Efficient Algorithms for MPEG-4 AAC-ELD, AAC-LD and AAC-LC Filterbanks

Ravi K. Chivukula[†], Yuriy A. Reznik[‡1], Venkat Devarajan[†]
[†]The University of Texas at Arlington, Email: {ravikiran.chivukula,venkat}@uta.edu
[‡]Stanford University, Email: yreznik@stanford.edu

Abstract

Recently, MPEG has completed work on a new low-delay audio codec called MPEG-4 AAC Enhanced Low Delay (ELD) targeting low bit rate, full-duplex communication applications such as audio and video conferencing. The AAC-ELD profile combines low delay SBR filterbanks with a new low delay core coder filterbank to achieve both high coding efficiency and low algorithmic delay. In this paper, we propose an efficient mapping of the AAC-ELD core coder filterbanks to the well known MDCT. This provides a fast algorithm for the new filterbanks. Since AAC-LD and AAC-LC profiles also use MDCT filterbanks, this mapping enables efficient joint implementation of filterbanks for all 3 profiles. We also present a very efficient 15-point DCT-II algorithm that is useful in all 3 profiles for frame lengths of 960 and 480. This algorithm requires just 17 multiplications and 67 additions. The design structure and complexity analysis for the filterbanks is also provided.

1. Introduction

Traditionally, speech and audio coding paradigms have been significantly different. Speech coding is primarily based on source modeling [1], and achieving low round-trip algorithmic delay [2] is considered essential for the intended applications (full-duplex communication systems). However, most speech codecs are only efficient in encoding single-speaker material and are unsuitable for generic audio content [6].

On the other hand, audio coding is traditionally based on modeling and exploiting the psychoacoustic characteristics of human auditory system [3]. Most of the codecs are intended for *perceptually transparent* reproduction of any generic music material. However, these codecs usually operate on long frame lengths for good frequency selectivity. They also typically use orthogonal filterbanks such as Modified Discrete Cosine Transform (MDCT) [5], due to which, the delay contributed by the filterbank depends on the length of the prototype low-pass filter [8]. Hence, they are usually characterized by high algorithmic delays, making them unsuitable for full-duplex communication. The MPEG-4 AAC-LC [9] is a well-known example of this type of codec.

MPEG-4 AAC Low Delay (LD) is the first audio codec that addressed the issue of high algorithmic delay [9]. It reduces delay by halving the frame length from 1024/960 to 512/480; by removing block switching (thereby avoiding the look-ahead delay) and by minimizing the use of a bit reservoir in the encoder. Though it could reduce the delay down to 20ms, it still requires bit rates close to 64kbps per channel in order to deliver satisfactory audio quality [6].

Recently, MPEG standardized a new algorithm - Enhanced Low Delay AAC (AAC-ELD) [6, 7, 22]. This codec addresses the drawbacks of AAC-LD by incorporating the low-delay spectral band replication (LD-SBR) tool and a new low-delay core coder filterbank. While the SBR technology [23-24] improves coding efficiency, LD-SBR tool also minimizes the introduced delay by avoiding the use of variable time grid [6, 22] and by using low-delay analysis and synthesis quadrature mirror filter (QMF) banks [22]. The delay of the new core coder filterbank is independent of filter length [6, 8] and hence, a window with multiple overlap for good frequency selectivity can be used. Parts of the window that access future input values are zeroed out, thus reducing the delay further. With these modifications, AAC-ELD achieves an algorithmic delay of only 31ms and delivers satisfactory audio quality at bit rates as low as 32kbps per channel [22].

[1] On leave from Qualcomm Inc, San Diego, CA.

In this paper, we present an efficient mapping of the AAC-ELD core coder filterbanks to the well known MDCT [5]. This mapping involves only permutations, sign changes and additions (only for analysis filterbank). Since many fast algorithms exist for MDCT, this mapping essentially provides a fast algorithm to implement the new filterbanks. Further, AAC-LC and AAC-LD profiles also use MDCT filterbanks. Thus, this mapping provides a common framework for the joint implementation of filterbanks in all 3 profiles. We also present a very efficient algorithm for 15-point DCT-II useful for frame lengths of 960 and 480. This algorithm is based on mapping the DCT to an equal length real input DFT. Complexity analysis of the AAC-ELD core coder filterbanks is provided at the end.

2. Definitions

The MPEG-4 AAC ELD core coder analysis and synthesis filterbanks are defined as follows [7]:

$$X(k) = -2\sum_{n=-N}^{N-1} z(n)\cos\left(\frac{2\pi}{N}\left(n+n_0\right)\left(k+\frac{1}{2}\right)\right) \quad for\ 0 \le k < \frac{N}{2}$$

$$x(n) = -\frac{2}{N}\sum_{k=0}^{\frac{N}{2}-1} X(k)\cos\left(\frac{2\pi}{N}\left(n+n_0\right)\left(k+\frac{1}{2}\right)\right) \quad for\ 0 \le n < 2N$$

where, $n_0 = (-N/4+1/2)$, *z(n)* denotes windowed input data samples, *X(k)* denotes subband coefficients, *x(n)* denotes reconstructed samples (prior to aliasing cancellation). *N* is 1024 or 960.

The MDCT and IMDCT are defined as follows [5,8]:

$$\tilde{X}(k) = 2.\sum_{n=0}^{N-1} z(n)\cos\left(\frac{2\pi}{N}(n+p_0)\left(k+\frac{1}{2}\right)\right); k = 0,....,N/2-1,$$

$$\tilde{x}(n) = \frac{2}{N}\sum_{k=0}^{N/2-1} \tilde{X}(k)\cos\left(\frac{2\pi}{N}(n+p_0)\left(k+\frac{1}{2}\right)\right); n = 0,1,....,N-1$$

where, $p_0 = (N/4+1/2)$, $\tilde{X}(k)$ denotes MDCT spectrum coefficients, $\tilde{x}(n)$ denotes reconstructed samples (prior to aliasing cancellation) and *N* is the length of the input sequence.

Hereafter, for brevity, we will use the terms DCT and IDCT to refer to DCT-II and IDCT-II transforms respectively without the normalization factors [4].

3. Mapping the Analysis Filterbank to MDCT

In the case of the analysis filterbank, for $0 \le k < \frac{N}{2}$,

$$\begin{aligned}
X(k) &= -2.\sum_{n=-N}^{-1} z(n)\cos\left[\frac{2\pi}{N}\left(n+n_0\right)\left(k+\frac{1}{2}\right)\right] \\
&\quad -2.\sum_{n=0}^{N-1} z(n)\cos\left[\frac{2\pi}{N}\left(n+n_0\right)\left(k+\frac{1}{2}\right)\right] \\
&= -2.\sum_{n=0}^{N-1} z(n-N)\cos\left[\frac{2\pi}{N}\left(n-N+n_0\right)\left(k+\frac{1}{2}\right)\right] \\
&\quad -2.\sum_{n=0}^{N-1} z(n)\cos\left[\frac{2\pi}{N}\left(n+n_0\right)\left(k+\frac{1}{2}\right)\right] \\
&= -2.\sum_{n=0}^{N-1}\left\{z(n)-z(n-N)\right\}\cos\left[\frac{2\pi}{N}\left(n+n_0\right)\left(k+\frac{1}{2}\right)\right] \\
&= -2.\sum_{n=0}^{N-1}\left\{z(n)-z(n-N)\right\}\cos\left[\frac{2\pi}{N}\left(n+p_0-\frac{N}{2}\right)\left(k+\frac{1}{2}\right)\right] \\
&= -2.(-1)^k\sum_{n=0}^{N-1}\left\{z(n)-z(n-N)\right\}\sin\left[\frac{2\pi}{N}\left(n+p_0\right)\left(k+\frac{1}{2}\right)\right]
\end{aligned}$$

$$\begin{aligned}
&X\left(\frac{N}{2}-1-k\right) = \\
&-2.(-1)^{\left(\frac{N}{2}-1-k\right)}\sum_{n=0}^{N-1}\left\{z(n)-z(n-N)\right\}\sin\left[\frac{2\pi}{N}\left(n+p_0\right)\left(\frac{N}{2}-1-k+\frac{1}{2}\right)\right] \\
&= (-1)^{\left(\frac{N}{2}-1-k\right)}.2\sum_{n=0}^{N-1}\left[(-1)^{\left(n+\frac{N}{4}+1\right)}\left\{z(n)-z(n-N)\right\}\right]\cos\left[\frac{2\pi}{N}\left(n+p_0\right)\left(k+\frac{1}{2}\right)\right]
\end{aligned}$$

We note that the summation on the RHS represents MDCT. Thus, the algorithm for implementing the analysis filterbank can be formulated as follows:

1. Form the sequence {*z(n)* – *z(n-N)*} for $0 \le n < N$,
2. Invert the signs of the even indexed samples if *N/4* is even or invert the signs of odd-indexed samples if *N/4* is odd,
3. Apply MDCT,
4. Reverse the order of the output,
5. Invert the signs of the odd-indexed samples if *N/2* is even or invert the signs of even-indexed samples if *N/2* is odd.

The flow graph for the analysis filterbank is shown in Fig. 1 assuming *N/4* is even. It can be observed that the complexity of the filterbank is *N* subtractions plus the complexity of MDCT.

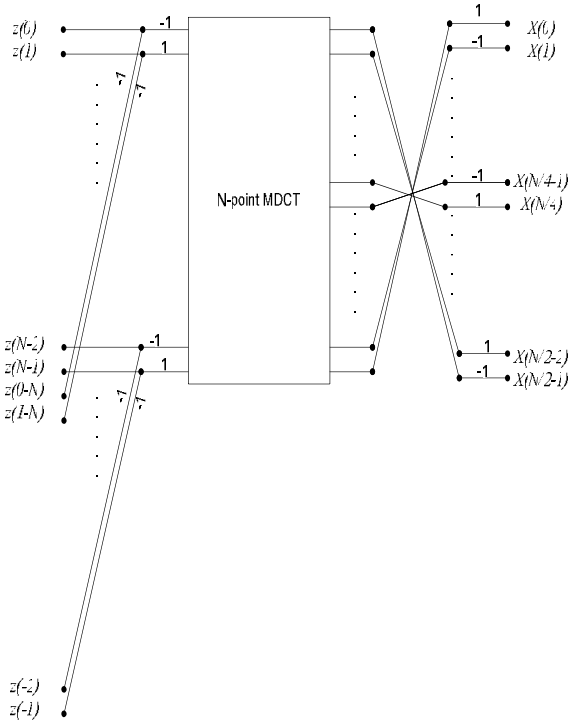

Fig 1. Flow graph for analysis filterbank

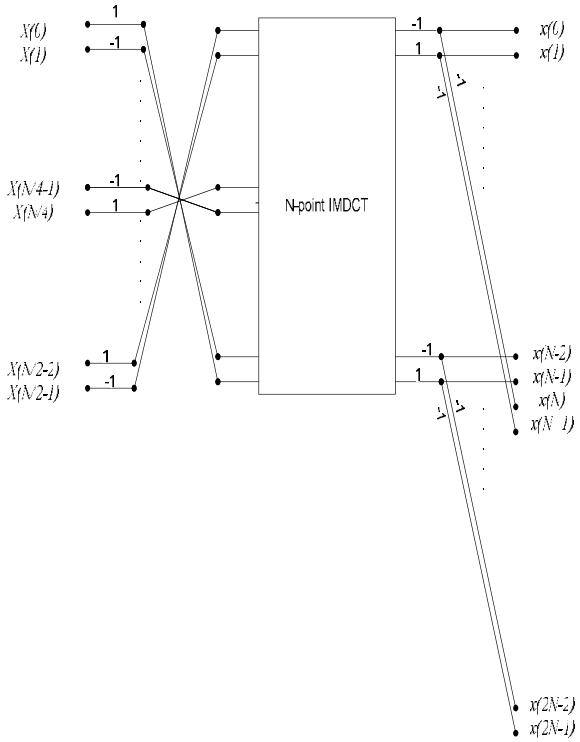

Fig 2. Flow graph for synthesis filterbank

4. Mapping the Synthesis Filterbank to IMDCT

In the case of synthesis filterbank, for $0 \le n < N$,

$$x(n+N) = -\frac{2}{N}\sum_{k=0}^{\frac{N}{2}-1} X(k)\cos\left[\frac{2\pi}{N}\left(n+N+n_0\right)\left(k+\frac{1}{2}\right)\right]$$

$$= \frac{2}{N}\sum_{k=0}^{\frac{N}{2}-1} X(k)\cos\left[\frac{2\pi}{N}\left(n+n_0\right)\left(k+\frac{1}{2}\right)\right]$$

$$= -x(n)$$

For $0 \le n < N$,

$$x(n) = -\frac{2}{N}\sum_{k=0}^{N/2-1} X(k)\cos\left[\frac{2\pi}{N}\left(n+p_0-\frac{N}{2}\right)\left(k+\frac{1}{2}\right)\right]$$

$$= -\frac{2}{N}\sum_{k=0}^{N/2-1} (-1)^k X(k)\sin\left[\frac{2\pi}{N}\left(n+p_0\right)\left(k+\frac{1}{2}\right)\right]$$

$$= -\frac{2}{N}\sum_{k=0}^{N/2-1} (-1)^{\left(\frac{N}{2}-1-k\right)} X\left(\frac{N}{2}-1-k\right)\sin\left[\frac{2\pi}{N}\left(n+p_0\right)\left(\frac{N}{2}-1-k+\frac{1}{2}\right)\right]$$

$$= \frac{2}{N}(-1)^{\left(n+\frac{N}{4}+1\right)}\sum_{k=0}^{N/2-1} (-1)^{\left(\frac{N}{2}-1-k\right)} X\left(\frac{N}{2}-1-k\right)\cos\left[\frac{2\pi}{N}\left(n+p_0\right)\left(k+\frac{1}{2}\right)\right]$$

We note that the summation on the RHS is an IMDCT.

Thus, the algorithm for the synthesis filterbank is:

1. Invert the signs of the odd-indexed spectral coefficients, *X(k)*, if *N/2* is even or invert the signs of even-indexed coefficients if *N/2* is odd,
2. Reverse the order of the above sequence,
3. Apply IMDCT,
4. Invert the signs of the even-indexed output samples if *N/4* is even or invert the signs of odd-indexed samples if *N/4* is odd; these form the first *N* output points of the filterbank,
5. The remaining *N* output samples are obtained by inverting the signs of the first *N* samples.

The flow graph for the synthesis filterbank is shown in Fig. 2 assuming *N/4* is even. It can be seen that the complexity of the synthesis filterbank is the same as that of IMDCT.

5. Implementation of MDCT

The problem of efficient implementation of MDCT and IMDCT transforms has already been well studied, and a number of algorithms can be found in the literature – see Malvar [5] and references therein. However, most of such existing fast algorithms (e.g. FFT-based algorithm of Duhamel et al. [10]), are only

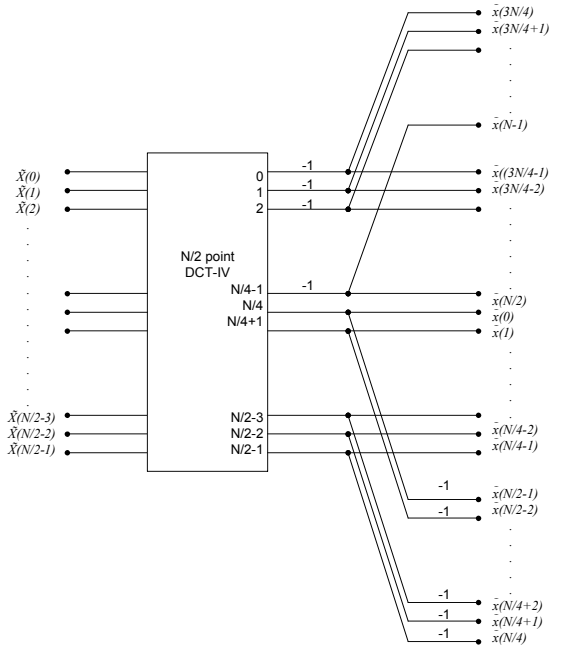

Fig 3. Mapping IMDCT to DCT-IV [11]

suitable for implementing transforms with power-of-2 lengths.

A more general framework for implementing MDCT/IMDCT of even lengths has been recently described by Cheng and Hsu [11].

Their algorithm maps MDCT/IMDCT to DCT-IV. The DCT-IV in turn can be mapped to DCT/IDCT with pre/post additions and multiplications [12]. Using the involutory property of the DCT-IV matrix, it is possible to merge the pre/post multiplications in DCT-IV with the windowing stage, thus reducing the number of multiplications and the storage requirement. The flow
graph for mapping IMDCT to DCT-IV is shown in Fig. 3. The flow graph for mapping DCT-IV to IDCT is shown in Fig. 4.

An algorithm that is applicable for any even length DCT is given by Kok [12]. Kok's algorithm is a decimation-in-frequency strategy, splitting an *N*-point DCT into two *N/2*-point DCTs. The algorithm was shown to be optimal for both power-of-2 lengths and also even lengths. Thus, the algorithm can be used even for MDCT block lengths such as 960/120 in AAC-LC and 480 in AAC-LD and AAC-ELD.

Usage of this overall scheme leads to a very efficient MDCT implementation and we assume this scheme in our complexity analysis given in section 7. Details of this algorithm are also given in [13].

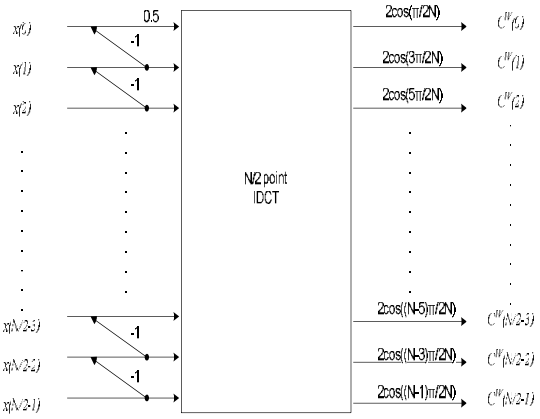

Fig 4. Mapping DCT-IV to IDCT [12]

6. Fast 15-point DCT Algorithm

As noted in section 5, DCT-based MDCT/IMDCT algorithms are computationally very efficient. Generally, radix-2 algorithms (which split an *N*-point transform into two *N/2*-point transforms) such as [12] are used for the implementation of DCT. Recursive application of such algorithms for transform lengths like 960 (64×15) and 480 (32×15) eventually leads to a 15-point DCT implementation. Hence, fast algorithms for 15-point DCT are critical for the overall performance of the MDCT algorithm.

An *N*-point DCT, $X_C(k)$, of a sequence $x(n)$ is defined as follows (ignoring the normalization factors) [4]:

$$X_C(k) = \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi(2n+1)k}{2N}\right) \; ; k = 0, ..., N-1$$

Heideman [14] showed that if *N* is odd, the DCT can be mapped to an equal length real-input DFT with just input and output permutations and sign changes at the output. Thus, the computational complexity of an odd-length DCT is equal to that of an odd-length real DFT. Hence, efficient algorithms for 15-point real DFT can be used to implement a 15-point DCT.

A 15-point DFT can be efficiently implemented using the Winograd Fourier Transform Algorithm (WFTA) [15, 16]. The WFTA for 15-point DFT uses Winograd 3-point and 5-point DFT modules in a prime factor mapping. Because of the structure of the 3-point and 5-point modules, it is possible to *nest* together the multiplications in the individual modules, thus reducing the total number of multiplications. See [15-17] for details.

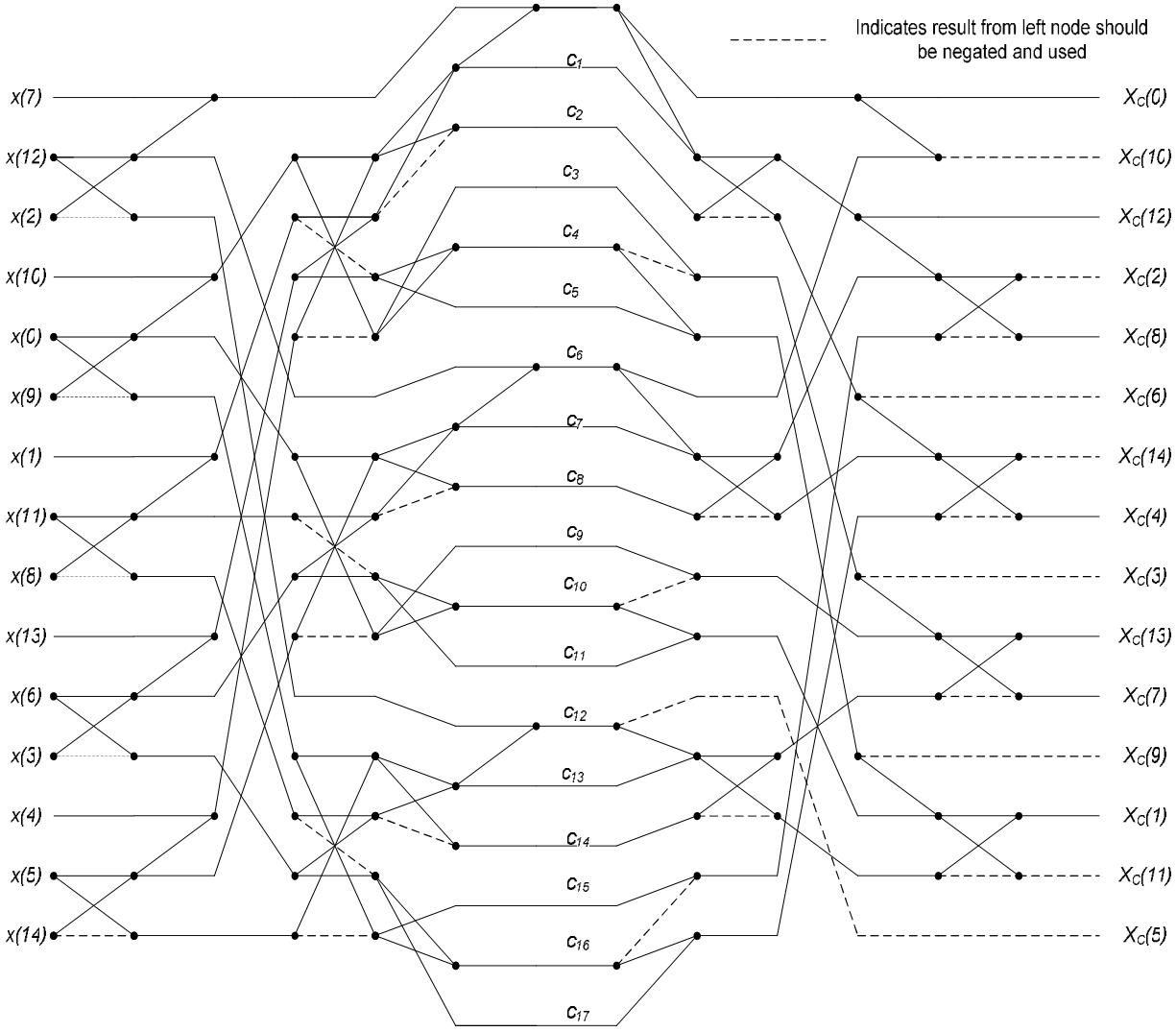

Fig 5. Flow graph for 15-point DCT-II

The 15-point real WFTA, and hence the 15-point DCT, can be implemented with 17 multiplications and 67 additions [17, 18]. The resulting FFT algorithm is the least complex among the surveyed literature [17-21]. The flow graph for the 15-point DCT is shown in Fig. 5. IDCT can be obtained by transposing this flow graph. The constants used in the figure are defined below:

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

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

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

$$c_6 = \cos v - 1; \quad c_7 = c_1 c_6; \quad c_8 = c_2 c_6$$

$$c_9 = c_3 c_6; \quad c_{10} = c_4 c_6; \quad c_{11} = c_5 c_6$$

$$c_{12} = \sin v; \quad c_{13} = c_1 c_{12}; \quad c_{14} = c_2 c_{12}$$

$$c_{15} = -c_3 c_{12}; \quad c_{16} = -c_4 c_{12}; \quad c_{17} = -c_5 c_{12}$$

7. Complexity Analysis

In this section, we discuss the computational complexity of the AAC-ELD filterbanks. We assume that the MDCT algorithm discussed in section 5 is used for these filterbanks. Since N is either 1024 or 960, we give the analysis assuming N is of the form 2^m or 15×2^m $(m \geq 3)$.

Let $RM_A(N)$ and $RA_A(N)$ denote, respectively, the number of real multiplications and additions required for the analysis filterbank and the preceding windowing operation. Let $RM_S(N)$ and $RA_S(N)$ denote the corresponding numbers for the synthesis filterbank and the succeeding windowing and overlap-add operation. *N/8* samples of the window are actually zeros and hence, multiplications and additions involving these coefficients need not be counted. Then,

$$RM_A\left(N=2^m\right)=RM_S\left(N=2^m\right)=\frac{mN}{4}+\frac{13N}{8}$$

$$RA_A\left(N=2^m\right)=RA_S\left(N=2^m\right)=\frac{3mN}{4}+\frac{5N}{8}$$

$$RM_A(N=15\times 2^m)=RM_S(N=15\times 2^m)=$$
$$2N+2^{m-1}RM_D(15)+\frac{(2m-3)N}{8}$$

$$RA_A(N=15\times 2^m)=RA_S(N=15\times 2^m)=$$
$$(59+RA_D(15)).2^{m-1}+\frac{(6m-7)N}{8}$$

where, $RM_D(15)$ and $RA_D(15)$ are the number of multiplications and additions for 15-point DCT. From section 6 we have, $RM_D(15) = 17$, $RA_D(15) = 67$.

Thus, for $N = 1024$ we have 4224 multiplications and 8320 additions; for $N = 960$ we have 3544 multiplications and 7512 additions.

8. Summary

In this paper, we presented an algorithm for mapping the MPEG-4 AAC-ELD filterbanks to MDCT/IMDCT. This mapping requires only sign changes, permutations and additions. The mapping can be used to derive a fast algorithm for the filterbanks. Further, the mapping provides a common framework for the joint implementation of filterbanks in AAC-LC, LD and ELD profiles. We also presented a very efficient 15-point DCT algorithm that is useful for block lengths of 960 and 480. We also provided a complexity analysis for the AAC-ELD filterbanks for the possible block lengths.

9. References

[1] A.M. Kondoz, “Digital Speech: Coding for Low Bit Rate Communication Systems”, 2nd Ed., Wiley, 2004.

[2] A. Spanias, “Speech Coding: A Tutorial Review”, *Proc. IEEE*, vol. 82, pp. 1541-1582, Oct. 1994.

[3] T. Painter and A. Spanias, “Perceptual Coding of Digital Audio”, *Proc. IEEE*, vol. 88, pp. 451-515, Apr. 2000.

[4] K. R. Rao and P. Yip, “Discrete Cosine Transform: Algorithms, Advantages, Applications”, New York, Academic Press, 1990.

[5] H. Malvar, “Signal Processing with Lapped Transforms”, Artech House, Boston, 1992.

[6] M. Schnell et al., “Enhanced MPEG-4 Low Delay AAC – Low Bitrate High Quality Communication”, *122nd Convention of AES*, Vienna, Austria, May 2007.

[7] ISO/IEC 14496-3:2005/FPDAM9, “Enhanced Low Delay AAC”, Apr. 2007.

[8] G.D.T. Schuller and T. Karp, “Modulated Filterbanks with Arbitrary System Delay: Efficient Implementations and the Time Varying Case”, *IEEE Transactions on Signal Processing*, vol. 48, no. 3, pp. 737-748, March 2000.

[9] ISO/IEC 14496-3: Subpart 4: “General Audio Coding (GA) - AAC, TwinVQ, BSAC”.

[10] P. Duhamel; Y. Mahieux and J.P. Petit, "A Fast Algorithm for the Implementation of Filter Banks Based on `Time Domain Aliasing Cancellation'," *in Proc. ICASSP-91,* pp. 2209-2212 vol. 3, 14-17 Apr 1991.

[11] M.-H. Cheng and Y.-H. Hsu, "Fast IMDCT and MDCT Algorithms - A Matrix Approach," *IEEE Transactions on Signal Processing*, vol. 51, no. 1, pp. 221-229, Jan. 2003.

[12] C.W. Kok, "Fast Algorithm for Computing Discrete Cosine Transform," *IEEE Transactions on Signal Processing,* vol. 45, no. 3, pp. 757-760, Mar 1997.

[13] R.K. Chivukula and Y.A. Reznik, “Efficient Implementation of a Class of MDCT/IMDCT Filterbanks for Speech and Audio Coding Applications”, *accepted for ICASSP 2008*.

[14] M.T. Heideman, “Computation of an Odd-Length DCT from a Real-Valued DFT of the Same Length”, *IEEE Transactions on Signal Processing*, vol. 40, no. 1, pp. 54-61, Jan 1992.

[15] S. Winograd, “On Computing the Discrete Fourier Transform”, *Mathematics of Computation*, vol. 32, no. 141, pp. 175-199, Jan 1978.

[16] H.F. Silverman, “An Introduction to Programming the Winograd Fourier Transform Algorithm (WFTA)”, *IEEE Trans. ASSP*, vol. 25, no. 2, pp. 152-165, April 1977.

[17] C.S. Burrus and T.W. Parks, “DFT/FFT and Convolution Algorithms – Theory and Implementation”, Wiley, New York, 1985.

[18] H.V. Sorensen et al. “Real-Valued Fast Fourier Transform Algorithms”, *IEEE Trans. ASSP*, vol. 35, no. 6, pp. 849-863, June 1987.

[19] C.S. Burrus and P.W. Eschenbacher, “An In-Place, In-Order Prime Factor FFT Algorithm”, *IEEE Trans. ASSP*, vol. 29, no. 4, pp. 806-817, Aug 1981.

[20] D.P. Kolba and T.W. Parks, “A Prime Factor FFT Algorithm Using High-Speed Convolution”, *IEEE Trans. ASSP*, vol. 25, pp. 281-294, Aug 1977.

[21] M.T. Heideman, C.S. Burrus and H.W. Johnson, “Prime Factor FFT Algorithms for Real-Valued Series”, *Proc. IEEE ICASSP*, pp. 492-495, San Diego, March 1984.

[22] M. Schnell et al., “Low Delay Filterbanks for Enhanced Low Delay Audio Coding”, *IEEE Workshop on Appl. Signal Proc. to Audio and Acoustics*, pp. 235-238, Oct. 2007.

[23] M. Dietz et al., “Spectral Band Replication, a Novel Approach in Audio Coding”, *12th AES Convention*, Munich, Germany, Apr. 2002.
[24] P. Ekstrand, “Bandwidth Extension of Audio Signals by Spectral Band Replication”, *Proc. 1st IEEE Benelux Workshop on Model based Processing and Coding of Audio*, Leuven, Belgium, Nov. 2002.