Hardware realization of 2-D Discrete Wavelet Transform for DSP applications

Aditya Mandloi, Rajnikant

Abstract— This work presents an efficient VLSI architecture of a high speed, low power 2-D Discrete Wavelet Transform. The proposed architecture reduces the hardware complexity and memory accesses. Our designs were realized using HAAR wavelet transform in VHDL language and MATLAB for optimizing the throughput and area requirements. The system is modeled and synthesized using Xilinx Virtex-II Field Programmable Gate Array (FPGA) device. The simulation results obtained through Mentor graphics Model-Sim software version 6.3.The system computing rate is up to 280 M samples/s, memory uses only 22 kilo byte for 8×8 image size and Maximum combinational path delay is 44.155ns.In this way, the developed design requests less memory area and provide very high-speed processing.

Index Terms— Xilinx FPGA, Discrete Wavelet Transform (DWT), image compression, VHDL, JPEG2000, MATLAB, Haar

1 INTRODUCTION

The revolution in the technology is making computer advanced day by day and so does the use of digital images. Along with this comes the serious issue of storing and transferring the large volume of data representing the images when uncompressed. Multimedia (graphics, audio and video) data requires quite a large storage capacity and transmission bandwidth. Despite of rapid growth in mass storage density, increased speed of the processor and the performance of the digital communication systems, the demand for data storage capacity and data transmission bandwidth keep on increasing the capabilities of on hand technologies [1].

Discrete Wavelet Transform (DWT) has found many applications in digital signal processing, due to the efficient computation and the sufficient properties for non-stationary signal analysis [2]. Nowadays it has become one of the most used techniques for image compression. The image is actually a kind of redundant data i.e. it contains the same information from certain perspective of view. By using data compression techniques, it is possible to remove some of the redundant information contained in images. Image compression minimizes 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 certain amount of disk or memory space [3]-[5].

In this work, we have implemented the compression algorithm using Haar wavelet, in VHDL programming language and in MATLAB. For that we have taken 8X8 matrix in both, with an objective to reduce complexity of understanding of the basic concepts. This new arrival has shown a great promise toward Data/image compression. The latest JPEG2000 standard is also based on this transform coding/algorithm [6]-[8]. In this paper, only Grey-scale image are considered. However, wavelet transform and compression techniques are equally applicable to the color images with three color components. We have to apply this transform to each of this color component independently and have to treat the result as an array of vectored valued wavelet coefficients.

- *Aditya mandloi is currently working as a assistant professor in Electronics and communication department atMedi-Caps Institute of Tecnology and Management, Indore(M.P.),INDIA, E-mail: aditya.mandloi@gmail.com*
- *Rajnikant is currently working as a assistant professor in Electronics and communication department atMedi-Caps Institute of Tecnology and Management, Indore(M.P.),INDIA,E-mail: kantjee@gmail.com*

2 INTERPRETATION OF IMAGE AS DATA

It also reduces the time necessary for images to be sent over the Internet or downloaded from web pages. There are a number of various methods in which image files can be compressed [1]. There are two main common compressed graphic image formats namely Joint Joint Photographic Experts Group (JPEG, usually pronounced as JAY-pehg) and Graphic Interchange Format (GIF) for the use in the Internet. Color image compression can be approximately regarded as compression of multiple grayscale images, which are either compressed entirely one at a time, or are compressed by alternately interleaving 8x8 sample blocks from each in turn[9]-[11].

The paper is organized as follows; Section-III describes the principle of image compression, its later subsections features the properties of discrete wavelet transform and application of wavelet as image compression tool. Section-IV deals with mathematical analysis of image compression using Haar wavelet. Section-V illustrates the RTL schematic while simulation results are presented in section-VI. Section-VII illustrates hardware utilization summary and Section-VIII concludes the paper.

3 PRINCIPLE OF IMAGE COMPRESSION

An ordinary characteristic of most images is the correlation of neighboring pixels and therefore hold redundant information. The foremost task then is to find out less correlated representation of the image. Two elementary components of compression are redundancy and irrelevancy reduction. Redundancy reduction aims at removing duplication from the signal source image. Irrelevancy reduction omits parts of the signal that is not noticed by the signal receiver, namely the Human Visual System (HVS). In general, three types of redundancy can be identified: (a) Spatial Redundancy or correlation between neighboring pixel values, (b) Spectral Redun-

dancy or correlation between different color planes or spectral bands and (c) Temporal Redundancy or correlation between adjacent frames in a sequence of images especially in video applications. Image compression research aims at reducing the number of bits needed to represent an image by removing the spatial and spectral redundancies as much as possible [8].

3.1 Wavelet for Image Compression

Wavelet transform exploits both the spatial and frequency correlation of data by dilations (or contractions) and translations of mother wavelet on the input data. It supports the multi-resolution analysis of data i.e. it can be applied to different scales according to the details required, which allows progressive transmission and zooming of the image without the need of extra storage. Another encouraging feature of wavelet transform is its symmetric nature that is both the forward and the inverse transform has the same complexity, building fast compression and decompression routines [8]-[11]. Its characteristics well suited for image compression include the ability to take into account of Human Visual System's (HVS) characteristics, very good energy compaction capabilities, robustness under transmission, high compression ratio etc. The implementation of wavelet compression scheme is very similar to that of sub-band coding scheme: the signal is decomposed using filter banks. The output of the filter banks is downsampled, quantized, and encoded. The decoder decodes the coded representation, up-samples and recomposes the signal. Wavelet transform divides the information of an image into approximation and detail sub-signals. The approximation subsignal shows the general trend of pixel values and other three detail sub-signals show the vertical, horizontal and diagonal details or changes in the images. If these details are very small (threshold) then they can be set to zero without significantly changing the image. The greater the number of zeros the greater the compression ratio. If the energy retained (amount of information retained by an image after compression and decompression) is 100% then the compression is lossless as the image can be reconstructed exactly. This occurs when the threshold value is set to zero, meaning that the details have not been changed. If any value is changed then energy will be lost and thus lossy compression occurs. As more zeros are obtained, more energy is lost. Therefore, a balance between the two needs to be found out [12].

3.2 2-D Discrete Wavelet Transform

The basic idea of 2-D architecture is similar to 1-D architecture. A 2-D DWT can be seen as a 1-D wavelet transform along the rows and then a 1-D wavelet transform along the columns, as illustrated in Figure 5. The 2-D DWT operates in a straightforward manner by inserting array transposition between the two 1-D DWT. The rows of the array are processed first with only one level of decomposition. This essentially divides the array into two vertical halves, with the first half storing the average coefficients, while the second vertical half stores the detail coefficients. This process is repeated again with the columns, resulting in four sub-bands (see figure 1) within the array defined by filter output.

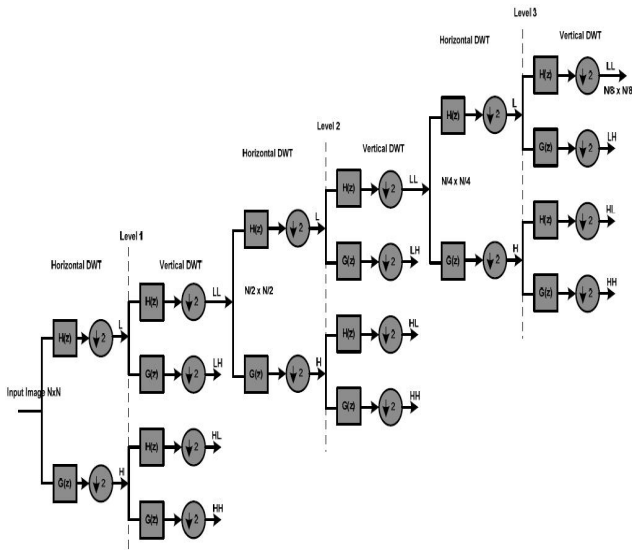

Fig. 1. Three-level 2-D DWT decomposition

The LL sub-band represents an approximation of the original image, the LL1 sub-band can be considered as a 2:1 subsampled (both horizontally and vertically) version of the original image. The other three sub-bands HL1, LH1, and HH1 contain higher frequency detail information (mostly local discontinuities in the edges of the image). This process is repeated for as many levels of decomposition as are desired. The JPEG2000 standard specifies five levels of decomposition [1], although three are usually considered acceptable in hardware.

4 Mathematical Analysis of Compression Using Haar Wavelet

Consider a 8X8 matrix (a part of large image), shown here which is represented by 'A', whose grey-scale representation is shown in figure 2. Here the different contrast levels are shown according to their values in the matrix.

$$A = \begin{bmatrix} 60 & 52 & 59 & 8 & 28 & 53 & 19 & 24 \\ 14 & 28 & 58 & 12 & 59 & 1 & 12 & 55 \\ 38 & 39 & 26 & 12 & 29 & 43 & 12 & 54 \\ 31 & 50 & 57 & 38 & 26 & 24 & 43 & 37 \\ 57 & 58 & 3 & 17 & 54 & 53 & 19 & 31 \\ 48 & 47 & 22 & 12 & 33 & 32 & 34 & 57 \\ 29 & 11 & 52 & 0 & 12 & 45 & 9 & 52 \\ 1 & 25 & 0 & 47 & 43 & 27 & 44 & 41 \end{bmatrix}$$

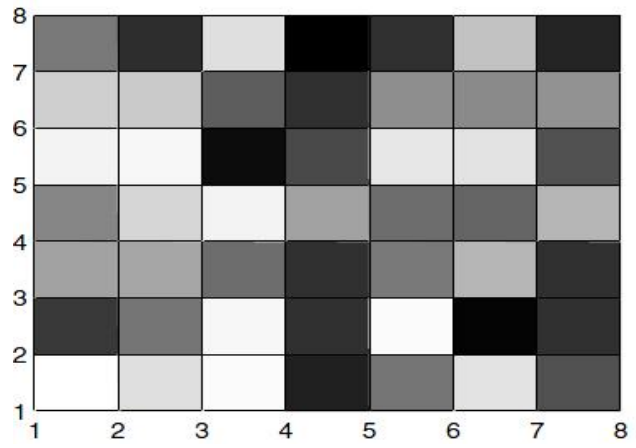

Fig. 2. 8X8 Gray scale image

From our original data, Extract the first row of A, and name it A^*.

$$A^* = [60\ 52\ 59\ 8\ 28\ 53\ 19\ 24] \quad (1)$$

We can think of it as four pairs of numbers. They are 60 and 52, 59 and 8, 28 and 53, and 19 and 24. The second step is to average each pair and put them into the first four entries of a row E that has the same size as A^*. Let B be the first four entries of E, that is,

$$B = [(60+52)/2\ (59+8)/2\ (28+53)/2\ (19+24)/2] \quad (2)$$

$$B = [56\ 33.5\ 40.5\ 21.5] \quad (3)$$

Now, by extracting the first number of each pair from A^*, we form a row C, which is,

$$C = [60\ 59\ 28\ 19] \quad (4)$$

Let D be $D = C - B$, and put D into the last four entries of E, to obtain, $E = [B.D]$ which is,

$$E = [56\ 33.5\ 40.5\ 21.5\ 4\ 25.5\ -12.5\ -2.5] \quad (5)$$

Apply the same method to B, to obtain a row F, which is,

$$F = [44.75\ 31\ 11.25\ 9.5] \quad (6)$$

Replace the first four entries of E by the entries in F, to obtain new row E,

$$E = [44.75\ 31\ 11.25\ 9.5\ 4\ 25.5\ -12.5\ -2.5] \quad (7)$$

Applying the same method above, we have

$$E = [37.875\ 6.875\ 11.25\ 9.5\ 4\ 25.5\ -12.5\ -2.5] \quad (8)$$

The numbers we obtain from the subtraction are called detail coefficients of analysis. Table-I shows result after summarizing each step, we have

TABLE 1

60	52	59	8	28	53	19	24
56	33.5	40.5	21.5	4	25.5	-12.5	-2.5
44.75	31	11.25	9.5	4	25.5	-12.5	-2.5
37.875	6.875	11.25	9.5	4	25.5	-12.5	-2.5

Clearly, the process can be generalized to strings of any length: strings of length 2k require k steps of averaging and differencing. We treat each row of the matrix, to obtain a totally different matrix which is shown below,

$$\begin{bmatrix} 37.88 & 6.88 & 11.25 & 9.50 & 4.00 & 25.50 & -12.50 & -2.50 \\ 29.88 & -1.88 & -7.00 & -1.75 & -7.00 & 23.00 & 29.00 & -21.50 \\ 31.63 & -2.88 & 9.75 & 1.50 & -0.50 & 7.00 & -7.00 & -21.00 \\ 38.25 & 5.75 & -3.50 & -7.5 & -9.50 & 9.50 & 1.00 & 3.00 \\ 36.50 & -2.75 & 23.75 & 14.25 & -0.50 & -7.00 & 0.50 & -6.00 \\ 35.63 & -3.38 & 15.25 & -6.5 & -0.50 & -7.00 & 0.50 & -6.00 \\ 26.25 & -3.25 & -3.00 & -1.00 & 9.00 & 26.00 & -16.5 & -21.5 \\ 28.50 & -10.25 & -5.25 & -3.75 & -12.00 & -23.50 & 8.00 & 1.50 \end{bmatrix}$$

Not only this, we also treat each column of the matrix the same manner after we treated each row. Let us call this semi-final matrix T, whose rows and columns have been treated.

$$T = \begin{bmatrix} 33.06 & -1.47 & 5.16 & 0.59 & -2.00 & 8.19 & 0.38 & -9.94 \\ 1.34 & 3.44 & -2.53 & -0.16 & -1.25 & 8.06 & 2.25 & -0.56 \\ -0.53 & 0.53 & -0.50 & 3.44 & 1.75 & 8.00 & 5.63 & -1.50 \\ 4.34 & 1.84 & 11.81 & 3.13 & 0.75 & -1.13 & 2.38 & 0.63 \\ 4.00 & 4.38 & 9.13 & 5.63 & 5.50 & 1.25 & -20.75 & 9.50 \\ -3.31 & -4.31 & 6.63 & 4.50 & 4.50 & -1.25 & -4.00 & -12.00 \\ 0.44 & 0.31 & 4.25 & 10.38 & -0.50 & -6.00 & 0.00 & 2.75 \\ -1.13 & 3.50 & 1.13 & 1.38 & 10.50 & 24.75 & -12.25 & -11.50 \end{bmatrix}$$

We call this procedure wavelet transform. The point of the wavelet transform is that regions of little variation in the original data manifest themselves as small or zero element in the wavelet transformed version. We see that, comparing the original matrix and the last matrix, the data has become smaller.

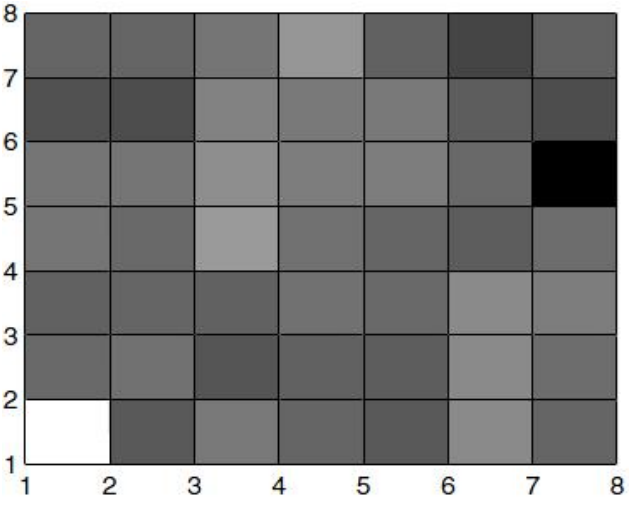

Fig. 3. Compressed gray scale image

Consider the row vector we discussed above

$$A = [60\ 52\ 59\ 8\ 28\ 53\ 19\ 24] \quad (9)$$

and the row vector

$$E = [37.875\ 6.875\ 11.25\ 9.5\ 4\ 25.5\ -12.5\ -2.5] \quad (10)$$

If we just transformed E, and do nothing, we just did half of the work. Think about the numbers which are close to zero. You

make them to be zero. It will make the storing and transferring process way easier. (Think about a 100×100 matrix with all the components are zero) In addition, we don't need a very precise version of a picture, but a approximated picture, which is good enough to let our perception to be stimulated. The first one is the original picture, and the second one has been compressed. The definition of wavelet compression is, fix a nonnegative threshold value, and decree that any detail coefficient in the wavelet transformed data whose magnitude is less than or equal to threshold value will be reset to zero, then rebuild an approximation of the original data using this doctored version of the wavelet transformed data .

The threshold for the second picture is 1. Even though we reset the numbers (they are less than or equal zero) to be zero, which means we lost the original data, the picture is still good. That is because we set an appropriate threshold to zero the lower coefficient out. In above section, we find the average of a pair of numbers, then subtract the average from the first number of that pair. The result is the detail coefficient. That is, the low detail coefficient means that there is not much different in neighboring pixels, such as pure white gradually change to light yellow. Think about a pair of numbers don't have a big difference, then the coefficient will be very small. High differences mean that neighboring pixels jumped from light to dark, or vice verse. The larger the absolute value of the detail coefficient, the higher the difference, and the shaper the changing of the color. For the previous matrix T we mentioned compressed matrix (without thresholding) as

$$T = \begin{bmatrix} 33.06 & -1.47 & 5.16 & 0.59 & -2.00 & 8.19 & 0.38 & -9.94 \\ 1.34 & 3.44 & -2.53 & -0.16 & -1.25 & 8.06 & 2.25 & -0.56 \\ -0.53 & 0.53 & -0.50 & 3.44 & 1.75 & 8.00 & 5.63 & -1.50 \\ 4.34 & 1.84 & 11.81 & 3.13 & 0.75 & -1.13 & 2.38 & 0.63 \\ 4.00 & 4.38 & 9.13 & 5.63 & 5.50 & 1.25 & -20.75 & 9.50 \\ -3.31 & -4.31 & 6.63 & 4.50 & 4.50 & -1.25 & -4.00 & -12.00 \\ 0.44 & 0.31 & 4.25 & 10.38 & -0.50 & -6.00 & 0.00 & 2.75 \\ -1.13 & 3.50 & 1.13 & 1.38 & 10.50 & 24.75 & -12.25 & -11.50 \end{bmatrix}$$

If we set the threshold to be 5, and perform thresholding of matrix T, it get transformed to a new matrix MyT which is shown below,

$$MyT = \begin{bmatrix} 33.06 & 0.00 & 5.16 & 0.00 & 0.00 & 8.19 & 0.00 & -9.94 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 8.06 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 8.06 & 5.63 & 0.00 \\ 0.00 & 0.00 & 11.81 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 9.13 & 5.63 & 5.50 & 0.00 & -20.75 & 9.50 \\ 0.00 & 0.00 & 6.63 & 0.00 & 0.00 & 0.00 & 0.00 & -12.00 \\ 0.00 & 0.00 & 0.00 & 10.38 & 0.00 & -6.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 10.50 & 24.75 & -12.25 & -11.50 \end{bmatrix}$$

A_n is the matrix obtained from the reversing steps.

$$A_n = \begin{bmatrix} 52.84 & 41.84 & 43.03 & -5.47 & 23.56 & 53.81 & 27.00 & 27.88 \\ 23.59 & 34.59 & 61.28 & 12.78 & 53.81 & 1.06 & 19.25 & 58.13 \\ 44.84 & 44.84 & 29.53 & 13.03 & 27.44 & 38.69 & 11.13 & 55.00 \\ 31.59 & 31.59 & 42.78 & 26.28 & 27.44 & 38.69 & 35.13 & 31.00 \\ 50.03 & 50.03 & 10.22 & 21.97 & 43.44 & 43.44 & 12.75 & 32.63 \\ 50.03 & 50.03 & 22.22 & 9.97 & 22.69 & 22.69 & 33.50 & 53.38 \\ 36.91 & 15.91 & 64.59 & 14.84 & 20.81 & 45.31 & 11.63 & 54.50 \\ 15.91 & 36.91 & 15.09 & 64.34 & 45.31 & 20.81 & 34.63 & 31.50 \end{bmatrix}$$

A_n Matrix is not exactly similar to original matrix A. But it is very much similar to it. Now we show grey scale representation of this recovered matrix.

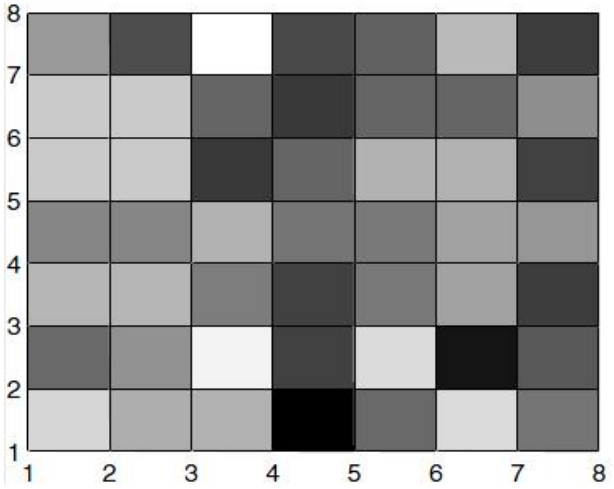

Fig. 4. Recovered matrics(grey-scale)

It is all about MATLAB. But when we talk about VHDL it is difficult to work with floating point. It is obvious that when we perform averaging and differencing on matrix, values should be in floating point. In VHDL we get value in integer not in floating point, example after applying same algorithm of averaging and differencing on matrix A in VHDL we get

$$T = \begin{bmatrix} 33 & 0 & 5 & 0 & 0 & 8 & 0 & 9 \\ 1 & 3 & -2 & 0 & 1 & 8 & 2 & 0 \\ 0 & 0 & 0 & 3 & 1 & 8 & 5 & -1 \\ 4 & 1 & 11 & 3 & 0 & 1 & 2 & 0 \\ 4 & 4 & 9 & 5 & 5 & 1 & 20 & 9 \\ -3 & -4 & 6 & 4 & 4 & 1 & 4 & 1 \\ 0 & 0 & 4 & 10 & 0 & 6 & 0 & 2 \\ -1 & 3 & 1 & 1 & 10 & 24 & 12 & -11 \end{bmatrix}$$

5 RTL SCHEMATIC

Now we see how the things are happening in VHDL. The RTL schematic shows W is input matrix of 8x8 with maximum value of any input in this matrix is 64(2^6) and after applying

algorithm we get output matrix C of 1x64 with max value of 128(2^7) .

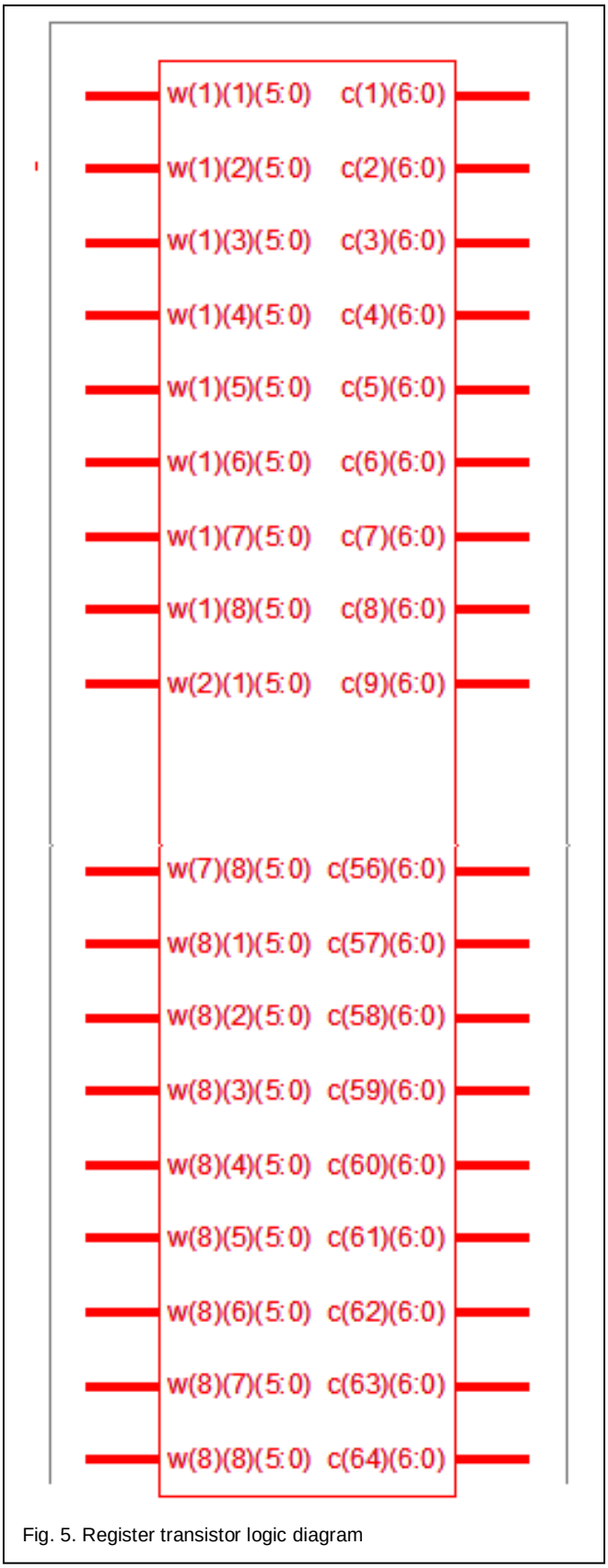

Fig. 5. Register transistor logic diagram

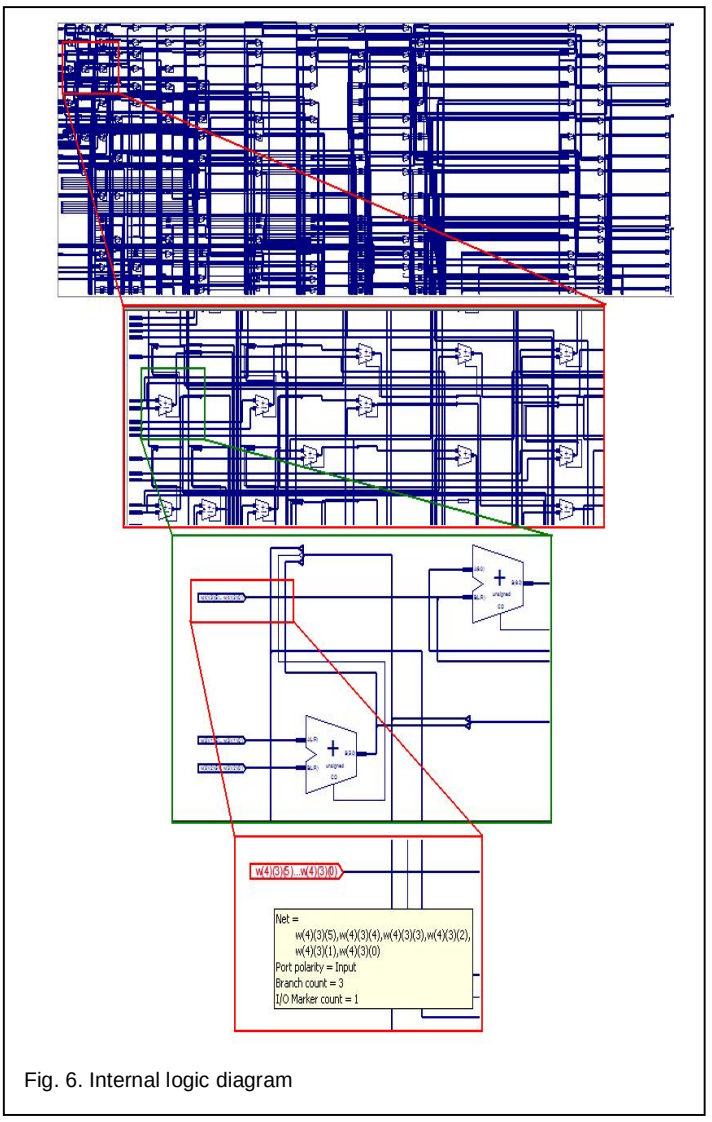

Fig. 6. Internal logic diagram

6 Simulation Results

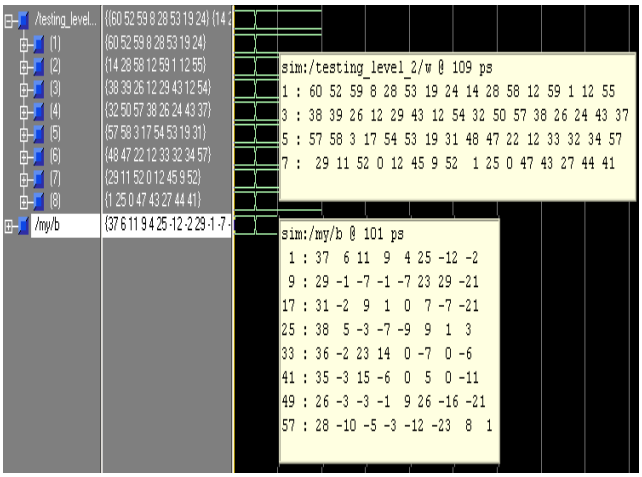

Fig. 7. After applying row operation on 8X8 matrix we get a waveform

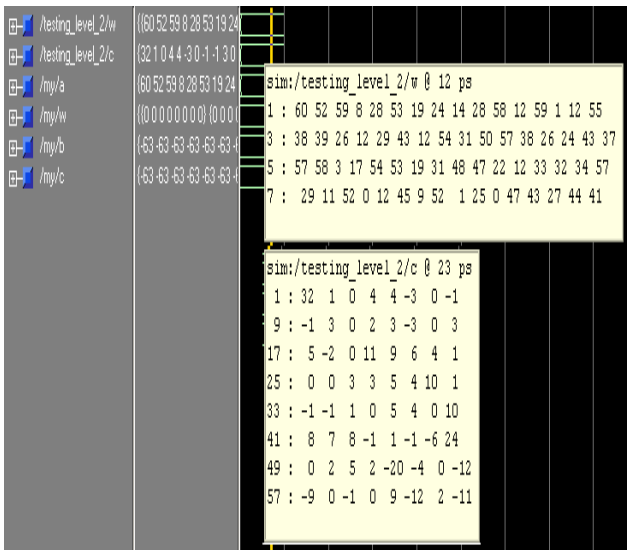

Fig. 8. After applying row and column operation on 8X8 matrix we get a waveform.

7 Device Utilization Summary

TABLE 2

Number of Slices	2322
Number of Slice Flip Flops	535
Number of 4 input LUTs	4108
No of Adder / subtractor	546

Power summary: I(mA) P(mW)

Total estimated power consumption: 348

8 Conclusion

In this paper, we have proposed VLSI architecture for 2-D DWT to meet the requirements of real-time image. The advantages of the proposed architecture are saving on chip area, fast computing time, low power consumption, and low control complexity. The performance of the proposed architecture is compared in terms of number of multipliers, number of adders, storage size, computing time, control complexity and hardware utilization. This hardware is designed to be used as part of a complete high performance and low power JPEG2000 encoder system for digital applications. The proposed architecture has been correctly verified by the VHDL Language. The methodology will, in general, result in different architectures for different target memory technologies, such as ASIC and FPGA implementations.

References

[1] K. K. Parhi, "*VLSI Digital Signal Processing Systems - Design and Implementation*", John Wiley, USA, 1999.

[2] Talukder, K.H. and Harada, K., *A Scheme of Wavelet Based Compression of 2D Image*, Proc. IMECS, Hong Kong, pp. 531-536, June 2006.

[3] "*JPEG2000 Image Coding System*", JPEG 2000 final committee draft version 1.0, March 2000 (available from http://www.jpeg.org/FCD15444-1.htm)

[4] D. Taubman and M. Marchellin, JPEG2000: *Image Compression Fundamentals, Standards and Practice*. Kluwer Academic Publishers, 2002.

[5] Q. P. Huang, R. Z. Zhou, and Z. L. Hong, "*Low memory and low complexity VLSI implementation of JPEG2000 codec*," IEEE Trans. Consum. Electron. vol. 50, no. 2, pp. 638–646, May 2004.

[6] Andra, K., Acharya,T., Chakrabarti, C.: *A High- Performance JPEG2000 Architecture*. IEEE Transactions on Circuits and Systems for Video Technology, vol. 13. No. 3. 209-218, (2003).

[7] B.-F.Wu and C.-F. Lin, "*An efficient architecture for JPEG2000 coprocessor*," IEEE Trans. Consum. Electron. vol. 50, no. 4, pp. 1183–1189, Nov. 2004.

[8] R F Woods, A Cassidy and J Gray, "*VLSI architectures for Field Programmable Gate Arrays: A case study*", IEEE Symposium on FPGAs for Custom Computing Machines (FCCM'96), Napa, USA, April 1996, pp 1-8.

[9] Mohit Kumar Singh, Md. Tauheed Khan, Prabhash Singh, Sanjay Singh Yadav,Sumit Kumar Singh" *image compression techniques: discrete cosine transform vs discrete wavelet transform*" IJESR,Volume-2,Issue-4,Article No-11,pp.261-268, April 2012.

[10] S. Lewis and G. Knowles, "*Image Compression Using the 2-D Wavelet Transform*" IEEE Trans. on Image Processing, Vol. I . NO. 2, PP. 244 - 250, APRIL 1992.

[11] Ronald A. DeVore, Bjorn Jawerth, and Bradley J. Lucier, Member,"*Image Compression Through Wavelet Transform Coding*" IEEE Trans. on Information Theory, Vol. 38. NO. 2, pp. 719-746, MARCH 1992.

[12] M. Vishwanath, R. Owens, and M. J. Irwin, "*VLSI architectures for the discrete wavelet transform*," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process. vol. 42, no. 5, pp. 305–316, May 1995.

[13] H. Liao, M.K. Mandal, and B.F. Cockburn, "*Efficient Architectures for 1-D and 2-D Lifting-Based Wavelet Transform*," IEEE Transactions on Signal Processing, vol. 52, no. 5, 2004, pp. 1315–1326.

[14] K. Andra, C. Chakrabarti, and T. Acharya, "*A VLSI Architecturefor Lifting-Based Forward and Inverse Wavelet Transform*,"IEEE Trans. of Signal Processing, vol. 50, no. 4, 2002, pp. 966–977.

[15] P.-Y. Chen, "*VLSI Implementation for One-Dimensional Multilevel Lifting-Based Wavelet Transform*," IEEE Transactions on Computers, vol. 53, no. 4, 2004.