Tradeoffs in Modified Discrete Cosine Transform Implementations

Xin Yang, ShiChang Shi, and Alfred K. Wong
Department of Electrical & Electronic Engineering, Hong Kong University
xyang@eee.hku.hk

Abstract • The performance dependence of modified discrete cosine transform (MDCT) on hardware architecture is investigated. The oddly stacked architecture is found to be superior to direct computation in terms of accuracy, power consumption, and circuit area.

I. INTRODUTION

Improvements in integrated circuit technology have enabled electronic systems to be built on single chips. Optimal implementation of such systems on a chip (SOC) requires detailed study of the tradeoffs between system performance and hardware parameters. In this study, we consider the tradeoffs involved in the implementation of modified discrete cosine transform (MDCT).

The MDCT is employed in subband coding schemes as the analysis filter bank based on time-domain aliasing cancellation (TDAC). It is a critical block of the MP3 audio coding standard, the first international standard [1,2] for digital compression of high-fidelity audio. The high coding gain of the standard arises from its time to frequency mapping by a polyphase filterbank. The outputs from the subband filter are then passed through a MDCT to obtain higher frequency resolution. This process is illustrated in Fig.1.

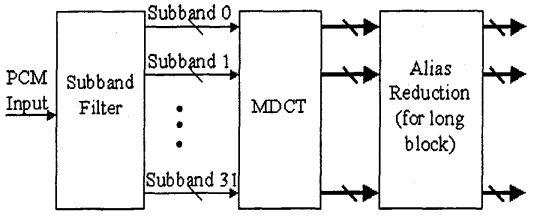

Figure 1. Schematic of the MP3 encoder filterbank.

Operation of the MDCT block in Fig.1 includes two parts: windowing and MDCT calculation. In the first part, the subband signals from the filterbank are windowed, resulting in 4 types of filter blocks: **normal, start, stop** and **short**. For example, the **normal** block is defined as follow [1]:

$$z_k = x'_k \sin(\frac{\pi}{36}(i+\frac{1}{2})) \qquad for\ k=0\ to\ 35 \qquad (1)$$

where x'_k is the subband input signal. The window length k and coefficients $C_k = \sin(\frac{\pi}{36}(i+\frac{1}{2}))$ are different for the various block types.

In the second part of the MDCT calculation, a discrete cosine transform is performed on the windowed subband data z_k:

$$X_i = \sum_{k=0}^{n-1} z_k \cos(\frac{\pi}{2n}(2k+1+\frac{n}{2})(2i+1)) \qquad (2)$$

$$for\ i=0\ to\ \frac{n}{2}-1$$

Because the windowing operation is basically a multiplying process, the first part is straightforward. We focus on the second part–MDCT calculation–in the following discussion.

The biggest impediment to efficient implementation of the MDCT is the length of the windowed data. With lengths of 36 (for normal, start, and stop blocks) and 12 (for short block), the transformation cannot be implemented with traditional fast algorithms which operate on data lengths of 2^m. Special circuit architectures are needed. The tradeoffs involved in the implementation of these MDCT architectures are discussed in the next section.

II. ANALYSYS

Architecture

0-7803-6677-8/01/$10.00©2001 IEEE.

A lot of fast discrete cosine transform (DCT) algorithms have been presented [3,4]. However, most of them apply only to data sequence lengths of $n=2^m$. They are thus not applicable to the MDCT under our current discussion.

Three architectures suitable for data lengths of 36 and 12 are discussed below. In the analysis, we illustrate the techniques by assuming a data length of 36.

a. Direct Computation

This method realizes the MDCT by directly implementing the MDCT equation:

$$X_i = \sum_{k=0}^{n-1} z_k \cos(\frac{\pi}{2n}(2k+1+\frac{n}{2})(2i+1)) \quad (3)$$

$$for\ i = 0\ to\ \frac{n}{2}-1$$

The input data sequence z_k is multiplied by the transform matrix $D=[d_{ki}]$, where

$$d_{ki} = \cos(\frac{\pi}{2n}(2k+1+\frac{n}{2})(2i+1)) \quad (4)$$

$$k = 0,1,\cdots 35 \quad i = 0,1,\cdots 18$$

It is a matrix with a dimension of 36×18. For a block length of 36 (n=36), this requires $36\times 18 = 648$ multipliers and $35\times 18 = 630$ adders. The hardware cost is very high.

b. Oddly Stacked MDCT

By combining and rotating the input data, the MDCT can be calculated by performing a discrete cosine transform (DCT) and a discrete sine transform (DST) on data sequences of reduced length [5]. Furthermore, the DST can be computed using the DCT [6] hardware. This scheme reduces the hardware requirement to 56 multipliers and 136 adders[7]. Compared with the direct computation implementation, the hardware cost is reduced greatly but the delay is larger.

c. Recursive MDCT

This implementation is based on the recursive algorithm for general length DCTs [8]. The recursive expression of MDCT is:

$$P_m(i) = x(m) + 2\cos(\theta_i)P_{m-1}(i)$$

$$-\frac{\cos[(\frac{M+1}{2})\theta_i]}{\cos[(\frac{M-1}{2})\theta_i]}x(m-1) - P_{m-2}(i) \quad (5)$$

$$i = 0,1,\cdots\frac{n}{2}-1$$

where M=n/2

x(m) is the input data sequence

θ_i is defined as $\theta_i = (i+\frac{1}{2})\frac{\pi}{2n}$

$P_m(i)$ is the intermediate result, and

$X_i = P_{35}(i)\times(-\cos[(\frac{M-1}{2})\theta_i])$ is the final result.

This implementation is suitable for parallel VLSI computing. For a data length of 36, to compute 18 points of output, one needs $2\times 36\times 18 = 1296$ multiplications and $2\times 36\times 18+18 = 1314$ additions. In hardware, it can be realized with 3 multipliers and 3 adders. The recursive nature of this architecture means that the hardware cost is low. In practice, however, the run time and error are the weaknesses. It requires 36 cycles to calculate one output and $36\times 18 = 628$ cycles to produce one MDCT. In addition, fixed-point computation in the chip induces round-off error. This error accumulates in each iteration. The resulting error is therefore much larger in this architecture compared with the previous implementations if the same number of bits is used to represent the numbers. In other words, more bits are required to give the same accuracy. Synthesis results indicate that the round-off error is unacceptable. So we will not consider this architecture in next section.

Bitwidth

In direct hardware implementation of the MDCT, not only does the architecture require examination, the number of bits used to represent the number also deserves attention. In computation of the MDCT with equation (2), the input signal and the transform coefficients are represented by a finite number of bits. The number of bits used to represent the input data (iw) is driven by the dynamic range of the input. One does not have much latitude for optimization. On the other hand, the number of bits used to represent the coefficients c_k (cw) can be optimized. The key is to find a cw long enough such that accuracy of the algorithm is adequate with minimal delay, power

consumption, and area. It turns out that the optimal cw is architecture dependent, as shown in the next section.

III. SYNTHESIS

In this section, we examine the dependence of MDCT performance on the number of bits used to represent the coefficients (cw). The performance criteria are accuracy (error), power, delay, and area. The error of each implementation is determined by

$$\frac{1}{N}\sum_{i=0}^{N-1}(X_i - \hat{X}_i)^2 \qquad (6)$$

where the X_i's are the MDCT outputs of the hardware and the $\hat{X}_i$'s represent the exact results.

Figure 2 shows the performance dependence on cw with different implementations. The results were estimated by synthesis from a 250nm standard cell library using Synopsys Module Compiler [9]. For direct computation, as cw increases, the error decreases while the area, delay, and power consumption increases. Increasing cw form 8 to 12 decrease the error from 35.899 (100%) to 0.708 (1.9%). But the area is increased from 3.07mm^2 to 7.68mm^2, the power consumption increases from 26.2 normalized unit to 63.5 normalized unit, and the delay increases from 9.34ns to 12.47ns. There is not much improvement in accuracy beyond a cw of 12. So the optimum cw should be 12.

For the oddly stacked implementation, the behavior is similar. For cw's less than 8, the error decreases sharply with increasing coefficient width. For cw's larger than 8, the error remains relatively constant, but the area, delay, and power keep rising as cw increases. A cw of 8 in the oddly stacked implementation achieves the same accuracy as a cw of 12 in direct computation. This is because when the computation is done with fixed-point arithmetic, the round-off error is induced. Increasing the number of arithmetic operations in general increases the error. The number of computations in the oddly stacked MDCT is less than that in direct computation. The error is therefore less. That means the direct computation implementation requires more bits to achieve the same accuracy. In this case, 4 extra bits are needed for the direct computation implementation.

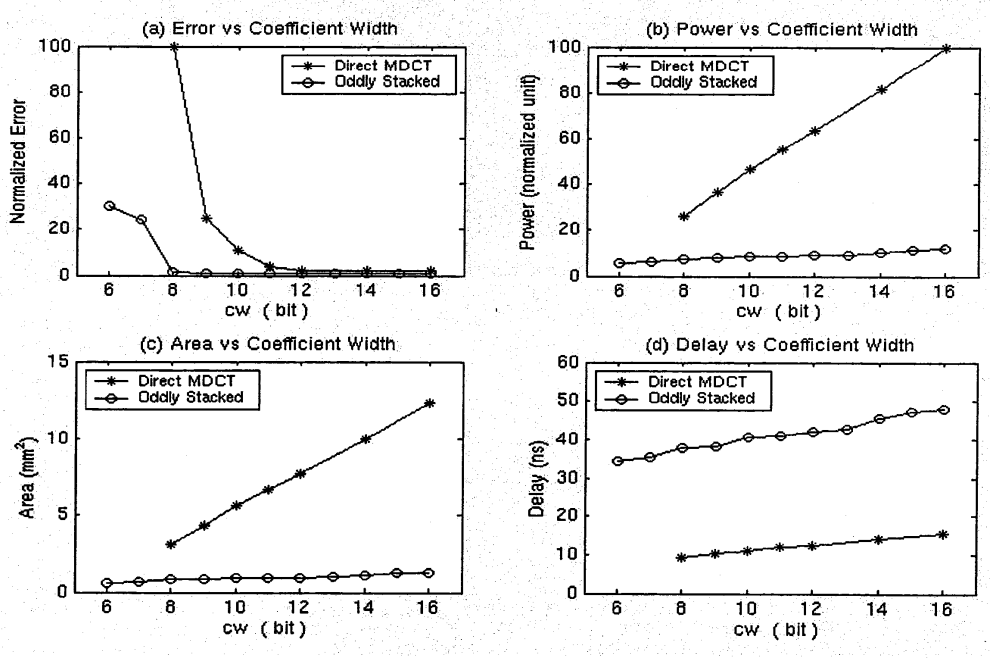

Figure 2. The performance of MDCT as a function of the coefficient bitwidth (cw). The input bitwidth is fixed at 8.

Table 1. Comparison between direct computed MDCT and oddly stacked MDCT

			Delay (ns)	Area (mm^2)	Normalized Power	Normalized Error
1	cw=12	Direct Computation	10.65	8.10	63.5	1.9
	cw=12	Oddly Stacked	44.77	1.09	8.98	0.81
2	cw=12	Direct Computation	10.65	8.10	63.5	1.9
	cw=8	Oddly Stacked	35.06	0.68	7.48	1.17
3	cw=12	Direct Computation	10.65	8.10	63.5	1.9
	cw=8	Oddly Stacked (×4)	8.765	2.72	29.92	1.17

Table 1 compares the MDCT implementations by direct computation and by the oddly stacked method. For the same cw of 12, the area of the oddly stacked MDCT is one eighth that of direct computation. Power consumption is one seventh and the accuracy of the oddly stacked MDCT also better. Nevertheless, the direct computation architecture is about 4.5 times faster.

Comparing the two implementations with approximately the same amount of error, we choose cw=12 for direct computation and cw=8 for the oddly stacked method. In this situation, the advantages of the oddly stacked architecture in area and power are further enhanced. Delay of the circuit is also shortened, although it is still 3.5 times slower than direct computation.

The slower performance of the oddly stacked architecture can be improved by parallel hardware implementation. If 4 oddly stacked circuits are used in parallel, the speed can be improved by a factor of 4, while the area and power consumption will be increased by approximately a factor of 4. This situation shown in the last two lines in Table 1. The area in the oddly stacked implementation is a third that of direct computation, and power is only a half. The oddly stacked architecture is therefore superior to direct computation.

IV. CONCLUSION

In this paper, we compared 3 different hardware implementations of MDCT. Recursive implementation suffers from inaccuracies, while direct computation is inefficient in terms of area and power. The oddly stacked architecture provides the best performance in terms of accuracy, delay, circuit area, and power consumption.

References:

1. ISO/IEC International Standard IS 11172-3 "Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbits/s-Part 3: Audio".
2. K.Brandenburg, and G.Stol, "The ISO/MPEG-Audio Codec: A Generic Standard for Coding of High Quality Digital Audio," 92nd AES-Convention, preprint 3336, Vienna 1992.
3. 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. IEEE Int. Conf. ASSP '91, Toronto, ON, Canada, May 1991, pp.2209-2212
4. Kober, V. and Cristobal, G., "Fast recursive algorithms for short-time discrete cosine transform" Electronics Letters, Vol.35 Issue:15, pp.1236-1238, 22July.1999
5. V.Britanak and K.R.Rao, "New fast algorithms for the unified forward and inverse MDCT/MDST computation," ICTR SAS, Tech. Rep., May 2000
6. Z.Wang, "A fast algorithm for the discrete sine transform implemented by the fast cosine transform," IEEE Trans. ASSP, vol. ASSP-30, pp.814-815, Oct. 1982.
7. M.T.Heidenman, "Computation of an odd-length DCT from a real-valued DFT of the same length," IEEE Trans. Signal Processing, vol.40, pp.54-61, Oct. 1992
8. L.P.Chau and W.C.Siu, "Recursive algorithm for the forward and inverse discrete cosine transform with general length", Electron. Lett., vol.30, no. 3, pp.197-198, Feb.1994.
9. Module Compiler User Guide, ©Synopsys, Inc, 2000.