ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 9, March 2013

Designing of IDWT System in Verilog HDL

Paresh Kamble, Apurva Khanzode

***Abstract*— Wavelet transform has gained importance from its application in pure mathematics to applied sciences. Various studies proved its advantage in image processing and data compression. This paper shows the comparison of software implementation of Inverse Discrete Wavelet Transform in Matlab with that in Verilog HDL. In this approach Haar wavelet was implemented. The system design in Verilog HDL was based on structural approach.**

***Index Terms*—Inverse Discrete Wavelet Transform, image compression, Verilog Implementation.**

I. INTRODUCTION

A majority of internet bandwidth is used for image and video. Large utilization of hand held devices and new communication standards pose a limit on the utilization of usable bandwidth. Among the several compression standards available, the JPEG image compression standard is in wide spread use today. JPEG uses the Discrete Cosine Transform (DCT) as the transform, applied to 8-by-8 blocks of image data. The newer standard JPEG2000 is based on the Wavelet Transform (WT) [7 8]. Wavelet based techniques such as JPEG2000 for image compression has a lot more to offer than conventional methods in terms of compression & decompression ratio. Currently wavelet implementations are still under development lifecycle and in the process of being perfected. Wavelet Transform offers multi-resolution image analysis, which appears to be well matched to the low level characteristic of human vision.

II. INVERSE DISCRETE WAVELET TRANSFORM

A. Wavelet Transform

The Discrete Wavelet Transform structure design needs an analysis filter. One filter of the analysis (wavelet transform) pair is a low pass filter (LPF), while the other is a high pass filter (HPF), Figure 1 [3 4 5]. Each filter has a down-sampler after it, to make the transform efficient. Fig. 2 shows the synthesis (inverse wavelet transform) pair, consisting of an inverse low pass filter (ILPF) and an inverse high pass filter (IHPF), each proceeded by an up-sampler [1 2].

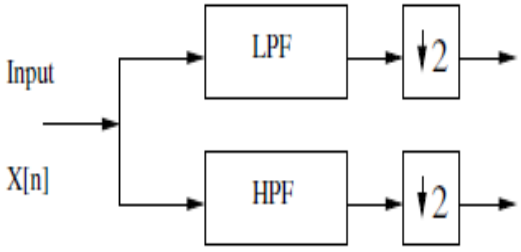

Fig. 1 Analysis Filter

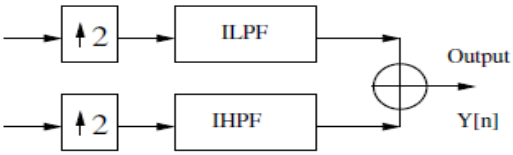

Fig. 2 Synthesis Filter

A low pass filter produces the average signal, while a high pass filter produces the detail signal. This correlation of frequency domain with spatial domain has been used as basis of the paper. Performing sub-band coding using the above filters we get the following structure as shown below. It generates LL, HL, LH & HH component of the original signal.

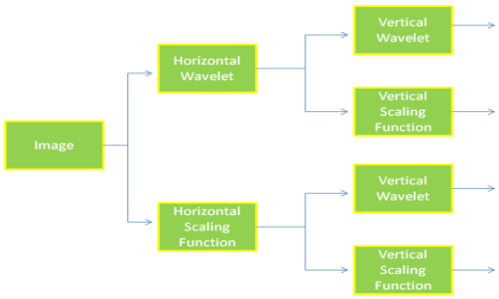

Fig. 3 2D Division of an Image Based On Wavelet Transform

B. Spatial domain work

Taking the above discussions into consideration, we tried to build a system where the spatial domain representations were used instead of frequency domain. It included replacing the LPF and HPF by Average and Difference components respectively.

Fig. 4 Frequency Domain and Spatial Domain Comparison

While performing Wavelet Transform it is needed to apply the 1D transform column wise first and then row wise. The additional time required can be reduced by using the equations devised. The equations for the Discrete Wavelet transform are shown below. It includes the 4 neighboring pixels A, B, C & D. After application of the DWT equations the pixels result into another set of pixels; U, V, W & X which form the DWT image.

Fig. 5 Pixel Locations of Original Image (Left) & DWT Image (Right)

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 9, March 2013

1) U = (A + B + C + D) / 4
2) V = (A - B + C - D) / 2
3) W = (A + B - C - D) / 2
4) X = A - B - C + D **Eq. 1**

In order to derive the IDWT equations from the DWT equations simple arithmetic operations were followed. It resulted into a set of 4 IDWT equations which replicated the way such that A, B, C & D were obtained from U, V, W & X.

Fig. 6 Pixel locations of DWT image (Left) & IDWT image (Right)

1) A = (4*U + 2*V + 2*W + X)/4
2) B = (4*U - 2*V + 2*W - X)/4
3) C = (4*U + 2*V - 2*W - X)/4
4) D = (4*U - 2*V - 2*W + X)/4 **Eq.2**

III. SYSTEM IMPLEMENTATION IN VERILOG HDL

Xilinx ISE was the most preferred choice for implementing the proposed design using Verilog HDL; the reason being its user friendly design and flexibility in working environment. The specialized architecture for the design required two 16K RAM, their respective controllers, the main processing block, a tri state buffer, a multiplexer and an output data read block. The architecture is shown below:

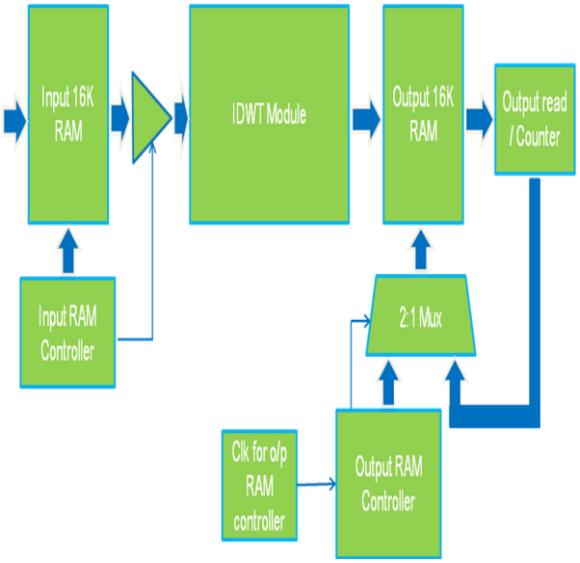

Fig. 7 Proposed Architecture of IDWT System

The above system requires gray scale image as input. It can be given by converting image into text file using Matlab software.

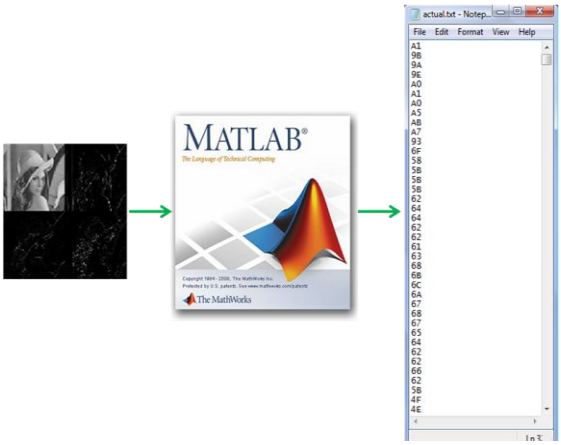

Fig. 8 DWT Image Converted To 'Actual.Txt' File Using Matlab

The first block of the above architecture consists of conversion logic of dwt image file into an 'actual.txt' text file which is done through Matlab. The text file is then saved in input 16K RAM memory.

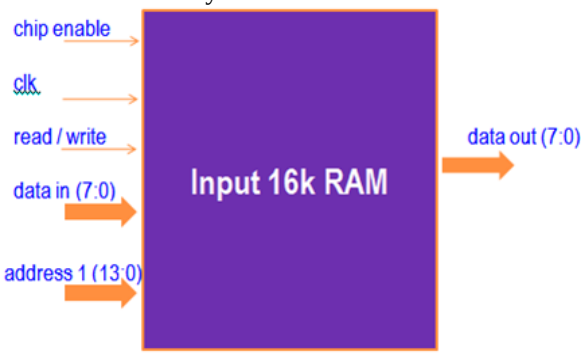

Fig. 9 Block Diagram of Input 16K Memory

The file 'inp_16k_mem.v' reads the file into the block ram generated. The data is stored in the memory and ready to be used for further processing.

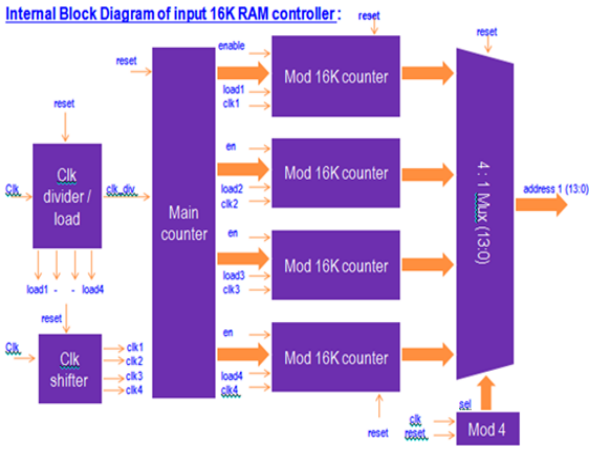

Fig. 10 Internal Block Diagram of Input 16K Memory.

The data in the memory needs to be given to the main processing block but in a specified sequence. The sequence includes an order which goes like 0, 8192, 64, 8256, 1, 8193, 65, 8257,..... This order offers the corresponding component of each of the four quadrants to the main processing block. This sequence is generated by the input memory controller

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 9, March 2013

block whose logic is written in 'main_inp_mem_controller.v' file.

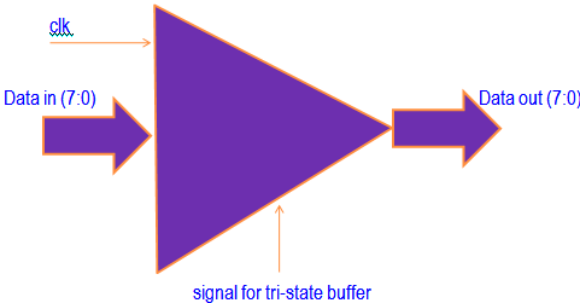

Fig. 11 Block Diagram of Tri State Buffer

The tri state buffer avoids the entry of garbage data in the main IDWT module once all the 16384 data values are transferred to the module. The buffer is activated via a counter incorporated in the input memory controller. The block of main clock to output memory controller plays a major role in synchronizing the system. The IDWT module takes 4 data as input and then gives its first output. It is required for the output memory controller to hold the address generation till the first result is available at the output of IDWT module. Just as the data appears, the address generation starts thus now the data can be stored at certain locations in the output memory. This block provides a delay of 4 clock pulses thereby synchronizing the system.

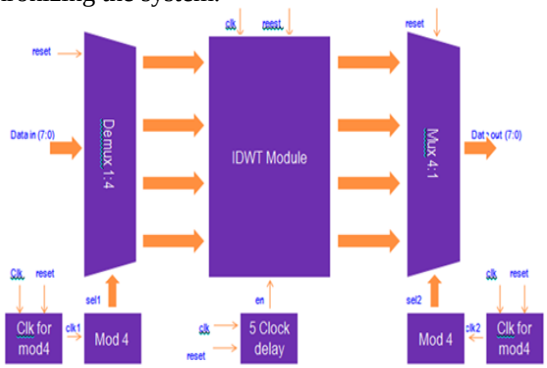

Fig. 12 Block Diagram of Overall IDWT Module

The above block shows the main IDWT block which contains the idwt module, 1:4 demux (8-bit), 4:1 mux (8-bit), a clock delay, 2 mod 4 counters, and clock for the mod-4 counters.

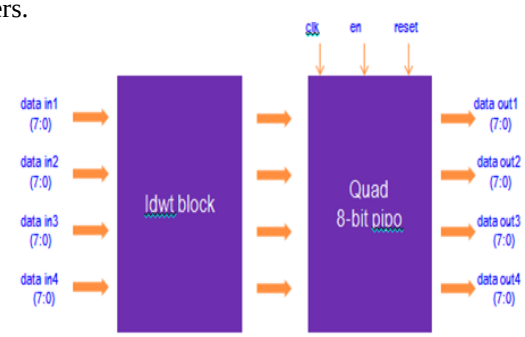

Fig. 13 Internal Block Diagram of IDWT Module

The above block shows the internal construction of the idwt module. This module consists of the idwt block which has the 4 idwt equations. The results of the equations need to be stored somewhere; so the quad 8-bit pipo registers along with latches are used. The de-multiplexer allows one data value to be entered at a time and multiplexer allows single value to leave the idwt module at an instance.

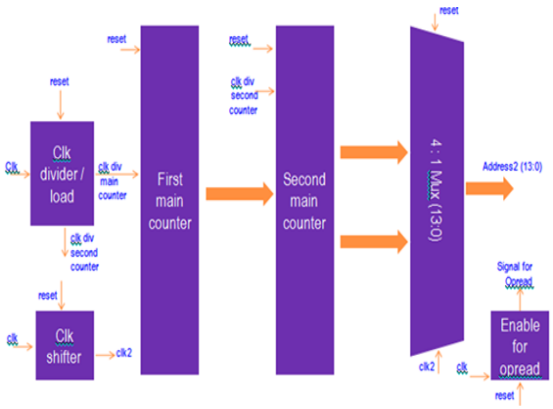

Fig. 14 Internal Block Diagram of Output Memory Controller Module

The above block shows the output memory controller. Here the first main counter generates the address 0, 256, 512, ...whereas the second counter generates the addresses as 0, 1, 127, 128, 2, 3, 129, 130, ...The clock divider provides the clock signal which is divided by 128 to be send to first main counter and clock signal for second main counter as well. The 4:1 multiplexer operates at the rate of the clock shifter pulse. The block also generates an enable signal for output read block so that it can start its memory reading operations. The same signal is given to external 14 bit 2:1 multiplexer.

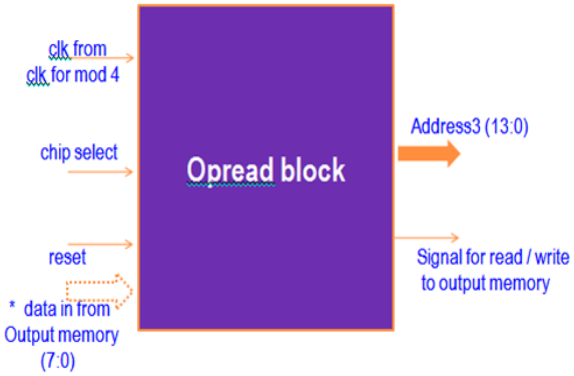

Fig. 15 Block Diagram of Output Read Block

The op read block has a clock signal from output memory controller which enables it. It has a two faceted purpose. In case of simulation use only, the data is stored in the 'rec_data.txt' text file. During synthesis since the file reading syntax are non synthesizable so can't be used. Rather, the block acts only as a serial 16384 counter. It sends serial addresses from 0 to 16383 in a serial fashion thus making the data available at the output of the architecture.

The 14 bit 2:1 multiplexer plays a major role here. It has one input from the output memory controller and other from the opead block. The counter from the output memory controller generates a select line for the multiplexer in order to shift the output from that of the output memory controller generated addresses to the op read block generated serial addresses from 0 to 16383.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 9, March 2013

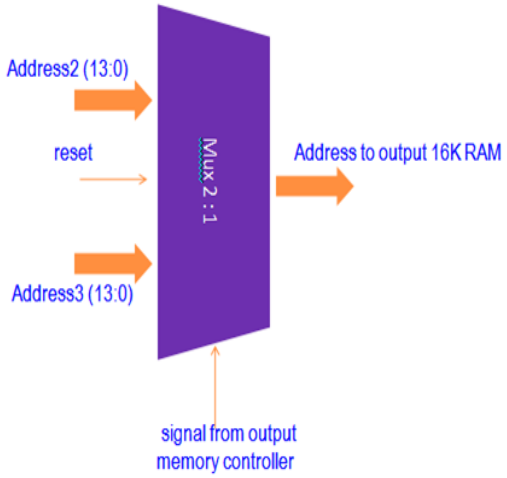

Fig. 16 Block Diagram of 14 Bit 2:1 Multiplexer

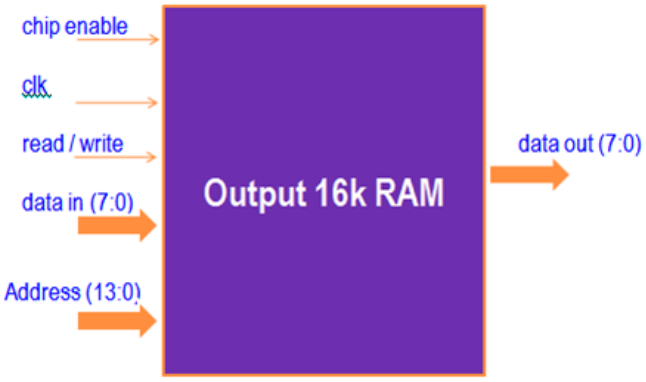

Fig. 17 Block Diagram of Output 16K Memory Block

The output memory is a 16K 8-bit memory. It can read and write 8-bit data if 14 bit addresses are provided to it.

IV. HARDWARE IMPLEMENTATION

Xilinx ISE 9.1i was used as a synthesis tool for the IDWT architecture built. Depending on the hugeness of architecture, FPGA of that comparison was to be employed. It thus included the family of Virtex 5 FPGA with XC5VLX50 device, package as FF324 and speed of -3.

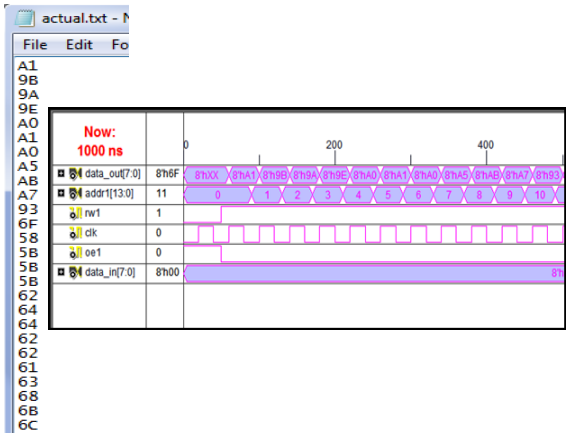

Fig. 18 Simulation Result for Input 16K Memory

It can be seen that the series of 14 bit addresses reads the same data stored in actual.txt which is stored in 16K input RAM. The code for the 16K memory is written in 'inp_16k_mem.v'. The controller, as shown in previous section, consists of clock divider blocks, clock shifter, main counter, 4 mod 16K counters, 13 bit 4:1 multiplexer & a mod 4 counter. The simulation results display the order of addresses to be sent to the input memory so that the corresponding data should be passed on to the IDWT block. The code is written in 'main_inp_mem_controller.v'.

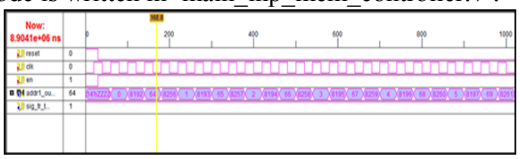

Fig. 19 Simulation Results for Input Memory Controller

Tri state buffer is the block which avoids garbage data to be passed on to the IDWT block once whole data i.e. 16384 pixel values are passed on. The code for this is written in 'tri_state_buffer.v'. The simulation result shows the importance of the 'e' signal i.e. the 'signal for tri state latch' for controlling the input 'a' to output 'f' of the 8-bit buffer.

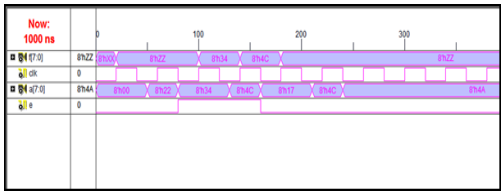

Fig. 20 Simulation Results for Tri state Buffer

While considering the synchronization of the whole architecture it is very important to look after the initialization of the address generation process for the output memory. Since the IDWT block requires 4 clock cycles to process the first resulting data, the output memory controller must wait for 4 clock cycles. This task is performed by the 'main_clk_fr_opt_mem_cntlr.v' program.

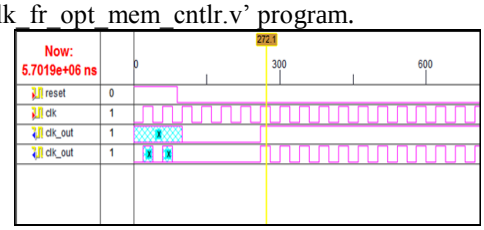

Fig. 21 Simulation Results for Clk for Output Memory Controller

This block is the heart of this architecture since it contains the set of IDWT equations which are going to help reconstruct the original image from the DWT image. Along with the equations, it also consists of latches and 4 8-bit parallel-input-parallel-output registers. They are required to temporarily save the result before being sent at its output port. The code is written in 'mimp_idwt_module.v'.

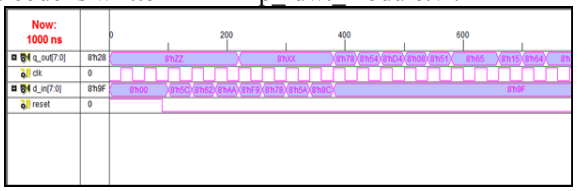

Fig. 22 Simulation Results for main IDWT block

The output memory controller is the output counterpart of the input memory controller. Its function is to provide addresses to the output memory to store the data at

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 9, March 2013

appropriate location so that it can be read out by a serial 13 bit counter. The order happens to be 0, 1, 128, 129, 2, 3, 130, 131, ... The simulation results show the generation of the addresses in the required order.

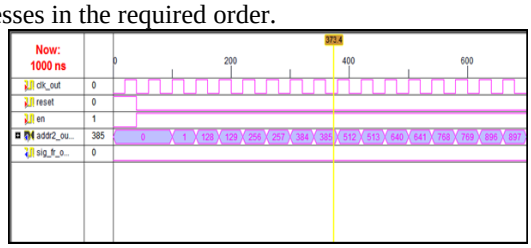

Fig. 23 Simulation Results for Output Memory Controller

This block is used for two purposes. First purpose is during simulation when the data is required to be stored in a text file. The second purpose is during synthesis when the block acts as a counter and is only used to generate data at the output port. The code is written in 'op_read.v'.

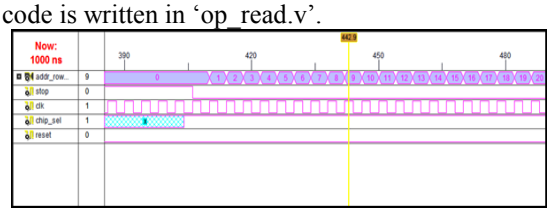

Fig. 24 Simulation Results for Output Read Block

The output memory stores the reconstructed data and makes available as and when needed.

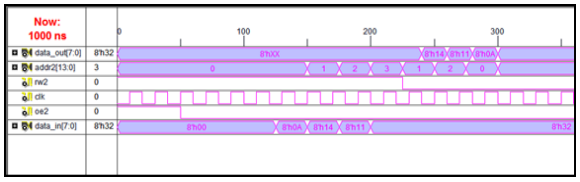

Fig. 25 Simulation Results for Output 16K Memory

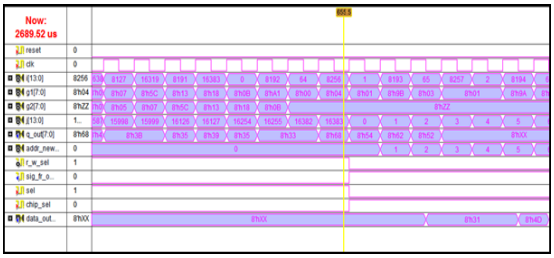

Fig. 26 Final Simulation Results for the Whole IWDT Architecture

The first of the three simulations shows the initial clock cycles where the input memory controller starts generating addresses and input memory reads out the corresponding data into the idwt module. The output memory controller starts after 4 clock cycles thereby synchronizing the architecture. The second simulation shows the end of the writing process of the output memory and the start of its reading process. The third simulation shows the end of output memory reading process either in a text file or making the data available at the output port.

V. MATLAB RESULTS

The 'rec_actual.txt' text file stores the reconstructed image pixel values in decimal. The text file is provided to the matlab code which converts the 1D pixel values into an image.

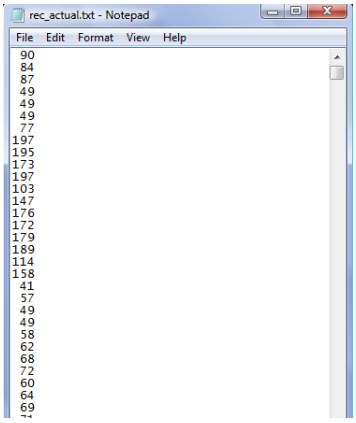

Fig. 27 'Rec_Actual.Txt' File Retrieved From Output Data Of IDWT Architecture

The text file 'rec_actual.txt' is formed into an image as in the figure shown below.

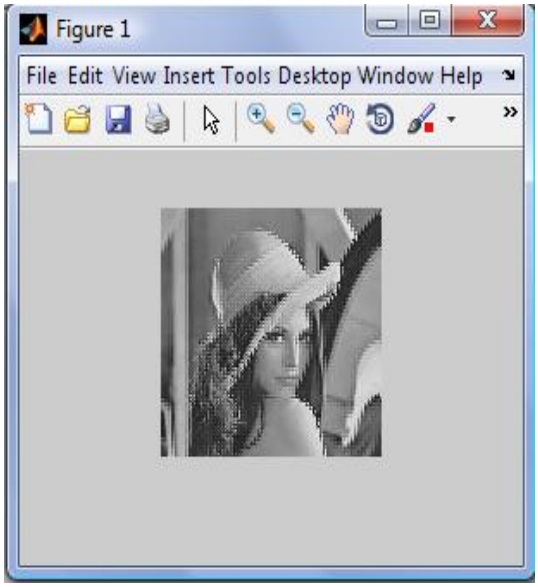

Fig. 28 Image Reconstructed In Matlab from 'Rec_Actual.Txt' File

Below is a table which shows parametric comparison between the above mentioned images.

Table 1 Parametric Comparison Of Various Reconstruction Methods Based On SNR, PSNR, RMSE & MAE.

Parameters	Reconstruction methods			
	Fixed point	Floating point	Standard idwt2()	Verilog HDL
SNR (Signal to Noise Ratio) (in dB)	83.2768	319.9462	309.6960	14.8622
PSNR (Peak Signal to Noise Ratio) (in dB)	88.9194	325.5888	315.3386	20.5048
RMSE (Root Mean Square Error)	0.0046	6.7262e-015	2.1892e-014	12.0773
MAE (Mean Absolute Error)	2.9357e-005	2.9104e-017	1.4246e-016	0.045283

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 2, Issue 9, March 2013

VI. CONCLUSION

This project work is an attempt to build an IDWT architecture using Verilog HDL. Though the reproduction of image appears visually fine but parametric comparison shows a large amount of variations in the SNR, PSNR, and RMSE & MAE values. Such variation is undesirable. It was necessary to find a cause for the discrepancy. While debugging it came to our notice that inclusion of signed modules in the architecture may improve the results by and large. Second issue which came to our notice was the problem being faced while making the code compatible with FPGA. We were able to realize only the IDWT module on Cyclone II 2C35 FPGA. If clock issues are solved then whole system can be realized on the same. Wavelet transform has lot many advantages than any other transform. It just needs to be explored.

REFERENCES

[1] P.C. Wu, C.T. L.G. Chen, "An Efficient Architecture for Two-Dimensional Inverse Discrete Wavelet Transform," IEEE International Symposium on Circuits and Systems, Vol. 2, May 2002, pp. II-312-II-315.

[2] P.C. Wu, L.G. Chen, "An Efficient Architecture for Two-Dimensional Discrete Wavelet Transform," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 4, April 2001, pp. 536-545.

[3] Tze-Yun Sung, "Two Fast Architectures for Two-Dimensional Discrete Wavelet Transform and Its Inversion Using Direct Form",

[4] Tze-Yun Sung & Yaw-Shih Shieh, "An Efficient Architecture for 2-D Inverse Discrete Wavelet Transform with Multiplierless Operation", IEEE ICSS2005 International Conference On Systems & Signals.

[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", World Academy of Science, Engineering and Technology 45 2008, pp. 538-543.

[6] Amara Graps, An Introduction to Wavelets, IEEE Computational Sciences and Engineering, Vol. 2, No 2, Summer 1995, pp 50-61.

[7] Robi Polikar "The Wavelet tutorial" http://engineering.rowan.edu/~polikar/WAVELETS/Wttutorial.html

[8] FAIZAL IQBAL, "WAVELET TRANSFORM BASED IMAGE COMPRESSION ON FPGA".

AUTHOR'S PROFILE

Paresh Kamble is working as Lecturer in Department of Electronics & Telecommunication Engineering at Rajiv Gandhi College of Engineering & Research, Nagpur. He did his MTech from Ramdeobaba College of Engineering & Management, Nagpur.

Apurva Khanzode is working as Lecturer in Department of Electronics & Telecommunication Engineering at Rajiv Gandhi College of Engineering & Research, Nagpur. She did her MTech from Yeshwantrao Chavhan College of Engineering, Nagpur.