VLSI Implementation of Mallat's Fast Discrete Wavelet Transform Algorithm with Reduced Complexity

Yuanbin Guo, Hongzhong Zhang, Xuguang Wang, Joseph R. Cavallaro

ECE - Rice University, 6100 Main Street. Houston, TX, 77005-1892.

***Abstract* – This paper proposes a novel VLSI architecture to compute the DWT (discrete wavelet transform) coefficients using Mallat's algorithm with reduced complexity. We studied the commonality embedded in the mirror filters of the algorithm and use a PLA as an Address Generator (PAG) to load the data for cascaded FIR computation. By using an embedded down-sampling process in the control signal design, we reduced the complexity by saving storage and computation. The prototyping design is implemented and fabricated using AMI 1.5 micron CMOS process through MOSIS.**

I. INTRODUCTION

Discrete Wavelet Transform (DWT) is a very useful tool in time-frequency analysis because of its excellent localization both in time and frequency [1]. It has been very successful in several areas such as image compression, communication and denoising. DWT is a good alternate to FFT (Fast Fourier Transform) in most applications. For the same reason as DCT and FFT ASICs, there is good demand to investigate efficient implementation architecture of VLSI design for DWT and push it into the real-world high-speed IC industry.

Mallat's pyramid algorithm is considered the most important algorithm to calculate DWT coefficients and it plays the similarly important role in wavelet transform as FFT has been in Fourier Transform. Mallat's algorithm is basically a collection of cascaded FIR filtering operations by a pair of mirror filters and down-sampling procedure in each scale. The problem is stated briefly as follows. Given a signal $\{x(n)\}$, we are expected to generate a set of wavelet coefficients $\{d_j^k\}$ and scaling coefficients $\{c_{J_0}^k\}$ from a pair of mirror FIR filters $\{h_0(n)\}$, $\{h_1(n)\}$. The computation of one scale includes two sets of coefficients: wavelet coefficients, $\{d_j^k\}$, that are the detailed representation of the signal and scaling coefficients, $\{c_{J_0}^k\}$, that are the coarse representation of signal.

In this paper, we first described Mallat's DWT algorithm. We made a complexity analysis and studied commonality in the algorithm. We found some embedded redundancy in the mirror FIR filtering and down-sampling process. Based on some important observations, we derived a VLSI architecture with reduced computational and storage complexity compared to the original algorithm. A Programmable Logic Array (PLA) Address Generator (PAG) is applied to generate index and control signals to load the data to a MAC (Multiplier Accumulator) unit. The prototype design is implemented with AMI 1.5 micron CMOS process.

II. DWT & MALLAT'S PYRAMID ALGORITHM

According to wavelet transform, a set of time series *f(t)* can be approximated by the smooth version projection onto multiple scaling subspace and the detailed version projection to the wavelet subspaces, as in

$$f(t)=\sum_k c_{J_0(k)}2^{J_0/2}\phi(2^{J_0}t-k)+\sum_k\sum_{j=1}^{J_0}d_j(k)2^{j/2}\psi(2^j t-k) \quad (1)$$

$\phi(t)$'s are scaling functions in multiple scales and $\psi(t)$'s are wavelet functions. They form an orthogonal basis of the signal space. In (1), *j* is the index of scale with a range from $\{0, J_0\}$, and *k* is the time shift factor. Given the orthogonal basis, signal *f(t)* can be represented by a set of scaling coefficients $\{c_{J_0}(k)\}$ and wavelet coefficients $\{d_j(k)\}_1^{J_0}$, where J_0 is a desired scale.

Mallat's algorithm states that the coefficients $\{c_{J_0}(k)\}$ and $\{d_j(k)\}_1^{J_0}$ can be calculated by cascade FIR with a pair of filters: low-pass filter $H_0(\omega)$ derived from the scaling function and high-pass filter $H_1(\omega)$ derived from wavelet functions [1]. The filters have the property to make them mirror filters. In Fig.1., $H_0(\omega)$ and $H_1(\omega)$ are the frequency response of predefined filter coefficients $\{h_0(n)\}_1^L$ and $\{h_1(n)\}_1^L$, respectively. $\{h_0(n)\}_1^L$ and $\{h_1(n)\}_1^L$ are computed from specific wavelet construction methods and they have a symmetric property which makes,

$$h_1(n)=(-1)^{n+1}h_0(L+1-n). \quad (2)$$

Here *L* is the length of the filters. In Fig.1, Daub-4 wavelet is used as an example to demonstrate the mirror filters.

The computation constitutes of two steps recursively: initially the original signal *f(n)* with length N ($N=2^J$) is considered as the scale 0. They are both low-pass filtered by $H_0(\omega)$ and high-pass filtered by $H_1(\omega)$. The FIR filtering is computed through circular convolution by making the data periodic.

0-7803-7206-9/01/$17.00 © 2001 IEEE

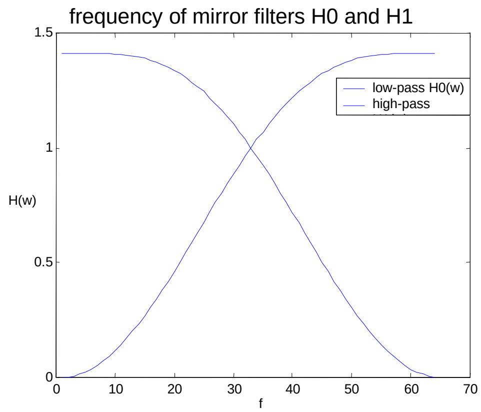

Fig.1 Mirror filter of H_0 and H_1 for Daub-4 wavelet

This first step generated a set of coefficients $\{c_1^{'}(k)\}_{k=1}^{N+L-1}$ and $\{d'_1(k)\}_{k=1}^{N+L-1}$ with length N+L-1. In the second step, a down-sampler decimates the coefficients to get the scaling coefficients $\{c_1(k)\}_{k=1}^{N/2}$ and wavelet coefficients $\{d_1(k)\}_{k=1}^{N/2}$ in the first scale.

$$\begin{cases} c_1(k) = c'_1(2*k-1) \\ d_1(k) = d'_1(2*k-1) \end{cases} \tag{3}$$

In the next scale computation, $\{d_1(k)\}_{k=1}^{N/2}$ is kept as the final result but $\{c_1(k)\}_{k=1}^{N/2}$ is used as the data and decomposed recursively as the above procedure until it reaches the desired scale J_0. This will form a pyramid structure of cascaded FIR and decimating computation as in Fig.2. The final coefficients are $\{c_{J_0}(k)\}$ in the last scale J_0 and $\{d_j(k)\}_1^{J_0}$ in the previous scales. The procedure is depicted in Fig.2. and is also called pyramid algorithm because of the structure.

III. COMPLEXITY ANALYSIS AND OPTIMATION

For a space and complexity constraint VLSI ASIC design, the key challenge is the complexity in both storage and computation. Generally, we need two FIR computations for each scale with the same length filters $\{h_0(n)\}_1^L$ and $\{h_1(n)\}_1^L$ for half the length of the previous scale. Typically a DWT is block-based computation, i.e. a block of data is first collected before the computation begins. Then the window shifts to another block after the computation of coefficients. So we analyze the computation in one block.

The FIR computation for one scale can be captured by equations:

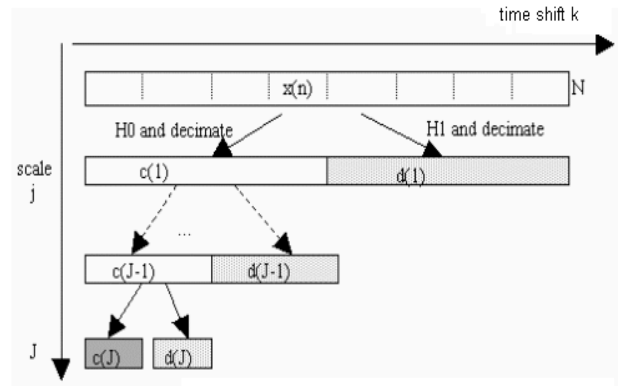

Fig. 2 Mallat's Pyramid algorithm for DWT computation

$$\begin{cases} \overline{\mathbf{C}}_j = \overline{\mathbf{H}}_0 * \overline{\mathbf{C}}_{j-1} \\ \overline{\mathbf{D}}_j = \overline{\mathbf{H}}_1 * \overline{\mathbf{C}}_{j-1} \end{cases} \tag{4}$$

where

$$\begin{cases} \overline{\mathbf{C}}_j = [c_j(1)\ c_j(2)\ ...c_j(N/2^j)]^T \\ \overline{\mathbf{D}}_j = [d_j(1)\ d_j(2)\ ...d_j(N/2^j)]^T \end{cases} \tag{5}$$

and

$$\begin{cases} \overline{\mathbf{H}}_0 = [\overline{h}_0(1)\ \overline{h}_0(2)...\ \overline{h}_0(N/2^j)]^T \\ \overline{\mathbf{H}}_1 = [\overline{h}_1(1)\ \overline{h}_1(2)...\ \overline{h}_1(N/2^j)]^T \end{cases}, \tag{6}$$

where $\overline{h}_m(k)$ are the row vectors constitutes the filter coefficients and padding zeros for both wavelet filters and scaling filters(m=0,1). They have the form of $[0\ \ \ldots\ \ 0\ \ h_4\ \ h_3\ \ h_2\ \ h_1\ \ 0\ \ \cdots\ \ 0]$. For an easier explanation, we use an example with L=4, N=8 to denote the matrix in the equation, as in Fig. 3.

In this example, we have $\{x(n) \mid n \in (1,8)\}$. The circular convolution will generate ***N+L-1*** coefficients $\{c_6\ c_7\ c_8\ c_1\ c_2 ... c_8\}$. It is obviously naïve to compute the direct multiplication of these two big matrixes since the matrix ***H*** is highly redundant and has size (***N***+***L***-1)*(***N***+***L***-1) with only L coefficients.

$$\begin{bmatrix} c_6 \\ c_7 \\ c_8 \\ c_1 \\ c_2 \\ \vdots \\ c_8 \end{bmatrix} = \begin{bmatrix} h_4 & h_3 & h_2 & h_1 & 0 & 0 & \cdots & & \cdots & 0 \\ 0 & h_4 & h_3 & h_2 & h_1 & 0 & & 0 & & \vdots \\ 0 & 0 & h_4 & h_3 & h_2 & h_1 & \ddots & & 0 & \\ \vdots & & \ddots & h_4 & h_3 & h_2 & h_1 & \ddots & & \vdots \\ & 0 & & \ddots & h_4 & h_3 & h_2 & h_1 & 0 & 0 \\ 0 & & 0 & & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & & & \cdots & 0 & h_4 & h_3 & h_2 & h_1 \end{bmatrix} * \begin{bmatrix} x_6 \\ x_7 \\ x_8 \\ x_1 \\ x_2 \\ \vdots \\ x_8 \end{bmatrix}$$

Fig. 3. Matrix Multiplication of Circular Convolution

We noticed that the computation for different coefficients is just shifting the filter through a window of the data. If we follow the ***FIR + decimation*** architecture and use the standard FIR computation with shifted operation of the index, we then will have to first compute and store ***N+L-1*** coefficients and then decimate. Moreover, this computation will need storage for two filters $\{h_0(n)\},\{h_1(n)\}$.

However we made several observations here: 1) The filters $\{h_0(n)\},\{h_1(n)\}$ are mirror filters and only differ in the index and sign; 2) a down-sampling procedure after the FIR introduces redundancy in the computation and we waste half the resources to compute those coefficients that we do not want; 3). By applying the standard FIR operation, we will need an extra decimating procedure to pick the desired coefficients; the storage for the intermediate coefficients $\{c'_1(k)\}_{k=1}^{N+L-1}$ and $\{d'_1(k)\}_{k=1}^{N+L-1}$ is redundant.

For a detailed study, we expand the expression of only those scaling/wavelet coefficients of interest,

$$\begin{cases}(c/d)_1(1)=h_{0/1}(4)*x(1)+h_{0/1}(3)*x(2)+h_{0/1}(2)*x(3)+h_{0/1}(1)*x(4)\\(c/d)_1(2)=h_{0/1}(4)*x(3)+h_{0/1}(3)*x(4)+h_{0/1}(2)*x(5)+h_{0/1}(1)*x(6)\\(c/d)_1(3)=h_{0/1}(4)*x(5)+h_{0/1}(3)*x(6)+h_{0/1}(2)*x(7)+h_{0/1}(1)*x(8)\\(c/d)_1(4)=h_{0/1}(4)*x(7)+h_{0/1}(3)*x(8)+h_{0/1}(2)*x(1)+h_{0/1}(1)*x(2)\end{cases}\quad(7)$$

and the relation of the filters are,

$$\begin{cases}h_1(1)=h_0(4)\\h_1(2)=-h_0(3)\\h_1(3)=h_0(2)\\h_1(4)=-h_0(1)\end{cases}\quad(8)$$

Notice that: 1). Each coefficient now only has 4 multiplications and 3 additions; 2). Wavelet filter $\{h_1(n)\}_1^L$ is only different from scaling filter $\{h_0(n)\}_1^L$ in the index and the sign; 3). The computation of $\{d_j(k)\}$ and $\{c_j(k)\}$ are also only different in the indices of the two sets of data: *{x(n)}* and $\{h_0(n)\}_1^L$. 4). The computation of redundant coefficients {*c2', c4', c6', c8*} has no effect on the desired coefficients {*c1', c3', c5', c7'*} which are actually the final coefficients {*c(1),c(2),c(3),c(4)*}.

Recall that in VLSI design the adder can have one input bit control signal "sub" to switch between addition and subtraction. So if we know the index and sign of each coefficient at each computation cycle, we will be able to get the correct results.

The key idea is to embed the decimation into the FIR computation. Rather than using a shift-register to generate the ordered index of the FIR coefficients as in a standard FIR computation, we design a PLA to generate both the address and sign for *x(n)* and *h(n)* at a particular step. Because the number of MAC is fixed for each coefficient as ***L***, it is very convenient for the PLA to implement a finite state machine (FSM) with very simple states. We only require four states for all coefficients. And since the index is generated by the PLA, it can be random and flexible rather than the exact pattern of shifting-order in a FIR computation.

We achieve a significant improvement in the storage and computation complexity with this design. TABLE.I summarized the improvement. Here ***N*** is the length of data block and ***L*** is the length of the filter coefficients. For simplicity, we also use the number of the example,i.e. ***N***=8, ***L***=4.

TABLE I. Reduced Complexity after Optimization

	Mallat's	Optimized	VLSI blocks
Storage	2(N+L-1)+N+2L=38	N+2L=16	Memory /Latch
*/+	2(Nh+L-1)*L=88	N*L=32	Accumulator/Multiplier
Overhead	Filter Mirror Down-sampler	--	Logic, PLA

IV. VLSI ARCHITECTURE

The top level architecture of our design is depicted in Fig.4. The core of this design consists of the following parts: a 8*8 Booth Recoding multiplier; a 12b*12b accumulator; internal memory unit to store the data x(n) and filter coefficients $\{h_0(n)\}_1^L$; PLA block to generate the address to load {*x(n)*} and $\{h_0(n)\}_1^L$ and relevant control signals such as "sub" signal for the full adder, and several "load" control signals for the latches separating the different modules. A MUX is used to multiplex the input data and update of the scaling coefficients $\{c_j(k)\}$ to the memory; an I/O interface with the external MPU(Micro Processor Unit). The main PLA is controlled by three counters corresponding to three level of periodicity: the index of coefficients k, the wavelet/scaling coefficient switch(CDSwitch) and the scale index j.

During computation of the DWT coefficients, an external MPU first inputs the data {*x(n)*} and scaling filter $\{h_0(n)\}_1^L$. Then the DWT chip switches to computation mode by "X/CSW" MUX control signal. The PLA will generate the relevant address to the decoder of memory unit according to the current scale, type of coefficient and index of the currently computed coefficient. The main PLA uses a predefined lookup table to generate the address and sign for both {*x(n)*} and $\{h_0(n)\}_1^L$. When it is in the mode of computation of scaling coefficients, $\{h_0(n)\}_1^L$ decrease in order and no

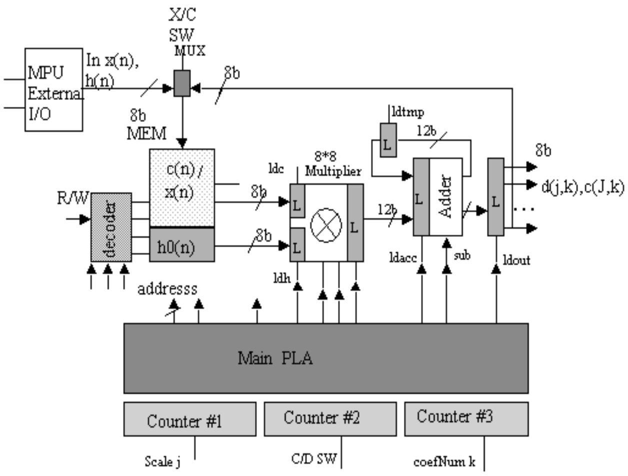

Fig. 4. VLSI architecture of Embedded Decimation with PLA Address Generator (PAG) in Mallat's Algorithm

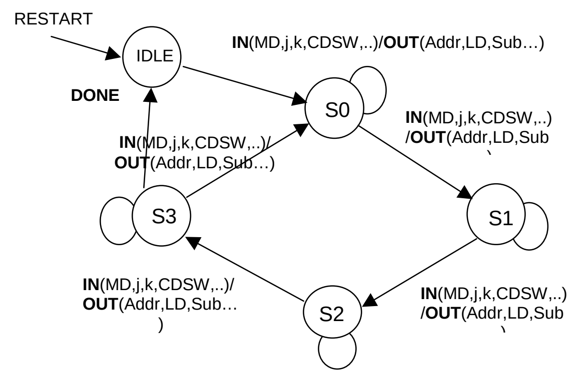

Fig. 5. Basic FSM for the main PLA

"sub" signal is produced. When it is in the mode of wavelet coefficients, x(n) is the same as in scaling mode, but the index and sign of $\{h_1(n)\}_1^L$ are generated according to the relation in (8).

When all the coefficients are computed in one block of $\{x(n)\}$, the MPU will feed in another block and this procedure continues recursively.

V. PLA DESIGN

In the proposed architecture, only one MAC is required (However we can use multiple MACs for a pipelined computation. This is a possible enhancement). The design of the PLA is essential to the algorithm. Since there are several coefficients to compute, the state transitions of the PLA might be very complicated if we do not extract the commonality among the computation of coefficients.

Notice in the computation of four scaling/wavelet coefficients in (4), we have 3 additions and 4 multiplications for each coefficient. We notice further that in general, when the filter length is $\boldsymbol{L}$, each coefficient in the complete Mallat's pyramid tree has $\boldsymbol{L}$-1 additions and $\boldsymbol{L}$ multiplications. This is very important since we can design the PLA with $\boldsymbol{L}$ as constant number of the basic state and use counter to generate the switch control signals of different coefficients.

A simplified FSM is depicted in Fig. 5. In the STARTDWT state, original signals and the filter $\{h_0(n)\}_1^L$ are inputted to the memory. After the data has become valid, the whole system is set to idle state (state0). All latches and other registers should be initialized after the RESTART is asserted. Then a STARTDWT (SD) signal is sent to the chip. From this point, the DWT chip is controlled by the internal main PLA and the calculation cycle begins.

In one cycle, the FSM includes four basic states {*s0*, *s1*, *s2*, *s3*}, each for one multiplication. The transition of the state is caused by the completion of one multiplication operation. After each multiplication, a "LDACC" signal for the accumulator is generated. After four states, one coefficient is computed. Then it updates the number of coefficient and enters another cycle to compute the second coefficient. After the completion of scaling coefficient, C/DSW is updated to generate address and sign of the wavelet filter and the corresponding $\{x(n)\}$. Then after one scale, it continues to the next scale. After the last coefficient of the last scale is computed, the state returns to idle and waits for the next block data. In the simplified FSM depicted in Fig.5, the bubbles are the states. The lines indicate the transition of the state based on control signal values. The output of PLA is also depicted on the lines with the form *"input/output"* above the lines.

Basically, there are three different periods. The first one is for addition, which equals to 4 multiplication cycles; the second one is for switching of different coefficients, i.e. d_i or c_i, which equals to the sum of 4 cycles of additions; the third one is for the number of coefficients computed in the current scale. So, three counters are designed for these three different periods. They send out corresponding control signals to the main PLA (Fig. 6). This first counter generates **MD**, **AD** and **XH** switch signal for main PLA. **MD** is asserted every four clock cycles to activate the state transition to another multiplication state. Meanwhile, the loading controls for the accumulator starts. After every four **MD**, an **AD** is asserted, which means a coefficient has been computed. **XH** is used to switch between loading data and wavelet coefficient.

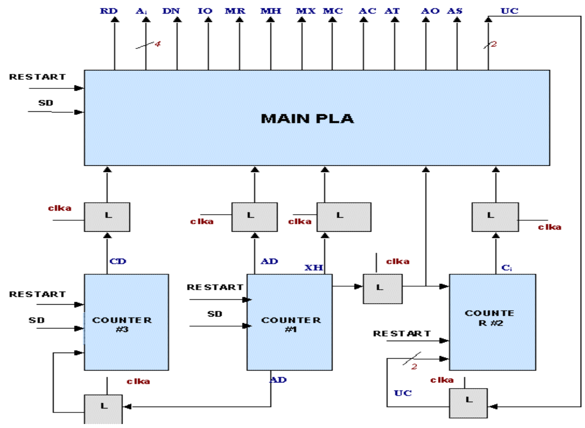

Fig. 6. PLA design with counters

The major inputs and outputs of this PLA design are summaried in TABLE.II.

TABLE II. PLA FSM Inputs/Outputs

Inputs	From	Note
RESTART	Input pin	Reset the PLA to idle states
SD	Input pin	Start the DWT process
CD	Counter #3	Switch calculation between d_i & c_i
AD	Counter #1	Accumulation Done
XH	Counter #1	Loading data / Loading coefficients
MD	Counter #2	Multiplication Done
C_i (i= 0,1)	Counter #2	Choose which to be calculated
Outputs	**Destination**	
RD	Output pin	Data is ready for output
MH	Multiplier	Load Wavelet Coefficient to Multiplier
MX	Multiplier	Load data to Multiplier
MC	Multiplier	Clear input latch of Multiplier
MR	Memory	Read Memory Enable
A_i(i=1~4)	Memory	Address for Decoder to Load Data
MD	Main PLA, Counter2	Multiplication Done
AC	Accumulator	Clear input latch of Accumulator
AS	Accumulator	Change to subtraction mode
AO	Accumulator	Accumulator output enable
AT	Accumulator	Load temporary result.

VI. CONCLUSION

The design is implemented and fabricated using the AMI 1.5 micron CMOS process through the MOSIS prototyping service. The mask layout is shown in Fig. 7. The size of the chip core is 2.2*mm** 2.2*mm*. Although this is not the most advanced technique in fabrication, it is enough for the purpose of prototyping. The simulation result showed correct computation of the design with reduced complexity compared to the original Mallat's algorithm. A detailed description of the work can be found in [3].

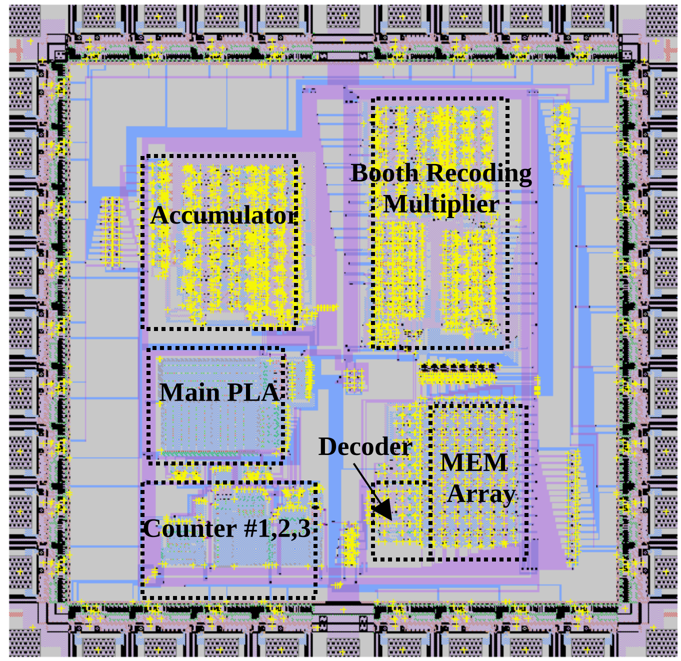

Fig. 7. The complete layout of DWT ASIC

ACKNOWLEDGEMENTS

The authors are sincerely grateful to MOSIS for the support of fabrication and to the anonymous reviewers for their helpful comments.

REFERENCES

[1]. Stéphane Mallat, "A wavelet tour of signal processing", Academic Press, 1999 .

[2]. Weste & Eshraghian, "Principle of CMOS VLSI design", second edition, Addison Wesley, 1993.

[3]. DWT VLSI Design Project Home page: http://www.ece.rice.edu/~ybguo/422/Finalreport/index_422rpt.htm.

[4]. M. Vishwanath, "The recursive pyramid algorithm for the discrete wavelet transform",*IEEE Trans. on Signal Processing*, vol 42, no. 3, 1994, pp. 673–676.

[5]. M. Vishwanath, R.M. Owens, and M.J. Irwin, "VLSI architectures for the discrete wavelet transform," *IEEE Trans. On Circuits and Systems-II*, vol. 42, no. 5, 1995, pp. 305–316.

[6]. A. Grzeszczak, M.K. Mandal, S. Panchanathan, and T. Yeap, "VLSI implementation of discrete wavelet transform," *IEEE Transactions on VLSI Systems*, vol. 4, no. 4, 1996, pp. 421–433.