Fast Algorithm for the Modulated Complex Lapped Transform

Henrique S. Malvar

January 2005

Technical Report
MSR-TR-2005-2

We present a complete derivation for a new algorithm for fast computation of the modulated complex lapped transform (MCLT), which we have previously presented. In particular, we present explicit formulas and flowgraphs not only for the direct transform, but also for the inverse transform. For a length-M MCLT, the direct transform algorithm is based on computing a length-$2M$ fast Fourier transform (FFT) plus M butterfly-like stages, without data shuffling. That reduces the number of operations and memory accesses needed to compute the MCLT when compared to previous algorithms. Compared to the original DCT-IV-based algorithm, a software implementation of the new algorithm running in a personal computer leads to 25% faster execution.

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://research.microsoft.com

1. Introduction

The modulated complex lapped transform (MCLT) is a cosine-modulated filter bank that maps overlapping blocks of a real-valued signal into complex-valued blocks of transform coefficients [1]. Thus, the MCLT performs a frequency decomposition that is similar to that obtained with the commonly-used discrete Fourier transform (DFT) filter bank [2]. The main advantage of the MCLT over DFT filter banks is that the MCLT allows for three simple reconstruction formulas, using all coefficients, only the cosine coefficients, or only the sine coefficients. That can be useful in audio enhancement and encoding systems; for example a noise reduction module can be implemented in the MCLT domain, and after enhancement only the cosine coefficients need to be sent to an encoder [1]. In fact, the MCLT cosine coefficients comprise the modulated lapped transform (MLT) coefficients for the same signal, and the MLT is quite popular in audio coding [3]. The MCLT has been used in applications where magnitude-phase decomposition is needed, as well as interfacing to coding systems, such as in audio watermarking [4], audio stream identification [5], acoustic echo cancellation [6], and microphone array beamforming [7].

In this report we present a new algorithm for computing the MCLT, for a sine window. This new algorithm leads to the simplest structure and the lowest computational complexity among MCLT algorithms reported to date. Thus, it should be quite attractive for practical applications, either in software or hardware platforms. The algorithm is the same as the one we presented in [8], but in this report we present more detailed derivations, including explicit formulas and the corresponding flowgraph for the inverse MCLT.

2. Modulated complex lapped transforms

A length-M MCLT of a length-$2M$ signal block $\{x(n)\}$ is defined by [1]

$$X(k) = \sum_{n=0}^{2M-1} x(n)\,p(n,k) \tag{1}$$

where the frequency index k varies from 0 to $M - 1$, n is the time index, and $p(n, k)$ are the analysis basis functions, defined by $p(n,k) = p_c(n,k) - j\,p_s(n,k)$, where $j \triangleq \sqrt{-1}$ and

$$\begin{aligned} p_c(n,k) &= \sqrt{\frac{2}{M}}\; h(n) \cos\left[\left(n + \frac{M+1}{2}\right)\left(k + \frac{1}{2}\right)\frac{\pi}{M}\right] \\ p_s(n,k) &= \sqrt{\frac{2}{M}}\; h(n) \sin\left[\left(n + \frac{M+1}{2}\right)\left(k + \frac{1}{2}\right)\frac{\pi}{M}\right] \end{aligned} \tag{2}$$

where $h(n)$ is the window function. Thus, we can also write $X(k) = X_c(k) - jX_s(k)$, with

$$X_c(k) = \sum_{n=0}^{2M-1} x(n)\,p_c(n,k) \qquad X_s(k) = \sum_{n=0}^{2M-1} x(n)\,p_s(n,k) \tag{3}$$

We see that $X_c(k)$, the cosine components of $X(k)$, are the MLT of $x(n)$. The window $h(n)$ can be designed in many ways, and a common choice [1], [2] is

$$h(n) = -\sin\left[\left(n + \frac{1}{2}\right)\frac{\pi}{2M}\right] \tag{4}$$

The inverse MCLT can be computed via the reconstruction formula

$$y(n) = \beta_c \sum_{k=0}^{M-1} X_c(k)\,p_c(n,k) + \beta_s \sum_{k=0}^{M-1} X_s(k)\,p_s(n,k) \tag{5}$$

where the coefficients β_c and β_s can be chosen in several ways. In [1] we propose:

i) $\beta_c = 1$ and $\beta_s = 0$ (cosine only);

ii) $\beta_c = 0$ and $\beta_s = 1$ (sine only);

iii) $\beta_c = \beta_s = ½$.

It is easy to see that the formula (5) works for any β_c and β_s such that $\beta_c + \beta_s = 1$.

The choice $\beta_c = \beta_s = ½$ is particularly useful. Besides making the MCLT an overcomplete signal decomposition whose basis functions form a tight frame with a frame gain of two [9], with that choice the formula (5) leads to $y(n) = x(n)\,h^2(n)$, i.e., there is no time-domain aliasing, and the superposition of consecutive overlapping blocks

leads to the perfect reconstruction $y = x$, because $h^2(n) + h^2(n + M) = 1$ [2]. For $\beta_c \neq \beta_s$, then $y(n)$ for a single block also contains time-aliased versions of $x(n)$. In all cases, after overlapping of the reconstructed blocks, the reconstructed signal $y(n)$ matches the original signal $x(n)$, because the overlapped squared windows add to one and time-domain aliasing terms, if present, are canceled [1], [2]. Nevertheless, in applications where a signal is processed in the MCLT domain (i.e. the MCLT of each block is computed, transformed, and an inverse MCLT is computed to generate the processed block), aliasing cancellation cannot be perfect, because the MCLT coefficients are being modified. Thus, it is better to chose $\beta_c = \beta_s = ½$ so as to avoid time-domain aliasing[1].

As suggested in [1], an efficient way to compute the MCLT is to borrow from the fast MLT algorithms in [2]. The cosine component could be computed via a MLT algorithm based on the type-IV discrete cosine transform (DCT-IV), and the sine component via a slightly modified MLT algorithm based on the type-IV discrete sine transform (DST-IV). In that way, the computational complexity of the MCLT would be exactly twice that of the MLT, that is $M(\log_2 M + 5)$ multiplications and $M(3\log_2 M + 3)$ additions, for M a power of two. The fast MCLT algorithm presented in [1] shares some of the windowing operations between the two transforms, reducing the computational complexity to $M(\log_2 M + 4)$ multiplications and $M(3\log_2 M + 2)$ additions.

3. New fast MCLT algorithm

One disadvantage of the fast MCLT algorithm in [1] is that it does not use a fast Fourier transform (FFT) directly. It maps the MCLT into DCT-IVs, which can be computed via length-$M/2$ complex FFTs, but through a process that requires data shuffling [2]. Data shuffling means additional data buffer reads and writes, which can take a considerable fraction of the total processing time, especially in general-purpose processors, where access to data memory is relatively slow (compared to access to internal CPU registers).

[1] Note that frequency-domain aliasing cannot be avoided, because the frequency selectivity of each subband filter defined by each analysis or synthesis basis functions cannot be perfect, that is, there is always some signal leaking into other subbands. For most applications, though, frequency-domain aliasing does not lead to significant distortions.

We now describe a new MCLT algorithm that uses a single length-2*M* real FFT, with no additional data shuffling. We start by assuming that *M* is even, and that the window function *h*(*n*) is the common sine window, which can be written as a sum of two complex exponentials

$$h(n) = -\sin\left[\left(n+\frac{1}{2}\right)\frac{\pi}{2M}\right] = -\frac{j}{2}\left[W_{8M}(2n+1) - W_{8M}(-2n-1)\right] \tag{6}$$

Where $W_M(r)$ is the common notation for the complex exponential used in Fourier transforms, namely

$$W_M(r) \triangleq \exp\left(\frac{-j2\pi r}{M}\right) \tag{7}$$

The same notation can be used to map (1)–(3) to

$$X(k) = \sqrt{\frac{2}{M}} \sum_{n=0}^{2M-1} x(n)\,h(n)\,W_{8M}\left[(2k+1)(2n+M+1)\right] \tag{8}$$

Combining (6) and (8), we obtain

$$\begin{aligned} X(k) = &-j\sqrt{\frac{1}{2M}} \sum_{n=0}^{2M-1} x(n)\,W_{8M}(2n+1)\,W_{8M}\left[(2k+1)(2n+M+1)\right] \\ &+j\sqrt{\frac{1}{2M}} \sum_{n=0}^{2M-1} x(n)\,W_{8M}(-2n-1)\,W_{8M}\left[(2k+1)(2n+M+1)\right] \end{aligned} \tag{9}$$

Using three basic properties of the complex exponential, $W_M(a)W_M(b) = W_M(a+b)$, $W_{2M}(2r) = W_M(r)$, and $j = W_4(-1)$, it is easy to see that

$$\begin{aligned} W_{8M}(2n+1)\,W_{8M}\left[(2k+1)(2n+M+1)\right] &= W_{2M}\left[(k+1)n\right]W_{4M}(k+1)\,W_8(2k+1) \\ W_{8M}(-2n-1)\,W_{8M}\left[(2k+1)(2n+M+1)\right] &= W_{2M}(kn)\,W_{4M}(k)\,W_8(2k+1) \\ W_8(2k+1) &= jW_8\left[2(k+1)+1\right] \end{aligned} \tag{10}$$

Then, combining (9) and (10), we get

$$X(k) = jV(k) + V(k+1) \tag{11}$$

where

$$V(k) \triangleq c(k)U(k)$$

$$c(k) \triangleq W_8(2k+1)W_{4M}(k) \tag{12}$$

$$U(k) \triangleq \sqrt{\frac{1}{2M}} \sum_{n=0}^{2M-1} x(n)W_{2M}(kn)$$

We recognize $U(k)$ as a normalized[2] $2M$-point FFT of the input block $x(n)$. Then, it is clear from (11) and (12) that the MCLT coefficients $X(k)$ can be obtained by first computing the FFT of the input vector to obtain $U(k)$ and then performing the additional operations with the factors $c(k)$, as shown in Figure 1. The coefficients $c(k)$ satisfy $|c(k)| = 1$, thus they are rotation operations. That guarantees good numerical properties of the algorithm, i.e. rounding errors due to fixed- precision arithmetic have a minimal influence on the final result.

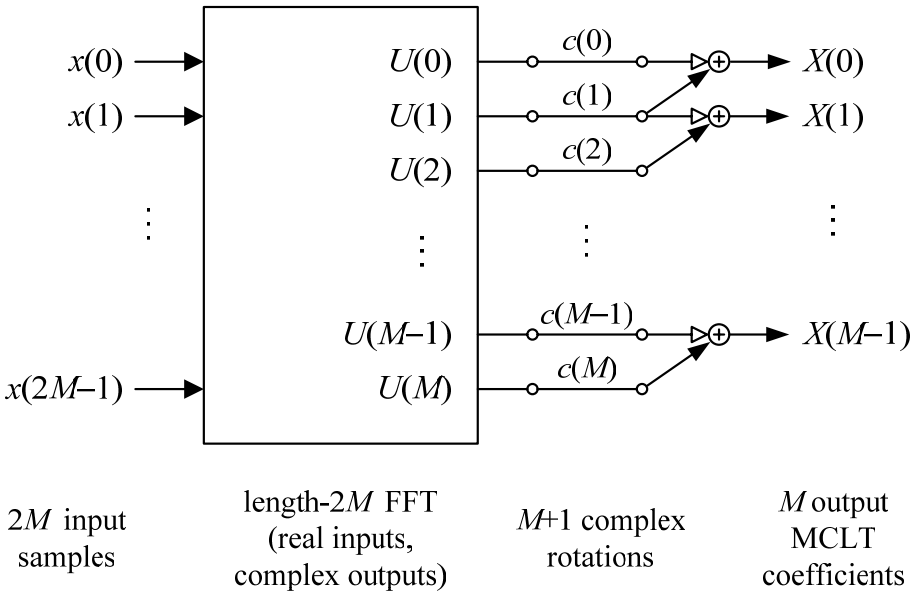

Figure 1: Flowgraph of the new fast MCLT algorithm. The input data is first transformed through an FFT for real inputs. The FFT coefficients are then combined to form the MCLT coefficients. Arrows with hollow heads mean multiplication by j.

[2] By normalized we mean with orthonormal basis functions.

4. Fast inverse MCLT

To derive the inverse MCLT relationships for a fast FFT-based algorithm, we first assume that we are interested in the inverse MCLT formula (5) with $\beta_c = \beta_s = ½$, that is

$$y(n) = \frac{1}{2}\left[\sum_{k=0}^{M-1} X_c(k)\,p_c(n,k) + \sum_{k=0}^{M-1} X_s(k)\,p_s(n,k)\right] \tag{13}$$

As we commented before, in this case there's a simple relationship between the output $y(n)$, of the inverse MCLT and the input $x(n)$ to the direct MCLT, namely

$$y(n) = x(n)h^2(n) \tag{14}$$

Let us consider the length-$2M$ FFT of $y(n)$, which is defined by

$$Y(k) \triangleq \sqrt{\frac{1}{2M}} \sum_{n=0}^{2M-1} y(n)\,W_{2M}(kn) \tag{15}$$

Substituting (14) into (15), we get

$$Y(k) = \sqrt{\frac{1}{2M}} \sum_{n=0}^{2M-1} x(n)\,h(n)\,W_{2M}(kn)\,h(n) \tag{16}$$

where it is clear that our intent is to relate $Y(k)$ to MCLT equation in (8). Replacing the rightmost term $h(n)$ by its representation via complex exponentials in (6), we obtain

$$Y(k) = \left(\frac{j}{4}\right)\sqrt{\frac{2}{M}} \sum_{n=0}^{2M-1} x(n)\,h(n)\,W_{2M}(kn)\left[W_{8M}(-2n-1) - W_{8M}(2n+1)\right] \tag{17}$$

where we have used the fact that $\sqrt{1/2M}\,(j/2) = \sqrt{2/M}\,(j/4)$.

Using again basic properties of the complex exponential, it is easy to see that

$$\begin{aligned} j\,W_{2M}(kn)\,W_8(-2n-1) &= W_{8M}\left[(2k-1)(2n+M+1)\right] W_{8M}\left[-(2k+1)\right] W_{4M}(-k) \\ j\,W_{2M}(kn)\,W_8(2n+1) &= -j\,W_{8M}\left[(2k+1)(2n+M+1)\right] W_{8M}\left[-(2k+1)\right] W_{4M}(-k) \end{aligned} \tag{18}$$

Thus, replacing (18) into (17), we obtain

$$Y(k) = \frac{c^*(k)}{4}\left[X(k-1) - jX(k)\right] \tag{19}$$

where $X(k)$ are the MCLT coefficients, the superscript * denotes complex conjugation, and the modulation $c(k)$ is the same as that in (12), namely $c(k) \triangleq W_8(2k+1)W_{4M}(k)$.

Equation (19) leads to a simple algorithm for computing the inverse MLCT: given the MCLT coefficients $X(k)$, for $k = 0, 1, \ldots, M-1$, we compute the FFT coefficients $Y(k)$ via (19) and then we use an inverse real FFT[3] to compute the output signal $y(n)$, for $n = 0, 1, \ldots, 2M-1$. Note that we need $2M$ complex values of $Y(k)$, but we have only M values of $X(k)$. However, since output signal $y(n)$ is real, we know that its FFT coefficients $Y(k)$ must satisfy the conjugate symmetry property

$$Y(2M-k) = Y^*(k) \tag{20}$$

Plus, we know that $Y(0)$ and $Y(M)$ must be real-valued. Thus, we cannot just use (19) to compute $Y(0)$ and $Y(M)$. After some manipulations based on (19) and (20), we arrive at

$$\begin{aligned} Y(0) &= \frac{1}{\sqrt{8}}\left[\mathrm{Re}\{X(0)\} + \mathrm{Im}\{X(0)\}\right] \\ Y(M) &= -\frac{1}{\sqrt{8}}\left[\mathrm{Re}\{X(M-1)\} + \mathrm{Im}\{X(M-1)\}\right] \end{aligned} \tag{21}$$

where $\mathrm{Re}\{\}$ and $\mathrm{Im}\{\}$ mean taking the real and imaginary parts, respectively.

We then have a complete four-step algorithm for the inverse MCLT:

1. Map the input MCLT coefficients $X(k)$ into $Y(k)$, using (19), for $k = 1, 2, \ldots, M-1$.
2. Use (21) to compute $Y(0)$ and $Y(M)$.
3. Use (20) to complete the vector $Y(k)$, for $k = M+1, M+2, \ldots, 2M-1$.
4. Use an inverse real FFT to compute the output vector $y(n)$ from $Y(k)$.

These steps are represented by the flowgraph in Figure 2.

[3] By "an inverse real FFT" we mean an inverse FFT that takes complex FFT coefficients that satisfy the conjugate symmetry property and compute the corresponding real-valued time-domain signal.

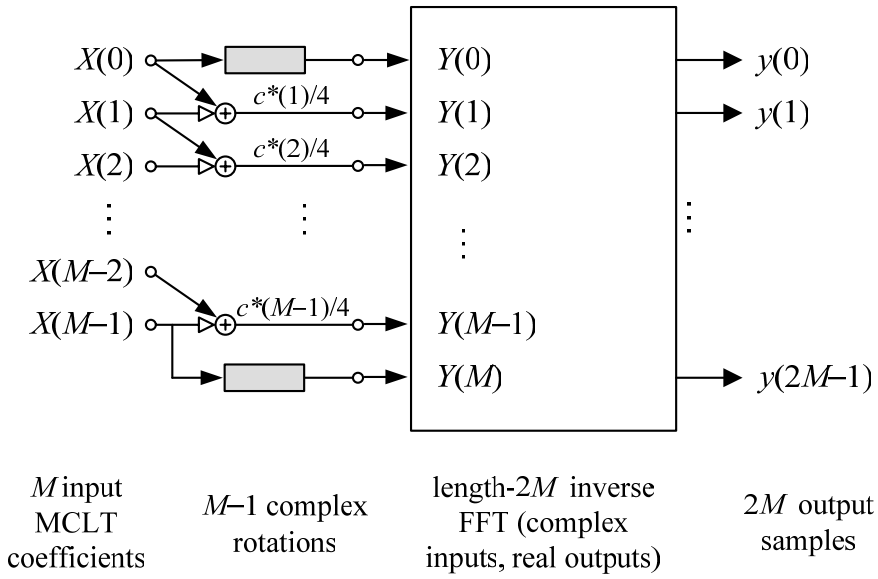

Figure 2: Flowgraph of the new fast inverse MCLT algorithm. The input MCLT coefficients are combined to form a new set of coefficients; those are then sent to an inverse real FFT to generate the output signal. Arrows with hollow heads mean multiplication by $-j$, and the shaded boxes represent the special mappings in (21).

In the Appendix we show reference implementations of the direct and inverse MCLT algorithms presented here, in the form of Matlab routines. Note that those routines were not written for fast execution; their purpose is to verify the mappings in (11)–(12) and (19)–(21). For example, they don't make use of the fast recursion for computing the modulation terms $c(k)$, as described in the next section.

5. Computational complexity

The coefficients $c(k)$ can be generated recursively, to avoid either the use of tables or the computation of trigonometric functions. From (12) we can write

$$c(0) = W_8(1) = \cos(\pi/4) - j\sin(\pi/4) \tag{22}$$

and

$$c(k) = \gamma c(k-1), \qquad \text{with} \quad \gamma \triangleq W_{4M}(M+1) \tag{23}$$

Thus, in a loop that computes the MCLT coefficients for each k in increasing order, each coefficient $c(k)$ can be generating by just multiplying the previous value $c(k-1)$ by a fixed complex-valued constant γ. Therefore, to implement the fast direct and inverse MCLT algorithms, as in Figures 1 and 2, we need only complex multiplications and additions, as well as a length-$2M$ FFT.

One of the advantages of our new fast MCLT algorithms in Figures 1 and 2 is that they do not require data shuffling, as in the MLT and MCLT algorithms in [1], [2]. That reduces execution time in software implementations and simplifies VLSI chip structures in hardware implementations.

From the derivations above, we conclude that the computational complexity of our direct MCLT algorithm is that of a length-$2M$ FFT for real inputs, plus the operations in (12). The length-$2M$ real-input FFT requires $M(\log_2 M-2)+2$ multiplications and $M(3\log_2 M-2)+4$ additions, for M equal to a power of two [2]. Besides the FFT, we need to perform $M+1$ complex multiplications and M complex additions, according to (12). Each complex multiplication can be performed with three multiplications and three additions [2]. Additionally, $U(0)$ and $U(M)$ are real, $c(0)$ and $c(M)$ have identical real and imaginary parts (except for sign), and $c(M/2)=W_4(M/2+1)$, which is either purely real or purely imaginary for $M > 2$. Then, the rotations by $c(k)$ take $3M-2$ multiplications and $5M-6$ additions. Thus, the total computational complexity of our new fast direct MCLT algorithm is $M(\log_2 M+1)$ multiplications and $M(3\log_2 M+3)-2$ additions.

In Table 1 we compare the complexity of our proposed algorithm to those of the algorithms in [1], [10], and [11]. Our new algorithm is similar to the one presented in [10], except that the latter involves complex multiplications before and after the FFT, and thus a complex-input FFT must be used, approximately doubling the computational complexity. With our proposed algorithm, not only we avoid data shuffling, but also reduce the total complexity when compared to the algorithm in [1] ($3M$ fewer multiplications and $M-2$ extra additions, for a net savings of $2M+2$ operations). For "C" implementations running on a personal computer with a Pentium 4 processor, the proposed algorithm runs about 25% faster than the one in [1], and about 50% faster than the one in [10].

Algorithm	Number of real multiplications	Number of real additions	Data shuffling?
Ref. [1]	$M(\log_2 M+4)$	$M(3\log_2 M+2)$	Yes
Ref. [10]	$M(2\log_2 M+3)+2$	$M(6\log_2 M+3)+4$	No
Ref. [11]	$M(\log_2 M+1)$	$M(3\log_2 M+4)$	No
Proposed	$M(\log_2 M+1)$	$M(3\log_2 M+3)-2$	No

Table 1: Computational complexity of several fast MCLT algorithms.

The algorithm recently presented in [11] has the advantage that it can be used for any symmetric window function $h(n)$, whereas our new algorithm applies only to the sine window in (6) (that is not a major limitation, since most applications use the sine window). Also, the algorithm in [11] maps the MCLT into two length-M discrete cosine transforms (DCTs), each one responsible for the real and the imaginary parts of the MCLT coefficients, respectively. The algorithm in [11] also avoids data reshuffling. So, in practice the choice between the algorithm presented here and the one in [11] depends on the availability of either an FFT or a DCT, for the desired target platform. For applications requiring biorthogonal transforms [12], that is, if $h(n)$ is not the sine window in (6), the algorithm in [11] is preferable.

The direct MCLT algorithm presented in Section 3 is for the computation of the MCLT of a single block. For the next block, a new FFT of length $2M$ must be computed. Even though the first half of the new block is identical to the second half of the previous block, that property cannot be used to speed up the FFT computation. Similarly, the inverse MCLT algorithm presented in Section4 is for the computation of the inverse MCLT of a single block. To reconstruct the signal, that block must be overlapped and added to the next reconstructed block, after a shift on M samples [1], [2].

6. Conclusion

We introduced a new fast algorithm for the computation of the MCLT. The proposed algorithm improves over the original algorithm in [1] by reducing the total number of operations and removing the need for data shuffling. It leads to about half the complexity

of the algorithm in [10]. By using a larger FFT as its first step, and applying a simple modification to the FFT output to generate the MCLT coefficients, the proposed algorithm also leads to a short code size whenever an efficient real FFT module is available, which is usually the case. The corresponding inverse MCLT algorithm has a similar structure, and thus it is also simple to implement in software or hardware.

References

[1] H. S. Malvar, "A modulated complex lapped transform and its applications to audio processing," *Proc. Int. Conf. Acoust., Speech, Signal Process.*, Phoenix, AZ, Mar. 1999, pp. 1421–1424.

[2] H. S. Malvar, *Signal Processing with Lapped Transforms*. Boston, MA: Artech House, 1992.

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

[4] D. Kirovski and H. Malvar, "Robust spread-spectrum audio watermarking," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing,* Salt Lake City, UT, pp. 1345–1348, Apr. 2001.

[5] C. J. C. Burges, J. C. Platt, and S. Jana, "Extracting noise-robust features from audio data," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing,* Orlando, FL, pp. 1021–1024, May 2002.

[6] J. Stokes and H. S. Malvar, "Acoustic echo cancellation with arbitrary playback sampling rate," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing,* Montreal, Canada, pp. IV-153– IV-156, May 2004.

[7] I. Tashev and H. S. Malvar, "A new beamformer design algorithm for microphone arrays," *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing,* Philadelphia, PA, Mar. 2005.

[8] H. S. Malvar, "Fast algorithm for the modulated complex lapped transform," *IEEE Signal Processing Letters,* vol. 10, pp. 8–10, Jan. 2003.

[9] V. Goyal, J. Kovacevic, and M. Vetterli, "Quantized frame expansions as source-channel codes for erasure channels," *Proc. IEEE Data Compression Conf.*, Snowbird, UT, pp. 326–335, Mar. 1999.

[10] H.-M. Tai, and C. Jing, "Design and efficient implementation of a modulated complex lapped transform processor using pipelining technique," *IEICE Trans. Fundamentals*, vol. E84-A, no. 5, pp. 1280–1286, May 2001.

[11] Q. Dai and X. Chen, "New algorithm for modulated complex lapped transform with symmetrical window function," *IEEE Signal Processing Lett.*, vol. 11, pp. 925–928, Dec. 2004.

[12] 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.

Appendix: Matlab reference implementations

We present here reference implementations for the direct and inverse MCLT algorithms in Sections 3 and 4. The routines below are not written for fast execution; their purpose is to verify the mappings in (11)–(12) and (19)–(21).

A.1. Direct transform

```
function X = fmclt(x)

% FMCLT - Compute MCLT of a vector via double-length FFT
%
% H. Malvar, September 2001  --  (c) 1998-2001 Microsoft Corp.
%
% Syntax:  X = fmclt(x)
%
% Input:   x : real-valued input vector of length 2*M
%
% Output:  X : complex-valued MCLT coefficients, M subbands

% in Matlab, by default j = sqrt(-1)

% determine # of subbands, M
L = length(x);
M = L/2;

% normalized FFT of input
U = sqrt(1/(2*M)) * fft(x);

% compute modulation function
k = [0:M]';
c = W(8,2*k+1) .* W(4*M,k);

% modulate U into V
V = c .* U(1:M+1);

% compute MCLT coefficients
X = j * V(1:M) + V(2:M+1);

return;


% Local function: complex exponential

function w = W(M,r)

w = exp(-j*2*pi*r/M);

return;
```

A.2. Inverse transform

```
function y = fimclt(X)

% FIMCLT - Compute IMCLT of a vector via double-length FFT
%
% H. Malvar, September 2001  --  (c) 1998-2001 Microsoft Corp.
%
% Syntax:  y = fimclt(X)
%
% Input:   X : complex-valued MCLT coefficients, M subbands
%
% Output:  y : real-valued output vector of length 2*M

% in Matlab, by default j = sqrt(-1)

% determine # of subbands, M
M = length(X);

% allocate vector Y
Y = zeros(2*M,1);

% compute modulation function
k = [1:M-1]';
c = W(8,2*k+1) .* W(4*M,k);

% map X into Y
Y(2:M) = (1/4) * conj(c) .* (X(1:M-1) - j * X(2:M));

% determine first and last Y values
Y(1)   =   sqrt(1/8) * (real(X(1)) + imag(X(1)));
Y(M+1) = - sqrt(1/8) * (real(X(M)) + imag(X(M)));

% complete vector Y via conjugate symmetry property for the
% FFT of a real vector (not needed if the inverse FFT
% routine is a "real FFT", which should take only as input
% only M+1 coefficients)
Y(M+2:2*M) = conj(Y(M:-1:2));

% inverse normalized FFT to compute the output vector
% output of ifft should have zero imaginary part; but
% by calling real(.) we remove the small rounding noise
% that's present in the imaginary part
y = real(ifft(sqrt(2*M) * Y));

return;


% Local function: complex exponential

function w = W(M,r)

w = exp(-j*2*pi*r/M);

return;
```