FPGA Implementation of a Modulated Complex Lapped Transform for Watermarking Systems

Jose Juan Garcia-Hernandez, Claudia Feregrino-Uribe and Rene Cumplido
National Institute for Astrophysics, Optics and Electronics
Puebla, Mexico
Email: {jjuan,cferegrino,rcumplido}@inaoep.mx

Abstract

*The Modulated Complex Lapped Transform (MCLT) is a 2x oversampled DFT filter bank. The MCLT has showed to be a good selection in audio compression and watermarking systems. It has been demonstrated that a length-*M *MCLT can be mapped to a length-*2M *Fast Fourier Transform plus* M *butterfly-like stages without data shuffling, therefore, the MCLT is appropriate for efficient hardware implementations. This paper presents an efficient implementation of the MCLT in a Field Programmable Gate Array (FPGA). Results are presented and discused.*

1 Introduction

The fast growth of the internet has increased the easy reproduction and retransmission of multimedia contents and, as a consequence, both legal and unauthorized data manipulation has also grown. One possible solution to this problem is the use of watermarking techniques, in which an imperceptible and statistically undetectable signature to multimedia content is added. A watermark must completely characterize the person who embedded it and in order to be used to prove the intellectual property of a digital media, any unauthorized removing or manipulation of the watermark must render the digital media useless.

Several algorithms for watermarking embedding and detection of watermarks have been proposed [1, 15, 7, 13, 6, 2]. The most efficient algorithms process the digital contents in the transform domain, mainly in the Discrete Cosine Transform (DCT) domain, however, the reconstructed signals using block transforms based systems exhibit the block artifact effect.

In order to beat block artifacts a family of lapped transforms was developed [9]; modulated lapped transform (MLT) is a member of that family, MLT uses *2M* samples in order to compute *M* coefficients. The MLT has been used in several audio coding standards [14]. However, MLT coefficients are only real, so, there is no phase information. In [10], the author proposed the Modulated Complex Lapped Transform (MCLT), which is an extension of MLT, but with complex components, also, fast MCLT algorithms based on discrete cosine transform and discrete sine transform were presented.

The MCLT domain has been satisfactorily used in audio watermarking [7, 8, 19], due to its no block artifact property [9]. Watermarking systems in MCLT domain have shown greater transparency than watermarking systems in other domains [5].

On the other hand, some applications of watermarking systems like broadcasting monitoring and live performance record require to guarantee their operation in real time [16, 12], moreover, multi-channel processing is very desirable.

In order to develop a real-time watermarking system it is possible to choose between two main platforms: Digital Signal Processors (DSP) and Field Programmable Gate Arrays (FPGA). The first one has been previously reported [3, 4], however, those implementations do not exploit the possible parallelism of several watermarking algorithms, therefore, FPGA implementation seems to be an interesting option. In [17] authors presented an FFT based fast algorithm and its CPLD implementation of the MCLT, however, that algorithm uses one pre-processing and one post-processing stage. Malvar showed in [11] that it is necessary only one post-processing stage after the FFT for the MCLT computing and one pre-processing stage before IFFT for the IMCLT computing.

MCLT implementation presented in this paper is one stage of a whole real-time watermarking system under development. The requirements for this MCLT implementation are: input data with format Q15, output data with format 9Q15 (along this paper we use *aQb* syntax, where *a* is number of bits used to represent the integer part and *b* is the number of bits used to represent the fractional part) and $M = 128$. This proposed architecture can be used as a coprocessor or as a module in specialized architectures

978-0-7695-3474-9/08 $25.00 © 2008 IEEE
DOI 10.1109/ReConFig.2008.35

for watermarking systems that are continuously required to perform the MCLT.

The outline of the paper is as follows: Section 2 presents the Malvar's fast algorithm via FFT. The circuit design, simulation results and hardware resources are presented in Section 3. Finally the conclusions are given in Section 4.

2 Malvar's Fast Algorithm

2.1 Fast MCLT algorithm

MCLT is a particular kind of a 2x oversampled generalized DFT filter bank whose basis are:

$$p(n,k) = p_c(n,k) - jp_s(n,k) \tag{1}$$

$$p_c(n,k) = h(n)\sqrt{\frac{2}{M}}\cos(phase) \tag{2}$$

$$p_s(n,k) = h(n)\sqrt{\frac{2}{M}}\sin(phase) \tag{3}$$

with:

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

and

$$phase = \left(n+\frac{M+1}{2}\right)\left(k+\frac{1}{2}\right)\frac{\pi}{M} \tag{5}$$

Where n is the time-domain index, k is the frequency-domain index, M is the sample block length and $j=\sqrt{-1}$. The MCLT coefficients of input vector $\mathbf{x}$ are calculated as $X(k) = X_c(k) - jX_s(k)$ with:

$$\begin{aligned} X_c(k) &= \sum_{n=0}^{2M-1} x(n)p_c(n,k), \\ X_s(k) &= \sum_{n=0}^{2M-1} x(n)p_s(n,k) \end{aligned} \tag{6}$$

If M is even, it is possible to write $h(n)$ like:

$$h(n) = -\frac{j}{2}[W_{8M}(2n+1) - W_{8M}(-2n-1)] \tag{7}$$

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

$$W_M(r) = \exp\left(\frac{-j2\pi r}{M}\right) \tag{8}$$

Combining (6) and (7), we obtain (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 $j = W_4(-1)$ and after some manipulations we get

$$X(k) = jV(k) + V(k+1) \tag{10}$$

where

$$\begin{aligned} V(k) &= c(k)U(k) \\ c(k) &= W_8(2k+1)W_{4M}(k) \\ U(k) &= \sqrt{\frac{1}{2M}}\sum_{n=0}^{2M-1} x(n)W_{2M}(kn) \end{aligned} \tag{11}$$

$U(k)$ is a $2M$ point FFT with orthonormal basis function of the input block $x(n)$, which it means that MCLT coefficients can be computed by first computing FFT of $x(n)$ to obtain $U(k)$ and then to carry out the operations with factors $c(k)$.

2.2 Fast inverse MCLT

There is a simple relation between the output $y(n)$ of the inverse MCLT and the input $x(n)$ to the direct MCLT, namely

$$y(n) = x(n)h^2(n) \tag{12}$$

Considering the length-$2M$ FFT of $y(n)$ and the relationship (12), we get

$$Y(k) = \sqrt{\frac{1}{2M}}\sum_{n=0}^{2M-1} x(n)h(n)W_{2M}(kn)h(n) \tag{13}$$

Where $Y(k)$ are the FFT coefficients of the output $y(n)$. Replacing the rightmost term $h(n)$ by its representation in (7), we obtain (14)

Using basic properties of the complex exponential, equation (15) is obtained.

Thus, replacing (15) into (14), we obtain,

$$Y(k) = \frac{c^*(k)}{4}[X(k-1) - jX(k)] \tag{16}$$

Where $X(k)$ are the MCLT coefficients, the superscript * denotes complex conjugation, and the modulation $c(k)$ is the same as that in (11). Using (16) we compute the M first FFT coeficientes of $y(n)$, but it is well known that FFT coefficients must satisfy the conjugate symmetry property

$$Y(2M-k) = Y^*(k) \tag{17}$$

Finally, we know that $Y(0)$ and $Y(M)$ must be real-valued, after some manipulations,

$$\begin{aligned} Y(0) &= \frac{1}{\sqrt{8}}[\Re\{X(0)\} + \Im\{X(0)\}] \\ Y(M) &= -\frac{1}{\sqrt{8}}[\Re\{X(M-1)\} + \Im\{X(M-1)\}] \end{aligned} \tag{18}$$

with $\Re$ and $\Im$ taking the real and imaginary parts, respectively.

$$X(k) = -j\sqrt{\frac{1}{2M}}\sum_{n=0}^{2M-1} x(n)W_{8M}(2n+1)W_{8M}[(2k+1)(2n+M+1)] + j\sqrt{\frac{1}{2M}}\sum_{n=0}^{2M-1} x(n)W_{8M}(-2n-1)W_{8M}[(2k+1)(2n+M+1)] \quad (9)$$

$$Y(k) = \left(\frac{j}{4}\right)\sqrt{\frac{2}{M}}\sum_{n=0}^{2M-1} x(n)h(n)W_{2M}(kn)[W_{8M}(-2n-1) - W_{8M}(2n+1)] \quad (14)$$

$$\begin{aligned} jW_{2M}(kn)W_8(-2n-1) &= W_{8M}[(2k-1)(2n+M+1)]W_{8M}[-(2k+1)]W_{4M}(-k) \\ jW_{2M}(kn)W_8(2n+1) &= W_{8M}[(2k+1)(2n+M+1)]W_{8M}[-(2k+1)]W_{4M}(-k) \end{aligned} \quad (15)$$

Malvar shows in [11] that equations (10), (16), (17) and (18), used with FFT processors are the fastest MCLT/IMCLT algorithms developed to date. Next section shows the implementation of the equation (10) and FFT processor, corresponding to the MCLT processor, and the implementation of the equations (16), (17) and (18) and IFFT processor, corresponding to the inverse MCLT processor.

3 FPGA Implementation

Figure 1 shows the direct MCLT processor. There are two blocks: an FFT processor and butterfly-like stage that performs equation 10. The FFT processor is implemented using a pre-designed core [18] configured in streaming mode.

The *c* factors are stored in a ROM using format Q15 in the butterfly-like stage, it also contains a register in order to store $V(k+1)$ when $X(k)$ is computed and the next clock cycle that value becomes $V(k)$. Figure 2 shows the butterfly-like structure, where *xk_re* and *xk_im* are the real and imaginary components of FFT output, *xk*, respectively, *xk_index* is the index of FFT value being processed, *c_re* and *c_im* are the real and imaginary components of factors *c* respectively, *V_re* and *V_im* are the real and imaginary components of *V* respectively and *sal_re* and *sal_im* are the real and imaginary components of *sal* MCLT coefficients respectively.

When *start* goes high it begins the loading phase, input data $xn\_re(xn\_index)$ should arrive three cycles later than the xn_index it matches [18], therefore, it is possible to use input data from an external memory or a frame buffer. The MCLT processor was developed in streaming mode, so, after an initial latency of around 615 clock cycles, it begins outputing MCLT values $X(sal\_dir) = sal\_re(sal\_dir) + jsal\_im(sal\_dir)$ and *dv* goes high. There is a M clock cycles latency due to it is necesary to load $2M$ input samples

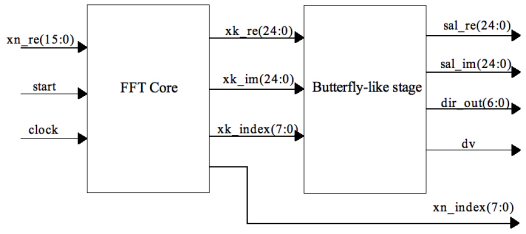

Figure 1. Direct MCLT processor

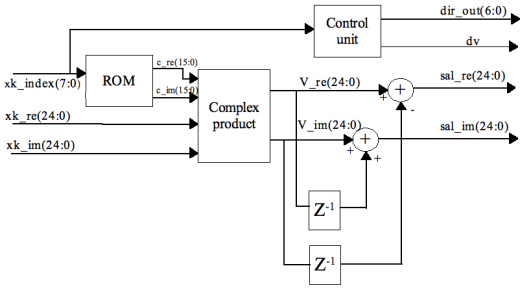

Figure 2. Butterfly-like stage for the direct MCLT processor

Table 1. FPGA's resources utilized for MCLT/IMCL implementations.

	Direct MCLT	Inverse MCLT
External IOBS	89	134
RAMB16s	7	14
Slices	2301	3545
BUFGMuxs	1	1
DSP48s	58	58
Max. Clock Frequency (MHz)	91.5	72.3
Throughput (MSPS)	91.5	72.3

in order to get M MCLT coefficients.

The $X(sal\_dir)$ values are presented in 9Q15 format, which is a constrain imposed for the whole real-time watermarking system under development. The calculations carried out in the butterfly-like stage are 40 bit wide because c factors are in Q15 format and xk samples are in 9Q15 format, therefore, a product between a Q15 number and a 9Q15 number results in a 9Q30 number, so it is necessary to truncate to the most significative twenty five bits in order to satisfy the constrain previously imposed.

For the purpose of simulation the MCLT processor was implemented in a Virtex-4 xc4vsx35-12ff668 FPGA, after *Place and Route* procedure the maximum clock rate is around 91 MHz. Due to the MCLT processor is designed in streaming mode and, after the initial latency, the MCLT processor gives a valid MCLT coefficient each clock cycle it is possible to consider a length-128 MCLT computing in $2.8\ \mu s$. The performance demonstrated by our processor suggests it can be used for applications of multi-channel, for example, in a typical block-based audio processing application, each 128 samples block is captured in $2.9\ ms$, if our MCLT processor is able to carry out a length-128 MCLT computing in $2.8\ \mu s$ then it is possible to process around 1035 channels simultaneously. In a software implementation running on a Apple iMac, G5-based workstation with and 1.9 GHz processor and 2 GB of RAM it was able to perform a length-128 MCLT computing in $625\ \mu s$. The system proposed in this paper performs around 220 times faster that this software implementation. For a multi-broadcasting monitoring application that performance is very useful. The processor presented in [17] is able to perform a length-16 MCLT in $6.06\ \mu s$, however, it is unfair to compare that implementation with our processor because the first one is implemented in a CPLD with smaller performance in comparison with the FPGA that we are using, but there are no more MCLT implementations using configurable structures reported in the literature.

The inverse MCLT processor was implemented in a similar form, c^* factors are stored in a ROM in the butterfly-like stage block in figure 3. In this block equations (16), (17) and (18) are computed. In the watermarking system

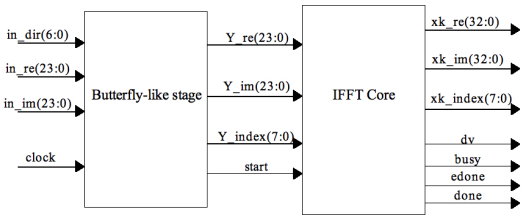

Figure 3. Inverse MCLT processor

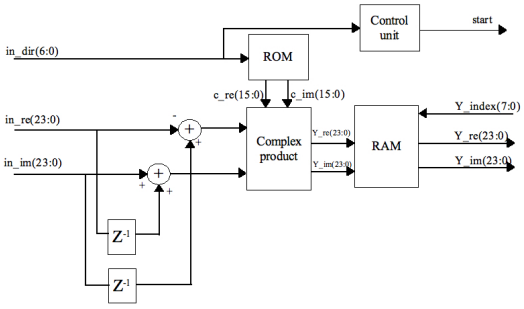

Figure 4. Butterfly-like stage for the inverse MCLT processor

under development, MCLT coefficients are watermarked in a sequential form, therefore, only two watermarked coefficients, $in\_re(in\_dir)+jin\_im(in\_dir)$ and $in\_re(in\_dir-1)+jin\_im(in\_dir-1)$ are stored in a register system similar to the direct MCLT processor, however, it is necessary to store $Y(k)$ values in a RAM in order to keep them accessible to the IFFT core. Figure 4 shows that butterfly-like structure, where *in_re* and *in_im* are the real and imaginary components of the watermarked sample *in*, respectively, *in_index* is the index of watermarked sample being processed, *c_re* and *c_im* are the real and imaginary components of factors c^* respectively, *Y_re* and *Y_im* are the real and imaginary components of *Y*. Internal control signal, generated in the *control unit* block in figure 4, begins the loading process for IFFT core in the right-hand block in figure 3 and control signals of IFFT core indicate when inverse MCLT is done. The *busy* signal will go high when IFFT is being computed, *edone* goes high one clock cycle immediately after *done* goes active, *done* will transition high for one clock cycle when the transform calculation has completed, and finally, *dv* goes high when there is a valid value $xk\_re(xk\_index)+jxk\_im(xk\_index)$. The inverse MCLT processor was also implemented in a Virtex-4 xc4vsx35-12ff668 FPGA, after *Place and Route* procedure the maximum clock rate is around 72 MHz. Due to, again, the inverse MCLT processor is designed in streaming mode it is possible to consider a length-128 inverse MCLT computing in 3.5 μs. Table 1 shows the FPGA resources utilized for, both, direct MCLT and inverse MCLT implementations, after *Place and Route* procedure. From table 1 it can be seen that the direct MCLT processor utilizes a minor number of slices and RAM16s components than the inverse MCLT processor does, it is due to the inverse MCLT processor uses a RAM stage and the direct MCLT processor does not. Moreover, the input samples for the inverse MCLT processor are 24 bit wide and, for the direct MCLT processor they are 16 bit wide, then a greater amount of slices for the inverse MCLT processor is necessary. The throughput is affected for the same wide input conditions, in the direct MCLT processor it is 91.5 mega samples per second (MSPS) and for the inverse MCLT processor it is 72.3 MSPS.

4 Conclusions

This paper presents efficient MCLT and inverse MCLT processors, based in the fastest MCLT algorithm available currently. Implementations in state-of-the-art FPGAs are presented and discussed too. The real-time watermarking system constrains are covered satisfactorily. The computing time for each processor suggest that in a watermarking-based multi-broadcasting application our implementations will be very adequate. Although our implementations are part of a watermarking system these can be used in other different digital signal processing tasks such as noise cancellation and acoustic echo cancellation with same precision requirements. The MCLT/IMCT implementations presented in this paper have shown to be the fastest implementations reported currently.

Acknowledgment

The authors would like to thank CONACyT for financial support.

References

[1] P. Bassia and I. Pitas. Robust audio watermarking in the time-domain. *IEEE Transactions on Multimedia*, 3(2):232–241, June 2001.

[2] W. Bender, D. Gruhl, and N. Morimoto. Techniques for data hidding. *IBM Journal*, 2003.

[3] J. Garcia-Hernandez, M. Nakano, and H. Perez. Real time implementation of low complexity audio watermarking algorithm. In *Proc. Third International Workshop on Random Fields and Processing in Inhomogeneous Media*, October 2005.

[4] J. Garcia-Hernandez, M. Nakano, and H. Perez. Real-time mclt audio watermarking and comparison of several whitening methods in receptor side. In *Proceedings of Eigth IEEE ISM 2006, San Diego, CA, USA*, pages 991–997, 2006.

[5] J. J. Garcia-Hernandez, M. Nakano-Miyatake, and H. Perez-Meana. Data hiding in audio signal using rational dither modulation. *IEICE Electron. Express*, 5(7):217–222, 2008.

[6] J. Haitsma, M. van der Veen, T. Kalker, and F. Bruekers. Audio watermarking for monitoring and copy protection. In *ACM Multimedia Workshop Marina Del Ray, USA*, pages 119–122, 2000.

[7] D. Kirovski and H. Malvar. Robust covert communication over a public audio channel using spread spectrum. In *4th International Information Hiding Workshop*, April 2001.

[8] D. Kirovski and H. Malvar. Spread spectrum watermarking of audio signals. *IEEE Transactions on Signal Processing*, 51(4):1020–1033, April 2003.

[9] H. S. Malvar. *Signal Processing with Lapped Transforms*. Artech House, Inc., 1992.

[10] H. S. Malvar. A modulated complex lapped transform and its applications to audio processing. Technical report, Microsoft Research, 1999.

[11] H. S. Malvar. Fast algorithm for the modulated complex lapped transform. Technical report, Microsoft Research, 2005.

[12] T. Mizrahi. Real-time implementation for digital watermarking in audio signals using perceptual masking. Technical report, Signal and Image Processing Lab., Dept. of EE, Technion, 2002.

[13] C. Neubauer, J. Herre, and K. Brandenburg. Continuous steganographic data transmission using uncompressed audio. In *Information Hiding Second International Workshop*, April 1998.

[14] S. Shlien. The modulated lapped transform, its time-varying forms, and its applications to audio coding standars. *IEEE Transactions on Speech and Audio Processing*, 5:359–366, 1997.

[15] M. D. Swanson, B. Zhu, and A. H. Tewfik. Robust audio watermarking using perceptual masking. *Signal Processing*, 66:337–355, 1998.

[16] R. Tachibana. Sonic watermarking. *EURASIP Journal on Applied Signal Processing*, pages 1956–1954, 2004.

[17] H.-M. Tai and C. Jing. Design and efficient implementation of a modulated complex lapped transform processor using pipelining technique. *IEICE Trans Fundamentals*, E84-A(5):1280–1286, May 2001.

[18] Xilinx. Inc. *Fast Fourier Transform v4.1*, April 2007.

[19] R. Zezula and J. Misurec. Audio signal watermarking in mclt domain with the aid of 2d pattern. In *Proceedings of 2nd International Conference on Digital Telecommunications, ICDT '07*, 2007.