IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 8, Issue 4, Ver. I (Jul. - Aug. 2018), PP 23-47
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Comparative Study Of Fpga Implementation Of Parallel 1-D Fft Algorithm Using Radix-2 And Radix-4 Butterfly Elements

Ms. Ridhima Vijay Benadikar[1], Prof. Dr.(Mrs.) V. Jayashree[2]
[1](*Research Student, Electronics department,DKTE society's Textile & Engineering Institute, Ichalkaranji , India,*
[2](*Professor, Electronics department, DKTE society's Textile & Engineering Institute, Ichalkaranji, India,*
Corresponding Author: Ms. Ridhima Vijay Benadikar

Abstract : *Fast Fourier Transform (FFT) algorithms are widely used in many areas of science and engineering. Some of the most widely known FFT algorithms are Radix-2 algorithm and Radix-4 algorithm. In this paper, these two algorithms are implemented and their performances are compared. The key properties, e.g., area and power consumption, of the FFT processor depend mainly on the implementation of butterfly operations. Radix-2 butterfly and Radix-4 butterfly element is described with VHDL and synthesized on FPGA, target device 6slx100fgg484-3. After this the device utilization summary and timing summary is compared. The comparison shows that that 1-D FFT processor using Radix-2 butterfly element requires less number of slice registers, slice LUTs and fully used LUT-FF pairs as compared to Radix-4 butterfly element. Utilization of DSP48Es is almost negligible for 1-D FFT processor which uses Radix-2 butterfly element but the radix-4 is more efficient algorithm in terms of computation time. If the choice of algorithm is to be made solely based on memory usage and area consumption with respect to number of slice register used, number of slice LUT's and LUT's FF, the Radix-2 algorithm is better. The proposed processor organization allows the area of the FFT implementation to be traded against the computation time, thus the final structure can be easily tailored according to the requirements of the given application.*

Date of Submission:18-08-2018 Date of acceptance:03-09-2018

I. INTRODUCTION

FFT's can be broadly classified in pipeline FFT architectures and parallel FFT architecture. The Fast Fourier Transform (FFT) is an efficient algorithm to compute the Discrete Fourier Transform (DFT). The pipeline FFT is a particular class of FFT algorithms which can compute the FFT in a sequential manner; it achieves real-time actions with non-stop processing when data is continually given through the processor. When real-time large scale signal processing needs became prevalent, pipeline FFT architectures can provide solution [1] [2]. Several different 1-D's FFT architectures based on different decomposition techniques, such as the Radix-2 Multipath Delay Commutator (R2MDC), Radix-2 Single-Path Delay Feedback (R2SDF), Radix-4 Single-Path Delay Commutator (R4SDC), and Radix-22 Single-Path Delay Feedback (R22SDF) have been researched. Recently, Radix-22 to Radix-24 single path delay (SDF) FFT's were studied and compared; and R23SDF was implemented [3]. It is seen to be an area efficient for 2 or 3 multi-path channels. Pipelined FFT architectures cannot offer a solution for processing large FFT's because they consume a large amount of hardware area. This makes them inappropriate for implementation on a single FPGA chip. Parallel FFT Architecture help **t**o increase the performance For this numerous algorithms were proposed which can be implemented in hardware or software. These algorithms are known as Fast Fourier transforms (FFT). The first major FFT algorithm was proposed by Cooley and Tukey. Several FFT algorithms were proposed with a time complexity of O (n log n). Some of them are Radix-2 butterfly algorithm, Radix-4 butterfly algorithm and Split Radix algorithm. There are many forms of parallel systems available viz., shared memory multiprocessors and message based multi-processors [3].All butterflies in parallel approach for 1-D FFT would mean that all butterfly computations can be performed in parallel. All butterflies in a stage can be performed in parallel and then at the end of the stage, the results can be gathered. All nodes can do computation on the result of the first stage in parallel and output of the second stage can be gathered again and so on. This provides maximum scope for parallelism. [1]. This motivated us to implement architectures of 1-D FFT using Radix-2 & 1-D FFT using

Radix-4 in VHDL and deployed the same in target device 6slx100fgg484-3. Also the performance comparison between the two is carried w.r.to Device utilization summary & Timing summary.

The paper organization consists of Section II that explains theoretical background of Radix-2 and radix-4 butterfly element. Section III explains in detail architecture of implemented 1-D FFT with the help of 1-D FFT using Radix-2 and Radix-4 butterfly module respectively. After detailing theory and experimental procedure, results from MATLAB and VHDL implementation of Radix-2 BE and Radix-4 BE are presented in Section IV and section V respectively. Finally section VI highlights the comparison results of 1-D architecture for Radix-2 and Radix-4 butterfly PE. And section VII and VIII discusses the conclusion and future scope respectively.

II. THEORETICAL BACKGROUND OF RADIX-2 AND RADIX-4 BUTTERFLY ELEMENT

1. Radix-2 butterfly element

Usually the computation of the $N = 2^v$ point DFT is carried out by the divide-and conquer approach. The *N*-point data sequence is split into two *N*/2-point data sequences viz.; $f_1(n)$ and $f_2(n)$, equivalent to the even-numbered and odd-numbered samples of $x(n)$, respectively, that is [B1][B2].

$$f_1(n) = x(2n) \quad (1)$$

$$f_2(n) = x(2n+1), \qquad n = 0,1,\ldots,\frac{N}{2}-1 \quad (2)$$

Thus $f_1(n)$ and $f_2(n)$ are obtained by decimating input $x(n)$ by a factor of two, and the resulting FFT algorithm is called a *decimation-in-time algorithm***.** Now the N-point DFT can be represented in terms of the decimated sequences of DFT's as follows:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \qquad k = 0,1,\ldots,N-1$$

$$= \sum_{n\ even} x(n) W_N^{kn} + \sum_{n\ odd} x(n) W_N^{kn}$$

$$= \sum_{m=0}^{\left(\frac{N}{2}\right)-1} x(2m) W_N^{2mk} + = \sum_{m=0}^{\left(\frac{N}{2}\right)-1} x(2m+1) W_N^{k(2m+1)} \quad (3)$$

But $W_N^2 = W_{N/2}$. With this substitution, the equation can be expressed as

$$X(k) = \sum_{m=0}^{\left(\frac{N}{2}\right)-1} f_1(m) W_{\frac{N}{2}}^{km} + W_N^k \sum_{m=0}^{\left(\frac{N}{2}\right)-1} f_2(m) W_{\frac{N}{2}}^{km}$$

$$= F_1(k) + W_N^k F_2(k), \quad k=0, 1,..., N-1 \quad (4)$$

Where $F_1(k)$ represent the *N*/2-point DFT's of the sequences $f_1(m)$ and $F_2(k)$ represent the represent the *N*/2-point DFT of the sequences $f_2(m)$. Since $F_1(k)$ and $F_2(k)$ are periodic, with period *N*/2, so we have $F_1(k+N/2) = F_1(k)$ and $F_2(k+N/2) = F_2(k)$. In addition, the twiddle factor $W_N^{k+N/2} = -W_N^k$. Hence the equation may be expressed as;

$$X(k) = F_1(k) + W_N^k F_2(k), \qquad k = 0,1,\ldots,\frac{N}{2} \quad (5)$$

$$X(k+\tfrac{N}{2}) = F_1(k) - W_N^k F_2(k) \quad k = 0,1,\ldots,\frac{N}{2} \quad (6)$$

Here it is quite obviously seen that the direct computation of $F_1(k)$ and $F_2(k)$ both require $(N/2)^2$ complex multiplications. Furthermore, there are *N*/2 additional complex multiplications required to calculate $W_N^k F_2(k)$.

By computing *N*/4-point DFTs, we obtain the *N*/2-point DFTs $F_1(k)$ and $F_2(k)$ from the relations;

$$F_1(k) = F\{f_1(2n)\} + W_{N/2}^k F\{f_1(2n+1)\}, \quad (7)$$

$$F_1\left(k+\frac{N}{4}\right) = F\{f_1(2n)\} - W_{\frac{N}{2}}^{k} F\{f_1(2n+1)\}, \quad (8)$$

$$F_2(k) = F\{f_2(2n)\} + W_{N/2}^{k} F\{f_2(2n+1)\}, \quad (9)$$

$$F_2\left(k+\frac{N}{4}\right) = F\{f_2(2n)\} - W_{\frac{N}{2}}^{k} F\{f_2(2n+1)\}, \quad (10)$$

Where

$k = 0,1,\dots,\frac{N}{4}-1;\quad n = 0,1,\dots,\frac{N}{4}-1$ And $F\{*\}$ *represents Fourier Transform*

The decimation of the data sequence can be repeated until the resulting sequences are reduced to one-point sequences. For $N = 2^v$, this decimation can be performed v = $\log_2 N$ times. This reduces the total number of complex multiplications to (N/2) $\log_2 N$ and the number of complex additions to $N\log_2 N$. Basic butterfly architecture for Radix 2 is shown in figure below.

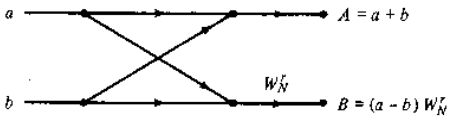

Figure 1. Butterfly Architecture for Radix-2

2. Radix-4 Butterfly element

For a DFT with the number of data points N is a power of 4 (i.e., $N = 4^v$), we can, use a radix-2 butterfly element for the computation. However, in this case, employing a radix-r FFT algorithm can be more efficient computationally. Here radix 4 DIT FFT algorithm is explained briefly. For 4-point DFT the N-point input sequence is split or decimated into four subsequence's, $x(4n)$, $x(4n+1)$, $x(4n+2)$, $x(4n+3)$, n = 0, 1, ... , N/4-1. [B1][B2]

$$X(p,q) = \sum_{l=0}^{3}\left[W_N^{lq} F(l,q)\right] W_4^{lp} \quad (11)$$

$$F(l,q) = \sum_{m=0}^{\left(\frac{N}{4}\right)-1} x(l,m) W_{\frac{N}{4}}^{mq} \quad (12)$$

$p = 0,1,2,3;\quad l = 0,1,2,3;\quad q = 0,1,2,\dots,\frac{N}{4}-1$ And

$$x(l,m) = x(4m+1) \quad (13)$$

$$X(p,q) = X\left(\frac{N}{4}\right)p + q \quad (14)$$

Thus the four N/4-point DFT's F (l, q) obtained in equations are combined to yield the N-point DFT[4]. The expression for combining the N/4-point DFT's defines a radix-4 decimation-in-time butterfly. Thus the N/4-point DFT can be expressed in matrix form as,

$$\begin{bmatrix} X(0,q) \\ X(1,q) \\ X(2,q) \\ X(3,q) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} W_N^{0} F(0,q) \\ W_N^{q} F(1,q) \\ W_N^{2q} F(2,q) \\ W_N^{3q} F(3,q) \end{bmatrix} \quad (15)$$

The radix-4 butterfly is depicted in Figure 2 (a) and a more compact form is shown in Figure 2(b). Here each butterfly involves three complex multiplications, since $W_N^0 = 1$, and 12 complex additions.

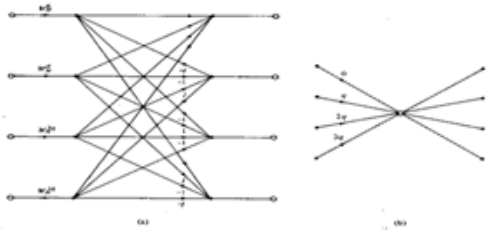

Figure 2. Butterfly Architecture for Radix-4

The architecture of 1-D FFT is explained below.

3. Architecture of 1-D FFT core

The proposed parallel 1-D FFT processor (PE) as shown in Figure 3 consists of dual port RAM memory, Address Generation Unit (AGU), butterfly unit and Look up tables (LUT). The butterfly operation is the heart of the FFT processor. It takes data words from memory and computes the FFT using principle of pipelining.. The AGU provides the I/O RAM's and twiddle coefficient look-up tables (LUT's) with the correct addresses. The AGU keeps track of the mode of operation and generates the necessary addresses as the address generation during data input, output and FFT computation processes are different. It also performs bit-reversal of output data at the end of each FFT execution. Respective LUT and FFT are then computed.

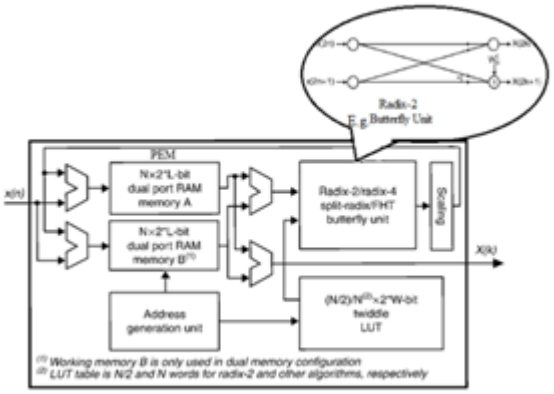

Figure 3. Block Diagram of 1-D FFT with single butterfly element

The twiddle LUT ROM's stores the twiddle coefficients (sine and cosine values) used in the FFT computation. For butterfly unit operation, the twiddle coefficients are taken from the ROM. The results are written back to the same memory locations as an in-place algorithm is used.

To accomplish this, Radix-2 butterfly PE is designed using a 16 point DFT. It requires 32 twiddle coefficients with 4 stages for FFT computation as shown in figure 5[3]. In each stage eight, radix-2 butterflies are used. Instead of using different butterflies in this architecture uses only one butterfly unit in each stage.

1-D FFT architecture using Radix-2 and Radix-4 algorithms reported by I.S.Uzun et al. has been implemented and presented in Section III and Section IV respectively. These are mostly used for practical application due to their simple structure with constant butterfly geometry and possibility of performing them in place.

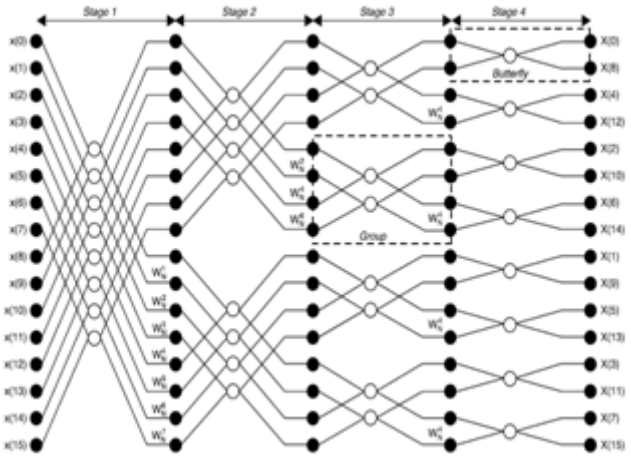

Figure 4. signal flow graph of 16 point DIT FFT

II. IMPLEMENTATION OF 1-D FFT USING RADIX-2 & RADIX-4 BUTTERFLY ELEMENT (BE)

Introduction

1. Part-I: 1-D FFT implementation using RADIX-2 BE

Implemented block diagram for 1-D radix-2 FFT as shown in Figure 5 consists of A)MCU B)AGU C) A single Radix-2 Butterfly processing element, D)FFT RAM employing 1-D in-place computation , E)Coefficient ROM. Design of these employ 1-D in-place computation[5][6]. The different ROM memory stores the Coefficients as 16-bit format. Design of these subcomponents of 1-D radix-2 FFT are explained further.

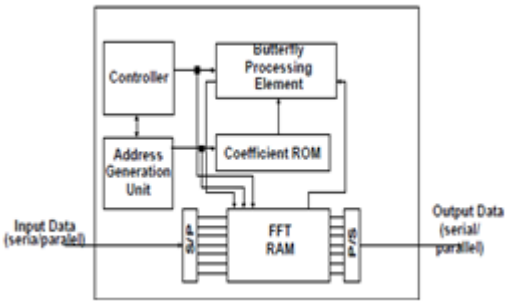

Figure 5. Architecture of 1-D FFT processor using Radix-2 Butterfly elemen

A. Master control unit (MCU) for 1-D FFT

A Moore machine based controller is used to determine the sequence of events based on the feedback received from other functional units. Implemented MCU and its state diagram is as shown in Figure 6 and 7 respectively.

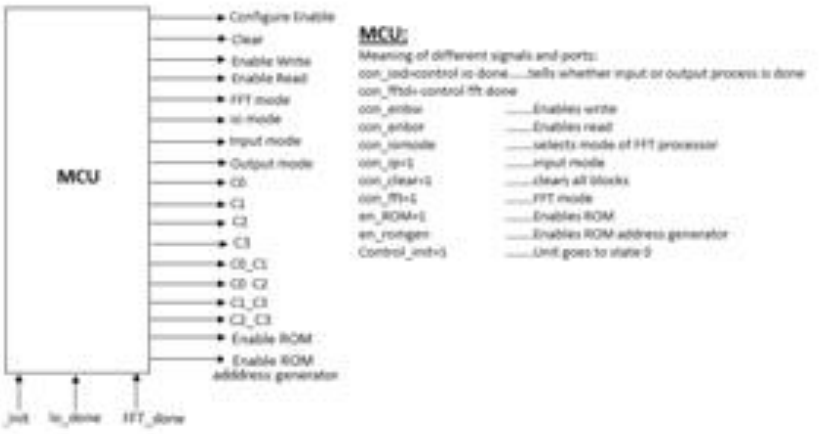

Figure 6. Block schematic of control unit

The state diagram of MCU is as shown in Figure 7 and state Table of master controller unit, has 8 states as given in Table I which is self explanatory.

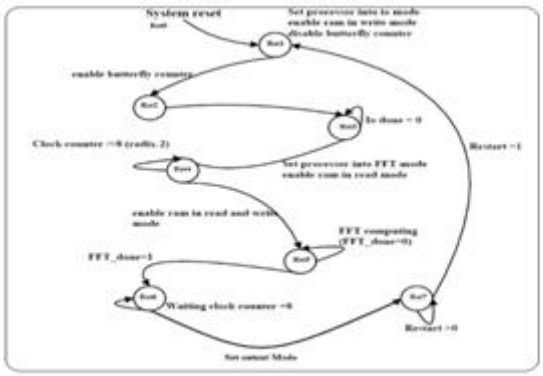

Figure 7. State diagram of Master control unit

TABLE I. STATE TABLE FOR MCU

States	Operation
rst0	FFT processor selects value of N
rst1	FFT processor enables the RAM into write mode
rst2	FFT processor enables all the units
rst3	FFT processor starts taking data inputs which are stored in RAM for FFT processing
rst4	FFT calculation starts
rst5	FFT calculation starts while previous output is being read
rst6	When counter reaches 8, final output of each stage is calculated and it is written into the RAM
rst7	FFT processor starts reading the outputs.

B. Address generation unit

The purpose of the address generation unit is to generate and access the RAM and the coefficient ROM with the correct addresses. Address generation during input output and FFT computation processes are different.AGU has to keep track of the mode of operation of the system supplied by the control Unit and generate the required addresses Figure 9 shows a block level representation of the AGU.

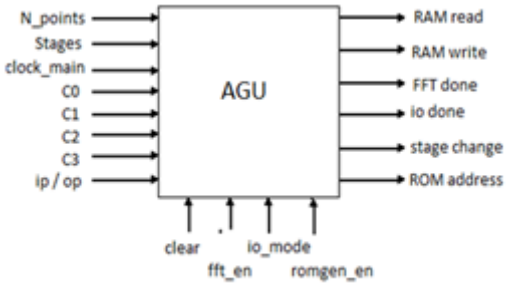

Figure 8. Block schematic of implemented AGU

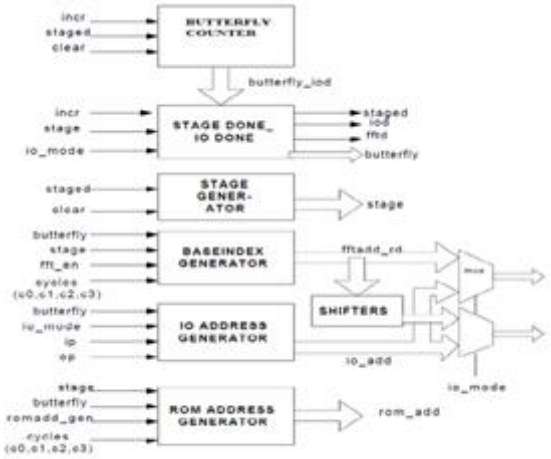

Figure 9: Architectural block diagram of Address Generation Unit.

IO stage has to keep track of butterfly computation in a particular stage. It uses IO Done signal for this. For 16-point FFT there are 8 butterflies per stage and 4 data words per butterfly (2 real and 2 imaginary) so an up counter is used. Same butterfly is used in every stage of FFT processing. Three signals called "iod", "staged" and "butterfly" are generated. "iod" is generated when either the Data Input or Output process is finished. The "butterfly" is the counter output which is also used for addressing the RAM during Input and Output Processes and providing basic timing for the FFT process. The "staged" signal is generated when the current "butterfly" count is 8, then stage generator increments the stage generator by one. The current stage in the FFT computation is given by the stage generator. The stage generator supplies the base index generator of the current the stage that is being computed at present. The butterfly counter is of 3 bits since 16-point FFT, has 4 stages Hence the stage generator and a 3- bit counter is incremented once every 8 butterfly counts (by the "stage done" signal). It

generates a signal called "FFT done" when the stage the count is eight. The controller is informed that the FFT computation process is complete and FFT processor can start the data output process.

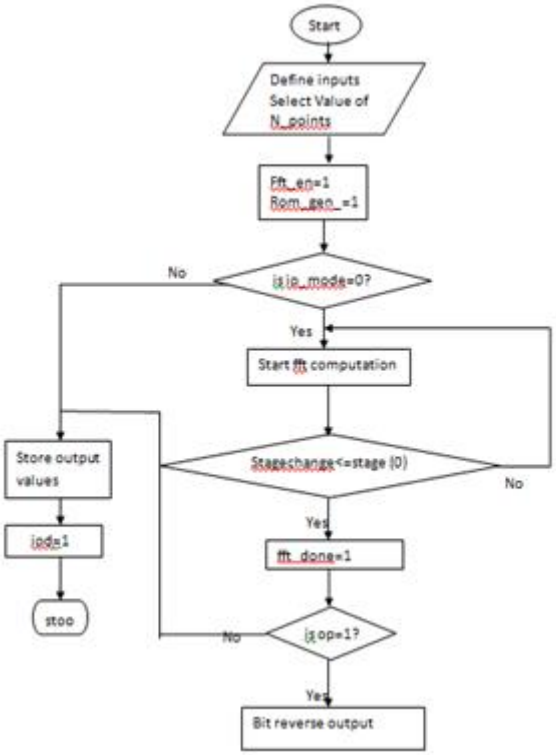

Figure 10: Flowchart of AGU

The block IO Address Generator from the AGU generates addresses for RAM during the data input and output processes. During the data input process the output of the butterfly generator "butterfly" can be used for addressing the RAM. However, during the data output process, output data should be bit-reversed while being written into the RAM. The addresses for bit-reversed outputs are selected by the multiplexer's of AGU once in the output process starts. The flowchart of AGU is as shown in figure.

C. Radix-2 butterfly element for 1-D FFT

The implemented radix-2 butterfly processing block schematic design is as shown in figure 11. The butterfly processing unit accepts two complex numbers and a complex twiddle factor as inputs and computes the DFT which results in real and imaginary parts of each number are stored separately. Each complex number is read from two memory banks of coefficient ROM. Reading the two inputs from RAM requires four cycles to complete. Pipeline concept is adopted for other later inputs which are still being read in, with calculations for the first inputs are already set in. As the inputs are passed on to the following processes, new inputs continue to be read in for the next operation. As calculations complete, the outputs are written back into the exact banks that they were read from. This constitutes an in-place computation and removes the requirement of additional buffer memory that is necessary with some types of FFT processors. The inputs c0 to c3 are the 4 cycles required to compute single butterfly operation.

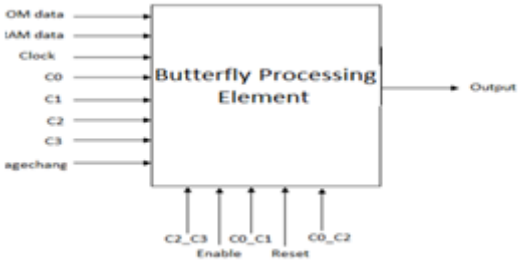

Figure 11. Block schematic of Butterfly PE

This type of implementation of the butterfly processor aids to an increase in computational speed at a cost of increased silicon area relative to using a serial parallel multiplier. The butterfly processing element takes four cycles to compute a two-point FFT hence has a latency of five cycles. The architecture of implemented of butterfly processing element is as shown in figure 12. Here signals ROM data, RAM data and output are of 16 bits each because we are giving 16 bit input to the FFT generator. As we are using single butterfly element it takes 32 cycles to complete one stage.

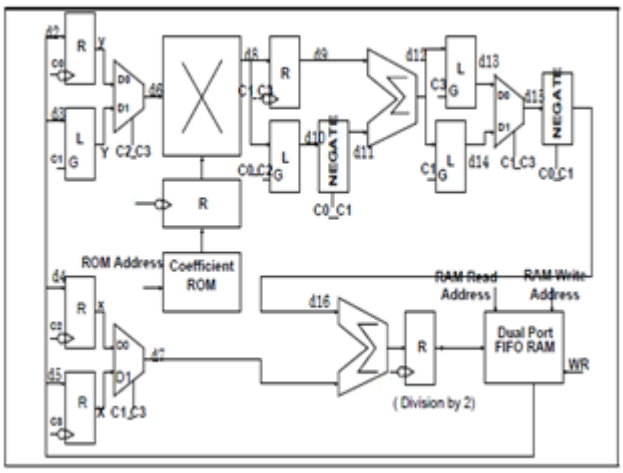

Figure 12. Radix-2 Butterfly Processor

The figure 13 shows the flowchart for implementation of butterfly processing element which is self explanatory.

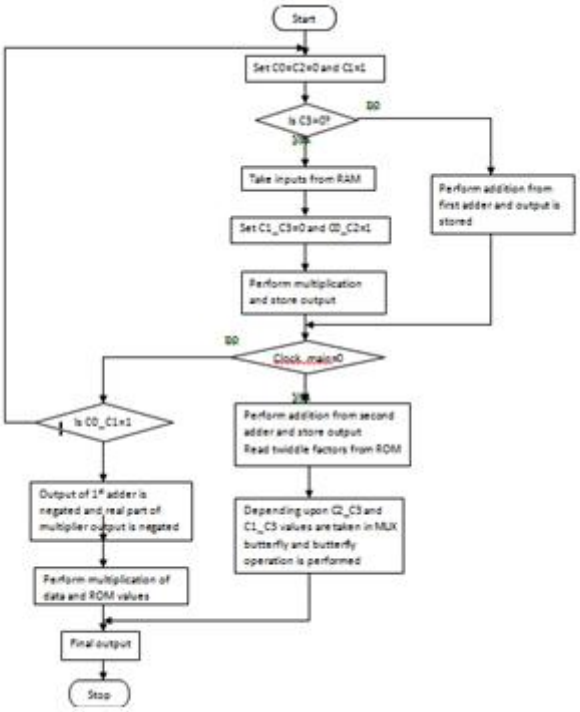

Figure 13. Flow chart of Butterfly processing element.

D. Designing of RAM for 1-D FFT & implementation of "In- place" algorithm

In "In-place" FFT is simply an FFT that is calculated entirely inside its original sample memory which does not require additional buffer memory. It is necessary to first shuffle the sequence x (n) by reversing the binary representation of the index. Calculations done sequentially and results are stored in the memory, so that at each calculation of butterfly the input can be read from there. The loading phase works as per the Table II with respect to io_mode signal.

TABLE I. MODE FOR IN-PLACE ALGORITHM

io_mode='0'	we='1'	data is written in to the RAM
io_mode='1'	we='0'	data is read from the RAM

As shown in Figure 14, RAM has FFT data, io data and output all of 16 bits. Read and write addresses are of 13 bits. It has three control lines which are a) Mode select b) For reading c) writing of RAM, Read and write addresses are of 13 bits. Clock is a reference signal.

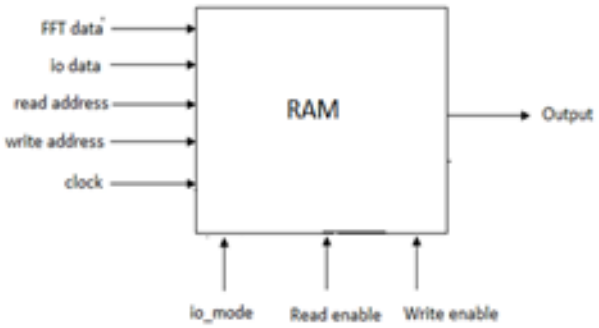

Figure 14. Block schematic of implemented RAM.

E. Coefficient ROM

Twiddle factors are integral part of FFT computation and these are stored in RAM memory in software implementation. It indicates the need for large memory and power consumption.
ROM table is an alternative approach for storing the twiddle factors which overcomes the drawback of RAM approach which is as shown in Figure 15. Twiddle factors are calculated using formula and are stored in ROM

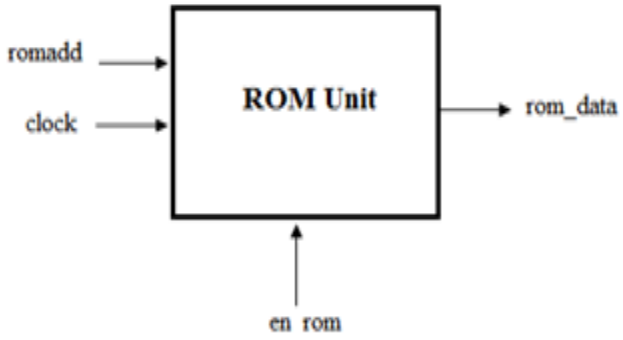

Figure 15. Block schematic of ROM.

$$W_{N=e^{-j2\pi/N}} \qquad (16)$$

16 point FFT is implemented which requires 32 twiddle factors by using symmetry property of twiddle factor only 16 twiddle factors are used and are stored in the ROM. Input signal, "romadd" is of eleven bits and output signal "rom" data is of 16 bits because twiddle factor values are of 16 bits.

III. EXPERIMENTAL RESULTS OF 1-D FFT USING RADIX-2 BE

Results on implementation of 1-D FFT using Radix-2 BE are presented further. Results and observations on implemented 1-D FFT in xilinx platform with 6slx100fgg484-3 as a target device are explained further. RTL schematic of individual blocks in 1-D FFT processor Viz.; A,B,C,D,E,F are presented further along with the Top level entire FFT module.

A. RTL schematic of MCU

Figure 16 shows the RTL schematic of MCU. Master control unit which consists of two sub units, control unit and cycle's unit. Cycle's unit consists of cycle generator and cycle's waveform generator.

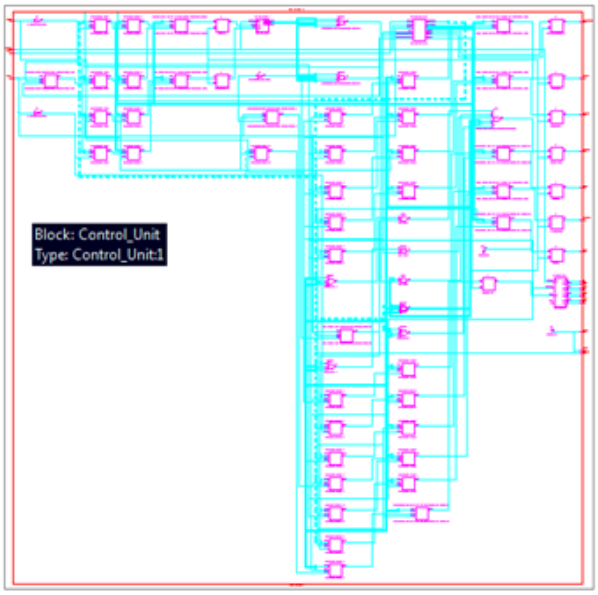

Figure 16. RTL schematic of Master Control Unit.

B. RTL schematic of Address generation unit

Figure 17 represents RTL schematic of Address Generation Unit. Its consists of butterfly counter, stage counter, IO stagedone, address index generator, address shifters, Multiplexer's, ROM address generator and clock multiplexer. It is seen that there is a latency of five cycles i.e., if "y" is read from the RAM during cycle "c0", "y1" is written into the same location during cycle "c1" as it was read after 5 cycles. This makes the read address to shift in each on these five cycles. The output of the last shifter is then given as the write address. The ROM Address Generator provides the ROM with the correct address for collecting the Twiddle factors. Clock multiplexer chooses between clock and cycles.

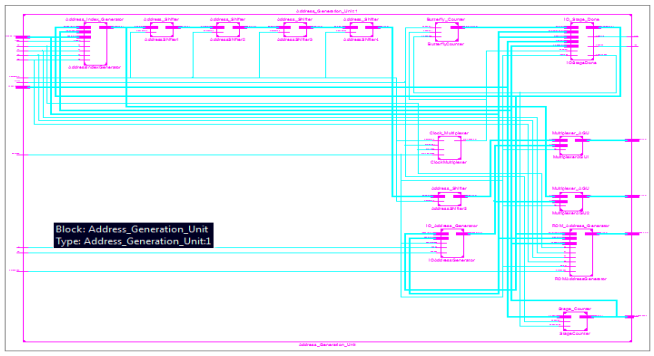

Figure 17. RTL schematic of Address Generation Unit.

C. RTL schematic of Butterfly processing element

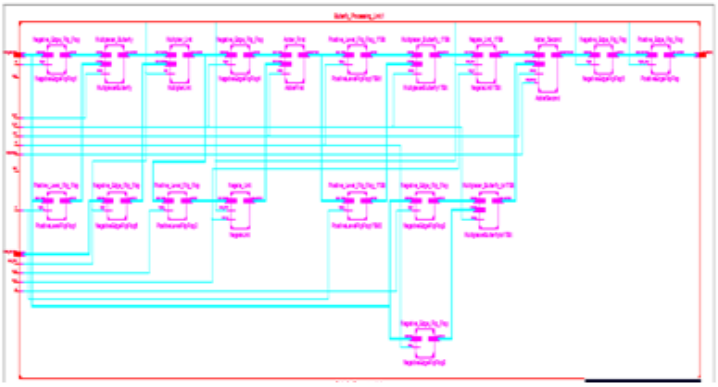

Figure 18. RTL schematic of Butterfly processing unit

The butterfly processing element consists of clock_main , clock , reset and enable input signals, c0, c1, c2, c3, c0_c1, c2_c3, c0_c2, c1_c3 and stage change control signals. Signals data_rom and ram_data are of 16 bits each. And final output signal out_data is of 16 bits.

D. The RTL schematic of RAM

It is shown in Figure 19. In RAM module, address width is 13 bits and data width is 16 bits. First bit of address represents the value of N for FFT.

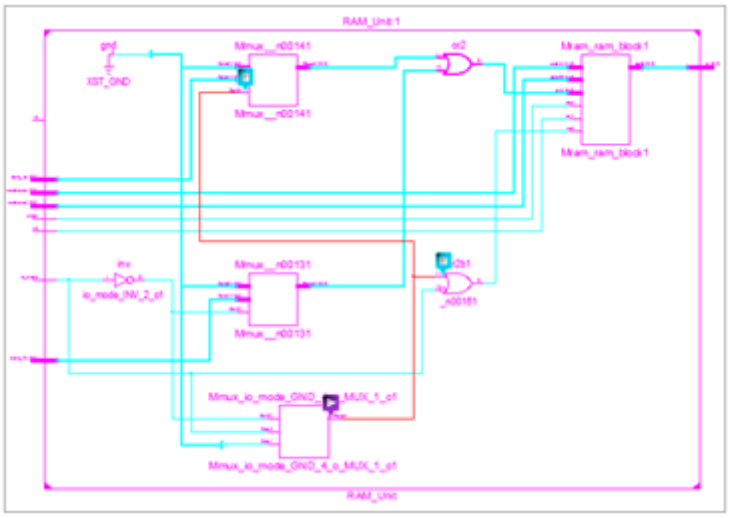

Figure 19 RTL schematic of RAM.

E. RTL schematic of Coefficient ROM

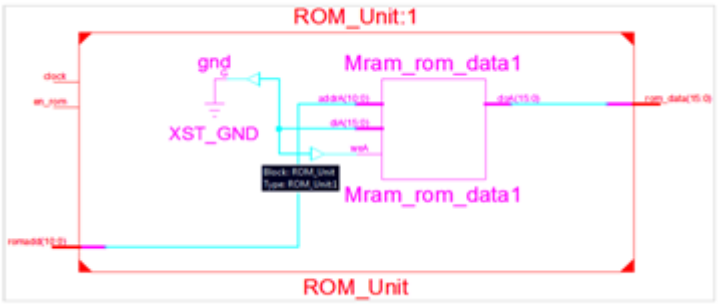

Figure 20 RTL schematic of ROM

Figure 20 represents RTL schematic of ROM. It consists of input signals clock, en_rom and romadd, romadd is of 11 bits. Output of ROM module is signal rom_data which is of 16 bits

F. The RTL shematic of 1-D FFT using Radix-2 butterfly Element

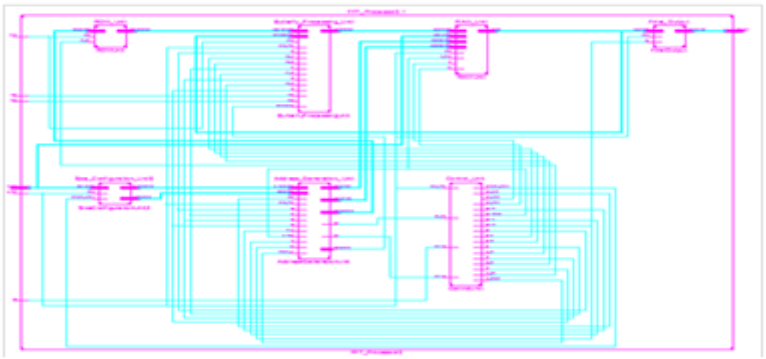

Figure 21. RTL Schematic for 1-D FFT using Radix-2 butterfly Element

The implemented 1-D FFT processor using Radix-2 butterfly element with its RTL schematic is as shown in Figure 21.

Simulation results obtained for 1-D FFT

The input 0000000000010000 decides value of N i.e.N point FFT. Input sequence given to FFT processor is generated randomly from the MATLAB which is given below,

x(0)=1000000110001001
x(1)=0000000111101011
x(2)=1000000001110000
x(3)=0000001010100000
x(4)=1000001000001001
x(5)=1000000110000010
x(6)=1000001001000100
x(7)=1000001001010100
x(8)=0000001001011101
x(9)=0000000010000010
x(10)=0000000001010001
x(11)=1000001001000101
x(12)=0000001001000010
x(13)=0000000011000111
x(14)=1000000011110100
x(15)=0000000000010101

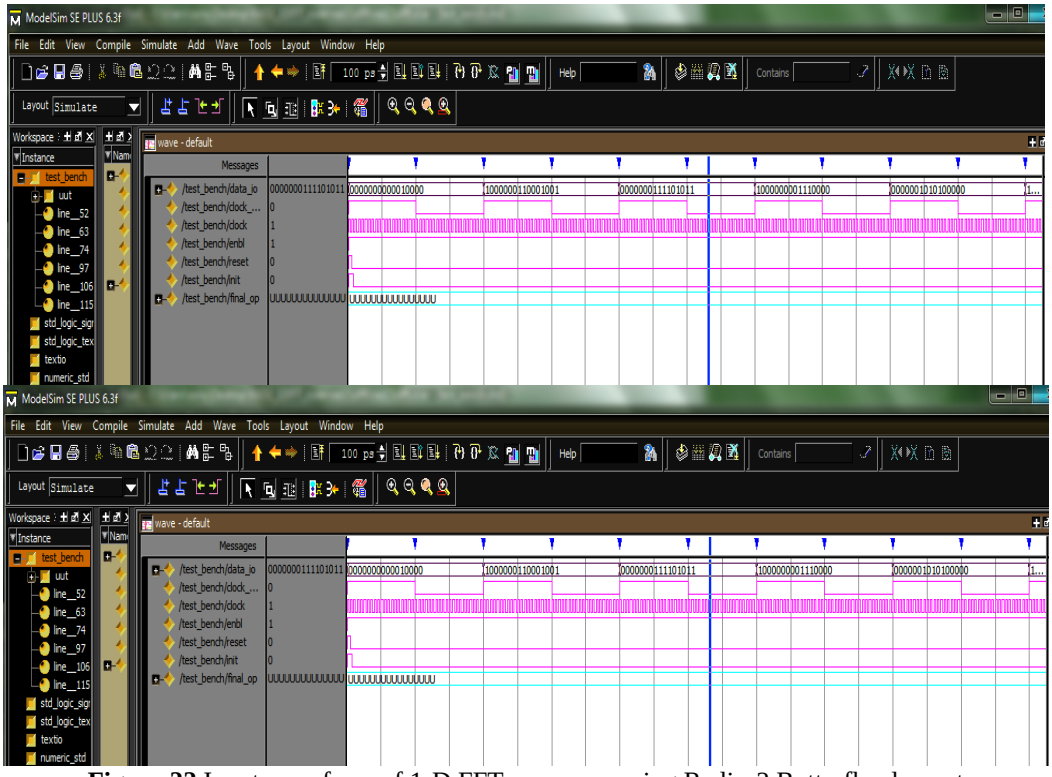

Figure 22 Input waveform of 1-D FFT processor using Radix-2 Butterfly element

MODELSIM Output: The output waveform of 1-D FFT processor using Radix-2 Butterfly element is shown in Figure 23. Input generated from MATLAB is applied to FFT processsor and figure 23 shows the output generated from 1-D FFT processor. Input is generated from the MATLAB and given as input to 1-D FFT processor implemented in VHDL. The ouput results obtained from the designed 1-D FFT processor is comapred with the MATLAB output.

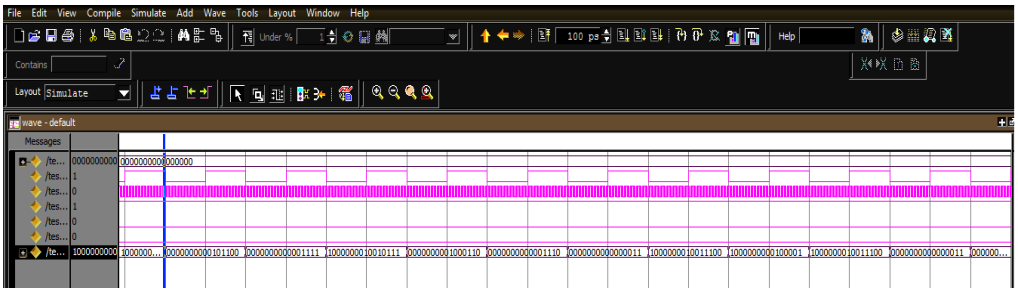

Figure 23 Simulation Results in 16 Point 1-D FFT using Radix-2 Algorithm

Comparison of MATLAB output and MODELSIM output

The output of FFT processor designed and output of MATLAB is compared and verified as shown in Figure 24.[9] It can be seen that output of our FFT processor and output of MATLAB's FFT processor is exactly same.

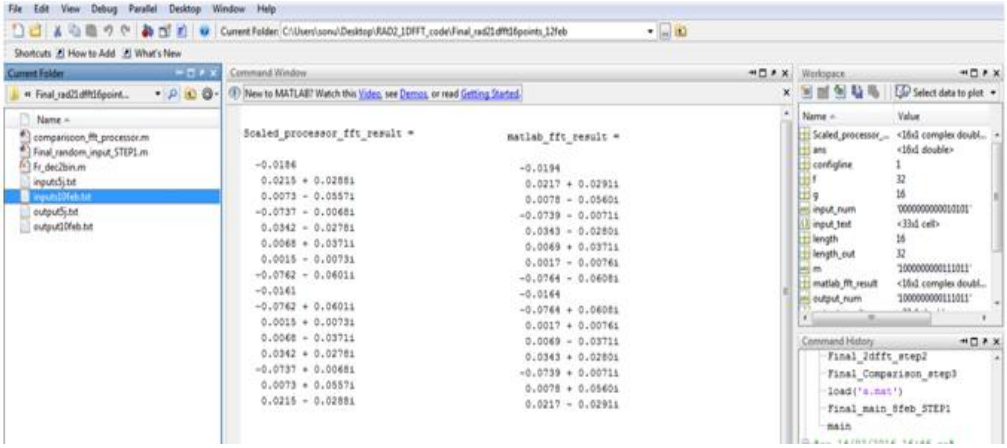

Figure 24 Simulation and MATLAB results for1-D FFT using Radix-2 butterfly element

Part-II Implementation of 1-D FFT using Radix-4 BE

For the design and experimentation of 1D FFT using Radix-4 BE the target device used was 6slx100fgg484-3. The details of Design and implemented results are explained further. Architecture of 1-D FFT using Radix-4 butterfly element is as shown in figure 25[5] [6]

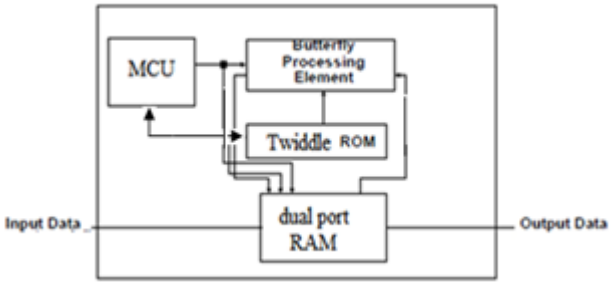

Figure 25 Architecture of 1-D FFT using Radix-4 butterfly element

Architecture of 1-D FFT using Radix-4 butterfly element consists of 4 sub components with certain variations w.r.to 1-D FFT PE using Radix-2 BE viz.; A.MCU B.Butterfly processing element (Switch,Three 64-bit registers,Three 256-bit registers and Swap Unit), C.RAM, D. Twiddle ROM (with ROM, Four 32-bit registers,Four 2-input floating-point multipliers and Two 32-bit floating-point adders)[7][8] .

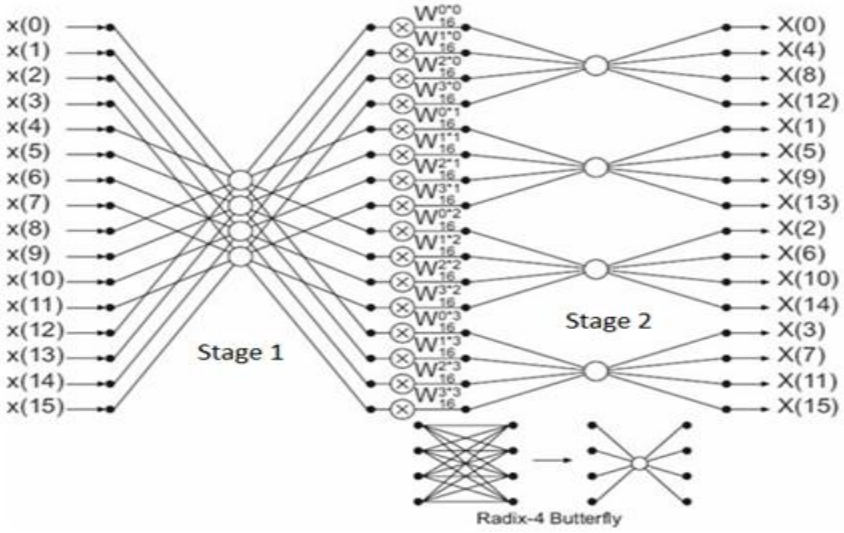

Figure 26 Signal flow graph of 16 point Radix-4 FFT

The 1-D architecture using Radix-2 BE has a single butterfly core. This uses Pipelined FFT architectures which is not suitable solution for processing large FFTs since the power consumption by them is large as amount of hardware area is large making them unsuitable for implementation on a single FPGA chip (especially for 2-D FFT implementations). The simplified architectural block diagram and flow chart of the 1-D FFT using Radix-4 design is depicted in signal flow graph of Figure 26.

Signal flow graph in Figure 26 is for 16 point radix-4 FFT. To compute 16 point FFT using radix-4 butterfly processing element it requires only 2 stages unlike 4 stages for Radix 2 algorithm.

A. Master Control Unit (MCU)

In this MCU architecture master control unit and address generation unit are merged together. The MCU generate all the logic needed to control the other components in the FFT processor. The state machine stores and generates all the control signals for the FFT processor's operation at every step, with respect to the clock. A reset signal resets the state machine counter. Further this signal act as the beginning of a new FFT calculation. At the last the FFT processor asserts a done signal to communicate the completion of the FFT. The Master Control Unit is a 4-stage state machine that is responsible for directing the flow of the entire data path throughout the entire calculation of the FFT.

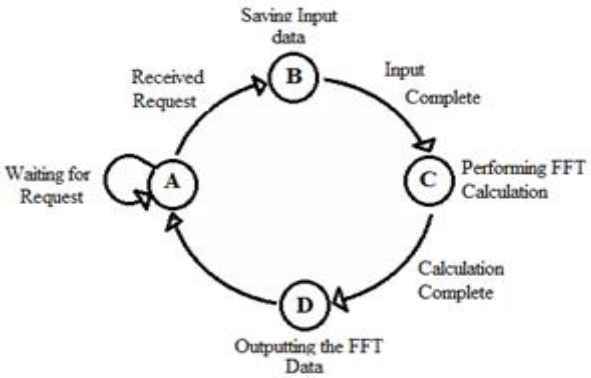

Figure 27 State diagram of MCU

Table I. State Table For Mcu Of Radix-4 Fft Pe

State	Operation
State A	MCU waits for a request to perform FFT calculation MCU send busy signal while calculating FFT Proceeds to state B after FFT calculation is done
State B	Saves input to RAM unit When the RAM is filled with data, the MCU proceeds to state C
State C	All the computation is done After all stages of the FFT have been computed, MCU goes to state D
State D	Sends " Done" signal to indicate that the output of the FFT calculation is received in next clock pulse

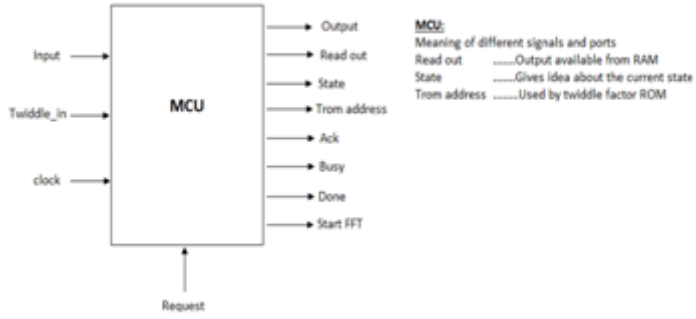

Figure 28 Block schematic of implemented MCU

B. Radix-4 butterfly element for 1-D FFT processor

Radix-4 butterfly element for 1-D FFT consists of a. Swap Unit b. Switch c. Three 64-bit registers and d. Three 256-bit registers.

Swap unit is used to reduce the number of complex multiplications; the swap unit is responsible for preparing the four values currently being used to calculate the 4-point DFT. The unit will take the four inputs simultaneously from the four separate memories. The inputs will then be used to calculate the values needed to execute (5) to (6).

$$X(4k) = \sum_{n=0}^{\frac{N}{4}-1}\left[x(n) + x(n+N/4) + x\left(n+\frac{N}{2}\right) + x(n+\frac{3N}{4})\right] W_N^0 W_{N/4}^{kn} \quad (5)$$

$$X(4k+1) = \sum_{n=0}^{\frac{N}{4}-1}\left[x(n) - jx(n+N/4) - x\left(n+\frac{N}{2}\right) + jx(n+\frac{3N}{4})\right] W_N^n W_{N/4}^{kn} \quad (6)$$

$$X(4k+2) = \sum_{n=0}^{\frac{N}{4}-1}\left[x(n) - x(n+N/4) + x\left(n+\frac{N}{2}\right) - x(n+\frac{3N}{4})\right] W_N^{2n} W_{N/4}^{kn} \quad (7)$$

$$X(4k+3) = \sum_{n=0}^{\frac{N}{4}-1}\left[x(n) + jx(n+N/4) - x\left(n+\frac{N}{2}\right) - jx(n+\frac{3N}{4})\right] W_N^{3n} W_{N/4}^{kn} \quad (8)$$

Depending on the twiddle factor's value needed, this unit will do one of three calculations to the inputs: multiply by -1, multiply by j or multiply by –j multiplying by -1 is easily done by inverting the sign. The real and imaginary parts require swapping and the sign of the new real part needs inverted for the process of multiplication by j factor. The final task of multiplying by –j is done by switching the real and imaginary parts as well as inverting the sign of the new imaginary part. All the necessary values are then outputted to various registers in order to be used in the adder unit. Figure 30 shows the block schematic of swap unit. State signal is of 2 bits. A to D inputs are of 16 hexadecimal bits. Final output of SWAP unit is 256-bit output that contains a packed form of the four 64-bit data points to be used by the Twiddle Unit for the butterfly calculation.

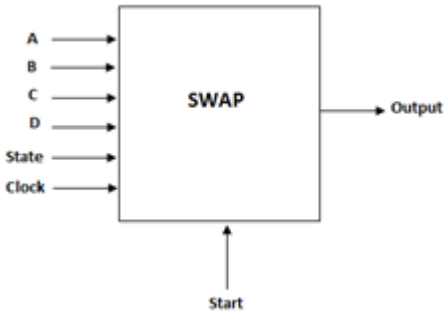

Figure 29 Block schematic and implemented SWAP unit

Four calculations to be performed in swap unit are as shown in Table IV for Arithmetic operations.

Table I. Arithmatic Operation Associated With State

State	Operation
01	Four inputs "a" "b" "c" and "d" are passed
10	Passes "a" without modification and multiplies "b" by j, "c" by -1 and "d" by j
11	Passes "a" and "c" without modification and Multiplies "b" by j, "d" by -1
00	Passes "a" without modification and Multiplies "b" by j, "c" by -1 and "d" by j

Butterfly in stage 2 consists of two 4-input adders, one for the real part and one for the imaginary part. The four input adders consist of three 2-input adders. For butterfly stage 1 input is of 64 bits and output is of 256

bits. State Butterfly performs a simple set of commands. The first step is to take the data from the RAM unit and load one of the three 64-bit registers depending on the switch selection. The three 64-bit registers' data in conjunction with a clear path connection from the switch make up the four butterfly inputs. These four butterfly inputs are then passed onto the Swap Unit, which performs a simple arithmetic computation of multiplying by ±1 or ±j.

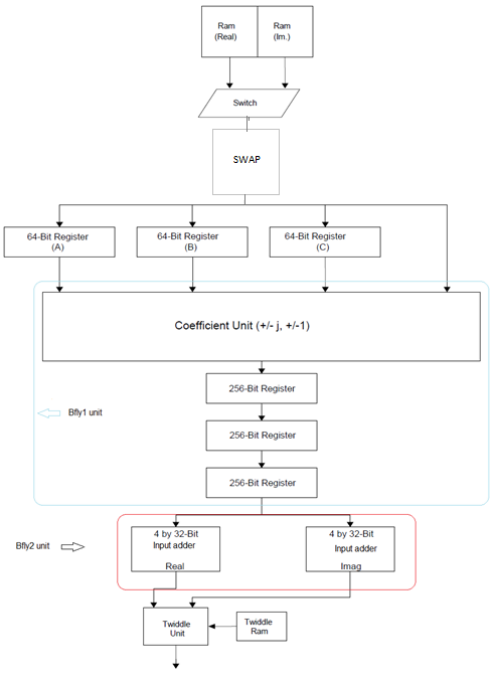

Figure 30 Radix-4 Butterfly processing unit

This is done by either switching the real and imaginary parts, inverting the sign bit or both. The 256-bit register consists of both the real and imaginary parts that are used to compute a single point of each butterfly operation. This data is passed to the same butterfly in stage 2. The 256-bit registers enable control unit to select the next four butterfly inputs to go through the switch loading the three 64-bit registers, one at a time, with the next four butterfly operands signal is of 2 bits. For butterfly stage 2 inputs is of 256 bits and output is of 64 bits.

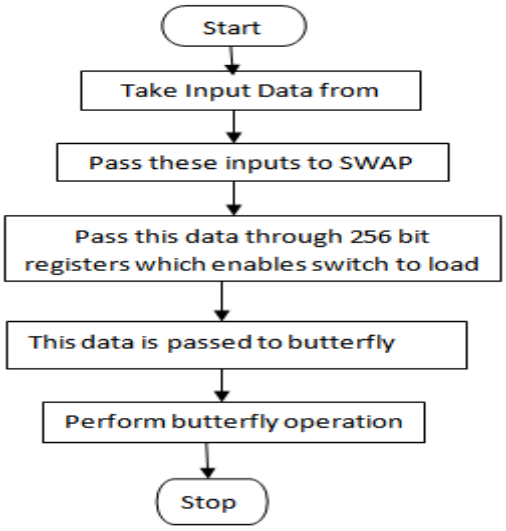

Figure 31 Flowchart of Radix-4 Butterfly

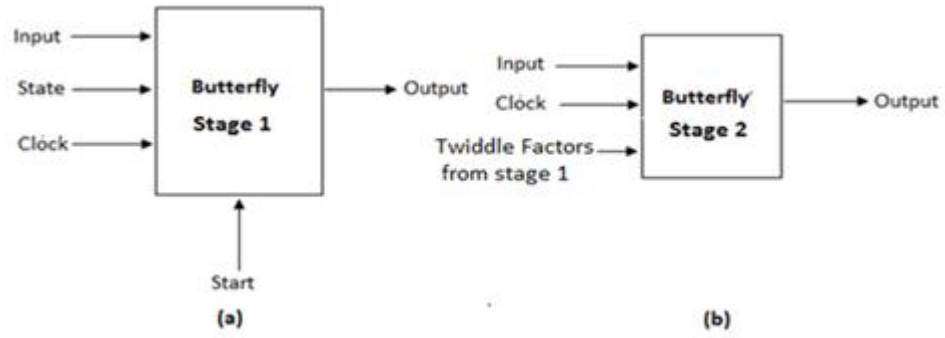

Figure 32 Block schematic of Radix-4 Butterfly processing unit

C. RAM

The RAM unit is responsible for storing the data input, all intermediary stage outputs and the final data output. The concept of in place algorithm is explained in previous section.

D. Twiddle ROM unit

Twiddle ROM is comprised of the Twiddle Unit and the Twiddle ROM, The Twiddle ROM is preloaded at download time and is parameterized depending on the length, *N*, of the FFT. The architecture of twiddle ROM consists of the following components. Twiddle ROM, Four 32-bit registers, Four 2-input floating-point multipliers and two 32-bit floating-point adders. This unit calculates both the real and imaginary part simultaneously. The twiddle ROM unit receives the real and imaginary values from the intermediate calculation in previous stage's intermediary calculation. This value is then multiplied with the twiddle value from the preloaded Twiddle ROM, which is essentially a complex multiplication.

The selected Twiddle Unit was chosen because of the ease of control. The simplification comes with a cost of doubling the amount of required floating-point units, but in turn it removes the necessity for a separate Twiddle Control Unit and several multiplexers that were required for the proposed Twiddle Unit

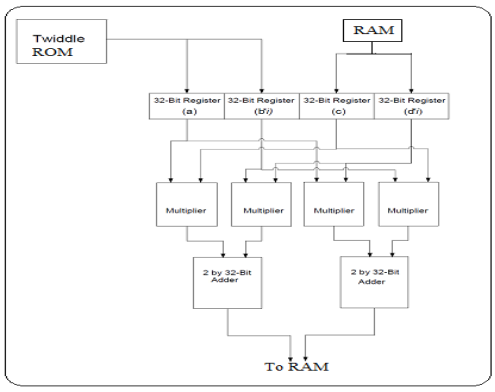

Figure 33 Architeccture of Twiddle Unit

Figure 33 shows the implemented block schematics of twiddle ROM consist of input, output, enable ROM and Trom address. Input and output is of 64 bits and Trom address is of 4 bits.

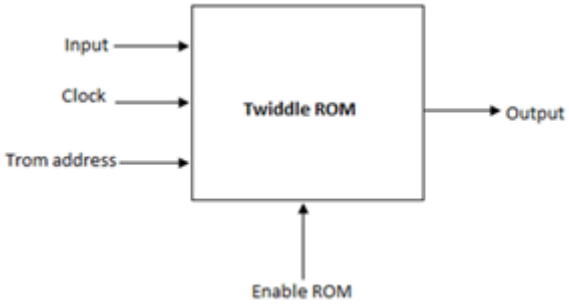

Figure 34 Block schematic of implemented twiddle ROM

E. Design of floating-point multiplier

The floating-point multiplier unit consists of four stages as shown in figure 36.

Table I. State Table Of Floating Point Multiplier

Stage	Operation
Stage 1	Takes two 32 bit floating point i/p unpacks them(fraction , exponent and sign) and calculated the sign of the result
Stage 2	Adds exponent of two inputs as well as computes the multiplication of the fractional numbers
Stage 3	Normalizes the result to the IEEE format for single precision floating point numbers
Stage 4	Writes the results of the multiplication. If all MSBs of the normalized product are 1's, rounding can generate a carry-out. In that case, normalization (stage 3) has to be done again.

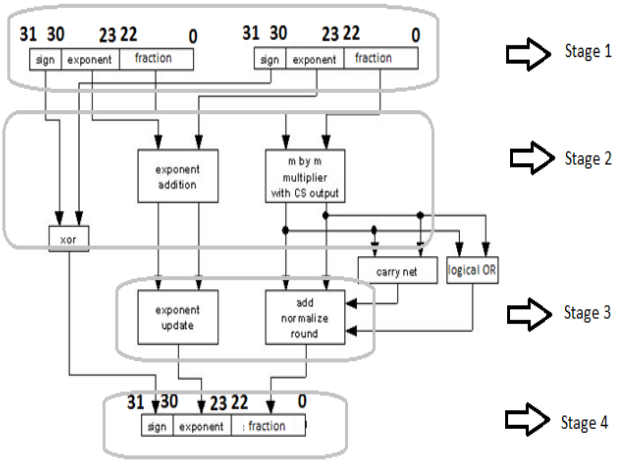

Figure 35 Floating point Multiplier

Flowchart of implemented floating point multiplier is as shown in Figure 37 consists of blocks corresponding to various stages which is as self explanatory. Given floating-point numbers $\text{Operand}_A = (s_A, e_A, f_A)$ and $\text{Operand}_B = (s_B, e_B, f_B)$, the stages for computing $\text{Operand}_A * \text{Operand}_B$.

F. Floating point adder

The conventional floating-point addition algorithm has five stages, which are, exponent difference, pre-alignment, addition, normalization and rounding. Given floating-point numbers $\text{Operand}_A = (s_A, e_A, f_A)$ and $\text{Operand}_B = (s_B, e_B, f_B)$, the stages for computing Operand_A + Operand_B are described in flowchart of adder figure 36.

- Find exponent difference $d = e_A - e_B$ *if* $e_A < e_B$, and swap position of mantissas. Set larger exponent as tentative exponent of result.

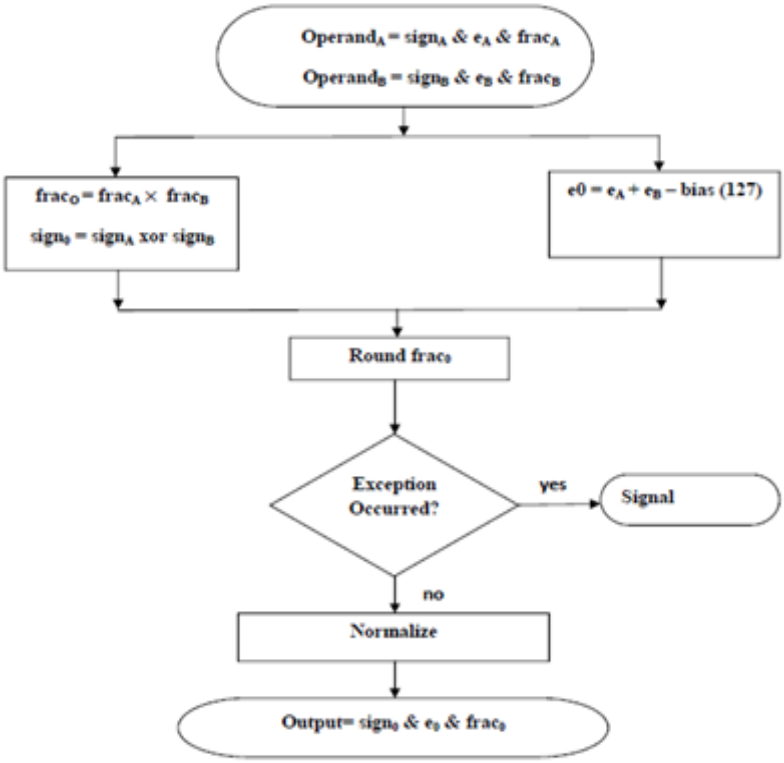

Figure 36 Flowchart of floating point multiplier

- Pre-align mantissas by shifting smaller mantissa right by d bits. Get tentative result for mantissa by adding or subtracting mantissas
- Perform Normalization.
- Shift result left and decrement exponent by the number of leading zeros to compensate for leading-zeros in the tentative result. If tentative result overflows, shift right and increment exponent by 1-bit. Round mantissa result. If it overflows due to rounding, shift right and increment exponent by 1-bit.

Exceptions

The IEEE standard defines five types of exceptions that should be signalled through a one bit status flag when encountered.

Invalid Operation: Some arithmetic operations are invalid; the result of an invalid operation shall be an isNaN (Not a number).The following are some arithmetic operations which are invalid operations and that give as a result isNaN signal.

Addition or subtraction: $\infty + (-\infty)$, Multiplication: $\pm 0 \times \pm \infty$

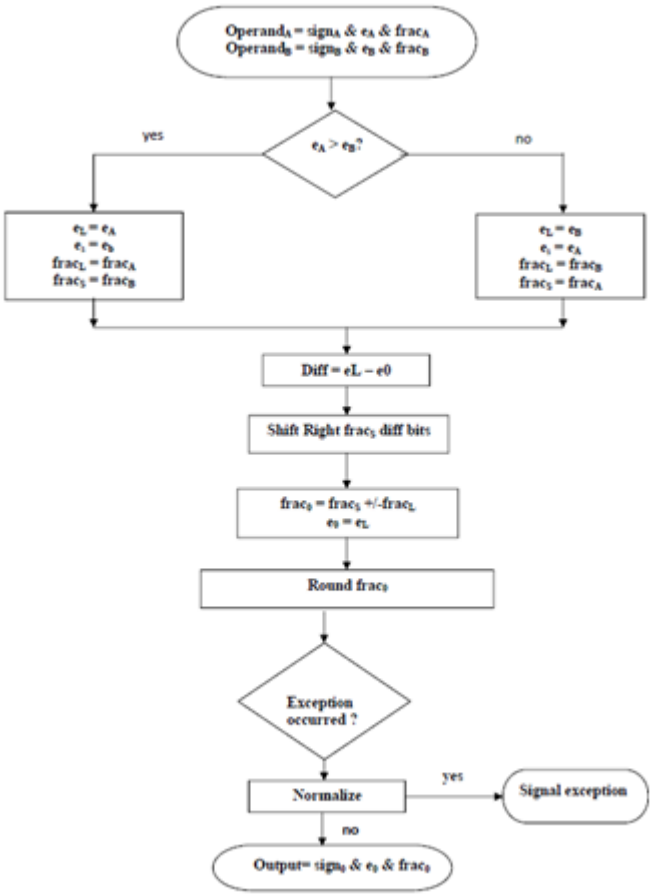

Figure 37 Flowchart of floating point adder

Overflow: The overflow exception is signalled whenever the result exceeds the maximum value that can be represented due to the restricted exponent range.
Infinity: This exception is signalled whenever the result is infinity without regard to how that occurred.
Zero: This exception is signalled whenever the result is zero without regard to how that occurred.

IV. EXPERIMENTAL RESULTS OF 1-D FFT USING RADIX-4 BE

The entire implementation of 1-D FFT using radix-4 butterfly element was done in VHDL with 6slx100fgg484-3 as target device.

1. RTL Schematic of individual blocks of 1-D FFT processor

Significance of RTL (Register transfer level) is explained in previous section. Individual RTL schematic of each block is given in figure 38. Detailed RTL of individual blocks are discussed in further section.

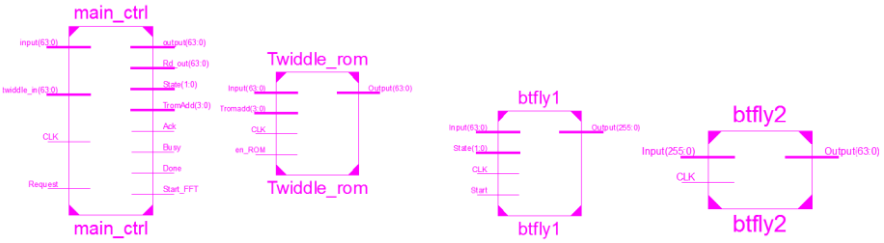

Figure 38 :RTL schematic of all blocks of 1-D FFT using Radix-4

2. RTL schematic of Master Control Unit and RAM

As shown, module main control unit consists of two sub modules Master Control Unit and RAM. It has four input signals clock, Request, input and twiddle_in out of which clock and request are reference signals and input and twiddle_in are of 64 bits i.e 16 hexadecimal bits. It has four control signals which are Start_FFT,

Busy, Done and Ack. Signals State is output signals consist of 2 bits and is used by SWAP unit. TromAdd is nothing but a twiddle addresses. Rd_out is a final output signal

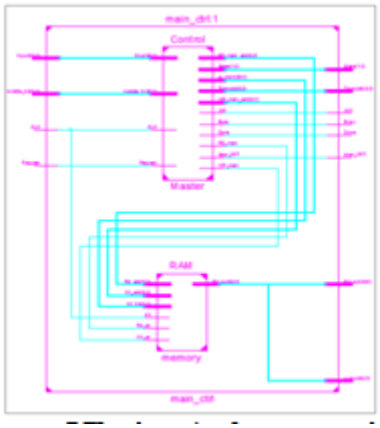

Figure 39 :RTL schematic of master control unit and RAM

3. RTL schematic Radix-4 Butterfly processing element

As shown in figure 40 and 41, for our convenience operation of butterfly processing element is divided into two parts butterfly element 1 and 2. As shown in figure 39 butterfly element 1 is consists of Switch, Three 64-bit registers, Three 256-bit registers, and Swap Unit.

It consists of input, output, clock, start and state signals. "State" is used throughout the data path, as a means to determine what calculation in the particular butterfly is to be performed. Output is of 256 bits, because it contains a packed form of the four 64-bit data points to be used by the Twiddle Unit for the butterfly calculation. Butterfly element 2 is consists of two adders. It has three signals input, output and clock. Final output will be addition of real and imaginary parts hence 64bits (32 bit for real and 32 bits for imaginary).

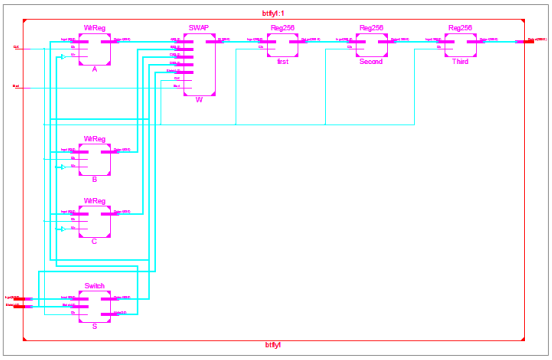

Figure 40 RTL schematic of Butterfly element 1

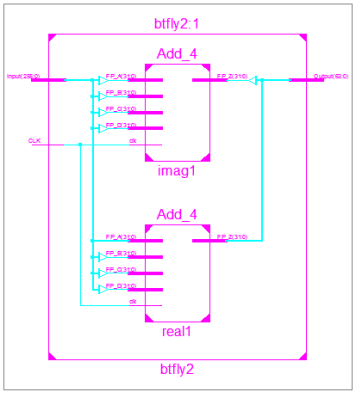

Figure 41 ;RTL schematic of butterfly element II

4. RTL schematic Twiddle ROM

As shown in figure 41, this module consists of two sub modules ROM and twiddle module. Twiddle module consists of floating point multiplier and floating point adders. ROM is nothing but a ROM memory

where pre-calculated twiddle factors are stored. It consists of 64 bit input and output, refrence signal clock. Signal tromadd which gives address twiddle factor require for FFT calculation from twiddle ROM. en_ROM which enables ROM to start FFT calculation.

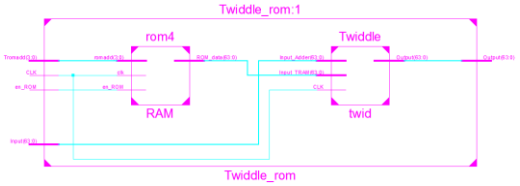

Figure 42:RTL schematic of twiddle ROM

5. RTL schematic 1-D FFT processor

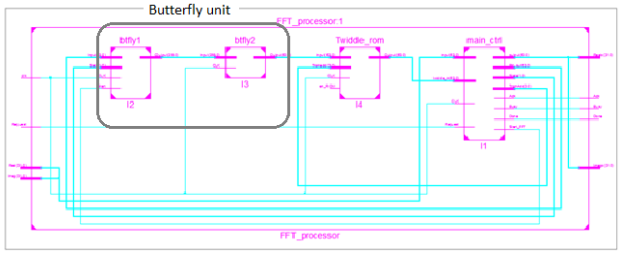

Figure 43 :RTL schematic of 1-D FFT using Radix-4 butterfly

As shown in figure 42, 1-D FFT processor consists of butterfly element, twiddle ROM and main control unit. It has two reference signals clock and request, when request comes then FFT starts taking inputs. It has three control signals Busy, Done and Ack, these signals indicates whether FFT is busy or it is done calculating.

B. Simulation results obtained for 1-D FFT processor using Radix-4 PE

Input sequence given are x(0) to x(15) each of 64 bit

x(0)=C1B2762CC235880C
x(1)=C22125B442016214
x(2)=419BDCE6C1925206
x(3)=423416BCC23A38B8
x(4)=C0C4049EC13D81A7
x(5)=41D469D641EC28F0
x(6)=C1FA807DBF830407
x(7)=C0AE1FC8416A19CF
x(8)=41A77DEA41CBBFD5
x(9)=C1B32E10418FC31B
x(10)=41782824C206F48E
x(11)=C21866A3BE278560
x(12)=4237E5C8C17F6201
x(13)=41086DABC1DCF350
x(14)=41C9037EC1C3EC85
x(15)=3F188022419F42E9

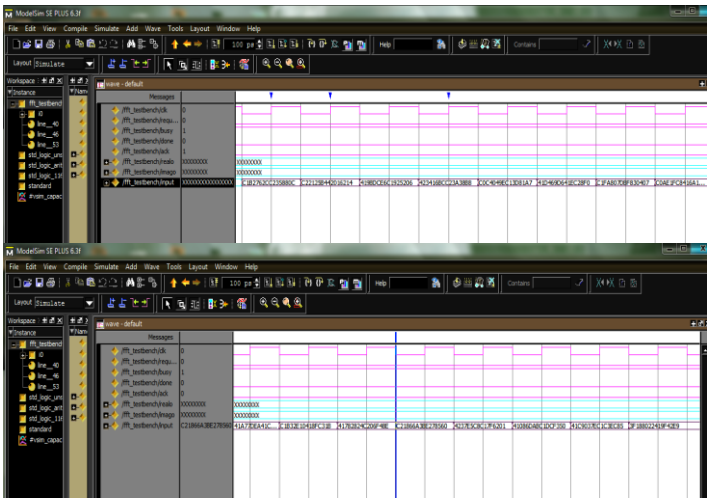

Figure 44: Input waveform of 1-D FFT using Radix-4 butterfly element

Output:

MATLAB output

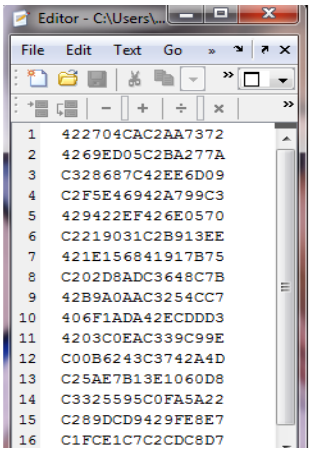

Figure 45 :MATLAB output for 1-D FFT processor

The output of 1-D FFT is as shown in figure 44 The input and output both are represented in hexadecimal format consists of 16 hexadecimal bits. Out of 16 bits first 8 bits represents the real part of the output and last 8 bits represents the imaginary part of the output. The output is compared with output of MATLAB. It can be seen from the figure that output of 1-D FFT processor designed by us exactly matches with the output of 1-D FFT processor which we have got through MATLAB.

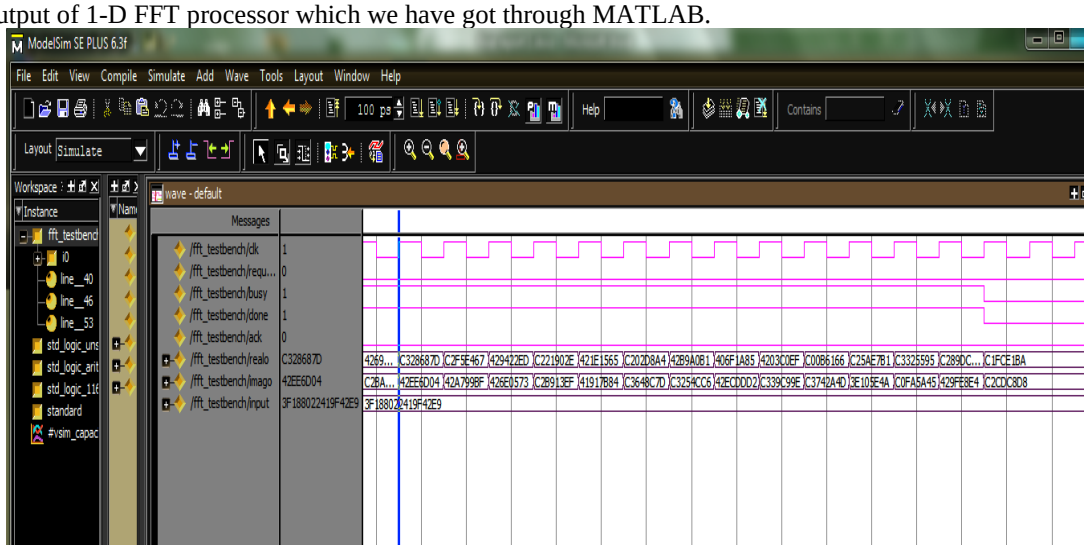

Figure 46 :Simulation Results of 16 point 1-D FFT using Radix-4

V. EXPERIMENTAL RESULTS AND COMPARISON OF RADIX-2 BE AND RADIX-4 BE

1. Comparison of Synthesis report summary of Radix-2 & Radix-4 1-D FFT processor

As the architecture is common for Radix-2 and Radix-4 FFT processor therefore the synthesis report generated for both the methods are similar.

TABLE:COMPARISON OF DEVICE UTILIZATION SUMMARY FOR RADIX-2, 1-D FFT AND RADIX-4, 1-D FFT

Logic Utilization	**Target device used : 6slx100fgg484-3**				
	Available	***Radix-2(1-D FFT)***		***Radix-4 (1-D FFT)***	
		Used	Utilization	Used	Utilization
Number of Slice Registers	126576	388	0%	4208	3%
Number of Slice LUTs	63288	3009	4%	6671	10%
Number of fully used LUT-FF pairs	3170	227	7%	2619	31%
Number of bonded IOBs	326	35	10%	133	40%
Number of Block RAM/FIFO	268	2	0%	1	0%
Number of BUFG/BUFGCTRLs	16	4	25%	1	6%

With respect to the comparison table VI, it can be concluded as follows

- Number of LUT's used double for Radix-4 with respect to Radix-2, Number of LUT FF pairs is 10 times in Radix-4 with respect to Radix-2.
- IOB's used is 4 times in Radix-4 with respect to Radix-2. Buffers and RAM used is reduced where as number of DSP's are 16 as compared to 1 DSP block used for radix-2.
- Number of Slice Registers is 4 times more for radix-4 as compared to radix-2 Whereas Numbers of Slice LUT's are two times more for Radix-4 as compared to Radix-2. Number of fully used LUT-FF pairs and Number of bonded IOBs are 4 times more in Radix-4.
- Both Number of Block RAM/FIFO and Number of BUFG/BUFGCTRLs are reduced in Radix-4 as compared to Radix-2. DSP blocks usage increased by 8 times in Radix-4 2-D FFT.

2. Comparison of Timing summary for 1-D FFT processor's using Radix-2 and Radix-4 butterfly elements

As depicted in Table VII, the following observations are made.

- Maximum output required time after clock is found to reduces from 5.037ns to 3.732ns in Radix-4 FFT.
- Total REAL time to synthesis (Xst) completion of Radix-4 FFT is 1.5 times of Radix-2.
- Total memory usage is 1.33 times more in Radix-4 as compared to Radix-2.

TABLE;COMPARISON OF TIMING SUMMARY FOR 1-D FFT PROCESSOR'S USING RADIX-2 AND RADIX-4 BUTTERFLY ELEMENTS

Timing summary	**1-D FFT**	
	Radix-2	***Radix-4***
Speed grade	-3	
Minimum Period/ Maximum frequency	12.244ns/ 81.67MHz	14.264ns/ 70.108MHz
Maximum o/p required time after clock	3.597ns	5.037ns
Minimum i/p arrival time before clock	14.040ns	3.151ns
Total REAL time to Xst completion	57.00 sec	105.00 sec
Total CPU time to Xst completion	56.82 sec	104.16 sec
Total memory usage in kilobytes	311876	3133220
Selected Device	6slx100fgg484-3	

VI. CONCLUSION

In this paper, FPGA-based architecture is reported for performing a parallel 1-D FFT. In this design, for both the Radix-2 and Radix-4 1-D FFT processor use of single butterfly element is demonstrated, in contrast with the usual approach where multiple butterfly elements are used for computing FFT without considering area and power consumption. The key design decision is to implement 1-D FFT, so as to increase speed and efficiency of FFT as much as possible. With the results from table I and II, the conclusion is that 1-D FFT processor using Radix-2 butterfly element requires less number of slice registers, slice LUTs and fully used LUT-FF pairs as compared to Radix-4 butterfly element. Utilization of DSP48Es is almost negligible for 1-D FFT processor which uses Radix-2 butterfly element than Radix-4 butterfly element. It is consistently seen in

Table VII that, the radix-4 is more efficient algorithm in terms of computation time. If the choice of algorithm is to be made solely based on memory usage and area with respect to number of slice register used, number of slice LUT's and LUT's FF, the Radix-2 algorithm is better. Minimum input arrival time is 1.3 times for Radix-2 FFT compared to radix-4 FFT. Thus it can be concluded that area utilization and power consumption is less for Radix-2 butterfly element FFT processor which uses than Radix-4 butterfly element. RAM memory requirement of designed Radix-2 FFT processors is also very less reducing power consumption further.

VII. FUTURE SCOPE

The future scope for this work is implementation of FFT architecture using higher-order Radix for the FFT, as small data samples are unusual on real-world applications. This may reduce the resource usage on the FPGA, so that the available space could be used to accommodate more computing cores, leading to new alternatives for the parallel algorithms. Also use of more than one butterfly processing element is recommended for faster processing. Most of the cells used to build the FFT processor have been optimized for speed rather than area and power consumption. These blocks can be redesigned for reduced area and power consumption

REFERENCES

[1]. Ediz Çetin, Richard C. S. Morling and Izzet Kale, **"An Integrated 256-point Complex FFT Processor for Real-time Spectrum Analysis and Measurement",** IEEE Proceedings of Instrumentation and Measurement Technology Conference, vol. 1, pp. 96-101, May 1997

[2]. Thomas lenart and Viktor Owall "Architecture for dynamic data scaling in 2/4/8K pipeline FFT cores", IEEE transaction on very large scale integration systems, Vol.12, NO.11 November 2006.

[3]. Erling H. Wold, Alvin M. Despain, "**Pipeline and Parallel-Pipeline FFT Processorsfor VLSI Implementations**", IEEE transactions on computers, Vol. c-33, No. 5, May 1984

[4]. K.Sreekanth Yadav, V.Charishma, Neelima koppala, "**Design and simulation of 64 point FFT using Radix 4 algorithm for FPGA Implementation**", International Journal of Engineering Trends and Technology, Volume-4, Issue-2, 2013

[5]. Markus Puschel, Martin Rotteler, "Cooley-Tukey FFT like algorithm for the discrete traingle transform", IEEE 11th Digital Signal Processing Workshop & IEEE Signal Processing Education Workshop, 2004.I.S.Uzun, A.Amira and A. Bouridane, "FPGA implementations of fast Fourier transforms for real-time signal and image processing", IEEE Proc. Image signal Process, vol. 152, no. 3, pp. 283–296, Jun. 2005.

[6]. B1. Proakis,J.G., Manolakis, D.G., " Digital Signal Processing" 3rd Edition, PHI Publication 2004

[7]. B2. Mitra, S. K., "Digital Signal Processing" 3rd Edition, Tata Mc. Graw Hill Publications

[8]. B3. Capman, S.J., "MATLAB Programming for Engineers", 3rd Edition, Thomson learning 2005.

Ms. Ridhima Vijay Benadikar "Comparative Study Of Fpga Implementation Of Parallel 1-D Fft Algorithm Using Radix-2 And Radix-4 Butterfly Elements "IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) , vol. 8, no. 4, 2018, pp. 23-47