World Academy of Science, Engineering and Technology
International Journal of Electronics and Communication Engineering
Vol:2, No:1, 2008

Comparison between Haar and Daubechies Wavelet Transformations on FPGA Technology

Fatma H. Elfouly, Mohamed I. Mahmoud, Moawad I. M. Dessouky, and Salah Deyab

***Abstract*—**Recently, the Field Programmable Gate Array (FPGA) technology offers the potential of designing high performance systems at low cost. The discrete wavelet transform has gained the reputation of being a very effective signal analysis tool for many practical applications. However, due to its computation-intensive nature, current implementation of the transform falls short of meeting real-time processing requirements of most application. The objectives of this paper are implement the Haar and Daubechies wavelets using FPGA technology. In addition, the Bit Error Rate (BER) between the input audio signal and the reconstructed output signal for each wavelet is calculated. From the BER, it is seen that the implementations execute the operation of the wavelet transform correctly and satisfying the perfect reconstruction conditions. The design procedure has been explained and designed using the stat-of-art Electronic Design Automation (EDA) tools for system design on FPGA. Simulation, synthesis and implementation on the FPGA target technology has been carried out.

***Keywords*—**Daubechies wavelet, discrete wavelet transform, Haar wavelet, Xilinx FPGA.

I. INTRODUCTION

THE wavelet transform is an emerging signal processing technique that can be used to represent real-life non-stationary signals with high efficiency [1]. Indeed, the wavelet transform is gaining momentum to become an alternative tool to traditional time-frequency representation techniques such as the discrete Fourier transform and the discrete cosine transform. By virtue of its multi-resolution representation capability, the wavelet transform has been used effectively in vital applications such as transient signal analysis [2], numerical analysis [3], computer vision [4], and image compression [5], among many other audiovisual applications. Wavelet transform is mostly needed to be embedded in consumer electronics, and thus a single chip hardware implementation is more desirable than a multi-chip parallel system implementation.

Several VLSI architectures have been proposed for the implementation of the discrete wavelet transform. The first architecture, presented by Knowles [6], uses many large multiplexes for storing intermediate results. Parhi and Nishitani proposed a folded architecture that has shorter latency [7], however, it requires complex routing and control network.

Chakabarti [8] proposed a systolic architecture, but also it requires many parallel hardware and complex routing. In general, custom VLSI circuits are inherently inflexible and their development is costly and time consuming, and thus they are not an attractive option for implementing the wavelet transform.

FPGA becomes the most applicable microelectronic technology in many recent applications such as communication, mobile telephone, etc. This is due to the relatively high capacity and low cost of the FPGA and also, short design cycle and short time to market when using EDA tools. Since the FPGAs provide a new implementation platform for the discrete wavelet transform. FPGAs maintain the advantages of the custom functionality of VLSI ASIC devices, while avoiding the high development costs and the inability to make design modifications after production [9]. Furthermore, FPGAs inherit design flexibility and adaptability of software implementations.

The appearances of the computer Aided Design (CAD) tools of electronic circuit design have been led to a reduction in the period of the design cycle. The CAD tools are developed to the Electronic Design Automation (EDA) tools that leading to a radial reduction in the design cycle and time to market. In consequence, the EDA tools suppliers transform all different types of design entry into VHDL in order to guarantee the portability of the design. It's necessary to state here that the recent designs can't be achieved without the EDA tools due to the size and complexity of the needed circuits. The organization of this paper is as follows: in section II, the wavelet transform structure is surveyed and concentration is mainly on Daubechies and Haar wavelets. In section III, gives the design of the Daubechies and Haar wavelets using FPGA technology. In section IV, simulation results of the designed circuits are presented. Section V gives synthesis of the Daubechies and Haar wavelets on the chosen FPGA. Finally, the conclusions of this work are summarized followed by a list of references.

M. I. Mahmoud, M. I. M. Dessouky, S. Deyab are with Faculty of Electronic Engineering, Menouf, Egypt. F. H. Elfouly is with HIE, Alshorouk Academy, Cairo, Egypt.[1]

II. Wavelet Processing Algorithm

The discrete wavelet transform (DWT) is simply implemented by a 2- band reconstruction block as shown in Fig. 1. [10]

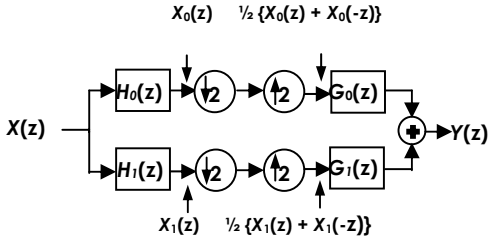

Fig. 1 The 2-band reconstruction block

The input signal $X(z)$ is split by two filters $H_0(z)$ and $H_1(z)$ into a low pass component X_0 and a high pass component X_1 , both of which are decimated (down-sampled) by 2:1. In order to reconstruct the signal, a pair of reconstruction filters $G_0(z)$ and $G_1(z)$ and usually the filters are designed such that output signal $Y(z)$ is identical to the input $X(z)$. This is known as the condition for perfect reconstruction (PR) [10].The art of finding good wavelet lies in the design of the set of filters, to achieve various tradeoffs between spatial and frequency domain characteristics while satisfying the perfect reconstruction condition. Now we are going to discuss the condition for perfect reconstruction. As shown in figure1 the process of decimation and interpolation by 2:1 at the output of $H_0(z)$ and $H_1(z)$ effectively sets all odd samples of these signals to zero.

For the Low pass branch (with LPF $H_0(z)$), this is equivalent to:

1-multiplying $X_0(n)$ by $\frac{1}{2}(1+(-1)^n)$.

2- $X_0(z)$ is converted to $\frac{1}{2}\{X_0(z)+X_0(-z)\}$.

3- Similarly for High pass branch (with HPF $H_1(z)$) $X_1(z)$ is converted to

$\frac{1}{2}\{X_1(z)+X_1(-z)\}$. Then

$$Y(z)=\tfrac{1}{2}\{X_0(z)+X_0(-z)\}G_0(z)+\tfrac{1}{2}\{X_1(z)+X_1(-z)\}G_1(z). \quad (1)$$

$$=\tfrac{1}{2}X(z)\{H_0(z)G_0(z)+H_1(z)G_1(z)\}+\tfrac{1}{2}X(-z)\{H_0(-z)G_0(z)+H_1(-z)G_1(z)\} \quad (2)$$

The first PR condition: requires aliasing cancellation and forces the above term in $X(-z)$ to be zero[10]. Hence

$$H_0(-z)\,G_0(z)+H_1(-z)\,G_1(z)=0 \quad (3)$$

Which can be achieved if:[10]

$$H_1(z)=z^{-k}G_0(-z) \text{ and } G_1(z)=z^{-k}H_0(-z) \quad (4)$$

Where k must be odd (usually k = ±1).

The second PR condition: is that the transfer function form $X(z)$ to $Y(z)$ should be unity

i.e. $H_0(z)\,G_0(z)+H_1(z)G_1(z)=2$ (5)

If we define a product filter

$$P(z)=H_0(z)G_0(z) \quad (6)$$

$$P(-z)=H_1(z)G_1(z) \quad (7)$$

Using equations (5),(6), equation (4) becomes :

$$H_0(z)G_0(z)+H_1(z)G_1(z)=P(z)+P(-z)=2 \quad (8)$$

This needs to be true for all z and, since the odd powers of z in $P(z)$ cancel with those in $P(-z)$, it requires that $P_0 = 1$ and that $P_n = 0$ for all n even and non zero[10].$P(z)$ is the transfer function of low pass branch in Fig.2.1 and $P(-z)$ is that of the high pass branch. For compression applications $P(z)$ should be zero-phase to minimize distortions when the high pass branch is quantized to zero; so to obtain PR it must be of the form:

$$p(z)=\ldots+P_5z^5+P_3z^3+P_1z+1+P_1z^{-1}+P_3z^{-3}+P_5z^{-5}+\ldots \quad (9)$$

We may now define a design method for the PR filters to be as follows:

Choose $P_1,P_3,P_5,\ldots$ in (9) to give a zero-phase low pass product filter $P(z)$ with good characteristics.

Factorize $P(z)$ into $H_0(z)$ and $G_0(z)$, preferably so that the two filters have similar low pass frequency responses.

Calculate $H_1(z)$ and $G_1(z)$ from equations(2.4).

To simplify the tasks of choosing $P(z)$ and factorizing it, based on the zero-phase symmetry we transform $P(z)$ into $P_t(z)$ such that:[10]

$$P(z)=P_t(Z)=1+p_{t,1}Z+p_{t,3}Z^3+p_{t,5}Z^5+\ldots \quad (10)$$

$$\text{Where } Z=\tfrac{1}{2}(z+z^{-1}) \quad (11)$$

To obtain the frequency response, let $z=e^{j\omega T_s}$:

$$\therefore Z=\tfrac{1}{2}(e^{j\omega T_s}+e^{-j\omega T_s})=\cos(\omega T_s) \quad (12)$$

Hence Z is purely real, varying from 1 at $\omega T_s=0$ to -1 at $\omega T_s=\pi$. Substituting this into $P_t(Z)$ gives the frequency response of P.

A. Haar wavelet transform

A Haar wavelet is the simplest type of wavelet [11]. In discrete form, Haar wavelets are related to a mathematical operation called the Haar transform. The Haar transform serves as a prototype for all other wavelet transforms. Like all wavelet transforms, the Haar transform decomposes a discrete signal into two subsignals of half its length. One subsignal is a running average or trend; the other subsignal is a running difference or fluctuation.

The Haar wavelet transform has a number of advantages [11]:

- It is conceptually simple.
- It is fast.
- It is memory efficient, since it can be calculated in place without a temporary Array.
- It is exactly reversible without the edge effects that are a problem with other Wavelet transforms.

The Haar transform also has limitations [12], which can be a problem with for some applications. In generating each of averages for the next level and each set of coefficients, the

Haar transform performs an average and difference on a pair of values. Then the algorithm shifts over by two values and calculates another average and difference on the next pair. The high frequency coefficient spectrum should reflect all high frequency changes. The Haar window is only two elements wide. If a big change takes place from an even value to an odd value, the change will not be reflected in the high frequency coefficients. The audio de-noising by Haar wavelet is not always so effective, because the transform can't compress the energy of the original signal into a few high-energy values lying above the noise threshold. Also if we try to use the Haar wavelet for threshold compression of audio signal, we get poor results. So Haar wavelet transform is not useful in compression and noise removal of audio signal processing. Haar 2-tap wavelet can be used to perform the Haar wavelet transforms. The filters coefficients corresponding to this wavelet type are shown in Table I.

TABLE I
HAAR 2-TAP WAVELET COEFFICIENTS [10]

H0	H1	G0	G1
0.5	1	1	0.5
0.5	-1	1	-0.5

B. Daubechies wavelet transform

The Daubechies wavelet transforms are defined in the same way as the Haar wavelet transform by computing the running averages and differences via scalar products with scaling signals and wavelets the only difference between them consists in how these scaling signals and wavelets are defined[11]. The Daubechies wavelet is more complicated than the Haar wavelet. Daubechies wavelets are continuous; thus, they are more computationally expensive to use than the Haar wavelet,

This wavelet type has balanced frequency responses but non-linear phase responses. Daubechies wavelets use overlapping windows, so the high frequency coefficient spectrum reflects all high frequency changes.

TABLE II
DAUBECHIES 4-TAP WAVELET COEFFICIENTS [10]

H0	H1	G0	G1
0.4830	0.1294	-0.1294	0.4830
0.8365	0.2241	0.2241	-0.8365
0.2241	-0.8365	0.8365	0.2241
-0.1294	0.4830	0.4830	0.1294

Audio de-noising and compression is more sonically pleasing with the Daubechies wavelet than with the Haar wavelet. The Daubechies 4 filter can be used to perform the Daubechies wavelet transforms [12]. The filters coefficients corresponding to this wavelet type are shown in Table II.

III. DESIGN OF HAAR AND DAUBECHIES WAVELET TRANSFORM USING FPGA

Due to the progress in the intensive integration of electronic devices and circuits, the components on the FPGA chip (gates, I/O, buffers, look up tables LUT, registers … etc.) becomes an ultra high number and then it is out of the human manipulation capabilities. Therefore the design on FPGA needs a powerful computer program (software). This program is constructed by a very specialized and expertise software developers that working in cooperation with the original FPGA manufacturer. Nowadays, the electronic design automation EDA tools are suitable for this task. A limited number of EDA factories have a full design flow for EDA of FPGA but Mentor Graphics tools: (FPGA Advantage^{R}) flow is available in our laboratory. This EDA tools has the following components:

1- (HDL - designers^{R}) [13] for data entry that represent the project design.

2- (Modelsim^{R}) [14] for simulation that generate the machine code directly from the complier, which provides faster compilation and some speed up in runtime.

3- (Leonardo - spectrum^{R}) [15] that used for synthesis and implementation of the design on the target technology.

When the design cycle is completed – that is the project elements are processed through all the mentioned above tools – and the results have been accepted; then the implementation of the design is downloaded to the chosen FPGA device.

The architecture of FPGA can handle the implementation of any combinational or sequential logic functions. As well, it accepts the implementation of the basic mathematical operations (addition, subtraction, multiplication and division). Therefore, the FPGA is suitable for the implementation of the discrete wavelet transform.

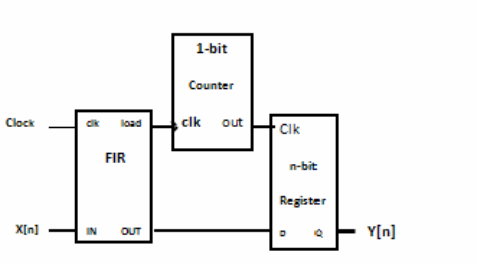

Fig. 2 Implementation of the basic blocks Forward DWT

The design is started with Forward Haar and Daubechies wavelet transforms implementation, the basic building block of the forward Haar or Daubechies discrete wavelet transform filter bank is the decimator which consists of an FIR filter followed by a down-sampling operator [16]. Down-sampling an input sequence x[n] by an integer value of 2, consists of generating an output sequence y[n] according to the relation y[n] = x[2n]. Accordingly, the sequence y[n] has a sampling rate equal to half of that of x[n]. We implemented the decimator as shown in Fig.2.

An active-high output control pin, labeled load, has been implemented in FIR filter structure and connected directly to the CLK input of a 1-bit counter. The input port of the FIR filter is connected to the input samples source, the input port of the FIR filter is connected to the input samples source, whereas the output port is connected to a parallel-load register. The register loads its input bits in parallel upon receiving a high signal on its load input from the 1-bit counter, and blocks its input otherwise. Assuming unsigned 8-bit input samples, the decimator operates as follows. When the load signal is activated, every time the FIR completes a filter operation, it triggers the counter to advance to the next state. If the new state is 1, the parallel-load register is activated, and it stores the data received at its input from the FIR filter. If the new state is 0, the register is disabled, and consequently the FIR output is blocked from entering the register, and ultimately discarded. The above procedure repeats, so that when the state machine has 1 on its output, the FIR data is stored, and when it has a 0 on its output, the FIR data is discarded.

Inverse DWT implementation for Haar and Daubechies wavelets, The basic building block of the inverse Haar or Daubechies discrete wavelet transform filter bank is the interpolator which consists of an FIR filter proceeded by an up-sampling operator [16]. The up-sampler inserts an equidistant zero-valued sample between every two consecutive samples on the input sequence x[n] to develop an output sequence y[n] such that y[n] = x[n/2] for even indices of n, and 0 otherwise. The sampling rate of the output sequence y[n] is thus twice as large as the sampling rate of the original sequence x[n]. We implemented the interpolation filter as shown in Fig.3. The input port of the FIR filter is connected to the output port of the up-sampling block;

whereas the input port of the up-sampling block which is described by a state machine is connected directly to the input samples source. The operation of the state machine depends on the load signal received from FIR filter; it triggers the state machine to advance to the next state. If the load signal is 1, the input sample will appear at the output port of the state machine. Otherwise the output will be zero.

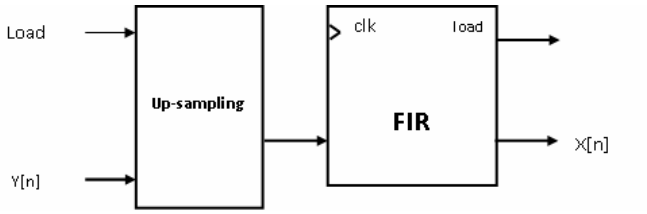

Fig. 3 Implementation of the basic blocks Inverse DWT

A. Data entry:

Design Entry is the process of creating the design and entering it into the development system .This part is explained in chapter 4. By the EDA tools such as (FPGA Advantage^{R}), the designer can create the design description with one or more of the following five methods; the block diagrams, state machine diagrams, flow charts, truth tables and/or VHDL code. The mentioned types of graphical descriptions are automatically converted - by the tool - to a fast and efficient HDL description. The hierarchical design capability of the EDA tools simplifies the design task.

At the top level, of the hierarchy, a global design can be made in the form of system entity block as shown in Fig.4. It appears as a block connected with all the inputs and all outputs of our design with its declaration and number of bits.

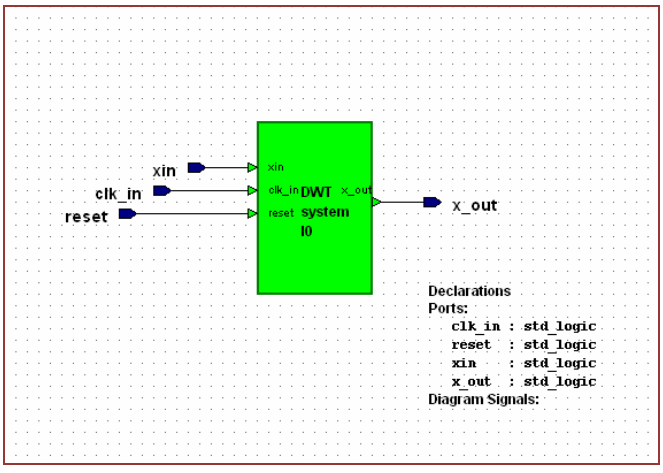

Fig.4 Entity block diagram of the design

The second level represented the system components, in the form of a block diagram, as illustrated in Fig. 5. The details of each component can be entered to the EDA tool using the suitable method of those mentioned above.

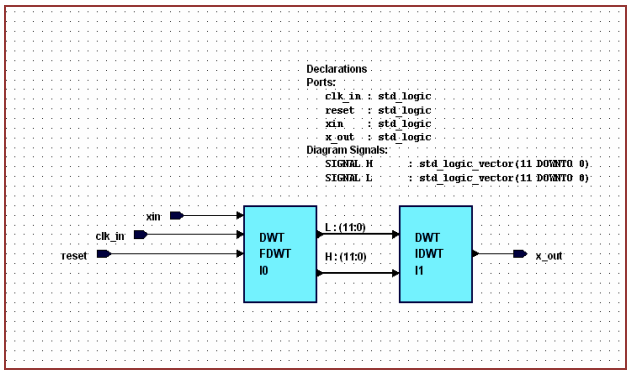

Fig. 5 A global system design representation

IV. Simulation of DWT for FPGA

Before any hardware implementation and testing is performed, all the design modules are tested for correct functionality using the FPGA Advantage[R] MODELSIM functional simulation tool. The MODELSIM tool provided the necessary environment for complete functional simulation of the target hardware. The MODELSIM tool features high speed and target hardware platform adaptability. It also allows for modification and verification of the design simultaneously. For this purpose a test bench facility is available in the EDA tool which is the most suitable method to run a complete simulation for the design. It was described with the VHDL code. The test bench provides access to text file which contains the data of the tested input signal. The input data is generated by MATLAB program. Fig.6. illustrates an example to the VHDL code of the test bench only.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;
Use STD.textio.all;

ENTITY dhbhaarl_tester IS
   PORT(
      clkt : IN     std_logic;
      xin  : OUT    std_logic
   );

-- Declarations

END dhbhaarl_tester ;

-- hds interface_end
ARCHITECTURE sim OF dhbhaarl_tester IS
file inFile : text is in "D:\input.txt";
  signal y: std_logic;
    signal yl: std_logic;
BEGIN
data:process(clkt)
        VARIABLE  inLine      : LINE;
     VARIABLE  dataRead    : real;
        variable    dataReadFromFile :integer;
        variable adc_out:std_logic;
        BEGIN
                  if (clkt'event and clkt = '1') then
                    IF (NOT ENDFILE(inFile)) THEN
                          READLINE(inFile, inLine);
                       READ(inLine, dataRead);
                       datareadfromfile:=integer(dataread);
                          if datareadfromfile=1 then
                                    adc_out:='1';
                             else
                                   adc_out:='0';
                             end if;
                    END IF;
                                end if;
                                y<=adc_out;
         END PROCESS;
        xin <=y;
```

Fig.6 VHDL code of the test bench.

The designed test bench has been run and Our completed design for the wavelet transform achieved a throughput of 19.2 K Samples per second, which means that it is capable of taking in a new sample of the input signal approximately every 52.1 µs, and producing a reconstructed signal at the output at the same rate. First, we need to define the type of signal that we shall be analyzing with the Haar and Daubechies wavelet. Throughout the simulation we shall be working extensively with discrete signal. An example of sample values is the set of data values stored in a computer audio file, such as tada.wav file that quantized and converted to a binary code by the MATLAB program. The reconstructed signal by the Haar and Daubechies wavelet transform look like the original audio signal as shown in Fig.7 and Fig.8. The test bench results of are presented from top to down as: the input audio signal and the reconstructed output. The Bit Error Rate between the input and the reconstructed audio signal gives quantitative evidence for the performance of the wavelet implementations. The BER is simply defined to be

BER = Errors/Total Number of Bits.

For the Daubechies and Haar wavelet transforms, the BER is zero, which prove that the implementations execute the operation of the wavelet transform correctly and verifying the perfect reconstruction conditions.

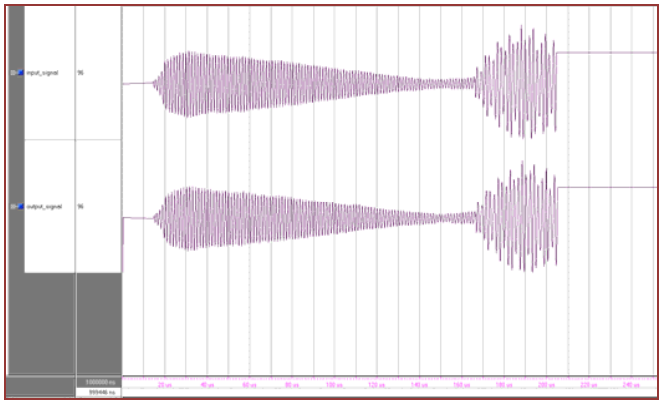

Fig.7 Simulation results of the Daubechies wavelet

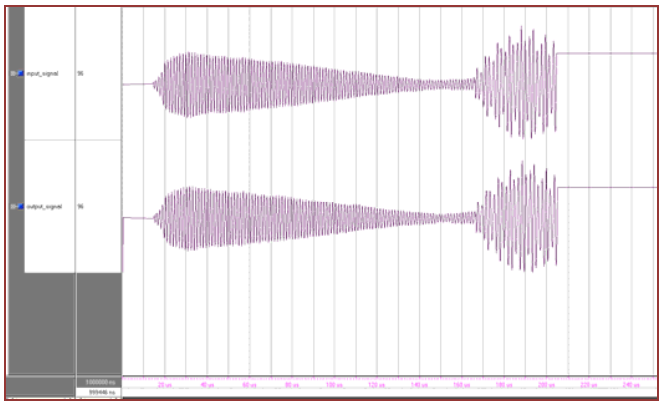

Fig. 8 Simulation results of the Haar wavelet

V. Synthesis of DWT on FPGA

In the case of satisfied simulation results, it is not a guarantee that the real FPGA will also function, the synthesis phase will started. A synthesis tool is used to include the propagation delay of the real scheme using the delay of each element that used in the design including the internal wiring connections. This time of propagation is calculated from any input to any output to detect the path that has the long propagation time or it is normally called the (critical path). The designer must takes into consideration these critical paths, which in some cases block the action of a desired function. Normally the design is an iterative process to compromise among many parameters or criteria. This means that; we may go through the design steps as many as we can to improve and optimize the result of our design.

A. Synthesis results:

We have implemented the design using Xilinx FPGA device, XC4000XL. This device contains 4000k gates and can operate at a maximum clock speed of 1MHz. Fig. 9 and Fig. 10, illustrate the critical path of the designed Daubechies and Haar wavelet transforms respectively.

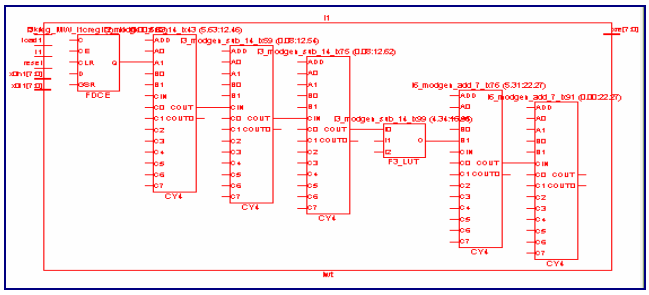

Fig. 9 Critical path of the Daubechies wavelet

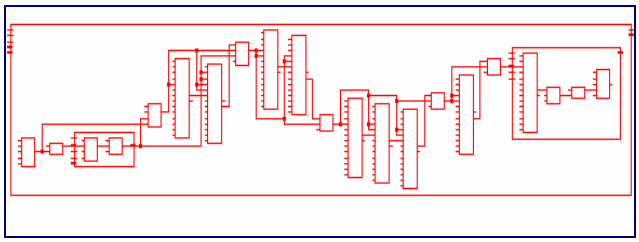

Fig. 10 Critical path of the Daubechies wavelet

VI. Conclusion

This paper proposed an efficient implementation of the Daubechies and Haar wavelet transform and compared between both of them using FPGA technology. The suggested design is tested. The simulation and synthesis results of the suggested design are presented. Based on the results obtained by the simulation and synthesis of the design we can say that the implementation on FPGA gives a fast and reliable realization of wavelet transform and inverse wavelet transform.

References

[1] Ali, M., 2003. Fast Discrete Wavelet Transformation Using FPGAs and Distributed Arithmetic. International Journal of Applied Science and Engineering, 1, 2: 160-171.

[2] Riol, O. and Vetterli, M. 1991 Wavelets and signal processing. IEEE Signal Processing Magazine, 8, 4: 14-38.

[3] Beylkin, G., Coifman, R., and Rokhlin,V. 1992.Wavelets in Numerical Analysis in Wavelets and Their Applications. New York: Jones and Bartlett, 181-210.

[4] Field, D. J. 1999. Wavelets, vision and the statistics of natural scenes. Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences, 357, 1760: 2527-2542.

[5] Antonini, M., Barlaud, M., Mathieu, P., and Daubechies, I.1992.Image coding using wavelet transform.IEEE Transactions on Image Processing, 1, 2: 205-220.

[6] Knowles, G. 1990. VLSI architecture for the discrete wavelet transform. Electron Letters, 26, 15: 1184-1185.

[7] Parhi, K. and Nishitani, T. 1993.VLSI architectures for discrete wavelet transforms. IEEE Transactions on VLSI Systems, 191-202.

[8] Chakabarti, C. and Vishwanath, M. 1995. Efficient realizations of the discrete and continuous wavelet transforms: from single chip implementations to mappings on SIMD array computers. IEEE Transactions on Signal Processing , 43, 3: 759-771.

[9] Seals, R. and Whapshott, G. 1997. Programmable Logic: PLDs and FPGAs. UK: Macmillan.

[10] Nick, K. and Julian, M. 1997. Wavelet Transform in Image Processing. Signal Processing and Prediction I, Eurasip, ICT press, 23-34.

[11] James S. Walker. 1999. A Primer on Wavelets and Scientific Applications.

[12] Applying the Haar Wavelet Transform to Time Series Information

[13] HDL Designer Series User Manual, Software Version 2003.1,9 April 2003, Mentor Graphics Corporation 1996-2003.

[14] Modelsim 5.6 SE Performance Guidelines, Model Technology February 2002, User's Manual, Version 5.6e, Mentor Graphics Corporation 1996-2002.

[15] Lenardo Spectrum User's Manual, Mentor Graphics Corporation 1990-2003.

[16] Vaidyanathan, P. 1993. Multirate Systems and Filter Banks. New Jersey: Prentice Hall.