

An MDCT Hardware Accelerator for MP3 Audio

Xingdong Dai
LSI Corporation
1110 American Parkway NE
Allentown, PA 18109
xingdong.dai@lsi.com

Meghanad D. Wagh
Dept. of ECE
Lehigh University
Bethlehem, PA 18015
mdw0@lehigh.edu

Abstract—With the increasing popularity of MP3 audio, there is a need to develop cost and power efficient architectures for the MP3 encoder and decoder. This paper describes dedicated architectures for computing the modified discrete cosine transform (MDCT) and its inverse (IMDCT). Recent profiling studies have shown that these operations represent about 30% of the total MP3 computations. MP3 format defines two frame sizes that can occur in the same data stream. We have developed the most efficient algorithms for MDCT and IMDCT suitable for both sizes. Unlike previous algorithms, our computations can be unified in a single ASIC architecture. This unified architecture implemented in 90 nm TSMC library is still 25% smaller and 25% faster than any previous single frame size architectures designed in the same technology. In addition, at 128 Kbits/sec data rates, our algorithms save nearly 1800 multiplications per second (18%) which can help reduce the power consumption.

I. INTRODUCTION

The MPEG-1/2 layer-III (MP3) standard is widely employed in music industry because of its efficient audio compression. A key enabler in MP3 coding is the perfect reconstruction (PR) cosine modulated filter bank based on the concept of time domain aliasing cancellation (TDAC) [1]. For the encoder, this analysis filter bank is realized by applying the *modified discrete cosine transform* (MDCT) to sliding blocks of data, while the synthesis filter bank of the decoder uses the *inverse modified discrete cosine transform* (IMDCT). Since the MDCT and IMDCT require intensive computations, fast algorithms and efficient implementations for these transforms is important to the realization of high quality audio compression, especially when most MP3 audio players are battery operated [2]–[4].

The N point MDCT of a sequence $\{x(i)\}$ is defined as

$$X(k) = \sum_{i=0}^{N-1} x(i) \cos\left(\frac{\pi(2i+1+\frac{N}{2})(2k+1)}{2N}\right), \\ 0 \leq k < N/2.$$

The inverse MDCT (IMDCT) is defined as

$$x(i) = \frac{2}{N} \sum_{k=0}^{\frac{N}{2}-1} X(k) \cos\left(\frac{\pi(2i+1+\frac{N}{2})(2k+1)}{2N}\right), \\ 0 \leq i < N.$$

Note that MDCT converts N signal samples into only $N/2$ transform samples. MP3 standard defines two data frames of 1152 and 384 samples. These frames are further divided in 32

subbands to be processed by the MDCT/IMDCT, resulting a long block of 36 samples and a short block of 12 samples. The switch from a long block to a short block is called window switching and it is used to suppress distortions frequently associated with frequency domain coding of an audio signal. The ability to process both long and short blocks is crucial to efficient implementations of MP3 audio encoder and decoder.

Because the MP3 block lengths are not power of two, only a few fast algorithms are published and the focus has been on software implementations [5]–[7]. However, over the years the cost of application specific architectures has decreased substantially. In addition, such dedicated architectures can be embedded in field programmable gate arrays making them just as flexible as software routines. Such an approach permits one to utilize a low-cost and low-power processor to work on data management and other computationally simple tasks while the dedicated hardware can take care of the computationally challenging tasks.

This paper is devoted to the development of the MDCT/IMDCT hardware accelerator for MP3 audio applications. Based on a group theoretic partitioning of the transform kernel, our solutions employ carefully crafted bilinear algorithms which can be directly mapped to hardware architectures. We show for the first time that both the long and short MP3 audio blocks can be seamlessly processed with a single hardware architecture. Our MDCT/IMDCT algorithms require only 9 multiplications for the short block and 36 multiplications for the long block. These may be compared with the current best algorithms for the same tasks which use at least 11 and 43 multiplications respectively [5]–[7]. However, the main advantage of the proposed algorithms is its bilinearity which implies that all the multiplications are independent and can be carried out concurrently. Thus when implemented in hardware, our algorithms have only one multiplication on the critical path for both the long and short block MDCT/IMDCT. Previous reported algorithms have at least 2 multiplications along the critical path [5]–[7]. As a result, our VLSI implementations reduce the critical path delay by about 25% while simultaneously saving about 25% of the chip area.

II. ALGORITHM AND ARCHITECTURE

A bilinear algorithm is made up of an addition stage followed by a stage of independent multiplications and a final addition stage. Our procedure consists of (a) converting the

MDCT/IMDCT computation to a DCT of type IV (using additions and subtractions only), (b) decomposing the DCT kernel into cyclic convolutions and Hankel matrix products, and finally, (c) employing bilinear algorithms for each of the convolutions and Hankel matrix products to obtain the required bilinear algorithm for the MDCT. We illustrate each of these steps in this section.

A. Conversion of MDCT into DCT

To obtain MDCT using a DCT, introduce a new data sequence

$$y(i) = \begin{cases} -x(i + 3N/4), & \text{if } 0 \leq i < N/4, \\ x(i - N/4), & \text{if } N/4 \leq i < N. \end{cases}$$

Then defining

$$z(i) = y(i) - y(N - 1 - i), \quad 0 \leq i < N/2,$$

an N point MDCT can be expressed as an $N/2$ point DCT-IV as

$$X(k) = \sum_{i=0}^{N/2-1} z(i) \cos\left(\frac{\pi(2i+1)(2k+1)}{2N}\right), \quad 0 \leq k < N/2.$$

Thus the MDCT computation of short and long blocks is transformed into DCTs of 6 and 18 points respectively.

B. Conversion of IMDCT into DCT

To obtain the IMDCT, we first compute the $N/2$ point type-IV DCT of X as

$$z(i) = \frac{2}{N} \sum_{k=0}^{N/2-1} X(k) \cos\left(\frac{\pi(2i+1)(2k+1)}{2N}\right), \quad 0 \leq i < N/2.$$

Then defining a new data sequence

$$y(i) = \begin{cases} z(i), & \text{if } 0 \leq i \leq N/2 - 1, \\ -z(N - 1 - i), & \text{if } N/2 \leq i < N, \end{cases}$$

One can recover the signal sequence as

$$x(i) = \begin{cases} y(i + N/4), & \text{if } 0 \leq i < 3N/4, \\ -y(i - 3N/4), & \text{if } 3N/4 \leq i < N. \end{cases}$$

Thus the IMDCT computation of short and long blocks is also transformed into DCTs of 6 and 18 points respectively.

C. DCT algorithm for MP3 short block

Recall that a DCT-IV of an N point sequence $\{x(i)\}$ is given by

$$X(k) = \sum_{i=0}^{N-1} x(i) \cos\left(\frac{\pi(2i+1)(2k+1)}{4N}\right), \quad 0 \leq k < N. \quad (1)$$

For MP3 short block, the DCT length $N = 6$. To compute (1) for this N , we partition the signal and transform indices in two sets: set $S_1 = \{1, 4\}$ is made up of those i 's for which $(2i+1)$ is a multiple of 3 and set $S_2 = \{0, 2, 3, 5\}$, of the rest. We compute those transform components together whose indices

are in the same set. Further, the summation in (1) is carried out over the two signal index sets separately. We will denote the computation of $X(k)$ with signal component indices restricted to sets S_1 and S_2 by $X_1(k)$ and $X_2(k)$ respectively. Clearly, $X(k) = X_1(k) + X_2(k)$.

Let p denote the DCT kernel element $\cos(\pi p/4N)$ and \bar{p} , the element $-\cos(\pi p/4N)$. Then from (1), the transform components $X_1(k)$, $k \in S_1$ are given by:

$$\begin{bmatrix} X_1(1) \\ X_1(4) \end{bmatrix} = \begin{bmatrix} 9 & \bar{3} \\ \bar{3} & 9 \end{bmatrix} \begin{bmatrix} x(1) \\ x(4) \end{bmatrix}. \quad (2)$$

Since the constant matrix in (2) is a Hankel matrix, one can use a bilinear algorithm to obtain this product.

Similarly, using the same shorthand notation, transform components $X_2(k)$, $k \in S_1$ are given by:

$$\begin{bmatrix} X_2(1) \\ X_2(4) \end{bmatrix} = \begin{bmatrix} 3 & 9 \\ 9 & \bar{3} \end{bmatrix} \begin{bmatrix} x(0) - x(3) \\ -x(5) - x(2) \end{bmatrix}. \quad (3)$$

Note that (3) exploits the fact that some kernel matrix elements are equal in magnitude and therefore the signal sequence may be folded to reduce the number of multiplications. Since the constant matrix in (3) is a Hankel matrix, one can once again use a bilinear algorithm to obtain this product.

When both the signal and transform indices are restricted to set S_2 , the computation can be transformed into a multidimensional convolution. To do this, observe that indices in S_2 are elements of $A(8N)$, a group formed by non-negative integers less than and relatively prime to $8N$ under the operation of multiplication modulo $8N$. The structure of this group can be used to order the indices in S_2 as well as to define a sign function. By using the order $\{0, 5, 3, 2\}$ suggested by $A(8N)$ and the corresponding sign function $\{1, -1, 1, 1\}$, one can express $X_2(k)$, $k \in S_2$ as:

$$\begin{bmatrix} X_2(0) \\ -X_2(5) \\ X_2(3) \\ X_2(2) \end{bmatrix} = \begin{bmatrix} 1 & \bar{11} & 7 & 5 \\ \bar{11} & \bar{1} & 5 & \bar{7} \\ 7 & 5 & 1 & 11 \\ 5 & \bar{7} & 11 & \bar{1} \end{bmatrix} \begin{bmatrix} x(0) \\ -x(5) \\ x(3) \\ x(2) \end{bmatrix}. \quad (4)$$

The structure of the matrix in (4) is predicted by the structure of the group $A(8N)$. By partitioning the matrix in (4) as shown, one can see that it is a block cyclic matrix with each block being a Hankel matrix. Thus this computation corresponds to a two dimensional convolution with a 2 point cyclic convolution along one dimension and a 2 point Hankel product along the other. Again, appropriate bilinear algorithms for these small lengths can be combined to obtain the bilinear algorithm for (4).

Finally, when the signal indices are restricted to the set S_1 , one the computation of $X_2(k)$, $k \in S_2$ gives:

$$\begin{bmatrix} X_1(0) \\ X_1(5) \\ X_1(3) \\ X_1(2) \end{bmatrix} = \begin{bmatrix} 3 & 9 \\ \bar{9} & 3 \\ \bar{3} & \bar{9} \\ \bar{9} & 3 \end{bmatrix} \begin{bmatrix} x(1) \\ x(4) \end{bmatrix}. \quad (5)$$

