Proc. IEEE Symposium on Advances in Digital Filtering and Signal Processing, Victoria, Canada, June 1998

FAST ALGORITHMS FOR ORTHOGONAL AND BIORTHOGONAL MODULATED LAPPED TRANSFORMS

Henrique Malvar

Microsoft Research
One Microsoft Way
Redmond, Washington 98052, USA

ABSTRACT

New algorithms for the computation of orthogonal and biorthogonal modulated lapped transforms (MLTs) are presented. The new structures are obtained by combining the MLT window operators with stages from a recently introduced structure for the type-IV discrete cosine transform (DCT-IV). The net result is fewer multiplications and additions than previously reported algorithms. For the orthogonal MLT, in particular, the new structure requires the computation of a slightly modified DCT-IV and some extra additions, but no further multiplications; so it demonstrates that the multiplicative complexity of the orthogonal MLT is the same as that of the DCT-IV.

1. INTRODUCTION

The modulated lapped transform (MLT) is an efficient tool for localized frequency decomposition of signals [1]. It is a particular form of a cosine-modulated filter bank [2] that allows for perfect reconstruction, has no blocking artifacts, and has almost optimal performance for transform coding of a wide variety of signals. Because of those properties, the MLT is being used in most modern audio coding systems, such as Dolby AC-3, MPEG-2 Layer III, and others [3].

Fast algorithms for the computation of the direct and inverse MLTs are based on a cascade of window operators and a type-IV discrete cosine transform (DCT-IV) [1]. The DCT-IV can be efficiently mapped to either the discrete Fourier transform (DFT) or the discrete cosine transform (DCT) [1]. Therefore, one can easily generate efficient MLT implementations by leveraging existing DFT or DCT hardware or software.

In [4],[5] fast MLT algorithms with fewer multiplications than the fast MLT in [1] were obtained by combining the MLT window with the first stage of a DCT-IV based on a half-length DFT for complex inputs. The price paid by the reduced multiplicative complexity was the need for additional data storage. In this paper we present a new MLT implementation based on a recently introduced DCT-IV structure [6]. The resulting MLT structure leads to additional savings in the number of operations, without the requirement of additional data buffers. It also leads to savings in operations for biorthogonal MLTs [7], whereas the structures in [4],[5] can only be used for orthogonal MLTs.

2. MLT AND DCT-IV

The MLT is based on the oddly-stacked time-domain aliasing cancellation (TDAC) filter bank introduced by Princen, Johnson, and Bradley [8]. Its basis functions can be obtained by cosine modulation of smooth windows, in the form [1],[7]

$$p_a(n,k) = h_a(n)\sqrt{\frac{2}{M}}\cos\left[\left(n+\frac{M+1}{2}\right)\left(k+\frac{1}{2}\right)\frac{\pi}{M}\right]$$
$$p_s(n,k) = h_s(n)\sqrt{\frac{2}{M}}\cos\left[\left(n+\frac{M+1}{2}\right)\left(k+\frac{1}{2}\right)\frac{\pi}{M}\right] \quad (1)$$

where $p_a(n,k)$ and $p_s(n,k)$ are the basis functions for the direct (analysis) and inverse (synthesis) transforms, and $h_a(n)$ and $h_s(n)$ are the analysis and synthesis windows, respectively. The time index n varies from 0 to $2M-1$ and the frequency index k varies from 0 to $M-1$, where M is the block size. The MLT is the TDAC for which the windows generate a lapped transform with maximum DC concentration, that is [1]

$$h_a(n) = h_s(n) = \sin\left[\left(n+\frac{1}{2}\right)\frac{\pi}{2M}\right] \quad (2)$$

The direct transform matrix $\mathbf{P}_a$ is the one whose entry in the n-th row and k-th column is $p_a(n,k)$. Similarly, the inverse transform matrix $\mathbf{P}_s$ is the one with entries $p_s(n,k)$. For a block $\mathbf{x}$ of $2M$ input samples of a signal $x(n)$, its corresponding vector $\mathbf{X}$ of transform coefficients is computed by $\mathbf{X} = \mathbf{P}_a^T\mathbf{x}$. For a vector $\mathbf{Y}$ of processed transform coefficients, the reconstructed $2M$-sample vector $\mathbf{y}$ is given by $\mathbf{y} = \mathbf{P}_s\mathbf{Y}$. Reconstructed $\mathbf{y}$ vectors are su-

perimposed with M-sample overlap [1], generating the reconstructed signal $y(n)$.

It is interesting to compare the MLT with the DCT-IV. For a signal $u(n)$, its length-M orthogonal DCT-IV is defined by [9]

$$U(k) \equiv \sqrt{\frac{2}{M}} \sum_{n=0}^{M-1} u(n) \cos\left[\left(n+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\frac{\pi}{M}\right] \qquad (3)$$

We see that the frequencies of the cosine functions that form the DCT-IV basis are $(k+1/2)\pi/M$, the same as those of the MLT. Therefore, we would naturally expect the existence of a simple relationship between the two transforms. That relationship is the basis of the fast MLT algorithm described in the next section.

3. FAST MLT

For a signal $x(n)$ with MLT coefficients $X(k)$ determined by the functions in (1), it is easy to show that $X(k) = U(k)$ if $u(n)$ in (3) is related to $x(n)$, for $n = 0,1,\ldots, M/2-1$, by [1]

$$u(n+M/2) = \Delta_M\{x(M-1-n)h_a(M-1-n) - x(n)h_a(n)\}$$
$$u(M/2-1-n) = x(M-1-n)h_a(n) + x(n)h_a(M-1-n)$$

where $\Delta_M\{\cdot\}$ is the M-sample (one block) delay operator.

Thus, the MLT can be computed from a DCT-IV as shown in the simplified flowgraph in Figure 1. The inverse MLT flowgraph is obtained in a similar way [1]. Note that the DCT-IV is its own inverse (since it is symmetrical and orthogonal). If $Y(k) = X(k)$, i.e., without any modification of the transform coefficients (or subband signals), then it is easy to see that cascading the direct and inverse MLT flowgraphs in Figure 1 leads to $y(n) = x(n-2M)$, where M samples of delay come from the blocking operators and another M samples come from the internal overlapping operators of the MLT (the z^{-M} operators).

The flowgraphs in Figure 1 do not assume the MLT sine window in (2). They lead to perfect reconstruction as long as the butterflies in the inverse transform are the inverses of those in the direct transform. That is true if [10]

$$h_a(n) = \frac{h_s(n)}{h_s^2(n) + h_s^2(M-1-n)}$$

Therefore, they can be used to compute the biorthogonal MLT, in which the synthesis window $h_s(n)$ has a particular form that not only improves the stopband attenuation of the synthesis filters in (1), but is also appropriate for the generation of multiresolution MLTs [7].

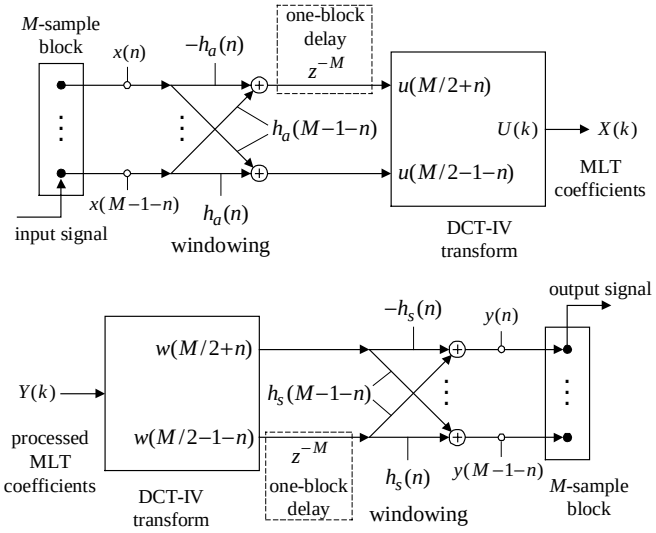

Figure 1. Flowgraphs for the fast MLT transform. Top: direct; bottom: inverse [1], n = 0, 1, …, M/2–1. These structures can also be used for the biorthogonal MLT [7].

The computational complexity of the fast MLT in Figure 1 is that of the DCT-IV plus the calculation of M/2 butterflies. For an orthogonal MLT, i.e. for $h_a(n) = h_s(n)$, each butterfly can be computed with three multiplications and three additions [1]. Thus, to evaluate a direct and an inverse MLT we need to compute two DCT-IVs plus 3M multiplications and 3M additions. Furthermore, we need M additional memory locations (to store half a block in each the direct and inverse MLTs).

An approach to reduce the computational complexity of the MLT is to combine the butterflies in Figure 1 with the first stage of computations internal to the DCT-IV. In [4], it was shown that if the DCT-IV is computed with the algorithm based on a length-M/2 complex-valued DFT [1], then M/2 multiplications can be saved in the direct MLT, at the expense of additional M/2 memory locations. In [5] it was shown that similar merging of butterflies can be applied to the inverse MLT, leading a total savings of M multiplications, at the cost of additional M memory locations[1]. In the next section we show that additional savings are possible if we start from a different DCT-IV.

[1] The inverse MLT flowgraph in Fig. 4 of [5] actually uses 2M delays, but M of them are redundant and can be removed.

4. NEW ALGORITHM - GENERAL CASE

An efficient algorithm for the DCT-IV has been presented recently [6] as part of a recursive DCT algorithm. Whereas the DCT and DCT-IV algorithms discussed in [1],[11] are all based on mappings into DFTs via orthogonal butterflies, the algorithm in [6] uses nonorthogonal stages, including diagonal matrices. Similar recursive algorithms have been developed in [9], [12], but they include flowgraph branches with gains given by $1/\cos[(r+1/2)\pi/2M]$, which can be very large (greater than 150 for M = 256, for example), leading to numerical instability.

The key result for the DCT-IV algorithm in [6] is the observation that $U(k)$ in (3) satisfies

$$\begin{aligned} U(k)+U(k-1)&=V(k),\ \text{for } k>0 \\ U(0)&=2V(0) \end{aligned} \tag{4}$$

where

$$V(k)\equiv\sqrt{\frac{2}{M}}\sum_{n=0}^{M-1}v(n)\cos\left[\left(n+\frac{1}{2}\right)\frac{k\pi}{M}\right]$$

which we recognize as the DCT (with a $\sqrt{2}$ scaling factor for the DC coefficient) of the sequence $v(n)$, given by

$$v(n)=2x(n)c(n),\qquad c(n)\equiv\cos\left[\left(n+\frac{1}{2}\right)\frac{\pi}{2M}\right] \tag{5}$$

Figure 2 shows a simplified flowgraph for the DCT-IV algorithm based on (4)–(5). Since the DCT-IV is symmetric, it can also be implemented by the transpose of the flowgraph, i.e., by flowing the signals from left to right. The diagonal matrix C has its entries equal to c(n), and the matrix B corresponds to the mapping U(k) = V(k) – U(k – 1), according to (4).

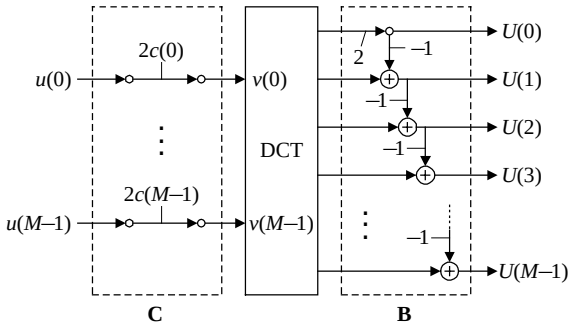

Figure 2. Flowgraph for computing the DCT-IV by means of a DCT [6].

The DCT in Figure 2 can be computed, for example, by mapping it to a real-valued FFT of length M via the FFCT algorithm [11]. Wee see from Figure 2 that computation of the DCT-IV requires M multiplications, M – 1 additions, and a DCT. Let us assume that the block size M is a power of 2. If a DCT is computed via the FFCT (which takes $(M/2)\log_2 M$ multiplications and $(3M/2)\log_2 M - M + 1$ additions), then the algorithm in Figure 2 has a complexity of $(M/2)\log_2 M + M$ multiplications and $(3M/2)\log_2 M$ additions. That matches the minimum complexity attainable for the DCT-IV [1].

Now let's use the flowgraph in Figure 2 for the DCT-IV block in the fast MLT of Figure 1. The result is shown in Figure 3. The mapping from $x(n)$ to $v(n)$ becomes the cascade of the butterflies of Figure 1 with the diagonal matrix **C** of Figure 2. Similarly, by using the transpose of the flowgraph in Figure 2 for the DCT-IV block in the fast inverse MLT of Figure 1, we get the inverse MLT flowgraph also shown in Figure 3.

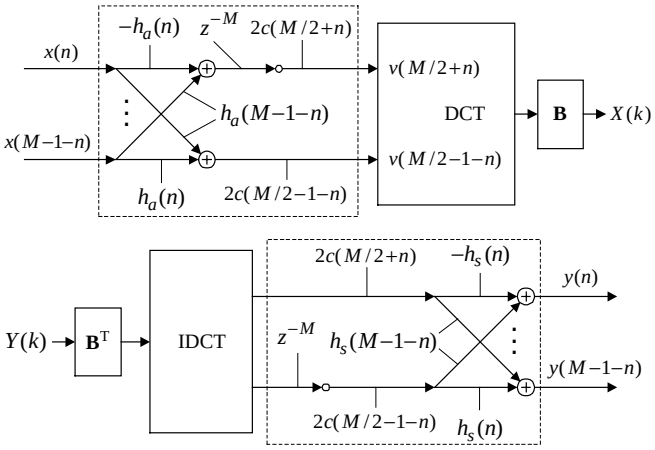

Figure 3. Fast MLT algorithm based on a length-M DCT. Top: direct; bottom: inverse. The operator **B** is the same as the one in Figure 2. The operators inside the dashed rectangles can be efficiently computed with the structures in Figure 4 (for biorthogonal MLTs) or Figure 5 (for the orthogonal MLT with the sine window in (2)).

We can reduce the computational complexity of the fast MLT in Figure 3 by scaling the butterfly coefficients, and applying the inverse scaling to the $c(n)$ gains. By doing that, we can replace the blocks inside the dashed rectangles in Figure 3 by those shown in Figure 4, where

$$\begin{gathered}\phi_a(n)=h_a(n)/h_a(M-1-n),\quad \phi_s(n)=h_s(n)/h_s(M-1-n)\\ \delta_a(n)=2c(n)h_a(M/2+n),\quad \delta_s(n)=2c(n)h_s(M/2+n)\\ \delta_a(M/2+n)=2c(M/2+n)h_a(M-1-n)\\ \delta_s(M/2+n)=2c(M/2+n)h_s(M-1-n)\end{gathered}\tag{6}$$

For most window designs, we have $1/2<h_a(n),h_s(n)<2$ for $M/2\le n<M$ [1],[7]. Therefore, the divisions in (6) do not to significantly increase the dynamic ranges of $\phi_a(n)$ and $\phi_s(n)$.

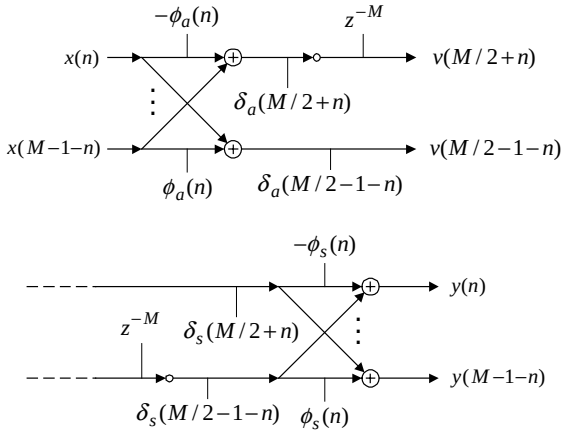

Figure 4. Reduced-complexity butterfly structures for the fast MLT structure of Figure 3. Top: structure for the direct transform; bottom: structure for the inverse transform. These butterflies define the Type-I fast MLT.

With the modified butterflies in Figure 4, we replace the computation of one butterfly (which needs three multiplications and three additions [1]) and two additional multiplications by a modified butterfly that takes four multiplications and two additions. Thus, we save one multiplication and one addition per butterfly stage. For the computation of a direct and an inverse MLT, we save a total of *M* multiplications and *M* additions.

5. NEW ALGORITHM FOR THE MLT WITH SINE WINDOW

We can obtain additional savings in computations for the case of the orthogonal MLT with the windows defined in (2). Comparing (2) with (5), we see that all butterfly gains in Figure 3 are sines or cosines of the same angles. It is easy to show that the butterflies in Figure 4 can be replaced by those in Figure 5, with

$$\begin{gathered}p(n)\equiv c(n)\big[h(n)+c(n)\big],\quad q(n)\equiv c(n)\big[h(n)-c(n)\big]\\ n=0,1,\ldots,M/2-1.\end{gathered}\tag{7}$$

The multiplications by $2^{1/2}$ in Figure 5 are just a constant scaling factor for all coefficients. If the DCT is implemented via the FFCT [11], for example, those factors can be absorbed in the output rotation butterflies. We could also work with scaled coefficients, and replace those factors at both the direct and inverse transforms by factors of two (which are just shifts) at the inverse transform only, for example. For coding applications those scaling factors can be embedded in the quantization step sizes.

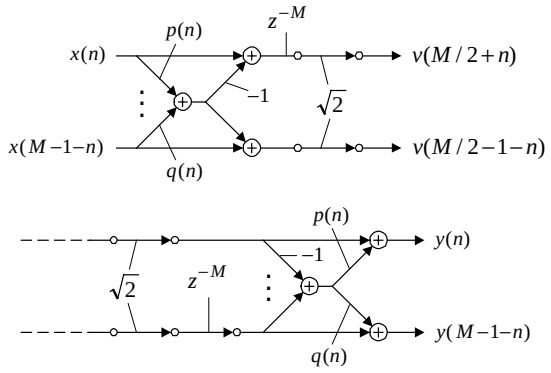

Figure 5. Reduced-complexity butterfly structures for the fast MLT structure of Figure 3. Top: direct transform; bottom: inverse transform. These butterflies define the Type-II fast MLT, which implements the windows in (2).

Hence, the Type-II fast MLT replaces the computation of one butterfly and two additional multiplications (the block inside the dashed rectangle in Figure 3) by a modified butterfly that needs only two multiplications and three additions. The computational complexity of the Type-II fast MLT, for either the direct or inverse transform, is that of the DCT plus *M* multiplications and $5M/2-1$ additions, for a total of $(M/2)\log_2 M+M$ multiplications and $(3M/2)\log_2 M+3M$ additions. That is equivalent to the computation of a DCT-IV and 3*M* extra additions.

A comparison of the computational complexity of various MLT algorithms is shown in Table 1, in terms of overhead with respect to the computation of two DCT-IVs. For the sine window of (2), which is nearly optimal for most coding applications [1], the new Type-II fast MLT saves *M* multiplications and *M* memory locations when compared to [5]. For the biorthogonal MLT, with its more flexible window choices, the new Type-I fast MLT algorithm saves *M* multiplications and *M* additions when compared to [1],[7].

Table 1. Computational complexity of MLT algorithms for direct and inverse transforms. Numbers are in addition to those required to compute two DCT-IVs.

Algorithm	Extra multiplications	Extra additions	Extra memory locations	Window choice
Ref. [1],[7]	3 M	3 M	M	any
New, Type I	2 M	2 M	M	any
Ref. [4]	2 M	3 M	1.5 M	(2) only
Ref. [5]	M	3 M	2 M	(2) only
New, Type II	0	3 M	M	(2) only

For example, Table 2 shows the total computation and memory resources needed by the various algorithms for M = 32. The new Type-II MLT saves 30% in multiplications when compared to [1], or 12% in multiplications and 25% in memory size when compared to [5].

Table 2. Computational complexity of various algorithms for direct and inverse transforms, M = 32.

Algorithm	Multiplications	Additions	Memory locations	Window choice
DCT-IV	224	480	64	–
Ref. [1],[7]	320	576	96	any
New, Type I	288	544	96	any
Ref. [4]	288	576	112	(2) only
Ref. [5]	256	576	128	(2) only
New, Type II	224	576	96	(2) only

6. CONCLUSION

We presented two new algorithms for fast MLT computation, with lower computational complexity than previous ones. Similar to the approaches in [4],[5], the computational savings are based on merging the MLT window operator with the first stage of a fast DCT-IV algorithm. Unlike the structures in [4],[5], the new algorithms do not require additional data storage.

The proposed Type-I fast MLT matches the computational complexity of [5] without being restricted to the sine window of (2). For that window, the proposed Type-II MLT reduces the total operation count by M, and shows that the multiplicative complexity of the MLT with the sine window of (2) is the same as that of the DCT-IV. It is not possible to reduce that multiplicative complexity even further, except at the cost of an excessive number of additions (otherwise, the MLT could be used to compute the DFT with a lower multiplicative complexity than the split-radix FFT, which is not possible [1],[4]).

REFERENCES

[1] H. S. Malvar, Signal Processing with Lapped Transforms. Norwood, MA: Artech House, 1992.

[2] P. P. Vaidyanathan, *Multirate Systems and Filter Banks.* Englewood Cliffs, NJ: Prentice Hall, 1993.

[3] S. Shlien, "The modulated lapped transform, its time-varying forms, and applications to audio coding," *IEEE Trans. Speech Audio Processing*, vol. 5, pp. 359–366, July 1997.

[4] P. Duhamel, Y. Mahieux, and J. P. Petit, "A fast algorithm for the implementation of filter banks based on time domain aliasing cancellation," *Proc. IEEE ICASSP*, Toronto, Canada, May 1991, pp. 2209–2212.

[5] D. Sevic and M. Popovic, "A new efficient implementation of the oddly stacked Princen-Bradley filter bank," *IEEE Signal Processing Lett.*, vol. 1, pp. 166–168, Nov. 1994.

[6] C. W. Kok, "Fast algorithm for computing discrete cosine transform," *IEEE Trans. Signal Processing*, vol. 45, pp. 757–760, Mar. 1997.

[7] H. S. Malvar, "Biorthogonal and nonuniform lapped transforms for transform coding with reduced blocking and ringing artifacts," *IEEE Trans. Signal Processing*, vol. 46, pp. 1043–1053, Apr. 1998.

[8] J. Princen, A. W. Johnson, and A. B. Bradley, "Subband/ transform coding using filter bank designs based on time domain aliasing cancellation," *Proc. IEEE ICASSP*, Dallas, TX, Apr. 1987, pp. 2161–2164.

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

[10] G. Smart and A. B. Bradley, "Filter bank design based on time domain aliasing cancellation with nonidentical windows," *Proc. IEEE ICASSP*, Adelaide, Australia, Apr. 1994, pp. III-185–III-188.

[11] M. Vetterli and H. J. Nussbaumer, "Simple FFT and DCT algorithms with reduced number of operations," *Signal Processing*, vol. 6, pp. 267–278, 1984.

[12] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 1243–1245, Dec. 1984.