Comparison of Implementations between Haar Wavelet Transform and FFT on FPGA

Mousa K. Wali, Zahra Qasim

Abstract: ***This research presents a design for both Fast Fourier Transform (FFT) and Fast Discrete wavelet transform (FDWT) by using the simplest algorithms in their respective design. Both the Haar wavelet and Decimation in Frequency implementations were performed. The performance of these two transformations was verified through their implementation using both VHDL and Xilinx System Generator (XSG) applied by MATLAB version R2012a and ISE design suite version 14.6.Finally, the difference between them is shown by comparing the speed of performance and signal processing in addition to the size of the design.***

Keywords: FFT, FDWT, Simulink, VHDL

I. INTRODUCTION

For the first time in 1909, Jean Morlet presented the concepts of the wavelet and its theory**.** Stephane Mallat developed an algorithm to implement (WT) in 1988. WT has played an important role in the past two decades in solving the problems found in physics, engineering, geography, science, earth and etc. compared with Fast Fourier Transform (FFT).Although FFT plays an important role in signal processing applications to convert the signal time domain to the frequency domain and vice versa [1], Fourier Transform has many limitations such as loss of time domain information during its transformation in addition, it should be applied along the entire signal and it does not allow the user to analyze the frequency changes of the input signal during any given time duration [2][3]. However, wavelets are useful for dividing the input signal into different frequency components depending on the characteristics of a particular wavelet function and providing useful information of the input signal simultaneously in both time and frequency domains [4].

The development of VLSI circuits is costly and time-consuming. Therefore these circuits are not a radical option for implementing wavelet transform. The Field Programmable Gate Array (FPGA) technology has a relatively high capacity and low cost as well as the flexibility of the design and adaptation of the software implementations. As a result, this technology is the most applicable technology in many modern applications and is a good option for implementing wavelet transforms [5]

II. THE FAST FOURIER TRANSFORM AND ITS ALGORITHMS

The Fast Fourier Transform is the quick and effective implementation of a discrete Fourier transform [6].

Revised Version Manuscript Received on 04 June, 2018.

Dr. Mousa K.Wali, Assistant Professor, Dean of Electrical Engineering Technical College, Middle Technical University, Baghdad, Iraq. E-mail: musawali@yahoo.com

Zahra Qasim, MSc Student, Department of Computer Engineering, Electrical Engineering Technical College, Middle Technical University, Baghdad, Iraq. E-mail: zahraqqq1992@gmail.com

It is one of the most fundamental blocks in digital signal processing because it is computationally intensive. FFT is some of the fast algorithms that were derived for the DFT computation [7]. It is based on the basic principle of analysis the computation of the DFT of a sequence of length N into sequentially smaller DFT. The FFT generates the same results as the DFT, but reduces the computational complexity from *O* (N^2) to *O* (*N log* (*N*)) [8].Where N represent the number of samples of the signal.

2.1. Decimation in Frequency

The DIF can be derived by decimating the output sequence *X(k)* into sets of smaller and smaller subsequences[7]**.** These results in the equations [1-3]:

$$\boldsymbol{X(K) = \sum_{n=0}^{N/2-1} x(n) W_N^{nk} + \sum_{n=N/2}^{N-1} x(n) W_N^{nk}} \qquad \boldsymbol{Eq(1)}$$

The equation reduced to:

$$\boldsymbol{X(K) = \sum_{n=0}^{N/2-1} \left[x(n) + (-1)^k x\left(n + \frac{N}{2}\right) \right] W_N^{nk}} \qquad \boldsymbol{Eq\ (2)}$$

Equation (2) can be divided into the odd and even index parts:

$$\boldsymbol{X(2K) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) + x\left(n + \frac{N}{2}\right) \right],}$$

$$\mathbf{X(2K+1) = \sum_{n=0}^{\frac{N}{2}-1} \left[x(n) - x\left(n + \frac{N}{2}\right) \right],\ where\ k = 0, 1, \ldots., \frac{N}{2} - 1} \qquad \boldsymbol{E(3)}$$

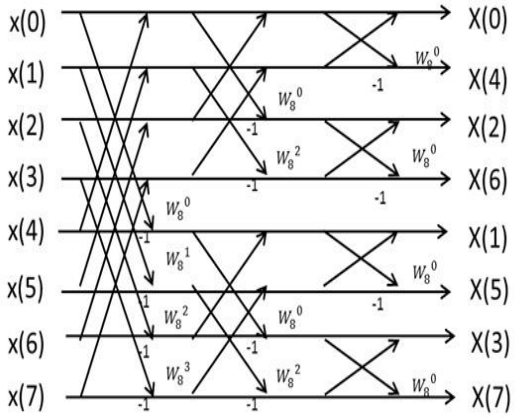


Figure1: The Complete Eight-Point DIF Fast Fourier Transform

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication Pvt. Ltd.

Through the figure1, it can be seen that the inputs are in normal order whereas the output is in bit-reversed order. Thus, this output forces some addition computation to place the output in normal order. It is more important to note that calculations can be performed in-place, thus reducing the amount of memory required to calculate the DFT [9].

WT is a mode to transform an array of N input samples from their decimal values to an array of M output wavelet coefficients as shown in Fig.(2) of our Proposed methodology. The wavelet coefficients indicate the engagement between the desirable wavelet function and the input signal samples [1].

III. WAVELET PROCESSING ALGORITHM

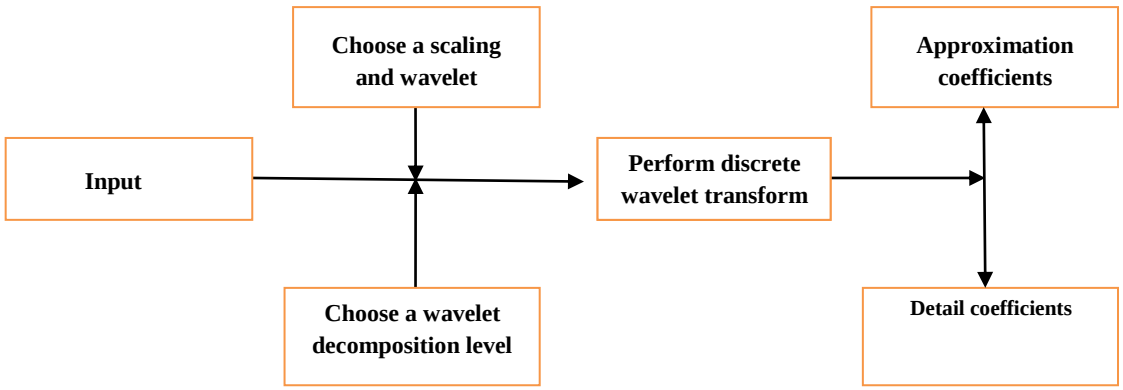


Figure 2: Proposed Methodology for Implementing DWT

Equation (4) represents the general equation for the mother wavelet $\psi(t)$ [1].

$$\psi(t) = \frac{1}{\sqrt{b}}\,\psi\left[\frac{t-a}{b}\right] \qquad a, b \in R, b \neq 0 \qquad Eq(4)$$

Where *a* is the shifting parameter, *b* is the wavelet scaling parameter which locate the size of the wavelet, and *R* is the wavelet space. If the value of *b* is large, the wavelet can localize the low frequency (LF) component of the input signal and vice versa. WT is classified into two main types: the first is the continuous wavelet Transform (CWT) and the second is the Discrete Wavelet Transform (DWT).

In DWT-based implementation, the scaling and shifting parameters such as "*b*" and "*a*" are represented in logarithmic scales as $b= 2^j$ and $a = k2^j$, where "*j*" is number of decomposition levels and k is constant integer. Therefore, the resultant wavelet function of equation (4) for DWT is given in equation (5) [1]:

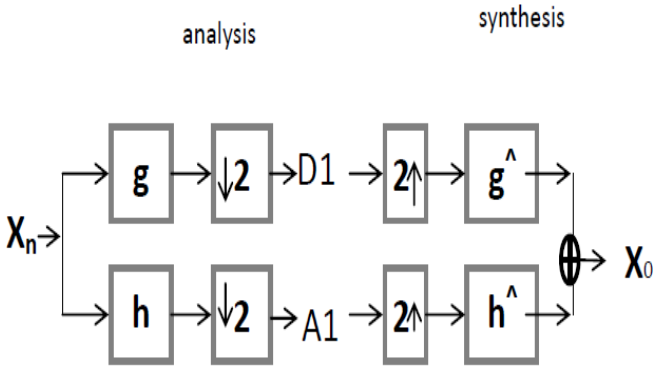
$$\psi_{j,k} = 2^{-j/2}\,\psi(2^{-j}t - k) \qquad Eq(5)$$

In equation (5) the factor 2^{-j} is used to normalize each wavelet function to maintain a constant norm which is independent on decomposition level [1]. Therefore, the approximation and detail coefficient of the input signal "x" processed with DWT is represented as in equation (6) and (7) respectively:

$$X_{a,l}[n] = \sum_{k=0}^{k-1} x_{a-1,}L[2n-k]g[k] \qquad and \qquad Eq\ (6)$$

$$X_{a,H}[n] = \sum_{k=0}^{k-1} x_{a-1}, L[2n-k]h[k], \qquad Eq(7)$$

Where h[k] is the low pass filter coefficients and g[k] is the high pass filter coefficients, which are derived from the scaling function and wavelet function, and they applied as in figure 3. *X1, H[n] - Approximation coefficients; x1, L[n] - Detail coefficients*



Figue3: Filter Bank Implementation of DWT and Inverse DWT

3.1. Haar Wavelet Transform

It is known that the simplest type of wavelet is the haar wavelet. For other Wavelet conversions, Haar transform is an initial model for them, the Haar transform decomposes a discrete signal into two sub signals of half its length [5].

The haar Wavelet cannot be distinguished because it is not continuous, but this technical disadvantage can be a feature for analyzing signals with sudden transformations, such as monitoring the failure of a tool in the machines [5].

The Haar scaling function is this simple unit-width, unit-height pulse function $\Phi(t)$ shown in figure (4a) [10] and it is obvious that $\Phi(2t)$ can be used to construct $\Phi(t)$ as in Eq. (8):

$$\Phi(t) = \Phi(2t) + \Phi(2t-1) \qquad Eq(8)$$

The Haar wavelet function that is associated with the scaling function is $\Psi(t)$shown in Fig. (4b) which construct as in Eq. (9):

$$\Psi(t) = \Phi(2t) + \Phi(2t-1) \qquad Eq(9)$$

(a) **(b)**

Figure4: (a) Haar Scaling Function, (b) Haar wavelet function.

The following two relationships in Eq. (10) and Eq. (11) show that wavelet and scaling coefficients on scale (j) concerned to the scaling coefficients at scale (j+1):

$$A_j(k) = \sum_m h(m-2k)\, A_{j+1}(m) \qquad Eq(10)$$

$$B_j(k) = \sum_m g(m-2k)\, B_{j+1}(m) \qquad Eq(11)$$

The filters coefficients corresponding to this wavelet type are shown in Table 1. H0 and H1 are the input decomposition filters and G0 and G1 are the output reconstruction filters.

Table 1. Haar 2-tap Wavelet Coefficients |

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

IV. FPGA IMPLEMENTATION OF FFT AND FDWT VIA SYS GEN

The presenting of field programmable gate arrays has made it appropriate to provide hardware for application of specific calculation design. The changes in designs in FPGA's can be achieved at a few hours. Both Fast Fourier Transform and Fast Discrete wavelet transform proposed in this paper is designed using MATLAB and synthesized using Xilinx Project Navigator Xilinx ISE14.6a. The summary of the device is explained in the table 2.

Table2. Summary of FPGA Features

Device Family	Spartan 3A
Device	XC3Sd3400a
Package	Fg676
Speed Grade	-4

4.1. Serial Radix-2 DIF FFT

Serial Radix-2 DIF FFT means that the samples enter the system to be processed sequentially and not in parallel. Figure (5) shows the internal execution mechanism of the Serial Radix-2 FFT

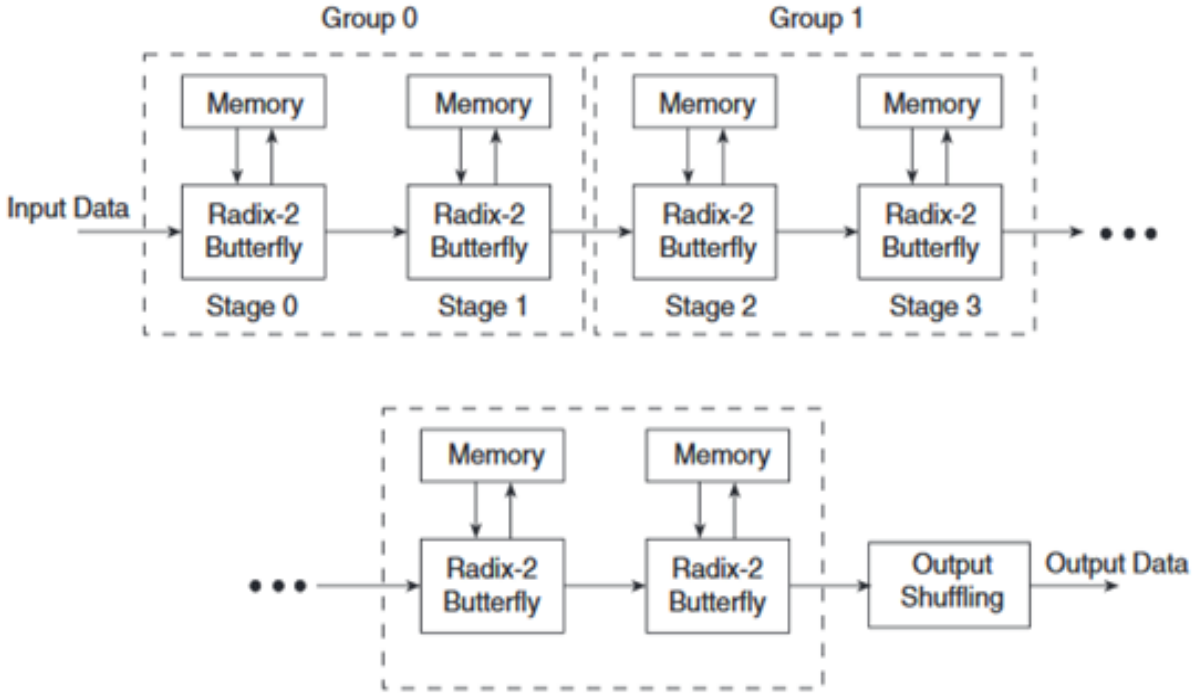


Figure5: Internal Execution Mechanism of the Serial Radix-2 FFT

Figure (6) shows16 point DIT, Radix-2 FFT has been designed using Simulink in MATLAB along with SG.

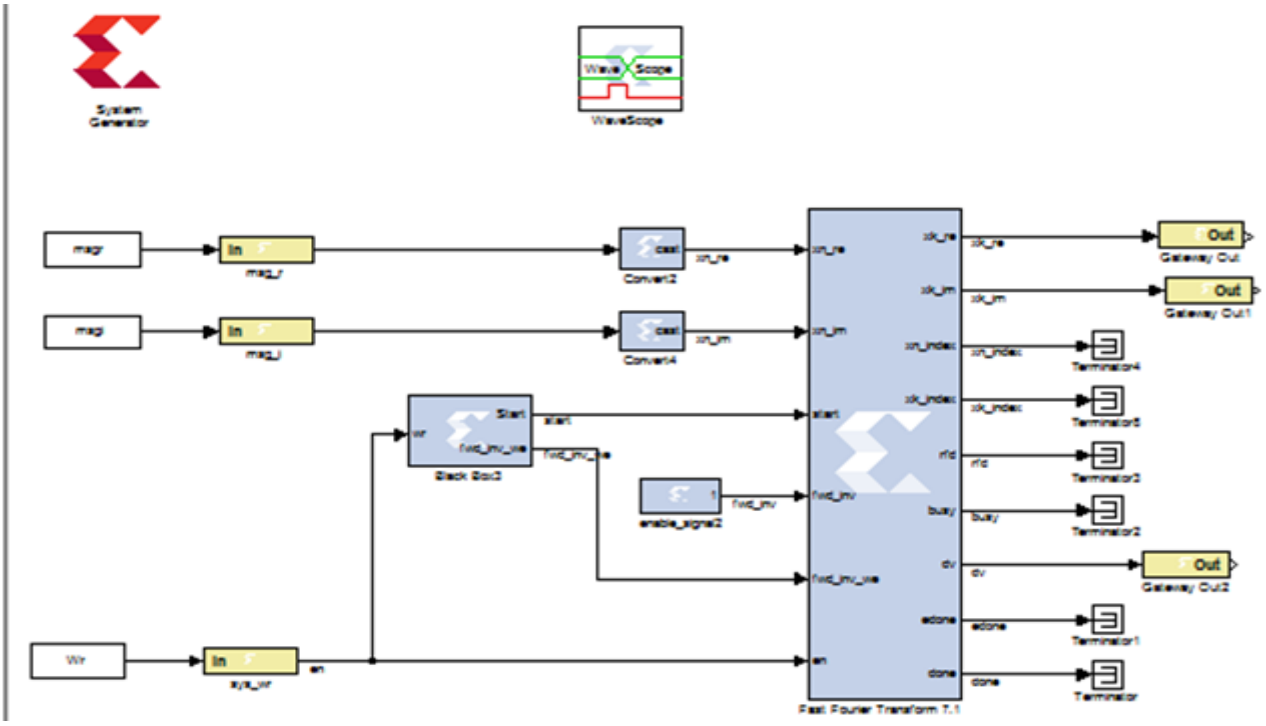


Figure6: Implementation of serial FFT in MATLAB using Simulink.

4.2. Haar Wavelet Implementation

The implementation of Haar wavelet by the MATLAB Simulink (Version R2012a) is shown in Figure (7). Each block of such model will be constructed in the SG (Version 14.6).

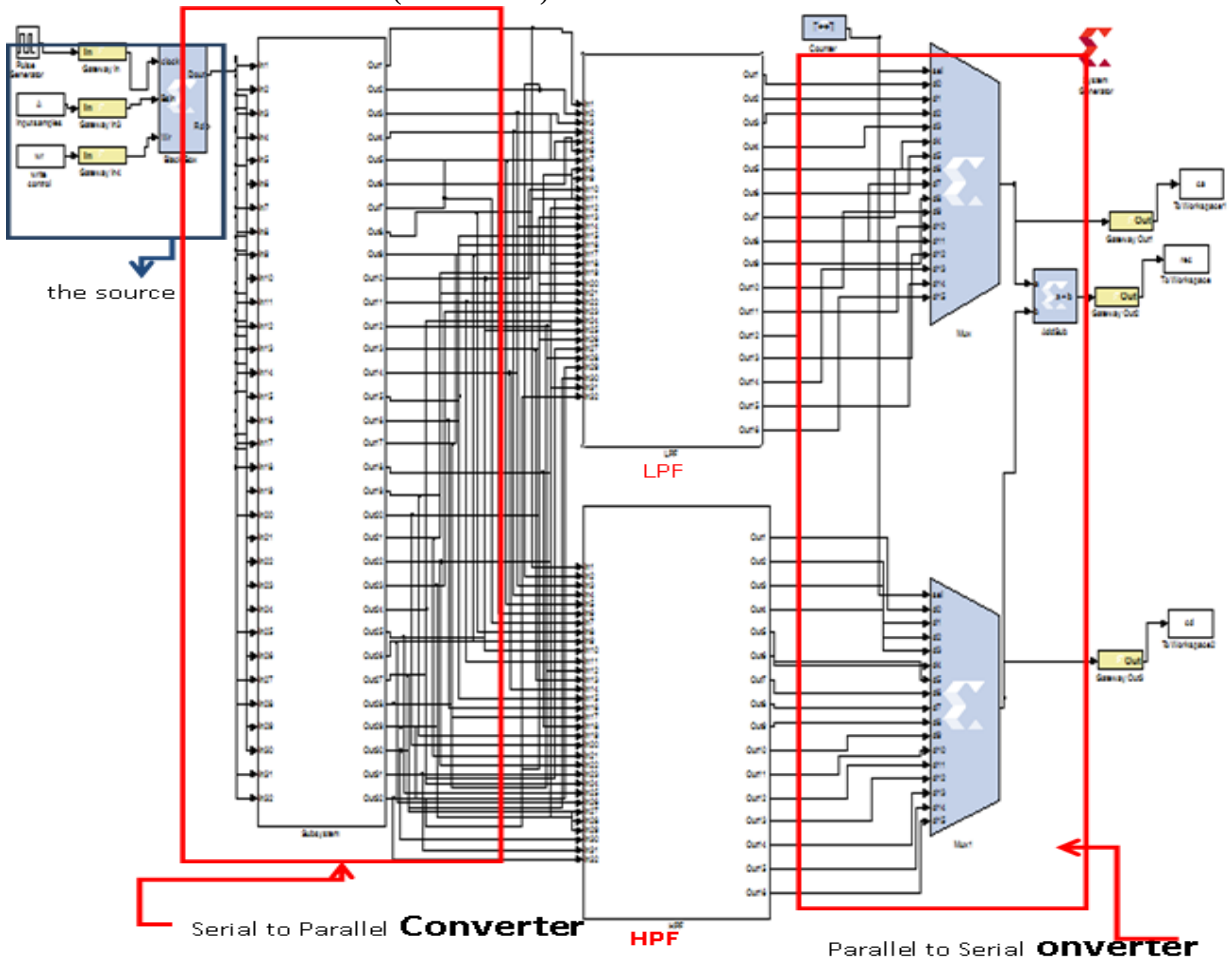


Figure7: Design of Haar wavelet block in SG design

The blocks shown in Figure (8) represent the source for generating and importing the data.

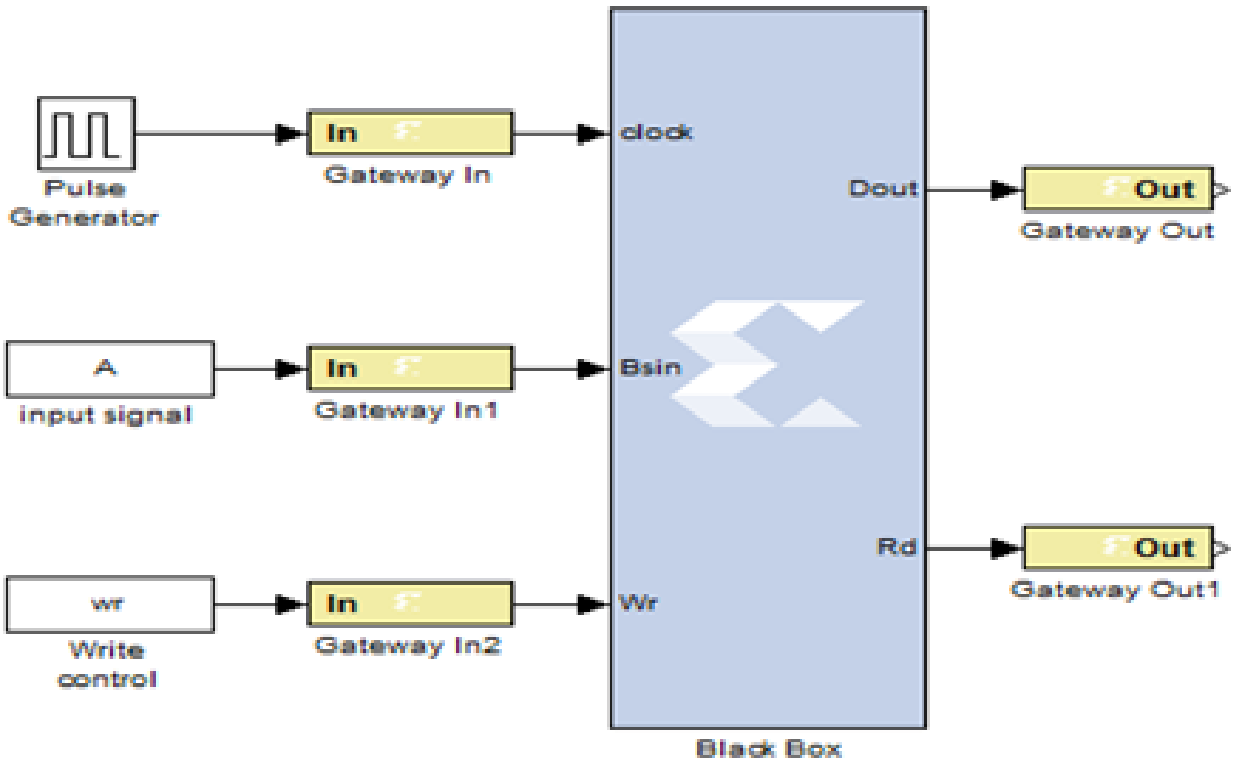


Figure 8: Source for Generating and Importing the Data Blocks in SG Design.

The data is generated by Matlab code and then stored and sent as a block of serial data via the VHDL code.

After the data are serialized from the black box using the VHDL, and by using the slices blocks data it converted from the serial to the parallel which shown as in Figure (9).

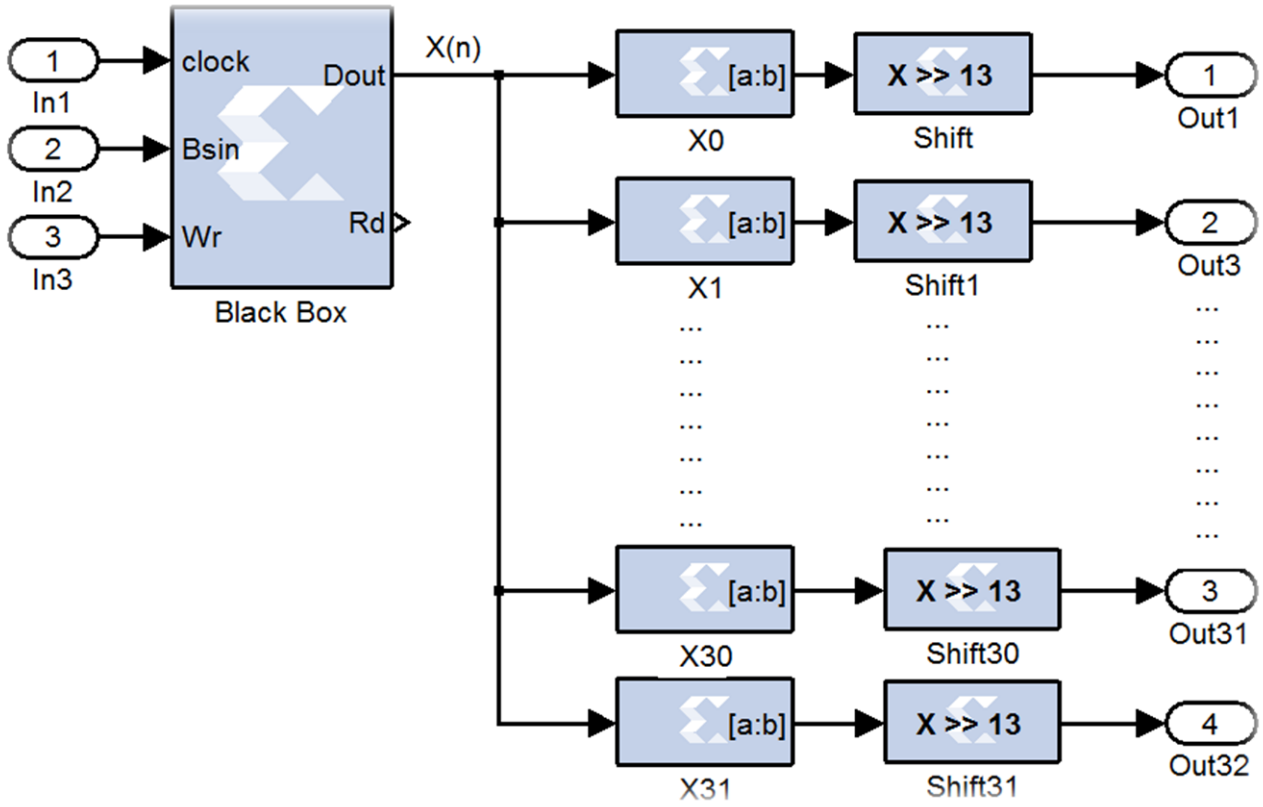


Figure 9: Serial to Parallel Converter Design in SG

After entering data and converting it in parallel, the next step is to apply Haar wavelet to these data. This is done using the block shown in Figure (10).

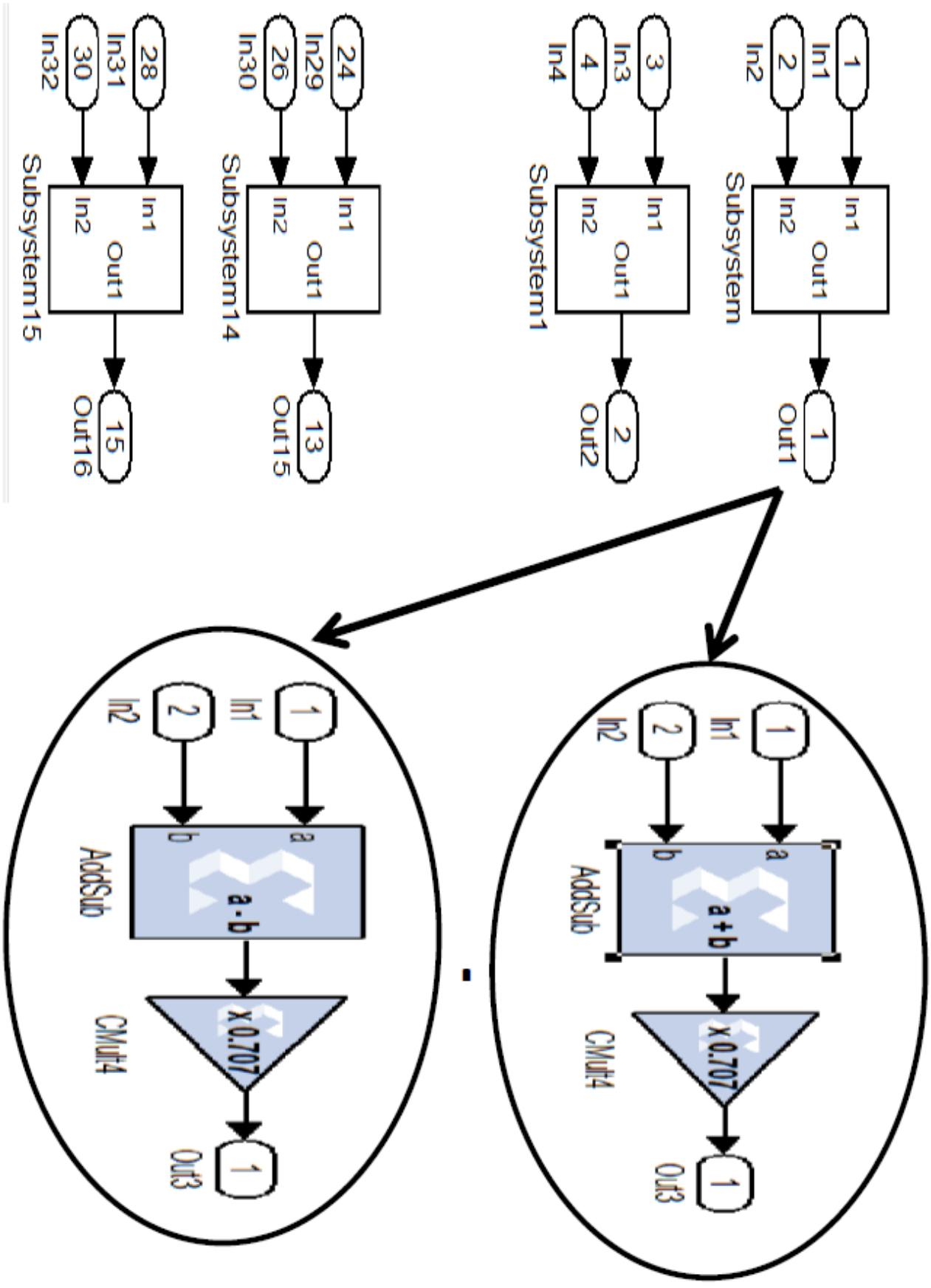


Figure 6: (a) subsystems inside LPF&HPF Haar blocks. (b) LPF Operations. (c) HPF Operations.

Finally, after the data is processed by a Haar wavelet, the results are then converted from parallel to serial using two multiplexes, one for the low filter and one for the high filter,Figure (11) shows how one of these multiplexes is designed.

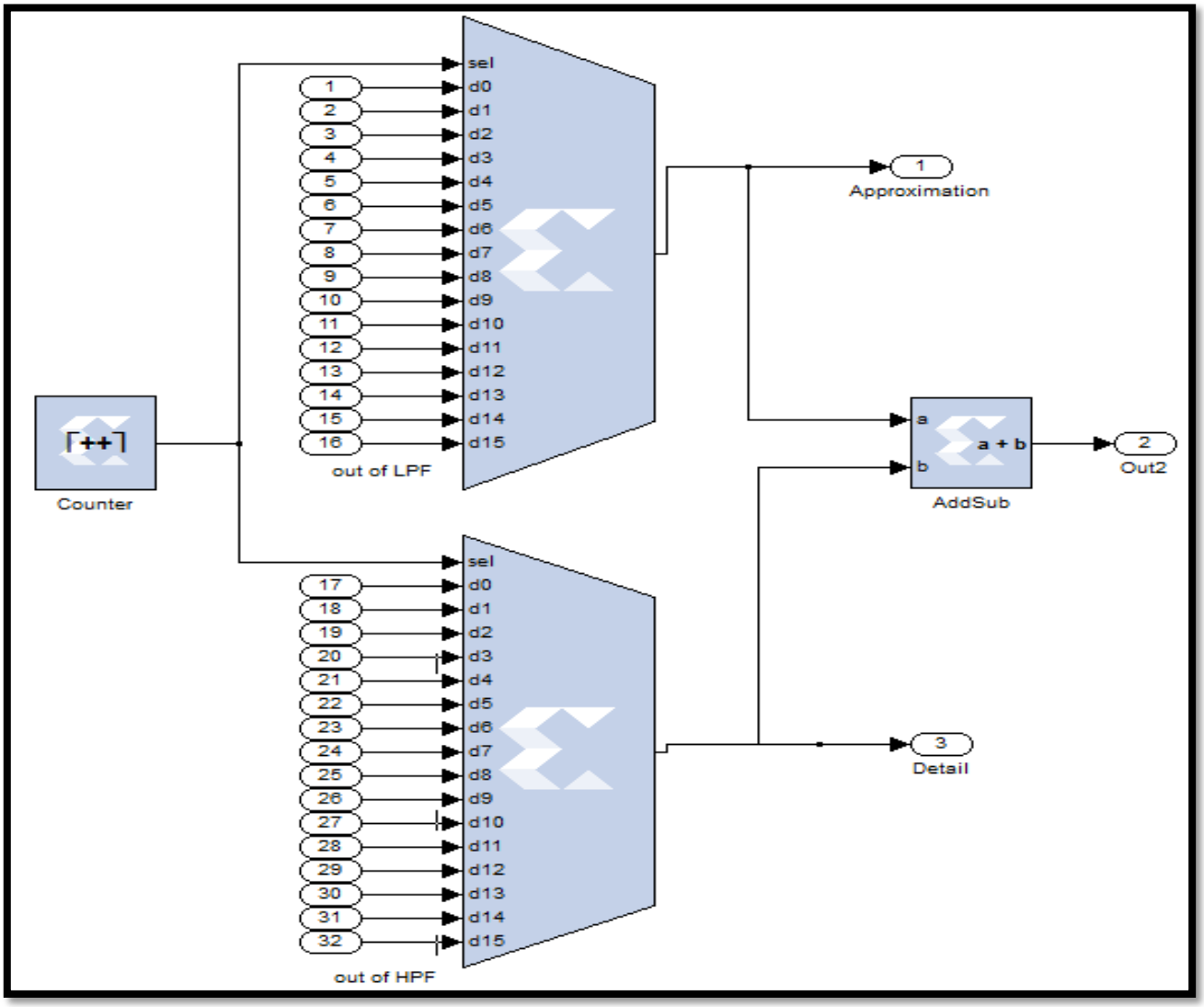


Figure 11: The Parallel to Serial Converter in SG Design

V. RESULTS

Figure (6, 7) shows the full SG designs. All parts were executed using the SG block which was explained in detail previously. Both FFT and FDWT are designed using MATLAB R2012a and synthesized using Xilinx Project Navigator Xilinx ISE 14.6. To verify the reliability of the algorithms implemented, one type of information source was used. Figure (12) shows the input signal is generated by Matlab code and then stored to be send as a block of serial data via the VHDL code.

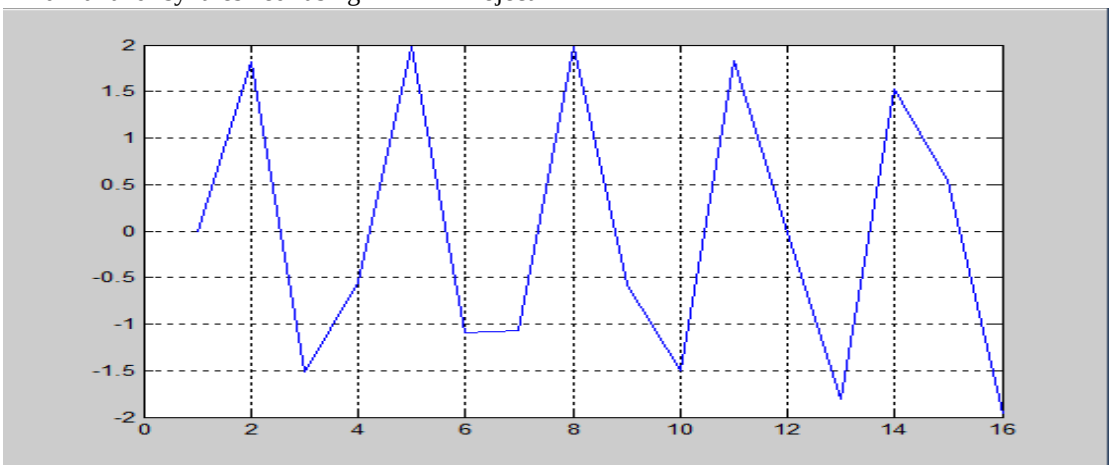


Figure12: Wave Data Plotted in MATLAB

The results of both the FFT and the Approximation and detail coefficients of Haar wavelet transform are shown in Fig. 13 and 14, respectively.

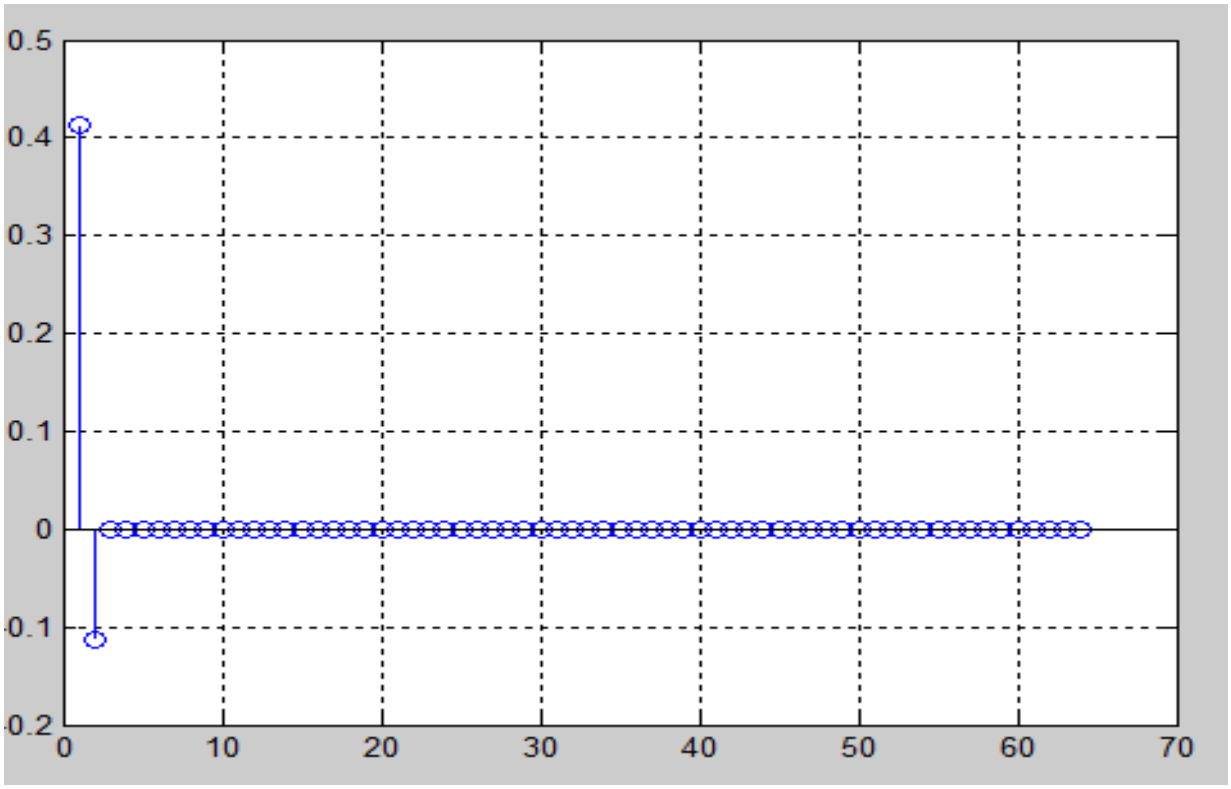


Figure 13: Simulation Results of the FFT.

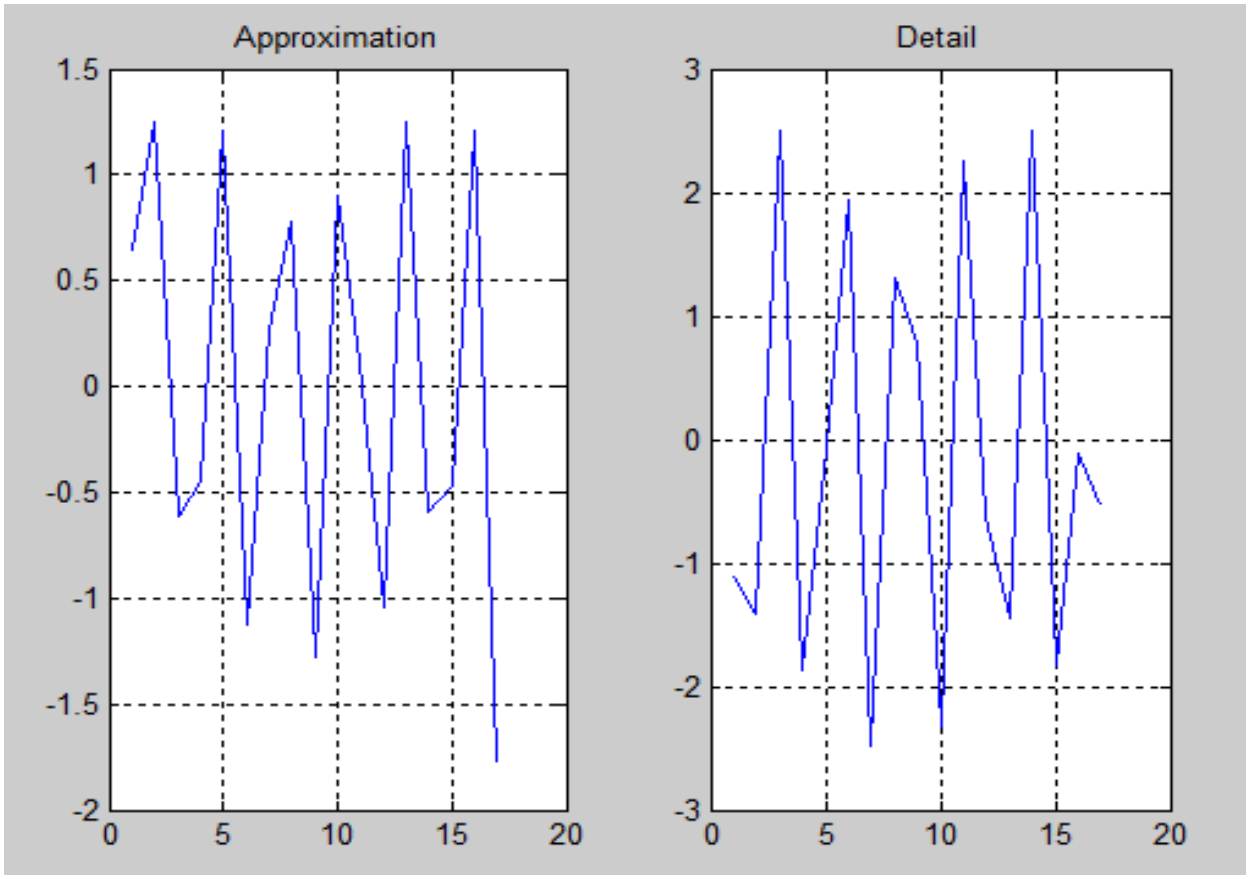


Figure 14: Simulation results of the Haar wavelet.

The estimated resources required for the implementation of both FFT and Haar wavelet on XILINX Spartan 3A XC3Sd3400a are presented in Table 1 and 2, respectively.

Table1: Hardware Resources of 32 Point FFT

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	1,709	47,744	3%
Number of 4 input LUTs	1,638	47,744	3%
Number of occupied Slices	1,224	23,872	5%
Number of Slices containing only related logic	1,224	1,224	100%
Number of Slices containing unrelated logic	0	1,224	0%
Total Number of 4 input LUTs	1,739	47,744	3%
Number used as logic	1,094		
Number used as a route-thru	101		
Number used as Shift registers	544		
Number of bonded IOBs	134	469	28%
Number of BUFGMUXs	2	24	8%
Number of DSP48As	10	126	7%
Number of RAMB16BWERs	2	126	1%
Average Fanout of Non-Clock Nets	1.80		

Table 2: Hardware Resources of 32 point Haar Wavelet

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	1,034	47,744	2%
Number of 4 input LUTs	3,391	47,744	7%
Number of occupied Slices	2,193	23,872	9%
Number of Slices containing only related logic	2,193	2,193	100%
Number of Slices containing unrelated logic	0	2,193	0%
Total Number of 4 input LUTs	3,746	47,744	7%
Number used as logic	3,391		
Number used as a route-thru	355		
Number of bonded IOBs	67	469	14%
Number of BUFGMUXs	2	24	8%
Average Fanout of Non-Clock Nets	2.60		

A comparison of the results of the implementation of Haar and FFT architectures are summarized in Table 3. In terms of area, DIF FFT (32 point) architecture requires less area (6.0%) with 1574 mW power consumption at 340 MHz speed. While, HAAR architecture requires 10% additional area and consumes less power 1282 mw.

In relation to the speed (minimum delay), Haar wavelet is approximately executed with same speed as DIF FFT. The important goal is gating very low power (292mw) compared with (1574mw) of DIF FFT.

Table 3: Comparison of The Proposed Designs Of Both HAAR Wt and FFT

	Number of FFs	**Number of Slice**	**Number of LUTs**	**Number of IOBs**	**Power Consumption (mW)**	**Min. delay**
Haar WT	1034	2193	3746	67	292	3.370ns(290Hz)
DIF FFT	1019	1416	5346	67	1574	2.941ns(340MHz)

VI. CONCLUSION

The comparison between DIF FFT and Haar wavelet transform during implementation of both of them has been presented in this paper using the FPGA technique. The FPGA implementation of FFT and Haar wavelet transform via system generator were presented. The result shows that wavelet transform executed with very less power than FFT with approximately 5 times even it takes more architecture Area but this is considered no problem as far as it is implemented by FPGA.

REFERENCES

1. Mousa Wali K., Murugappan M., mathematical implementation of hybrid fast fourier transform and discrete wavelet transform for developing graphical user interface using visual basic for signal processing applications, Journal of Mechanics in Medicine and Biology, Volume: 12, August 2012.
2. Sifuzzaman M, Islam MR, Application of wavelet transform and its advantages compared to Fourier transform, J Phys Sci 13:121134, 2009.
3. Mallat S, A Wavelet Tour of Signal Processing, Academic Press, New York, 1999.
4. Valens C, A Really Friendly Guide to Wavelets, 1999.
5. Mohamed Mahmoud I., Moawad Dessouky I. M., Comparison between Haar and Daubechies Wavelet Transformions on FPGA Technology, Proceedings of world academy of science, ISSN: 1307-6884, Volume 20, April 2007.
6. Rao K.R., Kim D.N., Fast Fourier Transform: Algorithms and Applications, Springer Dordrecht Heidelberg London New York, ISSN: 1860-4862, 2010.
7. Dr. André Kokkeler, Implementation of Fixed-Point FFT (512, 1K, 2K, 4K) on FPGA, University of Twente, the Netherlands, November, 18, 2010.
8. Nuo Li, ASIC FFT Processor for MB-OFDM UWB System, Faculty of Electrical Engineering, Mathematics and Computer Science Delft University of Technology , 2008.
9. Francis Edward Nicklous,The Design Simulation And Synthesis Of A Pipelined Vhdl Floating-Point Radix-4 Fast Fourier Transform Data Path, Temple University, August, 2010.