FPGA Implementation of 1D and 2D DWT Architecture using Modified Lifting Scheme

M. NAGABUSHANAM
Department of Electronics and Communication Engineering
M.S. Ramaiah Institute of Technology, Affiliated to VTU, Bangalore – 560054
E-mail: nagabushanam1971@gmail.com
Tel: 09986019083, +91-080-23600822 / 23603123; Fax: +91-080-23600822

S. RAMACHANDRAN
Professor, Dept of ECE, SJBIT, Bangalore
E-mail: ramachandr@gmail.com

P.KUMAR
Professor, Dept of ECE, K.S.Rangasamy coll of Tech
Tiruchengode, Namakkal District, TamilNadu-637215
E-mail: kumar_ksrct@yahoo.co

Abstract
Image compression is one of the prominent topics in image processing that plays a very important role in reducing image size for real-time transmission and storage. Many of the standards recommend the use of DWT for image compression. The computational complexity of DWT imposes a major challenge for the real-time use of DWT-based image compression algorithms. In this paper, we propose a modified lifting scheme for computing the approximation and detailed coefficients of DWT. The modified equations use, right shift operators and 6-bit multipliers. The hierarchy levels in computation are reduced to one; thereby minimizing the delay and increasing throughput. The design implemented on Virtex-5 FPGA operates at 180 MHz and consumes less than 1W of power. The design occupies less than 1% of the LUT resources on FPGA. The architecture developed is suitable for real-time image processing on FPGA platform.

Key Words: DWT, Image compression, BZFAD multiplier, FPGA, Lifting scheme

1. Introduction

Wavelet transformation is a widely used technique for image processing applications. Unlike traditional transforms such as the Fast Fourier Transform (FFT) and Discrete Cosine Transform (DCT), the Discrete Wavelet Transform (DWT) holds both time and frequency information, and is based on a multi-resolution analysis framework [1]. IMAGE compression technique based on 2-D discrete wavelet transform (DWT) has already gained superiority over traditional JPEG based discrete cosine transform and is standardized in forms like JPEG2000 [1]. DWT is recommended to be used in video coding standards such as H.261-3, MPEG1-2,4 by rendering the quality features like better peak signal-to-noise ratio (PSNR), absence of blocky artifacts in low bit rates. Furthermore, it has the added provision of highly scalable compression, which is mostly coveted in modern communications over heterogeneous channels like the Internet [2]. The DWT is implemented by the lifting scheme in JPEG 2000. Compared to the traditional convolution, computing complexity of lifting scheme [3][4] is reduced about by half. Two-dimensional DWT is usually implemented directly in [5], that is, firstly computing the image along line, then along column. The approach needs large memory to store intermediate results. But the line-based architecture [6]-[8] minimizes the memory. W. Chang[8] describes a line-based architecture for 2-D DWT using lifting scheme, which consists of one row filter, one column filter and on-chip memory. It requires 9N storage cells for computing N×N image using 9/7 wavelet and O(N2) clock cycles (ccs). K. Andra [7] to generalizes the lifting-based architecture, which consists of two row processors, two column processors and two memory modules. But the memory control logic of the architecture is complex, and O(N2)ccs are needed for an N×N image. G. Dillen [6] describes a combined line-based architecture for IDWT which needs two row processors. It is necessary to increase the resource requirement to compute N×N image in

O(N2/2) ccs. The schemes reported address computation of DWT coefficients based on various inputs and computation order. In [9] a modified algorithm for lifting computation was presented; the critical path delay for the lifting equations is 5*Tm* + 8*Ta*, where *Tm* and *Ta* denote the multiplier and adder delay respectively [9]. The primary reason behind this large delay is stacking of multipliers from the inputs to outputs. To inhibit the effect, the mechanism of flipping has been introduced in [9] which scales the delay down to 3*Tm* + 4*Ta*. As a fruitful result, the processing speed increases significantly when the flipped equations are mapped into hardware. With the flipping and shifting of lifting coefficients, the delay of the critical path was reduced. However, the lifting coefficients are non integers thus need a suitable number representation scheme for hardware implementation. In this work, we propose an algorithm that operates on integer lifting coefficients, and in order to further reduce delay and power, a pipelined architecture is proposed and implemented.

Section II presents the lifting based DWT algorithm, Section III presents the modified lifting DWT algorithm, section IV presents results and discussion and section V presents the conclusion.

2. Lifting Based DWT Scheme

In the traditional implementation of DWT, a pair of finite impulse response filters (FIR) (high-pass and low-pass filter) is applied in parallel. Mallat's pyramid algorithm [9] computes the one dimensional (1-D) convolution based DWT at different levels of resolution. The first level decomposition can be represented by using the block diagram illustrated in Figure 1.

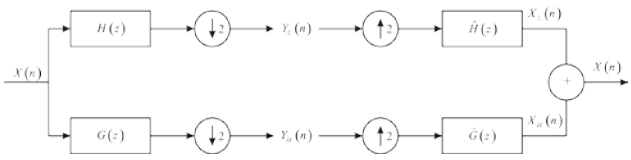

Figure 1 Single 1D-DWT Block [5]

The input sequence $X(n)$ in Figure 1 is convolved using quadrature mirror filters $H(z)$ and $G(z)$, and the outputs obtained are decimated by a factor of two. This reduces the time resolution by half and conversely doubles the frequency resolution by two. The 1D-DWT input signal $X(n)$ produces two sub-band coefficients $Y_L(n)$ and $Y_H(n)$ for a stage of decomposition. In the synthesis stage, scaling and wavelet coefficients $Y_L(n)$ and $Y_H(n)$ are treated inversely by up-sampling and filtering with low pass $\hat{H}(z)$ and high pass $\hat{G}(z)$ filters to perform reconstruction. This stage is also called Inverse Discrete Wavelet Transform (IDWT) [5]. The convolution-based 1-D DWT requires both a large number of arithmetic computations and a large memory for storage. Such features are not desirable for either high speed or low-power image processing applications. Recently, a new mathematical formulation for wavelet transformation has been proposed in [10], [11] as a light-weighted computation method for performing wavelet transforms with low power techniques. The main feature of the lifting-based wavelet transform is to break-up the high pass and the low pass wavelet filters into a sequence of smaller filters. The lifting scheme requires fewer computations compared to the convolution-based DWT. Therefore the computational complexity is reduced to almost a half of that needed with a convolution based approach. The lifting-based wavelet transform basically consists of three steps, which are called split, lifting, and scaling, respectively, as shown in Figure 2.

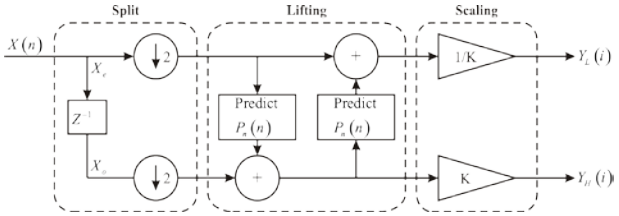

Figure 2 Lifting Scheme Implementation of 1D-DWT [7]

Lifting scheme is called as the second generation wavelet [5] because of the construction of bi-orthogonal wavelets which donot require the Fourier transform. The lifting scheme, which leads to a faster, fully in-place implementation of the wavelet transform, is attractive for both high throughput and low-power applications. The basic idea of lifting scheme is first to compute a trivial wavelet (or lazy wavelet transform) by splitting the original 1-D signal into odd and even indexed sub sequences, and then modifying these values using alternating prediction and updating steps. The lifting scheme algorithm can be described as follows:

1. **Split step:** The original signal, $X(n)$, is split into odd and even samples (lazy wavelet transform).
2. **Lifting step:** This step is executed as N sub-steps (depending on the type of the filter),

where the odd and even samples are filtered by the prediction and update filters, $P_n(n)$ and $U_n(n)$.

3. **Normalization or Scaling step**:After N lifting steps, scaling coefficients K and 1/K are applied respectively to the odd and even samples in order to obtain the low pass subband$(Y_L(i))$, and the high-pass subband$(Y_H(i))$.

Figure 3 shows the detailed block diagram of lifting-based DWT algorithm.

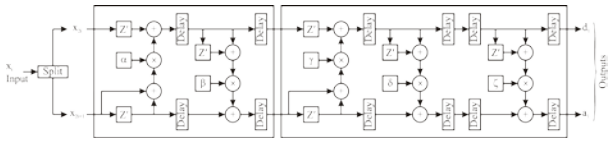

Figure 3 Block diagram of Lifting-based Algorithm [12]

The z^{-1} blocks are for delay; α, β, γ, δ, ζ are the lifting coefficients, and the shaded blocks are registers. The diagram shows the lifting scheme for Daubechies[9, 7]biorthogonal filter. The lifting equation based on 9/7 filter coefficients can be discussed in detail based on the equations given in Eq. (1) – Eq. (8). There are four stages in DWT computation, the split stage, predict and update stage and the scaling stage. When using 9/7 filters, there are two predict and two update stages [13] [14]. Various architectures for realizing DWT based on distributive arithmetic parallel distributive arithmetic, and modified distributive arithmetic have been discussed in [15] [16] [17] [18].

1. Split step

$$Xe \leftarrow X(2i) \quad \textit{Even Samples} \quad \ldots (1)$$

$$Xo \leftarrow X(2i+1) \quad \textit{Odd Samples} \quad \ldots (2)$$

In this stage the input sequence X is split into two output sequences of even and odd samples, and are stored in two separate registers.

2. Lifting Steps

For (9, 7) filter, $N = 2$

Predict P1**:** $D(i) = Xo(i) + a\left[Xe(i) + Xe(i+1)\right] \ldots (3)$

Update U1: $S(i) = Xe(i) + b\left[D(i-1) + D(i)\right] \ldots (4)$

Predict P2: $Y_H(i) = D(i) + c\left[S(i) + S(i+1)\right] \ldots (5)$

Update U2: $Y_L(i) = S(i) + d\left[Y_H(i-1) + Y_H(i)\right] \ldots (6)$

In the predict stage, the even samples are averaged, and added with the odd samples to predict the next even and odd sample. In the update phase, the differences between the predicted sample and the actual sample is updated.

3. Scaling Step

$$YH(i) = KY_H(i) \quad \ldots (7)$$

$$YL(i) = \frac{1}{K} Y_L(i) \quad \ldots (8)$$

In the last stage, the DWT coefficients are scaled to obtain the approximation and detail coefficients. The predict, update and scaling coefficients a, b, c, d and K are derived from 9/7 filter coefficients. Where,

$a = -1.586134342$, $b = -0.0529801185$,

$c = 0.882911076$, $d = -0.443506852$

and $K = 1.149604398$. In the lifting scheme computation Eq. (1) to Eq. (8), even and odd samples are added and are scaled with coefficients a, b, c and d in the lifting steps and added. As the input sequences arrive at the input memory, the split logic splits the samples into even and odd, thus one clock cycle is required. In the next step, to compute the P1 sample, present odd sample (i) and the present even sample (i) along with next even sample (i+1) is required to compute the D(i) sample. Therefore this operation requires two clock cycles to obtain data and to add the samples, one clock cycle for multiplication by *a* coefficient, and one clock cycle for addition with odd sample. Thus the lifting steps P1 has a latency of 4 clocks, similarly U2 has a latency of 4 clocks, and thus the combined predict 1 and update 1 phase has a latency of 8 clock cycles. The predict 2 and update 2 have a latency of 8 clock cycles. The final step which is scaling has a latency of one clock cycle. Thus the computation of approximation and detail coefficient has a latency of 20 clock cycles. The computation of predict and update samples at every stage is dependent on the present sample and the previous sample, hence there is a delay of 2 clock cycles; thus the throughput is 4 clock cycles as there are two stages of predict and two stages of update.

3. Modified Lifting based Algorithm

In the lifting equations discussed in the previous section, the latency is observed to be 20 clockcycles and the throughput is of 4 clock cycles, as the intermediate output depends upon present and next samples. Also the lifting scheme coefficients are fractions and thus require fixed point or floating point number representation. The arithmetic units such as multipliers and adders need to be designed

to operate on fixed or floating point numbers. In order to improve the throughput and reduce latency, a modified lifting scheme algorithm is discussed in this section. The modified algorithm eliminates the use of fractional coefficients and thus requires integer based adders and multipliers.

Considering the predict 1 and update 1 expressions as given in Eq. (3) and Eq. (4), S(i) samples depend upon Xe(i), D(i) and D(i-1) samples, D(i) sample requires Xo(i), Xe(i) and Xe(i+1) samples. Thus, in order to reduce this interdependency, Eq. (3) is substituted in Eq. (4), and is reduced to Eq. (9),

UpdateU1:

$$S(i) = Xe(i) + b\begin{bmatrix} Xo(i-1) + a\left[Xe(i-1) + Xe(i)\right] \\ + Xo(i) + a\left[Xe(i) + Xe(i+1)\right] \end{bmatrix}$$

Simplified as **U1:**

$$S(i) = (1+2ab)\,Xe(i) + bXo(i-1) + ab\left[Xe(i-1) + Xe(i+1)\right] \quad \ldots (9)$$

The coefficients and samples are regrouped and a minimized equation is obtained in Eq. (9). Similarly Eq. (5) and Eq. (6) are modified to obtain Eq. (10). Similarly, the U2 equations can be expressed as:

Update U2:

$$Y_L(i) = S(i) + d\begin{bmatrix} D(i-1) + c\left[S(i-1) + S(i)\right] \\ + D(i) + c\left[S(i) + S(i+1)\right] \end{bmatrix} \ldots (10)$$

Simplified as **U2:**

$$Y_L(i) = (1+dc)\,S(i) + dD(i-1) + dc\left[S(i-1) + S(i+1)\right]$$

Substituting Eq. (9) and Eq. (10) into Eq. (7) the expression for YL coefficient is obtained in Eq. (11).

$$YL(i) = 1/K\left(Y_L(i)\right) = \left[(1+dc)/K\right]S(i) + \left[d/K\right]D(i-1) + \left[dc/K\right]\left[S(i-1) + S(i+1)\right] \quad \ldots (11)$$

Regrouping the samples and coefficients, the modified expression for YL is given in Eq. (11). The new coefficients that are obtained are further scaled by 256, and the integer part is retained. The rounded coefficients are used for computation of Eq. (11). Table 1 shows the reduced coefficients and their expressions. Numerical values of reduced coefficients, scaled coefficients and rounded coefficients.

$$YL(i) = \left[A1\right]S(i) + \left[A2\right]D(i-1) + \left[A3\right]\left[S(i-1) + S(i+1)\right]$$

The coefficients are scaled to integer values and are presented in Table 1.

Table 1 Rounded and Scaled Coefficients

Coefficients	Values	256* Coefficients/256	Rounded Coefficients
$1+2ab$	+1.08403358539	+227.512/256	+278/256
b	−0.0529801185	−13.56/256	−14/256
ab	+0.08403358539	+21.51/256	+22/256
$1+dc$	0.60842288808	+155.75/256	+156/256
d	−0.443506852	−113.53/256	−114/256
dc	−0.39157711191	−100.24/256	−100/256
$A1 = ((1+dc)/K)$	0.529245442291	135.48683322/256	135/256
$A2 = (d/K)$	−0.38579084400	−98.762456290/256	−99/256
$A3 = (dc/K)$	−0.34061900990	−87.1984665536/256	−87/256

From the numerical values shown in Table 1, the scaled coefficients are rounded off to the nearest integer. As the coefficients are scaled by 256, after every multiplication operation is performed, the product obtained is right shifted by 8 bits.In other words, the LSB 8-bits are removed and the MSB bits are retained, thus performing the inverse scaling operation. The advantage of this process is that the arithmetic units operate on integer samples and thus reduce complexity of the hardware.Also the right shift operation performed after multiplication retains the information after every operation. The simplified equations have a latency of 3 clock cycles and reduce throughput by 2 clock cycles as compared with the basic lifting scheme equations. In addition, the fractional coefficients are scaled to integers and the division operation scaled by 256 is performed by right-shift operations, thus minimizing the circuit complexity.

As discussed previously, the computation of YH term is also carried out in a similar manner. The equations (3) and (9) are used to show the procedure for computing the modified YH term. From the lifting equations presented previously, $D(i)$, $S(i)$ and $S(i+1)$ are given by:

$$D(i) = Xo(i) + a\left[Xe(i) + Xe(i+1)\right]$$

$$S(i) = (1+2ab)\,Xe(i) + bXo(i-1) + ab\left[Xe(i-1) + Xe(i+1)\right]$$

$$S(i+1) = (1+2ab)\,Xe(i+1) + bXo(i) + ab\left[Xe(i) + Xe(i+2)\right]$$

Substituting, $D(i)$, $S(i)$ and $S(i+1)$ in $Y_H(i)$ we get,

$$Y_H(i) = \left[Xo(i) + aXe(i) + aXe(i+1)\right] + c\begin{bmatrix} (1+2ab)\,Xe(i) + bXo(i-1) \\ + ab\left[Xe(i-1)\right] + (1+2ab)\,Xe(i+1) \\ + bXo(i) + ab\left[Xe(i) + Xe(i+2)\right] \end{bmatrix}$$

Substituting $Y_H(i)$ in $YH(i)$ and simplifying the terms reduces the above equation as in Eq.(13)

$$YH(i) = A4\left[Xe(i) + aXe(i+1)\right] + A5\left[Xe(i-1) + Xe(i+2)\right] + A6\,Xo(i) + A7\,Xo(i-1)$$

The terms A4, A4, A6 and A7 are defined in table 2 and are scaled to integer values. As the coefficients are fractions, fixed point arithmetic units are required; scaling coefficients to integers, integer arithmetic units are required to realize the algorithm.

Table 2 Rounded and scaled coefficient in computing detail coefficients

Coefficients	Expression	Value	256* Coefficients/256	Rounded Coefficients
A4	$K(a+c(1+2ab)+abc)$	−0.6378406406	−163.28720400/256	−163
A5	$K(abc)$	0.0852939594	21.83525361/256	22
A6	$K(bc+1)$	−1.095829659	−280.5323928/256	−280
A7	Kcb	0.0537747384	13.7663330/256	14

The $YL(i)$ and $YH(i)$ terms are obtained based on the equations discussed above and are presented in Eq. (12) and Eq. (13).

$$YL(i) = 135*S(i) - 99*D(i-1) - 87*\left[S(i-1) + S(i+1)\right] \quad \ldots (12)$$

$$YH(i) = -163*\left[Xe(i) + Xe(i+1)\right] + 22*\left[Xe(i-1) + Xe(i+2)\right] - 280*Xo(i) + 14*Xo(i-1) \quad \ldots (13)$$

The coefficients require more than 8 bits for representation; in order to reduce the multiplier size, the coefficients are factored and the equations are described in (14),

$$YL(i) = (64+71)*S(i) - (64+35)*D(i-1) - (64+23)*\left[S(i-1+S(i+1))\right] \quad \ldots (14)$$

The samples are multiplied by a constant coefficient 64, which can be realized by right-shifting the samples by 8; hence the coefficients are split into two parts as $(64+x)$, where $x = 71, 35, 23$. The modified equations are in (15)

$$YL(i) = (71)*S(i) + \text{right shift } 6\left(S(i)\right) - (35)*D(i-1) - \text{right shift } 6\left(D(i-1)\right) - (23)*\left[S(i-1) + S(i+1)\right] - \text{right shift } 6\left(\left[S(i-1) + S(i+1)\right]\right) \quad \ldots (15)$$

Figure 4 shows the architecture derived for the modified lifting equations shown in Equation (15). The coefficients 71, 35 and 23 can be represented using six bits, and hence reduces the size of the multiplier architecture. The right-shift operations also eliminate the use of multipliers. The overall architecture designed as above, reduces the computational complexity of YL computations.

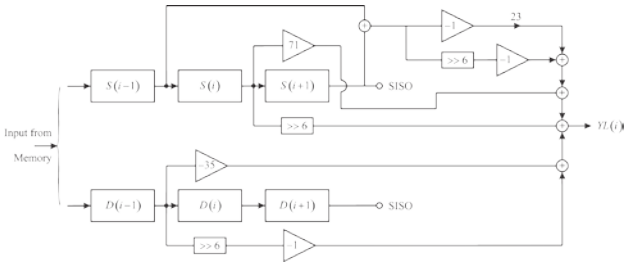

Figure 4 Modified Architecture for YL Computation

Similarly the YH term can be realized as in (16),

$$YH(i) = -35*\left[Xe(i) + Xe(i+1)\right] - \text{right shift } 7\left(\left[Xe(i) + Xe(i+1)\right]\right) + 22*\left[Xe(i-1) + Xe(i+2)\right] - 24*Xo(i) - \text{right shift } 7\left(Xo(i)\right) - \text{right shift } 7\left(Xo(i) + 14*Xo(i-1)\right) \quad \ldots (16)$$

Figure 5 shows the architecture derived for modified lifting scheme based in Equation (16).

Figure 5 Modified Architecture for YH Computation

From Eq. (15) and Eq. (16), the size of the multiplier is reduced to six bits and right shift7 and rightshift6 operators are used; hence the overall computational complexity has been optimized. Wallace tree multiplier [19] and carrysave adder are used for architecture implementation on FPGA. The 2D-DWT architecture is shown in figure 6, which consists of two stages of 1D-DWT realized using modified lifting scheme, in the 2D-DWT three blocks of YL computation and three blocks of YH computation along with intermediate memory decomposes the N×N image into 4 sub bands of N/2. The computation time is as discussed in table-3. The HDL model for the modified equations is developed and verified for its functionality using

Model Sim. The developed equations are synthesized using ISE and implemented on FPGA. The next section discusses the functional and synthesis results of the modified lifting scheme equations.

Table 3

Computation Time	
1D DWT using Modified Lifting	
Image Size	N×N
Number of Rows	N rows of N elements
Number of stages in modified lifting scheme	3 stages in YL Computation.
	4 stages in YH computation.
Arithmetic unit in modified Lifting Scheme (AU)	Multiplier Adder
Latency	5 clock cycles for YL [3 clocks for SISO + 2 clocks for A.U]
	6 clock cycles for YH [4 clocks for SISO + 2 clocks for A.U]
Throughput	½ YH computed simultaneously
Total time	$[N^2 + 5]$ clocks
2D DWT using Modified Lifting	
Computation time	Memory rearrangement time + 1D DWT computation time.
	[N^2 clocks + N^2 + 5 clocks]

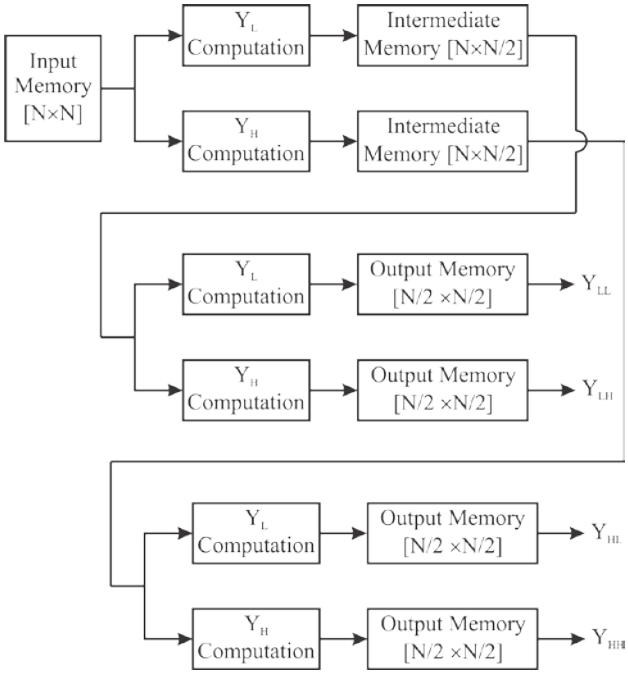

Figure 6 2D-DWT using Modified Lifting Scheme

5. Software Modelling

The HDL model for the modified lifting based DWT algorithm is simulated in Model Sim. The input image which is of size N x N is reorganized to N*N x 8 bit memory. The image is read row wise and is stored in the memory. The HDL model reads the input data from the memory and computes the L and H sub-band outputs. The L and H sub-band components are stored in two separate intermediate memories. In the second level, the L and H subbands are accessed simultaneously and are processed using two 1-D DWT architectures that consist of high pass and low pass filters. The second stage produces four outputs and hence the image data is subdivided into 4 sub bands of LL, LH, HL and HH. The flow chart shown in Figure 7, where the frame i is read and 1-D DWT is computed, and the intermediate result is stored in another ROM (ROM2), further the data is read out and 1D-DWT is computed in column wise and the result is stored in ROM 3. The process is repeated in all rows and all frames. In the second stage, the process is repeated with two different 1-D DWT architectures that operate in parallel.

Figure 7 Modified lifting based 2D DWT computation flow chart for level 1

Figure 8 shows the flow chart for 2D-DWT computation for two levels. In the second stage, two 1-D DWT processes are carried out in parallel to compute four sub bands of input data.

Figure 8 Modified lifting based 2D DWT computation flow chart for level 2

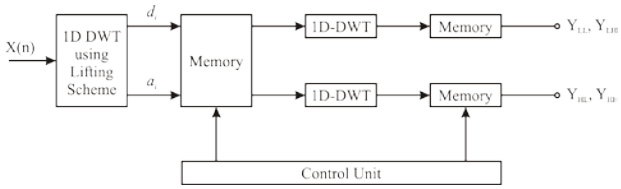

Figure 9 2D-DWT Architecture

The 2-D DWT architecture using the proposed modified lifting scheme is as shown in figure 9. The top level architecture consists of three 1-D DWT

processors realized using modified lifting scheme as discussed in equations 12 and 13.

6. Result and Discussion

A Test Bench model is developed using HDL and is used in functional simulation of an HDL model for modified DWT algorithm. Figure 10 shows the simulation results of 1D DWT computation for a given set of input vectors.

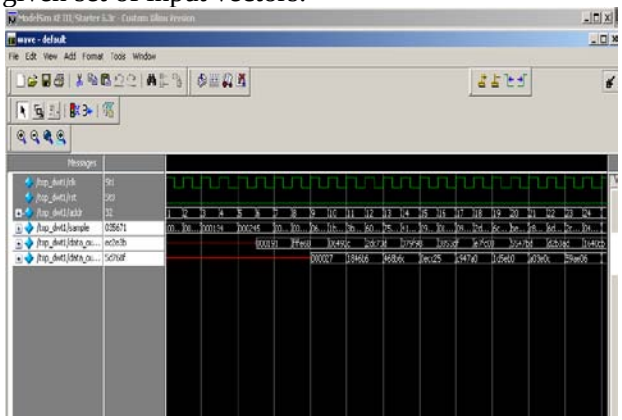

Figure 10 1D DWT output

An input clock and andRST signal trigger the memory read operations. The memory controller provides the address to the memory to read the row elements. The address is between location 1 and location 1024. The address of the memory enables the corresponding memory location and the data is read and is stored in the variable sample. After the initial latency the L output is computed after 5 clock cycles and the H output is computed after 6 clock cycles. The L and H output are stored in separate memory locations and are used for the computation of 2D DWT. Figure 11 shows the simulation results of 2D DWT. The L and H output samples are processed simultaneously to obtain LL, LH and HL, HL respectively. The clock and RST initiate the 2D DWT computation; the memory controller generates the corresponding address and is used in reading the intermediate memory for 2D DWT computation.

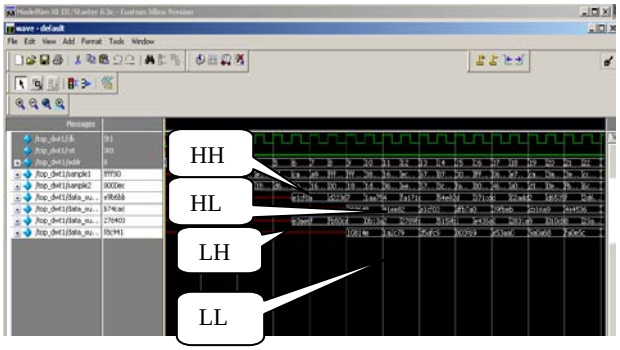

Figure 11 2D DWT output results

The functionally verified netlist is synthesized using Xilinx ISE and is implemented on Virtex-5 110 million gate device. The RTL model is synthesized by optimizing for area, power and speed. Figure 10 shows the top level block diagram of the netlist after synthesis. The RTL schematics of the proposed design with interconnects between the various blocks. It is a technology independent schematic.

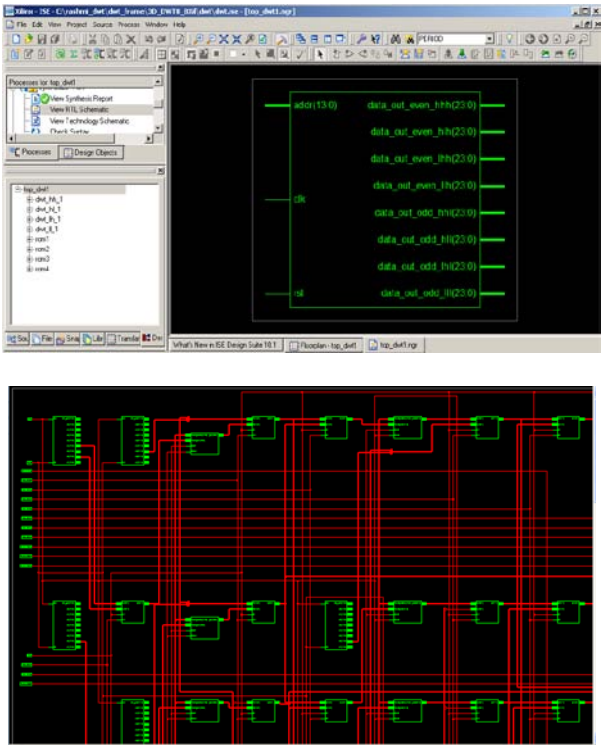

Figure 12 Top level diagram of 2D DWT

From the synthesis net list the following are captured and listed below:

- Total power dissipated: 0.98 W
- Minimum period: 5.54 ns (Maximum Frequency: 180.34 MHz)
- Number of BUFG/BUFGCTRLs: 6 out of 32 18%
- Number of bonded IOBs : 259 out of 640 40%
- Number of fully used LUT-FF pairs: 789 out of 9759 8%

The synthesized netlist is implemented on FPGA development kit, and Chipscope debugging is performed. The PC is interfaced with the development kit for programming. Once the interfacing is done, the corresponding programming file for the top module is generated. The target device is then configured so that the generated programming file can be successfully dumped on Virtex-5 as shown in Figure 13. The design is then

analyzed using Chip Scope Pro. The Chip Scope output is observed. The Chip Scope output is successfully compared with Model Sim simulation output.

Figure 13 FPGA Implementationand Debugging

The results obtained prove that the proposed DWT/IDT architecture is suitable for image transformation. The debugging results are found to be compatible with Model Sim simulation results.

Conclusion:

In this paper, we proposed a fast algorithm for 1D and 2D DWT computation based on lifting scheme algorithm. The six stages of lifting scheme are reduced to a single stage computation, by back substitution of intermediate stages.

The lifting coefficients have been reduced to seven constants; the derived constants are scaled to integers scaling and the modified lifting equation are realized. The derived equations are further modified to reduce the multiplier size from 9 bit to 6 bit, and the use of right shift operators further reduces the computational complexity. The modified architecture is designed to compute 1D and 2D DWT; the derived architecture is modeled using Verilog and simulated using ModelSim. Suitable test vectors are chosen to verify the logic correctness of 1D and 2D architectures. The verified model is synthesized using Xilinx ISE targeting Virtex-5 FPGA consisting of 110 million gates. The results obtained show that the proposed design operates at maximum frequency of 180MHz, and consumes power less than 1W, with less than 1% resource utilization. The present work is compared with[18] & [20] and presented in Table-4. The developed architecture can be extended to compute 3D DWT architecture. Suitable multipliers and adders can be adopted to design the modified architecture to improve the speed and power consumption.

Table 4 Comparison

Patent	Ref [18]	Present work	Ref [20]
Frequency of Operation	256MHz	180MHz	120MHz
Slice Registers	1,152 out of 19,200 (5%)	1,152 out of 19,200 (1%)	158 out of 4928(3%)
Power Dissipation	<30%	<1W	---
FPGA Type	Virtex – V	Virtex – V	Virtex – II

References

[1] Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal Process. Mag.*, vol. 18, no. 5, pp. 36–58, Sep. 2001.

[2] J.-R. Ohm, M. van der Schaar, and J. W. Woods, "Interframe wavelet coding: Motion picture representation for universal scalability," *J. Signal Process. Image Commun.*, vol. 19, no. 9, pp. 877–908, Oct. 2004.

[3] Daubechies I, Sweldens W. "Factoring wavelet transforms into lifting schemes," J. Fourier Anal. Appl,vol.4,pp.247~269,1998.

[4] Sweldens W. "The lifting scheme: a new philosophy in biorthogonal wavelet constructions," Proc SPIE,1995.

[5] M. Vishwanath,R.M. Owens,M.J.Irwin, "VLSI architecture for the discrete wavelet transform," IEEE trans. On circuit and systems-11, vol.42, no.5, pp.305-316,1995.

[6] G.Dillen,B.Georis. "Combined line-based architecture for the 5-3 and 9-7 wavelet transform for JPEG2000," IEEE Transaction on Circuits and Systems for Video Technology, vol.13, no.9, pp.944-950, 2003.

[7] Andra K, Chakrabarti C. "A VLSI architecture for lifting-based forward and inverse wavelet transform," IEEE Trans on Signal Processing, vol.50, no.4, pp.966-977, 2002.

[8] Wei-HsinChang,Yew-San Lee, Wen-shiawPeng, Chen-Yi Lee, "A Line-based, memory efficient and programmable architecture for 2D DWT using lifting

sheme," IEEE International Symposium on Circuits and Systems, 2001.

[9] S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Trans. Pattern Analysis And Machine Intelligence*, vol. 11, no. 7, 1989, pp. 674-693.

[10] K. Seth S.Srinivasan. "VLSI implementation of 2-D DWT/IDWT Cores using 9/7-tap filter banks based on the non-expansive symmetric extension scheme," Proceeding of the 15th international conference on VLSI Design, 2002.

[11] Tze-Yun Sung,Hsi-Chin Hsin Yaw-Shih Shieh and Chun-Wang Yu, "Low-Power Multiplierless 2-D DWT and IDWT Architectures Using 4-tap DaubechiesFilters",Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies,December 2006.

[12] NagabushanamM., and RamachandranS. (2012), 'Fast implementation of lifting based DWT architecture for image compression', Global journal of computer science & Technology(F),volume- XII, Issue XI, version 1,pp.23-29.(2012).

[13] Anirban Das, AnindyaHazara, and Swapna Banerjee, "An Efficient Architecture for 3-D Discrete Wavelet Transform" *IEEE* Transactinos on circuit and systems for video technology, vol. 20, no. 2, February 2010.

[14] TinkuAcharya and ChaitaliChakrabarti, "A Survey on Lifting-based Discrete Wavelt Transform Architectures," Springer Science, Journal of VLSI Signal Processing 42, 321-339, 2006

[15] NagabushanamM.,Cyril PrasannaRaj,and Ramachandran,S.'ModifiedVLSI implementation of DA-DWT for image compression',International Journal of Signal and Imaging Systems Engineering, published by inderscience,swizerland in Oct-2012,Vol. 5, No. 3, pp.167-174.

[16] Nagabushanam,M., Cyril Prasanna Raj, and Ramachandran,S.(2009),'Designand implementation of parallel and pipelinedDistributive arithmetic based discrete wavelet transformIP core', EJSR International Journal, Vol. 35.No. 3, pp.379–392.

[17] NagabushanamM., Cyril Prasanna Raj, and Ramachandran,S. (2011), 'Design and FPGA implementation of Modified Distributive arithmetic based DWT-IDWT processor for image compression', International Conference on Communication and Signal Processing, February, NIT, Calicut, India, p.69.

[18] Nagabushanam M and RamachandranS. (2012), 'Fast implementation of lifting based 1D/2D/3D DWT-IDWT architecture for image compression', International journal of computer Applications, volume-XII, Issue XI, version 1,pp.23-29.(2012).

[19] C. S. Wallace, "*A Suggestion for a Fast Multiplier*", IEEE Trans. computers, vol 13, no. 2, pp 14 – 17, 1964.

[20] Durgasowjanya, K.N.H. Srinivas and P. VenkataGanapathi'*FPGA Implementation of efficient VLSI Architecture for fixed point 1-D DWT using Lifting Scheme'*, International Journal of VLSI Design and Communication Systems (VLSICS), Vol.3, No.4, August 2012.