomposed into two $\frac{N}{4}$-point DCTs-III. For $N = 12$ its arithmetic complexity is 11 multiplications, 27 additions, and 2 shifts. In the case of $N = 36$, the 9-point DCT-III is again factorized into 4-point and 5-point DCTs-III thus causing the increase in arithmetic complexity (81 multiplications and 149 additions).

5.4.4.2 DCT-IV/SDCT-II-Based MDCT Implementation [50]

Substituting $M = \frac{N}{2}$ and $m = n$ into (5.107), (5.109), and (5.110) we get complete formulae defining the DCT-IV/SDCT-II-based efficient MDCT implementation [50]. Transform coefficients $\{c_k^{IV}\}$ and $\{s_k^{II}\}$ are, respectively, given by (5.108) and (5.111), and $u_n = y_n \cos\frac{\pi(2n+1)}{2N}$. The regular signal flow graph identical for the computation both of the forward and backward MDCT for $N = 12$ is shown in

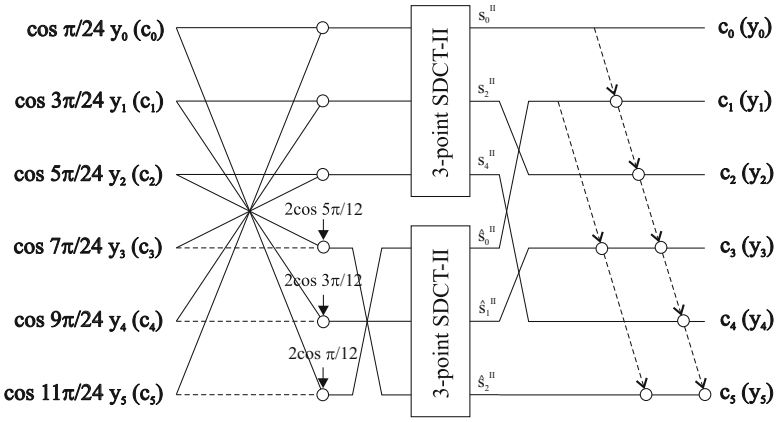

Fig. 5.15 Signal flow graph for the SDCT-IV/DCT-II-based forward/backward MDCT computation for $N = 12$

Fig. 5.15. Comparing Figs. 5.14 and 5.15 it can be seen that the factors 2 and $\frac{1}{2}$ in Fig. 5.14 are removed and 3-point DCT-II modules are replaced by 3-point SDCT-II ones in Fig. 5.15. The optimized efficient 3/9-point SDCT-II modules are presented in Appendix D.5. The time domain aliased data sequence $\{\hat{x}_n\}$ is recovered after the backward MDCT from $\{y_n\}$ according to (5.48).

The total computational complexity of the forward MDCT is $3\frac{N}{4}$ multiplications and $7\frac{N}{4}-2$ additions ($5\frac{N}{4}-2$ additions for the backward MDCT) plus the arithmetic complexity of two identical $\frac{N}{4}$-point SDCT-II modules. Thus, for the short block the forward/backward MDCT computation in MP3 requires 11 multiplications, 27/21 additions, and 2 shifts, while for the long block the forward/backward MDCT computation requires 43 multiplications, 129/111 additions, and 4 shifts.

Note 17: Twiddle factors $2\cos\frac{\pi(2n+1)}{2N}$ or $\cos\frac{\pi(2n+1)}{2N}$ applied to the data sequence $\{y_n\}$ in DCT-IV/(S)DCT-II-based MDCT implementations can be effectively absorbed into the windowing operation in the complete analysis/synthesis MDCT (MLT) filter banks.

5.4.4.3 DCT-IV/SDCT-II-Based MDCT Implementation [66]

An interesting efficient method of computing the MDCT in MP3 has been proposed in [66]. In terms of the multiplicative complexity these DCT-IV/SDCT-II-based MDCT implementations require exactly about one multiplication more compared to [38, 40], specifically, 10 multiplications for the short audio block and 37 multiplications for the long audio block. They are based on a recursive sparse block matrix factorization of DCT-II matrices of order 6 and 18 scaled by a factor

of $\sqrt{2}$, denoted, respectively, by $\sqrt{2}\ \boldsymbol{C}_6^{II}$ and $\sqrt{2}\ \boldsymbol{C}_{18}^{II}$. Since the approach in derivation of efficient MDCT implementations [66] is very valuable it merits to be presented in a detailed and compact form both for the short and long audio blocks separately.

At first, substituting $M = \frac{N}{2}$ and $m = n$ into (5.106) we obtain the basic formulae of DCT-IV/DCT-II-based efficient MDCT implementations defined as

$$c_k^{IV} = \sum_{n=0}^{\frac{N}{2}-1} u_n \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right] - c_{k-1}^{IV},$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{5.114}$$

where

$$u_n = 2\, y_n \cos\frac{\pi(2n+1)}{2N}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{5.115}$$

$\{y_n\}$ in (5.115) is given by (5.36) or (5.47) requiring $\frac{N}{2}$ additions, and transform coefficients $\{c_k^{IV}\}$ are computed from $\{c_k^{II}\}$ according to (5.108). But, using a simple trick, specifically, introducing the scaling factor $\sqrt{2}$ into the sum in (5.114) and immediately the scaling factor $\frac{1}{\sqrt{2}}$ into the right-hand side of (5.115), we obtain equivalent forms of Eqs. (5.114) and (5.115), respectively, as [66]

$$c_k^{IV} = \sum_{n=0}^{\frac{N}{2}-1} u_n \sqrt{2} \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right] - c_{k-1}^{IV}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{5.116}$$

where

$$u_n = \frac{2}{\sqrt{2}}\, y_n \cos\frac{\pi(2n+1)}{2N} = \sqrt{2}\, y_n \cos\frac{\pi(2n+1)}{2N},$$
$$n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{5.117}$$

The cosine transform kernel in (5.116) is recognized as the $\frac{N}{2}$-point SDCT-II scaled by the factor $\sqrt{2}$. The computational complexity of (5.116) is given by the complexity of $\frac{N}{2}$-point SDCT-II plus $\frac{N}{2} - 1$ recursive additions and 1 shift (multiplication by $\frac{1}{2}$) according to (5.108). Equation (5.117) requires $\frac{N}{2}$ multiplications.

Note 18: It is important to note that in the original paper [66] the corresponding Eq. (5.117) involves reciprocal values of cosines defined as $\frac{1}{2\sqrt{2}\cos\frac{\pi(2n+1)}{2N}}$. This fact can cause the numerical instability of efficient MDCT algorithms, particularly for the long block.

Secondly, let $\boldsymbol{T}_M$ be an orthogonal matrix of order M (M is an even integer) with elements $\{t_{k,m}\}$, $k,\ m = 0, 1, \ldots, M-1$, whose basis vectors (rows) are symmetric and anti-symmetric, i.e.,

$$t_{2k,m} = t_{2k,M-1-m}, \qquad t_{2k+1,m} = -\,t_{2k+1,M-1-m}, \quad k,\ m = 0, 1, \ldots, \frac{M}{2} - 1.$$

Just $\boldsymbol{C}_M^{II}$ or $\sqrt{2}\ \boldsymbol{C}_M^{II}$ is a such matrix. It is well known that the reordered matrix $\hat{\boldsymbol{T}}_M$ can be factored into the following (recursive) sparse block matrix factorization [119]

$$\hat{\boldsymbol{T}}_M = \begin{pmatrix} \boldsymbol{A}_{\frac{M}{2}} & \bar{\boldsymbol{A}}_{\frac{M}{2}} \\ \\ \bar{\boldsymbol{B}}_{\frac{M}{2}} & -\boldsymbol{B}_{\frac{M}{2}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_{\frac{M}{2}} & \boldsymbol{0} \\ \\ \boldsymbol{0} & \boldsymbol{B}_{\frac{M}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{M}{2}} & \boldsymbol{J}_{\frac{M}{2}} \\ \\ \boldsymbol{J}_{\frac{M}{2}} & -\boldsymbol{I}_{\frac{M}{2}} \end{pmatrix}, \qquad \begin{aligned} \bar{\boldsymbol{A}}_{\frac{M}{2}} &= \boldsymbol{A}_{\frac{M}{2}}\, \boldsymbol{J}_{\frac{M}{2}}, \\ \\ \bar{\boldsymbol{B}}_{\frac{M}{2}} &= \boldsymbol{B}_{\frac{M}{2}}\, \boldsymbol{J}_{\frac{M}{2}}, \end{aligned} \tag{5.118}$$

where $\boldsymbol{I}_{\frac{M}{2}}$ is the identity matrix and $\boldsymbol{J}_{\frac{M}{2}}$ is reverse ordered identity matrix, both of order $\frac{M}{2}$, and $\boldsymbol{0}$'s are null matrices. Block sub-matrices $(\boldsymbol{A}_{\frac{M}{2}} \quad \bar{\boldsymbol{A}}_{\frac{M}{2}})$ represent the even-indexed symmetric basis vectors of $\boldsymbol{T}_M$, while block sub-matrices $(\bar{\boldsymbol{B}}_{\frac{M}{2}} \quad -\boldsymbol{B}_{\frac{M}{2}})$ represent the odd-indexed anti-symmetric basis vectors of $\boldsymbol{T}_M$.

In summary, (5.116), (5.117), and (5.118) provide the basis in derivation of DCT-IV/SDCT-II-based MDCT implementations in MP3 [66]. Extracting the $\frac{N}{2}$-point SDCT-II from (5.116) we have

$$c_k^{II} = \sum_{n=0}^{\frac{N}{2}-1} u_n\ \sqrt{2}\ \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{5.119}$$

where $\{u_n\}$ is given by (5.117). Equation (5.119) can be represented in the equivalent matrix-vector form as

$$[\mathbf{c}^{II}]^T = \sqrt{2}\ \boldsymbol{C}_{\frac{N}{2}}^{II}\ \mathbf{u}^T. \tag{5.120}$$

Further development both for the short and long audio blocks is focused only on the efficient factorization of SDCT-II matrices $\sqrt{2}\ \boldsymbol{C}_6^{II}$ and $\sqrt{2}\ \boldsymbol{C}_{18}^{II}$.

5.4.4.4 Case N = 12 (Short Data Block)

Consider the matrix-vector product (5.120) for the explicit form of $\sqrt{2}\ \boldsymbol{C}_6^{II}$. Since the reordered SDCT-II matrix $\sqrt{2}\ \hat{\boldsymbol{C}}_6^{II}$ can be factored into the sparse block matrix factorization defined by (5.118), substituting (5.118) into the matrix-vector product (5.120) we have

$$\begin{pmatrix} c_0^{II} \\ c_2^{II} \\ c_4^{II} \\ -- \\ c_1^{II} \\ c_3^{II} \\ c_5^{II} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_3 & \boldsymbol{0} \\ \\ \boldsymbol{0} & \boldsymbol{B}_3 \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_3 & \boldsymbol{J}_3 \\ \\ \boldsymbol{J}_3 & -\boldsymbol{I}_3 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ -- \\ u_3 \\ u_4 \\ u_5 \end{pmatrix}, \tag{5.121}$$

where matrices $\boldsymbol{A}_3$ and $\boldsymbol{B}_3$ are, respectively, given by

$$\boldsymbol{A}_3 = \begin{pmatrix} \sqrt{2} & \sqrt{2} & \sqrt{2} \\ \\ \sqrt{2}\cos\frac{\pi}{6} & 0 & -\sqrt{2}\cos\frac{\pi}{6} \\ \\ \sqrt{2}\cos\frac{\pi}{3} & -\sqrt{2} & \sqrt{2}\cos\frac{\pi}{3} \end{pmatrix}, \quad \boldsymbol{B}_3 = \begin{pmatrix} \sqrt{2}\cos\frac{5\pi}{12} & 1 & \sqrt{2}\cos\frac{\pi}{12} \\ \\ -1 & -1 & 1 \\ \\ \sqrt{2}\cos\frac{\pi}{12} & -1 & \sqrt{2}\cos\frac{5\pi}{12} \end{pmatrix},$$

where $\cos\frac{\pi}{6} = \frac{\sqrt{3}}{2}$, $\cos\frac{\pi}{3} = \frac{1}{2}$ in $\boldsymbol{A}_3$. Further, scaling $\boldsymbol{C}_6^{II}$ by $\sqrt{2}$ produces a useful trigonometric identity between two elements of $\boldsymbol{B}_3$ resulting in saving 1 nontrivial multiplication in the corresponding block matrix-vector product. It is defined as

$$\sqrt{2}\cos\frac{5\pi}{12} = \sqrt{2}\cos\frac{\pi}{12} - 1. \tag{5.122}$$

Performing sequentially block matrix-vector products on the right-hand side of (5.121), and introducing a new vector $\mathbf{v}$ we get

$$\begin{aligned} v_0 &= u_0 + u_5, \quad v_1 = u_1 + u_4, \quad v_2 = u_2 + u_3, \\ v_3 &= u_2 - u_3, \quad v_4 = u_1 - u_4, \quad v_5 = u_0 - u_5, \end{aligned} \tag{5.123}$$

and

$$\begin{aligned} c_0^{II} &= \sqrt{2}\,(v_0 + v_1 + v_2), \\ c_2^{II} &= \sqrt{2}\cos\frac{\pi}{6}\,(v_0 - v_2), \\ c_4^{II} &= \sqrt{2}\cos\frac{\pi}{3}\,(v_0 + v_2) - \sqrt{2}\,v_1, \\ c_1^{II} &= \sqrt{2}\cos\frac{\pi}{12}\,(v_3 + v_5) - (v_3 - v_4), \\ c_3^{II} &= -(v_3 + v_4) + v_5, \\ c_5^{II} &= \sqrt{2}\cos\frac{\pi}{12}\,(v_3 + v_5) - (v_4 + v_5). \end{aligned} \tag{5.124}$$

Now, from (5.124) we are able to complete an efficient computation of SDCT-II coefficients $\{c_k^{II}\}$ for the short audio block in the form of the following linear code:

$$m_1 = \frac{\sqrt{2}}{2}\,(v_0 + v_2) \qquad m_2 = \sqrt{2}\,v_1,$$
$$m_3 = \sqrt{2}\cos\tfrac{\pi}{6}\,(v_0 - v_2)\; m_4 = \sqrt{2}\cos\tfrac{\pi}{12}\,(v_3 + v_5),$$

$$\begin{aligned} c_0^{II} &= 2\,m_1 + m_2, \\ c_1^{II} &= m_4 - (v_3 - v_4), \\ c_2^{II} &= m_3, \\ c_3^{II} &= -(v_3 + v_4) + v_5, \\ c_4^{II} &= m_1 - m_2, \\ c_5^{II} &= m_4 - (v_4 + v_5). \end{aligned} \tag{5.125}$$

Equation (5.123) requires 6 additions while (5.125) requires 4 multiplications, 11 additions, and 1 shift (multiplication by 2). Equations (5.123) and (5.125) define the fast computational structure for the computation of 6-point SDCT-II which totally requires 4 multiplications, 17 additions, and 1 shift.

Equations (5.116), (5.117), (5.123), and (5.125) define the identical fast computational structure both for the forward and backward MDCT computations. $\{y_n\}$ given by (5.36) or (5.47) in (5.117) requires 6 additions, (5.117) requires 6 multiplications, and (5.116) for the computation of transform coefficients $\{c_k^{IV}\}$ from $\{c_k^{II}\}$ according to (5.108) requires 5 recursive additions and 1 shift. Thus, for the short audio block the forward/backward MDCT in MP3 requires totally 10 multiplications, 28/22 additions, and 2 shifts. The time domain aliased data sequence $\{\hat{x}_n\}$ is recovered after the backward MDCT from $\{y_n\}$ according to (5.48).

5.4.4.5 Case N = 36 (Long Data Block)

Now, consider the matrix-vector product (5.120) for the explicit form of $\sqrt{2}\,\boldsymbol{C}_{18}^{II}$. Similarly, the reordered SDCT-II matrix $\sqrt{2}\,\hat{\boldsymbol{C}}_{18}^{II}$ can be factored into the sparse block matrix factorization defined by (5.118), and substituting (5.118) into (5.120) we have

$$\begin{pmatrix} c_0^{II} \\ c_2^{II} \\ c_4^{II} \\ c_6^{II} \\ c_8^{II} \\ c_{10}^{II} \\ c_{12}^{II} \\ c_{14}^{II} \\ c_{16}^{II} \\ -- \\ c_1^{II} \\ c_3^{II} \\ c_5^{II} \\ c_7^{II} \\ c_9^{II} \\ c_{11}^{II} \\ c_{13}^{II} \\ c_{15}^{II} \\ c_{17}^{II} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_9 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B}_9 \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_9 & \boldsymbol{J}_9 \\ \boldsymbol{J}_9 & -\boldsymbol{I}_9 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ -- \\ u_9 \\ u_{10} \\ u_{11} \\ u_{12} \\ u_{13} \\ u_{14} \\ u_{15} \\ u_{16} \\ u_{17} \end{pmatrix}, \tag{5.126}$$

where matrices $\boldsymbol{A}_9$ and $\boldsymbol{B}_9$ are, respectively, given by

$$\boldsymbol{A}_9 = \sqrt{2} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \cos\frac{\pi}{18} & \cos\frac{\pi}{6} & \cos\frac{5\pi}{18} & \cos\frac{7\pi}{18} & 0 & -\cos\frac{7\pi}{18} & -\cos\frac{5\pi}{18} & -\cos\frac{\pi}{6} & -\cos\frac{\pi}{18} \\ \cos\frac{\pi}{9} & \cos\frac{\pi}{3} & -\cos\frac{4\pi}{9} & -\cos\frac{2\pi}{9} & -1 & -\cos\frac{2\pi}{9} & -\cos\frac{4\pi}{9} & \cos\frac{\pi}{3} & \cos\frac{\pi}{9} \\ \cos\frac{\pi}{6} & 0 & -\cos\frac{\pi}{6} & -\cos\frac{\pi}{6} & 0 & \cos\frac{\pi}{6} & \cos\frac{\pi}{6} & 0 & -\cos\frac{\pi}{6} \\ \cos\frac{2\pi}{9} & -\cos\frac{\pi}{3} & -\cos\frac{\pi}{9} & \cos\frac{4\pi}{9} & 1 & \cos\frac{4\pi}{9} & -\cos\frac{\pi}{9} & -\cos\frac{\pi}{3} & \cos\frac{2\pi}{9} \\ \cos\frac{5\pi}{18} & -\cos\frac{\pi}{6} & -\cos\frac{7\pi}{18} & \cos\frac{\pi}{18} & 0 & -\cos\frac{\pi}{18} & \cos\frac{7\pi}{18} & \cos\frac{\pi}{6} & -\cos\frac{5\pi}{18} \\ \cos\frac{\pi}{3} & -1 & \cos\frac{\pi}{3} & \cos\frac{\pi}{3} & -1 & \cos\frac{\pi}{3} & \cos\frac{\pi}{3} & -1 & \cos\frac{\pi}{3} \\ \cos\frac{7\pi}{18} & -\cos\frac{\pi}{6} & \cos\frac{\pi}{18} & -\cos\frac{5\pi}{18} & 0 & \cos\frac{5\pi}{18} & -\cos\frac{\pi}{18} & \cos\frac{\pi}{6} & -\cos\frac{7\pi}{18} \\ \cos\frac{4\pi}{9} & -\cos\frac{\pi}{3} & \cos\frac{2\pi}{9} & -\cos\frac{\pi}{9} & 1 & -\cos\frac{\pi}{9} & \cos\frac{2\pi}{9} & -\cos\frac{\pi}{3} & \cos\frac{4\pi}{9} \end{pmatrix},$$

$$\boldsymbol{B}_9 = \sqrt{2}\begin{pmatrix}
\cos\frac{17\pi}{36} & \cos\frac{5\pi}{12} & \cos\frac{13\pi}{36} & \cos\frac{11\pi}{36} & \frac{\sqrt{2}}{2} & \cos\frac{7\pi}{36} & \cos\frac{5\pi}{36} & \cos\frac{\pi}{12} & \cos\frac{\pi}{36} \\
-\cos\frac{5\pi}{12} & -\frac{\sqrt{2}}{2} & -\cos\frac{\pi}{12} & -\cos\frac{\pi}{12} & -\frac{\sqrt{2}}{2} & -\cos\frac{5\pi}{12} & \cos\frac{5\pi}{12} & \frac{\sqrt{2}}{2} & \cos\frac{\pi}{12} \\
\cos\frac{13\pi}{36} & \cos\frac{\pi}{12} & \cos\frac{7\pi}{36} & \cos\frac{17\pi}{36} & -\frac{\sqrt{2}}{2} & -\cos\frac{\pi}{36} & -\cos\frac{11\pi}{36} & \cos\frac{5\pi}{12} & \cos\frac{5\pi}{36} \\
-\cos\frac{11\pi}{36} & -\cos\frac{\pi}{12} & -\cos\frac{17\pi}{36} & \cos\frac{5\pi}{36} & \frac{\sqrt{2}}{2} & -\cos\frac{13\pi}{36} & -\cos\frac{\pi}{36} & -\cos\frac{5\pi}{12} & \cos\frac{7\pi}{36} \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
-\cos\frac{7\pi}{36} & -\cos\frac{5\pi}{12} & \cos\frac{\pi}{36} & -\cos\frac{13\pi}{36} & -\frac{\sqrt{2}}{2} & \cos\frac{5\pi}{36} & \cos\frac{17\pi}{36} & -\cos\frac{\pi}{12} & \cos\frac{11\pi}{36} \\
\cos\frac{5\pi}{36} & -\cos\frac{5\pi}{12} & -\cos\frac{11\pi}{36} & \cos\frac{\pi}{36} & -\frac{\sqrt{2}}{2} & -\cos\frac{17\pi}{36} & \cos\frac{7\pi}{36} & -\cos\frac{\pi}{12} & \cos\frac{13\pi}{36} \\
-\cos\frac{\pi}{12} & \frac{\sqrt{2}}{2} & -\cos\frac{5\pi}{12} & -\cos\frac{5\pi}{12} & \frac{\sqrt{2}}{2} & -\cos\frac{\pi}{12} & \cos\frac{\pi}{12} & -\frac{\sqrt{2}}{2} & \cos\frac{5\pi}{12} \\
\cos\frac{\pi}{36} & -\cos\frac{\pi}{12} & \cos\frac{5\pi}{36} & -\cos\frac{7\pi}{36} & \frac{\sqrt{2}}{2} & -\cos\frac{11\pi}{36} & \cos\frac{13\pi}{36} & -\cos\frac{5\pi}{12} & \cos\frac{17\pi}{36}
\end{pmatrix},$$

Performing the product of the rightmost block matrix with vector $\mathbf{u}^T$ on the right-hand side of (5.126), and introducing a new vector $\mathbf{v}$ we get

$$\begin{array}{lll}
v_0 = u_0 + u_{17}, & v_1 = u_1 + u_{16}, & v_2 = u_2 + u_{15}, \\
v_3 = u_3 + u_{14}, & v_4 = u_4 + u_{13}, & v_5 = u_5 + u_{12}, \\
v_6 = u_6 + u_{11}, & v_7 = u_7 + u_{10}, & v_8 = u_8 + u_9, \\
v_9 = u_8 - u_9, & v_{10} = u_7 - u_{10}, & v_{11} = u_6 - u_{11}, \\
v_{12} = u_5 - u_{12}, & v_{13} = u_4 - u_{13}, & v_{14} = u_3 - u_{14}, \\
v_{15} = u_2 - u_{15}, & v_{16} = u_1 - u_{16}, & v_{17} = u_0 - u_{17}.
\end{array} \tag{5.127}$$

Equation (5.127) requires 18 additions. Let us investigate the matrix $\boldsymbol{A}_9$. Considering the central (5th) column of $\boldsymbol{A}_9$ only, with respect to (5.126) the product of $\boldsymbol{A}_9$ with the first half of vector $\mathbf{v}^T$ given by (5.127), i.e., $\boldsymbol{A}_9\ [v_0,\ v_1,\ v_2,\ v_3,\ v_4,\ v_5,\ v_6,\ v_7,\ v_8]^T$ results in values $\pm\sqrt{2}\ v_4$ for the even-indexed transform coefficients c_0^{II}, c_4^{II}, c_8^{II}, c_{12}^{II}, c_{16}^{II}. On the other hand, if we perform the scalar product of the first row of $\boldsymbol{A}_9$ with vector $[v_0,\ v_1,\ v_2,\ v_3,\ v_4,\ v_5,\ v_6,\ v_7,\ v_8]^T$ we obtain directly the transform coefficient c_0^{II} as

$$c_0^{II} = \sqrt{2}\ [(v_0 + v_8) + (v_1 + v_7) + (v_2 + v_6) + (v_3 + v_5)] + \sqrt{2}\ v_4. \tag{5.128}$$

Elimination of the first row and of central (5th) column from $\boldsymbol{A}_9$ leads to an 8×8 square matrix whose basis vectors (rows) are again symmetric and anti-symmetric. Consequently, the resulting reordered 8×8 matrix can be factored into the sparse block matrix factorization (5.118) as

$$\begin{pmatrix} c_4^{II} \\ c_8^{II} \\ c_{12}^{II} \\ c_{16}^{II} \\ -- \\ c_2^{II} \\ c_6^{II} \\ c_{10}^{II} \\ c_{14}^{II} \end{pmatrix} = \begin{pmatrix} -\sqrt{2}\, v_4 \\ \sqrt{2}\, v_4 \\ -\sqrt{2}\, v_4 \\ \sqrt{2}\, v_4 \\ -- \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} \boldsymbol{E}_4 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{F}_4 \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_4 & \boldsymbol{J}_4 \\ \boldsymbol{J}_4 & -\boldsymbol{I}_4 \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ -- \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{pmatrix}, \tag{5.129}$$

where matrices $\boldsymbol{E}_4$ and $\boldsymbol{F}_4$ are, respectively, given by

$$\boldsymbol{E}_4 = \begin{pmatrix} \sqrt{2}\cos\frac{\pi}{9} & \frac{\sqrt{2}}{2} & -\sqrt{2}\cos\frac{4\pi}{9} & -\sqrt{2}\cos\frac{2\pi}{9} \\ \sqrt{2}\cos\frac{2\pi}{9} & -\frac{\sqrt{2}}{2} & -\sqrt{2}\cos\frac{\pi}{9} & \sqrt{2}\cos\frac{4\pi}{9} \\ \frac{\sqrt{2}}{2} & -\sqrt{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \sqrt{2}\cos\frac{4\pi}{9} & -\frac{\sqrt{2}}{2} & \sqrt{2}\cos\frac{2\pi}{9} & -\sqrt{2}\cos\frac{\pi}{9} \end{pmatrix},$$

$$\boldsymbol{F}_4 = \begin{pmatrix} \sqrt{2}\cos\frac{7\pi}{18} & \sqrt{2}\cos\frac{5\pi}{18} & \sqrt{2}\cos\frac{\pi}{6} & \sqrt{2}\cos\frac{\pi}{18} \\ -\sqrt{2}\cos\frac{\pi}{6} & -\sqrt{2}\cos\frac{\pi}{6} & 0 & \sqrt{2}\cos\frac{\pi}{6} \\ \sqrt{2}\cos\frac{\pi}{18} & -\sqrt{2}\cos\frac{7\pi}{18} & -\sqrt{2}\cos\frac{\pi}{6} & \sqrt{2}\,cos\frac{5\pi}{18} \\ -\sqrt{2}\cos\frac{5\pi}{18} & \sqrt{2}\cos\frac{\pi}{18} & -\sqrt{2}\cos\frac{\pi}{6} & \sqrt{2}\cos\frac{7\pi}{18} \end{pmatrix}.$$

Performing the product of the rightmost block matrix with vector $\mathbf{v}^T$ on the right-hand side of (5.129), and introducing a new vector $\mathbf{w}$ we get

$$\begin{aligned} &w_0 = v_0 + v_8, \quad w_1 = v_1 + v_7, \quad w_2 = v_2 + v_6, \quad w_3 = v_3 + v_5, \\ &w_4 = v_4, \\ &w_5 = v_3 - v_5, \quad w_6 = v_2 - v_6, \quad w_7 = v_1 - v_7, \quad w_8 = v_0 - v_8, \end{aligned} \tag{5.130}$$

Equation (5.130) requires eight additions. Now, it remains for us to evaluate the computation of even-indexed transform coefficients $\{c_{2k}^{II}\}$, $k = 0, 1, \ldots, 8$, and to evaluate the products of matrix $\boldsymbol{E}_4$ with the first half of vector $\mathbf{w}^T$ (including the component w_4) given by (5.130), and of matrix $\boldsymbol{F}_4$ with the second half of vector $\mathbf{w}^T$ given by (5.130).

At first, consider the matrix-vector product $\boldsymbol{E}_4\ [w_0,\ w_1,\ w_2,\ w_3]^T$. Taking into account (5.128) for the transform coefficient c_0^{II}, and using the trigonometric identity

$$\sqrt{2}\cos\frac{4\pi}{9} = \sqrt{2}\left(\cos\frac{\pi}{9} - \cos\frac{2\pi}{9}\right),$$

after some algebraic manipulations for the even-indexed transform coefficients $c_0^{II},\ c_4^{II},\ c_8^{II},\ c_{12}^{II},\ c_{16}^{II}$, we get

$$\begin{aligned}
c_0^{II} &= 2\frac{\sqrt{2}}{2}(w_0 + w_2 + w_3) + \sqrt{2}\,w_1 + \sqrt{2}\,w_4,\\
c_4^{II} &= \sqrt{2}\,\cos\frac{\pi}{9}(w_0 - w_2) + \sqrt{2}\,\cos\frac{2\pi}{9}(w_2 - w_3) + \frac{\sqrt{2}}{2}w_1 - \sqrt{2}\,w_4,\\
c_8^{II} &= \sqrt{2}\,\cos\frac{2\pi}{9}(w_0 - w_3) - \sqrt{2}\,\cos\frac{\pi}{9}(w_2 - w_3) - \frac{\sqrt{2}}{2}w_1 + \sqrt{2}\,w_4,\\
c_{12}^{II} &= \frac{\sqrt{2}}{2}(w_0 + w_2 + w_3) - \sqrt{2}\,w_1 - \sqrt{2}\,w_4,\\
c_{16}^{II} &= \sqrt{2}\,\cos\frac{\pi}{9}(w_0 - w_3) - \sqrt{2}\,\cos\frac{2\pi}{9}(w_0 - w_2) - \frac{\sqrt{2}}{2}w_1 + \sqrt{2}\,w_4.
\end{aligned} \tag{5.131}$$

Noting that $(w_0 - w_3) = (w_0 - w_2) + (w_2 - w_3)$, we can generate an efficient computation of transform coefficients $c_0^{II},\ c_4^{II},\ c_8^{II},\ c_{12}^{II},\ c_{16}^{II}$, in the form of the following linear code:

$$\begin{aligned}
m_1 &= \tfrac{\sqrt{2}}{2}(w_0 + w_2 + w_3),\\
m_2 &= \tfrac{\sqrt{2}}{2}w_1,\\
m_3 &= \sqrt{2}\,w_4,\\
e &= m_2 - m_3 \qquad\qquad f = 2m_2 + m_3,\\
m_4 &= \sqrt{2}\cos\tfrac{\pi}{9}(w_0 - w_2),\\
m_5 &= \sqrt{2}\cos\tfrac{2\pi}{9}(w_2 - w_3),\\
m_6 &= -\sqrt{2}\cos\tfrac{4\pi}{9}(w_0 - w_3),
\end{aligned}$$

$$
\begin{aligned}
c_0^{II} &= 2\,m_1 + f,\\
c_4^{II} &= m_4 + m_5 + e,\\
c_8^{II} &= m_6 + m_4 - e,\\
c_{12}^{II} &= m_1 - f,\\
c_{16}^{II} &= m_5 - m_6 - e.
\end{aligned}
\tag{5.132}
$$

Equation (5.132) requires 6 multiplications, 15 additions, and 2 shifts (multiplications by 2).

Now, consider the matrix-vector product $\boldsymbol{F}_4\ [w_5,\ w_6,\ w_7,\ w_8]^T$. Using the trigonometric identity (note that $\cos\frac{\pi}{18} = \sin\frac{4\pi}{9}$, $\cos\frac{5\pi}{18} = \sin\frac{2\pi}{9}$, $\cos\frac{7\pi}{18} = \sin\frac{\pi}{9}$)

$$
\sqrt{2}\cos\frac{7\pi}{18} = \sqrt{2}\left(\cos\frac{\pi}{18} - \cos\frac{5\pi}{18}\right),
$$

after some algebraic manipulations for the even-indexed transform coefficients $c_2^{II},\ c_6^{II},\ c_{10}^{II},\ c_{14}^{II}$, we get

$$
\begin{aligned}
c_2^{II} &= \sqrt{2}\,\cos\frac{\pi}{18}\,(w_5 + w_8) - \sqrt{2}\,\cos\frac{5\pi}{18}\,(w_5 - w_6) + \sqrt{2}\,\cos\frac{\pi}{6}\,w_7,\\
c_6^{II} &= \sqrt{2}\,\cos\frac{\pi}{6}\,(-w_5 - w_6 + w_8),\\
c_{10}^{II} &= \sqrt{2}\,\cos\frac{\pi}{18}\,(w_5 - w_6) + \sqrt{2}\,\cos\frac{5\pi}{18}\,(w_6 + w_8) - \sqrt{2}\,\cos\frac{\pi}{6}\,w_7,\\
c_{14}^{II} &= \sqrt{2}\,\cos\frac{\pi}{18}\,(w_6 + w_8) - \sqrt{2}\,\cos\frac{5\pi}{18}\,(w_5 + w_8) - \sqrt{2}\,\cos\frac{\pi}{6}\,w_7.
\end{aligned}
\tag{5.133}
$$

Similarly, noting that $(w_5 + w_8) = (w_5 - w_6) + (w_6 + w_8)$, we can generate an efficient computation of transform coefficients $c_2^{II},\ c_6^{II},\ c_{10}^{II},\ c_{14}^{II}$, in the form of the following linear code:

$$
\begin{aligned}
m_1 &= \sqrt{2}\cos\frac{\pi}{6}\,w_7,\\
m_2 &= \sqrt{2}\cos\frac{\pi}{6}\,(-w_5 - w_6 + w_8),\\
m_3 &= \sqrt{2}\cos\frac{\pi}{18}\,(w_5 - w_6),
\end{aligned}
$$

$$m_4 = \sqrt{2}\cos\frac{5\pi}{18}\,(w_6 + w_8),$$

$$m_5 = \sqrt{2}\cos\frac{7\pi}{18}\,(w_5 + w_8),$$

$$\begin{aligned} c_2^{II} &= m_5 + m_4 + m_1, \\ c_6^{II} &= m_2, \\ c_{10}^{II} &= m_3 + m_4 - m_1, \\ c_{14}^{II} &= m_5 - m_3 - m_1. \end{aligned} \tag{5.134}$$

Equation (5.134) requires 5 multiplications and 11 additions. Thus, the evaluation of block matrix-vector product (5.129) for the computation of even-indexed transform coefficients $\{c_{2k}^{II}\}$, $k = 0, 1, \ldots, 8$, according to (5.130), (5.132), and (5.134) requires totally 11 multiplications, 34 additions, and 2 shifts.

Finally, for the computation of odd-indexed transform coefficients $\{c_{2k+1}^{II}\}$, $k = 0, 1, \ldots, 8$, consider the product of matrix $\boldsymbol{B}_9$ with the second half of vector $\mathbf{v}^T$ given by (5.127), i.e., $\boldsymbol{B}_9\,[v_9,\ v_{10},\ v_{11},\ v_{12},\ v_{13},\ v_{14},\ v_{15},\ v_{16},\ v_{17}]^T$. At first, we partition the matrix $\boldsymbol{B}_9$ into two parts. The first part consists of the 2nd, 5th, and 8th rows of $\boldsymbol{B}_9$ corresponding, respectively, to transform coefficients c_3^{II}, c_9^{II}, c_{15}^{II} in the matrix-vector product. Performing sequentially scalar products of the 2nd, 5th and 8th rows of $\boldsymbol{B}_9$ with the vector $[v_9,\ v_{10},\ v_{11},\ v_{12},\ v_{13},\ v_{14}\ v_{15},\ v_{16},\ v_{17}]^T$, and using the trigonometric identity (5.122), after some algebraic manipulations we get

$$\begin{aligned} c_3^{II} &= \sqrt{2}\,\cos\frac{\pi}{12}\,(-v_9 - v_{11} - v_{12} - v_{14} + v_{15} + v_{17}) \\ &\quad +(v_9 - v_{10} - v_{13} + v_{14} - v_{15} + v_{16}), \\ c_9^{II} &= (v_9 + v_{10} - v_{11} - v_{12} + v_{13} + v_{14} - v_{15} - v_{16} + v_{17}), \\ c_{15}^{II} &= \sqrt{2}\,\cos\frac{\pi}{12}\,(-v_9 - v_{11} - v_{12} - v_{14} + v_{15} + v_{17}) \\ &\quad +(v_{10} + v_{11} + v_{12} + v_{13} - v_{16} - v_{17}). \end{aligned} \tag{5.135}$$

From (5.134) we can generate an efficient computation of transform coefficients c_3^{II}, c_9^{II}, c_{15}^{II}, in the form of the following linear code:

$$\begin{aligned} &e = v_{10} + v_{13} - v_{16}, \qquad f = v_9 + v_{14} - v_{15},\ g = v_{11} + v_{12} - v_{17}, \\ &m_1 = -\sqrt{2}\,\cos\tfrac{\pi}{12}\,(f + g), \end{aligned}$$

$$
\begin{aligned}
c_3^{II} &= m_1 - e + f, \\
c_9^{II} &= e + f - g, \\
c_{15}^{II} &= m_1 + e + g.
\end{aligned} \tag{5.136}
$$

Equation (5.136) requires 1 multiplication and 13 additions. Further, noting that the 2nd, 5th, and 8th columns of $\boldsymbol{B}_9$ (corresponding to the components $v_{10},\ v_{13},\ v_{16}$ of vector $\mathbf{v}^T$) is composed from elements $\pm\sqrt{2}\ \cos\frac{5\pi}{12}$, ± 1, and $\pm\sqrt{2}\ \cos\frac{\pi}{12}$, again using the trigonometric identity (5.122), remaining matrix-vector product can be represented as

$$
\begin{pmatrix} c_1^{II} \\ c_5^{II} \\ c_7^{II} \\ c_{11}^{II} \\ c_{13}^{II} \\ c_{17}^{II} \end{pmatrix} =
\begin{pmatrix}
\sqrt{2}\ \cos\frac{\pi}{12}\ (v_{10} + v_{16}) - (v_{10} - v_{13}) \\
\sqrt{2}\ \cos\frac{\pi}{12}\ (v_{10} + v_{16}) - (v_{13} + v_{16}) \\
-\sqrt{2}\ \cos\frac{\pi}{12}\ (v_{10} + v_{16}) + (v_{13} + v_{16}) \\
-\sqrt{2}\ \cos\frac{\pi}{12}\ (v_{10} + v_{16}) + (v_{10} - v_{13}) \\
-\sqrt{2}\ \cos\frac{\pi}{12}\ (v_{10} + v_{16}) + (v_{10} - v_{13}) \\
-\sqrt{2}\ \cos\frac{\pi}{12}\ (v_{10} + v_{16}) + (v_{13} + v_{16})
\end{pmatrix}
$$

$$
+\sqrt{2}
\begin{pmatrix}
\cos\frac{17\pi}{36} & \cos\frac{13\pi}{36} & \cos\frac{11\pi}{36} & \cos\frac{7\pi}{36} & \cos\frac{5\pi}{36} & \cos\frac{\pi}{36} \\
\cos\frac{13\pi}{36} & \cos\frac{7\pi}{36} & \cos\frac{17\pi}{36} & -\cos\frac{\pi}{36} & -\cos\frac{11\pi}{36} & \cos\frac{5\pi}{36} \\
-\cos\frac{11\pi}{36} & -\cos\frac{17\pi}{36} & \cos\frac{5\pi}{36} & -\cos\frac{13\pi}{36} & -\cos\frac{\pi}{36} & \cos\frac{7\pi}{36} \\
-\cos\frac{7\pi}{36} & \cos\frac{\pi}{36} & -\cos\frac{13\pi}{36} & \cos\frac{5\pi}{36} & \cos\frac{17\pi}{36} & \cos\frac{11\pi}{36} \\
\cos\frac{5\pi}{36} & -\cos\frac{11\pi}{36} & \cos\frac{\pi}{36} & -\cos\frac{17\pi}{36} & \cos\frac{7\pi}{36} & \cos\frac{13\pi}{36} \\
\cos\frac{\pi}{36} & \cos\frac{5\pi}{36} & -\cos\frac{7\pi}{36} & -\cos\frac{11\pi}{36} & \cos\frac{13\pi}{36} & \cos\frac{17\pi}{36}
\end{pmatrix}
\begin{pmatrix} v_9 \\ v_{11} \\ v_{12} \\ v_{14} \\ v_{15} \\ v_{17} \end{pmatrix}. \tag{5.137}
$$

Substituting the following trigonometric identities

$$
\begin{aligned}
\sqrt{2}\ \cos\frac{\pi}{36} &= \cos\tfrac{2\pi}{9} + \cos\tfrac{5\pi}{18}, & \sqrt{2}\ \cos\tfrac{17\pi}{36} &= \cos\frac{2\pi}{9} - \cos\frac{5\pi}{18}, \\
\sqrt{2}\ \cos\frac{5\pi}{36} &= \cos\tfrac{\pi}{9} + \cos\tfrac{7\pi}{18}, & \sqrt{2}\ \cos\tfrac{13\pi}{36} &= \cos\frac{\pi}{9} - \cos\frac{7\pi}{18}, \\
\sqrt{2}\ \cos\frac{7\pi}{36} &= \cos\tfrac{4\pi}{9} + \cos\tfrac{\pi}{18}, & \sqrt{2}\ \cos\tfrac{11\pi}{36} &= \cos\frac{4\pi}{9} - \cos\frac{\pi}{18},
\end{aligned}
$$

into the matrix on right-hand side of (5.137), performing the matrix-vector product, and then using the trigonometric identities

$$\cos\frac{4\pi}{9} = \cos\frac{\pi}{9} - \cos\frac{2\pi}{9}, \qquad \cos\frac{7\pi}{18} = \cos\frac{\pi}{18} - \cos\frac{5\pi}{18},$$

after rigorous algebraic manipulations, for the odd-indexed transform coefficients c_1^{II}, c_5^{II}, c_7^{II}, c_{11}^{II}, c_{13}^{II}, c_{17}^{II}, we finally obtain

$$\begin{aligned}
c_1^{II} &= \sqrt{2}\,\cos\frac{\pi}{12}\,(v_{10}+v_{16}) - (v_{10}-v_{13}) \\
&\quad + \cos\frac{\pi}{9}\,(v_{11}-v_{12}+v_{14}+v_{15}) + \cos\frac{2\pi}{9}\,(v_9+v_{12}-v_{14}+v_{17}) \\
&\quad - \cos\frac{\pi}{18}\,(v_{11}-v_{12}-v_{14}-v_{15}) - \cos\frac{5\pi}{18}\,(v_9-v_{11}+v_{15}-v_{17})
\end{aligned}$$

$$\begin{aligned}
c_5^{II} &= \sqrt{2}\,\cos\frac{\pi}{12}\,(v_{10}+v_{16}) - (v_{13}+v_{16}) \\
&\quad + \cos\frac{\pi}{9}\,(v_9+v_{11}+v_{15}+v_{17}) - \cos\frac{2\pi}{9}\,(v_{11}-v_{12}+v_{14}+v_{15}) \\
&\quad - \cos\frac{\pi}{18}\,(v_9-v_{11}+v_{15}-v_{17}) + \cos\frac{5\pi}{18}\,(v_9-v_{12}-v_{14}-v_{17})
\end{aligned}$$

$$\begin{aligned}
c_7^{II} &= -\sqrt{2}\,\cos\frac{\pi}{12}\,(v_{10}+v_{16}) + (v_{13}+v_{16}) \\
&\quad + \cos\frac{\pi}{9}\,(v_9+v_{12}-v_{14}+v_{17}) - \cos\frac{2\pi}{9}\,(v_9+v_{11}+v_{15}+v_{17}) \\
&\quad - \cos\frac{\pi}{18}\,(v_9-v_{12}-v_{14}-v_{17}) + \cos\frac{5\pi}{18}\,(v_{11}-v_{12}-v_{14}-v_{15})
\end{aligned}$$

$$\begin{aligned}
c_{11}^{II} &= -\sqrt{2}\,\cos\frac{\pi}{12}\,(v_{10}+v_{16}) + (v_{10}-v_{13}) \\
&\quad - \cos\frac{\pi}{9}\,(v_9+v_{12}-v_{14}+v_{17}) + \cos\frac{2\pi}{9}\,(v_9+v_{11}+v_{15}+v_{17}) \\
&\quad - \cos\frac{\pi}{18}\,(v_9-v_{12}-v_{14}-v_{17}) + \cos\frac{5\pi}{18}\,(v_{11}-v_{12}-v_{14}-v_{15})
\end{aligned}$$

$$\begin{aligned}
c_{13}^{II} &= -\sqrt{2}\,\cos\frac{\pi}{12}\,(v_{10}+v_{16}) + (v_{10}-v_{13}) \\
&\quad + \cos\frac{\pi}{9}\,(v_9+v_{11}+v_{15}+v_{17}) - \cos\frac{2\pi}{9}\,(v_{11}-v_{12}+v_{14}+v_{15}) \\
&\quad + \cos\frac{\pi}{18}\,(v_9-v_{11}+v_{15}-v_{17}) - \cos\frac{5\pi}{18}\,(v_9-v_{12}-v_{14}-v_{17})
\end{aligned}$$

$$
\begin{aligned}
c_{17}^{II} = &-\sqrt{2}\,\cos\frac{\pi}{12}\,(v_{10}+v_{16}) + (v_{13}+v_{16}) \\
&+\cos\frac{\pi}{9}\,(v_{11}-v_{12}+v_{14}+v_{15}) + \cos\frac{2\pi}{9}\,(v_9+v_{12}-v_{14}+v_{17}) \\
&+\cos\frac{\pi}{18}\,(v_{11}-v_{12}-v_{14}-v_{15}) + \cos\frac{5\pi}{18}\,(v_9-v_{11}+v_{15}-v_{17}).
\end{aligned}
\tag{5.138}
$$

Noting that

$$
\begin{aligned}
(v_9+v_{11}+v_{15}+v_{17}) &= (v_{11}-v_{12}+v_{14}+v_{15}) + (v_9+v_{12}-v_{14}+v_{17}), \\
&= [(v_{11}-v_{12}) + (v_{14}+v_{15})] + [(v_9+v_{12}) - (v_{14}-v_{17})],
\end{aligned}
$$

$$
\begin{aligned}
(v_9-v_{12}-v_{14}-v_{17}) &= (v_{11}-v_{12}-v_{14}-v_{15}) + (v_9-v_{11}+v_{15}-v_{17}), \\
&= [(v_{11}-v_{12}) - (v_{14}+v_{15})] + [(v_9-v_{12}) + (v_{15}-v_{17})],
\end{aligned}
$$

from (5.138) we can generate an efficient computation of transform coefficients $c_1^{II}, c_5^{II}, c_7^{II}, c_{11}^{II}, c_{13}^{II}, c_{17}^{II}$, in the form of the following linear code:

$$
\begin{aligned}
&m_2 = \sqrt{2}\,\cos\tfrac{\pi}{12}\,(v_{10}+v_{16}), \\
&d = m_2 - (v_{10}-v_{13}), \qquad e = m_2 - (v_{13}+v_{16}),
\end{aligned}
$$

$$
\begin{aligned}
&p = v_{11}-v_{12}, \qquad q = v_{14}+v_{15}, \\
&r = v_9+v_{12}, \qquad s = v_{14}-v_{17}, \\
&a = p+q, \qquad b = r-s, \\
&m_3 = \cos\tfrac{\pi}{9}\,a, \\
&m_4 = \cos\tfrac{2\pi}{9}\,b, \\
&m_5 = \cos\tfrac{4\pi}{9}\,(a+b), \\
&f = m_3+m_4, \qquad g = m_5+m_4, \qquad h = m_5-m_3,
\end{aligned}
$$

$$
\begin{aligned}
&r = v_9-v_{12}, \qquad s = v_{15}-v_{17}, \\
&a = p-q, \qquad b = r+s, \\
&m_6 = \;\;\cos\tfrac{\pi}{18}\,a, \\
&m_7 = \;\;\cos\tfrac{5\pi}{18}\,b, \\
&m_8 = -\cos\tfrac{7\pi}{18}\,(a+b), \\
&i = m_6+m_7, \qquad j = m_8+m_6, \qquad l = m_8-m_7,
\end{aligned}
$$

$$
\begin{aligned}
c_1^{II} &= \quad d + f - i, \\
c_5^{II} &= \quad e + g + j, \\
c_7^{II} &= -e + h + l, \\
c_{11}^{II} &= -d - h + l, \\
c_{13}^{II} &= -d + g - j, \\
c_{17}^{II} &= -e + f + i.
\end{aligned} \tag{5.139}
$$

Equation (5.139) requires 7 multiplications and 35 additions. Thus, the evaluation of block matrix-vector product $\boldsymbol{B}_9\ [v_9,\ v_{10},\ v_{11},\ v_{12},\ v_{13},\ v_{14},\ v_{15},\ v_{16},\ v_{17}]^T$ for the computation of odd-indexed transform coefficients $\{c_{2k+1}^{II}\}$, $k = 0, 1, \ldots, 8$, according to (5.136) and (5.139) requires 8 multiplications and 48 additions. Equations (5.127), (5.130), (5.132), (5.134), (5.136), and (5.139) define the fast computational structure for the computation of 18-point SDCT-II which requires totally 19 multiplications, 100 additions, and 2 shifts.

Equations (5.116), (5.117), (5.127), (5.130), (5.132), (5.134), (5.136), and (5.139) define the identical fast computational structure both for the forward and backward MDCT computation. $\{y_n\}$ given by (5.36) or (5.47) in (5.117) requires 18 additions, (5.117) requires 18 multiplications, and (5.116) for the computation of transform coefficients $\{c_k^{IV}\}$ from $\{c_k^{II}\}$ according to (5.108) requires 17 recursive additions and 1 shift. Thus, for the long audio block the forward/backward MDCT in MP3 requires totally 37 multiplications, 135/117 additions, and 3 shifts. The time domain aliased data sequence $\{\hat{x}_n\}$ is recovered after the backward MDCT from $\{y_n\}$ according to (5.48).

5.4.5 *MDCT Efficient Implementations Based on the Evenly Stacked MDCT*

Another class of developed efficient MDCT implementations in MP3 [26, 39, 64] is based on a relation between the MDCT block transform given by (5.9) and the corresponding evenly stacked MDCT block transform [137]. Note that the forward and backward MDCT block transforms given by (5.9) and (5.10), respectively, represent the oddly stacked MDCT block transforms. Although the oddly stacked MDCT [7] and evenly stacked MDCT [137] are quite different TDAC filter banks, there exists a close relation between them, and consequently, the efficient computation of oddly stacked MDCT can be realized via the evenly stacked MDCT and vice versa only by simple pre- and post-processing of input and output data sequences [26]. This fact allows to handle evenly and oddly stacked MDCT block transforms in a unified framework.

Let $\{x_n\}$, $n = 0, 1, \ldots, N-1$ be an input data sequence. The forward and backward evenly stacked MDCT block transforms are, respectively, defined as [26, 137]

$$\overset{E}{c}_k = \sqrt{\frac{1}{N}} \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right) k\right], \quad k = 0, 1, \ldots, \frac{N}{2}-1, \quad \overset{E}{c}_{\frac{N}{2}} = 0, \tag{5.140}$$

$$\hat{x}_n^E = \sqrt{\frac{1}{N}} \sum_{k=0}^{\frac{N}{2}-1} \epsilon_k \overset{E}{c}_k \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right) k\right], \quad n = 0, 1, \ldots, N-1, \tag{5.141}$$

where $\epsilon_0 = 1$ for $k = 0$, $\epsilon_k = 2$ for $k = 1, 2, \ldots, \frac{N}{2}-1$, and $\{\hat{x}_n^E\}$ is the time domain aliased data sequence. Using trigonometric identity $\cos(\alpha+\beta) = 2\cos\alpha\ \cos\beta - \cos(\alpha-\beta)$ to the oddly stacked MDCT transform kernel given by (5.9), setting $\alpha = \frac{\pi}{N}(2n+1+\frac{N}{2})k$ and $\beta = \frac{\pi}{2N}(2n+1+\frac{N}{2})$, we get the relation between the unnormalized oddly and evenly stacked MDCT block transforms defined as [26]

$$c_k = \sum_{n=0}^{N-1} \left(2\, x_n \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)\right]\right) \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right) k\right] - c_{k-1},$$
$$k = 0, 1, \ldots, \frac{N}{2}-1, \tag{5.142}$$

where the transform coefficients $\{c_k\}$ and $\{\overset{E}{c}_k\}$ satisfy the following conditions:

$$\begin{aligned} 2\, c_0 &= \overset{E}{c}_0, \quad \text{if} \quad k = 0, \\ c_k + c_{k-1} &= \overset{E}{c}_k, \quad \text{if} \quad k > 0. \end{aligned} \tag{5.143}$$

Thus, the N-point oddly stacked MDCT is converted to the evenly stacked MDCT of the same size at the cost of additional N pre-multiplications and $N-1$ recursive post-additions. This relation is quite similar to that of between the DCT-IV and the DCT-II given by (5.106) and (5.108). Hence, an efficient implementation of the oddly stacked MDCT in MP3 relies on a suitable fast algorithm for the evenly stacked MDCT computation valid for N being an even integer. Some fast algorithms for the evenly stacked MDCT computation are presented in [118].

Now consider the efficient computation of the forward evenly stacked MDCT given by (5.140). Applying a permutation defined as [39]

$$y_n = \begin{cases} x_{n-\frac{N}{4}}, & n = \frac{N}{4}, \frac{N}{4}+1, \ldots, N-1, \\ x_{\frac{3N}{4}+n}, & n = 0, 1, \ldots, \frac{N}{4}-1, \end{cases} \tag{5.144}$$

and using the symmetry property of the cosine transform kernel, (5.140) can be rewritten as

$$c_k^E = \sum_{n=0}^{N-1} y_n \cos\left[\frac{\pi(2n+1)k}{N}\right] = \sum_{n=0}^{\frac{N}{2}-1} (y_n + y_{N-1-n}) \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right],$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{5.145}$$

The transform kernel in (5.145) is recognized as an $\frac{N}{2}$-point DCT-II. Finally, combining (5.144) and (5.145) we get

$$c_k^E = \sum_{n=0}^{\frac{N}{2}-1} y_n \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{5.146}$$

where

$$y_n = \begin{cases} x_{\frac{3N}{4}+n} + x_{\frac{3N}{4}-1-n}, & n = 0, 1, \ldots, \frac{N}{4} - 1, \\ x_{n-\frac{N}{4}} + x_{\frac{3N}{4}-1-n}, & n = \frac{N}{4}, \frac{N}{4}+1, \ldots, \frac{N}{2} - 1. \end{cases} \tag{5.147}$$

Equation (5.146) implies that the N-point forward evenly stacked MDCT given by (5.140) is converted to the $\frac{N}{2}$-point DCT-II of the data sequence $\{y_n\}$ defined by (5.147). Because we need a suitable even-length fast DCT-II algorithm, further development is simply based on (5.109)–(5.111) by substituting $M = \frac{N}{2}$ and $u_m = y_m$ for $m = n$.

Since the transform kernel of the evenly stacked MDCT is not symmetric, efficient computing the backward evenly stacked MDCT given by (5.141) can be realized by reversing a signal flow graph for the forward evenly stacked MDCT computation and performing inverse operations. Note that if backward evenly stacked MDCT computation is only required, then MDCT coefficients in (5.141) should be scaled by ϵ_k.

5.4.5.1 MDCT Efficient Implementation Based on the Evenly Stacked MDCT [39, 64]

Referring to (5.146) and (5.147), substituting $M = \frac{N}{2}$ and $m = n$ into Eqs. (5.109) and (5.110) we have

$$c_{2k}^{II} = \sum_{n=0}^{\frac{N}{4}-1} \left(u_n + u_{\frac{N}{2}-1-n}\right) \cos\left[\frac{\pi(2n+1)k}{2(N/4)}\right],$$

$$c^{II}_{2k+1} = \sum_{n=0}^{\frac{N}{4}-1} \left[2 \left(u_n - u_{\frac{N}{2}-1-n} \right) \cos \frac{\pi(2n+1)}{N} \right] \cos \left[\frac{\pi(2n+1)k}{2(N/4)} \right] - c^{II}_{2k-1},$$

$$k = 0, 1, \ldots, \frac{N}{4} - 1, \tag{5.148}$$

and the evenly stacked MDCT transform coefficients $\{c^E_k\}$ are obtained from (5.111) by setting $c^E_k = c^{II}_k$ and $\hat{c}^E_k = \hat{c}^{II}_k$. Note that (5.113) and (5.148) are identical. However, the data sequence $\{u_n\}$ defined as

$$u_n = \begin{cases} -2 \cos \frac{\pi(2n+1)}{2N} \left(x_{\frac{3N}{4}+n} + x_{\frac{3N}{4}-1-n} \right), & n = 0, 1, \ldots, \frac{N}{4} - 1, \\ 2 \cos \frac{\pi(2n+1)}{2N} \left(x_{n-\frac{N}{4}} - x_{\frac{3N}{4}-1-n} \right), & n = \frac{N}{4}, \frac{N}{4}+1, \ldots, \frac{N}{2} - 1, \end{cases} \tag{5.149}$$

is obtained by incorporating the terms $2x_n \cos\left[\frac{\pi}{2N}(2n + 1 + \frac{N}{2}\right]$ from (5.142) for n equal to $\frac{3N}{4} + n$, $\frac{3N}{4} - 1 - n$ and $n - \frac{N}{4}$ into the permutation (5.147). The final oddly stacked MDCT coefficients $\{c_k\}$ are obtained from (5.143). The corresponding regular signal flow graph for the forward MDCT computation in MP3 based on the evenly stacked MDCT for $N = 12$ is shown in Fig. 5.16. The backward MDCT computation can be realized simply by reversing the signal flow graph for the forward MDCT computation and performing inverse operations. The optimized efficient 3/9-point DCT-II and DCT-III modules are presented in Appendix D.4.

The total computational complexity of the forward MDCT is $3\frac{N}{4}$ multiplications and $7\frac{N}{4} - 2$ additions ($5\frac{N}{4} - 2$ additions for the backward MDCT) plus the arithmetic complexity of two identical $\frac{N}{4}$-point DCT-II/DCT-III modules. Thus, for the short block the forward/backward MDCT computation in MP3 requires 11 multiplications, 27/21 additions, and 4 shifts, while for the long block the forward/backward MDCT computation requires 43 multiplications, 129/111 additions, and 6 shifts.

The efficient MDCT implementation based on the evenly stacked MDCT can be further improved in terms of the structural simplicity. Comparing the data sequence $\{u_n\}$ given by (5.149) with that of defined in (5.112) and (5.113) for the DCT-IV/DCT-II-based efficient MDCT implementation, one can observe that they generate the same permuted data sequence (see Figs. 5.14 and 5.16). This fact indicates that there exists a close relation between those two different efficient MDCT implementations. Therefore, the signal flow graph identical for the forward/backward MDCT efficient implementation shown in Fig. 5.14 can be also used for the forward/backward MDCT efficient implementation based on the evenly stacked MDCT. With respect to (5.149) and Fig. 5.14, the time domain aliased data sequence $\{\hat{x}_n\}$ is recovered after the backward MDCT from $\{y_n\}$ as

$$\hat{x}_{\frac{3N}{4}+n} = \hat{x}_{\frac{3N}{4}-1-n} = -y_n, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1,$$

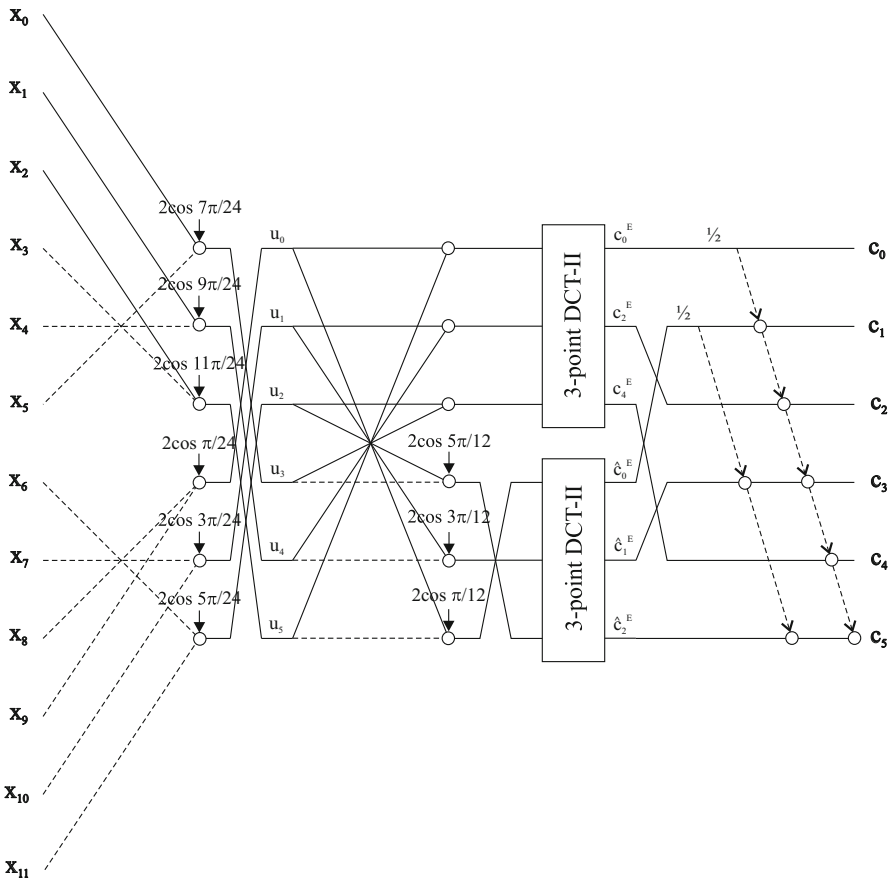

Fig. 5.16 Signal flow graph for the forward/backward MDCT computation for $N = 12$ based on the evenly stacked MDCT

$$\hat{x}_{n-\frac{N}{4}} = y_n, \quad \hat{x}_{\frac{3N}{4}-1-n} = -y_n, \quad n = \frac{N}{4}, \frac{N}{4}+1, \ldots, \frac{N}{2}-1. \tag{5.150}$$

5.4.5.2 MDCT Efficient Implementation Based on the Evenly Stacked MDCT [26]

Complete formulae of a refined fast algorithm for the evenly stacked MDCT computation are given by [26, 118]

$$c_{2k}^{E} = (-1)^k \sum_{n=0}^{\frac{N}{4}-1} \left(y_n'' + y_{\frac{N}{2}-1-n}'' \right) \cos\left[\frac{\pi(2n+1)k}{2(N/4)} \right],$$

$$c^{E}_{\frac{N}{2}-2k-1} = (-1)^{\frac{N}{4}-k} \sum_{n=0}^{\frac{N}{4}-1} \left[2\, z_n\, \cos \frac{\pi(2n+1)}{N} \right] \cos \left[\frac{\pi(2n+1)k}{2(N/4)} \right] + c^{E}_{\frac{N}{2}-2k+1},$$

$$k = 0, 1, \ldots, \frac{N}{4} - 1, \tag{5.151}$$

where

$$y_n = 2\, x_n\, \cos \left[\frac{\pi}{2N} \left(2n + 1 + \frac{N}{2} \right) \right], \qquad n = 0, 1, \ldots, N-1,$$

$$y'_n = y_n - y_{N-1-n}, \quad y''_n = y_n + y_{N-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1,$$

$$z_n = (-1)^n \left(y'_n + y'_{\frac{N}{2}-1-n} \right), \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{5.152}$$

The transform coefficients $\{c^{E}_k\}$ and $\{c^{II}_k\}$ satisfy the following conditions:

$$c^{E}_{\frac{N}{2}-1} = (-1)^{\frac{N}{4}}\, 2\, c^{II}_{\frac{N}{4}-1} \quad \text{if} \quad k = 0,$$

$$c^{E}_{\frac{N}{2}-2k-1} - c^{E}_{\frac{N}{2}-2k+1} = (-1)^{\frac{N}{4}-k}\, c^{II}_{\frac{N}{4}-k-1} \quad \text{if} \quad k > 0. \tag{5.153}$$

The final oddly stacked MDCT coefficients $\{c_k\}$ are obtained from (5.143). The corresponding regular signal flow graph for the forward MDCT computation in MP3 based on the evenly stacked MDCT for $N = 12$ is shown in Fig. 5.17. The backward MDCT computation can be realized simply by reversing the signal flow graph for the forward MDCT computation and performing inverse operations. The optimized efficient 3/9-point DCT-II and DCT-III modules are presented in Appendix D.4.

The total computational complexity of the forward MDCT is $5\frac{N}{4}$ multiplications and $9\frac{N}{4} - 2$ additions ($7\frac{N}{4} - 2$ additions for the backward MDCT) plus the arithmetic complexity of two identical $\frac{N}{4}$-point DCT-II/DCT-III modules. Thus, for the short block the forward/backward MDCT computation in MP3 requires 17 multiplications, 33/27 additions, and 4 shifts, while for the long block the forward/backward MDCT computation requires 61 multiplications, 147/129 additions, and 6 shifts.

Note 19: Twiddle factors $2\cos\left[\frac{\pi}{2N}(2n+1+\frac{N}{2}\right]$ applied to the data sequence $\{x_n\}$ can be effectively absorbed into the windowing operation in the complete analysis/synthesis MDCT (MLT) filter banks.

5.4.6 *Mixed-Radix Efficient MDCT Implementations*

Mixed-radix fast MDCT algorithms or fast MDCT algorithms for the composite lengths [28, 44, 60, 67, 68] are obtained by the combination of radix-q fast

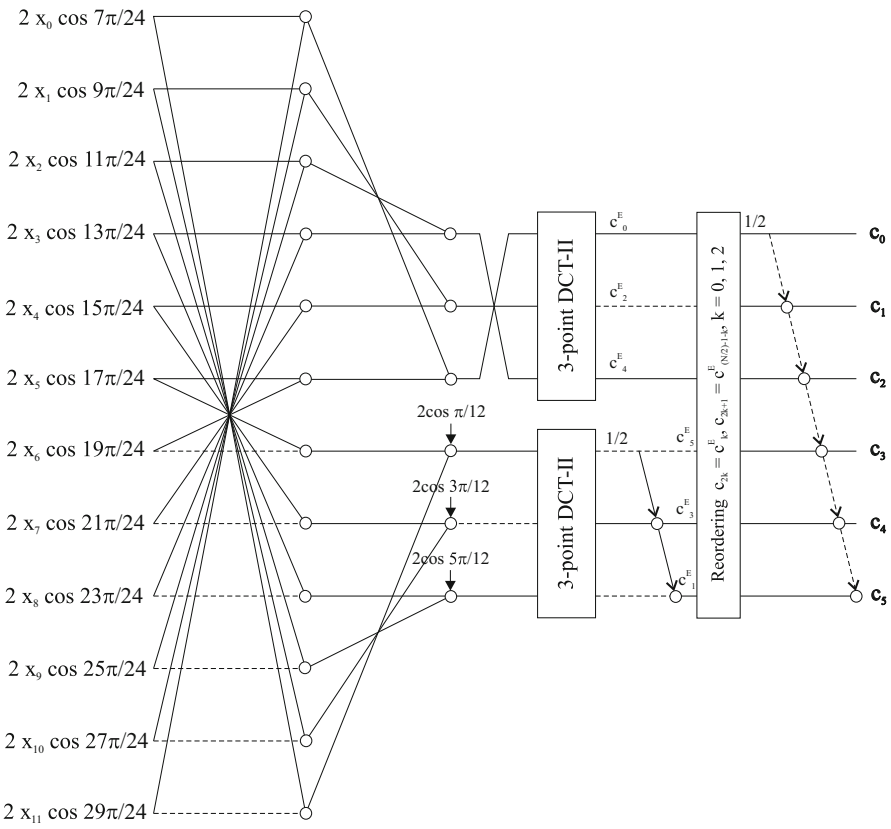

Fig. 5.17 Signal flow graph for the forward/backward MDCT computation for $N = 12$ based on the evenly stacked MDCT

algorithms (q is an odd integer) and radix-2 fast algorithms. For a particular radix number q, the radix-q fast algorithms recursively divide the entire transform computation into q shorter $\frac{N}{q}$-point transforms. The computation of higher-order transforms is obtained by the recursive reuse of lower-order transforms. A generalized mixed-radix decomposition method for the composite lengths $N = q^m \times 2^p$, $m,\ p > 0$, is described in [114]. It enables to optimize the performance of such algorithms in terms of the arithmetic complexity during the decomposition process for some special lengths [114].

Note 20: Importantly, the strictly radix-q fast MDCT algorithm cannot be constructed, since from the MDCT definition it follows that q^m is not divisible by 2. Therefore, for the MDCT only mixed-radix fast algorithms may be constructed.

Among the proposed mixed-radix fast algorithms which can be adopted for the efficient MDCT implementation in MP3 are three mixed-radix fast MDCT algorithms: the first one with $q = 3$ obtained by the decimation in frequency (DIF) method for the composite lengths $N = 3^m \times 2^p$, $m,\ p > 0$ [28, 60, 68], the second one obtained by the decimation in time (DIT) method for the composite lengths

$N = 3^m \times 2^p$, $m > 0$, $p > 1$ [67], and the third is a generalized mixed-radix MDCT algorithm for composite lengths $N = q^m \times 2^p$, $m > 0$, $p > 1$, where $q = 3,\ 5$, and 9 [44].

5.4.6.1 DIF Mixed-Radix Fast MDCT Algorithms [28, 60, 68]

Based on [60, 68] complete formulae of the improved and extended DIF mixed-radix fast MDCT algorithm [28] for the composite lengths $N = 3^m \times 2^p$, $m,\ p > 0$, are given by

$$c_{3k+1} = \sum_{n=0}^{\frac{N}{3}-1} \left[x_{\frac{N}{3}-1-n} - \left(x_{\frac{2N}{3}-1-n} - x_{N-1-n} \right) \right] \cos \left[\frac{\pi}{2(N/3)} \left(2n+1+\frac{N}{6} \right) (2k+1) \right],$$

$$\begin{aligned} c_{3k} &= a_k + b_k, \\ c_{3k+2} &= a_k - b_k, \qquad\qquad k = 0, 1, \ldots, \frac{N}{6} - 1, \end{aligned} \tag{5.154}$$

where

$$\begin{aligned} a_k = -\sum_{n=0}^{\frac{N}{3}-1} & \left[u_n \frac{1}{2} \sin \frac{\pi(2n+1)}{N} - v_n \frac{\sqrt{3}}{2} \cos \frac{\pi(2n+1)}{N} \right] \\ & \times \cos \left[\frac{\pi}{2(N/3)} \left(\frac{2N}{3} - 2n - 1 + \frac{N}{6} \right) (2k+1) \right], \end{aligned}$$

$$\begin{aligned} b_k = (-1)^k \sum_{n=0}^{\frac{N}{3}-1} & \left[u_n \frac{1}{2} \cos \frac{\pi(2n+1)}{N} + v_n \frac{\sqrt{3}}{2} \sin \frac{\pi(2n+1)}{N} \right] \\ & \times \cos \left[\frac{\pi}{2(N/3)} \left(2n+1+\frac{N}{6} \right) (2k+1) \right], \\ & k = 0, 1, \ldots, \frac{N}{6} - 1, \end{aligned} \tag{5.155}$$

and

$$u_n = 2x_n + x_{\frac{N}{3}+n} - x_{\frac{2N}{3}+n}, \qquad v_n = x_{\frac{N}{3}+n} + x_{\frac{2N}{3}+n}, \qquad n = 0, 1, \ldots, \frac{N}{3} - 1. \tag{5.156}$$

Thus, the forward N-point MDCT is obtained from the computation of three $\frac{N}{3}$-point MDCTs. The cosine transform kernel in the first sum of (5.155) is recognized as the MDCT kernel in reverse order, whereby $\cos\phi_{\frac{N}{3}-1-n,k} = (-1)^{k+1}\sin\phi_{n,k}$.

Since the short and long block sizes in MP3 are the composite lengths $N = 3^m \times 4$, $m > 0$, their DIF mixed-radix decompositions are, respectively, given by

$$\begin{aligned} N &= 12 = 3 \times 2^2 = 3 \times 4, \\ N &= 36 = 3^2 \times 2^2 = 3^2 \times 4 = 3 \times (3 \times 4) = 3 \times 12. \end{aligned} \tag{5.157}$$

From DIF mixed-radix decompositions (5.157) it can be seen that the 12-point MDCT is decomposed into three 4-point MDCTs, and 36-point MDCT into three 12-point MDCTs. In general, the computational complexity of improved DIF mixed-radix fast MDCT algorithm [28] for the composite lengths $N = 3^m \times 2^p$, $m,\ p > 0$, is associated with the value of p. For the composite lengths $N = 3^m \times 4$, $m > 0$, the multiplicative complexity can be reduced as follows. For the value of $n = \frac{1}{2}(\frac{N}{4} - 1)$ the angle is $\frac{\pi}{4}$, and the expressions between square brackets in (5.155) are:

$$\begin{aligned} u_n \frac{1}{2}\sin\frac{\pi}{4} - v_n \frac{\sqrt{3}}{2}\cos\frac{\pi}{4} &= \frac{\sqrt{2}}{4}\left(u_n - \sqrt{3}v_n\right), \\ u_n \frac{1}{2}\cos\frac{\pi}{4} + v_n \frac{\sqrt{3}}{2}\sin\frac{\pi}{4} &= \frac{\sqrt{2}}{4}\left(u_n + \sqrt{3}v_n\right), \end{aligned}$$

requiring three multiplications and two additions (instead of four multiplications and two additions), hence saving one multiplication. Then, the total arithmetic complexity (M_N are multiplications and A_N are additions) is given by [28]

$$M_N = 3 \times M_{\frac{N}{3}} + \frac{4N}{3} - 1, \quad A_N = 3 \times A_{\frac{N}{3}} + \frac{8N}{3}, \quad \text{with} \quad M_4 = 3,\ A_4 = 5,$$

where $\frac{N}{3}$ multiplications by 2 are counted as additions which can be efficiently implemented as shift operations. The regular signal flow graph for the forward MDCT computation for $N = 12$ is shown in Fig. 5.18. The backward MDCT computation can be realized simply by reversing the signal flow graph for the forward MDCT computation and performing inverse operations. The required optimized efficient forward/backward $4/2$-point MDCT modules are presented in Appendix E.1. The total arithmetic complexity of the forward 12-point MDCT is 24 multiplications and 47 additions, whereby 4 multiplications by 2 are counted as additions. Since the forward 12-point MDCT is recursively reused for the forward 36-point MDCT, the total arithmetic complexity of the forward 36-point MDCT computation is 119 multiplications and 237 additions, whereby similarly 24 multiplications by 2 are counted as additions. The backward MDCT computation requires exactly $\frac{N}{2}$ less additions than that of the forward MDCT.

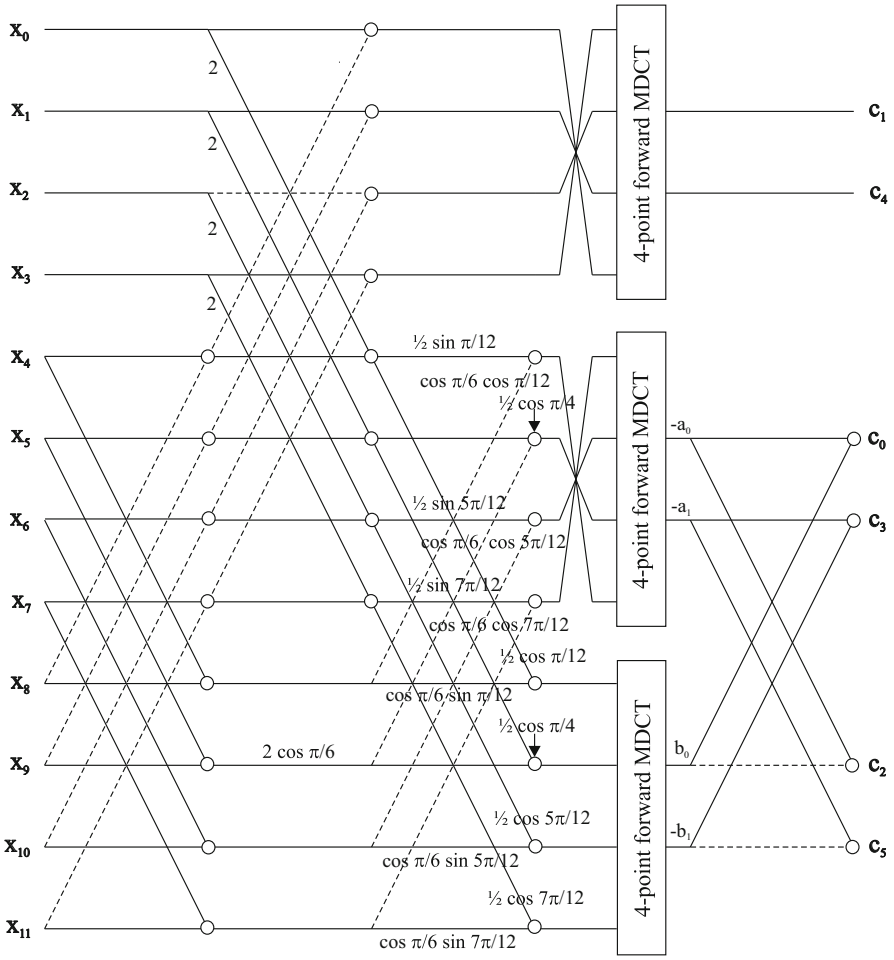

Fig. 5.18 Signal flow graph for the forward MDCT computation for $N = 12$ (3×4 mixed-radix decomposition) based on the improved DIF mixed-radix fast algorithm [28]

Note 21: When the terms $\sqrt{3}v_n$ are considered to be precomputed, then (5.155) can be alternatively implemented by the bilinear computational structure (see Appendix F.3) as

$$\begin{pmatrix} -\frac{1}{2}\cos\frac{\pi(2n+1)}{N} & \frac{1}{2}\sin\frac{\pi(2n+1)}{N} \\ \frac{1}{2}\sin\frac{\pi(2n+1)}{N} & \frac{1}{2}\cos\frac{\pi(2n+1)}{N} \end{pmatrix} \begin{pmatrix} \sqrt{3}v_n \\ u_n \end{pmatrix}, \qquad n = 0, 1, \ldots, \frac{N}{3} - 1,$$

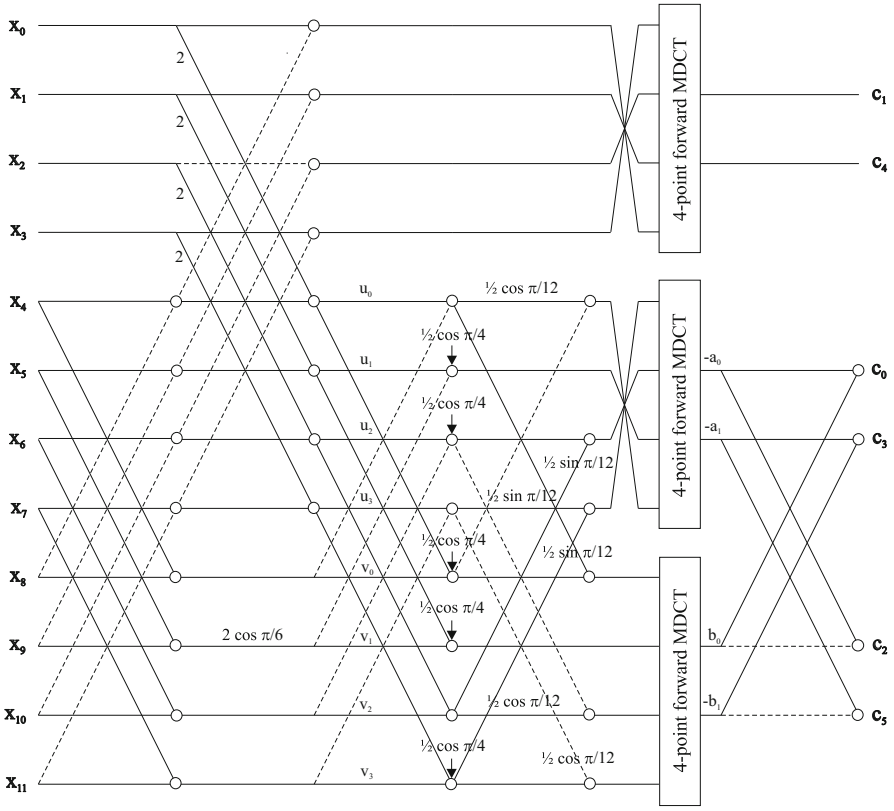

Fig. 5.19 Signal flow graph for the optimized forward MDCT computation for $N = 12$ based on the improved DIF mixed-radix fast algorithm [28]

however, at the cost of $\frac{N}{3}$ more additions. In the bilinear computational structure all the multiplications are independent and can be realized concurrently (preferred in hardware implementations).

The multiplicative complexity as well as the number of unique nontrivial angles for the forward 12-point MDCT computation based on the improved DIF mixed-radix fast MDCT algorithm can be further reduced by an optimization procedure [28] which is valid only for $N = 12$. The corresponding regular signal flow graph for the optimized forward 12-point MDCT computation is shown in Fig. 5.19. The optimized computation of the forward/backward 12-point MDCT requires totally 21 multiplications and 53/47 additions, wherein 4 multiplications by 2 are counted as additions. Since the forward/backward 12-point MDCT is recursively reused for the forward/backward 36-point MDCT computation, the total arithmetic complexity of the forward/backward 36-point MDCT computation is 110 multiplications and 255/237 additions, whereby 24 multiplications by 2 are counted as additions.

5.4.6.2 DIT Mixed-Radix Fast MDCT Algorithm [67]

Complete formulae of the DIT mixed-radix fast MDCT algorithm for the composite lengths $N = 3^m \times 2^p$, $m > 0,\ p > 1$, are given by [67]

$$\begin{aligned}
c_k &= e_k + \left[\cos\frac{\pi(2k+1)}{N} f_k + \sin\frac{\pi(2k+1)}{N} g_k\right],\\
c_{\frac{N}{3}-1-k} &= -e_k + \frac{1}{2}\left[\cos\frac{\pi(2k+1)}{N} f_k + \sin\frac{\pi(2k+1)}{N} g_k\right]\\
&\quad -\frac{\sqrt{3}}{2}\left[\sin\frac{\pi(2k+1)}{N} f_k - \cos\frac{\pi(2k+1)}{N} g_k\right],\\
c_{\frac{N}{3}+k} &= -e_k + \frac{1}{2}\left[\cos\frac{\pi(2k+1)}{N} f_k + \sin\frac{\pi(2k+1)}{N} g_k\right]\\
&\quad +\frac{\sqrt{3}}{2}\left[\sin\frac{\pi(2k+1)}{N} f_k - \cos\frac{\pi(2k+1)}{N} g_k\right],\\
&\qquad k = 0, 1, \ldots, \frac{N}{6} - 1,
\end{aligned} \tag{5.158}$$

where

$$\begin{aligned}
e_k &= \sum_{n=0}^{\frac{N}{3}-1} x_{3n+1} \cos\left[\frac{\pi}{2(N/3)}\left(2n+1+\frac{N}{6}\right)(2k+1)\right],\\
f_k &= \sum_{n=0}^{\frac{N}{3}-1} (x_{3n} + x_{3n+2}) \cos\left[\frac{\pi}{2(N/3)}\left(2n+1+\frac{N}{6}\right)(2k+1)\right],\\
g_{\frac{N}{6}-1-k} &= \sum_{n=0}^{\frac{N}{3}-1} (-1)^{n+\frac{N}{12}} (x_{3n} - x_{3n+2}) \cos\left[\frac{\pi}{2(N/3)}\left(2n+1+\frac{N}{6}\right)(2k+1)\right],\\
&\qquad k = 0, 1, \ldots, \frac{N}{6} - 1.
\end{aligned} \tag{5.159}$$

Thus, the forward N-point MDCT is again obtained from the computation of three $\frac{N}{3}$-point MDCTs. The input data sequence $\{x_n\}$ is reordered into three sub-sequences $\{x_{3n+1}\}$, $\{x_{3n}\}$ and $\{x_{3n+2}\}$, $n = 0, 1, \ldots, \frac{N}{3} - 1$.

Similarly, since the short and long block sizes in MP3 are the composite lengths $N = 3^m \times 4$, $m > 0$, their DIT mixed-radix decompositions are the same as DIF mixed-radix decompositions given in (5.157). Further, for the composite lengths $N = 3^m \times 4$, $m > 0$, the multiplicative complexity of the DIT mixed-radix fast MDCT algorithm can be reduced as follows. For the value of $k = \frac{1}{2}(\frac{N}{4} - 1)$ the angle is $\frac{\pi}{4}$, and from (5.158) we have

$$\frac{1}{2}\left(\cos\frac{\pi}{4}f_k+\sin\frac{\pi}{4}g_k\right)=\frac{1}{2}\cos\frac{\pi}{4}(f_k+g_k),$$
$$\frac{\sqrt{3}}{2}\left(\sin\frac{\pi}{4}f_k-\cos\frac{\pi}{4}g_k\right)=\cos\frac{\pi}{6}\cos\frac{\pi}{4}(f_k-g_k),$$

requiring two multiplications and two additions (instead of four multiplications and two additions), hence saving two multiplications. Then, the total arithmetic complexity is given by [67]

$$M_N=3\times M_{\frac{N}{3}}+\frac{2N}{3}-2,\quad A_N=3\times A_{\frac{N}{3}}+\frac{5N}{3},\quad \text{with}\quad M_4=3,\ A_4=5,$$

plus $\frac{N}{6}$ shift operations. The regular signal flow graph for the forward MDCT computation for $N=12$ is shown in Fig. 5.20. The backward MDCT computation

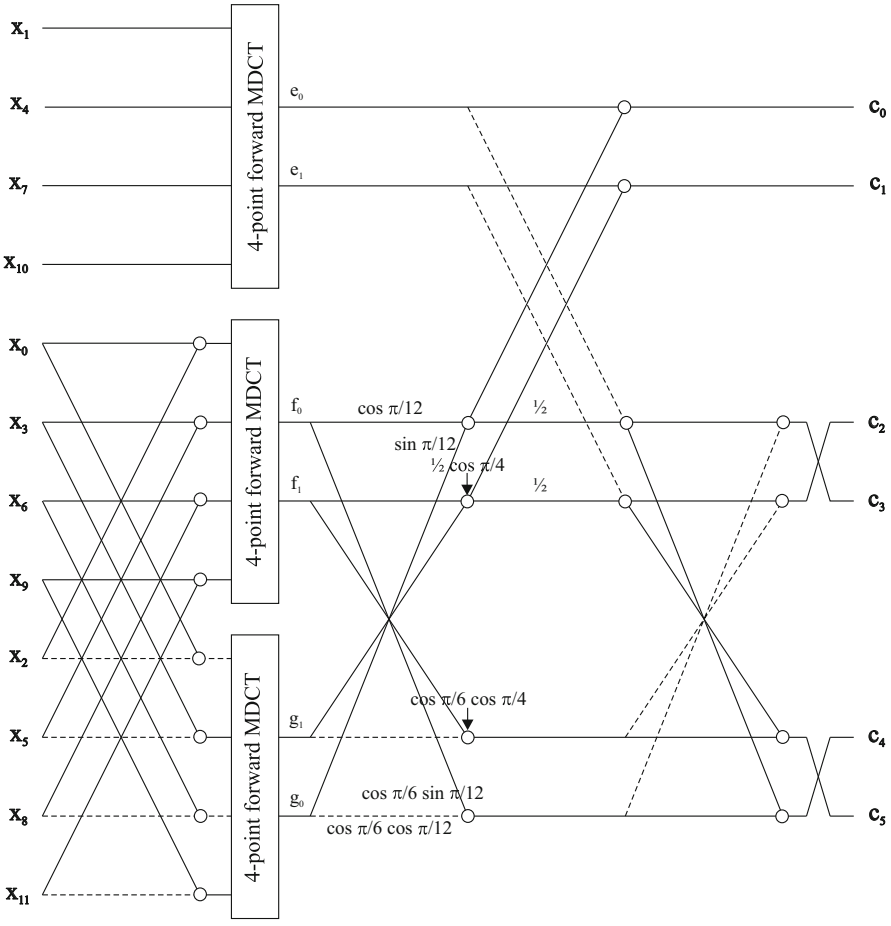

Fig. 5.20 Signal flow graph for the forward MDCT computation for $N=12$ (3×4 mixed-radix decomposition) based on the DIT mixed-radix fast algorithm

can be realized simply by reversing the signal flow graph for the forward MDCT computation and performing inverse operations. The required optimized efficient forward/backward 4/2-point MDCT modules are presented in Appendix E.1. The total arithmetic complexity of the forward/backward 12-point MDCT is 15 multiplications, 35/29 additions and 2 shifts. Since the forward/backward 12-point MDCT is recursively reused for the forward/backward 36-point MDCT, the total arithmetic complexity of the forward/backward 36-point MDCT computation is 67 multiplications, 165/147 additions, and 6 shifts.

5.4.6.3 Combined DIF Radix-2 and DIT Mixed-Radix Fast MDCT Algorithms [44, 67]

In principle, the mixed-radix decomposition process can start with any radix number. However, a lower count of arithmetic operations is achieved, if the decomposition process starts with the smallest radix number, i.e., with the radix-2 [114]. A such DIF radix-2 recursive fast MDCT algorithm for any even-lengths N divisible by 4 has been proposed in [67].

Complete formulae of the DIF radix-2 fast recursive MDCT algorithm for the even-lengths N divisible by 4 are given by [67]

$$c_{2k} = (-1)^{\lfloor \frac{k+1}{2} \rfloor} \sum_{n=0}^{\frac{N}{2}-1} x_n^{(k)} \cos\left[\frac{\pi}{2(N/2)}\left(2n+1+\frac{N}{4}\right)(2k+1)\right],$$

$$c_{2k+1} = (-1)^{\lfloor \frac{k+1}{2} \rfloor} \sum_{n=0}^{\frac{N}{2}-1} x_{\frac{N}{2}-1-n}^{(k)} \cos\left[\frac{\pi}{2(N/2)}\left(2n+1+\frac{N}{4}\right)(2k+1)\right],$$

$$k = 0, 1, \ldots, \frac{N}{4}-1, \tag{5.160}$$

where $\lfloor \, . \, \rfloor$ is the lower integer part of argument, and the terms $\{x_n^{(k)}\}$, $\{x_{\frac{N}{2}-1-n}^{(k)}\}$ are given by

$$x_n^{(k)} = \begin{cases} \left(x_n - x_{\frac{N}{2}-1-n}\right)\cos\frac{\pi(2n+1)}{2N} + \left(x_{\frac{N}{2}+n} + x_{N-1-n}\right)\sin\frac{\pi(2n+1)}{2N}, & \text{if k is even,} \\ -\left(x_n - x_{\frac{N}{2}-1-n}\right)\sin\frac{\pi(2n+1)}{2N} + \left(x_{\frac{N}{2}+n} + x_{N-1-n}\right)\cos\frac{\pi(2n+1)}{2N}, & \text{if k is odd,} \end{cases}$$

$$x_{\frac{N}{2}-1-n}^{(k)} = \begin{cases} -\left(x_n - x_{\frac{N}{2}-1-n}\right)\sin\frac{\pi(2n+1)}{2N} + \left(x_{\frac{N}{2}+n} + x_{N-1-n}\right)\cos\frac{\pi(2n+1)}{2N}, & \text{if k is even,} \\ \left(x_n - x_{\frac{N}{2}-1-n}\right)\cos\frac{\pi(2n+1)}{2N} + \left(x_{\frac{N}{2}+n} + x_{N-1-n}\right)\sin\frac{\pi(2n+1)}{2N}, & \text{if k is odd,} \end{cases}$$

$$n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{5.161}$$

The radix-2 recursive fast MDCT algorithm decomposes an N-point MDCT into two $\frac{N}{2}$-point MDCTs. The decomposition process is recursively repeated until the lower-order even-length MDCTs remain $(3^m \times 2^{p-1})$-point MDCTs for the composite lengths and 4-point MDCTs for the 2^p-lengths. Equation (5.161) defines the block of $\frac{N}{4}$ Givens–Jacobi rotations.

In general, for the composite lengths $N = 3^m \times 2^p$, $m, p > 0$, the total arithmetic complexity of the DIF radix-2 recursive fast MDCT algorithm is given by [67]

$$M_N = 2 \times M_{\frac{N}{2}} + \frac{3N}{4}, \quad A_N = 2 \times A_{\frac{N}{2}} + \frac{5N}{4}.$$

Since the short and long block sizes in MP3 are composite lengths $N = 4 \times 3^m$, $m > 0$, being even-lengths, their corresponding radix-2 decompositions are, respectively, given by

$$\begin{aligned} N &= 12 = 2^2 \times 3 = 2 \times (2 \times 3) = 2 \times 6, \\ N &= 36 = 2^2 \times 3^2 = 2 \times (2 \times 3^2) = 2 \times 18. \end{aligned} \tag{5.162}$$

From radix-2 decompositions (5.162) it can be seen that the 12-point MDCT is decomposed into two 6-point MDCTs, and the 36-point MDCT into two 18-point MDCTs. The forward 18-point MDCT can be generated from the improved DIF mixed-radix fast MDCT algorithm given by (5.154), (5.155), and (5.156) by recursively reuse three forward 6-point MDCTs (3×6 mixed-radix DIF decomposition with $M_6 = 1, A_6 = 6$ plus 1 shift). Its arithmetic complexity is 21 multiplications, 66 additions, and 2 shifts. Therefore, for such efficient MDCT implementation in MP3 we need two separate computational structures, one for the 12-point MDCT and one for the 36-point MDCT without the possibility to reuse 12-point MDCT for the 36-point MDCT computation. The total arithmetic complexity of such forward 36-point MDCT implementation is shown in Table 5.1.

The MDCT algorithm with radix-2 decomposition for $N = 12 = 2 \times 6$ leads to the regular signal flow graph for the forward 12-point MDCT computation shown in Fig. 5.21. The backward MDCT computation can be realized simply by reversing the signal flow graph for the forward MDCT computation and performing inverse operations. The required optimized efficient 6/3-point MDCT modules are presented in Appendix E.2. Efficient implementations of Givens–Jacobi rotations are presented in Appendices F.2 and F.3. The total arithmetic complexity of the forward/backward 12-point MDCT computation is 11 multiplications, 27/21 additions, and 2 shifts being the most efficient implementation among all mixed-radix 12-point MDCT implementations. On the other hand, the forward/backward 36-point MDCT computation based on the MDCT algorithm with radix-2 decomposition ($N = 36 = 2 \times 18$) requires totally 43 multiplications, 129/111 additions, and 4 shifts being also the most efficient implementation among all mixed-radix

Table 5.1 Arithmetic complexity of various efficient implementations of the forward 36-point MDCT obtained by combining radix-2 and DIF or DIT mixed-radix decompositions

Decomposition of 36-point MDCT	Decomposition of lower-order even-length MDCT	The arithmetic complexity	Remark
Radix-2 (2 × 18)	DIF (3 × 6)	69M+177A+6S	
DIF (3 × 12)	DIF (3 × 4)	110M+255A	24A are M by 2
DIT (3 × 12)	DIT (3 × 4)	67M+165A+6S	
DIF (3 × 12)	Radix-2 (2 × 6)	80M+177A+6S	12A are M by 2
DIT (3 × 12)	Radix-2 (2 × 6)	55M+141A+18S	
DIF (3 × 12)	DIT (3 × 4)	92M+201A+6S	12A are M by 2
DIT (3 × 12)	DIF (3 × 4)	85M+219A+6S	12A are M by 2

M multiplication, *A* addition, *S* shift

36-point MDCT implementations [44]. The required optimized efficient 18/9-point MDCT modules are presented in Appendix E.3.

In order to recursively reuse the forward and backward 12-point MDCTs for the forward and backward 36-point MDCT, we may combine radix-2 and DIF or DIT mixed-radix decompositions to obtain various efficient implementations of the forward/backward 36-point MDCT. We have three different 12-point MDCT efficient implementations (shown in Figs. 5.19, 5.20, and 5.21) and two different 36-point MDCT implementations resulting in six various 36-point MDCT efficient implementations with different arithmetic complexities. They are summarized in Table 5.1. The best 36-point forward/backward MDCT implementation is obtained by combining the DIT mixed-radix decomposition for 36-point MDCT with the radix-2 decomposition for 12-point MDCT. It requires 55 multiplications, 141/123 additions, and 18 shifts.

5.4.7 MDCT Implementations Based on Recursive/Regressive Filter Structures

Quite different class of algorithms which can be adopted for an efficient forward/backward MDCT implementation in MP3 are algorithms based on recursive or regressive filter structures [34, 70–88]. Following the approaches originally developed for the DCT-II and its inverse, DCT-III [113, 123, 124, 141], the forward/backward MDCT kernels are converted to recursive (or recurrent) equations based on:

- Recurrence formulae for Chebyshev polynomials of the second and third kinds [141]. Typical recursive forward/backward MDCT implementations [73, 88] follow the DCT-II-based approach [141].

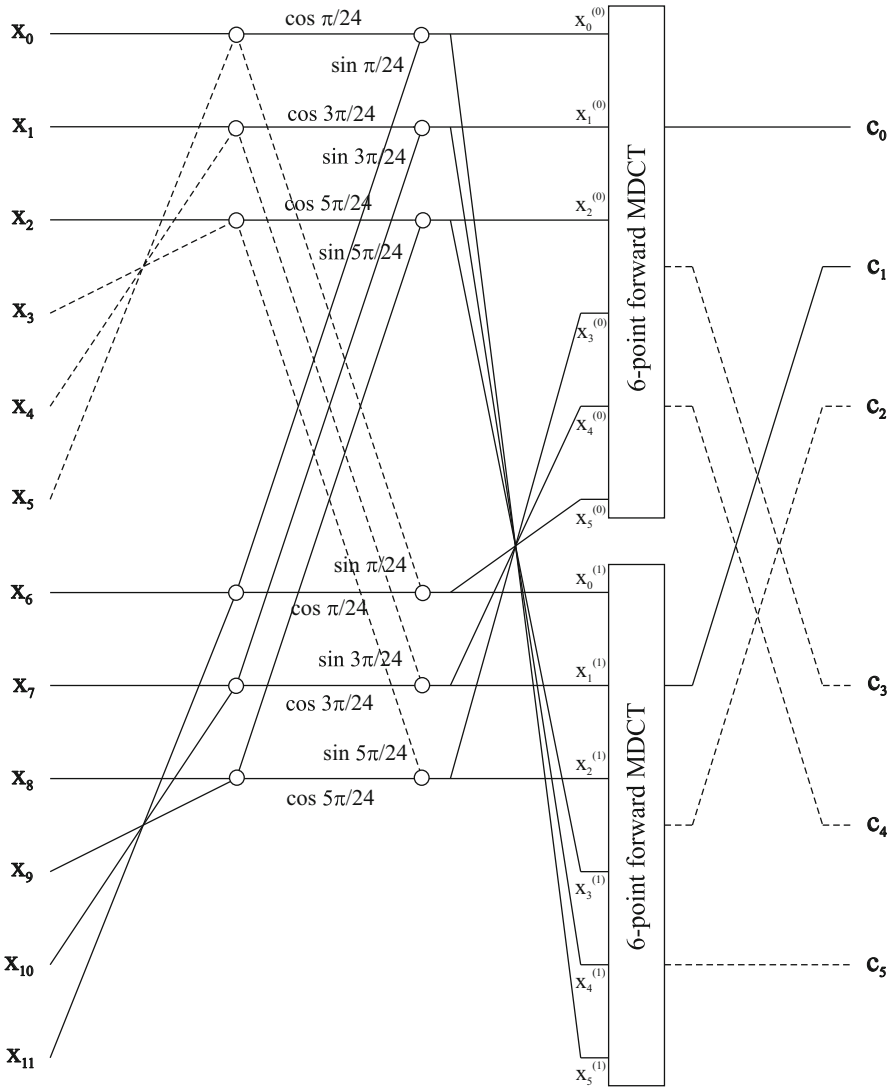

Fig. 5.21 Signal flow graph for the forward MDCT computation for $N = 12$ (2×6 mixed-radix decomposition) based on the DIF radix-2 recursive fast algorithm

- Sinusoidal recursive formulae exploiting Goertzel recursive formula [129]. Such typical recursive forward/backward MDCT implementations [85, 87] follow the DCT-II-based approach [123, 124].
- Clenshaw's recurrence formula (used for upward or downward ordering) [113]. A typical recursive forward/backward MDCT implementation [86] follows the DCT-II-based approach [113].

- Goertzel digital filters, where the forward/backward MDCT are expressed in the form of discrete time domain convolution sums [74].
- Recurrence formulae for the cosine and sine functions [75].

We recall that the N-point MDCT using the permutation (5.36) or (5.47) can optionally be converted to the DCT-IV of half size which may be subsequently converted to the DCT-II of the same size at the cost of additional $\frac{N}{2}$ pre-multiplications and $\frac{N}{2}-1$ recursive post-additions. Then, following the above mentioned approaches the recursive/regressive filter structures have been developed [70–72, 78, 79, 83]. The DCT-IV-based recursive MDCT filter structures are presented in [34, 76, 77, 80–82, 84]. Since the DCT-IV is self-inverse, both the forward and backward MDCT computations are implemented by an identical recursive/regressive filter structure. Although these recursive algorithms are not so efficient in terms of the arithmetic complexity, they can be represented by regular recursive or regressive filter structures which provide efficient online schemes particularly suitable for parallel VLSI implementations of the variable- or general-lengths forward/backward MDCT.

In the following subsections several representative efficient MDCT implementations based on recursive or regressive filter structures are discussed in more detail. We note that in the derivation of recursive algorithms [73, 85–87] the definitions of the forward and backward MLT given by (5.11) and (5.12), respectively, are used. We recall that $N = 2M$, where M is the number of transform coefficients.

5.4.7.1 Recursive MDCT Implementation [73]

Complete formulae for the recursive computation of one MDCT coefficient c_k in the forward MDCT are defined as [73]

$$c_k = -p_{N-1} \cos\left[\left(\frac{M-1}{2}\right)\theta_k\right], \quad \theta_k = \frac{\pi(2k+1)}{2M}, \quad k = 0, 1, \ldots, M-1, \tag{5.163}$$

where the term p_{N-1} is obtained from the recursive equation given by

$$p_n = x_n + p_{n-1}\, 2\cos\theta_k - \frac{\cos\left[\left(\frac{M+1}{2}\right)\theta_k\right]}{\cos\left[\left(\frac{M-1}{2}\right)\theta_k\right]} x_{n-1} - p_{n-2}, \quad n = 0, 1, \ldots, N-1, \tag{5.164}$$

with the following initial conditions: $p_{-1} = p_{-2} = 0$ and $x_{-1} = x_{N-1}$. For $n = 0$ we get

$$p_0 = x_0 - \frac{\cos\left[\left(\frac{M+1}{2}\right)\theta_k\right]}{\cos\left[\left(\frac{M-1}{2}\right)\theta_k\right]} x_{N-1}. \tag{5.165}$$

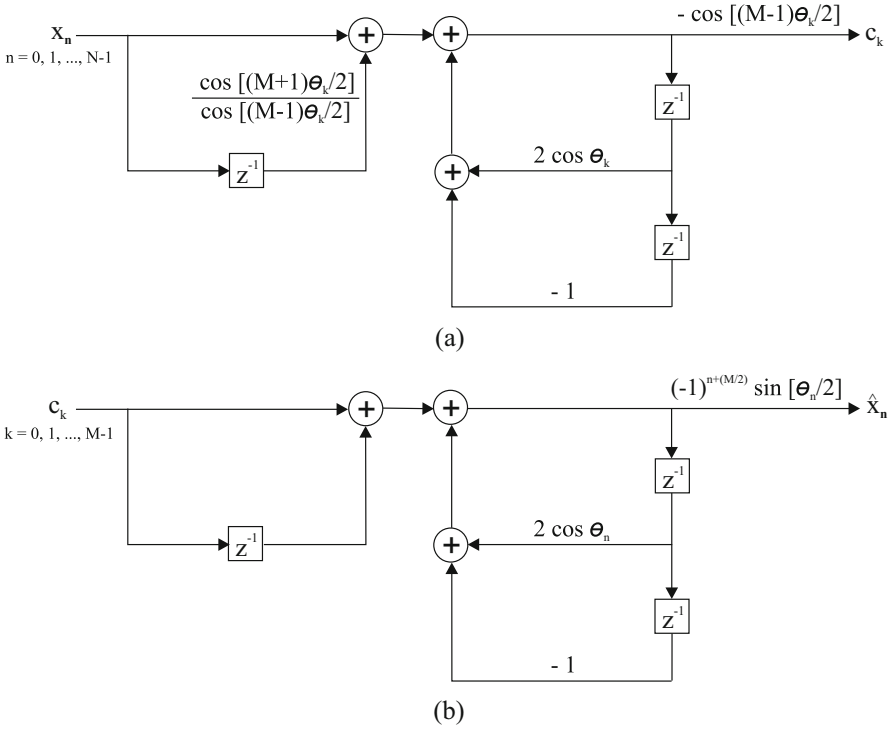

Fig. 5.22 Regressive filter structures [73] for (**a**) the recursive forward MDCT computation, (**b**) recursive backward MDCT computation (M is even)

From (5.163), (5.164), and (5.165) it can be seen that for the computation of one coefficient c_k we need $2N$ multiplications and $3N-2$ additions. The corresponding regressive filter structure for the parallel VLSI computing of the forward MDCT is shown in Fig. 5.22a. z^{-1} denotes one-sample delay operator. The input data sequence $\{x_n\}$ is in natural order. The regressive filter structure in Fig. 5.22a for the recursive forward MDCT computation consists of three multipliers, three adders, and three delay operators.

On the other hand, complete formulae for the recursive computation of one time domain aliased sample $\hat{x}_n$ in the backward MDCT (M is even) are defined as [73]

$$\hat{x}_n = p_{M-1}\,(-1)^{n+\frac{M}{2}}\,\sin\frac{\theta_n}{2}, \quad \theta_n = \frac{\pi}{M}\left(n+\frac{M+1}{2}\right), \quad n = 0,1,\ldots,N-1, \tag{5.166}$$

where the term p_{M-1} is obtained from the recursive equation given by

$$p_k = c_k + p_{k-1}\,2\cos\theta_n + c_{k-1} - p_{k-2}, \quad k = 0,1,\ldots,M-1, \tag{5.167}$$

with the following initial conditions: $p_{-1} = p_{-2} = 0$ and $c_{-1} = 0$. For $k = 0$ we get

$$p_0 = c_0. \tag{5.168}$$

From (5.166), (5.167), and (5.168) it follows that for the computation of one time domain aliased sample $\hat{x}_n$ we need M multiplications and $3M - 3$ additions. Using the symmetry property of $\{\hat{x}_n\}$ given by (5.14) it is sufficient to compute only $\frac{N}{2}$ time domain aliased samples. The corresponding regressive filter structure for the parallel VLSI implementation of the backward MDCT (M is even) is shown in Fig. 5.22b. The input data sequence $\{c_k\}$ is in natural order. The regressive filter structure in Fig. 5.22b for the recursive backward MDCT computation consists of two multipliers, three adders, and three delay operators.

5.4.7.2 Recursive MDCT Implementation [85, 87]

Complete formulae for the recursive computation of one MDCT coefficient c_k in the forward MDCT are defined as [85, 87]

$$c_k = v_0 \cos\left[\left(\frac{M+1}{2}\right)\theta_k\right] - v_1 \cos\left[\left(\frac{M-1}{2}\right)\theta_k\right],$$
$$\theta_k = \frac{\pi(2k+1)}{2M}, \quad k = 0, 1, \ldots, M-1, \tag{5.169}$$

where the terms v_0 and v_1 are obtained from the recursive equation given by

$$v_n = x_n + v_{n+1}\, 2\cos\theta_k - v_{n+2}, \qquad n = N-1, N-2, \ldots, 0, \tag{5.170}$$

with the following initial condition: $v_N = v_{N+1} = 0$. For $n = N - 1$ we get

$$v_{N-1} = x_{N-1}. \tag{5.171}$$

From (5.169), (5.170), and (5.171) it follows that for the computation of one coefficient c_k we need $N+1$ multiplications and $2N-1$ additions. The corresponding regressive filter structure for the parallel VLSI implementation of the forward MDCT is shown in Fig. 5.23a. The input data sequence $\{x_n\}$ is in reverse order. The regressive filter structure for the recursive forward MDCT computation consists of three multipliers, three adders, and two delay operators.

Complete formulae for the recursive computation of one time domain aliased sample $\hat{x}_n$ in the backward MDCT are defined as [85, 87]

$$\hat{x}_n = (v_0 - v_1)\cos\frac{\theta_n}{2}, \quad \theta_n = \frac{\pi}{M}\left(n + \frac{M+1}{2}\right), \quad n = 0, 1, \ldots, N-1, \tag{5.172}$$

where the terms v_0 and v_1 are obtained from the recursive equation given by

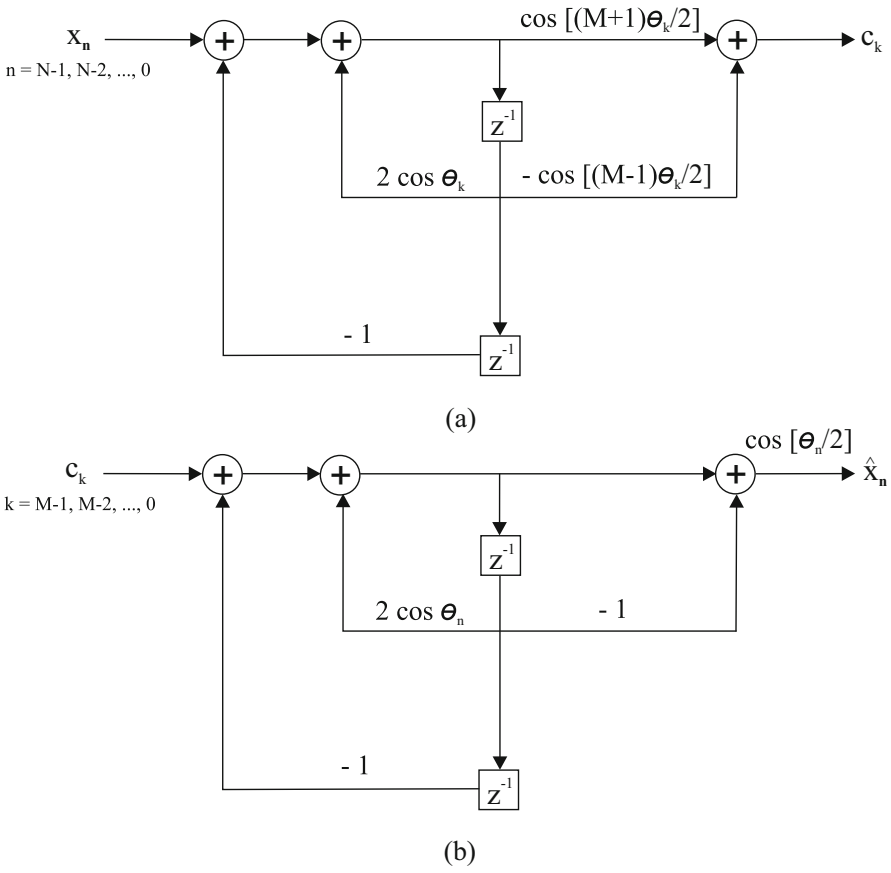

Fig. 5.23 Regressive filter structures [87] for (**a**) the recursive forward MDCT computation, (**b**) recursive backward MDCT computation

$$v_k = c_k + v_{k+1}\, 2\cos\theta_n - v_{k+2}, \qquad k = M-1, M-2, \ldots, 0, \tag{5.173}$$

with the following initial condition: $v_M = v_{M+1} = 0$. For $k = M-1$ we get

$$v_{M-1} = c_{M-1}. \tag{5.174}$$

From (5.172), (5.173), and (5.174) it follows that for the computation of one time domain aliased sample $\hat{x}_n$ we need M multiplications and $2M-1$ additions. Again, using the symmetry property of $\{\hat{x}_n\}$ given by (5.14) it is sufficient to compute only $\frac{N}{2}$ time domain aliased samples. The corresponding regressive filter structure for the parallel VLSI implementation of the backward MDCT is shown in Fig. 5.23b. The input data sequence $\{c_k\}$ is in reverse order. The regressive filter structure for the recursive backward MDCT computation consists of two multipliers, three adders, and two delay operators.

5.4.7.3 Recursive MDCT Implementation [86]

Complete formulae for the recursive computation of one MDCT coefficient c_k in the forward MDCT (Clenshaw's recurrence formula is used for the upward ordering) are defined as [86]

$$c_k = a_{N-2} \cos\left[\left(\frac{M+1}{2}\right)\theta_k\right] - a_{N-1} \cos\left[\left(\frac{M-1}{2}\right)\theta_k\right],$$
$$\theta_k = \frac{\pi(2k+1)}{2M}, \quad k = 0, 1, \ldots, M-1, \tag{5.175}$$

where the terms a_{N-2} and a_{N-1} are obtained from the recursive equation given by

$$a_n = x_n + a_{n-1}\, 2\cos\theta_k - a_{n-2}, \quad n = 0, 1, \ldots, N-1, \tag{5.176}$$

with the following initial condition: $a_{-2} = a_{-1} = 0$. For $n = 0$ we get

$$a_0 = x_0. \tag{5.177}$$

From (5.175), (5.176), and (5.177) it follows that for the computation of one coefficient c_k we need $N+1$ multiplications and $2N-1$ additions. The corresponding regressive filter structure for the parallel VLSI computing of the forward MDCT is shown in Fig. 5.24a. The input data sequence $\{x_n\}$ is in natural order. The regressive filter structure for the recursive forward MDCT computation consists of three multipliers, three adders and two delay operators. If Clenshaw's recurrence formula is used for the downward ordering, then the corresponding regressive filter structure is identical to that of shown in Fig. 5.23a.

Complete formulae for the recursive computation of one time domain aliased sample $\hat{x}_n$ in the backward MDCT (Clenshaw's recurrence formula is used for the upward ordering, and M is even) are defined as [86]

$$\hat{x}_n = (a_0 - a_1)\cos\frac{\theta_n}{2}, \quad \theta_n = \frac{\pi}{M}\left(n + \frac{M+1}{2}\right), \quad n = 0, 1, \ldots, N-1, \tag{5.178}$$

where the terms a_0 and a_1 are obtained from the recursive equation given by

$$a_k = c_k + a_{k+1}\, 2\cos\theta_n - a_{k+2}, \quad k = M-1, M-2, \ldots, 0, \tag{5.179}$$

with the following initial condition: $a_M = a_{M+1} = 0$. For $k = M-1$ we get

$$a_{M-1} = c_{M-1}. \tag{5.180}$$

From (5.178), (5.179), and (5.180) it follows that for the computation of one time domain aliased sample $\hat{x}_n$ we need M multiplications and $2M-1$ additions. Again, using the symmetry property of $\{\hat{x}_n\}$ given by (5.14) it is sufficient to compute only

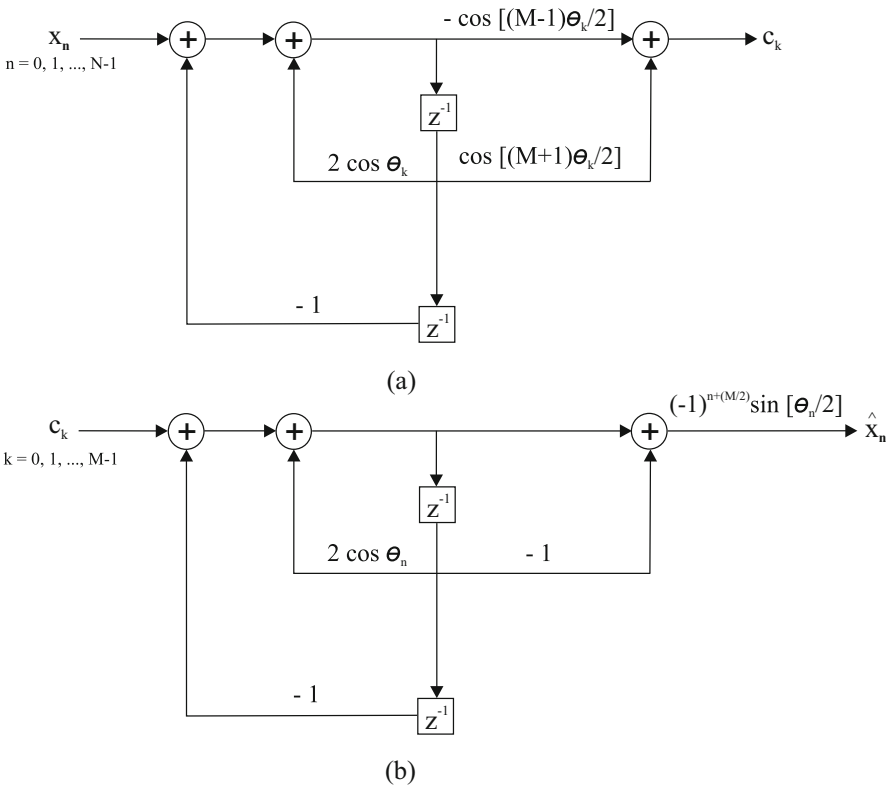

Fig. 5.24 Regressive filter structures [86] for (**a**) the recursive forward MDCT computation, (**b**) recursive backward MDCT computation

$\frac{N}{2}$ time domain aliased samples. The corresponding regressive filter structure for the parallel VLSI implementation of the backward MDCT is shown in Fig. 5.24b. The input data sequence $\{c_k\}$ is in natural order. The regressive filter structure for the recursive backward MDCT computation consists of two multipliers, three adders, and two delay operators. If Clenshaw's recurrence formula is used for the downward ordering, then the corresponding regressive filter structure is identical to that of shown in Fig. 5.23b.

5.4.8 Comparison of Efficient MDCT Implementations in MP3

Improvements in integrated circuit technology have enabled audio codecs to be built on a single chip. An optimal hardware implementation of such systems on a chip (SOC) requires detailed analysis of the trade-offs between system performance and

hardware parameters such as the delay, memory requirements, clock cycle counts, accuracy, power consumption, and circuit area. A simple low-cost and power-efficient (parallel) hardware architecture for the audio encoder, and in particular, for battery operated audio decoder is desired [110].

In general, when a fast algorithm is intended to be implemented in hardware/software for real-time processing, besides arithmetic complexity the design criteria such as I/O data access scheme (serial-in serial-out or parallel-in parallel-out), data indexing scheme, structural simplicity (using the identical or separate fast computational structures, common module sharing in fast computational structures for different block sizes), regularity (complete/incomplete recursive algorithm, butterfly-like repetitive stages and in-place computation), modularity (repetitive using basic modules in a corresponding fast computational structure) [67], and numerical precision and stability need to be considered for the evaluation of algorithm performance [110]. Note that the inherent parallelism in an investigated fast algorithm is not frequently exploited [104]. Essentially, using the identical fast computational structures both for the forward and backward transforms as well as sharing common modules for different block sizes significantly reduce the design time and hardware resources of a potential MP3 audio codec.

All presented fast computational structures for the efficient implementations of the forward/backward MDCT block transforms in MP3 are regular with butterfly-like stages and modular. Besides the arithmetic complexity for the short and long audio blocks, for evaluation and comparison of efficient MDCT implementations in MP3 the following criteria are specified (acronyms are used in Table 5.2 in the last column):

- Using complex arithmetic (CA) or real arithmetic (RA).
- Required input/internal/output permutations (IP/InP/OP) in the fast computational structure.
- Required sign changes (SC) in the fast computational structure.
- Required recursive post-addition stage(s) (RPAS) in the fast computational structure.
- Completely recursive fast algorithm (CRFA).
- Reversing the fast forward MDCT computational structure for the backward MDCT computation (RevFCS). This results in two separate computational structures for the encoder and two for the decoder.
- Using the identical fast computational structures (IFCS) with simple pre- and post-processing data sequences both for the forward and backward MDCT (one for the short and one for the long block).
- Reusing the fast computational structure (RFCS) for the short block in the fast computational structure for the long block.
- Unifying the fast computational structures (UFCS), i.e., integrating fast computational structures for the short and long blocks into a single one.
- Optimizing the fast computational structures (OFCS) in context of the complete analysis/synthesis MDCT (MLT) filter banks (see Sect. 5.5 for efficient imple-

Table 5.2 Comparison of efficient implementations of the forward/backward MDCT block transforms in MP3 in terms of the arithmetic complexity and specified criteria

MDCT algorithm	Forward/backward MDCT computation		Remarks
	$N = 12$	$N = 36$	
DFT/FFT-based [37]	20M+42/36A+2S (3M+3A can be saved)	70M+174/156A+4S (3M+3A can be saved)	CA, OP, SC, IFCS
DFT/FFT-based [42]	17M+39/33A+2S	67M+171/153A+4S	CA, OP, SC, IFCS
DFT/FFT-based [43, 52]	17M+33/27A+2S	67M+153/135A+4S	CA, InP, OP, SC, IFCS
DCT-II/DST-II-based [24] refined version of [32]	11M+39/33A+2S	43M+165/147A+4S	RA, InP, OP, SC, RevFCS
DCT-II/DST-II-based [56]	11M+27/21A+2S	43M+129/111A+4S	RA, InP, OP, SC, RevFCS
DCT-IV-based [114]	11M+33/27A+2S	61M+169/151A+6S	RA, IP, InP, IFCS, OFCS
DCT-IV-based [25]	11M+27/21A+2S	43M+129/111A+4S	RA, IP, InP, OP, SC, IFCS, OFCS
DCT-IV-based [31]	11M+27/21A+2S	43M+129/111A+4S	RA, IP, InP, IFCS, OFCS
DCT-IV-based [38, 40]	9M+27/21A+2S	36M+148/130A+2S	RA, IP, OP, SC, RFCS, UFCS, OFCS
DCT-IV-based [29]	11M+27/21A+2S (9M+27/21A+2S)	43M+129/111A+4S	RA, IFCS, OFCS
DCT-IV/DCT-II-based (representative)	11M+27/21A+4S	43M+129/111A+6S	RA, InP, RPAS,IFCS
DCT-IV/SDCT-II-based [50]	11M+27/21A+2S	43M+129/111A+6S	RA,InP, RPAS, IFCS
DCT-IV/SDCT-II-based [66]	10M+28/22A+2S	37M+135/117A+3S	RA, InP, RPAS, IFCS
Evenly stacked MDCT-based [39, 64]	11M+27/21A+4S	43M+129/111A+6S	RA, InP, RPAS, RevFCS
Evenly stacked MDCT-based [26]	17M+33/27A+4S	61M+147/129A+6S	RA,InP, RPAS, RevFCS
Improved DIF mixed-radix [28]	21M+49/43A+4S	110M+231/213A+24S	RA,InP,OP, SC, RevFCS
DIT mixed-radix [67]	15M+35/29A+2S	67M+165/147A+6S	RA, IP, InP, RevFCS
Combined radix-2 and DIT mixed-radix [67]	11M+27/21A+2S	55M+141/123A+18S	RA, InP, SC CRFA,RevFCS

M real multiplication, *A* real addition, *S* shift

mentations of windowing&overlap and windowing&overlap&add procedures via the block of Givens–Jacobi rotations or 2-point Hankel matrix products).

Comparison of efficient implementations of the forward and backward MDCT (MLT) block transforms in MP3 in terms of the arithmetic complexity and specified criteria above is summarized in Table 5.2.

The DFT/FFT-based MDCT implementations in MP3 are quite efficient in terms of the structural simplicity. We recall that the DCT-IV-based MDCT implementations are the most efficient both in terms of the arithmetic complexity and structural simplicity. In particular, the structure of DCT-IV algorithms for the most efficient MDCT implementation [38, 40] allows to create a single unified architecture to process both the short and long audio blocks. Moreover, if we use these the most efficient MDCT algorithms in the implementation of complete TDAC analysis/synthesis MDCT (MLT) filter banks, then they can result in the most efficient and compact architecture of TDAC fast analysis/synthesis MDCT (MLT) filter banks in MP3 audio encoder/decoder.

Finally, comparison of efficient implementations of the forward and backward MDCT block transforms in MP3 based on recursive/regressive filter structures (for one coefficient/sample computation) is summarized in Table 5.3. The last column in Table 5.3 specifies the complexity of the corresponding regressive filter structure. In regressive filter structures [73, 85, 87] for the forward/backward MDCT computation the input data sequences $\{x_n\}$ and $\{c_k\}$ are in natural order, while in the regressive filter structure [86] they are in reverse order. Although these recursive algorithms are not so efficient in terms of the arithmetic complexity, regressive structures provide an efficient scheme for the parallel VLSI implementation of the variable-length MDCT.

Table 5.3 Comparison of efficient implementations of the forward and backward MDCT block transforms in MP3 based on recursive/regressive filter structures (for one coefficient/sample computation)

MDCT algorithm	Forward/backward MDCT computation		Complexity of corresponding regressive filter structure
	$N = 12$	$N = 36$	
Chiang and Liu [73]		24/6M+34/15A 72/18M+106/51A	3/2 multipliers, 3 adders and 3 delay elements
Nikolajevič and Fettweis [86]	13/6M+23/11A	37/18M+71/35A	3/2 multipliers, 3 adders and 2 delay elements
Nikolajevic and Fettweis [85, 87]	13/6M+23/11A	37/18M+71/35A	3/2 multipliers, 3 adders and 2 delay elements

5.5 Fast Analysis/Synthesis MDCT (MLT) Filter Banks

With respect to the notation introduced in Sect. 5.2.2 we recall that the TDAC analysis MDCT filter bank given by (5.4) in processing two adjacent overlapped data blocks $\{x_n^{(t)}\}$ and $\{x_n^{(t+1)}\}$ consists of the windowing&overlap procedure and transforming adjacent data blocks by the forward MDCT block transform (realized by a fast computational structure). Vice versa, the TDAC synthesis MDCT filter bank given by (5.5) in processing two adjacent blocks of transform coefficients $\{c_k^{(t)}\}$ and $\{c_k^{(t+1)}\}$ consists of the backward MDCT transformation of coefficients (realized by a fast computational structure) and the windowing&overlap&add procedure to perfectly reconstruct the original data sequence in the overlapped part.

For a given N the implementation of complete TDAC analysis/synthesis MDCT filter banks using the direct approach requires totally $\frac{N}{2}(N+2)$ multiplications and $\frac{N}{2}(N-1)$ additions. Specifically, this requires for the short audio block 84 multiplications and 66 additions, while for the long audio block 684 multiplications and 630 additions. However, if the forward/backward MDCT is implemented by a fast DCT-IV computational structure, then the total arithmetic complexity of the complete TDAC analysis/synthesis MDCT filter banks implementation can be further reduced/optimized as well as their corresponding computational structures can be represented in more compact form.

5.5.1 *Fast Analysis MDCT (MLT) Filter Bank*

Consider the TDAC analysis MDCT filter bank given by (5.4). The fast TDAC analysis MDCT filter bank with incorporated windowing operation is defined as [31]

$$c_k^{(t)} = \sum_{n=0}^{\frac{N}{2}-1} y_n^{(t)} \cos\left[\frac{\pi}{4(N/2)}(2n+1)(2k+1)\right], \qquad k = 0, 1, \ldots, \frac{N}{2}-1, \tag{5.181}$$

where

$$\begin{aligned} y_{\frac{N}{4}+n}^{(t)} &= \quad w_n\, x_n^{(t)} - w_{\frac{N}{2}-1-n}\, x_{\frac{N}{2}-1-n}^{(t)}, \\ y_{\frac{N}{4}-1-n}^{(t)} &= -w_{\frac{N}{2}+n}\, x_{\frac{N}{2}+n}^{(t)} - w_{N-1-n}\, x_{N-1-n}^{(t)}, \quad n = 0, 1, \ldots, \frac{N}{4}-1. \end{aligned} \tag{5.182}$$

The cosine transform kernel in (5.181) is the $\frac{N}{2}$-point forward DCT-IV of $\{y_n^{(t)}\}$. $\{w_n\}$ in (5.182) is generally a symmetric windowing function satisfying the perfect reconstruction conditions given by (5.7). MP3 audio encoder/decoder adopted

the sine windowing function given by (5.8). Thus, (5.181) and (5.182) for the sine windowing function define the fast analysis MLT filter bank [5, 6]. We note that (5.182) with the incorporated windowing operation includes also the permutation defined by (5.47).

5.5.2 Fast Synthesis MDCT (MLT) Filter Bank

Now consider the TDAC synthesis MDCT filter bank given by (5.5). Since the DCT-IV matrix is self-inverse, the data sequence $\{y_n^{(t)}\}$ is recovered by the inverse $\frac{N}{2}$-point DCT-IV of $\{c_k^{(t)}\}$ as [31]

$$y_n^{(t)} = \sum_{k=0}^{\frac{N}{2}-1} c_k^{(t)} \cos\left[\frac{\pi}{4(N/2)}(2n+1)(2k+1)\right], \quad n = 0, 1, \ldots, \frac{N}{2}-1. \tag{5.183}$$

Equation (5.183) defines the fast TDAC synthesis MDCT filter bank, or equivalently, the fast synthesis MLT filter bank with the sine windowing function [5, 6]. The time domain aliased data sequence $\{\hat{x}_n^{(t)}\}$ is obtained from $\{y_n^{(t)}\}$ by applying the inverse permutation to (5.182) as

$$\begin{aligned}
\hat{x}_n^{(t)} &= \quad w_n \, y_{\frac{N}{4}+n}^{(t)}, \\
\hat{x}_{\frac{N}{2}-1-n}^{(t)} &= -\, w_{\frac{N}{2}-1-n} \, y_{\frac{N}{4}+n}^{(t)}, \\
\hat{x}_{\frac{N}{2}+n}^{(t)} &= -\, w_{\frac{N}{2}+n} \, y_{\frac{N}{4}-1-n}^{(t)} \quad = -\, w_{\frac{N}{2}-1-n} \, y_{\frac{N}{4}-1-n}^{(t)}, \\
\hat{x}_{N-1-n}^{(t)} &= -\, w_{N-1-n} \, y_{\frac{N}{4}-1-n}^{(t)} = -\, w_n \, y_{\frac{N}{4}-1-n}^{(t)}, \\
& n = 0, 1, \ldots, \frac{N}{4} - 1.
\end{aligned} \tag{5.184}$$

Equations (5.183) and (5.184) define the fast synthesis MDCT filter banks. The fast DCT-IV computational structures for the efficient implementations of forward/backward MDCT (MLT) block transforms in MP3 are presented in Sect. 5.4.3.

The windowing&overlap procedure in the fast TDAC analysis MDCT filter bank and windowing&overlap&add procedure given by (5.6) in the fast TDAC synthesis MDCT filter bank can be represented by regular cascades of Givens–Jacobi rotations (see Appendix F.1), or alternatively, by regular cascades of 2-point Hankel matrix-vector products (see Appendix A.2) as follows.

5.5.3 *Efficient Implementation of the Windowing&Overlap Procedure*

In the TDAC analysis MDCT filter bank between two adjacent overlapped data blocks $\{x_n^{(t)}\}$ and $\{x_n^{(t+1)}\}$ the following relations hold:

$$x^{(t)}_{\frac{N}{2}+n} = x^{(t+1)}_{n}, \qquad x^{(t)}_{N-1-n} = x^{(t+1)}_{\frac{N}{2}-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{5.185}$$

Using (5.185) and the symmetry property of the sine windowing function, and denoting $c_n = \cos\frac{\pi(2n+1)}{2N}$, $s_n = \sin\frac{\pi(2n+1)}{2N}$, Eq. (5.182) is rewritten as

$$\begin{aligned} y^{(t)}_{\frac{N}{4}+n} &= \quad s_n\, x^{(t)}_{n} \quad - c_n\, x^{(t)}_{\frac{N}{2}-1-n}, \\ y^{(t)}_{\frac{N}{4}-1-n} &= -c_n\, x^{(t+1)}_{n} - s_n\, x^{(t+1)}_{\frac{N}{2}-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{5.186}$$

For clarity, in processing the succeeding overlapped data blocks $0, 1, \ldots, t,\ t+1,\ t+2, \ldots$, the associated $\frac{N}{4}$-point data sub-sequences $\{y^{(.)}_{\frac{N}{4}+n}\}$ and $\{y^{(.)}_{\frac{N}{4}-1-n}\}$ are, respectively, given by

$$\begin{aligned} y^{(0)}_{\frac{N}{4}+n} &= 0, && \text{(initialization step)}, \\ \hline y^{(0)}_{\frac{N}{4}-1-n} &= -c_n\, x^{(1)}_{n} - s_n\, x^{(1)}_{\frac{N}{2}-1-n}, & & \\ y^{(1)}_{\frac{N}{4}+n} &= \quad s_n\, x^{(1)}_{n} - c_n\, x^{(1)}_{\frac{N}{2}-1-n}, & & \{y^{(1)}_{\frac{N}{4}+n}\} \text{ is } \frac{N}{4}\text{-sample delayed data sub-block}, \\ \hline &\vdots & & \\ \\ y^{(t)}_{\frac{N}{4}+n} &= \quad s_n\, x^{(t)}_{n} - c_n\, x^{(t)}_{\frac{N}{2}-1-n}, & & \\ \hline y^{(t)}_{\frac{N}{4}-1-n} &= -c_n\, x^{(t+1)}_{n} - s_n\, x^{(t+1)}_{\frac{N}{2}-1-n}, & & \\ y^{(t+1)}_{\frac{N}{4}+n} &= \quad s_n\, x^{(t+1)}_{n} - c_n\, x^{(t+1)}_{\frac{N}{2}-1-n}, & & \{y^{(t+1)}_{\frac{N}{4}+n}\} \text{ is } \frac{N}{4}\text{-sample delayed data sub-block}, \\ \hline y^{(t+1)}_{\frac{N}{4}-1-n} &= -c_n\, x^{(t+2)}_{n} - s_n\, x^{(t+2)}_{\frac{N}{2}-1-n}, & & \\ y^{(t+2)}_{\frac{N}{4}+n} &= \quad s_n\, x^{(t+2)}_{n} - c_n\, x^{(t+2)}_{\frac{N}{2}-1-n}, & & y^{(t+2)}_{\frac{N}{4}+n} \text{ is } \frac{N}{4}\text{-sample delayed data sub-block}, \\ \hline &\vdots & & \end{aligned} \tag{5.187}$$

By introducing the $\frac{N}{4}$-sample delay operator denoted by $\Delta_{\frac{N}{4}}\{.\}$, the windowing&overlap procedure in the fast analysis MLT filter bank can be represented in the matrix-vector form as

$$\begin{pmatrix} y^{(t)}_{\frac{N}{4}-1-n} \\ \\ \Delta_{\frac{N}{4}}\left\{y^{(t+1)}_{\frac{N}{4}+n}\right\} \end{pmatrix} = -\begin{pmatrix} c_n & s_n \\ \\ -s_n & c_n \end{pmatrix} \begin{pmatrix} x^{(t+1)}_{n} \\ \\ x^{(t+1)}_{\frac{N}{2}-1-n} \end{pmatrix}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{5.188}$$

or alternatively, as

$$\begin{pmatrix} y^{(t)}_{\frac{N}{4}-1-n} \\ \\ \Delta_{\frac{N}{4}}\left\{y^{(t+1)}_{\frac{N}{4}+n}\right\} \end{pmatrix} = \begin{pmatrix} -s_n & -c_n \\ \\ -c_n & s_n \end{pmatrix} \begin{pmatrix} x^{(t+1)}_{\frac{N}{2}-1-n} \\ \\ x^{(t+1)}_{n} \end{pmatrix}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{5.189}$$

with initial condition $y^{(0)}_{\frac{N}{4}+n} = 0$. Equation (5.188) represents the block of $\frac{N}{4}$ Givens–Jacobi rotations, while (5.189) represents the block of $\frac{N}{4}$ (2-point) Hankel matrix-vector products. Efficient implementations of Givens–Jacobi rotations and 2-point Hankel matrix-vector products are presented in Appendices F.2 and F.3. The windowing&overlap procedure including the permutation (5.47) requires $3\frac{N}{4}$ multiplications and $3\frac{N}{4}$ additions. Two corresponding compact computational structures of the fast analysis MLT filter bank with incorporated windowing&overlap procedure for the MP3 encoder are shown in Fig. 5.25a, b.

5.5.4 *Efficient Implementation of the Windowing&Overlap&Add Procedure*

Given two succeeding blocks of transform coefficients $\{c^{(t)}_k\}$ and $\{c^{(t+1)}_k\}$. After transforming the transform coefficients by the backward MLT (MDCT) block transform realized by the $\frac{N}{2}$-point DCT-IV, using the symmetry property of sine windowing function, and denoting $c_n = \cos\frac{\pi(2n+1)}{2N}$, $s_n = \sin\frac{\pi(2n+1)}{2N}$, from (5.184) it follows that the time domain aliased data sequences $\{\hat{x}^{(t)}_n\}$ and $\{\hat{x}^{(t+1)}_n\}$ are, respectively, given by

$$\begin{aligned} \hat{x}^{(t)}_{\frac{N}{2}+n} &= -c_n\, y^{(t)}_{\frac{N}{4}-1-n}, \\ \hat{x}^{(t)}_{N-1-n} &= -s_n\, y^{(t)}_{\frac{N}{4}-1-n}, \end{aligned}$$

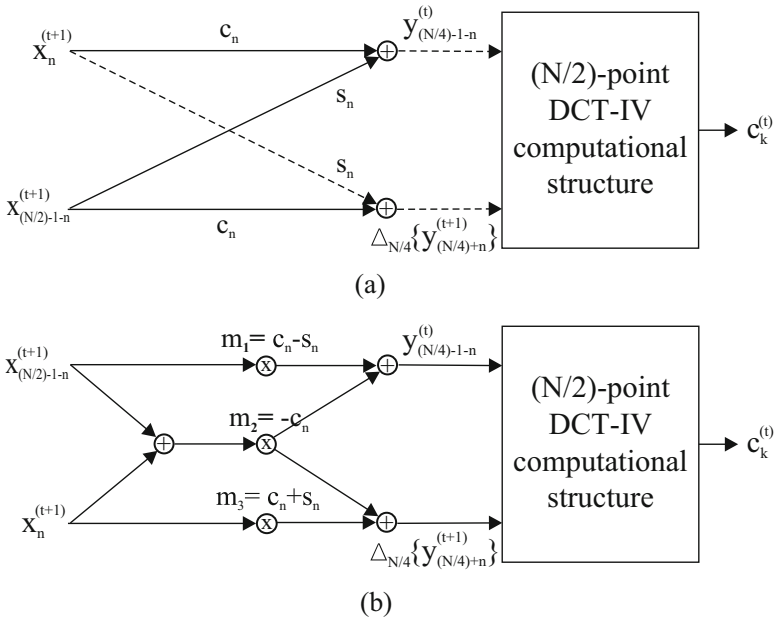

Fig. 5.25 Compact computational structures of the fast analysis MLT filter bank with incorporated windowing&overlap procedure represented by (**a**) Givens–Jacobi rotations, (**b**) 2-point Hankel matrix-vector products

$$\begin{aligned} \hat{x}_n^{(t+1)} &= s_n\, y_{\frac{N}{4}+n}^{(t+1)}, \\ \hat{x}_{\frac{N}{2}-1-n}^{(t+1)} &= -c_n\, y_{\frac{N}{4}+n}^{(t+1)}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{5.190}$$

For two adjacent time domain aliased data blocks $\{\hat{x}_n^{(t)}\}$ and $\{\hat{x}_n^{(t+1)}\}$ recovered by the TDAC synthesis MDCT filter bank, the following relations hold in the overlapped part:

$$x_n^{(t+1)} = \hat{x}_{\frac{N}{2}+n}^{(t)} + \hat{x}_n^{(t+1)}, \qquad x_{\frac{N}{2}-1-n}^{(t+1)} = \hat{x}_{N-1-n}^{(t)} + \hat{x}_{\frac{N}{2}-1-n}^{(t+1)}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{5.191}$$

In order to perfectly reconstruct the original data sequence $\{x_n^{(t+1)}\}$ in the overlapped part, let us substitute the appropriate time domain aliased data sequences $\{\hat{x}_n^{(t)}\}$ from (5.190) into (5.191). Using $\frac{N}{4}$-sample delay operator $\Delta_{\frac{N}{4}}\{.\}$ the windowing&overlap&add procedure in the fast synthesis MLT filter bank is expressed as

$$x_n^{(t+1)} = -c_n\, \Delta_{\frac{N}{4}}\{y_{\frac{N}{4}-1-n}^{(t)}\} + s_n\, y_{\frac{N}{4}+n}^{(t+1)},$$

$$x_{\frac{N}{2}-1-n}^{(t+1)} = -s_n \, \Delta_{\frac{N}{4}}\{y_{\frac{N}{4}-1-n}^{(t)}\} - c_n \, y_{\frac{N}{4}+n}^{(t+1)}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \quad (5.192)$$

Equation (5.192) can be represented in the matrix-vector form as

$$\begin{pmatrix} x_n^{(t+1)} \\ \\ x_{\frac{N}{2}-1-n}^{(t+1)} \end{pmatrix} = - \begin{pmatrix} c_n & -s_n \\ \\ s_n & c_n \end{pmatrix} \begin{pmatrix} \Delta_{\frac{N}{4}}\{y_{\frac{N}{4}-1-n}^{(t)}\} \\ \\ y_{\frac{N}{4}+n}^{(t+1)} \end{pmatrix}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1, \quad (5.193)$$

or alternatively, as

$$\begin{pmatrix} x_n^{(t+1)} \\ \\ x_{\frac{N}{2}-1-n}^{(t+1)} \end{pmatrix} = \begin{pmatrix} s_n & -c_n \\ \\ -c_n & -s_n \end{pmatrix} \begin{pmatrix} y_{\frac{N}{4}+n}^{(t+1)} \\ \\ \Delta_{\frac{N}{4}}\{y_{\frac{N}{4}-1-n}^{(t)}\} \end{pmatrix}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1. \quad (5.194)$$

Again, (5.193) represents the block of $\frac{N}{4}$ Givens–Jacobi rotations, while (5.194) represents the block of $\frac{N}{4}$ (2-point) Hankel matrix-vector products. The windowing&overlap&add procedure requires $3\frac{N}{4}$ multiplications and $3\frac{N}{4}$ additions. Two corresponding compact computational structures of the fast synthesis MLT filter bank with incorporated windowing&overlap&add procedure are shown in Fig. 5.26a, b.

The efficient implementation of complete analysis and synthesis MLT (MDCT) filter banks consists of the $\frac{N}{2}$-point identical fast DCT-IV computational structure, windowing&overlap and windowing&overlap&add procedures. If we adopt the fast $\frac{N}{2}$-point DCT-IV computational structures [38, 40] (requiring 9 multiplications, 21 additions plus 2 shifts for the short block, and 36 multiplications, 130 additions plus 2 shifts for the long block), then the complete analysis/synthesis MLT (MDCT) filter banks in MP3 will require totally 18 multiplications, 30 additions plus 2 shifts for the short audio block, and 63 multiplications, 157 additions plus 2 shifts for the long audio block.

Note 22: Similar compact computational structures of the fast TDAC analysis MDCT filter bank with incorporated windowing&overlap procedure, and of the fast TDAC synthesis MDCT filter bank with incorporated windowing&overlap&add procedure for arbitrary symmetric windowing function $\{w_n\}$ can be easily constructed as follows.

Consider the fast TDAC analysis MDCT filter bank with incorporated windowing&overlap procedure defined by (5.181) and (5.182) for arbitrary symmetric windowing function. Using (5.185) and the symmetry property of windowing function given by (5.7), (5.182) is rewritten as

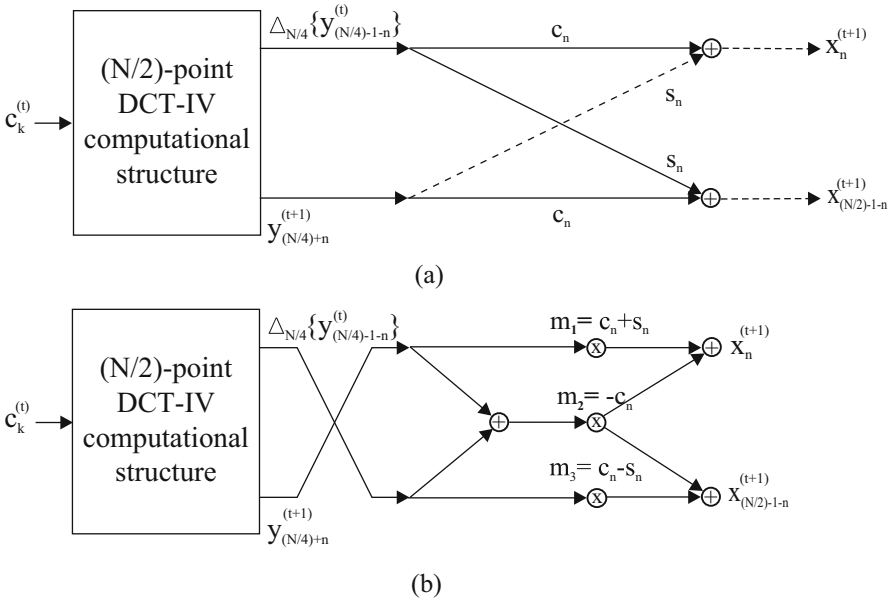

Fig. 5.26 Compact computational structures of the fast synthesis MLT filter bank with incorporated windowing&overlap&add procedure represented by (**a**) Givens–Jacobi rotations, (**b**) 2-point Hankel matrix-vector products

$$
\begin{aligned}
y^{(t)}_{\frac{N}{4}+n} &= \quad w_n\, x^{(t)}_n \quad - \; w_{\frac{N}{2}-1-n}\, x^{(t)}_{\frac{N}{2}-1-n}, \\
y^{(t)}_{\frac{N}{4}-1-n} &= -\, w_{\frac{N}{2}-1-n}\, x^{(t+1)}_n - \; w_n\, x^{(t+1)}_{\frac{N}{2}-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1.
\end{aligned}
\tag{5.195}
$$

Following the approach described in Sect. 5.5.3 the windowing&overlap procedure is represented in the following matrix-vector product

$$
\begin{pmatrix} y^{(t)}_{\frac{N}{4}-1-n} \\ \\ \Delta_{\frac{N}{4}}\{y^{(t+1)}_{\frac{N}{4}+n}\} \end{pmatrix} = \begin{pmatrix} -w_n & -w_{\frac{N}{2}-1-n} \\ \\ -w_{\frac{N}{2}-1-n} & w_n \end{pmatrix} \begin{pmatrix} x^{(t+1)}_{\frac{N}{2}-1-n} \\ \\ x^{(t+1)}_n \end{pmatrix},
$$

$$
n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{5.196}
$$

with initial condition $y^{(0)}_{\frac{N}{4}+n} = 0$. Equation (5.196) represents 2-point Hankel matrix-vector product whose efficient implementation via the bilinear algorithm is presented in Appendix A.2. The corresponding compact computational structure of the fast TDAC analysis MDCT filter bank with incorporated windowing&overlap

procedure for arbitrary symmetric windowing function $\{w_n\}$ can be obtained from Fig. 5.25b by setting constants m_1, m_2 and m_3 to the following values

$$m_1 = -w_n + w_{\frac{N}{2}-1-n}, \quad m_2 = -w_{\frac{N}{2}-1-n}, \quad m_3 = w_n - w_{\frac{N}{2}-1-n}. \quad (5.197)$$

Now, consider the fast TDAC synthesis MDCT filter bank defined by (5.183) and (5.184) with incorporated windowing&overlap&add procedure for arbitrary symmetric windowing function. At first, deriving the time domain aliased data sequences $\{\hat{x}_n^{(t)}\}$ and $\{\hat{x}_n^{(t+1)}\}$ from (5.184), and then following the approach described in Sect. 5.5.4, the windowing&overlap&add procedure is represented in the following matrix-vector product

$$\begin{pmatrix} x_n^{(t+1)} \\ \\ x_{\frac{N}{2}-1-n}^{(t+1)} \end{pmatrix} = \begin{pmatrix} w_n & -w_{\frac{N}{2}-1-n} \\ \\ -w_{\frac{N}{2}-1-n} & -w_n \end{pmatrix} \begin{pmatrix} y_{\frac{N}{4}+n}^{(t+1)} \\ \\ \Delta_{\frac{N}{4}}\{y_{\frac{N}{4}-1-n}^{(t)}\} \end{pmatrix},$$

$$n = 0, 1, \ldots, \frac{N}{4} - 1. \quad (5.198)$$

Again, (5.198) represents the 2-point Hankel matrix-vector product. The corresponding compact computational structure of the fast TDAC synthesis MDCT filter bank with incorporated windowing&overlap&add procedure for arbitrary symmetric windowing function $\{w_n\}$ can be obtained from Fig. 5.26b by setting constants m_1, m_2 and m_3 to the following values

$$m_1 = w_n + w_{\frac{N}{2}-1-n}, \quad m_2 = -w_{\frac{N}{2}-1-n}, \quad m_3 = -w_n + w_{\frac{N}{2}-1-n}. \quad (5.199)$$

5.6 Summary

Various efficient implementations of the forward and backward MDCT (MLT) block transforms in the MP3 audio coding standard developed in the time period 1990–2012 have been described and compared, including the efficient implementation of (pseudo-) QMF filter banks for completeness. The efficient MDCT (MLT) implementations have been discussed in the context of complete (fast) TDAC analysis/synthesis MDCT filter banks associated with the sine windowing function or equivalently, of the fast analysis/synthesis MLT filter banks in the MP3 encoder and decoder. For each efficient implementation of the forward/backward MDCT (MLT), block transforms have been presented: complete formulae or sparse matrix factorizations, the corresponding signal flow graph for short audio block, and the total arithmetic complexity as well as useful comments related to improving the arithmetic complexity and a possible structural simplification of the algorithm. Finally, the fast TDAC analysis/synthesis MDCT filter banks or equivalently, the fast

analysis/synthesis MLT filter banks for MP3 encoder/decoder have been discussed in detail. Appendices provide all the necessary supporting efficient optimized short-length computational modules and tools for completing efficient forward/backward MDCT (MLT) implementations. Essentially, the chapter provides:

- Efficient implementations of the hybrid analysis/synthesis filter banks in MP3 audio coding standard.
- History, theoretical background of the developed fast MDCT (MLT) algorithms, the relations among them (particularly among fast algorithms in different categories), as well as the relations among discrete sinusoidal unitary transforms of reduced sizes.

All the presented fast MDCT (MLT) algorithms can also be readily used for the 2^n-length data blocks. We hope that the chapter will serve as an excellent reference in designing not only MP3 audio encoder/decoder but in general, any audio encoder/decoder as well as it will be an inspiration to further advanced research.

Problems and Exercises

1. Investigating (5.19), (5.20), and (5.22) for the efficient implementation of backward pseudo-QMF block transform given by (5.2), derive the local symmetry properties of time domain aliased data sequence $\{\hat{x}_n\}$, in general, for any N divisible by 4.
2. Implement by a computer program the fast algorithm for forward pseudo-QMF block transform computation defined by (5.15), (5.17), and (5.18) for any N divisible by 4. Determine its arithmetic complexity. Fast recursive DCT-III computational structure is presented in Appendix C.1.1 or it is ready for using in [116].
3. Implement by a computer program the fast algorithm for backward pseudo-QMF block transform computation defined by (5.19), (5.20), and (5.22) for any N divisible by 4. Similarly, determine its arithmetic complexity. Fast recursive DCT-II computational structure is presented in Appendix C.1.1 or it is ready for using in [116].
4. Implement by a computer program the complete synthesis pseudo-QMF filter bank with windowing operation for MP3 decoder including a recursive optimized algorithm [10, 11]. Then investigate its computational complexity. How many arithmetic operations can be saved compared to the direct approach without applying the recursive optimized algorithm?
5. Verify the equivalence of permutations (5.36) and (5.47) for $N = 8$ and 16.
6. In this chapter the efficient MDCT (MLT) implementations in MP3 (including improved and modified) based on the discrete sinusoidal unitary transforms (DFT, DCT, and DST) are classified into the following categories:

- DFT/FFT-based (Sect. 5.4.1),
- DCT-II/DST-II-based (Sect. 5.4.2),
- DCT-IV-based (Sect. 5.4.3),
- DCT-IV/(S)DCT-II-based (Sect. 5.4.4).

The corresponding signal flow graphs are presented for the short audio block only ($N = 12$). At first, verify their correctness by computer programs. Then, investigating each algorithm in each category for the forward/backward MDCT (MLT) computation draw the corresponding signal flow graph for long audio block ($N = 36$). Similarly, verify their correctness by computer programs.

7. Following the previous Exercise 6 repeat the same problems for efficient MDCT (MLT) implementations in MP3 (including improved and modified):

   - Based on the corresponding evenly stacked MDCT (Sect. 5.4.5),
   - Mixed-radix MDCT algorithms (Sect. 5.4.6).

8. Implement by a computer program the most efficient DCT-IV-based MDCT algorithms in terms of the minimal multiplicative complexity [38, 40] tailored to MP3 described in Sect. 5.4.3 for the forward/backward MDCT block transforms computation both for the short and long audio blocks.
9. For the fast DCT-IV-based MDCT algorithm [29] described in Sect. 5.4.3, the most efficient MDCT implementation for the short audio block achieving the minimal multiplicative complexity (nine multiplications) has been derived (see (5.90) and (5.91) in Note 5.4.3.8). Consider (5.47), (5.86), and (5.88). Extending the terms $\{a_n\}$ and $\{b_n\}$ in (5.88) according to (5.86), and then using the following trigonometric identities

$$\cos\frac{\pi}{24}\cos\frac{\pi}{6} = \frac{1}{2}\left[\cos\frac{\pi}{8} + \cos\frac{5\pi}{24}\right], \qquad \cos\frac{\pi}{24}\sin\frac{\pi}{6} = \frac{1}{2}\left[\sin\frac{\pi}{8} + \sin\frac{5\pi}{24}\right],$$

$$\cos\frac{5\pi}{24}\cos\frac{\pi}{6} = \frac{1}{2}\left[\cos\frac{\pi}{24} + \sin\frac{\pi}{8}\right], \qquad \cos\frac{5\pi}{24}\sin\frac{\pi}{6} = \frac{1}{2}\left[-\sin\frac{\pi}{24} + \cos\frac{\pi}{8}\right],$$

$$\sin\frac{\pi}{24}\cos\frac{\pi}{6} = \frac{1}{2}\left[-\sin\frac{\pi}{8} + \sin\frac{5\pi}{24}\right], \qquad \sin\frac{\pi}{24}\sin\frac{\pi}{6} = \frac{1}{2}\left[\cos\frac{\pi}{8} - \cos\frac{5\pi}{24}\right],$$

$$\sin\frac{5\pi}{24}\cos\frac{\pi}{6} = \frac{1}{2}\left[\sin\frac{\pi}{24} + \cos\frac{\pi}{8}\right], \qquad \sin\frac{5\pi}{24}\sin\frac{\pi}{6} = \frac{1}{2}\left[\cos\frac{\pi}{24} - \sin\frac{\pi}{8}\right],$$

and

$$\cos\frac{\pi}{6}\cos\frac{\pi}{8} = \frac{1}{2}\left[\cos\frac{\pi}{24} + \sin\frac{5\pi}{24}\right], \qquad \sin\frac{\pi}{6}\cos\frac{\pi}{8} = \frac{1}{2}\left[\sin\frac{\pi}{24} + \cos\frac{5\pi}{24}\right],$$

$$\cos\frac{\pi}{6}\sin\frac{\pi}{8} = \frac{1}{2}\left[-\sin\frac{\pi}{24} + \cos\frac{5\pi}{24}\right], \qquad \sin\frac{\pi}{6}\sin\frac{\pi}{8} = \frac{1}{2}\left[\cos\frac{\pi}{24} - \sin\frac{5\pi}{24}\right],$$

derive an alternative efficient DCT-IV-based MDCT implementation for the short audio block achieving the minimal multiplicative complexity. Draw the corresponding modified fast 6-point DCT-IV computational structure.

10. As is indicated by Note 5.4.3.9 in Sect. 5.4.3 for the fast DCT-IV-based MDCT algorithm [29], the open problem remains to derive a modified 18-point fast DCT-IV computational structure with the minimal multiplicative complexity (36 multiplications) for the long audio block. Consider (5.47), (5.86), and (5.92). Try to derive a modified 18-point fast DCT-IV computational structure with the minimal multiplicative complexity for the long audio block. We note that this problem is classified as a difficult one. Hint: At first, extract from (5.92) for the pairs of coefficients c_{2k-1}, c_{2k} and c_{17-2k}, c_{18-2k}, $k = 1, 2, 3, 4$, the unique algebraic expressions containing the terms $\{a_n\}$ and $\{b_n\}$, $n = 1, 4$ and 7 extended according to (5.86). Then follow the approach indicated in Problem 8. For the remaining unique algebraic expressions, try to complete the modified 18-point fast DCT-IV computational structure.
11. Implement by a computer program the DCT-IV/SDCT-II-based MDCT algorithm [66] described in Sect. 5.4.4 also tailored to MP3 for the computation of forward/backward MDCT block transforms: (a) For the short audio block defined by (5.47), (5.48), (5.116), (5.117), (5.123), and (5.125), and (b) For the long audio block defined by (5.47), (5.48), (5.116), (5.117), (5.127), (5.130), (5.132), (5.134), (5.136), and (5.139).
12. Consider again the DCT-IV/SDCT-II-based MDCT algorithm [66] described in Sect. 5.4.4 for the short audio block. Based on (5.121) the computation of odd-indexed SDCT-II coefficients c_1^{II}, c_3^{II}, and c_5^{II} can be derived alternatively with the same computational complexity using the following trigonometric identities:

$$\sqrt{2}\,\cos\frac{\pi}{12} = \cos\frac{\pi}{6} + \cos\frac{\pi}{3}, \qquad \sqrt{2}\,\cos\frac{5\pi}{12} = \cos\frac{\pi}{6} - \cos\frac{\pi}{3}.$$

    Derive the alternative computation of odd-indexed SDCT-II coefficients c_1^{II}, c_3^{II}, and c_5^{II}.
13. The most efficient MDCT implementations among all mixed-radix MDCT algorithms are based on the combined DIF radix-2 and DIT mixed-radix fast MDCT algorithms [44, 67] described in Sect. 5.4.6. Implement by a computer program the 12-point MDCT (see signal flow graph shown in Fig. 5.21) and then 36-point MDCT for MP3. The required optimized efficient 6/3-point MDCT modules are presented in Appendix E.2 while the optimized efficient 18/9-point MDCT modules are presented in Appendix E.3.
14. By computer programs verify the correctness of forward and backward MDCT implementations based on recursive/regressive filter structures described in Sect. 5.4.7.
15. For MP3 encoder implement by a computer program compact DCT-IV computational structures of the fast analysis MLT filter bank with incorporated windowing&overlap procedure represented by Givens–Jacobi rotations (see Fig. 5.25a), and then represented by 2-point Hankel matrix-vector products (see Fig. 5.25b).
16. For MP3 decoder implement by a computer program compact DCT-IV computational structures of the fast synthesis MLT filter bank with incorporated windowing&overlap&add procedure represented by Givens–Jacobi rotations

(see Fig. 5.26a), and then represented by 2-point Hankel matrix-vector products (see Fig. 5.26b).

17. For any MDCT-based audio encoder based on (5.196) and (5.197), implement by a computer program compact DCT-IV computational structure of the fast analysis MDCT filter bank with incorporated windowing&overlap procedure represented by 2-point Hankel matrix-vector products for arbitrary symmetric windowing function.
18. Similarly, for any MDCT-based audio decoder based on (5.198) and (5.199), implement by a computer program compact DCT-IV computational structure of the fast synthesis MDCT filter bank with incorporated windowing&overlap&add procedure represented by 2-point Hankel matrix-vector products for arbitrary symmetric windowing function.

References

1. M. Bosi, R.E. Goldberg, *Introduction to Digital Audio Coding and Standards* (Springer, New York, 2003), Chapters 11 and 12, pp. 265–332
2. Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s, Part 3: Audio, ISO/IEC JTC1/SC29/WG11 MPEG, International Standard 11172-3 (MPEG-1), 1992
3. Information Technology – Generic Coding of Moving Pictures and Associated Audio, Part 3: Audio. ISO/IEC JTC1/SC29/WG11 MPEG, International Standard 13818-3 (MPEG-2), 1994
4. A.W. Johnson, A.B. Bradley, Adaptive transform coding incorporating time domain aliasing cancellation. Speech Comm. **6**, 299–308 (1987)
5. H.S. Malvar, Lapped transforms for efficient transform/sub-band coding. IEEE Trans. Acoust. Speech Signal Process. **38**(6), 969–978 (1990)
6. H.S. Malvar, *Signal Processing with Lapped Transforms* (Artech House, Norwood, MA, 1992)
7. J.P. Princen, A.W. Johnson, A.B. Bradley, Sub-band/transform coding using filter bank designs based on time domain aliasing cancellation, in *Proceedings of the IEEE ICASSP'87*, Dallas, TX, April 1987, pp. 2161–2164
8. K.R. Rao, J.J. Hwang, *Techniques and Standards for Image, Video, and Audio Coding* (Prentice Hall, Upper Saddle River, NJ, 1996), Chapter 10, pp. 242–272

Fast Analysis/Synthesis OMF (Pseudo-QMF) Filter Banks

9. D.-Y. Chan, J.-F. Yang and C.-C. Fang, Fast implementation of MPEG audio coder using recursive formula with fast discrete cosine transforms. IEEE Trans. Speech Audio Process. **4**(2), 144–148 (1996)
10. P. De Smet, I. Bruyland, Optimized MPEG audio decoding using recursive sub-band synthesis windowing, in *Proc. of the IEEE ICASSP'2002*, Vol. 3, Orlando, FL, May 2002, pp. 3160–3163
11. P. De Smet, I. Bruyland, Optimized recursive subband synthesis windowing for implementing efficient MPEG audio decoders. IEEE Signal Process Lett. **10**(10), 303–306 (2003)
12. W. Jianxin, D. Zaiwang, A fast algorithm for modified discrete cosine transform, in *Proc. of the IEEE International Conference on Communication Technology (ICCT'96)*, vol. 1, Beijing, China, May 1996, pp. 445–448

13. K. Konstantinides, Fast subband filtering in MPEG audio coding. IEEE Signal Process Lett. **1**(2), 26–28 (1994)
14. K.W. Law, K.C.K. Lee, Digital filter for sub-band synthesis. U. S. Patent #6,917,913, Motorola, Inc., Schaumburg, IL, July 2005
15. C.D. Murphy, K. Anandakumar, Real-time MPEG-1 audio coding and decoding on a DSP chip. IEEE Trans. Consum. Electron. **43**(1), 40–47 (1997)
16. D. Pan, Digital audio compression. Digit. Tech. J. **5**(2), 28–40 (1993)
17. D. Pan, A tutorial on MPEG/audio compression. IEEE Multimedia **2**, 60–74 (1995)
18. S.-H. Park, Y.-B. Kim, S.-W. Kim, Y.-S. Seo, Fast algorithm on MPEG/audio subband filtering, in *99th AES Convention*, New York, October 1995, Preprint #4090
19. S.-C. Tai, C.-C. Wang, C.-Y. Lin, FFT and IMDCT circuit sharing in DAB receiver. IEEE Trans. Broadcast. **49**(2), 124–131 (2003)

Fast MLT (MDCT) Algorithms

20. M.J. Absar, S. George, Fast modified discrete cosine transform method, US Patent Application WO 01/33411 A1, ST Microelectronics Asia Pacific, Singapore, May 2001
21. K.C. Anup, B.A. Kumar, A new efficient implementation of TDAC synthesis filterbanks based on radix-2 FFT, in *Proceedings of the 14th EUSIPCO Signal Processing Conference*, Florence, Italy, September 2006
22. G. Baum, Method, apparatus, and computer program for performing a modified discrete cosine transform, Patent Application #EP1445706, Thomson Brandt Gmbh, Germany, August 2004
23. G. Baum, Method, circuit, and computer program product for performing a modified discrete cosine transform, US Patent Application #2006003666, Thomson Licensing, Boulogne-Billancourt, France, August 2004
24. V. Britanak, The refined efficient implementation of the MDCT in MP3 and comparison with other methods, Technical Report II SAS-2002-2, Department of Numerical Methods and Algorithms, Institute of Informatics, Slovak Academy of Sciences, Bratislava, September 2002
25. V. Britanak, The fast DCT-IV/DST-IV computation via the MDCT. Signal Process. **83**(8), 1803–1813 (2003)
26. V. Britanak, An efficient computing of oddly stacked MDCT/MDST via evenly stacked MDCT/MDST and vice versa. Signal Process. **85**(7), 1353–1374 (2005)
27. V. Britanak, New universal rotation-based fast computational structures for an efficient implementation of the DCT-IV/DST-IV and analysis/synthesis MDCT/MDST filter banks. Signal Process. **89**(11), 2213–2232 (2009)
28. V. Britanak, Improved and extended mixed-radix decimation in frequency fast MDCT algorithm. Comput. Informatics **29**(6), 1001–1012 (2010)
29. V. Britanak, New efficient implementations of the forward/backward MDCT in MP3 audio coding standard. Signal Process. **90**(2), 536–547 (2010)
30. V. Britanak, A survey of efficient MDCT implementations in MP3 audio coding standard: retrospective and state of the art. Signal Process. **91**(4), 624–672 (2011)
31. V. Britanak, H.J. Lincklaen Arriëns, Fast computational structures for an efficient implementation of the complete TDAC analysis/synthesis MDCT/MDST filter banks. Signal Process. **89**(7), 1379–1394 (2009)
32. V. Britanak, K.R. Rao, An efficient implementation of the forward and inverse MDCT in MPEG audio coding. IEEE Signal Process Lett. **8**(2), 48–51 (2001) (Erratum, IEEE Signal Processing Lett. **8**(10), 279 (2001))
33. V. Britanak, K.R. Rao, A new fast algorithm for the unified forward and inverse MDCT/MDST computation. Signal Process. **82**(3), 433–459 (2002)
34. D.-Y. Chan, J.-F. Yang, S.-Y. Chen, Regular implementation algorithms of time domain aliasing cancellation. IEE Proc. Vision Image Signal Process. **143**(6), 387–392 (1996)

35. M.-H. Cheng, Y.-H. Hsu, Fast IMDCT and MDCT algorithms – a matrix approach. IEEE Trans. Signal Process. **51**(1), 221–229 (2003)
36. Y.-K. Cho, T.-H. Song, H.-S. Kim, An optimized algorithm for computing the modified discrete cosine transform and its inverse transform, in *Proceedings of the IEEE Region 10 Conference TENCON'2004*, Chiang Mai, Thailand, November 2004, pp. 626–628
37. S. Cramer, R. Gluth, Computationally efficient real-valued filter banks based on a modified O^2 DFT, in *Proceedings of EUSIPCO'90, Signal Processing V: Theories and Applications* (Elsevier, Barcelona, 1990), pp. 585–588
38. X. Dai, M.D. Wagh, An MDCT hardware accelerator for MP3 audio, in *Proceedings of the IEEE Symposium on Application Specific Processors (SASP'2008)*, Anaheim, CA, June 2008, pp. 121–125
39. X. Dai, M.D. Wagh, Fast algorithm for modulated complex lapped transform. IEEE Signal Process. Lett. **16**(1), 30–33 (2009)
40. X. Dai, M.D. Wagh, Bilinear algorithms and VLSI implementations of forward and inverse MDCT with applications to MP3 audio, U.S. Patent WO 2009/1000210 A2, August 2009
41. W. Dou, R. Liu, J. Wang, Z. Dong, One of the fast DSP-based inverse MDCT algorithm. J. Tsinghua Univ. (Sci&Tech) **40**(3), 99–103 (2000)
42. P. Duhamel, Y. Mahieux, J.P. Petit, A fast algorithm for the implementation of filter banks based on time domain aliasing cancellation, in *Proceedings of the IEEE ICASSP'91*, Toronto, Canada, May 1991, pp. 2209–2212
43. R. Gluth, Regular FFT-related transform kernels for DCT/DST-based polyphase filter banks, in *Proceedings of the IEEE ICASSP'91*, Toronto, Canada, May 1991, pp. 2205–2208
44. Z.G. Gui, Y. Ge, D.Y. Zhang, J.S. Wu, Generalized fast mixed-radix algorithm for the computation of forward and inverse MDCTs. Signal Process. **92**(2), 363–373 (2012)
45. Y.-T. Hwang, S.-C. Lai, A novel MDCT/IMDCT computing kernel design, in *Proceedings of the IEEE Workshop on Signal Processing Systems Design and Implementation*, Athens, Greece, November 2005, pp. 526–531
46. C.Y. Jing, H.-M. Tai, Fast algorithm for computing modulated lapped transform. Electronics Lett. **37**(12), 796–797 (2001)
47. C.Y. Jing, H.-M. Tai, Design and implementation of a fast algorithm for modulated lapped transform. IEE Proc. Vision Image Signal Process. **149**(1), 27–32 (2002)
48. H.-S. Kim, Y.-K. Cho, W.-P. Lee, A new optimized algorithm for computation of MDCT and its inverse transform, in *Proceedings of the IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS'2004)*, Seoul, Korea, November 2004, pp. 528–530
49. O. Lashko, Modulated lapped transform application in image coding and effective algorithm of it realization, in *Proceedings of the International Conference TCSET'2002*, Lviv-Slavsko, Ukraine, February 2002, pp. 243–244
50. S.-W. Lee, Improved algorithm for efficient computation of the forward and backward MDCT in MPEG audio coder. IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process. **48**(10), 990–994 (2001)
51. T. Li, R. Zhang, R. Yang, H. Huang, F. Lin, A unified computing kernel for MDCT/IMDCT in modern audio coding standards, in *Proceedings of the IEEE International Symposium on Communication and Information Technologies (ISCIT'2007)*, Sydney, Australia, October 2007, pp. 546–550
52. B. Lincoln, An experimental high fidelity perceptual audio coder. Project in MUS420 Win97, Center for Computer Research in Music and Acoustics, Stanford University, CA, 1998, web site: http://ccrma-www.stanford.edu/~bosse/proj/proj.html
53. C.-M. Liu, W.-C. Lee, A unified fast algorithm for cosine modulated filter banks in current audio coding standards. J. Audio Eng. Soc. **47**(12), 1061–1075 (1999)
54. H.S. Malvar, Fast algorithms for orthogonal and biorthogonal modulated lapped transforms, in *Proceedings of IEEE Symposium on Advances in Digital Filtering and Signal Processing*, Victoria, Canada, June 1998, pp. 159–163

55. S.B. Marovich, Faster MPEG-1 layer III audio decoding, Technical Report HPL-2000-66, Computer Systems and Technology Laboratory, HPL Laboratories, Palo Alto, June 2000
56. V. Nikolajevič, G. Fettweis, Improved implementation of MDCT in MP3 audio coding, in *Proceedings of the IEEE 10th Asian-Pacific Conference on Communications and 5th International Symposium on Multi-Dimensional Mobile Communications (APCC/MDMC'2004)*, vol. 1, Tsinghua University, Beijing, China, August-September 2004, pp. 309–312
57. N. Rama Murthy, M.N.S. Swamy, A parallel/pipeline algorithm for the computation of MDCT and IMDCT, in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'2003)*, vol. IV, Banghok, Thailand, May 2003, pp. 540–543
58. T. Sakamoto, M. Taruki, T. Hase, A fast MPEG-audio layer III algorithm for 32-Bit MCU. IEEE Trans. Consum. Electron. **45**(3), 986–993 (1999)
59. X. Shao, S.G. Johnson, Type-IV DCT, DST and MDCT algorithms with reduced number of arithmetic operations. Signal Process. **88**(6), 1313–1326 (2008)
60. H. Shu, X. Bao, C. Toumoulin, L. Luo, Radix-3 algorithm for the fast computation of forward and inverse MDCT. IEEE Signal Process Lett. **14**(2), 93–96 (2007)
61. D. Ševič, M. Popovič, A new efficient implementation of the oddly stacked Princen-Bradley filter bank. IEEE Signal Processing Lett. **1**(11), 166–168 (1994)
62. P. Singh, W. Moreno, N. Ranganathan, H. Neinhaus, A flexible MPEG audio decoder layer III chip architecture, in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'98)*, vol. IV, Monterey, CA, May 1998, pp. 37–40
63. T. Sporer, K. Brandenburg, B. Edler, The use of multirate filter banks for coding of high quality digital audio, in *Proceedings of the 6th European Signal Processing Conference (EUSIPCO)*, vol. 1, Amsterdam, Netherlands, June 1992, pp. 211–214
64. T.K. Truong, P.D. Chen, T.C. Cheng, Fast algorithm for computing the forward and inverse MDCT in MPEG audio coding. Signal Process. **86**(5), 1055–1060 (2006)
65. P.-S. Wu, Y.-T. Hwang, Efficient IMDCT core designs for audio signal processing, in *Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS'2003)*, Seoul, Korea, August 2003, pp. 275–280
66. G. Wu, E.-H. Yang, A new efficient method of computing MDCT in MP3 audio coding, in *Proceedings of the IEEE International Conference on Multimedia Information Networking and Security (MINES'2010)*, Nanjing Yiangsu, China, November 2010, pp. 63–67
67. J.-S. Wu, H.-Z. Shu, L. Senhadji, L.-M. Luo, Mixed-radix algorithm for the computation of forward and inverse MDCTs. IEEE Trans. Circuits Syst. Regul. Pap. **56**(4), 784–794 (2009)
68. J. Wu, L. Wang, L. Senhadji, H. Shu, Improved radix-3 decimation-in-frequency algorithm for the fast computation of forward and inverse MDCT, in *Proceedings of the IEEE International Conference on Audio, Language and Image Processing (ICALIP'2010)*, Shanghai, China, November 2010, pp. 694–699
69. L. Zhao, T. Zhang, C.-T. Liu, Analysis of IMDCT fast algorithms, in *Proceedings of the IEEE International Conference on Audio, Language and Image Processing (ICALIP'2010)*, Shanghai, China, November 2010, pp. 719–721

Recursive/Regressive MDCT Filter Structures

70. C.-H. Chen, C.-B. Wu, B.-D. Liu, J.-F. Yang, Recursive architectures for the forward and inverse modified discrete cosine transforms, in *Proceedings of the IEEE Workshop on Signal Processing Systems: Design and Implementation (SiPS'2000)*, Lafayette, LA, October 2000, pp. 50–59
71. C.-H. Chen, B.-D. Liu, J.-F. Yang, Recursive architectures for realizing modified discrete cosine transform and its inverse. IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process. **50**(1), 38–45 (2003)

72. Z.-Y. Cheng, C.-H. Chen, B.-D. Liu, J.-F. Yang, Unified selectable fixed-coefficient recursive structures for computing DCT, IMDCT, and sub-band synthesis filtering, in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'2004)*, vol. 3, Vancouver, Canada, May 2004, pp. 557–560
73. H.C. Chiang, J.C. Liu, Regressive implementations for the forward and inverse MDCT in MPEG audio coding. IEEE Signal Process. Lett.**3**(4), 116–118 (1996)
74. T.W. Fox, A. Carriera, Goertzel implementations of the forward and inverse modified discrete cosine transform, in *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering (CCECE'2004)*, vol. 4, Niagara Falls, Canada, May 2004, pp. 2371–2374
75. R. Koenig, T. Stripf, J. Becker, A novel recursive algorithm for bit-efficient realization of arbitrary length inverse modified cosine transforms, in *Proceedings of the International Conference on Design, Automation and Test in Europe (DATE'2008)*, Munich, Germany, March 2008, pp. 604–609
76. S.-C. Lai, S.-F. Lei, C.-H. Luo, Common architecture design of novel recursive MDCT and IMDCT algorithms for application to AAC, AAC in DRM, and MP3 codecs. IEEE Trans. Circuits Syst. Express Briefs **56**(10) 793–797 (2009)
77. S.-C. Lai, Y.-P. Yeh W.-C. Tseng, S.-F. Lei, Low-cost and high-accuracy design of fast recursive MDCT/MDST/IMDCT/IMDST algorithms and their realization. IEEE Trans. Circuits Syst. Express Briefs **59**(1), 65–69 (2012)
78. S.-F. Lei, S.-C. Lai, Y.-T. Hwang, C.-H. Luo, A high-precision algorithm for the forward and inverse MDCT using the unified recursive architecture, in *Proceedings of the IEEE International Symposium on Consumer Electronics*, Algarve, Portugal, April 2008, pp. 1–4
79. S.-F. Lei, S.-C. Lai, P.-Y. Cheng, C.-H. Luo, Low complexity and fast computation for recursive MDCT and IMDCT algorithms. IEEE Trans. Circuits Syst. Express Briefs **57**(7), 571–575 (2010)
80. H. Li, P. Li, Y. Wang, A compact hardware accelerator structure for realizing fast IMDCT computation, in *Proceedings of the IEEE 1st Asia-Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia'2009)*, Shanghai, China, January 2009, pp. 317–320
81. H. Li, P. Li, Q. Tang, L. Gao, A new decomposition algorithm of DCT-IV/DST-IV for realizing fast IMDCT computation. IEEE Signal Process. Lett. **16**(9) 735–738 (2009)
82. L. Li, H. Miao, X. Li, D. Guo, Efficient architectures of MDCT/IMDCT implementation for MPEG audio codec, in *Proceedings of the IEEE 3rd International Conference on Anti-Counterfeiting, Security and Identification in Communications (ASID'2009)*, Hong Kong, China, August 2009, pp. 156–159
83. H. Li, P. Li, Y. Wang, An efficient hardware accelerator architecture for implementing fast IMDCT computation. Signal Process. **90**(8), 2540–2545 (2010)
84. H. Li, Y. Wang, P. Li, Y. Li, A new recursive decomposition algorithm to calculate IMDCT, in *2013 AASRI Conference on Parallel and Distributed Systems, AASRI Procedia*, vol. 5, 2013, pp. 177–182
85. V. Nikolajevic, G. Fettweis, New recursive algorithms for the forward and inverse MDCT, in *Proceedings of the IEEE Workshop on Signal Processing Systems: Design and Implementation (SiPS'2001)*, Antwerp, Belgium, September 2001, pp. 51–57
86. V. Nikolajevič, G. Fettweis, Computation of forward and inverse MDCT using Clenshaw's recurrence formula. IEEE Trans. Signal Process. **51**(5), 1439–1444 (2003)
87. V. Nikolajevič, G. Fettweis, New recursive algorithms for the unified forward and inverse MDCT/MDST. J. VLSI Signal Process. Syst. Signal Image Video Technol. **34**(3), 203–208 (2003)
88. T.-H. Tsai, J.-N. Liu, Architecture design for MPEG-2 AAC filterbank decoder using modified regressive method, in *Proceedings of the IEEE ICASSP'2002*, Vol. 3, Orlando, FL, May 2002, pp. 3216–3219

Software/Hardware MLT (MDCT) Implementations in MP3

89. K.H. Bang, J.S. Kim, Y.C. Park, D.H. Youn, Design optimization of a dual MP3/AAC decoder, in *Proceedings of the IEEE ICASSP'2002*, vol. 4, Orlando, FL, May 2002, pp. 3776–3779
90. J.A.A. Cardona, S. Jayakumar, Real-time MP3 encoder implemented in DSP hardware, in *Proceedings of the International Conference on Signal Processing and Global DSP Expo*, Dallas, TX, March-April 2003
91. I. Fältman, M. Hast, A. Lundgren, S. Malki, E. Montnemery, A. Rangevall, J. Sandvall, M. Stamenkovic, A hardware implementation of an MP3 decoder, Digital IC-Project, LTH, Lund Institute of Technology, Sweden, May 2003
92. M. Galabov, Implementation of IMDCT block of an MP3 decoder through optimization on the DCT matrix. Radioengineering **13**(4), 22–26 (2004)
93. V. Gurkhe, Optimization of an MP3 decoder on the ARM processor, in *Proceedings of the IEEE Region 10 Asia-Pacific Conference TENCON'2003*, Bangalore, India, October 2003, Poster 280, pp. 1475–1478
94. H.-S. Kim, S.-H. Kim, K.-S. Chung, T.-H. Han, Low power implementation of MDCT/IMDCT for MP3 audio decoder, in *Proceedings of the IEEE International Conference on SoC Design (ISOCC'2010)*, Incheon, Korea, November 2010, pp. 143–146
95. K.H. Lee, K.-S. Lee, T.-H. Hwang, Y.-C. Park, D.H. Youn, An architecture and implementation of MPEG audio layer III decoder using dual-core DSP. IEEE Trans. Consum. Electron. **47**(4), 928–933 (2001)
96. K.-S. Lee, H.-O. Oh, Y.-C. Park, D.H. Youn, High quality MPEG-audio layer 3 algorithm for a 16-bit DSP, in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'2001)*, vol. 2, Sydney, Australia, May 2001, pp. 205–208
97. W. Lee, K. You, W. Sung, Software optimization of MPEG audio layer-III for a 32-bit RISC processor, in *Proceedings of the IEEE Asia-Pacific Conference on Circuits and Systems*, vol. 1, Bali, Indonesia, October 2002, pp. 435–438
98. P. Malik, Highly scalable IP core to accelerate the forward/backward modified discrete cosine transform in MP3 implemented to FPGA and low-power ASIC. IET Circuits Devices Syst. **5**(5), 351–359 (2011)
99. P. Malik, M. Balaz, M. Simlastik, A. Luczyk, W. Pleskacz, Various MDCT implementations in 0.35μm CMOS, in *Proceedings of the IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS'2008)*, Bratislava, Slovak Republic, April 2008, pp. 170–173
100. P. Malik, M. Ufnal, A.W. Luczyk, M. Balaz, W.A. Pleskacz, MDCT/IMDCT low power implementations in 90nm CMOS technology for MP3 audio, in *Proceedings of the IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS'2009)*, Liberec, Czech republic, April 2009, pp. 144–147
101. M. Pohronska, P. Malik, M. Balaz, FPGA implementation of fully parallel fast MDCT algorithm, in *Proceedings of the IEEE international Conference EUROCON'2009*, Saint Petersburg, Russia, May 2009, pp. 161–166
102. C. Ran, T.-Z. Shen, Low-complexity MP3 decoder based on Broadcom embedded platform. J. Beijing Inst. Technol. (English Edition), **20**(1), 94–99 (2011)
103. M. Samin, J. Frounchi, Low-power MP3 decoder implemented in XILINX VIRTEX-4 FPGA. Eur. J. Sci. Res. **15**(3), 386–395 (2006)
104. P. Sundareson, A re-evaluation of fundamental transform structures for efficient implementation on semi-parallel DSP architectures, in *112th AES Convention*, Munich, Germany, May 2002, Preprint #5614
105. W. Sung, H. Chang, W. Lee, S. Ryu, Speaking partner – an ARM7-based multimedia handheld device, in *Proceedings of the IEEE Workshop on Signal Processing Systems: Design and Implementation (SiPS'2002)*, San Diego, CA, October 2002, pp. 218–221

106. T.-H. Tsai, C.-N. Liu, A configurable common filterbank processor for multi-standard audio decoder. IEICE Trans. Fundam. **E90-A**(9), 1913–1923 (2007)
107. T.-H. Tsai, Y.-C. Yang, Low power and cost effective VLSI design for an MP3 audio decoder using an optimized synthesis-subband approach. IEE Proc. Comput. Digital Tech. **151**(3), 245–251 (2004)
108. T.-H. Tsai, Y.-C. Yang, C.-N. Liu, A hardware/software co-design of MP3 audio decoder. J. VLSI Signal Process. Syst. Signal Image Video Technol. **41**, 111–127 (2005)
109. X. Wang, W. Dou, Z. Hou, An improved audio encoding architecture based on 16-bit fixed-point DSP, in *Proceedings of the IEEE International Conference on Communications, Circuits and Systems, and West Sino Expositions (ICCCAS and WeSino Expo'2002)*, vol. 2, Chengdu City, China, June 2002, pp. 918–921
110. X. Yang, S. Shi, A.K. Wong, Tradeoffs in modified discrete cosine transform implementations, *Proceedings of the 4th International Conference on ASIC*, Shanghai, China, October 2001, pp. 370–373
111. Y. Yao, Q. Yao, P. Liu, Z. Xiao, Embedded software optimization for MP3 decoder implemented on RISC core. IEEE Trans. Consum. Electron. **50**(4), 1244–1249 (2004)
112. W. Zhang, P. Liu, Z.-B. Zhai, A hardware/software co-optimization approach for embedded software of MP3 decoder. J. Zhejiang Univ. Sci. A **8**(1), 49–56 (2007)

Supporting Literature

113. M.F. Aburdene, J. Zheng, R.J. Kozick, Computation of discrete cosine transform using Clenshaw's recurrence formula. IEEE Signal Process. Lett. **2**(8), 155–156 (1995)
114. G. Bi, Y. Zeng, *Transforms and Fast Algorithms for Signal Analysis and Representations* (Birkhäuser, Boston, 2004), Chap. 6, pp. 211–212
115. G. Bonnerot, M. Bellanger, Odd-time odd-frequency discrete Fourier transform for symmetric real-valued series. Proc. IEEE **64**, 392–393 (1976)
116. V. Britanak, On the discrete cosine transform computation. Signal Process. **40**(2–3), 183–194 (1994)
117. V. Britanak, Comments on fast radix-9 algorithm for the DCT-IV computation. IEEE Signal Process. Lett. **16**(11), 1005–1006 (2009)
118. V. Britanak, K.R. Rao, A unified fast MDCT/MDST computation in the evenly stacked analysis/synthesis system. Circuits Syst. Signal Process. **21**(4), 415–426 (2002)
119. V. Britanak, P. Yip, K.R. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations* (Academic Press, Elsevier Science, Amsterdam, 2007)
120. C.S. Burrus, T.W. Parks, *DFT/FFT and Convolution Algorithms* (Wiley, New York, 1985)
121. S.C. Chan, K.L. Ho, Direct methods for computing discrete sinusoidal transforms. IEE Proc. F: Radar Signal Process. **137**(6), 433–442 (1990)
122. Y.H. Chan, W.-C. Siu, Mixed-radix discrete cosine transform. IEEE Trans. Signal Process. **41**(11), 3157–3161 (1993)
123. L.-P. Chau, W.-C. Siu, Recursive algorithm for the discrete cosine transform with general lengths. Electron. Lett. **30**(3), 197–198 (1994); (Errata, Electron. Lett. **30**(7), 608 (1994))
124. L.-P. Chau, W.-C. Siu, Efficient recursive algorithm for the inverse discrete cosine transform. IEEE Signal Process. Lett. **7**(10), 276–277 (2000)
125. B. Edler, Coding of audio signals with overlapping block transform and adaptive window functions. Frequenz **43**(9), 252–256 (1989) (in German)
126. B. Edler, Aliasing reduction in sub-bands of cascaded filter banks with decimation. Electron. Lett. **28**(12), 1104–1106 (1992)
127. D.F. Elliott, K.R. Rao, *Fast Transforms: Algorithms, Analyzes, Applications* (Academic Press, New York, 1982), pp. 128–131

128. F.R. Gantmacher, *The Theory of Matrices*, 2nd edn. (Nauka, Moscow, 1966) (in Russian); English translation: vol. 1 and 2, Chelsea, New York, 1959
129. G. Goertzel, An algorithm for the evaluation of finite trigonometric series. Amer. Math. Monthly **65** 34–35 (1958)
130. H.-W. Hsu, C.-M. Liu, Fast radix-q and mixed-radix algorithms for type-IV DCT. IEEE Signal Process. Lett. **15**(12), 910–913 (2008)
131. S.G. Johnson, M. Frigo, A modified split-radix FFT with fewer arithmetic operations. IEEE Trans. Signal Process. **55**(1), 111–119 (2007)
132. C.W. Kok, Fast algorithm for computing discrete cosine transform. IEEE Trans. Signal Process. **45**(3), 757–760 (1997)
133. T. Mochizuki, Perfect reconstruction conditions for adaptive blocksize MDCT. IEICE Trans. Fundam. **E77-A**(5), 894–899 (1994)
134. V. Muddhasani, M.D. Wagh, Bilinear algorithms for discrete cosine transforms of prime lengths. Signal Process. **86**(9) (2006), 2393–2406
135. H.J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms* (Springer-Verlag, New York, 1981), pp. 144–148
136. G. Plonka, M. Tasche, Fast and numerically stable algorithms for discrete cosine transforms. Linear Algebra Appl. **394**(1), 309–345 (2005)
137. J.P. Princen, A.B. Bradley, Analysis/synthesis filter bank design based on time domain aliasing cancellation. IEEE Trans. Acoust. Speech Signal Process. **ASSP-34**(5), 1153–1161 (1986)
138. N. Rama Murthy, M.N.S. Swamy, On the algorithms for the computation of even discrete cosine transform-2 (EDCT-2) of real sequences. IEEE Trans. Circuits Syst. **37**(5), 625–627 (1990)
139. K.R. Rao, D.N. Kim, J.J. Hwang, *Fast Fourier Transform: Algorithms and Applications* (Springer, Heidelberg, 2010)
140. M.D. Wagh, H. Ganesh, A new algorithm for the discrete cosine transform of arbitrary number of points. IEEE Trans. Comput. **C-29**(4), 269–277 (1980)
141. Z. Wang, G.A. Julien, W.C. Miller, Recursive algorithm for the forward and inverse discrete cosine transform with arbitrary length. IEEE Signal Process. Lett. **1**(7), 101–102 (1994)

Chapter 6
Perfect Reconstruction Cosine/Sine-Modulated Filter Banks in the Dolby Digital (Plus) AC-3 Audio Coding Standards

6.1 Introduction

The Dolby Digital (AC-3) [1, 2, 5, 6, 8, 9, 12, 16] and the Dolby Digital Plus or Enhanced AC-3 (E-AC-3) [4, 7, 10, 11] audio coding standards developed by the Dolby Laboratories are currently the key enabling technologies for high-quality compression of digital audio signals. The AC-3 is a multichannel (5.1 channels) transform-based audio coding system that uses perceptual coding principles to provide high-quality data rate reduction for the transmission of audio signals. First released in 1991 for cinema industry needs and standardized in 1995, the AC-3 went through many stages of refinements, improvements, and fine-tuning. The resulting algorithm is currently in use in a number of standard applications in consumer electronics including the North American HDTV standard, DVD-Video standard, Blu-ray Disc standard, and Digital Video Broadcasting (DVB) standard. The E-AC-3 system released in 2005 [4, 7, 10, 11] extends the capabilities of the existing AC-3 system to better meet the needs of the emerging broadcast and multimedia applications. The E-AC-3 is essentially the advanced version of AC-3 providing increased coding efficiency, flexibility and wider range of supported bit-rates, expanded channel formats (up to 15.1 channels), and reproduction circumstances while preserving a high level of compatibility and interoperability with existing AC-3 system [4, 7, 10]. An excellent overview of the current AC-3 and E-AC-3 codecs is presented in [10].

For the time/frequency transformation of an audio data block, and vice versa, the AC-3 and E-AC-3 have adopted the modified discrete cosine transform (MDCT) as a perfect reconstruction cosine-modulated filter bank based on the concept of time domain aliasing cancellation [14]. Besides the MDCT, the AC-3 defines additional two variants of cosine-modulated filter banks called the first and second short transforms. They are actually real-valued polyphase filter banks, where all channels are shifted versions of the same prototype low-pass filter, and these filter banks are derived directly from the type-IV discrete cosine/sine transform (DCT-IV/DST-IV)

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1_6

kernels [24]. Moreover, the current AC-3 and E-AC-3 codecs for better spectral estimation and for phase angle adjustment have adopted the modified discrete sine transform (MDST) which together with the MDCT forms a complex filter bank. Specifically, the MDCT as the real part and MDST as the imaginary part compose a complex filter bank called the modulated complex lapped transform (MCLT) [39, 44, 48].

The chapter is devoted to the perfect reconstruction cosine/sine-modulated filter banks used in the Dolby AC-3 and E-AC-3 audio coding standards. Specifically, the chapter presents:

- Definitions of the analysis/synthesis AC-3 filter banks including the customized Kaiser-Bessel Derived windowing function, and general symmetry properties of AC-3 transforms both in the time and frequency domains.
- Efficient unified implementations of AC-3 transforms with corresponding signal flow graphs, and comparison of efficient unified implementations in terms of computational complexity and structural simplicity.
- Matrix representations of windowing procedure, AC-3 filter banks, their properties and useful relations among the AC-3 transform (sub-)matrices.
- Relations between the frequency coefficients and the time domain aliasing data sequences of AC-3 transforms.
- Fast algorithm for conversion of frequency coefficients of AC-3 transforms directly in the frequency domain for efficient E-AC-3 to AC-3 bit stream conversion and AC-3 to E-AC-3 bit stream transcoding. The fast conversion algorithm is compared with standard conversion methods in terms of computational complexity and memory requirements.
- Conversion methods of the MDCT to MDST frequency coefficients (exact and approximate), or equivalently the methods for construction of complex MCLT filter bank directly in the frequency domain. Exact and approximate conversion methods are compared in terms of computational complexity, structural simplicity, and memory requirements.

We note that the chapter summarizes many new research results achieved in the last few years related to the perfect reconstruction cosine/sine-modulated filter banks used in the Dolby AC-3 and E-AC-3 codecs. More specifically, it presents more efficient solutions of several open problems originally formulated in [4, 10, 37]. In general, some efficient methods can be used in any MDCT-based audio coding scheme/standard/system.

6.2 Definitions of AC-3 Filter Banks

The AC-3 defines analysis and synthesis filter banks with a variable parameter α, respectively, as [6]

$$c_k^{(\alpha)} = -\frac{2}{N}\sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}(2n+1)(2k+1) + \frac{\pi}{4}(2k+1)(1+\alpha)\right],$$

$$k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{6.1}$$

$$\hat{x}_n^{(\alpha)} = -\sum_{k=0}^{\frac{N}{2}-1} c_k^{(\alpha)} \cos\left[\frac{\pi}{2N}(2n+1)(2k+1) + \frac{\pi}{4}(2k+1)(1+\alpha)\right],$$

$$n = 0, 1, \ldots, N-1, \tag{6.2}$$

where

$$\alpha = \begin{cases} -1 & \text{for the first short transform } (N = 256) \\ 0 & \text{for the long transform } (N = 512) \\ +1 & \text{for the second short transform } (N = 256). \end{cases}$$

$\{x_n\}$, $n = 0, 1, \ldots, N-1$ represents the input audio data sequence which is windowed by a symmetrical customized Kaiser-Bessel derived (KBD) windowing function before its transformation [2, 9, 11–13, 15]. $\{\hat{x}_n^{(\alpha)}\}$, $n = 0, 1, \ldots, N-1$, represents the recovered time domain aliased data sequence which does not correspond to the original data sequence $\{x_n\}$.

Two algebraic terms of cosine transform kernels in (6.1) and (6.2) can be combined, and AC-3 analysis and synthesis filter banks, respectively, may be rewritten in more simplified equivalent forms as

$$c_k^{(\alpha)} = -\frac{2}{N}\sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}(1+\alpha)\right)(2k+1)\right],$$

$$k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{6.3}$$

$$\hat{x}_n^{(\alpha)} = -\sum_{k=0}^{\frac{N}{2}-1} c_k^{(\alpha)} \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}(1+\alpha)\right)(2k+1)\right],$$

$$n = 0, 1, \ldots, N-1. \tag{6.4}$$

It is important to note that the normalization factor $\frac{2}{N}$ in the analysis AC-3 filter bank (6.1) or (6.3) with incorporated windowing operation should be reduced to $\frac{4}{N}$ to maintain the perfect reconstruction conditions in the overlapped part when the windowing and overlap-add procedure is applied. With respect to the definitions of AC-3 transforms it is assumed through the chapter that $M = \frac{N}{2}$ denotes the size of

two short transforms and N is an even integer divisible by 4. Further, for simplicity, the scaling factor $-\frac{2}{N}$, $(-\frac{4}{N})$ in (6.1) and (6.3), and the sign "–" in (6.2) and (6.4) are omitted.

6.2.1 Parametrized KBD Windowing Function

The parametrized KBD windowing function has been constructed by a numerical normalization procedure proposed by the Dolby Laboratories [2, 9, 11, 15]. This numerical normalization procedure generally guarantees that any customized symmetric windowing function satisfying perfect reconstruction conditions can be derived from an initial symmetric nonnegative kernel function of the length $\frac{N}{2}+1$. The KBD windowing function $\{w_n\}$ which is identical both for the analysis and synthesis AC-3 filter banks with 50% overlapping is obtained by applying a transformation of the form [2, 15]

$$w_n = \sqrt{\frac{\sum_{j=0}^{n} v_j}{\sum_{j=0}^{\frac{N}{2}} v_j}}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1, \tag{6.5}$$

where $\{v_j\}$ represents an appropriately chosen symmetric nonnegative kernel function. In designing of the KBD windowing function, the Dolby Laboratories adopted the parametrized symmetric Kaiser-Bessel function with a variable parameter β defined as

$$v_n = \frac{I_0\left(\pi\ \beta\ \sqrt{1-\left[\frac{n-\frac{N}{4}}{\frac{N}{4}}\right]^2}\right)}{I_0(\pi\ \beta)}, \qquad n = 0, 1, \ldots, \frac{N}{2}, \tag{6.6}$$

where $I_0(x)$ is the modified zeroth order Bessel function given by

$$I_0\ (x) = \sum_{k=0}^{\infty}\left[\frac{(\frac{x}{2})^k}{k!}\right]^2. \tag{6.7}$$

Since the KBD windowing function is symmetric, i.e., $w_{N-1-n} = w_n$ for $n = 0, 1, \ldots, \frac{N}{2}-1$, only half of its values is sufficient to be stored. The KBD windowing function in the AC-3 and E-AC-3 is used with parameter $\beta = 5$. Plot of the KBD windowing function generated by the procedure (6.5) for $N = 512$ and $\beta = 5$ is shown in Fig. 6.1. For the shape comparison, plot of the sine windowing function is also included.

We note that the KBD windowing function is also employed in the MPEG-2/4 AAC [2, 3], specifically, with $\beta = 4$ for steady-state conditions (long windowing function), and with $\beta = 6$ for transients (short windowing function).

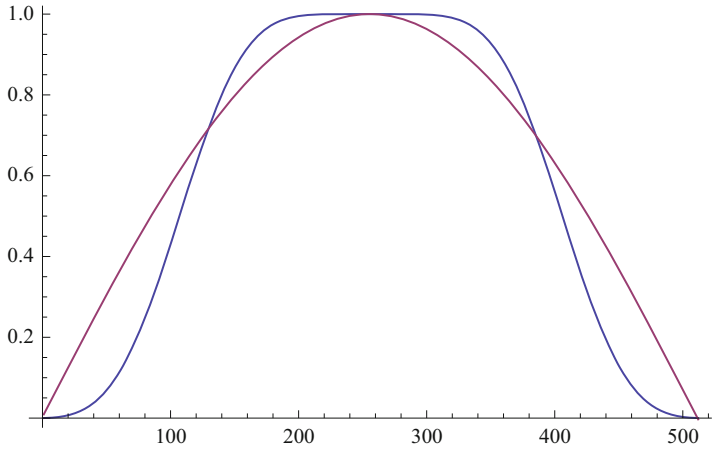

Fig. 6.1 Plot of the KBD windowing function for $N = 512$ and $\beta = 5$. For comparison plot of the sine windowing function is also shown

6.2.2 *How the AC-3 and E-AC-3 Systems Transform Audio Data Blocks*

Compared to the MPEG-2/4 AAC approach [2, 3], the AC-3 [1, 2, 5, 6, 8, 9, 12, 16] maintains the perfect reconstruction of filter banks while avoiding transitional blocks. In the AC-3 transform block-size switching procedure, a long block of $N = 512$ samples or two short blocks of $M = 256$ samples is employed (with the time resolution of 10.7 or 5.3 ms, respectively, for the sampling frequency of 48 kHz). The windowed long block is transformed when the spectrum remains stationary, or varies only slowly with time resulting in 256 unique nonzero frequency coefficients. During transients, when the signal changes rapidly in time, shorter blocks are constructed to reduce pre-echo effects by taking windowed long 512-sample block and splitting it into two segments each containing 256 samples. The first half of long block is transformed separately from the second half of that block. Each half-block produces 128 unique nonzero frequency coefficients. This is identical to the number of frequency coefficients produced by a single long block, but with two times improved temporal resolution. Frequency coefficients from those two half-blocks are interleaved together on a coefficient-by-coefficient basis to form a single audio block of 256 coefficients being processed identically. The diagram of time-to-frequency transformation of audio data blocks and vice versa in the AC-3 encoder/decoder is shown in Fig. 6.2.

In the current architecture of AC-3 [4, 10, 11] blocks of frequency coefficients are grouped into continuous frames. The AC-3 frame length is fixed at 1536 frequency coefficients per input channel corresponding to six 256-coefficients blocks. Transformed blocks are then quantized and transmitted as the so-called

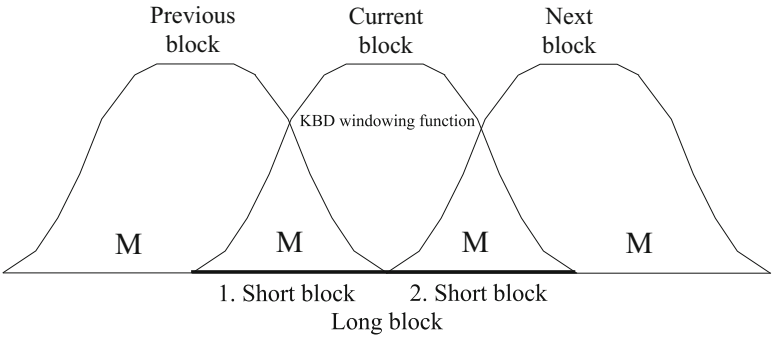

Fig. 6.2 Diagram of time-to-frequency transformation of audio data blocks and vice versa in the AC-3 encoder/decoder

spectral envelope with the associated side information. A similar, mirror image procedure is applied in the decoder during signal reconstruction. The current AC-3 encoder obtains better spectral power estimation in terms of improving the fidelity through the power energy summation of the MDCT frequency coefficients and frequency coefficients of the corresponding MDST [10].

As was mentioned earlier, the E-AC-3 [4, 7, 10, 11] is the advanced version of AC-3 providing increased coding efficiency, flexibility, and wider range of supported bit-rates, expanded channel formats (up to 15.1 channels) and reproduction circumstances while preserving a high level of compatibility and interoperability with existing AC-3 system [4, 7, 10]. The E-AC-3 preserves frame structure of six 256-coefficients blocks while also allows for shorter frames composed of one, two, or three frequency coefficients blocks. This feature enables to transport audio at data rates in formats limiting the amount of data per frame, such as DVD. The E-AC-3 utilizes new powerful coding tools such as an improved filter bank, improved quantization, enhanced channel coupling with phase preservation, spectral extension, and transient pre-noise processing. The improved filter bank is an adaptive hybrid transform composed of two linear transforms connected in cascade [10, 11]. The first transform is identical to that of employed AC-3: The windowed long (MDCT) transform producing 256 unique nonzero frequency coefficients. For frames containing audio signals which are stationary, a second linear transform can optionally be applied by E-AC-3 encoder, and inverted by the decoder. It is a non-windowed, non-overlapped type-II discrete cosine transform (DCT-II). When the DCT-II is applied, six 256-coefficients blocks are converted to a single 1536-coefficients block. This increases the frequency resolution resulting in the significantly improved coding efficiency and perceptual coding performance for stationary audio signals [10, 11]. Enhanced channel coupling process in the encoder and decoder requires the phase information for angle adjustment and therefore, besides the MDCT the corresponding MDST is also generated. The transient detector is similar but more sensitive to that employed in the standard AC-3 encoder.

However, although the E-C-3 in the presence of a transient can be switched into AC-3 block switching mode, the E-AC-3 decoder processes transient segments in a different way by the transient pre-noise coding tool [10, 11].

Essentially, the E-AC-3 bit streams are similar in nature to AC-3 bit streams, but are not backwards compatible, i.e., they are not decodable by AC-3 decoders. Annex E of [7] specifies E-AC-3 the bit stream syntax for decoding process.

6.2.3 Symmetry Properties of AC-3 Transforms

When investigating general mathematical properties of the filter banks, and when developing fast computational structures for their efficient implementation, they are frequently considered for simplicity as the block transforms applied to a single (windowed) data block. In the following subsections the symmetry properties of the AC-3 block transforms applied to the single audio data block are investigated in detail [29, 30].

6.2.3.1 The Forward and Backward Long (MDCT) Block Transforms

For $\alpha = 0$ in (6.3) and (6.4) let us denote $c_k = c_k^{(0)}$ and $\hat{x}_n = \hat{x}_n^{(0)}$. The AC-3 forward and backward long block transforms are, respectively, given by

$$c_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \qquad k=0,1,\ldots,\frac{N}{2}-1, \tag{6.8}$$

$$\hat{x}_n = \sum_{k=0}^{\frac{N}{2}-1} c_k \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \qquad n=0,1,\ldots,N-1, \tag{6.9}$$

and they correspond, respectively, to the forward and backward MDCT block transforms [14]. The symmetry properties of data sequences $\{c_k\}$ and $\{\hat{x}_n\}$ as well as the general mathematical and special properties of the MDCT block transform are presented in Chap. 3. Essentially, the forward and backward MDCT block transforms with the KBD windowing function (in general with any symmetric windowing function) can be efficiently realized by the fast analysis and synthesis MDCT filter banks based on an $\frac{N}{2}$-point DCT-IV [30]. This approach results in an identical fast computational structure with simple pre- and post-processing of data sequences both for the forward and backward MDCT computation.

6.2.3.2 The Forward and Backward First Short Block Transforms

For $\alpha = -1$ in (6.3) and (6.4), the forward and backward first short block transforms are, respectively, defined as

$$c_k^{(-1)} = \sum_{n=0}^{M-1} x_n^{(c)} \cos\left[\frac{\pi}{2M}(2n+1)(2k+1)\right], \quad k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{6.10}$$

$$\hat{x}_n^{(-1)} = \sum_{k=0}^{\frac{M}{2}-1} c_k^{(-1)} \cos\left[\frac{\pi}{2M}(2n+1)(2k+1)\right], \quad n = 0, 1, \ldots, M-1, \tag{6.11}$$

where $\{x_n^{(c)}\}$ represents the input data sequence taken from the first half of long audio block, i.e., $x_n^{(c)} = x_n$, $n = 0, 1, \ldots, M-1$. Substituting $M-1-k$ for k into (6.10) and $M-1-n$ for n into (6.11) we obtain

$$c_{M-1-k}^{(-1)} = -c_k^{(-1)}, \qquad k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{6.12}$$

demonstrating that $\{c_k^{(-1)}\}$ possesses the even anti-symmetry property while $\{\hat{x}_n^{(-1)}\}$ has the following symmetry:

$$\hat{x}_{M-1-n}^{(-1)} = -\hat{x}_n^{(-1)}, \qquad n = 0, 1, \ldots, \frac{M}{2} - 1. \tag{6.13}$$

Using the symmetry of the cosine transform kernel, i.e., substituting $M-1-n$ for n into (6.10), the forward first short transform can be written as

$$c_k^{(-1)} = \sum_{n=0}^{\frac{M}{2}-1} y_n^{(c)} \cos\left[\frac{\pi}{4(M/2)}(2n+1)(2k+1)\right], \quad k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{6.14}$$

where

$$y_n^{(c)} = x_n^{(c)} - x_{M-1-n}^{(c)}, \qquad n = 0, 1, \ldots, \frac{M}{2} - 1. \tag{6.15}$$

Since the DCT-IV is self-inverse, the backward first short transform is given by

$$y_n^{(c)} = \sum_{n=0}^{\frac{M}{2}-1} c_k^{(-1)} \cos\left[\frac{\pi}{4(M/2)}(2n+1)(2k+1)\right], \quad k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{6.16}$$

and the time domain aliased data sequence $\{\hat{x}_n^{(-1)}\}$ is recovered from $\{y_n^{(c)}\}$ by applying an inverse permutation to that of (6.15) defined as

$$\hat{x}_n^{(-1)} = y_n^{(c)}, \qquad \hat{x}_{M-1-n}^{(-1)} = -y_n^{(c)}, \qquad n = 0, 1, \ldots, \frac{M}{2} - 1. \tag{6.17}$$

The cosine transform kernels in (6.14) and (6.16) are recognized as $\frac{M}{2}$-point forward DCT-IV of $\{y_n^{(c)}\}$ and $\frac{M}{2}$-point inverse DCT-IV of $\{c_k^{(-1)}\}$, respectively.

6.2.3.3 The Forward and Backward Second Short Block Transforms

For $\alpha = +1$ in (6.3) and (6.4), the forward and backward second short block transforms are, respectively, defined as

$$c_k^{(+1)} = \sum_{n=0}^{M-1} x_n^{(s)} \cos\left[\frac{\pi}{2M}(2n + 1 + M)(2k + 1)\right], \quad k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{6.18}$$

$$\hat{x}_n^{(+1)} = \sum_{k=0}^{\frac{M}{2}-1} c_k^{(+1)} \cos\left[\frac{\pi}{2M}(2n + 1 + M)(2k + 1)\right], \quad n = 0, 1, \ldots, M - 1, \tag{6.19}$$

where $\{x_n^{(s)}\}$ represents the input data sequence taken from the second half of long audio block, i.e., $x_n^{(s)} = x_{M+n}$, $n = 0, 1, \ldots, M-1$. Similarly, substituting $M-1-k$ for k into (6.18) and $M - 1 - n$ for n into (6.19) we obtain

$$c_{M-1-k}^{(+1)} = -c_k^{(+1)}, \qquad k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{6.20}$$

and $\{c_k^{(+1)}\}$ possesses also the even anti-symmetry property while $\{\hat{x}_n^{(+1)}\}$ has the following symmetry:

$$\hat{x}_{M-1-n}^{(+1)} = \hat{x}_n^{(+1)}, \qquad n = 0, 1, \ldots, \frac{M}{2} - 1. \tag{6.21}$$

Applying a permutation to the input data sequence $\{x_n^{(s)}\}$ in (6.18) [25]

$$y_n^{(s)} = \begin{cases} -x_{\frac{M}{2}+n}^{(s)}, & n = 0, 1, \ldots, \frac{M}{2} - 1, \\ x_{n-\frac{M}{2}}^{(s)}, & n = \frac{M}{2}, \frac{M}{2} + 1, \ldots, M - 1, \end{cases} \tag{6.22}$$

and using the symmetry of the cosine transform kernel, i.e., substituting $M - 1 - n$ for n, the forward second short transform takes the following form:

$$c_k^{(+1)} = \sum_{n=0}^{\frac{M}{2}-1} y_n^{(s)} \cos\left[\frac{\pi}{4(M/2)}(2n+1)(2k+1)\right], \quad k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{6.23}$$

where

$$y_n^{(s)} = -x_{\frac{M}{2}+n}^{(s)} - x_{\frac{M}{2}-1-n}^{(s)}, \qquad n = 0, 1, \ldots, \frac{M}{2} - 1. \tag{6.24}$$

Since the DCT-IV is self-inverse, the backward second short transform is given by

$$y_n^{(s)} = \sum_{n=0}^{\frac{M}{2}-1} c_k^{(+1)} \cos\left[\frac{\pi}{4(M/2)}(2n+1)(2k+1)\right], \quad k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{6.25}$$

whereby the time domain aliased data sequence $\{\hat{x}_n^{(+1)}\}$ is recovered from $\{y_n^{(s)}\}$ by applying an inverse permutation to that of (6.24) defined as

$$\hat{x}_{\frac{M}{2}+n}^{(+1)} = -y_n^{(s)}, \qquad \hat{x}_{\frac{M}{2}-1-n}^{(+1)} = -y_n^{(s)}, \qquad n = 0, 1, \ldots, \frac{M}{2} - 1. \tag{6.26}$$

The cosine transform kernels in (6.23) and (6.25) are again recognized as $\frac{M}{2}$-point forward DCT-IV of $\{y_n^{(s)}\}$ and $\frac{M}{2}$-point inverse DCT-IV of $\{c_k^{(+1)}\}$, respectively.

In summary, from Eqs. (6.14)/(6.16) and (6.23)/(6.25) it can be seen that the M-point forward/backward first and second short transforms are, respectively, reduced to $\frac{M}{2}$-point forward/inverse DCT-IV.

6.3 Efficient Unified Implementations of AC-3 Transforms

Motivated by a specific definition of AC-3 filter banks, some proposed fast MDCT algorithms [19, 21, 30] and existing fast computational structures for the discrete sinusoidal unitary transforms computation of real data sequences [21–23, 25, 27, 28] have been investigated for the alternate and simultaneous (online) efficient implementation of AC-3 transforms. However, in all investigated unified approaches due to different transform lengths of the long (MDCT) and two short transforms, the efficient implementations have required a fast computational structure being reconfigurable with respect to the block length or they have required a modification in the internal structure of the fast algorithm. As the most efficient solution in terms of the arithmetic complexity and structural simplicity for the efficient implementation of the long (MDCT) and two short transforms the AC-3 [6] and E-AC-3 [7] have adopted a reconfigurable complex DFT/FFT-based fast algorithm [24, 53].

In the following subsections such efficient unified implementations of AC-3 transforms are discussed in more detail.

6.3.1 *Efficient Implementations of AC-3 Transforms Adopted in the AC-3 and E-AC-3 Codecs*

In the theory of fast MDCT algorithms it is well known that using a simple permutation applied to the input (windowed) data sequence, the MDCT can always be converted to the DCT-IV of half size. It is widely accepted that the DCT-IV-based fast MDCT algorithms are the most efficient both in terms of arithmetic complexity and structural simplicity [30].

Consider the N-point forward long (MDCT) transform given by (6.8). In fact, using a permutation applied to the input data sequence $\{x_n\}$ defined as

$$y_n = \begin{cases} -x_{\frac{3N}{4}+n} - x_{\frac{3N}{4}-1-n}, & n = 0, 1, \ldots, \frac{N}{4} - 1, \\ \\ x_{n-\frac{N}{4}} - x_{\frac{3N}{4}-1-n}, & n = \frac{N}{4}, \frac{N}{4} + 1, \ldots, \frac{N}{2} - 1, \end{cases} \tag{6.27}$$

or alternatively, applying a permutation defined as

$$\begin{aligned} y_{\frac{N}{4}+n} &= x_n - x_{\frac{N}{2}-1-n}, \\ y_{\frac{N}{4}-1-n} &= -x_{\frac{N}{2}+n} - x_{N-1-n}, \\ & n = 0, 1, \ldots, \frac{N}{4} - 1, \end{aligned} \tag{6.28}$$

the N-point forward long (MDCT) transform is converted to the $\frac{N}{2}$-point forward DCT-IV of $\{y_n\}$ as follows:

$$\begin{aligned} c_k &= \sum_{n=0}^{\frac{N}{2}-1} y_n \cos\left[\frac{\pi}{4(N/2)}(2n+1)(2k+1)\right], \\ & k = 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{6.29}$$

Although the permutations (6.27) and (6.28) seem to be different, they generate exactly the same data sequence $\{y_n\}$ [30]. Since the DCT-IV is self-inverse, with respect to (6.29) the backward long (MDCT) transform given by (6.9) is realized by the inverse $\frac{N}{2}$-point DCT-IV of $\{c_k\}$ as

$$y_n = \sum_{k=0}^{\frac{N}{2}-1} c_k \cos\left[\frac{\pi}{4(N/2)}(2n+1)(2k+1)\right],$$
$$n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{6.30}$$

The time domain aliased data sequence $\{\hat{x}_n\}$ can be recovered from $\{y_n\}$ applying an inverse permutation to that of (6.28) as

$$\hat{x}_n = y_{\frac{N}{4}+n}, \qquad \hat{x}_{\frac{N}{2}-1-n} = -y_{\frac{N}{4}+n},$$
$$\hat{x}_{\frac{N}{2}+n} = -y_{\frac{N}{4}-1-n}, \qquad \hat{x}_{N-1-n} = -y_{\frac{N}{4}-1-n},$$
$$n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{6.31}$$

Finally, Eqs. (6.14), (6.15) and (6.16), (6.17) define the fast algorithms, respectively, for the M-point forward and backward first short transforms computation based on the $\frac{M}{2}$-point forward and inverse DCT-IV. Similarly, Eqs. (6.23), (6.24) and (6.25), (6.26) define the fast algorithms, respectively, for the M-point forward and backward second short transforms also based on the $\frac{M}{2}$-point forward and inverse DCT-IV. As an example, signal flow graph for the online computation of the first and second short transforms for $M = 8$ is shown in Fig. 6.3.

Now, it is sufficient only to specify a suitable fast DCT-IV algorithm/computational structure valid for any N being an integer divisible by 4. Although generally, many 2^n-length [24, 49, 53] and even-length fast DCT-IV algorithms/computational structures are available [19, 30] (see Appendces C.2.1 and C.2.2), the AC-3 [6] and E-AC-3 [7] codecs have adopted a 2^n-length fast DCT-IV algorithm based on the identical complex forward FFT of half size [17, 24] which combines theoretical efficiency with very regular structure and achieves the lowest multiplicative complexity. For $N = 2^n$ its arithmetic complexity is $\frac{N}{2}(n+2)$ real multiplications and $\frac{3N}{2}n$ real additions [53]. The fast algorithm for the forward/inverse DCT-IV computation based on the identical complex forward FFT of half size [24] with corresponding fast computational structure is presented in Appendix C.2.1.

Thus, for the efficient implementation of N-point ($N = 512 = 2^9$, $n = 9$) forward/backward long (MDCT) transforms the AC-3 and E-AC-3 employ the $\frac{N}{2}$-point DCT-IV fast algorithm which is mapped into the identical $\frac{N}{4}$-point forward complex FFT module. Then, the forward long (MDCT) transform given by (6.29) taking into account Eq. (6.27) or (6.28), as well as the backward long (MDCT) transform given by (6.30) with overlap/add procedure require $\frac{N}{4}(n-1)$ real multiplications and $\frac{N}{4}(3n-1)$ real additions.

On the other hand, for the efficient implementation of M-point ($M = 256 = 2^8$, $m = 8$) forward/backward short transforms the AC-3 employ additionally the $\frac{M}{2}$-point DCT-IV fast algorithm which is similarly mapped into the identical $\frac{M}{4}$-point forward complex FFT module. Then, the forward first/second short transforms

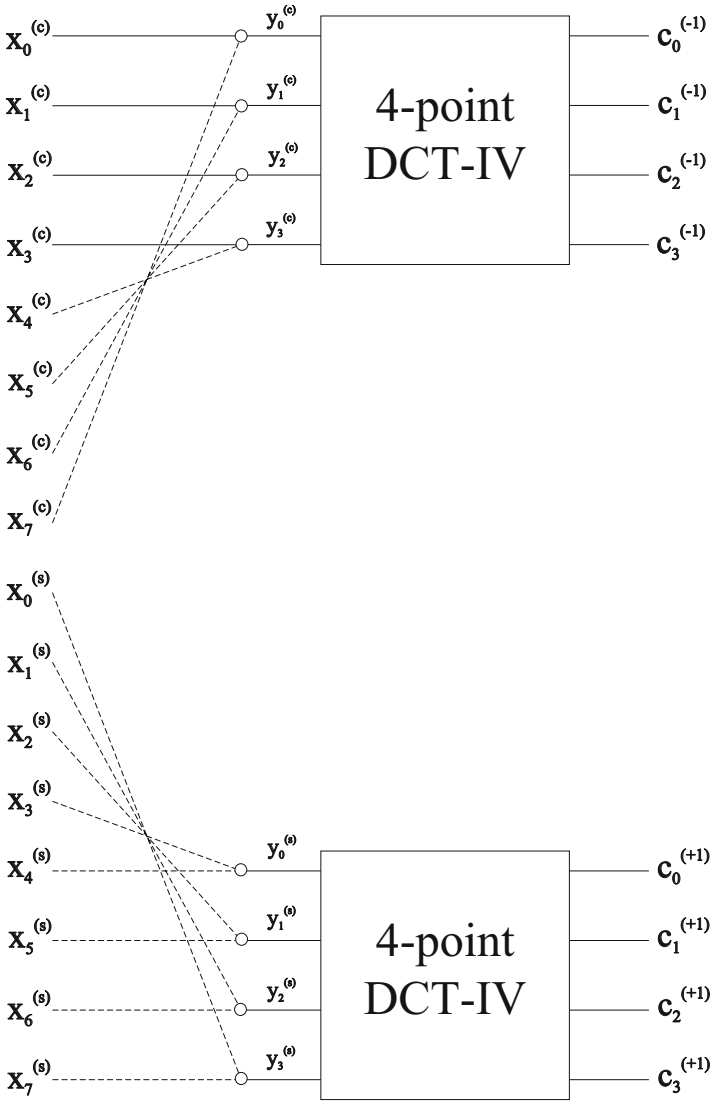

Fig. 6.3 Signal flow graph for the online computation of the first and second short transforms for $M = 8$

given by (6.14)/(6.23) and (6.15)/(6.24) require $\frac{M}{4}(m-1)$ real multiplications and $\frac{M}{4}(3m-1)$ real additions. The backward first/second short transforms given by (6.16)/(6.25) with overlap/add procedure have exactly the same arithmetic complexity.

Recently, new recursive algorithms for the 2^n length DCT-IV/DST-IV and MDCT computation have been presented in [26] requiring fewer total real multiplications

and additions than algorithms published up to now. They are based on a new improved FFT algorithm being actually the modified split-radix FFT with fewer arithmetic operations. For the N-point DCT-IV, $N = 2^n$, $n > 2$, the total number of arithmetic operations is asymptotically reduced from $2Nn + N$ to $\frac{17}{9}Nn + \frac{31}{27}N$. Since the DCT-IV and MDCT are closely related, this improved DCT-IV algorithm immediately implies an improved MDCT algorithm.

6.3.2 *Efficient Implementations of AC-3 Transforms Based on One Unified Transform Kernel*

An interesting method for the unified efficient implementation of AC-3 short transforms has been proposed in [27, 28]. First, the cosine transform kernel of AC-3 filter banks with the variable parameter α in (6.1) and (6.2) is rewritten into one unified equivalent form which corresponds to the cosine transform kernel of the long (MDCT) transform. Consequently, the short transforms can be implemented by the long (MDCT) transform formula with simple pre-processing of data sequences.

Consider the AC-3 analysis filter bank given by (6.1). Extending the second term of cosine transform kernel containing $(1 + \alpha)$ we have

$$c_k^{\alpha} = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}(2n+1)(2k+1) + \frac{\pi}{4}(2k+1) + \frac{\pi}{4}(2k+1)\alpha\right], \tag{6.32}$$

and combining the first and third terms in the cosine transform kernel of (6.32), after a simple algebraic manipulation we get

$$\begin{aligned} c_k^{\alpha} &= \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\alpha\right)(2k+1) + \frac{\pi}{4}(2k+1)\right] \\ &= \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}\left(2\left(n+\frac{N}{4}\alpha\right)+1\right)(2k+1) + \frac{\pi}{4}(2k+1)\right]. \end{aligned} \tag{6.33}$$

Finally, substituting $m = n + \frac{N}{4}\alpha$ ($n = m - \frac{N}{4}\alpha$) into (6.33) we obtain a unified equivalent form of the cosine transform kernel of the AC-3 analysis filter bank as [27, 28]

$$\begin{aligned} c_k^{\alpha} = &\sum_{m=\frac{N}{4}\alpha}^{N-1+\frac{N}{4}\alpha} x_{m-\frac{N}{4}\alpha} \cos\left[\frac{\pi}{2N}(2m+1)(2k+1) + \frac{\pi}{4}(2k+1)\right], \\ &k = 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{6.34}$$

For $\alpha = 0$ in Eq. (6.34) we have

$$c_k^{(0)} = \sum_{m=0}^{N-1} x_m \cos\left[\frac{\pi}{2N}(2m+1)(2k+1) + \frac{\pi}{4}(2k+1)\right]$$
$$= \sum_{m=0}^{N-1} x_m \cos\left[\frac{\pi}{2N}\left(2m+1+\frac{N}{2}\right)(2k+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{6.35}$$

which actually for $m = n$ is the forward long (MDCT) block transform given by (6.8). When parameter $\alpha \neq 0$, the data sequence $\{x_{m-\frac{N}{4}\alpha}\}$ in (6.34) can be interpreted as a shifting of the original data samples by $\frac{N}{4}$ samples with respect to the cosine transform kernel. In fact, exploiting the anti-periodicity property of MDCT transform kernel with the period N, i.e., by substituting $m + N$ for m into (6.35), we have [30]

$$\cos\left[\frac{\pi}{2N}\left(2m+1+\frac{N}{2}+2N\right)(2k+1)\right]$$
$$= -\cos\left[\frac{\pi}{2N}\left(2m+1+\frac{N}{2}\right)(2k+1)\right],$$

and it can be shown that for $\alpha = \mp 1$, the original data sequence $\{x_m\}$ has to be circularly shifted to the left/right in the period N by $\frac{N}{4}$ samples, respectively, followed by sign changes of $\frac{N}{4}$ circularly shifted samples.

Specifically, for $\alpha = -1$ in (6.34) the forward first short transform for $N = M$ is given by

$$c_k^{(-1)} = \sum_{m=-\frac{M}{4}}^{\frac{3M}{4}-1} x_{m+\frac{M}{4}} \cos\left[\frac{\pi}{2M}(2m+1)(2k+1) + \frac{\pi}{4}(2k+1)\right],$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{6.36}$$

while for $\alpha = +1$ in (6.34) the forward second short transform for $N = M$ is given by

$$c_k^{(+1)} = \sum_{m=\frac{M}{4}}^{\frac{5M}{4}-1} x_{m-\frac{M}{4}} \cos\left[\frac{\pi}{2M}(2m+1)(2k+1) + \frac{\pi}{4}(2k+1)\right],$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1. \tag{6.37}$$

Applying the following permutation to the input data sequence $\{x_m\}$ (actually corresponding to circular shifting to the left by $\frac{M}{4}$ samples)

$$y_m^{(-1)} = \begin{cases} x_{\frac{M}{4}+m}, & m = 0, 1, \ldots, \frac{3M}{4} - 1, \\ -x_{m-\frac{3M}{4}}, & m = \frac{3M}{4}, \frac{3M}{4} + 1, \ldots, M - 1, \end{cases} \tag{6.38}$$

the forward first short transform given by (6.36) takes the form

$$\begin{aligned} c_k^{(-1)} &= \sum_{m=0}^{M-1} y_m^{(-1)} \cos\left[\frac{\pi}{2M}(2m+1)(2k+1) + \frac{\pi}{4}(2k+1)\right] \\ &= \sum_{m=0}^{M-1} y_m^{(-1)} \cos\left[\frac{\pi}{2M}\left(2m+1+\frac{M}{2}\right)(2k+1)\right], \\ & k = 0, 1, \ldots, \frac{M}{2} - 1. \end{aligned} \tag{6.39}$$

On the other hand, applying the following permutation to the input data sequence $\{x_m\}$ (actually corresponding to circular shifting to the right by $\frac{M}{4}$ samples)

$$y_m^{(+1)} = \begin{cases} -x_{\frac{3M}{4}+m}, & m = 0, 1, \ldots, \frac{M}{4} - 1, \\ x_{m-\frac{M}{4}}, & m = \frac{M}{4}, \frac{M}{4} + 1, \ldots, M - 1, \end{cases} \tag{6.40}$$

the forward second short transform given by (6.37) takes the form

$$\begin{aligned} c_k^{(+1)} &= \sum_{m=0}^{M-1} y_m^{(+1)} \cos\left[\frac{\pi}{2M}(2m+1)(2k+1) + \frac{\pi}{4}(2k+1)\right], \\ &= \sum_{m=0}^{M-1} y_m^{(+1)} \cos\left[\frac{\pi}{M}\left(2m+1+\frac{M}{2}\right)(2k+1)\right], \\ & k = 0, 1, \ldots, \frac{M}{2} - 1. \end{aligned} \tag{6.41}$$

Equations (6.39) and (6.41) imply that both the forward first and second short transforms computation after a proper permutation of the original data sequence $\{x_m\}$ (Eqs. (6.38) and (6.40)) can be implemented via any fast computational structure for the forward long (MDCT) transform of size $M = \frac{N}{2}$. If the DCT-IV-based fast MDCT algorithm is applied to the computation of the forward first and

second short transforms, then one may simply use Eq. (6.27) or (6.28) for $N = M$ and $n = m$ to convert the M-point short transforms to $\frac{M}{2}$-point forward DCT-IV given by (6.29).

By the similar procedure for the AC-3 synthesis filter bank given by (6.2) with the variable parameter α, or simply directly from (6.34) we obtain

$$\hat{x}^{(\alpha)}_{m-\frac{N}{4}\alpha} = \sum_{k=0}^{\frac{N}{2}-1} c_k^{\alpha} \cos\left[\frac{\pi}{2N}(2m+1)(2k+1) + \frac{\pi}{4}(2k+1)\right],$$

$$m = \frac{N}{4}\alpha, \frac{N}{4}\alpha + 1, \ldots, N-1+\frac{N}{4}\alpha. \tag{6.42}$$

For $\alpha = 0$ in Eq. (6.42) we have

$$\hat{x}^{(0)}_m = \sum_{k=0}^{\frac{N}{2}-1} c_k^{(0)} \cos\left[\frac{\pi}{2N}(2m+1)(2k+1) + \frac{\pi}{4}(2k+1)\right]$$

$$= \sum_{k=0}^{\frac{N}{2}-1} c_k^{(0)} \cos\left[\frac{\pi}{2N}\left(2m+1+\frac{N}{2}\right)(2k+1)\right],$$

$$m = 0, 1, \ldots, N-1, \tag{6.43}$$

which actually for $m = n$ is the backward long (MDCT) block transform given by (6.9). For $\alpha = -1$ in (6.42) and $N = M$ the backward first short transform is given by

$$\hat{x}^{(-1)}_{m+\frac{M}{4}} = \sum_{k=0}^{\frac{M}{2}-1} c_k^{(-1)} \cos\left[\frac{\pi}{2M}(2m+1)(2k+1) + \frac{\pi}{4}(2k+1)\right],$$

$$= \sum_{k=0}^{\frac{M}{2}-1} c_k^{(-1)} \cos\left[\frac{\pi}{2M}\left(2m+1+\frac{M}{2}\right)(2k+1)\right],$$

$$m = -\frac{M}{4}, -\frac{M}{4}+1, \ldots, \frac{3M}{4}-1. \tag{6.44}$$

Similarly, for $\alpha = +1$ in (6.42) and $N = M$ the backward second short transform is given by

$$\hat{x}^{(+1)}_{m-\frac{M}{4}} = \sum_{k=0}^{\frac{M}{2}-1} c_k^{(+1)} \cos\left[\frac{\pi}{2M}(2m+1)(2k+1) + \frac{\pi}{4}(2k+1)\right],$$

$$= \sum_{k=0}^{\frac{M}{2}-1} c_k^{(+1)} \cos\left[\frac{\pi}{2M}\left(2m+1+\frac{M}{2}\right)(2k+1)\right],$$

$$m = \frac{M}{4}, \frac{M}{4}+1, \ldots, \frac{5M}{4}-1. \tag{6.45}$$

Exploiting the symmetry properties of the time domain aliased data sequences $\{\hat{x}_m^{(-1)}\}$ and $\{\hat{x}_m^{(+1)}\}$ given by (6.13) and (6.21), respectively, only half of samples is sufficient to be computed. Equations (6.44) and (6.45) imply that the data sequences $\{\hat{x}_m^{(-1)}\}$ and $\{\hat{x}_m^{(+1)}\}$ are recovered, respectively, by the backward first and second short transforms implemented via any fast computational structure for the backward long (MDCT) transform of size $M = \frac{N}{2}$.

6.3.3 *Efficient Implementations of AC-3 Transforms via the Fast MDCT Computational Structure*

The fast MDCT algorithm originally proposed in [21] and refined in [18] besides the computation of long transform can be also adopted for the alternate efficient implementation of two short transforms in the AC-3. Complete formulae of the fast MDCT algorithm [21] are presented in Chap. 4. The corresponding refined fast MDCT computational structure for $N = 16$ is shown in Fig. 6.4. The computation of backward long (MDCT) transform is realized simply by reversing the fast MDCT computational structure and performing inverse operations. For 2^n lengths the N-point long (MDCT) transform requires $\frac{N}{4}(n+1)$ real multiplications and $\frac{N}{4}(3n+3)$ real additions.

In order to adopt the fast MDCT computational structure for the alternate efficient implementation of two short transforms in the AC-3, consider the first step in the derivation of the fast MDCT algorithm defined as [21]

$$z_{2k} = (-1)^k \frac{\sqrt{2}}{2} \sum_{n=0}^{\frac{N}{2}-1} x_n^{'} \cos\left[\frac{\pi}{2N}(2n+1)(4k+1)\right]$$

$$-x_n^{''} \sin\left[\frac{\pi}{2N}(2n+1)(4k+1)\right],$$

$$k = 0, 1, \ldots, \frac{N}{2}-1, \tag{6.46}$$

where

$$x_n^{'} = x_n - x_{N-1-n}, \qquad x_n^{''} = x_n + x_{N-1-n}, \quad n = 0, 1, \ldots, \frac{N}{2}-1. \tag{6.47}$$

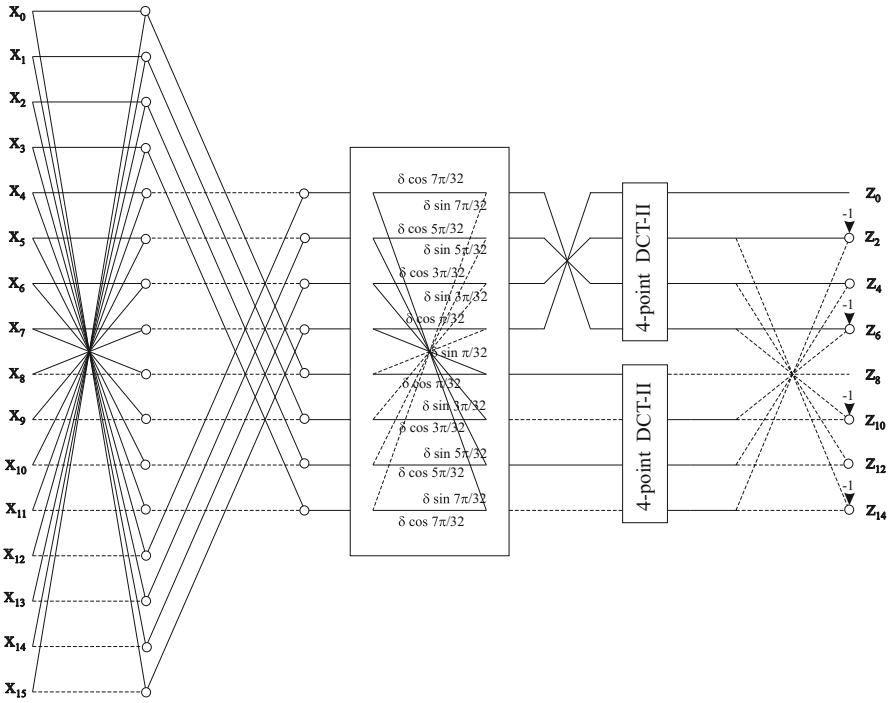

Fig. 6.4 Refined fast MDCT computational structure for $N = 16$, $\delta = \frac{\sqrt{2}}{2}$

Equation (6.46) is actually derived exploiting the even anti-symmetry property of MDCT coefficients [21]. This fact allows us to consider the even-indexed MDCT coefficients only. The odd-indexed MDCT coefficients can be deduced from the even anti-symmetry property.

Based on the even anti-symmetry property (6.12) of the first short transform given by (6.10), let us consider its even-indexed coefficients. Then, using the symmetry of cosine transform kernel we have

$$\begin{aligned} c_{2k}^{(-1)} &= \sum_{n=0}^{M-1} x_n^{(c)} \cos\left[\frac{\pi}{2M}(2n+1)(4k+1)\right] \\ &= \sum_{n=0}^{\frac{M}{2}-1} (x_n^{(c)} - x_{M-1-n}^{(c)}) \cos\left[\frac{\pi}{2M}(2n+1)(4k+1)\right], \\ & k = 0, 1, \ldots, \frac{M}{2} - 1. \end{aligned} \tag{6.48}$$

The data sequence $\{x_n^{(c)} - x_{M-1-n}^{(c)}\}$ in (6.48) has exactly the same form as $\{x_n'\}$ defined in (6.47) for $N = M$. In fact, from (6.46) it follows that the first short transform can be computed by the fast MDCT computational structure setting $x_n'' = 0$ for $n = 0, 1, \ldots, \frac{M}{2} - 1$, where x_n'' is defined in (6.47) for $N = M$. It means that the upper half after the first butterfly stage of the fast MDCT computational structure (see Figs. 6.4 and 6.5) is set to zero. Final coefficients of the first short transform are obtained as

$$c_{2k}^{(-1)} = z_{2k}, \qquad c_{2k+1}^{(-1)} = -z_{M-2-2k}, \quad k = 0, 1, \ldots, \frac{M}{4} - 1. \tag{6.49}$$

On the other hand, the second short transform given by (6.18) may be expressed in the following equivalent form

$$c_k^{(+1)} = (-1)^{k+1} \sum_{n=0}^{M-1} x_n^{(s)} \sin\left[\frac{\pi}{2M}(2n+1)(2k+1)\right],$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1. \tag{6.50}$$

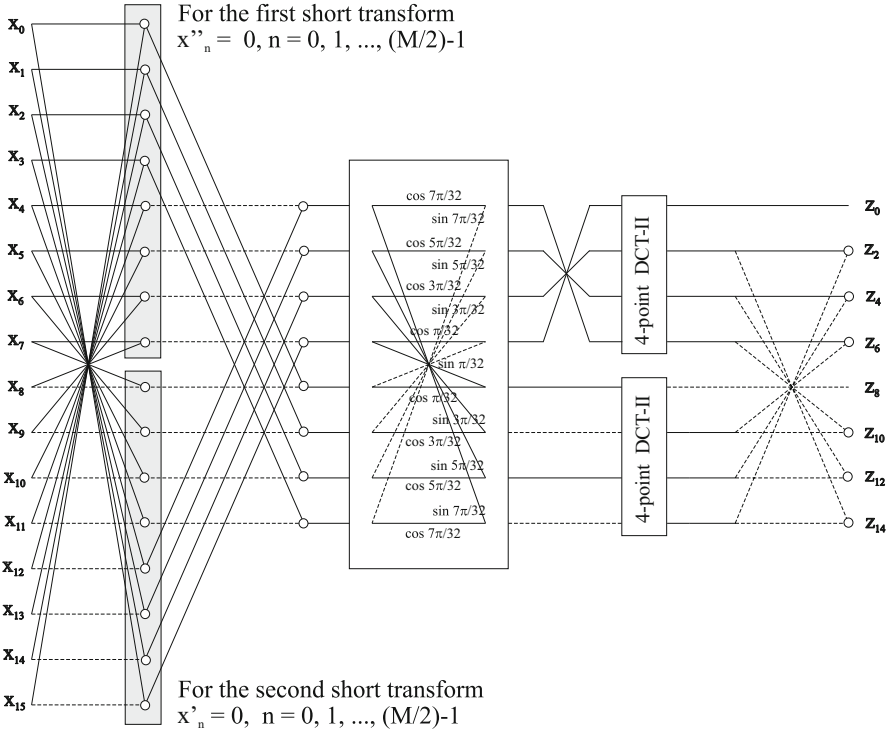

Fig. 6.5 Modified fast MDCT computational structure adopted for the alternate computation of two short transforms in the AC-3 for $M = 16$

Again, based on the even anti-symmetry property (6.20), let us consider the even-indexed coefficients of the second short transform given by (6.50). Using the symmetry of sine transform kernel we have

$$c_{2k}^{(+1)} = -\sum_{n=0}^{\frac{M}{2}-1} \left(x_n^{(s)} + x_{M-1-n}^{(s)}\right) \sin\left[\frac{\pi}{2M}(2n+1)(4k+1)\right],$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{6.51}$$

and the data sequence $\{x_n^{(s)} + x_{M-1-n}^{(s)}\}$ in (6.51) has exactly the same form as $\{x_n^{''}\}$ in (6.47) for $N = M$. Similarly, from (6.46) it follows that the second short transform can be computed by the fast MDCT computational structure setting $x_n^{'} = 0$ for $n = 0, 1, \ldots, \frac{M}{2} - 1$, where $x_n^{'}$ is defined in (6.47) for $N = M$. It means that the lower half after the first butterfly stage of the fast MDCT computational structure (see Figs. 6.4 and 6.5) is set to zero. Final coefficients of the second short transform are obtained as

$$c_{2k}^{(+1)} = z_{2k}, \qquad c_{2k+1}^{(+1)} = -z_{M-2-2k}, \quad k = 0, 1, \ldots, \frac{M}{4} - 1. \tag{6.52}$$

For the computation of two short transforms in the AC-3 the fast MDCT computational structure should be slightly modified. Comparison of (6.46) with (6.48) and (6.51) implies that the factor $(-1)^k \frac{\sqrt{2}}{2}$ in (6.46) has to be eliminated. The corresponding modified fast MDCT computational structure for the alternate computation of two short transforms in the AC-3 for $M = 16$ is shown in Fig. 6.5. The computation of backward short transforms is realized simply by reversing the modified fast MDCT computational structure and performing inverse operations. For $M = 2^m$ lengths the computation of M-point first or second short transform requires $\frac{M}{4}(m+1)$ real multiplications and $\frac{M}{4}(3m-1)$ real additions.

6.3.4 *Efficient Implementations of AC-3 Transforms via the Fast O^2DFT and DFT-IV Computational Structures*

A computationally efficient DFT/FFT-based MDCT algorithm originally proposed for the realization of real-valued single sideband analysis/synthesis filter banks (with perfect reconstruction as well as with nearly or almost perfect reconstruction) has been proposed in [23]. Due to the same even anti-symmetry property of the long (MDCT) transform coefficients and O^2DFT ones (see Appendix B.1), the efficient forward/backward long (MDCT) computation is based on the fast O^2DFT algorithm (see Appendix B.2) derived for odd/even symmetric real-valued data sequences. Complete formulae of the fast O^2DFT algorithm [23] for the efficient

implementation of long (MDCT) transform are presented in Chap. 4. For $N = 2^n$ the long (MDCT) transform computation requires $\frac{N}{4}(n+1)$ real multiplications and $\frac{N}{4}(3n+1)$ real additions.

The O^2DFT is equivalent to the generalized DFT of type IV (GDFT-IV) of real-valued data sequences defined as [20]

$$f_k^{IV} = \sum_{n=0}^{M-1} x_n \exp\left[-i\,\frac{2\pi}{4M}(2n+1)(2k+1)\right],$$
$$k = 0, 1, \ldots, M-1, \tag{6.53}$$

where $\{f_k^{IV}\}$ are GDFT-IV transform coefficients and $i = \sqrt{-1}$. The corresponding fast GDFT-IV computational structure [20] can be adopted for the online computation of two short transforms in the AC-3 as follows. Consider the first short transform defined by (6.10) and the second short transform defined by (6.50). Using the symmetry properties of cosine and sine transform kernels, the GDFT-IV given by (6.53) can be decomposed as

$$f_k^{IV} = \sum_{n=0}^{\frac{M}{2}-1} x_n^{'} \cos\left[\frac{\pi(2n+1)(2k+1)}{4(M/2)}\right] - i\,x_n^{''} \sin\left[\frac{\pi(2n+1)(2k+1)}{4(M/2)}\right],$$
$$k = 0, 1, \ldots, M-1, \tag{6.54}$$

where $\{x_n^{'}\}$ and $\{x_n^{''}\}$ are given by (6.47) for $N = M$. Using Eq. (6.50), the fast GDFT-IV computational structure can be modified for the online computation of two short transforms in the AC-3. It is shown for $M = 16$ in Fig. 6.6. For $M = 2^m$ the first/second short transform computation requires $\frac{M}{4}(m+1)$ real multiplications and $\frac{M}{4}(3m+1)$ real additions.

6.3.5 *Comparison of Existing Efficient Implementations of AC-3 Transforms*

The arithmetic complexity of $N = 2^n$-length fast algorithms for the forward/backward long (MDCT) transform computation is summarized in Table 6.1. The arithmetic complexity of the $M = 2^m$ length forward/backward first and second short transform computation is obtained, if we replace N by M and n by m in Table 6.1.

The DCT-IV-based efficient implementation of AC-3 transforms via the identical complex forward FFT of half size combines theoretical efficiency with very regular structure and achieves the lowest multiplicative complexity. Due to using different block lengths of AC-3 transforms, it is necessary only reconfigure it. On the other

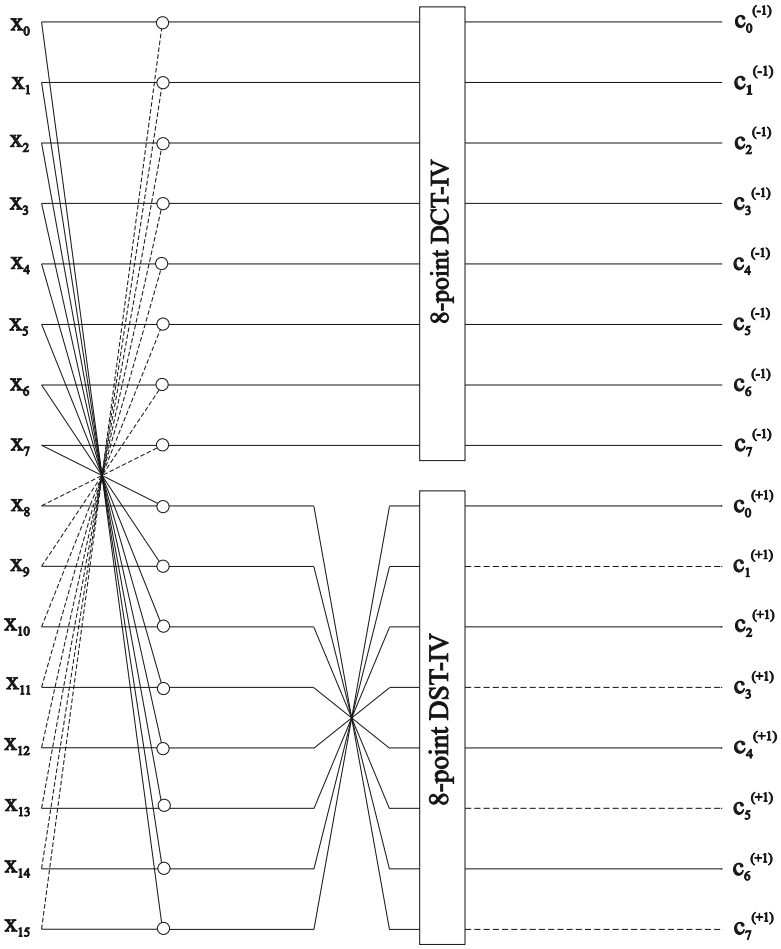

Fig. 6.6 Modified fast GDFT-IV computational structure for the online computation of two short transforms in the AC-3 for $M = 16$

Table 6.1 Summary of arithmetic complexity of discussed 2^n-length fast algorithms for the forward/backward long (MDCT) transform

$N = 2^n$-point fast algorithm	# of real mults	# of real adds
DCT-IV-based via $\frac{N}{4}$-point complex FFT	$\frac{N}{4}(n-1)$	$\frac{N}{4}(3n-1)$
Fast MDCT computational structure [18, 21]	$\frac{N}{4}(n+1)$	$\frac{N}{4}(3n+3)$
Fast O^2DFT algorithm [23]	$\frac{N}{4}(n+1)$	$\frac{N}{4}(3n+1)$

hand, the fast MDCT computational structure [21] requires a modification in its internal structure with respect to the block length.

6.3.6 *The Efficient Implementation of Adaptive Hybrid Transform*

We recall that the E-AC-3 utilizes an improved filter bank. The improved filter bank is an adaptive hybrid transform (AHT) composed of two linear transforms connected in cascade [10, 11]. The first transform is identical to that of employed in AC-3: the windowed 512-point long (MDCT) transform producing 256 unique nonzero frequency coefficients. For frames containing audio signals which are stationary, a second linear transform can optionally be applied by E-AC-3 encoder, and inverted by the decoder. It is the non-windowed and non-overlapped DCT-II. After the time-to-frequency transformation of successive overlapped audio blocks, six blocks of MDCT coefficients are packed and transformed by the DCT-II resulting in a block of 1536 coefficients. An efficient implementation of the DCT-II and its inverse, the DCT-III, is also briefly discussed for completeness.

Since the block length is a composite number, i.e., $1536 = 2^9 \times 3$, the composition of even-length fast recursive DCT-II algorithm and 3-point DCT-II module provides an efficient implementation of DCT-II in the E-AC-3 encoder. Complete formulae of the even-length fast recursive DCT-II (DCT-III) algorithm are presented in Appendix C.1.1, whereas the required efficient optimized 3-point DCT-II/DCT-III modules are presented in Appendix D.4. The arithmetic complexity of 1536-point DCT-II (DCT-III) computation is 7424 multiplications and 22 273 additions.

6.4 Matrix Representations of AC-3 Transforms

An alternative way to represent the AC-3 block transforms is in the matrix-vector form. Matrix representations are very powerful tools to analyze filter bank characteristics of the single data block both in the time and frequency domains. In the following subsections the matrix representations of windowing procedure and AC-3 transforms, their properties, and useful relations among transform (sub-)matrices are investigated and presented in detail [29, 30].

6.4.1 *Windowing Procedure*

Let $\boldsymbol{W}_N$ be a diagonal matrix with elements $\{w_n\}$ on the main diagonal defined as

$$\boldsymbol{W}_N = \begin{pmatrix} \boldsymbol{W}_{\frac{N}{2}}^{(1)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_{\frac{N}{2}}^{(2)} \end{pmatrix}, \tag{6.55}$$

representing the KBD windowing function, where $\boldsymbol{W}_{\frac{N}{2}}^{(1)}$ and $\boldsymbol{W}_{\frac{N}{2}}^{(2)}$ represent the first and second half of the windowing function, respectively. In order to eliminate time domain aliasing, the windowing functions of two succeeding data blocks have to satisfy the perfect reconstruction conditions. We recall that they are given by

$$\begin{aligned} &w_n^2 + w_{\frac{N}{2}+n}^2 = w_n^2 + w_{\frac{N}{2}-1-n}^2 = 1, \\ &w_n = w_{N-1-n}, \quad \text{or} \quad w_{\frac{N}{2}+n} = w_{\frac{N}{2}-1-n}, \quad n = 0, 1, \ldots, \frac{N}{2} - 1, \end{aligned} \tag{6.56}$$

or alternatively, in the matrix form as

$$\begin{aligned} &\boldsymbol{W}_{\frac{N}{2}}^{(1)}\, \boldsymbol{W}_{\frac{N}{2}}^{(1)} + \boldsymbol{W}_{\frac{N}{2}}^{(2)}\, \boldsymbol{W}_{\frac{N}{2}}^{(2)} = \boldsymbol{I}_{\frac{N}{2}}, \\ &\boldsymbol{W}_{\frac{N}{2}}^{(2)} = \boldsymbol{J}_{\frac{N}{2}}\, \boldsymbol{W}_{\frac{N}{2}}^{(1)}\, \boldsymbol{J}_{\frac{N}{2}}, \end{aligned} \tag{6.57}$$

where $\boldsymbol{I}_{\frac{N}{2}}$ is the identity matrix and $\boldsymbol{J}_{\frac{N}{2}}$ is the reverse ordered identity matrix both of order $\frac{N}{2}$. The matrix $\boldsymbol{W}_N$ given by (6.55) represents the windowing procedure by the symmetric KBD function applied to an audio data block in the encoder or to a recovered time domain aliased data block in the decoder.

6.4.2 *Forward/Backward Long (MDCT) Transform*

Consider the forward and backward MDCT block transforms defined by (6.8) and (6.9), respectively. Let the cosine transform kernel in the forward MDCT block transform (6.8) be represented by an $\frac{N}{2} \times N$ matrix $\boldsymbol{C}_{\frac{N}{2}\times N}$ with elements

$$\begin{aligned} &\left\{\boldsymbol{C}_{\frac{N}{2}\times N}\right\}_{k,n} = \cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)(2k+1)\right], \\ &k = 0, 1, \ldots, \frac{N}{2} - 1, \quad n = 0, 1, \ldots, N-1. \end{aligned} \tag{6.58}$$

Further, the cosine transform kernel in the backward MDCT block transform (6.9) is represented by the matrix $\left[\boldsymbol{C}_{\frac{N}{2}\times N}\right]^T = \boldsymbol{C}_{N\times\frac{N}{2}}$, where T denotes transposition. Next,

let $\mathbf{x}^T = [x_0, x_1, \ldots, x_{N-1}]^T$, $\mathbf{c}^T = \left[c_0, c_1, \ldots, c_{\frac{N}{2}-1}\right]^T$ and $\hat{\mathbf{x}}^T = [\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_{N-1}]^T$ be column vectors. Then, the forward and backward MDCT block transforms given by (6.8) and (6.9), respectively, including the windowing operation and the normalization factor $\frac{4}{N}$ can be written in the equivalent matrix-vector form as

$$\mathbf{c}^T = \frac{4}{N}\, \boldsymbol{C}_{\frac{N}{2}\times N}\, \boldsymbol{W}_N\, \mathbf{x}^T, \tag{6.59}$$

$$\hat{\mathbf{x}}^T = \boldsymbol{W}_N\, [\boldsymbol{C}_{\frac{N}{2}\times N}]^T\, \mathbf{c}^T. \tag{6.60}$$

Based on the matrix representation of the MDCT it was shown in [50] that the transposed MDCT matrix $\left[\boldsymbol{C}_{\frac{N}{2}\times N}\right]^T$ denoted by $\boldsymbol{C}^{+}_{N\times\frac{N}{2}}$ is the pseudoinverse of its corresponding forward transform matrix. Hence, the forward and backward MDCT block transforms are actually the pseudoinverse pair (see Appendix A.1).

The pseudoinverse matrix [51, 52] and its properties provide an elegant mathematical tool to characterize the MDCT, and in general, any perfect reconstruction cosine-/sine-modulated filter bank in a matrix representation. If we consider the forward MDCT given by (6.59) to be overdetermined systems of linear equations, then the time domain aliased data samples $\{\hat{x}_n\}$ in (6.60) for given MDCT coefficients can be interpreted as least squares solutions, i.e., solutions with minimum norm [51]. For products of the matrix $\boldsymbol{C}_{\frac{N}{2}\times N}$ and its pseudoinverse, $\boldsymbol{C}^{+}_{N\times\frac{N}{2}}$, the following relations hold:

$$\boldsymbol{C}_{\frac{N}{2}\times N}\, \boldsymbol{C}^{+}_{N\times\frac{N}{2}} = \frac{N}{2}\, \boldsymbol{I}_{\frac{N}{2}}, \tag{6.61}$$

and

$$\boldsymbol{C}^{+}_{N\times\frac{N}{2}}\, \boldsymbol{C}_{\frac{N}{2}\times N} = \frac{N}{4} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & -\boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{0} & \boldsymbol{0} \\ -\boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{J}_{\frac{N}{4}} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} = \frac{N}{4} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} - \boldsymbol{J}_{\frac{N}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{\frac{N}{2}} + \boldsymbol{J}_{\frac{N}{2}} \end{pmatrix}, \tag{6.62}$$

where $\boldsymbol{I}_{\frac{N}{2}}$ is the identity matrix, $\boldsymbol{J}_{\frac{N}{2}}$ is the reverse ordered identity matrix both of order $\frac{N}{2}$, and $\boldsymbol{0}'s$ are null matrices. For clarity, the matrix $\boldsymbol{C}_{\frac{N}{2}\times N}$ in explicit form for $N = 8$ is given by

$$\boldsymbol{C}_{4\times 8}=\begin{pmatrix} \cos\frac{5\pi}{16} & \cos\frac{7\pi}{16} & -\cos\frac{7\pi}{16} & -\cos\frac{5\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\cos\frac{3\pi}{16} \\ -\cos\frac{\pi}{16} & -\cos\frac{5\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{\pi}{16} & \cos\frac{7\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & \cos\frac{7\pi}{16} \\ \cos\frac{7\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{7\pi}{16} & \cos\frac{\pi}{16} & -\cos\frac{5\pi}{16} & -\cos\frac{5\pi}{16} & \cos\frac{\pi}{16} \\ \cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \cos\frac{\pi}{16} & -\cos\frac{3\pi}{16} & \cos\frac{5\pi}{16} & -\cos\frac{7\pi}{16} & -\cos\frac{7\pi}{16} & \cos\frac{5\pi}{16} \end{pmatrix}.$$

6.4.3 *Forward/Backward First and Second Short Transforms*

Consider the forward and backward first/second short block transforms defined by (6.10)/(6.18) and (6.11)/(6.19), respectively. Let the cosine transform kernels of the first and second short transforms in (6.10) and (6.18), respectively, be represented by the $\frac{M}{2}\times M$ matrices $\boldsymbol{C}^{(1)}_{\frac{M}{2}\times M}$ and $\boldsymbol{C}^{(2)}_{\frac{M}{2}\times M}$ with elements

$$\left\{\boldsymbol{C}^{(1)}_{\frac{M}{2}\times M}\right\}_{k,n}=\cos\left[\frac{\pi}{2M}(2n+1)(2k+1)\right],$$

$$\left\{\boldsymbol{C}^{(2)}_{\frac{M}{2}\times M}\right\}_{k,n}=\cos\left[\frac{\pi}{2M}(2n+1+M)(2k+1)\right],$$

$$k=0,1,\ldots,\frac{M}{2}-1,\quad n=0,1,\ldots,M-1. \tag{6.63}$$

Using the notation introduced in the previous subsection, the forward first and second short transforms given by (6.10) and (6.18), respectively, including the windowing operation and the normalization factor $\frac{4}{M}$ can be written in the equivalent matrix-vector form as

$$[\mathbf{c}^{(-1)}]^T=\frac{4}{M}\,\boldsymbol{C}^{(1)}_{\frac{M}{2}\times M}\,\boldsymbol{W}^{(1)}_{\frac{N}{2}}\,[\mathbf{x}^{(c)}]^T, \tag{6.64}$$

$$[\mathbf{c}^{(+1)}]^T=\frac{4}{M}\,\boldsymbol{C}^{(2)}_{\frac{M}{2}\times M}\,\boldsymbol{W}^{(2)}_{\frac{N}{2}}\,[\mathbf{x}^{(s)}]^T, \tag{6.65}$$

while the backward first and second short transforms given by (6.11) and (6.19), respectively, including the windowing operation can be written as

$$[\hat{\mathbf{x}}^{(-1)}]^T=\boldsymbol{W}^{(1)}_{\frac{N}{2}}\,[\boldsymbol{C}^{(1)}_{\frac{M}{2}\times M}]^T\,[\mathbf{c}^{(-1)}]^T, \tag{6.66}$$

$$[\hat{\mathbf{x}}^{(+1)}]^T=\boldsymbol{W}^{(2)}_{\frac{N}{2}}\,[\boldsymbol{C}^{(2)}_{\frac{M}{2}\times M}]^T\,[\mathbf{c}^{(+1)}]^T. \tag{6.67}$$

Similarly as for the forward/backward MDCT block transforms, it can be shown that the forward/backward first and second short block transforms are actually the pseudoinverse pairs (see Appendix A.1). In fact, for the product of matrix $\boldsymbol{C}^{(1)}_{\frac{M}{2}\times M}$ and its pseudoinverse denoted by $[\boldsymbol{C}^{(1)}_{M\times\frac{M}{2}}]^{+}$, as well as for the product of matrix $\boldsymbol{C}^{(2)}_{\frac{M}{2}\times M}$ and its pseudoinverse denoted by $[\boldsymbol{C}^{(2)}_{M\times\frac{M}{2}}]^{+}$, the following relations hold:

$$\boldsymbol{C}^{(1)}_{\frac{M}{2}\times M}\,[\boldsymbol{C}^{(1)}_{M\times\frac{M}{2}}]^{+} = \boldsymbol{C}^{(2)}_{\frac{M}{2}\times M}\,[\boldsymbol{C}^{(2)}_{M\times\frac{M}{2}}]^{+} = \frac{M}{2}\,\boldsymbol{I}_{\frac{M}{2}}, \tag{6.68}$$

where $\boldsymbol{I}_{\frac{M}{2}}$ is the identity matrix of order $\frac{M}{2}$, and

$$[\boldsymbol{C}^{(1)}_{M\times\frac{M}{2}}]^{+}\,\boldsymbol{C}^{(1)}_{\frac{M}{2}\times M} = \frac{M}{4}\begin{pmatrix} \boldsymbol{I}_{\frac{M}{2}} & -\boldsymbol{J}_{\frac{M}{2}} \\ -\boldsymbol{J}_{\frac{M}{2}} & \boldsymbol{I}_{\frac{M}{2}} \end{pmatrix} = \frac{M}{4}\,(\boldsymbol{I}_M - \boldsymbol{J}_M), \tag{6.69}$$

$$[\boldsymbol{C}^{(2)}_{M\times\frac{M}{2}}]^{+}\,\boldsymbol{C}^{(2)}_{\frac{M}{2}\times M} = \frac{M}{4}\begin{pmatrix} \boldsymbol{I}_{\frac{M}{2}} & \boldsymbol{J}_{\frac{M}{2}} \\ \boldsymbol{J}_{\frac{M}{2}} & \boldsymbol{I}_{\frac{M}{2}} \end{pmatrix} = \frac{M}{4}\,(\boldsymbol{I}_M + \boldsymbol{J}_M), \tag{6.70}$$

where $\boldsymbol{I}_M$ is the identity matrix and $\boldsymbol{J}_M$ is the reverse ordered identity matrix both of order M. For clarity, the matrices $\boldsymbol{C}^{(1)}_{\frac{M}{2}\times M}$ and $\boldsymbol{C}^{(2)}_{\frac{M}{2}\times M}$ in explicit forms for $M = 4$ are given by

$$\boldsymbol{C}^{(1)}_{2\times 4} = \begin{pmatrix} \cos\frac{\pi}{8} & \cos\frac{3\pi}{8} & -\cos\frac{3\pi}{8} & -\cos\frac{\pi}{8} \\ \cos\frac{3\pi}{8} & -\cos\frac{\pi}{8} & \cos\frac{\pi}{8} & -\cos\frac{3\pi}{8} \end{pmatrix},$$

$$\boldsymbol{C}^{(2)}_{2\times 4} = \begin{pmatrix} -\cos\frac{3\pi}{8} & -\cos\frac{\pi}{8} & -\cos\frac{\pi}{8} & -\cos\frac{3\pi}{8} \\ \cos\frac{\pi}{8} & -\cos\frac{3\pi}{8} & -\cos\frac{3\pi}{8} & \cos\frac{\pi}{8} \end{pmatrix}.$$

6.4.4 Useful Relations Among the AC-3 Transform Matrices

Denote by $t_{k,n} = \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right]$ the elements of matrix $\boldsymbol{C}_{\frac{N}{2}\times N}$ given by (6.58). We recall that the k-th row basis vector of $\boldsymbol{C}_{\frac{N}{2}\times N}$ in the first half possesses the even anti-symmetry and in the second half possesses the even symmetry property, respectively, given by

$$t_{k,\frac{N}{2}-1-n} = -\,t_{k,n}, \qquad t_{k,\frac{N}{2}+n} = t_{k,N-1-n},$$
$$\forall\ k, \quad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{6.71}$$

Similarly, denoting by $t^{(1)}_{k,n} = \cos\left[\frac{\pi}{2M}(2n+1)(2k+1)\right]$ the elements of matrix $\boldsymbol{C}^{(1)}_{\frac{M}{2}\times M}$ given by (6.63), and denoting by $t^{(2)}_{k,n} = \cos\left[\frac{\pi}{2M}(2n+1+M)(2k+1)\right]$ the elements of matrix $\boldsymbol{C}^{(2)}_{\frac{M}{2}\times M}$ given by (6.63), it can be verified by proper substitution that the k-th row basis vector of $\boldsymbol{C}^{(1)}_{\frac{M}{2}\times M}$ exhibits the even anti-symmetry property given by

$$t^{(1)}_{k,M-1-n} = -\,t^{(1)}_{k,n}, \quad \forall\ k, \quad n = 0, 1, \ldots, \frac{M}{2} - 1, \tag{6.72}$$

while the k-th row basis vector of $\boldsymbol{C}^{(2)}_{\frac{M}{2}\times M}$ exhibits the even symmetry property given by

$$t^{(2)}_{k,M-1-n} = t^{(2)}_{k,n}, \quad \forall\ k, \quad n = 0, 1, \ldots, \frac{M}{2} - 1. \tag{6.73}$$

The even anti-symmetry and even symmetry properties of row basis vectors given by (6.72) and (6.73) are quite similar to those of time domain aliased data sequences $\{\hat{x}^{(-1)}_n\}$ and $\{\hat{x}^{(+1)}_n\}$ given by (6.13) and (6.21), respectively.

Now, let the matrix $\boldsymbol{C}_{\frac{N}{2}\times N}$ given by (6.58) and its pseudoinverse, $\boldsymbol{C}^{+}_{N\times\frac{N}{2}}$, be split into two blocks defined as

$$\boldsymbol{C}_{\frac{N}{2}\times N} = \left(\boldsymbol{K}_{\frac{N}{2}}\ \boldsymbol{L}_{\frac{N}{2}}\right), \quad \boldsymbol{C}^{+}_{N\times\frac{N}{2}} = \begin{pmatrix} \boldsymbol{K}^{+}_{\frac{N}{2}} \\ \boldsymbol{L}^{+}_{\frac{N}{2}} \end{pmatrix}, \tag{6.74}$$

where $\boldsymbol{K}_{\frac{N}{2}}$, $\boldsymbol{L}_{\frac{N}{2}}$, $\boldsymbol{K}^{+}_{\frac{N}{2}}$ and $\boldsymbol{L}^{+}_{\frac{N}{2}}$ are square nonsingular matrices of order $\frac{N}{2}$ with elements

$$\{\boldsymbol{K}_{\frac{N}{2}}\}_{k,n} = \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$

$$\{\boldsymbol{L}_{\frac{N}{2}}\}_{k,n} = \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{3N}{2}\right)(2k+1)\right], k,\ n = 0, 1, \ldots, \frac{N}{2} - 1,$$

$$\{\boldsymbol{K}_{\frac{N}{2}}^{+}\}_{n,k} = \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$

$$\{\boldsymbol{L}_{\frac{N}{2}}^{+}\}_{n,k} = \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{3N}{2}\right)(2k+1)\right],$$

$$n,\ k = 0, 1, \ldots, \frac{N}{2}-1. \tag{6.75}$$

From (6.71) it follows that the k-th column basis vector of $\boldsymbol{K}_{\frac{N}{2}}^{+}$ is even anti-symmetric, whereas the k-th column basis vector of $\boldsymbol{L}_{\frac{N}{2}}^{+}$ is even symmetric, and immediately from (6.62) we have

$$\boldsymbol{K}_{\frac{N}{2}}^{+}\,\boldsymbol{K}_{\frac{N}{2}} = \frac{N}{4}\begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & -\boldsymbol{J}_{\frac{N}{4}} \\ -\boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} = \frac{N}{4}\,(\boldsymbol{I}_{\frac{N}{2}} - \boldsymbol{J}_{\frac{N}{2}}), \tag{6.76}$$

$$\boldsymbol{L}_{\frac{N}{2}}^{+}\,\boldsymbol{L}_{\frac{N}{2}} = \frac{N}{4}\begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{J}_{\frac{N}{4}} \\ \boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} = \frac{N}{4}\,(\boldsymbol{I}_{\frac{N}{2}} + \boldsymbol{J}_{\frac{N}{2}}). \tag{6.77}$$

Finally, comparing (6.76) to (6.69) and (6.77) to (6.70) for $M = \frac{N}{2}$ we get

$$\left[\boldsymbol{C}_{M\times\frac{M}{2}}^{(1)}\right]^{+}\,\boldsymbol{C}_{\frac{M}{2}\times M}^{(1)} = \frac{1}{2}\,\boldsymbol{K}_{M}^{+}\,\boldsymbol{K}_{M}, \qquad \left[\boldsymbol{C}_{M\times\frac{M}{2}}^{(2)}\right]^{+}\,\boldsymbol{C}_{\frac{M}{2}\times M}^{(2)} = \frac{1}{2}\,\boldsymbol{L}_{M}^{+}\,\boldsymbol{L}_{M}. \tag{6.78}$$

Let us consider the following matrix products:

$$\boldsymbol{C}_{\frac{M}{2}\times M}^{(1)}\boldsymbol{W}_{\frac{N}{2}}^{(1)}\boldsymbol{W}_{\frac{N}{2}}^{(2)}\left[\boldsymbol{C}_{M\times\frac{M}{2}}^{(2)}\right]^{+}, \qquad \boldsymbol{C}_{\frac{M}{2}\times M}^{(2)}\boldsymbol{W}_{\frac{N}{2}}^{(1)}\boldsymbol{W}_{\frac{N}{2}}^{(2)}\left[\boldsymbol{C}_{M\times\frac{M}{2}}^{(1)}\right]^{+},$$

$$\boldsymbol{C}_{\frac{M}{2}\times M}^{(1)}\boldsymbol{W}_{\frac{N}{2}}^{(2)}\,\boldsymbol{W}_{\frac{N}{2}}^{(1)}\boldsymbol{L}_{\frac{N}{2}}^{+}, \qquad \boldsymbol{C}_{\frac{M}{2}\times M}^{(2)}\boldsymbol{W}_{\frac{N}{2}}^{(2)}\,\boldsymbol{W}_{\frac{N}{2}}^{(1)}\boldsymbol{K}_{\frac{N}{2}}^{+}.$$

Since according to (6.71), (6.72), (6.73), (6.74), and (6.75) the row/column basis vectors of matrices in the corresponding matrix products are either even anti-symmetric/symmetric or even symmetric/anti-symmetric, and due to the symmetric property of windowing function, the matrix products satisfy the following relations:

$$\boldsymbol{C}_{\frac{M}{2}\times M}^{(1)}\,\boldsymbol{W}_{\frac{N}{2}}^{(1)}\,\boldsymbol{W}_{\frac{N}{2}}^{(2)}\left[\boldsymbol{C}_{M\times\frac{M}{2}}^{(2)}\right]^{+} = \boldsymbol{C}_{\frac{M}{2}\times M}^{(2)}\,\boldsymbol{W}_{\frac{N}{2}}^{(1)}\,\boldsymbol{W}_{\frac{N}{2}}^{(2)}\left[\boldsymbol{C}_{M\times\frac{M}{2}}^{(1)}\right]^{+} = \boldsymbol{0}_{\frac{M}{2}}, \tag{6.79}$$

$$\boldsymbol{C}_{\frac{M}{2}\times M}^{(1)}\,\boldsymbol{W}_{\frac{N}{2}}^{(2)}\,\boldsymbol{W}_{\frac{N}{2}}^{(1)}\,\boldsymbol{L}_{\frac{N}{2}}^{+} = \boldsymbol{C}_{\frac{M}{2}\times M}^{(2)}\,\boldsymbol{W}_{\frac{N}{2}}^{(2)}\,\boldsymbol{W}_{\frac{N}{2}}^{(1)}\,\boldsymbol{K}_{\frac{N}{2}}^{+} = \boldsymbol{0}_{\frac{M}{2}\times\frac{N}{2}}, \tag{6.80}$$

where $\mathbf{0}_{\frac{M}{2}}$ and $\mathbf{0}_{\frac{M}{2}\times\frac{N}{2}}$ are null matrices. Relations (6.79) and (6.80) arise from the simple fact that the scalar product of even anti-symmetric/symmetric or even symmetric/anti-symmetric nonzero vectors is always zero. In fact, the matrix products $\boldsymbol{W}_{\frac{N}{2}}^{(1)}\ \boldsymbol{W}_{\frac{N}{2}}^{(2)}$ and $\boldsymbol{W}_{\frac{N}{2}}^{(2)}\ \boldsymbol{W}_{\frac{N}{2}}^{(1)}$ in (6.79) and (6.80) do not disturb the symmetry properties of row/column basis vectors. A slightly different proof of (6.79) and (6.80) is presented in [31].

Given the frequency coefficients of the long (MDCT) transform or given the frequency coefficients of two short transforms in the frequency domain. Does exist a relation between the frequency coefficients of long transform and those of two short transforms, and vice versa?

6.5 Relations Between the Frequency Coefficients and the Time Domain Aliasing Data Sequences of AC-3 Transforms

Based on the matrix representation of AC-3 transforms, a systematic investigation of their properties and relations among transform matrices enables us to derive a relation between the frequency coefficients of the long (MDCT) and those of two short transforms [29, 31]. This relation has an impact on the current implementation of AC-3 analysis filter banks which can be simplified in the encoder. Similarly, a simple relation between the aliased data sequence recovered by the backward long (MDCT) and those of two short transforms can be derived. This relation shows why the perfect reconstruction property between the long (MDCT) and two short transforms is maintained although phase shifts of cosine transform kernels of two short transforms are relatively different to that of the long (MDCT) transform [29]. Again, this relation has an impact on the current implementation of AC-3 synthesis filter banks which can be simplified in the decoder.

6.5.1 Relation Between Frequency Coefficients of the Long and Two Short Transforms

Consider the windowing and overlap-add procedure in the AC-3 decoder to reconstruct the current data block denoted by $\mathbf{x}_t = \left[\mathbf{x}_t^{(c)},\ \mathbf{x}_t^{(s)}\right]$ where $\mathbf{x}_t^{(c)}$ is the first half and $\mathbf{x}_t^{(s)}$ is the second half of $\mathbf{x}_t$, and the subscript t denotes the data-block number. Further, let $\mathbf{c}_t^T$, $\left[\mathbf{c}_t^{(-1)}\right]^T$ and $\left[\mathbf{c}_t^{(+1)}\right]^T$ be the column vectors representing the frequency coefficients of the long (MDCT), the first and second short transforms of the tth transformed data block, respectively. Since the AC-3 transforms satisfy the perfect reconstruction constraints in the overlapped parts of two adjacent data blocks,

the tth data block $\mathbf{x}_t = \left[\mathbf{x}_t^{(c)},\ \mathbf{x}_t^{(s)}\right]$ can be perfectly reconstructed by the windowing and overlap-add procedure formulated in the matrix-vector representation as follows (note that the tth current data block which is processed is the long block) [31]:

$$[\mathbf{x}_t^{(c)}]^T = \begin{cases} \boldsymbol{W}_{\frac{N}{2}}^{(1)}\ \boldsymbol{K}_{\frac{N}{2}}^{+}\ \mathbf{c}_t^T + \boldsymbol{W}_{\frac{N}{2}}^{(2)}\left[\boldsymbol{C}_{M\times\frac{M}{2}}^{(2)}\right]^{+}\left[\mathbf{c}_{t-1}^{(+1)}\right]^T, & \text{if in the previous data block} \\ & \text{the short transforms are adopted,} \\ \boldsymbol{W}_{\frac{N}{2}}^{(1)}\ \boldsymbol{K}_{\frac{N}{2}}^{+}\ \mathbf{c}_t^T + \boldsymbol{W}_{\frac{N}{2}}^{(2)}\ \boldsymbol{L}_{\frac{N}{2}}^{+}\ \mathbf{c}_{t-1}^T, & \text{if in the previous data block} \\ & \text{the long transform is adopted,} \end{cases} \tag{6.81}$$

$$[\mathbf{x}_t^{(s)}]^T = \begin{cases} \boldsymbol{W}_{\frac{N}{2}}^{(2)}\ \boldsymbol{L}_{\frac{N}{2}}^{+}\ \mathbf{c}_t^T + \boldsymbol{W}_{\frac{N}{2}}^{(1)}\left[\boldsymbol{C}_{M\times\frac{M}{2}}^{(1)}\right]^{+}\left[\mathbf{c}_{t+1}^{(-1)}\right]^T, & \text{if in the following data block the} \\ & \text{short transforms are adopted,} \\ \boldsymbol{W}_{\frac{N}{2}}^{(2)}\ \boldsymbol{L}_{\frac{N}{2}}^{+}\ \mathbf{c}_t^T + \boldsymbol{W}_{\frac{N}{2}}^{(1)}\ \boldsymbol{K}_{\frac{N}{2}}^{+}\ \mathbf{c}_{t+1}^T, & \text{if in the following data block the long} \\ & \text{transform is adopted.} \end{cases} \tag{6.82}$$

According to (6.64) and (6.65), the frequency coefficients of two short transforms in the tth data block can be expressed in the matrix-vector form as

$$\begin{pmatrix} \left[\mathbf{c}_t^{(-1)}\right]^T \\ \left[\mathbf{c}_t^{(+1)}\right]^T \end{pmatrix} = \frac{4}{M}\begin{pmatrix} \boldsymbol{C}_{\frac{M}{2}\times M}^{(1)}\ \boldsymbol{W}_{\frac{N}{2}}^{(1)} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{C}_{\frac{M}{2}\times M}^{(2)}\ \boldsymbol{W}_{\frac{N}{2}}^{(2)} \end{pmatrix}\begin{pmatrix} \left[\mathbf{x}_t^{(c)}\right]^T \\ \left[\mathbf{x}_t^{(s)}\right]^T \end{pmatrix}. \tag{6.83}$$

Substituting the expressions from (6.81) and (6.82) into the right-hand side of (6.83) for $[\mathbf{x}_t^{(c)}]^T$ and $[\mathbf{x}_t^{(s)}]^T$, using relations (6.79) and (6.80) and noting that the current block is the long block, after some algebraic manipulations we obtain [31]

$$\begin{pmatrix} \left[\mathbf{c}_t^{(-1)}\right]^T \\ \left[\mathbf{c}_t^{(+1)}\right]^T \end{pmatrix} = \frac{4}{N}\begin{pmatrix} \boldsymbol{C}_{\frac{M}{2}\times M}^{(1)}\ \boldsymbol{W}_{\frac{N}{2}}^{(1)}\ \boldsymbol{W}_{\frac{N}{2}}^{(1)}\ \boldsymbol{K}_{\frac{N}{2}}^{+} \\ \boldsymbol{C}_{\frac{M}{2}\times M}^{(2)}\ \boldsymbol{W}_{\frac{N}{2}}^{(2)}\ \boldsymbol{W}_{\frac{N}{2}}^{(2)}\ \boldsymbol{L}_{\frac{N}{2}}^{+} \end{pmatrix}\mathbf{c}_t^T. \tag{6.84}$$

Equation (6.84) defines the relation between the frequency coefficients of the long (MDCT) transform and two short transforms in the tth data block. Given frequency coefficients of the long (MDCT) transform. Then, the frequency coefficients of two short transforms can be simply obtained from those of the long (MDCT) transform via a conversion matrix defined on the right-hand side of (6.84). Note that the authors in [31] did not further investigate the mathematical properties and general structure of the conversion matrix in detail.

The existence of conversion matrix has an impact on the current implementation of AC-3 analysis filter banks which can be simplified in the encoder. A fast computational structure for the forward long (MDCT) transform computation is required only, which can run in parallel with the transient detector. If a transient is detected, the frequency coefficients of the long (MDCT) transform can be converted to those of the short transforms via the conversion matrix. However, it is not necessary. The frequency coefficients of the long (MDCT) transform may be further transmitted with code bit indicating the presence/absence of transient. Thus, the fast computational structure for the forward short transforms computation with associated memory tables for the cosine/sine twiddle factors (128 table values) [6] can be completely eliminated so saving memory resources. On the other hand, in the AC-3 decoder the frequency coefficients of two short transforms can be converted to those of the long (MDCT) transform via the transposed conversion matrix, if it is required. However, as we will see in the next subsection, the conversion matrix may not be used at all.

6.5.2 *Relation Between Time Domain Aliased Data Sequences Recovered by the Backward Long and Two Short Transforms*

Consider the forward/backward long (MDCT) transform in the matrix-vector form defined by (6.59)/(6.60). Substituting (6.59) into (6.60) (i.e., performing the forward and backward MDCT) and using relation (6.62) we have

$$\begin{aligned}\hat{\mathbf{x}}^T &= \frac{4}{N}\, \boldsymbol{W}_N\, \boldsymbol{C}^{+}_{N\times\frac{N}{2}}\, \boldsymbol{C}_{\frac{N}{2}\times N}\, \boldsymbol{W}_N \\ \mathbf{x}^T &= \boldsymbol{W}_N \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} - \boldsymbol{J}_{\frac{N}{2}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_{\frac{N}{2}} + \boldsymbol{J}_{\frac{N}{2}} \end{pmatrix} \boldsymbol{W}_N\, \mathbf{x}^T. \end{aligned} \tag{6.85}$$

Based on (6.85), in [50] it has been shown that the recovered time domain aliased data sequence $\{\hat{x}_n\}$ can be derived in terms of the original data sequence $\{x_n\}$.

On the other hand, consider the forward/backward short transforms in the matrix-vector form defined by (6.64)/(6.66) and (6.65)/(6.67). Substituting (6.64)/(6.65) into (6.66)/(6.67) (i.e., performing the forward and backward first/second short transforms) and using relations (6.69)/(6.70) we have

$$\begin{aligned}\left[\hat{\mathbf{x}}^{(-1)}\right]^T &= \frac{4}{M}\, \boldsymbol{W}^{(1)}_M\, \left[\boldsymbol{C}^{(1)}_{M\times\frac{M}{2}}\right]^{+}\, \boldsymbol{C}^{(1)}_{\frac{M}{2}\times M}\, \boldsymbol{W}^{(1)}_{\frac{N}{2}}\, \left[\mathbf{x}^{(c)}\right]^T \\ &= \boldsymbol{W}^{(1)}_M\, (\boldsymbol{I}_M - \boldsymbol{J}_M)\, \boldsymbol{W}^{(1)}_M\, \left[\mathbf{x}^{(c)}\right]^T, \end{aligned} \tag{6.86}$$

$$\left[\hat{\mathbf{x}}^{(+1)}\right]^T = \frac{4}{M}\, \boldsymbol{W}_M^{(2)} \left[\boldsymbol{C}_{M\times\frac{M}{2}}^{(2)}\right]^+ \boldsymbol{C}_{\frac{M}{2}\times M}^{(2)}\, \boldsymbol{W}_M^{(2)} \left[\mathbf{x}^{(s)}\right]^T$$

$$= \boldsymbol{W}_M^{(2)}\, (\boldsymbol{I}_M + \boldsymbol{J}_M)\, \boldsymbol{W}_M^{(2)} \left[\mathbf{x}^{(s)}\right]^T, \tag{6.87}$$

where $\mathbf{x}^T = \left[\mathbf{x}^{(c)}, \mathbf{x}^{(s)}\right]^T$ and $\hat{\mathbf{x}}^T = \left[\hat{\mathbf{x}}^{(-1)}, \hat{\mathbf{x}}^{(+1)}\right]^T$. In (6.86) and (6.87) we can clearly observe that the corresponding time domain aliased data sequences $\{\hat{x}^{(-1)}\}$ and $\{\hat{x}^{(+1)}\}$ can be respectively derived in terms of the input data sequences $\{x^{(c)}\}$ and $\{x^{(s)}\}$, and hence, in terms of the original data sequence $\{x_n\}$ too. Indeed, the block matrices on the right-hand sides of (6.86) and (6.87) for $M = \frac{N}{2}$ coincide with those of on the right-hand side of (6.85). Using (6.5) and (6.74), substituting $M = \frac{N}{2}$ into (6.85) and performing block matrix products we get

$$\begin{pmatrix} \left[\hat{\mathbf{x}}^{(-1)}\right]^T \\ \\ \left[\hat{\mathbf{x}}^{(+1)}\right]^T \end{pmatrix} = \begin{pmatrix} \boldsymbol{W}_M^{(1)}\, (\boldsymbol{I}_M - \boldsymbol{J}_M)\, \boldsymbol{W}_M^{(1)} & \mathbf{0} \\ \\ \mathbf{0} & \boldsymbol{W}_M^{(2)}\, (\boldsymbol{I}_M + \boldsymbol{J}_M)\, \boldsymbol{W}_M^{(2)} \end{pmatrix} \begin{pmatrix} \left[\mathbf{x}^{(c)}\right]^T \\ \\ \left[\mathbf{x}^{(s)}\right]^T \end{pmatrix}. \tag{6.88}$$

Comparing (6.88) and Eqs. (6.86) and (6.87) concatenated into one matrix form it can be seen that they are equivalent. Finally, exploiting relations (6.69), (6.70), (6.76), (6.77), and (6.78) reveals a close relation between the time domain aliased data sequence recovered by the backward long (MDCT) and those of two short transforms given by

$$\hat{x}_n^{(-1)} = \frac{1}{2}\, \hat{x}_n, \qquad \hat{x}_n^{(+1)} = \frac{1}{2}\, \hat{x}_{\frac{N}{2}+n}, \qquad n = 0, 1, \ldots, M-1. \tag{6.89}$$

Equation (6.89) also implies that although phase shifts of cosine transform kernels of two short transforms are relatively different from that of the long (MDCT) transform, the perfect reconstruction property between the long (MDCT) and two short transforms is maintained.

The relations given by (6.89) have an impact on the current implementation of AC-3 synthesis filter banks which can be simplified in the decoder. A fast computational structure for the backward long (MDCT) transform computation is required only. If the presence of a transient is indicated, the time domain aliased data samples recovered by the backward long (MDCT) transform can be windowed and converted to those of recovered and windowed by the backward short transforms and vice versa using (6.89). It means that the conversion matrix in the AC-3 encoder and decoder may not be used at all.

6.6 Fast Algorithm for Conversion of Frequency Coefficients of AC-3 Transforms

We recall that although the E-AC-3 bit streams are similar in nature to AC-3 ones (they use the same MDCT filter bank, bit-allocation process, and framing structure), are not backwards compatible, i.e., they are not decodable by AC-3 decoders [7]. Therefore, the Dolby Laboratories developed an efficient method to convert an E-AC-3 bit stream to an AC-3 one, the so-called E-AC-3 to AC-3 conversion, to ensure the compatibility with the large installed base of AC-3 decoders. The conversion procedure is designed to minimize loss in audio quality while keeping the complexity at a level suitable for low-cost consumer devices [10, 11]. On the other hand, the so-called AC-3 to E-AC-3 transcoding, is used to distribute 5.1-channel audio content that has already been encoded in the AC-3. The E-AC-3 bit stream is created by transcoding an AC-3 bit stream to an E-AC-3 bit stream at lower bit rate [10, 11].

In the following subsection, standard methods for conversion of frequency coefficients including their arithmetic complexity and memory requirements are described. They are used in the current E-AC-3 to/from AC-3 bit stream conversion/transcoding.

6.6.1 Standard Methods for Conversion of Frequency Coefficients

We recall that for time-to-frequency transformation of an audio data block and vice versa, both the AC-3 and E-AC-3 use the identical long (MDCT) transform while the AC-3 uses the additional two short transforms when a transient signal is detected. We note that for efficient implementations of AC-3 transforms, the M-point ($M = 2^m$) DCT-IV requires $\frac{M}{2}(m+2)$ real multiplications and $\frac{3M}{2}m$ real additions, while $\frac{M}{2}$-point DCT-IV requires $\frac{M}{4}(m+1)$ real multiplications and $\frac{3M}{4}(m-1)$ real additions. The use of different filter banks has an impact on the E-AC-3 to/from AC-3 bit stream conversion/transcoding.

The E-AC-3 employs always the long data blocks. For the E-AC-3 to AC-3 bit stream conversion, when the current block is long without transient signal, then no conversion for the AC-3. However, when the long data block contains a transient signal, the frequency coefficients of the long (MDCT) transform have to be converted to those of two short transforms for the AC-3 to cancel pre-echo effects. The standard conversion method (see Fig. 6.7) involves the computation of three backward long transforms (of previous, current and following blocks), three (complete) windowing procedures, two overlap/add procedures, and two forward short transforms. This requires $\frac{M}{2}(4m+19)$ real multiplications, $\frac{3M}{2}(4m+1)$ real additions, and $2M$ memory locations.

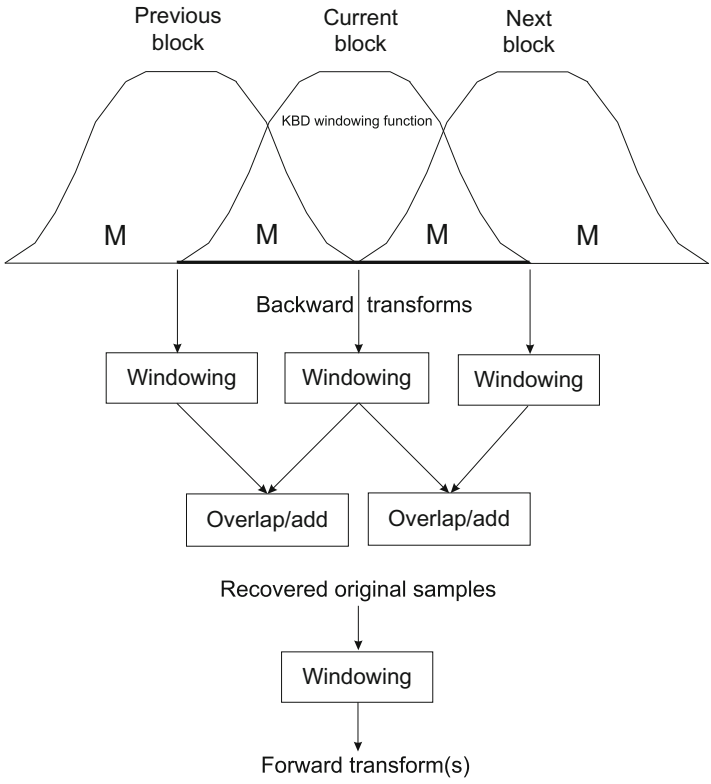

Fig. 6.7 Standard method for conversion of frequency coefficients

On the other hand, for the AC-3 to E-AC-3 bit stream transcoding, when the current block is long, then no conversion for the E-AC-3. However, when the frequency coefficients of AC-3 two short transforms are available, then they have to be converted to those of the long (MDCT) transform for the E-AC-3. The standard conversion method (see Fig. 6.7) involves the computation of two backward short transforms plus two backward long transforms in the worst case (provided by the previous and following blocks are long), or two backward short transforms in the best case (provided by the previous and following blocks are short), three (complete) windowing procedures, two overlap/add procedures, and finally one forward long (MDCT) transform. This requires in the worst case similarly $\frac{M}{2}(4m + 19)$ real multiplications and $\frac{3M}{2}(4m + 1)$ real additions, while in the best case $\frac{M}{2}(3m + 16)$ real multiplications, $\frac{3M}{2}$ $3m$ real additions, and in both cases $2M$ memory locations.

In general, it can be seen that the standard methods for conversion of frequency coefficients involve a partial decoding and encoding during the conversion processes. The existence of relation between transform coefficients via the conversion matrix enables to realize the E-AC-3 to/from AC-3 bit stream

conversion/transcoding in a simplified and more transparent way, thus minimizing the amount of partial decoding and encoding during the conversion processes.

6.6.2 The Conversion Matrix and Its Properties

In the following discussion for simplicity we drop the subscript t in (6.84). Denoting the square nonsingular conversion matrix by $\boldsymbol{V}_M$, its upper and lower block sub-matrices $\boldsymbol{V}^{(1)}_{\frac{M}{2}\times\frac{N}{2}}$ and $\boldsymbol{V}^{(2)}_{\frac{M}{2}\times\frac{N}{2}}$, respectively, by

$$\begin{aligned}\boldsymbol{V}^{(1)}_{\frac{M}{2}\times\frac{N}{2}} &= \boldsymbol{C}^{(1)}_{\frac{M}{2}\times M}\boldsymbol{W}^{(1)}_{\frac{N}{2}}\boldsymbol{W}^{(1)}_{\frac{N}{2}}\boldsymbol{K}^{+}_{\frac{N}{2}},\\ \boldsymbol{V}^{(2)}_{\frac{M}{2}\times\frac{N}{2}} &= \boldsymbol{C}^{(2)}_{\frac{M}{2}\times M}\boldsymbol{W}^{(2)}_{\frac{N}{2}}\boldsymbol{W}^{(2)}_{\frac{N}{2}}\boldsymbol{L}^{+}_{\frac{N}{2}},\end{aligned}$$

we have

$$\begin{pmatrix}\left[\mathbf{c}^{(-1)}\right]^T\\ \left[\mathbf{c}^{(+1)}\right]^T\end{pmatrix} = \frac{4}{N}\begin{pmatrix}\boldsymbol{V}^{(1)}_{\frac{M}{2}\times\frac{N}{2}}\\ \boldsymbol{V}^{(2)}_{\frac{M}{2}\times\frac{N}{2}}\end{pmatrix}\mathbf{c}^T = \frac{4}{N}\,\boldsymbol{V}_M\,\mathbf{c}^T. \tag{6.90}$$

Now, investigate the elements of $\boldsymbol{V}^{(1)}_{\frac{M}{2}\times\frac{N}{2}}$ and $\boldsymbol{V}^{(2)}_{\frac{M}{2}\times\frac{N}{2}}$ in detail. According to (6.63), (6.71), (6.72), (6.73), and (6.75), and performing the scalar products of row/column basis vectors, the corresponding elements of sub-matrices $\boldsymbol{V}^{(1)}_{\frac{M}{2}\times\frac{N}{2}}$ and $\boldsymbol{V}^{(2)}_{\frac{M}{2}\times\frac{N}{2}}$ are, respectively, given by

$$\begin{aligned}v^{(1)}_{k,m} &= \frac{4}{N}\sum_{n=0}^{\frac{N}{2}-1} w_n^2\,\cos\left[\frac{\pi}{2M}(2n+1)(2k+1)\right]\\ &\quad\times\cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2m+1)\right],\\ v^{(2)}_{k,m} &= \frac{4}{N}\sum_{n=0}^{\frac{N}{2}-1} w^2_{\frac{N}{2}-1-n}\,\cos\left[\frac{\pi}{2M}(2n+1+M)(2k+1)\right]\\ &\quad\times\cos\left[\frac{\pi}{2N}\left(2n+1+\frac{3N}{2}\right)(2m+1)\right],\\ &\quad k=0,1,\ldots,\frac{M}{2}-1,\quad m=0,1,\ldots,\frac{N}{2}-1.\end{aligned} \tag{6.91}$$

Equation (6.91) is based on the simple fact that the scalar product of even anti-symmetric/anti-symmetric or even symmetric/symmetric nonzero vectors is always nonzero. Since between the corresponding cosine terms under sums of (6.91) the following trigonometric identities hold:

$$\begin{aligned}
&\cos\left[\frac{\pi}{2M}(2n+1+M)(2k+1)\right] = (-1)^{k+1}\\
&\sin\left[\frac{\pi}{2M}(2n+1)(2k+1)\right],\\
&\cos\left[\frac{\pi}{2N}\left(2n+1+\frac{3N}{2}\right)(2m+1)\right] = (-1)^{m+1}\\
&\sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2m+1)\right],
\end{aligned} \tag{6.92}$$

substituting $\frac{M}{2}-1-k$ for k into the first sine term, and then substituting $\frac{N}{2}-1-m$ for m into the second sine term on the right-hand side of (6.92) we get

$$\begin{aligned}
&\cos\left[\frac{\pi}{2M}(2n+1+M)(2k+1)\right] = (-1)^{\frac{M}{2}-k}(-1)^n\\
&\cos\left[\frac{\pi}{2M}(2n+1)(2k+1)\right],\\
&\cos\left[\frac{\pi}{2N}\left(2n+1+\frac{3N}{2}\right)(2m+1)\right] = (-1)^{\frac{N}{2}-m}(-1)^{n+\frac{N}{4}}\\
&\cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2m+1)\right],
\end{aligned} \tag{6.93}$$

and (6.91) can be rewritten in the final form as

$$\begin{aligned}
v_{k,m}^{(1)} &= \frac{4}{N}\sum_{n=0}^{\frac{N}{4}-1}\left(w_n^2 + w_{\frac{N}{2}-1-n}^2\right)\cos\left[\frac{\pi}{2M}(2n+1)(2k+1)\right]\\
&\quad\times\cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2m+1)\right],\\
v_{\frac{M}{2}-1-k,\frac{N}{2}-1-m}^{(2)} &= (-1)^{k+m}\,v_{k,m}^{(1)},\\
k &= 0,1,\ldots,\frac{M}{2}-1,\quad m = 0,1,\ldots,\frac{N}{2}-1.
\end{aligned} \tag{6.94}$$

However, since the KBD windowing function is symmetric and satisfies the perfect reconstruction conditions (6.56), the expression under sum of (6.94), $w_n^2 + w_{\frac{N}{2}-1-n}^2 = 1$ (see Eq. (6.56)), may be eliminated. It means that the elements of conversion matrix $\boldsymbol{V}_M$ do not depend on the KBD windowing function. Equation (6.94) implies that the elements $v_{k,m}^{(2)}$ of sub-matrix $\boldsymbol{V}_{\frac{M}{2}\times\frac{N}{2}}^{(2)}$ are the same in magnitude compared to the elements $v_{k,m}^{(1)}$ of sub-matrix $\boldsymbol{V}_{\frac{M}{2}\times\frac{N}{2}}^{(1)}$ except for their reverse ordering and proper sign changes. Hence, only elements in the upper half of the conversion matrix $\boldsymbol{V}_M$ are unique. Therefore, only sub-matrix $\boldsymbol{V}_{\frac{M}{2}\times\frac{N}{2}}^{(1)}$ is sufficient to be precomputed and stored, thus saving memory resources. In fact, the conversion matrix $\boldsymbol{V}_M$ possesses the following regular general block structure:

$$\boldsymbol{V}_M = \begin{pmatrix} \boldsymbol{A}_{\frac{M}{2}}^{(1)} & \boldsymbol{A}_{\frac{M}{2}}^{(2)} \\ \\ (-1)^{\frac{N}{4}} \left(\bar{\boldsymbol{J}}_{\frac{M}{2}} \boldsymbol{A}_{\frac{M}{2}}^{(2)} \bar{\boldsymbol{J}}_{\frac{M}{2}}^{T}\right) & \bar{\boldsymbol{J}}_{\frac{M}{2}}^{T} \boldsymbol{A}_{\frac{M}{2}}^{(1)} \bar{\boldsymbol{J}}_{\frac{M}{2}} \end{pmatrix}, \tag{6.95}$$

where $\boldsymbol{A}_{\frac{M}{2}}^{(1)}$ and $\boldsymbol{A}_{\frac{M}{2}}^{(2)}$ are square sub-matrices both of order $\frac{M}{2}$, and $\bar{\boldsymbol{J}}_{\frac{M}{2}}$ is the matrix of order $\frac{M}{2}$ with alternating ± 1 elements on the opposite main diagonal defined as

$$\bar{\boldsymbol{J}}_{\frac{M}{2}} = \begin{pmatrix} 0 & & & & 1 \\ & & & -1 & \\ & & \cdot & & \\ & \cdot & & & \\ & \cdot & & & \\ & & 1 & & \\ (-1)^{\frac{M}{2}+1} & & & & 0 \end{pmatrix}. \tag{6.96}$$

Note that if data block lengths are powers of 2, the sign changing factor $(-1)^{\frac{N}{4}}$ in (6.95) can be removed, and $\bar{\boldsymbol{J}}_{\frac{M}{2}}^{T} = -\bar{\boldsymbol{J}}_{\frac{M}{2}}$.

In general, between the matrix $\boldsymbol{V}_M$ and its transpose the following relation holds:

$$\boldsymbol{V}_M \, \boldsymbol{V}_M^T = \boldsymbol{V}_M^T \, \boldsymbol{V}_M = \frac{1}{2} \boldsymbol{I}_M, \tag{6.97}$$

i.e., the matrices $\boldsymbol{V}_M$ are orthogonal. If the matrices $\boldsymbol{V}_M$ are properly scaled by the factor $\sqrt{2}$, then they are orthonormal, their determinant is unity and they have ***QR*** and ***PLUS*** factorizations [49, 54]. The scaling factor $\sqrt{2}$ can be simply absorbed into (6.90).

6.6.3 Conversion Procedures in the Matrix-Vector Forms

Given the frequency coefficients of the long (MDCT) transform, they can be converted to the frequency coefficients of two short transforms via the conversion matrix $\boldsymbol{V}_M$ according to

$$\begin{pmatrix} \left[\mathbf{c}^{(-1)}\right]^T \\ \left[\mathbf{c}^{(+1)}\right]^T \end{pmatrix} = \boldsymbol{V}_M \, \mathbf{c}^T, \tag{6.98}$$

or alternatively, by more practical formulae defined as

$$\begin{aligned} c_k^{(-1)} &= \sum_{m=0}^{M-1} v_{k,m}^{(1)} \, c_m, \\ c_{\frac{M}{2}-1-k}^{(+1)} &= \sum_{m=0}^{M-1} (-1)^{k+m} \, v_{k,m}^{(1)} \, c_{M-1-m}, \quad k = 0, 1, \ldots, \frac{M}{2} - 1. \end{aligned} \tag{6.99}$$

In order to obtain the true frequency coefficients of two short transforms $\{c_k^{(-1)}\}$ and $\{c_k^{(+1)}\}$, after conversion they have to be scaled by $\frac{\sqrt{2}}{2}$. Equation (6.99) is valid for any value of $M = \frac{N}{2}$, where N is even integer divisible by 4.

Let the frequency coefficients of two short transforms be given. Using the orthonormality property of the conversion matrix $\boldsymbol{V}_M$, from (6.98) we directly obtain

$$\mathbf{c}^T = \boldsymbol{V}_M^T \begin{pmatrix} \left[\mathbf{c}^{(-1)}\right]^T \\ \left[\mathbf{c}^{(+1)}\right]^T \end{pmatrix}, \tag{6.100}$$

and hence, the frequency coefficients of two short transforms can be converted to those of the long (MDCT) transform via the transposed conversion matrix $\boldsymbol{V}_M^T$, or alternatively, according to the more practical formula defined as

$$c_m = \sum_{k=0}^{\frac{M}{2}-1} v_{k,m}^{(1)} \, c_k^{(-1)} + (-1)^{k+m} \, v_{\frac{M}{2}-1-k,\frac{N}{2}-1-m}^{(1)} \, c_k^{(+1)}, \qquad m = 0, 1, \ldots, M-1. \tag{6.101}$$

Equations (6.98)/(6.100) and (6.99)/(6.101) require M^2 multiplications, $M(M-1)$ additions and half of the matrix $\boldsymbol{V}_M$ (the sub-matrix $\boldsymbol{V}^{(1)}_{\frac{M}{2}\times M}$), i.e., we need to store $\frac{1}{2}M^2$ elements.

The conversion procedures defined in the matrix-vector form are still computationally intensive and have high memory requirements compared to the standard conversion methods. Since the conversion matrix after proper scaling is the orthonormal matrix with very regular general block structure, the open problem as stated in [29] has to be solved: The existence of a generalized sparse block matrix factorization of the conversion matrix $\boldsymbol{V}_M$ which would define a fast conversion algorithm. Such generalized sparse block matrix factorization of the conversion matrix $\boldsymbol{V}_M$ indeed exists [32, 33], and is discussed in the following subsection.

6.6.4 *Fast Algorithm for Conversion of Frequency Coefficients*

Investigate in detail the explicit forms of conversion matrix $\boldsymbol{V}_M$ defined by (6.91) and scaled by the factor $\sqrt{2}$ for $M = 2,\ 4$ and 8. For $M = 2$ we have

$$\boldsymbol{V}_2 = \sqrt{2}\begin{pmatrix} \cos\frac{\pi}{4}\cos\frac{3\pi}{8} & -\cos\frac{\pi}{4}\cos\frac{\pi}{8} \\ \cos\frac{\pi}{4}\cos\frac{\pi}{8} & \cos\frac{\pi}{4}\cos\frac{3\pi}{8} \end{pmatrix} = \begin{pmatrix} \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} \\ \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \end{pmatrix},$$

while for $M = 4$ the conversion matrix $\boldsymbol{V}_4$ given by

$$\frac{\sqrt{2}}{2}\begin{pmatrix} \cos\frac{\pi}{8}\cos\frac{5\pi}{16}+\cos\frac{3\pi}{8}\cos\frac{7\pi}{16} & -\cos\frac{\pi}{8}\cos\frac{\pi}{16}-\cos\frac{3\pi}{8}\cos\frac{5\pi}{16} & \cos\frac{\pi}{8}\cos\frac{7\pi}{16}+\cos\frac{3\pi}{8}\cos\frac{3\pi}{16} & \cos\frac{\pi}{8}\cos\frac{3\pi}{16}-\cos\frac{3\pi}{8}\cos\frac{\pi}{16} \\ \cos\frac{3\pi}{8}\cos\frac{5\pi}{16}-\cos\frac{\pi}{8}\cos\frac{7\pi}{16} & -\cos\frac{3\pi}{8}\cos\frac{\pi}{16}+\cos\frac{\pi}{8}\cos\frac{5\pi}{16} & \cos\frac{3\pi}{8}\cos\frac{7\pi}{16}-\cos\frac{\pi}{8}\cos\frac{3\pi}{16} & \cos\frac{3\pi}{8}\cos\frac{3\pi}{16}+\cos\frac{\pi}{8}\cos\frac{\pi}{16} \\ \cos\frac{3\pi}{8}\cos\frac{3\pi}{16}+\cos\frac{\pi}{8}\cos\frac{\pi}{16} & -\cos\frac{3\pi}{8}\cos\frac{7\pi}{16}+\cos\frac{\pi}{8}\cos\frac{3\pi}{16} & -\cos\frac{3\pi}{8}\cos\frac{\pi}{16}+\cos\frac{\pi}{8}\cos\frac{5\pi}{16} & -\cos\frac{3\pi}{8}\cos\frac{5\pi}{16}+\cos\frac{\pi}{8}\cos\frac{7\pi}{16} \\ -\cos\frac{\pi}{8}\cos\frac{3\pi}{16}+\cos\frac{3\pi}{8}\cos\frac{\pi}{16} & \cos\frac{\pi}{8}\cos\frac{7\pi}{16}+\cos\frac{3\pi}{8}\cos\frac{3\pi}{16} & \cos\frac{\pi}{8}\cos\frac{\pi}{16}+\cos\frac{3\pi}{8}\cos\frac{5\pi}{16} & \cos\frac{\pi}{8}\cos\frac{5\pi}{16}+\cos\frac{3\pi}{8}\cos\frac{7\pi}{16} \end{pmatrix}$$

is factorized into the following matrix product

$$\boldsymbol{V}_4 = \frac{\sqrt{2}}{2}\begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} & 0 & \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} & & \\ & & \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ 0 & & \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} \end{pmatrix}\begin{pmatrix} \sin\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{\pi}{16} & \cos\frac{3\pi}{16} \\ \sin\frac{\pi}{16} & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} \\ \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \sin\frac{\pi}{16} \\ \cos\frac{3\pi}{16} & -\sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{3\pi}{16} \end{pmatrix}.$$

The key to the derivation of a fast conversion algorithm is an observation that the properly scaled factored (block) matrices on the right-hand sides of $\boldsymbol{V}_2$ and $\boldsymbol{V}_4$ matrices are actually the orthonormal DCT-IV matrices $\boldsymbol{C}_2^{IV}$ and $\boldsymbol{C}_4^{IV}$, but with lower and upper halves exchanged (see the rightmost matrices on the right-hand sides of $\boldsymbol{V}_2$ and $\boldsymbol{V}_4$) when compared with the explicit orthonormal forms of DCT-IV matrices presented in [49]. For clarity, the explicit forms of orthonormal matrices $\boldsymbol{C}_2^{IV}$ and $\boldsymbol{C}_4^{IV}$ are, respectively, given by

$$\boldsymbol{C}_2^{IV} = \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} \end{pmatrix}, \quad \boldsymbol{C}_4^{IV} = \frac{\sqrt{2}}{2} \begin{pmatrix} \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \sin\frac{\pi}{16} \\ \cos\frac{3\pi}{16} & -\sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{3\pi}{16} \\ \sin\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{\pi}{16} & \cos\frac{3\pi}{16} \\ \sin\frac{\pi}{16} & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} \end{pmatrix}.$$

Hence, the conversion matrices $\boldsymbol{V}_2$ and $\boldsymbol{V}_4$ may be represented by the following sparse block matrix factorizations:

$$\boldsymbol{V}_2 = \begin{pmatrix} \boldsymbol{C}_1^{IV} & 0 \\ 0 & \boldsymbol{C}_1^{IV} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \boldsymbol{C}_2^{IV},$$

$$\boldsymbol{V}_4 = \frac{\sqrt{2}}{2} \begin{pmatrix} \boldsymbol{C}_2^{IV} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_2^{IV} \end{pmatrix} \begin{pmatrix} \boldsymbol{0} & \boldsymbol{I}_2 \\ \boldsymbol{I}_2 & \boldsymbol{0} \end{pmatrix} \boldsymbol{C}_4^{IV},$$

where $\sqrt{2}\ \boldsymbol{C}_1^{IV} = 1$, and $\boldsymbol{I}_2$ is the identity matrix of order 2. In fact, investigating the conversion matrix $\boldsymbol{V}_8$ by the same procedure (see the orthonormal matrix $\boldsymbol{C}_8^{IV}$ in [49]) we find that it may be represented by the similar sparse block matrix factorization as

$$\boldsymbol{V}_8 = \frac{\sqrt{2}}{4} \begin{pmatrix} \boldsymbol{C}_4^{IV} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_4^{IV} \end{pmatrix} \begin{pmatrix} \boldsymbol{0} & \boldsymbol{I}_4 \\ \boldsymbol{I}_4 & \boldsymbol{0} \end{pmatrix} \boldsymbol{C}_8^{IV}.$$

Hence, for $M = 2^m$, $m > 0$, scaling $\boldsymbol{V}_M$ by $\sqrt{2}$ produces its generalized sparse block matrix factorization defined as

$$\boldsymbol{V}_M = \begin{pmatrix} \boldsymbol{C}_{\frac{M}{2}}^{IV} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_{\frac{M}{2}}^{IV} \end{pmatrix} \begin{pmatrix} \boldsymbol{0} & \boldsymbol{I}_{\frac{M}{2}} \\ \boldsymbol{I}_{\frac{M}{2}} & \boldsymbol{0} \end{pmatrix} \boldsymbol{C}_M^{IV}, \tag{6.102}$$

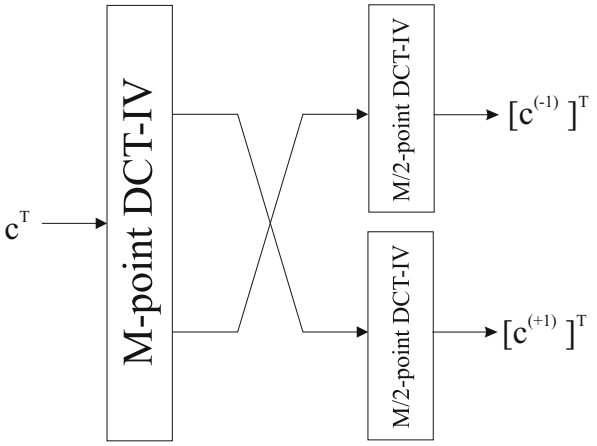

Fig. 6.8 Block diagram of the fast in-place conversion algorithm

where $\boldsymbol{C}_{\frac{M}{2}}^{IV}$ and $\boldsymbol{C}_{M}^{IV}$ are DCT-IV matrices of order $\frac{M}{2}$ and M, respectively, $\boldsymbol{I}_{\frac{M}{2}}$ is the identity matrix of order $\frac{M}{2}$, and $\mathbf{0}'s$ are null matrices. Equations (6.98) and (6.102) define the fast conversion algorithm. The corresponding block diagram of the fast in-place conversion algorithm is shown in Fig. 6.8. Based on the frequency coefficients of the long (MDCT) transform according to Fig. 6.8, they are at first transformed by the M-point DCT-IV, then two halves of the output vector are exchanged and finally, each half is transformed separately by the $\frac{M}{2}$-point DCT-IV. The final frequency coefficients of two short transforms are normalized by the factor $\frac{4}{N}$. Note that the memory requirements to store the half of $\boldsymbol{V}_M$ matrix are completely eliminated.

Since the DCT-IV matrices are self-inverse, i.e., $\boldsymbol{C}_{M}^{IV} = [\boldsymbol{C}_{M}^{IV}]^{-1} = [\boldsymbol{C}_{M}^{IV}]^{T}$ [49], by transposing (6.102) we obtain the generalized sparse block matrix factorization of the transposed conversion matrix $\boldsymbol{V}_{M}^{T}$ as

$$\boldsymbol{V}_{M}^{T} = \boldsymbol{C}_{M}^{IV} \begin{pmatrix} \mathbf{0} & \boldsymbol{I}_{\frac{M}{2}} \\ \boldsymbol{I}_{\frac{M}{2}} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{C}_{\frac{M}{2}}^{IV} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{C}_{\frac{M}{2}}^{IV} \end{pmatrix}. \tag{6.103}$$

Equations (6.100) and (6.103) define the fast conversion algorithm to convert the frequency coefficients of two short transforms to those of the long (MDCT) transform. It means that the block diagram in Fig. 6.7 is performed in the reverse direction. The final frequency coefficients of the long (MDCT) transform are normalized by the factor $\frac{4}{M}$.

In general, the arithmetic complexity of the fast conversion algorithm for $M = 2^m$ is given by that of the M-point DCT-IV plus that of two $\frac{M}{2}$-point DCTs-IV. Using the fast algorithm for M-point DCT-IV computation having the lowest achievable arithmetic complexity [53], the fast conversion algorithm requires $\frac{M}{2}(2m + 3)$ real

multiplications and $\frac{3M}{2}(2m-1)$ real additions. Compared to the matrix-vector products given by (6.98) and (6.100) which require M^2 multiplications, $M(M-1)$ additions, and the memory to store $\frac{1}{2}\ M^2$ elements of $\boldsymbol{V}_M$, the fast conversion algorithm is superior both in terms of the arithmetic complexity and memory requirements.

6.6.5 *Comparison of Discussed Conversion Methods and Consequences*

The arithmetic complexity and memory requirements of discussed conversion methods are summarized in Table 6.2. It can be seen that the fast in-place conversion algorithm is efficient in terms of the arithmetic complexity and memory requirements. It may be applied to the E-AC-3 to/from AC-3 bit stream conversion/transcoding directly in the frequency domain without partial decoding/encoding, thus saving more than 50% of total arithmetic operations and eliminating completely memory requirements compared to the standard conversion methods. The relationship between the long (MDCT) transform and two short transforms via the orthonormal conversion matrix guarantees that no errors are introduced during the E-AC-3 to/from AC-3 bit stream conversion/transcoding.

In summary, the fast algorithm for conversion of frequency coefficients of AC-3 transforms has the following advantages:

- It does not depend on the KBD windowing function.
- It simplifies the implementation of AC-3 analysis and synthesis filter banks in the encoder and decoder.
- It is efficient in terms of the structural simplicity, arithmetic complexity and memory requirements.
- Conversion procedures can be realized directly in the frequency domain without partial decoding and encoding.

Table 6.2 Summary of arithmetic complexity and memory requirements of conversion methods

Conversion method	# of real mults	# of real adds	Memory
Matrix-vector products	M^2	$M(M-1)$	$\frac{1}{2}\ M^2$
Dolby E-AC-3 to AC-3	$\frac{M}{2}(4m+19)$	$\frac{3M}{2}(4m+1)$	$2M$
Dolby AC-3 to E-AC-3 (in the worth case)	$\frac{M}{2}(4m+19)$	$\frac{3M}{2}(4m+1)$	$2M$
Dolby AC-3 to E-AC-3 (in the best case)	$\frac{M}{2}(3m+16)$	$\frac{3M}{2}\ 3m$	$2M$
Fast algorithm	$\frac{M}{2}(2m+3)$	$\frac{3M}{2}(2m-1)$	None

- Although many fast algorithms for the DCT-IV computation are available the existing AC-3 fast computational modules, i.e., the $\frac{N}{4}$-point and $\frac{M}{4}$-point forward complex FFT modules may be simply reused in the conversion procedures.
- It minimizes the amount of partial decoding/encoding and memory requirements during the conversion and transcoding processes.

6.7 Conversion of the MDCT to MDST Frequency Coefficients

The MDCT being the long transform in the AC-3 and E-AC-3 and the corresponding modified discrete sine transform (MDST) [21] are perfect reconstruction cosine/sine-modulated filter banks based on the concept of time domain aliasing cancellation. The MDCT as the real part and the MDST as the imaginary part compose a complex filter bank called the modulated complex lapped transform (MCLT) [44, 48]. Equivalently, the MDCT and MDST can be respectively viewed as the real and imaginary components of the well-known modified O^2DFT [23, 36, 37, 39]. The MCLT carries the magnitude and phase information which are useful measures in many perceptual audio coders for spectral analysis. Therefore, the MCLT has been adopted in the current AC-3 and E-AC-3 for spectral adjustment (extension) and channel coupling [10], in MP3 [45] and MPEG AAC [46, 47] audio streaming applications for audio packet loss concealment, as well as in MPEG-2/4 AAC to obtain the spatial parameter representation for stereo and multichannel audio coding [35, 36]. The construction of MCLT at the encoder is almost trivial using a fast MCLT algorithm (see Chap. 4). However, at the decoder, when the time domain samples are not available, the so-called direct transform-based method has to be applied, i.e., the available frequency representation (MDCT coefficients of three adjacent blocks) is backwards transformed to reconstruct the time domain representation and then transformed back to the frequency domain to obtain the MDST coefficients [10]. Therefore, the key question is how to compute the MDST coefficients from given MDCT coefficients directly in the frequency domain, or in other words, how to construct the complex MCLT filter bank from available MDCT coefficients.

Several methods have been developed [34–38, 40, 41, 43, 45–47] in order to obtain the MDST coefficients at the decoder directly in the frequency domain from current and neighboring blocks of MDCT coefficients. Fraunhofer researchers [40, 41] proposed a method to obtain the approximate MDST coefficients for the current block as a weighted sum of the surrounding MDCT coefficients. By using trigonometric identities, simple analytical formulae have been proposed in [45] to compute the MDST coefficients from three consecutive blocks of MDCT coefficients. However, analytical formulae are valid only for the rectangular and sine windowing functions. Using a sophisticated analytical procedure, trigonometric identities and identities for sums of trigonometric series, Dolby Labs developed

an exact and approximate method for arbitrary symmetric windowing function [37, 38]. Exact MDST coefficients for the current block are expressed as the sum of two modified convolution or filtering operations: the first filtering operation is performed on the MDCT coefficients of the previous and following blocks, and the second one is performed on the MDCT coefficients of the current block. On the other hand, based on the block matrix representation of the MDCT and MDST filter banks, a relation between the MDCT and MDST coefficients in the frequency domain has been derived in [43, 46, 47]. The MDST coefficients for the current block are obtained from three consecutive blocks of MDCT coefficients via conversion matrices. However, the authors did not investigate the properties of conversion matrices in detail. A method formulated in the matrix-vector form has been presented in [35, 36]. Due to applying different windowing functions for the MDCT and the MDST (the sine windowing function for MDCT and the cosine windowing function for MDST), the final matrix-vector products are different compared to [43, 46, 47]. Consequently, the MDST coefficients are approximately obtained only from previous and following blocks of MDCT coefficients (the current block is completely eliminated). Recently, a generalized exact and approximate conversion method of the MDCT to MDST coefficients directly in the frequency domain has been proposed for arbitrary symmetric windowing function [34]. Based on the compact block matrix representation of the MDCT and MDST filter banks, on their properties and on relations among transform sub-matrices, a relation in the matrix-vector form between the MDCT and MDST coefficients in the frequency domain is derived. Given MDCT coefficients of three consecutive data blocks at a decoder, the MDST coefficients of the current data block can be obtained by combining the MDCT coefficients of the previous, current, and following blocks via conversion matrices. Because the conversion matrices have a very regular structure, the matrix-vector products are reduced to simple analytical formulae. We note that the conversion methods do not disturb the MDCT coefficients at the decoder, i.e., the quality of recovered signal is preserved.

Although the exact conversion methods enable us to compute the exact MDST coefficients only in specified one or more frequency ranges, thus significantly reducing the computational complexity, the computation of complete set of MDST coefficients still requires a high number of arithmetic operations compared to the direct transform-based conversion method. As an alternative, efficient and flexible approximate conversion methods have been constructed [34, 37]. With properly selected parameters they can produce acceptable approximate results with much lower computational complexity. Moreover, they are very flexible to compute the approximate MDST coefficients in different frequency ranges with different accuracies including the exact computation. Therefore, the approximate conversion methods have a potential to be used in many MDCT-based audio decoders, and particularly at resource-limited and low-cost decoders for spectral analysis to obtain the magnitude and phase information.

In the following sections several exact and approximate conversion methods of the MDCT to MDST coefficients in the frequency domain are discussed including the direct transform-based method, their computational complexity, and memory

requirements. In general, almost all methods can be adopted in any MDCT-based audio codec, when the spectral information is required.

6.7.1 Analysis/Synthesis MDCT and MDST Filter Banks

We recall that the analysis and synthesis MDCT filter banks are, respectively, defined as [14]

$$c_k^{(t)} = \frac{4}{N} \sum_{n=0}^{N-1} w_n \, x_n^{(t)} \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{6.104}$$

$$\hat{x}_n^{(t)} = w_n \sum_{k=0}^{\frac{N}{2}-1} c_k^{(t)} \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$
$$n = 0, 1, \ldots, N-1, \tag{6.105}$$

where the superscript t denotes the data-block number, $\{x_n^{(t)}\}$ is the input data block, $\{c_k^{(t)}\}$ are MDCT frequency coefficients, and $\{\hat{x}_n^{(t)}\}$ is the time domain aliased data sequence. N is the data block length, assumed to be a multiple of 4. Although $\{w_n\}$ in the AC-3 and E-AC-3 represents the KBD windowing function, in this section it is assumed to be an arbitrary symmetric windowing function satisfying perfect reconstruction conditions given by (6.56), and $\{w_n\}$ is the same for analysis and synthesis filter banks. Analytical forms of commonly used symmetric windowing functions in audio coding applications satisfying (6.56) are discussed in Chap. 3.

The corresponding analysis and synthesis MDST filter banks are, respectively, defined as [30]

$$s_k^{(t)} = \frac{4}{N} \sum_{n=0}^{N-1} w_n \, x_n^{(t)} \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{6.106}$$

$$\hat{y}_n^{(t)} = w_n \sum_{k=0}^{\frac{N}{2}-1} s_k^{(t)} \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$
$$n = 0, 1, \ldots, N-1, \tag{6.107}$$

where $\{s_k^{(t)}\}$ are MDST frequency coefficients and $\{\hat{y}_n^{(t)}\}$ is the time domain aliased data sequence.

Since there exists a relation between the MDST and the MDCT [21, 30], the MDST computation can be realized via any fast MDCT computational structure. Consequently, in the current AC-3 and E-AC-3 encoder/decoder, the MDST may be implemented by the adopted reconfigurable complex DFT/FFT-based fast algorithm with simple pre- and post-processing of data sequences.

The MDCT as the real part and MDST as the imaginary part compose the complex MCLT filter bank for the tth data block defined as [44, 48]

$$p_k^{(t)} = c_k^{(t)} - i\, s_k^{(t)}, \qquad i = \sqrt{-1}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{6.108}$$

which is used to compute spectral measures, the magnitude and phase angle. Indeed, the current AC-3 encoder obtains better spectral power estimation in terms of improving the fidelity through the power energy summation of the MDCT frequency coefficients and frequency coefficients of the corresponding MDST. On the other hand, new coding tools in the E-AC-3 encoder and decoder, spectral adjustment (extension) and enhanced channel coupling process, require the phase information for angle adjustment and therefore, besides the MDCT the corresponding MDST is also generated [10].

6.7.2 Magnitude and Phase Angle of Spectral Coefficients

The magnitude of spectral coefficient $p_k^{(t)}$ in the tth data block is defined as [37]

$$|p_k^{(t)}| = \sqrt{\left[c_k^{(t)}\right]^2 + \left[s_k^{(t)}\right]^2}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{6.109}$$

whereas the phase angle $\varphi_k^{(t)}$ of spectral coefficient $p_k^{(t)}$ is defined as [37]

$$\varphi_k^{(t)} = \arctan\left[\frac{s_k^{(t)}}{c_k^{(t)}}\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{6.110}$$

6.7.3 The Direct Transform-Based Method

Given the MDCT coefficients of three consecutive data blocks (the previous, current, and following) at the decoder, i.e., $t-1$, t and $t+1$. According to (6.104) the MDCT coefficients $\{c_m^{(t-1)}\}$, $\{c_m^{(t)}\}$ and $\{c_m^{(t+1)}\}$ are, respectively, given by

$$
\begin{aligned}
c_m^{(t-1)} &= \frac{4}{N}\sum_{n=0}^{N-1} w_n\, x_n^{(t-1)} \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2m+1)\right],\\
c_m^{(t)} &= \frac{4}{N}\sum_{n=0}^{N-1} w_n\, x_n^{(t)} \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2m+1)\right],\\
c_m^{(t+1)} &= \frac{4}{N}\sum_{n=0}^{N-1} w_n\, x_n^{(t+1)} \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2m+1)\right],\\
& m = 0, 1, \ldots, \frac{N}{2}-1.
\end{aligned}
\tag{6.111}
$$

The direct transform-based method [37, 38] to compute the exact MDST coefficients for the current time domain data block t requires three computations of the synthesis MDCT filter bank given by (6.105), defined as

$$
\begin{aligned}
\hat{x}_n^{(t-1)} &= w_n \sum_{m=0}^{\frac{N}{2}-1} c_m^{(t-1)} \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2m+1)\right],\\
\hat{x}_n^{(t)} &= w_n \sum_{m=0}^{\frac{N}{2}-1} c_m^{(t)} \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2m+1)\right],\\
\hat{x}_n^{(t+1)} &= w_n \sum_{m=0}^{\frac{N}{2}-1} c_m^{(t+1)} \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2m+1)\right],\\
& n = 0, 1, \ldots, N-1,
\end{aligned}
\tag{6.112}
$$

and two overlap-add procedures to recover the current time domain data block $\{x_n^{(t)}\}$ defined as

$$
x_n^{(t)} = \begin{cases} \hat{x}_{\frac{N}{2}+n}^{(t-1)} + \hat{x}_n^{(t)}, & n = 0, 1, \ldots, \frac{N}{2}-1, \\ \hat{x}_n^{(t)} + \hat{x}_{n-\frac{N}{2}}^{(t+1)}, & n = \frac{N}{2}, \frac{N}{2}+1, \ldots, N-1, \end{cases}
\tag{6.113}
$$

followed by the analysis MDST filter bank given by (6.106) applied to $\{x_n^{(t)}\}$. For clarity of the direct transform-based conversion method see Fig. 6.7.

Since the direct transform-based method is a block operation, it yields always the complete set of exact MDST coefficients for the current data block. In practical implementations, to simplify computation only one synthesis MDCT filter bank, one overlap-add procedure and one analysis MDST filter bank are required, provided that a decoder retained the results of previous two synthesis MDCT filter banks [37]. If we adopt for the data block length $N = 2^n$ the most efficient fast MDCT algorithm

with the arithmetic complexity of $\frac{N}{4}(n+1)$ real multiplications and $\frac{N}{4}(3n-1)$ real additions (the synthesis MDCT filter bank requires exactly $\frac{N}{2}$ less real additions) [21, 30], then the practical implementation of direct transform-based method will actually require $\frac{N}{2}(n+5)$ real multiplications and $\frac{N}{2}(3n-1)$ real additions. Thus, for the computation of one unique MDST frequency coefficient the direct transform-based method will require exactly $n+5$ real multiplications and $3n-1$ real additions, i.e., totally $4n+4$ arithmetic operations.

Splitting the sum of analysis MDST filter bank given by (6.106) into two parts as

$$\begin{aligned} s_k^{(t)} &= \frac{4}{N} \sum_{n=0}^{\frac{N}{2}-1} w_n \, x_n^{(t)} \, \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right] \\ &\quad + \frac{4}{N} \sum_{n=\frac{N}{2}}^{N-1} w_n \, x_n^{(t)} \, \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \\ &\quad k = 0, 1, \ldots, \frac{N}{2}-1, \end{aligned} \tag{6.114}$$

and then substituting expressions of the overlap-add procedures from the right-hand sides of (6.113) into both sums for $x_n^{(t)}$ in (6.114) we get

$$\begin{aligned} s_k^{(t)} &= \frac{4}{N} \sum_{n=0}^{\frac{N}{2}-1} w_n \, (\hat{x}_{\frac{N}{2}+n}^{(t-1)} + \hat{x}_n^{(t)}) \, \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right] \\ &\quad + \frac{4}{N} \sum_{n=\frac{N}{2}}^{N-1} w_n \, (\hat{x}_n^{(t)} + \hat{x}_{n-\frac{N}{2}}^{(t+1)}) \, \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \\ &\quad k = 0, 1, \ldots, \frac{N}{2}-1. \end{aligned} \tag{6.115}$$

Finally, substituting proper expressions for the terms $\hat{x}_{\frac{N}{2}+n}^{(t-1)}$, $\hat{x}_n^{(t)}$ and $\hat{x}_{n-\frac{N}{2}}^{(t+1)}$ from the right-hand sides of (6.112) into (6.115), i.e., substituting $\frac{N}{2}+n$ and $n-\frac{N}{2}$ for n into appropriate sum of (6.112), we obtain

$$\begin{aligned} s_k^{(t)} &= \frac{4}{N} \sum_{n=0}^{\frac{N}{2}-1} w_n \left\{ w_{\frac{N}{2}+n} \sum_{m=0}^{\frac{N}{2}-1} (-1)^{m+1} \, c_m^{(t-1)} \, \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2m+1)\right] \right\} \\ &\quad \times \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right] \end{aligned}$$

$$
\begin{aligned}
&+\frac{4}{N}\sum_{n=0}^{\frac{N}{2}-1} w_n \left\{ w_n \sum_{m=0}^{\frac{N}{2}-1} c_m^{(t)} \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2m+1)\right]\right\} \\
&\times \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right] \\
&+\frac{4}{N}\sum_{n=\frac{N}{2}}^{N-1} w_n \left\{ w_n \sum_{m=0}^{\frac{N}{2}-1} c_m^{(t)} \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2m+1)\right]\right\} \\
&\times \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right] \\
&+\frac{4}{N}\sum_{n=\frac{N}{2}}^{N-1} w_n \left\{ w_{n-\frac{N}{2}} \sum_{m=0}^{\frac{N}{2}-1} (-1)^m\, c_m^{(t+1)} \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2m+1)\right]\right\} \\
&\times \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right], \qquad k=0,1,\ldots,\frac{N}{2}-1. \qquad (6.116)
\end{aligned}
$$

Equation (6.116) provides the basis for the derivation of two frequency domain-based conversion methods [45] and [37, 38] which are discussed in the next subsections.

6.7.4 *Conversion Method for the Rectangular and Sine Windowing Functions*

Simple analytical formulae have been proposed in [45] to compute the exact MDST coefficients of the current time domain data block t from three consecutive blocks of MDCT coefficients $\{c_m^{(t-1)}\}$, $\{c_m^{(t)}\}$ and $\{c_m^{(t+1)}\}$. However, analytical formulae are valid only for the rectangular and sine windowing functions.

Taking Eq. (6.116), adjusting the range of n in the last two sums from $\frac{N}{2}, \frac{N}{2}+1,\ldots,N-1$ to $0,1,\ldots,\frac{N}{2}-1$, using the trigonometric identities for products $\sin(\alpha)\sin(\beta)$, $\cos(\alpha)\sin(\beta)$, $\cos(\alpha)\cos(\beta)$, and finally exchanging the order of sums, after some algebraic manipulations the conversion method is defined by [45]

$$
\begin{aligned}
s_0^{(t)} &= e\left(c_0^{(t)} - c_1^{(t)}\right) + \sum_{m=0}^{\frac{N}{2}-1}(-1)^m f_{0,m}\, c_m^{(t-1)} + \sum_{m=0}^{\frac{N}{2}-1} g_{0,m}\, c_m^{(t+1)}, \quad \text{for } k=0, \\
s_k^{(t)} &= e\left(c_{k-1}^{(t)} - c_{k+1}^{(t)}\right) + \sum_{m=0}^{\frac{N}{2}-1}(-1)^m f_{k,m}\, c_m^{(t-1)} + (-1)^k \sum_{m=0}^{\frac{N}{2}-1} g_{k,m}\, c_m^{(t+1)},
\end{aligned}
$$

$$k = 1, 2, \ldots, \frac{N}{2} - 2,$$

$$s^{(t)}_{\frac{N}{2}-1} = e\left(c^{(t)}_{\frac{N}{2}-1} + c^{(t)}_{\frac{N}{2}-2}\right) + \sum_{m=0}^{\frac{N}{2}-1} (-1)^m f_{\frac{N}{2}-1,m}\, c^{(t-1)}_m - \sum_{m=0}^{\frac{N}{2}-1} g_{\frac{N}{2}-1,m}\, c^{(t+1)}_m, \text{ for } k = \frac{N}{2} - 1. \tag{6.117}$$

The real-valued constant e, and real-valued elements of $\{f_{k,m}\}$, $\{g_{k,m}\}$ are, respectively, defined as [45]

$$\begin{aligned} e &= \frac{2}{N} \sum_{n=0}^{\frac{N}{2}-1} \left(w_n^2 - w_{\frac{N}{2}+n}^2\right) \sin\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)\right], \\ f_{k,m} &= \frac{2}{N} \sum_{n=0}^{\frac{N}{2}-1} w_n\, w_{\frac{N}{2}+n} \left\{\cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)(k + m + 1)\right]\right. \\ &\quad \left. - \cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)(k - m)\right]\right\}, \\ g_{k,m} &= \frac{2}{N} \sum_{n=0}^{\frac{N}{2}-1} w_n\, w_{\frac{N}{2}+n} \left\{\cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)(k + m + 1)\right]\right. \\ &\quad \left. + \cos\left[\frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)(k - m)\right]\right\}, \quad k, m = 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{6.118}$$

For the sine windowing function the arithmetic complexity of conversion method defined by (6.117) is given by $\frac{N}{2}(N + 1)$ multiplications and the same number of additions. The precomputed values e, $\{f_{k,m}\}$ and $\{g_{k,m}\}$ in (6.118) require a memory to store $\frac{N^2}{2} + 1$ real values. Notice that for rectangular windowing function the constant $e = 0$, and hence the MDST coefficients do not depend on the current block of MDCT coefficients.

6.7.5 *Dolby Conversion Method*

The Dolby conversion method [37, 38] for arbitrary symmetric windowing function has been derived by a sophisticated analytical procedure using the trigonometric identities and identities for sums of trigonometric series. Principally, the derivation of exact conversion method is similarly based on Eq. (6.116).

In the following only analytical formulae of the Dolby exact conversion method in a compact form are presented. All intermediate steps in the derivation of conversion method and details in the form of lemmas with their proofs as well as

possible simplifications for the rectangular and sine windowing functions can be found in [37, 38].

Keeping up the original notation used in [37, 38], the exact even-indexed MDST coefficients are given by

$$s_{2k}^{(t)} = a_{2k} + b_{2k} = \sum_{p=0}^{N-1} z_p^{(1)}\, h_{2k-p}^{(1)} + \sum_{q=0}^{\frac{N}{2}-1} z_{2q+1}^{(2)}\, h_{2k-(2q+1)}^{(2)}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{6.119}$$

where

$$a_{2k+1} = a_{N-2-2k}, \qquad b_{2k+1} = b_{N-2-2k}, \quad k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{6.120}$$

The spectral components $\{z_p^{(1)}\}$ and $\{z_{2q+1}^{(2)}\}$ are, respectively, defined by

$$z_p^{(1)} = (-1)^{p+1}\, c_p^{(t-1)} + c_p^{(t+1)}, \qquad p = 0, 1, \ldots, N-1, \tag{6.121}$$

$$z_{2q+1}^{(2)} = c_{2q+1}^{(t)}, \qquad q = 0, 1, \ldots, \frac{N}{2} - 1. \tag{6.122}$$

The precomputed discrete sequences $\{h_p^{(1)}\}$ and $\{h_{2q+1}^{(2)}\}$ are, respectively, defined by

$$h_p^{(1)} = \frac{2}{N} \sum_{n=0}^{\frac{N}{2}-1} w_n\, w_{\frac{N}{2}+n}\, \cos\left[\frac{\pi}{N}\left(2n + 1 + \frac{N}{2}\right) p\right], \quad h_{-p}^{(1)} = h_p^{(1)},$$
$$p = 0, 1, \ldots, N-1, \tag{6.123}$$

$$h_{2q+1}^{(2)} = \frac{4}{N} \sum_{n=0}^{\frac{N}{2}-1} w_n^2\, \sin\left[\frac{\pi}{N}\left(2n + 1 + \frac{N}{2}\right)(2q+1)\right], \quad h_{-(2q+1)}^{(2)} = -h_{2q+1}^{(2)},$$
$$q = 0, 1, \ldots, \frac{N}{2} - 1. \tag{6.124}$$

Note that $\{h_{2q+1}^{(2)}\}$ is defined only in odd values. The odd-indexed MDST coefficients are deduced from even symmetry properties of MDST coefficients [21] as

$$s_{2k+1}^{(t)} = s_{N-2-2k}^{(t)}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{6.125}$$

The discrete sequence $\{h_p^{(1)}\}$ given by (6.123) has the odd anti-symmetry property:

$$h_{N-p}^{(1)} = -h_p^{(1)}, \qquad p = 1, 2, \ldots, \frac{N}{2} - 1, \tag{6.126}$$

while the discrete sequence $\{h_{2q+1}^{(2)}\}$ given by (6.124), has the even symmetry property:

$$h_{N-(2q+1)}^{(2)} = h_{2q+1}^{(2)}, \qquad q = 0, 1, \ldots, \frac{N}{2} - 1. \tag{6.127}$$

The Dolby exact conversion method consists of two parts. Equation (6.119) may be interpreted as a summation of two modified convolution or filtering operations of two discrete symmetric sequences $\{h_p^{(1)}\}$ and $\{h_{2q+1}^{(2)}\}$ with two sets of intermediate spectral components $\{z_p^{(1)}\}$ and $\{z_{2q+1}^{(2)}\}$ derived from previous, current and following blocks of MDCT coefficients. Since discrete sequences $\{h_p^{(1)}\}$ and $\{h_{2q+1}^{(2)}\}$ depend on the employed symmetric windowing function, their analytical expressions are further simplified by the sophisticated analytical procedures for two specific windowing functions, rectangular and sine [37, 38]. In particular, for the rectangular windowing function the discrete sequence $\{h_{2q+1}^{(2)}\}$ reduces to zero, i.e., $h_{2q+1}^{(2)} = 0$, for $q = 0, 1, \ldots, \frac{N}{2} - 1$, and hence the MDST coefficients do not depend on the current block of MDCT coefficients. For the sine windowing function, the discrete sequence $\{h_p^{(1)}\}$ is nonzero only for two values of p.

6.7.5.1 Computational Complexity and Memory Requirements

The arithmetic complexity of Dolby exact conversion method for the rectangular windowing function is $\frac{N}{4}(N+2)$ multiplications, $\frac{N}{4}(N+4)$ additions, and it requires a memory for $3\frac{N}{2}$ values. For the sine windowing function $\frac{N}{4}(N+6)$ multiplications and $\frac{N}{4}(N+8)$ additions are required, and memory for $2N + \frac{N}{4}$ values. Exploiting the symmetry properties of $\{h_p^{(1)}\}$ and $\{h_{2q+1}^{(2)}\}$, the exact conversion method for KBD windowing function requires $\frac{N}{4}(3N-2)$ multiplications, $\frac{N}{4}$ $3N$ additions and the memory for $2N + \frac{N}{4}$ values.

6.7.6 Generalized Conversion Method Based on the Compact Block Matrix Representation

A generalized conversion method of the MDCT to MDST coefficients directly in the frequency domain has been proposed in [34] for arbitrary symmetric windowing function. Based on the compact block matrix representation of the MDCT and MDST filter banks, on their properties and on relations among transform sub-matrices, a relation in the matrix-vector form between the MDCT and MDST

coefficients in the frequency domain is derived. Given MDCT coefficients of three consecutive data blocks at a decoder, the MDST coefficients of the current data block can be obtained by combining the MDCT coefficients of the previous, current, and following blocks via conversion matrices. Since the forms of conversion matrices depend on the employed windowing function, a specific solution for each windowing function is derived. Because the conversion matrices have a very regular structure, the matrix-vector products are reduced to simple analytical formulae. The generalized conversion method [34] presented in the next subsections is more efficient and structurally simpler both in terms of arithmetic complexity and memory requirements compared to existing exact frequency domain-based conversion methods.

It is important to note that in the next subsections the data-block number t is also used as the subscript in vector notations.

6.7.6.1 Matrix Representations of MDCT and MDST Filter Banks

Consider the MDCT and MDST filter banks given by (6.104)/(6.105) and (6.106)/(6.107), respectively. Matrix representations of the analysis/synthesis MDCT filter banks or equivalently, of the forward/backward long AC-3 transform with incorporated windowing function are presented in Sect. 6.4.2, whereas the symmetry properties of row basis vector of matrix $\boldsymbol{C}_{\frac{N}{2}\times N}$ are discussed in Sect. 6.4.4.

Let the sine transform kernel in the analysis MDST filter bank (6.106) be represented by a matrix $\boldsymbol{S}_{\frac{N}{2}\times N}$ with elements

$$\{\boldsymbol{S}_{\frac{N}{2}\times N}\}_{k,n} = \sin\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2}-1,\ n = 0, 1, \ldots, N-1. \tag{6.128}$$

Note that the kth row basis vector of the matrix $\boldsymbol{S}_{\frac{N}{2}\times N}$ exhibits even symmetry in its first half and even anti-symmetry property in its second half [30]. The sine transform kernel in the synthesis MDST filter bank (6.107) is represented by the matrix $\boldsymbol{S}^T_{\frac{N}{2}\times N}$, where T denotes transposition. The analysis/synthesis MDST filter banks given by (6.106)/(6.107) can be respectively represented in the equivalent matrix-vector form as [30]

$$\boldsymbol{s}_t^T = \frac{4}{N}\,\boldsymbol{S}_{\frac{N}{2}\times N}\,\boldsymbol{W}_N\,\boldsymbol{x}_t^T, \qquad \hat{\boldsymbol{y}}_t^T = \boldsymbol{W}_N\,\boldsymbol{S}^T_{\frac{N}{2}\times N}\,\boldsymbol{s}_t^T, \tag{6.129}$$

where the matrix $\boldsymbol{W}_N$ is given by (6.55), and $\boldsymbol{x}_t^T$, $\boldsymbol{s}_t^T$, $\hat{\boldsymbol{y}}_t^T$ are appropriate column vectors. Based on the matrix representation of the MDST it was shown in [50] that the transposed MDST matrix $\left[\boldsymbol{S}_{\frac{N}{2}\times N}\right]^T$ denoted by $\boldsymbol{S}^+_{N\times\frac{N}{2}}$ is the pseudoinverse of its

corresponding forward transform matrix. Hence, the forward and backward MDST block transforms are actually the pseudoinverse pair (see Appendix A.1). In fact, for products of the matrix $\boldsymbol{S}_{\frac{N}{2}\times N}$ and its pseudoinverse, $\boldsymbol{S}^{+}_{N\times\frac{N}{2}}$, the following relations hold:

$$\boldsymbol{S}_{\frac{N}{2}\times N}\,\boldsymbol{S}^{+}_{N\times\frac{N}{2}} = \frac{N}{2}\,\boldsymbol{I}_{\frac{N}{2}}, \tag{6.130}$$

and

$$\boldsymbol{S}^{+}_{N\times\frac{N}{2}}\,\boldsymbol{S}_{\frac{N}{2}\times N} = \frac{N}{4}\begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_{\frac{N}{4}} & -\boldsymbol{J}_{\frac{N}{4}} \\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{J}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} = \frac{N}{4}\begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} + \boldsymbol{J}_{\frac{N}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{\frac{N}{2}} - \boldsymbol{J}_{\frac{N}{2}} \end{pmatrix}, \tag{6.131}$$

where $\boldsymbol{I}_{\frac{N}{2}}$ is the identity matrix, $\boldsymbol{J}_{\frac{N}{2}}$ is the reverse ordered identity matrix both of order $\frac{N}{2}$, and $\boldsymbol{0}'s$ are null matrices.

We recall that there exists a close relation between MDCT and MDST matrices [30]. Actually, the matrix $\boldsymbol{S}_{\frac{N}{2}\times N}$ is related to $\boldsymbol{C}_{\frac{N}{2}\times N}$ by

$$\boldsymbol{S}_{\frac{N}{2}\times N} = (-1)^{\frac{N}{4}}\,\boldsymbol{J}_{\frac{N}{2}}\,\boldsymbol{C}_{\frac{N}{2}\times N}\,\boldsymbol{D}_N, \tag{6.132}$$

where $\boldsymbol{D}_N$ is a diagonal odd sign-changing matrix of order N defined as $\boldsymbol{D}_N = \mathrm{diag}\{1,-1,1,\ldots,-1\}$. For clarity, the matrix $\boldsymbol{S}_{\frac{N}{2}\times N}$ in explicit form for $N = 8$ is given by

$$\boldsymbol{S}_{4\times 8} = \begin{pmatrix} \cos\frac{3\pi}{16} & \cos\frac{\pi}{16} & \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{7\pi}{16} & -\cos\frac{7\pi}{16} & -\cos\frac{5\pi}{16} \\ \cos\frac{7\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & \cos\frac{7\pi}{16} & \cos\frac{\pi}{16} & \cos\frac{5\pi}{16} & -\cos\frac{5\pi}{16} & -\cos\frac{\pi}{16} \\ -\cos\frac{\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{5\pi}{16} & -\cos\frac{\pi}{16} & \cos\frac{7\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{7\pi}{16} \\ \cos\frac{5\pi}{16} & -\cos\frac{7\pi}{16} & -\cos\frac{7\pi}{16} & \cos\frac{5\pi}{16} & -\cos\frac{3\pi}{16} & \cos\frac{\pi}{16} & -\cos\frac{\pi}{16} & \cos\frac{3\pi}{16} \end{pmatrix}.$$

6.7.6.2 Relations Among MDCT and MDST Sub-Matrices

Now, let the matrices $\boldsymbol{C}_{\frac{N}{2}\times N}$ given by (6.58) and $\boldsymbol{S}_{\frac{N}{2}\times N}$ given by (6.128) be split into the following two blocks:

$$\boldsymbol{C}_{\frac{N}{2}\times N} = \begin{pmatrix} \boldsymbol{K}_{\frac{N}{2}} & \boldsymbol{L}_{\frac{N}{2}} \end{pmatrix}, \qquad \boldsymbol{S}_{\frac{N}{2}\times N} = \begin{pmatrix} \boldsymbol{P}_{\frac{N}{2}} & \boldsymbol{Q}_{\frac{N}{2}} \end{pmatrix}, \tag{6.133}$$

where $\boldsymbol{K}_{\frac{N}{2}}$, $\boldsymbol{L}_{\frac{N}{2}}$, $\boldsymbol{P}_{\frac{N}{2}}$ and $\boldsymbol{Q}_{\frac{N}{2}}$ are square nonsingular matrices of order $\frac{N}{2}$. Splitting the matrices $\boldsymbol{C}_{\frac{N}{2}\times N}$ and $\boldsymbol{S}_{\frac{N}{2}\times N}$ into two blocks plays a key role in the derivation of a relation between MDCT and MDST coefficients directly in the frequency domain. Based on the relation (6.132), the matrix pairs $\boldsymbol{K}_{\frac{N}{2}}$, $\boldsymbol{P}_{\frac{N}{2}}$ and $\boldsymbol{L}_{\frac{N}{2}}$, $\boldsymbol{Q}_{\frac{N}{2}}$ are closely related as

$$\begin{aligned}\boldsymbol{P}_{\frac{N}{2}} &= (-1)^{\frac{N}{4}}\boldsymbol{J}_{\frac{N}{2}}\boldsymbol{K}_{\frac{N}{2}}\boldsymbol{D}_{\frac{N}{2}},\\ \boldsymbol{Q}_{\frac{N}{2}} &= (-1)^{\frac{N}{4}}\boldsymbol{J}_{\frac{N}{2}}\boldsymbol{L}_{\frac{N}{2}}\boldsymbol{D}_{\frac{N}{2}},\end{aligned} \tag{6.134}$$

where the elements of $\boldsymbol{K}_{\frac{N}{2}}$ and $\boldsymbol{L}_{\frac{N}{2}}$ are given by (6.75). According to (6.134) the elements of $\boldsymbol{P}_{\frac{N}{2}}$ and $\boldsymbol{Q}_{\frac{N}{2}}$ are, respectively, given by

$$\begin{aligned}\{\boldsymbol{P}_{\frac{N}{2}}\}_{\frac{N}{2}-1-k,n} &= (-1)^{n+\frac{N}{4}}\ \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],\\ \{\boldsymbol{Q}_{\frac{N}{2}}\}_{\frac{N}{2}-1-k,n} &= (-1)^{n+\frac{N}{4}}\ \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{3N}{2}\right)(2k+1)\right],\\ k,\ n &= 0,1,\ldots,\frac{N}{2}-1.\end{aligned} \tag{6.135}$$

Then, the pseudoinverse matrices $\boldsymbol{C}^{+}_{N\times\frac{N}{2}}$ and $\boldsymbol{S}^{+}_{N\times\frac{N}{2}}$ (or transposed versions of $\boldsymbol{C}_{\frac{N}{2}\times N}$ and $\boldsymbol{S}_{\frac{N}{2}\times N}$) are, respectively, given by

$$\boldsymbol{C}^{+}_{N\times\frac{N}{2}} = \begin{pmatrix}\boldsymbol{K}^{+}_{\frac{N}{2}}\\ \boldsymbol{L}^{+}_{\frac{N}{2}}\end{pmatrix}, \qquad \boldsymbol{S}^{+}_{N\times\frac{N}{2}} = \begin{pmatrix}\boldsymbol{P}^{+}_{\frac{N}{2}}\\ \boldsymbol{Q}^{+}_{\frac{N}{2}}\end{pmatrix}, \tag{6.136}$$

and the row vectors of $\boldsymbol{K}_{\frac{N}{2}}/\boldsymbol{L}_{\frac{N}{2}}$ and $\boldsymbol{P}_{\frac{N}{2}}/\boldsymbol{Q}_{\frac{N}{2}}$ become the column vectors of $\boldsymbol{K}^{+}_{\frac{N}{2}}/\boldsymbol{L}^{+}_{\frac{N}{2}}$ and $\boldsymbol{P}^{+}_{\frac{N}{2}}/\boldsymbol{Q}^{+}_{\frac{N}{2}}$.

6.7.6.3 Relation Between MDCT and MDST Coefficients in the Frequency Domain

Consider the windowing and overlap-add procedure at the decoder to reconstruct the current time domain data block t denoted by $\boldsymbol{x}_t = \left[\boldsymbol{x}_t^{(1)}\ \ \boldsymbol{x}_t^{(2)}\right]$, where $\boldsymbol{x}_t^{(1)}$ is the first half and $\boldsymbol{x}_t^{(2)}$ is the second half of $\boldsymbol{x}_t$. Since the MDCT filter bank satisfies the perfect reconstruction constraints in the overlapped parts of three consecutive blocks $t-1$, t and $t+1$, the tth current time domain data block $\boldsymbol{x}_t = \left[\boldsymbol{x}_t^{(1)}\ \ \boldsymbol{x}_t^{(2)}\right]$ can be perfectly reconstructed by the windowing and overlap-add procedures formulated

in the matrix-vector form as follows:

$$\begin{aligned}\left[\boldsymbol{x}_t^{(1)}\right]^T &= \boldsymbol{W}_{\frac{N}{2}}^{(2)}\boldsymbol{L}_{\frac{N}{2}}^{+}\,\boldsymbol{c}_{t-1}^{T} + \boldsymbol{W}_{\frac{N}{2}}^{(1)}\boldsymbol{K}_{\frac{N}{2}}^{+}\,\boldsymbol{c}_{t}^{T},\\ \left[\boldsymbol{x}_t^{(2)}\right]^T &= \boldsymbol{W}_{\frac{N}{2}}^{(2)}\boldsymbol{L}_{\frac{N}{2}}^{+}\,\boldsymbol{c}_{t}^{T} + \boldsymbol{W}_{\frac{N}{2}}^{(1)}\boldsymbol{K}_{\frac{N}{2}}^{+}\,\boldsymbol{c}_{t+1}^{T}.\end{aligned} \tag{6.137}$$

On the other hand, using (6.55), (6.129) and (6.133) the MDST coefficients in tth data block can be expressed in the matrix-vector form as

$$\boldsymbol{s}_t^T = \frac{4}{N}\left(\boldsymbol{P}_{\frac{N}{2}}\,\boldsymbol{W}_{\frac{N}{2}}^{(1)}\left[\boldsymbol{x}_t^{(1)}\right]^T + \boldsymbol{Q}_{\frac{N}{2}}\,\boldsymbol{W}_{\frac{N}{2}}^{(2)}\left[\boldsymbol{x}_t^{(2)}\right]^T\right). \tag{6.138}$$

Equations (6.137) and (6.138) represent the direct transform-based conversion method in the matrix-vector representation to compute the exact MDST coefficients of the current time domain data block t if the MDCT coefficients of three consecutive blocks $t-1$, t and $t+1$ are given.

In order to obtain a relation between MDCT and MDST coefficients directly in the frequency domain, substituting expressions from (6.137) for $[\boldsymbol{x}_t^{(1)}]^T$ and $[\boldsymbol{x}_t^{(2)}]^T$ into the right-hand side of (6.138) we have

$$\boldsymbol{s}_t^T = \frac{4}{N}\left(\boldsymbol{U}_{\frac{N}{2}}\,\boldsymbol{c}_{t-1}^{T} + \left(\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}\right)\,\boldsymbol{c}_{t}^{T} + \boldsymbol{V}_{\frac{N}{2}}\,\boldsymbol{c}_{t+1}^{T}\right), \tag{6.139}$$

where

$$\begin{aligned}\boldsymbol{U}_{\frac{N}{2}} &= \boldsymbol{P}_{\frac{N}{2}}\boldsymbol{W}_{\frac{N}{2}}^{(1)}\boldsymbol{W}_{\frac{N}{2}}^{(2)}\boldsymbol{L}_{\frac{N}{2}}^{+}, \quad \boldsymbol{V}_{\frac{N}{2}} = \boldsymbol{Q}_{\frac{N}{2}}\boldsymbol{W}_{\frac{N}{2}}^{(2)}\boldsymbol{W}_{\frac{N}{2}}^{(1)}\boldsymbol{K}_{\frac{N}{2}}^{+},\\ \boldsymbol{G}_{\frac{N}{2}} &= \boldsymbol{P}_{\frac{N}{2}}\boldsymbol{W}_{\frac{N}{2}}^{(1)}\boldsymbol{W}_{\frac{N}{2}}^{(1)}\boldsymbol{K}_{\frac{N}{2}}^{+}, \quad \boldsymbol{H}_{\frac{N}{2}} = \boldsymbol{Q}_{\frac{N}{2}}\boldsymbol{W}_{\frac{N}{2}}^{(2)}\boldsymbol{W}_{\frac{N}{2}}^{(2)}\boldsymbol{L}_{\frac{N}{2}}^{+},\end{aligned} \tag{6.140}$$

are nonsingular square matrices of order $\frac{N}{2}$. Equations (6.139) and (6.140) define the relation among the MDST coefficients of the current time domain data block t and the MDCT coefficients of three consecutive blocks $t-1$, t and $t+1$. Square matrices $\boldsymbol{U}_{\frac{N}{2}}$, $\boldsymbol{V}_{\frac{N}{2}}$ and $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ are called conversion matrices.

The same relation between the MDCT and MDST coefficients in the frequency domain defined by (6.139) and (6.140) has been obtained in [43, 46, 47]. However, the authors did not further investigate the properties of conversion matrices in detail. On the other hand, a relation between the MDCT and MDST coefficients in the matrix-vector form presented in [35, 36] is based on applying different windowing functions for the MDCT and the MDST, specifically, the sine windowing function for MDCT and the cosine windowing function for MDST. Consequently, the final matrix-vector products are different compared to (6.139) and (6.140), and MDST coefficients are approximately obtained only from previous and following blocks of MDCT coefficients (the current block is completely eliminated).

As the first step in the derivation of the generalized conversion method for arbitrary symmetric windowing function, we shall investigate the elements of matrices $\boldsymbol{U}_{\frac{N}{2}}$, $\boldsymbol{V}_{\frac{N}{2}}$ and $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ in detail. Actually, the conversion matrices have very regular structure and possess interesting symmetry properties which can be used to reduce the matrix-vector products in (6.139) to simple general analytical formulae with minimal memory requirements.

6.7.6.4 Conversion Matrices $\boldsymbol{U}_{\frac{N}{2}}$, $\boldsymbol{V}_{\frac{N}{2}}$ and Their Properties

Using (6.57), (6.75), (6.135), and (6.136), and performing the scalar products of row/column vectors, the corresponding elements of matrices $\boldsymbol{U}_{\frac{N}{2}}$ and $\boldsymbol{V}_{\frac{N}{2}}$ in (6.139) and (6.140) are, respectively, given by

$$u_{\frac{N}{2}-1-k,m} = \frac{4}{N} \sum_{n=0}^{\frac{N}{4}-1} 2\, w_n\, w_{\frac{N}{2}-1-n}\, (-1)^{n+\frac{N}{4}}\, \cos\phi_{n,k}\, \cos\psi_{n,m},$$

$$v_{\frac{N}{2}-1-k,m} = \frac{4}{N} \sum_{n=0}^{\frac{N}{4}-1} 2\, w_{\frac{N}{2}-1-n}\, w_n\, (-1)^{n+\frac{N}{4}}\, \cos\psi_{n,k}\, \cos\phi_{n,m},$$

$$k,\ m = 0, 1, \ldots, \frac{N}{2} - 1, \tag{6.141}$$

where $\phi_{n,p} = \frac{\pi}{2N}\left(2n + 1 + \frac{N}{2}\right)(2p+1)$ and $\psi_{n,p} = \frac{\pi}{2N}\left(2n + 1 + \frac{3N}{2}\right)(2p+1)$ for $p = k$ and m. Since

$$\cos\psi_{n,p} = (-1)^{p+1}\ \sin\phi_{n,p}, \tag{6.142}$$

then, substituting $\frac{N}{2} - 1 - p$ for p into the sine term in (6.142) we get the following trigonometric identity:

$$\cos\psi_{n,\frac{N}{2}-1-p} = (-1)^p\ (-1)^{n+\frac{N}{4}}\ \cos\phi_{n,p}. \tag{6.143}$$

Using (6.142) and (6.143) for $p = k$ and m we can rewrite (6.141) into the final form as

$$u_{\frac{N}{2}-1-k,\frac{N}{2}-1-m} = (-1)^m\, \frac{4}{N} \sum_{n=0}^{\frac{N}{4}-1} 2\, w_n\, w_{\frac{N}{2}-1-n}\, \cos\phi_{n,k}\, \cos\phi_{n,m},$$

$$v_{k,m} = (-1)^k \frac{4}{N} \sum_{n=0}^{\frac{N}{4}-1} 2\, w_n\, w_{\frac{N}{2}-1-n}\, \cos\phi_{n,k}\, \cos\phi_{n,m},$$
$$k,\ m = 0, 1, \ldots, \frac{N}{2} - 1. \tag{6.144}$$

From (6.144), it can be easily seen that the elements of $\boldsymbol{U}_{\frac{N}{2}}$ and $\boldsymbol{V}_{\frac{N}{2}}$ are the same in magnitude except for their proper sign changes and reverse ordering. Indeed, the elements $u_{k,m}$ are related to $v_{k,m}$ and vice versa by

$$u_{k,m} = (-1)^{k+m}\, v_{\frac{N}{2}-1-k,\frac{N}{2}-1-m},$$
$$v_{k,m} = (-1)^{k+m}\, u_{\frac{N}{2}-1-k,\frac{N}{2}-1-m},$$
$$k,\ m = 0, 1, \ldots, \frac{N}{2} - 1. \tag{6.145}$$

Further, between the elements above and under the main diagonal of $\boldsymbol{U}_{\frac{N}{2}}$ and $\boldsymbol{V}_{\frac{N}{2}}$ we can observe the following symmetries:

$$u_{k,m} = (-1)^{k+m}\, u_{m,k},$$
$$v_{k,m} = (-1)^{k+m}\, v_{m,k}, \quad k \neq m, \quad k < m,$$
$$k,\ m = 0, 1, \ldots, \frac{N}{2} - 1. \tag{6.146}$$

If we would realize the matrix-vector products $\boldsymbol{U}_{\frac{N}{2}}\boldsymbol{c}_{t-1}^T$ and $\boldsymbol{V}_{\frac{N}{2}}\boldsymbol{c}_{t+1}^T$ in (6.139), then from (6.145) it follows that it is sufficient to use only one of the matrices $\boldsymbol{U}_{\frac{N}{2}}$ and $\boldsymbol{V}_{\frac{N}{2}}$. Let the conversion matrix $\boldsymbol{V}_{\frac{N}{2}}$ be precomputed according to (6.144). Equation (6.146) implies that only the upper triangular part of $\boldsymbol{V}_{\frac{N}{2}}$ including elements on the main diagonal must be stored, i.e., we need totally $\frac{N}{8}(N+2)$ elements. However, the memory requirements can be further minimized as follows.

Using the trigonometric identity $2\cos\phi_{n,k}\cos\phi_{n,m} = \cos(\phi_{n,k} + \phi_{n,m}) + \cos(\phi_{n,k} - \phi_{n,m})$, the elements $v_{k,m}$ in (6.144) can be written as

$$v_{k,m} = (-1)^k \frac{4}{N} \sum_{n=0}^{\frac{N}{4}-1} w_n\, w_{\frac{N}{2}-1-n}\, \cos\left[\frac{\pi}{N}\left(2n + 1 + \frac{N}{2}\right)(k + m + 1)\right] +$$
$$(-1)^k \frac{4}{N} \sum_{n=0}^{\frac{N}{4}-1} w_n\, w_{\frac{N}{2}-1-n}\, \cos\left[\frac{\pi}{N}\left(2n + 1 + \frac{N}{2}\right)(k - m)\right],$$
$$k,\ m = 0, 1, \ldots, \frac{N}{2} - 1. \tag{6.147}$$

Let us define the following discrete sequence $\{f_s\}$ by

$$f_s = \frac{4}{N} \sum_{n=0}^{\frac{N}{4}-1} w_n \, w_{\frac{N}{2}-1-n} \, \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right) s\right],$$

$$f_{-s} = f_s, \quad s = 0, 1, \ldots N-1, \tag{6.148}$$

whose values depend on the block length and a specific windowing function. It can be easily verified that the discrete sequence $\{f_s\}$ possesses the odd anti-symmetry property given by

$$f_{N-s} = -f_s, \qquad s = 1, 2, \ldots, \frac{N}{2} - 1, \quad \text{and } f_{\frac{N}{2}} = 0. \tag{6.149}$$

Then, Eq. (6.147) can be written in the simplified form as

$$v_{k,m} = (-1)^k \, (f_{k+m+1} + f_{k-m}), \qquad k, \; m = 0, 1, \ldots, \frac{N}{2} - 1. \tag{6.150}$$

Finally, substituting $\frac{N}{2} - 1 - k$ for k and $\frac{N}{2} - 1 - m$ for m in (6.150) and using (6.145) after some algebraic manipulations, the elements $u_{k,m}$ are given by

$$u_{k,m} = (-1)^m \, (f_{k+m+1} - f_{k-m}), \qquad k, \; m = 0, 1, \ldots, \frac{N}{2} - 1. \tag{6.151}$$

The symmetry properties given by (6.145) and (6.146) can be easily derived from (6.147) using its representation via $\{f_s\}$. Equations (6.150) and (6.151) indicate the following important fact. For given k, m, the elements $v_{k,m}$ and $u_{k,m}$ can be generated by combining values f_{k+m+1} and f_{k-m} of the discrete sequence $\{f_s\}$. Thus, if we use the odd anti-symmetry property of $\{f_s\}$ given by (6.149), then we need to store only $\frac{N}{2}$ elements instead of $\frac{N}{8}(N+2)$.

It is interesting to compare Eqs. (6.151) and (6.150) with elements $\{f_{k,m}\}$ and $\{g_{k,m}\}$, respectively, defined in (6.118). Except for sign changes, they are equivalent. However, the authors in their conversion method [45] observed neither the symmetry properties of conversion matrices nor the discrete sequence $\{f_s\}$ and its symmetry property which reduce the memory requirements significantly.

6.7.6.5 Conversion Matrix $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ and Its Properties

Similarly, using (6.57), (6.75), (6.135), and (6.136), and performing the scalar products of row/column vectors, the elements of matrix $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ in (6.139) and (6.140) denoted by $(g+h)_{k,m}$ are given by

$$(g+h)_{\frac{N}{2}-1-k,m} = \frac{4}{N}\sum_{n=0}^{\frac{N}{2}-1}(-1)^{n+\frac{N}{4}}\left(w_n^2\cos\phi_{n,k}\cos\phi_{n,m} + w_{\frac{N}{2}-1-n}^2\cos\psi_{n,k}\cos\psi_{n,m}\right),$$
$$k,\ m = 0,1,\ldots,\frac{N}{2}-1. \tag{6.152}$$

Compared to the conversion matrix $\boldsymbol{V}_{\frac{N}{2}}$, the derivation of a general close form of $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ is somewhat more complicated. Firstly, using (6.142) for $p = k$ and m, subsequently substituting $\frac{N}{2}-1-n$ for n in (6.152) and using trigonometric identities

$$\cos\phi_{\frac{N}{2}-1-n,p} = -\cos\phi_{n,p}, \qquad \sin\phi_{\frac{N}{2}-1-n,p} = \sin\phi_{n,p}, \tag{6.153}$$

for $p = k$ and m, Eq. (6.152) can be written as

$$(g+h)_{\frac{N}{2}-1-k,m} = \frac{4}{N}\sum_{n=0}^{\frac{N}{4}-1}(-1)^{n+\frac{N}{4}}\left(w_n^2 - w_{\frac{N}{2}-1-n}^2\right)$$
$$\left(\cos\phi_{n,k}\cos\phi_{n,m} - (-1)^{k+m}\sin\phi_{n,k}\sin\phi_{n,m}\right),$$
$$k,\ m = 0,1,\ldots,\frac{N}{2}-1. \tag{6.154}$$

Then, by substituting $\frac{N}{2}-1-k$ for k into (6.154), and using trigonometric identities

$$\cos\phi_{n,\frac{N}{2}-1-k} = (-1)^{n+\frac{N}{4}}\sin\phi_{n,k}, \qquad \sin\phi_{n,\frac{N}{2}-1-k} = (-1)^{n+\frac{N}{4}}\cos\phi_{n,k}, \tag{6.155}$$

Eq. (6.154) takes the following form

$$(g+h)_{k,m} = \frac{4}{N}\sum_{n=0}^{\frac{N}{4}-1}\left(w_n^2 - w_{\frac{N}{2}-1-n}^2\right)\left(\sin\phi_{n,k}\cos\phi_{n,m} + (-1)^{k+m}\cos\phi_{n,k}\sin\phi_{n,m}\right),$$
$$k,\ m = 0,1,\ldots,\frac{N}{2}-1. \tag{6.156}$$

Applying trigonometric identities to the terms $\sin\phi_{n,k}\cos\phi_{n,m}$ and $\cos\phi_{n,k}\sin\phi_{n,m}$ in (6.156), and by using

$$\sin\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)(k+m+1)\right]$$
$$= (-1)^{\frac{k+m}{2}}\cos\left[\frac{\pi(2n+1)}{N}(k+m+1)\right],\ \text{when } (k+m) \text{ is even},$$

$$\sin\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)(k-m)\right]$$
$$=(-1)^{\frac{k-m-1}{2}}\cos\left[\frac{\pi(2n+1)}{N}(k-m)\right],\ \text{when } (k+m) \text{ is odd,} \tag{6.157}$$

we finally obtain a general close form of $\boldsymbol{G}_{\frac{N}{2}}+\boldsymbol{H}_{\frac{N}{2}}$ defined as

$$(g+h)_{k,m}=\frac{4}{N}\sum_{n=0}^{\frac{N}{4}-1}\begin{cases}\left(w_n^2-w_{\frac{N}{2}-1-n}^2\right)\ (-1)^{\frac{k+m}{2}}\ \cos\left[\frac{\pi(2n+1)}{N}(k+m+1)\right], \\ (k+m)\ \text{is even}, \\ \left(w_n^2-w_{\frac{N}{2}-1-n}^2\right)\ (-1)^{\frac{k-m-1}{2}}\ \cos\left[\frac{\pi(2n+1)}{N}(k-m)\right], \\ (k+m)\ \text{is odd}, \end{cases} \tag{6.158}$$

for $k,\ m=0,1,\ldots,\frac{N}{2}-1$. Note that $w_n^2-w_{\frac{N}{2}-1-n}^2=\left(w_n+w_{\frac{N}{2}-1-n}\right)\left(w_n-w_{\frac{N}{2}-1-n}\right)<0$ for $n=0,1,\ldots,\frac{N}{4}-1$.

The conversion matrix $\boldsymbol{G}_{\frac{N}{2}}+\boldsymbol{H}_{\frac{N}{2}}$ has interesting symmetry properties. Specifically, between the elements in the upper/left and lower/right half of $\boldsymbol{G}_{\frac{N}{2}}+\boldsymbol{H}_{\frac{N}{2}}$ the following symmetries can be observed:

$$(g+h)_{k,m}=(-1)^{k+m}(g+h)_{\frac{N}{2}-1-k,\frac{N}{2}-1-m},\qquad k,\ m=0,1,\ldots,\frac{N}{2}-1, \tag{6.159}$$

while between the elements under and above the main/opposite main diagonal of $\boldsymbol{G}_{\frac{N}{2}}+\boldsymbol{H}_{\frac{N}{2}}$ the following symmetries can be observed:

$$(g+h)_{k,m}=(-1)^{k+m}(g+h)_{m,k},\quad k\neq m,\quad k<m,\quad k,\ m=0,1,\ldots,\frac{N}{2}-1. \tag{6.160}$$

After we realize the matrix-vector product $\left(\boldsymbol{G}_{\frac{N}{2}}+\boldsymbol{H}_{\frac{N}{2}}\right)\boldsymbol{c}_t^T$ in (6.139), it is sufficient to store either only the upper half or only the upper triangular part of $\boldsymbol{G}_{\frac{N}{2}}+\boldsymbol{H}_{\frac{N}{2}}$ including elements on the main diagonal, i.e., totally either $\frac{N^2}{8}$ or $\frac{N}{8}(N+2)$ elements, respectively. However, we shall see that the memory requirements may be reduced to $\frac{N}{4}$ as follows.

Let us derive the elements of $\boldsymbol{G}_{\frac{N}{2}}+\boldsymbol{H}_{\frac{N}{2}}$ in the explicit form according to Eq. (6.158). There are two cases:

1. If $k+m$ is even, then it can be written as $k+m=2q$, $q=0,1,\ldots,\frac{N}{4}-1$, and the elements $(g+h)_{k,m}$ are given by

$$(g+h)_{k,m} = \frac{4}{N}\sum_{n=0}^{\frac{N}{4}-1}\left(w_n^2 - w_{\frac{N}{2}-1-n}^2\right)\ (-1)^q\ \cos\left[\frac{\pi}{N}(2n+1)(2q+1)\right],$$
$$q = 0, 1, \ldots, \frac{N}{4}-1. \tag{6.161}$$

Additionally, substituting $\frac{N}{2}-1-k$ for k and $\frac{N}{2}-1-m$ for m in (6.158) when $k+m$ is even, we have

$$(g+h)_{k,m} = (g+h)_{\frac{N}{2}-1-m,\frac{N}{2}-1-k}, \qquad k+m \ \text{ is even.} \tag{6.162}$$

2. If $k+m$ is odd, then $k-m$ is also odd and it can be written as $k-m = \pm(2q+1)$, $q = 0, 1, \ldots, \frac{N}{4}-1$, and the elements $(g+h)_{k,m}$ are given by

$$(g+h)_{k,m} = \frac{4}{N}\sum_{n=0}^{\frac{N}{4}-1}\begin{cases}\left(w_n^2 - w_{\frac{N}{2}-1-n}^2\right)\ (-1)^q\ \cos\left[\frac{\pi}{N}(2n+1)(2q+1)\right], \\ \text{if } k > m, \\ \left(w_n^2 - w_{\frac{N}{2}-1-n}^2\right)\ (-1)^{q-1}\ \cos\left[\frac{\pi}{N}(2n+1)(2q+1)\right], \\ \text{if } k < m,\end{cases} \tag{6.163}$$
$$q = 0, 1, \ldots, \frac{N}{4}-1.$$

Equations (6.161), (6.162), and (6.163) directly imply that the matrix $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ contains only $\frac{N}{4}$ unique elements in magnitude. Denoting them by β_q, based on (6.161) they are defined as

$$\beta_q = (-1)^q\,\frac{4}{N}\sum_{n=0}^{\frac{N}{4}-1}\left(w_n^2 - w_{\frac{N}{2}-1-n}^2\right)\ \cos\left[\frac{\pi}{4(N/4)}(2n+1)(2q+1)\right],$$
$$q = 0, 1, \ldots, \frac{N}{4}-1. \tag{6.164}$$

Values of $\{\beta_q\}$ again depend on the block length and on a specific windowing function. Equation (6.164) corresponds to an $\frac{N}{4}$-point DCT-IV of $\{w_n^2 - w_{\frac{N}{2}-1-n}^2\}$.

In general, the forms of conversion matrices $\boldsymbol{V}_{\frac{N}{2}}$ and $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ depend on the employed symmetric windowing function. The unique elements of $\boldsymbol{V}_{\frac{N}{2}}$ are given by (6.150), and the unique elements of $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ are given by (6.164). Since the conversion matrices have the very regular structure, the matrix-vector products in (6.139) can be reduced to simple analytical formulae.

6.7.6.6 General Analytical Formulae of the Exact Conversion Method

With respect to (6.139) let us divide the matrix-vector products into two parts as

$$s_k^{(t)} = a_k + b_k, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{6.165}$$

where the terms a_k and b_k are, respectively, defined as

$$a_k = \sum_{m=0}^{\frac{N}{2}-1} u_{k,m}\, c_m^{(t-1)} + v_{k,m}\, c_m^{(t+1)}, \tag{6.166}$$

$$b_k = \sum_{m=0}^{\frac{N}{2}-1} (g+h)_{k,m}\, c_m^{(t)}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{6.167}$$

6.7.6.7 General Analytical Formula for the Computation of a_k

According to (6.150) and (6.151), and replacing the index m by p, the computation of a_k given by (6.166) can be written as

$$\begin{aligned} a_k &= \sum_{p=0}^{\frac{N}{2}-1} (-1)^p\, (f_{k+p+1} - f_{k-p})\, c_p^{(t-1)} + (-1)^k\, (f_{k+p+1} + f_{k-p})\, c_p^{(t+1)} \\ &= \sum_{p=0}^{\frac{N}{2}-1} \left[(-1)^p\, c_p^{(t-1)} + (-1)^k\, c_p^{(t+1)} \right] f_{k+p+1} - \left[(-1)^p\, c_p^{(t-1)} - (-1)^k\, c_p^{(t+1)} \right] f_{k-p}, \\ & \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{6.168}$$

Finally, considering indices k in (6.168) separately for even and odd values, i.e., for $2k$ and $2k+1$, respectively, we obtain the analytical formulae for the computation of a_k as

$$\begin{aligned} a_{2k} &= \sum_{p=0}^{\frac{N}{2}-1} \left[(-1)^p\, c_p^{(t-1)} + c_p^{(t+1)} \right] f_{2k+p+1} - \left[(-1)^p\, c_p^{(t-1)} - c_p^{(t+1)} \right] f_{2k-p}, \\ a_{2k+1} &= \sum_{p=0}^{\frac{N}{2}-1} \left[(-1)^p\, c_p^{(t-1)} - c_p^{(t+1)} \right] f_{2k+p+2} - \left[(-1)^p\, c_p^{(t-1)} + c_p^{(t+1)} \right] f_{2k+1-p}, \\ & \qquad k = 0, 1, \ldots, \frac{N}{4} - 1, \end{aligned} \tag{6.169}$$

where data sequences $\{(-1)^p\ c_p^{(t-1)} + c_p^{(t+1)}\}$ and $\{(-1)^p\ c_p^{(t-1)} - c_p^{(t+1)}\}$ for $p = 0, 1, \ldots, \frac{N}{2} - 1$ are precomputed requiring N additions. Since $f_{-s} = f_s$ note that $f_{2k-p} = f_{|2k-p|}$ when $2k - p < 0$, and $f_{2k+1-p} = f_{|2k+1-p|}$ when $2k + 1 - p < 0$. Subscript indices $2k + p + 1$, $2k + p + 2$, $|2k - p|$ and $|2k + 1 - p|$ satisfy the following conditions: $2k+p+1 \in \{1, 2, \ldots, N-2\}$, $2k+p+2 \in \{2, 3, \ldots, N-1\}$, $|2k - p|$ and $|2k + 1 - p| \in \{0, 1, \ldots, \frac{N}{2} - 1\}$. In general, for arbitrary symmetric windowing function, exploiting the fact that $f_{2k+p+1} = 0$ when $2k + p + 1 = \frac{N}{2}$, or $f_{2k+p+2} = 0$, when $2k+p+2 = \frac{N}{2}$, the analytical formula (6.169) requires $\frac{N}{2}(N-1)$ multiplications, $\frac{N}{2}$ N additions and needs the memory to store $3\frac{N}{2}$ values. However, for the rectangular and sine windowing functions, the arithmetic complexity of the computation of a_{2k} and a_{2k+1} given by (6.169) can be further reduced by identifying additional zeroth values of the discrete sequence $\{f_s\}$ as follows.

Substituting the analytical form of rectangular windowing function into (6.148), the discrete sequence $\{f_s\}$ is given by

$$f_s = \frac{2}{N} \sum_{n=0}^{\frac{N}{4}-1} \cos\left[\frac{\pi}{N}\left(2n + 1 + \frac{N}{2}\right)\ s\right], \quad s = 0, 1, \ldots N-1. \tag{6.170}$$

Based on the identity for the sum of trigonometric series defined as

$$\sum_{n=0}^{\frac{N}{4}-1} \cos\left[\frac{\pi}{N}\left(2n + 1 + \frac{N}{2}\right)\ 2r\right] = 0, \qquad |r| = 1, 2, \ldots, \frac{N}{2} - 1, \tag{6.171}$$

it can be seen that by substituting $s = 2r$ in (6.170) and using the identity (6.171), the terms $f_{2r} = 0$ for $r = 1, 2, \ldots, \frac{N}{2} - 1$ including the case $f_{\frac{N}{2}} = 0$. For the rectangular windowing function the discrete sequence $\{f_s\}$ has $\frac{N}{4}$ additional zero values. Therefore, in the computation of a_k for indices k, p satisfying $2k + p + 1 = 2k + p + 2 = |2k - p| = |2k + 1 - p| = 2r$, $r = 1, 2, \ldots, \frac{N}{2} - 1$ (excluding the case $f_{\frac{N}{2}} = 0$), $\frac{N}{2}\left(\frac{N}{2} - 2\right)$ multiplications and the same number of additions may be saved.

Similarly, substituting the analytical form of sine windowing function into (6.148) and using trigonometric identity after some algebraic manipulations, the discrete sequence $\{f_s\}$ is given by

$$f_s = \frac{2}{N} \sum_{n=0}^{\frac{N}{4}-1} \sin\left[\frac{\pi(2n+1)}{N}\right] \cos\left[\frac{\pi}{N}\left(2n + 1 + \frac{N}{2}\right)\ s\right], \quad s = 0, 1, \ldots N-1. \tag{6.172}$$

Based on the identity for the sum of trigonometric series defined as

$$\sum_{n=0}^{\frac{N}{4}-1} \sin\left[\frac{\pi(2n+1)}{N}\right] \cos\left[\frac{\pi}{N}\left(2n+1+\frac{N}{2}\right)(2r+1)\right] = 0,$$

$$r = 1, 2, \ldots, \frac{N}{2} - 2, \tag{6.173}$$

by substituting $s = 2r + 1$ into (6.172) and using the identity (6.173), the terms $f_{2r+1} = 0$ for $r = 1, 2, \ldots, \frac{N}{2} - 2$. For the sine windowing function the discrete sequence $\{f_s\}$ has also $\frac{N}{4}$ additional zero values. Hence, in the computation of a_k for indices k, p satisfying $2k + p + 1 = 2k + p + 2 = |2k - p| = |2k + 1 - p| = 2r + 1$, $r = 1, 2, \ldots, \frac{N}{2} - 2$, similarly, $\frac{N}{2}(\frac{N}{2} - 2)$ multiplications and the same number of additions may be saved.

6.7.6.8 Analytical Formulae for the Computation of b_k

The computation of b_k given by (6.167) can also be further optimized for the specific windowing function in terms of the number of multiplications and memory requirements. Indeed, it follows immediately for the rectangular windowing function from the general analytical close form (6.158) that the matrix $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ is null matrix, and therefore $b_k = 0$ for $\forall\ k$. It means that b_k does not depend at all on the MDCT coefficients of the current data block.

By exploiting the special structure of matrix $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$, specific analytical formulae for (and in general, for any symmetric windowing function), and the sine windowing function are derived separately in the following subsections.

A. Analytical Formulae for the KBD Windowing Function

The analytical form of KBD windowing functions is relatively complicated, and consequently, the expression $w_n^2 - w_{\frac{N}{2}-1-n}^2$ in (6.158) is also relatively complicated. Therefore, numerical values only are used. Exploiting the special structure of matrix $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ containing only $\frac{N}{4}$ unique elements in magnitude, b_k given by (6.167) for KBD (and in general, for any symmetric windowing function) can be realized as

$$b_k = \sum_{q=0}^{\frac{N}{8}-1} \begin{cases} \beta_q \left(c^{(t)}_{2q-k} - c^{(t)}_{2q+1+k} \right) + \beta_{\frac{N}{4}-1-q} \left(c^{(t)}_{\frac{N}{2}-2-2q-k} - c^{(t)}_{\frac{N}{2}-1-2q+k} \right), \\ \qquad \text{if } 2q-k \geq 0, \text{ and } 2q+1+k < \frac{N}{2}-1, \\ \beta_q \left(c^{(t)}_{k-2q-1} - c^{(t)}_{2q+1+k} \right) + \beta_{\frac{N}{4}-1-q} \left(c^{(t)}_{\frac{N}{2}-2-2q-k} + c^{(t)}_{\frac{N}{2}+2q-k} \right), \\ \qquad \text{if } 2q-k < 0, \text{ and } 2q+1+k \leq \frac{N}{2}-1, \\ \beta_q \left(c^{(t)}_{k-2q-1} + c^{(t)}_{N-2-2q-k} \right) + \beta_{\frac{N}{4}-1-q} \left(c^{(t)}_{k-\frac{N}{2}+1+2q} + c^{(t)}_{\frac{N}{2}+2q-k} \right), \\ \qquad \text{otherwise}, \end{cases} \tag{6.174}$$

for $k = 0, 1, \ldots, \frac{N}{2} - 1$, where multipliers β_q are given in (6.164). If $\frac{N}{4}$ is even, the analytical formula (6.174) requires $\frac{N}{2}\ \frac{N}{4}$ multiplications and $\frac{N}{2}\left(\frac{N}{2}-1\right)$ additions, and needs the memory to store $\frac{N}{4}$ values.

If $\frac{N}{4}$ is odd, then for fixed value of the index $q = \frac{1}{2}\left(\frac{N}{4}-1\right)$ we need additionally to compute

$$\begin{aligned} b'_k &= \beta_q \left(c^{(t)}_{2q-k} - c^{(t)}_{2q+1+k} \right), \\ b'_{\frac{N}{2}-1-k} &= \beta_q \left(c^{(t)}_{2q-k} + c^{(t)}_{2q+1+k} \right), \qquad k = 0, 1, \ldots, \frac{N}{4}-1, \end{aligned} \tag{6.175}$$

and add the results to appropriate terms b_k. This requires $\frac{N}{2}$ multiplications and N additions.

B. Analytical Formula for the Sine Windowing Function

Analytical form of the sine windowing function is simple and the expression $w_n^2 - w_{\frac{N}{2}-1-n}^2$ is equal to $-\cos\frac{\pi(2n+1)}{N}$ in (6.158). Analysis of the general close form of $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ is based on Eqs. (6.161), (6.162), (6.163) and (6.164), and an identity for the sum of trigonometric series which is defined by

$$\sum_{n=0}^{\frac{N}{4}-1} \cos\left[\frac{\pi}{N}(2n+1)\ 2r\right] = 0, \qquad |r| = 1, 2, \ldots, \frac{N}{2}-1. \tag{6.176}$$

Let us derive the elements $\{\beta_q\}$ given in (6.164) in the explicit form for the sine windowing function. Then, we have

$$\beta_q = (-1)^{q+1}\,\frac{4}{N}\sum_{n=0}^{\frac{N}{4}-1}\cos\left[\frac{\pi}{N}(2n+1)\right]\,\cos\left[\frac{\pi}{N}(2n+1)(2q+1)\right],$$
$$q = 0, 1, \ldots, \frac{N}{4}-1. \tag{6.177}$$

Consider two cases in (6.177), specifically, $q = 0$ and $q > 0$:

1. If $q = 0$, then using the trigonometric identity $\cos^2\alpha = \frac{1}{2}(1+\cos 2\alpha)$ and the identity (6.176) we get

$$\beta_0 = -\frac{4}{N}\sum_{n=0}^{\frac{N}{4}-1}\cos^2\left[\frac{\pi}{N}(2n+1)\right] = -\frac{4}{N}\sum_{n=0}^{\frac{N}{4}-1}\left(\frac{1}{2}+\frac{1}{2}\cos\left[\frac{\pi}{N}(2n+1)\,2\right]\right)$$
$$= -\frac{4}{N}\left(\sum_{n=0}^{\frac{N}{4}-1}\frac{1}{2}+\frac{1}{2}\sum_{n=0}^{\frac{N}{4}-1}\cos\left[\frac{\pi}{N}(2n+1)\,2\right]\right) = -\frac{4}{N}\,\frac{N}{8} = -\frac{1}{2}. \tag{6.178}$$

2. If $q > 0$, then using the trigonometric identity for cosine products and the identity (6.176), we get

$$\beta_q = \frac{4}{N}\,(-1)^{q+1}\sum_{n=0}^{\frac{N}{4}-1}\frac{1}{2}\left(\cos\left[\frac{\pi}{N}(2n+1)\,2q\right]\right.$$
$$\left.+\cos\left[\frac{\pi}{N}(2n+1)\,2(q+1)\right]\right) = 0,$$
$$q = 1, 2, \ldots, \frac{N}{4}-1. \tag{6.179}$$

Equations (6.178) and (6.179) imply that only $\beta_0 = -\frac{1}{2}$ is nonzero. This defines the nonzero elements of $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ for which $k + m = 0$ in equations (6.161) and (6.162), and the elements for which $k + m$ is odd and $k - m = \pm 1$ in Eq. (6.163), i.e.,

$$(g+h)_{0,0} = (g+h)_{\frac{N}{2}-1,\frac{N}{2}-1} = -\frac{1}{2},$$
$$(g+h)_{k+1,k} = -\frac{1}{2}, \qquad (g+h)_{k,k+1} = \frac{1}{2},$$
$$k = 0, 1, \ldots, \frac{N}{2}-2. \tag{6.180}$$

As a result, for the sine windowing function the matrix $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ corresponds to a tridiagonal matrix defined as

$$\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}} = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & 0 & & & \dots & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 & & \dots & 0 \\ 0 & -\frac{1}{2} & 0 & \frac{1}{2} & 0 & \dots & 0 \\ \vdots & & \vdots & & & & \vdots \\ 0 & \dots & 0 & -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \dots & & 0 & -\frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \dots & & & 0 & -\frac{1}{2} & -\frac{1}{2} \end{pmatrix}, \tag{6.181}$$

and b_k given by (6.167) can be realized as

$$b_k = \begin{cases} -\frac{1}{2}\left(c_0^{(t)} - c_1^{(t)}\right) & \text{when} \quad k = 0, \\ -\frac{1}{2}\left(c_{k-1}^{(t)} - c_{k+1}^{(t)}\right) & \text{when} \quad k = 1, 2, \dots, \frac{N}{2} - 2, \\ -\frac{1}{2}\left(c_{\frac{N}{2}-2}^{(t)} + c_{\frac{N}{2}-1}^{(t)}\right) & \text{when} \quad k = \frac{N}{2} - 1. \end{cases} \tag{6.182}$$

The analytical formula (6.182) requires $\frac{N}{2}$ multiplications (they can be implemented as shift operations) and $\frac{N}{2}$ additions. Unlike the rectangular windowing function, b_k for the sine windowing function is dependent on the MDCT coefficients of the current data block. However, since the tridiagonal matrix $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ defined by (6.181) is sparse, the dependence is relatively small.

A family of sine squared windowing functions for harmonic analysis has been analyzed in [42]. They can be written in a general analytical form as

$$w_n = \alpha - (1 - \alpha)\ \cos\frac{\pi(2n+1)}{N}, \qquad n = 0, 1, \dots, N-1. \tag{6.183}$$

Equation (6.183) for $\alpha = \frac{1}{2}$ defines the Hann windowing function (see also [2], p. 107), and for $\alpha \approx 0.54$ it defines the Hamming windowing function. Both windowing functions are special cases of the Blackman windowing function [42]. We easily find that the expression $w_n^2 - w_{\frac{N}{2}-1-n}^2 = -4\alpha(1-\alpha)\ \cos\frac{\pi(2n+1)}{N}$. In particular, when $\alpha = \frac{1}{2}$ the expression $w_n^2 - w_{\frac{N}{2}-1-n}^2$ is the same as for sine windowing function, and hence, based on (6.178) for the Hann windowing function we obtain the same solution $\beta_0 = -\frac{1}{2}$. For the Hamming windowing function, $\beta_0 = 2\alpha(1-\alpha)$, where $\alpha \approx 0.54$, and the tridiagonal structure of $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ is also preserved for it.

6.7.6.9 Computational Complexity and Memory Requirements

The generalized conversion method for arbitrary symmetric windowing function defined by (6.165), (6.169), (6.174), (6.175), and (6.182) is valid for any value of N divisible by 4. Let the discrete sequence $\{f_s\}$ given by (6.148) be precomputed for $s = 0, 1, \ldots, \frac{N}{2} - 1$. Similarly, let the data sequences $\{(-1)^p \; c_p^{(t-1)} + c_p^{(t+1)}\}$ and $\{(-1)^p \; c_p^{(t-1)} - c_p^{(t+1)}\}$ be precomputed for $p = 0, 1, \ldots, \frac{N}{2} - 1$. Then, the exact MDST coefficients are computed for the specific symmetric windowing function as follows:

- For the rectangular windowing function we use only formula (6.169) requiring totally $\frac{N}{4}(N + 2)$ multiplications, $\frac{N}{4}(N + 4)$ additions and the memory for $3\frac{N}{2}$ values.
- For the sine windowing function we use formulae (6.169) and (6.182) requiring totally $\frac{N}{4}(N + 2)$ multiplications, $\frac{N}{2}$ shifts, $\frac{N}{4}(N + 8)$ additions and the memory for $3\frac{N}{2}$ values.
- Finally, for the KBD windowing function we use formulae (6.169) and (6.174), where multipliers β_p are given by (6.164). When $\frac{N}{4}$ is even, formulae (6.169) and (6.174) require totally $\frac{N}{4}\left(\frac{5N}{2} - 2\right)$ multiplications, $\frac{N}{4}\; 3N$ additions and the memory for $3\frac{N}{2} + \frac{N}{4}$ values. If $\frac{N}{4}$ is odd, then we use additionally formula (6.175) requiring $\frac{N}{2}$ multiplications and N additions.

6.7.6.10 Comparison of Exact Conversion Methods

The generalized exact conversion method for sine windowing function and the existing method for audio packet loss concealment in MP3 decoder [45] (see Sect. 6.7.4) are actually equivalent in terms of analytical expressions, although they have been derived by different procedures. Indeed, for the sine windowing function the constant e in (6.118) coincides with the element $(g + h)_{0,0} = -\frac{1}{2}$ or with the multiplier $\beta_0 = -\frac{1}{2}$ in (6.178). Similarly, two-dimensional arrays $\{f_{k,m}\}$ and $\{g_{k,m}\}$ in (6.117) with proper sign changes coincide with conversion matrices $\boldsymbol{U}_{\frac{N}{2}}$ and $\boldsymbol{V}_{\frac{N}{2}}$, respectively. However, because the authors in [45] observed neither the symmetry properties of $\{f_{k,m}\}$ and $\{g_{k,m}\}$ nor the discrete sequence $\{f_s\}$ given by (6.148) and its symmetry property, the efficiency of generalized exact conversion method for the sine windowing function is superior both in terms of the arithmetic complexity and memory requirements compared to [45] (see Table 6.3).

In principle, the generalized conversion method is similar to the Dolby conversion method [37, 38], although both methods have been derived by quite different procedures. The exact MDST coefficients in the current data block are expressed as linear combinations of previous, current, and following blocks of MDCT coefficients. While the Dolby conversion method has been derived by a sophisticated analytical procedure using the trigonometric identities and identities for sums of trigonometric series, the generalized conversion method is based on

Table 6.3 Comparison of the arithmetic complexity and memory requirements of the frequency domain-based conversion methods for the exact computation of the complete set of MDST coefficients for commonly used windowing functions

Method	Windowing function	# of mults	# of adds	Memory
[45]	Only sine	$\frac{N}{2}(N+1)$	$\frac{N}{2}(N+1)$	$\frac{N^2}{2}$
[37, 38]	Sine	$\frac{N}{4}(N+6)$	$\frac{N}{4}(N+8)$	$2N+\frac{N}{4}$
[37, 38]	KBD	$\frac{N}{4}(3N-2)$	$\frac{N}{4}\,3N$	$2N+\frac{N}{4}$
[34]	Sine	$\frac{N}{4}(N+2)$	$\frac{N}{4}(N+8)$	$3\frac{N}{2}$
[34]	KBD	$\frac{N}{4}(\frac{5N}{2}-2)$	$\frac{N}{4}\,3N$	$3\frac{N}{2}+\frac{N}{4}$

the compact block matrix representation of the MDCT and MDST filter banks and detailed investigation of the symmetry properties of conversion matrices in corresponding matrix-vector products.

In general, both the Dolby and generalized conversion methods consist of two parts. Although the resulting analytical formulae for the computation of $\{a_k\}/\{a_{2k}\}$ and $\{b_k\}/\{b_{2k}\}$ seem to be different (compare Eqs. (6.169), (6.174) with (6.119) and (6.120)), they give exactly the same numerical results. In fact, Eq. (6.169) corresponds to the first sum in (6.119), while Eq. (6.174), (6.175) for the KBD windowing function and Eq. (6.182) for the sine windowing function correspond to the second sum in (6.119). Further, the discrete sequence $\{h_p^{(1)}\}$ given by (6.123) coincides with $\{f_p\}$ given by (6.148), and the discrete sequence $\{\beta_q\}$ given by (6.164) coincides with the first half of elements of the discrete sequence $\{h_{2q+1}^{(2)}\}$ given by (6.124). Note that $f_{\frac{N}{2}} = 0$ and hence, $h_{\frac{N}{2}}^{(1)} = 0$. In particular, comparing the second sum in (6.119) with Eq. (6.182) for the sine windowing function reveals that only two nonzero values of $\{h_{2q+1}^{(2)}\}$ are equal to $\pm\frac{1}{2}$ ($\beta_0 = -\frac{1}{2}$). It means that the multiplicative complexity of Dolby conversion method can be further reduced. The arithmetic complexity and memory requirements for rectangular windowing function are the same in both methods.

It can be seen from Table 6.3 that the generalized conversion method for the exact MDST computation is better both in terms of arithmetic complexity and memory requirements. The achieved reduced multiplicative complexity in generalized conversion method for the KBD and sine windowing functions compared to the Dolby conversion method is based on the fact that the formulae (6.174) and (6.182) for the computation of $\{b_k\}$ directly incorporate the symmetry property of $\{h_{2q+1}^{(2)}\}$ by combining pairs of MDCT coefficients multiplied by the same value of β_q or equivalently by $h_{2q+1}^{(2)}$. Further, in generalized conversion method for the sine windowing function the formula (6.182) for the computation of $\{b_k\}$ is simpler and more elegant compared to the Dolby conversion method. Comparison of the arithmetic complexity and memory requirements of the frequency domain-based conversion methods for the exact computation of the complete set of MDST coefficients for commonly used windowing functions, when $\frac{N}{4}$ is even, is

summarized in Table 6.3. One can see that the generalized conversion method is more efficient both in terms of arithmetic complexity and memory requirements, compared to exact frequency domain-based conversion methods.

6.7.7 *Efficient and Flexible Approximate Generalized Conversion Methods*

The exact MDST coefficients of the current time domain data block in the Dolby conversion method are expressed in terms of linear combinations of exact MDCT coefficients of previous, current, and following blocks. In [37, 38] the general form of exact conversion method for arbitrary windowing function defined by (6.119)–(6.125) has been used to construct an approximate conversion method.

The construction is based on two important properties of discrete sequences $\{h_p^{(1)}\}$ defined by (6.123) and $\{h_{2q+1}^{(2)}\}$ defined by (6.124). Since both $\{h_p^{(1)}\}$ and $\{h_{2q+1}^{(2)}\}$ are symmetric and decay relatively quickly, each discrete sequence may be truncated to significantly shorter length. Exploiting these properties, the general form of exact conversion method defined by (6.119)–(6.125) has been rewritten to form the approximate conversion method which uses simply the truncated versions of $\{h_p^{(1)}\}$ and $\{h_{2q+1}^{(2)}\}$ in convolution operations defined by (6.119). Consequently, an approximation of MDST coefficients is obtained with much lower complexity. In general, the longer the truncated versions of discrete sequences, the higher the computational complexity and the more accurate the approximated MDST coefficients.

In order to specify the computational complexity and to enable a performance analysis of the approximate conversion method [37, 38], an optional parameter has been defined: the total number of nonzero coefficients or taps (denoted by $ntaps_{\text{total}}$) to be used in both of the truncated versions $\{h_p^{(1)}\}$ and $\{h_{2q+1}^{(2)}\}$. Specifically, values of $ntaps_{\text{total}} = 5,\ 13,\ 19$ and 25 have been selected to investigate the approximate conversion method. The selection and distribution of these taps between discrete sequences $\{h_p^{(1)}\}$ and $\{h_{2q+1}^{(2)}\}$ depend on their decaying rates. The inspection reveals that for increasing indices the discrete sequence $\{h_{2q+1}^{(2)}\}$ decays faster than $\{h_p^{(1)}\}$. It means that $\{h_p^{(1)}\}$ is more dominant in the approximate computation and therefore, more taps have to be allocated to $\{h_p^{(1)}\}$ than to $\{h_{2q+1}^{(2)}\}$. The number of taps allocated to $\{h_p^{(1)}\}$ and $\{h_{2q+1}^{(2)}\}$ is denoted by $ntaps_{I,III}$ and $ntaps_{II}$, respectively, whereby $ntaps_{\text{total}} = ntaps_{I,III} + ntaps_{II}$ determines the computational complexity of Dolby approximate conversion method [37, 38].

In order to investigate and compare the Dolby approximate conversion method with an approximate version of generalized conversion method in a unified framework, let us define two new optional parameters $p_{\max}$ and $q_{\max}$ and relate them to $ntaps_{I,III}$, $ntaps_{II}$, and $ntaps_{\text{total}}$. Taking into account the symmetry properties of $\{h_p^{(1)}\}$ and $\{h_{2q+1}^{(2)}\}$, let $p_{\max}$ denote the maximal value of the subscript index p for which values of $\{h_p^{(1)}\}$, $p = 0, 1, \ldots, p_{\max}$, are involved in the approximate

computation of a_{2k}^{*} in (6.119), whereby $p_{\max} \in \{0, 1, \ldots, \frac{N}{2} - 1\}$. Similarly, let $q_{\max}$ denote the maximal value of the subscript index q for which values of $\{h_{2q+1}^{(2)}\}$, $q = 0, 1, \ldots, q_{\max} - 1$, are involved in the approximate computation of b_{2k}^{*} in (6.119), whereby $q_{\max} \in \{1, 2, \ldots, \frac{N}{4}\}$. Then, $p_{\max}$ and $q_{\max}$ are related to $ntaps_{I,III}$, $ntaps_{II}$ and $ntaps_{\text{total}}$ by

$$\begin{aligned} ntaps_{I,III} &= 2p_{\max} + 1, \quad ntaps_{II} = 2q_{\max}, \\ ntaps_{\text{total}} &= 2p_{\max} + 1 + 2q_{\max}, \end{aligned} \tag{6.184}$$

and the distribution of taps $ntaps_{I,III}$ and $ntaps_{II}$ between $\{h_p^{(1)}\}$ and $\{h_{2q+1}^{(2)}\}$, respectively, is related to

$$p_{\max} = 2q_{\max} + 1, \tag{6.185}$$

i.e., approximately two times more taps are allocated to $ntaps_{I,III}$ compared to $ntaps_{II}$. Choosing $p_{\max} = \frac{N}{2} - 1$ and $q_{\max} = \frac{N}{4} - 1$ corresponds to the Dolby exact conversion method.

Since the discrete sequence $\{f_p\}$ coincides with $\{h_p^{(1)}\}$ and the discrete sequence $\{\beta_q\}$ coincides with the first half of $\{h_{2q+1}^{(2)}\}$, following the approach of [37, 38] we can construct the approximate method [34] from the generalized exact conversion method defined by (6.169) and (6.174). Let $p_{\max}$ and $q_{\max}$ be given. At first, in the approximate computation of a_k^{*} according to (6.169) those values of f_{2k+p+1}, f_{2k-p}, f_{2k+p+2} and f_{2k+1-p} are involved, whose subscript indices satisfy: $2k + p + 1,\ 2k + p + 2,\ |2k - p|,\ |2k + 1 - p| \le p_{\max}$. On the other hand, since the discrete sequence $\{\beta_q\}$ decays very fast, for larger block sizes the multipliers $\beta_{\frac{N}{4}-1-q}$ in (6.174) may be neglected, and (6.174) for the approximate computation of b_k^{*} can be rewritten in the simplified form as follows:

$$b_k^{*} = \sum_{q=0}^{q_{\max}-1} \begin{cases} \beta_q \left(c_{2q-k}^{(t)} - c_{2q+1+k}^{(t)} \right), & \text{if } 2q - k \ge 0, \text{ and } 2q + 1 + k < \frac{N}{2} - 1, \\ \beta_q \left(c_{k-2q-1}^{(t)} - c_{2q+1+k}^{(t)} \right), & \text{if } 2q - k < 0, \text{ and } 2q + 1 + k \le \frac{N}{2} - 1, \\ \beta_q \left(c_{k-2q-1}^{(t)} + c_{N-2-2q-k}^{(t)} \right), & \text{otherwise}, \end{cases}$$

$$k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{6.186}$$

With respect to (6.174), $q_{\max} \in \{1, 2, \ldots, \frac{N}{8}\}$, i.e., $q_{\max}$ has half range compared to that of the Dolby approximate method. Further, it is important to note that although

the Dolby and generalized approximate method defined by quite different formulae, for given $p_{\max}$ and $q_{\max}$ both approximate methods generate the same algebraic expressions.

6.7.7.1 Computational Analysis

In order to quantify the computational complexity of the generalized conversion method including Dolby approximate method for completeness, expressions in terms of optional parameters $p_{\max}$ and $q_{\max}$ are derived specifying the average total number of arithmetic operations required per unique MDST coefficient for several different 2^n length blocks and the different total number of taps $ntaps_{\text{total}} = 7,\ 13,\ 19,\ 25$ specified in [37]. Then, the computational complexity of approximate conversion methods for the complete set of MDST coefficients is compared to that of the direct transform-based method.

A. Arithmetic Complexity of the Dolby Approximate Conversion Method

The average total number of arithmetic operations required per unique MDST coefficient by the Dolby approximate method in terms of optional parameters $p_{\max}$, $q_{\max}$ is given by

1. Computation of $\{z_p^{(1)}\}$ given by (6.121):		2 adds on average
2. Computation of a_{2k}^* in (6.119):	$(2p_{\max} + 1)$ mults	$2p_{\max}$ adds
3. Computation of b_{2k}^* in (6.119):	$2q_{\max}$ mults	$(2q_{\max} - 1)$ adds
4. Computation of $a_{2k}^* + b_{2k}^*$:		1 add
	$2p_{\max} + 1 + 2q_{\max}$ mults	$2p_{\max} + 2q_{\max} + 2$ adds
The total number of operations:	$4p_{\max} + 4q_{\max} + 3$	

Thus derived arithmetic complexity per unique MDST coefficient in terms of $p_{\max}$ and $q_{\max}$ coincides with that derived in [37]. Table 6.4 summarizes the arithmetic complexity of the Dolby approximate method in terms of the average total number of operations required per unique MDST coefficient for $ntaps_{\text{total}} = 7,\ 13,\ 19,\ 25$, whereas Table 6.5 summarizes the arithmetic complexity in terms of the total number of operations required for the complete set of MDST coefficients for different 2^n length blocks and $ntaps_{\text{total}} = 7,\ 13,\ 19,\ 25$.

Table 6.4 The arithmetic complexity in terms of the average number of total operations required per unique MDST coefficient by the Dolby approximate method for $ntaps_{\text{total}} = 7, 13, 19, 25$, and generalized approximate method for $ntaps_{\text{total}} = 6, 11, 16, 21$

Total # of taps ($ntaps_{\text{total}}$)	$ntaps_{I,III}$ of $\{h_p^{(1)}\}/\{f_p\}$	$ntaps_{II}$ of $\{h_{2q+1}^{(2)}\}/\{\beta_q\}$	Value of $p_{\max}$	Value of $q_{\max}$	# of mults	# of adds	Total # of operations
7/6	5	2/1	2	1	7/6	8	15/14
13/11	9	4/2	4	2	13/11	14	27/25
19/16	13	6/3	6	3	19/16	20	39/36
25/21	17	8/4	8	4	25/21	26	51/47

Table 6.5 The arithmetic complexity in terms of the total number of operations required for the complete set of MDST coefficients for different 2^n length blocks by the practical direct transform-based method, Dolby approximate method for $ntaps_{\text{total}} = 7, 13, 19, 25$, and generalized approximate method for $ntaps_{\text{total}} = 6, 11, 16, 21$

2^n-length blocks	Direct method	Dolby method				Generalized method			
		7/15	13/27	19/39	25/51	6/14	11/25	16/36	21/47
128	2048	960	1728	2496	3264	896	1600	2304	3008
256	4608	1920	3456	4992	6528	1792	3200	4608	6016
512	10240	3840	6912	9984	13056	3584	6400	9216	12032
1024	22258	7680	13824	19986	26112	7168	13280	18432	24064
2048	49152	15360	27648	39936	52224	14336	25600	36864	48128

From Table 6.4, it can be observed that the Dolby approximate method for obtaining the complete set of MDST coefficients requires fewer arithmetic operations than the direct transform-based method for most combinations of block sizes and specified $ntaps_{\text{total}}$. Furthermore, the advantage of the Dolby approximate method increases over the direct transform-based method as the block size increases [37]. In a specific case of the sine windowing function, since the values of discrete sequence $\{h_p^{(1)}\}$ coincide with those of $\{f_p\}$ (see Eqs. (6.172) and (6.173)), the values $h_{2r+1}^{(1)} = 0$ for $r = 1, 2, \ldots, \frac{N}{2} - 2$. $ntaps_{II}$ is always equal to 2 for any value of $ntaps_{\text{total}}$, and therefore, $ntaps_{\text{total}} = ntaps_{I,III} + 2$. Consequently, for a fixed $ntaps_{\text{total}}$ more taps are allocated to $ntaps_{I,III}$, and proper identifying of the values $h_{2r+1}^{(1)} = 0$ results in further reduction of the total number of arithmetic operations.

B. Arithmetic Complexity of General Approximate Conversion Method

The average total number of arithmetic operations required per unique MDST coefficient by the generalized approximate method in terms of optional parameters $p_{\max}$ and $q_{\max}$ is given by

1. Computation of $\{(-1)^p\, c_p^{(t-1)} + c_p^{(t+1)}\}$:		1 add on average
2. Computation of $\{(-1)^p\, c_p^{(t-1)} - c_p^{(t+1)}\}$:		1 add on average
3. Computation of a_k^* in (6.168):	$(2p_{\max} + 1)$ mults	$2p_{\max}$ adds
4. Computation of b_k^* given by (6.186):	$q_{\max}$ mults	$(2q_{\max} - 1)$ adds
5. Computation of $a_k^* + b_k^*$:		1 add
	$2p_{\max} + 1 + q_{\max}$ mults	$2p_{\max} + 2q_{\max} + 2$ adds
The total number of operations:	$4p_{\max} + 3q_{\max} + 3$	

Table 6.4 summarizes the arithmetic complexity of the generalized approximate method in terms of the average total number of operations required per unique MDST coefficient for $ntaps_{\text{total}} = 6,\ 11,\ 16,\ 21$. Comparing the expressions for arithmetic complexity per unique MDST coefficient of the Dolby and generalized approximate method, one can see that to obtain the same approximated MDST coefficient, generalized approximate method requires lower number of multiplications with less total number of $ntaps_{\text{total}}$ and hence, less total number of operations than the Dolby approximate method. The achieved reduced multiplicative complexity is based on the fact that (6.186) for the computation of $\{b_k^*\}$ directly incorporates the symmetry property of $\{h_{2q+1}^{(2)}\}$ by combining pairs of MDCT coefficients multiplied by the same value of β_q. Consequently, since $\{f_p\}$ is more dominant in the approximate computation than $\{\beta_q\}$, increasing the number of taps for $\{f_p\}$ in generalized approximate method for the same value of $ntaps_{\text{total}}$ will lead to obtaining the approximated MDST coefficient with higher accuracy, while the total number of required operations is approximately the same compared to the Dolby approximate method.

Table 6.5 summarizes the arithmetic complexity in terms of the total number of operations required for the complete set of MDST coefficients by the generalized approximate method for different 2^n-length blocks and $ntaps_{\text{total}} = 6,\ 11,\ 16,\ 21$. It can be observed that the generalized approximate method requires significantly lower total number of operations than the Dolby approximate method and direct transform-based methods for almost all combinations of block sizes, as well as lower number of $ntaps_{\text{total}}$. This advantage is more apparent over the Dolby approximate and direct transform-based methods as the block size increases. The advantage is even more evident for the sine windowing function. In this specific case, the computation of $\{b_k\}$ is realized by (6.182) requiring one multiplication by $\frac{1}{2}$ and two additions per unique MDST coefficient. Then, for a fixed $ntaps_{\text{total}}$ exact identifying of the values $f_p = 0$ (see Eqs. (6.172) and (6.173)) results in a significant reduction of the total number of arithmetic operations. The memory requirements of the generalized approximate conversion method are the same as for the exact conversion method.

6.7.7.2 Performance Analysis

The performance of Dolby approximate conversion method has been analyzed in [37] by measuring the difference between the exact and approximate spectral components, magnitudes defined by (6.109) and phase angles defined by (6.110), for different block lengths, windowing functions, source signals, and accuracy specifications in terms of $ntaps_{\text{total}}$. In this performance analysis, special attention was given to $ntaps_{\text{total}}$ parameter because it provides the best control over the behavior of the approximate method. The exact spectral components were computed using the exact MDCT and exact MDST, while approximate spectral components were computed using the exact MDCT and approximate MDST. The source input signal has been modeled either as a noise source being the white noise uniformly distributed between -32767 and $+32768$, or as a sinusoidal source being 1440.324 Hz sine wave having the amplitude of 32768. To be more general, the source signals have been normalized to $< -1, 1 >$ range. In simulations 512-point analysis/synthesis KBD windowing function applied to the noise source signal has been used, as a typical application for the AC-3 (E-AC-3) audio coding standard [10], and 2048-point analysis/synthesis sine windowing function applied to the sinusoidal source signal, as a typical application for the MPEG AAC audio coding standard [3].

The performance analysis of Dolby approximate method has involved measuring the difference between the exact and approximate spectral components by investigating [37]:

- *The accuracy of approximation for* $ntaps_{\text{total}}$ shown in Table 6.2. Simulations have shown that the approximation of spectral components is improved as $ntaps_{\text{total}}$ increases. While for $ntaps_{vtotal} = 7$ the approximation is fairly poor, for $ntaps_{\text{total}} \geq 19$ the errors are small enough and obtained approximation may be acceptable for some audio codec configurations, while saving considerable computational power at the decoder.
- *The effects of a longer block size, different source signal, and different windowing function on the quality of approximation*. Simulations have shown that errors are distributed relatively evenly across the spectra both for the noise and sinusoidal sources, with only a few isolated frequency coefficients having larger errors.
- *The error statistics of approximation (average, maximum, median, and variance) for different source signals and all combinations of block sizes and* $ntaps_{\text{total}}$. Simulations have shown that for increasing $ntaps_{\text{total}}$ in general, all error statistics decrease. The block length does not seem to have a predictable effect on the error statistics.

One can find in [37] all detailed results of experiments.

Since the discrete sequence $\{f_p\}$ coincides with $\{h_p^{(1)}\}$ and the discrete sequence $\{\beta_q\}$ coincides with the first half of $\{h_{2q+1}^{(2)}\}$, the above brief summary of performance analysis is also valid for the generalized approximate conversion method [34]. The performance analysis of generalized approximate method involved also

measures the difference between the exact and approximate spectral components. In contrast to [37], for 512-point KBD windowing function applied to the same noise source signal, additionally the following problems have been investigated [34]:

- *Effects of the distribution of taps between $\{f_p\}$ and $\{\beta_q\}$ and their impact on the accuracy of approximation by measuring the average, maximum, and root-mean-square (RMSE) errors for a given $ntaps_{\text{total}}$*. The error statistics of approximate spectral components have been measured for various combinations of allocated taps $p_{\max}$ and $q_{\max}$. It was observed that for $ntaps_{\text{total}} = 19$ and 20, fixing the number of $\{f_p\}$ taps ($p_{\max}$) and increasing the number of $\{\beta_q\}$ taps ($q_{\max}$), the error statistics significantly decreased compared to those for $ntaps_{\text{total}} = 18$, i.e., the approximation accuracy has improved significantly. On the other hand, by increasing the number of $\{f_p\}$ taps ($p_{\max}$) while fixing the number of $\{\beta_q\}$ taps ($q_{\max}$), the error statistics remain similar, but both are significantly higher. This indicates that the distribution of taps for $ntaps_{\text{total}} = 18,\ 19,\ 20$ has considerable impact on the accuracy of the approximation method. However, for increasing $ntaps_{\text{total}} > 20$ almost all error statistics decrease smoothly. It was also shown that the Dolby approximate method with $ntaps_{\text{total}} = 25$ and generalized approximate method with $ntaps_{\text{total}} = 21$ give exactly the same error statistics. It means that the approximate spectral components computed by the generalized approximate method with almost the same computational complexity are obtained with higher accuracy.
- *Decaying rates of $\{f_p\}$ and $\{\beta_q\}$ for different block lengths and different windowing functions*. In general, proper selection and distribution of taps depend on the decaying rates of discrete sequences $\{f_p\}$ and $\{\beta_q\}$ whose values depend on the block length and employed windowing function. As an example, the first ten elements of $\{f_p\}$ and $\{\beta_q\}$ for 512-point KBD windowing function are shown in Table 6.6.

  In order to get an insight into decaying rates of $\{f_p\}$ and $\{\beta_q\}$, the following simple and useful experiment was carried out. For each windowing function and block length were searched for such indices $p_{\max}$ and $q_{\max}$ for which the elements of both sequences $\{f_p\}$, where $p \leq p_{\max}$, and $\{\beta_q\}$, where $q < q_{\max}$,

Table 6.6 The first ten elements of $\{f_p\}$ and $\{\beta_q\}$ for 512-point KBD windowing function

p, q	$\{f_p\}$	$\{\beta_q\}$
0	1.933E-001	−5.914E-001
1	−1.729E-001	−1.079E-001
2	1.228E-001	−1.793E-002
3	−6.796E-002	−1.429E-003
4	−8.512E-003	−2.302E-005
5	2.065E-003	−3.022E-008
6	−6.588E-004	−8.216E-009
7	2.045E-004	1.098E-008
8	1.604E-005	−3.151E-009
9	−3.921E-006	−1.539E-009

have a magnitude greater than a pre-specified constant. Thus, for the approximate method values $p_{\max}$ and $q_{\max}$ will determine truncation limits or the number of allocated taps to $\{f_p\}$ and $\{\beta_q\}$, respectively, when higher accuracy of the approximate spectral components is required. For the KBD windowing function and constants 10^{-5} and 10^{-6}, the values of $p_{\max}$ and $q_{\max}$ do not change for different block lengths in contrast to the sine windowing function. This explains the behavior of the approximation method in terms of accuracy by specific distribution of the taps between $\{f_p\}$ and $\{\beta_q\}$ for 512-point KBD windowing function and $ntaps_{\text{total}} = 18,\ 19,\ 20$. Experimental results confirm the conclusion given in [37] that the decay rates of $\{f_p\}$ and $\{\beta_q\}$ for different windowing functions are influenced by the increased frequency resolution property of a specific windowing function. The better frequency resolution of the windowing function, the faster is the decay rate of $\{f_p\}$ and $\{\beta_q\}$. We recall that the taps in the Dolby approximate method [37] are distributed according to Eq. (6.185), i.e., approximately two times more taps are allocated to $ntaps_{I,III}$ than to $ntaps_{II}$. Indeed, it was observed that for KBD windowing function and the constant 10^{-5} approximately two times more taps are allocated to $\{f_p\}$ than to $\{\beta_q\}$. However, for the constant 10^{-6} approximately three times more taps are allocated to $\{f_p\}$ than to $\{\beta_q\}$ for higher accuracy. For the sine windowing function and increasing block length, it is interesting to note the changing values of $p_{\max}$. They seem to approach a constant value [34].

One can find in [34] all detailed results of experiments. However, three essential open problems still remain to be solved:

1. Why the decaying rates of $\{f_p\}$ and $\{\beta_q\}$ for the KBD windowing function do not depend on the block length? An asymptotic investigation of the behavior of $\{f_p\}$ and $\{\beta_q\}$ as the block length goes to the infinity has to be carried out. Note that even/odd terms of $\{f_p\}$ given by (6.148) and $\{\beta_q\}$ given by (6.164) can be written in the following equivalent forms as

$$f_{2q} = (-1)^q \frac{4}{N} \sum_{n=0}^{\frac{N}{4}-1} w_n\, w_{\frac{N}{2}-1-n}\, \cos\left[\frac{\pi}{N}(2n+1)2q\right],$$

$$f_{\frac{N}{2}-1-2q} = (-1)^q \frac{4}{N} \sum_{n=0}^{\frac{N}{4}-1} (-1)^n w_n\, w_{\frac{N}{2}-1-n}\, \cos\left[\frac{\pi}{N}(2n+1)(2q+1)\right],$$

$$q = 0, 1, \ldots, \frac{N}{4} - 1,$$

$$\beta_q = (-1)^q \frac{4}{N} \sum_{n=0}^{\frac{N}{4}-1} (w_n + w_{\frac{N}{2}-1-n})(w_n - w_{\frac{N}{2}-1-n})$$

$$\times \cos\left[\frac{\pi}{N}(2n+1)(2q+1)\right].$$

2. Why $\{\beta_q\}$ generally decays faster than $\{f_p\}$?
3. Given $p_{\max}$ and $q_{\max}$, can the approximation accuracy be estimated? Such a formula would be very useful for the approximate conversion method in real audio applications.

Compared to the direct transform-based method, the approximate conversion methods for approximation of spectral components at the decoder have the following advantages:

- With properly selected parameters $p_{\max}$ and $q_{\max}$ they can produce acceptable approximate results, while saving considerable computational power particularly at resource-limited and low-cost decoders.
- They are very flexible to compute the approximate MDST coefficients in different frequency bands with different accuracies including the exact computation.

6.8 Summary

The perfect reconstruction cosine/sine-modulated filter banks used in the Dolby AC-3 and E-AC-3 audio coding standards have been discussed, specifically, definitions of the analysis/synthesis AC-3 filter banks, their general symmetry properties both in the time and frequency domains, and their efficient unified implementations with corresponding signal flow graphs. The efficient unified implementations of AC-3 transforms are compared in terms of computational complexity and structural simplicity. Matrix representations of AC-3 filter banks, their properties, and relations among transform (sub-)matrices provided the basis to derive relations between the frequency coefficients and the time domain aliasing data sequences of AC-3 transforms, and in particular, the derivation of the fast conversion algorithm of frequency coefficients of AC-3 transforms directly in the frequency domain for efficient E-AC-3 to AC-3 bit stream conversion and AC-3 to E-AC-3 bit stream transcoding. The fast conversion algorithm is compared with standard conversion methods in terms of computational complexity and memory requirements. Finally, exact and approximate conversion methods of the MDCT to MDST frequency coefficients, or equivalently the methods for construction of complex MCLT filter bank directly in the frequency domain have been presented. Exact and approximate conversion methods are compared in terms of computational complexity, structural simplicity, and memory requirements.

Problems and Exercises

1. Based on Eqs. (6.5), (6.6), and (6.7) for a given block size N, write a computer program to generate the KBD windowing function with variable parameter β. If it is possible, display the graph of generated KBD windowing function for

the selected parameter β. Note that the modified zeroth order Bessel function given by (6.7) converges rapidly after several iterations for a given argument and accuracy.

2. In the AC-3 and E-AC-3 encoder/decoder, the KBD windowing procedure is realized via lookup tables before/after the forward/backward long (MDCT) transform computation (standard method). Following the approach described in Chap. 5 for MP3 audio coding standard, construct a fast analysis MDCT filter bank (including the forward long transform with incorporated KBD windowing and overlap procedures), and a fast synthesis MDCT filter bank (including the backward long transform with incorporated KBD windowing, overlap, and add procedures). Draw corresponding general compact computational structures and verify them by a computer program. Finally, compare the arithmetic complexity of standard method with the implemented fast analysis/synthesis MDCT filter banks.
3. Consider Eqs. (6.14) and (6.15) for $n = m$ defining the fast DCT-IV-based algorithm for the forward first short transform computation. Further, consider Eqs. (6.38) and (6.39) defining the forward first short transform in the unified form of forward long (MDCT) transform. When we adopt the DCT-IV-based fast MDCT algorithm given by (6.27) or (6.28), and (6.29) for its efficient computation, whereby $N = M$ and $n = m$, show that the data sequences $\{y_m^{(c)}\}$ in (6.15) and $\{y_m^{(-1)}\}$ in (6.39) are equivalent.
4. Similarly, consider Eqs. (6.23) and (6.24) for $n = m$ defining the fast DCT-IV-based algorithm for the forward second short transform computation. Further, consider Eqs. (6.40) and (6.41) defining the forward second short transform in the unified form of forward long (MDCT) transform. When we adopt the DCT-IV-based fast MDCT algorithm given by (6.27) or (6.28), and (6.29) for its efficient computation, whereby $N = M$ and $n = m$, show that the data sequences $\{y_m^{(s)}\}$ in (6.24) and $\{y_m^{(+1)}\}$ in (6.41) are equivalent.
5. Implement the computation of forward first short transform derived in the unified form of forward long (MDCT) transform given by (6.38) and (6.39) and the computation of forward second short transform derived in the unified form of forward long (MDCT) transform given by (6.40) and (6.41), via a fast DCT-IV-based MDCT computational structure.
6. Implement the alternate computation of forward/backward first and second short transforms via the modified fast MDCT computational structure discussed in Sect. 6.3.3, and verify Eqs. (6.48), (6.49) and (6.51), (6.52).
7. Implement the alternate computation of forward/backward first and second short transforms by the modified fast GDFT-IV computational structure discussed in Sect. 6.3.4.
8. In the E-AC-3 encoder after the time-to-frequency transformation of successive overlapped audio blocks, six blocks of MDCT coefficients are packed and transformed by the DCT-II resulting in the block of 1536 coefficients. In the E-AC-3 decoder, this block of DCT-II coefficients is inverted by the inverse DCT-II, DCT-III. The block length is a composite number, i.e., $1536 = 2^9 \times 3$.

Implement the efficient DCT-II/DCT-III computation for the composite lengths $2^n \times 3,\ n > 0$. The even-length fast recursive DCT-II algorithm is presented in Appendix C.1.1, whereas the required efficient optimized 3-point DCT-II/DCT-III modules are presented in Appendix D.4.

9. Verify the products of the matrix $\boldsymbol{C}_{\frac{N}{2}\times N}$ and its pseudoinverse, $\boldsymbol{C}^{+}_{N\times\frac{N}{2}}$, given by (6.61) and (6.62) for $N = 4$ and 8.
10. Verify the products of matrix $\boldsymbol{C}^{(1)}_{\frac{M}{2}\times M}$ and its pseudoinverse, $\left[\boldsymbol{C}^{(1)}_{M\times\frac{M}{2}}\right]^{+}$, given by (6.68) and (6.69), as well as the products of matrix $\boldsymbol{C}^{(2)}_{\frac{M}{2}\times M}$ and its pseudoinverse, $\left[\boldsymbol{C}^{(2)}_{M\times\frac{M}{2}}\right]^{+}$, given by (6.68) and (6.70), for $M = 4$ and 8.
11. Prove the symmetry properties of row basis vectors of matrices $\boldsymbol{C}_{\frac{N}{2}\times N}$, $\boldsymbol{C}^{(1)}_{\frac{M}{2}\times M}$ and $\boldsymbol{C}^{(2)}_{\frac{M}{2}\times M}$ defined by (6.71), (6.72) and (6.73), respectively.
12. Prove Eqs. (6.79) and (6.80).
13. Derive Eq. (6.84) using Eqs. (6.81), (6.82), and (6.83).
14. Investigate the regular general block structure of conversion matrix $\boldsymbol{V}_M$ defined by (6.95) for $M = 4$ and 8.
15. Verify the orthogonality property of conversion matrix $\boldsymbol{V}_M$ defined by (6.97) for $M = 4$ and 8, and then its orthonormality property.
16. Implement the fast algorithm for conversion of frequency coefficients of AC-3 transforms defined by (6.98), (6.102) and (6.100), (6.103).
17. From (6.116) derive the conversion method of MDCT to MDST coefficients defined by (6.117) and (6.118) for rectangular and sine windowing functions.
18. Implement by a computer program the conversion method defined by (6.117) and (6.118) for rectangular and sine windowing functions. Investigate the real-valued constant e, and real-valued elements of $\{f_{k,m}\}$ and $\{g_{k,m}\}$ for rectangular and sine windowing functions.
19. Implement the Dolby exact conversion method defined by (6.119)–(6.124) for commonly used symmetric windowing functions.
20. Verify the products of the matrix $\boldsymbol{S}_{\frac{N}{2}\times N}$ and its pseudoinverse, $\boldsymbol{S}^{+}_{N\times\frac{N}{2}}$, given by (6.130) and (6.131) for $N = 4$ and 8.
21. Based on (6.137) and (6.138) derive the relation between MDCT and MDST coefficients in the frequency domain defined by (6.139) and (6.140).
22. Using (6.57), (6.75), (6.135), and (6.136), derive the elements $\{u_{k,m}\}$ and $\{v_{k,m}\}$ of conversion matrices $\boldsymbol{U}_{\frac{N}{2}}$ and $\boldsymbol{V}_{\frac{N}{2}}$, respectively, in (6.141) and then derive (6.144) from (6.141). Similarly, derive the elements $\{v_{k,m}\}$ in (6.147) from (6.144).
23. Verify the relation between the elements $\{u_{k,m}\}$ and $\{v_{k,m}\}$ given by (6.145), and their symmetry properties given by (6.146) for $N = 4$ and 8.
24. Using (6.57), (6.75), (6.135), and (6.136), derive the elements $\{(g+h)_{k,m}\}$ of conversion matrix $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ in (6.152) and derive (6.154) from (6.152). Then, derive the general close form of $\boldsymbol{G}_{\frac{N}{2}} + \boldsymbol{H}_{\frac{N}{2}}$ given by (6.158) from (6.154).

25. Verify the symmetry properties of elements $\{(g + h)_{k,m}\}$ given by (6.159) and (6.160) for $N = 4$ and 8.
26. Try to generalize the conversion method defined by (6.117) and (6.118) for arbitrary symmetric windowing function. Hint: Instead of the real-valued constant e, consider the terms $\{e_{k,m}\}$. Then, investigate real values of $\{e_{k,m}\}$, $\{f_{k,m}\}$ and $\{g_{k,m}\}$ for commonly used symmetric windowing functions.
27. Implement by a computer program the generalized exact conversion method defined by (6.165), (6.169), (6.174), (6.175), and (6.182) for commonly used symmetric windowing functions.
28. Consider the computation of spectral measures, the magnitudes, and phase angles of spectral coefficients defined by (6.109) and (6.110), respectively. For a given source input signal and two input parameters $p_{\max}$ and $q_{\max}$, implement by a computer program the Dolby approximate conversion method defined by (6.119)–(6.124) for the KBD windowing function.
29. Similarly, for the same source input signal and two input parameters $p_{\max}$ and $q_{\max}$, implement by a computer program the generalized approximate conversion method defined by (6.169) and (6.186) for the KBD windowing function. Compare the exact spectral measures with those computed by the generalized approximate conversion method and by the Dolby approximate method for the same input parameters $p_{\max}$ and $q_{\max}$.
30. At the end of this chapter three essential open problems related to the approximate conversion methods are formulated. Try to solve them.

References

1. M. Bosi, S.E. Forshay, High quality audio coding for HDTV: an overview of AC-3, in *Proceedings of the 7th International Workshop on HDTV*, Torino, Italy, Oct 1994
2. M. Bosi, R.E. Goldberg, *Introduction to Digital Audio Coding and Standards* (Springer Science+Business Media, New York, 2003), Chap. 5, pp. 103–147, Chap. 14, pp. 371–400
3. M. Bosi et al., ISO/IEC MPEG-2 advanced audio coding, in *101st AES Convention*, Los Angeles, CA, Nov 1996, Preprint #4382
4. G.A. Davidson, M.A. Isnardi, L.D. Fielder, M.S. Goldman, C.C. Todd, ATSC video and audio coding. Proc. IEEE **94**(1), 60–76 (2006)
5. M.F. Davis, The AC-3 multichannel coder, in *95th AES Convention*, New York, NY, Oct 1993, Preprint #3774
6. Digital Audio Compression (AC-3) ATSC Standard, Document A/52/10 of advanced television systems committee (ATSC), Audio specialist group T3/S7, Washington D.C., Dec 1995
7. Digital Audio Compression Standard (AC-3, E-AC-3), Revision B, Document A/52B of advanced television systems committee (ATSC), Washington, D.C., Dec 2012
8. L.D. Fielder, D.P. Robinson, AC-2 and AC-3: the technology and its applications, in *5th Australian Regional Convention*, Sydney, Australia, April 1995, Preprint #4022
9. L.D. Fielder, M. Bosi, G.A. Davidson, M.F. Davis, C.C. Todd, S. Vernon, AC-2 and AC-3: low-complexity transform-based audio coding, in AES publication collected papers on *Digital Audio Bit-Rate Reduction*, ed. by N. Gilchrist, C. Grewin (Audio Engineering Society, San Francisco, 1996), pp. 54–72

10. L.D. Fielder et al., Introduction to Dolby digital plus, an enhancement to the Dolby digital coding system, in *117th AES Convention*, San Francisco, CA, Oct 2004, Preprint #6196
11. V.K. Madisetti (ed.), *The Digital Signal Processing Handbook*, vol. 3, 2nd edn., Video, Speech, and Audio Signal Processing and Associated Standards (CRC Press, Boca Raton, 2010), Part I-3: Dolby digital audio coding standards, pp. 3.1–3.46
12. V.K. Madisetti, D.B. Williams (eds.), Digital audio coding: Dolby AC-3, in *The Digital Signal Processing Handbook* (CRC Press, Boca Raton, 1998), Chap. 41, pp. 41.1–41.21
13. T. Painter, A. Spanias, Perceptual coding of digital audio. Proc. IEEE **88**(4), 451–513 (2000)
14. J.P. Princen, A.W. Johnson, A.B. Bradley, Sub-band/transform coding using filter bank designs based on time domain aliasing cancellation, in *Proceedings of the IEEE ICASSP'87*, Dallas, TX, April 1987, pp. 2161–2164
15. A. Spanias, T. Painter, V. Atti, *Audio Signal Processing and Coding* (Wiley, Hoboken, 2007), Chap. 6, pp. 145–193, Chap. 10, pp. 263–342
16. C.C. Todd, G.A. Davidson, M.F. Davis, L.D. Fielder, B.D. Link, S. Vernon, AC-3: flexible perceptual coding for audio transmission and storage, in *96th AES Convention*, Amsterdam, Feb 1994, Preprint #3796

Efficient Unified Implementations of AC-3 Transforms

17. M. Bosi-Goldberg, Analysis-/synthesis-filtering system with efficient oddly-stacked single-band filter bank using time-domain aliasing cancellation, US Patent Application 5,890,106, Dolby Laboratories, San Francisco, CA, March 1999
18. V. Britanak, The refined efficient implementation of the MDCT in MP3 and comparison with other methods. Technical Report II SAS-2002-02, Sept 2002
19. V. Britanak, New universal rotation-based fast computational structures for an efficient implementation of the DCT-IV/DST-IV and analysis/synthesis MDCT/MDST filter banks. Signal Process. **89**(11), 2213–2232 (2009)
20. V. Britanak, K.R. Rao, The fast generalized discrete Fourier transforms: a unified approach to the discrete sinusoidal transforms computation. Signal Process. **79**(12), 135–150 (1999)
21. V. Britanak, K.R. Rao, A new fast algorithm for the unified forward and inverse MDCT/MDST computation. Signal Process. **82**(3), 433–459 (2002)
22. Y.-C. Chen, C.-W. Tsai, J.-L. Wu, Fast time-frequency transform algorithms and their applications to real-time software implementation of AC-3 codec. IEEE Trans. Consum. Electron. **44**(2), 413–423 (1998)
23. S. Cramer, R. Gluth, Computationally efficient real-valued filter banks based on a modified O^2 DFT, in *Proceedings of EUSIPCO'90, Signal Processing V: Theories and Applications* (Elsevier Science Publishers B.V., Barcelona, Sept 1990), pp. 585–588
24. R. Gluth, Regular FFT-related transform kernels for DCT/DST–based polyphase filter banks, in *Proceedings of the IEEE ICASSP'91*, Toronto, Canada, May 1991, pp. 2205–2208
25. C.-M. Liu, W.-C. Lee, A unified fast algorithm for cosine-modulated filter banks in current audio coding standards. J. Audio Eng. Soc. **47**(12), 1061–1075 (1999)
26. X. Shao, S.G. Johnson, Type-IV DCT, DST, and MDCT algorithms with reduced numbers of arithmetic operations. Signal Process. **88**(6), 1313–1326 (2008)
27. T. Zhang, S. Liu, J. He, H. Zhang, A new algorithm on short window MDCT for Dolby AC3, in *Proceedings of the International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS'2007)*, Xiamen, China, Nov–Dec 2007, pp. 478–481
28. T. Zhang, J. He, C. Chen, On the relationship of MDCT transform kernels in AC-3, in *Proceedings of the International Conference on Audio, Language and Image Processing, (ICALIP'2009)*, Shanghai, China, July 2008, pp. 839–842

Matrix Representations of AC-3 Transforms, Relations Between the Frequency Coefficients and the Time Domain Aliasing Data Sequences

29. V. Britanak, On properties, relations, and simplified implementations of filter banks in the Dolby Digital (Plus) AC-3 audio coding standards. IEEE Trans. Audio Speech Lang. Process. **19**(5), 1231–1241 (2011)
30. V. Britanak, H.J. Lincklaen Arriëns, Fast computational structures for an efficient implementation of the complete TDAC analysis/synthesis MDCT/MDST filter banks. Signal Process. **89**(7), 1379–1394 (2009)
31. S.-W. Lee, C.-M. Liu, Transformation from 512–point transform coefficients to 256–point transform coefficients for Dolby AC-3 decoder. Electron. Lett. **35**(19), 1614–1615 (1999)

Fast Algorithm for Conversion of Frequency Coefficients of AC-3 Transforms

32. V. Britanak, Fast conversion algorithm for the Dolby Digital (Plus) AC-3 audio coding standards. IEEE Signal Process Lett. **19**(12), 910–913 (2012)
33. V. Britanak, Fast conversion algorithm for the Dolby Digital (Plus) AC-3 audio coding standards. Presented in the IEEE ICASSP'2013, Vancouver, Canada, May 2013

Conversion Methods of the MDCT to MDST Coefficients Directly in the Frequency domain

34. V. Britanak, New generalized conversion method of the MDCT to MDST coefficients in the frequency domain for arbitrary symmetric windowing function. Digital Signal Process. **23**(5), 1783–1797 (2013)
35. S. Chen, R. Hu, S. Zhang, Estimating spatial cues for audio coding in MDCT domain, in *Proceedings of the IEEE International Conference on Multimedia and Expo (IMCE'2009)*, Cancun, Mexico, June–July 2009, pp. 53–56
36. S. Chen, N. Xiong, J.H. Park, M. Chen, R. Hu, Spatial parameters for audio coding: MDCT domain analysis and synthesis. Multimed. Tools Appl. **48**(2), 225–246 (2010)
37. C.I. Cheng, Method for estimating magnitude and phase in the MDCT domain, in *116th AES Convention*, Berlin, Germany, May 2004, Preprint #6091
38. C.I. Cheng, M.J. Smithers, D.N. Lathrop, Improved coding techniques using estimated spectral magnitude and phase derived from MDCT coefficients, US Patent Application US2005/001499, Dolby Laboratories, San Francisco, CA, Aug 2005
39. G.A. Davidson, S.D. Vernon, Method and apparatus for efficient implementation of single-sideband filter banks providing accurate measures of spectral magnitude and phase, US Patent Application #5,727,119, Dolby Laboratories, San Francisco, CA, March 1998
40. B. Edler, S. Geyersberger, Arrangement and method for the generation of a complex spectral representation of a time-discrete signal, International Patent Application WO 2004/013839 A1, Fraunhofer Institute, München, Germany, Feb 2004
41. B. Edler, S. Geyersberger, Device and method for generating a complex spectral representation of a discrete–time signal, US Patent Application No. 11/044786, Fraunhofer Institute, München, Germany, Jan 2005

42. F.J. Harris, On the use of windows for harmonic analysis with the discrete Fourier transform. Proc. IEEE **66**(1), 51–83 (1978)
43. F. Kuech, B. Edler, Aliasing reduction for modified discrete cosine transform domain filtering and its application to speech enhancement, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, Oct 2007, pp. 131–134
44. H. Malvar, A modulated complex lapped transform and its applications to audio processing, in *Proceedings of the IEEE ICASSP'99*, Phoenix, AR, May 1999, pp. 1421–1424
45. H. Ofir, D. Malah, I. Cohen, Audio packet loss concealment in a combined MDCT-MDST domain. IEEE Signal Process. Lett. **14**(12), 1032–1035 (2007)
46. S.-U. Ryu , K. Rose, A frame loss concealment technique for MPEG AAC, in *120th AES Convention*, Paris, France, May 2006, Preprint #6662
47. S.-U. Ryu, K. Rose, An MDCT domain frame-loss concealment technique for MPEG advanced audio coding, in *Proceedings of the IEEE ICASSP'2007*, vol. I, Honolulu, HI, April 2007, pp. 273–276
48. B.-J. Yoon, H.S. Malvar, Coding overcomplete representations of audio using the MCLT, in *Proceedings of the IEEE Data Compression Conference*, Snowbird, UT, March 2008, pp. 152–161

Supporting Literature

49. V. Britanak, P. Yip, K.R. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations* (Academic Press/Elsevier Science, Amsterdam, 2007)
50. V. Britanak, A note on the MDCT/MDST and pseudoinverse matrix. Comput. Inform. **23**(3), 205–214 (2004)
51. F.R. Gantmacher, *The Theory of Matrices*, 2nd edn. (Nauka, Moscow, 1966) (in Russian), English translation: Vol. 1 and 2, Chelsea, New York, 1959
52. G.H. Golub, C.F. Van Loan, *Matrix Computations*, 3rd edn. (The Johns Hopkins University Press, Baltimore, 1996)
53. H.S. Malvar, *Signal Processing with Lapped Transforms* (Artech House, Norwood, 1992)
54. G.W. Stewart, *Matrix Algorithms, Volume I: Basic Decompositions* (SIAM Society for Industrial and Applied Mathematics, Philadephia, 1998)

Chapter 7
Spectral Band Replication Compression Technology: Efficient Implementations of Complex Exponential- and Cosine-Modulated QMF Banks

7.1 Introduction

Spectral Band Replication (SBR) [1, 2, 4] is an enhancement compression technology originally invented, developed, and marketed by Coding Technologies. In November 2007 Coding Technologies company merged into Dolby Laboratories, now the leading international company in the audio coding field. The SBR is a bandwidth extension method [2] which significantly improves the compression efficiency (coding gain) of perceptual audio and speech coding schemes. SBR cannot be used as a stand-alone coder, it always operates in conjunction with a conventional codec, a core codec. The SBR acts as pre-processing to the encoder, and as post-processing to the decoder. In general, the SBR can be combined with any conventional (even not necessary perceptual) audio/speech codec. The SBR is based on the fact that in most cases there are large dependencies or strong correlation between the characteristics of lower and higher frequency content of an audio signal. Consequently, the high frequency part can be reconstructed from the low frequency part, or in other words, the SBR is able to recreate the missing high frequency components of a decoded audio signal in a perceptually accurate way by reusing signal information from the decoded low frequency part, thus allowing a much higher audio quality at low data rates. Therefore, transmission of the high frequency part is not necessary, only the low frequency part and a small set of control data need to be carried in the bit stream to guarantee an optimal reconstruction of high frequencies. Thus, the core codec is responsible only for coding and transmitting the lower part of the original signal spectrum [1, 4].

SBR-enhanced codecs have the major advantage of being backward and forward compatible to the core codec. This permits to integrate the SBR technology to existing systems, thus enabling a smooth transition from a conventional audio coder to its more efficient SBR-enhanced version. In December 2001, the SBR was chosen as initial reference model for the MPEG standardization process for bandwidth extension [2], a work item finalized in March 2003 [23]. Indeed, the SBR technology

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1_7

has been initially successfully integrated into three existing international audio coding standards: MPEG-1 audio layer II [10], MPEG-1/2 audio layer III known as MP3 [12], and MPEG-2/4 AAC (Advanced Audio Coding) [7], all being parts of the open ISO/MPEG standard. The SBR-enhanced version of MPEG-1 audio layer II [10] has been adopted as a source coder by the Digital Audio Broadcasting (DAB) system for digital radio services [8]. The SBR-enhanced version of MP3 [12], called mp3PRO, released and marketed by Thomson Multimedia led to several both software- and hardware-based commercial products [2]. The combination of SBR and MPEG-4 AAC, the so-called MPEG-4 High Efficiency AAC (HE-AAC) or aacPlus standard [13–18, 22, 23], has been adopted as a source coder by the advanced DAB digital broadcasting system (DAB+) [8], by the Digital Radio Mondiale (DRM) universal openly standardized digital broadcasting system [6, 8, 9], as well as by the XM Satellite Radio [11] being one of two satellite-based digital radio services (XM Satellite Radio and Sirius Satellite Radio) used in United States and Canada. In July 2008, XM Satellite Radio and Sirius Satellite Radio merged forming Sirius XM Radio. Recently, the combination of SBR and modified MPEG-4 AAC has been adopted by the MPEG-D Unified Speech and Audio Coding (USAC) standard for high-quality coding of both speech and music signals [19–21]. The MPEG committee also completed development of a new audio coding standard, the MPEG-4 AAC Enhanced Low Delay (AAC-ELD) [24–30]. In order to minimize algorithmic delays while maintaining high coding efficiency at low bit rates, the AAC-ELD uses a special low delay optimized version of SBR technology. Both the HE-AAC and AAC-ELD currently belong to the state-of-the-art audio codecs. Targeted applications involve digital audio broadcasting (proprietary digital radio systems, digital radio standards), TV broadcasting, mobile communications, commercial electronics (HDTV, DVD), internet audio/video streaming, and in particular, the real-time bidirectional audio/video teleconferencing. A comprehensive list of real-world audio applications can be found in [17].

In this chapter, the complete unified efficient low-cost implementations of complex exponential-modulated and real-valued cosine-modulated analysis and synthesis QMF banks used in the standard SBR (HQ-SBR and LP-SBR) and LD-SBR (HQ-LD-SBR and LP-LD-SBR) encoder and decoder are presented. They are based on existing fast algorithms for the discrete cosine/sine transform of type IV (DCT-IV/DST-IV), and discrete cosine transforms of type II and III (DCT-II/DCT-III). The unified efficient implementations are efficient in terms of the computational complexity, regularity, and structural simplicity. In general, for each QMF bank is presented:

- Definition in its equivalent block transform with a common parameter M representing the number of sub-bands. The parameter M has a fixed value of $M = 64$ both for the number of sub-bands and time shift factors in the transform kernels.
- Its general symmetry property in the frequency or time domain which is very useful for the derivation of a fast algorithm.
- The derivation of a fast algorithm with corresponding computational complexity.

All fast algorithms are analyzed in detail in terms of the regularity and structural simplicity for a potential real-time low-cost implementation in hardware or software. Appendices contain the required fast DCT-IV and DCT-II/DCT-III computational structures to complete the unified efficient low-cost implementations of complex exponential-modulated and real-valued cosine-modulated analysis and synthesis QMF banks.

7.2 Overview of the SBR Compression Technology

There are two versions of the SBR technology in the encoder and decoder: Standard SBR and low delay SBR (LD-SBR). Basic principles how the both standard SBR and LD-SBR work in the encoder and decoder are well described in [1, 2, 4, 23].

7.2.1 High-Quality and Low-Power SBR and LD-SBR

Central to the operation of both standard SBR and LD-SBR are dedicated complex exponential-modulated and real-valued cosine-modulated (low delay) quadrature mirror filter (QMF) banks as the basic mathematical tools to analyze and synthesize audio signals. While the standard SBR for the encoder defines only one complex exponential-modulated analysis QMF bank [18], the standard SBR for the decoder defines two types of analysis and synthesis QMF banks depending on the application: Complex exponential-modulated QMF banks forming the high quality SBR (HQ-SBR), and real-valued cosine-modulated QMF banks forming the low power SBR (LP-SBR) [3, 5]. Since the LD-SBR [25] is derived from the standard SBR with some modifications, similarly, the LD-SBR for the encoder defines also only one complex exponential-modulated low delay analysis QMF bank and two types of low delay analysis and synthesis QMF banks for the decoder forming the high quality LD-SBR (HQ-LD-SBR) and low power LD-SBR (LP-LD-SBR). The main difference between HQ-SBR (HQ-LD-SBR) and LP-SBR (LP-LD-SBR) is how the data is represented during the SBR process. In the HQ-SBR (HQ-LD-SBR) all subsequent calculations are realized in complex arithmetic. The LP-SBR (LP-LD-SBR) operates with real-valued cosine-modulated (low delay) analysis and synthesis QMF banks, and hence subsequent real-valued arithmetic, to reduce the computational complexity. In situations, where a lower sampling rate is sufficient, for example in portable devices, the SBR can run in a down-sampled mode and down-sampled versions of complex and real-valued (low delay) synthesis QMF banks can be employed. The complex exponential-modulated (low delay) QMF banks are intended for use in applications requiring the best possible audio quality at a given bit rate, while the real-valued (low delay) QMF banks are intended to be lower complexity versions that still produce acceptable results in terms of audio quality and bit rate [33, 34].

7.2.2 Motivation to Develop Efficient Implementations of QMF Banks

The modulation stages of the QMF banks in the ISO/MPEG standard documents [18, 25] are defined using matrix-vector products with the number of sub-bands being 64 or 32 and with fixed values of time shift factors in the transform kernels. They are the most time-consuming procedures in decoders. Indeed, from the study of the HE-AAC codec [23] it is known that the SBR process in the decoder may constitute from 50% to 75% of its total computational complexity. Audio and speech decoders are used in mobile or portable devices, where the processing power and battery resources are limited. Consequently, in such environments a particular attention must be paid to reducing the decoder complexity. Moreover, if the decoder is intended to be implemented in hardware, it is well known that the computational complexity and memory requirements are directly related to the power consumption and/or chip size as well as the implementation costs [23]. This is a reason why the HE-AAC and AAC-ELD define two types of QMF banks in the decoders. Nevertheless, even when the LP-SBR (LP-LD-SBR) is used, the contribution of the QMF banks to the overall decoder complexity is significant [33, 34]. Therefore, efficient implementations of the QMF banks are of great importance.

7.2.3 Existing Efficient Implementations of QMF Banks

After finalizing the MPEG-4 HE-AAC standard in 2004 [18], several efficient implementations of QMF banks have been developed both for the HQ-SBR [39, 40] and LP-SBR decoder [32, 38]. The real and imaginary parts of the complex exponential modulation step of the QMF are, respectively, mapped into the DCT-IV and the corresponding DST-IV of reduced sizes [39, 40], while the cosine modulation of real-valued QMF banks is mapped either into the DCT-II or its inverse, DCT-III, of a reduced size [32, 38]. Thus, many available fast DCT-IV/DST-IV computational structures [43, 45–47, 51, 52] offer efficient implementations of the QMF banks in HQ-SBR, and the fast DCT-II/DCT-III computational structures [42, 44, 46, 48, 51, 57] in LP-SBR. The definitions of the low delay QMF banks in the MPEG-4 AAC-ELD standard [25] are basically similar to those in the standard SBR. Only the windowing function and modulation differ. Efficient implementations of the QMF banks both for the HQ-LD-SBR and LP-LD-SBR have been proposed in [33, 34]. Similarly, the real and imaginary parts of the complex exponential modulation step of the low delay QMF are mapped into the DCT-IV and the corresponding DST-IV of reduced sizes. The cosine modulation of real-valued low delay QMF banks is the same as the real part of the corresponding complex exponential-modulated low delay QMF banks. In general, since there exists a relation between the DCT-IV and DCT-II [44, 46, 48], the DCT-IV may always be converted to the DCT-II of the same size at the cost of additional multiplications

and recursive additions. This approach allows absorbing additional multiplications in the windowing stage, thus further reducing the multiplicative complexity (see Appendix C.3). The complete unified efficient low-cost implementations of complex exponential-modulated and real-valued cosine-modulated analysis and synthesis QMF banks used in the standard SBR (HQ-SBR and LP-SBR) and LD-SBR (HQ-LD-SBR and LP-LD-SBR) encoder and decoder are presented in [31].

Besides the aforementioned developed efficient implementations of QMF banks there are the so-called firmware implementations for the standard SBR decoder available. Specifically, a fixed-point firmware reference (FFR) code for various computer platforms licensed by Coding Technologies [35, 36] which is optimized in terms of the memory usage and processing power. Further, QMF functions for the SBR decoder developed by INTEL are described in its Integrated Performance Primitives Reference Manual [41]. However, for a reader perhaps the most interesting and easily available is the free-ware open source code for efficient implementations of QMF banks in the standard SBR decoder, the so-called FAAD2 software package [37].

7.3 QMF Banks: Definitions, Symmetry Properties, and Efficient Implementations

The theory of complex (low delay) QMF banks [2, 18, 40] used in the standard SBR and LD-SBR technologies is a complex exponential extension of the theory of real-valued cosine-modulated (low delay) QMF banks [54–56]. The excellent overview of the technical details of complex exponential-modulated and real-valued cosine-modulated QMF banks used in the standard SBR is presented in [2].

7.3.1 Standard SBR QMF Banks: Definitions and Symmetry Properties

In the following subsections the definitions and symmetry properties of complex exponential-modulated analysis/(down-sampled) synthesis QMF banks used in the standard HQ-SBR are presented.

7.3.1.1 Complex Exponential-Modulated Analysis QMF Bank in the Encoder

The complex exponential-modulated M-band ($M = 64$) analysis QMF bank in the encoder as a block transform is defined as [18]

$$p_k = \sum_{n=0}^{2M-1} u_n \exp\left[i \frac{\pi}{2M} \left(k + \frac{1}{2} \right) (2n+1) \right],$$
$$k = 0, 1, \ldots, M-1, \tag{7.1}$$

where $\{p_k\}$ are complex-valued sub-band coefficients and $\{u_n\}$ is a $2M$-point windowed data sequence. The flowchart of one loop of the sub-band filtering procedure in the encoder is shown in [18] (see Fig. 4.B.16). Since exp [.] denotes the complex exponential function and $i = \sqrt{-1}$ is the imaginary unit, Eq. (7.1) can be extended as

$$p_k = \sum_{n=0}^{2M-1} u_n \cos\left[\frac{\pi}{4M} (2k+1)(2n+1) \right] + i \sum_{n=0}^{2M-1} u_n \sin\left[\frac{\pi}{4M} (2k+1)(2n+1) \right],$$
$$k = 0, 1, \ldots, M-1, \tag{7.2}$$

or equivalently, decomposed into two parts, the real and the imaginary part, respectively, as

$$\Re e\,\{p_k\} = \sum_{n=0}^{2M-1} u_n \cos\left[\frac{\pi}{4M} (2k+1)(2n+1) \right],$$
$$\Im m\,\{p_k\} = \sum_{n=0}^{2M-1} u_n \sin\left[\frac{\pi}{4M} (2k+1)(2n+1) \right], \qquad k = 0, 1, \ldots, M-1. \tag{7.3}$$

One can note that the cosine and sine transform kernels in (7.2) and (7.3) are symmetric with respect to the time and frequency indices n and k. Substituting $2M - 1 - k$ for k into (7.2) for the cosine and sine transform kernels we have:

$$\begin{aligned} &\cos\left[\frac{\pi}{4M} (4M-1-2k)(2n+1) \right] + i \sin\left[\frac{\pi}{4M} (4M-1-2k)(2n+1) \right] \\ &= -\cos\left[\frac{\pi}{4M} (2k+1)(2n+1) \right] + i \sin\left[\frac{\pi}{4M} (2k+1)(2n+1) \right] \\ &= -\exp\left[-i \frac{\pi}{4M} (2k+1)(2n+1) \right], \end{aligned} \tag{7.4}$$

implying that $\{p_k\}$ has a conjugate symmetry property given by

$$p_{2M-1-k} = -p_k^*, \qquad k = 0, 1, \ldots, M-1, \tag{7.5}$$

where the symbol $*$ denotes complex conjugation. The conjugate symmetry property given by (7.5) indicates that only M complex-valued sub-band coefficients are unique, if we take into account the negative frequencies for $k = M, M+1, \ldots, 2M-1$. Further, for Eq. (7.5) to hold, the data sequence $\{u_n\}$ needs to be real-valued.

7.3.1.2 HQ-SBR QMF Banks in the Decoder

Complex Exponential-Modulated Analysis QMF Bank

The complex exponential-modulated $\frac{M}{2}$-band ($M = 64$) analysis QMF bank in the decoder as a block transform is defined as [18]

$$p_k = 2 \sum_{n=0}^{M-1} u_n \exp\left[i \frac{\pi}{M} \left(k + \frac{1}{2}\right)\left(2n - \frac{1}{2}\right)\right],$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{7.6}$$

where $\{p_k\}$ are complex-valued sub-band coefficients and $\{u_n\}$ is an M-point windowed data sequence. The flowchart of one loop of sub-band filtering procedure in the decoder is shown in [18] (see Fig. 4.41). Equation (7.6) can be similarly extended as

$$p_k = 2 \sum_{n=0}^{M-1} u_n \cos\left[\frac{\pi}{4M}(2k+1)(4n-1)\right]$$
$$+2i \sum_{n=0}^{M-1} u_n \sin\left[\frac{\pi}{4M}(2k+1)(4n-1)\right],$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{7.7}$$

or equivalently, decomposed into the real and the imaginary part, respectively, as

$$\Re e\,\{p_k\} = 2 \sum_{n=0}^{M-1} u_n \cos\left[\frac{\pi}{4M}(2k+1)(4n-1)\right],$$
$$\Im m\,\{p_k\} = 2 \sum_{n=0}^{M-1} u_n \sin\left[\frac{\pi}{4M}(2k+1)(4n-1)\right],$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1. \tag{7.8}$$

Substituting $M-1-k$ for k into (7.7) for the cosine and sine transform kernels we have:

$$\begin{aligned}
&\cos\left[\frac{\pi}{4M}(2M-1-2k)(4n-1)\right]+i\ \sin\left[\frac{\pi}{4M}(2M-1-2k)(4n-1)\right]\\
&=-\sin\left[\frac{\pi}{4M}(2k+1)(4n-1)\right]-i\ \cos\left[\frac{\pi}{4M}(2k+1)(4n-1)\right]\\
&=-i\ \exp\left[-i\ \frac{\pi}{4M}(2k+1)(4n-1)\right], \qquad (7.9)
\end{aligned}$$

demonstrating that $\{p_k\}$ has a conjugate symmetry property given by

$$p_{M-1-k}=-i\ p_k^*, \qquad k=0,1,\ldots,\frac{M}{2}-1, \qquad (7.10)$$

where the symbol $*$ denotes complex conjugation. The conjugate symmetry property (7.10) indicates that only $\frac{M}{2}$ complex-valued sub-band coefficients are unique. Further, based on (7.10), the real part of $\{p_k\}$ is related to the imaginary part in (7.8) by

$$\Re e\{p_k\}=-\Im m\{p_{M-1-k}\}, \quad k=0,1,\ldots,M-1. \qquad (7.11)$$

According to (7.11) it is sufficient to consider either the real or imaginary part of $\{p_k\}$ only, if the index k is extended to $M-1$.

Complex Exponential-Modulated Synthesis QMF Bank

The complex exponential-modulated M-band ($M=64$) synthesis QMF bank in the decoder as a block transform is defined as [18]

$$\begin{aligned}
v_n &= \frac{1}{M}\ \Re e\left\{\sum_{k=0}^{M-1} p_k\ \exp\left[i\ \frac{\pi}{2M}\left(k+\frac{1}{2}\right)(2n+1-4M)\right]\right\},\\
&\quad n=0,1,\ldots,2M-1, \qquad (7.12)
\end{aligned}$$

where $\{p_k\}$ are complex-valued sub-band coefficients and $\{v_n\}$ is a $2M$-point real-valued time domain data sequence. The flowchart of one loop of synthesis filtering procedure in the decoder is shown in [18] (see Fig. 4.42). The real part of complex exponential-modulated synthesis QMF bank defined by (7.12) after performing the complex multiplications under the sum is given by

$$v_n = \frac{1}{M}\sum_{k=0}^{M-1} \Re e\{p_k\}\ \cos\left[\frac{\pi}{4M}(2k+1)(2n+1-4M)\right]$$
$$-\frac{1}{M}\sum_{k=0}^{M-1} \Im m\{p_k\}\ \sin\left[\frac{\pi}{4M}(2k+1)(2n+1-4M)\right],$$
$$n = 0, 1, \ldots, 2M-1. \tag{7.13}$$

Complex Exponential-Modulated Down-Sampled Synthesis QMF Bank

The complex exponential-modulated $\frac{M}{2}$-band ($M = 64$) down-sampled synthesis QMF bank in the decoder as a block transform is defined as [18]

$$v_n = \frac{1}{M}\,\Re e\left\{\sum_{k=0}^{\frac{M}{2}-1} p_k\ \exp\left[i\,\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(2n+\frac{1}{2}-2M\right)\right]\right\},$$
$$n = 0, 1, \ldots, M-1. \tag{7.14}$$

where $\{p_k\}$ are complex-valued sub-band coefficients and $\{v_n\}$ is an M-point real-valued time domain data sequence. The flowchart of one loop of synthesis down-sampled filtering procedure in the decoder is shown in [18] (see Fig. 4.43). The real part of complex exponential-modulated down-sampled synthesis QMF bank defined by (7.14) after performing the complex multiplications under the sum is given by

$$v_n = \frac{1}{M}\sum_{k=0}^{\frac{M}{2}-1} \Re e\{p_k\}\ \cos\left[\frac{\pi}{4M}(2k+1)(4n+1-4M)\right]$$
$$-\frac{1}{M}\sum_{k=0}^{\frac{M}{2}-1} \Im m\{p_k\}\ \sin\left[\frac{\pi}{4M}(2k+1)(4n+1-4M)\right],$$
$$n = 0, 1, \ldots, M-1. \tag{7.15}$$

7.3.1.3 LP-SBR QMF Banks in the Decoder

In the following subsections the definitions and symmetry properties of real-valued cosine-modulated analysis/(down-sampled) synthesis QMF banks used in the standard LP-SBR are presented.

Real-Valued Cosine-Modulated Analysis QMF Bank

The real-valued cosine-modulated $\frac{M}{2}$-band ($M = 64$) analysis QMF bank in the decoder as a block transform is defined as [18]

$$c_k = 2 \sum_{n=0}^{M-1} u_n \cos\left[\frac{\pi}{M}\left(k + \frac{1}{2}\right)\left(2n - \frac{3M}{2}\right)\right],$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{7.16}$$

where $\{c_k\}$ are real-valued sub-band coefficients and $\{u_n\}$ is an M-point windowed data sequence. The flowchart of one loop of sub-band filtering procedure in the decoder is shown in [18] (see Fig. 4.49). Substituting $M - 1 - k$ for k into (7.16) we get

$$c_{M-1-k} = c_k, \qquad k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{7.17}$$

demonstrating that $\{c_k\}$ has an even symmetry property, and hence only $\frac{M}{2}$ real-valued sub-band coefficients are unique if the range of k is extended from $\frac{M}{2} - 1$ to $M - 1$. Equation (7.16) can be written in the following equivalent form:

$$c_k = 2 \sum_{n=0}^{M-1} u_n \cos\left[\frac{\pi}{2M}(2k + 1)\left(2n - \frac{3M}{2}\right)\right],$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1. \tag{7.18}$$

Real-Valued Cosine-Modulated Synthesis QMF Bank

The real-valued cosine-modulated M-band ($M = 64$) synthesis QMF bank in the decoder as a block transform is defined as [18]

$$v_n = \frac{2}{M} \sum_{k=0}^{M-1} c_k \cos\left[\frac{\pi}{2M}\left(k + \frac{1}{2}\right)(2n - M)\right],$$
$$n = 0, 1, \ldots, 2M - 1, \tag{7.19}$$

where $\{c_k\}$ are real-valued sub-band coefficients and $\{v_n\}$ is a $2M$-point time domain data sequence. The flowchart of one loop of synthesis filtering procedure in the decoder is shown in [18] (see Fig. 4.50). Equation (7.19) can be written in the following equivalent form:

$$v_n = \frac{2}{M} \sum_{k=0}^{M-1} c_k \cos\left[\frac{\pi}{4M}(2k+1)(2n-M)\right],$$
$$n = 0, 1, \ldots, 2M-1. \tag{7.20}$$

Real-Valued Cosine-Modulated Down-Sampled Synthesis QMF Bank

The real-valued cosine-modulated $\frac{M}{2}$-band ($M = 64$) down-sampled synthesis QMF bank in the decoder as a block transform is defined as [18]

$$v_n = \frac{2}{M} \sum_{k=0}^{\frac{M}{2}-1} c_k \cos\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(2n-\frac{M}{2}\right)\right],$$
$$n = 0, 1, \ldots, M-1, \tag{7.21}$$

where $\{c_k\}$ are real-valued sub-band coefficients and $\{v_n\}$ is an M-point time domain data sequence. The flowchart of one loop of synthesis down-sampled filtering procedure in the decoder is shown in [18] (see Fig. 4.51). Equation (7.21) can be written in the following equivalent form:

$$v_n = \frac{2}{M} \sum_{k=0}^{\frac{M}{2}-1} c_k \cos\left[\frac{\pi}{2M}(2k+1)\left(2n-\frac{M}{2}\right)\right],$$
$$n = 0, 1, \ldots, M-1. \tag{7.22}$$

7.3.2 *Efficient Implementations of the QMF Banks in the HE-AAC*

In general, each complex-valued or real-valued QMF bank in the ISO/MPEG standards [18, 25] consists of a windowing step followed by a block transform. The windowing step is in itself quite computational demanding as each block of M samples requires a filtering step using 10M coefficients. The filtering step is as complex as one of the block transforms for the real or imaginary part, thus representing one-third of the complexity in a complex-valued QMF and half of the complexity of a real-valued QMF.

In the following efficient implementations of QMF banks as block transforms in the HE-AAC codec are presented. Without loss of generality, the normalization factors from definitions of the QMF banks are omitted in the derivations of fast algorithms.

7.3.2.1 Complex Exponential-Modulated Analysis QMF Bank in the Encoder

Consider the complex exponential-modulated analysis QMF bank in the encoder given by (7.3). Using the symmetry properties of cosine and sine transform kernels, i.e., substituting $2M-1-n$ for n we immediately obtain its efficient implementation defined as

$$\Re e\,\{p_k\} = \sum_{n=0}^{M-1}(u_n - u_{2M-1-n})\ \cos\left[\frac{\pi}{4M}(2k+1)(2n+1)\right],$$

$$\Im m\,\{p_k\} = \sum_{n=0}^{M-1}(u_n + u_{2M-1-n})\ \sin\left[\frac{\pi}{4M}(2k+1)(2n+1)\right],$$

$$k = 0, 1, \ldots, M-1, \tag{7.23}$$

where the real part $\Re e\,\{p_k\}$ is reduced to an unnormalized M-point forward DCT-IV of $\{u_n - u_{2M-1-n}\}$ while the imaginary part $\Im m\,\{p_k\}$ is reduced to an unnormalized M-point forward DST-IV of $\{u_n + u_{2M-1-n}\}$ which may subsequently be (but not necessary) converted to the DCT-IV. The efficient implementation of complex exponential-modulated analysis QMF bank in the encoder requires $2M$ additions and the computation of two M-point forward DCTs-IV.

7.3.2.2 HQ-SBR QMF Banks in the Decoder

The efficient implementations of complex exponential-modulated QMF banks for the HQ-SBR in the HE-AAC decoder proposed in [39, 40] are based on the DCT-IV or FFT. Their derivations here are presented in more general, more transparent, and compact form.

Complex Exponential-Modulated Analysis QMF Bank

We recall that for computation of the complex exponential-modulated analysis QMF bank given by (7.8), based on the property (7.11) it is sufficient to compute only either the real (preferred) or the imaginary part of $\{p_k\}$.

Let us investigate the transform kernel $\cos\left[\frac{\pi}{4M}(2k+1)(4n-1)\right]$ for the real part in (7.8). Substituting $n = \frac{m+1}{2}$ leads to the transform kernel $\cos\left[\frac{\pi}{4M}(2k+1)(2m+1)\right]$ which corresponds to the real part in (7.8). This fact indicates that there exists a mapping between the time indices n and m in the range of $0, 1, \ldots, M-1$. Indeed, comparing both the transform kernels for specific values of n and m as follows:

n	$\cos\left[\frac{\pi}{4M}(2k+1)(4n-1)\right]$	**m**	$\cos\left[\frac{\pi}{4M}(2k+1)(2m+1)\right]$
0	$\cos\left[\frac{\pi}{4M}(2k+1)(-1)\right]$	0	$\cos\left[\frac{\pi}{4M}(2k+1)(1)\right]$
$\frac{M}{2}$	$\cos\left[\frac{\pi}{4M}(2k+1)(2M-1)\right]$	$M-1$	$\cos\left[\frac{\pi}{4M}(2k+1)(2M-1)\right]$
$n+1$	$\cos\left[\frac{\pi}{4M}(2k+1)(4n+3)\right]$ $n=0,1,\ldots,\frac{M}{2}-2$	$2m+1$	$\cos\left[\frac{\pi}{4M}(2k+1)(4m+3)\right]$ $m=0,1,\ldots,\frac{M}{2}-2$
$M-1-n$	$-\cos\left[\frac{\pi}{4M}(2k+1)(4n+5)\right]$ $n=0,1,\ldots,\frac{M}{2}-2$	$2m+2$	$\cos\left[\frac{\pi}{4M}(2k+1)(4m+5)\right]$ $m=0,1,\ldots,\frac{M}{2}-2$

results in the unique one-to-one mapping defined by

$$y_0 = u_0, \qquad y_{M-1} = u_{\frac{M}{2}},$$
$$y_{2n+1} = u_{n+1}, \quad y_{2n+2} = -u_{M-1-n}, \qquad n = 0, 1, \ldots, \frac{M}{2} - 2, \tag{7.24}$$

and the real part of (7.8) can be equivalently written as

$$\Re e\,\{p_k\} = \sum_{n=0}^{M-1} y_n \cos\left[\frac{\pi}{4M}(2k+1)(2n+1)\right],$$
$$k = 0, 1, \ldots, M-1. \tag{7.25}$$

Using the mapping (7.24), the real part of the complex exponential-modulated analysis QMF bank given by (7.8) is converted to an unnormalized M-point forward DCT-IV of $\{y_n\}$. Hence, its efficient implementation requires the computation of only one M-point forward DCT-IV. The imaginary part $\Im m\,\{p_k\}$ is easily constructed from (7.11).

Complex Exponential-Modulated Synthesis QMF Bank

Consider the complex exponential-modulated synthesis QMF bank given by (7.13). Using trigonometric identities for the cosine and sine transform kernels we can immediately eliminate time shift factors $4M$, and Eq. (7.13) can be written in the equivalent form as

$$v_n = -\sum_{k=0}^{M-1} \Re e\{p_k\} \cos\left[\frac{\pi}{4M}(2k+1)(2n+1)\right]$$
$$+ \sum_{k=0}^{M-1} \Im m\{p_k\} \sin\left[\frac{\pi}{4M}(2k+1)(2n+1)\right],$$
$$n = 0, 1, \ldots, 2M-1. \tag{7.26}$$

The first sum in (7.26) is an unnormalized M-point inverse DCT-IV, while the second sum is the corresponding unnormalized M-point inverse DST-IV which may subsequently be (but not necessary) converted to the DCT-IV. Further, substituting $2M-1-n$ for n into (7.26) we get

$$\begin{aligned} v_{2M-1-n} &= \sum_{k=0}^{M-1} \Re e\{p_k\} \cos\left[\frac{\pi}{4M}(2k+1)(2n+1)\right] \\ &\quad + \sum_{k=0}^{M-1} \Im m\{p_k\} \sin\left[\frac{\pi}{4M}(2k+1)(2n+1)\right], \\ &\quad n = 0, 1, \ldots, M-1. \end{aligned} \tag{7.27}$$

Combining (7.26) and (7.27) and using the definitions of the inverse DCT-IV and DST-IV, respectively, given by (C.10) and (C.12) from Appendix C.2, the final time domain sequence $\{v_n\}$ is obtained as

$$\begin{aligned} v_n &= -x_n^{(c)} + x_n^{(s)}, \\ v_{2M-1-n} &= x_n^{(c)} + x_n^{(s)}, \\ &\quad n = 0, 1, \ldots, M-1. \end{aligned} \tag{7.28}$$

The efficient implementation of the complex exponential-modulated synthesis QMF bank requires $2M$ additions and the computation of two M-point inverse DCTs-IV.

Complex Exponential-Modulated Down-Sampled Synthesis QMF Bank

There exist at least three methods how to compute the complex exponential-modulated down-sampled synthesis QMF bank in the decoder given by (7.15). The first method is based on a direct relation between Eqs. (7.13) and (7.15). Specifically, substituting $n = \frac{m}{2}$ into (7.15) and zero-padding the real and imaginary parts of $\{p_k\}$ for $k = \frac{M}{2}, \frac{M}{2}+1, \ldots, M-1$ we have

$$\begin{aligned} v_{\frac{m}{2}} &= \sum_{k=0}^{M-1} \Re e\{p_k\} \cos\left[\frac{\pi}{4M}(2k+1)(2m+1-4M)\right] \\ &\quad - \sum_{k=0}^{M-1} \Im m\{p_k\} \sin\left[\frac{\pi}{4M}(2k+1)(2m+1-4M)\right], \\ &\quad m = 2n, \qquad n = 0, 1, \ldots, M-1. \end{aligned} \tag{7.29}$$

Equation (7.29) corresponds to the complex exponential-modulated synthesis QMF bank given by (7.13). Therefore, its efficient implementation defined

by (7.26)–(7.28) may simply be re-used for the down-sampled synthesis QMF bank. The time domain sequence of the down-sampled synthesis QMF bank is obtained from $\{v_n\}$ by taking the even-indexed samples, i.e., $\{v_{2n}\}$, $n = 0, 1, \ldots, M-1$.

The second method is similarly based on the existing efficient implementation of complex exponential-modulated synthesis QMF bank. Using trigonometric identities for the cosine and sine transform kernels in (7.15), we can immediately eliminate the shift factors $4M$, and Eq. (7.15) can be written as

$$\begin{aligned} v_n = &- \sum_{k=0}^{\frac{M}{2}-1} \Re e\{p_k\} \cos\left[\frac{\pi}{4M}(2k+1)(4n+1)\right] \\ &+ \sum_{k=0}^{\frac{M}{2}-1} \Im m\{p_k\} \sin\left[\frac{\pi}{4M}(2k+1)(4n+1)\right], \\ &n = 0, 1, \ldots, M-1. \end{aligned} \tag{7.30}$$

Further, substituting $M-1-n$ for n into (7.30) we get

$$\begin{aligned} v_{M-1-n} = &\sum_{k=0}^{\frac{M}{2}-1} \Re e\{p_k\} \cos\left[\frac{\pi}{4M}(2k+1)(4n+3)\right] \\ &+ \sum_{k=0}^{\frac{M}{2}-1} \Im m\{p_k\} \sin\left[\frac{\pi}{4M}(2k+1)(4n+3)\right], \\ &n = 0, 1, \ldots, \frac{M}{2}-1. \end{aligned} \tag{7.31}$$

Now, if the real and imaginary parts of $\{p_k\}$ are zero-padded for $k = \frac{M}{2}, \frac{M}{2}+1, \ldots, M-1$, and we use the definitions of inverse DCT-IV and DST-IV, respectively, given by (C.10) and (C.12) from Appendix C.2, then Eqs. (7.30) and (7.31) can be written in the following form:

$$\begin{aligned} v_n &= -x_{2n}^{(c)} + x_{2n}^{(s)}, \\ v_{M-1-n} &= x_{2n+1}^{(c)} + x_{2n+1}^{(s)}, \\ &n = 0, 1, \ldots, \frac{M}{2}-1. \end{aligned} \tag{7.32}$$

Although both discussed methods above can simply reuse the existing efficient implementation of the complex exponential-modulated synthesis QMF bank given by (7.13), they still require the computation of two unnormalized M-point inverse DCTs-IV.

However, the third, more elegant method, requires the computation of only one unnormalized M-point inverse DCT-IV as follows. Consider Eqs. (7.30) and (7.31). If we substitute $M-1-k$ for $k=\frac{M}{2},\frac{M}{2}+1,\ldots,M-1$, into the second sums of (7.30) and (7.31), then they can be written as

$$v_n = -\sum_{k=0}^{\frac{M}{2}-1} \Re e\{p_k\}\ \cos\left[\frac{\pi}{4M}(2k+1)(4n+1)\right] + \sum_{k=\frac{M}{2}}^{M-1} \Im m\{p_{M-1-k}\}\ \cos\left[\frac{\pi}{4M}(2k+1)(4n+1)\right], \tag{7.33}$$

$$v_{M-1-n} = \sum_{k=0}^{\frac{M}{2}-1} \Re e\{p_k\}\ \cos\left[\frac{\pi}{4M}(2k+1)(4n+3)\right] + \sum_{k=\frac{M}{2}}^{M-1} \Im m\{p_{M-1-k}\}\ \cos\left[\frac{\pi}{4M}(2k+1)(4n+3)\right], \quad n=0,1,\ldots,\frac{M}{2}-1. \tag{7.34}$$

Introducing new data sequences $\{y_k\}$ and $\{z_k\}$ and packing the real and imaginary parts of $\{p_k\}$ in (7.33) and (7.34), respectively, as

$$y_k = -\Re e\{p_k\}, \qquad y_{\frac{M}{2}+k} = \Im m\{p_{\frac{M}{2}-1-k}\}, \qquad k=0,1,\ldots,\frac{M}{2}-1, \tag{7.35}$$

and

$$z_k = \Re e\{p_k\}, \qquad z_{\frac{M}{2}+k} = -\Im m\{p_{\frac{M}{2}-1-k}\}, \qquad k=0,1,\ldots,\frac{M}{2}-1, \tag{7.36}$$

Eqs. (7.33) and (7.34) are reduced to unnormalized M-point inverse DCTs-IV of $\{y_k\}$ and $\{z_k\}$, respectively, as

$$v_n = \sum_{k=0}^{M-1} y_k\ \cos\left[\frac{\pi}{4M}(2k+1)(4n+1)\right] = x_{2n}^{(c)}, \tag{7.37}$$

$$v_{M-1-n} = \sum_{k=0}^{M-1} z_k\ \cos\left[\frac{\pi}{4M}(2k+1)(4n+3)\right], \quad n=0,1,\ldots,\frac{M}{2}-1. \tag{7.38}$$

But, one can easily see that the data sequence $\{y_k\}$ is closely related to $\{z_k\}$ by

$$z_k = -\,y_k, \qquad k = 0, 1, \ldots, M-1. \tag{7.39}$$

Consequently, only one M-point inverse DCT-IV is sufficient to be computed. Indeed, using the definition of the inverse DCT-IV given by (C.10) from Appendix C.2, the final time domain sequence is obtained as

$$v_n = x^{(c)}_{2n}, \qquad v_{M-1-n} = -\,x^{(c)}_{2n+1}, \qquad k = 0, 1, \ldots, \frac{M}{2} - 1. \tag{7.40}$$

7.3.2.3 LP-SBR QMF Banks in the Decoder

The efficient implementations of the real-valued cosine-modulated QMF banks for the LP-SBR in HE-AAC decoder have been proposed in [32, 38]. They are based on the DCT-II or its inverse, the DCT-III. In principle, these derivations closely follow a method for the derivation of efficient implementations of analysis and synthesis pseudo-QMF banks used in the MPEG-1/2 audio coding standard [49, 50].

Real-Valued Cosine-Modulated Analysis QMF Bank

Consider the real-valued cosine-modulated analysis QMF bank given by (7.18). In order to eliminate the time shift factor $\frac{3M}{2}$ in (7.18), let us apply the following permutation to the input data sequence $\{u_n\}$ defined as

$$y_n = \begin{cases} u_{\frac{3M}{4}+n}, & n = 0, 1, \ldots, \frac{M}{4} - 1, \\ \\ -\,u_{n-\frac{M}{4}}, & n = \frac{M}{4}, \frac{M}{4} + 1, \ldots, M-1, \end{cases} \tag{7.41}$$

and we have

$$\begin{aligned} c_k &= \sum_{n=0}^{\frac{M}{4}-1} u_{\frac{3M}{4}+n} \cos\left[\frac{\pi(2k+1)n}{M}\right] + \sum_{n=\frac{M}{4}}^{M-1} -u_{n-\frac{M}{4}} \cos\left[\frac{\pi(2k+1)n)}{M}\right] \\ &= \sum_{n=0}^{M-1} y_n \cos\left[\frac{\pi(2k+1)n}{M}\right], \qquad k = 0, 1, \ldots, \frac{M}{2} - 1. \end{aligned} \tag{7.42}$$

Using the symmetry property of cosine transform kernel, i.e., substituting $M-n$ for $n = 1, 2, \ldots, \frac{M}{2} - 1$, into the sum on the right-hand side of (7.42), we get

$$c_k = y_0 + \sum_{n=1}^{\frac{M}{2}-1} (y_n - y_{M-n}) \cos\left[\frac{\pi(2k+1)n}{2(M/2)}\right],$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1. \tag{7.43}$$

The cosine transform kernel in Eq. (7.43) is recognized as an unnormalized $\frac{M}{2}$-point DCT-III of $\{y_n - y_{M-n}\}$. Finally, combining Eqs. (7.41) and (7.43) we obtain an efficient implementation of the real-valued cosine-modulated analysis QMF bank defined as

$$c_k = \sum_{n=0}^{\frac{M}{2}-1} y_n \cos\left[\frac{\pi(2k+1)n}{2(M/2)}\right], \qquad k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{7.44}$$

where

$$y_n = \begin{cases} u_{\frac{3M}{4}}, & n = 0, \\ u_{\frac{3M}{4}+n} + u_{\frac{3M}{4}-n}, & n = 1, 2, \ldots, \frac{M}{4} - 1, \\ -u_{n-\frac{M}{4}} + u_{\frac{3M}{4}-n}, & n = \frac{M}{4}, \frac{M}{4} + 1, \ldots, \frac{M}{2} - 1. \end{cases} \tag{7.45}$$

We note that when M is a power of two, then it is always divisible by 4. The efficient implementation of the real-valued cosine-modulated analysis QMF bank requires $\frac{M}{2} - 1$ additions and the computation of one $\frac{M}{2}$-point DCT-III.

Real-Valued Cosine-Modulated Synthesis QMF Bank

Now, consider the real-valued cosine-modulated synthesis QMF bank given by (7.20). Similarly, in order to eliminate the time shift factor M in (7.20), subsequently substituting $\frac{M}{2} + n$, $\frac{M}{2}$, $\frac{M}{2} - n$, $\frac{3M}{2} + n$, $\frac{3M}{2}$ and $\frac{3M}{2} - n$ into (7.20), and using relations (C.3) and (C.4) from Appendix C.1, we have

$$v_{\frac{M}{2}+n} = \sum_{k=0}^{M-1} c_k \cos\left[\frac{\pi(2k+1)n}{2M}\right] = x_n, \qquad n = 1, 2, \ldots, \frac{M}{2},$$
$$v_{\frac{M}{2}} = \sum_{k=0}^{M-1} c_k = x_0,$$
$$v_{\frac{M}{2}-n} = \sum_{k=0}^{M-1} c_k \cos\left[\frac{\pi(2k+1)n}{2M}\right] = x_n, \qquad n = 1, 2, \ldots, \frac{M}{2},$$

$$
\begin{aligned}
v_{\frac{3M}{2}+n} &= -\sum_{k=0}^{M-1} c_k\,(-1)^k \sin\left[\frac{\pi(2k+1)n}{2M}\right] = -x_{M-n}, \qquad n = 1, 2, \ldots, \frac{M}{2}-1,\\
v_{\frac{3M}{2}} &= 0,\\
v_{\frac{3M}{2}-n} &= \sum_{k=0}^{M-1} c_k\,(-1)^k \sin\left[\frac{\pi(2k+1)n}{2M}\right] = x_{M-n}, \qquad n = 1, 2, \ldots, \frac{M}{2}-1,
\end{aligned}
\tag{7.46}
$$

where

$$
x_n = \sum_{k=0}^{M-1} c_k \cos\left[\frac{\pi(2k+1)n}{2M}\right], \qquad n = 0, 1, \ldots, M-1. \tag{7.47}
$$

Equation (7.47) is recognized as an unnormalized M-point DCT-II of $\{c_k\}$. Using (7.46), the time domain aliased data sequence $\{v_n\}$ is recovered from $\{x_n\}$ as

$$
\begin{aligned}
v_{\frac{M}{2}} &= x_0, & v_{\frac{3M}{2}} &= 0,\\
v_{\frac{M}{2}-n} &= x_n, & v_{\frac{M}{2}+n} &= v_{\frac{M}{2}-n}, & n &= 1, 2, \ldots, \frac{M}{2},\\
v_{\frac{3M}{2}-n} &= x_{M-n}, & v_{\frac{3M}{2}+n} &= -v_{\frac{3M}{2}-n}, & n &= 1, 2, \ldots, \frac{M}{2}-1.
\end{aligned}
\tag{7.48}
$$

Equations (7.47) and (7.48) define the efficient implementation of the real-valued cosine-modulated synthesis QMF bank requiring the computation of one M-point DCT-II.

Real-Valued Cosine-Modulated Down-Sampled Synthesis QMF Bank

Comparing Eqs. (7.20) and (7.22) it can be easily seen that by substituting $\frac{M}{2}$ for M into (7.20) we get the real-valued cosine-modulated down-sampled synthesis QMF bank given by (7.22). Therefore, the efficient implementation of the real-valued cosine-modulated synthesis QMF bank defined by (7.47) and (7.48) may also be used for the down-sampled synthesis QMF bank by taking $\frac{M}{2}$ instead of M (M is divisible by 4) as follows:

$$
\begin{aligned}
v_{\frac{M}{4}} &= x_0, & v_{\frac{3M}{4}} &= 0,\\
v_{\frac{M}{4}-n} &= x_n, & v_{\frac{M}{4}+n} &= v_{\frac{M}{4}-n}, & n &= 1, 2, \ldots, \frac{M}{4},\\
v_{\frac{3M}{4}-n} &= x_{\frac{M}{4}-n}, & v_{\frac{3M}{4}+n} &= -v_{\frac{3M}{4}-n}, & n &= 1, 2, \ldots, \frac{M}{4}-1.
\end{aligned}
\tag{7.49}
$$

where

$$x_n = \sum_{k=0}^{\frac{M}{2}-1} c_k \cos\left[\frac{\pi(2k+1)n}{2(M/2)}\right], \qquad n = 0, 1, \ldots, \frac{M}{2} - 1. \tag{7.50}$$

It requires the computation of one unnormalized $\frac{M}{2}$-point DCT-II.

7.3.3 LD-SBR QMF Banks: Definitions and Symmetry Properties

In the following subsections the definitions and symmetry properties of complex exponential-modulated low delay (LD) analysis/(down-sampled) synthesis QMF banks used in the HQ-LD-SBR are presented.

7.3.3.1 Complex Exponential-Modulated LD Analysis QMF Bank in the Encoder

The complex exponential-modulated M-band ($M = 64$) LD analysis QMF bank, or the so-called CLDFB (complex low delay filter bank) analysis filter bank in the encoder as a block transform is defined as [25]

$$p_k = \sum_{n=0}^{2M-1} u_n \exp\left[i \frac{\pi}{2M}\left(k + \frac{1}{2}\right)(2n + 1 - 3M)\right], \quad k = 0, 1, \ldots, M-1. \tag{7.51}$$

The sub-band filtering by the analysis CLDFB filter bank is basically similar to the complex exponential-modulated analysis QMF bank used in the standard SBR encoder defined by (7.1). Only the windowing function and the modulation differ [25]. The flowchart of one loop of sub-band filtering procedure is shown in [18] (see Fig. 4.B.16). Equation (7.51) can be extended as

$$\begin{aligned} p_k = & \sum_{n=0}^{2M-1} u_n \cos\left[\frac{\pi}{4M}(2k+1)(2n+1-3M)\right] \\ & + i \sum_{n=0}^{2M-1} u_n \sin\left[\frac{\pi}{4M}(2k+1)(2n+1-3M)\right], \\ & k = 0, 1, \ldots, M-1, \end{aligned} \tag{7.52}$$

or equivalently, decomposed into the real and imaginary part, respectively, as

$$\Re e\,\{p_k\} = \sum_{n=0}^{2M-1} u_n \cos\left[\frac{\pi}{4M}(2k+1)(2n+1-3M)\right],$$

$$\Im m\,\{p_k\} = \sum_{n=0}^{2M-1} u_n \sin\left[\frac{\pi}{4M}(2k+1)(2n+1-3M)\right],$$

$$k = 0, 1, \ldots, M-1. \tag{7.53}$$

Substituting $2M-1-k$ for k into (7.52) we have:

$$\cos\left[\frac{\pi}{4M}(4M-1-2k)(2n+1-3M)\right] + i \sin\left[\frac{\pi}{4M}(4M-1-2k)(2n+1-3M)\right]$$

$$= -\cos\left[\frac{\pi}{4M}(2k+1)(2n+1-3M)\right] + i \sin\left[\frac{\pi}{4M}(2k+1)(2n+1-3M)\right]$$

$$= -\exp\left[-i\frac{\pi}{4M}(2k+1)(2n+1-3M)\right], \tag{7.54}$$

implying that $\{p_k\}$ has a conjugate symmetry property given by

$$p_{2M-1-k} = -p_k^*, \qquad k = 0, 1, \ldots, M-1, \tag{7.55}$$

where the symbol $*$ denotes complex conjugation. The conjugate symmetry property given by (7.55) indicates that only M complex-valued sub-band coefficients are unique if the range of k is extended from $\frac{M}{2}$ to $M-1$.

Note 1: The LD-SBR encoder can use either the complex exponential-modulated analysis QMF bank defined by (7.1) or the CLDFB defined by (7.51). If the CLDFB is used, the synchronization between the core coder and SBR is equal to the HE-AAC case. If the complex exponential-modulated analysis QMF bank defined by (7.1) is used, then the SBR must operate on additionally 256 ($4M$) delayed audio samples with respect to the core encoder [25].

7.3.3.2 HQ-LD-SBR QMF Banks in the Decoder

The sub-band as well as synthesis filtering procedures described for the standard HQ-SBR in the decoder (the flowcharts in Figs. 4.41, 4.42, and 4.43 in [18]) are basically similar for the HQ-LD-SBR. Only the windowing function and the modulation differ.

Complex Exponential-Modulated LD Analysis QMF Bank

The complex exponential-modulated $\frac{M}{2}$-band ($M = 64$) LD analysis QMF bank in the decoder as a block transform is defined as [25]

$$p_k = 2 \sum_{n=0}^{M-1} u_n \exp\left[i \frac{\pi}{M} \left(k + \frac{1}{2} \right) \left(2n + 1 - \frac{3M}{2} \right) \right],$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1. \tag{7.56}$$

Equation (7.56) can be extended as

$$p_k = 2 \sum_{n=0}^{M-1} u_n \cos\left[\frac{\pi}{2M} (2k + 1) \left(2n + 1 - \frac{3M}{2} \right) \right]$$
$$+2i \sum_{n=0}^{M-1} u_n \sin\left[\frac{\pi}{2M} (2k + 1) \left(2n + 1 - \frac{3M}{2} \right) \right],$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{7.57}$$

or equivalently, decomposed into the real and imaginary part, respectively, as

$$\Re e \{p_k\} = 2 \sum_{n=0}^{M-1} u_n \cos\left[\frac{\pi}{2M} (2k + 1) \left(2n + 1 - \frac{3M}{2} \right) \right],$$
$$\Im m \{p_k\} = 2 \sum_{n=0}^{M-1} u_n \sin\left[\frac{\pi}{2M} (2k + 1) \left(2n + 1 - \frac{3M}{2} \right) \right],$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1. \tag{7.58}$$

Comparing Eqs. (7.53) and (7.58) it can be seen that they are closely related. Specifically, substituting $\frac{M}{2}$ for M into (7.53) we get (7.58), i.e., the complex exponential-modulated LD analysis QMF bank in the decoder defined by (7.56) is the complex exponential-modulated LD analysis QMF bank in the encoder defined by (7.51) for the size $\frac{M}{2}$.

Complex Exponential-Modulated LD Synthesis QMF Bank

The complex exponential-modulated M-band ($M = 64$) LD synthesis QMF bank in the decoder as a block transform is defined as [25]

$$v_n = \frac{1}{M}\,\Re e\left\{\sum_{k=0}^{M-1} p_k\,\exp\left[i\,\frac{\pi}{2M}\left(k+\frac{1}{2}\right)(2n+1-M)\right]\right\},$$
$$n = 0, 1, \ldots, 2M-1. \tag{7.59}$$

The real part of under the sum in (7.59) is given by

$$v_n = \frac{1}{M}\sum_{k=0}^{M-1}\Re e\{p_k\}\,\cos\left[\frac{\pi}{4M}(2k+1)(2n+1-M)\right]$$
$$-\frac{1}{M}\sum_{k=0}^{M-1}\Im m\{p_k\}\,\sin\left[\frac{\pi}{4M}(2k+1)(2n+1-M)\right],$$
$$n = 0, 1, \ldots, 2M-1. \tag{7.60}$$

Complex Exponential-Modulated LD Down-Sampled Synthesis QMF Bank

The complex exponential-modulated $\frac{M}{2}$-band ($M = 64$) LD down-sampled synthesis QMF bank in the decoder as a block transform is defined as [25]

$$v_n = \frac{1}{M}\,\Re e\left\{\sum_{k=0}^{\frac{M}{2}-1} p_k\,\exp\left[i\,\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(2n+1-\frac{M}{2}\right)\right]\right\},$$
$$n = 0, 1, \ldots, M-1. \tag{7.61}$$

The real part under sum in (7.61) is given by

$$v_n = \frac{1}{M}\sum_{k=0}^{\frac{M}{2}-1}\Re e\{p_k\}\,\cos\left[\frac{\pi}{2M}(2k+1)\left(2n+1-\frac{M}{2}\right)\right]$$
$$-\frac{1}{M}\sum_{k=0}^{\frac{M}{2}-1}\Im m\{p_k\}\,\sin\left[\frac{\pi}{2M}(2k+1)\left(2n+1-\frac{M}{2}\right)\right],$$
$$n = 0, 1, \ldots, M-1. \tag{7.62}$$

Comparing Eqs. (7.60) and (7.62) it can be seen that they are closely related. Indeed, substituting $\frac{M}{2}$ for M into (7.60) excluding the normalization factor we get (7.62), i.e., the complex exponential-modulated LD down-sampled synthesis QMF bank defined by (7.61) is the complex exponential-modulated LD synthesis QMF bank defined by (7.59) for the size $\frac{M}{2}$.

7.3.3.3 LP-LD-SBR QMF Banks in the Decoder

Similarly, the sub-band as well as synthesis filtering procedures described for the standard LP-SBR in the core decoder (the flowcharts in Figs. 4.49, 4.50, and 4.51 in [18]) are basically similar for the LP-LD-SBR. Only the windowing function and the modulation differ.

Real-Valued Cosine-Modulated LD Analysis QMF Bank

The real-valued cosine-modulated $\frac{M}{2}$-band ($M = 64$) LD analysis QMF bank in the decoder as a block transform is defined as [25]

$$c_k = 2 \sum_{n=0}^{M-1} u_n \cos\left[\frac{\pi}{M}\left(k + \frac{1}{2}\right)\left(2n + 1 - \frac{3M}{2}\right)\right],$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1. \tag{7.63}$$

Substituting $M - 1 - k$ for k into (7.63) we get

$$c_{M-1-k} = -c_k, \qquad k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{7.64}$$

demonstrating that $\{c_k\}$ has an even anti-symmetry property, and hence only $\frac{M}{2}$ real-valued sub-band coefficients are unique if the range of k is extended from $\frac{M}{2}$ to $M - 1$. Equation (7.63) can be written in the following equivalent form:

$$c_k = 2 \sum_{n=0}^{M-1} u_n \cos\left[\frac{\pi}{2M}(2k + 1)\left(2n + 1 - \frac{3M}{2}\right)\right],$$
$$k = 0, 1, \ldots, \frac{M}{2} - 1. \tag{7.65}$$

Comparing Eqs. (7.65) and (7.58) one can see that Eq. (7.65) corresponds to the real part of the complex exponential-modulated LD analysis QMF filter bank defined by (7.56).

Real-Valued Cosine-Modulated LD Synthesis QMF Bank

The real-valued cosine-modulated M-band ($M = 64$) LD synthesis QMF bank in the decoder as a block transform is defined as [25]

$$v_n = \frac{2}{M} \sum_{k=0}^{M-1} c_k \cos\left[\frac{\pi}{2M}\left(k+\frac{1}{2}\right)(2n+1-M)\right], \quad n = 0, 1, \ldots, 2M-1. \tag{7.66}$$

Equation (7.66) can be written in the following equivalent form:

$$v_n = \frac{2}{M} \sum_{k=0}^{M-1} c_k \cos\left[\frac{\pi}{4M}(2k+1)(2n+1-M)\right], \quad n = 0, 1, \ldots, 2M-1. \tag{7.67}$$

Comparing Eqs. (7.67) and (7.60) reveals that Eq. (7.67) corresponds to the first sum on the right-hand side of (7.60) which defines the real part complex exponential-modulated LD synthesis QMF filter bank (7.59).

Real-Valued Cosine-Modulated LD Down-Sampled Synthesis QMF Bank

The real-valued cosine-modulated $\frac{M}{2}$-band ($M = 64$) LD down-sampled synthesis QMF bank in the decoder as a block transform is defined as [25]

$$v_n = \frac{2}{M} \sum_{k=0}^{\frac{M}{2}-1} c_k \cos\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(2n+1-\frac{M}{2}\right)\right], \quad n = 0, 1, \ldots, M-1. \tag{7.68}$$

Equation (7.68) can be written in the following equivalent form:

$$v_n = \frac{2}{M} \sum_{k=0}^{\frac{M}{2}-1} c_k \cos\left[\frac{\pi}{2M}(2k+1)\left(2n+1-\frac{M}{2}\right)\right], \quad n = 0, 1, \ldots, M-1. \tag{7.69}$$

Similarly, comparing Eqs. (7.69) and (7.62) reveals that Eq. (7.69) corresponds to the first sum on the right-hand side of (7.62) which is the complex exponential-modulated LD synthesis QMF filter bank (7.61). Moreover, comparing Eqs. (7.69) and (7.67) it can be seen that they are closely related. Indeed, substituting $\frac{M}{2}$ for M into (7.67) excluding the normalization factor we get (7.69), i.e., the real-valued cosine-modulated LD down-sampled synthesis QMF bank defined by (7.68) is the real-valued cosine-modulated LD synthesis QMF bank defined by (7.66) for the size $\frac{M}{2}$.

Note 2: The SBR technology in the MPEG-4 HE-AAC and AAC-ELD codecs use similar QMF banks. Perhaps the most important distinction between standard SBR

and LD-SBR is that the time shift factors in the transform kernels are even numbers in HE-AAC, and odd numbers in AAC-ELD [33]. This even/odd difference comes from the fact that the low pass prototype filter used in HE-AAC is odd symmetric while the filter used in AAC-ELD is even symmetric.

7.3.4 *Efficient Implementations of the LD QMF Banks in the AAC-ELD*

In the following efficient implementations of LD QMF banks as block transforms in the AAC-ELD codec are presented. Again, without loss of generality, the normalization factors from the definitions of the LD QMF banks are omitted in the derivations of fast algorithms.

7.3.4.1 Complex Exponential-Modulated LD Analysis QMF Bank in the Encoder

Consider the complex exponential-modulated LD analysis QMF bank, or equivalently, the CLDFB analysis filter bank given by (7.53). Obviously, in order to eliminate the time shift factors $3M$ in (7.53), applying the following permutation to the input data sequence $\{u_n\}$ as

$$y_n = \begin{cases} u_{\frac{3M}{2}+n}, & n = 0, 1, \ldots, \frac{M}{2} - 1, \\ -u_{n-\frac{M}{2}}, & n = \frac{M}{2}, \frac{M}{2} + 1, \ldots, 2M - 1, \end{cases} \tag{7.70}$$

the real and imaginary part of the CLDFB analysis filter bank can be, respectively, written as

$$\begin{aligned} \Re e\,\{p_k\} &= \sum_{n=0}^{2M-1} y_n \cos\left[\frac{\pi}{4M}(2k+1)(2n+1)\right], \\ \Im m\,\{p_k\} &= \sum_{n=0}^{2M-1} y_n \sin\left[\frac{\pi}{4M}(2k+1)(2n+1)\right], \qquad k = 0, 1, \ldots, M-1. \end{aligned} \tag{7.71}$$

Subsequently, using the symmetry property of cosine and sine transform kernels, i.e., substituting $2M - 1 - n$ for n into both sums of (7.71), we get

$$\Re e\ \{p_k\} = \sum_{n=0}^{M-1} (y_n - y_{2M-1-n})\ \cos\left[\frac{\pi}{4M}(2k+1)(2n+1)\right],$$
$$\Im m\ \{p_k\} = \sum_{n=0}^{M-1} (y_n + y_{2M-1-n})\ \sin\left[\frac{\pi}{4M}(2k+1)(2n+1)\right],\ k = 0, 1, \ldots, M-1. \tag{7.72}$$

In Eq. (7.72) the real part $\Re e\ \{p_k\}$ is recognized as an unnormalized M-point forward DCT-IV of $\{y_n - y_{2M-1-n}\}$ while the imaginary part $\Im m\ \{p_k\}$ as an unnormalized M-point forward DST-IV of $\{y_n + y_{2M-1-n}\}$ which may subsequently be (but not necessary) converted to the forward DCT-IV. Combining (7.70) with expressions $\{y_n - y_{2M-1-n}\}$ and $\{y_n + y_{2M-1-n}\}$ leads, respectively, to the following mappings

$$y_n - y_{2M-1-n} = \begin{cases} u_{\frac{3M}{2}+n} + u_{\frac{3M}{2}-1-n}, & n = 0, 1, \ldots, \frac{M}{2}-1, \\ \\ -u_{n-\frac{M}{2}} + u_{\frac{3M}{2}-1-n}, & n = \frac{M}{2}, \frac{M}{2}+1, \ldots, M-1, \end{cases} \tag{7.73}$$

and

$$y_n + y_{2M-1-n} = \begin{cases} u_{\frac{3M}{2}+n} - u_{\frac{3M}{2}-1-n}, & n = 0, 1, \ldots, \frac{M}{2}-1, \\ \\ -u_{n-\frac{M}{2}} - u_{\frac{3M}{2}-1-n}, & n = \frac{M}{2}, \frac{M}{2}+1, \ldots, M-1. \end{cases} \tag{7.74}$$

The efficient implementation of CLDFB analysis QMF bank in the encoder requires $2M$ additions and the computation of two M-point unnormalized forward DCTs-IV.

7.3.4.2 HQ-LD-SBR QMF Banks in the Decoder

The efficient implementations of complex exponential-modulated LD QMF banks for the HQ-LD-SBR as well as the efficient implementations of the real-valued cosine-modulated LD QMF banks for the LP-LD-SBR in AAC-ELD decoder have been proposed in [33, 34]. They are based on the DCT-IV or FFT.

Complex Exponential-Modulated LD Analysis QMF Bank

We recall that the complex exponential-modulated LD analysis QMF bank given by (7.58) is actually the CLDFB analysis filter bank in the encoder given by (7.53), but for the size $\frac{M}{2}$. Therefore, the efficient implementation of the CLDFB analysis filter bank defined by (7.72)–(7.74) can simply be reused for the complex exponential-modulated LD analysis QMF bank by taking $\frac{M}{2}$ instead of M (M is divisible by 4) as follows:

$$\Re e\ \{p_k\} = \sum_{n=0}^{\frac{M}{2}-1} (y_n - y_{M-1-n})\ \cos\left[\frac{\pi}{4(M/2)}(2k+1)(2n+1)\right],$$

$$\Im m\ \{p_k\} = \sum_{n=0}^{\frac{M}{2}-1} (y_n + y_{M-1-n})\ \sin\left[\frac{\pi}{4(M/2)}(2k+1)(2n+1)\right],$$

$$k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{7.75}$$

where

$$y_n - y_{M-1-n} = \begin{cases} u_{\frac{3M}{4}+n} + u_{\frac{3M}{4}-1-n}, & n = 0, 1, \ldots, \frac{M}{4} - 1, \\ -u_{n-\frac{M}{4}} + u_{\frac{3M}{4}-1-n}, & n = \frac{M}{4}, \frac{M}{4}+1, \ldots, \frac{M}{2} - 1, \end{cases} \tag{7.76}$$

and

$$y_n + y_{M-1-n} = \begin{cases} u_{\frac{3M}{4}+n} - u_{\frac{3M}{4}-1-n}, & n = 0, 1, \ldots, \frac{M}{4} - 1, \\ -u_{n-\frac{M}{4}} - u_{\frac{3M}{4}-1-n}, & n = \frac{M}{4}, \frac{M}{4}+1, \ldots, \frac{M}{2} - 1. \end{cases} \tag{7.77}$$

This requires M additions and the computation of two $\frac{M}{2}$-point unnormalized forward DCTs-IV.

Complex Exponential-Modulated LD Synthesis QMF Bank

Consider the real part of the complex exponential-modulated LD synthesis QMF bank given by (7.60) in the following equivalent form:

$$v_n = y_n^{(1)} - y_n^{(2)}, \qquad n = 0, 1, \ldots, 2M - 1, \tag{7.78}$$

where

$$y_n^{(1)} = \sum_{k=0}^{M-1} \Re e\{p_k\}\ \cos\left[\frac{\pi}{4M}(2k+1)(2n+1-M)\right],$$

$$y_n^{(2)} = \sum_{k=0}^{M-1} \Im m\{p_k\}\ \sin\left[\frac{\pi}{4M}(2k+1)(2n+1-M)\right]. \tag{7.79}$$

In order to eliminate the time shift factors M in (7.79), subsequently substituting $\frac{M}{2}+n$, $\frac{M}{2}-1-n$, $\frac{3M}{2}+n$, and $\frac{3M}{2}-1-n$ for $n = 0, 1, \ldots, \frac{M}{2}-1$ into the first sum of (7.79), and using the relation (C.16) from Appendix C.2 we, respectively, have

$$
\begin{aligned}
y^{(1)}_{\frac{M}{2}+n} &= \sum_{k=0}^{M-1} \Re e\{p_k\} \cos\left[\frac{\pi}{4M}(2k+1)(2n+1)\right] = x^{(c)}_n, \\
y^{(1)}_{\frac{M}{2}-1-n} &= \sum_{k=0}^{M-1} \Re e\{p_k\} \cos\left[\frac{\pi}{4M}(2k+1)(2n+1)\right] = x^{(c)}_n, \\
y^{(1)}_{\frac{3M}{2}+n} &= -\sum_{k=0}^{M-1} \Re e\{p_k\} (-1)^k \sin\left[\frac{\pi}{4M}(2k+1)(2n+1)\right] = -x^{(c)}_{M-1-n}, \\
y^{(1)}_{\frac{3M}{2}-1-n} &= \sum_{k=0}^{M-1} \Re e\{p_k\} (-1)^k \sin\left[\frac{\pi}{4M}(2k+1)(2n+1)\right] = x^{(c)}_{M-1-n}, \\
& n = 0, 1, \ldots, \frac{M}{2} - 1.
\end{aligned} \tag{7.80}
$$

The data sequence $\{x^{(c)}_n\}$ in (7.80) corresponds to an unnormalized M-point inverse DCT-IV of $\Re e\{p_k\}$. One can immediately see that the data sequence $\{y^{(1)}_n\}$ has the following local symmetries

$$
y^{(1)}_{\frac{M}{2}+n} = y^{(1)}_{\frac{M}{2}-1-n}, \qquad y^{(1)}_{\frac{3M}{2}-1-n} = -y^{(1)}_{\frac{3M}{2}+n}, \qquad n = 0, 1, \ldots, \frac{M}{2} - 1. \tag{7.81}
$$

Repeating exactly the same procedure for the second sum of (7.79) and using the relation (C.14) from Appendix C.2 we have

$$
\begin{aligned}
y^{(2)}_{\frac{M}{2}+n} &= \sum_{k=0}^{M-1} \Im m\{p_k\} \sin\left[\frac{\pi}{4M}(2k+1)(2n+1)\right] = x^{(s)}_n, \\
y^{(2)}_{\frac{M}{2}-1-n} &= -\sum_{k=0}^{M-1} \Im m\{p_k\} \sin\left[\frac{\pi}{4M}(2k+1)(2n+1)\right] = -x^{(s)}_n, \\
y^{(2)}_{\frac{3M}{2}+n} &= \sum_{k=0}^{M-1} \Im m\{p_k\} (-1)^k \cos\left[\frac{\pi}{4M}(2k+1)(2n+1)\right] = x^{(s)}_{M-1-n}, \\
y^{(2)}_{\frac{3M}{2}-1-n} &= \sum_{k=0}^{M-1} \Im m\{p_k\} (-1)^k \cos\left[\frac{\pi}{4M}(2k+1)(2n+1)\right] = x^{(s)}_{M-1-n}, \\
& n = 0, 1, \ldots, \frac{M}{2} - 1,
\end{aligned} \tag{7.82}
$$

where the data sequence $\{x^{(s)}_n\}$ in (7.82) corresponds to an unnormalized M-point inverse DST-IV of $\Im m\{p_k\}$, and the data sequence $\{y^{(2)}_n\}$ has the following local symmetries

$$y^{(2)}_{\frac{M}{2}+n} = -\, y^{(2)}_{\frac{M}{2}-1-n}, \qquad y^{(2)}_{\frac{3M}{2}-1-n} = y^{(2)}_{\frac{3M}{2}+n},$$
$$n = 0, 1, \ldots, \frac{M}{2} - 1. \tag{7.83}$$

After the computation of M-point inverse DCT-IV of $\Re e\{p_k\}$ and M-point inverse DST-IV of $\Im m\{p_k\}$, the final time domain data sequence $\{v_n\}$ is obtained by two ways as follows: (a) creating the data sequence $\{y^{(1)}_n\}$ using (7.80), then creating the data sequence $\{y^{(2)}_n\}$ using (7.82) followed by applying (7.78), or (b) by combining Eqs. (7.78), (7.80), and (7.82) the data sequence $\{v_n\}$ is directly obtained as

$$v_{\frac{M}{2}+n} = x^{(c)}_n - x^{(s)}_n, \qquad v_{\frac{3M}{2}+n} = -\, x^{(c)}_{M-1-n} - x^{(s)}_{M-1-n},$$
$$v_{\frac{M}{2}-1-n} = x^{(c)}_n + x^{(s)}_n, \qquad v_{\frac{3M}{2}-1-n} = x^{(c)}_{M-1-n} - x^{(s)}_{M-1-n},$$
$$n = 0, 1, \ldots, \frac{M}{2} - 1. \tag{7.84}$$

The efficient implementation of the complex exponential-modulated LD synthesis QMF bank requires $2M$ additions and the computation of two M-point unnormalized inverse DCTs-IV.

Complex Exponential-Modulated LD Down-Sampled Synthesis QMF Bank

We recall that the complex exponential-modulated LD down-sampled synthesis QMF bank given by (7.62) is the complex exponential-modulated LD synthesis QMF bank given by (7.60) but for the size $\frac{M}{2}$. Consequently, the efficient implementation of the complex exponential-modulated LD synthesis QMF bank defined by (7.78), (7.80), (7.82), and (7.84) can be reused for the complex exponential-modulated LD down-sampled synthesis QMF bank by taking $\frac{M}{2}$ instead of M (M is divisible by 4). The final time domain data sequence $\{v_n\}$ is obtained as

$$v_{\frac{M}{4}+n} = x^{(c)}_n - x^{(s)}_n, \qquad v_{\frac{3M}{4}+n} = -\, x^{(c)}_{\frac{M}{2}-1-n} - x^{(s)}_{\frac{M}{2}-1-n},$$
$$v_{\frac{M}{4}-1-n} = x^{(c)}_n + x^{(s)}_n, \qquad v_{\frac{3M}{4}-1-n} = x^{(c)}_{\frac{M}{2}-1-n} - x^{(s)}_{\frac{M}{2}-1-n},$$
$$n = 0, 1, \ldots, \frac{M}{4} - 1, \tag{7.85}$$

where

$$x^{(c)}_n = \sum_{k=0}^{\frac{M}{2}-1} \Re e\{p_k\} \cos\left[\frac{\pi}{4(M/2)}(2k+1)(2n+1)\right],$$

$$x_n^{(s)} = \sum_{k=0}^{\frac{M}{2}-1} \Im m\{p_k\} \sin\left[\frac{\pi}{4(M/2)}(2k+1)(2n+1)\right], \quad n = 0, 1, \ldots, \frac{M}{2} - 1. \tag{7.86}$$

This requires M additions and the computation of two $\frac{M}{2}$-point unnormalized inverse DCTs-IV.

7.3.4.3 LP-LD-SBR QMF Banks in the Decoder

The derivations of efficient implementations for real-valued cosine-modulated LD QMF banks in the LP-LD-SBR core decoder closely follow those of the corresponding complex exponential-modulated LD QMF banks in the HQ-LD-SBR core decoder. They are exactly defined as their real parts.

Real-Valued Cosine-Modulated LD Analysis QMF Bank

The efficient implementation of the real-valued cosine-modulated LD analysis QMF bank given by (7.65) is obtained from the real part (7.75) and mapping (7.76) by taking $\frac{M}{2}$ instead of M (M is divisible by 4) as

$$c_k = \sum_{n=0}^{\frac{M}{2}-1} (y_n - y_{M-1-n}) \cos\left[\frac{\pi}{4(M/2)}(2k+1)(2n+1)\right], \quad k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{7.87}$$

where

$$y_n - y_{M-1-n} = \begin{cases} u_{\frac{3M}{4}+n} + u_{\frac{3M}{4}-1-n}, & n = 0, 1, \ldots, \frac{M}{4} - 1, \\ -u_{n-\frac{M}{4}} + u_{\frac{3M}{4}-1-n}, & n = \frac{M}{4}, \frac{M}{4}+1, \ldots, \frac{M}{2} - 1. \end{cases} \tag{7.88}$$

It requires $\frac{M}{2}$ additions and the computation of one $\frac{M}{2}$-point unnormalized forward DCT-IV.

Real-Valued Cosine-Modulated LD Synthesis QMF Bank

The efficient implementation of the real-valued cosine-modulated LD synthesis QMF bank given by (7.67) is obtained from (7.80) and (7.84). The time domain aliased data sequence $\{v_n\}$ is given by

$$v_{\frac{M}{2}+n} = x_n^{(c)}, \qquad v_{\frac{3M}{2}+n} = -x_{M-1-n}^{(c)},$$
$$v_{\frac{M}{2}-1-n} = x_n^{(c)}, \qquad v_{\frac{3M}{2}-1-n} = x_{M-1-n}^{(c)},$$
$$n = 0, 1, \ldots, \frac{M}{2} - 1, \tag{7.89}$$

where

$$x_n^{(c)} = \sum_{k=0}^{M-1} c_k \cos\left[\frac{\pi}{4M}(2k+1)(2n+1)\right], \quad n = 0, 1, \ldots, M-1. \tag{7.90}$$

It requires only the computation of one M-point unnormalized inverse DCT-IV.

Real-Valued Cosine-Modulated LD Down-Sampled Synthesis QMF Bank

The efficient implementation of the real-valued cosine-modulated LD down-sampled synthesis QMF bank given by (7.69) is obtained directly from (7.89) and (7.90) by taking $\frac{M}{2}$ instead of M (M is divisible by 4). The time domain aliased data sequence $\{v_n\}$ is given by

$$v_{\frac{M}{4}+n} = x_n^{(c)}, \qquad v_{\frac{3M}{4}+n} = -x_{\frac{M}{2}-1-n}^{(c)},$$
$$v_{\frac{M}{4}-1-n} = x_n^{(c)}, \qquad v_{\frac{3M}{4}-1-n} = x_{\frac{M}{2}-1-n}^{(c)}, \qquad n = 0, 1, \ldots, \frac{M}{4} - 1, \tag{7.91}$$

where

$$x_n^{(c)} = \sum_{k=0}^{\frac{M}{2}-1} c_k \cos\left[\frac{\pi}{4(M/2)}(2k+1)(2n+1)\right], \quad n = 0, 1, \ldots, \frac{M}{2} - 1. \tag{7.92}$$

It requires the computation of one $\frac{M}{2}$-point unnormalized inverse DCT-IV.

7.4 Comparison of the Efficient QMF Bank Implementations

The arithmetic complexity of the complex exponential-modulated (LD) QMF banks in the HQ-SBR and HQ-LD-SBR encoder and decoder implemented using the direct method as block transforms are summarized in Table 7.1, while the arithmetic complexity of the real-valued cosine-modulated (LD) QMF banks in the LP-SBR and LP-LD-SBR decoder implemented using the direct method as block transforms are summarized in Table 7.2.

Table 7.1 The arithmetic complexity of the complex exponential-modulated (LD) QMF banks in the HQ-SBR and HQ-LD-SBR encoder and decoder implemented using the direct method as block transforms (M—real multiplication, A—real addition)

Complex exponential-modulated (LD) QMF bank	Direct implementation
Analysis (LD) QMF in encoder	16,384 M + 16,256 A
Analysis (LD) QMF in decoder	4096 M + 4032 A
Synthesis (LD) QMF in decoder	16,384 M + 16,512 A
Down-sampled synthesis (LD) QMF in decoder	4096 M + 4032 A

Table 7.2 The arithmetic complexity of the real-valued cosine-modulated (LD) QMF banks in the LP-SBR and LP-LD-SBR decoder implemented using the direct method as block transforms (M—real multiplication, A—real addition)

Real-valued cosine-modulated (LD) QMF bank	Direct implementation
Analysis (LD) QMF	2048 M + 2016 A
Synthesis (LD) QMF	8192 M + 8064 A
Down-sampled synthesis (LD) QMF	2048 M + 1984 A

7.4.1 Efficient QMF Banks Implementations in the HE-AAC

7.4.1.1 HQ-SBR QMF Banks

The efficient implementations of all complex exponential-modulated QMF banks in the HQ-SBR encoder and decoder rely on only one type of M-point forward or inverse DCT-IV, where $M = 64$. In the cases when two M-point DCTs-IV are needed to be computed, one for the real part and one for the imaginary part, they can be realized in parallel. If the FFT-based fast DCT-IV computational structure identical both for the forward and inverse DCT-IV computation is adopted (see Appendix C.2.1), it results in the unified efficient implementation of all complex QMF banks in the HQ-SBR encoder and decoder. Moreover, it can lead to a very fast, compact, and low-cost hardware (VLSI) architecture in terms of the regularity, structural simplicity, and minimal memory requirements without the need to reconfigure the fast DCT-IV computational structure.

On the other hand, if the DCT-II-based fast DCT-IV computational structure is adopted (see Appendix C.3), although it is more efficient in terms of the multiplicative complexity, it has to be inverted for the inverse DCT-IV computation. This fact results in two separate fast computational structures, one for the forward and one for the inverse DCT-IV. Moreover, the windowing function has to be modified too. The arithmetic complexity of the efficient implementations of the complex exponential-modulated QMF banks in the HQ-SBR encoder and decoder is summarized in Table 7.3.

Table 7.3 The arithmetic complexity of the efficient implementations of the complex exponential-modulated QMF banks in the HQ-SBR encoder and decoder

Complex exponential-modulated QMF banks in the HQ-SBR encoder/decoder	Fast DCT-IV computational structure	
	FFT-based	DCT-II-based
Analysis QMF in encoder	512 M + 1280 A	384 M + 1280 A
Analysis QMF in decoder	256 M + 576 A	192 M + 576 A
Synthesis QMF in decoder	512 M + 1280 A	384 M + 1280 A
Down-sampled synthesis QMF in decoder	256 M + 576 A	192 M + 576 A

Table 7.4 The arithmetic complexity of the efficient implementations of the real-valued cosine-modulated QMF banks in the LP-SBR decoder

Real-valued cosine-modulated QMF banks in the LP-SBR decoder	Fast DCT-II/III computational structure	
	Classic algorithm	DCT-IV-based
Analysis QMF	80 M + 240 A	144 M + 304 A
Synthesis QMF	192 M + 513 A	320 M + 639 A
Down-sampled synthesis QMF	80 M + 209 A	144 M + 271 A

7.4.1.2 LP-SBR QMF Banks

The efficient implementations of the real-valued cosine-modulated QMF banks in the LP-SBR decoder, although of much lower arithmetic complexity, require the computation of one $\frac{M}{2}$-point DCT-III and one reconfigurable M-/$\frac{M}{2}$-point DCT-II, where $M = 64$. This fact indicates that one fast reconfigurable DCT-II computational structure is needed (see Appendix C.1.1) which has to be inverted for the $\frac{M}{2}$-point DCT-III computation. Thus, two separate fast computational structures are required. The same is valid for the DCT-IV-based fast DCT-II/III computational structures (see Appendix C.3), however, with the higher arithmetic complexity.

The FFT-based fast DCT-IV computational structure used in the HQ-SBR may be also shared in the LP-SBR. The arithmetic complexity of the efficient implementations of the real-valued cosine-modulated QMF banks in the LP-SBR decoder is summarized in Table 7.4.

7.4.2 Efficient LD QMF Banks Implementations in AAC-ELD

7.4.2.1 HQ-LD-SBR QMF Banks

In contrast to the efficient implementations of the complex QMF banks in the HQ-SBR encoder and decoder, where only one type of M-point forward and inverse DCT-IV is required, the efficient implementations of all complex exponential-modulated LD QMF banks in the HQ-LD-SBR encoder and decoder rely on one reconfigurable M/$\frac{M}{2}$-point forward or inverse DCT-IV, where $M = 64$. When

two M-point or two $\frac{M}{2}$-point DCTs-IV are needed to be computed, they can be realized in parallel. Obviously, the FFT-based fast DCT-IV computational structure (see Appendix C.2.1) or the DCT-II-based fast DCT-IV computational structure may be adopted (see Appendix C.3) but in the reconfigurable form. Therefore, the conclusions mentioned in subsection 7.4.1.1 are almost valid for the HQ-LD-SBR encoder and decoder. The arithmetic complexity of the complex exponential-modulated LD QMF banks efficient implementations in the HQ-LD-SBR encoder and decoder is summarized in Table 7.5.

7.4.2.2 LP-LD-SBR QMF Banks

Since the real-valued cosine-modulated LD QMF banks in the LP-LD-SBR decoder are defined as the real parts of corresponding complex exponential-modulated LD QMF banks in the HQ-LD-SBR decoder, their efficient implementations are also based on one reconfigurable $M/\frac{M}{2}$-point forward or inverse DCT-IV, where $M = 64$. Thus, the reconfigurable FFT-based or DCT-II-based fast DCT-IV computational structures provide very compact systems for the unified efficient implementation of all QMF banks both in the HQ-LD-SBR encoder and decoder and LP-LD-SBR decoder. The arithmetic complexity of the real-valued cosine-modulated LD QMF banks efficient implementations in the LP-LD-SBR decoder is summarized in Table 7.6.

In summary, the reconfigurable FFT-based fast DCT-IV computational structure provides a transform engine for the efficient implementations of all complex and real-valued QMF banks in the HQ-SBR, LP-SBR, HQ-LD-SBR, and LP-LD-SBR encoder and decoder. Since many commercial FFT software and hardware

Table 7.5 The arithmetic complexity of the complex exponential-modulated LD QMF banks efficient implementations in the HQ-LD-SBR encoder and decoder

Complex exponential-modulated LD QMF banks in the HQ-LD-SBR encoder/decoder	Fast DCT-IV computational structure	
	FFT-based	DCT-II-based
Analysis LD QMF in encoder	512 M + 1280 A	384 M + 1280 A
Analysis LD QMF in decoder	224 M + 544 A	160 M + 544 A
Synthesis LD QMF in decoder	512 M + 1280 A	384 M + 1280 A
Down-sampled synthesis LD QMF in decoder	224 M + 544 A	160 M + 544 A

Table 7.6 The arithmetic complexity of the real-valued cosine-modulated LD QMF banks efficient implementations in the LP-LD-SBR decoder

Real-valued cosine-modulated LD QMF banks in the LP-LD-SBR decoder	Fast DCT-IV computational structure	
	FFT-based	DCT-II-based
Analysis QMF	112 M + 272 A	80 M + 272 A
Synthesis QMF	256 M + 576 A	192 M + 576 A
Down-sampled synthesis QMF	112 M + 240 A	80 M + 240 A

products are available on websites: S/W tools, chips, digital signal processors (DSP), hardware/software implementations on DSP, VLSI, and FPGA [53], this fact can result in a low-cost implementation of the QMF banks in the HQ-SBR, LP-SBR HQ-LD-SBR, and LP-LD-SBR decoders.

7.5 Summary

The complete unified efficient low-cost implementations of the complex and real-valued analysis/synthesis QMF banks used in the standard SBR encoder and decoder (HQ-SBR and LP-SBR) and their low delay versions used in the LD-SBR encoder and decoder (HQ-LD-SBR and LP-LD-SBR) have been presented. They are based on the fast DCT-IV and DCT-II/DCT-III computational structures and are efficient in terms of the computational complexity, regularity, and structural simplicity. The efficient implementations of the QMF banks used in the standard SBR (HQ-SBR and LP-SBR) as well as used in the LD-SBR encoder and decoder (HQ-LD-SBR and LP-LD-SBR) are separately compared with respect to the arithmetic complexity. In particular, all the fast algorithms have been analyzed in detail in terms of the regularity and structural simplicity for a potential real-time low-cost implementation in software or hardware.

Problems and Exercises

1. Verify the even symmetry property of sub-band coefficients $\{c_k\}$ given by (7.17) of the real-valued cosine-modulated analysis QMF bank used in the standard LP-SBR decoder.
2. Verify the even anti-symmetry property of sub-band coefficients $\{c_k\}$ given by (7.64) of the real-valued cosine-modulated LD analysis QMF bank used in the LP-LD-SBR decoder.
3. For the standard SBR in HE-AAC encoder, implement by a computer program the fast algorithm for the computation of complex exponential-modulated analysis QMF bank defined by (7.23), where $M = 64$. For the efficient implementation, use both the FFT-based fast M-point DCT-IV computational structure (see Appendix C.2.1), and the DCT-II-based fast M-point DCT-IV computational structure (see Appendix C.3). Compare their arithmetic complexity.
4. For the standard HQ-SBR in HE-AAC decoder, implement by a computer program the unified fast computation of:
   - Complex exponential-modulated analysis QMF bank defined by (7.24) and (7.25),

- Complex exponential-modulated synthesis QMF bank defined by (7.26)–(7.28), and
- Complex exponential-modulated down-sampled synthesis QMF bank defined by (7.35), (7.37), and (7.40),

using only one FFT-based fast M-point DCT-IV computational structure, where $M = 64$.

5. Verify the correctness of the first alternative method for the efficient implementation of complex exponential-modulated down-sampled synthesis QMF bank defined by (7.29) via the efficient implementation of complex exponential-modulated synthesis QMF bank defined by (7.26)–(7.28).
6. Verify the correctness of the second alternative method for the efficient implementation of complex exponential-modulated down-sampled synthesis QMF bank defined by (7.30)–(7.32) via the efficient implementation of complex exponential-modulated synthesis QMF bank defined by (7.26)–(7.28).
7. For the standard LP-SBR in HE-AAC decoder, implement by a computer program the unified fast computation of:

   - Real-valued cosine-modulated analysis QMF bank defined by (7.44) and (7.45),
   - Real-valued cosine-modulated synthesis QMF bank defined by (7.47) and (7.48), and
   - Real-valued cosine-modulated down-sampled synthesis QMF bank defined by (7.49) and (7.50),

   using one reconfigurable (M-point and $\frac{M}{2}$-point) fast DCT-II/III computational structure, where $M = 64$.
8. For the LD-SBR in AAC-ELD encoder, implement by a computer program the fast algorithm for the computation of complex exponential-modulated LD analysis QMF bank defined by (7.72)–(7.74), where $M = 64$. For the efficient implementation, use both the FFT-based fast M-point DCT-IV computational structure (see Appendix C.2.1), and the DCT-II-based fast M-point DCT-IV computational structure (see Appendix C.3). Compare their arithmetic complexity.
9. For the HQ-LD-SBR in AAC-ELD decoder, implement by a computer program the unified fast computation of:

   - Complex exponential-modulated LD analysis QMF bank defined by (7.75)–(7.77),
   - Complex exponential-modulated LD synthesis QMF bank defined by (7.80), (7.82), and (7.84), and
   - Complex exponential-modulated LD down-sampled synthesis QMF bank defined by (7.85) and (7.86),

   using only reconfigurable (M-point and $\frac{M}{2}$-point) FFT-based fast DCT-IV computational structure, where $M = 64$.

10. Similarly, for the LP-LD-SBR in AAC-ELD decoder, implement by a computer program the unified fast computation of:

    - Real-valued cosine-modulated LD analysis QMF bank defined by (7.87) and (7.88),
    - Real-valued cosine-modulated LD synthesis QMF bank defined by (7.89) and (7.90), and
    - Real-valued cosine-modulated LD down-sampled synthesis QMF bank defined by (7.91) and (7.92),

    using only reconfigurable (M-point and $\frac{M}{2}$-point) FFT-based fast DCT-IV computational structure, where $M = 64$.

References

1. M. Dietz, L. Liljeryd, K. Kjörling, O. Kunz, Spectral band replication, a novel approach in audio coding, in *112th AES Convention*, Munich, May 2002. Preprint #5553
2. P. Ekstrand, Bandwidth extension of audio signals by Spectral Band Replication, in *Proceedings of the 1st IEEE Benelux Workshop on Model based Processing and Coding of Audio (MPCA'2002)*, Leuven, November 2002, pp. 53–58
3. H.-W. Hsu, W.-C. Lee, C.-M. Liu, H.-Y. Tseng, C.-H. Yang, High quality, low power QMF bank design for SBR, parametric coding, and MPEG surround decoders, in *122nd AES Convention*, Vienna, May 2007. Preprint #7000
4. L. Liljeryd, P. Ekstrand, L. Henn, K. Kjörling, Source coding enhancement using spectral-band replication, US Patent Application #7283955 B2, Coding Technology Ab, Stockholm, October 2003
5. O. Shimada et al., A low power SBR algorithm for the MPEG-4 audio standard and its DSP implementation, in *116th AES Convention*, Berlin, May 2004. Preprint #6048

Standard SBR Integrated into MPEG-1 Layer 2, MP3 and MPEG-4 AAC

6. Digital Radio Mondiale, available on web site: http://www.drm.org/
7. A. Ehret, M. Dietz, K. Kjörling, State-of-the-art audio coding for broadcasting and mobile applications, in *114th AES Convention*, Amsterdam, March 2003. Preprint #5834
8. W. Hoeg, T. Lauterbach (eds.), *Digital Audio Broadcasting: Principles and Applications of DAB, DAB+*, 3rd edn. (Wiley, Chichester, 2009), pp. 93–165. Chapter 3: Audio services and applications
9. S. Meltzer, R. Böhm, F. Henn, SBR enhanced audio codecs for Digital Broadcasting such as Digital Radio Mondiale (DRM), in *112th AES Convention*, Munich, April 2002. Preprint #5559
10. M. Schug, A. Gröschel, M. Beer, F. Henn, Enhancing audio coding efficiency of MPEG layer-2 with Spectral Band Replication (SBR) for DigitalRadio (EUREKA 147/DAB) in a backwards compatible way, in *114th AES Convention*, Amsterdam, March 2003. Preprint #5850
11. XM Satellite Radio, available on web site: http://www.xmradio.com/
12. T. Ziegler, A. Ehret, P. Ekstrand, M. Lutzky, Enhancing mp3 with SBR: features and capabilities of the new mp3PRO algorithm, in *112th AES Convention*, Munich, May 2002. Preprint #5560

Standard SBR in HE-AAC (aacPlus) and USAC

13. A.C. den Brinker et al., An overview of the coding standard MPEG-4 audio Amendments 1 and 2: HE-AAC, SSC and HE-AAC v2. EURASIP J. Audio Speech Music Process. (2009). Article ID 468971, 21 pp.
14. M. Dietz, S. Meltzer, CT-aacPlus – a state-of-the-art audio coding scheme, in *European Broadcasting Union Technical Review*, vol. 291, July 2002, 7 pp.
15. D. Frerichs, New MPEG-4 High-Efficiency AAC audio: enabling new applications, Document of Coding Technologies, Inc., April 2003, 8 pp.
16. Y. Gao, Audio coding standard overview: MPEG-4 AAC, HE-AAC, and HE-AAC v2, in *Mobile Multimedia Broadcasting Standards: Technology and Practice*, chap. 21, ed. by F.-L. Luo (Springer Science + Business Media LLC, New York, 2009), pp. 607–627
17. J. Herre, M. Dietz, MPEG-4 high-efficiency AAC coding. IEEE Signal Process. Mag. **25**(3), 137–142 (2008)
18. Information Technology – Coding of Audio–Visual Objects – Part 3: Audio, Subpart 4: General Audio Coding (GA) — AAC, TwinVQ, BSAC, ISO/IEC 14496-3:2005(E) (2005)
19. M. Neuendorf et al., A novel scheme for low bitrate Unified Speech and Audio Coding – MPEG RM0, in *126th AES Convention*, Munich, May 2009. Preprint #7713
20. M. Neuendorf et al., The ISO/MPEG Unified Speech and Audio Coding standard – consistent high quality for all content types and at all bit rates, in *132nd AES Convention*, Budapest, April 2012. Preprint #8654, also published in J. Audio Eng. Soc. **61**(12), 956–977 (2013)
21. S. Quackenbush, MPEG unified speech and audio coding. IEEE MultiMedia **20**(2), 72–78 (2013)
22. S.-U. Ryu, K. Rose, J.-H. Chang, Effective high frequency regeneration based on sinusoidal modeling for MPEG-4 HE-AAC, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, October 2005, pp. 211–214
23. M. Wolters, K. Kjörling, D. Homm, H. Purnhagen, A closer look into MPEG-4 high efficiency AAC, in *115th AES Convention*, New York, October 2003. Preprint #5871

Low Delay SBR in AAC-ELD

24. T. Friedrich, G. Schuller, Spectral Band Replication tool for very low delay audio coding applications, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, October 2007, pp. 199–202
25. Information Technology – Coding of Audio-Visual Objects – Part 3: Audio, Amendment 9: Enhanced Low Delay AAC, ISO/IEC 14496-3:2005/FDAM 9:2007(E), Shenzen, October 2007
26. M. Schnell et al., Enhanced MPEG-4 low delay AAC – low bit rate high quality communication, in *122nd AES Convention*, Vienna, May 2007. Preprint #6998
27. M. Schnell et al., Low delay filter banks for enhanced low delay audio coding, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, October 2007, pp. 235–238
28. M. Schnell et al., MPEG-4 enhanced low delay AAC – a new standard for high quality communication, in *125nd AES Convention*, San Francisco, CA, October 2008. Preprint #7503
29. M. Werner, G. Schuller, An enhanced SBR tool for low-delay applications, in *127th AES Convention*, New York, NY, October 2009. Preprint #7876
30. M. Werner, G. Schuller, An SBR tool for very low delay applications with flexible crossover frequency. *Proceedings of the IEEE ICASSP'2010*, Dallas, TX, March 2010, pp. 353–356

Fast (Low Delay) HQ-SBR and LP-SBR QMF Banks

31. V. Britanak, Spectral Band Replication (SBR) compression technology: survey of the unified efficient low-cost implementations of complex exponential- and cosine-modulated QMF banks. Signal Process. **115**(10), 49–65 (2015)
32. L.-G. Chen, S.-W. Huang, T.-H. Tsai, Fast filterbanks for the low power MPEG high efficiency advanced audio coding decoder, in *118th AES Convention*, Barcelona, May 2005. Preprint #6336
33. R.K. Chivukula, Y. Reznik, Low complexity Spectral Band Replication (SBR) filterbanks, US Patent Application #20100262427, Qualcomm Inc., San Diego, CA, October 2010
34. R.K. Chivukula, Y.A. Reznik, V. Devarajan, M. Jayendra-Lakshman, Fast algorithms for low-delay SBR filterbanks in MPEG-4 AAC-ELD. IEEE Trans. Audio Speech Lang. Process. **20**(3), 1022–1031 (2012)
35. Coding Technologies, Embedded aacPlus decoder implementations – product sheet, June 2013
36. Coding Technologies, Embedded HE-AAC: implementation guideline, June 2013
37. FAAD2 – open source free-ware Advanced Audio (AAC) decoder including SBR decoding, Free software at website http://www.audiocoding.com/ by M. Bakker, Nero AG, January 2011
38. S.-W. Huang, T.-H. Tsai, L.-G. Chen, Fast decomposition of filterbanks for the state-of-the-art audio coding. IEEE Signal Process. Lett. **12**(10), 693–696 (2005)
39. J. Huang, G. Du, D. Zhang, Y. Song, L. Geng, M. Gao, VLSI design of resource shared complex-QMF bank for HE-AAC decoder, in *Proceedings of the 8th IEEE International Conference on ASIC (ASICON'2009)*, Changsha, October 2009, pp. 796–799
40. H.-W. Hsu, C.-M. Liu, W.-C. Lee, Fast complex quadrature mirror filterbanks for MPEG-4 HE-AAC, in *121st AES Convention*, San Francisco, CA, October 2006. Preprint #6871
41. Intel(R) Integrated Performance Primitives Reference Manual, Vol. 1: Signal Processing, QMF Functions (Quadrature mirror filter banks used by MPEG-4 SBR decoder), Intel Developer Zone, available at web site: http://software.intel.com/

Supporting Literature

42. V. Britanak, On the discrete cosine transform computation. Signal Process. **40**(2–3), 183–194 (1994)
43. V. Britanak, New universal rotation-based fast computational structures for an efficient implementation of the DCT-IV/DST-IV and analysis/synthesis MDCT/MDST filter banks. Signal Process. **89**(11), 2213–2232 (2009)
44. V. Britanak, P. Yip, K.R. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations* (Academic Press Inc., Elsevier Science, Amsterdam, 2007)
45. V. Britanak, H.J. Lincklaen Arriëns, Fast computational structures for an efficient implementation of the complete TDAC analysis/synthesis MDCT/MDST filter banks. Signal Process. **89**(7), 1379–1394 (2009)
46. S.C. Chan, K.L. Ho, Direct methods for computing discrete sinusoidal transforms. IEE Proc. Part F: Radar Signal Process. **137**(6), 433–442 (1990)
47. R. Gluth, Regular FFT-related transform kernels for DCT/DST-based polyphase filter banks, in *Proceedings of the IEEE ICASSP'91*, Toronto, May 1991, pp. 2205–2208
48. C.W. Kok, Fast algorithm for computing discrete cosine transform. IEEE Trans. Signal Process. **45**(3), 757–760 (1997)
49. K. Konstantinides, Fast subband filtering in MPEG audio coding. IEEE Signal Process. Lett. **1**(2), 26–28 (1994)

50. C.-M. Liu, W.-C. Lee, A unified fast algorithm for cosine modulated filter banks in current audio coding standards. J. Audio Eng. Soc. **47**(12), 1061–1075 (1999)
51. H.S. Malvar, *Signal Processing with Lapped Transforms*, chap. 2 (Artech House, Norwood, MA, 1992), pp. 71–75
52. N. Rama Murthy, M.N.S. Swamy, On the algorithms for the computation of even discrete cosine transform-2 (EDCT-2) of real sequences. IEEE Trans. Circuits Syst. **37**(5), 625–627 (1990)
53. K.R. Rao, D.N. Kim, J.J. Hwang, *Fast Fourier Transform: Algorithms and Applications* (Springer Science + Business Media B.V., New York, 2010)
54. G.D.T. Schuller, T. Karp, Modulated filter banks with arbitrary system delay: efficient implementations and time-varying case. IEEE Trans. Signal Process. **48**(3), 737–748 (2000)
55. G.D.T. Schuller, M.J.T. Smith, New framework for modulated perfect reconstruction filter banks. IEEE Trans. Signal Process. **44**(8), 1941–1954 (1996)
56. P.P. Vaidyanathan, Cosine modulated filter banks, in *Multirate Systems and Filter Banks*, chap. 8 (Prentice–Hall, Englewood Cliffs, NJ, 1993), pp. 353–393
57. M. Vetterli, H.J. Nussbaumer, Simple FFT and DCT algorithms with reduced number of operations. Signal Process. **6**(4), 267–278 (1984)

Chapter 8
Efficient Implementations of Perfect Reconstruction Low Delay Cosine-Modulated Filter Banks in the MPEG-4 AAC-ELD

8.1 Introduction

The MPEG committee has recently completed development of a new audio coding standard, MPEG-4 Advanced Audio Coding-Enhanced Low Delay (AAC-ELD) [1], targeted towards high-quality real-time (interactive) bidirectional communication applications, such as audio and video conferencing. Common audio coding schemes and state-of-the-art MPEG audio coding standards, such as MPEG-4 AAC Low Complexity (AAC-LC), High Efficiency AAC (HE-AAC) [2, 3], and AAC Low Delay (AAC-LD) [4–6], utilize for the time-to-frequency transformation of an audio data block and vice versa, the well-known perfect reconstruction cosine-modulated filter banks, the time domain aliasing cancellation modified discrete cosine transform (TDAC-MDCT) [34]. In order to achieve the high coding efficiency and low algorithmic delay, the AAC-ELD combines a low delay-optimized Spectral Band Replication (SBR) compression technology (see Chap. 7) known from the HE-AAC [2, 3] and a perfect reconstruction low delay cosine-modulated filter bank, called the low delay MDCT (LD-MDCT) [8–10]. Very recently, MPEG finished the standardization of a low delay MPEG surround as a parametric stereo coding tool for the AAC-ELD codec. The combination of both technologies, the parametric stereo coding tool and AAC-ELD, is also known as the AAC-ELD v2 [7, 11]. The applications of AAC-ELD v2 codec involve broadcasting and mobile videoconferencing. Essentially, the AAC-LC, HE-AAC, and AAC-LD audio codecs form the basis of AAC-ELD.

Although the use of LD-MDCT substantially reduces the algorithmic delays, the transform operations in the AAC-ELD codec are still computationally intensive and the LD-MDCT filter banks need to have fast algorithms, and in particular, when the block length is a composite number. Therefore, this chapter is concentrated

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1_8

on the perfect reconstruction analysis/synthesis LD-MDCT filter banks used in the AAC-ELD codec and mainly on their efficient implementations. Specifically, this chapter presents:

- Definitions of the analysis/synthesis LD-MDCT (and TDAC-MDCT) filter banks, and general symmetry properties of LD-MDCT block transforms both in the time and frequency domains.
- Relations between the LD-MDCT and TDAC-MDCT block transforms in the analytical forms as well as in the equivalent matrix representations. This fact enables us to map the LD-MDCT into TDAC-MDCT block transform and provides the basis for a unified approach to efficiently implement both the LD-MDCT and TDAC-MDCT block transforms in the state-of-the-art MPEG audio codecs.
- Efficient implementations of the even-length analysis/synthesis LD-MDCT filter banks based on the TDAC-MDCT as well as the efficient implementations without mapping the LD-MDCT to TDAC-MDCT. For each fast LD-MDCT algorithm all the complete formulae are derived.
- All the fast even-length LD-MDCT algorithms are investigated and compared in terms of arithmetic complexity and structural simplicity.

Finally, consequences of the fast even-length LD-MDCT algorithms to other existing audio broadcasting standards and speech communication codecs are discussed, when the block sizes are composite numbers.

8.2 Definitions of the LD-MDCT and TDAC-MDCT Filter Banks

8.2.1 Analysis/Synthesis LD-MDCT Filter Banks

The analysis and synthesis LD-MDCT filter banks are, respectively, defined as [1, 8–10]

$$c_k^{(i)-\mathrm{LD}} = -2\sum_{n=-N}^{N-1} w_n^{(a)}\, x_n^{(i)}\, \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1-\frac{N}{2}\right)\right], \quad k=0,1,\ldots,\frac{N}{2}-1, \tag{8.1}$$

$$\begin{aligned}\hat{x}_n^{(i)-\mathrm{LD}} &= -\frac{2}{N} w_n^{(s)} \sum_{k=0}^{\frac{N}{2}-1} c_k^{(i)-\mathrm{LD}}\, \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1-\frac{N}{2}\right)\right], \\ & n = 0,1,\ldots,2N-1,\end{aligned} \tag{8.2}$$

where the superscript (i) denotes the data-block number, $\{x_n^{(i)}\}$ is the input data block of the length $2N$ (N is divisible by 4), $\{c_k^{(i)-\mathrm{LD}}\}$ are LD-MDCT frequency coefficients, and $\{\hat{x}_n^{(i)-\mathrm{LD}}\}$ is the time domain aliased data sequence. $\{w_n^{(a)}\}$ represents an asymmetric low delay analysis windowing function, while $\{w_n^{(s)}\}$ represents an asymmetric low delay synthesis windowing function which is simply time-reversed replica of the low delay analysis windowing function, i.e., $w_n^{(s)} = w_{2N-1-n}^{(a)}$, $n = 0, 1, \ldots, 2N-1$. The low delay analysis windowing functions for $N = 1024$ and 960 are tabulated in [1]. Plots of low delay analysis and synthesis windowing functions for $N = 960$ are shown in [8–10]. Note that the first $\frac{N}{8}$ values of $\{w_n^{(a)}\}$ both for $N = 1024$ and 960 are implicitly equal to zero [1].

The original data sequence $\{x_n^{(i)}\}$ is perfectly reconstructed by adding outputs of the synthesis LD-MDCT filter banks of four succeeding data blocks $(i-3)$, $(i-2)$, $(i-1)$, and (i) overlapped by $\frac{N}{2}$ samples (the so-called overlap/add procedure) as follows:

$$x_n^{(i)} = \hat{x}_n^{(i)-\mathrm{LD}} + \hat{x}_{\frac{N}{2}+n}^{(i-1)-\mathrm{LD}} + \hat{x}_{N+n}^{(i-2)-\mathrm{LD}} + \hat{x}_{\frac{3N}{2}+n}^{(i-3)-\mathrm{LD}}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{8.3}$$

8.2.2 Analysis/Synthesis TDAC-MDCT Filter Banks

We recall that the analysis and synthesis TDAC-MDCT filter banks are, respectively, defined as [34]

$$c_k^{(i)-\mathrm{TDAC}} = \frac{4}{N} \sum_{n=0}^{N-1} w_n \, x_n^{(i)} \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}\right)\right], \quad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{8.4}$$

$$\hat{x}_n^{(i)-\mathrm{TDAC}} = w_n \sum_{k=0}^{\frac{N}{2}-1} c_k^{(i)-\mathrm{TDAC}} \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}\right)\right], \quad n = 0, 1, \ldots, N-1, \tag{8.5}$$

where the superscript (i) denotes the data-block number, $\{x_n^{(i)}\}$ is the input data block of the length N (N is divisible by 4), $\{c_k^{(i)-\mathrm{TDAC}}\}$ are TDAC-MDCT frequency coefficients, and $\{\hat{x}_n^{(i)-\mathrm{TDAC}}\}$ is the time domain aliased data sequence. $\{w_n\}$ represents a symmetric windowing function being identical both for analysis and synthesis TDAC-MDCT filter banks.

The original data sequence $\{x_n^{(i)}\}$ is perfectly reconstructed by adding outputs of the synthesis TDAC-MDCT filter banks of two succeeding data blocks $(i-1)$ and (i) overlapped by $\frac{N}{2}$ samples (the overlap/add procedure) as follows:

$$x_n^{(i)} = \hat{x}_{\frac{N}{2}+n}^{(i-1)-\mathrm{TDAC}} + \hat{x}_n^{(i)-\mathrm{TDAC}}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{8.6}$$

8.2.3 General Comments on LD-MDCT and TDAC-MDCT Filter Banks

The purpose of low delay filter banks is to reduce their reconstruction delay independently of the prototype filter length, while still maintaining the perfect reconstruction property. The theory and methods for the design and implementation of perfect reconstruction modulated filter banks with arbitrary system delay are well described and analyzed in [31, 35, 36]. The resulting low delay filter banks have the same cosine modulation function as the TDAC-MDCT, but they can have longer windowing functions which can be nonsymmetric with a generalized or low reconstruction delay. In fact, the LD-MDCT has a similar cosine modulation kernel as TDAC-MDCT, but substantial delay reduction is achieved by utilizing a nonsymmetric windowing function with a low reconstruction delay and with multiple overlap. The asymmetric windowing function allows to reduce the overlap towards future samples and at the same time its impulse response is extended towards past samples. This cannot be accomplished with TDAC-MDCT which employs a symmetric windowing function and thus has a system delay identical to the block size minus one [8–10].

Obviously, when investigating and developing fast algorithms/computational structures for an efficient implementation of (low delay) analysis and synthesis filter banks, their complete analytical forms are frequently considered as the forward/backward block transforms applied to a single data block. Without loss of generality, we may omit the data-block number (i) and normalization factors from the definitions of LD-MDCT filter banks given by (8.1) and (8.2), and TDAC-MDCT filter banks given by (8.4) and (8.5). Similarly, assuming that the input/output data sequences are windowed before/after their transformation, we may omit the windowing functions too.

8.2.4 Forward/Backward LD-MDCT as the Block Transforms

The forward and backward LD-MDCT block transforms are, respectively, defined as

$$c_k^{\mathrm{LD}} = -\sum_{n=-N}^{N-1} x_n \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1-\frac{N}{2}\right)\right], \quad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{8.7}$$

$$\hat{x}_n^{\text{LD}} = -\sum_{k=0}^{\frac{N}{2}-1} c_k^{\text{LD}} \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1-\frac{N}{2}\right)\right], \quad n=0,1,\ldots,2N-1. \tag{8.8}$$

The $2N$-point forward LD-MDCT given by (8.7) can be reduced to N-point forward LD-MDCT as follows. Splitting the summation (8.7) into two parts, we get [13, 14]

$$\begin{aligned} c_k^{\text{LD}} &= -\sum_{n=-N}^{-1} x_n \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1-\frac{N}{2}\right)\right] \\ &\quad -\sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1-\frac{N}{2}\right)\right] \\ &= -\sum_{n=0}^{N-1} x_{n-N} \cos\left[\frac{\pi}{2N}(2k+1)\left(-2N+2n+1-\frac{N}{2}\right)\right] \\ &\quad -\sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1-\frac{N}{2}\right)\right] \\ &= -\sum_{n=0}^{N-1} (x_n - x_{n-N}) \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1-\frac{N}{2}\right)\right], \\ &\quad k=0,1,\ldots,\frac{N}{2}-1. \end{aligned} \tag{8.9}$$

However, the indexing x_{n-N} in (8.9) can be changed to x_{N+n}. In fact, substituting $n-N$ and then $N+n$ for $n=0,1,\ldots,N-1$ into the cosine transform kernel in (8.9) we find that it corresponds to $-\cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1-\frac{N}{2}\right)\right]$. This fact actually explains the periodicity property of $\{x_n\}$. Consequently, Eq. (8.9) may be rewritten into a new form defined as

$$c_k^{\text{LD}} = -\sum_{n=0}^{N-1} (x_n - x_{N+n}) \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1-\frac{N}{2}\right)\right], \quad k=0,1,\ldots,\frac{N}{2}-1. \tag{8.10}$$

Thus, the $2N$-point forward LD-MDCT given by (8.7) is reduced to N-point forward LD-MDCT given by (8.10). Equation (8.10) implies that the forward LD-MDCT block transform given by (8.7) can be written in an alternative useful form as

$$c_k^{\text{LD}} = \sum_{n=-N}^{N-1} x_{N+n} \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1-\frac{N}{2}\right)\right], \quad k=0,1,\ldots,\frac{N}{2}-1. \tag{8.11}$$

8.2.4.1 Symmetry Properties of the Forward/Backward LD-MDCT Block Transforms

Substituting $N-1-k$ for k into (8.10) we get

$$c_{N-1-k}^{\text{LD}} = -\,c_k^{\text{LD}}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{8.12}$$

demonstrating that the $\{c_k^{\text{LD}}\}$ has the even anti-symmetry property. On the other hand, substituting $N+n$ for n into (8.8) we obtain

$$\hat{x}_{N+n}^{\text{LD}} = -\,\hat{x}_n^{\text{LD}}, \qquad n = 0, 1, \ldots, N-1. \tag{8.13}$$

Consequently, only half the samples of $\{\hat{x}_n^{\text{LD}}\}$ in (8.8) is sufficient to be computed. Using the symmetry property (8.13) the backward LD-MDCT block transform is defined as

$$\hat{x}_n^{\text{LD}} = -\sum_{k=0}^{\frac{N}{2}-1} c_k^{\text{LD}} \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1-\frac{N}{2}\right)\right], \quad n = 0, 1, \ldots, N-1. \tag{8.14}$$

8.2.5 *Forward/Backward TDAC-MDCT as the Block Transforms*

The forward and backward TDAC-MDCT block transforms are, respectively, defined as [28]

$$c_k^{\text{TDAC}} = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}\right)\right], \quad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{8.15}$$

$$\hat{x}_n^{\text{TDAC}} = \sum_{k=0}^{\frac{N}{2}-1} c_k^{\text{TDAC}} \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}\right)\right], \quad n = 0, 1, \ldots, N-1. \tag{8.16}$$

The symmetry properties of data sequences $\{c_k^{\text{TDAC}}\}$ and $\{\hat{x}_n^{\text{TDAC}}\}$ as well as the general mathematical and special properties of the TDAC-MDCT block transforms are presented in [28] (see also Chap. 3).

It can be easily seen that the cosine transform kernels of the LD-MDCT in (8.10) and (8.14), and those of the TDAC-MDCT in (8.15) and (8.16), differ only by the shift factor $\mp\frac{N}{2}$.

8.2.6 *TDAC-MDCT and LD-MDCT Transforms in the Current Audio Codecs*

We recall that the state-of-the-art MPEG-4 audio coding standards: AAC-LC, HE-AAC [2, 3], and AAC-LD [4–6], utilize for the time-to-frequency transformation of the audio data block and vice versa, the TDAC-MDCT transform. In order to adapt to the signal characteristics, the AAC-LC and HE-AAC use the so-called block size switching procedure. Specifically, when the audio signal is stationary, the long block of size $N = 2048$ is used. When a transient signal is detected, then eight short blocks of size $N = 256$ are used. The HE-AAC alternatively defines the long block of size $N = 1920$ and short block of size $N = 240$. The advanced Digital Audio Broadcasting (DAB+) system [20] for digital radio services as well as the Digital Radio Mondiale (DRM) [19], universal openly standardized digital broadcasting system for all broadcasting frequencies, have adopted the HE-AAC codec with the TDAC-MDCT transform of the length $N = 1920$ (long block) or $N = 240$ (short block). We note that these block sizes are composite numbers, i.e., they are of the form $1920 = 2^7 \times 15$ and $240 = 2^4 \times 15$.

In general, the transform length influences the algorithmic delay (the theoretical minimum delay allowed by an algorithm due to computational speed, or other implementation circumstances). In order to reduce algorithmic delay, the AAC-LD has deactivated the block size switching procedure and block sizes have been reduced from $N = 2048$ to 1024 and from $N = 1920$ to 960 for the TDAC-MDCT transform. Finally, for the time-to-frequency transformation of the audio block and vice versa, AAC-ELD for the LD-MDCT transform defines the block of size $2N$, where $N = 1024$ or 960. Again, the block size $N = 960$ is a composite number, i.e., it is of the form $960 = 2^6 \times 15$.

8.3 Relations Between the LD-MDCT and TDAC-MDCT Block Transforms

In the following subsections, relations between the forward/backward LD-MDCT and the forward/backward TDAC-MDCT block transforms are discussed. They provide a unified approach to efficiently implement both the LD-MDCT and TDAC-MDCT via a fast TDAC-MDCT computational structure.

8.3.1 Relations Between the LD-MDCT and TDAC-MDCT

Consider the forward and backward LD-MDCT block transforms defined, respectively, by (8.10) and (8.14). Relations between the forward/backward LD-MDCT and forward/backward TDAC-MDCT block transforms have been discussed in [13, 14]. Specifically, rewriting the argument of cosine transform kernel in the forward/backward LD-MDCT given by (8.10)/(8.14) as

$$\cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}-N\right)\right] = (-1)^k \sin\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}\right)\right], \tag{8.17}$$

leads to mapping the forward/backward LD-MDCT into the forward/backward TDAC modified discrete sine transform (TDAC-MDST), respectively, as

$$c_k^{\text{LD}} = -(-1)^k \sum_{n=0}^{N-1}(x_n - x_{N+n}) \sin\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}\right)\right], \quad k = 0, 1, \ldots, \frac{N}{2}-1, \tag{8.18}$$

$$\hat{x}_n^{\text{LD}} = -\sum_{k=0}^{\frac{N}{2}-1}(-1)^k c_k^{\text{LD}} \sin\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}\right)\right], \quad n = 0, 1, \ldots, N-1. \tag{8.19}$$

Based on the relation between the TDAC-MDST and TDAC-MDCT [28], the TDAC-MDST may be subsequently converted to the TDAC-MDCT. Then, the analytical expressions for the forward and backward LD-MDCT computation based on the TDAC-MDCT are, respectively, given by [13, 14]

$$c_{\frac{N}{2}-1-k}^{\text{LD}} = (-1)^{\frac{N}{2}-k} \sum_{n=0}^{N-1}(-1)^{\frac{N}{4}+n}(x_n - x_{N+n}) \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}\right)\right], \quad k = 0, 1, \ldots, \frac{N}{2}-1, \tag{8.20}$$

$$\hat{x}_n^{\text{LD}} = (-1)^{\frac{N}{4}+1+n} \sum_{k=0}^{\frac{N}{2}-1}(-1)^{\frac{N}{2}-1-k} c_{\frac{N}{2}-1-k}^{\text{LD}} \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}\right)\right], \quad n = 0, 1, \ldots, N-1. \tag{8.21}$$

Equation (8.21) is obtained simply by substituting $\frac{N}{2}-1-k$ for k into (8.19). Since the term $\frac{N}{2}$ in Eqs. (8.20) and (8.21) is always even, then both equations can be rewritten, respectively, in the simplified forms as

$$c^{\text{LD}}_{\frac{N}{2}-1-k} = (-1)^k \sum_{n=0}^{N-1} (-1)^{\frac{N}{4}+n} \, (x_n - x_{N+n}) \, \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}\right)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2}-1, \tag{8.22}$$

$$\hat{x}^{\text{LD}}_n = (-1)^{\frac{N}{4}+n} \sum_{k=0}^{\frac{N}{2}-1} (-1)^k \, c^{\text{LD}}_{\frac{N}{2}-1-k} \, \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}\right)\right],$$
$$n = 0, 1, \ldots, N-1. \tag{8.23}$$

It can be seen that this approach involves reverse operations and sign changes with respect to both the time and frequency indices depending on whether $\frac{N}{4}$ is either even or odd.

8.3.2 *Simplified Relations Between the LD-MDCT and TDAC-MDCT*

Again, consider the forward and backward LD-MDCT block transforms defined, respectively, by (8.10) and (8.14). The key for derivation of simplified relations between the forward/backward LD-MDCT and the forward/backward TDAC-MDCT block transforms is an observation that by substituting $N-1-n$ for n into (8.10) we get [12]

$$c^{\text{LD}}_k = \sum_{n=0}^{N-1} (x_{N-1-n} - x_{2N-1-n}) \, \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}\right)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2}-1, \tag{8.24}$$

and the forward LD-MDCT given by (8.10) is immediately converted to the forward TDAC-MDCT of $\{x_{N-1-n} - x_{2N-1-n}\}$. Similarly, by substituting $N-1-n$ for n into (8.14) we get [12]

$$\hat{x}^{\text{LD}}_{N-1-n} = \sum_{k=0}^{\frac{N}{2}-1} c^{\text{LD}}_k \, \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}\right)\right], \quad n = 0, 1, \ldots, N-1, \tag{8.25}$$

and the backward LD-MDCT given by (8.14) is also immediately converted to the backward TDAC-MDCT of $\{c_k^{\text{LD}}\}$, but the time domain aliased data sequence $\{\hat{x}_n^{\text{LD}}\}$ is in reverse order. Equations (8.24) and (8.25) define simplified relations between the forward/backward LD-MDCT and the forward/backward TDAC-MDCT.

Let us investigate the simplified relations between LD-MDCT and TDAC-MDCT block transforms in their equivalent matrix forms. Let the cosine transform kernel of the forward LD-MDCT in (8.10) be represented by an $\frac{N}{2} \times N$ matrix $\boldsymbol{C}^{\text{LD}}_{\frac{N}{2}\times N}$, and the cosine transform kernel of the forward TDAC-MDCT in (8.15) be represented by an $\frac{N}{2} \times N$ matrix $\boldsymbol{C}^{\text{TDAC}}_{\frac{N}{2}\times N}$. Then, $\boldsymbol{C}^{\text{TDAC}}_{\frac{N}{2}\times N}$ is related to $\boldsymbol{C}^{\text{LD}}_{\frac{N}{2}\times N}$ by

$$\boldsymbol{C}^{\text{TDAC}}_{\frac{N}{2}\times N} = \boldsymbol{C}^{\text{LD}}_{\frac{N}{2}\times N} \times (-\boldsymbol{J}_N), \tag{8.26}$$

where $\boldsymbol{J}_N$ is the reflection matrix of order N. Equation (8.26) implies that the corresponding basis vectors (rows) of forward LD-MDCT and TDAC-MDCT matrices are reverse to each other, and their elements have opposite signs. For clarity, in the following the matrices $\boldsymbol{C}^{\text{LD}}_{\frac{N}{2}\times N}$ and $\boldsymbol{C}^{\text{TDAC}}_{\frac{N}{2}\times N}$ are, respectively, shown in explicit forms for $N = 8$:

$$\boldsymbol{C}^{\text{LD}}_{4\times 8} = \begin{pmatrix} \cos\frac{3\pi}{16} & \cos\frac{\pi}{16} & \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{7\pi}{16} & -\cos\frac{7\pi}{16} & -\cos\frac{5\pi}{16} \\ -\cos\frac{7\pi}{16} & \cos\frac{3\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{7\pi}{16} & -\cos\frac{\pi}{16} & -\cos\frac{5\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{\pi}{16} \\ -\cos\frac{\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{5\pi}{16} & -\cos\frac{\pi}{16} & \cos\frac{7\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{7\pi}{16} \\ -\cos\frac{5\pi}{16} & \cos\frac{7\pi}{16} & \cos\frac{7\pi}{16} & -\cos\frac{5\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \cos\frac{\pi}{16} & -\cos\frac{3\pi}{16} \end{pmatrix},$$

$$\boldsymbol{C}^{\text{TDAC}}_{4\times 8} = \begin{pmatrix} \cos\frac{5\pi}{16} & \cos\frac{7\pi}{16} & -\cos\frac{7\pi}{16} & -\cos\frac{5\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\cos\frac{3\pi}{16} \\ -\cos\frac{\pi}{16} & -\cos\frac{5\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{\pi}{16} & \cos\frac{7\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & \cos\frac{7\pi}{16} \\ \cos\frac{7\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{7\pi}{16} & \cos\frac{\pi}{16} & -\cos\frac{5\pi}{16} & -\cos\frac{5\pi}{16} & \cos\frac{\pi}{16} \\ \cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \cos\frac{\pi}{16} & -\cos\frac{3\pi}{16} & \cos\frac{5\pi}{16} & -\cos\frac{7\pi}{16} & -\cos\frac{7\pi}{16} & \cos\frac{5\pi}{16} \end{pmatrix}.$$

By transposing (8.26) we obtain the relation between matrices of the backward LD-MDCT given by (8.14) and backward TDAC-MDCT given by (8.16) as

$$\boldsymbol{C}^{\text{TDAC}}_{N\times\frac{N}{2}} = -\boldsymbol{J}_N \times \boldsymbol{C}^{\text{LD}}_{N\times\frac{N}{2}}. \tag{8.27}$$

Thus, by simple sign changes and order reversing applied to data sequences the forward/backward LD-MDCT is converted to the forward/backward TDAC-MDCT. The simplified relations between the forward/backward LD-MDCT and the forward/backward TDAC-MDCT block transforms given by (8.24)/(8.25) are simpler, more straightforward, and more transparent compared to (8.20)/(8.21) or (8.22)/(8.23).

8.4 Efficient Implementations of the LD-MDCT

Although the use of LD-MDCT substantially reduces the algorithmic delays, the transform operations in the AAC-ELD codec are still computationally intensive, and as a motivation, the LD-MDCT filter banks need to have fast algorithms, and in particular, when the block length is a composite number, i.e., $N = 960$. Several efficient implementations of the analysis and synthesis LD-MDCT filter banks have been developed up to now in [12–14]. The fast LD-MDCT algorithms in [13, 14] are exclusively based on the fast TDAC-MDCT. In [12], exploiting the simplified relations between the forward/backward LD-MDCT and forward/backward TDAC-MDCT, the improved fast TDAC-MDCT-based LD-MDCT algorithms as well as fast LD-MDCT algorithms without mapping the LD-MDCT to TDAC-MDCT are described. All these fast algorithms are just discussed and investigated in detail in this section.

In general, the relations between the forward/backward LD-MDCT and the forward/backward TDAC-MDCT block transforms provide the unified approach to efficiently implement both the LD-MDCT and TDAC-MDCT via a fast TDAC-MDCT computational structure. Moreover, since the AAC-LC, HE-AAC [2, 3], and AAC-LD [4–6] audio codecs use the TDAC-MDCT, the TDAC-MDCT-based fast LD-MDCT algorithms provide the unified efficient implementation of LD-MDCT and TDAC-MDCT transforms in all four codecs: AAC-ELD, AAC-LD, HE-AAC, and AAC-LC.

8.4.1 TDAC-MDCT-Based Fast LD-MDCT Algorithms

The relations between the forward/backward LD-MDCT and the forward/backward TDAC-MDCT defined by (8.20)/(8.21) or (8.22)/(8.23) enable us to derive fast algorithms for the LD-MDCT computation based on the TDAC-MDCT [13, 14]. They are presented in the following subsections.

8.4.1.1 TDAC-MDCT-Based Fast Forward LD-MDCT Algorithm

Consider the forward LD-MDCT block transform given by (8.22) expressed in the form:

$$c^{\text{LD}}_{\frac{N}{2}-1-k} = (-1)^k \sum_{n=0}^{N-1} u_n \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}\right)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2}-1, \tag{8.28}$$

where

$$u_n = (-1)^{\frac{N}{4}+n}(x_n - x_{N+n}), \qquad n = 0, 1, \ldots, N-1. \tag{8.29}$$

To eliminate the shift factor $+\frac{N}{2}$ in (8.28), by applying the following permutation to the data sequence $\{u_n\}$ [27]

$$y_n = \begin{cases} -u_{\frac{3N}{4}+n}, & n = 0, 1, \ldots, \frac{N}{4}-1, \\ u_{n-\frac{N}{4}}, & n = \frac{N}{4}, \frac{N}{4}+1, \ldots, N-1, \end{cases} \tag{8.30}$$

and using a symmetry property of the cosine transform kernel, the forward LD-MDCT given by (8.28) is reduced to

$$c^{\text{LD}}_{\frac{N}{2}-1-k} = (-1)^k \sum_{n=0}^{\frac{N}{2}-1} (y_n - y_{N-1-n}) \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2}-1. \tag{8.31}$$

The transform kernel in Eq. (8.31) is recognized as an $\frac{N}{2}$-point forward type-IV DCT (DCT-IV) [29] of $\{y_n - y_{N-1-n}\}$. Combining Eqs. (8.30) and (8.31) we obtain:

$$c^{\text{LD}}_{\frac{N}{2}-1-k} = (-1)^k \sum_{n=0}^{\frac{N}{2}-1} y_n \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right], \quad k = 0, 1, \ldots, \frac{N}{2}-1, \tag{8.32}$$

where

$$y_n = \begin{cases} -u_{\frac{3N}{4}+n} - u_{\frac{3N}{4}-1-n}, & n = 0, 1, \ldots, \frac{N}{4}-1, \\ u_{n-\frac{N}{4}} - u_{\frac{3N}{4}-1-n}, & n = \frac{N}{4}, \frac{N}{4}+1, \ldots, \frac{N}{2}-1, \end{cases} \tag{8.33}$$

and $\{u_n\}$ is defined by (8.29). However, the data sequence $\{y_n\}$ defined by (8.33) in terms of $\{u_n\}$ can be derived in terms of the original data sequence $\{x_n\}$. Indeed, combining Eqs. (8.29) and (8.33), the fast TDAC-MDCT-based algorithm for the N-point forward LD-MDCT computation is defined as

$$c^{\mathrm{LD}}_{\frac{N}{2}-1-k} = (-1)^k \sum_{n=0}^{\frac{N}{2}-1} (-1)^n \, y_n \, \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{8.34}$$

where

$$y_n = \begin{cases} -x_{\frac{3N}{4}+n} + x_{\frac{7N}{4}+n} + x_{\frac{3N}{4}-1-n} - x_{\frac{7N}{4}-1-n}, & n = 0, 1, \ldots, \frac{N}{4} - 1, \\ x_{n-\frac{N}{4}} - x_{\frac{3N}{4}+n} + x_{\frac{3N}{4}-1-n} - x_{\frac{7N}{4}-1-n}, & n = \frac{N}{4}, \frac{N}{4}+1, \ldots, \frac{N}{2} - 1, \end{cases} \tag{8.35}$$

8.4.1.2 TDAC-MDCT-Based Fast Backward LD-MDCT Algorithm

Now, consider the backward LD-MDCT block transform given by (8.21). To eliminate the shift factor $+\frac{N}{2}$ in (8.21), substituting subsequently $\frac{N}{4}+n$, $\frac{N}{4}-1-n$, $\frac{3N}{4}+n$ and $\frac{3N}{4}-1-n$ for $n = 0, 1, \ldots, \frac{N}{4}-1$, into (8.21), and using the relation (C.17) from Appendix C.2 we, respectively, have

$$\hat{x}^{\mathrm{LD}}_{\frac{N}{4}+n} = (-1)^n \sum_{k=0}^{\frac{N}{2}-1} (-1)^{\frac{N}{2}-1-k} \, c^{\mathrm{LD}}_{\frac{N}{2}-1-k} \, (-1)^k \, \sin\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right]$$
$$= -(-1)^n \sum_{k=0}^{\frac{N}{2}-1} (-1)^{\frac{N}{2}-1-k} \, c^{\mathrm{LD}}_{\frac{N}{2}-1-k}$$
$$\cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right] = -\, y_{\frac{N}{2}-1-n},$$

$$\hat{x}^{\mathrm{LD}}_{\frac{N}{4}-1-n} = (-1)^n \sum_{k=0}^{\frac{N}{2}-1} (-1)^{\frac{N}{2}-1-k} \, c^{\mathrm{LD}}_{\frac{N}{2}-1-k} \, (-1)^k \, \sin\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right]$$
$$= -(-1)^n \sum_{k=0}^{\frac{N}{2}-1} (-1)^{\frac{N}{2}-1-k} \, c^{\mathrm{LD}}_{\frac{N}{2}-1-k}$$
$$\cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right] = -\, y_{\frac{N}{2}-1-n},$$

$$\hat{x}^{\mathrm{LD}}_{\frac{3N}{4}+n} = (-1)^n \sum_{k=0}^{\frac{N}{2}-1} (-1)^{\frac{N}{2}-1-k}\, c^{\mathrm{LD}}_{\frac{N}{2}-1-k} \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right] = y_n,$$

$$\hat{x}^{\mathrm{LD}}_{\frac{3N}{4}-1-n} = -(-1)^n \sum_{k=0}^{\frac{N}{2}-1} (-1)^{\frac{N}{2}-1-k}\, c^{\mathrm{LD}}_{\frac{N}{2}-1-k} \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right] = -y_n,$$

$$n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{8.36}$$

where the data sequence $\{y_n\}$ actually corresponds to an $\frac{N}{2}$-point inverse DCT-IV of $\{(-1)^{\frac{N}{2}-1-k}\, c^{\mathrm{LD}}_{\frac{N}{2}-1-k}\}$ as follows

$$y_n = (-1)^n \sum_{k=0}^{\frac{N}{2}-1} (-1)^{\frac{N}{2}-1-k}\, c^{\mathrm{LD}}_{\frac{N}{2}-1-k} \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right],$$

$$= (-1)^{n+1} \sum_{k=0}^{\frac{N}{2}-1} (-1)^{k}\, c^{\mathrm{LD}}_{\frac{N}{2}-1-k} \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right],$$

$$n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{8.37}$$

We note that considering the relation (C.17), in two expressions corresponding to $\{\hat{x}^{\mathrm{LD}}_{\frac{N}{4}+n}\}$ and $\{\hat{x}^{\mathrm{LD}}_{\frac{N}{4}-1-n}\}$ on the right-hand sides of (8.36) the following simple trigonometric identity is valid:

$$(-1)^k \sin\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right] = \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right],$$

$$\text{for } \frac{N}{2} - 1 - n.$$

Equation (8.36) also indicates that $\{\hat{x}^{\mathrm{LD}}_n\}$, $n = 0, 1, \ldots, N-1$, has the following local symmetries:

$$\hat{x}^{\mathrm{LD}}_{\frac{N}{4}+n} = \hat{x}^{\mathrm{LD}}_{\frac{N}{4}-1-n}, \quad \hat{x}^{\mathrm{LD}}_{\frac{3N}{4}+n} = -\hat{x}^{\mathrm{LD}}_{\frac{3N}{4}-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{8.38}$$

Then, the time domain aliased data sequence $\{\hat{x}^{\mathrm{LD}}_n\}$ is recovered as

$$\hat{x}^{\mathrm{LD}}_{\frac{N}{4}+n} = \hat{x}^{\mathrm{LD}}_{\frac{N}{4}-1-n} = -y_{\frac{N}{2}-1-n},$$

$$\hat{x}^{\mathrm{LD}}_{\frac{3N}{4}+n} = y_n, \quad \hat{x}^{\mathrm{LD}}_{\frac{3N}{4}-1-n} = -y_n, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{8.39}$$

Equations (8.37) and (8.39) define the fast TDAC-MDCT-based algorithm for the backward LD-MDCT computation, whereby N is divisible by 4. Finally, the complete data sequence $\{\hat{x}_n^{\mathrm{LD}}\}$ with the length $2N$ is obtained according to (8.13).

We note that if we would alternatively consider the backward LD-MDCT block transform given by (8.23), then in the derivation procedure of fast backward LD-MDCT algorithm we can simply use the identity $(-1)^{\frac{N}{2}-1-k} = -(-1)^k$ in (8.36). It has been actually used in Eq. (8.37).

TDAC-MDCT-based fast algorithms for the forward N-point LD-MDCT computation given by (8.34) and (8.35), and the backward LD-MDCT computation given by (8.37) and (8.39) rely on a fast $\frac{N}{2}$-point DCT-IV computational structure, and involve reverse operations and sign changes with respect to both the time and frequency indices.

8.4.2 *Improved TDAC-MDCT-Based Fast LD-MDCT Algorithms*

As an alternative, the simplified relations between the forward/backward LD-MDCT and the forward/backward TDAC-MDCT defined by (8.24) and (8.25) enable us to improve versions of fast TDAC-MDCT-based algorithms for the forward and backward LD-MDCT computation presented in the previous subsections without reverse operations and sign changes with respect to both the time and frequency indices [12]. They are presented in the following subsections.

8.4.2.1 Improved TDAC-MDCT-Based Fast Forward LD-MDCT Algorithm

Consider the forward LD-MDCT block transform given by (8.24) expressed in the form:

$$c_k^{\mathrm{LD}} = \sum_{n=0}^{N-1} u_n \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1+\frac{N}{2}\right)\right], \quad k=0,1,\ldots,\frac{N}{2}-1, \tag{8.40}$$

where

$$u_n = x_{N-1-n} - x_{2N-1-n}, \qquad n = 0,1,\ldots,N-1. \tag{8.41}$$

Following the same derivation procedure defined by (8.30)–(8.33), and finally combining (8.33) and (8.41) we get

$$c_k^{\mathrm{LD}} = \sum_{n=0}^{\frac{N}{2}-1} y_n \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right], \qquad k = 0,1,\ldots,\frac{N}{2}-1, \tag{8.42}$$

where

$$y_n = \begin{cases} -x_{\frac{N}{4}-1-n} + x_{\frac{5N}{4}-1-n} - x_{\frac{N}{4}+n} + x_{\frac{5N}{4}+n}, & n = 0, 1, \ldots, \frac{N}{4}-1, \\ x_{\frac{5N}{4}-1-n} - x_{\frac{9N}{4}-1-n} - x_{\frac{N}{4}+n} + x_{\frac{5N}{4}+n}, & n = \frac{N}{4}, \frac{N}{4}+1, \ldots, \frac{N}{2}-1, \end{cases} \tag{8.43}$$

Equations (8.42) and (8.43) define the improved fast TDAC-MDCT-based algorithm for the N-point forward LD-MDCT computation, whereby N is divisible by 4.

This DCT-IV-based fast algorithm is well known in the theory of fast TDAC-MDCT algorithms, and it has been used for the efficient implementation of the TDAC-MDCT in MP3 audio coding standard [27].

8.4.2.2 Improved TDAC-MDCT-Based Fast Backward LD-MDCT Algorithm

Now, consider the backward LD-MDCT block transform given by (8.25). Again, to eliminate the shift factor $+\frac{N}{2}$ in (8.25), substituting subsequently $\frac{N}{4}+n$, $\frac{N}{4}-1-n$, $\frac{3N}{4}+n$ and $\frac{3N}{4}-1-n$ for $n = 0, 1, \ldots, \frac{N}{4}-1$, into (8.25), and using the relation (C.12) from Appendix C.2 we, respectively, have

$$\begin{aligned} \hat{x}^{\mathrm{LD}}_{\frac{N}{4}+n} &= -\sum_{k=0}^{\frac{N}{2}-1} c_k^{\mathrm{LD}} \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right] = -y_n, \\ \hat{x}^{\mathrm{LD}}_{\frac{N}{4}-1-n} &= -\sum_{k=0}^{\frac{N}{2}-1} c_k^{\mathrm{LD}} \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right] = -y_n, \\ \hat{x}^{\mathrm{LD}}_{\frac{3N}{4}+n} &= \sum_{k=0}^{\frac{N}{2}-1} c_k^{\mathrm{LD}} (-1)^k \sin\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right] = y_{\frac{N}{2}-1-n}, \\ \hat{x}^{\mathrm{LD}}_{\frac{3N}{4}-1-n} &= -\sum_{k=0}^{\frac{N}{2}-1} c_k^{\mathrm{LD}} (-1)^k \sin\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right] = -y_{\frac{N}{2}-1-n}, \\ & n = 0, 1, \ldots, \frac{N}{4}-1. \end{aligned} \tag{8.44}$$

The data sequence $\{y_n\}$ in (8.44) corresponds to an $\frac{N}{2}$-point inverse DCT-IV of $\{c_k^{\mathrm{LD}}\}$ as follows

$$y_n = \sum_{k=0}^{\frac{N}{2}-1} c_k^{\mathrm{LD}} \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right], \qquad n = 0, 1, \ldots, \frac{N}{2}-1, \tag{8.45}$$

and the time domain aliased data sequence $\{\hat{x}_n^{\mathrm{LD}}\}$ having the local symmetry properties given by (8.38) is recovered as

$$\hat{x}_{\frac{N}{4}+n}^{\mathrm{LD}} = \hat{x}_{\frac{N}{4}-1-n}^{\mathrm{LD}} = -\,y_n,$$

$$\hat{x}_{\frac{3N}{4}+n}^{\mathrm{LD}} = y_{\frac{N}{2}-1-n}, \quad \hat{x}_{\frac{3N}{4}-1-n}^{\mathrm{LD}} = -\,y_{\frac{N}{2}-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{8.46}$$

Equations (8.45) and (8.46) define the improved fast TDAC-MDCT-based algorithm for the backward LD-MDCT computation, whereby N is divisible by 4. The complete data sequence $\{\hat{x}_n^{\mathrm{LD}}\}$ with the length $2N$ is obtained according to (8.13).

The improved TDAC-MDCT-based fast algorithms for the forward N-point LD-MDCT computation given by (8.42) and (8.43), and the backward LD-MDCT computation given by (8.45) and (8.46) rely again on a fast $\frac{N}{2}$-point DCT-IV computational structure, but without reverse operations and sign changes with respect to both the time and frequency indices.

8.4.3 *Fast LD-MDCT Algorithms Without Mapping to TDAC-MDCT*

Fast algorithms for the forward/backward LD-MDCT computation presented in the previous subsections are based on relations between the forward/backward LD-MDCT and forward/backward TDAC-MDCT. However, this assumption is not necessary. In the following subsections fast algorithms for the forward and backward LD-MDCT computation are presented without mapping the LD-MDCT to TDAC-MDCT.

8.4.3.1 Fast Forward LD-MDCT Algorithm

Consider the forward LD-MDCT block transform given by (8.10) expressed in the form:

$$c_k^{\mathrm{LD}} = \sum_{n=0}^{N-1} v_n \cos\left[\frac{\pi}{2N}(2k+1)\left(2n+1-\frac{N}{2}\right)\right], \quad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{8.47}$$

where

$$v_n = x_{N+n} - x_n, \qquad n = 0, 1, \ldots, N-1. \tag{8.48}$$

Obviously, to eliminate the shift factor $-\frac{N}{2}$ in (8.47), by applying the following permutation to the data sequence $\{v_n\}$

$$y_n = \begin{cases} v_{\frac{N}{4}+n}, & n = 0, 1, \ldots, \frac{3N}{4} - 1, \\ -v_{n-\frac{3N}{4}}, & n = \frac{3N}{4}, \frac{3N}{4} + 1, \ldots, N - 1, \end{cases} \tag{8.49}$$

and using a symmetry property of the cosine transform kernel, the forward LD-MDCT given by (8.47) is reduced to

$$c_k^{\text{LD}} = \sum_{n=0}^{\frac{N}{2}-1} (y_n - y_{N-1-n}) \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right], \quad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{8.50}$$

The transform kernel in Eq. (8.50) is recognized as an $\frac{N}{2}$-point forward DCT-IV of $\{y_n - y_{N-1-n}\}$. Combining Eqs. (8.49) and (8.50) we obtain:

$$c_k^{\text{LD}} = \sum_{n=0}^{\frac{N}{2}-1} y_n \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right], \quad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{8.51}$$

where

$$y_n = \begin{cases} v_{\frac{N}{4}+n} + v_{\frac{N}{4}-1-n}, & n = 0, 1, \ldots, \frac{N}{4} - 1, \\ v_{\frac{N}{4}+n} - v_{\frac{5N}{4}-1-n}, & n = \frac{N}{4}, \frac{N}{4} + 1, \ldots, \frac{N}{2} - 1. \end{cases} \tag{8.52}$$

and $\{v_n\}$ is defined by (8.48). Finally, combining Eqs. (8.48) and (8.52) we get

$$y_n = \begin{cases} x_{\frac{5N}{4}+n} - x_{\frac{N}{4}+n} + x_{\frac{5N}{4}-1-n} - x_{\frac{N}{4}-1-n}, & n = 0, 1, \ldots, \frac{N}{4} - 1, \\ x_{\frac{5N}{4}+n} - x_{\frac{N}{4}+n} - x_{\frac{9N}{4}-1-n} + x_{\frac{5N}{4}-1-n}, & n = \frac{N}{4}, \frac{N}{4} + 1, \ldots, \frac{N}{2} - 1. \end{cases} \tag{8.53}$$

Equations (8.51) and (8.53) define the fast algorithm for the N-point forward LD-MDCT computation (N is divisible by 4), but without mapping the forward LD-MDCT to forward TDAC-MDCT.

8.4.3.2 Fast Backward LD-MDCT Algorithm

Now consider the backward LD-MDCT block transform given by (8.14). Then, substituting subsequently $\frac{N}{4} + n$, $\frac{N}{4} - 1 - n$, $\frac{3N}{4} + n$ and $\frac{3N}{4} - 1 - n$ into (8.14) and using the relation (C.12) leads to exactly the same Eq. (8.44) and to the fast backward LD-MDCT algorithm defined by Eqs. (8.45) and (8.46).

The fast algorithms for the forward N-point LD-MDCT given by (8.51) and (8.53), and the backward LD-MDCT computation given by (8.45) and (8.46) are based on a fast $\frac{N}{2}$-point DCT-IV computational structure with simple pre- and post-processing of data sequences. Since the fast TDAC-MDCT-based algorithms are also based on a fast $\frac{N}{2}$-point DCT-IV computational structure, this fact provides the unified approach for efficient implementation of both the even-length TDAC-MDCT and LD-MDCT transforms via one identical fast DCT-IV computational structure.

8.5 Computational Complexity and Comparison of Fast LD-MDCT Algorithms

For a given value of N being in general, an even integer (divisible by 4), the computational complexity of fast analysis LD-MDCT filter bank is given by the complexity of: Windowing procedure requiring $\frac{15N}{8}$ multiplications, converting the N-point forward LD-MDCT (or forward TDAC-MDCT) block transform to the $\frac{N}{2}$-point DCT-IV by applying the permutation described in (8.35), (8.43) or (8.53) requiring $\frac{11N}{8}$ additions (it is taken into account the fact that the low delay analysis windowing function has the first $\frac{N}{8}$ values implicitly equal to zero, i.e., $w_n^{(a)} = 0 \Rightarrow x_n = 0, n = 0, 1, \ldots, \frac{N}{8} - 1$), and the complexity of a fast $\frac{N}{2}$-point DCT-IV computational structure. On the other hand, the computational complexity of fast synthesis LD-MDCT filter bank is given by the complexity of: Fast $\frac{N}{2}$-point DCT-IV computational structure, windowing procedure requiring $\frac{15N}{8}$ multiplications, and the complexity of overlapping/add procedure given by (8.3) requiring $\frac{11N}{8}$ additions.

Since the AAC-ELD defines the block length to be $2N$ with $N = 1024$ or $N = 960$ [1], we need in general a fast even-length DCT-IV computational structure. For the block length $N = 1024$ being a power of two, a 2^n-length fast DCT-IV computational structure which combines theoretical efficiency with a very regular structure, and achieving the lowest multiplicative complexity is based on the complex FFT of half size [30, 33]. It is presented in Appendix C.2.1. In particular, for the block length $N = 960$ which is a composite number of the form $2^m \times q,\ m > 1$, where q is an odd positive integer, we need an even-length fast DCT-IV computational structure. Such even-length fast DCT-IV computational structure [28] is presented in Appendix C.2.2. Because the block length $960 = 2^6 \times 15$, $q = 15$ is the composite number we additionally need an efficient 15-point type-II/III DCT (DCT-II/III) module. The required efficient 15-point PFA DCT-II/III module is presented in Appendix D.7, or alternatively, the efficient 15-point WFTA DCT-II/III module is presented in Appendix D.8.

8.5.1 DCT-IV-Based Fast LD-MDCT Algorithms

Essentially, all the fast LD-MDCT (and TDAC-MDCT too) algorithms rely on the fast $\frac{N}{2}$-point DCT-IV computational structure which is identical both for the analysis and synthesis LD-MDCT (TDAC-MDCT) filter banks.

When $M = \frac{N}{2} = 2^m,\ m > 1$, ($M = 512 = 2^9$ for the LD-MDCT in AAC-ELD), the arithmetic complexity of fast analysis/synthesis LD-MDCT filter banks is given by

$$M_{2^m} = \frac{M}{2}(m+2) + \frac{15M}{4}, \qquad A_{2^m} = \frac{3M}{2}m + \frac{11M}{4}, \tag{8.54}$$

where M_{2^m} denotes the number of multiplications and A_{2^m} denotes the number of additions.

When $M = \frac{N}{2} = 2^m \times 15,\ m > 1$, is a composite number ($M = 480 = 2^5 \times 15$ for the LD-MDCT in AAC-ELD), taking into account the arithmetic complexity of efficient 15-point DCT-II/III module, the arithmetic complexity of fast analysis/synthesis LD-MDCT filter banks is given by

$$\begin{aligned} M_{2^m \times 15} &= 2^m \times M_{15}^{II} + \frac{M}{2}(m+2) + \frac{15M}{4}, \\ A_{2^m \times 15} &= 2^m \times A_{15}^{II} + \frac{M}{2}(3m+2) - 2^m + \frac{11M}{4}, \end{aligned} \tag{8.55}$$

where M_{15}^{II} denotes the number of multiplications and A_{15}^{II} denotes the number of additions of the efficient 15-point DCT-II/III module.

8.5.2 DCT-IV/DCT-II-Based Fast LD-MDCT Algorithms

We recall that the $\frac{N}{2}$-point DCT-IV can be converted to the DCT-II of the same size at the cost of $\frac{N}{2}$ additional multiplications and $\frac{N}{2} - 1$ recursive additions [32]. These multiplications may be simply absorbed into the windowing operation, thus further reducing the multiplicative complexity (saving $\frac{N}{2}$ multiplications) of a fast DCT-IV/DCT-II-based computational structure (see Appendix C.3). However, this optimized approach requires modifying the low delay analysis/synthesis windowing function and the fast DCT-IV/DCT-II-based computational structure has to be inverted at the decoder.

When $M = \frac{N}{2} = 2^m,\ m > 1$, ($M = 512$ for the LD-MDCT in AAC-ELD), assuming that the fast recursive DCT-II/III computational structure [32] (see also Appendix C.1.1) achieving the lowest multiplicative complexity is used, the arithmetic complexity of fast analysis/synthesis LD-MDCT filter banks is given by

$$M_{2^m} = \frac{M}{2}m + \frac{15M}{4}, \qquad A_{2^m} = \frac{3M}{2}m + \frac{11M}{4}. \tag{8.56}$$

When $M = \frac{N}{2} = 2^m \times 15,\ m > 0$ is a composite number ($M = 480$ for the LD-MDCT in AAC-ELD), taking into account the arithmetic complexity of efficient

15-point DCT-II/III module, the arithmetic complexity of fast analysis/synthesis LD-MDCT filter banks is given by

$$M_{2^m\times 15} = 2^m \times M_{15}^{II} + \frac{M}{2}\, m + \frac{15M}{4},$$
$$A_{2^m\times 15} = 2^m \times A_{15}^{II} + \frac{3M}{2}\, m - 2^m + \frac{15M}{4}. \tag{8.57}$$

8.5.3 Comparison of Fast LD-MDCT Algorithms

Comparison of fast LD-MDCT algorithms in terms of arithmetic complexity for $N = 1024$ and 960 and structural simplicity are shown in Tables 8.1 and 8.2. In Table 8.1 the fast LD-MDCT algorithms are implemented via the fast DCT-IV computational structure, and in Table 8.2 the fast LD-MDCT algorithms are implemented via the fast DCT-II/III computational structure. From Tables 8.1 and 8.2 one can see that all fast LD-MDCT algorithms have the same arithmetic complexity, but improved TDAC-MDCT-based fast algorithms as well as ones without mapping the LD-MDCT to TDAC-MDCT do not require reverse operations and sign changes with respect to the time and frequency indices.

Table 8.1 Comparison of fast LD-MDCT algorithms based on the fast DCT-IV computational structure in terms of arithmetic complexity for $N = 1024$ and 960 and structural simplicity (M denotes multiplication, A denotes addition)

LD-MDCT algorithm ($M = \frac{N}{2}$)	Fast analysis/synthesis LD-MDCT filter bank		Reverse operations and sign changes?
	$N = 1024$	$N = 960$	
TDAC-MDCT-based	4736 M and 8320 A	4024 M and 7512 A	Yes
Improved TDAC-MDCT-based	4736 M and 8320 A	4024 M and 7512 A	No
Without mapping to TDAC-MDCT	4736 M and 8320 A	4024 M and 7512 A	No

Table 8.2 Comparison of fast LD-MDCT algorithms based on the fast DCT-II/III computational structure in terms of arithmetic complexity for $N = 1024$ and 960 and structural simplicity (M denotes multiplication, A denotes addition)

Algorithm ($M = \frac{N}{2}$)	Fast analysis/synthesis LD-MDCT filter bank		Reverse operations and sign changes?
	$N = 1024$	$N = 960$	
TDAC-MDCT-based	4224 M and 8320 A	3544 M and 7512 A	Yes
Improved TDAC-MDCT-based	4224 M and 8320 A	3544 M and 7512 A	No
Without mapping to TDAC-MDCT	4224 M and 8320 A	3544 M and 7512 A	No

The direct fast mixed-radix TDAC-MDCT algorithms for composite lengths $2^m \times q^n$, $m,\ n > 0$, with $q = 3, 5$ and 9 have been developed in [15, 16]. However, in order to be used for the composite length $960 = 2^6 \times 15 = 2^5 \times 30$, an efficient 30-point TDAC-MDCT module has to be derived.

8.6 Discussion and Consequences of Fast LD-MDCT Algorithms

The fast even-length DCT-IV and DCT-IV/DCT-II fast computational structures together with the efficient 15-point PFA or WFTA DCT-II/III modules (see Appendices D.7 and D.8) discussed in this chapter and optimized efficient 5-point DCT-II/III modules (see Appendix D.7) provide a compact, modular, and flexible transform block building system which has an impact on an efficient implementation of the even-length TDAC-MDCT filter banks used in other existing audio broadcasting standards [19, 20] and speech communication codecs [17, 21–26], where the length of data block is a composite number of the form $2^m \times q$, whereby $q = 5$ or 15. By a composition of proper regular fast computational structure/module(s), the efficient implementation of the even-length TDAC-MDCT can be achieved as follows:

- The advanced Digital Audio Broadcasting (DAB+) system [20] for digital radio services as well as the Digital Radio Mondiale (DRM) [19], universal openly standardized digital broadcasting system, have adopted the HE-AAC codec [2, 3] which uses the TDAC-MDCT filter bank with the length $N = 1920$ (long block) or with $N = 240$ (short block). Since both lengths are composite numbers, i.e., $1920 = 2^7 \times 15$ and $240 = 2^4 \times 15$, the composition of fast even-length DCT-IV computational structure with the permutation defined by (8.33) for $x_n = u_n$, and 15-point PFA or WFTA DCT-II/III module provides the efficient implementation of forward/backward TDAC-MDCT in the DAB+ and DRM systems. According to (C.32) with $M = \frac{N}{2}$, $m = 6$ or 3, $q = 15$, $M_{15}^{II} = 17$, and $A_{15}^{II} = 67$, the arithmetic complexity of forward TDAC-MDCT computation is 4928 and 14,784 additions for the long block, and 436 multiplications and 1308 additions for the short block.

  When the fast even-length DCT-II/III computational structure is used, according to (C.8) the arithmetic complexity of forward TDAC-MDCT computation is 3968 and 13,825 additions for the long block, and 316 multiplications and 1189 additions for the short block. In both cases, the backward TDAC-MDCT with overlap/add operation requires exactly the same number of arithmetic operations.
- Issued ITU-T G.722.1 [21], ITU-T G.722.1C [25], G.719 [26], G.718 and G.729.1 [23] speech codecs, 3GPP2 EVRC-WB [17] vocoder, and ITU-T G.EV-VBR standard [22, 24] have adopted the TDAC-MDCT filter bank (actually the modulated lapped transform with associated sine windowing function [33]) with the length $N = 640$. Compared to ITU-T G.722.1, in the ITU-T G.722.1C speech

codec the transform length is doubled to $N = 1280$. Since block lengths are again composite numbers, i.e., $640 = 2^7 \times 5$ and $1280 = 2^8 \times 5$, the composition of fast even-length DCT-IV computational structure with the permutation defined by (8.33) for $x_n = u_n$, and efficient 5-point DCT-II/III modules provides the efficient implementation of forward/backward TDAC-MDCT in speech communications codecs. The required efficient 5-point DCT-II/III modules are presented in Appendix D.7. According to (C.32) with $M = \frac{N}{2}$, $m = 6$, $q = 5$, $M_5^{II} = 4$, and $A_5^{II} = 13$, the arithmetic complexity of the forward 640-point TDAC-MDCT is 1536 multiplications and 4288 additions. The arithmetic complexity of the forward 1280-point TDAC-MDCT is 3392 multiplications and 9536 additions.

When the fast even-length DCT-II/III computational structure is used, according to (C.8) the arithmetic complexity of forward TDAC-MDCT computation is 1216 multiplications and 3969 additions. In both cases, the backward TDAC-MDCT with overlap/add operation requires exactly the same number of arithmetic operations. The associated modified sine windowing function which is piecewise symmetric is shown in [18].

8.7 Summary

Definitions of the analysis/synthesis LD-MDCT (and TDAC-MDCT) filter banks, general symmetry properties of LD-MDCT block transforms both in the time and frequency domains, relations between the LD-MDCT and TDAC-MDCT transforms both in the analytical forms and in the equivalent matrix representations, and efficient implementations of the even-length analysis/synthesis LD-MDCT filter banks have been presented. For each fast LD-MDCT algorithm the complete formulae are derived. Relations between the LD-MDCT and TDAC-MDCT transforms enable us to map the LD-MDCT into TDAC-MDCT block transform. This fact provides the unified approach to efficiently implement both the LD-MDCT and TDAC-MDCT block transforms in state-of-the-art MPEG audio codecs. All the fast even-length LD-MDCT algorithms, TDAC-MDCT-based and without mapping the LD-MDCT to TDAC-MDCT, have been investigated and compared in terms of arithmetic complexity and structural simplicity.

Problems and Exercises

1. Following the procedure used in Eq. (8.9), reduce the $2N$-point forward LD-MDCT block transform given by (8.11) to the N-point forward LD-MDCT.
2. Verify the even anti-symmetry property of LD-MDCT frequency coefficients $\{c_k^{\text{LD}}\}$ given by (8.12).

3. Verify the symmetry property of time domain aliased data sequence $\{\hat{x}_n^{\text{LD}}\}$ given by (8.13).
4. Consider the forward LD-MDCT block transform defined by (8.10). Verify the simplified relation between the forward LD-MDCT and the forward TDAC-MDCT block transforms given by (8.24).
5. Consider the backward LD-MDCT block transform defined by (8.14). Verify the simplified relation between the backward LD-MDCT and the backward TDAC-MDCT block transforms given by (8.25).
6. Following the procedure used in Eq. (8.36), derive the fast TDAC-MDCT-based algorithm for backward LD-MDCT computation from Eq. (8.23).
7. Implement by a computer program the TDAC-MDCT-based fast forward LD-MDCT algorithm defined by (8.34) and (8.35), and TDAC-MDCT-based fast backward LD-MDCT algorithm defined by (8.37) and (8.39) for $N = 2^m,\ m > 2$. Note that in the efficient implementation you can use either a fast 2^m-length DCT-IV computational structure or a fast 2^m-length DCT-IV/DCT-II computational structure.
8. Implement by a computer program the improved TDAC-MDCT-based fast forward LD-MDCT algorithm defined by (8.42) and (8.43), and improved TDAC-MDCT-based fast backward LD-MDCT algorithm defined by (8.45) and (8.46) for $N = 2^m,\ m > 2$.
9. Implement by a computer program the fast forward LD-MDCT algorithm defined by (8.51) and (8.53), and fast backward LD-MDCT algorithm defined by (8.45) and (8.46) for $N = 2^m,\ m > 2$.
10. Compare all fast LD-MDCT algorithms (see 7, 8, and 9 above) in terms of structural simplicity.
11. Implement by a computer program the efficient 15-point PFA DCT-II/III module represented by the signal flow graph shown in Fig. D.1 in Appendix D.7. Thereby you can verify its correctness and also you will need it for the efficient implementation of fast LD-MDCT algorithms for $N = 960$.
12. Implement by a computer program the efficient 15-point WFTA DCT-II/III module represented by the signal flow graph shown in Fig. D.2 in Appendix D.8. Similarly, you can verify its correctness and you will need it for the alternative efficient implementation of fast LD-MDCT algorithms for $N = 960$.
13. Implement by a computer program the TDAC-MDCT-based fast forward LD-MDCT algorithm defined by (8.34) and (8.35), and TDAC-MDCT-based fast backward LD-MDCT algorithm defined by (8.37) and (8.39) for the composite length $N = 2^m \times 15,\ m > 0$. In the efficient implementation you can use either a fast even-length DCT-IV computational structure or a fast even-length DCT-IV/DCT-II computational structure. The required efficient 15-point PFA DCT-II/III module is presented in Appendix D.7, or alternatively, the efficient 15-point WFTA DCT-II/III module is presented in Appendix D.8.

14. Implement by a computer program the improved TDAC-MDCT-based fast forward LD-MDCT algorithm defined by (8.42) and (8.43), and improved TDAC-MDCT-based fast backward LD-MDCT algorithm defined by (8.45) and (8.46) for the composite length $N = 2^m \times 15,\ m > 0$. In the implementation use the efficient 15-point PFA DCT-II/III module, or alternatively, the efficient 15-point WFTA DCT-II/III module.
15. Implement by a computer program the fast forward LD-MDCT algorithm defined by (8.51) and (8.53), and fast backward LD-MDCT algorithm defined by (8.45) and (8.46) for the composite length $N = 2^m \times 15,\ m > 0$. In the implementation use the efficient 15-point PFA DCT-II/III module, or alternatively, the efficient 15-point WFTA DCT-II/III module.
16. Compare all fast LD-MDCT algorithms (see 13, 14, and 15 above) in terms of structural simplicity.
17. The direct fast mixed-radix TDAC-MDCT algorithms for composite lengths $2^m \times q^n$, $m,\ n > 0$, with $q = 3, 5$, and 9 have been developed in [15, 16]. In order to be used for the composite length $960 = 2^6 \times 15 = 2^5 \times 30$, an efficient 30-point TDAC-MDCT module has to be derived. Try to derive the efficient 30-point TDAC-MDCT module. Then, the composition of 2^m-length fast TDAC-MDCT algorithm [16] and the efficient 30-point TDAC-MDCT module provides the efficient implementation of the TDAC-MDCT (LD-MDCT) for composite lengths $N = 2^m \times 30,\ m > 0$. Implement by a computer program the improved TDAC-MDCT-based fast forward LD-MDCT algorithm defined by (8.42) and (8.43) based on the fast mixed-radix TDAC-MDCT algorithm. Finally, investigate its arithmetic complexity and compare with previous efficient implementations.
18. Based on the fast even-length DCT-IV (or DCT-IV/DCT-II) computational structure with permutation defined by (8.33) for $x_n = u_n$, and 15-point PFA or WFTA DCT-II/III module, implement the forward/backward TDAC-MDCT computation for the composite lengths $N = 1920$ and 240.
19. Implement by a computer program the efficient 5-point DCT-II/III modules presented in Appendix D.7. You will need them for the efficient implementation of fast TDAC-MDCT algorithms for $N = 640$.
20. Based on the fast even-length DCT-IV (or DCT-IV/DCT-II) computational structure with permutation defined by (8.33) for $x_n = u_n$, and efficient 5-point DCT-II/III modules (see Appendix D.7), implement the forward/backward TDAC-MDCT computation for the composite length $N = 640$.

References

1. Information Technology – Coding of Audio-Visual Objects – Part 3: Audio, Amendment 9: Enhanced Low Delay AAC, ISO/IEC 14496-3:2005/FDAM 9:2007(E), N9499, Shenzhen, October 2007

MPEG-4 AAC-LC and HE-AAC Audio Coding Standards

2. Information Technology – Coding of Audio-Visual Objects – Part 3: Audio, Subpart 4: General Audio Coding (GA)-AAC, TwinVQ, BSAC, ISO/IEC 14496–3:2005(E) (2005)
3. M. Wolters, K. Kjörling, D. Homm, H. Purnhagen, A closer look into MPEG-4 high efficiency AAC, in *115th AES Convention*, New York, NY, October 2003. Preprint #5871

MPEG-4 AAC-LD Audio Coding Standard

4. E. Allamanche, R. Geiger, J. Herre, T. Sporer, MPEG-4 low delay audio coding based on the AAC codec, in *106th AES Convention*, Munich, May 1999. Preprint #4929
5. J. Hilpert et al., Real-time implementation of the MPEG-4 low delay advanced audio coding algorithm (AAC-LD) on Motorola DSP56300, in *108th AES Convention*, Paris, February 2000. Preprint #5081
6. M. Lutzky, M. Schnell, M. Schmidt, R. Geiger, Structural analysis of low latency audio coding schemes, in *119th AES Convention*, New York, NY, October 2005. Preprint #6601

MPEG-4 AAC-ELD Audio Coding Standard

7. M. Lutzky, M.L. Valero, M. Schnell, J. Hilpert, AAC-ELD v2 – the new state of the art in high quality communication audio coding, in *131st AES Convention*, New York, NY, October 2011. Preprint #8516
8. M. Schnell et al., Enhanced MPEG-4 low delay AAC – low bitrate high quality communication, in *122nd AES Convention*, Vienna, May 2007. Preprint #6998
9. M. Schnell et al., Low delay filter banks for enhanced low delay audio coding, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, October 2007, pp. 235–238
10. M. Schnell et al., MPEG-4 enhanced low delay AAC – a new standard for high quality communication, in *125th AES Convention*, San Francisco, CA, October 2008. Preprint #7503
11. M.L. Valero et al., A new parametric stereo and multichannel extension for MPEG-4 enhanced low delay AAC (AAC-ELD), in *128th AES Convention*, London, May 2010. Preprint #8099

Efficient Implementations of LD-MDCT Filter Banks

12. V. Britanak, New fast algorithms for the low delay MDCT computation in the MPEG-4 AAC enhanced low delay audio coding standard. Signal Process. **105**(12), 410–418 (2014)
13. R.K. Chivukula, Y.A. Reznik, V. Devarajan, Efficient algorithms for MPEG-4 AAC-ELD, AAC-LD and AAC-LC filter banks, in *Proceedings of the IEEE International Conference on Audio, Language and Image Processing, (ICALIP'2008)*, Shanghai, July 2008, pp. 1629–1634
14. R.K. Chivukula, Y.A. Reznik, Y. Hu, V. Devarajan, Fast algorithms for low-delay TDAC filter banks in MPEG-4 AAC-ELD. IEEE Trans. Audio Speech Lang. Process. **22**(12), 1701–1712 (2014)
15. Z.G. Gui, Y. Ge, D.Y. Zhang, J.S. Wu, Generalized fast mixed-radix algorithm for the computation of forward and inverse MDCTs. Signal Process. **92**(2), 363–373 (2012)
16. J. Wu, H. Shu, L. Senhadji, L. Luo, Mixed-radix algorithm for the computation of forward and inverse MDCTs. IEEE Trans. Circuits Syst. I: Regul. Pap. **56**(4), 784–794 (2009)

Existing Audio Broadcasting and Speech Communication Codecs

17. 3GGP2 C.S0014-C v1.0, Enhanced variable rate codec, speech service Option 3, 68 and 70 for wide-band spread spectrum digital systems, 2007
18. R.K. Chivukula, Y.A. Reznik, Efficient implementation of a class of MDCT/IMDCT filter-banks for speech and audio coding, in *Proceedings of the IEEE ICASSP'2008*, Las Vegas, NV, March–April 2008, pp. 213–216
19. Digital Radio Mondiale (DRM): System Specification, ETSI ES 201 980 v3.1.1 (2009–08), ETSI Standard, August 2009 (available at website www.drm.org)
20. W. Hoeg, T. Lauterbach (eds.), *Digital Audio Broadcasting: Principles and Applications of DAB, DAB+ and DMB*, chap. 3, 3rd edn. (Wiley, Chichester, 2009), pp. 93–165
21. ITU-T Recommendation G.722.1 Annex C, Low-complexity coding at 24 and 32 kbit/s for hands-free operation in systems with low frame loss, Annex C: 14 kHz at 24, 32 and 48 kbit/s, May 2005
22. ITU-T SG16 Q9 – Contribution 199: extended high-level description of the Q9 EV-VBR baseline codec, June 2007
23. L. Laaksonen et al., Super wide–band extension of G.718 and G.729.1 speech codec, in *Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010*, Makuhari, September 2010
24. T. Vaillancourt et al., ITU-T EV-VBR: a robust 8–32 kbit/s scalable coder for error prone telecommunication channels, in *Proceedings of the 16th European Signal Processing Conference*, Lausanne, August 2008
25. M. Xie, D. Lindbergh, P. Chu, From ITU-T G.722.1 to ITU-T G.722.1 Annex C: a new low-complexity 14 kHz bandwidth audio coding standard, in *Proceedings of the IEEE ICASSP'2006*, vol. 5, Toulouse, May 2006, pp. 173–176, also published in J. Multimed. **2**(2), 65–76 (2007)
26. M. Xie, P. Chu, A. Taleb, M. Briand, ITU-T G.719: a new low-complexity full-band (20 kHz) audio coding standard for high quality conversational applications, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA'2009)*, New Paltz, NY, October 2009, pp. 265–268

Supporting Literature

27. V. Britanak, Survey of efficient MDCT implementations in MP3 audio coding standard: retrospective and state-of-the-art. Signal Process. **91**(4), 1379–1394 (2011)
28. V. Britanak, H.J. Lincklaen Arriëns, Fast computational structures for an efficient implementation of the complete TDAC analysis/synthesis MDCT/MDST filter banks. Signal Process. **89**(7), 1379–1394 (2009)
29. V. Britanak, P. Yip, K.R. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations* (Academic Press Inc., Elsevier Science, Amsterdam, 2007)
30. R. Gluth, Regular FFT-related transform kernels for DCT/DST-based polyphase filter banks, in *Proceedings of the IEEE ICASSP'91*, Toronto, May 1991, pp. 2205–2208
31. R. Gluth, A unified approach to transform-based FIR filter banks with special regard to perfect reconstruction systems, in *Proceedings of the IEEE ICASSP'93*, vol. III, Minneaopolis, MN, April 1993, pp. 157–160
32. C.W. Kok, Fast algorithm for computing discrete cosine transform. IEEE Trans. Signal Process. **45**(3), 757–760 (1997)
33. H.S. Malvar, *Signal Processing with Lapped Transforms*, chap. 2 (Artech House, Norwood, MA, 1992), pp. 71–75

34. J.P. Princen, A.W. Johnson, A.B. Bradley, Subband/transform coding using filter bank designs based on time domain aliasing cancellation, in *Proceedings of the IEEE ICASSP'87*, Dallas, TX, April 1987, pp. 2161–2164
35. G.D.T. Schuller, M.J.T. Smith, New framework for modulated perfect reconstruction filter banks. IEEE Trans. Signal Process. **44**(8), 1941–1954 (1996)
36. G.D.T. Schuller, T. Karp, Modulated filter banks with arbitrary system delay: efficient implementations and time-varying case. IEEE Trans. Signal Process. **48**(3), 737–748 (2000)

Chapter 9
Integer Approximate Cosine/Sine-Modulated Filter Banks

9.1 Introduction

Digital image/video/audio signals to be processed are quantized into M-bit representations and hence, are integer-valued. Specifically, for digital audio signals $M = 16$, 21 or $M = 24$ for high amplitude resolution signals. But the perfect reconstruction cosine/sine-modulated filter banks and cosine-modulated QMF banks are real-valued transforms which map integer signal into real-valued spectral coefficients. Although their fast algorithms reduce the computational complexity, due to floating-point finite-length representation and corresponding rounding-off errors, they cannot be used for lossless audio coding. Indeed, almost all modern perceptual audio coding schemes developed so far operate in floating-point arithmetic and therefore, are lossy in nature [78]. However, some audio coding applications require completely lossless preservation of the audio signal. In general, there exist at least three approaches to lossless audio coding. The first approach is using the linear (adaptive) prediction coding methods [2, 5, 32], and the second is transform-based perceptual coding approach [4, 8]. Third approach to realize the transform-based lossless audio coding is using a number theoretic transform such as the discrete Mersenne and Fermat transforms which are inherently integer transforms [1]. For large transform sizes, however, the number theoretic transforms require a large number of bits to represent the spectral coefficients. Moreover, the modulo arithmetic has to be employed [83].

Due to achieving the high compression ratios and relatively high-quality decoded audio of lossy codecs, further research has been focused to design a transform-based perceptual audio coding scheme extended to lossless coding. The basic principles of transform-based perceptual lossless audio coding scheme have been suggested in [4, 6–8]. Having a lossy perceptual audio codec as a core codec, in the encoder the input audio signal is transformed into frequency domain by a forward transform, and spectral coefficients are quantized (rounded to the nearest integers). Duplicating the decompression steps at the decoder side, i.e., applying

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1_9

the inverse transform to integer spectral coefficients followed by dequantization results in a residual error defined as the difference between the reconstructed and original signal. This residual error is fed into a lossless bit-conversion module for lossless coding, and finally is transmitted along with entropy coded integer spectral coefficients. In the decoder, the original reconstructed signal is obtained by adding signal reconstructed from lossy compression and lossless coded residual error provided that the implementation of inverse transform and dequantization is identical to that of the encoder.

All perfect reconstruction cosine/sine-modulated and cosine-modulated QMF filter banks in the existing international audio/speech coding standards and proprietary audio compression algorithms are used for larger transform sizes, typically $N = 64, 128, 256, 512, 1024, 2048, 4096$ for 2^n-lengths, or $N = 240, 640, 960, 1280, 1920$ for mixed-radix (composite) lengths. In this book it has been shown that the fast algorithms/computational structures for their efficient implementation are based on the discrete sinusoidal unitary transforms of reduced sizes, specifically, discrete cosine transforms of types II, III and IV (DCT-II, DCT-III, DCT-IV), and discrete Fourier Transform (DFT) or its FFT algorithm. An enabling technology for transform-based lossless audio coding is the integer transform. Generally, integer transform is a transform which maps integers to integers by a reversible (invertible) way so that it preserves all mathematical properties of the original real-valued transform such as perfect reconstruction, energy compaction property, and fast algorithm. Indeed, the integer modified discrete cosine transform (IntMDCT) or integer modulated lapped transform (IntMLT) enabled to design and implement this innovative coding technology for scalable lossy to lossless audio coding [68, 71]. The latest emerged MPEG standards for lossless audio coding support two formats: MPEG-4 Audio Lossless Coding (MPEG-4 ALS) [3, 5], MPEG-4 Scalable Lossless Coding (MPEG-4 SLS) [63, 68], and MPEG-4 High-Definition Scalable Advanced Audio Coding (MPEG-4 HD-AAC/SLS) [59]. ALS based on linear predictive coding supports only lossless compression whereas SLS and HD-AAC/SLS support a scalable to lossy compression in which the bit stream includes an AAC lossy representation for fast transcoding [32].

In this chapter, the local and global methods to integer approximation of perfect reconstruction cosine/sine-modulated filter banks and cosine-modulated QMF banks are discussed in detail. They are based on computational methods of linear algebra, matrix theory, and matrix computations, and in particular, on (block) matrix decompositions. In fact, the scalar and block matrix decompositions are powerful mathematical tools to construct the reversible (invertible) integer transforms. Discussed approximation methods enable to construct the complete integer analysis/synthesis filter banks for designing/building a transform-based lossless audio coding scheme. In general, any integer filter bank can be constructed as long as has a fast algorithm. Selected basic material from linear algebra and matrix theory which is fundamental for understanding approximation methods is presented in Appendix A.

9.2 Integer Transforms and Integer Filter Banks

Basic theory and methods developed for the integer approximation of discrete sinusoidal orthogonal (unitary) transforms, such as the DFT, the discrete Hartley transform (DHT), discrete cosine, and sine transforms (DCT/DST), are well described in [9, 12–14, 16–18, 21, 23–27]. In general, the discrete sinusoidal transform to be integer approximated is represented by the corresponding orthogonal (unitary) matrix or alternatively, by a rotation-based (recursive) sparse block matrix factorization of the transform matrix. Principally, there are two basic approaches to construct the integer transform [9]:

- By directly replacing the real-valued elements of the transform matrix by M-bit integers or by dyadic rational numbers so that all its mathematical properties are preserved. However, this approach is rather suitable for small transform sizes, typically for $N = 4,\ 8$ and 16. Indeed, the 8- point integer DCT (IntDCT) is used in the state-of-the-art international image/video coding standards [11, 15]. For larger transform sizes when $N > 16$, which are typically used in the international audio coding standards, the complexity of derivation of integer transform increases significantly, and hence the transform size is the main limiting factor of this approach to integer approximation.
- In the theory of fast algorithms a rotation-based (recursive) sparse block matrix factorization of the transform matrix defines a fast computational structure for its efficient implementation. Then a given transform can be approximated and implemented by a reversible integer-to-integer mapping or by a reversible integer transform as follows: Each 2×2 Givens–Jacobi rotation matrix is factored into a product of the so-called Gauss elementary matrices being the unit lower and upper triangular matrices (see Appendix F.2). The factored unit triangular matrices define computational structures for the efficient computation of 2×2 Givens–Jacobi rotations. For the reversible (invertible) integer approximation of the Givens–Jacobi rotation all off-diagonal elements in the factored unit triangular matrices are approximated by dyadic rational numbers (dyadic approximation), or alternatively, the computation is realized in the floating-point arithmetic followed immediately by the rounding operator (rounding to the nearest integer) applied to components of a rotated vector. However, the dyadic approximation as well as the approximation by rounding operator introduce an approximation error in each step. Although this approach offers simple and elegant solution to reversible (invertible) integer approximation of the transform, the resulting approximation error accumulated through the stages of fast computational structure becomes considerable for large transform sizes. Again, this approach is rather suitable for small transform sizes.

A comprehensive overview of integer approximation methods for the short-length DCT can be found in the book [9], for the short-length FFT in [23–27], and an overview of the latest research results in the short-length DCT approximations can be found in [11, 20].

In general, there are three essential aspects which we must take into account to the integer approximation of a complete analysis/synthesis filter bank in comparison with the integer approximation of the discrete sinusoidal unitary (orthogonal) transform:

- While the discrete sinusoidal transform is represented by the corresponding unitary (orthogonal) matrix or alternatively, by a rotation-based (recursive) sparse block matrix factorization of the transform matrix, the complete analysis and synthesis filter banks consist of several procedures which have to be integer approximated separately. Specifically, the analysis filter bank consists of the windowing procedure (multiplying the original audio data block by a real-valued windowing function), the time domain aliasing (TDA) procedure followed by a forward orthogonal block transform to compute the frequency coefficients. On the other hand, the synthesis filter bank consists of the inverse orthogonal block transform, and the windowing and overlap and add procedure or the time domain aliasing cancellation (TDAC) procedure to reconstruct the original audio data block.
- Almost all perfect reconstruction cosine/sine-modulated and cosine-modulated QMF filter banks in the existing international audio/speech coding standards and proprietary audio compression algorithms are used for larger transform sizes, typically $N = 64,\ 128,\ 256,\ 512,\ 1024,\ 2048,\ 4096$ for 2^n-lengths, or $N = 240,\ 640,\ 960,\ 1280,\ 1920$ for mixed-radix (composite) lengths. Perhaps the only exception is the MLT filter bank (or equivalently the MDCT filter bank associated with the sine windowing function), which is used in the MP3 audio coding standard with the mixed-radix lengths $N = 12$ (short block) or $N = 36$ (long block).
- All perfect reconstruction cosine/sine-modulated and cosine-modulated QMF filter banks discussed in this book are always converted to the orthogonal/orthonormal DCT-IV or DCT-II/III transforms of reduced sizes, providing thus their efficient implementation. Therefore, the design of fast DCT-IV and DCT-II/III computational structures for radix-2 and in particular, for mixed-radix (composite) lengths is very important for the integer approximation of filter banks in terms of the approximation accuracy and computational efficiency for lossless audio coding.

9.2.1 Desired Properties of Integer Transforms

In order to develop a progressive to lossless audio codec from a conventional lossy perceptual audio codec, in general, the key step is to replace the real-valued filter bank with its reversible (invertible) integer counterpart followed by replacing the entropy coding module with a lossless embedded entropy coding module (the quantization step is eliminated) [37, 38, 45, 46]. An ideal or optimal reversible (invertible) integer transform for lossless audio coding should preserve all mathematical properties of the original real-valued transform and should have the following properties [37, 38, 45, 46]:

1. **Integer-to-integer mapping**: The integer transform maps integer signal to integer frequency coefficients by one-to-one correspondence, i.e., each input value is mapped into one and only one output value, and vice versa.
2. **Perfect reconstruction**: By applying the forward and backward integer transforms the integer signal is exactly reconstructed from the integer frequency coefficients without any error.
3. **Constant data volume**: The dynamic range of output data is the same as that of the input data.
4. **Energy compaction property**: The integer transform compacts most of energy to a low number of frequency coefficients.
5. **Fast algorithm**: The forward and backward integer transforms are efficiently implemented by a fast computational structure.
6. **Minimal difference between the exact and approximated spectral values**: The frequency coefficients of integer transform should be as close as possible to those of the original real-valued transform.

Properties (1) and (2) ensure that the integer transform is indeed reversible. The property (3) requires that the reversible integer transform be designed with reference to a concept of normalized transform. The normalized transform is associated with an orthogonal sparse (recursive) matrix factorization of a transform matrix $\boldsymbol{T}$, or more exactly, it is associated with the unit determinant of transform matrix $\boldsymbol{T}$. A valid reversible integer transform represented by the transform matrix $\boldsymbol{T}$ cannot have determinant $\det(\boldsymbol{T}) < 1$, because such a transform will compact the data, i.e., multiple input integer values will be mapped to one output integer value, which contradicts the reversibility property. On the other hand, a reversible integer transform represented by the transform matrix $\boldsymbol{T}$ having determinant $\det(\boldsymbol{T}) > 1$ will expand the input data set by a factor of $|\det(\boldsymbol{T})|$ in the output data set. Therefore, a desired property of the reversible integer transform is that $\det(\boldsymbol{T}) = 1$ [46]. The property (4) ensures the high coding gain for audio data compression. The property (5) guarantees the minimal computational complexity of the integer transform. Finally, the property (6) implies that the difference between the exact and approximated frequency coefficients defined as a quantization noise is minimal.

As a summary, all properties (1)–(6) imply that the performance of integer transform should be as close as possible to that of the original real-valued transform [37, 46].

9.2.2 Normalized (Integer) Transform

When a transform is applied to a data block it has to be normalized to preserve the energy of input signal in the frequency domain. In the case of DCT-II, DCT-III and DCT-IV the normalization factor is $\sqrt{2/N}$ [9], and in the case of DFT the normalization factor is $1/\sqrt{N}$, where N is order of the matrix [41]. Then such normalized DCT-II, DCT-III, and DCT-IV matrices for $N > 2$ are orthonormal and their deter-

minants are unity. Such matrices are also called the unit matrices [9]. The normalized DFT matrix is symmetric and unitary and its determinant is a complex number whose modulus is one [26]. The concept of normalized transform is based on an orthogonal (recursive) sparse block matrix factorization of the transform matrix. For the normalized transform all factored matrices including butterfly ones have to be orthogonal. Orthogonal butterfly matrices are actually trivial rotation matrices with the rotation angle $\frac{\pi}{4}$. Consequently, no final normalization is needed and the dynamic range of the output frequency coefficients is the same as for the input signal. The orthogonal (recursive) sparse block matrix factorization of the transform matrix implies that the determinant of transform matrix is one. However, the existence of orthogonal butterflies in a corresponding fast computational structure increases the total computational complexity of the normalized transform [27, 46].

Similarly, if the corresponding integer transform is used for lossless coding applications, it has to be normalized. Naturally, the normalized integer transform is also associated with the orthogonal (recursive) sparse block matrix factorization of the transform matrix. Due to realization of orthogonal butterflies by trivial rotation matrices with the rotation angle $\frac{\pi}{4}$, the error caused by integer approximation increases too. In the case of long transforms this approximation error, i.e., the difference between exact and approximated transform derived in some measure such as the minimum square error (MSE), becomes considerable. The total accumulated approximation error propagated through the stages of fast computational structure has a significant impact on the lossless coding efficiency [27, 46].

9.2.3 Quality Measures of Integer Transforms for Coding

The reversible (invertible) integer transforms for lossless audio coding have two quality measures of coding [36, 41, 51]:

- **The approximation error**: Since integer transforms map the integer signal to integer frequency coefficients they are only approximations of the original real-valued transforms. The approximation error results from an accumulation of rounding errors in the approximated trivial and Givens–Jacobi rotations. As a result, the coding efficiency and hence compression ratio is limited by the approximation error, especially at higher frequencies, where audio signal usually has a very low energy.
- **The computational complexity**: This is a very important measure because in audio coding applications large size transforms are used, and in general, the computational complexity of integer transform is higher than that of the corresponding original real-valued transform.

An important factor for the approximation error is the total number of rounding operations applied to approximated rotations in succeeding stages of a fast computational structure. Every rounding operation introduces a rounding error which is interpreted as an approximation error of integer transform. Even though

this approximation error can be canceled out by the backward transform, the approximation error is accumulated and spreads in the frequency domain. Since all spectral coefficients have to be encoded this accumulated approximation error has an impact on lossless coding efficiency. Specifically, for the high frequency range, where audio signals usually contain a rather small amount of energy, the approximation error can be larger than the actual signal. The impact is more critical for larger transform sizes. Consequently, this fact becomes the main limiting factor for lossless coding efficiency [41, 51].

We recall that the discrete transforms in their fast computational structures contain butterflies being the trivial rotations with elements $\pm$ 1. For integer transforms these trivial rotations have to be modified since they result in an expansion of range of data. They are, therefore realized by the rotations with angle $\frac{\pi}{4}$. This is a reason, why the computational complexity of integer transforms is higher than that of the corresponding original real-valued transforms. Therefore, in order to improve the performance of reversible integer transform with the associated fast computational structure it is desirable to minimize as much as possible [36]:

- The approximation error by minimizing the total number of rounding operations,
- Computational complexity.

9.2.4 Orthogonal Recursive Sparse Block Matrix Factorizations

We mentioned that the normalized integer transform is associated with the orthogonal (recursive) sparse block matrix factorization of the transform matrix. Since all perfect reconstruction cosine/sine-modulated and cosine-modulated QMF filter banks discussed in this book are always converted to the orthogonal DCT-IV or DCT-II/III transforms of reduced sizes, in the following subsections are presented orthogonal (recursive) sparse block matrix factorizations of DCT-II, DCT-III, and DCT-IV matrices which we will need for the integer approximation of complete analysis/synthesis filter banks. In addition, since the DCT-IV of the length N can be implemented via the complex DFT of half size with identical pre- and post-rotation stages [82], the unitary recursive sparse block matrix factorization of the DFT matrix is also included.

9.2.4.1 Orthogonal Recursive Sparse Block Factorization of DCT-II/III Matrices

The orthogonal recursive sparse block matrix factorization of the DCT-II matrix of order N denoted by $\boldsymbol{C}_N^{II}$ with a scaling factor of $\sqrt{2}$ has been reported in [84]. For $N = 2^m,\ m > 1$, the scaled matrix $\boldsymbol{C}_N^{II}$ can be factored into the following product of orthogonal block matrices [9, 84], pp. 97–98:

$$\begin{aligned} \boldsymbol{C}_N^{II} &= \boldsymbol{P}_N^T \begin{pmatrix} \boldsymbol{C}_{\frac{N}{2}}^{II} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_{\frac{N}{2}}^{IV} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{J}_{\frac{N}{2}} \\ \boldsymbol{I}_{\frac{N}{2}} & -\boldsymbol{J}_{\frac{N}{2}} \end{pmatrix} \\ &= \boldsymbol{P}_N^T \begin{pmatrix} \boldsymbol{C}_{\frac{N}{2}}^{II} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_{\frac{N}{2}}^{IV} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{J}_{\frac{N}{2}} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{J}_{\frac{N}{2}} \\ \boldsymbol{J}_{\frac{N}{2}} & -\boldsymbol{I}_{\frac{N}{2}} \end{pmatrix}, \end{aligned} \tag{9.1}$$

where $\boldsymbol{0}s$ are null matrices, $\boldsymbol{I}_{\frac{N}{2}}$ is the identity matrix, $\boldsymbol{J}_{\frac{N}{2}}$ is the reverse ordered identity matrix, $\boldsymbol{C}_{\frac{N}{2}}^{II}$ is the DCT-II matrix, and $\boldsymbol{C}_{\frac{N}{2}}^{IV}$ is the DCT-IV matrix, all of order $\frac{N}{2}$. $\boldsymbol{P}_N$ is a permutation matrix of order N which when applied to a data vector corresponds to the reordering

$$\tilde{x}_n = x_{2n}, \quad \tilde{x}_{\frac{N}{2}+n} = x_{2n+1}, \quad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{9.2}$$

Note that $\boldsymbol{P}_N^T = \boldsymbol{P}_N^{-1}$. All factored block matrices on the right-hand side of (9.1) are orthogonal. Since $[\boldsymbol{C}_N^{II}]^T = [\boldsymbol{C}_N^{II}]^{-1} = \boldsymbol{C}_N^{III}$, by transposition or inversion of (9.1) we obtain the orthogonal recursive sparse block matrix factorization of the inverse scaled DCT-II matrix, the scaled DCT-III matrix $\boldsymbol{C}_N^{III}$.

9.2.4.2 Orthogonal Recursive Sparse Block Factorizations of the DCT-IV Matrix

Now we derive two simple and useful recursive fast DCT-IV algorithms. The first algorithm is derived by the decimation-in-frequency (DIF) decomposition method, and the second one is derived by the decimation-in-time (DIT) decomposition method. Each fast algorithm defines a recursive sparse block matrix factorization of the DCT-IV matrix and this fact enables us to construct the corresponding orthogonal recursive sparse block matrix factorization of the DCT-IV matrix.

Let the unnormalized forward and inverse DCT-IV transforms be defined, respectively, by Eqs. (C.9) and (C.10) in Appendix C.2, and let N being the length of transform be an even integer.

A. Fast DIF Recursive DCT-IV Algorithm

Following the DIF decomposition method used for the derivation of a mixed-radix fast MDCT algorithm [87], consider the following two algebraic expressions:

$$a_k = c_{2k}^{IV} + c_{2k+1}^{IV}, \qquad b_k = c_{2k}^{IV} - c_{2k+1}^{IV}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{9.3}$$

where

$$a_k = \sum_{n=0}^{N-1} x_n \left(\cos\left[\frac{\pi}{4N}(4k+1)(2n+1) \right] + \cos\left[\frac{\pi}{4N}(4k+3)(2n+1) \right] \right),$$

$$b_k = \sum_{n=0}^{N-1} x_n \left(\cos\left[\frac{\pi}{4N}(4k+1)(2n+1) \right] - \cos\left[\frac{\pi}{4N}(4k+3)(2n+1) \right] \right),$$

$$k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{9.4}$$

Using the trigonometric identities for the sum and difference of cosines we get

$$a_k = 2 \sum_{n=0}^{N-1} \left(x_n \cos \frac{\pi(2n+1)}{4N} \right) \cos\left[\frac{\pi}{2N}(2k+1)(2n+1) \right],$$

$$b_k = 2 \sum_{n=0}^{N-1} \left(x_n \sin \frac{\pi(2n+1)}{4N} \right) \sin\left[\frac{\pi}{2N}(2k+1)(2n+1) \right],$$

$$k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{9.5}$$

Substituting $N - 1 - n$ for n into both equations of (9.5) we have

$$a_k = 2 \sum_{n=0}^{\frac{N}{2}-1} \left(x_n \cos \frac{\pi(2n+1)}{4N} - x_{N-1-n} \sin \frac{\pi(2n+1)}{4N} \right) \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1) \right],$$

$$b_k = 2 \sum_{n=0}^{\frac{N}{2}-1} \left(x_n \sin \frac{\pi(2n+1)}{4N} + x_{N-1-n} \cos \frac{\pi(2n+1)}{4N} \right) \sin\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1) \right],$$

$$k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{9.6}$$

The cosine transform kernel in the first equation of (9.6) for $\{a_k\}$ corresponds to an $\frac{N}{2}$-point DCT-IV, while the sine transform kernel in the second equation of (9.6) for $\{b_k\}$ corresponds to an $\frac{N}{2}$-point DST-IV. Exploiting the relation between the DST-IV and the DCT-IV defined by (C.13) in Appendix C.2 we have

$$b_{\frac{N}{2}-1-k} = \sum_{n=0}^{\frac{N}{2}-1}(-1)^n \left(x_n \sin\frac{\pi(2n+1)}{4N} + x_{N-1-n}\cos\frac{\pi(2n+1)}{4N}\right) \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right], \quad k = 0, 1, \ldots, \frac{N}{2}-1. \tag{9.7}$$

Then the complete formulae of fast DIF recursive DCT-IV algorithm are defined as

$$a_k = \sum_{n=0}^{\frac{N}{2}-1} u_n \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right],$$
$$b_{\frac{N}{2}-1-k} = \sum_{n=0}^{\frac{N}{2}-1}(-1)^n v_n \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right], k = 0, 1, \ldots, \frac{N}{2}-1, \tag{9.8}$$

where

$$u_n = x_n \cos\frac{\pi(2n+1)}{4N} - x_{N-1-n}\sin\frac{\pi(2n+1)}{4N},$$
$$v_n = x_n \sin\frac{\pi(2n+1)}{4N} + x_{N-1-n}\cos\frac{\pi(2n+1)}{4N}, \qquad n = 0, 1, \ldots, \frac{N}{2}-1, \tag{9.9}$$

and based on (9.3) the final DCT-IV coefficients are given by

$$c^{IV}_{2k} = a_k + b_k, \qquad c^{IV}_{2k+1} = a_k - b_k, \qquad k = 0, 1, \ldots, \frac{N}{2}-1. \tag{9.10}$$

Thus, the N-point DCT-IV is decomposed into the block of $\frac{N}{2}$ Givens–Jacobi rotations, two $\frac{N}{2}$-point DCTs-IV, and the post-butterfly stage. The decomposition may be performed recursively until the short-length DCTs-IV remain. The fast DCT-IV computational structure derived by the DIF decomposition method for $N = 6$ is shown in Fig. 9.1.

The fast DCT-IV computational structure shown in Fig. 9.1 defines the recursive sparse block factorization of the matrix $\boldsymbol{C}^{IV}_N$ as

$$\boldsymbol{C}^{IV}_N = \boldsymbol{P}_N \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{J}_{\frac{N}{2}} \\ \boldsymbol{J}_{\frac{N}{2}} & -\boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{C}^{IV}_{\frac{N}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}^{IV}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_{\frac{N}{2}}\boldsymbol{J}_{\frac{N}{2}} \end{pmatrix} \boldsymbol{G}_N, \tag{9.11}$$

where $\boldsymbol{G}_N$ is the rotation matrix (Givens–Jacobi rotations) of order N given by

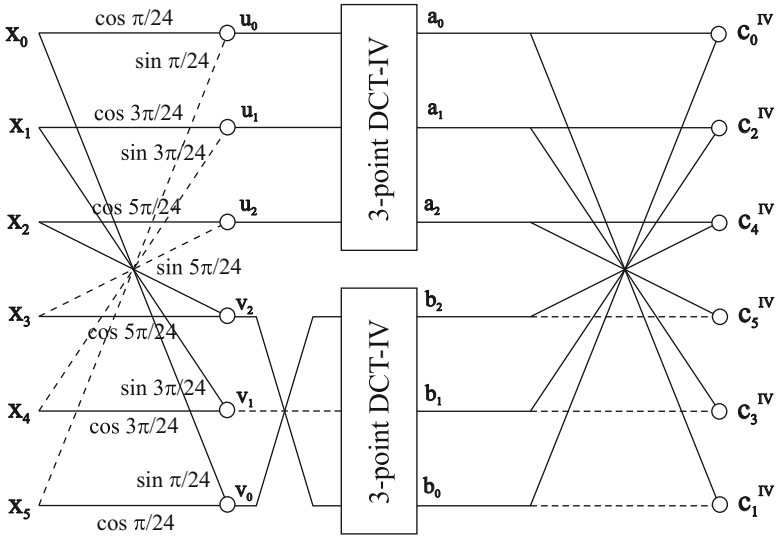

Fig. 9.1 The fast DCT-IV computational structure derived by the DIF decomposition method for $N = 6$

$$\boldsymbol{G}_N = \begin{pmatrix} \cos\frac{\pi}{4N} & & 0 & & -\sin\frac{\pi}{4N} \\ & \cos\frac{3\pi}{4N} & & -\sin\frac{3\pi}{4N} & \\ & & \ddots & & \\ & & \cos\frac{(N-1)\pi}{4N} \; -\sin\frac{(N-1)\pi}{4N} & & \\ 0 & & & & 0 \\ & & \sin\frac{(N-1)\pi}{4N} \; \cos\frac{(N-1)\pi}{4N} & & \\ & & \ddots & & \\ & \sin\frac{3\pi}{4N} & & \cos\frac{3\pi}{4N} & \\ \sin\frac{\pi}{4N} & & 0 & & \cos\frac{\pi}{4N} \end{pmatrix}, \quad (9.12)$$

and $\boldsymbol{D}_{\frac{N}{2}} = \text{diag}\{(-1)^k\}$, $k = 0, 1, \ldots, \frac{N}{2} - 1$ is the diagonal odd-sign changing matrix of order $\frac{N}{2}$. $\boldsymbol{P}_N$ is a permutation matrix of order N which reorders the transform coefficients such that the first half are even-indexed coefficients in natural order, while the second half are odd-indexed coefficients but in reverse order. Then, based on (9.11) and (9.12) the first orthogonal recursive sparse block factorization of the DCT-IV matrix $\boldsymbol{C}_N^{IV}$ is defined as

$$\boldsymbol{C}_N^{IV} = \boldsymbol{P}_N \, \frac{\sqrt{2}}{2} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{J}_{\frac{N}{2}} \\ \boldsymbol{J}_{\frac{N}{2}} & -\boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_{\frac{N}{2}}^{IV} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_{\frac{N}{2}} \boldsymbol{J}_{\frac{N}{2}} \end{pmatrix} \boldsymbol{G}_N. \quad (9.13)$$

B. Fast DIT Recursive DCT-IV Algorithm

Dividing the input data sequence $\{x_n\}$ into two parts, specifically into the even-indexed and odd-indexed samples, we have [74]

$$\begin{aligned} c_k^{IV} &= \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \cos\left[\frac{\pi}{4N}(2k+1)(4n+1)\right] \\ &\quad + \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \cos\left[\frac{\pi}{4N}(2k+1)(4n+3)\right] \\ &= \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \cos\left[\frac{\pi}{4N}(2k+1)[2(2n+1)-1]\right] \\ &\quad + \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \cos\left[\frac{\pi}{4N}(2k+1)[2(2n+1)+1]\right], \\ &\quad k = 0, 1, \ldots, N-1. \end{aligned} \tag{9.14}$$

Using the trigonometric identities for cosines of sum and difference of angles after some algebraic manipulation we get

$$c_k^{IV} = \cos\frac{\pi(2k+1)}{4N}\, a_k + \sin\frac{\pi(2k+1)}{4N}\, b_k, \qquad k = 0, 1, \ldots, N-1, \tag{9.15}$$

where

$$\begin{aligned} a_k &= \sum_{n=0}^{\frac{N}{2}-1} (x_{2n} + x_{2n+1}) \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right], \\ b_k &= \sum_{n=0}^{\frac{N}{2}-1} (x_{2n} - x_{2n+1}) \sin\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right], k = 0, 1, \ldots, N-1. \end{aligned} \tag{9.16}$$

The cosine transform kernel in the first equation of (9.16) for $\{a_k\}$ corresponds to an $\frac{N}{2}$-point DCT-IV of $\{x_{2n} + x_{2n+1}\}$, while the sine transform kernel in the second equation of (9.16) for $\{b_k\}$ corresponds to an $\frac{N}{2}$-point DST-IV of $\{x_{2n} - x_{2n+1}\}$. Similarly, exploiting the relation between the DST-IV and the DCT-IV defined by (C.13) in Appendix C.2 we have

$$b_{\frac{N}{2}-1-k} = \sum_{n=0}^{\frac{N}{2}-1} (-1)^n (x_{2n} - x_{2n+1}) \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{9.17}$$

Substituting $N-1-k$ for k into (9.16) we find that the data sequences $\{a_k\}$ and $\{b_k\}$ have the following symmetry properties

$$a_k = -a_{N-1-k}, \qquad b_k = b_{N-1-k}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{9.18}$$

and substituting $N-1-k$ for k into (9.15) and using (9.18), we finally obtain

$$c^{IV}_{N-1-k} = -\sin\frac{\pi(2k+1)}{4N}\, a_k + \cos\frac{\pi(2k+1)}{4N}\, b_k, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{9.19}$$

Then the complete formulae of fast DIT recursive DCT-IV algorithm are defined as

$$u_n = x_{2n} + x_{2n+1}, \qquad v_n = (-1)^n (x_{2n} - x_{2n+1}), \qquad n = 0, 1, \ldots, \frac{N}{2} - 1, \tag{9.20}$$

and

$$a_k = \sum_{n=0}^{\frac{N}{2}-1} u_n \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right],$$
$$b_{\frac{N}{2}-1-k} = \sum_{n=0}^{\frac{N}{2}-1} v_n \cos\left[\frac{\pi}{4(N/2)}(2k+1)(2n+1)\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{9.21}$$

The final DCT-IV coefficients are given by

$$c^{IV}_k = \cos\frac{\pi(2k+1)}{4N}\, a_k + \sin\frac{\pi(2k+1)}{4N}\, b_k,$$
$$c^{IV}_{N-1-k} = -\sin\frac{\pi(2k+1)}{4N}\, a_k + \cos\frac{\pi(2k+1)}{4N}\, b_k, \quad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{9.22}$$

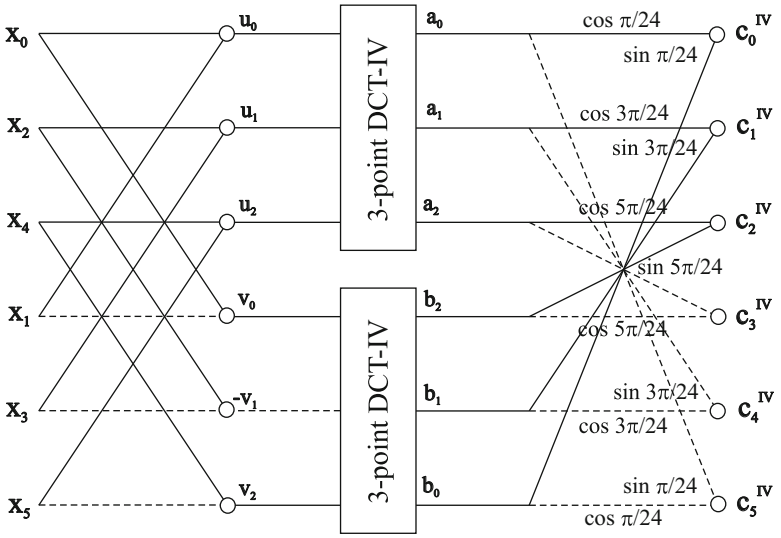

Fig. 9.2 The fast DCT-IV computational structure derived by the DIT decomposition method for $N = 6$

Thus, the N-point DCT-IV is decomposed into the pre-butterfly stage, two $\frac{N}{2}$-point DCTs-IV, and the block of $\frac{N}{2}$ Givens–Jacobi rotations. Again, the decomposition may be performed recursively until the short-length DCTs-IV remain. The fast DCT-IV computational structure derived by the DIT decomposition method for $N = 6$ is shown in Fig. 9.2.

The fast DCT-IV computational structure shown in Fig. 9.2 defines the recursive sparse block factorization of the DCT-IV matrix $\boldsymbol{C}_N^{IV}$ as

$$\boldsymbol{C}_N^{IV} = \boldsymbol{G}_N^T \begin{pmatrix} \boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_{\frac{N}{2}}^{IV} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{I}_{\frac{N}{2}} \\ \boldsymbol{I}_{\frac{N}{2}} & -\boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \boldsymbol{P}_N, \tag{9.23}$$

where $\boldsymbol{G}_N^T$ is the transposed rotation matrix $\boldsymbol{G}_N^T$ defined by (9.12). Note that $\boldsymbol{G}_N^T = \boldsymbol{G}_N^{-1}$. $\boldsymbol{P}_N$ is a permutation matrix of order N defined by (9.2). Then based on (9.23) the second orthogonal recursive sparse block factorization of the matrix $\boldsymbol{C}_N^{IV}$ is defined as

$$\boldsymbol{C}_N^{IV} = \boldsymbol{G}_N^T \begin{pmatrix} \boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_{\frac{N}{2}}^{IV} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_{\frac{N}{2}} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{I}_{\frac{N}{2}} \\ \boldsymbol{I}_{\frac{N}{2}} & -\boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \boldsymbol{P}_N. \tag{9.24}$$

Note 1: Since the DCT-IV matrix is symmetric and self-inverse, i.e., $\boldsymbol{C}_N^{IV} = [\boldsymbol{C}_N^{IV}]^T = [\boldsymbol{C}_N^{IV}]^{-1}$, by transposition or inversion of (9.13) and (9.24) we obtain alternative orthogonal recursive sparse block factorization of the matrix $\boldsymbol{C}_N^{IV}$.

Note 2: The fast DCT-IV computational structures derived by DIF and DIT decomposition methods are identical both for the forward and inverse DCT-IV computation. For the N-point DCT-IV computation their computational complexity is given by

$$M_N^{IV} = 2\, M_{\frac{N}{2}}^{IV} + \frac{3N}{2}, \qquad A_N^{IV} = 2\, A_{\frac{N}{2}}^{IV} + \frac{5N}{2},$$

where M_N^{IV} denotes the number of multiplications and A_N^{IV} denotes the number of additions with the following initial values: For 2^n-lengths $M_2^{IV} = 3$ and $A_2^{IV} = 3$, while generally if N is even, $M_3^{IV} = 1$ and $A_3^{IV} = 6$ and 1 shift for the scaled 3-point DCT-IV (see Appendix D.6).

9.2.4.3 Unitary Recursive Sparse Block Factorization of the DFT Matrix

Let $\boldsymbol{F}_N$ be the DFT matrix of order N with elements $\{1/\sqrt{N}\ e^{-i\frac{2\pi nk}{N}}\}$, where $i = \sqrt{-1}$ and $n,\ k = 0, 1, \ldots, N-1$ [24, 25]. Then $\boldsymbol{F}_N$ can be factored into the following product of unitary block matrices [27, 41]:

$$\boldsymbol{F}_N = \boldsymbol{P}_N \begin{pmatrix} \boldsymbol{F}_{\frac{N}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{F}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_{\frac{N}{2}} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{I}_{\frac{N}{2}} \\ \boldsymbol{I}_{\frac{N}{2}} & -\boldsymbol{I}_{\frac{N}{2}} \end{pmatrix}, \tag{9.25}$$

where $\boldsymbol{P}_N$ is the bit-reversal permutation matrix, $\boldsymbol{F}_{\frac{N}{2}}$ is the DFT matrix of order $\frac{N}{2}$, and $\boldsymbol{W}_{\frac{N}{2}}$ is a diagonal matrix of order $\frac{N}{2}$ defined as $\boldsymbol{W}_{\frac{N}{2}} = \text{diag}\{e^{-i\frac{2\pi k}{N}}\}$, $k = 0, 1, \ldots, \frac{N}{2} - 1$. Complex multiplications by each complex twiddle factor in the matrix $\boldsymbol{W}_{\frac{N}{2}}$ can be converted to the Givens–Jacobi rotations (see Appendix F.4).

Note that the normalized DFT matrix is symmetric, i.e., $\boldsymbol{F}_N^T = \boldsymbol{F}_N$, and is unitary, i.e., $\boldsymbol{F}_N^* = \boldsymbol{F}_N^{-1}$, $\boldsymbol{F}_N^*\, \boldsymbol{F}_N = \boldsymbol{F}_N\, \boldsymbol{F}_N^* = \boldsymbol{I}_N$, where the symbol $*$ denotes complex conjugation [26].

9.2.5 *Fast MDCT and MLT Analysis and Synthesis Filter Banks*

The complete analysis and synthesis MDCT filter banks are, respectively, defined as [85]

$$c_k^{(t)} = \sqrt{\frac{4}{N}} \sum_{n=0}^{N-1} w_n \, x_n^{(t)} \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{9.26}$$

$$\hat{x}_n^{(t)} = \sqrt{\frac{4}{N}} \, w_n \sum_{k=0}^{\frac{N}{2}-1} c_k^{(t)} \cos\left[\frac{\pi}{2N}\left(2n+1+\frac{N}{2}\right)(2k+1)\right],$$
$$n = 0, 1, \ldots, N-1, \tag{9.27}$$

where $\{x_n^{(t)}\}$ is the input data sequence, $\{c_k^{(t)}\}$ are the MDCT transform coefficients and $\{\hat{x}_n^{(t)}\}$ is the time domain aliased data sequence. The superscript (t) denotes the data-block number, and $\{w_n\}$ is a symmetric windowing function. The analysis/synthesis MDCT filter bank implicitly associated with the sine windowing function is the analysis/synthesis MLT filter bank [82]. In the analysis filter bank given by (9.26), for the data block t, N windowed time domain samples $\{x_n^{(t)}\}$ are used to calculate $\frac{N}{2}$ unique transform coefficients $\{c_k^{(t)}\}$. Vice versa, the tth data block of $\frac{N}{2}$ transform coefficients $\{c_k^{(t)}\}$ is used to calculate N windowed time domain aliased samples $\{\hat{x}_n^{(t)}\}$ with the synthesis filter bank given by (9.27).

In the analysis MDCT filter bank two succeeding data blocks t and $t+1$ are overlapped by $\frac{N}{2}$ samples so that for each data block $\frac{N}{2}$ new time domain samples are processed. For a smooth block overlapping a windowing function $\{w_n\}$ is applied to $\{x_n^{(t)}\}$ and $\{x_n^{(t+1)}\}$, the so-called windowing and overlap procedure. By applying analysis and synthesis MDCT filter banks, a time domain aliasing error is introduced. The aliasing error is canceled out (or perfect reconstruction is accomplished) by adding outputs of the synthesis MDCT filter bank of two succeeding windowed data blocks t and $t+1$ in the overlapped part, the so-called windowing and overlap and add procedure, or also called the time domain aliasing cancellation (TDAC) procedure as follows:

$$x_n^{(t+1)} = x_{\frac{N}{2}+n}^{(t)} = \hat{x}_{\frac{N}{2}+n}^{(t)} + \hat{x}_n^{(t+1)}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{9.28}$$

To ensure TDAC, the windowing functions of two succeeding data blocks have to satisfy the so-called perfect reconstruction conditions in their overlapped part. A sufficient condition for TDAC is given by Princen et al. [85]

$$w_n^2 + w_{\frac{N}{2}+n}^2 = 1, \quad \text{or} \quad w_n^2 + w_{\frac{N}{2}-1-n}^2 = 1,$$
$$w_n = w_{N-1-n}, \quad \text{or} \quad w_{\frac{N}{2}+n} = w_{\frac{N}{2}-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{9.29}$$

Thus, the analysis MDCT filter bank given by (9.26) in processing two adjacent overlapped data blocks $\{x_n^{(t)}\}$ and $\{x_n^{(t+1)}\}$ consists of the windowing and overlap

procedure and transforming adjacent data blocks by the forward MDCT block transform (realized by a fast DCT-IV computational structure). Vice versa, the synthesis MDCT filter bank given by (9.27) in processing two adjacent blocks of transform coefficients $\{c_k^{(t)}\}$ and $\{c_k^{(t+1)}\}$ consists of the backward MDCT transformation of coefficients (realized by the fast DCT-IV computational structure) and the windowing and overlap and add procedure to perfectly reconstruct the original data sequence in the overlapped part.

In the following subsections we will present the fast analysis and synthesis MDCT filter banks associated with an arbitrary symmetric windowing function $\{w_n\}$ which satisfies the perfect reconstruction conditions given by (9.29).

9.2.5.1 Fast Analysis MDCT Filter Bank

The fast analysis MDCT filter bank for the data block t with incorporated windowing procedure is defined as [76]

$$c_k^{(t)} = \sum_{n=0}^{\frac{N}{2}-1} y_n^{(t)} \cos\left[\frac{\pi}{4(N/2)}(2n+1)(2k+1)\right], \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{9.30}$$

where

$$\begin{aligned} y_{\frac{N}{4}+n}^{(t)} &= w_n x_n^{(t)} - w_{\frac{N}{2}-1-n} x_{\frac{N}{2}-1-n}^{(t)}, \\ y_{\frac{N}{4}-1-n}^{(t)} &= -w_{\frac{N}{2}+n} x_{\frac{N}{2}+n}^{(t)} - w_{N-1-n} x_{N-1-n}^{(t)}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{9.31}$$

Equations (9.30) and (9.31) define the fast analysis MDCT filter bank with incorporated windowing procedure. The cosine transform kernel in (9.30) is the $\frac{N}{2}$-point forward DCT-IV of $\{y_n^{(t)}\}$. Equation (9.31) includes besides the windowing also the permutation of input data sequence $\{x_n^{(t)}\}$, and frequently is called the TDA procedure.

9.2.5.2 Efficient Implementation of the Windowing and Overlap Procedure

In the analysis MDCT filter bank between two adjacent overlapped data blocks $\{x_n^{(t)}\}$ and $\{x_n^{(t+1)}\}$ the following relations hold:

$$x_{\frac{N}{2}+n}^{(t)} = x_n^{(t+1)}, \qquad x_{N-1-n}^{(t)} = x_{\frac{N}{2}-1-n}^{(t+1)}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{9.32}$$

Using (9.32) and the symmetry property of windowing function, (9.31) is rewritten as

$$\begin{aligned} y^{(t)}_{\frac{N}{4}+n} &= \quad w_n\, x^{(t)}_n \quad - \; w_{\frac{N}{2}-1-n}\, x^{(t)}_{\frac{N}{2}-1-n}, \\ y^{(t)}_{\frac{N}{4}-1-n} &= -\, w_{\frac{N}{2}-1-n}\, x^{(t+1)}_n - \; w_n\, x^{(t+1)}_{\frac{N}{2}-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{9.33}$$

For clarity, in processing the succeeding overlapped data blocks $0, 1, \ldots, t,\ t+1,\ t+2, \ldots$, the associated $\frac{N}{4}$-point data sub-sequences $\{y^{(.)}_{\frac{N}{4}+n}\}$ and $\{y^{(.)}_{\frac{N}{4}-1-n}\}$ are, respectively, given by

$$y^{(0)}_{\frac{N}{4}+n} = 0, \qquad \text{(initialization step)},$$

$$\begin{aligned} y^{(0)}_{\frac{N}{4}-1-n} &= -w_{\frac{N}{2}-1-n}\, x^{(1)}_n - w_n\, x^{(1)}_{\frac{N}{2}-1-n}, \\ y^{(1)}_{\frac{N}{4}+n} &= \quad w_n\, x^{(1)}_n - w_{\frac{N}{2}-1-n}\, x^{(1)}_{\frac{N}{2}-1-n}, \qquad \left\{y^{(1)}_{\frac{N}{4}+n}\right\} \text{ is } \frac{N}{4}\text{-sample delayed data sub-block}, \end{aligned}$$

$$\vdots$$

$$y^{(t)}_{\frac{N}{4}+n} = \quad w_n\, x^{(t)}_n - w_{\frac{N}{2}-1-n}\, x^{(t)}_{\frac{N}{2}-1-n},$$

$$\begin{aligned} y^{(t)}_{\frac{N}{4}-1-n} &= -w_{\frac{N}{2}-1-n}\, x^{(t+1)}_n - w_n\, x^{(t+1)}_{\frac{N}{2}-1-n}, \\ y^{(t+1)}_{\frac{N}{4}+n} &= \quad w_n\, x^{(t+1)}_n - w_{\frac{N}{2}-1-n}\, x^{(t+1)}_{\frac{N}{2}-1-n}, \qquad \left\{y^{(t+1)}_{\frac{N}{4}+n}\right\} \text{ is } \frac{N}{4}\text{-sample delayed data sub-block}, \end{aligned}$$

$$\begin{aligned} y^{(t+1)}_{\frac{N}{4}-1-n} &= -w_{\frac{N}{2}-1-n}\, x^{(t+2)}_n - w_n\, x^{(t+2)}_{\frac{N}{2}-1-n}, \\ y^{(t+2)}_{\frac{N}{4}+n} &= \quad w_n\, x^{(t+2)}_n - w_{\frac{N}{2}-1-n}\, x^{(t+2)}_{\frac{N}{2}-1-n}, \qquad \left\{y^{(t+2)}_{\frac{N}{4}+n}\right\} \text{ is } \frac{N}{4}\text{-sample delayed data sub-block}, \end{aligned}$$

$$\vdots$$

By introducing the $\frac{N}{4}$-sample delay operator denoted by $\Delta_{\frac{N}{4}}\{.\}$, the windowing and overlap procedure in the fast analysis MDCT filter bank can be represented in the following matrix-vector form:

$$\begin{pmatrix} y^{(t)}_{\frac{N}{4}-1-n} \\ \\ \Delta_{\frac{N}{4}}\left\{y^{(t+1)}_{\frac{N}{4}+n}\right\} \end{pmatrix} = - \begin{pmatrix} w_{\frac{N}{2}-1-n} & w_n \\ \\ -w_n & w_{\frac{N}{2}-1-n} \end{pmatrix} \begin{pmatrix} x^{(t+1)}_n \\ \\ x^{(t+1)}_{\frac{N}{2}-1-n} \end{pmatrix},$$

$$n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{9.34}$$

with initial condition $y^{(0)}_{\frac{N}{4}+n} = 0$. Since the windowing function satisfies the perfect reconstruction conditions given by (9.29), the determinant of 2×2 matrix on the right-hand side of (9.34) is unity. The windowing and overlap procedure including the permutation (9.31) requires $3\frac{N}{4}$ multiplications and $3\frac{N}{4}$ additions.

9.2.5.3 Fast Synthesis MDCT Filter Bank

Since the DCT-IV matrix is self-inverse, the data sequence $\{y^{(t)}_n\}$ is recovered by the inverse $\frac{N}{2}$-point DCT-IV of $\{c^{(t)}_k\}$ as [76]

$$y^{(t)}_n = \sum_{k=0}^{\frac{N}{2}-1} c^{(t)}_k \cos\left[\frac{\pi}{4(N/2)}(2n+1)(2k+1)\right], \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{9.35}$$

The time domain aliased data sequence $\{\hat{x}^{(t)}_n\}$ is obtained from $\{y^{(t)}_n\}$ by applying the inverse permutation to (9.31) as

$$\begin{aligned}
\hat{x}^{(t)}_n &= \quad w_n\, y^{(t)}_{\frac{N}{4}+n}, \\
\hat{x}^{(t)}_{\frac{N}{2}-1-n} &= -\, w_{\frac{N}{2}-1-n}\, y^{(t)}_{\frac{N}{4}+n}, \\
\hat{x}^{(t)}_{\frac{N}{2}+n} &= -\, w_{\frac{N}{2}+n}\, y^{(t)}_{\frac{N}{4}-1-n} \quad = -\, w_{\frac{N}{2}-1-n}\, y^{(t)}_{\frac{N}{4}-1-n}, \\
\hat{x}^{(t)}_{N-1-n} &= -\, w_{N-1-n}\, y^{(t)}_{\frac{N}{4}-1-n} = -\, w_n\, y^{(t)}_{\frac{N}{4}-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1.
\end{aligned} \tag{9.36}$$

Equations (9.35) and (9.36) define the fast synthesis MDCT filter bank.

9.2.5.4 Efficient Implementation of the Windowing and Overlap and Add Procedure

Given two succeeding blocks of transform coefficients $\{c^{(t)}_k\}$ and $\{c^{(t+1)}_k\}$. After transforming each block of transform coefficients by the $\frac{N}{2}$-point inverse DCT-IV transform, using the symmetry property of windowing function, from (9.36) it follows that the time domain aliased data sequences $\{\hat{x}^{(t)}_n\}$ and $\{\hat{x}^{(t+1)}_n\}$ are, respectively, given by

$$\begin{aligned}
\hat{x}^{(t)}_{\frac{N}{2}+n} &= -w_{\frac{N}{2}-1-n}\, y^{(t)}_{\frac{N}{4}-1-n}, \\
\hat{x}^{(t)}_{N-1-n} &= -w_n\, y^{(t)}_{\frac{N}{4}-1-n},
\end{aligned}$$

$$\begin{aligned} \hat{x}_n^{(t+1)} &= w_n \, y_{\frac{N}{4}+n}^{(t+1)}, \\ \hat{x}_{\frac{N}{2}-1-n}^{(t+1)} &= -w_{\frac{N}{2}-1-n} \, y_{\frac{N}{4}+n}^{(t+1)}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{9.37}$$

For two adjacent time domain aliased data blocks $\{\hat{x}_n^{(t)}\}$ and $\{\hat{x}_n^{(t+1)}\}$ recovered by the synthesis MDCT filter bank, the following relations hold in the overlapped part:

$$x_n^{(t+1)} = \hat{x}_{\frac{N}{2}+n}^{(t)} + \hat{x}_n^{(t+1)}, \qquad x_{\frac{N}{2}-1-n}^{(t+1)} = \hat{x}_{N-1-n}^{(t)} + \hat{x}_{\frac{N}{2}-1-n}^{(t+1)}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{9.38}$$

In order to perfectly reconstruct the original data sequence $\{x_n^{(t+1)}\}$ in the overlapped part, by substitution of the appropriate time domain aliased data sequences $\{\hat{x}_n^{(t)}\}$ from (9.37) into (9.38), and using $\frac{N}{4}$-sample delay operator $\Delta_{\frac{N}{4}}\{.\}$, the windowing and overlap and add procedure in the fast synthesis MDCT filter bank is expressed as

$$\begin{aligned} x_n^{(t+1)} &= -w_{\frac{N}{2}-1-n} \, \Delta_{\frac{N}{4}} \left\{ y_{\frac{N}{4}-1-n}^{(t)} \right\} + w_n \, y_{\frac{N}{4}+n}^{(t+1)}, \\ x_{\frac{N}{2}-1-n}^{(t+1)} &= -w_n \, \Delta_{\frac{N}{4}} \left\{ y_{\frac{N}{4}-1-n}^{(t)} \right\} - w_{\frac{N}{2}-1-n} \, y_{\frac{N}{4}+n}^{(t+1)}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1, \end{aligned} \tag{9.39}$$

or equivalently, it can be represented in the matrix-vector form as

$$\begin{pmatrix} x_n^{(t+1)} \\ x_{\frac{N}{2}-1-n}^{(t+1)} \end{pmatrix} = - \begin{pmatrix} w_{\frac{N}{2}-1-n} & -w_n \\ w_n & w_{\frac{N}{2}-1-n} \end{pmatrix} \begin{pmatrix} \Delta_{\frac{N}{4}} \left\{ y_{\frac{N}{4}-1-n}^{(t)} \right\} \\ y_{\frac{N}{4}+n}^{(t+1)} \end{pmatrix}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{9.40}$$

Similarly, from the perfect reconstruction conditions given by (9.29) it follows that the determinant of 2×2 matrix on the right-hand side of (9.40) is unity. The windowing and overlap and add procedure including the permutation (9.31) requires $3\frac{N}{4}$ multiplications and $3\frac{N}{4}$ additions.

Note 3: For the fast analysis and synthesis MLT filter banks the sine windowing function is defined by $w_n = \sin \frac{\pi(2n+1)}{2N}$ for $n = 0, 1, \ldots, N-1$ [82]. Denoting $w_n = s_n$, we have: $w_{\frac{N}{2}-1-n} = c_n = \cos \frac{\pi(2n+1)}{2N}$, $w_{\frac{N}{2}+n} = c_n$, and $w_{N-1-n} = s_n$ for $n = 0, 1, \ldots, \frac{N}{2} - 1$.

Thus, the efficient implementation of complete analysis and synthesis MDCT filter banks consists of the $\frac{N}{2}$-point identical fast DCT-IV computational structure, the windowing and overlap procedure (including the TDA procedure) and the

windowing and overlap and add procedure. Equations (9.34) and (9.40) provide the compact computational structures, respectively, for the efficient implementation of the analysis MDCT filter bank with incorporated windowing and overlap procedure, and the synthesis MDCT filter bank with incorporated windowing and overlap and add procedure for an arbitrary symmetric windowing function (for the fast analysis and synthesis MLT filter banks see Fig. 9.11a, b, respectively).

Having prepared the desired properties of integer transforms, quality measures of integer transforms for coding, having defined the concept of normalized (integer) transform with the corresponding orthogonal (unitary) recursive sparse block matrix factorizations of the DCT-II/III, DCT-IV, and DFT matrices, and having defined the fast MDCT and MLT analysis and synthesis filter banks, in the following sections, local and global methods to integer approximation of transforms and cosine/sine-modulated filter banks are presented. The local as well as global methods to integer approximation are based on the classical matrix theory and computational methods of linear algebra, and in particular, on (block) matrix decompositions [80, 81, 86] (see Appendices A.3–A.8).

9.3 Local Methods to Integer Approximation

The orthogonal (unitary) recursive sparse block matrix factorizations (9.1), (9.13), (9.24), and (9.25) define the corresponding fast computational structures. Each matrix factor is a block matrix with blocks being the orthogonal 2×2 rotation matrix and/or orthogonal 2×2 trivial rotation matrix with unit determinants. Thus, the construction of reversible (invertible) integer transform is reduced to the construction of only integer transforms of the length 2. In order to construct the reversible (invertible) integer transforms which map integers-to-integers, each orthogonal 2×2 matrix has to be factored into a product of invertible elementary matrices [12–14, 16, 17, 29–31, 35, 37–39, 45, 46, 48]. However, for large transform sizes the number of orthogonal 2×2 matrices increases. Therefore, the local methods to integer approximation are rather suitable for small transform sizes [54, 55].

9.3.1 LUL Matrix Factorizations of a 2-Point Block Transform

Let $\boldsymbol{T}_2$ be a matrix of order 2 with scalar elements $a, b, c, d \in R$ representing a 2-point block transform as

$$\boldsymbol{T}_2 = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \qquad b \neq 0, \tag{9.41}$$

with $\det(\boldsymbol{T}_2) = ad - bc = +1$. We show that the matrix $\boldsymbol{T}_2$ can be factored into a product of the Gauss elementary matrices being unit lower and unit upper triangular matrices. Indeed, since $\det(\boldsymbol{T}_2) = +1$, according to ***PLUS*** factorization theorem we can apply to the matrix $\boldsymbol{T}_2$ the ***PLUS*** factorization algorithm (see Appendix A.6).

The permutation matrix $\boldsymbol{P}$ in this case corresponds to the identity matrix and we can omit it. In the first step there must exist a number s_1 such that $a - s_1\ b = 1$ and hence $s_1 = \frac{a-1}{b}$ and we obtain the product of

$$\boldsymbol{T}_2\ \boldsymbol{S}_2^{-1} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -s_1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & b \\ c - s_1\ d & d \end{pmatrix}.$$

The second step is Gaussian elimination of the first column of $\boldsymbol{T}_2\ \boldsymbol{S}_2^{-1}$, what is equivalent to the pre-multiplication of $\boldsymbol{T}_2\ \boldsymbol{S}_2^{-1}$ by the Gauss elementary matrix $\boldsymbol{L}_2^{-1}$ as follows

$$\boldsymbol{L}_2^{-1}\ \boldsymbol{T}_2\ \boldsymbol{S}_2^{-1} = \begin{pmatrix} 1 & 0 \\ s_1\ d - c & 1 \end{pmatrix} \boldsymbol{T}_2 \begin{pmatrix} 1 & 0 \\ -s_1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix} = \boldsymbol{U}_2, \tag{9.42}$$

and we obtained the factorization of $\boldsymbol{L}_2^{-1}\ \boldsymbol{T}_2\ \boldsymbol{S}_2^{-1}$. From algebra of unit triangular matrices (see Appendix A.3) it follows that the inverse of a unit lower/unit upper triangular matrix is also a unit lower/unit upper triangular matrix, and we easily find their inverses as

$$\begin{pmatrix} 1 & 0 \\ -x & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 \\ x & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & -x \\ 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix}.$$

Pre-multiplying Eq. (9.42) by $\boldsymbol{L}_2$ (whereby its off-diagonal element $c - s_1\ d = \frac{d-1}{b}$) followed by the post-multiplying by $\boldsymbol{S}_2$ (whereby its off-diagonal element is s_1), we finally get the factorization of $\boldsymbol{T}_2 = \boldsymbol{LUS}$ (and hence $\boldsymbol{LUL}$ matrix factorization) given by

$$\boldsymbol{T}_2 = \begin{pmatrix} 1 & 0 \\ \frac{d-1}{b} & 1 \end{pmatrix} \begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{a-1}{b} & 1 \end{pmatrix}, \qquad \text{where } b \neq 0. \tag{9.43}$$

For the inverse matrix $\boldsymbol{T}_2^{-1}$ we obtain

$$\boldsymbol{T}_2^{-1} = \begin{pmatrix} 1 & 0 \\ -\frac{a-1}{b} & 1 \end{pmatrix} \begin{pmatrix} 1 & -b \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{d-1}{b} & 1 \end{pmatrix}. \tag{9.44}$$

The $\boldsymbol{LUL}$ matrix factorization of $\boldsymbol{T}_2$ defined by (9.43) has been presented in [12, 14, 29, 30], however without any discussion to its origin. Note that the factored matrices on the right-hand side of (9.43) and (9.44) are invertible. Thus, the $\boldsymbol{PLUS}$ factorization of $\boldsymbol{T}_2$ and $\boldsymbol{T}_2^{-1}$ into the product of Gauss elementary matrices given by (9.43) and (9.44), respectively, define computational structures for the efficient implementations of the forward and inverse 2-point block transforms. The corresponding $\boldsymbol{LUL}$ computational structures of $\mathbf{y} = \boldsymbol{T}_2\ \mathbf{x}$ and $\mathbf{x} = \boldsymbol{T}_2^{-1}\mathbf{y}$ are, respectively, shown in Fig. 9.3a, b, where $(x_0, x_1)^T$ is the input data vector and $(y_0, y_1)^T$ is the output data vector.

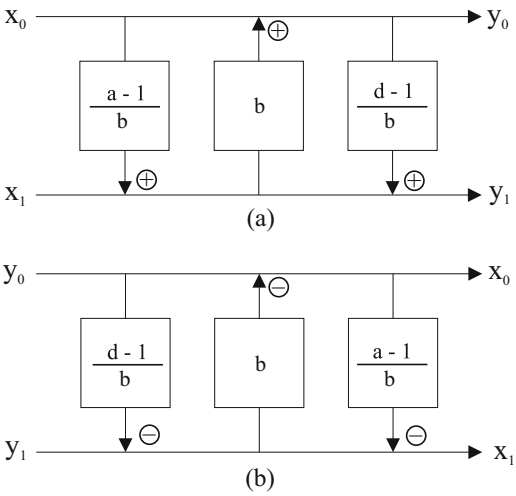

Fig. 9.3 ***LUL*** computational structures for the efficient implementation: (**a**) of $\mathbf{y} = \boldsymbol{T}_2\,\mathbf{x}$, and (**b**) of $\mathbf{x} = \boldsymbol{T}_2^{-1}\,\mathbf{y}$

From the implementation point of view, to invert $\boldsymbol{T}_2$ given by ***PLUS*** factorization, we simply need to subtract out what was added in the forward computation. The implementation of $\mathbf{y} = \boldsymbol{T}_2\ \mathbf{x}$ given by ***PLUS*** factorization compared to its direct implementation has the following advantages:

- The number of multiplications is reduced from four to three multiplications and overall arithmetic complexity is three multiplications and three additions.
- Leads to in-place implementation, i.e., without the need of auxiliary memory, which is the desired property in VLSI implementations.
- Multipliers in factor matrices can be quantized (using functions such as *round*, *floor* or *ceil*) to obtain the reversible integer-to-integer mapping.
- The perfect reconstruction property.

Note 4: A general matrix factorization theory of an invertible linear transform into a product of unit triangular matrices has been developed in [13, 18]. If a real nonsingular matrix of order N has its determinant equal to 1, then it has the ***PLUS*** factorization (for further details see Appendix A.6). As examples, the ***PLUS*** factorizations of the 4-point DFT and 8-point DCT-II are presented in [13].

9.3.2 *LUL (ULU) Matrix Factorizations of Givens–Jacobi Rotations*

A 2×2 orthogonal matrix $\boldsymbol{G}_2^{(\varphi)}$ is called Givens–Jacobi rotation by an angle φ, if it has the form [81]

$$\boldsymbol{G}_2^{(\varphi)} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}, \quad \left[\boldsymbol{G}_2^{(\varphi)}\right]^{-1} = \left[\boldsymbol{G}_2^{(\varphi)}\right]^{T} = \boldsymbol{G}_2^{(-\varphi)}. \tag{9.45}$$

where $\boldsymbol{G}_2^{(\varphi)}$ and $\boldsymbol{G}_2^{(-\varphi)}$ are inverses to each other. It is clear that det $(\boldsymbol{G}_2^{(\varphi)}) =$ det $(\boldsymbol{G}_2^{(-\varphi)}) = 1$. Orthogonal Givens–Jacobi rotations are also called the linear 2-norm rotation transforms, because they preserve the 2-norm of the rotated vectors. They play a central role in the least squares solutions of overdetermined systems of linear equations as well as symmetric eigenvalue problem [80, 81].

Although ***LUL*** matrix factorizations of Givens–Jacobi rotations $\boldsymbol{G}_2^{(\varphi)}$ and $\boldsymbol{G}_2^{(-\varphi)}$ be can obtained by applying the ***PLUS*** factorization algorithm to the matrix $\boldsymbol{G}_2^{(\varphi)}$, without loss of generality, they are simply obtained by substituting $a = d = \cos\varphi$, $b = -\sin\varphi$ and $c = \sin\varphi$ into (9.43) and (9.44). Then, the ***LUL*** factorizations of $\boldsymbol{G}_2^{(\varphi)}$ and $\boldsymbol{G}_2^{(-\varphi)}$ are given by

$$\boldsymbol{G}_2^{(\varphi)} = \begin{pmatrix} 1 & 0 \\ \frac{1-\cos\varphi}{\sin\varphi} & 1 \end{pmatrix} \begin{pmatrix} 1 & -\sin\varphi \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{1-\cos\varphi}{\sin\varphi} & 1 \end{pmatrix}, \tag{9.46}$$

and

$$\boldsymbol{G}_2^{(-\varphi)} = \begin{pmatrix} 1 & 0 \\ -\frac{1-\cos\varphi}{\sin\varphi} & 1 \end{pmatrix} \begin{pmatrix} 1 & \sin\varphi \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{1-\cos\varphi}{\sin\varphi} & 1 \end{pmatrix}, \quad \text{where} \quad \sin\varphi \neq 0. \tag{9.47}$$

We note that $\tan\frac{\varphi}{2} = \frac{1-\cos\varphi}{\sin\varphi} = \frac{\sin\varphi}{1+\cos\varphi}$. Although the matrix $\boldsymbol{G}_2^{(\varphi)}$ is orthogonal, the factored matrices on the right-hand sides of (9.46) and (9.47) are not longer orthogonal; however, they are still invertible. The corresponding ***LUL*** computational structures for the efficient implementation of the forward and inverse Givens–Jacobi rotations are, respectively, shown in Fig. 9.4a, b, where $(x_0, x_1)^T$ is the input data vector and $(y_0, y_1)^T$ is the rotated vector.

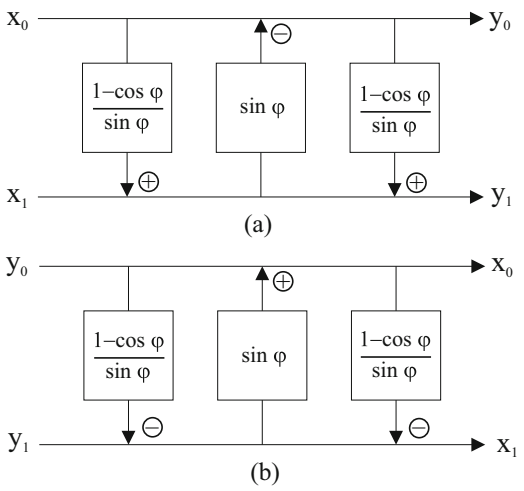

Fig. 9.4 ***LUL*** computational structures for the efficient implementation: (**a**) of $\mathbf{y} = \boldsymbol{G}_2^{(\varphi)}\mathbf{x}$, and (**b**) of $\mathbf{x} = \boldsymbol{G}_2^{(-\varphi)}\mathbf{y}$

Since $\boldsymbol{G}_2^{(\varphi)} = [\boldsymbol{G}_2^{(-\varphi)}]^T$, by transposing (9.47) we obtain the alternative ***ULU*** matrix factorizations of $\boldsymbol{G}_2^{(\varphi)}$ and $\boldsymbol{G}_2^{(-\varphi)}$ as

$$\boldsymbol{G}_2^{(\varphi)} = \begin{pmatrix} 1 & -\frac{1-\cos\varphi}{\sin\varphi} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \sin\varphi & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{1-\cos\varphi}{\sin\varphi} \\ 0 & 1 \end{pmatrix}, \qquad \text{where} \quad \sin\varphi \neq 0, \tag{9.48}$$

and

$$\boldsymbol{G}_2^{(-\varphi)} = \begin{pmatrix} 1 & \frac{1-\cos\varphi}{\sin\varphi} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\sin\varphi & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1-\cos\varphi}{\sin\varphi} \\ 0 & 1 \end{pmatrix}. \tag{9.49}$$

The corresponding ***ULU*** computational structures for the alternative efficient implementation of the forward and inverse Givens–Jacobi rotations can be simply obtained by a minor modification of ***LUL*** ones shown in Fig. 9.4. From the computational point of view, the ***LUL*** and ***ULU*** structures are equivalent.

Note 5: All orthogonal (unitary) recursive sparse block matrix factorizations (9.1), (9.13), (9.24), and (9.25) contain also trivial 2×2 rotation matrices of the form:

$$\boldsymbol{H}_2 = \frac{\sqrt{2}}{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{9.50}$$

The matrix $\boldsymbol{H}_2$ in (9.50) is recognized as the scaled 2×2 Hadamard matrix. If it is applied to a 2-point vector $\mathbf{x}^T$, then we have:

$$\begin{pmatrix} \cos\frac{\pi}{4} & \sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & -\cos\frac{\pi}{4} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{pmatrix} \begin{pmatrix} x_0 \\ -x_1 \end{pmatrix}. \tag{9.51}$$

From Eq. (9.51) it can be seen that the trivial rotation of the input vector $\mathbf{x}^T$ (and hence scaled Hadamard matrix $\boldsymbol{H}_2$) is converted to the forward Givens–Jacobi rotation with the angle $\frac{\pi}{4}$ of the modified input vector. Using (9.46) we get the ***LUL*** factorization of $\boldsymbol{G}_2^{(\frac{\pi}{4})}$ as

$$\boldsymbol{G}_2^{(\frac{\pi}{4})} = \begin{pmatrix} 1 & 0 \\ \tan\frac{\pi}{8} & 1 \end{pmatrix} \begin{pmatrix} 1 & -\sin\frac{\pi}{4} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \tan\frac{\pi}{8} & 1 \end{pmatrix}, \quad \text{where} \quad \tan\frac{\pi}{8} = \sqrt{2} - 1, \tag{9.52}$$

and using (9.48) we get the alternative ***ULU*** factorization of $\boldsymbol{G}_2^{(\frac{\pi}{4})}$ as

$$\boldsymbol{G}_2^{(\frac{\pi}{4})} = \begin{pmatrix} 1 & -\tan\frac{\pi}{8} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \sin\frac{\pi}{4} & 1 \end{pmatrix} \begin{pmatrix} 1 & -\tan\frac{\pi}{8} \\ 0 & 1 \end{pmatrix}. \tag{9.53}$$

Note 6: The second quasi-diagonal matrix on the right-hand side of (9.25) includes the diagonal matrix $\boldsymbol{W}_{\frac{N}{2}} = \text{diag}\{e^{-i\frac{2\pi k}{N}}\}$ for $k = 0, 1, \ldots, \frac{N}{2} - 1$. We recall that the complex multiplications by each complex twiddle factor in the matrix $\boldsymbol{W}_{\frac{N}{2}}$ can be converted to the forward Givens–Jacobi and trivial rotations (see Appendix F.4).

Principally, there are two approaches how to construct the reversible (invertible) integer transforms from the corresponding factored matrices $\boldsymbol{T}_2$, $\boldsymbol{T}_2^{-1}$ given by (9.43), (9.44), respectively, and the reversible (invertible) integer rotations from the corresponding factored matrices $\boldsymbol{G}_2^{(\varphi)}$, $\boldsymbol{G}_2^{(-\varphi)}$ given by (9.46), (9.47) or (9.48), (9.49), respectively, applied to the input vector $\mathbf{x}^T$:

- Including the nonlinear rounding operator in each factored unit triangular matrix (integer approximation by the rounding operator). Thus, multiplying each by the off-diagonal floating-point multiplier is followed immediately by the rounding operation, i.e., the result is rounded to the nearest integer [16].
- Replacing the off-diagonal floating-point multiplier in each factored unit triangular matrix by an appropriate dyadic rational number (dyadic approximation) [17].

Both approaches to integer approximation result in an approximation error in each computational step and this accumulated approximation error has to be minimized. The approximation error can be interpreted as the difference between results of the exact transform and the related integer transform in terms of some vector norm (for example, Euclidean or maximum).

In the following subsections each approach is investigated in detail separately, and the corresponding explicit estimates of approximation errors are also discussed as well as the optimization strategies to minimize the total approximation error.

9.3.2.1 Approximation by Rounding Operator and Estimate of Approximation Error

One can see that the orthogonal (unitary) recursive sparse block matrix factorizations (9.1), (9.13), (9.24), and (9.25) are composed from blocks being 2×2 Givens–Jacobi rotations $\boldsymbol{G}_2^{(\pm\varphi)}$ and $\boldsymbol{G}_2^{(\frac{\pi}{4})}$. Then, the first approach to construction of an invertible integer transform of order N can be formulated simply as follows: For a given invertible matrix $\boldsymbol{G}_2^{(\pm\varphi)}$ with rotation angles $\varphi \in (0, \frac{\pi}{4}\rangle$ and for arbitrary vector $\mathbf{x} \in Z^2$, we need to find a suitable integer approximation of $\boldsymbol{G}_2^{(\pm\varphi)}\mathbf{x}$ such that this process is invertible.

Due to practical reasons, consider the ***ULU*** factorizations of $\boldsymbol{G}_2^{(\varphi)}$ and $\boldsymbol{G}_2^{(-\varphi)}$ expressed, respectively, in more compact forms as

$$\boldsymbol{G}_2^{(\varphi)} = \begin{pmatrix} 1 & -\tan\frac{\varphi}{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \sin\varphi & 1 \end{pmatrix} \begin{pmatrix} 1 & -\tan\frac{\varphi}{2} \\ 0 & 1 \end{pmatrix}, \tag{9.54}$$

$$\boldsymbol{G}_2^{(-\varphi)} = \begin{pmatrix} 1 & \tan\frac{\varphi}{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\sin\varphi & 1 \end{pmatrix} \begin{pmatrix} 1 & \tan\frac{\varphi}{2} \\ 0 & 1 \end{pmatrix}, \text{where} \quad \sin\varphi \neq 0. \tag{9.55}$$

and the ***ULU*** factorization of $\boldsymbol{G}_2^{(\frac{\pi}{4})}$ given by (9.53). The ***ULU*** factorizations consist of nonorthogonal matrix factors which are still invertible, and they can be used for the construction of invertible integer transform as follows.

For $a \in R$ let $\lfloor a \rfloor = \max\{x \leq a : \ x \in Z\}$ and $\{a\} = a - \lfloor a \rfloor \in \langle 0, 1)$. Then $\{a\}$ is the noninteger part of a. Further, let $\text{round}(a) = \lfloor a + \frac{1}{2} \rfloor$ be the integer closest to a. Consider the last matrix factor of $\boldsymbol{G}_2^{(\varphi)}$ on the right-hand side of (9.54). One computational step in the matrix-vector form can be written as

$$\hat{\mathbf{y}} = \begin{pmatrix} 1 & -s \\ 0 & 1 \end{pmatrix} \mathbf{x}, \quad s \in R,$$

where $\mathbf{x} = [x_0, x_1]^T \in Z^2$. Then, $\hat{\mathbf{y}} = [\hat{y}_0, \hat{y}_1]^T$ can be approximated by $\mathbf{y} = [y_0, y_1]^T \in Z^2$ with

$$y_0 = x_0 - \left\lfloor sx_1 + \frac{1}{2} \right\rfloor = x_0 - \text{round}(sx_1), \quad y_1 = x_1.$$

This transform is invertible with

$$x_0 = y_0 + \left\lfloor sy_1 + \frac{1}{2} \right\rfloor = y_0 + \text{round}(sy_1), \quad x_1 = y_1.$$

Indeed, we have

$$y_1 = x_1, \quad y_0 + \text{round}(sy_1) = x_0 - \text{round}(sx_1) + \text{round}(sx_1) = x_0.$$

The following theorem is fundamental for the integer approximation of the ***ULU*** factored matrix $\boldsymbol{G}_2^{(\varphi)}$ by the rounding operator with explicit estimates of truncation errors [16].

Theorem 9.1 (Integer Approximation of the *ULU* Factored Matrix $\boldsymbol{G}_2^{(\varphi)}$ by the Rounding Operator and Explicit Truncation Error Estimates) *Let $\boldsymbol{G}_2^{(\varphi)}$ with $\varphi \in (0, \frac{\pi}{4})$ be the Givens–Jacobi matrix represented by **ULU** factorization. Then for arbitrary $\mathbf{x} = [x_0, x_1]^T \in Z^2$, a suitable integer approximation $\mathbf{y} = [y_0, y_1]^T \in Z^2$ of*

$\hat{\mathbf{y}} = \boldsymbol{G}_2^{(\varphi)}\mathbf{x}$ *is given by* $y_0 = z_2$, $y_1 = z_1$, *where*

$$z_0 = x_0 - round\left(x_1 \tan\frac{\varphi}{2}\right), \quad z_1 = x_1 + round\left(z_0 \sin\varphi\right),$$
$$z_2 = z_0 - round\left(z_1 \tan\frac{\varphi}{2}\right).$$

The procedure is invertible and its inverse is $x_0 = v_2$, $x_1 = v_1$, *where*

$$v_0 = y_0 + round\left(y_1 \tan\frac{\varphi}{2}\right), \quad v_1 = y_1 - round\left(v_0 \sin\varphi\right),$$
$$v_2 = v_0 + round\left(v_1 \tan\frac{\varphi}{2}\right).$$

The truncation error can be estimated by

$$\|\hat{\mathbf{y}} - \mathbf{y}\|_2 \le \sqrt{h(\varphi)}, \quad \|\hat{\mathbf{y}} - \mathbf{y}\|_\infty \le g(\varphi), \tag{9.56}$$

where functions $h(\varphi)$ *and* $g(\varphi)$ *are respectively given by*

$$h(\varphi) = \frac{3}{4} + \sin\varphi + \frac{1}{2}\cos\varphi + \frac{1}{4}\tan^2\frac{\varphi}{2}, \qquad g(\varphi) = \frac{1}{2}\left(1 + \tan\frac{\varphi}{2} + \cos\varphi\right).$$

The formulae for y_0, y_1 and x_0, x_1 (after inverse transform) directly follow by applying the input vector $\mathbf{x}$ to three matrices in (9.54). The elegant proof of truncation error estimates given by (9.56) can be found in [16]. Plot of the function $h(\varphi)$ is shown in Fig. 9.5, while plot of the function $g(\varphi)$ is shown in Fig. 9.6.

Let $\hat{\mathbf{y}} = \boldsymbol{G}_2^{(\varphi)}\mathbf{x}$ with arbitrary vector $\mathbf{x} \in Z^2$ be given and let $\mathbf{y}$ be its integer approximation. The special values for truncation errors $\|\hat{\mathbf{y}} - \mathbf{y}\|_2$ and $\|\hat{\mathbf{y}} - \mathbf{y}\|_\infty$ via the rounding procedure for rotation angles $\varphi \in \{\frac{\pi}{4}, \frac{\pi}{8}, \frac{\pi}{16}, \frac{3\pi}{16}\}$ can be obtained by substituting values of φ into (9.56). In particular, we obtain [16]

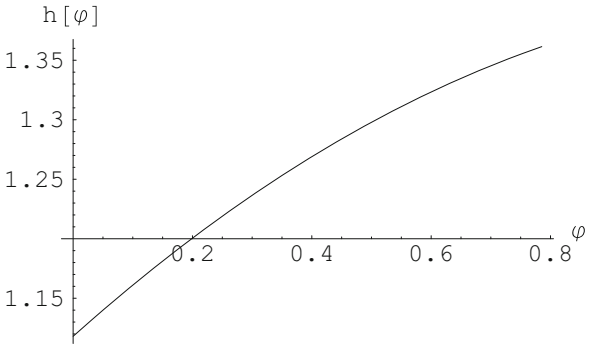

Fig. 9.5 Plot of the function $h(\varphi)$

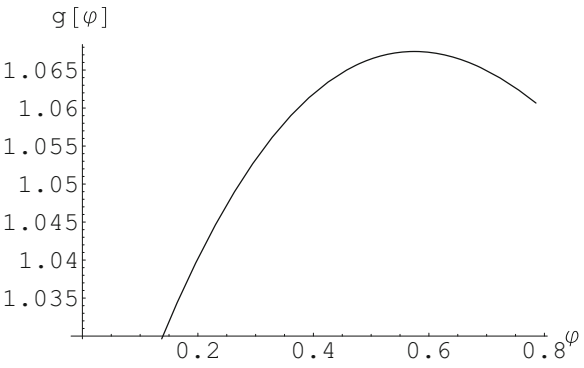

Fig. 9.6 Plot of the function $g(\varphi)$

$$\|\hat{\mathbf{y}}-\mathbf{y}\|_2 \;\le\; \begin{cases} 1.361453 & \text{for} \quad \varphi=\frac{\pi}{4}, \\ 1.266694 & \text{for} \quad \varphi=\frac{\pi}{8}, \\ 1.199128 & \text{for} \quad \varphi=\frac{\pi}{16}, \\ 1.320723 & \text{for} \quad \varphi=\frac{3\pi}{16}, \end{cases} \qquad \|\hat{\mathbf{y}}-\mathbf{y}\|_\infty \;\le\; \begin{cases} 1.060660 & \text{for} \quad \varphi=\frac{\pi}{4}, \\ 1.061396 & \text{for} \quad \varphi=\frac{\pi}{8}, \\ 1.039638 & \text{for} \quad \varphi=\frac{\pi}{16}, \\ 1.067408 & \text{for} \quad \varphi=\frac{3\pi}{16}. \end{cases}$$

Further, for all $\varphi \in \langle 0, \frac{\pi}{4}) $ we have

$$\|\hat{\mathbf{y}}-\mathbf{y}\|_\infty \;\le\; \max\left\{g(\varphi):\; \varphi \in \left\langle 0, \frac{\pi}{4}\right)\right\} \;\approx\; 1.067442.$$

9.3.2.2 Dyadic Approximation and Estimate of Approximation Error

The second approach to construction of an invertible integer transform of order N is based on the approximation of off-diagonal floating-point multipliers in factored unit triangular matrices by appropriate dyadic rational numbers.

Definition 9.1 The dyadic rational number is a binary fractional one of the form $\frac{k}{2^b}$, where $k, b \in N$ and k is an odd integer [9].

Multiplication by dyadic rational number can be implemented using only binary arithmetic. The multiplicand is first multiplied by numerator k and the result is shifted to the right by b bits. Neglecting shift operations, the minimum number of additions required for implementing a given binary fraction is equal to that for implementing its numerator k. An integer multiplication is equivalent to bit-shifting the multiplicand to the left by different numbers of bits and summing up these bit-shifted versions. The total numbers of shifts and additions required can be counted from the binary representation of the integer multiplier. For example, multiplication by $5 = (101)_2$ can be implemented by one addition and one shift. Similarly, multiplication by $7 = (111)_2$ can be implemented by two additions and two shifts, since $7 = 4 + 2 + 1 = (100)_2 + (10)_2 + (1)_2$. However, this is not the minimum number of additions needed to multiply a number by 7, because if we express $7 = 8 - 1 = (1000)_2 - (1)_2$, it is immediately clear that only one addition

and one shift are required. This fact is related to minimum-adder representation of integer multiplier k which is based on the concept of multiplicative irreducibility in terms of adders (see p. 221 in [9]).

Replacing the trigonometric values $\tan\frac{\varphi}{2}$ and $\sin\varphi$ by the dyadic rationals $a = \frac{\beta_a}{2^n}$ and $b = \frac{\beta_b}{2^n}$, $\beta_a,\ \beta_b,\ n \in N$, respectively, in the ***ULU*** factorization of $\boldsymbol{G}_2^{(\varphi)}$ given by (9.54), and multiplying the factored matrices we obtain the approximation matrix $\tilde{\boldsymbol{G}}_2^{(\varphi)}$ in the form:

$$\tilde{\boldsymbol{G}}_2^{(\varphi)} = \begin{pmatrix} 1-ab & -2a-a^2b \\ b & 1-ab \end{pmatrix}, \qquad \det\left(\tilde{\boldsymbol{G}}_2^{(\varphi)}\right) = 1. \tag{9.57}$$

The following theorem states how to estimate truncation errors caused by the dyadic approximation of ***ULU*** factored matrix $\boldsymbol{G}_2^{(\varphi)}$ [17].

Theorem 9.2 (Dyadic Approximation of the *ULU* Factored Matrix $\boldsymbol{G}_2^{(\varphi)}$ and Explicit Truncation Error Estimates) *Let $\boldsymbol{G}_2^{(\varphi)}$ be the matrix represented by **ULU** factorization, and $\tilde{\boldsymbol{G}}_2^{(\varphi)}$ be its approximation matrix with $\varphi \in (0, \frac{\pi}{2}\rangle$. Further, let $a = \frac{\beta_a}{2^n} \geq 0$ and $b = \frac{\beta_b}{2^n} \geq 0$, $\beta_a,\ \beta_b,\ n \in N$ be given with*

$$\left|\tan\frac{\varphi}{2} - a\right| \leq 2^{-j} \quad \text{and} \quad |\sin\varphi - b| \leq 2^{-j}$$

for some fixed $j \in N$ denoting the required approximation accuracy. Then for arbitrary $\mathbf{x} = [x_0, x_1]^T \in (-2^k, 2^k\rangle \cap Z^2$, a suitable integer approximation $\mathbf{y} = [y_0, y_1]^T \in Z^2$ of $\hat{\mathbf{y}} = \tilde{\boldsymbol{G}}_2^{(\varphi)}\,\mathbf{x}$ is given by $y_0 = z_2$, $y_1 = z_1$, where

$$z_0 = x_0 - round(x_1\ a), \quad z_1 = x_1 + round(z_0\ b), \quad z_2 = z_0 - round(z_1\ a).$$

The procedure is left-invertible and its left-inverse is $x_0 = w_2$, $x_1 = w_1$, where

$$w_0 = y_0 + round(y_1\ a), \quad w_1 = y_1 - round(w_0\ b), \quad w_2 = w_0 + round(w_1\ a).$$

Further, the component-wise truncation error can be estimated by

$$\begin{aligned} |\hat{y}_0 - y_0| &\leq \left[2 + a + a^2 + \sin\varphi\left(1 + a + \tan\frac{\varphi}{2}\right)\right] 2^{k-j} + \frac{1}{2}\,(2 + a - ab), \\ |\hat{y}_1 - y_1| &\leq (1 + a + \sin\varphi)\ 2^{k-j} + \frac{1}{2}\,(1 + b), \end{aligned} \tag{9.58}$$

where k denotes the number of bits to represent $\mathbf{x}$.

□

The formulae for y_0, y_1 and x_0, x_1 (after inverse transform) directly follow by applying the input vector $\mathbf{x}$ to three matrices in (9.54). The elegant proof of truncation error estimates (9.58) can be found in [17]. The component-wise error estimates given by (9.58) depend on the range of input vector $\mathbf{x}$ (k is fixed) and approximation quality. Improving the approximation (j is large) leads to arbitrary small error estimates for the first two terms due to the fact that the terms within brackets are bounded. On the other hand, in [17] it has been indicated that the component-wise error estimates of integer approximated $\boldsymbol{G}_2^{(\varphi)}\mathbf{x}$ by the rounding operator [16] are given by (see also Theorem 9.1)

$$|\hat{y}_0 - y_0| \;\le\; \frac{1}{2}\left(1 + \tan\frac{\varphi}{2} + \cos\varphi\right), \qquad |\hat{y}_1 - y_1| \;\le\; \frac{1}{2}\,(1 + \sin\varphi). \tag{9.59}$$

Note that the second terms, specifically $\frac{1}{2}\,(2 + a - ab)$ and $\frac{1}{2}\,(1 + b)$ in (9.58) represent the component-wise error estimates in (9.59).

Note 7: The so-called SOPOT (Sum-Of-Powers-Of-Two) approximation of a floating-point multiplier was introduced in [23, 28, 34]. The SOPOT approximation is defined as

$$\sum_{k=1}^{M} a_k\, 2^{b_k}, \quad a_k \in \{-1, 0, 1\}, \quad b_k \in \{-r, \ldots, -1, 0, 1, \ldots, r\},$$

where r is the range of multiplier and M is the number of terms. Thus, each floating-point multiplier can be replaced by the limited number of additions and shifts. However, the SOPOT approximation is actually a dyadic rational number after summing up the terms.

9.3.2.3 Optimization Strategies to Minimize the Approximation Error

Practical implementations of the ***LUL*** as well as ***ULU*** computational structures of $\boldsymbol{G}_2^{(\varphi)}$ and $\boldsymbol{G}_2^{(-\varphi)}$ by a rotation angle φ to integer approximation of transforms by the rounding operator or by the dyadic approximation can be further optimized to minimize the approximation error. There are two optimization strategies which have been proposed, one in [25] for the dyadic approximation of $\boldsymbol{G}_2^{(\varphi)}$, and one in [31, 45, 46] for the integer approximation by rounding operator. Again, due to practical reasons, consider the ***ULU*** factorization of $\boldsymbol{G}_2^{(\varphi)}$ given by (9.54) with off-diagonal elements denoted by $c_0 = \tan\frac{\varphi}{2} = \frac{1-\cos\varphi}{\sin\varphi} = \frac{\sin\varphi}{1+\cos\varphi}$ and $c_1 = \sin\varphi \neq 0$.

The first optimization strategy is based on controlling the dynamic range of off-diagonal floating-point multipliers in unit triangular matrices approximated by dyadic rationals [25]. The value of multiplier c_1 can be arbitrarily small and consequently, the dynamic range of multiplier c_0 can be arbitrarily large. Therefore, we should control the range of the multipliers c_0 and c_1 as follows. If the rotation angle $\varphi \in (-\frac{\pi}{2}, 0) \cup (0, \frac{\pi}{2})$, then the value of $\cos\varphi > 0$ and

$$|c_0| = \left|\tan\frac{\varphi}{2}\right| = \left|\frac{1-\cos\varphi}{\sin\varphi}\right|$$
$$= \left|\frac{\sin\varphi}{1+\cos\varphi}\right| = |\sin\varphi|\left|\frac{1}{1+\cos\varphi}\right| < |\sin\varphi| = |c_1| < 1, \qquad (9.60)$$

and multipliers c_0 and c_1 fall between -1 and $+1$. However, if the rotation angle $\varphi \in (-\pi, \frac{\pi}{2}) \cup (\frac{\pi}{2}, \pi)$, then the value of $\cos\varphi < 0$ and $|c_0| > 1$ which is not desirable. The absolute values of multipliers c_0 and c_1 can be controlled to be always less than or equal to 1 by replacing $\boldsymbol{G}_2^{(\varphi)}$ with its alternative equivalent form $-\boldsymbol{G}_2^{(\varphi+pi)}$ as [25]

$$\boldsymbol{G}_2^{(\varphi)} = -\boldsymbol{G}_2^{(\varphi+\pi)} = -\begin{pmatrix} -\cos\varphi & \sin\varphi \\ -\sin\varphi & -\cos\varphi \end{pmatrix} = -\begin{pmatrix} 1 & -\bar{c}_0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ c_1 & 1 \end{pmatrix}\begin{pmatrix} 1 & -\bar{c}_0 \\ 0 & 1 \end{pmatrix}, \qquad (9.61)$$

where

$$\bar{c}_0 = \frac{1+\cos\varphi}{\sin\varphi} = \frac{1}{\tan\frac{\varphi}{2}}, \quad \text{and} \quad c_1 = \sin\varphi \neq 0.$$

When rotation angle $\varphi \in (-\pi, \frac{\pi}{2}) \cup (\frac{\pi}{2}, \pi)$, then we have

$$|\bar{c}_0| = \left|\frac{1}{\tan\frac{\varphi}{2}}\right| = \left|\frac{1+\cos\varphi}{\sin\varphi}\right|$$
$$= \left|\frac{\sin\varphi}{1-\cos\varphi}\right| = |\sin\varphi|\left|\frac{1}{1-\cos\varphi}\right| < |\sin\varphi| = |c_1| < 1, \qquad (9.62)$$

and new multipliers $\bar{c}_0$ and c_1 fall again between -1 and $+1$. Consequently, the number of required additions in a multiplier-less implementation of integer transform is reduced. In order to obtain the corresponding ***ULU*** computational structure of $-\boldsymbol{G}_2^{(\varphi+pi)}$ we need to modify the ***ULU*** structure of $\boldsymbol{G}_2^{(\varphi)}$ replacing the multiplier c_0 by $\bar{c}_0$ and changing signs of the result.

The second optimization strategy is based on modeling the approximation error as a quantization noise introduced after each rounding operator in ***ULU*** structure, whereby this quantization noise in each step is independent and identically distributed. For improving the reversible integer rotation $\boldsymbol{G}_2^{(\varphi)}$ resulting in the smaller quantization noise, three other ***ULU*** structures have been proposed defined as [31, 45, 46]

$$\boldsymbol{G}_2^{(\varphi)} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\begin{pmatrix} 1 & -\frac{\sin\varphi+1}{\cos\varphi} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ \cos\varphi & 1 \end{pmatrix}\begin{pmatrix} 1 & -\frac{\sin\varphi+1}{\cos\varphi} \\ 0 & 1 \end{pmatrix}, \qquad (9.63)$$

$$\boldsymbol{G}_2^{(\varphi)} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{\sin\varphi - 1}{\cos\varphi} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \cos\varphi & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{\sin\varphi - 1}{\cos\varphi} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{9.64}$$

$$\boldsymbol{G}_2^{(\varphi)} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{\cos\varphi + 1}{\sin\varphi} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \sin\varphi & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{\cos\varphi + 1}{\sin\varphi} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{9.65}$$

where also $\cos\varphi \neq 0$. Compared to (9.54), additional ***ULU*** factored forms (9.63)–(9.65) include pre/post-swapping and pre/post-sign changing operations applied to the components of input and/or output vector, and they lead to different multipliers c_0 and c_1 for the same rotation angle φ. Now, the problem is to select the optimal ***ULU*** factored form for different rotation angle φ such that it achieves the lowest quantization noise in the MSE sense [46].

Let $[x, y]^T$ be the input vector, $[x', y']^T$ be the output vector, and Δ be the quantization noise of one rounding operator, i.e., $\text{round}(a) = a + \Delta$. Then the reversible integer rotation $\hat{\boldsymbol{G}}_2^{(\varphi)}$ applied to the vector $[x, y]^T$ is given by

$$\begin{aligned} z &= x - \text{round}(c_0\, y) = x - (c_0\, y + \Delta_0), \\ x' &= y + \text{round}(c_1\, z) = y + (c_1\, z + \Delta_1), \\ y' &= z - \text{round}(c_0\, x') = z - (c_0\, x' + \Delta_2). \end{aligned} \tag{9.66}$$

where Δ_i, $i = 0, 1, 2$ is the quantization noise of successive rounding operators. Further, let $\mathbf{e} = [\Delta x', \Delta y']^T$ be the quantization noise of the reversible integer rotation. The goal is to minimize the MSE, i.e., $E\,[\mathbf{e}\mathbf{e}^T] = E\,[(\Delta x')^2] + E\,[(\Delta y')^2]$ has to be minimal, where $E\,[.]$ is the expectation operator. Using Eq. (9.66) after some algebraic manipulation we obtain the terms $\Delta x'$ and $\Delta y'$, respectively, as

$$\Delta x' = c_1\Delta_0 - \Delta_1, \qquad \Delta y' = (1 - c_0c_1)\Delta_0 + c_0\Delta_1 + \Delta_2. \tag{9.67}$$

The MSE of the quantization noise of the reversible integer rotation is calculated as

$$E\,[(\Delta x')^2] + E\,[(\Delta y')^2] = \{(1 - c_0c_1)^2 + c_0^2 + c_1^2 + 2\}\, E\,[\Delta^2], \tag{9.68}$$

where $E\,[\Delta^2] = \frac{1}{12}$ is the average energy of quantization noise of a single rounding operator. If we plot the quantization noise versus the rotation angles for different ***ULU*** factored forms (9.54), (9.63)–(9.65), we can observe the plots as parabolic curves (see Fig. 2 in [46]) which indicate that for any single ***ULU*** factorization, the quantization noise is fairly large at certain rotation angles [46].

This fact enables us to select the optimal ***ULU*** factored form for different rotation angle φ as follows: When the rotation angle $\varphi \in (-\frac{\pi}{4}, \frac{\pi}{4})$, then the reversible integer

rotation $\boldsymbol{G}_2^{(\varphi)}$ is realized by (9.54). ***ULU*** factored form (9.63) is used when $\varphi \in (-\frac{3\pi}{4}, -\frac{\pi}{4})$, (9.64) when $\varphi \in (\frac{\pi}{4}, \frac{3\pi}{4})$, and (9.65) is used when $\varphi \in (\frac{3\pi}{4}, \frac{5\pi}{4})$.

Just discussed local methods have been used to integer approximation of the DCT-II, the DCT-IV [9, 13, 14, 16, 17], the DFT or FFT [23–25, 27, 45, 48], the lapped orthogonal transforms or the MLT [28–31, 33, 34, 54, 55], and of the MDCT [36–40, 45, 46]. Although the DCT-II, DCT-IV and DFT are linear transforms which are generated by the corresponding matrices, naturally, by introducing the nonlinear rounding operator into computational steps, the invertible (reversible) DCTs and DFTs are no longer linear mappings. Thus, such invertible integer transforms are understood to be nonlinear invertible mappings which act on Z^N (N-dimensional vector space over Z) and approximate the classical original transforms [16].

9.3.3 Modulo Transforms

A nice elegant theory and construction of a class of scaled reversible (rotation) transforms with integer elements and unit determinants has been developed in [19]. This class of integer transforms is built on a foundation of modulo arithmetic and critical quantization of transform coefficients. Therefore, these transforms are called the modulo transforms. Some interesting properties of modulo transforms in modular arithmetic lead to reversible integer rotations which are critically quantized without loss of information, as an alternative to ***LUL*** or ***ULU*** structures. The modulo transforms are non-expansive, and therefore, are suitable for lossless signal compression and digital image rotation [19].

A linear transform $\boldsymbol{A} \in R^{N\times N}$ applied to the vector $\mathbf{x} \in R$ is defined as

$$\mathbf{y}^T = \boldsymbol{A}\ \mathbf{x}^T,$$

where $\mathbf{y}$ is the transform domain representation of $\mathbf{x}$. For data compression, the transform $\boldsymbol{A}$ is required to be orthonormal, i.e., $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$, or it is required to be orthogonal, i.e., $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{A}^T\boldsymbol{A} = \lambda\ \boldsymbol{I}$, $\lambda \in R$. When elements of $\boldsymbol{A}$ are integers, $\boldsymbol{A}$ is an integer transform since it maps integers to integers. The original vector $\mathbf{x}$ can be recovered from $\mathbf{y}$ by the inverse transformation as

$$\mathbf{x}^T = \boldsymbol{A}^T\ \mathbf{y}^T, \qquad \text{or} \qquad \mathbf{x}^T = \frac{1}{\lambda}\boldsymbol{A}^T\ \mathbf{y}^T.$$

In general, any nontrivial linear transform $\boldsymbol{A}$ with integer elements results in an expansion of the range (or more exactly, the volume) of transformed data compared to the volume of the original data. In other words, the set of points located at integer lattice transformed by $\boldsymbol{A}$ is mapped to the volume expanded transform domain (or expanded bounding volume) increased by the expansion factor of $\text{vol}(\boldsymbol{A}) = |\det\ (\boldsymbol{A})|$. Due to the volume expansion such integer linear transforms cannot be used in lossless data compression. As an example, the expansion factor of non-

scaled 2×2 Hadamard matrix $\boldsymbol{H}_2$ given by (9.50) is $\text{vol}(\boldsymbol{H}_2) = |\det(\boldsymbol{H}_2)| = 2$. But, by scaled factored $\boldsymbol{H}_2$ with introduced rounding operator, the set of points located at integer lattice transformed by the scaled $\boldsymbol{H}_2$ is mapped back to the same transform domain volume as the time domain volume (normalized transform). Thus, the scaled $\boldsymbol{H}_2$ has unit volume [19].

In the following are presented the definitions leading to the concept of the critical quantization of transform space, a critically quantized transform, and modular arithmetic [19].

Definition 9.2 The mapping $F : \mathbf{x} \in Z^N \rightarrow \mathbf{y} \in Z^N$ is said to be reversible if $F(x_0) = F(x_1) \Rightarrow x_0 = x_1$.

Definition 9.3 The quantization $\boldsymbol{Q}$ of space Y is a partition of Y into some number K of discrete cells or quantization bins. K may be countably infinite.

Definition 9.4 A valid transform point is a point in the transform space Y that is mapped by an element of the original domain $X = Z^N$, and the transform $\boldsymbol{A}$ is invertible, i.e., $\det(\boldsymbol{A}) \neq 0$.

Definition 9.5 $\boldsymbol{Q}$ is a critical quantization of the transform space Y if every quantization bin contains one and only one valid transform point generated by transforming the integer lattice X.

Definition 9.6 $\boldsymbol{A}$ is a critically quantized transform if a one-to-one mapping exists between the integer lattice representing the domain of $\boldsymbol{A}$ and the integer lattice representing the range of $\boldsymbol{A}$, i.e., $X \rightarrow Y$. Therefore, a critically quantized linear transform $\boldsymbol{A}$ defines a reversible mapping with volume preservation, i.e., $\text{vol}(\boldsymbol{A}) = \det(\boldsymbol{A}) = 1$.

It is important to note that the cascades of transforms are critically quantized. When quantization is separable and uniform scalar, the effect of quantization along a certain dimension is basically a rounded division by the bin width along that particular dimension, and this division can be incorporated into the transform itself [19].

The simplest nontrivial linear transform is two-point transform. Let us now consider a two-point transform or 2×2 rotation matrix $\boldsymbol{R}_2$ given by

$$\boldsymbol{R}_2 = \begin{pmatrix} c & -s \\ s & c \end{pmatrix}, \qquad c, s \in R. \tag{9.69}$$

If $\boldsymbol{R}_2$ is orthonormal, then $c^2 + s^2 = 1$. Now, relax the restriction of orthonormality of $\boldsymbol{R}_2$ and instead restrict c and s to be integers, i.e., $\boldsymbol{R}_2$ is integer rotation matrix. Such rotation may be derived from rational approximations to the desired rotation angle of cosine and sine terms such that they have no common factors. Integers a and b having no common factors other than 1, i.e., $(a, b) = 1$, are said to be mutually prime or relatively prime [83]. The resulting rotation $\boldsymbol{R}_2$ is scaled orthonormal and it is an expansive transformation with an expansion factor of $\det(\boldsymbol{R}_2) = D = c^2 + s^2 \in Z^+$. However, integer rotation $\boldsymbol{R}_2$ can be reformulated as a non-expansive transformation as follows.

Note 8: For understanding the basic elements of elementary number theory, such as divisibility of integers and congruences (modulo relationships), the reader is referred to the book [83].

$\boldsymbol{R}_2$ is rank deficient (see Appendix A.1) in modular $D = c^2 + s^2$ arithmetic, i.e., $\det(\boldsymbol{R}_2) = c^2 + s^2 = D$ is zero in modular D arithmetic. For the transformation $\mathbf{y}^T = \boldsymbol{R}_2\, \mathbf{x}^T$, where $[x_0, x_1]^T$ and $[y_0, y_1]^T$ are input and output vectors, respectively, the transform coefficients y_0 and y_1 are redundant in modular D arithmetic. Indeed, considering the matrix-vector products $\mathbf{y}^T = \boldsymbol{R}_2\, \mathbf{x}^T$ and $\mathbf{x}^T = \boldsymbol{R}_2^T\, \mathbf{y}^T$, the quantity $c\, y_0 + s\, y_1$ is evaluated as

$$c\, y_0 + s\, y_1 = c(c\, x_0 - s\, x_1) + s(s\, x_0 + c\, x_1) = (c^2 + s^2)\, x_0 + 0\, x_1 = 0 \mod D. \tag{9.70}$$

Equation (9.70) is referred to as the modulo relationship. Since c and s have no common factors, c, s and D have no pairwise common factors. Then there exist inverses c^{-1} and s^{-1} in modular D arithmetic such that $c^{-1}\, c = s^{-1}\, s = 1 \mod D$. From Eq. (9.70) it follows that the modular relationship can also be expressed as

$$y_0 = c^{-1}\, s\, y_1 \mod D, \qquad \text{or} \qquad y_0 \equiv c^{-1}\, s\, y_1 \mod D, \tag{9.71}$$

which implies that in modular D arithmetic one transform coefficient can be recovered from the other, thus proving their redundancy. Referring to the second equation in (9.71) it is noted that two integers a and b are congruent mod D denoted by $a \equiv b \mod D$, if $D/(a - b)$, or in other words, D divides $(a - b)$ [83]. The modulo relationship (9.70) also implies its dual one

$$c\, y_1 - s\, y_0 = 0 \mod D, \tag{9.72}$$

since $c\, y_1 - s\, y_0 = c^{-1}\, (c^2\, y_1 - cs\, y_0) = c^{-1}\, (s^2\, y_1 + cs\, y_0) = c^{-1} s\, (s\, y_1 + c\, y_0) = 0 \mod D$. For a base D, any integer k can be represented in two parts corresponding to its quotient and remainder with divisor D. When a consistent rounding is applied for both positive and negative k, the remainder corresponds to $k \mod D$. Let the quotient be written as $k \text{ div } D$. Then,

$$k = (k \text{ div } D)\, D + (k \mod D). \tag{9.73}$$

Definition 9.7 The rounded division sdiv is defined for integer a and n as

$$\text{sdiv}(a, n) = \left\lfloor \frac{a + \lfloor \frac{n}{2} \rfloor}{n} \right\rfloor, \tag{9.74}$$

where $\lfloor . \rfloor$ denotes the rounding operator. When quantization bins are centered around multiples of n, the quantized index of a is equal to $\lfloor (a + \lfloor n/2 \rfloor)/n \rfloor$. The location of a within its quantization bin with respect to its center is given by the signed modulus *smod*. The range of $\text{smod}(a, n)$ is centered around zero for odd n. For even n, a consistent rounding rule is chosen for $\forall\ a$: for instance rounding is always toward $-\infty$ [19].

Definition 9.8 The sign modulo operation *smod* is defined as

$$\text{smod}(a, n) = a - n\ \text{sdiv}(a, n) \in \left\{ -\left\lfloor \frac{n}{2} \right\rfloor \ldots \left\lfloor \frac{n-1}{2} \right\rfloor \right\}. \tag{9.75}$$

Thus, any scaled orthonormal two-point transform $\boldsymbol{A}$ can be reformulated as a non-expansive or critically quantized by exploiting the modulo relationship, although the resulting transform may be only orthogonal. Any transform domain point satisfying modulo relationships (9.70) and (9.72) is a valid transform domain point generated by a pair of integers in the spatial domain. As a result, the Givens–Jacobi rotation given by (9.45) can be approximated arbitrarily closely by the integer rotation matrix $\boldsymbol{R}_2$ given by (9.69) such that c and s being integers have no common factors. Critically quantized transforms exploiting the modulo relationships are referred to the modulo transforms [19].

In order to construct modulo transforms, the perfectly square quantization partitions that guarantee the critical quantization are specific integer Pythagorean triples $(c, s, \sqrt{D})$, where D is a square.

9.3.3.1 Construction of Modulo Transforms

For the integer Pythagorean triples $(c, s, \sqrt{D})$, where D is the square, the effective forward and inverse integer rotations with orthonormal matrix are, respectively, defined as [19]

$$\hat{\boldsymbol{R}}_2 = \frac{1}{\sqrt{D}} \begin{pmatrix} c & -s \\ s & c \end{pmatrix}, \qquad \hat{\boldsymbol{R}}_2^{-1} = \frac{1}{\sqrt{D}} \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \qquad c,\ s,\ D \in Z. \tag{9.76}$$

The inverse integer rotation matrix recovers the original data without any error. Thus, this construction can be used to generate integer rotation matrices for all primitive Pythagorean triples $(c, s, \sqrt{D})$ satisfying $D = c^2 + s^2$ with c, s having no common factors. In the following is presented an essential theorem claiming the Pythagorean triads which satisfy conditions of the uniform and critical quantization [19].

Theorem 9.3 *Any Pythagorean triad satisfying* $c = \pm(\sqrt{D}-1)$ *or* $s = \pm(\sqrt{D}-1)$, *i.e.,* $\pm\sqrt{D} = c+1$ *or* $\pm\sqrt{D} = s+1$, *is critically quantized using uniform quantizers with square bins of size* $\sqrt{D} \times \sqrt{D}$, *when the quantization grid is arranged such that the origin lies at the center of its bins.*

□

Let $n > 1$ be an odd number. Theorem 9.3 is satisfied when the Pythagorean triads $(c, s, \sqrt{D})$, $s < c$, s is odd and c is even, are defined as [73]

$$s = n, \qquad c = \frac{1}{2}(n^2 - 1), \qquad \sqrt{D} = \frac{1}{2}(n^2 + 1), \tag{9.77}$$

where $\sqrt{D} = c + 1$, and hence the Pythagorean triads are simply given by $(c, s, c + 1)$.

Note 9: For $n > 1$ there are other Pythagorean triads which are defined by the following formulae [73]

$$s = 4n, \qquad c = 4n^2 - 1, \qquad \sqrt{D} = 4n^2 + 1.$$

Since $\sqrt{D} = c + 2$, they do not satisfy the conditions of Theorem 9.3, and consequently, they cannot be used to construct the modulo transforms. In fact, they will generate a mapping which will not be one-to-one. Famous Greek mathematician Euclid gave formulae however without a proof to generate all Pythagorean triads as [73]

$$s = t\,(a^2 - b^2), \qquad c = 2tab, \qquad \sqrt{D} = t\,(a^2 + b^2),$$

where t, a and b are arbitrary positive integers such that $a > b$, a and b have no prime factors in common, and one of a or b is odd, the other even.

From definition of $\tan\varphi = \frac{s}{c}$ and the relation $c^2 + s^2 = (c+1)^2$ we have

$$c = \frac{s^2 - 1}{2}, \qquad s < c, \tag{9.78}$$

which produces the rotation angles

$$\varphi = \arctan\left(\frac{2s}{s^2 - 1}\right). \tag{9.79}$$

In general, s and c may be interchanged to overcome the inherent constraints of odd s and even c. For the integer rotations defined as modulo transforms only rotations between zero and $\frac{\pi}{4}$ are of interest, since other rotations can be realized by negating the rotation angle or adding an arbitrary number of right angle rotations which are trivial transforms of the form $\begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix}^n$, or a combination of both. As the value of s increases, the angles get progressively smaller. There are countably infinite permissible rotation angles corresponding to the odd values of s. Based on (9.77)–(9.79) the first ten Pythagorean triads $(c, s, c+1)$ with corresponding angles in radians are shown in Table 9.1.

From Table 9.1 it can be inferred that there exist good approximations to small rotation angles. Computational structures for the forward and inverse integer rotations $\hat{\boldsymbol{R}}_2$, $\hat{\boldsymbol{R}}_2^{-1}$ based on the Pythagorean triads $(c, s, c+1)$, respectively, are shown in Fig. 9.7a, b. Boxes labeled as $\mathrm{div}(c+1)$ perform the operations $\mathrm{sdiv}(a, c+1)$ on the input a. The modulo transforms based on Pythagorean triads $(c, s, c+1)$ are implemented with integer multiplications and integer divisions.

Larger rotations may be realized as a finite-length cascade of smaller rotations of the form (9.79). As an example, consider the scaled Hadamard matrix $\boldsymbol{H}_2$ given by (9.50). Based on (9.51) the matrix-vector product $\mathbf{y}^T = \boldsymbol{H}_2\,\mathbf{x}^T$ may be converted

Table 9.1 Pythagorean triads satisfying the conditions of Theorem 9.3

Index	s	c	$\sqrt{D} = c+1$	Angle φ in radians
1	3	4	5	0.643 501 109
2	5	12	13	0.394 791 120
3	7	24	25	0.283 794 109
4	9	40	41	0.221 314 442
5	11	60	61	0.181 319 774
6	13	84	85	0.153 543 783
7	15	112	113	0.133 136 328
8	17	144	145	0.117 511 645
9	19	180	181	0.105 166 123
10	21	220	221	0.095 166 207

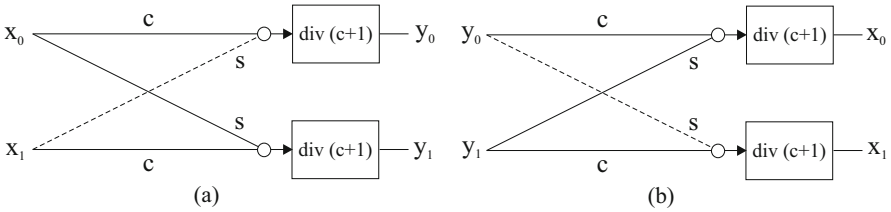

Fig. 9.7 Computational structures for the integer rotations based on Pythagorean triads $(c, s, c+1)$: (**a**) Forward, (**b**) Inverse

to the matrix-vector product $\mathbf{y}^T = \boldsymbol{G}_2^{(\frac{\pi}{4})}\hat{\mathbf{x}}^T$, where $\hat{\mathbf{x}} = [x_0, -x_1]$, and $\boldsymbol{G}_2^{(\frac{\pi}{4})}$ is the forward Givens–Jacobi rotation by the angle $\frac{\pi}{4}$. The best two-stage approximation of the desired rotation angle $\frac{\pi}{4} = 2\,\frac{\pi}{8}$ by two modulo transforms is obtained by two successive rotations of $\varphi = \arctan\,((2 \times 5)/(5^2 - 1)) = 0.394791$. Thus, each integer rotation is based on the Pythagorean triad $(5, 12, 13)$ approximating the rotation angle $\frac{\pi}{8} \cong 0.392699$. The effective integer rotation is given by the matrix

$$\begin{pmatrix} 12 & -5 \\ 5 & 12 \end{pmatrix}^2 = \begin{pmatrix} 119 & -120 \\ 120 & 119 \end{pmatrix}.$$

In Fig. 9.8 is shown the realization of approximation of $\mathbf{y}^T = \boldsymbol{G}_2^{(\frac{\pi}{4})}\hat{\mathbf{x}}$ by the cascade of two modulo transforms resulting in an orthonormal and reversible transform.

In general, the theory of modulo transforms can be extended to an $N \times N$ orthonormal transform $\boldsymbol{A}$. It is well known that based on $\boldsymbol{QR}$ factorization (see Appendix A.6), $\boldsymbol{A}$ can be decomposed and implemented by a product of at most $N(N-1)/2$ Givens–Jacobi rotations $\boldsymbol{G}_{i,j}^{(\varphi)}$. Thus, the transformation $\boldsymbol{A} = \prod_{i,j} \boldsymbol{G}_{i,j}^{(\varphi)}$ is quantized critically if $\forall\ \boldsymbol{G}_{i,j}^{(\varphi)}$ are quantized critically. It means that in the theory of modulo transforms any higher order orthonormal integer transform can be approximated by a cascade of modulo transforms which ensure the reversibility property. As an example, the orthonormal 4-point DCT-II approximated by modulo transforms is shown in [19].

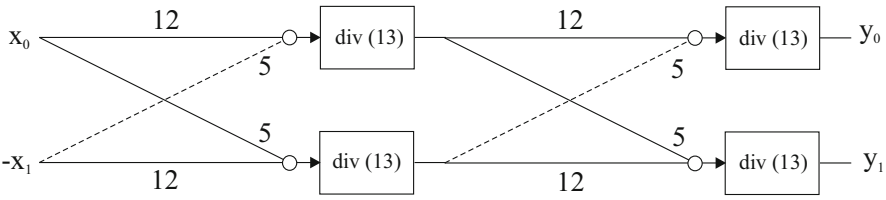

Fig. 9.8 Modulo transform approximation of $\mathbf{y}^T = \boldsymbol{G}_2^{(\frac{\pi}{4})}\hat{\mathbf{x}}$

9.3.3.2 Computational Aspects of Modulo Transforms

Modulo transforms based on Pythagorean triads $(c, s, c+1)$ satisfying Theorem 9.3 are implemented with integer multiplications and integer divisions. The forward two-point modulo transform is generated by Srinivasan [19]

$$\begin{aligned} y_0 &= \text{sdiv}(c\, x_0 - s\, x_1, c+1), \\ y_1 &= \text{sdiv}(s\, x_0 + c\, x_1, c+1), \end{aligned} \tag{9.80}$$

while the inverse two-point modulo transform is generated by

$$\begin{aligned} x_0 &= \text{sdiv}(\, c\, y_0 + s\, y_1, c+1), \\ x_1 &= \text{sdiv}(-s\, y_0 + c\, y_1, c+1), \end{aligned} \tag{9.81}$$

where

$$\text{sdiv}(a, c+1) = \left\lfloor \frac{a + \lfloor \frac{c+1}{2} \rfloor}{c+1} \right\rfloor, \tag{9.82}$$

and the divisor $c+1$ is always an odd number. A common method of realizing the integer division by a constant is by doing a multiply and shift. The multiplier is related to the reciprocal of the divisor $c+1$. An offset is added to prior shift to do rounded division as required. The size of accumulator for multiply operation is the bit range of dividend plus a constant. If the input range is $\pm\, R$ in each dimension, the dividend range in (9.80) and (9.81) is $\pm(c+s)\, R$. The dividend range can be reduced to $\pm(s+1)\, R$ because (9.80) is equivalent to

$$\begin{aligned} y_0 &= x_0 - \text{sdiv}(x_0 + s\, x_1, c+1), \\ y_1 &= x_1 + \text{sdiv}(s\, x_0 - x_1, c+1), \end{aligned} \tag{9.83}$$

and (9.81) is equivalent to [19]

$$\begin{aligned} x_0 &= y_0 + \text{sdiv}(-y_0 + s\, y_1, c+1), \\ x_1 &= y_1 - \text{sdiv}(s\, y_0 + y_1, c+1). \end{aligned} \tag{9.84}$$

Although (9.83) and (9.84) still involve two divisions, the multiply and shift division can be implemented using a smaller accumulator. It is noted that in practical implementations $\mathrm{sdiv}(a, c+1)$ operation may be replaced by the rounding operator round $(\frac{a}{c+1})$.

It is interesting to compare modulo transforms with ***LUL*** or ***ULU*** structures from computational point of view. The modulo transform components may be evaluated quite independently in the corresponding computational structure, whereas in ***LUL*** (***ULU***) structures a cascade of three multiplications must be implemented subsequently and in high precision arithmetic for the desired rotation to obtain the final accurate result. Moreover, intermediate integer approximations by the rounding operator or dyadic approximation lead to an effective reversible transformation which is not a pure rotation. On the other hand, the modulo transform always results in a rotation, even when the rotation angle is only approximated [19].

In summary, the modulo transforms have interesting properties in modular arithmetic. Based on these properties, certain integer rotations can be critically quantized without loss of information. Thus, modulo transforms are very suitable for lossless data compression as a viable alternative to the ***LUL*** or ***ULU*** structures for realizing the reversible integer transforms and reversible integer rotations.

9.3.4 Infinity-Norm Rotation Transforms

A class of reversible dynamic-range preserving one-to-one mappings to integer transform approximation, the so-called infinity-norm rotation transforms, has been developed in [22]. They are analogous to the general linear 2-norm rotation transforms such as the Givens–Jacobi rotations. While the linear 2-norm rotation transforms preserve the 2-norm (Euclidean norm) of the rotated vectors, the infinity-norm rotation transforms are piecewise linear ones preserving the infinity-norm of vectors. Besides the advantages of perfect reversibility property and of maintaining the constant dynamic range, the infinity-norm rotation transforms have also good energy-compact ability. Their in-place implementation is based on products of matrices being unit lower and unit upper triangular matrices with unit determinants [22].

In the following the theory of infinity-norm rotations and their important properties are discussed at first followed by their piecewise linear implementation preserving the constant dynamic range for integer-to-integer approximation.

9.3.4.1 Infinity-Norm Rotation

In general, a linear orthogonal transform is a rotation in Euclidean space, which preserves the 2-norm of the rotated vectors. Similarly, a rotation that preserves the dynamic range is a rotation in the infinity-norm space, called the infinity-norm rotation [22]. It is well known that the vector norms are derived from the general class of the so-called p-norms (for more details, see Appendix A.3).

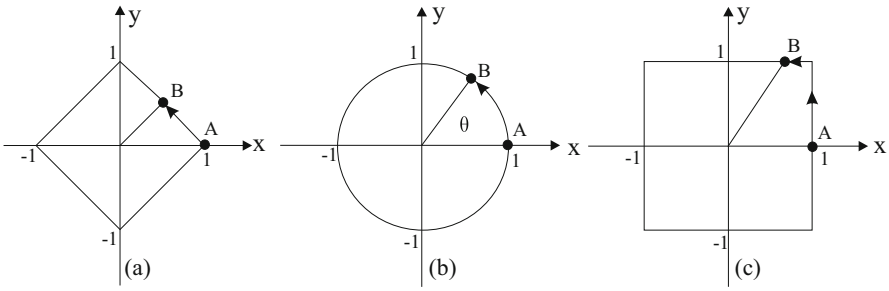

Fig. 9.9 Unit p-norm rotations in 2-D space: (**a**) 1-norm rotation, (**b**) 2-norm rotation, (**c**) infinity-norm rotation

Definition 9.9 (Notion of p-Norm Rotation That Preserves p-Norm of the Vectors After the Rotation) p-norm isometry set S_d^p is the set of vectors $\mathbf{x} = [x_0, x_1, \ldots, x_{n-1}]$ such that the p-norms of all vectors are equal to d, i.e.,

$$S_d^p = \{\mathbf{x} : \ \mathbf{x} \in l^p \wedge \|\mathbf{x}\|_p = d\}. \tag{9.85}$$

In general, S_d^p is called a p-norm circle, and S_1^p is called the unit circle of the p-norm isometry set. For illustration, a 1-norm circle in 2-D space is a rhomboid, a 2-norm circle is a normal circle in 2-D and a sphere in 3-D, the infinity-norm circle is a square in 2-D and a cube in 3-D (see Fig. 9.9).

Definition 9.10 (p-Norm Rotation) p-norm rotation $\boldsymbol{R}^{(p)}$ is the transform that preserves the p-norm of the vectors, and geometrically moves the end points of the vectors in the p-norm isometry set S_d^p, i.e.,

$$\boldsymbol{R}^{(p)} : \ S_d^p \ \rightarrow \ S_d^p, \ \forall \ \mathbf{x} \in S_d^p, \ \|\mathbf{x}\|_p = \|\boldsymbol{R}^{(p)} \ \mathbf{x}\|_p = d. \tag{9.86}$$

Thus, a rotation transform that preserves 1-norm, 2-norm, or the infinity-norm is a 1-norm rotation, 2-norm rotation, or an infinity-norm rotation, respectively. For illustration, these types of rotations in 2-D space are shown in Fig. 9.9. They are performed along the unit circles of 1-norm (the set S_1^1), 2-norm (the set S_1^2), and infinity norm (the set S_1^∞) counterclockwise with respect to the X axis.

Definition 9.11 (Angle of p-Norm Rotation) In 2-D space, the angle is the amount of p-norm rotation between two vectors in S_d^p, which is measured by the ratio of the runlength sweeping from the first vector to the second vector to the constant p-norm radius:

$$\theta_p = \frac{\text{runlength}}{\text{radius}}. \tag{9.87}$$

Thus, θ_2 = (arc length)/radius is the angle measure of a 2-norm rotation, and θ_∞ = (runlength)/(side length)/2 is the angle measure of an infinity-norm rotation.

After rotating a complete circle, $\theta_2 = 2\pi$ and $\theta_\infty = 8$. The equivalent relation between θ_∞ and θ_2 for $\theta_2 \in \langle 0, \frac{\pi}{4} \rangle$ is defined as

$$\theta_\infty = \tan \theta_2. \tag{9.88}$$

Especially, $\theta_\infty = 1$ when $\theta_2 = \frac{\pi}{4}$, and $\theta_\infty = 2$ when $\theta_2 = \frac{\pi}{2}$. Other relations can be found by combining the above rotations. These angle relations define the infinity-norm rotations preserving constant dynamic range to approximate the corresponding linear orthogonal transforms [22].

9.3.4.2 Properties of Infinity-Norm Rotations

Let $\boldsymbol{R}^{(\infty)}(\theta_\infty)$ denote the rotation from point A to point B along an infinity-norm isometry set by an angle θ_∞. The infinity-norm rotations have the following properties (some of them are also the properties of other p-norm rotations) [22]:

1. **Composition**: Composition of two infinity-norm rotations results in another infinity-norm rotation.
2. **Unique inverse**: An infinity-norm rotation has a unique inverse rotation, if the runlength is always the shortest.
3. **Group of infinity-norm rotations**: The set of all infinity-norm rotations of the same center and radius forms a group.
4. **Constant infinity norm**:

$$\|\mathbf{x}\|_\infty = \|\boldsymbol{R}^{(\infty)}(\theta_\infty)\, \mathbf{x}\|_\infty. \tag{9.89}$$

5. **Periodicity**:

$$\boldsymbol{R}^{(\infty)}(\theta_\infty)\, \mathbf{x} = \boldsymbol{R}^{(\infty)}(\theta_\infty + T)\, \mathbf{x}, \tag{9.90}$$

   where T is the period of the infinity rotation. In 2-D space, $T = 8$.
6. **Reversibility property**: For $\alpha_\infty,\ \beta_\infty \in < 0, T)$

$$\boldsymbol{R}^{(\infty)}(\alpha_\infty)\, \mathbf{x} = \boldsymbol{R}^{(\infty)}(\beta_\infty)\, \mathbf{x} \Leftrightarrow \alpha_\infty = \beta_\infty. \tag{9.91}$$

7. **Piecewise linearity**: In general, all sides of a p-norm isometry set S_d^p are linear in planes. Hence, the infinity-norm rotation from one point to another point on the same plane of S_d^∞ can be expressed as a piece of linear transforms. The piecewise linearity of infinity-norm rotation in 2-D plane is shown in Fig. 9.10.

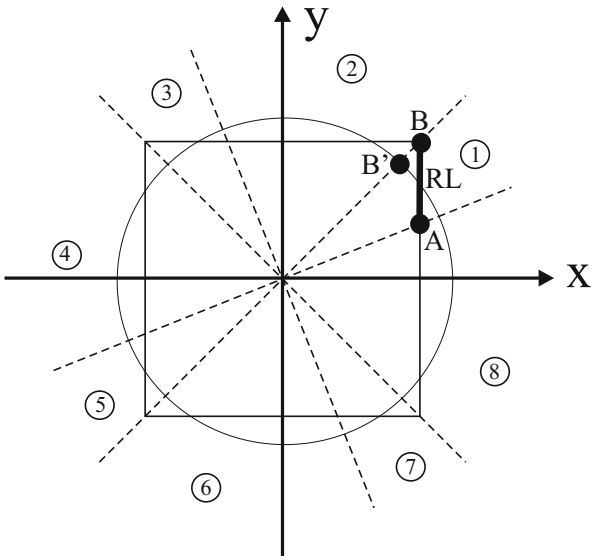

Fig. 9.10 The piecewise linearity of infinity-norm rotation in 2-D plane

9.3.4.3 Piecewise Linear Implementation of Infinity-Norm Rotations

All the infinity-norm rotations can be formulated as piecewise linear transforms in Euclidean space. With respect to Fig. 9.10 the *XY* plane is divided into eight regions by four dashed lines. In each region, the infinity-norm rotation is a linear transform. Denoting the *runlength* by *RL*, the corresponding formula for 2-D infinity-norm rotation of a vector $(x,\ y)^T$ is defined as [22]

$$\boldsymbol{R}^{(\infty)}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{cases} \begin{pmatrix} 1 & -\theta_\infty \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}, & y = a \ \wedge\ x \geq -a + \mathrm{RL}, \ \text{region 2}, \\ & y = -a \ \wedge\ x \leq -a - \mathrm{RL}, \ \text{region 6}, \\ \begin{pmatrix} 0 & -1 \\ 1 & 2 \end{pmatrix}\begin{pmatrix} 1 & -\theta_\infty \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}, & y = a \ \wedge\ x \leq -a + \mathrm{RL}, \ \text{region 3}, \\ & y = -a \ \wedge\ x \geq -a - \mathrm{RL}, \ \text{region 7}, \\ \begin{pmatrix} 1 & 0 \\ \theta_\infty & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}, & x = a \ \wedge\ y \geq a + \mathrm{RL}, \ \text{region 4}, \\ & x = -a \ \wedge\ y \leq a - \mathrm{RL}, \ \text{region 8}, \\ \begin{pmatrix} 2 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ \theta_\infty & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}, & x = a \ \wedge\ y \geq a - \mathrm{RL}, \ \text{region 1}, \\ & x = -a \ \wedge\ y \leq a + \mathrm{RL}, \ \text{region 5}, \end{cases} \tag{9.92}$$

where $a = \|(x, y)\|_\infty = \max\{|x|, |y|\}$. Matrices on the right-hand side of (9.92) are (unit and/or unit cross) lower and upper triangular, and using the algebra of (unit and/or unit cross) triangular matrices their inverses are easily obtained to define the inverse infinity-norm rotations. We note that the cross forms of (unit) triangular matrices are obtained from their original (unit) triangular forms by reversing the orders of their rows and columns [86].

Although the infinity-norm rotation can result in generally a non-integer output [see Eq. (9.88)], our interest is the reversible integer infinity-norm rotation. In this case, for the 2-D vector $(x,\ y)^T$ the value of *runlength* is defined as [22]

$$\text{runlength} = \text{round}\ (\|(x,\ y)\|_\infty\ .\ \theta_\infty), \tag{9.93}$$

where round() is the rounding operator. Thus, the infinity-norm rotation on the integer lattice can be formulated by (9.92) with *runlength* given by (9.93) [22]. To implement an infinity-norm rotation we need to find a proper rotation angle corresponding to the linear orthogonal transform to be approximated. For example, for the 2-norm rotation $\boldsymbol{R}^{(2)}$ by the angle $\frac{\pi}{4}$, according to (9.88) we find a rotation angle $\theta_\infty = 1$ for $\boldsymbol{R}^{(\infty)}$. Then, the integer approximation of $\boldsymbol{R}^{(2)}$ is obtained by $\boldsymbol{R}^{(\infty)}$ with θ_∞ implemented by (9.92) and (9.93).

In general, the theory of infinity-norm rotation transforms can be extended to a $N \times N$ orthonormal transform. It is well known that based on ***QR*** factorization (see Appendix A.6), the orthonormal transform can be decomposed and implemented by a product of at most $N(N-1)/2$ Givens–Jacobi rotations. It means that any higher order orthonormal integer transform can be obtained by a cascade of 2-D integer infinity-norm transforms which ensure the reversibility property. As an example, the orthonormal 3-point DCT-II approximated by a product of two infinity-norm rotations is shown in [22], where the orthonormal 3-point DCT-II is factored into the following rotation-based product of two matrices

$$\boldsymbol{C}_3^{II} = \sqrt{\frac{2}{3}} \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{3}}{2} & 0 & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & -1 & \frac{1}{2} \end{pmatrix} = \begin{pmatrix} 0 & \frac{\sqrt{3}}{3} & \frac{\sqrt{6}}{3} \\ 1 & 0 & 0 \\ 0 & -\frac{\sqrt{6}}{3} & \frac{\sqrt{3}}{3} \end{pmatrix} \begin{pmatrix} \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ 0 & 1 & 0 \\ \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \end{pmatrix}. \tag{9.94}$$

In summary, the infinity-norm rotation transforms enable us to construct the reversible integer rotations. Thus, they are also very suitable for lossless data compression as a viable alternative to the ***LUL*** or ***ULU*** structures, and to the modulo transforms for realizing the reversible integer transforms and reversible integer rotations.

9.3.5 *Construction of the Analysis/Synthesis IntMLT Filter Banks for MP3*

In order to illustrate just discussed local methods to integer approximation, specifically the reversible ***LUL*** or ***ULU*** computational structures, the reversible integer modulo transform rotations, and the reversible integer infinity-norm transform rotations, this subsection is focused on integer approximation of the complete analysis/synthesis MLT for MP3, or equivalently, the complete analysis/synthesis MDCT filter banks with the associated sine windowing function.

As we mentioned the local methods to integer approximation are rather suitable for the short transform sizes. Indeed, the MP3 audio coding standard [78] for the time-to-frequency transformation of an audio signal and vice versa has adopted the analysis and synthesis MLT filter banks [82] operating on the blocks of 12 samples (the short blocks) or the blocks of 36 samples (the long blocks). From theory of fast algorithms it is well known that using a permutation, the MLT or MDCT block transforms can be always converted to the DCT-IV of half size. Although many efficient implementations of the forward and backward MLT or MDCT block transforms for MP3 have been developed up to now (see Chap. 5 for more details) for integer approximation of the complete analysis and synthesis MLT filter banks, we will consider the fast analysis MDCT filter bank with the incorporated windowing and overlap procedure given by (9.30) and (9.34), and the fast synthesis MDCT filter bank with incorporated windowing and overlap and add procedure given by (9.35) and (9.40). In particular, from the definition of sine windowing function we have: $w_n = s_n = \sin\frac{\pi(2n+1)}{2N}$, and $w_{\frac{N}{2}-1-n} = c_n = \cos\frac{\pi(2n+1)}{2N}$, $n = 0, 1, \ldots, \frac{N}{4} - 1$. The corresponding compact computational structures for the efficient implementation of analysis MLT filter bank with incorporated windowing and overlap procedure, and synthesis MLT filter bank with incorporated windowing and overlap and add procedure are shown in Fig. 9.11a, b, respectively.

From Fig. 9.11 it can be seen that the compact computational structures for the efficient implementation of both the analysis and synthesis N-point MLT filter banks

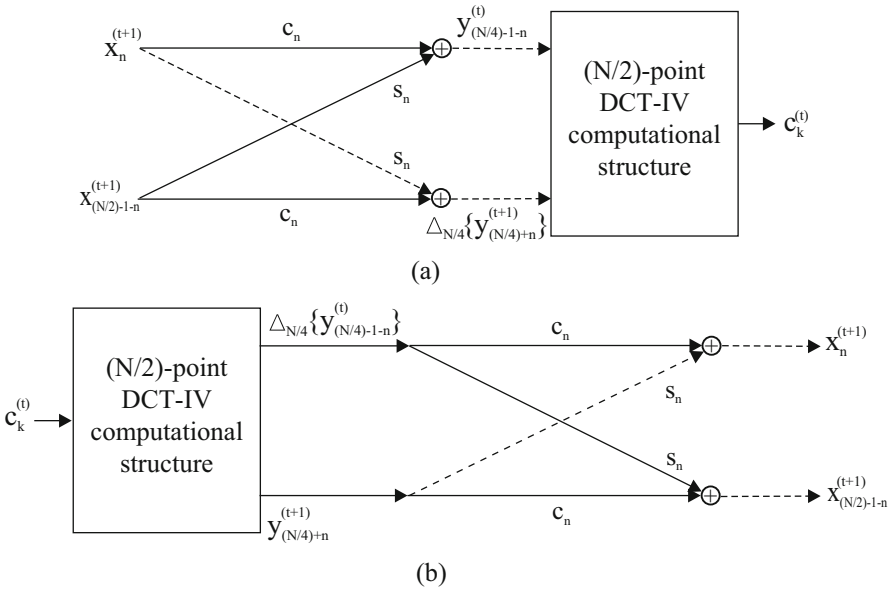

Fig. 9.11 Compact computational structures for the efficient implementation of: (**a**) analysis MLT filter bank with incorporated windowing and overlap procedure, (**b**) synthesis MLT filter bank with incorporated windowing and overlap and add procedure

consist of the regular cascade of $\frac{N}{4}$ Givens–Jacobi rotations derived in matrix-vector forms, respectively, as

$$\begin{pmatrix} y^{(t)}_{\frac{N}{4}-1-n} \\ \Delta_{\frac{N}{4}}\left\{y^{(t+1)}_{\frac{N}{4}+n}\right\} \end{pmatrix} = -\,\boldsymbol{G}_2^{\left(-\frac{\pi(2n+1)}{2N}\right)} \begin{pmatrix} x^{(t+1)}_{n} \\ x^{(t+1)}_{\frac{N}{2}-1-n} \end{pmatrix}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{9.95}$$

and

$$\begin{pmatrix} x^{(t+1)}_{n} \\ x^{(t+1)}_{\frac{N}{2}-1-n} \end{pmatrix} = -\,\boldsymbol{G}_2^{\left(\frac{\pi(2n+1)}{2N}\right)} \begin{pmatrix} \Delta_{\frac{N}{4}}\left\{y^{(t)}_{\frac{N}{4}-1-n}\right\} \\ y^{(t+1)}_{\frac{N}{4}+n} \end{pmatrix}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{9.96}$$

and the forward/inverse $\frac{N}{2}$-point DCT-IV computational structure. Thus, for the complete integer approximation of both analysis and synthesis MLT filter banks for MP3 we still need a suitable regular rotation-based fast DCT-IV algorithm/computational structure.

The fast even-length DCT-IV algorithm presented in Appendix C.2.2 has been decomposed exclusively into sums of the Givens–Jacobi rotations [75]. Actually, it has been derived from the fast analysis MLT filter bank with incorporated windowing and overlap procedure given by (9.30) [76]. For $N' = \frac{N}{4}$ being an odd integer (when $N = 12$, $N' = 3$, and when $N = 36$, $N' = 9$), the Givens–Jacobi rotation-based DCT-IV computational structure in the matrix-vector form is defined as [75]

$$\begin{pmatrix} c^{IV}_{2k} \\ c^{IV}_{\frac{N}{2}-1-2k} \end{pmatrix} = \sum_{n=0}^{\left\lfloor \frac{N'}{2} \right\rfloor - 1} [1,\ (-1)^{n+1}] \left\{ \boldsymbol{G}_2^{(-\varphi_{k,n})} \begin{pmatrix} a_n \\ b_n \end{pmatrix} + (-1)^k\, \boldsymbol{G}_2^{(\varphi_{k,n})} \begin{pmatrix} a_{N'-1-n} \\ b_{N'-1-n} \end{pmatrix} \right\},$$

$$\begin{pmatrix} c^{IV}_{\frac{N}{2}-2k} \\ c^{IV}_{2k-1} \end{pmatrix} = \sum_{n=0}^{\left\lfloor \frac{N'}{2} \right\rfloor - 1} [(-1)^n,\ 1] \left\{ \boldsymbol{G}_2^{(-\varphi_{k,n})} \begin{pmatrix} b_n \\ a_n \end{pmatrix} + (-1)^k\, \boldsymbol{G}_2^{(\varphi_{k,n})} \begin{pmatrix} b_{N'-1-n} \\ a_{N'-1-n} \end{pmatrix} \right\},$$

$$\varphi_{k,n} = \frac{\pi(2n+1)k}{2N'}, \quad k = 1, 2, \ldots, \left\lfloor \frac{N'}{2} \right\rfloor, \quad n = 0, 1, \ldots, \left\lfloor \frac{N'}{2} \right\rfloor - 1, \tag{9.97}$$

where $\lfloor . \rfloor$ denotes the lower integer part of an argument. The data sequences $\{a_n\}$ and $\{b_n\}$ are given by

$$\begin{pmatrix} a_n \\ b_n \end{pmatrix} = \boldsymbol{G}_2^{\left(-\frac{\pi(2n+1)}{2N}\right)} \begin{pmatrix} y^{(t)}_{n} \\ y^{(t)}_{\frac{N}{2}-1-n} \end{pmatrix}, \qquad n = 0, 1, \ldots, N' - 1, \tag{9.98}$$

where $\{y_n^{(t)}\}$ is the time domain aliased data sequence given by (9.31). Finally, for $k = 0$ the coefficients c_0^{IV} and $c_{\frac{N}{2}-1}^{IV}$ are given by

$$c_0^{IV} = a_{\frac{N'-1}{2}} + \sum_{n=0}^{\lfloor \frac{N'}{2} \rfloor - 1} (a_n + a_{N'-1-n}),$$

$$c_{\frac{N}{2}-1}^{IV} = (-1)^{\frac{N'-1}{2}+1} \, b_{\frac{N'-1}{2}} + \sum_{n=0}^{\lfloor \frac{N'}{2} \rfloor - 1} (-1)^{n+1} \, (b_n + b_{N'-1-n}). \tag{9.99}$$

The vectors $[1, \ (-1)^{n+1}]$ and $[(-1)^n, \ 1]$ under sums on the right-hand sides of (9.97) change the signs of components of rotated vectors after summing their corresponding components. Trivial rotations of the vector $[a_{\frac{N'-1}{2}}, b_{\frac{N'-1}{2}}]$ by the angles $-\frac{\pi}{2}k$ are realized by initializing summators for the coefficients c_{2k}^{IV} and $c_{\frac{N}{2}-1-2k}^{IV}$ as follows [75]

$$\begin{pmatrix} c_{2k}^{IV} \\ c_{\frac{N}{2}-1-2k}^{IV} \end{pmatrix} = \boldsymbol{G}_2^{(-\frac{\pi}{2}k)} \begin{pmatrix} a_{\frac{N'-1}{2}} \\ b_{\frac{N'-1}{2}} \end{pmatrix}, \quad k = 1, 2, \ldots, N'-1. \tag{9.100}$$

Now, consider the analysis MLT filter bank with incorporated windowing and overlap procedure for the short block when $N = 12$ (see Fig. 9.11a). For the windowing and overlap procedure given by (9.95) we have

$$\begin{pmatrix} y_2^{(t)} \\ \Delta_3 \{y_3^{(t+1)}\} \end{pmatrix} = -\boldsymbol{G}_2^{(-\frac{\pi}{24})} \begin{pmatrix} x_0^{(t+1)} \\ x_5^{(t+1)} \end{pmatrix}, \quad \begin{pmatrix} y_1^{(t)} \\ \Delta_3 \{y_4^{(t+1)}\} \end{pmatrix} = -\boldsymbol{G}_2^{(-\frac{\pi}{8})} \begin{pmatrix} x_1^{(t+1)} \\ x_4^{(t+1)} \end{pmatrix},$$

$$\begin{pmatrix} y_0^{(t)} \\ \Delta_3 \{y_5^{(t+1)}\} \end{pmatrix} = -\boldsymbol{G}_2^{(-\frac{5\pi}{24})} \begin{pmatrix} x_2^{(t+1)} \\ x_3^{(t+1)} \end{pmatrix}. \tag{9.101}$$

The windowing and overlap procedure given by (9.101) includes three inverse Givens–Jacobi rotations by three rotation angles $-\frac{\pi}{24}$, $-\frac{\pi}{8}$, and $-\frac{5\pi}{24}$. According to (9.98) we have

$$\begin{pmatrix} a_0 \\ b_0 \end{pmatrix} = \boldsymbol{G}_2^{(-\frac{\pi}{24})} \begin{pmatrix} y_0^{(t)} \\ y_5^{(t)} \end{pmatrix}, \quad \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} = \boldsymbol{G}_2^{(-\frac{\pi}{8})} \begin{pmatrix} y_1^{(t)} \\ y_4^{(t)} \end{pmatrix}, \quad \begin{pmatrix} a_2 \\ b_2 \end{pmatrix} = \boldsymbol{G}_2^{(-\frac{5\pi}{24})} \begin{pmatrix} y_2^{(t)} \\ y_3^{(t)} \end{pmatrix}. \tag{9.102}$$

Equation (9.102) includes again three inverse Givens–Jacobi rotations by three rotation angles $-\frac{\pi}{24}$, $-\frac{\pi}{8}$, and $-\frac{5\pi}{24}$. According to (9.100) and (9.97) for $k = 1$ and $n = 0$, the rotation angle $\varphi_{1,0} = \frac{\pi}{6}$, and we have

$$\begin{pmatrix} \overset{IV}{c_2} \\ \\ \overset{IV}{c_3} \end{pmatrix} = \begin{pmatrix} b_1 \\ -a_1 \end{pmatrix} + [1, -1] \left\{ \boldsymbol{G}_2^{\left(-\frac{\pi}{6}\right)} \begin{pmatrix} a_0 \\ b_0 \end{pmatrix} - \boldsymbol{G}_2^{\left(\frac{\pi}{6}\right)} \begin{pmatrix} a_2 \\ b_2 \end{pmatrix} \right\},$$

$$\begin{pmatrix} \overset{IV}{c_4} \\ \\ \overset{IV}{c_1} \end{pmatrix} = \begin{pmatrix} -a_1 \\ -b_1 \end{pmatrix} + \left\{ \boldsymbol{G}_2^{\left(-\frac{\pi}{6}\right)} \begin{pmatrix} b_0 \\ a_0 \end{pmatrix} - \boldsymbol{G}_2^{\left(\frac{\pi}{6}\right)} \begin{pmatrix} b_2 \\ a_2 \end{pmatrix} \right\}. \tag{9.103}$$

Equation (9.103) includes four forward or inverse Givens–Jacobi rotations by one rotation angle $\frac{\pi}{6}$ or $-\frac{\pi}{6}$. Finally, according to (9.99) we have

$$\overset{IV}{c_0} = a_1 + (a_0 + a_2), \qquad \overset{IV}{c_5} = b_1 - (b_0 + b_2). \tag{9.104}$$

Thus, for the short block the fast analysis MLT filter bank represented by the compact computational structure shown in Fig. 9.11a includes totally ten Givens–Jacobi rotations by four rotation angles, $-\frac{\pi}{24}$, $-\frac{\pi}{8}$, $-\frac{5\pi}{24}$, and $\mp\frac{\pi}{6}$.

Formulae for the fast synthesis MLT filter bank represented by the compact computational structure shown in Fig. 9.11b can be easily obtained by following the procedure indicated above.

9.3.5.1 IntMLT Implementation by the LUL or ULU Structures

For integer approximation of the analysis and synthesis MLT filter banks (IntMLT) by the ***LUL*** or ***ULU*** computational structures we need to approximate inverse Givens–Jacobi rotations $\boldsymbol{G}_2^{\left(-\frac{\pi}{24}\right)}$, $\boldsymbol{G}_2^{\left(-\frac{\pi}{8}\right)}$, $\boldsymbol{G}_2^{\left(-\frac{5\pi}{24}\right)}$, and forward/inverse Givens–Jacobi rotations $\boldsymbol{G}_2^{\left(\pm\frac{\pi}{6}\right)}$. We know that their integer approximation can be realized either by introducing the rounding operator in subsequent computational steps, or alternatively, the off-diagonal multipliers can be approximated by dyadic rationals. Using the ***LUL*** or the ***ULU*** computational structure is left on a reader. For the rotation angles $\frac{\pi}{24}$, $\frac{\pi}{8}$, $\frac{5\pi}{24}$, and $\frac{\pi}{6}$ the values of off-diagonal multipliers $\tan\frac{\varphi}{2}$ and $\sin\varphi$ as well as their relatively exact dyadic approximations are summarized in Table 9.2.

Note 10: The first integer approximation of the MLT block transform for MP3 audio coding standard using ***LUL*** structures has been reported in [54, 55]. It is based on the fast MDCT algorithm [77]. The required orthonormal 3-point and 9-point DCTs-II are factored exclusively into Givens–Jacobi rotations. As an example, for the short audio block the orthonormal 3-point DCT-II module is decomposed into two Givens–Jacobi rotations according to (9.94). However, integer approximation of the windowing operation for the complete analysis and synthesis MLT filter banks was not considered.

Table 9.2 Values of off-diagonal multipliers $\tan\frac{\varphi}{2}$ and $\sin\varphi$ and their dyadic approximations for ***LUL*** or ***ULU*** computational structures

Rotation angle	Value	Dyadic approximation	Approximated value
$\tan\frac{\pi}{48}$	0.065 543 463	17/256	0.066 406 250
$\sin\frac{\pi}{24}$	0.130 526 192	33/256	0.128 906 250
$\tan\frac{\pi}{16}$	0.198 912 367	51/256	0.199 218 750
$\sin\frac{\pi}{8}$	0.382 683 432	49/128	0.382 812 500
$\tan\frac{5\pi}{48}$	0.339 454 259	87/256	0.339 843 750
$\sin\frac{5\pi}{24}$	0.608 761 429	39/64	0.609 375 000
$\tan\frac{\pi}{12}$	0.267 949 192	69/256	0.269 531 250
$\sin\frac{\pi}{6}$	0.500 000 000	1/2	0.500 000 000

Table 9.3 Values of rotation angles in radians and their appropriate approximations by Pythagorean triads (c, s, c+1)

Rotation angle	Value in radians	Approximation by (c, s, c+1)
$\frac{\pi}{24}$	0.130 899 694	(15, 112, 113)
$\frac{\pi}{8}$	0.392 699 082	(5, 12, 13)
$\frac{5\pi}{24}$	0.654 498 469	(3, 4, 5)
$\frac{\pi}{6}$	0.523 598 776	(5, 12, 13) + (15, 112, 113)

9.3.5.2 IntMLT Implementation by the Modulo Transform Rotations

For the implementation of analysis and synthesis IntMLT filter banks by the modulo transforms we need to approximate inverse Givens–Jacobi rotations $\boldsymbol{G}_2^{(-\frac{\pi}{24})}$, $\boldsymbol{G}_2^{(-\frac{\pi}{8})}$, $\boldsymbol{G}_2^{(-\frac{5\pi}{24})}$ and forward/inverse Givens–Jacobi rotations $\boldsymbol{G}_2^{(\pm\frac{\pi}{6})}$ by Pythagorean triads $(c, s, c+1)$. With respect to Table 9.1, the values of rotation angles $\frac{\pi}{24}$, $\frac{\pi}{8}$, $\frac{5\pi}{24}$ and $\frac{\pi}{6}$ in radians and their appropriate approximations by Pythagorean triads $(c, s, c+1)$ are shown in Table 9.3.

9.3.5.3 IntMLT Implementation by the Infinity-Norm Transform Rotations

For the implementation of analysis and synthesis IntMLT filter banks by the infinity-norm rotation transforms for each forward/inverse Givens–Jacobi rotation we need to calculate the angle θ_∞ according to (9.88), and then the value of parameter RL according to (9.93). Values of angles θ_∞ are shown in Table 9.4. Integer approximation is realized by Eq. (9.92) using the rounding operator.

Discussed local methods to integer approximation of the MLT block transform (without integer approximation of the windowing and overlap procedure) for MP3 audio coding standard have been evaluated and compared in terms of the

Table 9.4 Values of angles $\theta_\infty = \tan\theta_2$

θ_2	$\theta_\infty = \tan\theta_2$
$\frac{\pi}{24}$	0.131 652 498
$\frac{\pi}{8}$	0.414 213 562
$\frac{5\pi}{24}$	0.767 326 988
$\frac{\pi}{6}$	0.577 350 269

computational speed and accuracy in [57] with the following conclusion: IntMLT approximated by the modulo transforms has the highest computational speed, while IntMLT approximated by the infinity-norm rotation transforms has the highest accuracy.

9.4 Global Methods to Integer Approximation

The local methods to integer approximation of a transform are applied to local parts, usually to the orthogonal/orthonormal transforms of length 2 being the Givens–Jacobi rotations and/or trivial rotations by the angle $\frac{\pi}{4}$ in the corresponding orthogonal (unitary) recursive sparse block matrix factorizations (9.1), (9.13), (9.24), and (9.25). However, for long transform sizes the construction and computational complexity as well as the number of rounding operations and approximation error increase considerably.

In order to improve the integer transforms in terms of improved construction, computational complexity and improved approximation accuracy, the global methods to integer approximation of a transform are used [15, 21, 36, 41–47, 49–53, 56]. They are based on theory, algebra of block matrices, and block matrix decompositions [80, 86] (see Appendix A.7). Essentially, in global approximation methods large parts of transforms are calculated in floating-point arithmetic, only output is component-wisely rounded and added/subtracted. In matrix terminology, instead of multiplying by scalar elements (see ***LUL*** or ***ULU*** structures), block matrix-vector multiplication is applied followed by component-wise rounding operators. Consequently, the construction and computational complexity as well as the number of rounding operations and approximation error are minimized. Thus, global approximation methods are suitable not only for large transform sizes but also for small transform sizes.

9.4.1 Generalized LUL and ULU Block Matrix Factorizations of Windowing & Overlap and Windowing & Overlap & Add Procedures

At first, in the following subsections the generalized ***LUL*** and ***ULU*** block matrix factorizations of the windowing&overlap and windowing&overlap&add procedures both for the fast analysis/synthesis MLT filter banks (see Fig. 9.11a, b), and fast

analysis/synthesis MDCT filter banks with an arbitrary symmetric windowing function given by (9.34) and (9.40) are derived. The generalized ***LUL*** and ***ULU*** block matrix factorizations consisting of products of lower and upper block triangular matrices with unit diagonal blocks are invertible, thus, they enable us to construct the reversible integer transforms in compact block matrix-vector forms.

9.4.1.1 The Analysis and Synthesis MLT Filter Banks

With respect to Fig. 9.11a the windowing and overlap procedure by the sine windowing function in the fast analysis MLT filter bank is represented in the matrix-vector form as

$$\begin{pmatrix} y^{(t)}_{\frac{N}{4}-1-n} \\ \Delta_{\frac{N}{4}} \left\{ y^{(t+1)}_{\frac{N}{4}+n} \right\} \end{pmatrix} = - \begin{pmatrix} c_n & s_n \\ -s_n & c_n \end{pmatrix} \begin{pmatrix} x^{(t+1)}_{n} \\ x^{(t+1)}_{\frac{N}{2}-1-n} \end{pmatrix}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{9.105}$$

where $c_n = \cos \frac{\pi(2n+1)}{2N}$ and $s_n = \sin \frac{\pi(2n+1)}{2N}$. We begin with a simple example. Consider (9.105) for $N = 16$. Then, for $n = 0, 1, 2, 3$ we have

$$\begin{pmatrix} y^{(t)}_{3} \\ \Delta_4 \left\{ y^{(t+1)}_{4} \right\} \end{pmatrix} = - \begin{pmatrix} c_0 & s_0 \\ -s_0 & c_0 \end{pmatrix} \begin{pmatrix} x^{(t+1)}_{0} \\ x^{(t+1)}_{7} \end{pmatrix}, \quad \begin{pmatrix} y^{(t)}_{2} \\ \Delta_4 \left\{ y^{(t+1)}_{5} \right\} \end{pmatrix} = - \begin{pmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{pmatrix} \begin{pmatrix} x^{(t+1)}_{1} \\ x^{(t+1)}_{6} \end{pmatrix},$$

$$\begin{pmatrix} y^{(t)}_{1} \\ \Delta_4 \left\{ y^{(t+1)}_{6} \right\} \end{pmatrix} = - \begin{pmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{pmatrix} \begin{pmatrix} x^{(t+1)}_{2} \\ x^{(t+1)}_{5} \end{pmatrix}, \quad \begin{pmatrix} y^{(t)}_{0} \\ \Delta_4 \left\{ y^{(t+1)}_{7} \right\} \end{pmatrix} = - \begin{pmatrix} c_3 & s_3 \\ -s_3 & c_3 \end{pmatrix} \begin{pmatrix} x^{(t+1)}_{3} \\ x^{(t+1)}_{4} \end{pmatrix}.$$

The set of above equations for $N = 16$ may be represented in a compact matrix-vector form as

$$\begin{pmatrix} y^{(t)}_{3} \\ y^{(t)}_{2} \\ y^{(t)}_{1} \\ y^{(t)}_{0} \\ - - - - - \\ \Delta_4\{y^{(t+1)}_{4}\} \\ \Delta_4\{y^{(t+1)}_{5}\} \\ \Delta_4\{y^{(t+1)}_{6}\} \\ \Delta_4\{y^{(t+1)}_{7}\} \end{pmatrix} = - \left(\begin{array}{cccc|cccc} c_0 & 0 & 0 & 0 & s_0 & 0 & 0 & 0 \\ 0 & c_1 & 0 & 0 & 0 & s_1 & 0 & 0 \\ 0 & 0 & c_2 & 0 & 0 & 0 & s_2 & 0 \\ 0 & 0 & 0 & c_3 & 0 & 0 & 0 & s_3 \\ \hline -s_0 & 0 & 0 & 0 & c_0 & 0 & 0 & 0 \\ 0 & -s_1 & 0 & 0 & 0 & c_1 & 0 & 0 \\ 0 & 0 & -s_2 & 0 & 0 & 0 & c_2 & 0 \\ 0 & 0 & 0 & -s_3 & 0 & 0 & 0 & c_3 \end{array} \right) \begin{pmatrix} x^{(t+1)}_{0} \\ x^{(t+1)}_{1} \\ x^{(t+1)}_{2} \\ x^{(t+1)}_{3} \\ - - - - \\ x^{(t+1)}_{7} \\ x^{(t+1)}_{6} \\ x^{(t+1)}_{5} \\ x^{(t+1)}_{4} \end{pmatrix},$$

or in the equivalent block matrix-vector form as

$$\begin{pmatrix} y_3^{(t)} \\ y_2^{(t)} \\ y_1^{(t)} \\ y_0^{(t)} \\ ----- \\ \Delta_4\{y_4^{(t+1)}\} \\ \Delta_4\{y_5^{(t+1)}\} \\ \Delta_4\{y_6^{(t+1)}\} \\ \Delta_4\{y_7^{(t+1)}\} \end{pmatrix} = - \begin{pmatrix} \boldsymbol{C}_4 & \boldsymbol{S}_4 \\ & \\ -\boldsymbol{S}_4 & \boldsymbol{C}_4 \end{pmatrix} \begin{pmatrix} x_0^{(t+1)} \\ x_1^{(t+1)} \\ x_2^{(t+1)} \\ x_3^{(t+1)} \\ ---- \\ x_7^{(t+1)} \\ x_6^{(t+1)} \\ x_5^{(t+1)} \\ x_4^{(t+1)} \end{pmatrix},$$

where $\boldsymbol{C}_4$ and $\boldsymbol{S}_4$ are diagonal matrices of order 4. The last equation indicates that the windowing and overlap procedure in the fast analysis MLT filter bank may be represented in a generalized block matrix-vector form as

$$\begin{pmatrix} y_{\frac{N}{4}-1-n}^{(t)} \\ ------ \\ \Delta_{\frac{N}{4}}\left\{y_{\frac{N}{4}+n}^{(t+1)}\right\} \end{pmatrix} = - \begin{pmatrix} \boldsymbol{C}_{\frac{N}{4}} & \boldsymbol{S}_{\frac{N}{4}} \\ & \\ -\boldsymbol{S}_{\frac{N}{4}} & \boldsymbol{C}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} x_n^{(t+1)} \\ ---- \\ x_{\frac{N}{2}-1-n}^{(t+1)} \end{pmatrix}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{9.106}$$

where $\boldsymbol{C}_{\frac{N}{4}} = \text{diag}\{\cos\frac{\pi(2n+1)}{2N}\}$ and $\boldsymbol{S}_{\frac{N}{4}} = \text{diag}\{\sin\frac{\pi(2n+1)}{2N}\}$, are diagonal matrices of order $\frac{N}{4}$. The first half of samples $\{x_n^{(t+1)}\}$ on the right-hand side of (9.106) is in natural order while the second half of samples $\{x_{\frac{N}{2}-1-n}^{(t+1)}\}$ is in reverse order.

In order to obtain a reversible integer approximation of (9.106), the nonsingular block matrix on the right-hand side of (9.106) is decomposed into the following product of lower and upper block triangular matrices with unit diagonal blocks as [51]

$$\begin{pmatrix} \boldsymbol{C}_{\frac{N}{4}} & \boldsymbol{S}_{\frac{N}{4}} \\ & \\ -\boldsymbol{S}_{\frac{N}{4}} & \boldsymbol{C}_{\frac{N}{4}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{T}_{\frac{N}{4}} \\ & \\ \boldsymbol{0}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{0}_{\frac{N}{4}} \\ & \\ -\boldsymbol{S}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{T}_{\frac{N}{4}} \\ & \\ \boldsymbol{0}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix}, \tag{9.107}$$

where $\boldsymbol{I}_{\frac{N}{4}}$ is the identity matrix, $\boldsymbol{0}_{\frac{N}{4}}$ is the null matrix, both of order $\frac{N}{4}$. $\boldsymbol{T}_{\frac{N}{4}} = \text{diag}\{\tan\frac{\pi(2n+1)}{4N}\}$, $n = 0, 1, \ldots, \frac{N}{4} - 1$, is the diagonal matrix of order $\frac{N}{4}$. Rounding operators are introduced after $\boldsymbol{T}_{\frac{N}{4}}$ and $\boldsymbol{S}_{\frac{N}{4}}$ matrix multiplications with a vector. Equation (9.107) defines the generalized ***ULU*** block matrix factorization. The corresponding generalized ***ULU*** computational structure is shown in Fig. 9.12, where $\{x_n\}$, $\{x_{\frac{N}{2}-1-n}\}$ are input data sub-sequences and $\{y_{\frac{N}{4}-1-n}\}$, $\{y_{\frac{N}{4}+n}\}$ are output data sub-sequences for $n = 0, 1, \ldots, \frac{N}{4} - 1$. It requires $\frac{3N}{4}$ rounding operators. Explicit estimates of component-wise accumulated approximation errors caused by the rounding operators we can obtain from Theorem 9.1.

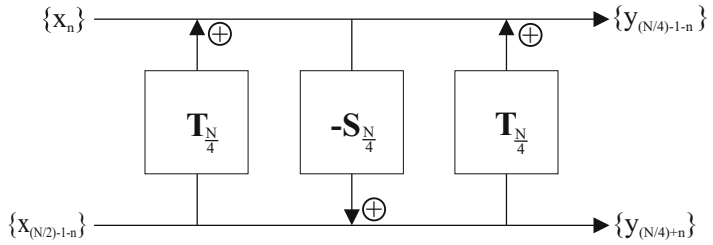

Fig. 9.12 Generalized ***ULU*** computational structure. $\{x_n\}$, $\{x_{\frac{N}{2}-1-n}\}$ are input data subsequences and $\{y_{\frac{N}{4}-1-n}\}$, $\{y_{\frac{N}{4}+n}\}$ are output data subsequences for $n = 0, 1, \ldots, \frac{N}{4} - 1$

On the other hand, with respect to Fig. 9.11b the windowing and overlap and add procedure in the fast synthesis MLT filter bank using the similar approach may be represented in the block matrix-vector form as

$$\begin{pmatrix} x_n^{(t+1)} \\ ---- \\ x_{\frac{N}{2}-1-n}^{(t+1)} \end{pmatrix} = -\begin{pmatrix} \boldsymbol{C}_{\frac{N}{4}} & -\boldsymbol{S}_{\frac{N}{4}} \\ & \\ \boldsymbol{S}_{\frac{N}{4}} & \boldsymbol{C}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} \Delta_{\frac{N}{4}} \left\{ y_{\frac{N}{4}-1-n}^{(t)} \right\} \\ ------ \\ y_{\frac{N}{4}+n}^{(t+1)} \end{pmatrix}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{9.108}$$

In order to obtain a reversible integer approximation of (9.108), transposing (9.107) and using the algebra of block triangular matrices we get the generalized ***LUL*** block matrix factorization of nonsingular block matrix on the right-hand side of (9.108) as

$$\begin{pmatrix} \boldsymbol{C}_{\frac{N}{4}} & -\boldsymbol{S}_{\frac{N}{4}} \\ & \\ \boldsymbol{S}_{\frac{N}{4}} & \boldsymbol{C}_{\frac{N}{4}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{0}_{\frac{N}{4}} \\ & \\ \boldsymbol{T}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & -\boldsymbol{S}_{\frac{N}{4}} \\ & \\ \boldsymbol{0}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{0}_{\frac{N}{4}} \\ & \\ \boldsymbol{T}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix}. \tag{9.109}$$

The corresponding generalized ***LUL*** computational structure can be obtained from the generalized ***ULU*** computational structure shown in Fig. 9.12 by reversing the arrows.

Note 11: Inverting (9.107) and using the algebra of block triangular matrices we get the alternative generalized ***ULU*** block matrix factorization of (9.109) defined as

$$\begin{pmatrix} \boldsymbol{C}_{\frac{N}{4}} & -\boldsymbol{S}_{\frac{N}{4}} \\ & \\ \boldsymbol{S}_{\frac{N}{4}} & \boldsymbol{C}_{\frac{N}{4}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & -\boldsymbol{T}_{\frac{N}{4}} \\ & \\ \boldsymbol{0}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{0}_{\frac{N}{4}} \\ & \\ \boldsymbol{S}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{0}_{\frac{N}{4}} \\ & \\ -\boldsymbol{T}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix}. \tag{9.110}$$

Similarly, inverting (9.109) and using the algebra of block triangular matrices we get the alternative generalized ***LUL*** block matrix factorization of (9.107) defined as

$$\begin{pmatrix} \boldsymbol{C}_{\frac{N}{4}} & \boldsymbol{S}_{\frac{N}{4}} \\ -\boldsymbol{S}_{\frac{N}{4}} & \boldsymbol{C}_{\frac{N}{4}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{0}_{\frac{N}{4}} \\ -\boldsymbol{T}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{S}_{\frac{N}{4}} \\ \boldsymbol{0}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{0}_{\frac{N}{4}} \\ -\boldsymbol{T}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix}. \tag{9.111}$$

Note 12: Consider the rotation matrix $\boldsymbol{G}_N$ of order N given by (9.12) in the orthogonal recursive sparse block matrix factorization (9.13) of the DCT-IV matrix $\boldsymbol{C}_N^{IV}$. Let us reorder the input data sequence $\{x_n\}$ such that samples in its first half are in natural order while samples in its second half are in reverse order. Then the reordered rotation matrix $\hat{\boldsymbol{G}}_N$ will have the following block matrix form

$$\hat{\boldsymbol{G}}_N = \begin{pmatrix} \boldsymbol{C}_{\frac{N}{2}} & -\boldsymbol{S}_{\frac{N}{2}} \\ \boldsymbol{S}_{\frac{N}{2}} & \boldsymbol{C}_{\frac{N}{2}} \end{pmatrix}, \tag{9.112}$$

where $\boldsymbol{C}_{\frac{N}{2}} = \mathrm{diag}\{\cos\frac{\pi(2n+1)}{4N}\}$ and $\boldsymbol{S}_{\frac{N}{2}} = \mathrm{diag}\{\sin\frac{\pi(2n+1)}{4N}\}$, $n = 0, 1, \ldots, \frac{N}{2} - 1$, are diagonal matrices of order $\frac{N}{2}$. Consequently, it can be decomposed into the generalized ***LUL*** block matrix form given by (9.109) or (9.110). Note that the diagonal matrix $\boldsymbol{T}_{\frac{N}{2}} = \mathrm{diag}\{\tan\frac{\pi(2n+1)}{8N}\}$ for $n = 0, 1, \ldots, \frac{N}{2} - 1$. Similarly, the reordered rotation matrix $\hat{\boldsymbol{G}}_N^T$ in the orthogonal recursive sparse block matrix factorization (9.24) of the DCT-IV matrix $\boldsymbol{C}_N^{IV}$ will have the following block matrix form

$$\hat{\boldsymbol{G}}_N^T = \begin{pmatrix} \boldsymbol{C}_{\frac{N}{2}} & \boldsymbol{S}_{\frac{N}{2}} \\ -\boldsymbol{S}_{\frac{N}{2}} & \boldsymbol{C}_{\frac{N}{2}} \end{pmatrix}, \tag{9.113}$$

and it can be decomposed into the generalized ***ULU*** block matrix form given by (9.107) or (9.111).

9.4.1.2 The Analysis and Synthesis MDCT Filter Banks

Just described approach for the analysis/synthesis MLT filter banks can be generalized for the fast analysis/synthesis MDCT filter banks with an arbitrary symmetric windowing function $\{w_n\}$. Consider the fast analysis MDCT filter bank with incorporated windowing and overlap procedure given by (9.34) and fast synthesis MDCT filter bank with incorporated windowing and overlap and add procedure given by (9.40). The matrices on the right-hand sides of (9.34) and (9.40) have unit determinant, and substituting $a = d = w_{\frac{N}{2}-1-n}$ and $b = w_n$ into (9.42) and (9.43)

we easily obtain their ***LUL*** matrix factorizations defined, respectively, as

$$\begin{pmatrix} w_{\frac{N}{2}-1-n} & w_n \\ -w_n & w_{\frac{N}{2}-1-n} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{(w_{\frac{N}{2}-1-n}-1)}{w_n} & 1 \end{pmatrix} \begin{pmatrix} 1 & w_n \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{(w_{\frac{N}{2}-1-n}-1)}{w_n} & 1 \end{pmatrix}, \quad w_n \neq 0, \tag{9.114}$$

and

$$\begin{pmatrix} w_{\frac{N}{2}-1-n} & -w_n \\ w_n & w_{\frac{N}{2}-1-n} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -\frac{(w_{\frac{N}{2}-1-n}-1)}{w_n} & 1 \end{pmatrix} \begin{pmatrix} 1 & -w_n \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{(w_{\frac{N}{2}-1-n}-1)}{w_n} & 1 \end{pmatrix}. \tag{9.115}$$

Note that alternative ***ULU*** and ***LUL*** matrix factorizations can be obtained by transposing or by inverting (9.114) and (9.115).

Following the simple example from previous subsection consider Eq. (9.34) for $N = 16$. Then, for $n = 0, 1, 2, 3$ the corresponding set of equations for $N = 16$ may be represented in a compact matrix-vector form as

$$\begin{pmatrix} y_3^{(t)} \\ y_2^{(t)} \\ y_1^{(t)} \\ y_0^{(t)} \\ \hline \Delta_4\{y_4^{(t+1)}\} \\ \Delta_4\{y_5^{(t+1)}\} \\ \Delta_4\{y_6^{(t+1)}\} \\ \Delta_4\{y_7^{(t+1)}\} \end{pmatrix} = - \left(\begin{array}{cccc|cccc} w_7 & 0 & 0 & 0 & w_0 & 0 & 0 & 0 \\ 0 & w_6 & 0 & 0 & 0 & w_1 & 0 & 0 \\ 0 & 0 & w_5 & 0 & 0 & 0 & w_2 & 0 \\ 0 & 0 & 0 & w_4 & 0 & 0 & 0 & w_3 \\ \hline -w_0 & 0 & 0 & 0 & w_7 & 0 & 0 & 0 \\ 0 & -w_1 & 0 & 0 & 0 & w_6 & 0 & 0 \\ 0 & 0 & -w_2 & 0 & 0 & 0 & w_5 & 0 \\ 0 & 0 & 0 & -w_3 & 0 & 0 & 0 & w_4 \end{array}\right) \begin{pmatrix} x_0^{(t+1)} \\ x_1^{(t+1)} \\ x_2^{(t+1)} \\ x_3^{(t+1)} \\ \hline x_7^{(t+1)} \\ x_6^{(t+1)} \\ x_5^{(t+1)} \\ x_4^{(t+1)} \end{pmatrix},$$

or in the equivalent block matrix-vector form as

$$\begin{pmatrix} y_3^{(t)} \\ y_2^{(t)} \\ y_1^{(t)} \\ y_0^{(t)} \\ \hline \Delta_4\{y_4^{(t+1)}\} \\ \Delta_4\{y_5^{(t+1)}\} \\ \Delta_4\{y_6^{(t+1)}\} \\ \Delta_4\{y_7^{(t+1)}\} \end{pmatrix} = - \begin{pmatrix} \hat{\boldsymbol{W}}_4 & \boldsymbol{W}_4 \\ -\boldsymbol{W}_4 & \hat{\boldsymbol{W}}_4 \end{pmatrix} \begin{pmatrix} x_0^{(t+1)} \\ x_1^{(t+1)} \\ x_2^{(t+1)} \\ x_3^{(t+1)} \\ \hline x_7^{(t+1)} \\ x_6^{(t+1)} \\ x_5^{(t+1)} \\ x_4^{(t+1)} \end{pmatrix},$$

where $\boldsymbol{W}_4$ and $\hat{\boldsymbol{W}}_4$ are diagonal matrices of order 4. Similarly, the last equation indicates that the windowing and overlap procedure in the fast analysis MDCT filter bank may be represented in a generalized block matrix-vector form as

$$\begin{pmatrix} y^{(t)}_{\frac{N}{4}-1-n} \\ ----- \\ \Delta_{\frac{N}{4}}\{y^{(t+1)}_{\frac{N}{4}+n}\} \end{pmatrix} = -\begin{pmatrix} \hat{\boldsymbol{W}}_{\frac{N}{4}} & \boldsymbol{W}_{\frac{N}{4}} \\ & \\ -\boldsymbol{W}_{\frac{N}{4}} & \hat{\boldsymbol{W}}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} x^{(t+1)}_{n} \\ --- \\ x^{(t+1)}_{\frac{N}{2}-1-n} \end{pmatrix}, \; n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{9.116}$$

where $\hat{\boldsymbol{W}}_{\frac{N}{4}} = \text{diag}\left\{\frac{(w_{\frac{N}{2}-1-n}-1)}{w_n}\right\}$ and $\boldsymbol{W}_{\frac{N}{4}} = \text{diag}\{w_n\}$ are diagonal matrices of order $\frac{N}{4}$.

In order to obtain a reversible integer approximation of (9.116), we need a block matrix factorization of nonsingular block matrix on the right-hand side of (9.116). In fact, the block matrix on the right-hand side of (9.116) can be decomposed into the following product of lower and upper block triangular matrices with unit diagonal blocks as

$$\begin{pmatrix} \hat{\boldsymbol{W}}_{\frac{N}{4}} & \boldsymbol{W}_{\frac{N}{4}} \\ & \\ -\boldsymbol{W}_{\frac{N}{4}} & \hat{\boldsymbol{W}}_{\frac{N}{4}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{0}_{\frac{N}{4}} \\ & \\ \hat{\boldsymbol{W}}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{W}_{\frac{N}{4}} \\ & \\ \boldsymbol{0}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{0}_{\frac{N}{4}} \\ & \\ \hat{\boldsymbol{W}}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix}, \tag{9.117}$$

where $\boldsymbol{I}_{\frac{N}{4}}$ is the identity matrix, $\boldsymbol{0}_{\frac{N}{4}}$ is the null matrix, both of order $\frac{N}{4}$. $\hat{\boldsymbol{W}}_{\frac{N}{4}} = \text{diag}\left\{\frac{(w_{\frac{N}{2}-1-n}-1)}{w_n}\right\}$ and $\hat{\boldsymbol{W}}_{\frac{N}{4}} = \text{diag}\{w_n\}$, $n = 0, 1, \ldots, \frac{N}{4} - 1$, are the diagonal matrices of order $\frac{N}{4}$. Equation (9.117) is considered as the generalized ***LUL*** block matrix factorization of (9.114).

Finally, the windowing and overlap and add procedure in the fast synthesis MDCT filter bank given by (9.40) using the similar approach may be represented in the block matrix-vector form as

$$\begin{pmatrix} x^{(t+1)}_{n} \\ --- \\ x^{(t+1)}_{\frac{N}{2}-1-n} \end{pmatrix} = -\begin{pmatrix} \hat{\boldsymbol{W}}_{\frac{N}{4}} & -\boldsymbol{W}_{\frac{N}{4}} \\ & \\ \boldsymbol{W}_{\frac{N}{4}} & \hat{\boldsymbol{W}}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} \Delta_{\frac{N}{4}}\left\{y^{(t)}_{\frac{N}{4}-1-n}\right\} \\ ------ \\ y^{(t+1)}_{\frac{N}{4}+n} \end{pmatrix}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{9.118}$$

In order to obtain a reversible integer approximation of (9.118), transposing (9.117) and using the algebra of block triangular matrices we get the generalized ***ULU*** block matrix factorization of nonsingular block matrix on the right-hand side of (9.118) as

$$\begin{pmatrix} \hat{\boldsymbol{W}}_{\frac{N}{4}} & -\boldsymbol{W}_{\frac{N}{4}} \\ & \\ \boldsymbol{W}_{\frac{N}{4}} & \hat{\boldsymbol{W}}_{\frac{N}{4}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \hat{\boldsymbol{W}}_{\frac{N}{4}} \\ & \\ \boldsymbol{0}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \boldsymbol{0}_{\frac{N}{4}} \\ & \\ \boldsymbol{W}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{4}} & \hat{\boldsymbol{W}}_{\frac{N}{4}} \\ & \\ \boldsymbol{0}_{\frac{N}{4}} & \boldsymbol{I}_{\frac{N}{4}} \end{pmatrix}. \tag{9.119}$$

Now, it remains for us only to construct reversible integer approximations of the DCT-II block transform $\boldsymbol{C}_N^{II}$ given by orthogonal sparse block matrix factorization (9.1), the DCT-IV block transform $\boldsymbol{C}_N^{IV}$ given by the orthogonal recursive sparse block matrix factorizations (9.13) and (9.24), and the DFT block transform $\boldsymbol{F}_N$ given by the unitary sparse block matrix factorization (9.25).

9.4.2 *Integer Transforms with an Expansion Factor*

A global method to derive integer transform from a given arbitrary invertible linear transform has been proposed in [15]. Principally, the method can be seen as an extension of integer transform with expansion factors [10]. The construction of invertible integer DCT is illustrated for the 8-point DCT-II.

For a general linear mapping $\hat{F} : R^N \rightarrow R^N$ given by $\hat{F}\ (\mathbf{x}) = \boldsymbol{H}_N\ \mathbf{x}$, where $\boldsymbol{H}_N$ is an invertible transform matrix, it can be found an invertible integer mapping $F : Z^N \rightarrow Z^N$ approximating $\hat{F}$ by

$$\hat{F}(\mathbf{x}) = \text{round}(\boldsymbol{H}_N\mathbf{x}),$$

if $\boldsymbol{H}_N$ satisfies the expansion condition $\langle -\frac{1}{2}, \frac{1}{2}\rangle^N \subseteq \boldsymbol{H}_N((-\frac{1}{2}, \frac{1}{2}\rangle^N)$, where $\boldsymbol{H}_N((-\frac{1}{2}, \frac{1}{2}\rangle^N) = \{\boldsymbol{H}_N\ \mathbf{r} : \mathbf{r} \in (-\frac{1}{2}, \frac{1}{2}\rangle^N\}$ is the image of unit cube $\langle -\frac{1}{2}, \frac{1}{2}\rangle^N$ under the linear mapping $\hat{F}$ generated by $\boldsymbol{H}_N$. However, the transform matrix $\boldsymbol{H}_N$ frequently does not satisfy this condition. In this case we can apply to $\boldsymbol{H}_N$ a suitable expansion factor $\alpha_N > 1$ such that $\alpha_N\boldsymbol{H}_N$ satisfies the expansion condition, i.e., $\alpha_N\ \boldsymbol{H}_N((-\frac{1}{2}, \frac{1}{2}\rangle^N)$ completely covers the unit cube $\langle -\frac{1}{2}, \frac{1}{2}\rangle^N$. Then, an invertible integer transform is simply given by

$$F(\mathbf{x}) = \text{round}(\alpha_N\ \boldsymbol{H}_N\mathbf{x}).$$

This nonlinear integer transform is very close to the exact (scaled) transform $\alpha_N\ \boldsymbol{H}_N\mathbf{x}$ and the error $\alpha_N\ \boldsymbol{H}_N\mathbf{x} - F(\mathbf{x})$ is at most $\frac{1}{2}$ in each component.

The idea is applied to the DCT-II matrix to derive invertible integer DCT-II. Generally, we just need to apply a fast DCT-II algorithm to the input data vector $\mathbf{x}$ and to compute in floating-point arithmetic $\alpha_N\ \boldsymbol{C}_N^{II}\mathbf{x}$, where α_N is a relatively small constant depending on the value of N. Finally, each DCT-II coefficient is rounded to the nearest integer. The proposed integer transform has two advantages:

- The fast algorithm being already implemented for the DCT-II can be directly applied.
- The difference between integer transform and exact (scaled) linear transform is controlled.

Now consider the discrete trigonometric transforms where DCT matrices are obviously orthogonal. But orthogonal matrices do not satisfy the expansion con-

dition and need to be multiplied with a suitable expansion factor. For a given trigonometric transform matrix $\boldsymbol{H}_N$ generating the linear mapping $\hat{F} : \mathbf{x} \rightarrow \boldsymbol{H}_N\mathbf{x}$ it is necessary to find a minimal expansion factor $\alpha > 0$ such that $\alpha\ \boldsymbol{H}_N$ satisfies the expansion condition. In the following theorem is shown how to determine the minimal expansion factor for trigonometric transform matrix [15].

Theorem 9.4 (The Nonlinear Integer Mapping for Trigonometric Transform Matrix with the Minimal Expansion Factor) *Let $\boldsymbol{H}_N$ be an invertible matrix and $\hat{F} : R^N \rightarrow R^N$ with $\hat{F}(\mathbf{x}) = \boldsymbol{H}_N\mathbf{x}$ be the linear mapping generated by $\boldsymbol{H}_N$. Then the nonlinear mapping $F(\mathbf{x}) : Z^N \rightarrow Z^N$ given by*

$$F(\mathbf{x}) = round(\alpha\ \boldsymbol{H}_N\mathbf{x}). \tag{9.120}$$

with $\alpha \geq \alpha_N = \|\boldsymbol{H}_N^{-1}\|_\infty$ is an invertible mapping, and we have

$$\mathbf{x} = round\left(\frac{1}{\alpha}\boldsymbol{H}_N^{-1}F(\mathbf{x})\right). \tag{9.121}$$

Moreover, it follows that the error is

$$\|\alpha\hat{F}(\mathbf{x}) - F(\mathbf{x})\|_\infty \leq \frac{1}{2}, \tag{9.122}$$

i.e., F is close to the scaled linear mapping $\alpha\ \hat{F}$.

□

The proof of Theorem 9.4 can be found in [15].

Specifically, for integer DCT-II the nonlinear invertible mapping $F(\mathbf{x}) : Z^N \rightarrow Z^N$ approximating the DCT-II is defined by

$$F(\mathbf{x}) = round(\alpha\ \boldsymbol{C}_N^{II}\mathbf{x}) \tag{9.123}$$

with

$$\alpha \geq \alpha_N = \frac{1}{\sqrt{N}} + \frac{1}{\sqrt{2N}}\left(\cot\frac{\pi}{4N} - 1\right). \tag{9.124}$$

The nonlinear mapping F is invertible, and we have

$$\mathbf{x} = \text{round}\left(\frac{1}{\alpha}\,[\boldsymbol{C}_N^{II}]^T F(\mathbf{x})\right) = \text{round}\left(\frac{1}{\alpha}\,\boldsymbol{C}_N^{III}F(\mathbf{x})\right). \tag{9.125}$$

Moreover, comparing $\hat{\mathbf{x}} = \alpha\ \boldsymbol{C}_N^{II}\mathbf{x}$ and $\mathbf{y} = F(\mathbf{x})$ component-wise, we find that

$$|y_j - \hat{x}_j| < \frac{1}{2}, \quad j = 0, 1, \ldots, N-1, \tag{9.126}$$

i.e., in all components, the nonlinear mapping F rounds the exact (scaled) DCT-II coefficients to the next integer. In particular, the constants α_N satisfy $\alpha_N \leq \sqrt{N}$, i.e., we can replace the normalization factor $\sqrt{\frac{2}{N}}$ in the definition of $\boldsymbol{C}_N^{II}$ by $\sqrt{2}$ and apply a fast DCT-II algorithm to this scaled DCT-II.

Based on the above discussion, the algorithm for integer N-point DCT-II computation is very simple. For integer input vector $\mathbf{x} \in Z^N$ the computation of the forward integer N-point DCT-II consists of the following steps:

1. Compute $\hat{\mathbf{x}} = \alpha\ \boldsymbol{C}_N^{II}\mathbf{x}$ by a fast DCT-II algorithm, where $\alpha \geq \alpha_N$ is chosen suitably, for example, take $\alpha_N = \sqrt{N}$.
2. Compute $\mathbf{y} = \text{round}(\hat{\mathbf{x}})$, where $\mathbf{y} \in Z^N$ approximates $\alpha\ \boldsymbol{C}_N^{II}\mathbf{x}$.

For example, when $N = 8$ we can just take the factor $\alpha = 2\sqrt{2} > \alpha_N$ and use a fast 8-point DCT-II algorithm for $2\sqrt{2}\ \boldsymbol{C}_N^{II}\mathbf{x}$.

The inverse integer N-point DCT-II algorithm is equivalently simple and consists of the following steps:

1. Compute $\hat{\mathbf{y}} = \frac{1}{\alpha}[\boldsymbol{C}_N^{II}]^T\mathbf{y} = \frac{1}{\alpha}\ \boldsymbol{C}_N^{III}\mathbf{y}$ by a fast inverse DCT-II algorithm, where α is chosen as in the forward integer DCT-II algorithm.
2. Compute $\mathbf{x} = \text{round}(\hat{\mathbf{y}})$, where $\mathbf{x} \in Z^N$ is the original input vector.

Using the Theorem 9.4 the integer transforms can be derived for other discrete trigonometric transforms analogously, i.e., also for the $\boldsymbol{C}_N^{IV}$.

9.4.3 *Multidimensional Computational Structure*

The multidimensional computational structure (MDCS) is based on the factorization of 2×2 diagonal scaling matrix with unit determinant into the product of three unit lower and unit upper triangular matrices as follows [35]

$$\begin{aligned}\begin{pmatrix} d & 0 \\ 0 & d^{-1} \end{pmatrix} &= \begin{pmatrix} -1 & 0 \\ d^{-1} & 1 \end{pmatrix}\begin{pmatrix} 1 & -d \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & d^{-1} \end{pmatrix} \\ &= \begin{pmatrix} -1 & 0 \\ d^{-1} & 1 \end{pmatrix}\begin{pmatrix} 1 & -d \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ d^{-1} & 1 \end{pmatrix} J_2, \quad d \in R,\ \ d \neq 0. \end{aligned} \tag{9.127}$$

The factorization (9.127) provides the basic idea to derive the MDCS computational structure. It can be extended to cases where each scalar element in the factored matrices is a square nonsingular matrix. In fact, for an arbitrary invertible matrix $\boldsymbol{T}_M$ of order M the following block matrix factorization is possible [36, 41, 51]

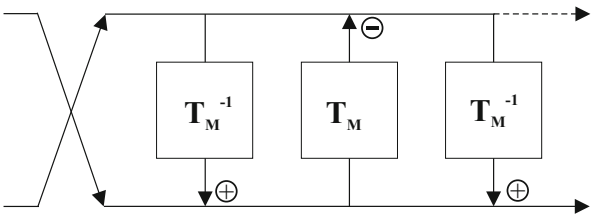

Fig. 9.13 MDCS computational structure

$$\begin{pmatrix} \boldsymbol{T}_M & \boldsymbol{0}_M \\ \boldsymbol{0}_M & \boldsymbol{T}_M^{-1} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{I}_M & \boldsymbol{0}_M \\ \boldsymbol{T}_M^{-1} & \boldsymbol{I}_M \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_M & -\boldsymbol{T}_M \\ \boldsymbol{0}_M & \boldsymbol{I}_M \end{pmatrix} \begin{pmatrix} \boldsymbol{0}_M & \boldsymbol{I}_M \\ \boldsymbol{I}_M & \boldsymbol{T}_M^{-1} \end{pmatrix}$$

$$= \begin{pmatrix} -\boldsymbol{I}_M & \boldsymbol{0}_M \\ \boldsymbol{T}_M^{-1} & \boldsymbol{I}_M \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_M & -\boldsymbol{T}_M \\ \boldsymbol{0}_M & \boldsymbol{I}_M \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_M & \boldsymbol{0}_M \\ \boldsymbol{T}_M^{-1} & \boldsymbol{I}_M \end{pmatrix} \begin{pmatrix} \boldsymbol{0}_M & \boldsymbol{I}_M \\ \boldsymbol{I}_M & \boldsymbol{0}_M \end{pmatrix}, \quad (9.128)$$

where $\boldsymbol{I}_M$ is the identity matrix, $\boldsymbol{0}_M$ is null matrix, both of order M. The MDCS computational structure is defined exactly by the factorization of quasi-diagonal matrix of order $2M$ into the product of lower and upper block triangular matrices of order with unit diagonal blocks. It is shown in Fig. 9.13.

An alternative block matrix factorization but of a cross quasi-diagonal matrix with opposite main diagonal is defined as [42]

$$\begin{pmatrix} \boldsymbol{0}_M & \boldsymbol{T}_M \\ \boldsymbol{T}_M^{-1} & \boldsymbol{0}_M \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_M & \boldsymbol{0}_M \\ -\boldsymbol{T}_M^{-1} & \boldsymbol{I}_M \end{pmatrix} \begin{pmatrix} -\boldsymbol{I}_M & \boldsymbol{T}_M \\ \boldsymbol{0}_M & \boldsymbol{I}_M \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_M & \boldsymbol{0}_M \\ \boldsymbol{T}_M^{-1} & \boldsymbol{I}_M \end{pmatrix}. \quad (9.129)$$

A reversible integer-to-integer mapping is simply constructed by the implementation of matrix-vector multiplications in floating-point arithmetic with the results rounded to the nearest integer. The MDCS computational structure defined by (9.128) or (9.129) requires $3M$ rounding operations. If the integer input vector is applied to a matrix on the right-hand side of (9.128), the first (second) half of integer values is processed by the matrix $\boldsymbol{T}_M^{-1}$ ($\boldsymbol{T}_M$) and then rounded to integer values before adding to the second (first) half of integer values. Hence, the large parts of the transforms are computed without rounding operations, only the results of matrix-vector multiplications are rounded and added. A detailed rounding error analysis of the MDCS computational structure is presented in [49–51].

The MDCS computational structure defined by (9.128) can be used for invertible integer approximations of a transform $\boldsymbol{T}_M$ or an invertible approximation of certain scaling operations. Moreover, substituting the block factorization (9.128) into a recursive sparse block matrix factorization of the transform matrix [for example, $\boldsymbol{C}_M^{IV}$, see Eqs. (9.13) and (9.24)] results in further reducing the overall computational complexity compared to non-integer implementation of the transform. The MDCS computational structure is especially suitable for large transform sizes because it minimizes the total approximation error and the number of rounding operations.

9.4.3.1 Integer Approximation of the DCT-IV (IntDCT-IV) via the MDCS

If the block matrix factorization (9.128) for $M = \frac{N}{2}$ is applied to the DCT-IV matrix $\boldsymbol{C}_{\frac{N}{2}}^{IV}$ and taking into account that $[\boldsymbol{C}_{\frac{N}{2}}^{IV}]^{-1} = \boldsymbol{C}_{\frac{N}{2}}^{IV}$, it becomes [36, 41]

$$\begin{pmatrix} \boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{0} \\ \\ \boldsymbol{0} & \boldsymbol{C}_{\frac{N}{2}}^{IV} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \\ \boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & -\boldsymbol{C}_{\frac{N}{2}}^{IV} \\ \\ \boldsymbol{0} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \\ \boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{0} & \boldsymbol{I}_{\frac{N}{2}} \\ \\ \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \end{pmatrix}. \tag{9.130}$$

The integer approximation of the DCT-IV (IntDCT-IV) via the MDCS computational structure (9.130) is simply obtained by applying rounding operators after each DCT-IV matrix multiplication. For the DCT-IV matrix $\boldsymbol{C}_N^{IV}$ the block matrix factorization (9.130) is substituted into the corresponding orthogonal recursive sparse block matrix factorization (9.13) and (9.24). In order to construct the IntDCT-IV via the MDCS computational structure, consider the orthogonal recursive sparse block matrix factorization (9.24) of the matrix $\boldsymbol{C}_N^{IV}$ written in more compact form as

$$\boldsymbol{C}_N^{IV} = \boldsymbol{G}_N^T \; \frac{\sqrt{2}}{2} \begin{pmatrix} \boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{0} \\ \\ \boldsymbol{0} & \boldsymbol{C}_{\frac{N}{2}}^{IV} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{I}_{\frac{N}{2}} \\ \\ \boldsymbol{D}_{\frac{N}{2}} & -\boldsymbol{D}_{\frac{N}{2}} \end{pmatrix} \boldsymbol{P}_N. \tag{9.131}$$

Now, for the construction of IntDCT-IV we need to factorize the first three (block) matrices on the right-hand side of (9.131). According to (9.113) the reordered rotation matrix $\hat{\boldsymbol{G}}_N^T$ can be factorized using the generalized $\boldsymbol{ULU}$ block matrix form given by (9.107) or (9.111). The second scaled block quasi-diagonal matrix can be factorized using the MDCS computational structure (9.130) taking into account the scaling factor $\frac{\sqrt{2}}{2}$ as

$$\begin{aligned} \frac{\sqrt{2}}{2} \begin{pmatrix} \boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{0} \\ \\ \boldsymbol{0} & \boldsymbol{C}_{\frac{N}{2}}^{IV} \end{pmatrix} &= \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \\ \boldsymbol{0} & \frac{1}{2}\boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} -\boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \\ \sqrt{2}\boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & -\frac{\sqrt{2}}{2}\boldsymbol{C}_{\frac{N}{2}}^{IV} \\ \\ \boldsymbol{0} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \\ &\quad \times \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \\ \sqrt{2}\boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{0} & \boldsymbol{I}_{\frac{N}{2}} \\ \\ \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \end{pmatrix}. \end{aligned} \tag{9.132}$$

Finally, the third block matrix on the right-hand side of (9.131) can be factorized using the block $\boldsymbol{LU}$ decomposition (see the next subsection) as

$$\begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{I}_{\frac{N}{2}} \\ \\ \boldsymbol{D}_{\frac{N}{2}} & -\boldsymbol{D}_{\frac{N}{2}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \\ \boldsymbol{D}_{\frac{N}{2}} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{I}_{\frac{N}{2}} \\ \\ \boldsymbol{0} & -2\boldsymbol{D}_{\frac{N}{2}} \end{pmatrix}. \tag{9.133}$$

Thus, the complete invertible factorization of the matrix $\boldsymbol{C}_N^{IV}$ requires $3N$ rounding operators. Note that the inversion of upper block triangular matrix on the right-hand side of (9.133) is given by

$$\begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{I}_{\frac{N}{2}} \\ \\ \boldsymbol{0} & -2\boldsymbol{D}_{\frac{N}{2}} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & -\frac{1}{2}\boldsymbol{D}_{\frac{N}{2}} \\ \\ \boldsymbol{0} & -\frac{1}{2}\boldsymbol{D}_{\frac{N}{2}} \end{pmatrix}. \tag{9.134}$$

The most straightforward way of using the MDCS computational structure for the implementation of IntMLT or integer MDCT (IntMDCT) is to apply the $\frac{N}{2}$-point DCT-IV to two blocks of time domain aliased signal simultaneously. These blocks can either be from two succeeding overlapped data blocks or from the left and right channel of a stereo audio signal, the so-called Stereo IntMDCT. The construction and implementation of Stereo IntMDCT is presented in detail in [36, 41, 51]. Stereo IntMDCT consists of the windowing and overlap procedure by the sine windowing function followed by the DCT-IV block transform. The windowing and overlap procedure is approximated by the generalized ***ULU*** block matrix factorization given by (9.107) while the DCT-IV block transform is approximated by the MDCS computational structure given by (9.130). A detailed rounding error analysis of Stereo IntMDCT can be found in [49–51].

9.4.3.2 Integer Approximation of the FFT (IntFFT) via the MDCS

It is well known that the $\frac{N}{2}$-point DCT-IV can be implemented by a complex DFT (FFT) of half size with identical pre- and post-rotation stages [82] (see also Appendix C.2.1). Thus, the N-point MLT or the MDCT, if are converted to the $\frac{N}{2}$-point DCT-IV, then they can be implemented alternatively by the $\frac{N}{4}$-point complex FFT.

Consider the unitary sparse block matrix factorization (9.25) of the DFT matrix $\boldsymbol{F}_N$, and in particular, the first quasi-diagonal matrix on the right-hand side of (9.25). Since the forward DFT may be realized by its inverse by swapping the real and imaginary part as both pre- and post-processing [79], the first quasi-diagonal matrix in (9.25) can be written in the following form [41]

$$\begin{pmatrix} \boldsymbol{F}_{\frac{N}{2}} & \boldsymbol{0} \\ \\ \boldsymbol{0} & \boldsymbol{F}_{\frac{N}{2}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \\ \boldsymbol{0} & i\,\boldsymbol{I}^*_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{F}_{\frac{N}{2}} & \boldsymbol{0} \\ \\ \boldsymbol{0} & \boldsymbol{F}^{-1}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \\ \boldsymbol{0} & i\,\boldsymbol{I}^*_{\frac{N}{2}} \end{pmatrix}, \tag{9.135}$$

where $\boldsymbol{I}^*_{\frac{N}{2}}$ is such a matrix that all elements of the matrix which are multiplied by $\boldsymbol{I}^*_{\frac{N}{2}}$ they become complex conjugate. If we substitute (9.135) into (9.25) we obtain the modified unitary sparse block matrix factorization (9.25) of the DFT

matrix $\boldsymbol{F}_N$. Finally, substituting $\boldsymbol{T}_M = \boldsymbol{F}_M$ for $M = \frac{N}{2}$ into the block matrix factorization (9.128), the corresponding MDCS computational structure of the middle quasi-diagonal matrix on the right-hand side of (9.135) is defined as [41]

$$\begin{pmatrix} \boldsymbol{F}_{\frac{N}{2}} & \boldsymbol{0}_{\frac{N}{2}} \\ \boldsymbol{0}_{\frac{N}{2}} & \boldsymbol{F}_{\frac{N}{2}}^{-1} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0}_{\frac{N}{2}} \\ \boldsymbol{F}_{\frac{N}{2}}^{-1} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & -\boldsymbol{F}_{\frac{N}{2}} \\ \boldsymbol{0}_{\frac{N}{2}} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0}_{\frac{N}{2}} \\ \boldsymbol{F}_{\frac{N}{2}}^{-1} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{0}_{\frac{N}{2}} & \boldsymbol{I}_{\frac{N}{2}} \\ \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0}_{\frac{N}{2}} \end{pmatrix}. \tag{9.136}$$

The integer approximation of the FFT (IntFFT) via the MDCS computational structure (9.136) is obtained by applying rounding operators after each DFT matrix multiplication. For the DFT matrix $\boldsymbol{F}_N$ the block matrix factorizations (9.135) and (9.136) are substituted into the corresponding unitary recursive sparse block matrix factorization (9.25). The block $\boldsymbol{W}_{\frac{N}{2}}$ in the second quasi-diagonal matrix on the right-hand side of (9.25) corresponds to the Givens–Jacobi rotations (see Appendix F.4), and consequently, its integer approximation can be obtained by the local approximation methods (***LUL*** computational structure given by (9.46) or by the ***ULU*** computational structure given by (9.48)). Finally, the scaled block butterfly matrix on the right-hand side of (9.25) can be factorized using the block ***LU*** decomposition (see the next subsection) as

$$\frac{\sqrt{2}}{2} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{I}_{\frac{N}{2}} \\ \boldsymbol{I}_{\frac{N}{2}} & -\boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \frac{\sqrt{2}}{2}\boldsymbol{I}_{\frac{N}{2}} & \frac{\sqrt{2}}{2}\boldsymbol{I}_{\frac{N}{2}} \\ \boldsymbol{0} & -\sqrt{2}\boldsymbol{I}_{\frac{N}{2}} \end{pmatrix}. \tag{9.137}$$

Thus, the complete invertible factorization of the matrix $\boldsymbol{F}$ requires $4N + \frac{N}{2}$ rounding operators.

9.4.4 Block LU, LDU, LUD, and PLUS Matrix Decompositions

Through this chapter so far it has been demonstrated that the scalar and partially block matrix decompositions are powerful mathematical tools to construct the reversible (invertible) integer approximation of transforms. Several block matrix decompositions of the $\boldsymbol{C}_N^{II}$ matrix [21], of the $\boldsymbol{C}_N^{IV}$ matrix [43, 44], and of the $\boldsymbol{F}_N$ matrix [45, 46] have been developed for their reversible integer approximation. Although the block matrix decompositions proposed in [21, 43–46] are correct, they have been presented almost without rigorous mathematical background except for [18]. The theory and algebra of block matrices are discussed in detail in Appendix A.7, while the block matrix decompositions are discussed in detail in Appendix A.8. Block ***LU***, ***LDU***, ***LUD*** and ***PLUS*** matrix decompositions in Appendix A.8 have been derived for a 2×2 block nonsingular matrix and are ready to be used for the reversible (invertible) integer approximation of transforms.

Consider a 2×2 block nonsingular matrix partitioned to four square blocks of the same order as

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix},$$

and let $\det(\boldsymbol{A}) \neq 0$ and $\det(\boldsymbol{B}) \neq 0$, i.e., $\boldsymbol{A}$ and $\boldsymbol{B}$ are nonsingular. In the following block $\boldsymbol{LU}$, $\boldsymbol{LUD}$, $\boldsymbol{LDU}$ and $\boldsymbol{PLUS}$ matrix decompositions of $\boldsymbol{M}$ are presented, where $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{L}$ and $\boldsymbol{S}$ are lower block triangular with unit diagonal blocks, $\boldsymbol{U}$ is an upper block triangular (with unit diagonal blocks), and $\boldsymbol{D}$ is a quasi-diagonal matrix.

According to (A.55) in Appendix A.8, the block $\boldsymbol{LU}$ matrix decomposition of $\boldsymbol{M}$ is defined by

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{U} \end{pmatrix}, \tag{9.138}$$

where $\boldsymbol{U} = \boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}$ is the Schur complement of $\boldsymbol{A}$ in $\boldsymbol{M}$.

According to (A.60) in Appendix A.8, the block $\boldsymbol{LDU}$ matrix decomposition of $\boldsymbol{M}$ is defined by

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{U} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{A}^{-1}\boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix}, \tag{9.139}$$

and according to (A.62) in Appendix A.8, the block $\boldsymbol{LUD}$ matrix decomposition of $\boldsymbol{M}$ is defined by

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{B}\boldsymbol{U}^{-1} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{U} \end{pmatrix}. \tag{9.140}$$

Finally, according to (A.63) in Appendix A.8, the block $\boldsymbol{PLUS}$ matrix decomposition of $\boldsymbol{M}$ is defined by

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{C} - \boldsymbol{D}\boldsymbol{H} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{G} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{H} & \boldsymbol{I} \end{pmatrix}, \tag{9.141}$$

where $\boldsymbol{H} = \boldsymbol{B}^{-1}\boldsymbol{A} - \boldsymbol{B}^{-1}$ and $\boldsymbol{G} = (\boldsymbol{D}\boldsymbol{B}^{-1}\boldsymbol{A} - \boldsymbol{C})\boldsymbol{B}$.

One can easily see that all block matrices on the right-hand sides of (9.138)–(9.141) are invertible. Similarly, a reversible integer-to-integer mapping is simply constructed by the implementation of matrix-vector multiplications in floating-point arithmetic with the results rounded to the nearest integer. In general, to optimize a block matrix factorization of a given block matrix, permutation matrices are considered in the factorization to permute rows and columns of a factored block matrix [18, 21, 45, 46]. However, this approach is suitable rather for low order block matrices.

9.4.4.1 IntDCT-IV via the Block Matrix Decompositions

In order to construct the IntDCT-IV via the block matrix decompositions, consider the orthogonal recursive sparse block matrix factorization of the matrix $\boldsymbol{C}_N^{IV}$ given by (9.24). Multiplying all block matrices on the right-hand side of (9.24) we get

$$\boldsymbol{C}_N^{IV} = \boldsymbol{G}_N^T \begin{pmatrix} \frac{\sqrt{2}}{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV} & \frac{\sqrt{2}}{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV} \\ \\ \frac{\sqrt{2}}{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}} & -\frac{\sqrt{2}}{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}} \end{pmatrix} \boldsymbol{P}_N. \tag{9.142}$$

At first, using (9.113) the reordered rotation matrix $\hat{\boldsymbol{G}}_N^T$ can be factorized using the generalized $\boldsymbol{ULU}$ block matrix form given by (9.107) or (9.111). The block matrix on the right-hand side of (9.142) is partitioned into four blocks $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ as follows:

$$\boldsymbol{A} = \boldsymbol{B} = \frac{\sqrt{2}}{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV}, \qquad \boldsymbol{C} = \frac{\sqrt{2}}{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}}, \qquad \boldsymbol{D} = -\frac{\sqrt{2}}{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}}.$$

According to (A.43) in Appendix A.8, the determinant of this block matrix is equal to $\det(\boldsymbol{AD} - \boldsymbol{ACA}^{-1}\boldsymbol{B}) = \det(-\boldsymbol{D}_{\frac{N}{2}})$, where $\boldsymbol{A}^{-1} = \sqrt{2}\;\boldsymbol{C}_{\frac{N}{2}}^{IV}$. Now, in order to obtain a specific block matrix decomposition of the block matrix on the right-hand side of (9.142), it is sufficient to substitute directly the blocks $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ into the appropriate block matrix decomposition (9.138)–(9.140) or (9.141), and to calculate necessary block elements in factored block matrices. For the block $\boldsymbol{LU}$ matrix decomposition we have

$$\boldsymbol{CA}^{-1} = \boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}}\boldsymbol{C}_{\frac{N}{2}}^{IV}, \qquad \boldsymbol{U} = \boldsymbol{D} - \boldsymbol{CA}^{-1}\boldsymbol{B} = -\sqrt{2}\boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}},$$

and the block $\boldsymbol{LU}$ matrix factorization is given by

$$\frac{\sqrt{2}}{2}\begin{pmatrix} \boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{C}_{\frac{N}{2}}^{IV} \\ \\ \boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}} & -\boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \\ \boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}}\boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \frac{\sqrt{2}}{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV} & \frac{\sqrt{2}}{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV} \\ \\ \boldsymbol{0} & -\sqrt{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}} \end{pmatrix}. \tag{9.143}$$

The block $\boldsymbol{LU}$ matrix factorization given by (9.143) requires $2N$ rounding operators, and the complete block matrix factorization of $\boldsymbol{C}_N^{IV}$ given by (9.142) requires $3N + \frac{N}{2}$ rounding operators. Further, the block $\boldsymbol{LDU}$ matrix factorization is given by

$$\frac{\sqrt{2}}{2}\begin{pmatrix} \boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{C}_{\frac{N}{2}}^{IV} \\ \\ \boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}} & -\boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \\ \boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}}\boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \frac{\sqrt{2}}{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{0} \\ \\ \boldsymbol{0} & -\sqrt{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}} \end{pmatrix}$$

$$\times \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{I}_{\frac{N}{2}} \\ \\ \boldsymbol{0} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix}, \tag{9.144}$$

whereas the block ***LUD*** matrix factorization is given by

$$\frac{\sqrt{2}}{2} \begin{pmatrix} \boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{C}_{\frac{N}{2}}^{IV} \\ \\ \boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}} & -\boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \\ \boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}}\boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & -\frac{1}{2}\boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}}\boldsymbol{C}_{\frac{N}{2}}^{IV} \\ \\ \boldsymbol{0} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix}$$
$$\times \begin{pmatrix} \frac{\sqrt{2}}{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{0} \\ \\ \boldsymbol{0} & -\sqrt{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}} \end{pmatrix}. \tag{9.145}$$

The block ***LDU*** matrix factorization given by (9.144) requires $\frac{3N}{2}$ rounding operators, and the complete block matrix factorization of $\boldsymbol{C}_N^{IV}$ given by (9.142) requires $3N$ rounding operators. On the other hand, The block ***LUD*** matrix factorization given by (9.145) requires $2N$ rounding operators, and the complete block matrix factorization of $\boldsymbol{C}_N^{IV}$ given by (9.142) requires $3N + \frac{N}{2}$ rounding operators. Finally, the block ***PLUS*** matrix factorization is given by

$$\frac{\sqrt{2}}{2} \begin{pmatrix} \boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{C}_{\frac{N}{2}}^{IV} \\ \\ \boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}} & -\boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \\ \boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}}\left(\sqrt{2}\boldsymbol{I}_{\frac{N}{2}} - \boldsymbol{C}_{\frac{N}{2}}^{IV}\right) & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \frac{\sqrt{2}}{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV} \\ \\ \boldsymbol{0} & -\boldsymbol{C}_{\frac{N}{2}}^{IV}\boldsymbol{D}_{\frac{N}{2}}\boldsymbol{C}_{\frac{N}{2}}^{IV} \end{pmatrix}$$
$$\times \begin{pmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{0} \\ \\ \boldsymbol{I}_{\frac{N}{2}} - \sqrt{2}\,\boldsymbol{C}_{\frac{N}{2}}^{IV} & \boldsymbol{I}_{\frac{N}{2}} \end{pmatrix}. \tag{9.146}$$

The block ***PLUS*** matrix factorization given by (9.146) requires $2N$ rounding operators, and the complete block matrix factorization of $\boldsymbol{C}_N^{IV}$ given by (9.142) requires $3N + \frac{N}{2}$ rounding operators. Note that the block ***PLUS*** matrix factorization (9.146) of $\boldsymbol{C}_N^{IV}$ is quite similar to that of [43, 44].

9.4.4.2 IntFFT via the Block Matrix Decompositions

Consider the unitary sparse block matrix factorization (9.25) of the DFT matrix $\boldsymbol{F}_N$. Again, multiplying all block matrices on the right-hand side of (9.24) we get

$$\boldsymbol{F}_N = \boldsymbol{P}_N \begin{pmatrix} \frac{\sqrt{2}}{2}\boldsymbol{F}_{\frac{N}{2}} & \frac{\sqrt{2}}{2}\boldsymbol{F}_{\frac{N}{2}} \\ \frac{\sqrt{2}}{2}\boldsymbol{F}_{\frac{N}{2}}\boldsymbol{W}_{\frac{N}{2}} & -\frac{\sqrt{2}}{2}\boldsymbol{F}_{\frac{N}{2}}\boldsymbol{W}_{\frac{N}{2}} \end{pmatrix}. \tag{9.147}$$

The corresponding block ***LU***, ***LDU***, ***LUD***, and ***PLUS*** matrix factorizations of $\boldsymbol{F}_N$ can be obtained following the same approach as for $\boldsymbol{C}_N^{IV}$ matrix.

9.4.5 Rounding Error Shaping Method

There is a serious problem when a signal is transformed by an integer transform. Every rounding operator introduces the rounding error which is accumulated and spread in the transform domain. Typically, the signal being compressed has energy concentrated near low frequencies and decays at high frequencies. The accumulated error spectrum is interpreted as the approximation error of the integer transform and limits the lossless coding performance. For the high frequency range, where signals usually contain a rather small amount of energy, the approximation error can be larger than the actual signal. The impact is more critical for larger transform sizes. When the spectral energy of signal is concentrated at low frequencies, it is possible to improve the coding efficiency by attenuating error spectrum at high frequencies, where error spectrum is dominant. Specifically, the error spectrum at high frequencies may be shaped towards the low frequencies.

A method for shaping the rounding error in the transform domain (error shaping filter) is presented in [49–51, 59, 68]. This error shaping filter is applied to the rounding operations so that the error spectrum is shaped towards the low frequencies. Consequently, the error spectrum is below the spectral envelope of the signal at high frequencies and the lossless coding performance is improved. Detailed design and analysis of rounding error shaping method can be found in [49–51, 59, 68].

9.5 MPEG-4 HD-AAC/SLS Scalable Lossless Audio Coding Standard

Almost all modern perceptual audio coding schemes developed up to now are based on the transform-based approach. Specifically, they employ sub-band analysis and synthesis filter banks, such as the MDCT or MLT, to obtain a block-wise representation of the audio signal in the frequency domain. They operate in floating-point arithmetic, and therefore are lossy in nature [78]. Due to increasing demand of delivery of high sampling rate and high resolution digital audio at lossless quality for high-quality applications, such as audio archiving systems, the lossy compression

became inappropriate since every bit in the original audio signal has to be preserved. On the other hand, the transition from lossy to lossless coding by a scalable way would facilitate the digital audio services to interchange compressed audio across various application domains using scalable lossy to lossless compressed formats. Thus, a scalable to lossless audio coding technology that will support both lossy and lossless audio compression simultaneously is desirable [58, 66]. Responding to these demands, the MPEG audio standardization group decided to start a new work item to explore a new innovative technology for lossless and near-lossless coding of audio signals by issuing a call for proposals for this relevant technology in 2002 [60–62].

As a response to this call two scalable lossy to lossless perceptual audio coders have been developed and evaluated: Advanced Audio Zip (AAZ), a fine grain scalable to lossless (SLS) audio coder [66, 67, 71, 72], and a scalable lossless enhancement of MPEG-4 AAC audio coder [58]. In particular, the AAZ was adopted as the Reference model for MPEG-4 audio SLS work [67, 71]. The enabling technology is the IntMDCT, an integer approximation of the MDCT. Indeed, based on the perfect reconstruction property and the close approximation of the MDCT, the IntMDCT allows to build a scalable lossless enhancement of the MDCT-based perceptual audio coding schemes. A general structure of the scalable lossless enhancement of MDCT-based perceptual audio codec is the following [58]:

- **Encoder:** The structure of encoder is an extension of a general structure of a core perceptual audio codec. In addition to the usual MDCT spectral coefficients, the IntMDCT spectral coefficients are calculated. For the lossless enhancement, the difference between the IntMDCT spectral coefficients and the inverse quantized MDCT spectral coefficients is calculated. These difference values are entropy coded and transmitted in the lossless enhancement bit stream.
- **Decoder:** To achieve lossless decoding, the transmitted difference values in the lossless enhancement bit-stream are decoded and added to the usual MDCT coefficients. In this way the IntMDCT spectral coefficients are reconstructed exactly, hence, the bit-exact audio signal is recovered.

For the extension to lossless operation of a MDCT-based perceptual audio scheme as the core codec, the lossy MPEG-4 AAC codec has been used [58, 67].

Research and standardization efforts of MPEG audio group led to the specification of SLS (scalable lossless coding) technology in the form of amendment to the MPEG-4 audio standard [63, 68–70]. As the extension of MPEG-AAC perceptual audio codec, the MPEG-4 SLS codec includes a scalable lossless audio coding solution that integrates the functionalities of high-compression, lossless audio coding, perceptual audio coding, and fine granular scalable audio coding into a single coder, while simultaneously provides the backward compatibility to existing MPEG-4 AAC codec at the bit-stream level [68, 70]. The scalability is achieved in the frequency domain using the IntMDCT. In order to achieve backward compatibility to the existing MPEG-4 AAC codec, the MPEG-4 SLS adopts two layer structure to code IntMDCT spectral coefficients: the AAC core layer, and a lossless enhancement layer working on the top of AAC architecture. These two

layers in the encoder generate the core layer bit stream which is MPEG-4 AAC compliant, and the lossless enhancement layer bit stream providing the scalability from lossy to lossless coding. AAC compliant bit stream is embedded in the final bit stream. Bit-exact reconstruction of the input original audio signal in the decoder is independent to the implementation accuracy of the AAC core codec. Since MPEG-4 SLS provides the fine granular bit rate scalability from lossy to lossless coding, it becomes a universal compression system for digital audio applications which up to now required different audio coding technologies. Moreover, the need for any transcoding is completely eliminated [68, 70]. It is important to note that for an efficient implementation of the IntMDCT, SLS technology combines local and global approximation methods discussed in this chapter. Similarly, other AAC coding tools, such as Mid/Side (M/S) stereo coding and Temporal Noise Shaping, are considered and implemented in an invertible integer way on the IntMDCT spectral coefficients. An interested reader can find an overview of the MPEG-4 SLS standard, its application scenarios, structure, and description of coding tools in [68].

The MPEG-4 SLS published in June 2006 as an ISO standard [63] still employed two filter banks, the MDCT filter bank which is inherent to AAC core codec, and the IntMDCT filter bank to enable the SLS technology. However, the presence of two filter banks has increased the implementation complexity of the codec. It is well known that the IntMDCT algorithm introduces rounding errors during the whole coding process. Therefore, it was of interest to study and analyze the effect of rounding errors introduced by IntMDCT algorithm for the operation of MPEG-4 SLS in lossy mode, or other words, to investigate the performance of IntMDCT filter bank under lossy operation [64]. Based on this analysis it was found that the error introduced by rounding operations in IntMDCT algorithm does not affect the perceptual quality of the coded audio signal under any circumstances. This fact suggested that the MDCT and IntMDCT filter banks are interchangeable in SLS codec at lossy bit rate. Consequently, the MPEG-4 SLS can use only IntMDCT filter bank instead of both. On the other hand, the difference between MDCT and IntMDCT in the frequency domain is visible as a small noise floor, however, typically much lower than the error introduced by perceptual coding. Thus, the IntMDCT allows for efficient coding of the quantization error in the frequency domain [64, 65]. It was concluded that the IntMDCT is only filter bank to be used in MPEG-4 SLS for both lossy and lossless operations.

In 2007 the MPEG audio group has successfully concluded the standardization process on enhanced SLS technology for lossless coding of high-definition (HD) audio signals—ISO/IEC MPEG-4 High-Definition Scalable Advanced Audio Coding (MPEG-4 HD-AAC/SLS) [59]. HD-AAC/SLS audio coding technology provides a fine grain scalable lossless extension of the MPEG-4 AAC perceptual audio coder up to fully lossless reconstruction at word lengths and sampling rates typically used for HD audio. Enhanced HD-AAC/SLS technology generates a universal digital audio format for a variety of (HD) applications including digital audio archiving, network audio streaming, portable audio players, digital VCD and DVD media, consumer electronics, and digital broadcasting [59].

9.6 Summary

The local and global methods to integer approximation of perfect reconstruction cosine/sine-modulated filter banks and cosine-modulated QMF banks have been discussed in detail. They are based on computational methods of linear algebra, matrix theory, and matrix computations, and in particular, on the scalar and block matrix decompositions. In fact, the scalar and block matrix decompositions are powerful mathematical tools to construct the reversible (invertible) integer transforms. In general, any integer filter bank can be constructed as long as it has a fast algorithm. IntMDCT or IntMLT enabled to design and implement the innovative coding technology for scalable lossy to lossless audio coding in the latest emerged MPEG standards for lossless audio coding, MPEG-4 SLS, and MPEG-4 HD-AAC/SLS.

Problems and Exercises

1. For the matrix $\boldsymbol{T}_2$ given by (9.41) find its inverse matrix.
2. Verify Eqs. (9.52) and (9.53).
3. Verify Eqs. (9.63)–(9.65).
4. Consider a two-dimensional integer lattice (grid) of size $[-a : a, -b : b]$, $a,\ b \in N$, with regularly spaced array of points. In order to verify one-to-one mapping, for a given rotation angle implement by a computer program the integer rotations (both forward and inverse) via:

   - ***LUL*** (***ULU***) computational structures,
   - Modulo transforms,
   - Infinity-norm rotation transforms,

   and visualize the original integer lattice versus rotated one.
5. With respect to Fig. 9.11, Eqs. (9.95), (9.96), and using the fast even-length DCT-IV algorithm defined by (9.97)–(9.100) implement by a computer program the complete analysis/synthesis IntMLT filter banks in MP3 audio coding standard for the short and long blocks via:

   - ***LUL*** (***ULU***) computational structures,
   - Modulo transforms,
   - Infinity-norm rotation transforms,

   and compare implementations in terms of computational speed and accuracy.
6. Repeat exercise 5 for the complete analysis/synthesis IntMDCT filter banks for an arbitrary symmetric windowing function, and compare implementations in terms of computational speed and accuracy.
7. Chapter 5 discusses the efficient implementations of the MLT in MP3 audio coding standard. Among many DCT-IV-based (DCT-IV/DCT-II-based) algorithms does(do) exist some algorithm(s) which would be suitable for the integer approximation?

8. Prove that the block matrices given by (9.107) and (9.109) are inverses to each other.
9. Implement by a computer program the IntDCT-IV by the global method with an expansion factor described in Sect. 9.4.2.
10. Implement by a computer program the IntDCT-IV by the multidimensional computational structure.
11. Implement by a computer program the IntFFT by the multidimensional computational structure.
12. Implement by a computer program the IntDCT-IV via the ***LU***, ***LUD***, ***LDU*** and ***PLUS*** block matrix decompositions using (9.142), and compare implementations in terms of the number of rounding operators and accuracy.
13. Repeat exercise 12 using the orthogonal recursive sparse block matrix factorization of the DCT-IV matrix given by (9.13).
14. Derive the ***LU***, ***LUD***, ***LDU*** and ***PLUS*** block matrix decompositions for the IntFFT using (9.147), and implement them. Similarly, compare implementations in terms of the number of rounding operators and accuracy.
15. Implement by a computer program the complete analysis/synthesis IntMLT filter bank in MP3 for the short and long blocks using the block matrix decompositions. Then compare the best implementations obtained by local and global approximation methods.
16. Implement by a computer program the complete analysis/synthesis IntMDCT filter bank for an arbitrary symmetric windowing function using the block matrix decompositions.
17. In this book, various perfect reconstruction cosine/sine-modulated filter banks and cosine-modulated QMF banks are discussed in detail, and in particular, their efficient algorithms. In general, any integer filter bank can be constructed as long as it has a fast algorithm. Select a filter bank, design, and construct its integer versions.

References

1. J.A.S. Angus, Finite fields transforms for lossless audio signal processing, in *105th AES Convention*, San Francisco, CA, September (1998). Preprint #483
2. M. Hans, R.W. Schafer, Lossless compression of digital audio. IEEE Signal Process. Mag. **18**(4), 21–32 (2001)
3. ISO/IEC 14496–3:2005/Amd.2:2006, Coding of audio–visual objects – Part 3: audio, amendment 2: audio lossless coding (ALS), New Audio Profiles and BSAC Extensions. International Standards Organization, Geneva, Switzerland (2006)
4. J. Koller, T. Sporer, K. Brandenburg, Improving lossless audio coding, in *Proceedings of the 17th International AES Conference: High–quality Audio Coding*, Florence, Italy, August (1999), pp. 299–306
5. T. Liebchen, Y.A. Reznik, MPEG-4 ALS: an emerging standard for lossless audio coding, in *Proceedings of Data Compression Conference (DCC'2006)*, Snowbird, UT, March (2006), pp. 439–448

6. T. Moriya, N. Iwakami, A. Jin, T. Mori, A design of lossy and lossless scalable audio coding, in *Proceedings of the IEEE ICASSP'2000*, Istanbul, June (2000), pp. 889–892
7. T. Moriya, A. Jin, T. Mori, K. Ikeda, T. Kaneko, Lossless scalable audio coder and quality enhancement, in *Proceedings of the IEEE ICASSP'2002*, vol. 2, Orlando, FL, May (2002), pp. 1829–1832
8. M. Purat, T. Liebchen, P. Noll, Lossless transform coding of audio signals, in *102nd AES Convention*, Munich, Germany, March (1997). Preprint #4414

Integer DCT/DST (IntDCT/IntDST)

9. V. Britanak, P. Yip, K.R. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*, Chap. 5 (Academic, Elsevier Science, Amsterdam, 2007), pp. 141–304
10. A.R. Calderbank, I. Daubechies, W. Sweldens, B.L. Yeo, Wavelet transforms that map integers to integers. Appl. Comput. Harmon. Anal. **5**, 332–369 (1998)
11. R.J. Cintra, F.M. Bayer, C.J. Tablada, Low-complexity 8-point DCT approximations based on integer functions. Signal Process. **99**(6), 201–214 (2014)
12. V.K. Goyal, Transform coding with integer-to-integer transforms. IEEE Trans. Inf. Theory **46**(2), 465–473 (2000)
13. P. Hao, Q. Shi, Matrix factorizations for reversible integer mapping. IEEE Trans. Signal Process. **49**(10), 2314–2324 (2001)
14. X. Li, B. Tao, M.T. Orchard, On implementing transforms from integers to integers, in *Proceedings of the IEEE International Conference on Image Processing (ICIP'98)*, Chicago, IL, October (1998), pp. 881–885
15. G. Plonka, A global method for invertible integer DCT and integer wavelet algorithms. Appl. Comput. Harmon. Anal. **16**(2), 90–110 (2004)
16. G. Plonka, M. Tasche, Invertible integer DCT algorithms. Appl. Comput. Harmon. Anal. **15**(1), 70–88 (2003)
17. M. Primbs, Worst–case error analysis of lifting-based fast DCT-algorithms. IEEE Trans. Signal Process. **53**(8), 3211–3218 (2005). Part 2
18. Y. She, P. Hao, On the necessity and sufficiency of PLUS factorizations. Linear Algebra Appl. **400**, 193–202 (2005)
19. S. Srinivasan, Modulo transforms – An alternative to lifting. IEEE Trans. Signal Process. **54**(5), 1864–1874 (2006)
20. C.J. Tablada, F.M. Bayer, R.J. Cintra, A class of DCT approximations based on the Feigh–Winograd algorithm. Signal Process. **113**(8), 38–51 (2015)
21. J. Wang, J. Sun, S. Yu, 1-D and 2-D transforms from integers to integers, in *Proceedings of the IEEE ICASSP'2003*, vol. 2, Hong Kong, April (2003), pp. 549–552
22. L. Yang, P. Hao, Infinity-norm rotation transforms. IEEE Trans. Signal Process. **57**(7), 2594–2603 (2009)

Integer FFT (IntFFT)

23. S.C. Chan, P.M. Yiu, An efficient multiplierless approximation of the fast Fourier transform using sum-of-powers-of–two (SOPOT) coefficients. IEEE Signal Process. Lett. **9**(10), 322–325 (2002)
24. S. Oraintara, Y.-J. Chen, T. Nguyen, Integer fast Fourier transform `(IntFFT)`, in *Proceedings of the IEEE ICASSP'2001*, Salt Lake City, UT, May (2001)

25. S. Oraintara, Y.-J. Chen, T. Nguyen, Integer fast Fourier transform. IEEE Trans. Signal Process. **50**(3), 607–618 (2002)
26. K.R. Rao, D.N. Kim, J.J. Hwang, Integer fast fourier transform, in *Fast Fourier Transform: Algorithms and Applications* (Springer Science + Business Media B. V., Heidelberg, 2010), pp. 111–126
27. Y. Yokotani, S. Oraintara, R. Geiger, G. Schuller, K.R. Rao, A comparison of integer fast Fourier transforms for lossless coding, in *Proceedings of the International Symposium on Communications and Information Technologies (ISCIT 2004)*, Sapporo, October (2004), pp. 1069–1073

Reversible Integer Lapped Orthogonal Transforms and Integer MLT

28. S.C. Chan, P.M. Yiu, Multiplier–less discrete sinusoidal and lapped transforms using sum-of-powers-of-two (SOPOT) coefficients, in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'2001)*, vol. 2, Sydney, May (2001), pp. 13–16
29. K. Komatsu, K. Sezaki, Design of lossless block transforms and filter banks for image coding. IEICE Trans. Fundam. **E82–A**(8), 1656–1664 (1999)
30. K. Komatsu, K. Sezaki, Design of lossless LOT and its performance evaluation, in *Proceedings of the IEEE ICASSP'2000*, vol. 4, Istanbul, June (2000), pp. 2119–2122
31. J. Li, A progressive to lossless embedded audio coder (PLEAC) with reversible modulated lapped transform, in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'2003)*, vol. 3, Baltimore, MD, July (2003), pp. 221–224
32. H.S. Malvar, Lossless and near-lossless audio compression using integer-reversible modulated lapped transforms, in *Proceedings of the IEEE Data Compression Conference (DCC'2007)*, Snowbird, UT, March (2007)
33. T.D. Tran, The LiftLT: fast-lapped transforms via lifting steps. IEEE Signal Process Lett. **7**(6), 145–148 (2000)

Integer MDCT (IntMDCT) and Approximation Error

34. S.C. Chan, W. Yiu, K.L. Ho, Multiplierless perfect reconstruction modulated filter banks with sum-of-powers-of-two coefficients. IEEE Signal Process. Lett. **8**(6), 163–166 (2001)
35. R. Geiger, G. Schuller, Integer low delay and MDCT filter banks, in *Proceedings of the 36th Asilomar Conference on Signals, Systems and Computers*, vol. 1, Pacific Grove, CA, November (2002), pp. 811–815
36. R. Geiger, Y. Yokotani, G. Schuller, Improved integer transforms for lossless audio coding, in *Proceedings of the 37th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, November (2003), pp. 2119–2123
37. R. Geiger, T. Sporer, J. Koller, K. Brandenburg, Audio coding based on integer transforms, in *111th AES Convention*, New York, NY, September (2001). Preprint #5471
38. R. Geiger, J. Herre, J. Koller, K. Brandenburg, IntMDCT – A link between perceptual and lossless audio coding, in *Proceedings of the IEEE ICASSP'2002*, vol. 2, Orlando, FL, May (2002), pp. 1813–1816
39. R. Geiger, J. Herre, G. Schuller, T. Sporer, Fine grain scalable perceptual and lossless audio coding based on IntMDCT, in *Proceedings of the IEEE ICASSP'2003*, vol. 5, Hong Kong, April (2003), pp. 445–448

40. R. Geiger, G. Schuller, T. Sporer, Fine grain scalable perceptual and lossless audio coding based on IntMDCT, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, October (2003), p. 50
41. R. Geiger, Y. Yokotani, G. Schuller, J. Herre, Improved integer transforms using multidimensional lifting, in *Proceedings of the IEEE ICASSP'2004*, vol. 2, Montreal, May (2004), pp. 1005–1008
42. H. Huang, R. Yu, X. Lin, S. Rahardja, Method for realising reversible integer type-IV discrete cosine transform. Electron. Lett. **40**(8), 514–515 (2004)
43. H. Huang, S. Rahardja, R. Yu, X. Lin, A fast algorithm of integer MDCT for lossless audio coding, in *Proceedings of the IEEE ICASSP'2004*, vol. 4, Montreal, May (2004), pp. 177–180
44. H. Huang, S. Rahardja, R. Yu, X. Lin, Integer MDCT with enhanced approximation of the DCT-IV. IEEE Trans. Signal Process. **54**(3), 1156–1159 (2006)
45. J. Li, Reversible FFT and MDCT via matrix lifting, in *Proceedings of the IEEE ICASSP'2004*, vol. 4, Montreal, May (2004), pp. 173–176
46. J. Li, Low noise reversible MDCT (RMDCT) and its application in progressive-to-lossless embedded audio coding. IEEE Trans. Signal Process. **53**(5), 1870–1880 (2005)
47. V.M. Prasad, C.D. Creusere, Analyzing reversible lapped transforms using Reng probing, in *Proceedings of the 40th Asilomar Conference on Signals, System and Computers*, Pacific Grove, CA, October–November (2006), pp. 873–877
48. Y. Yokotani, S. Oraintara, Lossless audio compression using integer modified discrete cosine transform, in *Proceedings of the IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS'2003)*, Awaji Island, December (2003), pp. 120–126
49. Y. Yokotani, R. Geiger, G. Schuller, S. Oraintara, K.R. Rao, Improved lossless audio coding using the noise-shaped IntMDCT, in *Proceedings of the IEEE 11th Digital Signal Processing Workshop and Signal Processing Education Workshop*, Taos Ski Valley, NM, August (2004), pp. 356–360
50. Y. Yokotani, S. Oraintara, R. Geiger, G. Schuller, K.R. Rao, Approximation error analysis for transform-based lossless audio coding, in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'2004)*, vol. 2, Dallas, TX, November–December (2004), pp. 595–599
51. Y. Yokotani, R. Geiger, G.D.T. Schuller, S. Oraintara, K.R. Rao, Lossless audio coding using the IntMDCT and rounding error shaping. IEEE Trans. Audio Speech Lang. Process. **14**(6), 2201–2211 (2006)
52. Y. Zhang, G. Gao, A scalable and lossless audio coding system based on integer transform, in *Proceedings of the IEEE International Symposium on Communications and Information Technologies (ISCIT'2006)*, Bangkok, September–October (2006), pp. 414–417
53. Y. Zhang, R. Hu, Scalable audio coding based on integer transform, in *Proceedings of the IEEE 1st International Conference on Communications and Networking in China (ChinaCom'2006)*, Beijing, October (2006), pp. 1–5

IntMDCT in MP3 Audio Coding

54. T. Krishnan, S. Oraintara, Fast and lossless implementation of the forward and inverse MDCT computation in MPEG audio coding, in *Proceedings of of the IEEE International Symposium on Circuits and Systems (ISCAS'2002)*, vol. 2, Phoenix, Scottsdale, AR, May (2002), pp. 181–184
55. T. Krishnan, S. Oraintara, The integer MDCT and its application in the MPEG layer III audio, in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'2003)*, vol. 4, Bangkok, May (2003), pp. 301–304

56. H.-J. Quan, T. Zhang, Y.-N. Che, L. Zhao, The application of integer MDCT in MP3 audio, in *Proceedings of the IEEE International Conference on Computer Science and Network Technology*, Dalian, December (2011), pp. 1153–1157
57. L. Wang, J. Wu, L. Senhadji, H. Shu, Comparison of three IntMDCT algorithms in audio compression. J. Southwest Univ. (Nat. Sci. Ed.). **42**(2), 259–264 (2012) [in Chinesse]

MPEG-4 HD-AAC/SLS Scalable Lossless Audio Coding Standard

58. R. Geiger, G. Schuller, J. Herre, R. Sperschneider, T. Sporer, Scalable perceptual and lossless audio coding based on MPEG-4 AAC, in *115th AES Convention*, New York, NY, October (2003). Preprint #5868
59. R. Geiger, R. Yu, J. Herre, S. Rahardja, S.-W. Kim, X. Lin, M. Schmidt, ISO/IEC MPEG-4 High-definition scalable advanced audio coding. J. Audio Eng. Soc. **55**(1/2), 27–43 (2007)
60. ISO/IEC JTC1/SC29/WG11 Moving Picture Experts Group, Call for proposals on MPEG-4 lossless audio coding. No. N5040, Awaji Island (2002)
61. ISO/IEC JTC1/SC29/WG11 Moving Picture Experts Group, Final call for proposals on MPEG-4 lossless audio coding, No. N5208, Shanghai, China, October (2002)
62. ISO/IEC JTC1/SC29/WG11 Moving Picture Experts Group, Workplan for audio scalable lossless audio coding (SLS). No. N5720, Trondheim, Norway, July (2003)
63. ISO/IEC 14496–3:2005/Amd.3:2006, Coding of audio–visual objects – Part 3: audio, amendment 3: scalable lossless coding (SLS). International Standards Organization, Geneva, Switzerland (2006)
64. T. Li, S. Rahardja, R. Yu, S. Koh, Study on rounding errors of IntMDCT in perceptual audio coding, in *Proceedings of the 7th IEEE International Symposium on Multimedia (ISM'2005)*, Irvine, CA, December (2005)
65. T. Li, S. Rahardja, R. Yu, S.N. Koh, On integer MDCT for perceptual audio coding. IEEE Trans. Audio Speech Lang. Process. **15**(8), 2236–2248 (2007)
66. R. Yu, X. Lin, S. Rahardja, C.C. Ko, A fine granular scalable perceptually lossy and lossless audio codec, in *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME'2003)*, vol. I, Baltimore, MD, July (2003), pp. 65–68
67. R. Yu, X. Lin, S. Rahardja, C.C. Ko, A scalable lossy to lossless audio coder for MPEG-4 lossless audio coding, in *Proceedings of the IEEE ICASSP'2004*, vol. III, Montreal, Canada, May (2004), pp. 1004–1007
68. R. Yu, R. Geiger, S. Rahardja, J. Herre, X. Lin, H. Huang, MPEG-4 scalable to lossless audio coding, in *117th AES Convention*, San Francisco, CA, October (2004). Preprint #6183
69. R. Yu, X. Lin, S. Rahardja, C. C. Ko, H. Huang, Improving coding efficiency for MPEG-4 audio scalable lossless coding, in *Proceedings of the IEEE ICASSP'2005*, vol. III, Philadephia, PA, May (2005), pp. 169–172
70. R. Yu, X. Lin, S. Rahardja, H. Huang, MPEG-4 scalable to lossless audio coding – Emerging international standard for digital audio compression, in *Proceedings of the IEEE 7th Workshop on Multimedia Signal Processing*, Shanghai, China, October–November (2005), pp. 1–4
71. R. Yu, S. Rahardja, X. Lin, C.C. Ko, A fine granular scalable to lossless audio coding. IEEE Trans. Audio Speech Lang. Process. **14**(4), 1352–1363 (2006)
72. R. Yu, T. Li, S. Rahardja, Perceptually enhanced bit–plane coding for scalable audio, in *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME'2006)*, Toronto, July (2006), pp. 1153–1156

Supporting Literature

73. T.M. Apostol, Historical introduction, in *Introduction to Analytic Number Theory* (Springer, New York, 1976), pp. 1–12
74. G. Bi, Y. Zeng, *Transforms and Fast Algorithms for Signal Analysis and Representations*, Chap. 6 (Birkhäuser, Boston, 2004), pp. 210–211
75. V. Britanak, New universal rotation-based fast computational structures for an efficient implementation of the DCT-IV/DST-IV and analysis/synthesis MDCT/MDST filter banks. Signal Process. **89**(11), 2213–2232 (2009)
76. V. Britanak, H.J. Lincklaen Arriëns, Fast computational structures for an efficient implementation of the complete TDAC analysis/synthesis MDCT/MDST filter banks. Signal Process. **89**(7), 1379–1394 (2009)
77. V. Britanak, K.R. Rao, A new fast algorithm for the unified forward and inverse MDCT/MDST computation. Signal Process. **82**(3), 433–459 (2002)
78. M. Bosi, R.E. Goldberg, Audio coding standards, in *Introduction to Digital Audio Coding and Standards*, Part II (Springer Science+Business Media, New York, 2003)
79. P. Duhamel, B. Piron, J.M. Etcheto, On computing the inverse DFT. IEEE Trans. Acoust. Speech Signal Process. **36**(2), 285–286 (1988)
80. F.R. Gantmacher, *The Theory of Matrices*, 2nd edn. (Nauka, Moscow, 1966) [in Russian], English translation: Vols. 1 and 2, (Chelsea, New York, 1959)
81. G.H. Golub, C.F. Van Loan, *Matrix Computations*, 3rd edn. (The Johns Hopkins University Press, Baltimore, MD, 1996)
82. H.S. Malvar, *Signal Processing with Lapped Transforms*, Chap. 2 (Artech House, Norwood, MA, 1992), pp. 71–75
83. H.J. Nussbaumer, Elements of number theory and polynomial algebra, in *Fast Fourier Transform and Convolution Algorithms*, Chap. 2 (Springer, Berlin, 1981), pp. 4–31
84. G. Plonka, M. Tasche, Fast and numerically stable algorithms for discrete cosine transforms. Linear Algebra Appl. **394**(1), 309–345 (2005)
85. J.P. Princen, A.W. Johnson, A.B. Bradley, Sub–band/transform coding using filter bank designs based on time domain aliasing cancellation, in *Proceedings of the IEEE ICASSP'87*, Dallas, TX, April (1987), pp. 2161–2164
86. G.W. Stewart, Gaussian elimination, in *Matrix Algorithms, Volume I: Basic Decompositions*, Chap. 3 (SIAM Society for Industrial and Applied Mathematics, Philadelpha, PA, 1998), pp. 149–250
87. J. Wu, H. Shu, L. Senhadji, L. Luo, Mixed-radix algorithm for the computation of forward and inverse MDCTs. IEEE Trans. Circuits Syst. Regul. Pap. **56**(4), 784–794 (2009)

Appendix A
Selected Mathematical Basics from Matrix Theory and Linear Algebra

In this appendix the mathematical basics from classical matrix theory and computational methods of linear algebra are summarized [17, 20–22, 24, 39, 43].

A.1 Generalized Inverse or Pseudoinverse Matrix

It is well known that if $\boldsymbol{A}$ is a real square nonsingular matrix, then there exists its unique inverse matrix denoted by $\boldsymbol{A}^{-1}$. However in general, if $\boldsymbol{A}$ is an $m \times n$ matrix ($m \neq n$), then the inverse matrix for $\boldsymbol{A}$ does not exist. Nevertheless, for an arbitrary $m \times n$ matrix $\boldsymbol{A}$ exists a generalized inverse or pseudoinverse matrix denoted by $\boldsymbol{A}^{+}$, which possesses some properties of the inverse matrix [22]. The pseudoinverse matrices are directly related to the minimum norm and they have played an important role in solution of overdetermined systems of linear equations (least squares problem) [21, 24].

Let $\boldsymbol{A}$ be a real $m \times n$ matrix of rank r (matrix $\boldsymbol{A}$ has the full rank if $r = \min(m, n)$, and it is rank deficient if $r < \min(m, n)$). For an arbitrary matrix $\boldsymbol{A}$ there exists exactly one $n \times m$ matrix $\boldsymbol{A}^{+}$ satisfying four Moore–Penrose conditions [21, 24]

$$\begin{aligned}
&\text{(a)} \quad \boldsymbol{A}\boldsymbol{A}^{+}\boldsymbol{A} = \boldsymbol{A}, \\
&\text{(b)} \quad \boldsymbol{A}^{+}\boldsymbol{A}\boldsymbol{A}^{+} = \boldsymbol{A}^{+}, \\
&\text{(c)} \quad (\boldsymbol{A}^{+}\boldsymbol{A})^{T} = \boldsymbol{A}^{+}\boldsymbol{A}, \quad (\boldsymbol{A}^{+}\boldsymbol{A})^{2} = \boldsymbol{A}^{+}\boldsymbol{A}, \\
&\text{(d)} \quad (\boldsymbol{A}\boldsymbol{A}^{+})^{T} = \boldsymbol{A}\boldsymbol{A}^{+}, \quad (\boldsymbol{A}\boldsymbol{A}^{+})^{2} = \boldsymbol{A}\boldsymbol{A}^{+}.
\end{aligned} \tag{A.1}$$

The matrix $\boldsymbol{A}^{+}$ is said to be the generalized inverse or pseudoinverse of $\boldsymbol{A}$. Conditions (c) and (d) emphasize the fact that the matrices $\boldsymbol{A}^{+}\boldsymbol{A}$ and $\boldsymbol{A}\boldsymbol{A}^{+}$ are Hermitian and involutory, i.e., they are symmetric and their second power is equal to the original matrix. In the case of $m \times n$ matrix $\boldsymbol{A}$, $m > n$, if rank $(\boldsymbol{A}) = n$, then

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1

the pseudoinverse matrix $\boldsymbol{A}^+$ is given by $\boldsymbol{A}^+ = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$, while if rank $(\boldsymbol{A}) = m = n$, then $\boldsymbol{A}^+ = \boldsymbol{A}^{-1}$. Generally, for an $m \times n$ matrix $\boldsymbol{A}$ of rank r we can perform the so-called skeleton decomposition $\boldsymbol{A} = \boldsymbol{BC}$, where $\boldsymbol{B}$ is $m \times r$ and $\boldsymbol{C}$ is $r \times n$ matrix [22]. If such a decomposition is known, then the pseudoinverse matrix $\boldsymbol{A}^+$ can be computed from the formula

$$\boldsymbol{A}^+ = \boldsymbol{C}^+\boldsymbol{B}^+ = \boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{C}^T)^{-1}(\boldsymbol{B}^T\boldsymbol{B})^{-1}\boldsymbol{B}^T. \tag{A.2}$$

The pseudoinverse matrix $\boldsymbol{A}^+$ can be alternatively obtained via the Singular Value Decomposition (***SVD***) [24] or ***QR*** matrix factorization [21]. According to the ***SVD*** for a real $m \times n$ matrix $\boldsymbol{A}$ of rank r there exists an orthogonal matrix $\boldsymbol{U}$ of order m, an orthogonal matrix $\boldsymbol{V}$ of order n, and $m \times n$ diagonal matrix $\boldsymbol{\Sigma}$ of rank r with positive diagonal elements that

$$\boldsymbol{U}^T\boldsymbol{A}\boldsymbol{V} = \boldsymbol{\Sigma}, \quad \boldsymbol{\Sigma} = \text{diag}\,\{\sigma_1, \sigma_2, \ldots, \sigma_r, 0, \ldots, 0\}, \tag{A.3}$$

where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ are called the singular values of $\boldsymbol{A}$. Because $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices, $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{I}$ and $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix, consequently

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T. \tag{A.4}$$

From the ***SVD*** decomposition it follows that the pseudoinverse matrix $\boldsymbol{A}^+$ is given by

$$\boldsymbol{A}^+ = \boldsymbol{V}\boldsymbol{\Sigma}^+\boldsymbol{U}^T, \quad \boldsymbol{\Sigma}^+ = \text{diag}\left\{\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \ldots, \frac{1}{\sigma_r}, 0, \ldots, 0\right\}, \tag{A.5}$$

and it satisfies four Moore–Penrose conditions. In particular, matrices $\boldsymbol{A}\boldsymbol{A}^+$ and $\boldsymbol{A}^+\boldsymbol{A}$ are given by Fiedler [21]

$$\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{U}\begin{bmatrix}\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}\boldsymbol{U}^T, \qquad \boldsymbol{A}^+\boldsymbol{A} = \boldsymbol{V}\begin{bmatrix}\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}\boldsymbol{V}^T, \tag{A.6}$$

where $\boldsymbol{0}$ is the null matrix. The computational methods for the ***SVD*** decomposition are presented in [24].

According to ***QR*** matrix factorization of a real $m \times n$ matrix $\boldsymbol{A}$, $m > n$, with linear independent columns there exists uniquely an $m \times n$ matrix $\boldsymbol{Q}$ and an $n \times n$ matrix $\boldsymbol{R}$ so that $\boldsymbol{Q}^T\boldsymbol{Q}$ is diagonal matrix with positive diagonal elements d_i, and $\boldsymbol{R}$ is the unit upper triangular matrix [21], that is

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}, \qquad \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{D} = \text{diag}\{d_1, \ldots, d_n\}. \tag{A.7}$$

If such $\boldsymbol{QR}$ factorization of the matrix $\boldsymbol{A}$ is known, then the pseudoinverse matrix $\boldsymbol{A}^+$ is given by

$$\boldsymbol{A}^+ = \boldsymbol{R}^{-1}\boldsymbol{D}^{-1}\boldsymbol{Q}^T, \qquad \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{D}. \tag{A.8}$$

The algorithm for computation of $\boldsymbol{QR}$ factorization is presented in [21].

A.2 Hankel Matrices and the Efficient Hankel Matrix-Vector Products

A symmetric matrix $\boldsymbol{H}_N$ of order N, in general with complex-valued elements, derived in the explicit form as [43]

$$\boldsymbol{H}_N = \{h_{i+j}\}_{i,j=0}^{N-1} = \begin{pmatrix} h_0 & h_1 & h_2 & \dots & h_{N-2} & h_{N-1} \\ h_1 & h_2 & h_3 & \dots & h_{N-1} & h_N \\ h_2 & h_3 & h_4 & \dots & h_N & h_{N+1} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ h_{N-2} & h_{N-1} & h_N & \dots & h_{2N-4} & h_{2N-3} \\ h_{N-1} & h_N & h_{N+1} & \dots & h_{2N-3} & h_{2N-2} \end{pmatrix}, \tag{A.9}$$

is called the Hankel matrix. From the structure of $\boldsymbol{H}_N$ one can observe that all elements on opposite diagonals are identical., i.e., it has a cyclic structure. Addition/subtraction of two Hankel matrices of the same order is a Hankel matrix. Algebraic theory and properties of Hankel matrices are presented in [43].

Hankel matrices have played key role in the representation of short-length cyclic convolutions and their efficient implementations by the so-called bilinear algorithms for the computation of N-point Hankel matrix-vector products (multiplications of $N \times N$ Hankel matrices and N-point vectors) [33]. The simplest case 2-point Hankel matrix-vector product is defined as [33]

$$\begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}, \qquad a, b, c \in R, \tag{A.10}$$

and the corresponding bilinear algorithm for 2-point Hankel matrix-vector product is shown in Fig. A.1. It consists of three sequential stages: The first and last stages of additions and a middle stage of multiplications which are independent and therefore, they can be realized concurrently (preferred in hardware implementations). A bilinear algorithm for 3-point Hankel matrix-vector product can be found in [33].

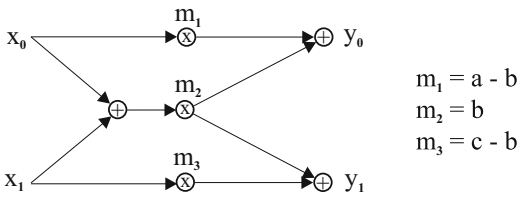

Fig. A.1 The bilinear algorithm for 2-point Hankel matrix-vector product

A.3 Algebra of Real Square Matrices

Let $\boldsymbol{A}$ and $\boldsymbol{B}$ are square nonsingular matrices of order N. Then, the transposition and inverse of the matrix product $\boldsymbol{AB}$ is, respectively, defined as [17, 20, 22, 24]

$$(\boldsymbol{AB})^T = \boldsymbol{B}^T\boldsymbol{A}^T, \qquad (\boldsymbol{AB})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}, \tag{A.11}$$

i.e., the transposition/inverse of the matrix product $\boldsymbol{AB}$ is the reverse product of the transposed/inverse matrices. Additionally, the transpose of the inverse matrix $(\boldsymbol{A}^{-1})^T$ is the inverse of the transpose [17, 24]

$$(\boldsymbol{A}^{-1})^T = (\boldsymbol{A}^T)^{-1}. \tag{A.12}$$

A.3.1 The Determinant

If $\boldsymbol{A} = \{a_{ij}\}$ is a square nonsingular matrix of order N, then the determinant of $\boldsymbol{A}$ is defined in terms of order $N-1$ determinants [24]

$$\det(\boldsymbol{A}) = \sum_{j=1}^{N} (-1)^{j+1}\, a_{1j}\, \det(\boldsymbol{A}_{1j}), \tag{A.13}$$

where $\boldsymbol{A}_{1j}$ is an $(N-1)\times(N-1)$ matrix obtained by deleting the first row and jth column of $\boldsymbol{A}$. Useful properties of the determinant include [17, 24]:

- $\det(\boldsymbol{AB}) = \det(\boldsymbol{BA}) = \det(\boldsymbol{A})\det(\boldsymbol{B})$, $\boldsymbol{A}$ and $\boldsymbol{B}$ are nonsingular matrices,
- $\det(\boldsymbol{A}^T) = \det(\boldsymbol{A})$,
- $\det(c\boldsymbol{A}) = c^N \det(\boldsymbol{A})$, $c \in R$,
- $\det(\boldsymbol{A}) \neq 0$ if $\boldsymbol{A}$ is nonsingular,

A.3.2 Orthogonal/Orthonormal Matrices

Matrix $\boldsymbol{A}$ is called orthogonal, if $\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. Additionally, if the norm of each row (basis vector) of the matrix is equal to 1, then the matrix is orthonormal. The orthogonal/orthonormal matrices possess a few useful properties [20]:

- The identity matrix $\boldsymbol{I}$ is orthogonal/orthonormal.
- If $\boldsymbol{A}$ is orthogonal/orthonormal, then $\boldsymbol{A}^{-1} = \boldsymbol{A}^T$.
- If $\boldsymbol{A}$ is orthogonal/orthonormal, then $\boldsymbol{A}^T$ is also orthogonal/orthonormal.
- The product of two orthogonal/orthonormal matrices is an orthogonal/orthonormal matrix.
- The determinant of orthogonal/orthonormal matrix is equal to ± 1. If $\det(\boldsymbol{A}) = +1$, then $\boldsymbol{A}$ is called to be eigenorthogonal/eigenorthonormal. Otherwise, if $\det(\boldsymbol{A}) = -1$, then A is called to be non-eigenorthogonal/non-eigenorthonormal.

A.3.3 Algebra of Triangular Matrices

A matrix with all elements under/above the main diagonal equal to zero is called an upper/lower triangular matrix. A unit triangular matrix is triangular matrix with 1s on the main diagonal. There are a few useful properties about products, determinants, and inverses of triangular matrices [24]:

- The product of two upper (lower) triangular matrices is upper (lower) triangular matrix.
- The product of two unit upper (unit lower) triangular matrices is unit upper (unit lower) triangular matrix.
- The determinant of upper or lower triangular matrix and in particular, the determinant of a diagonal matrix, is equal to the product of its diagonal elements.
- The determinant of unit upper or unit lower triangular matrix is equal to 1.
- The inverse of upper (lower) triangular matrix is upper (lower) triangular.
- The inverse of unit upper (unit lower) triangular matrix is unit upper (unit lower) triangular.

A.3.4 Matrix and Vector Norms

The matrix and vector norms are frequently used for the analysis of matrix algorithms in linear algebra and matrix computations. They provide a measure of distance on the space of matrices/vector space, or more precisely, the space of matrices/vector space together with matrix/vector norms define a metric space [17, 24]. In general, matrix norms are defined for an arbitrary matrix, i.e., also for nonsquare matrices.

The norm of a matrix $\boldsymbol{A} = \{a_{ij}\}$ is a real number $\|\boldsymbol{A}\|$ satisfying the following properties [17, 24]:

- $\|\boldsymbol{A}\| \geq 0, \quad (\|\boldsymbol{A}\| = 0 \text{ iff } \boldsymbol{A} = 0)$,
- $\|\alpha.\boldsymbol{A}\| = |\alpha|.\|\boldsymbol{A}\|, \quad \alpha \in R$, and in particular $\|-\boldsymbol{A}\| = \|\boldsymbol{A}\|$,
- $\|\boldsymbol{A} + \boldsymbol{B}\| \leq \|\boldsymbol{A}\| + \|\boldsymbol{B}\|$,
- $\|\boldsymbol{AB}\| \leq \|\boldsymbol{A}\|\|\boldsymbol{B}\|$, and in particular $\|\boldsymbol{A}^r\| \leq \|\boldsymbol{A}\|^r$, where $r > 0$ is integer.

One important inequality between the norms of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of the same type is given by

$$\|\boldsymbol{A}-\boldsymbol{B}\| \geq |\, \|\boldsymbol{B}\| - \|\boldsymbol{A}\| \,|. \tag{A.14}$$

The matrix norm is called canonical, if satisfies additional properties:

- $|a_{ij}| \leq \|\boldsymbol{A}\|$,
- The inequality $|\boldsymbol{A}| \leq |\boldsymbol{B}|$ implies that $\|\boldsymbol{A}\| \leq \|\boldsymbol{B}\|$.

Subscripts on $\|.\|$ are used to distinguish between various norms. The most frequently used and easily computed matrix norms are [17, 24]

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_i \sum_j |a_{ij}|^2}, \qquad \text{Frobenius norm,} \tag{A.15}$$

$$\|\boldsymbol{A}\|_1 = \max_i \sum_j |a_{ij}|, \qquad \text{1-norm,} \tag{A.16}$$

$$\|\boldsymbol{A}\|_\infty = \max_j \sum_i |a_{ij}|, \qquad \infty\text{-norm,} \tag{A.17}$$

It can be verified that matrix norms $\|\boldsymbol{A}\|_F$, $\|\boldsymbol{A}\|_1$ and $\|\boldsymbol{A}\|_\infty$ are canonical. For a vector $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$ these norms are defined as

$$\|\mathbf{x}\|_2 = |\mathbf{x}| = \sqrt{|x_1|^2 + |x_2|^2 + \ldots + |x_N|^2}, \qquad \text{Euclidean norm or 2-norm,} \tag{A.18}$$

$$\|\mathbf{x}\|_1 = |x_1| + |x_2| + \ldots + |x_N|, \tag{A.19}$$

$$\|\mathbf{x}\|_\infty = \max_i |x_i|, \qquad \text{maximum norm.} \tag{A.20}$$

A unit vector with respect to the norm $\|.\|$ is a vector $\mathbf{x}$ that satisfies $\|\mathbf{x}\| = 1$. We note that vector norms are derived from the class of the so-called vector p-norms defined as [24]

$$\|\mathbf{x}\|_p = (|x_1|^p + |x_2|^p + \ldots + |x_N|^p)^{\frac{1}{p}}, \quad p \geq 1.$$

A very important property concerning the vector norms is Cauchy–Schwartz inequality:

$$|\mathbf{x}\mathbf{y}^T| \leq \|\mathbf{x}\|_2 \, \|\mathbf{y}\|_2. \tag{A.21}$$

Finally, for the identity matrix of order N we have

$$\|\boldsymbol{I}\|_F = \sqrt{N}, \quad \|\boldsymbol{I}\|_1 = \|\boldsymbol{I}\|_\infty = 1.$$

A.3.5 Well- and Ill-Conditioned Matrices

In linear algebra the analysis of numerical sensitivity of direct methods for solving systems of linear equations represented by a square nonsingular matrix is associated with the notion of well- and ill-conditioned systems or matrices [21, 24]. Let a system of linear equations is represented by $\boldsymbol{A}\mathbf{x} = \mathbf{b}$, where $\boldsymbol{A}$ is a square nonsingular matrix, $\mathbf{x}$ and $\mathbf{b}$ are vectors. Aim of the analysis of numerical sensitivity is to examine how the relative errors in $\boldsymbol{A}$ and $\mathbf{b}$ affect the solution $\mathbf{x}$.

We say that the system of linear equations or matrix $\boldsymbol{A}$ is well-conditioned if has the following property: For some matrix norm $\|\boldsymbol{A}\|$, the number

$$\kappa\ (\boldsymbol{A}) = \|\boldsymbol{A}\|\ \|\boldsymbol{A}^{-1}\|, \tag{A.22}$$

satisfying always $\kappa\ (\boldsymbol{A}) \geq 1$, is not substantially larger than 1 (for example 100× or 1000×). Otherwise, the system of linear equations or matrix $\boldsymbol{A}$ is ill-conditioned. The number $\kappa\ (\boldsymbol{A})$ is called the condition number of $\boldsymbol{A}$ with respect to the matrix norm $\|\boldsymbol{A}\|$. Thus, the condition number $\kappa\ (\boldsymbol{A})$ quantifies the sensitivity of the $\boldsymbol{A}\mathbf{x} = \mathbf{b}$ problem. If the condition number $\kappa\ (\boldsymbol{A}) = 1$, then $\boldsymbol{A}$ is perfectly conditioned [24].

A.4 Elementary Rotation Matrices

Elementary rotation matrices $\boldsymbol{G}_{ij}$ are defined as [20]

$$\boldsymbol{G}_{ij} = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & c & \ldots & -s & \\ & & \vdots & \ddots & \vdots & \\ & & s & \ldots & c & \\ & & & & & \ddots \\ & & & & & & 1 \end{pmatrix}, \quad \boldsymbol{G}_{ij}^{-1} = \boldsymbol{G}_{ij}^{T} = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & c & \ldots & s & \\ & & \vdots & \ddots & \vdots & \\ & & -s & \ldots & c & \\ & & & & & \ddots \\ & & & & & & 1 \end{pmatrix}, \tag{A.23}$$

where $c^2 + s^2 = 1$, $c = \cos\varphi$, $s = \sin\varphi$ for some angle φ. The subscript indices $i, j \quad (i < j)$ indicate positions of elements c and s in the matrix. The elementary rotation matrices given by (A.23) differ from the identity matrix only by four elements placed on crossing two rows and two columns. They are orthogonal transformations playing a central role in the least squares solutions of overdetermined systems of linear equations and symmetric eigenvalue problems [24]. Elementary rotation matrices $\boldsymbol{G}_{ij}$ are known as the Givens–Jacobi rotations. They are eigenorthogonal, i.e., $\det\ (\boldsymbol{G}_{ij}) = \det\ \left(\boldsymbol{G}_{ij}^{T}\right) = +1$, whereby $\boldsymbol{G}_{ij}$ and

$\boldsymbol{G}_{ij}^{T}$ are inverses to each other. Through the pre-multiplications (multiplications on the left) and/or post-multiplications (multiplications on the right) of a nonsingular matrix by elementary rotation matrices we can reduce the given matrix into various canonical forms introducing zero elements in the matrix by properly choosing the rotation angle. For our purposes it is sufficient to consider the simplest case of 2×2 elementary rotation matrices.

A 2×2 orthogonal (eigenorthogonal) matrix is called Givens–Jacobi rotation, if it has the form [24]

$$\boldsymbol{G}_{\varphi} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}, \quad \boldsymbol{G}_{\varphi}^{-1} = \boldsymbol{G}_{\varphi}^{T} = \boldsymbol{G}_{-\varphi}. \tag{A.24}$$

The computational structures for efficient implementations of Givens–Jacobi rotations are presented in Appendices F.2 and F.3.

A.5 Elementary Transformations of Matrices

Elementary transformations are elementary matrices of a special form frequently performed upon matrices as follows [20]:

1. Multiplication to the elements of some row by a number α
2. Adding to the elements of some row numbers proportional (α multiple) to the elements of some preceding row
3. Adding to the elements of some row numbers proportional (α multiple) to the elements of some following row

Sometimes such elementary transformations are made upon the columns of the matrix. Any elementary transformation of the rows is equivalent to a pre-multiplication of the matrix by a nonsingular matrix of a special form. Thus, the operation (1) is equivalent to a pre-multiplication by the matrix

$$\begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & \alpha & & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}. \tag{A.25}$$

Operation (2) is equivalent to a pre-multiplication by the matrix

$$\begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & \vdots & \ddots & \\ & & \alpha & \dots & 1 \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}, \tag{A.26}$$

and the operation (3) is equivalent to a pre-multiplication by the matrix

$$\begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & \dots & \alpha \\ & & & \ddots & \vdots \\ & & & & 1 \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}. \tag{A.27}$$

Operations (1), (2), and (3) can be performed alternatively upon the columns in post-multiplications of the matrix. It can be seen that the elementary matrix of the form (2) defined by (A.26) is unit lower triangular, and is called Gauss elementary matrix. On the other hand, the elementary matrix of the form (3) defined by (A.27) is unit upper triangular. The inverses of elementary matrices of the form (A.26) and (A.27) are, respectively, defined as

$$\begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & \vdots & \ddots & \\ & & -\alpha & \dots & 1 \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & \dots & -\alpha \\ & & & \ddots & \vdots \\ & & & & 1 \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}. \tag{A.28}$$

Note: In general, the elementary rotation matrices can be thought of as the elementary transformations, which are orthogonal.

A.6 Matrix Decompositions

Matrix decompositions are factorizations of matrices into products of simpler matrices. In linear algebra and matrix computations, a real nonsingular matrix is reduced by elementary rotation matrices and elementary transformations into various canonical forms in order to simplify subsequent computational steps of a solved problem. Such procedures lead to various useful factorizations of the matrix into the products of structurally simpler matrices.

A.6.1 QR *Matrix Factorization*

There exist two basic methods to reduce a real nonsingular matrix of order N into equivalent upper triangular form. The first method is based on pre-multiplications of the matrix by elementary rotation matrices. This procedure leads to the well-known $\boldsymbol{QR}$ matrix factorization, where $\boldsymbol{Q}$ is an orthogonal matrix and $\boldsymbol{R}$ is an upper triangular matrix. The following theorem and corollaries state the $\boldsymbol{QR}$ matrix factorization [20].

Theorem 5 ([20], Chapter 1, p. 37) *Arbitrary real square nonsingular matrix*

$$\boldsymbol{A} = \begin{pmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \dots & a_{1n}^{(0)} \\ a_{21}^{(0)} & a_{22}^{(0)} & \dots & a_{2n}^{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(0)} & a_{n2}^{(0)} & \dots & a_{nn}^{(0)} \end{pmatrix}$$

can be reduced through successive pre-multiplications by elementary rotation matrices $\boldsymbol{G}_{ij}$ to an upper triangular matrix, whose all diagonal elements are positive besides the last one, that is,

$$\boldsymbol{A}^{(n-1)} = \boldsymbol{G}_{n-1,n} \dots \boldsymbol{G}_{13}\, \boldsymbol{G}_{12}\, \boldsymbol{A} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn}^{(n-1)} \end{pmatrix}. \tag{A.29}$$

Corollary 1 ($\boldsymbol{QR}$ Matrix Factorization) *Arbitrary real nonsingular matrix $\boldsymbol{A}$ is the product of an eigenorthogonal matrix $\boldsymbol{Q}$ and an upper triangular matrix $\boldsymbol{R}$, i.e., $\boldsymbol{A} = \boldsymbol{QR}$, where*

$$\boldsymbol{Q} = (\boldsymbol{G}_{n-1,n} \dots \boldsymbol{G}_{13}\, \boldsymbol{G}_{12})^{-1} \quad \textit{and} \quad \boldsymbol{R} = \boldsymbol{A}^{(n-1)}. \tag{A.30}$$

Corollary 2 *Any eigenorthogonal matrix A is the product of at most* $\frac{n(n-1)}{2}$ *elementary rotation matrices, i.e.,*

$$\boldsymbol{A} = \boldsymbol{G}_{12}^{-1}\, \boldsymbol{G}_{13}^{-1}\, \ldots\, \boldsymbol{G}_{n-1,n}^{-1}\, \boldsymbol{A}^{(n-1)}, \quad \textit{where} \quad \boldsymbol{A}^{(n-1)} = \boldsymbol{I}. \tag{A.31}$$

A.6.2 LU *and* LDU *Matrix Factorizations*

The second method to reduce a real nonsingular matrix to the upper triangular form is based on pre-multiplications of the matrix by elementary transformations, specifically, by elementary matrices of the form (2) defined by (A.26). The procedure leads to the well-known $\boldsymbol{LU}$ matrix factorization, where $\boldsymbol{L}$ is a lower triangular matrix and $\boldsymbol{U}$ is an upper triangular matrix. The following theorem states $\boldsymbol{LU}$ matrix factorization [20, 22, 24, 39].

Theorem 6 ([20], Chapter 1, p. 20, and Its General Form in [22], Chapter 2, p. 50) *On condition that the principal minors of the matrix*

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}$$

are not equal to zero, i.e., that $a_{11} \neq 0, \ldots, \det(\boldsymbol{A}) \neq 0$*, the matrix* $\boldsymbol{A}$ *may be represented as the product of a lower triangular matrix and an upper triangular matrix, that is*

$$\boldsymbol{A} = \boldsymbol{LU}.$$

□

Note: $\boldsymbol{LU}$ factorization of a matrix $\boldsymbol{A}$ will be uniquely determined if we prescribe values for the diagonal elements of one of the triangular matrices. It is convenient to consider, for example, that elements of $\boldsymbol{U}$ are equal to $u_{ii} = 1,\ i = 1, 2, \ldots, N$ [20, 22].

$\boldsymbol{LU}$ matrix factorization is actually originated from the method of Gaussian elimination used for solving systems of linear equations [20, 22, 24, 39]. One step of Gaussian elimination is equivalent to a pre-multiplication of the matrix by elementary matrix of the form (2) defined by (A.26) which actually is the unit lower triangular and it is called Gauss elementary matrix. Thus, the transition from original matrix $\boldsymbol{A}$ to its upper triangular form can be written as

$$\boldsymbol{W}_m\, \boldsymbol{W}_{m-1}\, \ldots\, \boldsymbol{W}_1\, \boldsymbol{A} = \boldsymbol{U}, \tag{A.32}$$

where $\boldsymbol{W}_i,\ i = 1, 2, \ldots, m$ are Gauss elementary matrices. Then, we have

$$\boldsymbol{A} = (\boldsymbol{W}_m\ \boldsymbol{W}_{m-1}\ \ldots\ \boldsymbol{W}_1)^{-1}\ \boldsymbol{U} = \boldsymbol{L}\boldsymbol{U}, \tag{A.33}$$

whereby this $\boldsymbol{LU}$ factorization is unique [24]. From the algebra of triangular matrices it follows that the matrix $\boldsymbol{L}$ is unit lower triangular because each $\boldsymbol{W}_i^{-1}$ is unit lower triangular and $\det(\boldsymbol{A}) = u_{11} \ldots u_{nn}$. Moreover, if $\det(\boldsymbol{A}) = +1$, then $\det(\boldsymbol{LU}) = \det(\boldsymbol{L}) \det(\boldsymbol{U}) = +1$, hence $\det(\boldsymbol{L}) = +1$ and $\det(\boldsymbol{U}) = +1$. Generally, if we take into account interchanges of two rows in the matrix during the factorization process (the so-called pivoting operation in Gaussian elimination), then we should consider in $\boldsymbol{LU}$ factorization pre-multiplied or post-multiplied permutation matrices $\boldsymbol{P}_i$.

In addition, according to the theorem ([22], Chapter 2, p. 53), an arbitrary matrix $\boldsymbol{A}$, whose all principal minors are not equal to zero, can be represented as the product of $\boldsymbol{A} = \boldsymbol{LDU}$, where $\boldsymbol{L}$ is an unit lower triangular matrix, $\boldsymbol{D}$ is a diagonal matrix and $\boldsymbol{U}$ is a unit upper triangular matrix.

A variant of Gaussian elimination is the Jordan elimination. Whereas the Gaussian elimination leads to an upper triangular matrix, the Jordan elimination leads to a diagonal matrix. One step of Jordan elimination is equivalent to a pre-multiplication of the matrix by

$$\begin{pmatrix} 1 & 0 & \ldots & \alpha_{1,i} & \ldots & 0 \\ 0 & 1 & \ldots & \alpha_{2,i} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \ldots & \alpha_{i-1,i} & \ldots & 0 \\ 0 & 0 & \ldots & 1 & \ldots & 0 \\ 0 & 0 & \ldots & \alpha_{i+1,i} & \ldots & 0 \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \alpha_{n,i} & \ldots & 1 \end{pmatrix}, \tag{A.34}$$

called the Gauss–Jordan elementary matrix. It can be easily verified that Gauss–Jordan elementary matrix is the product of elementary matrices of the form (2) defined by (A.26) and of form (3) defined by (A.27).

$\boldsymbol{LU}$ matrix factorization is the basic approach to factorize an invertible matrix into the product of triangular matrices and possibly permutation matrices taking into account row interchanges during the factorization process.

A.6.3 PLUS *Matrix Factorization*

In the special case, if a real nonsingular matrix $\boldsymbol{A}$ has its determinant equal to $+1$, i.e., $\det(\boldsymbol{A}) = +1$, then it can be formulated theorem for the general $\boldsymbol{PLUS}$ matrix factorization of $\boldsymbol{A}$ [27, 37] as follows.

Theorem 7 (*PLUS* Matrix Factorization [27, 37]) *A real square nonsingular matrix* $\boldsymbol{A}$ *has a factorization of* $\boldsymbol{A} = \boldsymbol{PLUS}$ *if and only if* $\det(\boldsymbol{A}) = \det(\boldsymbol{P}) = +1$, *where* $\boldsymbol{P}$ *is a permutation matrix,* $\boldsymbol{L}$ *is an unit lower triangular,* $\boldsymbol{U}$ *is an unit upper triangular, and* $\boldsymbol{S}$ *is an unit lower triangular matrix of the form*

$$\boldsymbol{S} = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ s_1 & s_2 & \dots & s_{n-1} & 1 \end{pmatrix}, \quad \textit{where} \quad \boldsymbol{S}^{-1} = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ -s_1 & -s_2 & \dots & -s_{n-1} & 1 \end{pmatrix}.$$

□

Note: In general, any real square nonsingular matrix $\boldsymbol{A}$ can be customarily factorized into three triangular matrices, $\boldsymbol{A} = \boldsymbol{PLUS}$ customizable factorization, where $\boldsymbol{P}$ is a permutation matrix (in some cases $\boldsymbol{P}$ may be the identity matrix), $\boldsymbol{U}$ is an upper triangular matrix of which the diagonal elements $d_1, d_2, \dots, d_N$ are customizable and they can be given by all means as long as its determinant is equal to that of $\boldsymbol{A}$ up to a possible sign adjustment, i.e., $\det(\boldsymbol{A}) = \det(\boldsymbol{U}) = d_1\ d_2\ \dots\ d_N$. $\boldsymbol{S}$ is a unit lower or upper triangular matrix of which all but $N-1$ off-diagonal elements are also flexibly customizable such as a single-row, single-column, bidiagonal matrix or other specially patterned matrices. Besides ***PLUS***, a customizable factorization also has other alternatives, ***LUSP***, ***PSUL*** and ***SULP*** for unit lower triangular $\boldsymbol{S}$, and ***PULS***, ***ULSP***, ***PSLU***, ***SLUP*** for unit upper triangular $\boldsymbol{S}$ [26].

We note that any nonsingular matrix $\boldsymbol{A}$ with $\det(\boldsymbol{A}) = -1$ can be scaled to have its $\det(\boldsymbol{A}) = +1$. More insight into the structure of ***PLUS*** factorization gives its factorization algorithm [27, 37]. Let $\boldsymbol{A}$ be a real nonsingular (invertible) matrix of order N

$$\boldsymbol{A} = \begin{pmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \dots & a_{1n}^{(0)} \\ a_{21}^{(0)} & a_{22}^{(0)} & \dots & a_{2n}^{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(0)} & a_{n2}^{(0)} & \dots & a_{nn}^{(0)} \end{pmatrix}.$$

Then, there must exist a permutation matrix $\boldsymbol{P}_1$ for row interchanges such that

$$\boldsymbol{P}_1\,\boldsymbol{A} = \begin{pmatrix} p_{11}^{(1)} & p_{12}^{(1)} & \cdots & p_{1n}^{(1)} \\ p_{21}^{(1)} & p_{22}^{(1)} & \cdots & p_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1}^{(1)} & p_{n2}^{(1)} & \cdots & p_{nn}^{(1)} \end{pmatrix},$$

and $p_{1n}^{(1)} \neq 0$, and hence there must exist a number s_1 such that $p_{11}^{(1)} - s_1\, p_{1n}^{(1)} = 1$. Then, we get $s_1 = \frac{p_{11}^{(1)}-1}{p_{1n}^{(1)}}$ and we obtain a product of

$$\boldsymbol{P}_1\, \boldsymbol{A}\, \boldsymbol{S}_{0,1} = \boldsymbol{P}_1\, \boldsymbol{A} \begin{pmatrix} 1 & 0 \ldots 0 \\ 0 & 1 \ldots 0 \\ \vdots & \vdots \ddots \vdots \\ -s_1 & 0 \ldots 1 \end{pmatrix} = \begin{pmatrix} 1 & p_{12}^{(1)} \ldots p_{1n}^{(1)} \\ p_{21}^{(1)} - s_1 p_{2n}^{(1)} & p_{22}^{(1)} \ldots p_{2n}^{(1)} \\ \vdots & \vdots \ddots \vdots \\ p_{n1}^{(1)} - s_1 p_{nn}^{(1)} & p_{n2}^{(1)} \ldots p_{nn}^{(1)} \end{pmatrix}.$$

The second step is Gaussian elimination of the first column and it is achieved by pre-multiplication of the product $\boldsymbol{P}_1\, \boldsymbol{A}\, \boldsymbol{S}_{0,1}$ by the Gauss elementary matrix $\boldsymbol{L}_1$ as follows

$$\boldsymbol{L}_1\, \boldsymbol{P}_1\, \boldsymbol{A}\, \boldsymbol{S}_{0,1} = \begin{pmatrix} 1 & 0 \ldots 0 \\ s_1 p_{2n}^{(1)} - p_{21}^{(1)} & 1 \ldots 0 \\ \vdots & \vdots \ddots \vdots \\ s_1 p_{nn}^{(1)} - p_{n1}^{(1)} & 0 \ldots 1 \end{pmatrix} \boldsymbol{P}_1\, \boldsymbol{A}\, \boldsymbol{S}_{0,1} = \begin{pmatrix} 1 & a_{12}^{(2)} \ldots a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} \ldots a_{2n}^{(2)} \\ \vdots & \vdots \ddots \vdots \\ 0 & a_{n2}^{(2)} \ldots a_{nn}^{(2)} \end{pmatrix}.$$

Continuing the factorization process for $k = 2, 3, \ldots, N-1$, where matrices $\boldsymbol{P}_k$ define the row interchanges among the kth through Nth rows to guarantee that the kth element in the N-th column are not equal to zero, i.e., $p_{kn}^{(k)} \neq 0$, matrices $S_{0,k}$ convert elements $a_{kk}^{(k)}$ into $1's$, where $s_k = \frac{p_{kk}^{(k)}-1}{p_{kn}^{(k)}}$, and matrices $\boldsymbol{L}_k$ represent row multipliers used for Gaussian elimination of column k. Completing the factorization process we get the product:

$$\boldsymbol{L}_{N-1}\boldsymbol{P}_{N-1} \ldots \boldsymbol{L}_2\boldsymbol{P}_2\boldsymbol{L}_1\boldsymbol{P}_1\boldsymbol{A}\boldsymbol{S}_{0,1}\boldsymbol{S}_{0,2} \ldots \boldsymbol{S}_{0,N-1} = \begin{pmatrix} 1 & a_{12}^{(N-1)} & \ldots & a_{1n}^{(N-1)} \\ 0 & 1 & \ldots & a_{2n}^{(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & a_{nn}^{(N-1)} \end{pmatrix} = \boldsymbol{U},$$

where $\boldsymbol{U}$ is the upper triangular. Since $\det(\boldsymbol{A}) = +1$, $a_{nn}^{(N-1)} = 1$ and $\boldsymbol{U}$ is the unit upper triangular. Having, respectively, multiplied all matrices $\boldsymbol{S}_{0,k}$ together, all permutation matrices $\boldsymbol{P}_k$ together, and all unit lower triangular matrices $\boldsymbol{L}_k$ together, we have one matrix $\boldsymbol{S}^{-1}$, one pre-multiplying matrix $\boldsymbol{P}^T$ and one unit lower triangular matrix $\boldsymbol{L}^{-1}$. From algebra of triangular matrices it follows that the inverse of a unit lower triangular matrix is also a unit lower triangular matrix and we have

$$\boldsymbol{S}_{0,1}\,\boldsymbol{S}_{0,2}\,\ldots\,\boldsymbol{S}_{0,N-1} = \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & 0 \\ -s_1 & -s_2 & \ldots & -s_{n-1} & 1 \end{pmatrix} = \boldsymbol{S}^{-1}$$

and

$$\begin{aligned} &\boldsymbol{L}_{N-1}\boldsymbol{P}_{N-1}\ldots\boldsymbol{L}_2\boldsymbol{P}_2\boldsymbol{L}_1\boldsymbol{P}_1 \\ &= \boldsymbol{L}_{N-1}\left(\boldsymbol{P}_{N-1}\boldsymbol{L}_{N-2}\boldsymbol{P}_{N-1}^T\right)\ldots\left(\boldsymbol{P}_{N-1}\ldots\boldsymbol{P}_2\boldsymbol{L}_1\boldsymbol{P}_2^T\ldots\,\boldsymbol{P}_{N-1}^T\right)(\boldsymbol{P}_{N-1}\ldots\boldsymbol{P}_2\boldsymbol{P}_1) \\ &= \boldsymbol{L}^{-1}\boldsymbol{P}^T, \end{aligned}$$

where

$$\boldsymbol{L}^{-1} = \boldsymbol{L}_{N-1}\left(\boldsymbol{P}_{N-1}\boldsymbol{L}_{N-2}\boldsymbol{P}_{N-1}^T\right)\ldots\left(\boldsymbol{P}_{N-1}\ldots\boldsymbol{P}_2\boldsymbol{L}_1\boldsymbol{P}_2^T\ldots\boldsymbol{P}_{N-1}^T\right)$$

and

$$\boldsymbol{P}^{-1} = \boldsymbol{P}^T = \boldsymbol{P}_{N-1}\ldots\boldsymbol{P}_2\boldsymbol{P}_1.$$

Hence, finally we obtain $\boldsymbol{L}^{-1}\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{S}^{-1} = \boldsymbol{U}$ or $\boldsymbol{A} = \boldsymbol{PLUS}$ matrix factorization.

A.7 Block Matrices and Algebra of Block Matrices

In linear algebra, matrix theory, and matrix computations the matrices with scalar elements are frequently partitioned into sub-matrices or blocks [20–22, 24, 39]. Such matrices are called block matrices. Although an $m \times n$ (non-square) matrix can be partitioned, in general, into rectangular (non-square) sub-matrices or blocks [22], pp. 55–64, for simplicity, we will consider square matrices only partitioned into square sub-matrices all being of the same order.

Let $\boldsymbol{A} = \{a_{ij}\}$, $i,\ j = 1, 2, \ldots, n$, be a square matrix with scalar elements. A partitioning of $\boldsymbol{A}$ is a representation of $\boldsymbol{A}$ in the form

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \ldots & \boldsymbol{A}_{1n} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \ldots & \boldsymbol{A}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{n1} & \boldsymbol{A}_{n2} & \ldots & \boldsymbol{A}_{nn} \end{pmatrix}, \tag{A.35}$$

where square sub-matrices $\boldsymbol{A}_{ij}$ are called blocks. Thus, the square matrix $\boldsymbol{A}$ in (A.35) is partitioned into $n \times n$ blocks, or equivalently, $\boldsymbol{A}$ is a square block matrix. The power of partitioning lies in the fact that the algebra of block matrices (sum and products) interacts nicely with the algebra of scalar matrices, as long as in general, the dimensions of partitions allow the indicated sums and products. Then, sums/products of block matrices are formed by treating the sub-matrices or blocks as scalars performing an ordinary addition/multiplication of sub-matrices. In this Appendix we will consider block matrices of the same order with the same partitioning. Such block matrices are called to be conformal.

Various forms of scalar matrices such as diagonal, (unit) upper and lower triangular have block analogues [22, 39]. A block matrix $\boldsymbol{A}$ is the quasi-diagonal, if $\boldsymbol{A}_{ij} = 0$ for $i \neq j$. A block matrix $\boldsymbol{A}$ is the upper block triangular or the upper quasi-triangular, if all blocks $\boldsymbol{A}_{ij} = 0$ for $i > j$, and is the lower block triangular or the lower quasi-triangular, if all blocks $\boldsymbol{A}_{ij} = 0$ for $i < j$. The quasi-diagonal matrix can be considered as a special form of the upper/lower block triangular matrix. In particular, an upper/lower block triangular matrix having on main diagonal the identity matrices then is the upper/lower block triangular with unit diagonal blocks.

A.7.1 Algebra of Block Triangular (Quasi-Diagonal) Matrices

There are a few useful properties about products, determinants, and inverses of block triangular (quasi-diagonal) matrices [22]:

- The product of two upper (lower) block triangular matrices is upper (lower) block triangular matrix.
- The product of two upper (lower) block triangular matrices with unit diagonal blocks is upper (lower) block triangular matrix with unit diagonal blocks.
- The determinant of upper or lower block triangular matrix, and in particular the determinant of quasi-diagonal matrix, is equal to the product of determinants of diagonal blocks (Laplace decomposition).
- The determinant of upper or lower block triangular matrix with unit diagonal blocks is equal to 1.
- The inverse of upper (lower) block triangular matrix is upper (lower) block triangular.
- The inverse of upper (lower) block triangular matrix with unit diagonal blocks is upper (lower) block triangular with unit diagonal blocks.

A.7.2 Block Elementary Transformations

Elementary transformations performed upon scalar matrices (see Appendix A.5) have also block analogues [22], specifically, for block matrices the operation (1) is equivalent to a pre-multiplication by the elementary block matrix

$$\begin{pmatrix} \boldsymbol{I} & & & & & \\ & \ddots & & & & \\ & & \boldsymbol{I} & & & \\ & & & \boldsymbol{X} & & \\ & & & & \boldsymbol{I} & \\ & & & & & \ddots \\ & & & & & & \boldsymbol{I} \end{pmatrix}, \tag{A.36}$$

operation (2) is equivalent to a pre-multiplication by the elementary block matrix

$$\begin{pmatrix} \boldsymbol{I} & & & & \\ & \ddots & & & \\ & & \boldsymbol{I} & & \\ & & \vdots & \ddots & \\ & & \boldsymbol{X} & \ldots & \boldsymbol{I} \\ & & & & & \ddots \\ & & & & & & \boldsymbol{I} \end{pmatrix}, \tag{A.37}$$

and the operation (3) is equivalent to a pre-multiplication by the elementary block matrix

$$\begin{pmatrix} \boldsymbol{I} & & & & \\ & \ddots & & & \\ & & \boldsymbol{I} & \ldots & \boldsymbol{X} \\ & & & \ddots & \vdots \\ & & & & \boldsymbol{I} \\ & & & & & \ddots \\ & & & & & & \boldsymbol{I} \end{pmatrix}. \tag{A.38}$$

where diagonal blocks $\boldsymbol{I}$ are identity matrices and $\boldsymbol{X}$ is a square sub-matrix or block, all of the same order. Operations (1), (2), and (3) can be performed alternatively upon the columns in post-multiplications of a block matrix.

It can be easily seen that the elementary block matrices are nonsingular. The elementary block matrix defined by (A.36) is quasi-diagonal and its determinant is equal to $\det(\boldsymbol{X})$. The elementary block matrix defined by (A.37) is the lower block triangular with unit diagonal blocks and its determinant is equal to 1. It is called the Gauss elementary block matrix. On the other hand, the elementary block matrix defined by (A.38) is the upper block triangular with unit diagonal blocks and its determinant is also equal to 1. Based on the algebra of block triangular

matrices, the inverses of elementary block matrices of the form (A.37) and (A.38) are, respectively, defined as

$$\begin{pmatrix} \boldsymbol{I} & & & & \\ & \ddots & & & \\ & & \boldsymbol{I} & & \\ & & \vdots & \ddots & \\ & & -\boldsymbol{X} & \dots & \boldsymbol{I} \\ & & & & & \ddots & \\ & & & & & & \boldsymbol{I} \end{pmatrix}, \quad \begin{pmatrix} \boldsymbol{I} & & & & \\ & \ddots & & & \\ & & \boldsymbol{I} & \dots & -\boldsymbol{X} \\ & & & \ddots & \vdots \\ & & & & \boldsymbol{I} \\ & & & & & \ddots & \\ & & & & & & \boldsymbol{I} \end{pmatrix}. \tag{A.39}$$

A.7.3 Generalized Gauss Algorithm or Block Gaussian Elimination

Let $\boldsymbol{A} = \{\boldsymbol{A}_{ij}\}$, $i,\ j = 1, 2, \dots, n$, defined by (A.35) be a square block matrix whose diagonal block $\boldsymbol{A}_{11}$ is a square nonsingular sub-matrix, i.e., $\det(\boldsymbol{A}_{11}) \neq 0$.

Pre-multiplying the first row of $\boldsymbol{A}$ by $-\boldsymbol{A}_{i1}\,\boldsymbol{A}_{11}^{-1}$, $i = 2, 3, \dots, n$, and subsequently adding the product to the ith row of $\boldsymbol{A}$ we get the following block matrix

$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \dots & \boldsymbol{A}_{1n} \\ \boldsymbol{0} & \boldsymbol{A}_{22}^{(1)} & \dots & \boldsymbol{A}_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{A}_{n2}^{(1)} & \dots & \boldsymbol{A}_{nn}^{(1)} \end{pmatrix}, \tag{A.40}$$

where

$$\boldsymbol{A}_{ij}^{(1)} = -\boldsymbol{A}_{i1}\,\boldsymbol{A}_{11}^{-1}\,\boldsymbol{A}_{1j} + \boldsymbol{A}_{ij}, \qquad i,\ j = 2, 3, \dots, n, \tag{A.41}$$

where $\boldsymbol{0}$s are null blocks. The transition from the block matrix $\boldsymbol{A}$ to the block matrix $\boldsymbol{B}$ in (A.40) is equivalent to the successive pre-multiplications (multiplications on the left) of $\boldsymbol{A}$ by the Gauss elementary block matrices defined by (A.37), where $\boldsymbol{X} = -\boldsymbol{A}_{i1}\,\boldsymbol{A}_{11}^{-1}$, $i = 2, 3, \dots, n$.

If the block $\boldsymbol{A}_{22}^{(1)}$ is again a square nonsingular sub-matrix, i.e., $\det\left(\boldsymbol{A}_{22}^{(1)}\right) \neq 0$, then the above process can be repeated to eliminate blocks in the second column of $\boldsymbol{B}$ under block $\boldsymbol{A}_{22}^{(1)}$. After $n-1$ steps the block matrix $\boldsymbol{A}$ is reduced to an upper block triangular form. Just described procedure defines the generalized Gauss algorithm for block matrices (block Gaussian elimination). Since the block matrix $\boldsymbol{A}$ is reduced to the upper block triangular form, then its determinant is equal to the determinant of the upper block triangular matrix.

A.7.4 Schur Formulae for the Determinant Calculation of a 2 × 2 Block Matrix

Consider the simplest case of 2×2 block matrix $\boldsymbol{M}$ partitioned to four square blocks of the same order n as

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix},$$

and let $\det(\boldsymbol{A}) \neq 0$, i.e., $\boldsymbol{A}$ is nonsingular. Our aim is to derive the so-called Schur formulae for the determinant calculation of block matrix $\boldsymbol{M}$ of order $2n$ via the determinants of order n [22].

Apply to the block matrix $\boldsymbol{M}$ the generalized Gauss algorithm. Pre-multiplying the first row of $\boldsymbol{M}$ by $-\boldsymbol{C}\boldsymbol{A}^{-1}$, and adding the product to the second row of $\boldsymbol{M}$ corresponds to pre-multiplication of $\boldsymbol{M}$ by the Gauss elementary block matrix defined by (A.37), where $\boldsymbol{X} = -\boldsymbol{C}\boldsymbol{A}^{-1}$, and we have

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{I} \end{pmatrix} \boldsymbol{M} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B} \end{pmatrix}. \tag{A.42}$$

Using the algebra of block triangular matrices from Eq. (A.42) we obtain

$$\det(\boldsymbol{M}) = \det(\boldsymbol{A})\ \det(\boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}) = \det(\boldsymbol{A}\boldsymbol{D} - \boldsymbol{A}\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}), \quad \det(\boldsymbol{A}) \neq 0. \tag{A.43}$$

Now let $\det(\boldsymbol{D}) \neq 0$, i.e., $\boldsymbol{D}$ is nonsingular and apply to the block matrix $\boldsymbol{M}$ the generalized Gauss algorithm again. Pre-multiplying the second row of $\boldsymbol{M}$ by $-\boldsymbol{B}\boldsymbol{D}^{-1}$, and adding the product to the first row of $\boldsymbol{M}$ corresponds to pre-multiplication of $\boldsymbol{M}$ by the Gauss elementary block matrix defined by (A.38), where $\boldsymbol{X} = -\boldsymbol{B}\boldsymbol{D}^{-1}$, we have

$$\begin{pmatrix} \boldsymbol{I} & -\boldsymbol{B}\boldsymbol{D}^{-1} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \boldsymbol{M} = \begin{pmatrix} \boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix}. \tag{A.44}$$

Similarly, using the algebra of block triangular matrices from Eq. (A.44) we obtain

$$\det(\boldsymbol{M}) = \det(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C})\ \det(\boldsymbol{D}) = \det(\boldsymbol{A}\boldsymbol{D} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{D}), \quad \det(\boldsymbol{D}) \neq 0. \tag{A.45}$$

Equations (A.43) and (A.45) define the Schur formulae for the determinant calculation of the block matrix $\boldsymbol{M}$ of order $2n$ via the determinants of order n. In particular, if the blocks $\boldsymbol{A}$ and $\boldsymbol{C}$ are commuting, i.e., $\boldsymbol{A}\boldsymbol{C} = \boldsymbol{C}\boldsymbol{A}$, as well as the blocks $\boldsymbol{C}$ and $\boldsymbol{D}$ are commuting, i.e., $\boldsymbol{C}\boldsymbol{D} = \boldsymbol{D}\boldsymbol{C}$, then the determinants of $\boldsymbol{M}$ are equal to [22]

$$\begin{aligned} &\det(\boldsymbol{M}) = \det(\boldsymbol{A}\boldsymbol{D} - \boldsymbol{C}\boldsymbol{B}), \quad \det(\boldsymbol{A}) \neq 0, \\ &\det(\boldsymbol{M}) = \det(\boldsymbol{A}\boldsymbol{D} - \boldsymbol{B}\boldsymbol{C}), \quad \det(\boldsymbol{D}) \neq 0. \end{aligned} \tag{A.46}$$

We can obtain additional six formulae by exchanging the roles of $\boldsymbol{A}$ and $\boldsymbol{D}$, and then the roles of $\boldsymbol{B}$ and $\boldsymbol{C}$ on the right-hand sides of (A.43), (A.45), and (A.46).

A.7.5 Frobenius Formula for the Inverse of a 2 × 2 Block Matrix

Consider a 2×2 block nonsingular matrix partitioned to four square blocks of the same order n as

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix},$$

and let $\det(\boldsymbol{A}) \neq 0$, i.e., $\boldsymbol{A}$ is nonsingular. We need to derive the inverse of $\boldsymbol{M}$, $\boldsymbol{M}^{-1}$.

In the first step we apply to the block matrix $\boldsymbol{M}$ the generalized Gauss algorithm pre-multiplying the first row of $\boldsymbol{M}$ by $-\boldsymbol{C}\boldsymbol{A}^{-1}$, and adding the product to the second row of $\boldsymbol{M}$. This operation corresponds to the pre-multiplication of $\boldsymbol{M}$ by the Gauss elementary block matrix as follows

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{I} \end{pmatrix} \boldsymbol{M} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix}, \tag{A.47}$$

where $\boldsymbol{S} = \boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}$. Since $\boldsymbol{M}$ is nonsingular, $\det(\boldsymbol{M}) = \det(\boldsymbol{A})\det(\boldsymbol{S}) \neq 0$, then $\det(\boldsymbol{S}) \neq 0$. Inverting Eq. (A.47) we get

$$\boldsymbol{M}^{-1} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{I} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix}^{-1}. \tag{A.48}$$

Now, the inverse block matrix on the right-hand side of (A.48) we will look for in the form

$$\begin{pmatrix} \boldsymbol{A}^{-1} & \boldsymbol{U} \\ \boldsymbol{0} & \boldsymbol{S}^{-1} \end{pmatrix}.$$

From the equality

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix} \begin{pmatrix} \boldsymbol{A}^{-1} & \boldsymbol{U} \\ \boldsymbol{0} & \boldsymbol{S}^{-1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix},$$

we find that $\boldsymbol{U} = -\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{S}^{-1}$, and hence

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{S}^{-1} \\ \boldsymbol{0} & \boldsymbol{S}^{-1} \end{pmatrix}. \tag{A.49}$$

Then from Eq. (A.48) we obtain

$$\boldsymbol{M}^{-1} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{I} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{S}^{-1} \\ \boldsymbol{0} & \boldsymbol{S}^{-1} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{I} \end{pmatrix}. \tag{A.50}$$

Finally, performing all products of block matrices on the right-hand side of (A.50) we get

$$\boldsymbol{M}^{-1} = \begin{pmatrix} \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{S}^{-1}\boldsymbol{C}\boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{S}^{-1} \\ -\boldsymbol{S}^{-1}\boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{S}^{-1} \end{pmatrix}, \qquad \boldsymbol{S} = \boldsymbol{D} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B}. \tag{A.51}$$

Equation (A.51) defines the Frobenius formula to obtain the inverse of block matrix $\boldsymbol{M}$. Alternatively, assuming that $\det(\boldsymbol{D}) \neq 0$ (instead of $\det(\boldsymbol{A}) \neq 0$), and exchanging the roles of blocks $\boldsymbol{A}$ and $\boldsymbol{D}$, we obtain the second form of Frobenius formula as

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{K}^{-1} & -\boldsymbol{K}^{-1}\boldsymbol{B}\boldsymbol{D}^{-1} \\ -\boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{K}^{-1} & \boldsymbol{D}^{-1} + \boldsymbol{D}^{-1}\boldsymbol{C}\boldsymbol{K}^{-1}\boldsymbol{B}\boldsymbol{D}^{-1} \end{pmatrix}, \quad \boldsymbol{K} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C}. \tag{A.52}$$

Additionally, the derivation of Frobenius formulae also nicely demonstrates how to obtain the formulae for inverses of 2×2 upper and lower block triangular matrices. The inverse of an upper block triangular matrix is defined by (A.49). The inverse of a lower block triangular matrix can be obtained as follows. From equality

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{B} & \boldsymbol{S} \end{pmatrix} \begin{pmatrix} \boldsymbol{A}^{-1} & \boldsymbol{0} \\ \boldsymbol{U} & \boldsymbol{S}^{-1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix},$$

we find that $\boldsymbol{U} = -\boldsymbol{S}^{-1}\boldsymbol{B}\boldsymbol{A}^{-1}$, and hence

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{B} & \boldsymbol{S} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{A}^{-1} & \boldsymbol{0} \\ -\boldsymbol{S}^{-1}\boldsymbol{B}\boldsymbol{A}^{-1} & \boldsymbol{S}^{-1} \end{pmatrix}. \tag{A.53}$$

In particular, Eqs. (A.49) and (A.53) imply ($\boldsymbol{A} = \boldsymbol{S} = \boldsymbol{I}$) that the inverses of 2×2 upper and lower block triangular matrices with unit diagonal blocks are, respectively, defined as

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{X} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{X} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix}, \qquad \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{X} & \boldsymbol{I} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{X} & \boldsymbol{I} \end{pmatrix}. \tag{A.54}$$

A.8 Block Matrix Decompositions

The block matrix decompositions are factorizations of block matrices into structurally simpler block matrices [20, 21, 27, 37, 39]. Similarly as for scalar matrices, in linear algebra and matrix computations a real nonsingular block matrix is reduced by elementary block transformations into various canonical block forms in order to simplify subsequent computational steps of a solved problem. Actually, the block ***QR***, ***LU***, ***LDU***, and ***PLUS*** matrix factorizations can be considered as generalized analogues of those formulated for scalar matrices. The block matrix factorization ***QR*** is discussed in [39], Chapter 4, p. 255.

A.8.1 *Schur Complement and Block* LU *Matrix Factorization*

Schur complement plays an important role in matrix computations and leads to a block $\boldsymbol{LU}$ matrix factorization [21, 39]. It has been heavily discussed in the previous Appendix A.7, specifically in the generalized Gauss algorithm, the determinant calculation and inverse of a 2×2 block matrix. In the following is generalized the notion of Schur complement.

Definition (Schur Complement, [39], Chapter 3, p. 157) Let $\boldsymbol{A}$ be partitioned in the form

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} \, \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} \, \boldsymbol{A}_{22} \end{pmatrix},$$

and suppose that $\boldsymbol{A}_{11}$ is nonsingular. Then the Schur complement of $\boldsymbol{A}_{11}$ in $\boldsymbol{A}$ is the matrix $\boldsymbol{S} = \boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}$.

□

The following theorem states the block $\boldsymbol{LU}$ matrix factorization of $\boldsymbol{A}$ [39].

Theorem 8 (Block *LU* Matrix Factorization, [39], Chapter 3, p. 158) *Let*

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} \, \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} \, \boldsymbol{A}_{22} \end{pmatrix},$$

where $\boldsymbol{A}_{11}$ is nonsingular, and let $\boldsymbol{S}$ be the Schur complement of $\boldsymbol{A}_{11}$ in $\boldsymbol{A}$. Then $\boldsymbol{A}$ is nonsingular if and only if $\boldsymbol{S}$ is nonsingular.

□

It is easy verified (see Eq. (A.42)) that $\boldsymbol{A}$ has a block $\boldsymbol{LU}$ matrix factorization of the form

$$\begin{pmatrix} \boldsymbol{A}_{11} \, \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} \, \boldsymbol{A}_{22} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{A}_{11} \, \boldsymbol{A}_{12} \\ \boldsymbol{0} \quad \boldsymbol{S} \end{pmatrix}. \tag{A.55}$$

The first factored matrix on the right-hand side of (A.55) is nonsingular, because it is the lower block triangular with unit diagonal blocks. Hence, $\boldsymbol{A}$ is nonsingular if and only if the second factored matrix is nonsingular. This factored matrix is the upper block triangular and is nonsingular if and only if its diagonal blocks are nonsingular. But by hypothesis the diagonal block $\boldsymbol{A}_{11}$ is nonsingular. Therefore, $\boldsymbol{A}$ is nonsingular if and only if the second diagonal block $\boldsymbol{S}$, Schur complement, is nonsingular [39]. Equation (A.55) defines the block $\boldsymbol{LU}$ matrix factorization of $\boldsymbol{A}$. If $\boldsymbol{A}$ is nonsingular and has a block $\boldsymbol{LU}$ matrix factorization, then this factorization is unique.

It is clear that the generalized Gauss algorithm generates a block $\boldsymbol{LU}$ matrix factorization for block matrices. Thus, the generalized Gauss algorithm or block Gaussian elimination provides the block $\boldsymbol{LU}$ factorization algorithm of a block matrix.

Note: A slightly different block $\boldsymbol{LU}$ matrix factorization of a 2×2 block matrix $\boldsymbol{A}$ whose determinant is equal to 1 has been presented in [27] in the following form

$$\begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1} & \boldsymbol{S} \end{pmatrix} \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix},$$

where $\det(\boldsymbol{A}_{11}) = 1$.

Note: From historical point of view the first block $\boldsymbol{LU}$ matrix factorization is due to Schur who wrote it in the following form [39]

$$\begin{pmatrix} \boldsymbol{A}_{11}^{-1} & \boldsymbol{0} \\ -\boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12} \end{pmatrix}.$$

Schur used the relation only to prove a theorem on determinants calculation (see Appendix A.7.4) and he did not investigate it further.

A.8.2 Properties of Schur Complements

Schur complements have important properties which do not depend on the choice of diagonal blocks in the block $\boldsymbol{LU}$ matrix factorization. They are summarized in the following two theorems.

Theorem 9 ([39], Chapter 3, p. 165) *In the partitioning*

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix},$$

suppose that $\boldsymbol{A}_{11}$ *is nonsingular. Then* $\boldsymbol{A}$ *has a block* $\boldsymbol{LU}$ *factorization*

$$\begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix} = \begin{pmatrix} \boldsymbol{L}_{11} & \boldsymbol{0} \\ \boldsymbol{L}_{21} & \boldsymbol{L}_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{U}_{11} & \boldsymbol{U}_{12} \\ \boldsymbol{0} & \boldsymbol{U}_{22} \end{pmatrix}, \tag{A.56}$$

where $\boldsymbol{L}_{11}$ *and* $\boldsymbol{U}_{11}$ *are nonsingular. Moreover, for any such factorization*

1. $\boldsymbol{L}_{11}\boldsymbol{U}_{11} = \boldsymbol{A}_{11}$,
2. $\boldsymbol{L}_{21}\boldsymbol{U}_{11} = \boldsymbol{A}_{21}$,
3. $\boldsymbol{L}_{11}\boldsymbol{U}_{12} = \boldsymbol{A}_{12}$,
4. $\boldsymbol{L}_{22}\boldsymbol{U}_{22} = \boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12} = \boldsymbol{A}_{22} - \boldsymbol{L}_{21}\boldsymbol{U}_{12}$,
5. $\boldsymbol{U}_{11}^{-1}\boldsymbol{U}_{12} = \boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}$,
6. $\boldsymbol{L}_{21}\boldsymbol{L}_{11}^{-1} = \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}$.

If in addition $\boldsymbol{A}$ *is nonsingular, then so are* $\boldsymbol{L}_{22}$, $\boldsymbol{U}_{22}$, *and hence the Schur complement* $\boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12} = \boldsymbol{L}_{22}\boldsymbol{U}_{22}$. *If we partition*

$$\boldsymbol{A}^{-1} = \begin{pmatrix} \boldsymbol{A}_{11}^{-1} & \boldsymbol{A}_{12}^{-1} \\ \boldsymbol{A}_{21}^{-1} & \boldsymbol{A}_{22}^{-1} \end{pmatrix},$$

then

$$\boldsymbol{A}_{22}^{-1} = \boldsymbol{U}_{22}^{-1}\boldsymbol{L}_{22}^{-1} = (\boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12})^{-1}.$$

□

Theorem shows that the Schur complement computed by a sequence of k steps of classical Gaussian elimination is the same as the Schur complement of the leading principal minor of order k [39].

The following theorem shows that any two sequences of scalar and block elimination that terminate at the same leading principal sub-matrix compute the same Schur complement. The proof of theorem can be found in [39].

Theorem 10 ([39], Chapter 3, p. 166) *Let* $\boldsymbol{A}$ *be partitioned in the form*

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \boldsymbol{A}_{13} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \boldsymbol{A}_{23} \\ \boldsymbol{A}_{31} & \boldsymbol{A}_{32} & \boldsymbol{A}_{33} \end{pmatrix},$$

and assume that

$$\boldsymbol{A}_{11} \quad \textit{and} \quad \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix} \quad \textit{are nonsingular.}$$

Let

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{S}_{11} & \boldsymbol{S}_{12} \\ \boldsymbol{S}_{21} & \boldsymbol{S}_{22} \end{pmatrix}$$

be the Schur complement of $\boldsymbol{A}_{11}$ *in* $\boldsymbol{A}$. *Then* $\boldsymbol{S}_{11}$ *is nonsingular, and the Schur complement of*

$$\begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix}$$

in $\boldsymbol{A}$ *is equal to the Schur complement of* $\boldsymbol{S}_{11}$ *in* $\boldsymbol{S}$.

A.8.3 Block LDU Matrix Factorization

Having defined the Schur complement, its properties, and the block ***LU*** factorization, the following theorem states the existence and uniqueness of the block ***LDU*** matrix factorization, where $\boldsymbol{D}$ is a quasi-diagonal matrix [21].

Theorem 11 (Block *LDU* Matrix Factorization [21], p. 29) *Let* $\boldsymbol{A} = \{\boldsymbol{A}_{ij}\}$, $i, j = 1, 2, \ldots, r$, *is a block matrix whose blocks*

$$\begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \ldots & \boldsymbol{A}_{1k} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \ldots & \boldsymbol{A}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{k1} & \boldsymbol{A}_{k2} & \ldots & \boldsymbol{A}_{kk} \end{pmatrix}$$

are nonsingular for $k = 1, 2, \ldots, r-1$. *Then there exist a lower block triangular matrix* $\boldsymbol{L}$ *with* r *unit diagonal blocks, an upper block diagonal matrix* $\boldsymbol{U}$ *with* r *unit diagonal blocks, and a quasi-diagonal matrix* $\boldsymbol{D}$ *such that*

$$\boldsymbol{A} = \boldsymbol{L}\,\boldsymbol{D}\,\boldsymbol{U},$$

where the block matrices $\boldsymbol{L}$, $\boldsymbol{U}$ *and* $\boldsymbol{D}$ *are defined uniquely. The diagonal blocks* $\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots, \boldsymbol{D}_r$ *of* $\boldsymbol{D}$ *satisfy the following relations:*

$$\boldsymbol{D}_1 = \boldsymbol{A}_{11},$$

$$\boldsymbol{D}_k = \left[\begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \ldots & \boldsymbol{A}_{1k} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \ldots & \boldsymbol{A}_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{k1} & \boldsymbol{A}_{k2} & \ldots & \boldsymbol{A}_{kk} \end{pmatrix} / \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \ldots & \boldsymbol{A}_{1,k-1} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \ldots & \boldsymbol{A}_{2,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{k-1,1} & \boldsymbol{A}_{k-1,2} & \ldots & \boldsymbol{A}_{k-1,k-1} \end{pmatrix} \right], \quad k = 2, 3, \ldots, r,$$

and diagonal blocks $\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots, \boldsymbol{D}_{r-1}$ *are nonsingular.* $\boldsymbol{D}_k = [(*)/(*)]$ *for* $k = 2, 3, \ldots, r$, *are the Shur complements.*

□

Although Theorem 11 does not give a factorization algorithm, in order to illustrate the existence of block ***LDU*** factorization, consider the simplest case of a nonsingular 2×2 block matrix $\boldsymbol{A}$ (i.e., $k = 2$ in Theorem 11) partitioned as

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix}.$$

Since $\boldsymbol{A}$ is nonsingular, based on (A.55) it has the unique block $\boldsymbol{LU}$ factorization which is actually obtained by pre-multiplying $\boldsymbol{A}$ (multiplication on the left) by the Gauss elementary block matrix being the upper block triangular with unit diagonal blocks as

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix}, \tag{A.57}$$

where $\boldsymbol{S} = \boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}$ is the Schur complement of $\boldsymbol{A}_{11}$ in $\boldsymbol{A}$. The factored matrix on the right-hand side of (A.57) is the upper block diagonal with nonsingular diagonal blocks $\boldsymbol{A}_{11}$ and $\boldsymbol{S}$. It can be further reduced to a quasi-diagonal form by post-multiplying (multiplication on the right) by the elementary block matrix as

$$\begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix},$$

and equivalently, we have

$$\begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix}. \tag{A.58}$$

Composing Eqs. (A.57) and (A.58) we have

$$\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix}. \tag{A.59}$$

Finally, by pre-multiplying the lower block triangular matrix with unit diagonal blocks by its inverse followed by post-multiplying the upper block triangular matrix with unit diagonal blocks by its inverse on both sides of (A.59) we get

$$\begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix}. \tag{A.60}$$

Equation (A.60) defines the block $\boldsymbol{LDU}$ factorization of the 2×2 block matrix $\boldsymbol{A}$. According to Theorem 11, the blocks $\boldsymbol{D}_1 = \boldsymbol{A}_{11}$ and $\boldsymbol{D}_2 = \boldsymbol{S}$.

Note: Now, consider the block $\boldsymbol{LU}$ matrix factorization of a nonsingular 2×2 block matrix $\boldsymbol{A}$ defined by (A.55), and in particular, the upper block triangular matrix on the right-hand side of (A.55)

$$\begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix}.$$

It can be alternatively reduced to a quasi-diagonal form as follows: By pre-multiplying the second row by $-\boldsymbol{A}_{12}\boldsymbol{S}^{-1}$, and adding the product to the first row corresponds to the pre-multiplication by the Gauss elementary block matrix and we have

$$\begin{pmatrix} \boldsymbol{I} & -\boldsymbol{A}_{12}\boldsymbol{S}^{-1} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix},$$

or equivalently

$$\begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{A}_{12}\boldsymbol{S}^{-1} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix}. \tag{A.61}$$

Finally, substituting Eq. (A.61) into (A.55) we get

$$\begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{A}_{12}\boldsymbol{S}^{-1} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S} \end{pmatrix}. \tag{A.62}$$

Equation (A.62) defines a modification of the block ***LDU***, the block ***LUD*** factorization of the 2×2 block matrix $\boldsymbol{A}$.

A.8.4 Block PLUS Matrix Factorization

The ***PLUS*** matrix factorization has also a block analogue. For any nonsingular block matrix $\boldsymbol{A}$ with square diagonal blocks, in the block version of factorization $\boldsymbol{A} = \boldsymbol{PLUS}$, $\boldsymbol{P}$ is still a permutation matrix at the scalar level (in some cases $\boldsymbol{P}$ may be the identity matrix) while $\boldsymbol{L}$ is a lower block triangular matrix with unit diagonal blocks, $\boldsymbol{U}$ is an upper block triangular matrix, and $\boldsymbol{S}$ is a lower block triangular matrix with unit diagonal blocks [37].

More insight into the structure of block ***PLUS*** factorization gives its factorization algorithm. Let $\boldsymbol{A}$ be a nonsingular square block matrix partitioned to $n \times n$ square blocks as

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11}^{(0)} & \boldsymbol{A}_{12}^{(0)} & \dots & \boldsymbol{A}_{1n}^{(0)} \\ \boldsymbol{A}_{21}^{(0)} & \boldsymbol{A}_{22}^{(0)} & \dots & \boldsymbol{A}_{2n}^{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{n1}^{(0)} & \boldsymbol{A}_{n2}^{(0)} & \dots & \boldsymbol{A}_{nn}^{(0)} \end{pmatrix}.$$

Then must exist a permutation matrix $\boldsymbol{P}_1$ for row interchanges such that

$$\boldsymbol{P}_1\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11}^{(1)} & \boldsymbol{A}_{12}^{(1)} & \dots & \boldsymbol{A}_{1n}^{(1)} \\ \boldsymbol{A}_{21}^{(1)} & \boldsymbol{A}_{22}^{(1)} & \dots & \boldsymbol{A}_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{n1}^{(1)} & \boldsymbol{A}_{n2}^{(1)} & \dots & \boldsymbol{A}_{nn}^{(1)} \end{pmatrix},$$

and the block $\boldsymbol{A}_{1n}^{(1)}$ is nonsingular. In the first step, there must exist a nonsingular matrix $\boldsymbol{H}_1$ such that $\boldsymbol{A}_{11}^{(1)} - \boldsymbol{H}_1\boldsymbol{A}_{1n}^{(1)} = \boldsymbol{I}$. Then, we get $\boldsymbol{H}_1 = \left[\boldsymbol{A}_{1n}^{(1)}\right]^{-1} - \boldsymbol{A}_{11}^{(1)}\left[\boldsymbol{A}_{1n}^{(1)}\right]^{-1}$ and obtain a product of

$$\boldsymbol{P}_1\boldsymbol{A}\boldsymbol{S}_1 = \boldsymbol{P}_1\boldsymbol{A}\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0}\ldots\boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}\ldots\boldsymbol{0} \\ \vdots & \vdots\ddots\vdots \\ -\boldsymbol{H}_1 & \boldsymbol{0}\ldots\boldsymbol{I} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{A}_{12}^{(1)}\ldots\boldsymbol{A}_{1n}^{(1)} \\ \boldsymbol{A}_{21}^{(1)} - \boldsymbol{H}_1\boldsymbol{A}_{2n}^{(1)} & \boldsymbol{A}_{22}^{(1)}\ldots\boldsymbol{A}_{2n}^{(1)} \\ \vdots & \vdots\ddots\vdots \\ \boldsymbol{A}_{n1}^{(1)} - \boldsymbol{H}_1\boldsymbol{A}_{nn}^{(1)} & \boldsymbol{A}_{n2}^{(1)}\ldots\boldsymbol{A}_{nn}^{(1)} \end{pmatrix}.$$

The second step is the block Gaussian elimination of the first column by pre-multiplication of the product $\boldsymbol{P}_1\boldsymbol{A}\boldsymbol{S}_1$ by the Gauss block elementary matrix $\boldsymbol{L}_1$ as follows:

$$\boldsymbol{L}_1\boldsymbol{P}_1\boldsymbol{A}\boldsymbol{S}_1 = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0}\ldots\boldsymbol{0} \\ \boldsymbol{H}_1\boldsymbol{A}_{2n}^{(1)} - \boldsymbol{A}_{21}^{(1)} & \boldsymbol{I}\ldots\boldsymbol{0} \\ \vdots & \vdots\ddots\vdots \\ \boldsymbol{H}_1\boldsymbol{A}_{nn}^{(1)} - \boldsymbol{A}_{n1}^{(1)} & \boldsymbol{0}\ldots\boldsymbol{I} \end{pmatrix}\boldsymbol{P}_1\boldsymbol{A}\boldsymbol{S}_1 = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{A}_{12}^{(2)}\ldots\boldsymbol{A}_{1n}^{(2)} \\ \boldsymbol{0} & \boldsymbol{A}_{22}^{(2)}\ldots\boldsymbol{A}_{2n}^{(2)} \\ \vdots & \vdots\ddots\vdots \\ \boldsymbol{0} & \boldsymbol{A}_{n2}^{(2)}\ldots\boldsymbol{A}_{nn}^{(2)} \end{pmatrix}.$$

Continuing the factorization process for $k = 2, 3, \ldots, n-1$, where matrices $\boldsymbol{P}_k$ define the row interchanges among the kth through nth rows to guarantee that the block $\boldsymbol{A}_{kn}^{(k)}$ is nonsingular, matrices $\boldsymbol{S}_k$ convert diagonal blocks $\boldsymbol{A}_{kk}^{(k)}$ to the identity matrix, where $\boldsymbol{H}_k = \left[\boldsymbol{A}_{kn}^{(k)}\right]^{-1} - \boldsymbol{A}_{kk}^{(k)}\left[\boldsymbol{A}_{kn}^{(k)}\right]^{-1}$, and matrices $\boldsymbol{L}_k$ represent row multipliers used for block Gaussian elimination of column k. Completing the factorization process we get the product:

$$\boldsymbol{L}_{n-1}\boldsymbol{P}_{n-1}\ldots\boldsymbol{L}_2\boldsymbol{P}_2\boldsymbol{L}_1\boldsymbol{P}_1\boldsymbol{A}\boldsymbol{S}_1\boldsymbol{S}_2\ldots\boldsymbol{S}_{n-1} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{A}_{12}^{(n-1)} & \ldots & \boldsymbol{A}_{1n}^{(n-1)} \\ \boldsymbol{0} & \boldsymbol{I} & \ldots & \boldsymbol{A}_{2n}^{(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \ldots & \boldsymbol{A}_{nn}^{(n-1)} \end{pmatrix} = \boldsymbol{U},$$

where $\boldsymbol{U}$ is the upper block triangular. If $\det(\boldsymbol{A}) = 1$, then $\det(\boldsymbol{A}_{nn}^{(n-1)}) = 1$. Having, respectively, multiplied all matrices $\boldsymbol{S}_k$ together, all permutation matrices $\boldsymbol{P}_k$ together, and all $\boldsymbol{L}_k$ being the lower block triangular matrices with unit diagonal blocks together, we have one matrix $\boldsymbol{S}^{-1}$, one pre-multiplying matrix $\boldsymbol{P}^T$, and one lower block triangular matrix $\boldsymbol{L}^{-1}$ with unit diagonal blocks. From algebra of block triangular matrices it follows that the inverse of a lower block triangular matrix with unit diagonal blocks is also a lower block triangular matrix with unit diagonal blocks and we have

$$\boldsymbol{S}_1\,\boldsymbol{S}_2\,\ldots\,\boldsymbol{S}_{n-1} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} & \ldots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \ldots & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \ldots & \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{H}_1 & -\boldsymbol{H}_2 & \ldots & -\boldsymbol{H}_{n-1} & \boldsymbol{I} \end{pmatrix} = \boldsymbol{S}^{-1}$$

and

$$\begin{aligned} &\boldsymbol{L}_{n-1}\boldsymbol{P}_{n-1}\ldots\boldsymbol{L}_2\boldsymbol{P}_2\boldsymbol{L}_1\boldsymbol{P}_1 \\ &= \boldsymbol{L}_{n-1}\left(\boldsymbol{P}_{n-1}\boldsymbol{L}_{n-2}\boldsymbol{P}_{n-1}^T\right)\ldots\left(\boldsymbol{P}_{n-1}\ \ldots\boldsymbol{P}_2\boldsymbol{L}_1\boldsymbol{P}_2^T\ldots\boldsymbol{P}_{n-1}^T\right)(\boldsymbol{P}_{n-1}\ldots\boldsymbol{P}_2\boldsymbol{P}_1) \\ &= \boldsymbol{L}^{-1}\boldsymbol{P}^T, \end{aligned}$$

where

$$\boldsymbol{L}^{-1} = \boldsymbol{L}_{n-1}\left(\boldsymbol{P}_{n-1}\boldsymbol{L}_{n-2}\boldsymbol{P}_{n-1}^T\right)\ldots\left(\boldsymbol{P}_{n-1}\ldots\boldsymbol{P}_2\boldsymbol{L}_1\boldsymbol{P}_2^T\ldots\boldsymbol{P}_{n-1}^T\right)$$

and

$$\boldsymbol{P}^{-1} = \boldsymbol{P}^T = \boldsymbol{P}_{N-1}\ldots\boldsymbol{P}_2\boldsymbol{P}_1.$$

Hence, finally we obtain $\boldsymbol{L}^{-1}\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{S}^{-1} = \boldsymbol{U}$ or $\boldsymbol{A} = \boldsymbol{PLUS}$ block factorization.

In order to illustrate the existence of block $\boldsymbol{PLUS}$ factorization, consider the simplest case of a nonsingular 2×2 block matrix $\boldsymbol{M}$ partitioned to square blocks as

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix},$$

where the block $\boldsymbol{B}$ is nonsingular and $\boldsymbol{P} = \boldsymbol{I}$.

According to the $\boldsymbol{PLUS}$ factorization algorithm must exist a nonsingular block $\boldsymbol{H}$ such that $\boldsymbol{A} - \boldsymbol{BH} = \boldsymbol{I}$. Then we get $\boldsymbol{H} = \boldsymbol{B}^{-1}\boldsymbol{A} - \boldsymbol{B}^{-1}$ and obtain a product of

$$\boldsymbol{M}\boldsymbol{S}^{-1} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix}\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{H} & \boldsymbol{I} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{B} \\ \boldsymbol{C} - \boldsymbol{DH} & \boldsymbol{D} \end{pmatrix}.$$

To complete factorization process, we apply the block Gauss elimination by pre-multiplying the product $\boldsymbol{M}\boldsymbol{S}^{-1}$ by the Gauss block elementary matrix $\boldsymbol{L}^{-1}$, and we finally get

$$\boldsymbol{L}^{-1}\boldsymbol{M}\boldsymbol{S}^{-1} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{DH} - \boldsymbol{C} & \boldsymbol{I} \end{pmatrix}\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix}\begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ -\boldsymbol{H} & \boldsymbol{I} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{B} \\ \boldsymbol{0} & (\boldsymbol{DH} - \boldsymbol{C})\boldsymbol{B} + \boldsymbol{D} \end{pmatrix} = \boldsymbol{U},$$

where $\boldsymbol{L}^{-1}$ and $\boldsymbol{S}^{-1}$ are lower block triangular with unit diagonal blocks, $\boldsymbol{U}$ is upper block triangular, and

$$(\boldsymbol{DH}-\boldsymbol{C})\boldsymbol{B}+\boldsymbol{D}=\boldsymbol{DB}^{-1}\boldsymbol{AB}-\boldsymbol{DB}^{-1}\boldsymbol{B}-\boldsymbol{CB}+\boldsymbol{D}=(\boldsymbol{DB}^{-1}\boldsymbol{A}-\boldsymbol{C})\boldsymbol{B}.$$

Using the algebra of block triangular matrices, pre-multiplying the product $\boldsymbol{L}^{-1}\boldsymbol{MS}^{-1}$ by $\boldsymbol{L}$, and post-multiplying by $\boldsymbol{S}$ we obtain the block $\boldsymbol{PLUS}$ factorization of $\boldsymbol{M}$ as

$$\begin{pmatrix}\boldsymbol{A}&\boldsymbol{B}\\ \boldsymbol{C}&\boldsymbol{D}\end{pmatrix}=\begin{pmatrix}\boldsymbol{I}&\boldsymbol{0}\\ \boldsymbol{C}-\boldsymbol{DH}&\boldsymbol{I}\end{pmatrix}\begin{pmatrix}\boldsymbol{I}&\boldsymbol{B}\\ \boldsymbol{0}&(\boldsymbol{DB}^{-1}\boldsymbol{A}-\boldsymbol{C})\boldsymbol{B}\end{pmatrix}\begin{pmatrix}\boldsymbol{I}&\boldsymbol{0}\\ \boldsymbol{H}&\boldsymbol{I}\end{pmatrix}, \tag{A.63}$$

where

$$\boldsymbol{H}=\boldsymbol{B}^{-1}\boldsymbol{A}-\boldsymbol{B}^{-1}.$$

If the blocks $\boldsymbol{A}$ and $\boldsymbol{B}$ are commuting, i.e., $\boldsymbol{AB}=\boldsymbol{BA}$, Eq. (A.63) can be simplified as

$$\begin{pmatrix}\boldsymbol{A}&\boldsymbol{B}\\ \boldsymbol{C}&\boldsymbol{D}\end{pmatrix}=\begin{pmatrix}\boldsymbol{I}&\boldsymbol{0}\\ \boldsymbol{C}-\boldsymbol{DH}&\boldsymbol{I}\end{pmatrix}\begin{pmatrix}\boldsymbol{I}&\boldsymbol{B}\\ \boldsymbol{0}&\boldsymbol{DA}-\boldsymbol{CB}\end{pmatrix}\begin{pmatrix}\boldsymbol{I}&\boldsymbol{0}\\ \boldsymbol{H}&\boldsymbol{I}\end{pmatrix}, \tag{A.64}$$

Appendix B
Odd-Time Odd-Frequency Discrete Fourier transform (O^2DFT)

B.1 Definitions and Symmetry Properties

The odd-time odd-frequency discrete Fourier transform (O^2DFT) has been introduced in digital signal processing as an alternative method to efficiently transform symmetric real-valued data sequences [1, 5]. Essentially, the fast O^2DFT algorithm for real-valued symmetric data sequences based on an $\frac{N}{4}$-point complex-valued DFT has played the key role in the efficient MDCT computation as well as the efficient computation of DCT-IV/DST-IV [16, 23, 35].

An arbitrary real-valued data sequence $\{x_n\},\ n = 0, 1, \ldots, N-1$, can be split into its odd and even parts, respectively, as

$$u_n = \frac{1}{2}\,(x_n - x_{N-1-n}), \qquad v_n = \frac{1}{2}\,(x_n + x_{N-1-n}), \qquad n = 0, 1, \ldots, N-1. \tag{B.1}$$

Then, odd part $\{u_n\}$ and even part $\{v_n\}$ have the corresponding odd and even symmetry, respectively, given by

$$u_n = -w_{N-1-n}, \qquad v_n = v_{N-1-n}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{B.2}$$

It is obvious that $x_n = u_n + v_n,\ n = 0, 1, \ldots, N-1$.

Let $\{x_n\},\ n = 0, 1, \ldots, N-1$, represent an input data sequence and $\{f_k\},\ k = 0, 1, \ldots, N-1$, represent O^2DFT frequency coefficients, where N is an even integer. Then, the forward and inverse O^2DFT are, respectively, defined as [1, 5]

$$f_k = O^2DFT\{x_n\} = \frac{1}{N}\sum_{n=0}^{N-1} x_n\, W_{4N}^{-(2n+1)(2k+1)}, \quad k = 0, 1, \ldots, N-1, \tag{B.3}$$

$$x_n = O^2DFT^{-1}\{f_k\} = \sum_{k=0}^{N-1} f_k\, W_{4N}^{(2n+1)(2k+1)}, \quad n = 0, 1, \ldots, N-1, \tag{B.4}$$

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1

where $W_N^{\mp nk} = e^{\mp i\frac{2\pi nk}{N}} = \cos\frac{2\pi nk}{N} \mp i \sin\frac{2\pi nk}{N}$, and $i = \sqrt{-1}$. If $\{x_n\}$ is real-valued, then $\{f_k\}$ possesses the basic conjugate symmetry property given by

$$f_{N-1-k} = -f_k^*, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{B.5}$$

where $*$ denotes complex conjugate. Due to this basic symmetry property it is sufficient to calculate the transform for even indices only [1, 5]. Similarly, if $\{f_k\}$ is real-valued, then

$$x_{N-1-n} = -x_n^*, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{B.6}$$

Additionally, when the input data sequence $\{x_n\}$ is real-valued and odd symmetric, i.e., $x_{N-1-n} = -x_n,\ n = 0, 1, \ldots, \frac{N}{2} - 1$, then $\{f_k\}$ is purely real and odd symmetric. When $\{x_n\}$ is even symmetric, i.e., $x_{N-1-n} = x_n,\ n = 0, 1, \ldots, \frac{N}{2} - 1$, then $\{f_k\}$ is purely imaginary and even symmetric. N is an integer multiple of 4.

The O^2DFT is equivalent to the generalized discrete Fourier transform of type IV (GDFT-IV) which is related to the generalized discrete Hartley transform of type IV (GDHT-IV) of real-valued data sequences [10]. Additionally, the real/imaginary parts of both the O^2DFT and GDFT-IV of real-valued data sequences are intrinsically related to the DCT-IV/DST-IV of half sizes [10, 23, 35]. Consequently, the DCT-IV/DST-IV computation can be efficiently realized via the double length fast O^2DFT algorithm of odd/even extended real-valued data sequences [23].

B.2 The Fast O^2DFT Algorithm

The following fast algorithm can be used for the efficient computation of the forward O^2DFT [5]

$$\begin{aligned} f_k = f_{2k} + i f_{\frac{N}{2}+2k} &= \frac{2}{N} \sum_{n=0}^{\frac{N}{4}-1} \left(x_{2n} - i\, x_{\frac{N}{2}+2n}\right) W_{4N}^{-(4n+1)(4k+1)} \\ &= \frac{2}{N} W_{8N}^{-(8k+1)} \sum_{n=0}^{\frac{N}{4}-1} \left[\left(x_{2n} - i\, x_{\frac{N}{2}+2n}\right) W_{8N}^{-(8n+1)}\right] W_{\frac{N}{4}}^{-nk}, \\ k &= 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{B.7}$$

The transform kernel $W_{4N}^{-(4n+1)(4k+1)} = W_{8N}^{-(8k+1)} W_{\frac{N}{4}}^{-nk} W_{8N}^{-(8n+1)}$ is uniformly split into two equal parts $\frac{N}{4}$-point forward complex-valued DFT (CDFT). The necessary separation process in the above fast algorithm is given by

$$f_{2k} = \Re e\,\{f_k\}, \qquad f_{\frac{N}{2}+2k} = \Im m\,\{f_k\}, \qquad k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{B.8}$$

This separation process is less obvious for real-valued even symmetric data sequences because the corresponding transform is purely imaginary, and it is given by Cramer and Gluth et al. [16]

$$f_{2k} = i\,\Im m\,\{f_k\}, \qquad f_{\frac{N}{2}+2k} = -i\,\Re e\,\{f_k\}, \qquad k = 0, 1, \ldots, \frac{N}{4} - 1. \tag{B.9}$$

Since for an arbitrary real-valued data sequence the following relation holds:

$$x_n = O^2DFT^{-1}\{f_k\} = O^2DFT\left\{f_k^*\right\}, \tag{B.10}$$

the inverse O^2DFT computation can be realized by the same algorithm as

$$\begin{aligned} x_n = x_{2n} + i\,x_{\frac{N}{2}+2n} &= \sum_{k=0}^{\frac{N}{4}-1} \left(f_{2k} + i\,f_{\frac{N}{2}+2k}\right)^* \; W_{4N}^{-(4n+1)(4k+1)} \\ &= W_{8N}^{-(8n+1)} \sum_{k=0}^{\frac{N}{4}-1} \left[\left(f_{2k} - i\,f_{\frac{N}{2}+2k}\right) \; W_{8N}^{-(8k+1)}\right] W_{\frac{N}{4}}^{-nk}, \\ n &= 0, 1, \ldots, \frac{N}{4} - 1. \end{aligned} \tag{B.11}$$

The necessary separation process in the above fast algorithm is given by

$$x_{2n} = \Re e\,\{x_n\}, \qquad x_{\frac{N}{2}+2n} = \Im m\,\{x_n\}, \qquad n = 0, 1, \ldots, \frac{N}{4} - 1. \tag{B.12}$$

The fast O^2DFT algorithm is not restricted to real-valued data sequences with the odd/even symmetry but it is also valid for any purely imaginary data sequences with the odd/even symmetry [16]. In this case $\{f_k\}$ has the basic conjugate symmetry property given by

$$f_{N-1-k} = f_k^*, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{B.13}$$

Appendix C
Fast DCT/DST Computational Structures

C.1 Fast DCT-II/DCT-III Computational Structures

The unnormalized forward N-point DCT-II is defined by Malvar [32] and Britanak et al. [11]

$$c_k^{II} = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi(2n+1)k}{2N}\right], \qquad k = 0, 1, \ldots, N-1, \tag{C.1}$$

while unnormalized inverse DCT-II, the DCT-III, is defined by Malvar [32] and Britanak et al. [11]

$$x_n = \sum_{k=0}^{N-1} c_k^{II} \cos\left[\frac{\pi(2n+1)k}{2N}\right], \qquad n = 0, 1, \ldots, N-1, \tag{C.2}$$

where $\{x_n\}$ is an input data sequence and $\left\{c_k^{II}\right\}$ are DCT-II transform coefficients.

Exchanging the roles of time and frequency indices n and k in (C.1), the DCT-II can be equivalently written as

$$x_n = \sum_{k=0}^{N-1} c_k^{II} \cos\left[\frac{\pi(2k+1)n}{2N}\right], \qquad k = 0, 1, \ldots, N-1. \tag{C.3}$$

Substituting $N - n$ for $n = 1, 2, \ldots, N$, into (C.3) we obtain the following useful relation:

$$x_{N-n} = \sum_{k=0}^{N-1} c_k^{II} (-1)^k \sin\left[\frac{\pi(2k+1)n}{2N}\right], \qquad n = 1, 2, \ldots, N, \tag{C.4}$$

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1

where the sine transform kernel in (C.4) is recognized as that of corresponding type-II discrete sine transform (DST-II) [11]. In contrast to the DCT-IV, the DCT-II is not symmetric transform. Indeed, representing the DCT-II defined by (C.1) by the matrix $\boldsymbol{C}_N^{II}$ and the DCT-III defined by (C.2) by the matrix $\boldsymbol{C}_N^{III}$ we have: $\left[\boldsymbol{C}_N^{II}\right]^{-1} = \boldsymbol{C}_N^{III} = \left[\boldsymbol{C}_N^{II}\right]^T$ or vice versa, $\left[\boldsymbol{C}_N^{III}\right]^{-1} = \boldsymbol{C}_N^{II} = \left[\boldsymbol{C}_N^{III}\right]^T$, where T denotes matrix transposition. It means that the DCT-III computation can be realized by a fast DCT-II computational structure which has to be inverted or performed in the reverse direction.

C.1.1 Fast Recursive Even-Length DCT-II

Although many fast (recursive) DCT-II/DCT-III algorithms for the $N = 2^n$ lengths are available in the literature [4, 6, 11, 13, 30, 32, 40], perhaps the most suitable algorithm in terms of the generality and simplicity is the well known even-length fast recursive DCT-II algorithm defined as [30]

$$
\begin{aligned}
c_{2k}^{II} &= \sum_{n=0}^{\frac{N}{2}-1} (x_n + x_{N-1-n}) \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right], \\
c_{2k+1}^{II} &= \sum_{n=0}^{\frac{N}{2}-1} \left((x_n - x_{N-1-n})\, 2 \cos\frac{\pi(2n+1)}{2N}\right) \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right] - c_{2k-1}^{II}, \\
k &= 0, 1, \ldots, \frac{N}{2} - 1.
\end{aligned}
\tag{C.5}
$$

Denoting the sum in the second expression of (C.5) by y_{2k+1}, the odd-indexed DCT-II frequency coefficients are obtained as

$$
\begin{aligned}
2\, c_1^{II} &= y_0, && \text{if } k = 0, \\
c_{2k+1}^{II} &= y_{2k+1} - y_{2k-1}, && k = 1, 2, \ldots, \frac{N}{2} - 1.
\end{aligned}
\tag{C.6}
$$

Thus, the even-length N-point DCT-II is decomposed into two $\frac{N}{2}$-point DCTs-II given by (C.5) and the recursive post-addition stage given by (C.6). When $N = 2^n$ (the so-called radix-2 algorithm), the decomposition defined by (C.5) and (C.6) is repeated recursively until only 2-point DCTs-II remain. The arithmetic complexity of computing 2^n-point DCT-II is given by Kok [30]

$$
M_{2^n}^{II} = \frac{N}{2}\, n, \qquad A_{2^n}^{II} = \frac{N}{2}\,(3n - 2) + 1,
\tag{C.7}
$$

where $M_{2^n}^{II}$ and $A_{2^n}^{II}$ denote, respectively, the number of multiplications and additions. When N is a composite number, i.e., it is of the form $N = 2^n \times q$, $n > 0$, where q is an odd positive integer (the so-called mixed-radix algorithm being a combination of radix-2 and radix-q algorithms), the decomposition defined by (C.5) and (C.6) is repeated recursively m times until only 2^n odd q-length DCTs-II remain [3, 4]. The arithmetic complexity of computing the N-point DCT-II for composite lengths is given by Kok [30]

$$M_{2^n \times q}^{II} = 2^n \times M_q^{II} + \frac{N}{2} n, \qquad A_{2^n \times q}^{II} = 2^n \times A_q^{II} + \frac{3N}{2} n - 2^n + 1, \tag{C.8}$$

where M_q^{II} and A_q^{II} denote, respectively, the number of multiplications and additions of an odd q-length DCT-II.

C.2 Fast DCT-IV/DST-IV Computational Structures

The unnormalized forward and inverse N-point DCT-IV transforms are, respectively, defined by Malvar [32] and Britanak et al. [11]

$$c_k^{IV} = \sum_{n=0}^{N-1} x_n^{(c)} \cos\left[\frac{\pi}{4N}(2k+1)(2n+1)\right], \qquad k = 0, 1, \ldots, N-1, \tag{C.9}$$

$$x_n^{(c)} = \sum_{k=0}^{N-1} c_k^{IV} \cos\left[\frac{\pi}{4N}(2k+1)(2n+1)\right], \qquad n = 0, 1, \ldots, N-1, \tag{C.10}$$

where $\left\{x_n^{(c)}\right\}$ is an input data sequence and $\left\{c_k^{IV}\right\}$ are DCT-IV transform coefficients. Since the cosine transform kernel in (C.9) and (C.10) is symmetric with respect to the time and frequency indices n and k, the DCT-IV is the symmetric transform. Indeed, representing the forward and inverse DCT-IV respectively given by (C.9) and (C.10) by the matrix $\boldsymbol{C}_N^{IV}$ we have: $\left[\boldsymbol{C}_N^{IV}\right]^{-1} = \boldsymbol{C}_N^{IV} = \left[\boldsymbol{C}_N^{IV}\right]^T$, i.e., $\boldsymbol{C}_N^{IV}$ is self-inverse. T denotes the matrix transposition. Consequently, the forward and inverse DCT-IV can be realized by an identical fast computational structure.

On the other hand, the unnormalized forward and inverse N-point DST-IV transforms are, respectively, defined by Britanak et al. [11]

$$s_k^{IV} = \sum_{m=0}^{N-1} x_n^{(s)} \sin\left[\frac{\pi}{4N}(2k+1)(2n+1)\right], \qquad k = 0, 1, \ldots, N-1, \tag{C.11}$$

$$x_n^{(s)} = \sum_{k=0}^{N-1} s_k^{IV} \sin\left[\frac{\pi}{4N}(2k+1)(2n+1)\right], \qquad n = 0, 1, \ldots, N-1, \tag{C.12}$$

where $\left\{x_n^{(s)}\right\}$ is an input data sequence and $\left\{s_k^{IV}\right\}$ are DST-IV transform coefficients. There exists a close relation between the DCT-IV and DST-IV and vice versa. Specifically, substituting $N-1-k$ for k into Eq. (C.11) and then $N-1-n$ for n into (C.12) we, respectively, obtain:

$$s_{N-1-k}^{IV} = \sum_{k=0}^{N-1} (-1)^n \, x_n^{(s)} \, \cos\left[\frac{\pi}{4N}(2k+1)(2n+1)\right], \qquad k = 0, 1, \ldots, N-1. \quad \text{(C.13)}$$

$$x_{N-1-n}^{(s)} = \sum_{k=0}^{N-1} s_k^{IV} \, (-1)^k \, \cos\left[\frac{\pi}{4N}(2k+1)(2n+1)\right], \qquad n = 0, 1, \ldots, N-1. \quad \text{(C.14)}$$

Further, substituting $N-1-k$ for k into Eq. (C.9) and then $N-1-n$ for n into (C.10) we, respectively, obtain:

$$c_{N-1-k}^{IV} = \sum_{k=0}^{N-1} (-1)^n \, x_n^{(c)} \, \sin\left[\frac{\pi}{4N}(2k+1)(2n+1)\right], \qquad k = 0, 1, \ldots, N-1. \quad \text{(C.15)}$$

$$x_{N-1-n}^{(c)} = \sum_{k=0}^{N-1} c_k^{IV} \, (-1)^k \, \sin\left[\frac{\pi}{4N}(2k+1)(2n+1)\right], \qquad n = 0, 1, \ldots, N-1. \quad \text{(C.16)}$$

Relations (C.13), (C.14), (C.15), and (C.16) imply that the forward and inverse DST-IV can be realized via the corresponding DCT-IV, or vice versa. Finally, substituting $N-1-k$ for k into (C.16) we obtain

$$x_{N-1-n}^{(c)} = (-1)^n \sum_{k=0}^{N-1} c_{N-1-k}^{IV} \, (-1)^{N-1-k} \, \cos\left[\frac{\pi}{4N}(2k+1)(2n+1)\right],$$
$$n = 0, 1, \ldots, N-1. \quad \text{(C.17)}$$

Relations (C.16) and (C.17) are very useful in the derivations of the fast algorithms for cosine-modulated filter banks.

C.2.1 *Fast 2^m-Length DCT-IV/DST-IV Computational Structures*

For the efficient DCT-IV computation various fast algorithms for $N = 2^n$ lengths may be adopted [7, 9, 11, 13, 23, 32, 35]. However, the most suitable 2^n-length fast DCT-IV algorithm which combines the theoretical efficiency with a very regular

computational structure and achieving the lowest multiplicative complexity [23, 32, 35] is based on a fast O^2DFT algorithm derived for odd/even symmetric real-valued data sequences [5]. For the forward N-point DCT-IV computation it is defined as [23, 32]

$$f_k = \exp\left(-i\,\frac{\pi(8k+1)}{8N}\right)\sum_{n=0}^{\frac{N}{2}-1}\left[\left(x_{2n}^{(c)} + i\,x_{N-1-2n}^{(c)}\right)\,\exp\left(-i\,\frac{\pi(8n+1)}{8N}\right)\right] \exp\left(-i\,\frac{2\pi nk}{N/2}\right),$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{C.18}$$

where $\{f_k\}$ are complex-valued DFT transform coefficients, and $i = \sqrt{-1}$ is the imaginary unit. Complex-valued twiddle factors $\exp\left(-i\,\frac{\pi(8k+1)}{8N}\right)$ and $\exp\left(-i\,\frac{\pi(8n+1)}{8N}\right)$ in (C.18), correspond to the identical blocks of $\frac{N}{2}$ Givens–Jacobi pre- and post-rotations. The complex exponential transform kernel $\exp\left(-i\,\frac{2\pi nk}{N/2}\right)$ corresponds to the $\frac{N}{2}$-point complex forward FFT. The final DCT-IV coefficients are obtained as

$$c_{2k}^{IV} = \Re e\,\{f_k\}, \qquad c_{N-1-2k}^{IV} = -\Im m\,\{f_k\}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{C.19}$$

On the other hand, the fast FFT-based algorithm for the inverse N-point DCT-IV computation is defined as [23, 32]

$$f_n = \exp\left(-i\,\frac{\pi(8n+1)}{8N}\right)\sum_{k=0}^{\frac{N}{2}-1}\left[\left(c_{2k}^{IV} + i\,c_{N-1-2k}^{IV}\right)\,\exp\left(-i\,\frac{\pi(8k+1)}{8N}\right)\right] \exp\left(-i\,\frac{2\pi nk}{N/2}\right),$$
$$n = 0, 1, \ldots, \frac{N}{2} - 1, \tag{C.20}$$

and the data sequence $\{x_n^{(c)}\}$ is obtained as

$$x_{2n}^{(c)} = \Re e\,\{f_n\}, \qquad x_{N-1-2n}^{(c)} = -\Im m\,\{f_n\}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{C.21}$$

Comparing Eqs. (C.18) and (C.20) for exchanged time and frequency indices n and k results in the identical fast DCT-IV computational structure both for the forward and inverse DCT-IV computation. Block diagram of the fast DCT-IV computational

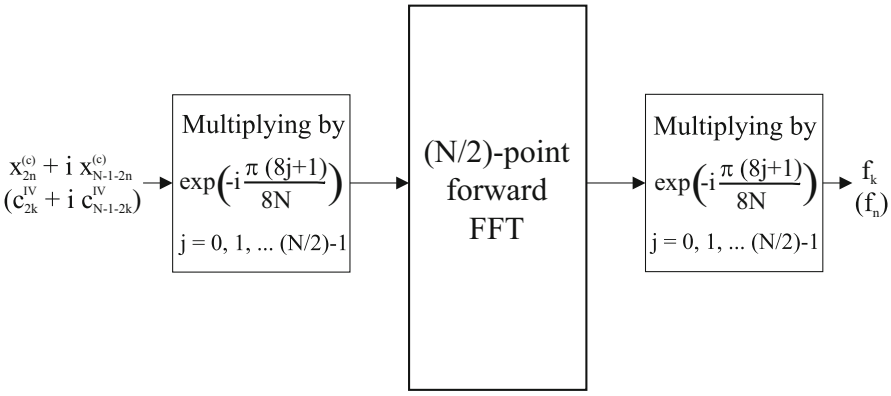

Fig. C.1 Block diagram of the fast DCT-IV computational structure based on FFT

structure based on FFT is shown in Fig. C.1. Its arithmetic complexity for $N = 2^n$ lengths is given by [32]

$$M_N^{IV} = \frac{N}{2}\,(n+2), \qquad A_N^{IV} = \frac{3N}{2}\,n, \tag{C.22}$$

where M_N^{IV} and A_N^{IV} denote, respectively, the number of real multiplications and real additions.

For completeness, the corresponding FFT-based fast algorithm for the forward N-point DST-IV computation is defined as [23]

$$f_k = \exp\left(-i\,\frac{\pi(8k+1)}{8N}\right)\sum_{n=0}^{\frac{N}{2}-1}\left[\left(x^{(s)}_{2n} - i\,x^{(s)}_{N-1-2n}\right)\exp\left(-i\,\frac{\pi(8n+1)}{8N}\right)\right]$$
$$\exp\left(-i\,\frac{2\pi nk}{N/2}\right),$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{C.23}$$

and the final DST-IV coefficients are obtained as

$$s^{IV}_{2k} = -\,\Im m\,\{f_k\}, \qquad s^{IV}_{N-1-2k} = \Re e\,\{f_k\}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1. \tag{C.24}$$

The fast FFT-based algorithm for the inverse N-point DST-IV computation is defined as [23]

$$f_n = \exp\left(-i\,\frac{\pi(8n+1)}{8N}\right)\sum_{k=0}^{\frac{N}{2}-1}\left[\left(s^{IV}_{2k} - i\,s^{IV}_{N-1-2k}\right)\exp\left(-i\,\frac{\pi(8k+1)}{8N}\right)\right]$$
$$\exp\left(-i\,\frac{2\pi nk}{N/2}\right),$$

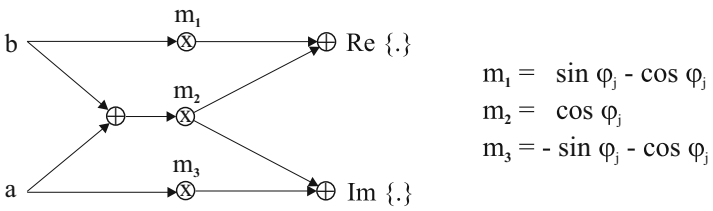

Fig. C.2 Bilinear computational structure for the efficient implementation of complex multiplications by twiddle factors $\exp(-i\,\varphi_j) = \cos\varphi_j - i\sin\varphi_j$, where $\varphi_j = \frac{\pi(8j+1)}{8N}$, $j = 0, 1, \ldots, \frac{N}{2} - 1$

$$n = 0, 1, \ldots, \frac{N}{2} - 1, \tag{C.25}$$

and the data sequence $\{x_n^{(s)}\}$ is obtained as

$$x_{2n}^{(s)} = -\,\Im m\,\{f_n\}, \qquad x_{N-1-2n}^{(s)} = \Re e\,\{f_n\}, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{C.26}$$

The fast DCT-IV computational structure shown in Fig. C.1 can easily be adopted for the DST-IV computation.

Two identical blocks of twiddle factors in Eqs. (C.18) and (C.20) are complex numbers of the form: $\exp(-i\,\varphi_j) = \cos\varphi_j - i\sin\varphi_j$, where $\varphi_j = \frac{\pi(8j+1)}{8N}$, $j = 0, 1, \ldots, \frac{N}{2} - 1$. Multiplication of two complex numbers $[a + ib]$ and $[\cos\varphi_j - i\sin\varphi_j]$ can be written in the equivalent matrix-vector notation as

$$\begin{pmatrix} \Re e\,\{.\} \\ \Im m\,\{.\} \end{pmatrix} = \begin{pmatrix} \cos\varphi_j & \sin\varphi_j \\ -\sin\varphi_j & \cos\varphi_j \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}, \tag{C.27}$$

where the 2×2 matrix on the right-hand side of (C.27) is a Givens–Jacobi rotation matrix [24]. Two identical blocks of twiddle factors in Eqs. (C.18) and (C.20) defining the fast DCT-IV computational structure which is shown in Fig. C.1 can be replaced by the bilinear computational structures for the efficient implementation of Givens–Jacobi rotations (see Appendix F.3). The bilinear computational structure shown for efficient implementation of complex multiplications by twiddle factors is shown in Fig. C.2.

C.2.2 Fast Even-Length DCT-IV Computational Structure

An even-length fast DCT-IV algorithm based on two DCTs-II of half sizes has been proposed in [9]. The so-called N-point fast DCT-IV computational structure with constant geometry is defined as

$$c_{2k}^{IV} = \sum_{n=0}^{\frac{N}{2}-1} a_n \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right] + b_n \sin\left[\frac{\pi(2n+1)k}{2(N/2)}\right],$$

$$c_{2k-1}^{IV} = \sum_{n=0}^{\frac{N}{2}-1} a_n \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right] - b_n \sin\left[\frac{\pi(2n+1)k}{2(N/2)}\right], \; k = 1, 2, \ldots, \frac{N}{2} - 1, \tag{C.28}$$

where

$$\begin{aligned} a_n &= \quad y_n \cos\frac{\pi(2n+1)}{4N} + y_{\frac{N}{2}-1-n} \sin\frac{\pi(2n+1)}{4N}, \\ b_n &= -y_n \sin\frac{\pi(2n+1)}{4N} + y_{\frac{N}{2}-1-n} \cos\frac{\pi(2n+1)}{4N}, \quad n = 0, 1, \ldots, \frac{N}{2} - 1. \end{aligned} \tag{C.29}$$

Thus, the N-point DCT-IV is decomposed into the block of $\frac{N}{2}$ Givens–Jacobi rotations given by (C.29), an $\frac{N}{2}$-point DCT-II and a corresponding $\frac{N}{2}$-point type-II discrete sine transform (DST-II), whose outputs are combined to obtain the final DCT-IV frequency coefficients. For $k = 0$ in the first sum and for $k = \frac{N}{2}$ in the second sum of (C.28), we, respectively, get

$$c_0^{IV} = \sum_{n=0}^{\frac{N}{2}-1} a_n, \qquad c_{N-1}^{IV} = -\sum_{n=0}^{\frac{N}{2}-1} (-1)^n b_n. \tag{C.30}$$

Further, denoting the $\frac{N}{2}$-point DCT-II of $\{a_n\}$ and $\frac{N}{2}$-point DST-II of $\{b_n\}$ in Eq. (C.28), respectively, by

$$c_k^{II} = \sum_{n=0}^{\frac{N}{2}-1} a_n \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right], \qquad s_k^{II} = \sum_{n=0}^{\frac{N}{2}-1} b_n \sin\left[\frac{\pi(2n+1)k}{2(N/2)}\right],$$
$$k = 1, 2, \ldots, \frac{N}{2} - 1,$$

and using the relation between the DCT-II and the DST-II [11], i.e., substituting $\frac{N}{2} - k$ for k into the second sum above we get

$$s_{\frac{N}{2}-k}^{II} = \sum_{n=0}^{\frac{N}{2}-1} (-1)^n b_n \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right] = \sum_{m=0}^{\frac{N}{2}-1} b'_n \cos\left[\frac{\pi(2n+1)k}{2(N/2)}\right],$$
$$k = 1, 2, \ldots, \frac{N}{2} - 1.$$

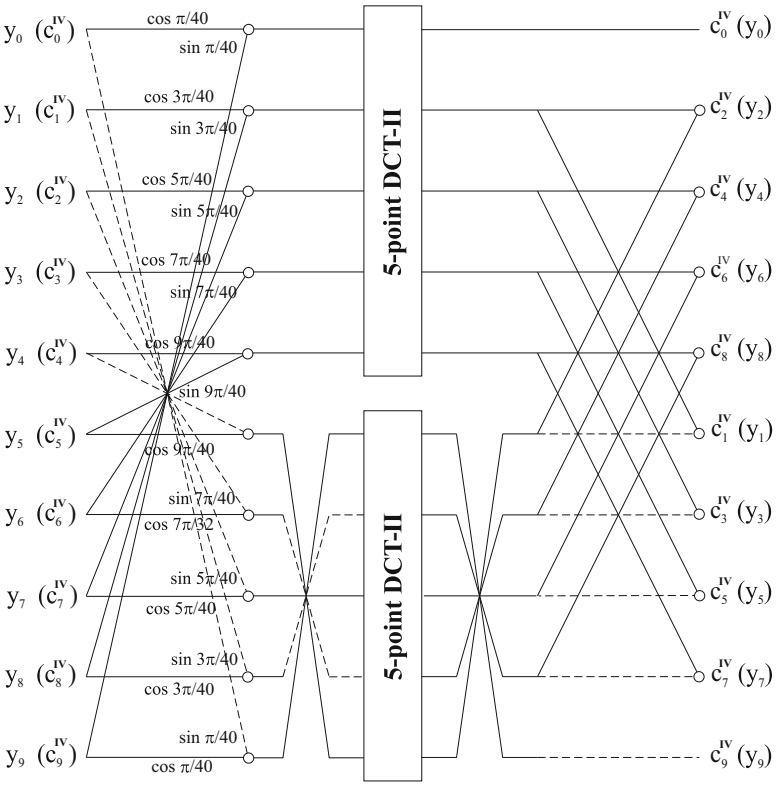

Fig. C.3 Regular 10-point fast DCT-IV computational structure [9]

Thus, the $\frac{N}{2}$-point DST-II of $\{b_n\}$ is converted to an $\frac{N}{2}$-point DCT-II of $\{b'_n\}$. Then Eqs. (C.28) and (C.30) can be rewritten in a simplified equivalent form as

$$c^{IV}_{2k} = c^{II}_{k} + s^{II}_{\frac{N}{2}-k}, \qquad c^{IV}_{0} = c^{II}_{0},$$

$$c^{IV}_{2k-1} = c^{II}_{k} - s^{II}_{\frac{N}{2}-k}, \qquad c^{IV}_{N-1} = -s^{II}_{\frac{N}{2}}, \qquad k = 1, 2, \ldots, \frac{N}{2} - 1. \quad \text{(C.31)}$$

Two unnormalized $\frac{N}{2}$-point DCTs-II in (C.31) can be implemented by the fast recursive DCT-II algorithm described in the Appendix C.1.1. As an example, the regular 10-point fast DCT-IV computational structure consisting of three stages is shown in Fig. C.3. The required efficient 5-point DCT-II/III modules additionally optimized together with their arithmetic complexity are presented in Appendix D.7. Note that the last butterfly stage in Fig. C.3 may be reorganized so that the DCT-IV frequency coefficients are obtained in natural order [9]. When $N = 2^n$, using

Eq. (C.7) the arithmetic complexity of the fast DCT-IV computational structure is given by (C.22). When N is a composite number ($N = 2^n \times q$, $n > 1$, q is an odd positive integer), using Eq. (C.8) the arithmetic complexity of the fast DCT-IV computational structure is given by

$$M^{IV}_{2^n \times q} = 2^n \times M^{II}_q + \frac{N}{2}(n+2), \qquad A^{IV}_{2^n \times q} = 2^n \times A^{II}_q + \frac{N}{2}(3n+2) - 2^n. \tag{C.32}$$

Since the DCT-IV is symmetric with respect to the time and frequency indices, i.e., the DCT-IV matrix is self-inverse, the forward and inverse DCT-IV can be realized by the identical fast computational structure.

C.3 Fast DCT-IV Algorithm Based on the DCT-II and Fast DCT-II/DCT-III Algorithm Based on DCT-IV

In general, for the computation of DCT-II/DCT-III we need two separate fast computational structures, one for the DCT-II and one for the DCT-III [6]. In order to minimize memory resources, it is possible to propose alternative fast DCT-II and DCT-III computational structures based on a relation between the DCT-IV and the DCT-II defined by a matrix product as follows. It is well known that the matrix $\boldsymbol{C}^{IV}_N$ is related to $\boldsymbol{C}^{II}_N$ matrix by Britanak et al. [11]

$$\boldsymbol{C}^{IV}_N = \boldsymbol{L}_N \, \boldsymbol{C}^{II}_N \, \boldsymbol{D}_N, \tag{C.33}$$

where $\boldsymbol{D}_N = \text{diag}\left\{2\cos\frac{\pi(2n+1)}{4N}\right\}$, $n = 0, 1, \ldots, N-1$, is the diagonal matrix of order N, and $\boldsymbol{L}_N$ is a lower triangular matrix of order N given by

$$\boldsymbol{L}_N = \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 & \cdots & 0 \\ -\frac{1}{2} & 1 & 0 & 0 & \cdots & 0 \\ \frac{1}{2} & -1 & 1 & 0 & \cdots & 0 \\ -\frac{1}{2} & 1 & -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -\frac{1}{2} & 1 & -1 & 1 & \cdots & 1 \end{pmatrix}. \tag{C.34}$$

Equation (C.33) has a twofold importance. Firstly, it provides an alternative fast DCT-IV algorithm realized via the DCT-II of the same size at the cost of additional N multiplications and $N-1$ recursive additions. However, in the efficient implementations of filter banks multiplications by the factors $2\cos\frac{\pi(2n+1)}{4N}$ can be simply absorbed into the windowing operations in the encoder/decoder, i.e., the windowing function is modified. On the other hand, for the inverse DCT-IV

computation a fast DCT-II computational structure has to be inverted. With respect to (C.7) the arithmetic complexity of fast forward/inverse DCT-IV computation via DCT-II/DCT-III is given by

$$M_N^{IV} = \frac{N}{2}\, n, \qquad A_N^{IV} = \frac{3N}{2}\, n. \tag{C.35}$$

Secondly, Eq. (C.33) provides a basis to construct alternative fast DCT-II and DCT-III computational structures via the DCT-IV of the same size. Matrices $\boldsymbol{D}_N$ and $\boldsymbol{L}_N$ in (C.33) are nonsingular, and therefore their inverse matrices exist. Performing matrix multiplications on both sides of (C.33), i.e., left-multiplying by $\boldsymbol{L}_N^{-1}$, then right-multiplying by inverse diagonal matrix $\boldsymbol{D}_N^{-1}$, we get

$$\boldsymbol{C}_N^{II} = \boldsymbol{L}_N^{-1}\, \boldsymbol{C}_N^{IV}\, \boldsymbol{D}_N^{-1}, \tag{C.36}$$

where $\boldsymbol{D}_N^{-1} = \text{diag}\ \left\{1/(2\cos\frac{\pi(2n+1)}{4N})\right\}$, $n = 0, 1, \ldots, N-1$, and $\boldsymbol{L}_N^{-1}$ is a lower bidiagonal matrix of order N given by

$$\boldsymbol{L}_N^{-1} = \begin{pmatrix} 2 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 1 \end{pmatrix}. \tag{C.37}$$

Equation (C.36) defines the fast DCT-II computational structure based on the DCT-IV. Further, by transposing both sides of (C.36) we get

$$\left[\boldsymbol{C}_N^{II}\right]^T = \boldsymbol{C}_N^{III} = \boldsymbol{D}_N^{-1}\, \boldsymbol{C}_N^{IV}\, \left[\boldsymbol{L}_N^{-1}\right]^T, \tag{C.38}$$

where $\left[\boldsymbol{L}_N^{-1}\right]^T$ is an upper bidiagonal matrix of order N given by

$$\left[\boldsymbol{L}_N^{-1}\right]^T = \begin{pmatrix} 2 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}. \tag{C.39}$$

Equation (C.38) defines the fast DCT-III computational structure again based on the DCT-IV. The diagonal matrix $\boldsymbol{D}_N^{-1}$ includes N pre- or post-multiplications. Both band bidiagonal matrices $\left[\boldsymbol{L}_N^{-1}\right]^T$ and $\boldsymbol{L}_N^{-1}$ include $N-1$ pre- or post-additions [22, 24, 39]. Hence, the DCT-III or DCT-II can be realized via the DCT-IV only by simple pre- or post-processing of input and output data sequences. If we adopt the fast DCT-IV computational structure defined by (C.18), according to (C.22) the arithmetic complexity of proposed fast DCT-II/DCT-III computational structures via the DCT-IV for $N = 2^n$ lengths is given by

$$M_N^{II/III} = \frac{N}{2}(n+4), \qquad A_N^{II/III} = \frac{N}{2}(3n+2) - 1. \tag{C.40}$$

Although the fast DCT-II and DCT-III computational structures via the DCT-IV require exactly $2N$ more multiplications and N more additions than classic fast DCT-II and DCT-III algorithms (see the arithmetic complexity given by (C.7)), nevertheless the composition of (C.18), (C.36), and (C.38) provides the elegant and simple compact computational system for the efficient implementation of all the discrete orthogonal transforms, DCT-IV, DCT-II, and DCT-III, in terms of structural simplicity, regularity, and memory requirements.

Appendix D
Optimized Efficient Short Odd-Length Complex DFT, Real-Valued DFT, and (S)DCT Modules

The optimized efficient odd-length modules represent the unscaled, unnormalized version of the appropriate discrete sinusoidal unitary transform. For each efficient odd-length module the corresponding total arithmetic complexity is also specified.

We note that in all odd-length efficient modules, additions and multiplications (shifts) are sequenced and must be executed in the specified order. When there are several expressions to a line, they are read left to right before proceeding to next line.

D.1 Table of Constants for All the Optimized Efficient Short Odd-Length Modules

$u = \frac{2\pi}{3}$ $d_1 = \cos(u)$ $d_2 = \sin(u)$

$u = \frac{2\pi}{9}$ $d_3 = \cos(4u)$ $d_4 = \cos(2u)$ $d_5 = \cos(u)$
$d_6 = \sin(4u)$ $d_7 = \sin(2u)$ $d_8 = \sin(u)$

$d_9 = 2\,d_2$ $d_{10} = 2\,d_3$ $d_{11} = 2\,d_4$ $d_{12} = 2\,d_5$
$d_{13} = 2\,d_6$ $d_{14} = 2\,d_7$ $d_{15} = 2\,d_8$

$d_{16} = \tan(\frac{u}{2})$ $d_{17} = \tan(\frac{u}{4})$ $d_{18} = \tan(u)$

D.2 Optimized Efficient 3/9-Point Complex-Valued DFT (CDFT) Modules

The optimized efficient 3/9-point CDFT modules have been derived from short-length Winograd Fourier transform algorithms for $N = 3$ and 9 [12, 19, 34]. The

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1

efficient 3/9-point CDFT modules represent fast algorithms for the forward 3/9-point DFT computation. In general, if the inverse 3/9-point DFT computation is required, then there exist two simple methods how to use a forward CDFT module for the inverse CDFT computation:

- Substituting $-u$ for u in the above table of constants and performing the forward CDFT module [19].
- Exchanging the real and imaginary parts in the input complex-valued data sequence, then performing the forward CDFT module and finally, exchanging the real and imaginary parts of the result [18].

D.2.1 3-point CDFT Module: 2 Real Mults, 12 Real Adds, and 2 Shifts

The input data sequence $\{x_0, x_1, x_2\}$ is complex-valued, the output data sequence $\{F_0, F_1, F_2\}$ is complex-valued and $i = \sqrt{-1}$.

$$a_1 = x_1 + x_2 \; a_2 = x_1 - x_2 \; a_3 = x_0 + a_1$$

$$m_1 = -i \; d_2 \; a_2 \; s_1 = d_1 \; a_1 = -\tfrac{1}{2} \; a_1$$

$$a_4 = x_0 + s_1$$

$$\begin{aligned} F_0 &= a_3 \\ F_1 &= a_4 + m_1 \\ F_2 &= a_4 - m_1 \end{aligned}$$

D.2.2 9-Point CDFT Module: 16 Real Mults, 84 Real Adds, and 4 Shifts

The input data sequence $\{x_0, x_1, \ldots, x_8\}$ is complex-valued, the output data sequence $\{F_0, F_1, \ldots, F_8\}$ is complex-valued, and $i = \sqrt{-1}$.

$$\begin{aligned} &a_1 = x_1 + x_8 \quad a_2 = x_1 - x_8 \quad a_3 = x_2 + x_7 \quad a_4 = x_2 - x_7 \\ &a_5 = x_3 + x_6 \quad a_6 = x_3 - x_6 \quad a_7 = x_4 + x_5 \quad a_8 = x_4 - x_5 \\ &a_9 = x_0 + a_5 \quad a_{10} = a_1 + a_3 \; a_{11} = a_{10} + a_7 \; a_{12} = a_3 - a_7 \\ &a_{13} = a_1 - a_7 \; a_{14} = a_1 - a_3 \; a_{15} = a_2 - a_4 \quad a_{16} = a_{15} + a_8 \\ &a_{17} = a_4 + a_8 \; a_{18} = a_2 - a_8 \; a_{19} = a_2 + a_4 \end{aligned}$$

$$m_1 = -i\, d_2\, a_6 \quad m_2 = -d_3\, a_{12} \quad m_3 = -d_4\, a_{13} \quad m_4 = -d_5\, a_{14}$$
$$m_5 = -i\, d_2\, a_{16} \quad m_6 = -i\, d_6\, a_{17} \quad m_7 = -i\, d_7\, a_{18} \quad m_8 = -i\, d_8\, a_{19}$$
$$s_1 = \tfrac{1}{2}\, a_5 \qquad s_2 = \tfrac{1}{2}\, a_{11}$$

$$a_{20} = x_0 - s_1 \quad a_{21} = a_{20} + m_2 \quad a_{22} = a_{20} - m_2 \quad a_{23} = a_{20} + m_3$$
$$a_{24} = m_1 + m_6 \quad a_{25} = m_1 - m_6 \quad a_{26} = m_1 + m_7 \quad a_{27} = a_{21} - m_4$$
$$a_{28} = a_{22} - m_3 \quad a_{29} = a_9 - s_2 \quad a_{30} = a_{23} + m_4 \quad a_{31} = a_{26} - m_8$$
$$a_{32} = m_7 - a_{25} \quad a_{33} = a_{24} + m_8$$

$$F_0 = a_9 + a_{11}$$
$$F_1 = a_{27} + a_{33}$$
$$F_2 = a_{28} + a_{32}$$
$$F_3 = a_{29} + m_5$$
$$F_4 = a_{30} + a_{31}$$
$$F_5 = a_{30} - a_{31}$$
$$F_6 = a_{29} - m_5$$
$$F_7 = a_{28} - a_{32}$$
$$F_8 = a_{27} - a_{33}$$

D.3 Optimized Efficient 3/9-Point Real-Valued DFT (RDFT) Modules

The optimized efficient 3/9-point forward and inverse RDFT modules are derived from the corresponding 3/9-point CDFT modules simply using the conjugate symmetry of DFT coefficients [38]. They are used to derive optimized efficient 3/9-point DCT-II and DCT-III modules.

D.3.1 3-Point Forward RDFT Module: 1 Mult, 4 Adds, and 1 Shift

The input data sequence $\{x_0, x_1, x_2\}$ is real-valued and the output data sequence $\{F_0, F_1, F_2\}$ is real-valued.

$$a_1 = x_1 + x_2 \; a_2 = x_1 - x_2$$

$$m_1 = -d_2\, a_2 \; s_1 = -\tfrac{1}{2}\, a_1$$

$$F_0 = x_0 + a_1$$
$$F_1 = x_0 + s_1$$
$$F_2 = m_1$$

D.3.2 3-Point Inverse RDFT Module: 1 Mult, 4 Adds, and 1 Shift

The input data sequence $\{F_0, F_1, F_2\}$ is real-valued and the output data sequence $\{x_0, x_1, x_2\}$ is real-valued.

$$m_1 = -d_2\ F_2\ s_1 = -\tfrac{1}{2}\ F_1$$
$$a_1 = F_0 + F_1\ a_2 = F_0 + s_1$$
$$x_0 = a_1$$
$$x_1 = a_2 + m_1$$
$$x_2 = a_2 - m_1$$

D.3.3 9-Point Forward RDFT Module: 8 Mults, 34 Adds, and 2 Shifts

The input data sequence $\{x_0, x_1, \ldots, x_8\}$ is real-valued and the output data sequence $\{F_0, F_1, \ldots, F_8\}$ is real-valued.

$$a_1 = x_1 + x_8 \quad a_2 = x_1 - x_8 \quad a_3 = x_2 + x_7 \quad a_4 = x_2 - x_7$$
$$a_5 = x_3 + x_6 \quad a_6 = x_3 - x_6 \quad a_7 = x_4 + x_5 \quad a_8 = x_4 - x_5$$
$$a_9 = x_0 + a_5 \quad a_{10} = a_1 + a_3\ a_{11} = a_{10} + a_7\ a_{12} = a_3 - a_7$$
$$a_{13} = a_1 - a_7\ a_{14} = a_1 - a_3\ a_{15} = a_2 - a_4 \quad a_{16} = a_{15} + a_8$$
$$a_{17} = a_4 + a_8\ a_{18} = a_2 - a_8\ a_{19} = a_2 + a_4$$

$$m_1 = -d_2\ a_6 \quad m_2 = -d_3\ a_{12}\ m_3 = -d_4\ a_{13}\ m_4 = -d_5\ a_{14}$$
$$m_5 = -d_2\ a_{16}\ m_6 = -d_6\ a_{17}\ m_7 = -d_7\ a_{18}\ m_8 = -d_8\ a_{19}$$
$$s_1 = \tfrac{1}{2}\ a_5 \qquad s_2 = \tfrac{1}{2}\ a_{11}$$

$$a_{20} = x_0 - s_1 \quad a_{21} = a_{20} + m_2\ a_{22} = a_{20} - m_2\ a_{23} = a_{20} + m_3$$
$$a_{24} = m_1 + m_6\ a_{25} = m_1 - m_6 \quad a_{26} = m_1 + m_7$$

$$
\begin{aligned}
F_0 &= a_9 + a_{11} \\
F_1 &= a_{21} - m_4 \\
F_2 &= a_{22} - m_3 \\
F_3 &= a_9 - s_2 \\
F_4 &= a_{23} + m_4 \\
F_5 &= a_{26} - m_8 \\
F_6 &= m_5 \\
F_7 &= m_7 - a_{25} \\
F_8 &= a_{24} + m_8
\end{aligned}
$$

D.3.4 9-Point Inverse RDFT Module: 8 Mults, 34 Adds, and 2 Shifts

The input data sequence $\{F_0, F_1, \ldots, F_8\}$ is real-valued and the output data sequence $\{x_0, x_1, \ldots, x_8\}$ is real-valued.

$$
\begin{aligned}
&a_1 = F_0 + F_3 \; a_2 = F_1 + F_2 \;\; a_3 = a_2 + F_4 \;\; a_4 = F_2 - F_4 \\
&a_5 = F_1 - F_4 \; a_6 = F_1 - F_2 \;\; a_7 = F_8 - F_7 \;\; a_8 = a_7 + F_5 \\
&a_9 = F_7 + F_5 \; a_{10} = F_8 - F_5 \; a_{11} = F_8 + F_7
\end{aligned}
$$

$$
\begin{aligned}
&m_1 = -d_2 \, F_6 \; m_2 = -d_3 \, a_4 \; m_3 = -d_4 \, a_5 \;\; m_4 = -d_5 \, a_6 \\
&m_5 = -d_2 \, a_8 \;\; m_6 = -d_6 \, a_9 \; m_7 = -d_7 \, a_{10} \; m_8 = -d_8 \, a_{11} \\
&s_1 = \tfrac{1}{2} \, F_3 \qquad s_2 = \tfrac{1}{2} \, a_3
\end{aligned}
$$

$$
\begin{aligned}
&a_{12} = F_0 - s_1 \;\; a_{13} = a_{12} + m_2 \; a_{14} = a_{12} - m_2 \;\; a_{15} = a_{12} + m_3 \\
&a_{16} = m_1 + m_6 \; a_{17} = m_1 - m_6 \;\; a_{18} = m_1 + m_7 \;\; a_{19} = a_{13} - m_4 \\
&a_{20} = a_{14} - m_3 \; a_{21} = a_1 - s_2 \;\; a_{22} = a_{15} + m_4 \; a_{23} = a_{18} - m_8 \\
&a_{24} = m_7 - a_{17} \; a_{25} = a_{16} + m_8
\end{aligned}
$$

$$
\begin{aligned}
x_0 &= a_1 + a_3 \\
x_1 &= a_{19} + a_{25} \\
x_2 &= a_{20} + a_{24} \\
x_3 &= a_{21} + m_5 \\
x_4 &= a_{22} + a_{23} \\
x_5 &= a_{22} - a_{23} \\
x_6 &= a_{21} - m_5 \\
x_7 &= a_{20} - a_{24} \\
x_8 &= a_{19} - a_{25}
\end{aligned}
$$

D.4 Optimized Efficient 3/9-Point DCT-II and DCT-III Modules

The optimized efficient 3/9-point DCT-II and DCT-III modules are, respectively, derived from the corresponding 3/9-point forward and inverse RDFT modules. There exists a relation between any odd-length DFT and DCT-II of the same length [28]. If N is an odd integer, then the DCT-II of length N can be computed by simply permuting the input data sequence $\{x_n\}$ using the index mapping defined as

$$\hat{x}_n = \begin{cases} x_{(-1)^{n+(N+1)/2+1}\, n+\frac{N-1}{2}}, & n = 0, 1, \ldots, \frac{N-1}{2}, \\ x_{(-1)^{n+(N+1)/2+1}\, (N-n)+\frac{N-1}{2}}, & n = \frac{N+1}{2}, \frac{N+3}{2}, \ldots, N-1, \end{cases} \tag{D.1}$$

and performing the identical-length DFT of $\{\hat{x}_n\}$ with a DFT algorithm for real-valued inputs or equivalently with a RDFT algorithm. Then, the DCT-II coefficients can be extracted as

$$\begin{aligned} C_{2k} &= (-1)^k\, \Re e\, \{\hat{F}_k\}, \qquad k = 0, 1, \ldots, \frac{N-1}{2}, \\ C_{N-2k} &= (-1)^{k+1}\, \Im m\, \{\hat{F}_k\}, \qquad k = 1, 2, \ldots, \frac{N-1}{2}. \end{aligned} \tag{D.2}$$

This relation enables us to derive odd-length DCT-II algorithms from the corresponding forward odd-length RDFT algorithms using the above equations. A similar relation holds between an odd-length DCT-III, being inverse of the DCT-II, and the inverse RDFT of the same length. Thus, the odd-length DCT-III algorithms can be derived from the corresponding inverse odd-length RDFT algorithms by using the inverse index mapping implied by above equations.

D.4.1 3-Point DCT-II Module: 1 Mult, 4 Adds, and 1 Shift

The input data sequence is $\{x_0, x_1, x_2\}$ and the output data sequence is $\{C_0, C_1, C_2\}$.

$$a_1 = x_0 + x_2 \;\; a_2 = x_0 - x_2$$

$$m_1 = d_2\; a_2 \;\; s_1 = -\tfrac{1}{2}\; a_1$$

$$\begin{aligned} C_0 &= x_1 + a_1 \\ C_1 &= m_1 \\ C_2 &= -x_1 - s_1 \end{aligned}$$

D.4.2 9-point DCT-II Module: 8 Mults, 34 Adds, and 2 Shifts

The input data sequence is $\{x_0, x_1, \ldots, x_8\}$ and the output data sequence is $\{C_0, C_1, \ldots, C_8\}$.

$$
\begin{array}{llll}
a_1 = x_3 + x_5 & a_2 = x_3 - x_5 & a_3 = x_6 + x_2 & a_4 = x_6 - x_2 \\
a_5 = x_1 + x_7 & a_6 = x_1 - x_7 & a_7 = x_8 + x_0 & a_8 = x_8 - x_0 \\
a_9 = x_4 + a_5 & a_{10} = a_1 + a_3 & a_{11} = a_{10} + a_7 & a_{12} = a_3 - a_7 \\
a_{13} = a_1 - a_7 & a_{14} = a_1 - a_3 & a_{15} = a_2 - a_4 & a_{16} = a_{15} + a_8 \\
a_{17} = a_4 + a_8 & a_{18} = a_2 - a_8 & a_{19} = a_2 + a_4 &
\end{array}
$$

$$
\begin{array}{llll}
m_1 = -d_2\, a_6 & m_2 = -d_3\, a_{12} & m_3 = -d_4\, a_{13} & m_4 = -d_5\, a_{14} \\
m_5 = -d_2\, a_{16} & m_6 = -d_6\, a_{17} & m_7 = -d_7\, a_{18} & m_8 = -d_8\, a_{19} \\
s_1 = \frac{1}{2}\, a_5 & s_2 = \frac{1}{2}\, a_{11} & &
\end{array}
$$

$$
\begin{array}{llll}
a_{20} = x_4 - s_1 & a_{21} = a_{20} + m_2 & a_{22} = a_{20} - m_2 & a_{23} = a_{20} + m_3 \\
a_{24} = m_1 + m_6 & a_{25} = m_1 - m_6 & a_{26} = m_1 + m_7 &
\end{array}
$$

$$
\begin{aligned}
C_0 &= a_9 + a_{11} \\
C_1 &= m_8 - a_{26} \\
C_2 &= m_4 - a_{21} \\
C_3 &= m_5 \\
C_4 &= a_{22} - m_3 \\
C_5 &= a_{25} - m_7 \\
C_6 &= s_2 - a_9 \\
C_7 &= a_{24} + m_8 \\
C_8 &= a_{23} + m_4
\end{aligned}
$$

D.4.3 3-Point DCT-III Module: 1 Mult, 4 Adds, and 1 Shift

The input data sequence is $\{C_0, C_1, C_2\}$ and the output data sequence is $\{x_0, x_1, x_2\}$.

$$
m_1 = -d_2\, C_1 \quad s_1 = \frac{1}{2}\, C_2
$$

$$
a_1 = C_0 - C_2 \quad a_2 = C_0 + s_1
$$

$$
\begin{aligned}
x_0 &= a_2 - m_1 \\
x_1 &= a_1 \\
x_2 &= a_2 + m_1
\end{aligned}
$$

D.4.4 9-Point DCT-III Module: 8 Mults, 34 Adds, and 2 Shifts

The input data sequence is $\{C_0, C_1, \ldots, C_8\}$ and the output data sequence is $\{x_0, x_1, \ldots, x_8\}$.

$$
\begin{aligned}
&a_1 = C_0 - C_6 \quad a_2 = C_4 - C_2 \quad a_3 = a_2 + C_8 \quad a_4 = C_4 - C_8 \\
&a_5 = -C_2 - C_8 \; a_6 = -C_2 - C_4 \; a_7 = C_7 + C_5 \quad a_8 = a_7 - C_1 \\
&a_9 = -C_5 - C_1 \; a_{10} = C_7 + C_1 \quad a_{11} = C_7 - C_5
\end{aligned}
$$

$$
\begin{aligned}
&m_1 = -d_2 \; C_3 \; m_2 = -d_3 \; a_4 \; m_3 = -d_4 \; a_5 \quad m_4 = -d_5 \; a_6 \\
&m_5 = -d_2 \; a_8 \quad m_6 = -d_6 \; a_9 \; m_7 = -d_7 \; a_{10} \; m_8 = -d_8 \; a_{11} \\
&s_1 = -\tfrac{1}{2} \; C_6 \quad s_2 = \tfrac{1}{2} \; a_3
\end{aligned}
$$

$$
\begin{aligned}
&a_{12} = C_0 - s_1 \quad a_{13} = a_{12} + m_2 \; a_{14} = a_{12} - m_2 \; a_{15} = a_{12} + m_3 \\
&a_{16} = m_1 + m_6 \; a_{17} = m_1 - m_6 \; a_{18} = m_1 + m_7 \; a_{19} = a_{13} - m_4 \\
&a_{20} = a_{14} - m_3 \; a_{21} = a_1 - s_2 \quad a_{22} = a_{15} + m_4 \; a_{23} = a_{18} - m_8 \\
&a_{24} = m_7 - a_{17} \; a_{25} = a_{16} + m_8
\end{aligned}
$$

$$
\begin{aligned}
x_0 &= a_{22} - a_{23} \\
x_1 &= a_{21} + m_5 \\
x_2 &= a_{20} - a_{24} \\
x_3 &= a_{19} + a_{25} \\
x_4 &= a_1 + a_3 \\
x_5 &= a_{19} - a_{25} \\
x_6 &= a_{20} + a_{24} \\
x_7 &= a_{21} - m_5 \\
x_8 &= a_{22} + a_{23}
\end{aligned}
$$

D.5 Optimized Efficient 3/9-Point Scaled DCT-II (SDCT-II) Modules

The optimized efficient 3/9-point SDCT-II modules are derived from the corresponding 3/9-point DCT-II modules [31].

D.5.1 3-Point SDCT-II Module: 1 Mult, 4 Adds, and 1 Shift

The input data sequence is $\{x_0, x_1, x_2\}$ and the output data sequence is $\{C_0, C_1, C_2\}$.

$$
\begin{aligned}
&a_1 = x_0 + x_2 \; a_2 = x_0 - x_2 \\
&m_1 = d_9 \; a_2
\end{aligned}
$$

$$
\begin{aligned}
C_0 &= x_1 + a_1 \\
C_1 &= m_1 \\
C_2 &= a_1 - 2\,x_1
\end{aligned}
$$

D.5.2 9-Point SDCT-II Module: 8 Mults, 34 Adds, and 2 Shifts

The input data sequence is $\{x_0, x_1, \ldots, x_8\}$ and the output data sequence is $\{C_0, C_1, \ldots, C_8\}$.

$$
\begin{aligned}
&a_1 = x_3 + x_5 \quad a_2 = x_3 - x_5 \quad a_3 = x_6 + x_2 \quad a_4 = x_6 - x_2 \\
&a_5 = x_1 + x_7 \quad a_6 = x_1 - x_7 \quad a_7 = x_8 + x_0 \quad a_8 = x_8 - x_0 \\
&a_9 = x_4 + a_5 \quad a_{10} = a_1 + a_3 \; a_{11} = a_{10} + a_7 \; a_{12} = a_3 - a_7 \\
&a_{13} = a_1 - a_7 \; a_{14} = a_1 - a_3 \; a_{15} = a_2 - a_4 \quad a_{16} = a_{15} + a_8 \\
&a_{17} = a_4 + a_8 \; a_{18} = a_2 - a_8 \; a_{19} = a_2 + a_4 \quad a_{20} = 2\,x_4 - a_5
\end{aligned}
$$

$$
\begin{aligned}
&m_1 = -d_9\,a_6 \quad m_2 = -d_{10}\,a_{12}\; m_3 = -d_{11}\,a_{13}\; m_4 = -d_{12}\,a_{14} \\
&m_5 = -d_9\,a_{16}\; m_6 = -d_{13}\,a_{17}\; m_7 = -d_{14}\,a_{18}\; m_8 = -d_{15}\,a_{19}
\end{aligned}
$$

$$
\begin{aligned}
&a_{21} = a_{20} + m_2 \; a_{22} = a_{20} - m_2 \; a_{23} = a_{20} + m_3 \; a_{24} = m_1 + m_6 \\
&a_{25} = m_1 - m_6 \quad a_{26} = m_1 + m_7
\end{aligned}
$$

$$
\begin{aligned}
C_0 &= a_9 + a_{11} \\
C_1 &= m_8 - a_{26} \\
C_2 &= m_4 - a_{21} \\
C_3 &= m_5 \\
C_4 &= a_{22} - m_3 \\
C_5 &= a_{25} - m_7 \\
C_6 &= a_{11} - 2\,a_9 \\
C_7 &= a_{24} + m_8 \\
C_8 &= a_{23} + m_4
\end{aligned}
$$

D.6 Optimized Efficient 3/9-Point Scaled DCT-IV (SDCT-IV) Modules

Scaled DCT-IV (SDCT-IV) of an input data sequence $\{x_n\}$ is defined as [29]

$$
c_k^{IV} = \sqrt{2} \sum_{m=0}^{N-1} x_n \cos\left[\frac{\pi}{4N}(2n+1)(2k+1)\right], \qquad k = 0, 1, \ldots, N-1. \tag{D.3}
$$

The optimized efficient 3/9-point SDCT-IV modules are derived from the radix-q DCT-IV algorithm [8, 29], where q is an odd positive integer. If the outputs of SDCT-IV modules are scaled by $\frac{\sqrt{2}}{2}$ the corresponding DCT-IV modules are obtained. Since the DCT-IV matrix is symmetric and self-inverse, the forward and inverse 3/9-point DCT-IV computation can be realized by the identical efficient module.

D.6.1 3-Point SDCT-IV Module: 1 Mult, 6 Adds, and 1 Shift

The input data sequence is $\{x_0, x_1, x_2\}$ and the output data sequence is $\{C_0, C_1, C_2\}$.

$$a_1 = x_0 + x_2 \; a_2 = x_0 - x_2 \; a_3 = x_1 + \tfrac{1}{2}\, a_2$$

$$m_1 = d_2 \, a_1$$

$$\begin{aligned} C_0 &= m_1 + a_3 \\ C_1 &= a_2 - x_1 \\ C_2 &= m_1 - a_3 \end{aligned}$$

D.6.2 9-Point SDCT-IV Module: 17 Mults, 53 Adds, and 3 Shifts

The input data sequence is $\{x_0, x_1, \ldots, x_8\}$ and the output data sequence is $\{C_0, C_1, \ldots, C_8\}$.

$$\begin{aligned} &a_1 = x_0 + x_8 \; a_2 = x_0 - x_8 \; a_3 = x_1 + x_7 \; a_4 = x_1 - x_7 \\ &a_5 = x_2 + x_6 \; a_6 = x_2 - x_6 \; a_7 = x_3 + x_5 \; a_8 = x_3 - x_5 \end{aligned}$$

$$\begin{aligned} &m_1 = d_{16} \, a_1 \quad a_9 = m_1 - a_2 \quad m_2 = d_8 \, a_9 \\ &a_{10} = a_1 - m_2 \; m_3 = d_{16} \, a_{10} \quad a_{11} = a_9 + m_3 \end{aligned}$$

$$m_4 = d_2 \, a_3 \quad a_{12} = m_4 + \tfrac{1}{2}\, a_4 \; s_1 = \tfrac{1}{2}\, a_{12} \quad a_{13} = s_1 - a_4$$

$$\begin{aligned} &m_5 = d_{17} \, a_5 \quad a_{14} = m_5 - a_6 \quad m_6 = d_6 \, a_{14} \\ &a_{15} = a_5 - m_6 \; m_7 = d_{17} \, a_{15} \quad a_{16} = a_{14} + m_7 \end{aligned}$$

$$\begin{aligned} &m_8 = d_{18} \, a_8 \quad a_{17} = a_7 - m_8 \; m_9 = d_7 \, a_{17} \\ &a_{18} = a_8 + m_9 \; m_{10} = d_{18} \, a_{18} \; a_{19} = a_{17} - m_{10} \end{aligned}$$

$$
\begin{array}{llll}
a_{20} = x_4 + a_{12} & a_{21} = a_{10} + a_{15} & a_{22} = a_{21} + a_{18} & a_{23} = x_4 - s_1 \\
a_{24} = a_{10} - a_{15} & a_{25} = a_{10} - a_{18} & a_{26} = a_{11} + a_{16} & a_{27} = a_{16} + a_{19} \\
a_{28} = a_{11} - a_{16} & a_{29} = a_{28} + a_{19} & a_{30} = a_{24} - a_{25} & a_{31} = a_{27} - a_{26}
\end{array}
$$

$$
\begin{array}{llll}
m_{11} = d_4\, a_{24} & m_{12} = d_5\, a_{25} & m_{13} = -d_3\, a_{30} & m_{14} = d_7\, a_{26} \\
m_{15} = d_8\, a_{31} & m_{16} = d_6\, a_{27} & m_{17} = d_2\, a_{29} &
\end{array}
$$

$$
\begin{array}{llll}
a_{32} = a_{20} + a_{22} & a_{33} = -a_{23} + m_{11} & a_{34} = a_{33} + m_{12} & a_{35} = a_{23} + m_{13} \\
a_{36} = a_{35} + m_{12} & a_{37} = -a_{20} + \frac{1}{2}\, a_{22} & a_{38} = a_{23} + m_{11} & a_{39} = a_{38} - m_{13} \\
a_{40} = a_{13} + m_{14} & a_{41} = a_{40} + m_{15} & a_{42} = a_{13} - m_{15} & a_{43} = a_{42} - m_{16} \\
a_{44} = -a_{13} + m_{14} & a_{45} = a_{44} - m_{16} &&
\end{array}
$$

$$
\begin{aligned}
C_0 &= a_{32} \\
C_1 &= a_{34} - a_{41} \\
C_2 &= a_{34} + a_{41} \\
C_3 &= a_{36} - a_{43} \\
C_4 &= a_{36} + a_{43} \\
C_5 &= a_{37} - m_{17} \\
C_6 &= a_{37} + m_{17} \\
C_7 &= a_{39} - a_{45} \\
C_8 &= a_{39} + a_{45}
\end{aligned}
$$

D.7 Efficient 15-Point PFA DCT-II/III Module

The efficient 15-point DCT-II module was derived in [2, 4] by the prime factor decomposition (PFA) method, and it is shown in Fig. D.1. Full lines represent transfer factors $+1$ while dashed lines represent transfer factors -1. Symbol $\circ$ represents addition. The corresponding 15-point PFA DCT-III module can be obtained by transposing the 15-point PFA DCT-II module, i.e., by its reversing and performing inverse operations.

The efficient 15-point PFA DCT-II module consists of the first stage of three 5-point DCT-II modules in parallel and the second stage of five 3-point DCT-II modules in parallel followed by the post-butterfly stage. The outputs and inputs between stages of 5-point and 3-point DCT-II modules are interconnected according to the following relations:

$$(In)_{3n} = (Out)_n, \quad (In)_{3n+1} = (Out)_{5+n}, \quad (In)_{3n+2} = (Out)_{10+n}, \quad n = 0, 1, 2, 3, 4.$$

No twiddle factors are required among stages. The required efficient 3-point DCT-II [3, 28] and DCT-III modules additionally optimized together with their arithmetic complexity are presented in Appendix D.4. Similarly, the required efficient 5-point

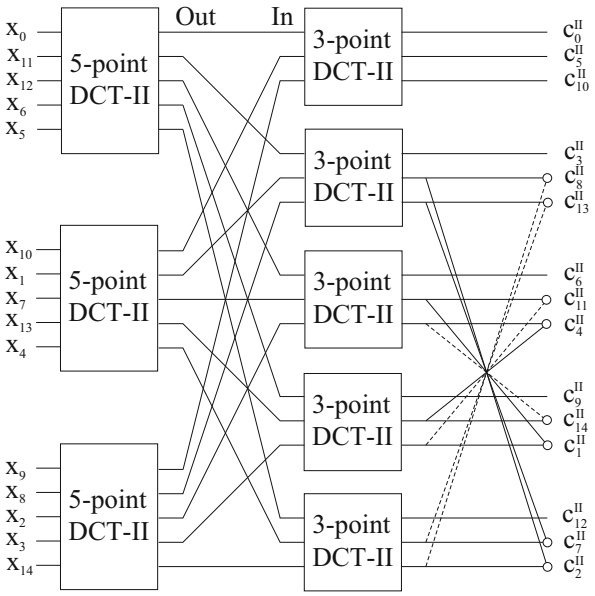

Fig. D.1 Efficient 15-point PFA DCT-II module [2, 4]

DCT-II [3, 28] and DCT-III modules additionally optimized together with their arithmetic complexity are presented below. The arithmetic complexity of efficient 15-point PFA DCT-II/DCT-III module is given by 17 multiplications, 67 additions, and 8 shift operations.

D.7.1 Optimized Efficient 5-Point DCT-II Module: 4 Mults, 13 Adds, and 1 Shift

The input data sequence is $\{x_0, x_1, x_2, x_3, x_4\}$ and the output data sequence is $\left\{c_0^{II}, c_1^{II}, c_2^{II}, c_3^{II}, c_4^{II}\right\}$. Note that the multiplication factor $\frac{1}{4} + \cos\frac{2\pi}{5} = \frac{\sqrt{5}}{4}$.

$$
\begin{aligned}
a_1 &= x_0 + x_4 & a_2 &= x_0 - x_4 \\
a_3 &= x_1 + x_3 & a_4 &= x_1 - x_3 \\
a_5 &= a_1 + a_3 & a_6 &= a_1 - a_3 \\
a_7 &= a_2 + a_4 & a_8 &= x_2 - \tfrac{1}{4}\, a_5
\end{aligned}
$$

$$
\begin{aligned}
m_1 &= a_6 \left(\tfrac{1}{4} + \cos\tfrac{2\pi}{5}\right) \\
m_2 &= a_7 \cos\tfrac{\pi}{10} \\
m_3 &= a_4 \left(\cos\tfrac{\pi}{10} - \cos\tfrac{3\pi}{10}\right) \\
m_4 &= a_2 \left(\cos\tfrac{\pi}{10} + \cos\tfrac{3\pi}{10}\right)
\end{aligned}
$$

D.7.2 Optimized Efficient 5-Point DCT-III Module: 4 Mults, 13 Adds, and 1 Shift

The input data sequence is $\left\{\overset{II}{c}_0, \overset{II}{c}_1, \overset{II}{c}_2, \overset{II}{c}_3, \overset{II}{c}_4\right\}$ and the output data sequence is $\{x_0, x_1, x_2, x_3, x_4\}$.

$$
\begin{aligned}
a_1 &= \overset{II}{c}_2 + \overset{II}{c}_4 & a_2 &= \overset{II}{c}_2 - \overset{II}{c}_4 \\
a_3 &= \overset{II}{c}_1 + \overset{II}{c}_3 & a_4 &= \overset{II}{c}_0 + \tfrac{1}{4}\, a_2
\end{aligned}
$$

$$
\begin{aligned}
m_1 &= a_1 \ \left(\tfrac{1}{4} + \cos\tfrac{2\pi}{5}\right) \\
m_2 &= a_3 \ \cos\tfrac{\pi}{10} \\
m_3 &= \overset{II}{c}_3 \ \left(\cos\tfrac{\pi}{10} - \cos\tfrac{3\pi}{10}\right) \\
m_4 &= \overset{II}{c}_1 \ \left(\cos\tfrac{\pi}{10} + \cos\tfrac{3\pi}{10}\right)
\end{aligned}
$$

$$
\begin{aligned}
a_5 &= m_2 - m_3 & a_6 &= m_4 - m_2 \\
a_7 &= a_4 + m_1 & a_8 &= a_4 - m_1
\end{aligned}
$$

$$
\begin{aligned}
x_0 &= a_7 + a_5 \\
x_1 &= a_8 + a_6 \\
x_2 &= \overset{II}{c}_0 - a_2 \\
x_3 &= a_8 - a_6 \\
x_4 &= a_7 - a_5
\end{aligned}
$$

D.8 Efficient 15-Point WFTA DCT-II/III Module

From Appendix D.4 it is known that the odd q-length DCT-II and DCT-III modules can be derived, respectively, from the corresponding odd q-length forward and inverse RDFT modules. We recall that there exists a relation between any odd q-length DFT and DCT-II of the same length [28]. This relation enables us to derive q-length DCT-II algorithms from the corresponding odd q-length forward RDFT algorithms. If N is an odd integer, and hence when $q = 15$, then a 15-point DCT-II can be derived by simply permuting the input data sequence defined by (D.1) and applying an identical-length RDFT algorithm for real-valued inputs. The DCT-II coefficients are extracted according to (D.2).

A 15-point DFT can be efficiently implemented by the Winograd Fourier transform algorithm (WFTA) [12] which uses efficient Winograd's prime-length 3- and 5-point DFT modules in a prime factor mapping. Combining the permutation defined by (D.1) and 15-point WFTA DFT algorithm, an efficient 15-point WFTA DCT-II module has been derived [14, 15, 36]. The corresponding sparse matrix

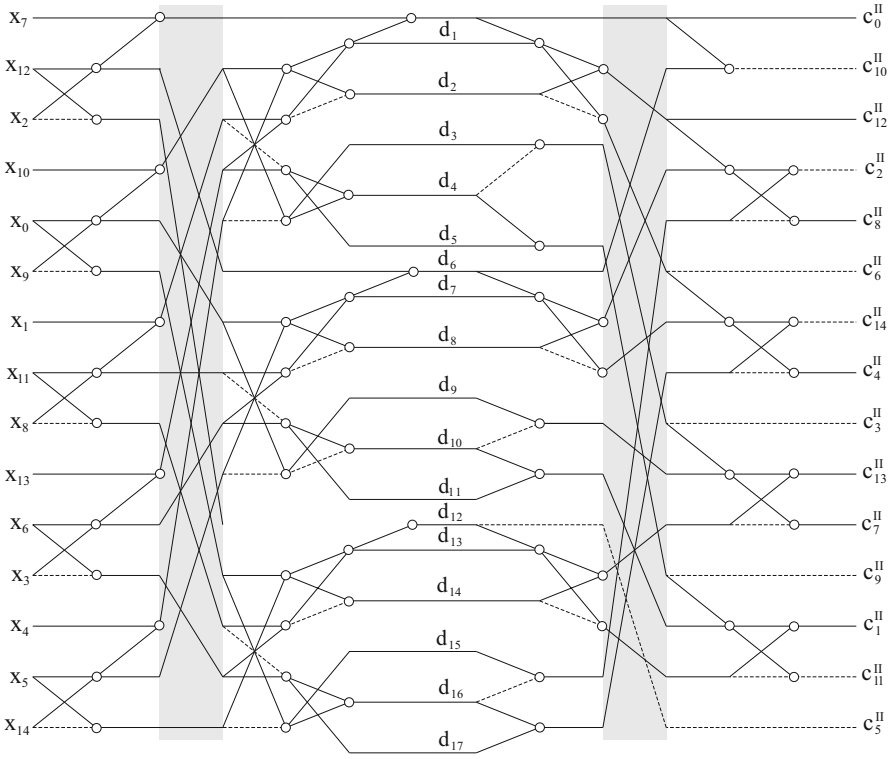

Fig. D.2 Efficient 15-point WFTA DCT-II module [14, 15]

factorization of 15×15 DFT matrix in detail is presented in [15]. The corresponding signal flow graph of efficient 15-point WFTA DCT-II module is shown in Fig. D.2. Full lines represent transfer factors $+1$ while dashed lines represent transfer factors -1. Symbol $\circ$ represents addition. The arithmetic complexity of 15-point WFTA DCT-II module is given by 17 multiplications and 67 additions. The 15-point WFTA DCT-III module can be obtained by transposing the 15-point WFTA DCT-II module, i.e., by its reversing and performing inverse operations.

Twiddle factors $\{d_i\}$, $i = 1, 2, \ldots, 17$ in the signal flow graph of efficient 15-point WFTA DCT-II module in Fig. D.2 are defined as follows [14, 15]:

$$u = -\frac{2\pi}{5}, \qquad v = -\frac{2\pi}{3},$$

$$\begin{aligned}
&d_1 = \tfrac{1}{2}(\cos u + \cos 2u) - 1, && d_2 = \tfrac{1}{2}(\cos u - \cos 2u), && d_3 = \sin u + \sin 2u,\\
&d_4 = \sin 2u, && d_5 = \sin u - \sin 2u, && d_6 = \cos v - 1,\\
&d_7 = d_1\, d_6, && d_8 = d_2\, d_6, && d_9 = d_3\, d_6,\\
&d_{10} = d_4\, d_6, && d_{11} = d_5\, d_6, && d_{12} = \sin v,\\
&d_{13} = d_1\, d_{12}, && d_{14} = d_2\, d_{12}, && d_{15} = -\, d_3\, d_{12},\\
&d_{16} = -\, d_4\, d_{12}, && d_{17} = -\, d_5\, d_{12}.
\end{aligned}$$

In [36] it has been shown that among 17 twiddle factors $\{d_i\}$, 3 are dyadic rational numbers of the form

$$d_1 = -\frac{5}{4},\ d_6 = -\frac{3}{2},\ d_7 = \frac{15}{8}.$$

Thus, efficient 15-point WFTA DCT-II/III module requires 14 nontrivial irrational multiplications and 67 additions. Multiplications by dyadic rationals can be implemented only by addition and shift operations [11].

Appendix E
Optimized Efficient Short-Length Forward/Backward MDCT Modules

E.1 Optimized Efficient 4/2-Point Forward/Backward MDCT Modules

The optimized efficient $4/2$-point forward/backward MDCT modules are derived from the DCT-IV-based fast MDCT algorithm [9]. Note that $\cos\frac{\pi}{8} + \sin\frac{\pi}{8} = \sqrt{2}\,\cos\frac{\pi}{8}$ and $\cos\frac{\pi}{8} - \sin\frac{\pi}{8} = \sqrt{2}\,\sin\frac{\pi}{8}$.

E.1.1 Forward 4-Point MDCT Module: 3 Mults and 5 Adds

The input data sequence is $\{x_0, x_1, x_2, x_3\}$ and the output data sequence is $\{c_0, c_1\}$.

$$\begin{array}{lll} a_1 = x_0 - x_1 & a_2 = -x_2 - x_3 & a_3 = a_1 + a_2 \\ m_1 = -a_1\,\sqrt{2}\,\sin\frac{\pi}{8} & m_2 = a_2\,\sqrt{2}\,\cos\frac{\pi}{8} & m_3 = a_3\,\cos\frac{\pi}{8} \end{array}$$

$$\begin{aligned} c_0 &= m_1 + m_3 \\ c_1 &= m_2 - m_3 \end{aligned}$$

E.1.2 Backward 2-Point MDCT Module: 3 Mults and 3 Adds

The input data sequence is $\{c_0, c_1\}$ and the output time domain aliased data sequence is $\{\hat{x}_0, \hat{x}_1, \hat{x}_2, \hat{x}_3\}$.

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1

$$a_1 = c_0 - c_1$$
$$m_1 = -c_0\ \sqrt{2}\ \sin\tfrac{\pi}{8}\ m_2 = -c_1\ \sqrt{2}\ \cos\tfrac{\pi}{8}\ m_3 = a_1\ \cos\tfrac{\pi}{8}$$

$$\begin{aligned}\hat{x}_0 &= \ \ m_1 + m_3\\ \hat{x}_1 &= -\hat{x}_0\\ \hat{x}_2 &= \ \ m_2 - m_3\\ \hat{x}_3 &= \ \ \hat{x}_2\end{aligned}$$

E.2 Optimized Efficient 6/3-Point Forward/Backward MDCT Modules

The optimized efficient 6/3-point forward/backward MDCT modules are derived directly from the MDCT matrix-vector representation [42].

E.2.1 Forward 6-Point MDCT Module: 1 Mult, 6 Adds, and 1 Shift

The input data sequence is $\{x_0, x_1, x_2, x_3, x_4, x_5\}$ and the output data sequence is $\{c_0, c_1, c_2\}$.

$$a_1 = x_0 - x_2 \quad a_2 = x_3 + x_5$$
$$a_3 = \tfrac{1}{2}\ a_1 - x_4\ m_1 = a_2\ \cos\tfrac{\pi}{6}$$

$$\begin{aligned}c_0 &= \ \ a_3 - m_1\\ c_1 &= -a_1 - x_4\\ c_2 &= \ \ a_3 + m_1\end{aligned}$$

E.2.2 Backward 3-Point MDCT Module: 1 Mult, 4 Adds, and 1 Shift

The input data sequence is $\{c_0, c_1, c_2\}$ and the output time domain aliased data sequence is $\{\hat{x}_0, \hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4, \hat{x}_5\}$.

$$a_1 = c_0 + c_2 \qquad a_2 = c_0 - c_2$$
$$m_1 = -\ a_2\ \cos\tfrac{\pi}{6}$$

$$
\begin{aligned}
\hat{x}_0 &= \tfrac{1}{2}\, a_1 - c_1 \\
\hat{x}_1 &= 0 \\
\hat{x}_2 &= -\hat{x}_0 \\
\hat{x}_3 &= m_1 \\
\hat{x}_4 &= -a_1 - c_1 \\
\hat{x}_5 &= \hat{x}_3
\end{aligned}
$$

E.3 Optimized Efficient 18/9-Point Forward/Backward MDCT Modules

Based on the generalized fast mixed–radix MDCT algorithm for composite lengths $2^n \times 9$ [25], the correct optimized efficient $18/9$-point forward/backward MDCT modules are derived directly from the forward/backward MDCT matrix-vector representation.

E.3.1 Forward 18-Point MDCT Module: 8 Mults, 42 Adds, and 2 Shifts

The input data sequence is $\{x_0, x_1, \ldots, x_{17}\}$ and the output data sequence is $\{c_0, c_1, \ldots, c_8\}$.

$$
\begin{aligned}
&a_1 = x_0 - x_8 \qquad a_2 = x_1 - x_7 \qquad a_3 = x_2 - x_6 \qquad a_4 = x_3 - x_5 \\
&a_5 = x_9 + x_{17} \qquad a_6 = x_{10} + x_{16} \qquad a_7 = x_{11} + x_{15} \qquad a_8 = x_{12} + x_{14}
\end{aligned}
$$

$$
\begin{aligned}
&a_9 = a_1 + a_3 \qquad a_{10} = a_9 + a_8 \\
&a_{11} = a_4 - a_5 \qquad a_{12} = a_{11} + a_7 \\
&a_{13} = a_2 + x_{13} \qquad a_{14} = \tfrac{1}{2}\, a_2 - x_{13} \qquad a_{15} = \tfrac{1}{2}\, a_{12} + a_{13}
\end{aligned}
$$

$$
\begin{aligned}
&m_1 = a_6 \cos\tfrac{\pi}{6} \qquad m_2 = a_{10} \cos\tfrac{\pi}{6} \\
&m_3 = (a_1 - a_3) \sin\tfrac{4\pi}{9} \qquad m_4 = (a_1 - a_8) \sin\tfrac{2\pi}{9} \qquad m_5 = (a_3 - a_8) \sin\tfrac{\pi}{9} \\
&a_{16} = m_4 + m_5 - m_1 \qquad a_{17} = m_3 - m_4 - m_1 a_{18} = m_3 + m_5 + m_1
\end{aligned}
$$

$$
\begin{aligned}
&m_6 = (a_4 + a_5) \cos\tfrac{4\pi}{9} \qquad m_7 = (a_4 - a_7) \cos\tfrac{2\pi}{9} \qquad m_8 = (a_5 + a_7) \cos\tfrac{\pi}{9} \\
&a_{19} = m_6 - m_8 + a_{14} \qquad a_{20} = m_7 + m_8 + a_{14} \qquad a_{21} = m_6 + m_7 - a_{14}
\end{aligned}
$$

$$
\begin{aligned}
c_0 &= a_{16} + a_{19} \\
c_1 &= -m_2 - a_{15} \\
c_2 &= -a_{17} + a_{20} \\
c_3 &= a_{18} - a_{21} \\
c_4 &= a_{12} - a_{13} \\
c_5 &= -a_{18} - a_{21} \\
c_6 &= a_{17} + a_{20} \\
c_7 &= m_2 - a_{15} \\
c_8 &= -a_{16} + a_{19}
\end{aligned}
$$

E.3.2 Backward 9-Point MDCT Module: 8 Mults, 34 Adds, and 2 Shifts

The input data sequence is $\{c_0, c_1, \ldots, c_8\}$ and the output time domain aliased data sequence is $\{\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_{17}\}$.

$$
\begin{array}{llll}
a_1 = c_0 - c_8 & a_2 = c_1 - c_7 & a_3 = c_2 - c_6 & a_4 = c_3 - c_5 \\
a_5 = c_0 + c_8 & a_6 = c_1 + c_7 & a_7 = c_2 + c_6 & a_8 = c_3 + c_5
\end{array}
$$

$$
\begin{array}{lll}
a_9 = a_1 - a_3 & a_{10} = a_9 - a_4 & \\
a_{11} = a_5 + a_7 & a_{12} = a_{11} + a_8 & \\
a_{13} = a_6 + c_4 & a_{14} = \frac{1}{2}\, a_6 - c_4 & a_{15} = \frac{1}{2}\, a_{12} - a_{13}
\end{array}
$$

$$
\begin{array}{ll}
m_1 = a_2 \cos\frac{\pi}{6} & m_2 = -\, a_{10} \cos\frac{\pi}{6}
\end{array}
$$

$$
\begin{array}{lll}
m_3 = (a_1 + a_3) \sin\frac{4\pi}{9} & m_4 = (a_1 + a_4) \sin\frac{2\pi}{9} & m_5 = (a_3 - a_4) \sin\frac{\pi}{9} \\
a_{16} = m_4 - m_5 & a_{17} = m_3 - m_4 & a_{18} = -m_3 + m_5 \\
m_6 = (a_5 - a_7) \cos\frac{4\pi}{9} & m_7 = (a_5 - a_8) \cos\frac{2\pi}{9} & m_8 = (a_7 - a_8) \cos\frac{\pi}{9} \\
a_{19} = m_6 + m_8 & a_{20} = -m_7 + m_8 & a_{21} = -m_6 - m_7
\end{array}
$$

$$
\begin{aligned}
\hat{x}_0 &= a_{16} - m_1 \\
\hat{x}_1 &= a_{15} \\
\hat{x}_2 &= a_{17} - m_1 \\
\hat{x}_3 &= a_{19} - a_{14} \\
\hat{x}_4 &= 0 \\
\hat{x}_5 &= -\hat{x}_3 \\
\hat{x}_6 &= -\hat{x}_2 \\
\hat{x}_7 &= -\hat{x}_1
\end{aligned}
$$

$$
\begin{aligned}
\hat{x}_8 &= -\hat{x}_0 \\
\hat{x}_9 &= \quad a_{20} + a_{14} \\
\hat{x}_{10} &= \quad m_2 \\
\hat{x}_{11} &= \quad a_{21} - a_{14} \\
\hat{x}_{12} &= \quad a_{18} - m_1 \\
\hat{x}_{13} &= -a_{12} - a_{13} \\
\hat{x}_{14} &= \quad \hat{x}_{12} \\
\hat{x}_{15} &= \quad \hat{x}_{11} \\
\hat{x}_{16} &= \quad \hat{x}_{10} \\
\hat{x}_{17} &= \quad \hat{x}_9
\end{aligned}
$$

Appendix F
Efficient Implementations of Givens–Jacobi Rotations

F.1 Definitions

A 2×2 orthogonal matrix denoted as $\boldsymbol{G}_{\varphi}$ is called Givens–Jacobi rotation, if it has the form [24]:

$$\boldsymbol{G}_{\varphi} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}, \quad \boldsymbol{G}_{\varphi}^{-1} = \boldsymbol{G}_{\varphi}^{T} = \boldsymbol{G}_{-\varphi}, \tag{F.1}$$

where $\boldsymbol{G}_{\varphi}$ and $\boldsymbol{G}_{-\varphi}$ are inverses to each other. T denotes matrix transposition.

F.2 LUL and ULU Computational Structures

The first method for an efficient implementation of Givens–Jacobi rotations is based on $\boldsymbol{LUL}$ matrix factorization of $\boldsymbol{G}_{\varphi}$, where $\boldsymbol{L}$ is a unit lower triangular and $\boldsymbol{U}$ is a unit upper triangular matrix. Provided that the determinant of $\boldsymbol{G}_{\varphi}$ and $\boldsymbol{G}_{-\varphi}$ is unity, i.e., $\det(\boldsymbol{G}_{\varphi}) = \det(\boldsymbol{G}_{-\varphi}) = +1$, then applying the $\boldsymbol{PLUS}$ factorization algorithm (see Appendix A.6) to $\boldsymbol{G}_{\varphi}$ (in this case $\boldsymbol{P} = \boldsymbol{I}$ is the identity matrix) we get $\boldsymbol{LUS}$ and hence $\boldsymbol{LUL}$ factorizations of $\boldsymbol{G}_{\varphi}$ and $\boldsymbol{G}_{-\varphi}$ defined as [11]

$$\boldsymbol{G}_{\varphi} = \begin{pmatrix} 1 & 0 \\ \tan\frac{\varphi}{2} & 1 \end{pmatrix} \begin{pmatrix} 1 & -\sin\varphi \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \tan\frac{\varphi}{2} & 1 \end{pmatrix}, \tag{F.2}$$

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1

$$\boldsymbol{G}_{-\varphi} = \begin{pmatrix} 1 & 0 \\ -\tan\frac{\varphi}{2} & 1 \end{pmatrix} \begin{pmatrix} 1 & \sin\varphi \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\tan\frac{\varphi}{2} & 1 \end{pmatrix}. \qquad \sin\varphi \neq 0. \tag{F.3}$$

Since $\boldsymbol{G}_{\varphi} = \boldsymbol{G}_{-\varphi}^{T}$ from (F.2) and (F.3) we obtain the alternative ***ULU*** factorizations of $\boldsymbol{G}_{\varphi}$ and $\boldsymbol{G}_{-\varphi}$ given by

$$\boldsymbol{G}_{\varphi} = \begin{pmatrix} 1 & -\tan\frac{\varphi}{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \sin\varphi & 1 \end{pmatrix} \begin{pmatrix} 1 & -\tan\frac{\varphi}{2} \\ 0 & 1 \end{pmatrix}, \quad \sin\varphi \neq 0. \tag{F.4}$$

$$\boldsymbol{G}_{-\varphi} = \begin{pmatrix} 1 & \tan\frac{\varphi}{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\sin\varphi & 1 \end{pmatrix} \begin{pmatrix} 1 & \tan\frac{\varphi}{2} \\ 0 & 1 \end{pmatrix}. \tag{F.5}$$

Although $\boldsymbol{G}_{\varphi}$ and $\boldsymbol{G}_{-\varphi}$ are eigenorthogonal, the factored matrices on the right-hand sides of (F.2), (F.3), (F.4), and (F.5), the so-called Gauss elementary matrices being unit lower and unit upper triangular matrices, are not longer orthogonal; however, they are still invertible. ***LUL*** and ***ULU*** factorizations of $\boldsymbol{G}_{\varphi}$ and $\boldsymbol{G}_{-\varphi}$ define the corresponding ***LUL*** and ***ULU*** computational structures, respectively, for the efficient implementation of forward and inverse Givens–Jacobi rotations. We note that from computational point of view, ***LUL*** and ***ULU*** structures are equivalent. The ***LUL*** computational structure for the efficient implementation of $\mathbf{y} = \boldsymbol{G}_{\varphi}\,\mathbf{x}$ and $\mathbf{x} = \boldsymbol{G}_{-\varphi}\,\mathbf{y}$, where $\mathbf{x} = [x_0, x_1]^T$ is the input vector and $\mathbf{y} = [y_0, y_1]^T$ is rotated vector, is shown in Fig. F.1.

The products of Givens–Jacobi rotations have the following useful properties

$$\boldsymbol{G}_{\alpha}\,\boldsymbol{G}_{\beta} = \boldsymbol{G}_{\beta}\,\boldsymbol{G}_{\alpha} = \begin{pmatrix} \cos(\alpha+\beta) & -\sin(\alpha+\beta) \\ \sin(\alpha+\beta) & \cos(\alpha+\beta) \end{pmatrix} = \boldsymbol{G}_{\alpha+\beta},$$

$$\boldsymbol{G}_{-\alpha}\,\boldsymbol{G}_{\beta} = \boldsymbol{G}_{\beta}\,\boldsymbol{G}_{-\alpha} = \boldsymbol{G}_{-(\alpha-\beta)},$$

$$\boldsymbol{G}_{-\alpha}\,\boldsymbol{G}_{-\beta} = \boldsymbol{G}_{-\beta}\,\boldsymbol{G}_{-\alpha} = \boldsymbol{G}_{-(\alpha+\beta)}. \tag{F.6}$$

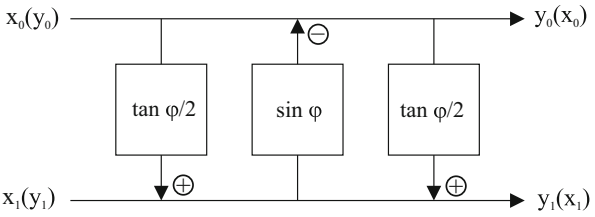

Fig. F.1 ***LUL*** computational structure for the efficient implementation of $\mathbf{y} = \boldsymbol{G}_{\varphi}\,\mathbf{x}$ and $\mathbf{x} = \boldsymbol{G}_{-\varphi}\,\mathbf{y}$

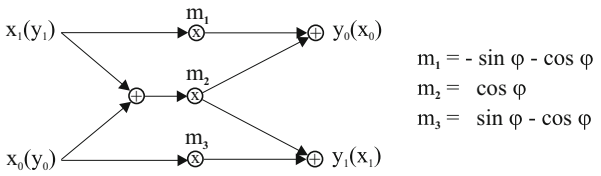

Fig. F.2 Bilinear computational structure for the efficient implementation of $\mathbf{y} = \boldsymbol{G}_{\varphi}\ \mathbf{x}$ and $\mathbf{x} = \boldsymbol{G}_{-\varphi}\ \mathbf{y}$

F.3 Bilinear Computational Structure

An alternative method to efficiently implement $\boldsymbol{G}_{\varphi}$ and $\boldsymbol{G}_{-\varphi}$ is to use the bilinear algorithm for the computation of 2-point Hankel matrix-vector product [33]. Exchanging columns in the matrix $\boldsymbol{G}_{\varphi}$ in (F.1) it becomes the Hankel matrix (see Appendix A.2) and therefore, the bilinear algorithm for 2-point Hankel matrix product can be applied to the reordered matrix $\boldsymbol{G}_{\varphi}$. The resulting bilinear computational structure for the efficient implementation of $\mathbf{y} = \boldsymbol{G}_{\varphi}\ \mathbf{x}$ and $\mathbf{x} = \boldsymbol{G}_{-\varphi}\ \mathbf{y}$ is shown in Fig. F.2.

LUL (***ULU***) and bilinear computational structures require 3 multiplications and 3 additions. However, while in the ***LUL*** (***ULU***) computational structure all the multiplications have to be performed sequentially, in the bilinear computational structure all the multiplications are independent and can be realized concurrently. Moreover, the ***LUL*** or ***ULU*** computational structure requires determinant of the matrix to be unity. Therefore, the scaled Givens–Jacobi rotations of the form $\alpha\ \boldsymbol{G}_{\varphi}$ and $\alpha\ \boldsymbol{G}_{-\varphi}$, where $\alpha \in R$ have to be realized by the bilinear computational structure.

F.4 Conversion of Complex Multiplication to the Givens–Jacobi Rotation

All twiddle factors in FFT algorithms are complex numbers with the unit magnitude, i.e., every twiddle factor can be expressed in the form:

$$e^{\mp i\,\varphi} = \cos\varphi\ \mp\ i\ \sin\varphi = c \mp is, \qquad c^2 + s^2 = 1,$$

where $i = \sqrt{-1}$. The complex multiplication $\mathbf{y} = (c \mp is)\ \mathbf{x}$, where $x = a + ib$, can be written in the equivalent matrix-vector notation as

$$\mathbf{y}^T = (c - is)\ \mathbf{x} = (1 \quad i) \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \mathbf{x}^T,$$

$$\mathbf{y}^T = (c + is)\ \mathbf{x} = (1 \quad i) \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \mathbf{x}^T, \qquad s \neq 0.$$

Thus, complex multiplications correspond to Givens–Jacobi rotations $\boldsymbol{G}_{\varphi}$ and $\boldsymbol{G}_{-\varphi}$ applied to the vector $\mathbf{x}$.

Appendix G
Symmetry/Anti-Symmetry and Periodicity/Anti-Periodicity of a Sequence (Function)

Special kinds of data sequences (functions), the symmetric and anti-symmetric sequences, the periodic and anti-periodic sequences, are fundamental notions in harmonic analysis, convolution, and correlation of signals [41].

G.1 Symmetry and Anti-Symmetry of a Sequence

A data sequence (finite or infinite) may be treated as samples taken from a continuous function. The symmetry or anti-symmetry of a data sequence reflects the symmetry of the function from which these samples are taken. Importantly, the symmetry or anti-symmetry of a data sequence depends on the choice of the symmetry center [41].

If one of the sampling points is chosen to be the symmetry center, the symmetry or anti-symmetry of a data sequence is referred to as odd symmetry or odd anti-symmetry. This symmetry type is referred to as odd symmetry type. Let $\{x_n\}$, $n = 0, 1, \ldots, M-1$, be a data sequence of the length M.

Definition 1 Sequence $\{x_n\}$ is said to be odd symmetric if $x_{-n} = x_n$.

Definition 2 Sequence $\{x_n\}$ is said to be odd anti-symmetric if $x_{-n} = -x_n$.

In the odd symmetry type, all sampling points are integers. As an example, the DFT is based on this symmetry type [41].

There is another symmetry type, called the even symmetry type. If the midpoint between two adjacent sampling points is chosen to be the symmetry center, the symmetry or anti-symmetry of a data sequence is referred to as even symmetry or even anti-symmetry. This type of symmetry is referred to as even symmetry type.

Definition 3 Sequence $\{x_n\}$ is said to be even symmetric if $x_{-n-1} = x_n$.

Definition 4 Sequence $\{x_n\}$ is said to be even anti-symmetric if $x_{-n-1} = -x_n$.

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1

In the even symmetry type, all sampling points are half integers. A number is called a half integer if it is the sum of an integer and $\frac{1}{2}$. Thus, there are two choices of symmetry type in both spatial and frequency domains [41].

G.2 Periodicity and Anti-Periodicity of a Sequence

Now, we recall the definitions of periodic and anti-periodic sequences. Let $\{x_n\}$, $n = 0, 1, \ldots, M-1$, be a data sequence of the length M.

Definition 5 A data sequence $\{x_n\}$ is called a periodic sequence if $x_{n+M} = x_n$, where $M > 0$ is called the period of periodic sequence $\{x_n\}$.

Definition 6 A data sequence $\{x_n\}$ is called an anti-periodic sequence if $x_{n+M} = -x_n$, where $M > 0$ is the period of anti-periodic sequence $\{x_n\}$.

An anti-periodic sequence $\{x_n\}$ may be treated as a periodic sequence with period $2M$ because $x_{n+2M} = -x_{n+M} = x_n$. However, the properties of $\{x_n\}$ depend only upon its values in one period M. The periodicity and anti-periodicity of sequences are closely related to their symmetry and anti-symmetry properties, respectively. Properties (sums and products) of periodic and anti-periodic sequences with a common period can be found in [41].

References

1. M. Bellanger, *Digital Processing of Signals: Theory and Practice*, 2nd edn. (Wiley, Chichester, NJ, 1989)
2. G. Bi, Index mapping for prime factor algorithm of discrete cosine transform. Electron. Lett. **35**(3), 198–200 (1999)
3. G. Bi, L.W. Yu, DCT algorithms for composite sequence lengths. IEEE Trans. Signal Process. **46**(3), 554–562 (1998)
4. G. Bi, Y. Zeng, *Transforms and Fast Algorithms for Signal Analysis and Representation*, Chapter 6 (Birkhäuser, Boston, 2004), pp. 207–245
5. G. Bonnerot, M. Bellanger, Odd-time odd-frequency discrete Fourier transform for symmetric real-valued series. Proc. IEEE **64**, 392–393 (1976)
6. V. Britanak, On the discrete cosine transform computation. Signal Process. **40**(2–3), 183–194 (1994)
7. V. Britanak, New universal rotation-based fast computational structures for an efficient implementation of the DCT-IV/DST-IV and analysis/synthesis MDCT/MDST filter banks. Signal Process. **89**(11), 2213–2232 (2009)
8. V. Britanak, Comments on fast radix-9 algorithm for the DCT-IV computation. IEEE Signal Process Lett.**16**(11), 1005–1006 (2009)
9. V. Britanak, H.J. Lincklaen Arriëns, Fast computational structures for an efficient implementation of the complete TDAC analysis/synthesis MDCT/MDST filter banks. Signal Process. **89**(7), 1379–1394 (2009)
10. V. Britanak, K.R. Rao, The fast generalized discrete Fourier transforms: A unified approach to the discrete sinusoidal transforms computation. Signal Process. **79**(12), 135–150 (1999)
11. V. Britanak, P. Yip, K.R. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations* (Academic Press Inc. Elsevier Science, Amsterdam, 2007)
12. C.S. Burrus, T.W. Parks, *DFT/FFT and Convolution Algorithms* (John Wiley, New York, 1985)
13. S.C. Chan, K.L. Ho, Direct methods for computing discrete sinusoidal transforms. IEE Proc. Part F: Radar and Signal Process. **137**(6), 433–442 (1990)
14. R.K. Chivukula, Y.A. Reznik, V. Devarajan, Efficient algorithms for MPEG-4 AAC-ELD, AAC-LD and AAC-LC filter banks, in *Proceedings of the IEEE International Conference on Audio, Language and Image Processing (ICALIP'2008)*, Shanghai, China, July 2008, pp. 1629–1634
15. R.K. Chivukula, Y.A. Reznik, Y. Hu, V. Devarajan, Fast algorithms for low-delay TDAC filter banks in MPEG-4 AAC-ELD. IEEE Trans. Audio Speech Lang. Process. **22**(12), 1701–1712 (2014)

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1

16. S. Cramer, R. Gluth, Computationally efficient real-valued filter banks based on a modified O^2 DFT, in *Proceedings of EUSIPCO'90, Signal Processing V: Theories and Applications* (Elsevier Science Publishers B. V., Barcelona, Spain, 1990), pp. 585–588
17. B.P. Demidovic, I.A. Maron, Chapter 7: Algebra of matrices, *Basics of Numerical Mathematics* (GIFML, Moscow, 1960) (in Russian)
18. P. Duhamel, B. Piron, J.M. Etcheto, On computing the inverse DFT. IEEE Trans. Acoust. Speech Signal Process. **36**(2), 285–286 (1988)
19. D.F. Elliott, K.R. Rao, *Fast Transforms: Algorithms, Analyses, Applications* (Academic Press, New York, 1982), pp. 128–131
20. D.K. Faddeev, V.N. Faddeeva, *Computational Methods of Linear Algebra* (GIFML, Moscow, 1963) (in Russian), (English translation: Dover Publications Inc., New York, 1959)
21. M. Fiedler, *Special Matrices and their Using in Numerical Mathematics* (SNTL, Prague, Czech Republic, 1981) (in Czech)
22. F.R. Gantmacher, *Theory of Matrices*, 2nd edn. (Nauka, Moscow, 1966) (in Russian), (English translation: Vols. 1 and 2, Chelsea, New York, 1959)
23. R. Gluth, Regular FFT-related transform kernels for DCT/DST-based polyphase filter banks, in *Proceedings of the IEEE ICASSP'91*, Toronto, Canada, May 1991, pp. 2205–2208
24. G.H. Golub, C.F. Van Loan, *Matrix Computations*, 3rd edn. (The Johns Hopkins University Press, Baltimore, MD, 1996)
25. Z.G. Gui, Y. Ge, D.Y. Zhang, J.S. Wu, Generalized fast mixed-radix algorithm for the computation of forward and inverse MDCTs. Signal Process. **92**(2), 363–373 (2012)
26. P. Hao, Customizable triangular factorizations of matrices. Linear Algebra Appl. **382**, 135–154 (2004)
27. P. Hao, Q. Shi, Matrix factorizations for reversible integer mapping. IEEE Trans. Signal Process. **49**(10), 2314–2324 (2001)
28. M.T. Heideman, Computation of an odd-length DCT from a real-valued DFT of the same length. IEEE Trans. Signal Process. **40**(1), 54–61 (1992)
29. H.W. Hsu, C.M. Liu, Fast radix-q and mixed-radix algorithms for type-IV DCT. IEEE Signal Process Lett. **15**(12), 910–913 (2008)
30. C.W. Kok, Fast algorithm for computing discrete cosine transform. IEEE Trans. Signal Process. **45**(3), 757–760 (1997)
31. S.W. Lee, Improved algorithm for efficient computation of the forward and backward MDCT in MPEG audio coder. IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process. **48**(10), 990–994 (2001)
32. H.S. Malvar, *Signal Processing with Lapped Transforms*, Chapter 2 (Artech House, Norwood, MA, 1992), pp. 71–75
33. V. Muddhasani, M.D. Wagh, Bilinear algorithms for discrete cosine transforms of prime lengths. Signal Process. **86**(9), 2393–2406 (2006)
34. H.J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms* (Springer, New York, 1981), pp. 144–148
35. N. Rama Murthy, M.N.S. Swamy, On the algorithms for the computation of even discrete cosine transform-2 (EDCT-2) of real sequences. IEEE Trans. Circuits Syst. **37**(5), 625–627 (1990)
36. Y. Reznik, R.K. Chivukula, Fast 15×15 transform for image and video coding applications, in *Proceedings of the IEEE Data Compression Conference (DCC'2009)*, Snowbird, UT, March 2009, p. 465
37. Y. She, P. Hao, On the necessity and sufficiency of PLUS factorizations. Linear Algebra Appl. **400**, 193–202 (2005)
38. H.V. Sorensen, D.L. Jones, M.T. Heideman, Real-valued fast Fourier transform algorithms. IEEE Trans. Acoust. Speech Signal Process. **35**(6), 849–863 (1987)
39. G.W. Stewart, *Matrix Algorithms, Volume I: Basic Decompositions* (SIAM Society for Industrial and Applied Mathematics, Philadephia, PA, 1998)

40. M. Vetterli, H.J. Nussbaumer, Simple FFT and DCT algorithms with reduced number of operations. Signal Process. **6**(4), 267–278 (1984)
41. Z. Wang, B.R. Hunt, The discrete W transform. Appl. Math. Comput. **16**, 19–48 (1985)
42. J.S. Wu, H.Z. Shu, L. Senhadji, L.M. Luo, Mixed-radix algorithm for the computation of forward and inverse MDCTs. IEEE Trans. Circuits Syst. I, Regul. Pap.**56**(4), 784–794 (2009)
43. I.S. Yohvidov, *Hankel and Toeplitz Matrices and Forms: Algebraic Theory*, Chapter II (NAUKA, Moscow, 1974), (English translation: Birkhäuser, Boston, 1982), pp. 53–95

Index

V. Britanak, K.R. Rao, *Cosine-/Sine-Modulated Filter Banks*,
DOI 10.1007/978-3-319-61080-1