

FAST ALGORITHMS FOR MDCT AND
LOW DELAY FILTERBANKS
USED IN AUDIO CODING

by

RAVI KIRAN CHIVUKULA

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2008

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to Prof. Venkat Devarajan, my thesis advisor, who has continuously supported me during the execution of this thesis. I have immensely benefited from his courses and advice. Thanks are due to Dr. Yuriy Reznik of Qualcomm Corp. R&D who has been my technical manager and mentor for this thesis. He has reviewed my work continuously and offered constructive criticism at various points of time. I would also like to thank Prof. K.R. Rao for extensively reviewing the thesis document and offering several insightful comments. I am also thankful to Prof. K.R. Rao and Prof. Soontorn Oraintara for serving on the thesis committee. The course work I took in the electrical engineering department of UTA has given me the necessary preparation to carry out this research. For this, I am indebted to the faculty of UTA EE department.

Finally, I would like to thank my mother who has always been a source of encouragement and support in my life.

February 29, 2008

ABSTRACT

FAST ALGORITHMS FOR MDCT AND LOW DELAY FILTERBANKS USED IN AUDIO CODING

Ravi Kiran Chivukula, M.S.

The University of Texas at Arlington, 2008

Supervising Professor: Venkat Devarajan

Modern audio and speech coding systems use filterbank and transform coding techniques to decorrelate the input signal. Cosine-modulated, perfect reconstruction, pseudo-QMF filterbanks are most commonly used for this purpose. In this thesis, we present three propositions. First, a fast algorithm for modified discrete cosine transform (MDCT) for transform lengths of the form 5×2^m and 15×2^m is presented. This algorithm is based on mapping the MDCT to DCT-II via DCT-IV and using the involutory property of the DCT-IV matrix. This leads to a reduction in the number of multiplications and constant memory requirement. The algorithm also uses very

efficient DCT-II modules for transform lengths of 5 and 15 which are derived from the Winograd Fourier Transform Algorithm. Second, the newly introduced MPEG-4 AAC Enhanced Low Delay filterbanks are mapped to MDCT. The mapping involves just input and output permutations, sign changes and additions. Since many fast algorithms exist for MDCT, this mapping essentially provides a fast algorithm for the new filterbanks. Third, we present a radix-5 decomposition for DCT-II useful for MDCT of length 5×2^m . This decomposition is useful for improving the precision of the fixed-point implementations of the algorithms. Complexity analysis is provided for all the algorithms and comparisons are made with existing algorithms.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	ii
ABSTRACT	iii
LIST OF ILLUSTRATIONS.....	viii
LIST OF TABLES.....	ix
LIST OF ACRONYMS	x
Chapter	Page
1. INTRODUCTION.....	1
1.1 Overview.....	1
1.2 Elements of an Audio Codec	2
1.3 The Need for a Class of New Fast Algorithms	4
1.4 Objectives of Efficient Algorithms.....	5
1.5 Organization of the Thesis	5
2. FAST MDCT ALGORITHMS.....	7
2.1 Introduction.....	7
2.2 Mapping MDCT to DCT-IV and DCT-II	9
2.3 Fast Algorithms for 5-point and 15-point DCT-II	14
2.3.1 Overview.....	14
2.3.2 Definitions... ..	15

2.3.3 Heideman's Mapping.....	16
2.3.4 DFT Algorithms for Real-Valued Input Data.....	17
2.3.5 Winograd Short-N DFT Modules.....	18
2.3.6 Prime Factor Mapping	20
2.3.7 Winograd Fourier Transform Algorithm (WFTA)	22
2.3.8 5-point DCT.....	25
2.3.9 15-point DCT.....	26
2.4 Modified Window Function.....	28
2.5 Complexity Analysis of the New MDCT Algorithm.....	30
3. MPEG-4 AAC ENHANCED LOW DELAY FILTERBANKS	32
3.1 Introduction.....	32
3.2 MPEG-4 AAC LD	33
3.3 MPEG-4 AAC Enhanced Low Delay Filterbanks.....	34
3.4 Mapping the ELD Filterbanks to MDCT.....	38
3.4.1 Definitions..	38
3.4.2 Mapping the ELD Analysis Filterbank to MDCT	40
3.4.3 Mapping the ELD Synthesis Filterbank to IMDCT	42
3.4.4 Complexity Analysis	45
3.4.4.1 Complexity of non-MDCT part of our algorithm	45
3.4.4.2 Complexity of MDCT.....	46
3.4.4.3 Complexity of our fast algorithm.....	46
4. RADIX-5 DECOMPOSITION OF DCT	48

4.1 Introduction.....	48
4.2 Radix-5 DCT.....	50
4.3 Complexity Analysis.....	53
5. CONCLUSIONS.....	55
Appendix	
A. MATLAB CODE FOR FAST 15-POINT DCT.....	56
B. ILLUSTRATION OF WFTA.....	60
REFERENCES	64
BIOGRAPHICAL INFORMATION.....	71

LIST OF ILLUSTRATIONS

Figure	Page
1.1 Block diagram of an audio encoder.....	2
1.2 Block diagram of an audio decoder.....	3
2.1 Mapping IMDCT to DCT-IV	11
2.2 Mapping DCT-IV to DCT-II.....	12
2.3 Kok's algorithm for a 10-point IDCT-II	14
2.4 Flow graph for 5-point DCT-II	26
2.5 Flow graph for 15-point DCT-II	27
2.6 Modified window function ($N = 640$)	29
3.1 Analysis and synthesis windows used in MPEG-4 AAC ELD core coder filterbanks	36
3.2 Sine window vs low delay window.....	37
3.3 Low overlap window vs low delay window.....	38
3.4 Implementation of ELD analysis filterbank by MDCT	42
3.5 Implementation of ELD synthesis filterbank by IMDCT	45

LIST OF TABLES

Table	Page
2.1 Comparison of proposed MDCT algorithm for $N = 640$	31
2.2 Comparison of proposed MDCT algorithm for $N = 1920$	31

LIST OF ACRONYMS

3GPP:	3 rd Generation Partnership Project
3GPP2:	3 rd Generation Partnership Project 2
AAC:	Advanced Audio Coding
AC-3:	Dolby Audio Codec 3
AMR-NB:	Adaptive Multi Rate – Narrow Band
AMR-WB:	Adaptive Multi Rate – Wide Band
CFA:	Common Factor Algorithm
CFM:	Common Factor Mapping
Codec:	Encoder-Decoder
DCT:	Discrete Cosine Transform
DCT-II:	Type II Discrete Cosine Transform
DCT-IV:	Type IV Discrete Cosine Transform
DFT:	Discrete Fourier Transform
ELD:	Enhanced Low Delay
ER:	Error Resilient
EVRC:	Enhanced Variable Rate Codec
EVRC-WB:	Enhanced Variable Rate Codec – Wide Band
EV-VBR:	Evolution-Variable Bit Rate
FFT:	Fast Fourier Transform

GSM:	Global System for Mobile Communications
IDCT:	Inverse Discrete Cosine Transform
IDCT-II:	Type II Inverse Discrete Cosine Transform
IEC:	International Engineering Consortium
IMDCT:	Inverse Modified Discrete Cosine Transform
ISO:	International Standards Organization
ITU:	International Telecommunication Union
LC:	Low Complexity
LD:	Low Delay
LPC:	Linear Predictive Coding
MDCT:	Modified Discrete Cosine Transform
MP3:	MPEG-1 Layer-3
MPEG:	Moving Pictures Expert Group
PFA:	Prime Factor Algorithm
PFM:	Prime Factor Mapping
QMF:	Quadrature Mirror Filter
RHS:	Right Hand Side
SBR:	Spectral Band Replication
WFTA:	Winograd Fourier Transform Algorithm

CHAPTER 1

INTRODUCTION

1.1 Overview

There has been a tremendous growth in the use of digital multimedia in the last two decades. Speech and audio content have always been an integral part of multimedia. This exponential growth in demand for digital audio has created requirements for high fidelity content at very low bit rates. Consequently, there has been much research in the past two decades on perceptually lossless low bit rate compression of speech and audio data. This has led to several international standards for speech and audio such as Adaptive Multi Rate (AMR) Narrow Band codec, AMR Wide Band codec and Enhanced Variable Rate Codec (EVRC) for speech and MPEG-1 Layer 3 (MP3), MPEG-4 Advanced Audio Coding (AAC) for audio; proprietary audio codecs such as Dolby AC-3, and an open source audio codec called Ogg Vorbis [1-6].

Speech codecs are based on vocal tract modeling for redundancy removal using a technique called Linear Predictive Coding (LPC) [7]. The human vocal tract is modeled as a time varying filter. The parameters of the filter (LPC coefficients) are transmitted or stored. Speech codecs are highly effective for single speaker material with the signals sampled at 8 kHz or 16 kHz. Their round-trip algorithmic delay is less than 30 ms making them useful for full-duplex communications purposes. On the other

hand, audio codecs achieve compression by exploiting the perceptual redundancies present in the audio signal [8, 51]. This perceptual redundancy is due to the limitations of the human auditory system. Audio and speech codecs operate on convenient blocks of the original signal called frames. To achieve good coding gain, audio codecs operate on longer frame lengths. In general, the audio signal can be assumed to be a quasi-stationary random process within each frame. However, occasionally there could be non-stationary parts to the signal where a shorter frame length allows better modeling. Therefore, audio codecs also use look-ahead buffers for detecting non-stationary parts of the signal and switch over to a shorter block length to increase the temporal resolution. Audio codecs are highly effective for any generic audio content but suffer from large algorithmic delays (due to long block lengths and look-ahead buffers) making them unsuitable for full-duplex communications.

1.2 Elements of an Audio Codec

A basic block diagram of an audio encoder is shown in Fig. 1.1.

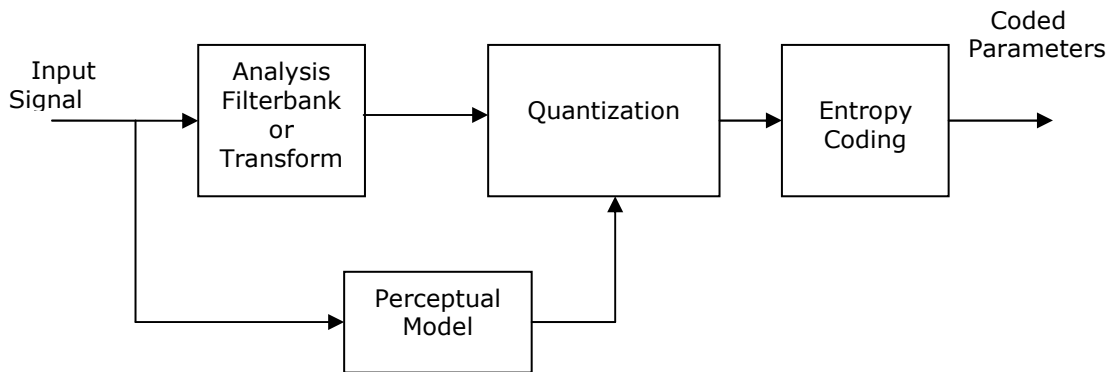


Fig. 1.1 Block diagram of an audio encoder

The analysis filterbank or the analysis transform splits the incoming signal into several frequency bands so as to exploit the statistical redundancies in the signal. Typically, cosine modulated, perfect reconstruction filterbanks such as the modified discrete cosine transform (MDCT) are used [4, 5, 26]. The perceptual model estimates the maximum inaudible amount of quantization noise that can be introduced in each sub band. The quantization and coding block does the actual quantization of the sub band samples based on the guidance information received from the perceptual model. The coded spectral coefficients are further entropy coded using Huffman codes, arithmetic codes etc. The parameters are then multiplexed into a bit stream which is either transmitted or stored.

The block diagram of an audio decoder is shown below in Fig 1.2.

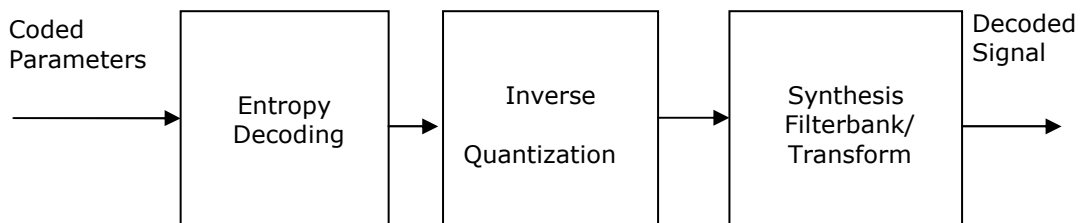


Fig 1.2 Block diagram of an audio decoder

In the decoder, the reverse process is followed. The bit stream parameters are demultiplexed from the bit stream, entropy decoded, inverse quantized and then the synthesis filterbank or the inverse transform is applied to produce the decoded signal.

In general, the most computationally intensive blocks in an audio codec are the analysis and synthesis transforms. Moreover, these codecs generally run on embedded platforms such as mobile phones which have limited computational power, memory

resources and battery power. Hence there is a distinct need for fast algorithms (detailed in the next section) for the implementation of these transforms.

In this thesis, we develop several novel fast algorithms for some of the most commonly used transforms in audio coding.

1.3 The Need for a Class of New Fast Algorithms

There has been a large amount of literature on fast algorithms for the standard filterbanks and transforms used in audio coding [9-15]. However, because of the new application scenarios, these algorithms are not directly applicable for some of the new codecs. The cases addressed in this thesis are described below:

1. MPEG has proposed new filterbanks for low delay communications [20-22]. This set of new tools or ‘profile’ is called MPEG-4 AAC ELD (Enhanced Low Delay). These filterbanks are different from the traditional MDCT in that they have longer overlap among frames than MDCT and, use a different prototype filter to reduce the introduced delay. It is therefore necessary to also have fast algorithms for these new filterbanks which do not currently exist.
2. Many of the fast algorithms for transforms such as MDCT are for powers-of-2 lengths [10-13]. In other words, the length of the transform is of the form 2^m . Because of MP3 codec, there also exist algorithms for lengths that have a factor of 3 [15,16]. However, the latest wide band speech codecs such as EVRC-Wide Band (EVRC-WB) and G.EV-VBR use transform lengths that have a factor of 5 [17-19] such as 160, 320

and 640. Moreover, because of the introduction of MPEG-4 AAC-ELD, transform lengths such as 960 and 480 have become more important. These lengths have a factor of 15. The existing algorithms are not directly applicable for these new cases.

3. Algorithms that are specific for powers-of-2 lengths have higher fixed-point precision than algorithms that are applicable for general lengths [23-24]. Hence, for fixed-point implementations it would be desirable to split the general length algorithm into several powers-of-2 length algorithms for improved precision and accuracy.

1.4 Objectives of Efficient Algorithms

In deriving fast algorithms in this thesis, the following objectives have been kept in mind:

1. Minimize the number of arithmetic operations – multiplications and additions. This will improve the battery life of the device.
2. Minimize data and code memory requirements. This will reduce the cost of the system.
3. Improve fixed-point precision of the existing algorithms.

1.5 Organization of the Thesis

In chapter 2, we derive fast algorithms for MDCT of lengths 5×2^m and 15×2^m . Complexity analysis and comparison figures are given at the end of the chapter. In chapter 3, we introduce the MPEG-4 AAC ELD filterbanks and then derive an algorithm that maps these filterbanks to MDCT. Complexity analysis is given at the end

of the chapter. In chapter 4, we derive a radix-5 decomposition of DCT. In chapter 5, we provide conclusions about the work and point to possible future work. Relevant Matlab code is provided in Appendices A and B.

CHAPTER 2

FAST MDCT ALGORITHMS

2.1 Introduction

The Modified Discrete Cosine Transform (MDCT) is widely used in speech and audio coding as analysis/synthesis filter bank with time domain alias cancellation property [31]. Since it is a lapped transform it is particularly useful in mitigating blocking artifacts that arise in audio coding because of quantization of the spectral coefficients. It usually represents one of the most computationally intensive parts of the codec and hence there exists a need for fast algorithms for implementing it.

This problem is well known, well studied, and numerous efficient algorithms have been proposed for solving it [10-16, 32]. Many of these proposed algorithms are derived for transforms of lengths $N = 2^m$. Among other transform sizes, ones including a factor of 3 (as prompted by the design of MP3 audio codec [3]) have also been thoroughly studied [15, 16]. However, fast algorithms for transforms of other sizes such as 5×2^m and 15×2^m do not exist, and engineers are often left with the necessity to modify or combine some of these techniques in their designs.

The need for transforms of sizes $N = 5 \times 2^m$ arises in the design of speech and audio codecs, which typically operate with sampling rates of 8 kHz or 16 kHz and have to use frames of only 10ms or 20ms duration. Examples of such algorithms include recently standardized ITU-T G.729.1, and 3GPP2 EVRC-WB vocoders [17, 18], and an

emerging ITU-T G.EV-VBR standard [19]. The need for transform sizes 15×2^m arises in the case of MPEG-4 AAC profiles. Transforms of sizes 960 and 480 are used in AAC-LC and AAC-ELD profiles respectively. To operate synchronously with speech codecs (which often have frame length of 160 samples), it is necessary to have frame lengths that are multiples of 160. This, combined with the requirement that the frame size be large enough for good frequency resolution, results in transform lengths of 960 and 480.

In this chapter, we present efficient algorithms for MDCT and IMDCT of sizes 5×2^m and 15×2^m ($m \geq 2$). Since MDCT is an orthogonal transform, the inverse transform can be obtained by transposing the flow graph of the forward transform and vice versa. Hence, the theory presented here is equally valid for both MDCT and IMDCT.

The rest of this chapter is organized as follows. In section 2.2, we explain the mapping of the MDCT into a DCT-IV and DCT-II with isolated pre/post-multiplications, allowing their subsequent absorption by the windowing stage. In section 2.3, we describe efficient implementations of 5-point and 15-point DCT-II algorithms, adopted in our MDCT design. The design of the merged window (because the window absorbs the pre/post multiplications in DCT-IV) is discussed in section 2.4. Finally, in section 2.5 we characterize the overall complexity of our proposed algorithms and compare it with other possible implementations. Parts of this chapter are also presented in [33].

2.2 Mapping MDCT to DCT-IV and DCT-II

Let $\{x(n)\}$, $n = 0, \dots, N-1$ represent an input sequence of samples (multiplied by the window), and let N denote the frame length. In this section, we consider MDCT and inverse MDCT (IMDCT), defined respectively as follows:

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi}{2N} \left(2n+1 + \frac{N}{2}\right) (2k+1)\right), \quad k = 0, \dots, \frac{N}{2}-1 \\ \tilde{x}(n) &= \sum_{k=0}^{\frac{N}{2}-1} X(k) \cos\left(\frac{\pi}{2N} \left(2n+1 + \frac{N}{2}\right) (2k+1)\right), \quad n = 0, \dots, N-1 \end{aligned} \quad (2.1)$$

where, $X(k)$ are the resulting MDCT coefficients and $\tilde{x}(n)$ are the reconstructed samples. Because of time domain aliasing, $\tilde{x}(n)$ will not be the same as $x(n)$. $\tilde{x}(n)$ has to be windowed and then overlap-added with the previous frame to cancel the time domain aliasing. For convenience, we ignore the normalization factors in all definitions. The normalization factors can also be incorporated in the window without affecting the computational complexity.

By adopting the matrix notation proposed by Cheng and Hsu [29], the $(N/2 \times N)$ MDCT matrix M can be defined as,

$$M(k, n) = \cos\left(\frac{\pi}{2N} \left(2n+1 + \frac{N}{2}\right) (2k+1)\right), \quad \begin{matrix} k = 0, \dots, \frac{N}{2}-1, \\ n = 0, \dots, N-1. \end{matrix} \quad (2.2)$$

and write,

$$\begin{aligned} \mathbf{X} &= M\mathbf{x} \\ \tilde{\mathbf{x}} &= M^T \mathbf{X} \end{aligned} \quad (2.3)$$

where,

$$\mathbf{X} = \left[X(0), \dots, X\left(\frac{N}{2}-1\right) \right]^T \quad (2.4)$$

$$\mathbf{x} = \left[x(0), \dots, x(N-1) \right]^T \quad (2.5)$$

$$\tilde{\mathbf{x}} = [\tilde{x}(0), \dots, \tilde{x}(N-1)]^T \quad (2.6)$$

We next map MDCT and IMDCT into an $N/2$ -point DCT-IV as follows [29]:

$$\mathbf{M}^T = \mathbf{P} \mathbf{S} \mathbf{C}_{N/2}^{IV} \quad (2.7)$$

$$\mathbf{M} = \mathbf{C}_{N/2}^{IV} \mathbf{S} \mathbf{P}^T \quad (2.8)$$

where,

$$\mathbf{P} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{N/4} \\ \mathbf{0} & -\mathbf{J}_{N/4} \\ \mathbf{J}_{N/4} & \mathbf{0} \\ \mathbf{I}_{N/4} & \mathbf{0} \end{bmatrix} \quad (2.9)$$

$\mathbf{I}_{N/4}$ is an $N/4 \times N/4$ identity matrix and $\mathbf{J}_{N/4}$ is an $N/4 \times N/4$ order reversal matrix defined as,

$$\mathbf{J}_{N/4} = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 & 0 \end{bmatrix}_{N/4 \times N/4} \quad (2.10)$$

$$\mathbf{S} = \begin{bmatrix} -\mathbf{I}_{N/4} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{N/4} \end{bmatrix} \quad (2.11)$$

and, $\mathbf{C}_{N/2}^{IV}$ is an $N/2 \times N/2$ DCT-IV matrix defined as

$$C_{N/2}^{IV}(k, n) = \cos\left(\frac{\pi}{2N}(2k+1)(2n+1)\right), \quad k, n = 0, \dots, \frac{N}{2}-1. \quad (2.12)$$

These relationships are shown in Fig. 2.1 for IMDCT. By using the involutory property of the DCT-IV matrix (i.e., the matrix is both symmetrical and orthogonal), it can be mapped into DCT-II as follows [30],

$$C_{N/2}^{IV} = D \left(C_{N/2}^H \right)^T L^T \quad (2.13)$$

where, D is a diagonal matrix with elements,

$$D(i,i) = 2 \cos \left(\frac{\pi}{2N} (2i+1) \right), \quad i = 0, \dots, \frac{N}{2} - 1 \quad (2.14)$$

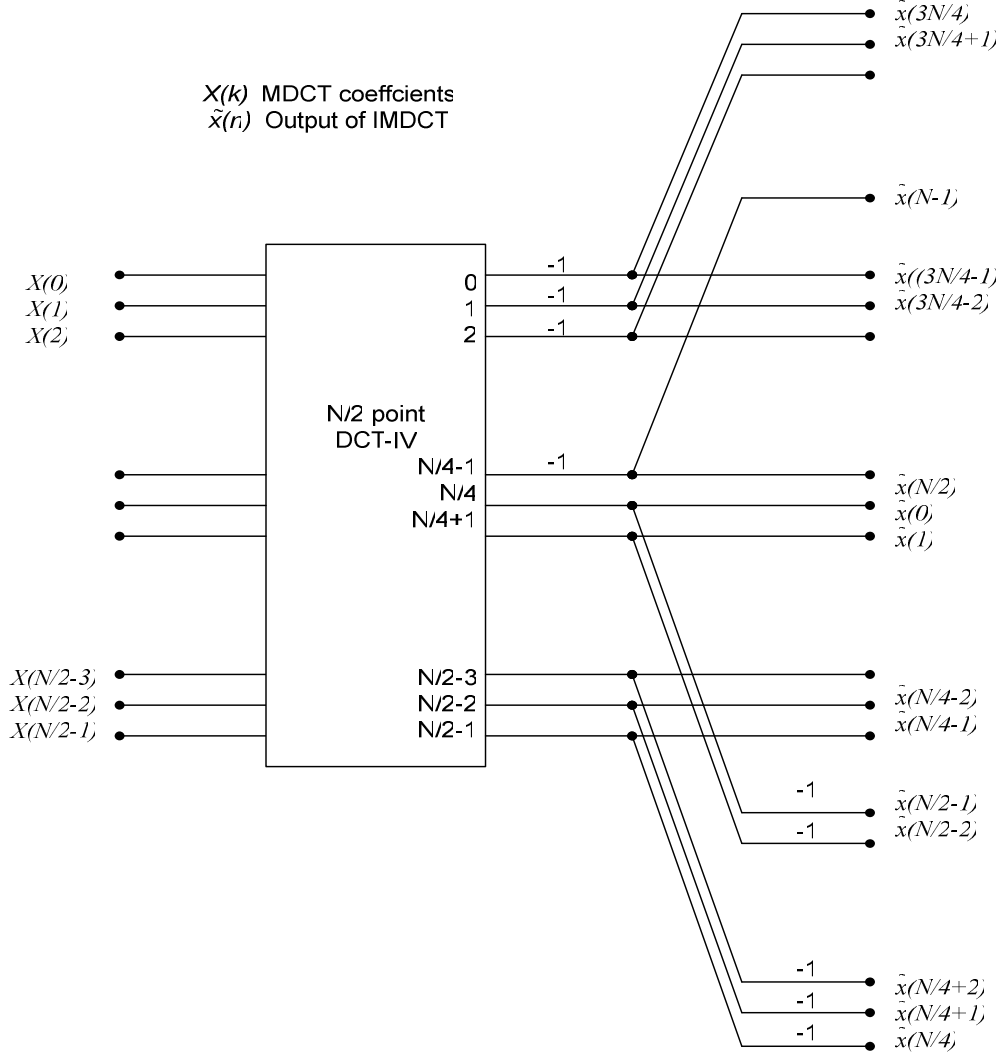


Fig. 2.1 Mapping IMDCT to DCT-IV [29]

$$L = \begin{bmatrix} 0.5 & 0 & 0 & 0 & \cdots & 0 \\ -0.5 & 1 & 0 & 0 & \cdots & 0 \\ 0.5 & -1 & 1 & 0 & \cdots & 0 \\ -0.5 & 1 & -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -0.5 & 1 & -1 & 1 & \cdots & 1 \end{bmatrix} \quad (2.15)$$

and, $C_{N/2}^H$ is an $N/2 \times N/2$ DCT-II matrix defined as

$$C_{N/2}^H(k, n) = \cos\left(\frac{\pi}{N}(2n+1)k\right), \quad k, n = 0, \dots, \frac{N}{2}-1 \quad (2.16)$$

The relationship between DCT-IV and DCT-II is shown in Fig. 2.2.

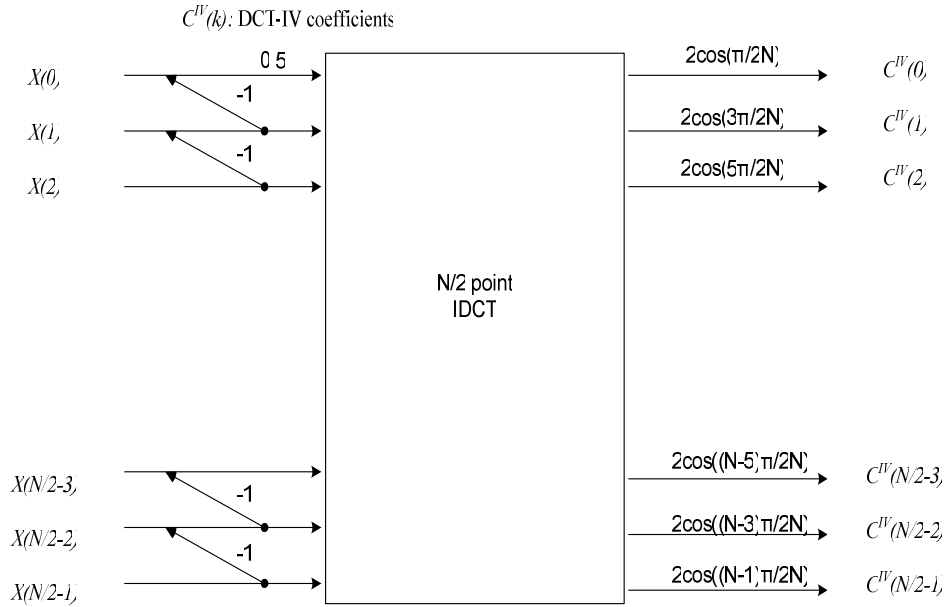


Fig. 2.2 Mapping DCT-IV to DCT-II [30]

As can be seen from Fig. 2.1 and 2.2, the multiplications contributed by matrix D in equation (2.14) can be absorbed into the subsequent windowing stage after the

application of IMDCT. This would save $N/2$ multiplications and $N/2$ constant memory locations.

As we can also see from the figures, the most computationally intensive block is the $N/2$ -point DCT-II/IDCT-II. Since we are interested in lengths of type 5×2^m and 15×2^m , we need a good algorithm that is applicable for even lengths. One of the most efficient algorithms for even length DCTs is the one presented by Kok [30] in which the author shows that the algorithm is better than the prime factorization algorithms available for DCT such as Yang [44], Lee [45] and Chan [46]. It is also optimal for power-of-2 lengths in the sense that it gives the same number of multiplications and additions as the best power-of-2 algorithms. Kok's algorithm is basically a decimation-in-frequency strategy, splitting an N -point DCT into two $N/2$ -point DCTs along with some pre/post additions and multiplications. Hence, the algorithm can be recursively executed till we are left with odd-length transforms; in our case we will have 5-point or 15-point DCT/IDCT. An illustration of Kok's algorithm for a 10-point IDCT is shown in Fig. 2.3.

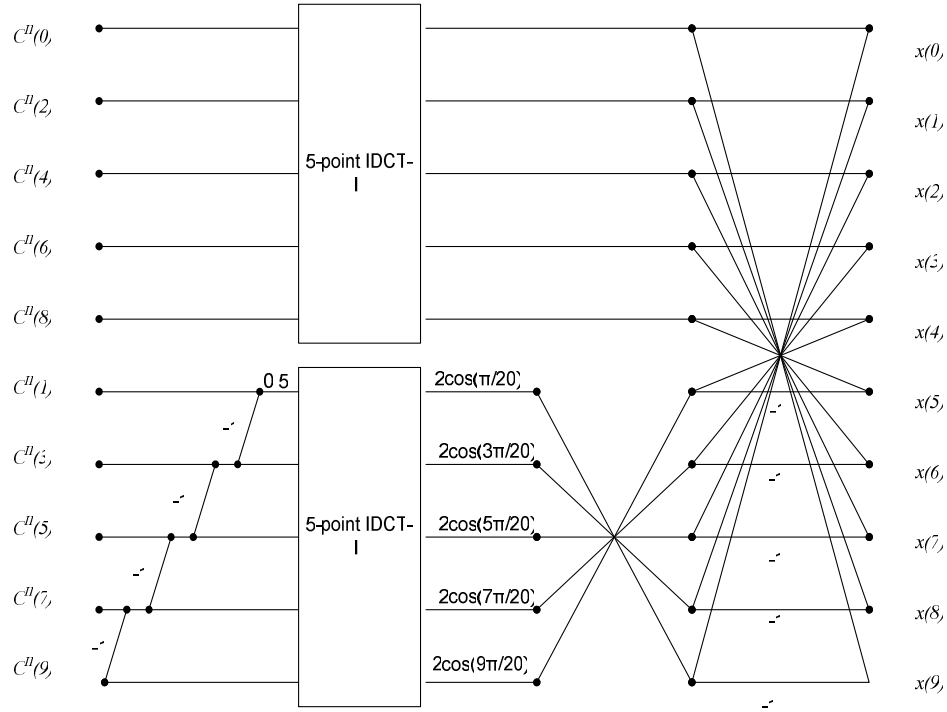


Fig. 2.3 Kok's algorithm for a 10-point IDCT-II [30]

Note that the discussion so far is applicable for any even length MDCT transform including power-of-2 lengths.

2.3 Fast Algorithms for 5-point and 15-point DCT-II

2.3.1 Overview

We need efficient DCT algorithms for the smaller lengths of 5 and 15 because there will be 2^{m-1} such blocks at the last stage of Kok's algorithm [30]. The 5-point DCT is usually implemented in a brute force manner, leading to 20 additions and multiplications. The 15-point DCT is usually implemented using prime factor algorithms for DCT [44, 45]. These algorithms split the 15-point DCT into a two

dimensional 3×5 DCT with input and output index mappings. These mappings also have additions at either the input [45] or output [46].

In this section, we present an efficient implementation of odd-length DCT by an FFT. We first discuss Heideman's mapping of odd-length DCTs to equal length real DFTs. This mapping involves only permutations at input and output and, sign changes at the output. Next, efficient algorithms for odd-length real input DFTs are presented which are based on Winograd short, prime-length DFT modules. Since Winograd DFT modules for prime length sequences have theoretically optimal number of multiplications, the DCT modules obtained from these DFT modules are very efficient.

Specifically, 5-point DCT is implemented by applying Heideman's mapping to 5-point Winograd DFT module. This algorithm requires 5 multiplications and 13 additions. 15-point DCT is implemented by applying Heideman's mapping to 15-point Winograd Fourier Transform Algorithm (WFTA). WFTA is essentially a prime factor algorithm for FFT along with a step called *nesting* which further reduces the number of multiplications. Details are given in the following sub sections. The resulting 15-point DCT algorithm takes 17 multiplications and 67 additions.

Our contribution is to use efficient DFT algorithms for implementing DCT rather than using the more complex prime factor algorithms for DCT or the brute-force approaches.

2.3.2 Definitions

For the purpose of this section, DCT-II and DFT are defined as follows:

$$\begin{aligned}
\text{DCT: } X_C(k) &= \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi(2n+1)k}{2N}\right) ; k = 0, \dots, N-1 \\
\text{DFT: } X_F(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} ; k = 0, \dots, N-1
\end{aligned} \tag{2.17}$$

where,

$$W_N = e^{-j\frac{2\pi}{N}}$$

We assume the data is real. We also ignore all normalization factors since they can be absorbed in other blocks such as windowing without affecting the computational complexity.

2.3.3 Heideman's Mapping

Typically to implement DCT through DFT, we need to use DFT of twice or four times the length of DCT [34]. Hence, it is generally inefficient to compute DCT by FFT. However, Heideman showed that an odd length DCT can be implemented by DFT of the same size with just input and output permutations and output sign changes [34]. This is possible because, for odd length DCT the set of cosines is exactly the same as the set of sines and cosines in DFT of the same length. This is however not true if the length is even. Thus, an odd length DCT can be efficiently implemented by a real valued DFT. The mapping given by Heideman is as follows:

Define a new sequence $\hat{x}(n)$:

$$\hat{x}(n) = \begin{cases} x\left((-1)^{n+(N+1)/2+1} n + \frac{N-1}{2}\right); & n = 0, 1, \dots, \frac{N-1}{2} \\ x\left((-1)^{n+(N+1)/2+1} (N-n) + \frac{N-1}{2}\right); & n = \frac{N+1}{2}, \dots, N-1 \end{cases} \tag{2.18}$$

$\hat{x}(n)$ is essentially a permutation of the input sequence $x(n)$. Let $\hat{X}_F(k)$ be the DFT of the sequence $\hat{x}(n)$. Then,

$$\begin{aligned} X_C(2k) &= (-1)^k \operatorname{Re} \left[\hat{X}_F(k) \right]; & k = 0, \dots, \frac{N-1}{2} \\ X_C(N-2k) &= (-1)^{k+1} \operatorname{Im} \left[\hat{X}_F(k) \right]; & k = 1, \dots, \frac{N-1}{2} \end{aligned} \quad (2.19)$$

(Note that the exponent for -1 in equation (9) of [34] is incorrect. The correct expression is shown in the above equations.) As can be seen from equation (2.19), we need to compute only the first $(N+1)/2$ output points for the DFT. The input mapping for $N = 15$ is given in Appendix A as part of the Matlab code for 15-point DCT.

For real input, DFT exhibits conjugate symmetry property, i.e.,

$$X_F(k) = X_F^*(N-k) \quad (2.20)$$

Hence, if N is odd, here too we need to compute just the first $(N+1)/2$ output points. Therefore, we conclude that the computational complexity of the DCT is exactly the same as that of DFT for odd N such as 5 and 15. The problem now boils down to implementing a very efficient FFT for real valued input data and an odd transform length.

2.3.4 DFT Algorithms for Real-Valued Input Data

There has been significant literature discussing efficient algorithms for real valued FFTs [35-38]. Basically, the algorithms fall into two categories: prime factor algorithms (PFA) and common factor algorithms (CFA). PFAs are used when the transform length N can be decomposed into two or more relatively prime factors. An example of PFA is the Winograd Fourier Transform Algorithm (WFTA). CFAs are

used when the factors are not relatively prime. Examples of CFA are the traditional Cooley-Tukey FFT algorithms and the split radix algorithms. Both PFAs and CFAs are based on permuting the input data into a two dimensional matrix so that the corresponding two dimensional DFT is a separable transform with minimum number of twiddle factors [39]. Especially for PFAs, the row and column DFTs are independent and are not related by any twiddle factors [39].

In our case, the length 15 can be decomposed into 5 and 3 which are mutually prime. Hence, PFAs are applicable in our case. Two PFA algorithms are available: the PFA algorithms developed by Burrus et al. [35-37, 39] and the WFTA. The WFTA algorithm uses Winograd short- N DFT modules for prime lengths N as building blocks. WFTA algorithms typically minimize the number of multiplications (by using a *nesting* procedure) at the expense of a slight increase in additions [39, 40]. However, for short lengths (including 15) [35, 36], the WFTA algorithm uses lesser number of multiplications and the same number of additions as PFA. Hence, we will be using the WFTA algorithm for our 15-point FFT. Our survey of the existing literature shows that the 15-point real WFTA is the least complex among the available algorithms [35-40].

2.3.5 Winograd Short- N DFT Modules

Winograd short- N DFT modules are the building blocks for constructing the WFTA for longer lengths. The short- N modules are defined for prime lengths. Specifically, we need 3-point and 5-point DFT modules for the 15-point transform. For 5-point DCT, we can simply use Winograd's 5-point DFT module combined with Heideman's mapping.

The Winograd DFT modules are based on a fast cyclic convolution algorithm for prime lengths using the theoretically minimum number of multiplications [39-41]. This optimum convolution algorithm can be mapped to DFT using Rader's technique [39] to give very efficient DFT modules for prime lengths.

In mathematical terms, Winograd's algorithm achieves a canonical decomposition of the DFT matrix as shown below. Using the matrix notation in [42],

$$D_N = S_N C_N T_N, \quad (2.21)$$

where,

D_N is the $N \times N$ DFT matrix,

S_N is a $N \times J$ matrix having only 0's, 1's and -1's,

C_N is a $J \times J$ diagonal matrix,

T_N is a $J \times N$ matrix having only 0's, 1's and -1's.

That is, S_N and T_N are just addition matrices and C_N is the multiplier matrix. Moreover, the elements of C_N matrix are either purely real or purely imaginary, so for real input data, we will have just one real multiplication for each element of C_N . Hence, the number of multiplications will be J . Winograd algorithm is powerful because, for small and prime N (such as 3, 5, 7, 11), J is very close to N . That is, we need only about N multiplications instead of N^2 multiplications required in brute force approach. For example, for $N = 3$, the S_3 , C_3 and T_3 can be derived from [41] as follows:

$$\begin{aligned}
S_3 &= \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & -1 \end{bmatrix}, \\
C_3 &= \text{diag} \left[1 \quad \cos\left(-\frac{2\pi}{3}\right)-1 \quad j \sin\left(-\frac{2\pi}{3}\right) \right], \\
T_3 &= \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & -1 \end{bmatrix}
\end{aligned} \tag{2.22}$$

Note that the S_N and T_N matrices can be factorized into sparse matrices to minimize the number of additions. For example S_3 and C_3 in equation (2.22) can be factorized as [41],

$$\begin{aligned}
S_3 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \\
T_3 &= \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & -1 \end{bmatrix}
\end{aligned} \tag{2.23}$$

2.3.6 Prime Factor Mapping

The idea behind mapping a one-dimensional array into a two-dimensional array is to divide the bigger problem into several smaller problems and solve each of the smaller problems effectively [39]. The mappings are based on modulo integer arithmetic to exploit the periodicity property of the complex exponentials in DFT.

Let $N = N_1 N_2$. Let N_1 and N_2 be co-prime. Let n, n_1, n_2, k, k_1, k_2 be the corresponding index variables in time and frequency domain. The mappings from n_1, n_2, k_1, k_2 to n and k can be defined as follows [39]:

$$\begin{aligned}
n &= \langle K_1 n_1 + K_2 n_2 \rangle_N \quad n_1 = 0, \dots, N_1 - 1; \quad n_2 = 0, \dots, N_2 - 1 \\
k &= \langle K_3 k_1 + K_4 k_2 \rangle_N \quad k_1 = 0, \dots, N_1 - 1; \quad k_2 = 0, \dots, N_2 - 1
\end{aligned} \tag{2.24}$$

where, $\langle \cdot \rangle_N$ represents modulo- N operation.

There always exist integers K_1, K_2, K_3 and K_4 such that the above mapping is unique (i.e., all values of n and k between 0 and $N-1$ can be generated by using all combinations of n_1, n_2 and k_1, k_2) [39]. The mapping is called a prime factor map (PFM) if [39],

$$\begin{aligned} K_1 &= aN_2 \text{ and } K_2 = bN_1, \\ K_3 &= cN_2 \text{ and } K_4 = dN_1 \end{aligned} \quad (2.25)$$

A solution that satisfies the PFM and also decouples the row and column DFTs is [39],

$$\begin{aligned} K_1 &= N_2, \quad K_2 = N_1, \\ K_3 &= N_2 \langle N_2^{-1} \rangle_{N_1}, \quad K_4 = N_1 \langle N_1^{-1} \rangle_{N_2} \end{aligned} \quad (2.26)$$

where, $\langle x^{-1} \rangle_N$ is defined as the smallest natural number that satisfies $\langle x^{-1} x \rangle_N = 1$ [39].

The DFT then becomes [39],

$$\hat{X}_F(k_1, k_2) = \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} \hat{x}(n_1, n_2) W_{N_1}^{n_1 k_1} W_{N_2}^{n_2 k_2} \quad (2.27)$$

where,

$$\begin{aligned} \hat{x}(n_1, n_2) &= x(\langle K_1 n_1 + K_2 n_2 \rangle_N), \\ \hat{X}_F(k_1, k_2) &= X_F(\langle K_3 k_1 + K_4 k_2 \rangle_N) \end{aligned} \quad (2.28)$$

That is, the DFT is first applied along the columns and then along the rows.

Matrix Interpretation: If we fix a value for n_1 and vary the values for n_2 from 0 to N_2-1 in the above mapping and do this for all values of n_1 from 0 to N_1-1 , we obtain a permuted sequence of values of n from 0 to $N-1$. Let P_i be the input permutation matrix

describing these steps. Similarly, we define P_o to be the output permutation matrix.

Then it can be shown that [42],

$$P_o \mathbf{X}_F = (D_{N_1} \otimes D_{N_2}) P_i \mathbf{x}; \otimes \text{ denotes the Kronecker product.} \quad (2.29)$$

where, the Kronecker product of two matrices A and B is defined as,

$$A_{M \times N} \otimes B_{P \times Q} = \begin{bmatrix} A(0,0)B & A(0,1)B & \cdots & A(0,N-1)B \\ A(1,0)B & A(1,1)B & \cdots & A(1,N-1)B \\ \vdots & \vdots & \ddots & \vdots \\ A(M-1,0)B & A(M-1,1)B & \cdots & A(M-1,N-1)B \end{bmatrix}_{MP \times NQ} \quad (2.30)$$

That is, a two-dimensional $N_1 \times N_2$ DFT can be viewed as a Kronecker product of the row and column DFT matrices D_{N_1} and D_{N_2} . This matrix representation and the corresponding permutation matrices for $N = 15$ are illustrated in the Matlab code of Appendix B.

2.3.7 Winograd Fourier Transform Algorithm (WFTA)

WFTA takes forward the matrix interpretation of the PFA given above. If we have short-N DFT modules for lengths N_1 and N_2 , then,

$$D_{N_1} \otimes D_{N_2} = (S_{N_1} C_{N_1} T_{N_1}) \otimes (S_{N_2} C_{N_2} T_{N_2}) \quad (2.31)$$

Using the properties of Kronecker products, it can be shown that [41-42],

$$\begin{aligned} D_{N_1} \otimes D_{N_2} &= (S_{N_1} \otimes S_{N_2})(C_{N_1} \otimes C_{N_2})(T_{N_1} \otimes T_{N_2}) \\ &= S_N C_N T_N \end{aligned} \quad (2.32)$$

This last step in equation (2.32) is called the *nesting* procedure in WFTA [42]. Because the multiplications are nested, this algorithm generally yields the lowest multiplication count. The above equations can be further simplified as follows:

Suppose,

$$P_i \mathbf{x} = \begin{bmatrix} e_0 \\ \cdot \\ \cdot \\ \cdot \\ e_{N_2-1} \\ e_{N_2} \\ \cdot \\ \cdot \\ \cdot \\ e_{2N_2-1} \\ \cdot \\ \cdot \\ \cdot \\ e_{(N_1-1)N_2} \\ \cdot \\ \cdot \\ \cdot \\ e_{N_1N_2-1} \end{bmatrix} \quad (2.33)$$

Define,

$$z_N = \begin{bmatrix} e_0 & \cdot & \cdot & \cdot & e_{N_2-1} \\ e_{N_2} & \cdot & \cdot & \cdot & e_{2N_2-1} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ e_{(N_1-1)N_2} & \cdot & \cdot & \cdot & e_{N_1N_2-1} \end{bmatrix} \quad (2.34)$$

Similarly, define Z_N for the output coefficients. P_i and P_o are as defined in equation (2.29). Then it can be shown that [42] (see also the corrections and comments in [43]),

$$Z_N = S_{N_1} \left[S_{N_2} \left[C \circ \left\{ T_{N_2} \left(T_{N_1} z_N \right)^T \right\} \right] \right]^T \quad (2.35)$$

where,

$$C(m, n) = C_{N_2}(m, m)C_{N_1}(n, n); \quad m = 0, \dots, M_{N_2} - 1, \quad n = 0, \dots, M_{N_1} - 1 \quad (2.36)$$

M_{N_1} and M_{N_2} are the lengths of the diagonals (i.e., the number of multiplications) of C_{N_1} and C_{N_2} respectively,

◦ denotes element-by-element product of two matrices of same dimensions. It is defined as follows,

$$\begin{aligned} \text{If } C_{M \times N} &= A_{M \times N} \circ B_{M \times N}, \text{ then,} \\ C(m, n) &= A(m, n)B(m, n); \quad m = 0, \dots, M - 1; \quad n = 0, \dots, N - 1 \end{aligned} \quad (2.37)$$

T denotes matrix transpose.

It can be seen from equation (2.36) that the number of multiplications is $M_{N_1} \times M_{N_2}$. Some of these are trivial multiplications by 1. When $N = 3 \times 5$, M_3 turns out to be 3 and $M_5 = 6$ [41]. So we have a total of 17 multiplications (excluding the multiplication by 1). Two of these are multiplications by 1.25 and 1.5. If we discount them as trivial multiplications (because they can implemented by adds and shifts) then we have only 15 non-trivial multiplications.

The number of complex additions is calculated as follows: Let A_{N_1} be the number of complex additions contributed together by S_{N_1} and T_{N_1} . Similarly define A_{N_2} . Then from the above matrix description we can derive the number of complex additions as ([39, 40]) $(N_2 A_{N_1} + M_{N_1} A_{N_2})$. If we interchanged N_1 and N_2 , the number would be $(N_1 A_{N_2} + M_{N_2} A_{N_1})$ which is different from the previous number. Hence the order is important here. When $N = 15$, we have $A_3 = 6$ and $A_5 = 17$ [41]. If we choose $N_1 = 3$ and

$N_2 = 5$, we get the number of additions as 81. If $N_1 = 5$ and $N_2 = 3$, we get 86. Hence, we choose $N_1 = 3$ and $N_2 = 5$. For real data, we use the fact that DFT exhibits conjugate symmetry. Hence, we need not calculate $(N-1)/2$ output points. Further, we note that adding a purely real number to a purely imaginary number is not an addition in the computational sense. Using these facts, it is possible to reduce $(N-1)$ additions from the above count. Hence, we require $81-14 = 67$ real additions for $N = 15$. This number tallies with that given in [35]. This also proves that our novel combination of Heideman's mapping and WFTA has accomplished the task of implementing a 15-point DCT using a minimum number of multiplications and additions. As a comparison, PFA algorithms for DCT or DFT take 25 multiplications and 67 additions. Therefore, the number of multiplications has been reduced by 9.

2.3.8 5-point DCT

The 5-point DCT-II algorithm can simply be derived by applying Heideman's mapping to Winograd's 5-point DFT module. For 5-point DFT we have the following [41]:

$$S_5 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.38)$$

$$u = \frac{-2\pi}{5};$$

$$C_5 = \text{diag} \left[1 \quad \frac{\cos u + \cos 2u}{2} - 1 \quad \frac{\cos u - \cos 2u}{2} \quad j(\sin u + \sin 2u) \quad j \sin 2u \quad j(\sin u - \sin 2u) \right] \quad (2.39)$$

$$T_5 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 \end{bmatrix} \quad (2.40)$$

The flow graph for the 5-point DCT is shown in Fig. 2.4. It requires 5 multiplications and 13 additions. The algorithm for IDCT can simply be obtained by transposing the flow graph since DCT is an orthogonal transform.

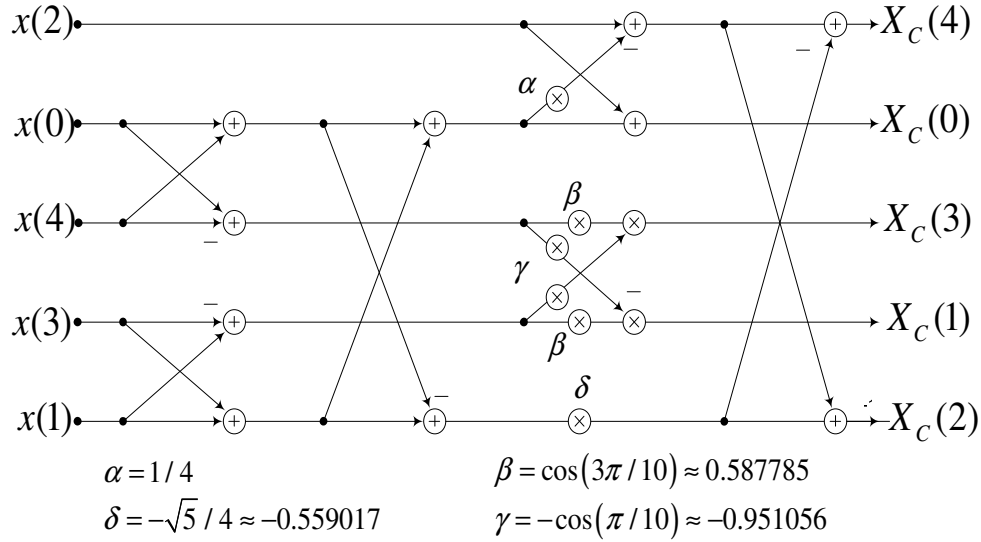


Fig. 2.4 Flow graph for 5-point DCT-II

2.3.9 15-point DCT

To summarize, we have developed and implemented a new 15-point DCT algorithm using Winograd's 3-point and 5-point DFT algorithms in the WFTA. We applied Heideman's mapping (equations (2.18), (2.19)) to the DFT algorithm to obtain

the DCT algorithm. The flow graph for the algorithm is shown in Fig. 2.5. Algorithm for IDCT can be obtained by transposing the flow graph.

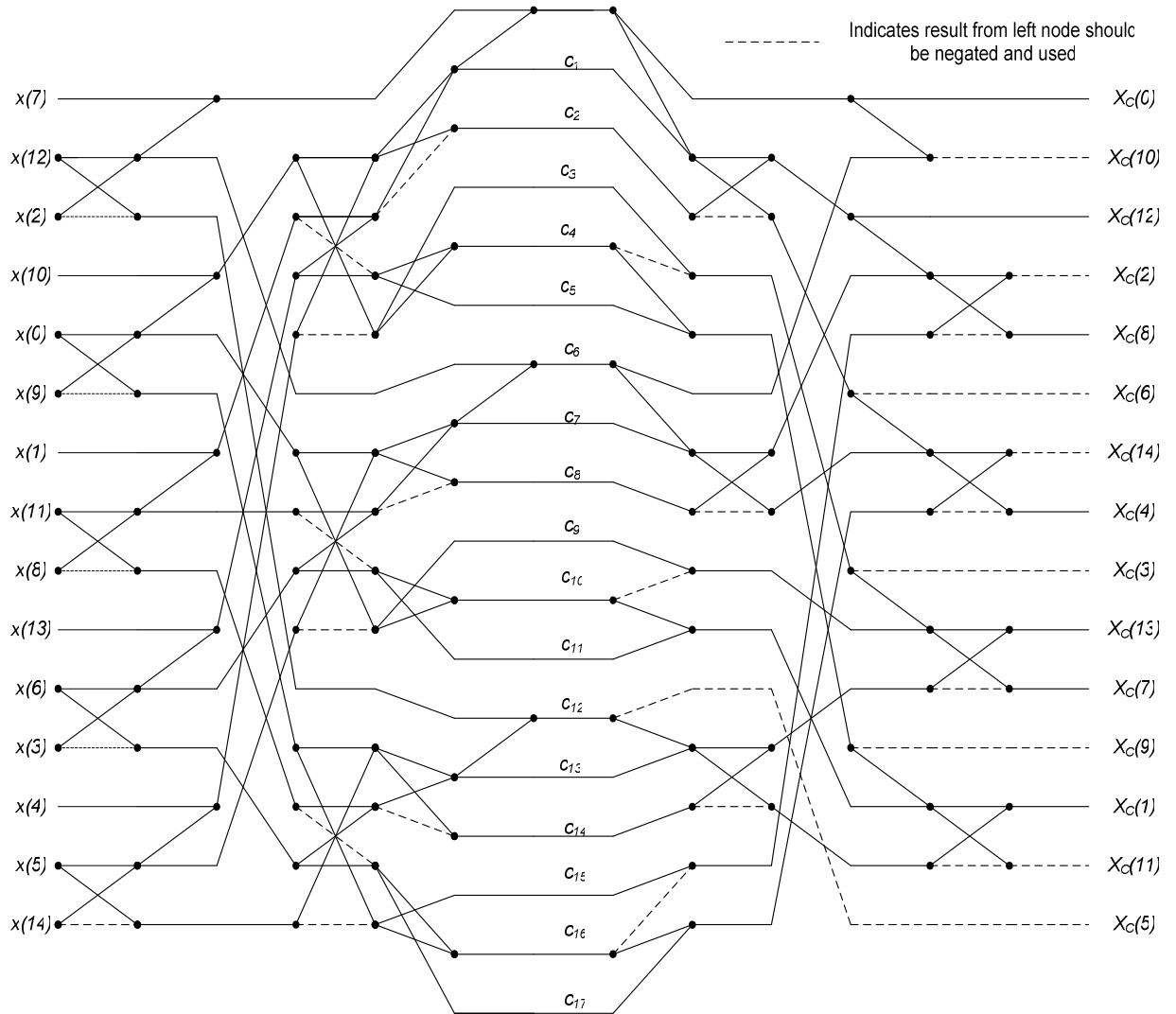


Fig. 2.5 Flow graph for 15-point DCT-II

The constants used in Fig. 2.5 are defined below:

$$u = -\frac{2\pi}{5}; \quad v = -\frac{2\pi}{3}$$

$$c_1 = \frac{\cos u + \cos 2u}{2} - 1; \quad c_2 = \frac{\cos u - \cos 2u}{2}$$

$$c_3 = \sin u + \sin 2u; \quad c_4 = \sin 2u; \quad c_5 = \sin u - \sin 2u$$

$$\begin{aligned}
c_6 &= \cos v - 1; & c_7 &= c_1 c_6; & c_8 &= c_2 c_6 \\
c_9 &= c_3 c_6; & c_{10} &= c_4 c_6; & c_{11} &= c_5 c_6 \\
c_{12} &= \sin v; & c_{13} &= c_1 c_{12}; & c_{14} &= c_2 c_{12} \\
c_{15} &= -c_3 c_{12}; & c_{16} &= -c_4 c_{12}; & c_{17} &= -c_5 c_{12}
\end{aligned} \tag{2.41}$$

The Matlab code for 15-point DCT is given in Appendix A. The code is tested applying the algorithm to all 15 basis vectors. The results were exactly the same as those of brute force approach. Since DCT is a linear transform, the fast algorithm should also work for any linear combination of the basis vectors, i.e., the algorithm should work for any arbitrary 15-point input.

2.4 Modified Window Function

As noted in section 2.2, the multiplications contributed by the matrix D (equation (2.14)) in implementing DCT-IV, can be absorbed into the windowing stage that precedes MDCT or accompanies IMDCT. In this section, we show that if we start with a symmetrical *sine* window, then the modified window is piece-wise symmetric.

The symmetrical *sine* window that is usually used with MDCT filterbanks [31] is defined as follows:

$$h(n) = \sin\left(\frac{(2n+1)\pi}{2N}\right), \quad n = 0, \dots, N-1 \tag{2.42}$$

The modified window function obtained by merging the matrix D with $h(n)$ (after some algebraic manipulation) is as follows:

$$w(n) = \begin{cases} 2 \cos\left(\frac{\pi}{2N}\left(2n + \frac{N}{2} + 1\right)\right) \sin\left(\frac{\pi(2n+1)}{2N}\right), & n = 0, \dots, \frac{N}{4} - 1, \\ -2 \cos\left(\frac{\pi}{2N}\left(2n + \frac{N}{2} + 1\right)\right) \sin\left(\frac{\pi(2n+1)}{2N}\right), & n = \frac{N}{4}, \dots, N-1. \end{cases} \tag{2.43}$$

For $0 \leq n < N/4$:

$$\begin{aligned} w\left(\frac{N}{4} - 1 - n\right) &= 2 \sin\left(\frac{\pi(2n+1)}{2N}\right) \cos\left(\frac{\pi}{2N}\left(2n + \frac{N}{2} + 1\right)\right) \\ &= w(n) \end{aligned} \quad (2.44)$$

Similarly, for $0 \leq n < 3N/4$:

$$\begin{aligned} w\left(n + \frac{N}{4}\right) &= 2 \sin\left(\frac{\pi(2n+1)}{2N}\right) \sin\left(\frac{\pi}{2N}\left(2n + \frac{N}{2} + 1\right)\right) \\ &= w(N-1-n) \end{aligned} \quad (2.45)$$

As can be observed from (2.44) and (2.45), the window is piece-wise symmetric. This is illustrated in Fig. 2.6 ($N = 640$). This means that the memory required to store the modified window is the same as that for the *sine* window ($N/2$ words). In effect, we have reduced $N/2$ words of memory requirement by not storing the factors of matrix D .

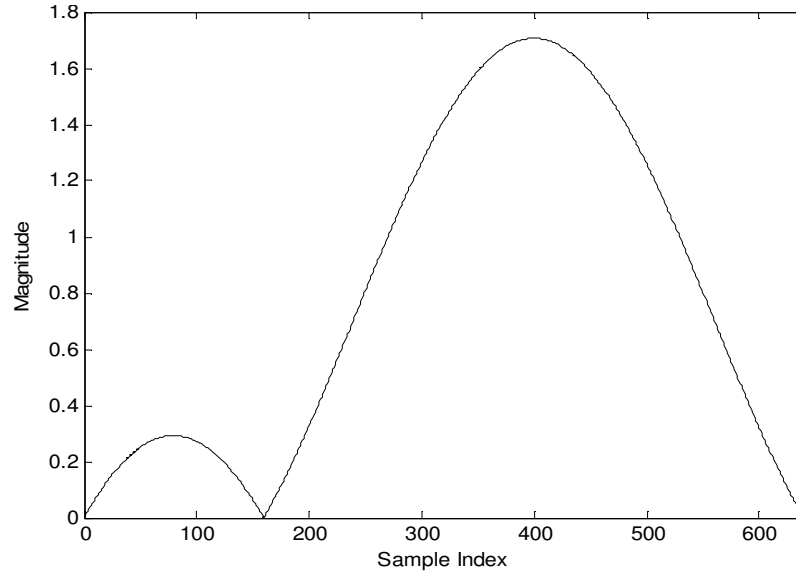


Fig. 2.6 Modified window function ($N = 640$)

2.5 Complexity Analysis of the New MDCT Algorithm

In this section, we consider the complexity analysis of the proposed MDCT algorithm in terms of the number of additions, multiplications and memory requirement. We consider the case of $N = p \times 2^m$ ($m \geq 2$), where p is either 5 or 15. Let $RM_F(N)$ and $RA_F(N)$ be, respectively, the number of multiplications and additions of an N -point MDCT along with windowing operation. Let $RM_I(N)$ and $RA_I(N)$ be, respectively, the number of multiplications and additions of an N -point IMDCT along with windowing and overlap-add operation. Let $RM_D(p)$ and $RA_D(p)$ be, respectively, the number of multiplications and additions of the fast 5-point and 15-point DCT-II. Then the computational complexity of the proposed algorithm is given by

$$RM_F(p \times 2^m) = RM_I(p \times 2^m) = p \times 2^m + 2^{m-1} RM_D(p) + p(m-1)2^{m-2} \quad (2.46)$$

$$RA_F(p \times 2^m) = RA_I(p \times 2^m) = p \times 2^m + 2^{m-1} RA_D(p) + 3p(m-1)2^{m-2} - 2^{m-1} \quad (2.47)$$

From section 2.3, $RM_D(5) = 5$, $RA_D(5) = 13$, $RM_D(15) = 17$, $RA_D(15) = 67$.

In Table 2.1, we present the results of complexity comparison of our algorithm with two possible implementations of MDCT of size $N = 640$ (5×2^7) using the well-known Britanak and Rao's algorithm [14]. In the first case, we assume that DCT-II of size 5 is implemented using straightforward matrix multiplication, and in the second case, we use same 5-point DCT-II as in our proposed algorithm. The complexity numbers presented in Table 3.1 include the complexity of windowing and overlap-add operations. This table also presents estimated constant memory requirements for these algorithms. Table 2.2 gives a similar complexity comparison for $N = 1920$ (15×2^7).

It can be seen from these tables that there are considerable gains in number of multiplications and additions and ROM requirement. The algorithm for $N = 5 \times 2^m$ was accepted in the floating point implementation of G.EV-VBR codec [19].

Table 2.1 Comparison of proposed MDCT algorithm for $N = 640$

*: without fast 5-point DCT (direct matrix multiplication, 20 mults and adds)

**: with fast 5-point DCT proposed in this chapter

Algorithm		Multi plica- tions	Addi- tions	Transform Memory # words	Window words	Memory #
MDCT with windowing	[14]*	3200	5376	640	320	
	[14]**	2240	4928	640	320	
	Proposed Algorithm	1920	4288	320	320	
IMDCT with windowing and overlap-add operation	[14]*	3200	5376	640	320	
	[14]**	2240	4928	640	320	
	Proposed Algorithm	1920	4288	320	320	

Table 2.2 Comparison of proposed MDCT algorithm for $N = 1920$

*: without fast 15-point DCT (direct matrix multiplication, 210 mults and adds)

**: with fast 15-point DCT proposed in this chapter

Algorithm		Multi plica- tions	Addi- tions	Transform Memory # words	Window words	Memory #
MDCT with windowing	[14]*	19200	25856	1920	960	
	[14]**	6848	16704	1920	960	
	Proposed Algorithm	5888	14784	960	960	
IMDCT with windowing and overlap-add operation	[14]*	19200	25856	1920	960	
	[14]**	6848	16704	1920	960	
	Proposed Algorithm	5888	14784	960	960	

CHAPTER 3

MPEG-4 AAC ENHANCED LOW DELAY FILTERBANKS

3.1 Introduction

Traditionally, speech and audio coding paradigms have been different. Speech coding was primarily based on source modeling [7, 25] and many speech codecs are characterized by low round-trip algorithmic delay [7] making them suitable for full-duplex communications. However, they were focused on single-speaker material and performed badly for music signals [20]. On the other hand, audio coding was based on the psychoacoustics of the human ear [8]. The codecs were intended for perceptually transparent reproduction of any generic music material. However, they usually have high algorithmic delays making them unsuitable for full-duplex communication purposes [8].

The algorithmic delay of a codec depends on the following factors [26]:

1. *Frame Length*: larger the frame length, more the delay because of the time required to buffer the input samples.
2. *Filterbank delay*: analysis and synthesis filterbanks used in the codec also introduce delay. This is contributed by the causal FIR filters used in the filterbank.
3. *Block switching*: audio codecs switch between long and short blocks to optimally handle non-stationary parts of the signal. Since block switching

cannot happen instantaneously, a certain amount of look-ahead is required to make the transition between long and short blocks smooth. This look-ahead contributes to the delay.

4. *Bit Reservoir*: the encoder generally uses a bit reservoir to optimize the number of bits used for each frame, yet maintaining an average bit rate over a period of time. This causes the decoder to wait till it receives the maximum possible number of bits in a frame, contributing to the algorithmic delay [27].

3.2 MPEG-4 AAC LD

The first effort toward making the audio codecs suitable for full-duplex communications was made with the standardization of MPEG-4 ER AAC LD [26]. LD stands for Low Delay. The algorithmic delay is reduced by making the following changes [26]:

1. The frame size is reduced from the usual 1024/960 points to 512/480 points.
2. Block switching is not used to avoid the delay due to look-ahead.
3. To handle non-stationary parts of the signal in the absence of block switching, the coder uses a low overlap *sine* window for transient regions. The low overlap also minimizes pre-echo artifacts produced by the temporal spreading of quantization noise in the non-stationary portions [26]. The low overlap window is defined as follows [26] (N , the frame length, is either 1024 or 960):

$$W(i) = \begin{cases} 0 & i = 0, \dots, 3N/16 - 1 \\ \sin\left(\frac{\pi(i - 3N/16 + 0.5)}{N/4}\right) & i = 3N/16, \dots, 5N/16 - 1 \\ 1 & i = 5N/16, \dots, 11N/16 - 1 \\ \sin\left(\frac{\pi(i - 9N/16 + 0.5)}{N/4}\right) & i = 11N/16, \dots, 13N/16 - 1 \\ 0 & i = 13N/16, \dots, N - 1 \end{cases}$$

4. The size of the bit reservoir is minimized to reduce the delay contributed by it.

These techniques reduce the algorithmic delay to about 20 ms (for 48 kHz sampling rate), making full-duplex communications possible. However, this profile is still a full bandwidth coder like the traditional low complexity (LC) profile [26]. Such profiles use large bit rates such as 64kbps per channel for transparent audio quality. Thus the coding efficiency of AAC-LD is low and is not suitable for low bandwidth channels. MPEG-4 AAC Enhanced Low Delay (ELD) profile addresses this problem by incorporating spectral band replication (SBR) technology [49-50] to improve the coding efficiency.

3.3 MPEG-4 AAC Enhanced Low Delay Filterbanks

MPEG-4 AAC ELD profile improves the coding efficiency of the AAC-LD profile by incorporating the SBR bandwidth extension tool. This allows perceptually transparent audio quality at bit rates as low as 32 kbps per channel [20]. A modified SBR tool with reduced delay is used in conjunction with a new low delay core coder filterbank to minimize the overall delay.

The new core coder filterbank uses a different window function with multiple overlap to reduce delay without affecting the frequency selectivity of the filterbank

[20]. This window is used in conjunction with the MPEG-4 AAC ELD filterbanks originally proposed in [28].

Delay reduction is obtained by zeroing out parts of the window that overlap with future input samples. For example, if the frame size is 480 samples, the length of the analysis window is 1920 samples (overlap of 4 frames), the last 120 samples of which are zeros. Similarly, the first 120 samples of the synthesis window are zeros. Thus, the delay of the analysis-synthesis filterbank chain is reduced from $480 + 480 = 960$ samples to $(480 - 120) + (480 - 120) = 720$ samples. The analysis and synthesis windows are shown in Fig. 3.1 along with the traditional sine window that is used in AAC-LD. In Fig. 3.1, X-axis represents sample index and Y-axis represents window amplitudes.

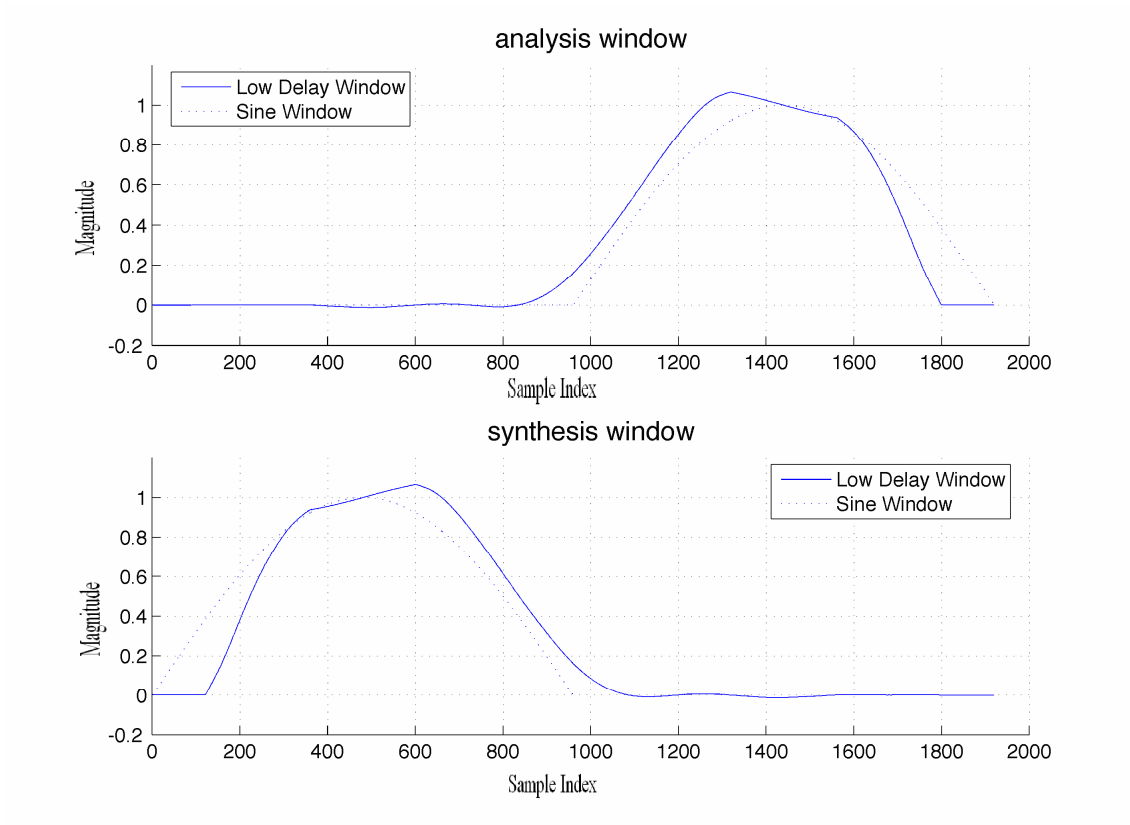


Fig. 3.1 Analysis and synthesis windows used in MPEG-4 AAC ELD core coder filterbanks [20]

Because of the longer overlap, the frequency selectivity of the new window function is comparable to that of the *sine* window (Fig. 3.2) and much better than that of the low-overlap window used in AAC-LD (Fig. 3.3).

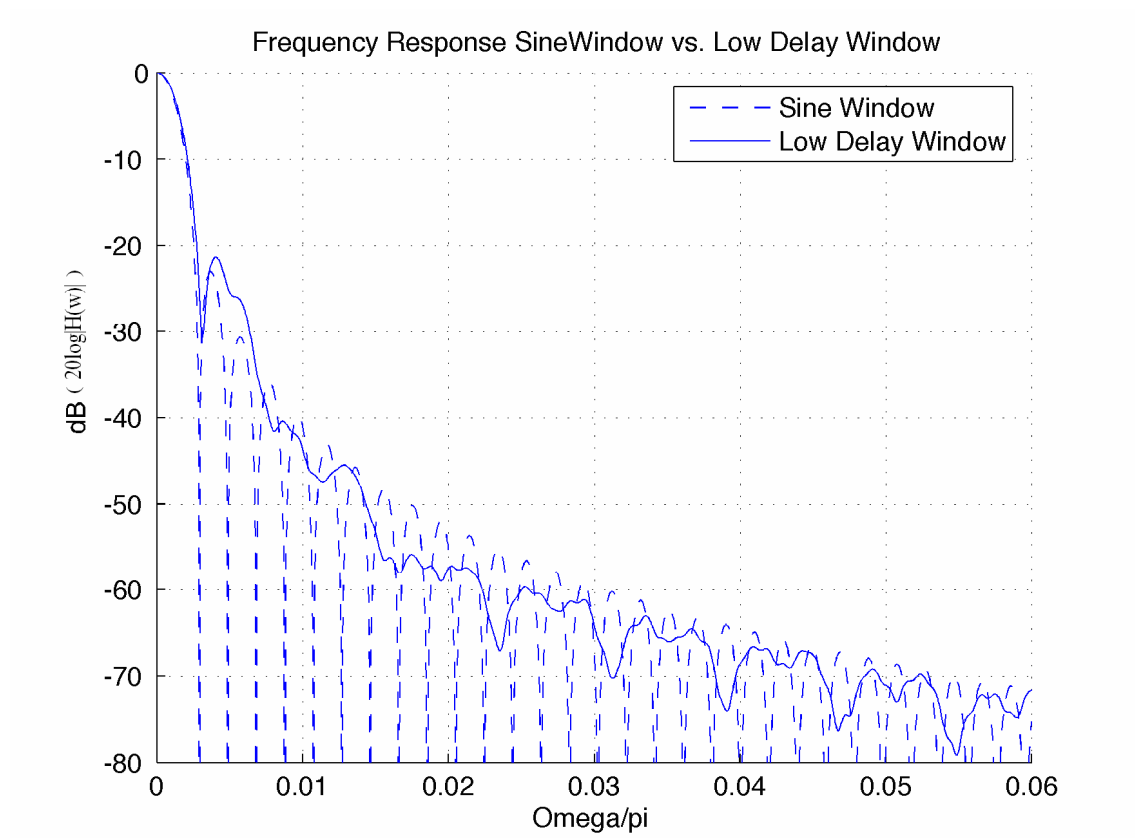


Fig. 3.2 Sine window vs low delay window [20]

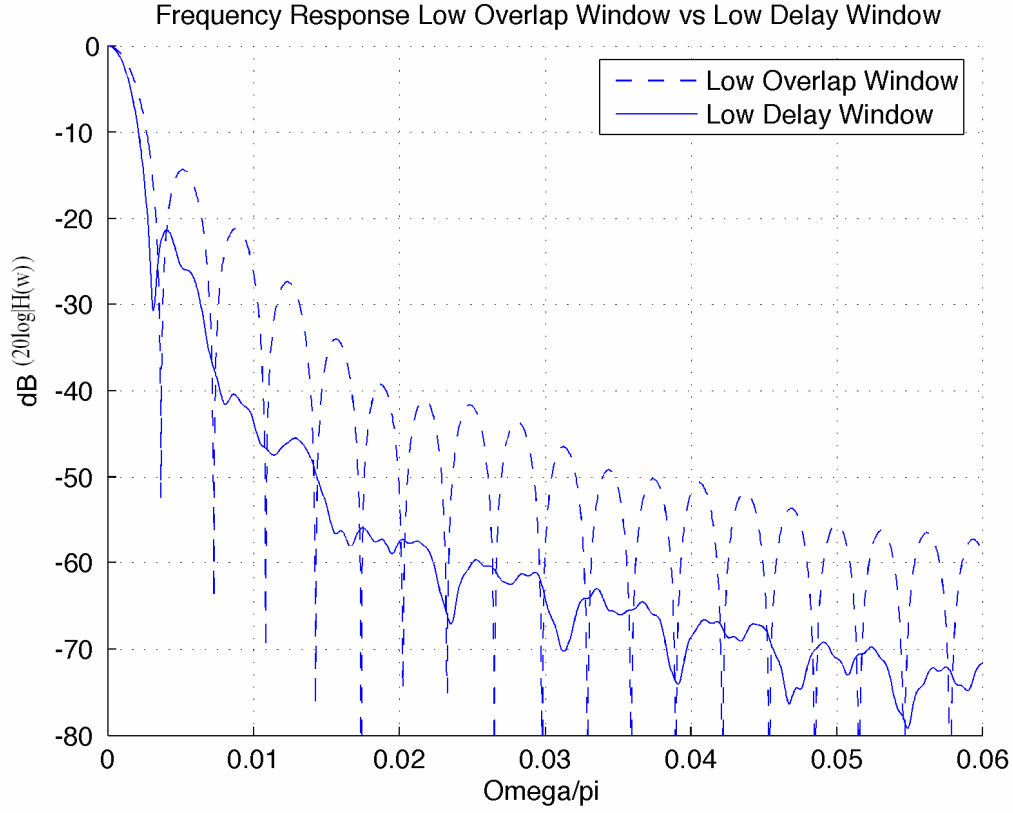


Fig. 3.3 Low overlap window vs low delay window [20]

In the next section, we describe our development of a mapping of the core coder ELD filterbanks to the well known MDCT. Our motivation for such development is the implementation of these new filterbanks using MDCT for which many fast algorithms already exist.

3.4 Mapping the ELD Filterbanks to MDCT

3.4.1 Definitions

The analysis filterbank in AAC-ELD is defined as follows [22]:

$$X_{i,k} = -2 \cdot \sum_{n=-N}^{N-1} z_{i,n} \cos\left(\frac{2\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right) \quad \text{for } 0 \leq k < N/2 \quad (3.1)$$

where,

- $z_{i,n}$ = input sequence multiplied by the analysis window
- n = sample index
- k = spectral coefficient index
- i = block index
- N = window length based on the window sequence value
- n_0 = $(-N / 2 + 1) / 2$

The synthesis filterbank in AAC-ELD is defined as follows [22]:

$$x_{i,n} = -\frac{2}{N} \sum_{k=0}^{\frac{N}{2}-1} spec[i][k] \cos\left(\frac{2\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right) \quad for \ 0 \leq n < 2N \quad (3.2)$$

where,

- n = sample index
- i = window index
- k = spectral coefficient index
- N = window length
- n_0 = $(-N / 2 + 1) / 2$
- $spec[i][k]$ = Received spectral coefficients

with $N = 960$ or 1024 .

The MDCT filterbank is defined as follows in [26]:

$$\hat{X}_{i,k} = 2 \cdot \sum_{n=0}^{N-1} \hat{z}_{i,n} \cos\left(\frac{2\pi}{N}(n+p_0)\left(k+\frac{1}{2}\right)\right), \ 0 \leq k < \frac{N}{2} \quad (3.3)$$

where:

- $\hat{z}_{i,n}$ = input sequence multiplied by the analysis window
- n = sample index
- k = spectral coefficient index
- i = block index
- N = window length
- p_0 = $(N / 2 + 1) / 2$

The IMDCT filterbank is defined as follows [26]:

$$\hat{x}_{i,n} = \frac{2}{N} \sum_{k=0}^{\frac{N}{2}-1} \text{spec}[i][k] \cos\left(\frac{2\pi}{N}(n+p_0)\left(k+\frac{1}{2}\right)\right) \quad \text{for } 0 \leq n < N \quad (3.4)$$

where:

n = sample index
 i = window index
 k = spectral coefficient index
 N = window length
 $p_0 = (N/2 + 1)/2$
 with $N = 1920$ or 2048 .

3.4.2 Mapping the ELD Analysis Filterbank to MDCT

Lemma 3.1: For $0 \leq k < \frac{N}{2}$,

$$X_{i, \left(\frac{N}{2}-1-k\right)} = (-1)^{\left(\frac{N}{2}-1-k\right)} \cdot 2 \sum_{n=0}^{N-1} \left\{ (-1)^{\left(n+\frac{N}{4}+1\right)} (z_{i,n} - z_{i,n-N}) \right\} \cos\left[\frac{2\pi}{N}(n+p_0)\left(k+\frac{1}{2}\right)\right] \quad (3.5)$$

Proof:

$$\begin{aligned}
 X_{i,k} &= -2 \cdot \left\{ \sum_{n=-N}^{-1} z_{i,n} \cos\left[\frac{2\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right] + \sum_{n=0}^{N-1} z_{i,n} \cos\left[\frac{2\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right] \right\} \\
 &= -2 \cdot \left\{ \sum_{n=0}^{N-1} z_{i,n-N} \cos\left[\frac{2\pi}{N}(n-N+n_0)\left(k+\frac{1}{2}\right)\right] + \sum_{n=0}^{N-1} z_{i,n} \cos\left[\frac{2\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right] \right\} \\
 &= -2 \cdot \sum_{n=0}^{N-1} (z_{i,n} - z_{i,n-N}) \cos\left[\frac{2\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right] \\
 &= -2 \cdot \sum_{n=0}^{N-1} (z_{i,n} - z_{i,n-N}) \cos\left[\frac{2\pi}{N}\left(n+p_0-\frac{N}{2}\right)\left(k+\frac{1}{2}\right)\right] \\
 &= -2 \cdot (-1)^k \sum_{n=0}^{N-1} (z_{i,n} - z_{i,n-N}) \sin\left[\frac{2\pi}{N}(n+p_0)\left(k+\frac{1}{2}\right)\right]
 \end{aligned}$$

$$\begin{aligned}
X_{i, \left(\frac{N}{2}-1-k\right)} &= -2 \cdot (-1)^{\left(\frac{N}{2}-1-k\right)} \sum_{n=0}^{N-1} (z_{i,n} - z_{i,n-N}) \sin \left[\frac{2\pi}{N} (n + p_0) \left(\frac{N}{2} - 1 - k + \frac{1}{2} \right) \right] \\
&= (-1)^{\left(\frac{N}{2}-1-k\right)} \cdot 2 \sum_{n=0}^{N-1} \left\{ (-1)^{\left(n+\frac{N}{4}+1\right)} (z_{i,n} - z_{i,n-N}) \right\} \cos \left[\frac{2\pi}{N} (n + p_0) \left(k + \frac{1}{2} \right) \right] \quad \square
\end{aligned}$$

It can be seen that the cosine kernel on RHS is the same as the MDCT kernel. MDCT is applied to the modified input shown in double braces. Hence the algorithm for implementing the analysis filterbank is:

1. Form the sequence $\{z_{i,n} - z_{i,(n-N)}\}$ for $0 \leq n < N$,
2. Invert the signs of the even indexed samples of this sequence if $N/4$ is even.
3. Invert the signs of the odd indexed samples of this sequence if $N/4$ is odd.
4. Apply MDCT,
5. Reverse the order of the output,
6. Invert the signs of the odd-indexed samples if $N/2$ is even,
7. Invert the signs of the even-indexed samples if $N/2$ is odd.

We can now develop the flow graph for the analysis filterbank as shown in Fig 3.4.

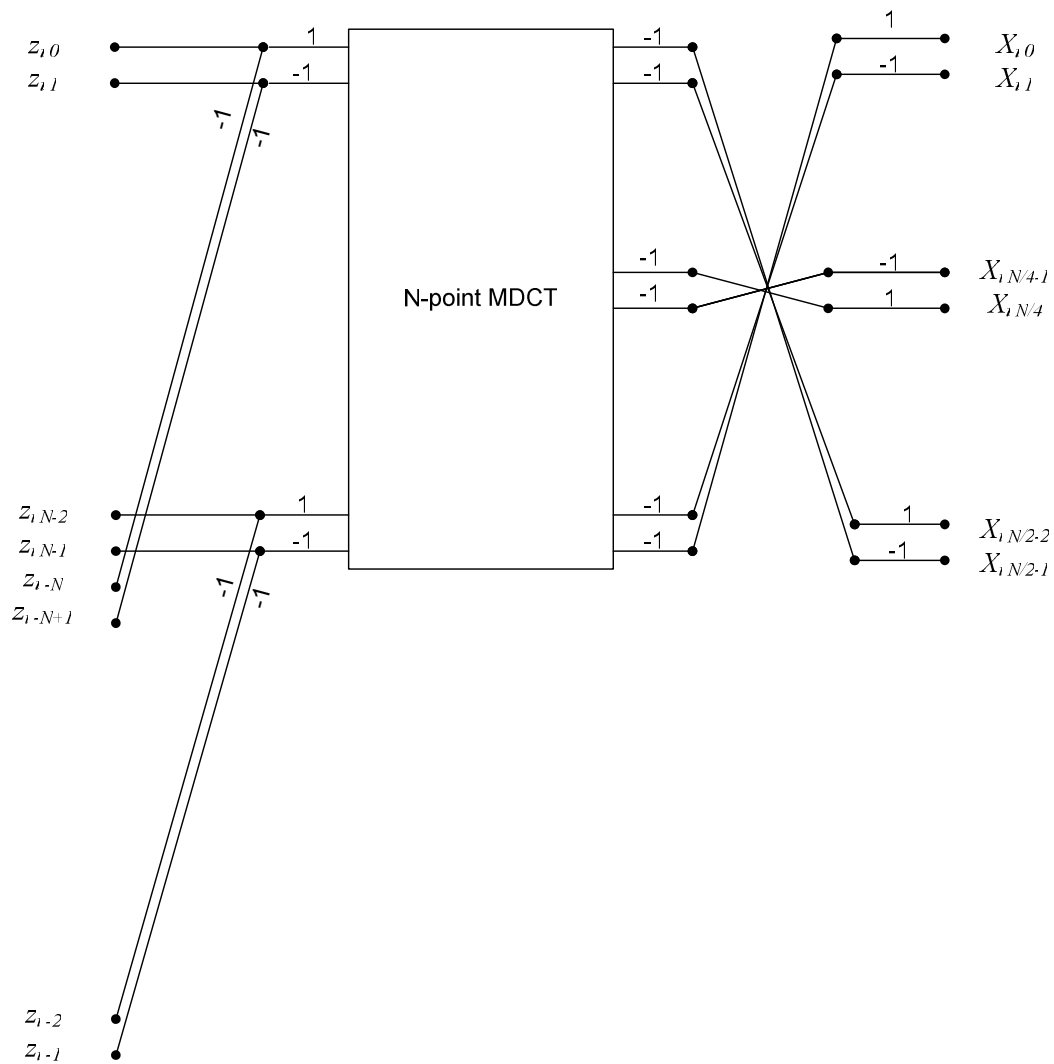


Fig. 3.4 Implementation of ELD analysis filterbank by MDCT

3.4.3 Mapping the ELD Synthesis Filterbank to IMDCT

A similar approach can be used to derive the mapping of the synthesis filterbank to IMDCT.

Lemma 3.2: $x_{i,n+N} = -x_{i,n}$ for $0 \leq n < N$. (3.6)

Proof:

$$\begin{aligned}
x_{i,n+N} &= -\frac{2}{N} \sum_{k=0}^{\frac{N}{2}-1} \text{spec}[i][k] \cos \left[\frac{2\pi}{N} (n+N+n_0) \left(k + \frac{1}{2} \right) \right] \\
&= \frac{2}{N} \sum_{k=0}^{\frac{N}{2}-1} \text{spec}[i][k] \cos \left[\frac{2\pi}{N} (n+n_0) \left(k + \frac{1}{2} \right) \right] \\
&= -x_{i,n}
\end{aligned}$$

From Lemma 3.2, we can observe that we need not calculate all the $2N$ output points. Instead, the last N output points can be obtained just by negating the first N output points. Now we focus on efficiently calculating the first N output points.

Lemma 3.3: For $0 \leq n < N$,

$$x_{i,n} = (-1)^{\binom{n+\frac{N}{4}+1}{2}} \cdot \frac{2}{N} \sum_{k=0}^{\frac{N}{2}-1} \left\{ (-1)^{\binom{\frac{N}{2}-1-k}{2}} \text{spec}[i] \left[\frac{N}{2} - 1 - k \right] \right\} \cos \left[\frac{2\pi}{N} (n+p_0) \left(k + \frac{1}{2} \right) \right] \quad (3.7)$$

Proof:

$$\begin{aligned}
x_{i,n} &= -\frac{2}{N} \sum_{k=0}^{\frac{N}{2}-1} \text{spec}[i][k] \cos \left[\frac{2\pi}{N} \left(n+p_0 - \frac{N}{2} \right) \left(k + \frac{1}{2} \right) \right] \\
&= -\frac{2}{N} \sum_{k=0}^{\frac{N}{2}-1} (-1)^k \text{spec}[i][k] \sin \left[\frac{2\pi}{N} (n+p_0) \left(k + \frac{1}{2} \right) \right] \\
&= -\frac{2}{N} \sum_{k=0}^{\frac{N}{2}-1} (-1)^{\binom{\frac{N}{2}-1-k}{2}} \text{spec}[i] \left[\frac{N}{2} - 1 - k \right] \sin \left[\frac{2\pi}{N} (n+p_0) \left(\frac{N}{2} - 1 - k + \frac{1}{2} \right) \right] \\
&= (-1)^{\binom{n+\frac{N}{4}+1}{2}} \cdot \frac{2}{N} \sum_{k=0}^{\frac{N}{2}-1} \left\{ (-1)^{\binom{\frac{N}{2}-1-k}{2}} \text{spec}[i] \left[\frac{N}{2} - 1 - k \right] \right\} \cos \left[\frac{2\pi}{N} (n+p_0) \left(k + \frac{1}{2} \right) \right] \quad \square
\end{aligned}$$

It can be seen that the cosine kernel on RHS is the same as the IMDCT kernel. IMDCT is applied to modified input as shown in the double braces. Hence the algorithm for implementing the ELD synthesis filterbank is:

1. Invert the signs of the odd-indexed spectral coefficients, $spec[i][k]$, if $N/2$ is even,
2. Invert the signs of the even-indexed spectral coefficients, $spec[i][k]$, if $N/2$ is odd,
3. Reverse the order of the above sequence,
4. Apply IMDCT,
5. Invert the signs of the even-indexed output samples if $N/4$ is even,
6. Invert the signs of the odd-indexed output samples if $N/4$ is odd,
7. The result of steps 5 and 6 form the first N outputs of the filterbank,
8. The remaining N output samples are obtained by inverting the signs of the first N samples.

The flow graph for implementing the synthesis filterbank can now be derived as shown in Fig. 3.5.

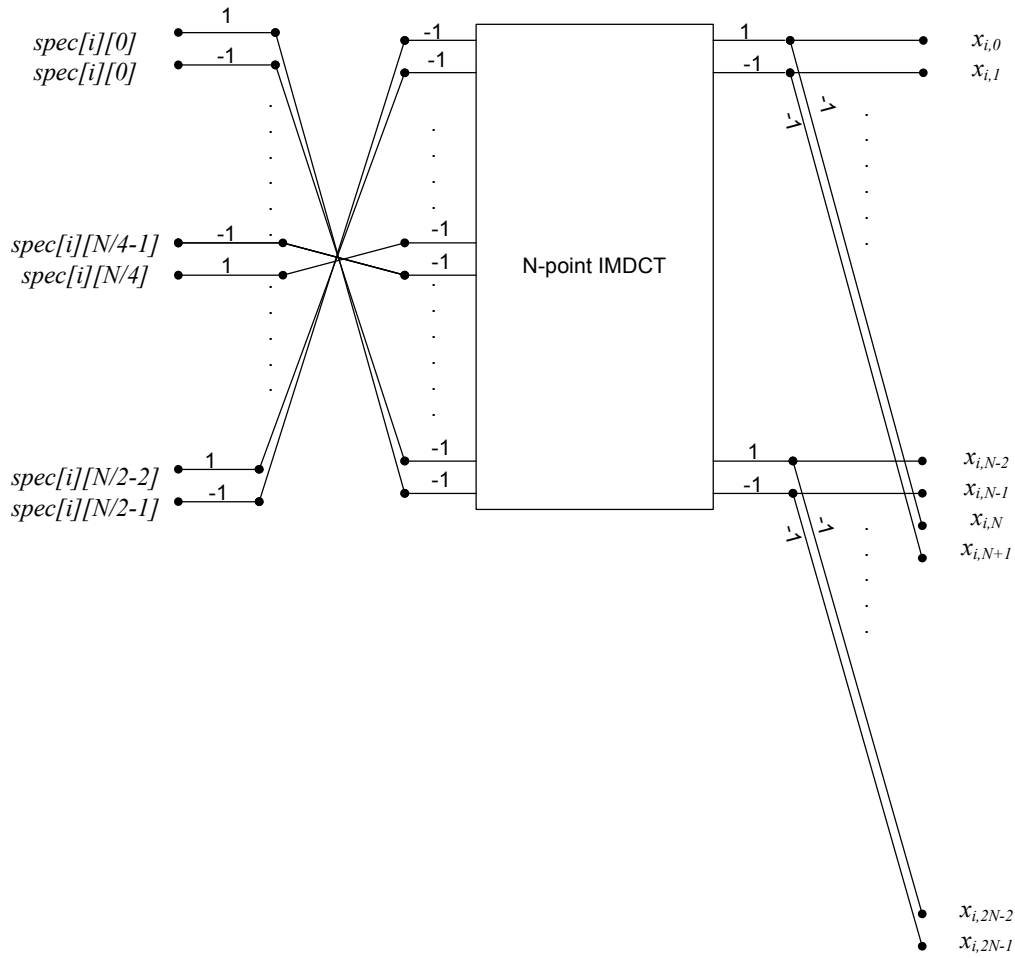


Fig. 3.5 Implementation of the ELD synthesis filterbank by IMDCT

3.4.4 Complexity Analysis

3.4.4.1 Complexity of non-MDCT part of our fast algorithm

For the mappings provided in section 3.4.2 and 3.4.3, it is clear that the complexity of the analysis filterbank is equal to the complexity of implementing an N -point MDCT plus N additions. Before the analysis filterbank is applied, we also require $2N$ multiplications for windowing. The complexity of the synthesis filterbank is the same as that of an N -point IMDCT. After the synthesis filterbank is applied, we also

require $2N$ multiplications and $3N/2$ additions for windowing and overlap-add operation.

3.4.4.2 Complexity of MDCT

We can use the efficient MDCT algorithm given in chapter 2 for implementing the ELD filterbanks. The complexity numbers given in section 2.5 are also applicable here. The complexity analysis of ELD filterbanks given in this section assumes the usage of the algorithms in chapter 2.

3.4.4.3 Complexity of our fast algorithm

Assume N is a power of 2. Let $RM_A(N)$ and $RA_A(N)$ denote, respectively, the number of real multiplications and additions required for the analysis filterbank along with windowing operation. Let $RM_S(N)$ and $RA_S(N)$ denote the corresponding numbers for the synthesis filterbank along with windowing and overlap-add operation. Then,

$$RM_A(N) = RM_S(N) = 0.25N(\log_2 N - 1) + 2N \quad (3.8)$$

$$RA_A(N) = RA_S(N) = 0.75N(\log_2 N + 1) \quad (3.9)$$

Thus for $N = 1024$, we require 4352 multiplications and 8448 additions. On the other hand, a brute force approach requires more than a million multiplications and additions.

The other commonly used length for MPEG-4 AAC-ELD is 960 which has a factor of 15. Assume $N = 15 \times 2^m$ ($m \geq 2$). Efficient algorithms for MDCT and IMDCT of this type of length are presented in chapter 2. Using the results from chapter 2, the computational complexity of the filterbanks can be shown as:

$$RM_A(15 \times 2^m) = RM_S(15 \times 2^m) = 15 \times 2^{m+1} + 2^{m-1} RM_D(15) + 15(m-1)2^{m-2} \quad (3.10)$$

$$RA_A(15 \times 2^m) = RA_S(15 \times 2^m) = (59 + RA_D(15))2^{m-1} + 45(m-1)2^{m-2} \quad (3.11)$$

where $RM_D(15)$ and $RA_D(15)$ are the multiplicative and additive complexities of the fast 15-point DCT algorithm presented in chapter 3. $RM_D(15) = 17$ and $RA_D(15) = 67$. Thus, for $N = 960$, we have 3664 multiplications and 7632 additions. Again, if we were to implement the filterbanks using brute force approach, it will take more than a million multiplications and additions.

CHAPTER 4

RADIX-5 DECOMPOSITION OF DCT

4.1 Introduction

In chapter 2, we presented efficient algorithms of MDCT and IMDCT for certain composite transform lengths. We observed that DCT-II is the major computational unit in those algorithms. In most of the cases, these algorithms are implemented on mobile devices which have limited processing power. Also, these devices use fixed-point processors to reduce the system cost. Hence, there is a need for implementing the algorithms in fixed point using integer arithmetic. Under these conditions, apart from computationally efficient algorithms, it is also important that the algorithms have good numerical accuracy.

It is shown in [23] and [24] that DCT algorithms following Hou's approach of recursive computation of DCT [47] (computing an N -point DCT from two $N/2$ -point DCTs) are numerically less accurate than other algorithms such as Vetterli's [48] especially if less bit depth is available for representing variables. Since Kok's algorithm used in chapter 2 is a realization of Hou's algorithm, its fixed point accuracy is not expected to be good at low bit depths.

In this thesis, we are interested in composite lengths such as $N = 5 \times 2^m$. As noted in chapter 2, there are not many DCT algorithms for even lengths. One alternative to Kok's algorithm is to use prime factor algorithms such as [44, 45] which split an N -

point DCT with mutually prime factors N_1 and N_2 (i.e., $N = N_1 N_2$) into a two-dimensional $N_1 \times N_2$ DCT. DCT is then applied along each of the dimensions and the results are combined through input and output permutations. For $N = 5 \times 2^m$, this results in a two dimensional 5×2^m DCT where we would have 5-point DCTs and 2^m -point DCTs. We could then apply numerically stable algorithms such as [48] for the 2^m -point DCTs. The short 5-point DCT modules also have good numerical accuracy. Hence, we would have a good fixed-point algorithm for the original length. However, apart from higher computational complexity, these algorithms also have complex input and output mappings and matrix transpose operations in between. For large transform lengths, these operations require large data moves and hence are very costly.

In this chapter, we develop a new radix-5 decomposition of DCT which may be useful for implementing accurate fixed-point DCT-II for lengths of the type $N = 5 \times 2^m$. We use an approach similar to radix-3 and radix-6 DCT algorithms presented in [46]. Application of radix-5 decomposition splits an N -point DCT into five $N/5$ -point DCTs with some pre and post processing. Thus, in our case, this would result in five 2^m -point DCTs each of which can be implemented using say, Vetterli's algorithm [48]. This algorithm doesn't have complex input and output permutations and hence can be easily implemented. It is not as efficient as Kok's algorithm but since the decomposition is applied only once in our case, the effect on overall computational complexity is minimal.

4.2 Radix-5 DCT

Assume N is a multiple of 5. For this section, we define DCT as follows:

$$\text{DCT: } X(k) = \sum_{n=0}^{N-1} x(n) \cos\left(\frac{(2n+1)k\pi}{2N}\right); \quad k = 0, \dots, N-1 \quad (4.1)$$

Now, from (4.1),

$$\begin{aligned} X(5k) &= \sum_{n=0}^{N-1} x(n) \cos\left\{\frac{(2n+1)k\pi}{2(N/5)}\right\}, \quad k = 0, \dots, \frac{N}{5}-1 \\ &= \sum_{i=0}^4 \sum_{n=\frac{iN}{5}}^{\frac{(i+1)N}{5}-1} x(n) \cos\left\{\frac{(2n+1)k\pi}{2(N/5)}\right\} \end{aligned} \quad (4.2)$$

We note that,

$$\begin{aligned} \sum_{n=\frac{N}{5}}^{\frac{2N}{5}-1} x(n) \cos\left\{\frac{(2n+1)k\pi}{2(N/5)}\right\} &= \sum_{n=0}^{\frac{N}{5}-1} x\left(\frac{2N}{5}-1-n\right) \cos\left\{\frac{\left[2\left(\frac{2N}{5}-1-n\right)+1\right]k\pi}{2(N/5)}\right\} = \sum_{n=0}^{\frac{N}{5}-1} x\left(\frac{2N}{5}-1-n\right) \cos\left\{\frac{(2n+1)k\pi}{2(N/5)}\right\} \\ \sum_{n=\frac{2N}{5}}^{\frac{3N}{5}-1} x(n) \cos\left\{\frac{(2n+1)k\pi}{2(N/5)}\right\} &= \sum_{n=0}^{\frac{N}{5}-1} x\left(\frac{2N}{5}+n\right) \cos\left\{\frac{\left[2\left(\frac{2N}{5}+n\right)+1\right]k\pi}{2(N/5)}\right\} = \sum_{n=0}^{\frac{N}{5}-1} x\left(\frac{2N}{5}+n\right) \cos\left\{\frac{(2n+1)k\pi}{2(N/5)}\right\} \\ \sum_{n=\frac{3N}{5}}^{\frac{4N}{5}-1} x(n) \cos\left\{\frac{(2n+1)k\pi}{2(N/5)}\right\} &= \sum_{n=0}^{\frac{N}{5}-1} x\left(\frac{4N}{5}-1-n\right) \cos\left\{\frac{\left[2\left(\frac{4N}{5}-1-n\right)+1\right]k\pi}{2(N/5)}\right\} = \sum_{n=0}^{\frac{N}{5}-1} x\left(\frac{4N}{5}-1-n\right) \cos\left\{\frac{(2n+1)k\pi}{2(N/5)}\right\} \\ \sum_{n=\frac{4N}{5}}^{N-1} x(n) \cos\left\{\frac{(2n+1)k\pi}{2(N/5)}\right\} &= \sum_{n=0}^{\frac{N}{5}-1} x\left(\frac{4N}{5}+n\right) \cos\left\{\frac{\left[2\left(\frac{4N}{5}+n\right)+1\right]k\pi}{2(N/5)}\right\} = \sum_{n=0}^{\frac{N}{5}-1} x\left(\frac{4N}{5}+n\right) \cos\left\{\frac{(2n+1)k\pi}{2(N/5)}\right\} \end{aligned}$$

Substituting these in the above equation and simplifying we get,

$$X(5k) = \sum_{n=0}^{N/5-1} \{a(n) + b(n) + c(n) + d(n) + e(n)\} \cos\left(\frac{(2n+1)k\pi}{2(N/5)}\right)$$

where,

$$\left. \begin{aligned} a(n) &= x(n), \\ b(n) &= x\left(\frac{2N}{5} - 1 - n\right), \\ c(n) &= x\left(\frac{2N}{5} + n\right), \\ d(n) &= x\left(\frac{4N}{5} - 1 - n\right), \\ e(n) &= x\left(\frac{4N}{5} + n\right) \end{aligned} \right\} n = 0, \dots, \frac{N}{5} - 1 \quad (4.3)$$

It can be seen that from equation (4.3) that $X(5k)$ is a $N/5$ -point DCT. Now consider:

$X(5k+i) + X(5k-i)$ for $i = 1, 2, 3$ and 4 .

$$\begin{aligned} X(5k+i) + X(5k-i) &= \sum_{n=0}^{N-1} x(n) \left\{ \cos\left(\frac{(2n+1)(5k+i)\pi}{2N}\right) + \cos\left(\frac{(2n+1)(5k-i)\pi}{2N}\right) \right\} \\ &= \sum_{n=0}^{N-1} \left\{ 2x(n) \cos\left(\frac{(2n+1)i\pi}{2N}\right) \right\} \cos\left(\frac{(2n+1)k\pi}{2(N/5)}\right) \end{aligned} \quad (4.4)$$

The RHS in (4.4) is similar to what we had for $X(5k)$ in (4.2). Hence following a similar procedure for (4.4), it can be converted to:

$$\begin{aligned} X(5k+i) + X(5k-i) &= \\ \sum_{n=0}^{\frac{N}{5}-1} \{a_i(n) + b_i(n) + c_i(n) + d_i(n) + e_i(n)\} \cos\left(\frac{(2n+1)k\pi}{2\left(\frac{N}{5}\right)}\right); & k = 0, \dots, \frac{N}{5} - 1; i = 1, \dots, 4 \end{aligned} \quad (4.5)$$

where,

$$\left. \begin{aligned}
a_i(n) &= 2a(n) \cos\left(\frac{(2n+1)(i\pi)}{2N}\right), \\
b_i(n) &= 2b(n) \cos\left(\frac{\left\{2\left(\frac{2N}{5}-1-n\right)+1\right\}(i\pi)}{2N}\right), \\
c_i(n) &= 2c(n) \cos\left(\frac{\left\{2\left(\frac{2N}{5}+n\right)+1\right\}(i\pi)}{2N}\right), \\
d_i(n) &= 2d(n) \cos\left(\frac{\left\{2\left(\frac{4N}{5}-1-n\right)+1\right\}(i\pi)}{2N}\right), \\
e_i(n) &= 2e(n) \cos\left(\frac{\left\{2\left(\frac{4N}{5}+n\right)+1\right\}(i\pi)}{2N}\right),
\end{aligned} \right\} n = 0, \dots, \frac{N}{5}-1 \quad (4.6)$$

Thus, we are able to split the N -point DCT into five $N/5$ -point DCTs. We note that,

$$X(k) = X(-k); \quad k = 0, \dots, N-1. \quad (4.7)$$

Hence,

$$\begin{aligned}
X(0) &= \sum_{n=0}^{N-1} x(n) \\
X(i) &= 0.5 \times \sum_{n=0}^{\frac{N}{5}-1} \{a_i(n) + b_i(n) + c_i(n) + d_i(n) + e_i(n)\}; \quad i = 1, \dots, 4.
\end{aligned} \quad (4.8)$$

The remaining DCT coefficients can be obtained from the recursive relation,

$$X(5k+i) = X_i(k) - X(5k-i), \quad k=1, \dots, \frac{N}{5}-1; \quad i=1, \dots, 4$$

where,

(4.9)

$$X_i(k) = \sum_{n=0}^{\frac{N}{5}-1} \{a_i(n) + b_i(n) + c_i(n) + d_i(n) + e_i(n)\} \cos \left(\frac{(2n+1)k\pi}{2\left(\frac{N}{5}\right)} \right)$$

4.3 Complexity Analysis

In this section, we give the complexity analysis of the algorithm assuming that N is a power of 5, i.e., $N = 5^m$. We also assume that in the last stage we will be using the fast 5-point DCT algorithm proposed in chapter 2 which can be implemented using 5 multiplications and 13 additions.

It is clear that in each stage we would be performing $4N$ multiplications by the *cosine* factors. Also, there are $4N$ additions in the $N/5$ -point DCTs. Additionally, we have $4N/5-4$ additions in the recursive computation part. Hence, we have in total $4(6N-5)/5$ additions. Let $RM(N)$ and $RA(N)$ denote the number of multiplications and additions respectively. Then,

$$\begin{aligned} RM(N) &= 4N + 5RM\left(\frac{N}{5}\right) \\ RA(N) &= \frac{4(6N-5)}{5} + 5RA\left(\frac{N}{5}\right) \end{aligned}$$
(4.10)

Solving these recursive equations we get,

$$\left. \begin{aligned} RM(5^m) &= (4m-3)5^m \\ RA(5^m) &= 12(2m-1)5^{m-1} + 1 \end{aligned} \right\} m > 0$$
(4.11)

Application of this algorithm once for MDCT of size 640, as discussed in chapter 2, results in five 64-point DCTs. The total complexity of MDCT would be 2880 multiplications and 4736 additions. However, this algorithm does not have any intricate input and output permutations or matrix transpose operations needed in PFA DCT algorithms.

CHAPTER 5

CONCLUSIONS

In this thesis, we have presented fast algorithms for several filterbanks used in audio coding. In particular, we presented fast algorithms for MDCT and IMDCT of lengths 5×2^m and 15×2^m , fast algorithms for the core coder filterbanks used in MPEG-4 AAC ELD. The MDCT algorithms were based on a mapping to DCT-IV which in turn is mapped to DCT-II. We used efficient algorithms for even length DCT in conjunction with our proposed fast 5-point and 15-point DCT modules. We showed that the modified window function is still piece-wise symmetric, thus requiring no extra memory to store it. We also derived a radix-5 decomposition of DCT which may be useful in the fixed-point implementation of the proposed MDCT algorithm. Complexity analysis for all the algorithms and comparisons are given.

The work presented in this thesis can be extended in the following ways:

1. Fast algorithms for the low delay SBR filterbanks used in MPEG-4 AAC ELD can also be derived.
2. An exhaustive fixed point error analysis of various MDCT algorithms can be carried out including the algorithms proposed in this thesis and the radix-5 DCT algorithm.
3. Prime factor algorithms directly for the MDCT can also be explored.

APPENDIX A

MATLAB CODE FOR FAST 15-POINT DCT

```

function [dct_15] = dct_15_unrolled(x)
%15 point DCT for real input using WFTA and Heideman's mapping.
%x is the input vector and dct_15 is the output vector (no scaling
done).
%Complexity: 17 multiplications and 67 additions

%application of T3 matrix...3 adds each
idx_map = [8 13 3];
s1 = x(idx_map(2)) + x(idx_map(3));
x(idx_map(3)) = x(idx_map(2)) - x(idx_map(3));
x(idx_map(2)) = s1;
x(idx_map(1)) = s1 + x(idx_map(1));

idx_map = [11 1 10];
s1 = x(idx_map(2)) + x(idx_map(3));
x(idx_map(3)) = x(idx_map(2)) - x(idx_map(3));
x(idx_map(2)) = s1;
x(idx_map(1)) = s1 + x(idx_map(1));

idx_map = [2 12 9];
s1 = x(idx_map(2)) + x(idx_map(3));
x(idx_map(3)) = x(idx_map(2)) - x(idx_map(3));
x(idx_map(2)) = s1;
x(idx_map(1)) = s1 + x(idx_map(1));

idx_map = [14 7 4];
s1 = x(idx_map(2)) + x(idx_map(3));
x(idx_map(3)) = x(idx_map(2)) - x(idx_map(3));
x(idx_map(2)) = s1;
x(idx_map(1)) = s1 + x(idx_map(1));

idx_map = [5 6 15];
s1 = x(idx_map(2)) + x(idx_map(3));
x(idx_map(3)) = x(idx_map(2)) - x(idx_map(3));
x(idx_map(2)) = s1;
x(idx_map(1)) = s1 + x(idx_map(1));

%application of T5 to the the transpose of above result...8 adds each
m = zeros(6,3); k = 1;

idx_map = [8 11 2 14 5];
s1 = x(idx_map(2)) + x(idx_map(5)); s2 = x(idx_map(2)) -
x(idx_map(5));
s3 = x(idx_map(4)) + x(idx_map(3)); s4 = x(idx_map(4)) -
x(idx_map(3));
s5 = s1 + s3; s6 = s1 - s3;
s7 = s4 + s2; s8 = s5 + x(idx_map(1));
m(1,k) = s8; m(2,k) = s5; m(3,k) = s6;
m(4,k) = s2; m(5,k) = s7; m(6,k) = s4;
k = k+1;

idx_map = [13 1 12 7 6];

```

```

s1 = x(idx_map(2)) + x(idx_map(5)); s2 = x(idx_map(2)) -
x(idx_map(5));
s3 = x(idx_map(4)) + x(idx_map(3)); s4 = x(idx_map(4)) -
x(idx_map(3));
s5 = s1 + s3; s6 = s1 - s3;
s7 = s4 + s2; s8 = s5 + x(idx_map(1));
m(1,k) = s8; m(2,k) = s5; m(3,k) = s6;
m(4,k) = s2; m(5,k) = s7; m(6,k) = s4;
k = k+1;

idx_map = [3 10 9 4 15];
s1 = x(idx_map(2)) + x(idx_map(5)); s2 = x(idx_map(2)) -
x(idx_map(5));
s3 = x(idx_map(4)) + x(idx_map(3)); s4 = x(idx_map(4)) -
x(idx_map(3));
s5 = s1 + s3; s6 = s1 - s3;
s7 = s4 + s2; s8 = s5 + x(idx_map(1));
m(1,k) = s8; m(2,k) = s5; m(3,k) = s6;
m(4,k) = s2; m(5,k) = s7; m(6,k) = s4;

%application of matrix C...17 multiplications (C(1,1) is 1)
u = -2*pi/5; v = -2*pi/3;
C = ...
    [1                                cos(v)-1
sin(v)
    (cos(u)+cos(2*u))/2-1            ((cos(u)+cos(2*u))/2-1)*(cos(v)-1)
    ((cos(u)+cos(2*u))/2-1)*sin(v)
    (cos(u)-cos(2*u))/2              (cos(u)-cos(2*u))/2*(cos(v)-1)
    (cos(u)-cos(2*u))/2*sin(v)
    (sin(u)+sin(2*u))                (sin(u)+sin(2*u))*(cos(v)-1)
    -(sin(u)+sin(2*u))*sin(v)
    sin(2*u)                         sin(2*u)*(cos(v)-1)
    -sin(2*u)*sin(v)
    (sin(u)-sin(2*u))                (sin(u)-sin(2*u))*(cos(v)-1)
    -(sin(u)-sin(2*u))*sin(v)];

m = C .* m;

%apply S5 matrix...5 adds each
Ev = zeros(8,1); Od = zeros(7,1);

%1st vector
s9 = m(1,1) + m(2,1); s12 = m(4,1) - m(5,1); s13 = m(5,1) + m(6,1);
s10 = s9 + m(3,1); s11 = s9 - m(3,1);
Ev(1) = m(1,1);
Ev(2) = s10; Od(1) = s12;
Ev(3) = s11; Od(2) = s13;

%2nd vector
s9 = m(1,2) + m(2,2); s12 = m(4,2) - m(5,2); s13 = m(5,2) + m(6,2);
s10 = s9 + m(3,2); s11 = s9 - m(3,2);
Ev(4) = m(1,2);

```

```

Ev(5) = s10; Od(3) = s12;
Ev(6) = s11; Od(4) = s13;

%3rd vector
s9 = m(1,3) + m(2,3); s12 = m(4,3) - m(5,3); s13 = m(5,3) + m(6,3);
s10 = s9 + m(3,3); s11 = s9 - m(3,3);
Od(5) = m(1,3);
Od(6) = s10; Ev(7) = s12;
Od(7) = s11; Ev(8) = s13;

%apply S3 matrix to the transpose of the above result...total 13 adds

%1st vector
Ev(4) = Ev(1) + Ev(4);

%2nd vector
Ev(5) = Ev(2) + Ev(5); Od(3) = Od(1) + Od(3);
r1 = Ev(5) + Ev(7); i1 = Od(3) + Od(6);
Ev(7) = Ev(5) - Ev(7); Od(6) = Od(3) - Od(6);
Ev(5) = r1; Od(3) = i1;

%3rd vector
Ev(6) = Ev(3) + Ev(6); Od(4) = Od(2) + Od(4);
r1 = Ev(6) + Ev(8); i1 = Od(4) + Od(7);
Ev(8) = Ev(6) - Ev(8); Od(7) = Od(4) - Od(7);
Ev(6) = r1; Od(4) = i1;

%finally apply the output permutation...
dct_15 = zeros(15,1);

dct_15(1) = Ev(1);
dct_15(3) = -Ev(5);
dct_15(5) = Ev(8);
dct_15(7) = -Ev(3);
dct_15(9) = Ev(7);
dct_15(11) = -Ev(4);
dct_15(13) = Ev(2);
dct_15(15) = -Ev(6);

dct_15(14) = Od(3);
dct_15(12) = -Od(7);
dct_15(10) = -Od(2);
dct_15(8) = Od(6);
dct_15(6) = -Od(5);
dct_15(4) = -Od(1);
dct_15(2) = Od(4);

```

APPENDIX B

ILLUSTRATION OF WFTA

```

function [] = fft_15()
P_inp = zeros(15,15);
P_out = zeros(15,15);

%Prime Factor Mapping (PFM) for N = 3*5
%3*5 means we first apply 5 point transforms and then 3 point
transforms.
%this will result in 67 real additions. if we use 5*3, we get 73 real
%additions. multiplications will be the same in either case. hence we
use
%the factorization 3*5

%mapping obtained by applying chinese remainder theorem as in the book
by
%Burrus
k = 1;
for n1 = 0:2
    for n2 = 0:4
        inp_idx_map(k) = mod(5*n1+3*n2,15);
        out_idx_map(k) = mod(10*n1+6*n2,15);
        k = k+1;
    end
end

inp_idx_map = inp_idx_map + 1;
out_idx_map = out_idx_map + 1;

%form the permutation matrices of input and output
for k = 1:15
    P_inp(k,inp_idx_map(k)) = 1;
    P_out(k,out_idx_map(k)) = 1;
end

%verify that the permuted transform matrix is equal to kronecker
product of
%prime factor transform matrices
P_out * fft(eye(15)) * inv(P_inp) - kron(fft(eye(3)),fft(eye(5)))

%define post addition matrix for tranform size 3; refer to Winograd
short-N
%DFT algorithms for the derivation of these matrices
S3 = [1      0      0
      1      1      1
      1      1     -1];

%multiplication matrix for length 3
C3 = diag([1 cos(-2*pi/3)-1 i*sin(-2*pi/3)]);

%pre additons matrix for length 3
T3 = [1      1      1
      0      1      1
      0      1     -1];

```

```

%post additions matrix for length 5
S5 = [1      0      0      0      0      0
      1      1      1      1     -1      0
      1      1     -1      0      1      1
      1      1     -1      0     -1     -1
      1      1      1     -1      1      0];

%multiplication matrix for length 5
u = -2*pi/5;
C5 = diag([1 (cos(u)+cos(2*u))/2-1 (cos(u)-cos(2*u))/2 ...
           i*(sin(u)+sin(2*u)) i*sin(2*u) i*(sin(u)-sin(2*u))]);

%pre additions matrix for length 5
T5 = [1      1      1      1      1
      0      1      1      1      1
      0      1     -1     -1      1
      0      1      0      0     -1
      0      1     -1      1     -1
      0      0     -1      1      0];

%verify eqn 16,17 in Silverman's paper
kron(S3,S5)*kron(C3,C5)*kron(T3,T5)-kron(fft(eye(3)),fft(eye(5)))

%form matrix C as defined eqn 24 of Silverman's paper
[r_C3,temp] = size(C3);
[r_C5,temp] = size(C5);

for j = 1:r_C5
    for q = 1:r_C3
        C(j,q) = (C5(j,j) * C3(q,q));
    end
end

%verify equation 24 in Silverman's paper....
fft_15 = zeros(15,15);
for vec = 1:15 %test for each basis vector
    clear z15; clear x; clear y;
    x = zeros(15,1);
    x(vec) = 1;

    %apply input permutation
    x = P_inp * x;

    %form matrix z15 as in eqn 22 of Silverman's paper
    q = 1;
    for j = 1:3
        for k = 1:5
            z15(j,k) = x(q);
            q = q+1;
        end
    end
end

```

```

end

z15 = S3 * (S5 * (C .* (T5 * (T3 * z15).')))).';

%form the output vector ... output scrambled
y = zeros(15,1);
q = 1;
for j = 1:3
    for k = 1:5
        y(q) = z15(j,k);
        q = q+1;
    end
end

%apply inverse output permutation to get the unscrambled output
y = inv(P_out) * y;
fft_15(1:end,vec) = y;
end
fft_15 = fft(eye(15))

```

REFERENCES

- [1] 3GPP TS 26.090, “AMR Speech Codec; Transcoding Functions”,
<http://www.3gpp.org/ftp/Specs/html-info/26090.htm>.
- [2] 3GPP2 C.S0014-0, “Enhanced Variable Rate Codec”, Version 1.0;
http://www.3gpp2.org/Public_html/specs/C.S0014-0_v1.0_revised.pdf.
- [3] ISO/IEC 11172-3:1993, “Information technology -- Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbps -- Part 3: Audio”.
- [4] Digital Audio Compression Standard (AC-3), Rev. B, Doc. A/52B, June 2005;
www.atsc.org/standards/a_52b.pdf.
- [5] ISO/IEC 13818-3: Information technology — Generic Coding of Moving Pictures and Associated Audio Information — Part 3: Audio, 1998.
- [6] Xiph.org Foundation, “Vorbis I Specification”,
http://xiph.org/vorbis/doc/Vorbis_I_spec.html.
- [7] Kondo, A.M., “Digital Speech: Coding for Low Bit Rate Communication Systems”, 2nd Ed., Wiley, 2004.
- [8] Painter, T., Spanias, A., “Perceptual Coding of Digital Audio”, *Proc. IEEE*, vol. 88, pp. 451-515, Apr. 2000.
- [9] Britanak V., Yip P., Rao K.R., “Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations”, Academic Press, 2006.

- [10] Duhamel, P.; Mahieux, Y.; Petit, J.P., "A Fast Algorithm for the Implementation of Filter Banks Based on `Time Domain Aliasing Cancellation'" in *Proc. Intl. Conf. Audio, Speech Signal Proc.,-1991*, pp. 2209-2212 vol. 3, 14-17 Apr 1991.
- [11] Fan, Y-H; Madisetti, V.K.; Mersereau, R.M., "On Fast Algorithms for Computing the Inverse Modified Discrete Cosine Transform," *Signal Processing Letters, IEEE* , vol. 6, no. 3, pp. 61-64, Mar 1999.
- [12] Chan, D.-Y.; Yang, J.-F.; Chen, S.-Y., "Regular Implementation Algorithms of Time Domain Aliasing Cancellation," *Vision, Image and Signal Processing, IEE Proceedings -* , vol. 143, no. 6, pp. 387-392, Dec 1996.
- [13] Chen, C. H., et al., "Recursive architectures for the forward and inverse modified discrete cosine transforms," *Signal Processing Systems, IEEE Workshop on*, pp. 50-59, Oct. 2000.
- [14] Britanak, V.; Rao, K.R., "A new fast algorithm for the unified forward and inverse MDCT/MDST computation" *Signal Processing*, vol. 82, issue 3, pp. 433-459, March 2002.
- [15] Shu, H.; Bao, X.; Toumoulin, C.; Luo, L., "Radix-3 Algorithm for the Fast Computation of Forward and Inverse MDCT," *Signal Processing Letters, IEEE* , vol. 14, no. 2, pp. 93-96, Feb. 2007.
- [16] Britanak, V.; Rao, K.R., "An Efficient Implementation of the Forward and Inverse MDCT in MPEG Audio Coding", *Signal Processing Letters, IEEE*, vol. 8, no. 2, pp. 48-51, Feb 2001.

- [17] 3GPP2 C.S0014-C v1.0, “Enhanced Variable Rate Codec, Speech Service Option 3, 68 and 70 for Wideband Spread Spectrum Digital Systems”, 2007.
- [18] ITU Recommendation G.722.1, “Low-complexity coding at 24 and 32 kbit/s for hands-free operation in systems with low frame loss”, 2005.
- [19] ITU-T SG16 Q9 – Contribution 199: Extended high-level description of the Q9 EV-VBR baseline codec, June 2007.
- [20] Schnell, M. et al., “Proposal for an Enhanced Low Delay Coding Mode”, M13958, *ISO/IEC JTC1/SC29/WG11*, Oct. 2006, China.
- [21] Schnell, M. et al., “Proposed Improvement for Enhanced Low Delay AAC”, M14230, *ISO/IEC JTC1/SC29/WG11*, Jan. 2007, Marrakech, Morocco.
- [22] ISO/IEC 14496-3:2005/FPDAM9, “Enhanced Low Delay AAC”, Apr. 2007.
- [23] Yun, I.D.; Lee, S.U.; “On the Fixed-Point-Error Analysis of Several Fast DCT Algorithms”, *Circuits, Syst., Video Tech., IEEE Trans. on*, vol. 3, no. 1, pp. 27-41, Feb 1993.
- [24] Yun, I.D.; Lee, S.U., “On the Fixed-Point-Error Analysis of Several Fast IDCT Algorithms”, *Circuits and Systems II – Analog and Digital Signal Processing , IEEE Trans. on*, vol. 42, no. 11, pp. 685-693, Nov 1995.
- [25] Spanias, A., “Speech Coding: A Tutorial Review”, *Proc. IEEE*, vol. 82, pp. 1541-1582, Oct. 1994.
- [26] ISO/IEC 14496-3: Subpart 4: “General Audio Coding (GA) - AAC, TwinVQ, BSAC”.

- [27] Herre, J., "Temporal Noise Shaping, Quantization and Coding Methods in Perceptual Audio Coding: A Tutorial Introduction", *High Quality Audio Coding, AES 17th Intl. Conference on*.
- [28] Schuller, G.D.T.; Karp, T., "Modulated Filter Banks with Arbitrary System Delay: Efficient Implementations and the Time-Varying Case", *Signal Processing, IEEE Trans. on*, vol. 48, no. 3, pp. 737-748, March 2000.
- [29] Cheng, M-H.; Hsu, Y-H., "Fast IMDCT and MDCT algorithms - a matrix approach," *Signal Processing, IEEE Transactions on*, vol. 51, no. 1, pp. 221-229, Jan. 2003.
- [30] Kok, C. W., "Fast Algorithm for Computing Discrete Cosine Transform," *Signal Processing, IEEE Transactions on*, vol. 45, no. 3, pp.757-760, Mar 1997.
- [31] Princen, J.; Johnson, A.; Bradley, A., "Subband / Transform coding using filter bank designs based on time domain aliasing cancellation," in *Proc. Intl. Conf. Audio, Speech Signal Proc.'87.*, vol. 12, pp. 2161-2164, Apr 1987.
- [32] Malvar, H., "Signal Processing with Lapped Transforms", Artech House, Boston, 1992.
- [33] Chivukula, R.K.; Reznik, Y.A., "Efficient Implementation of a Class of MDCT/IMDCT Filterbanks for Speech and Audio Coding Applications", accepted for *IEEE Intl. Conf. Audio, Speech Signal Proc. 2008*, Las Vegas.
- [34] Heideman, M.T.; "Computation of an Odd-Length DCT from a Real-Valued DFT of the Same Length", *Signal Proc., IEEE Trans. on*, vol. 40, no. 1, pp. 54-61, Jan 1992.

- [35] Sorensen, H.V.; et. al. “Real-Valued Fast Fourier Transform Algorithms”, *Acoust. Speech and Signal Proc., IEEE Trans. on*, vol. ASSP-35, no. 6, pp. 849-863, June 1987.
- [36] Burrus, C.S.; Eschenbacher, P.W.; “An In-Place, In-Order Prime Factor FFT Algorithm”, *Acoust. Speech and Signal Proc., IEEE Trans. on*, vol. ASSP-29, no. 4, pp. 806-817, Aug 1981.
- [37] Heideman, M.T.; Burrus, C.S.; Johnson, H.W.; “Prime Factor FFT Algorithms for Real-Valued Series”, *Proc. IEEE Intl. Conf. Audio, Speech Signal Proc.*, San Diego, pp. 28A.7.1-28A.7.4, March 1984.
- [38] Kolba, D.P.; Parks, T.W.; “A Prime Factor FFT Algorithm Using High-Speed Convolution”, *Acoust. Speech and Signal Proc., IEEE Trans. on*, vol. ASSP-25, pp. 281-294, Aug 1977.
- [39] Burrus, C.S.; Parks, T.W.; “DFT/FFT and Convolution Algorithms – Theory and Implementation”, Wiley, New York, 1985.
- [40] Nussbaumer, H.J.; “Fast Fourier Transform and Convolution Algorithms”, Springer, 1981.
- [41] Winograd, S.; “On Computing the Discrete Fourier Transform”, *Mathematics of Computation*, vol. 32, no. 141, pp. 175-199, Jan 1978.
- [42] Silverman, H.F.; “An Introduction to Programming the Winograd Fourier Transform Algorithm (WFTA)”, *Acoust. Speech and Signal Proc., IEEE Trans. on*, vol. ASSP-25, no. 2, pp. 152-165, April 1977.

- [43] Tseng, B.D.; Miller, W.C.; “Comments on ‘An Introduction to Programming the Winograd Fourier Transform Algorithm (WFTA)’ ”, *Acoust. Speech and Signal Proc., IEEE Trans. on*, vol. ASSP-26, no. 3, pp. 268-269, June 1978.
- [44] Yang, P.P.N.; Narasimha, M.J.; “Prime Factor Decomposition of the Discrete Cosine Transform and its Hardware Realization”, *Proc. IEEE Intl. Conf. Audio, Speech Signal Proc.-1985*, pp. 772-775, 1985.
- [45] Lee, B.G.; “Input and Output Index Mappings for a Prime-Factor-Decomposed Computation of Discrete Cosine Transform”, *Acoust. Speech and Signal Proc., IEEE Trans. on*, vol. 37, no. 2, pp. 237-244, Feb 1989.
- [46] Chan, Y.H.; Siu, W.C.; “Mixed-Radix Discrete Cosine Transform”, *Signal Proc., IEEE Trans. on*, vol. 41, no. 11, pp. 3157-3161, Nov. 1993.
- [47] Hou, H.S.; “A Fast Recursive Algorithm for Computing the Discrete Cosine Transform”, *Acoust. Speech and Signal Proc., IEEE Trans. on*, vol. ASSP-35, no. 10, pp. 1455-1461, Oct. 1987.
- [48] Vetterli, M.; Nussbaumer, H.J.; “Simple FFT and DCT Algorithms with Reduced Number of Operations”, *Signal Processing*, vol. 6, no. 4, pp. 267-278, 1984.
- [49] Dietz, M. et al., “Spectral Band Replication, a Novel Approach in Audio Coding”, *12th AES Convention*, Munich, Germany, Apr. 2002.
- [50] Ekstrand, P.; “Bandwidth Extension of Audio Signals by Spectral Band Replication”, *Model based Processing and Coding of Audio, Proc. 1st IEEE Benelux Workshop on*, pp. 53-58, Leuven, Belgium, Nov. 2002.

- [51] Bosi, M. and Goldberg, R.E.; “Introduction to digital audio coding standards”,
Norwell, MA: Kluwer, 2002.

BIOGRAPHICAL INFORMATION

Ravi Kiran Chivukula was born in Andhra Pradesh, India. He received his B.Tech. degree in Electronics and Communication Engineering from the National Institute of Technology, Warangal in 2002. From 2002 to 2006 he was with Emuzed India Pvt. Ltd. working on speech and audio coding systems and music synthesis engines. Since August 2006, he has been a graduate student at the University of Texas at Arlington. He is also a recipient of UTA Graduate Dean's Masters Fellowship. He worked as an intern in Qualcomm Corporate R&D from May 2007 to December 2007. Chivukula's research interests include digital signal processing, information theory and computer science. He is a member of Eta Kappa Nu and Tau Beta Pi.