ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 1, April 2013

2D-DWT Lifting Based Implementation using VLSI for JPEG2000

Mrs.A.F. Mulla[1], Dr.Smt.R.S.Patil [2]

Asst.Prof. ETC, BVC, Kolhapur, India[1]

Professor, ETX, DYPCT, Kolhapur, India[2]

***Abstract*:** This paper proposes the design of VLSI architecture for image compression. The architecture has been simulated using behavioral VHDL. VHDL allows as much of the design as possible to be portable and flexible to other synthesis tools. To perform the process of image compression VLSI architecture is designed using lifting based discrete wavelet transform (DWT) and it is implemented in Spartan 3EDK kit. The decomposition algorithm of this transform is designed and synthesized with the VHDL language and then implemented on the FPGA Spartan 3E starter kit (XC3S500E) to check validation of results and performance of design. The VHDL model is validated through simulation using ModelSim-Altera. . The motivation in designing is to reduce its complexity, enhance its performance and to make suitable development on a reconfigurable FPGA based platform for VLSI implementation. The main feature of the lifting based DWT scheme is to break up the high pass and low pass filters into a sequence of upper and lower triangular matrices and convert the filter implementation into banded matrix multiplications [1], [2]. Such a scheme has several advantages, including "in-place" computation of the DWT, integer-to-integer wavelet transform (IWT), symmetric forward and inverse transform, etc. Traditional DWT architectures are based on convolutions. Then, the second-generation DWTs, which are based on lifting algorithms, are available. The outputs generated by the row and column processors are stored in memory modules. The memory modules are divided into multiple banks to accommodate high computational bandwidth requirements. The lifting based DWT architecture has the advantage of lower computational complexities and higher efficiencies. Through the DWT, signals can be decomposed into different sub bands with both time and frequency information. Compared with convolution-based ones; lifting-based architectures not only have lower computation complexity but also require less memory.

Keywords: Architecture, Flexible, DWT, Lifting scheme, FPGA, DWT processor

I. Introduction

Digital images play an important role both in daily life applications as well as in areas of research and technology. Due to the increasing traffic caused by multimedia information and digitized form of representation of images; image compression has become a necessity. The two-dimensional Discrete Wavelet Transform (2D DWT) is nowadays established as a key operation in image processing. Transmission and storage of uncompressed video would be extremely costly and impractical.

With the increasing use of multimedia technologies, image compression requires higher performance. To address needs and requirements of multimedia and internet applications, many efficient image compression techniques, with considerably different features have been developed [1].

A. Image compression using lifting algorithm

Now-a-days image compressions have become more popular for bandwidth saving purpose. Image compression is minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the Internet or downloaded from Web pages.

Lifting-Based DWT!

The main feature of the lifting based DWT scheme is to break up the high pass and low pass filters into a sequence of upper and lower triangular matrices and convert the filter implementation into banded matrix multiplications.

Triangular matrices is a matrix of the form

$$L = \begin{bmatrix} l_{1,1} & & & & 0 \\ l_{2,1} & l_{2,2} & & & \\ l_{3,1} & l_{3,2} & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \\ l_{n,1} & l_{n,2} & \dots & l_{n,n-1} & l_{n,n} \end{bmatrix} \quad \text{.....(a)}$$

is called a lower triangular matrix or left triangular matrix, and analogously a matrix of the form

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 1, April 2013

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \dots & u_{1,n} \\ & u_{2,2} & u_{2,3} & \dots & u_{2,n} \\ & & \ddots & \ddots & \vdots \\ & & & \ddots & u_{n-1,n} \\ 0 & & & & u_{n,n} \end{bmatrix} \quad \text{.....(b)}$$

is called an upper triangular matrix or right triangular matrix. The variable L (standing for lower or left) is commonly used to represent a lower triangular matrix, while the variable U (standing for upper) or R (standing for right) is commonly used for upper triangular matrix. A matrix that is both upper and lower triangular is diagonal.
The lifting scheme is a new algorithm proposed for the implementation of the wavelet transforms [2]. It can reduce the computational complexity of DWT involved with the convolution implementation. Furthermore, the extra memory required to store the results of the convolution can also be reduced by in place computation of the wavelet coefficient with the lifting scheme. Lifting wavelet transform has advantages over the ordinary wavelet transform by way of reduction in memory required for its implementation. The basic principle of the lifting scheme is to factorize the poly phase matrix of a wavelet filter into a sequence of alternating upper and lower triangular matrices and a diagonal matrix [1], [2]. This leads to the wavelet implementation by means of banded-matrix multiplications.

Let h(z) and g(z) be the low pass and high pass analysis filters, and let h (z) and g (z) be the low pass and high pass synthesis filters. The corresponding poly phase matrices are defined as

$$\tilde{P}_1(Z) = \begin{bmatrix} k & 0 \\ 0 & \frac{1}{k} \end{bmatrix} \prod_{i=1}^{m} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \qquad (1)$$

$$\tilde{P}_2(Z) = \begin{bmatrix} k & 0 \\ 0 & \frac{1}{k} \end{bmatrix} \prod_{i=1}^{m} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \qquad (2)$$

Where k is a constant, ti(z) and si(z) are denoted as primary lifting and dual lifting polynomial respectively and m represents the total lifting steps required.
A lifting stage is comprised of the four steps namely Split, Predict, Update and scaling.

- Split step, where the input signal (image) split into coefficients at odd & and even positions.
- Predict step, where the even samples are multiplied by the time domain equivalent of t (z) and are added to the odd samples,
- Update step, where updated odd samples are multiplied, by the time domain equivalent of s (z) and are added to the even samples,
- Scaling step, where the even samples are multiplied by 1/k and odd samples by k.

The lifting scheme is an efficient tool for constructing second generation wavelets and has advantages such as faster implementation, fully in place calculation, perfect reconstruction with low memory and low computational complexity. This is mainly because it achieves higher Compression ratios, due to the sub band decomposition it involves, while it eliminates the `blocking' artifacts that deprive the reconstructed image of the desired smoothness and continuity. The high algorithmic performance of the 2D DWT in image compression justifies its use as the kernel of both the JPEG-2000 image compression standard.

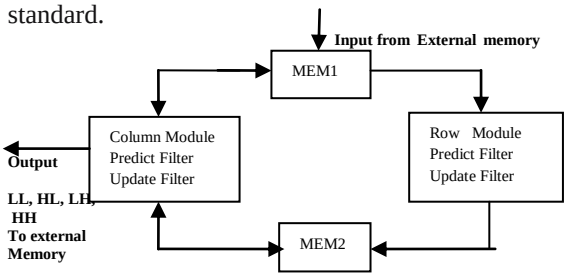


Figure 1. Lifting Based 2D DWT

The computations in each filter can be partitioned into prediction and update stages as shown in Figure 2. above. Here the row module reads the data from MEM1 performs the DWT along the rows (H and L) and writes the data into MEM2. The prediction filter of the column module reads the data from MEM2, performs column wise DWT along alternate rows (HH and LH) and writes the data into MEM2.The update filter of the column module reads the data from MEM2, performs column wise DWT along the remaining rows, and writes the LL data into MEM1 for higher octave computations and HL data to external memory. Final compressed image is stored into LL.HL is for recovering losses in compression. The core coding system of JPEG 2000 standard recommended two wavelet filters. If samples are numbered from 0, the even terms of the output terms form the low pass sub band and odd terms form the high pass sub band. The lifting based DWT has many advantages over the convolution based approach. Lifting based DWT typically requires less computation compared to convolution based approach (50% less) approximately.

B. Proposed VLSI Architecture

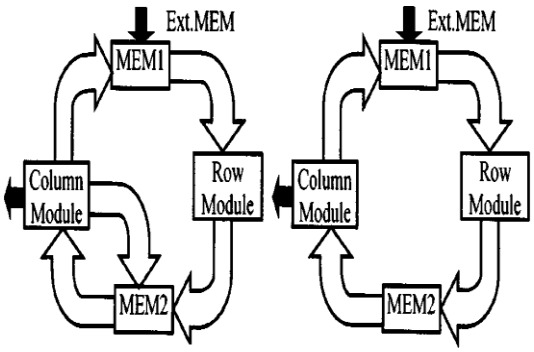


Figure 2. Data flow for 2M filters

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 1, April 2013

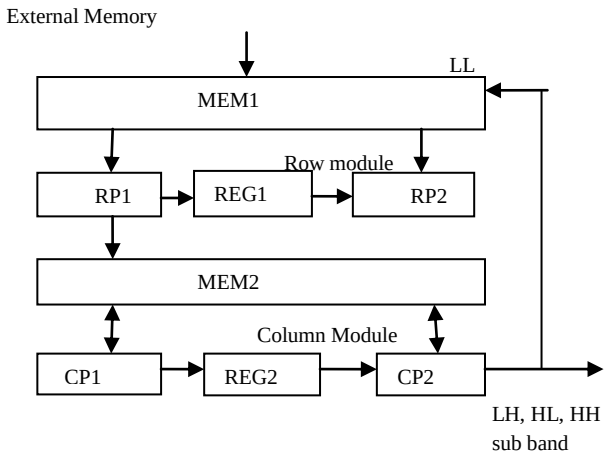


Figure 3. One Step Lifting Architecture

Case I- To perform the DWT, the architecture reads in the block of data, carries out the transform, and outputs the LH, HL, and HH data at each level of decomposition. The LL data is used for the next level of decomposition. At the end of the inverse Transform, the LL values of the next higher level are obtained. The transform values of the three sub bands (LH, HL, and HH) are read in, and the DWT is carried out on the new data set. The architecture, as shown in Figure 3, consists of a row module (two row processors RP1 and RP2 along with a register file REG1), a column module (two column processors CP1, CP2 and a register file REG2), and two mem0ry modules (MEM1, MEM2).

CaseII-In the case II (i.e., when lifting is implemented by two factorization matrices), processors RP1 and RP2 read the data from MEM1, perform the DWT along the rows, and write the data into MEM2. Processor CP1 reads the data from MEM2, performs the column wise DWT along alternate rows, and writes the HH and LH sub bands into MEM2 and Processor CP2 reads the data from MEM2, performs the column-wise DWT along the rows on which the CP1 did not work, and writes LL sub-band to MEM1 and HL sub-band to external memory the data Flow is shown in figure 3.

II. System Basic block diagram

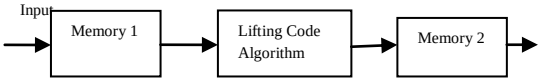


Figure 4. System Basic block diagram

***Memory 1*:** Input for this is BMP image of 8 bit resolution. It is RAM or ROM. Digital images can be compressed by eliminating redundant information present in the image, such as spatial redundancy, spectral redundancy, and temporal redundancy. For digital still and moving images, huge amount of disk space is required for storage and manipulation purpose so image compression is very essential to reduce storage need. Digital image is defined as a two dimensional function f(x, y), where x and y are spatial (plane) coordinates, and the amplitude of f at any pair of coordinates (x, y) is called intensity or grey level of the image at that point.

Lifting code algorithm:

The basic DWT can be realized by convolution-based implementation using the FIR-filters to do the transform. The input discrete signal X (n) is filtered by a low-pass filter (h) and a high-pass filter (g) at each transform level. The two output streams are then sub-sampled by simply dropping the alternate output samples in each stream to produce the low pass sub band YL and high-pass sub band YH. The equations are as in (3) (4). Figure5. shows the signal analysis in one dimensional (1-D) Discrete Wavelet Transform.

$$Y_L(n) = \sum_{i=0}^{N/2-1} h(2n-i)\, x(i) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3),$$

$$Y_H(n) = \sum_{i=0}^{N/2-1} g(2n-i)\, x(i) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

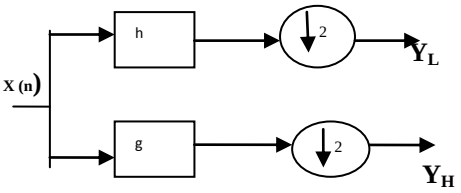


Figure 5. 1D DWT Decomposition

In convolution method, more hardware is required compared with that needed in Lifting scheme. It is replaced here with a multiplier less design. This output can be achieved with lifting approach with shifters and adders/ subtractors replacing multipliers. As a result of implementing DWT hardware, multipliers have been replaced by Shifters, thus giving less number of computations and make control complexity very simple. As the level of decomposition of image increases, more and more approximate and detailed information is available. Thus provides efficient mutiresolution analysis at different frequencies.

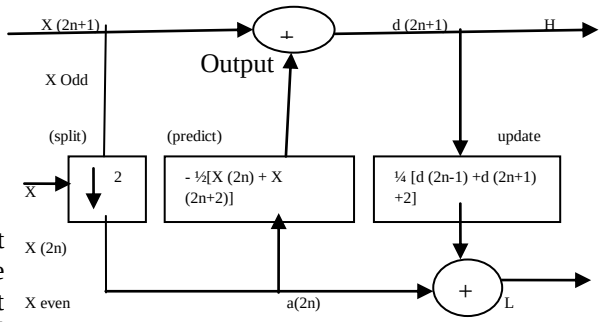


Figure 6. Lifting scheme decomposition of 5/3 filter

*A **Split***- Split the input signal (image) into coefficients at odd and even positions. In split step, the signal x (n) is

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 1, April 2013

split into even x (2n) and odd x (2n+1) points, because the maximum correlation between adjacent pixels can be utilized for the next predict step.

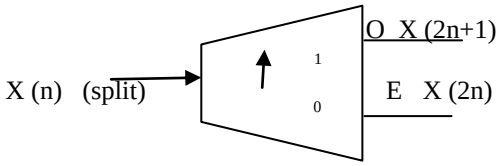


Figure 7. Split

B Predict- Perform a predict step the even samples are multiplied and then the results are added to the odd samples to generate the detailed coefficients (d) i.e., high pass coefficients. The predict step uses a function that approximates the data set. The differences between the approximation and the actual data replace the odd elements of the data set. The even elements are left unchanged and become the input for the next step in the transform. The predict step, where the odd value is "predicted" from the even value is described by the equation. Lifting means computing prediction and recording the detail.

$d = X_o - P(Xe)$ where d= difference

$$d(2n+1) = X(2n+1) - [X(2n) + X(2n+2)]/2 \quad \ldots\ldots\ldots\ldots (5)$$

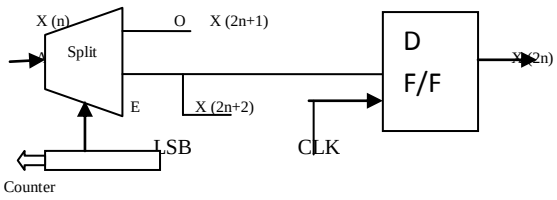


Figure 8. Predict d (2n+1) = X (2n+1) - [X (2n) + X (2n+2)] /2

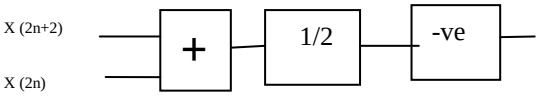


Figure 9. Realization of Predict P = - [X (2n) + X (2n+2)] /2

A] Realization of ½ blocks!

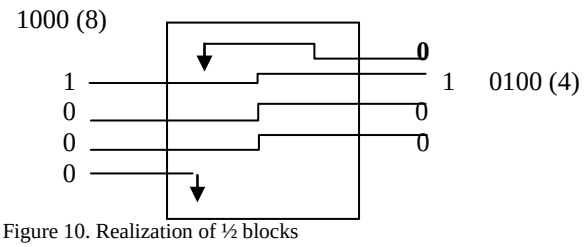
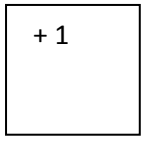


Figure 10. Realization of ½ blocks

B] Realization of -VE– 2's compliment!

-ve (P)



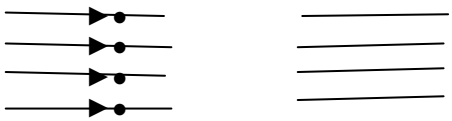


Figure 11. Realization of **2's compliment** blocks

C] Realization of +!

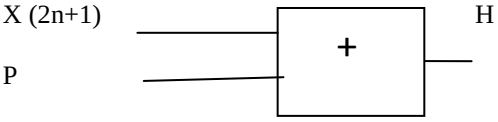


Figure12. Realization of + blocks H is high pass coefficient

C Update –

The detailed coefficients computed by the predict step is multiplied by the update factors and then the results are added to the even samples to get the coarse coefficient i.e., low pass coefficients. The update step replaces the even elements with an average. These results in a smoother input for the next step of the wavelet transform. The odd elements also represent an approximation of the original data set, which allows filters to be constructed. The update follows predict. The original values of the odd elements have been overwritten by the difference between the odd element and its even "predictor". So in calculating an average the update phase must operate on the differences that are stored in the odd elements.

$$a(2n) = X(2n) + ([d(2n-1) + d(2n+1) + 2])/4 \quad \ldots\ldots\ldots (6)$$

A] Realization of ¼ blocks !

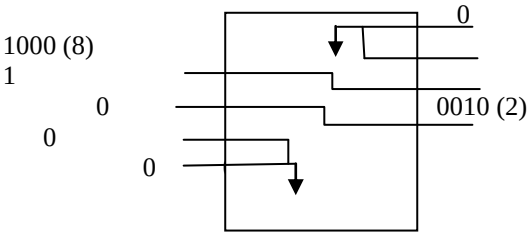


Figure 13.Realization of 1/4 blocks

B] Realization of X (2n)!

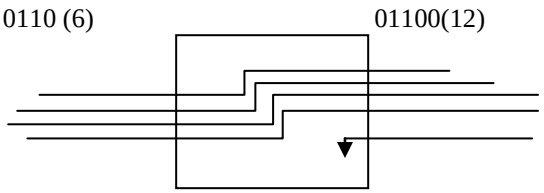


Figure 14. Realization of twice blocks

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 1, April 2013

c) ***Realization of Update is*** *a (2n) = X (2n) + ([d (2n-1) + d (2n +1) +2]) / 4!*

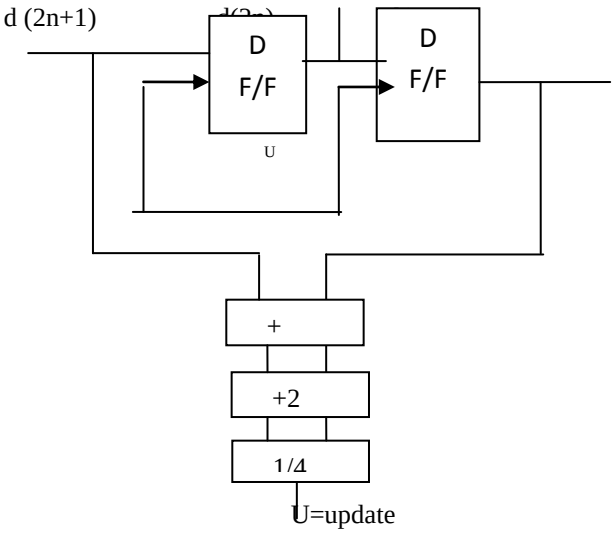


Figure 15. Realization of Update block

L = Low pass coefficient, U = Update

L = a (2n) + U

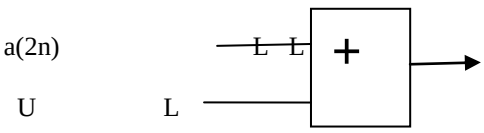


Figure 16. Realization of Low pass coefficient block

Waveforms are obtained after VHDL operations. In Waveforms data is compressed and it can be zoom.

A) After 500ns, 16 bit addition
a = 0100010000001111, b = 1111111111111111
a + b = sum = 10100010000001110

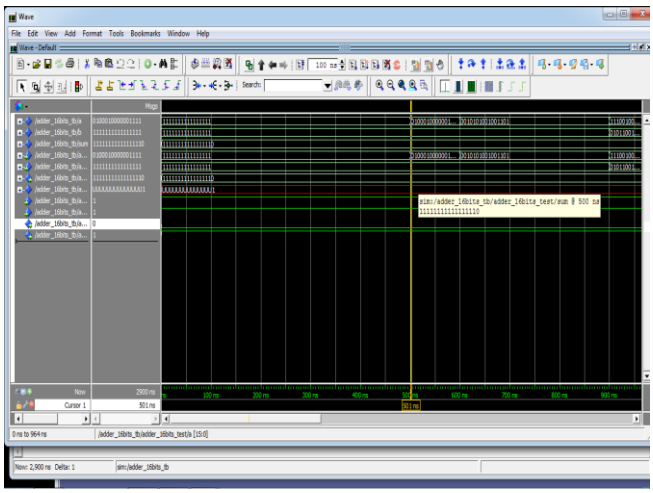


Figure17. Simulation of 16 bit additions after 500ns

B) After 600 ns, 16 bit addition
a = 0010101001001101
b = 1111111111111111
Sum = 1000101001001100

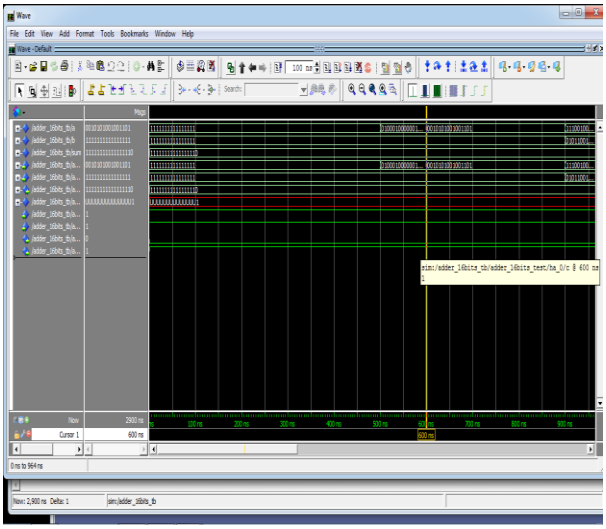


Figure18. Simulation of 16 bit additions after 600ns.

Simulation result of Split-

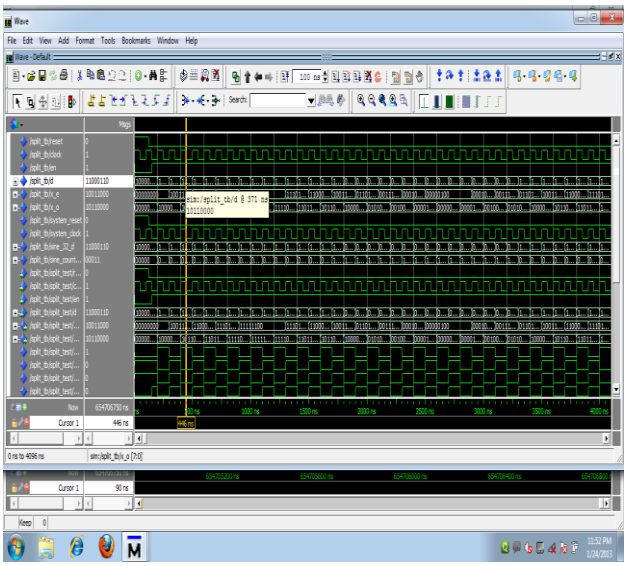


Figure19. Simulation result of Split step

Simulation result of Predict at 103ns

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 1, April 2013

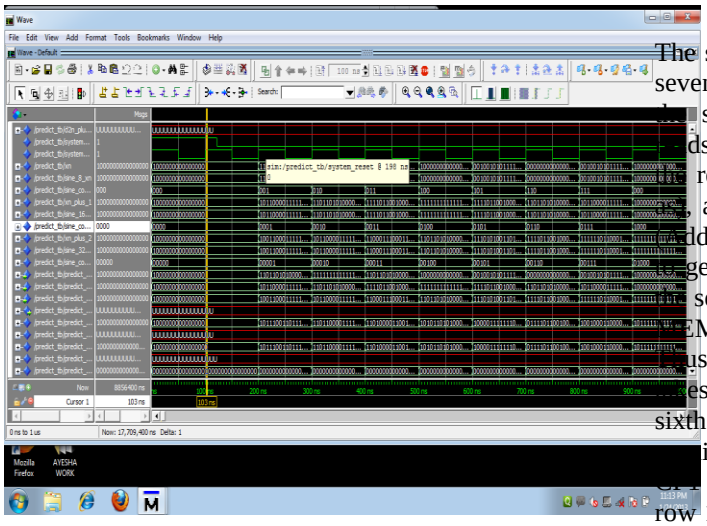


Figure20. Simulation result of Predict step at 103ns

Simulation result of Update at 200ns

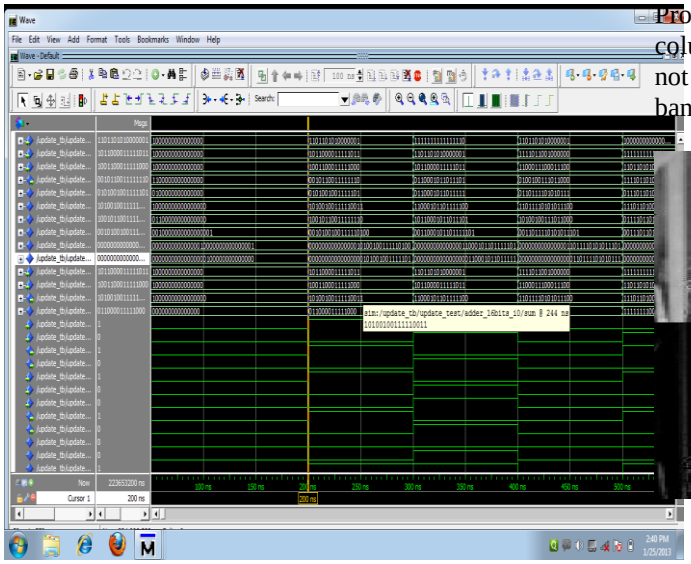


Figure 21. Simulation result of Update step at 200ns

Table I
Schedule for RP1 & RP2 for (5, 3) filter

Time	Rp1			Rp2		
	Adder1	Shifter	Adder2	Adder1	Shifter	Adder2
1	-	-	-	-	-	-
2	X0,0+X0,2	-	-	-	-	-
3	X0,2+X0,4	RA1	-	-	-	-
4	X0,4+X0,6	RA1	RS_X0,1	-	-	-
5	X0,6+X0,8	RA1	RS_X0,3	-	-	-
6	-	RA1	RS_X0,5	Y0,1,Y0,3	-	-
7	X2,0+X2,2	-	RS_X0,7	Y0,3,Y0,5	RA1	Y0,0
8	X2,2+X2,4	RA1	-	Y0,5,Y0,7	RA1	RS+Xo,2
9	X2,4+X2,6	RA1	RS_X2,1	-	RA1	RS+Xo,2
10	X2,6+X2,8	RA1	RS_X2,3	-	-	RS+Xo,2
11	-	RA1	RS_X2,5	Y2,1,Y2,3	-	Y0,8
12	X1,0+X1,2	-	RS_X2,7	Y2,3,Y2,5	RA1	Y2,0
13	X1,2+X1,4	RA1	-	Y2,5,Y2,7	RA1	RS+X2,2
14	X1,4+X1,6	RA1	RS_X1,1	-	RA1	RS+X,2,4

The schedule in Table I should be read as follows. In the seventh cycle, Adder1 of RP1 adds the elements and stores the sum in register RA1. The shifter (Shifter column) reads this sum in the next cycle (eighth cycle), carries out the required number of shifts (one right shift in this case) and stores the data in register RS. The second adder (Adder2) reads the value in RS and subtracts the element to generate in the next cycle (ninth cycle). The output of the second adder is stored in a suitable memory location in the MEM2 module and is also supplied to RP2 using REG1. Thus, to process a row of a 9 block, the RP1 processor takes four cycles. Adder 1 in RP2 starts computation in the sixth cycle. The gaps in the schedule for RP1 and RP2 are required to read the zeroth element of each row. Adder1 in CP2 starts in the 13th cycle to absorb the first element of row 1 computed byRP1 in the 14th cycle. Adder1 of CP2 starts after CP1 computes the first element in row 3 (25th cycle).

***Memory 2*:** Output is compressed image which is in RAM. Processor CP1 reads the data from MEM2, performs the column wise DWT along alternate rows, and writes the HH and LH sub bands into MEM2 and Ext.MEM. Processor CP2 reads the data from MEM2, performs the column-wise DWT along the rows on which the CP1 did not work, and writes LL sub-band to MEM1 and HL sub-band to Ext.MEM.

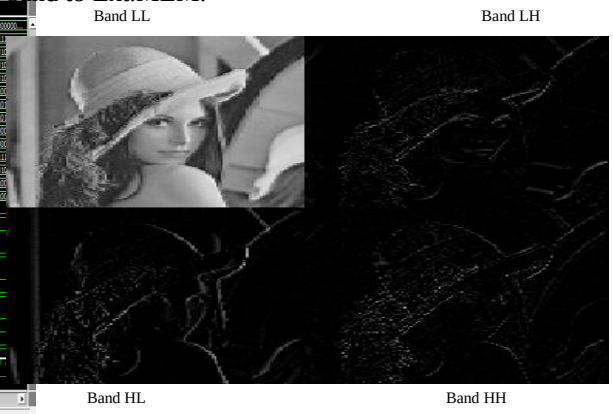


Figure 24. One level of 2D-DWT

The 2D-DWT, in first-level decomposition, the bus interface unit selects data (pixels) form input image. The transform module (Processing, processing band H and processing band L) decomposes to the four sub-bands LL1, LH1, HL1 and HH1, and saves LL1 band to the RAM module. After finishing the first level decomposition, the controller unit selects data from RAM module. The LL1 band is then sent to the module transform to perform the second level decomposition. The transform module decomposes the LL1band to the four sub-bands LL2, LH2, HL2 and HH2, and saves LL2 band to the RAM module for next level decomposition. This procedure repeats until the desired 3 levels (last level) decomposition is finished. The 1D-DWT, in first-level decomposition the bus interface unit selects data (pixels) form input image. The transform module

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 1, April 2013

(Processing) decomposes to the two sub-bands L1 and H1 and saves L1 band to the RAM module. In second level the controller unit selects data (band L1) from RAM module. The module transform (processing band L) decompose the band L1 to the two sub-bands L2 and H2 and save band L2 to the RAM module for next level decomposition.

V. RESULTS

We have applied the adaptive prediction algorithms to 256×256 8 bit images. The original image is transformed using adaptive prediction algorithms. Before any hardware implementation and testing is performed, all the VHDL modules are tested for correct functionality using the MODELSIM functional simulation tool. The MODELSIM tool provided the necessary environment for complete functional simulation of the target hardware (FPGAs). The MODELSIM tool provides high speed and target hardware platform adaptability. It also allows for modification and verification of the VHDL codes simultaneously. The hardware implementation is carried out by programming the FPGA through the parallel port of a computer. After implementing this design on the FPGA device, the resources utilization has been shown in the Table II and the maximum operation frequency is 215 MHz.

Table II
FPGA Synthesis results

Parameter	Values
Number of Slices	1835
Number of DFFs	1656
Number of LUTs	3359
Number of IOs	48
Number of GCLKs	1
Frequency (MHz)	108
Power (mW)	47

IMAGE QUALITY MEASURES (IQM)

1] MSE –MEAN SQURE ERROR

Let X (m, n) denotes the samples of original image, and X^T(m n) denotes the samples of compressed image. M and N are number of pixels in row and column directions respectively .Mean Square Error is given by:

$$MSE = 1/MN \sum_{m=1}^{M} \sum_{n=1}^{N} (X(m,n) - X^T(m,n)) \ldots\ldots (7)$$

MSE for Lena is 86.02 at .08bpp bit rate.
MSE for Lena is 67.84 at .1bpp bit rate.

2] PSNR – PEAK SIGNAL TO NOISE RATIO

$$PSNR = 10 \log_{10}((255)^2/ MSE)) \ldots\ldots\ldots (8)$$

The PSNR defined as:

$$PSNR = 10 * \log_{10} (MAX^2_I / MSE)$$
$$= 20 * \log_{10} (MAX_I / MSE^{½})$$
$$= 20 * \log_{10} (MAX_I) – 10*\log_{10} (MSE)$$

Where MAX_I = maximum possible pixel value of the image

When the pixels are 8 bits per sample, MAX_I is 255.

For Lena,

$$PSNR = 20 * \log_{10} (MAX_I) – 10*\log_{10} (MSE)$$
$$= 20* \log_{10}(255) -10 * \log_{10} (86.02)$$
$$= 28.785db.$$

Similarly for MSE = 67.84, PSNR is 29.82db.

3] MD-MAXIMUM DIFFERENCE

This parameter gives maximum difference in pixel values of two images.

$$MD = Max (|X (m, n)-X^T (m, n)|) \ldots\ldots\ldots\ldots (9)$$
$$= 81 \text{ at } .08bpp$$
$$= 75 \text{ at } .1bpp.$$

Figure25.a) Original Image of Lena

b) 1/4th compressed image of Lena

Thus Lena image is compressed with
PSNR = 28.785db (standard is between 25 to 30db),
MSE = 86.02.

REFERENCES:

Papers

[1] Alice Blessie, J. Nalini and S. C. Ramesh "Image Compression Using Wavelet Transform Based on the Lifting Scheme and its Implementation", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 1, May 2011.

[2] Sugreev Kaur and Rajesh Mehra "High Speed and Area Efficient 2D DWT Processor Based Image Compression", Signal and Image Processing: An International Journal (SIPIJ) Vol.1, No.2, December 2010 .DOI: 10.5121/sipij.2010.1203.

[3] A. Mansuri, A. Ahaitouf, and F. Abdi, "An Efficient VLSI Architecture and FPGA Implementation of High –Speed and Low Power 2-D DWT for (9,7) Wavelet Filter", IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.3, March 2009 BP : 2202 FES MOROCCO.

[4] M. Jeyaprakash, "FPGA Implementation of Discrete Wavelet Transform (DWT) for JPEG 2000", International Journal of Recent Trends in Engineering, Vol 2, No. 6, November 2009.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 1, April 2013

[5] Mountassar Maamoun, Mehdi Neggazi , Abdelhamid Meraghni,and Daoud Berkani, "VLSI Design of 2-D Discrete Wavelet Transform for Area-Efficient and High speed Image Computing".
[6] Ali M. - Haj Department of Computer Engineering, "An FPGA-Based Parallel Distributed Arithmetic Implementation of the 1-D Discrete Wavelet Transform", Informatics 29 (2005) 241-247, February 2004.
[7] Jie Guo; Ke-yan Wang; Cheng- ke Wu; Yun-song Li, "Efficient FPGA implementation of modified DWT for JPEG 2000 ";9th International.
[8] Kishore Andra, Chaitali Chakrabarti", A VLSI Architecture for Lifting-Based Forward and Inverse Wavelet Transform" IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 50, NO. 4, APRIL 2002.
[9] Kishore Andra, Chaitali Chakrabarti, Member, IEEE, and Tinku Acharya, Senior Member, IEEE, "A VLSI Architecture for Lifting-Based Forward and Inverse Wavelet Transform", IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 50, NO. 4, APRIL 2002.