Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 6.017

IJCSMC, Vol. 5, Issue. 11, November 2016, pg.133 – 142

Design and Simulation of 32 and 64 Point FFT Using Multiple Radix Algorithm

Miss. Priyanka R. Bhonde[1], Prof. R. D. Ghongade[2], Prof. R. D. Sushir[3]

[1]Electronic and Telecommunication from SGB Amravati University, India
[2]Electronic and Telecommunication from SGB Amravati University, India
[3]Electronic and Telecommunication from SGB Amravati University, India

[1] priyankabhonde69272@gmail.com; [2] rahulghongade@rediffmail.com; [3] rupeshsushir@gmail.com

Abstract— Always technical designers choice includes algorithms, flowcharts, programming etc. and at the end users requires the given input and application output. Based upon this view, this project focus on the advancement of the Fast Fourier Transform (FFT), by doing design and observing the simulated waveform of the 32 point FFT and 64 point FFT using MULTIPAL RADIX algorithm.
Fast Fourier Transform is an algorithm used to compute Discrete Fourier Transform (DFT) of a finite series. This project proposes the design of 32 and 64 point FFT using MULTIPAL RADIX Algorithm and it concentrate on Decimation-In-Time Domain (DIT) of the Fast Fourier Transform (FFT). Here I use Xilinx Design Suite 14.7 version, by using VHDL as a design entity and the stimulation result are stimulated on Model SIM. FFT computation technique is used in wide range of its Mathematics, Auto Correlation, Data Compression, Pattern Recognition etc. The simulation results shows the comparison of 32 and 64 point FFT in terms of speed and computational complexity.

Keywords— Fast Fourier Transform (FFT), Decimation-In-Time(DIT-FFT),Discrete Fourier Transform (DFT), Radix-2, Radix-4, Radix-8, VHDL, Twiddle factor.

I. INTRODUCTION

New techniques for digital signal processing have been developed to meet the increasing demands for higher speed in communications which can be used in both FPGA and OFDM environments. To meet out the high speed, less hardware, an efficient Radix scheme is to be employed. A promising Radix technique that is increasingly being adopted in the digital signal processing field for Field Programmable Gate Array (FPGA) and Orthogonal Frequency Division Multiplexing (OFDM) implementation. Orthogonal Frequency Division Multiplexing (OFDM) is a multi-carrier modulation technique in which a single high rate data-stream is divided into multiple low rate data-streams and is modulated using sub-carriers which are orthogonal to each other [19].

Frequency analysis of discrete signal can be conveniently performed on digital signal processors. In order to perform such an analysis one has to transform the signal from time domain to frequency domain representation. FFT is itself not a transformation but just a computational algorithm to evaluate Discrete Fourier Transform (DFT). Fast Fourier

transform (FFT) algorithms are computationally efficient algorithms that exploit these properties of Twiddle factor. It computes the DFT of N number of discrete data samples in (N log2N) time as opposed to (N^2) in the direct method. Discrete Fourier Transform (DFT) is of much importance in fields of signal processing. DFT is the most widely used transform of all the available transforms in digital signal processing. The DFT maps the input sequence X(n) into frequency domain.

II. LITERATURE SURVEY

The major obstacle in using the MRA is the high complexity. To reduce this complexity several methods have been previously developed, but before discussing these methods I will mention the papers published in various journals which adopts MRA as a design technique. In 2003 Wen-Chang Yeh and Chein-Wei Jen published the paper" High-Speed and Low-Power Split-Radix FFT " in the 'Signal Processing journal, IEEE, Volume 51, NO. 3, PP. 866-874. in which they optimized the general size memory FFT processer design. It sustain the multibank addressing for a high-radix structure without memory. In this project they used a low complexity index vector generator[2].

Radix-4 algorithm was first described in an initially little-appreciated paper publish" Design of 16-point Radix-4 Fast Fourier Transform in 0.18µm CMOS Technology " by Siva Kumar Palaniappan, Tun Zainal Azni Zulkifli in American Journal of Applied Sciences Volume 4 Issue8 PP. 570-575, design 16 point with CMOS technology and conclude it save 1.73% of power also reducer 55% of area. For that they use better and efficient architecture[7]. In the year 2014, authors V. Venkata, Lakshmi Dadal Naresh, R. Anil Kumar presented a paper titled as "Butterfly Design for Radix-4k DIT FFT " IJRCCT Volume 3, Issue 10, PP. 1348-1353, they projected that this paper reduced the computation and improve the speed of system. comparing with Radix 2 algorithm the Radix-4 save 75% of time[3]. Next in the year authors Asmita Haveliya [8] published the paper titled "Design and Simulation of 32-Point FFT Using Radix-2 Algorithm for FPGA Implementation", in Second International Conference on Advanced Computing & Communication Technologies PP. 168-171, 2012. . In this paper, they use radix-2 algorithm. The proposed FFT block of signal length 32 is been simulated and synthesized. The output is available in two form that is real and imaginary . The authors use Vertex 6 device family**,** XC6VLX75TL device FF484 Package, -1L Speed Grade. Minimum delay is 30.783ns , Total REAL time to XST completion: 1611.00 secs, Total CPU time to XST completion: 1611.39 secs, Total memory usage is 1112720 kilobytes[8].

The 32 point radix-2FFT for FPGA implementation is carried out in XILINK/SIMILINK by Kasina Madhusudhana Rao, V. Ravi Tejesvi, Anantha Rao, paper titled as " Verilog Implementation of 32 Point FFT Using Radix-2 Algorithm on FPGA Technology", (IOSR-JECE) Volume 9, Issue 1, VER. II, PP. 40-43, Jan. 2014. This paper introduces the butterfly structure and operational principle of radix-2 for 32 point FFT[9].

III. PROPOSED METHODOLOGY

In this project, we are going to implement the multiple Radix architecture for 32 and 64 point FFT. Also to increase the throughput of system i.e. High speed design of multiple Radix FFT architecture for 32 and 64 point, we are going to implement the pipelined FFT concept.

A. *MULTIPLE RADIX ALGORITHMS*

Multiple radix algorithm many radix algorithm such as radix-2, radix-4, radix-8 etc. Any radix algorithm is defined by their base i.e. if base is equals to 2 then it is known as radix-2, if base is equals to 4 then it is known as radix-4 and if base is equals to 8 then it is known as radix-8 algorithm. It represents by 2M where M represents the index/stage and its value is a positive integer. The base is decided the number of input and output to the system. The radix-2, radix-4, radix-8 have 2,4,8 input and output respectively. The computation of radix made

up of butterflies called Radix butterflies. E.g. The computation of radix-N made up of butterflies called Radix-N butterflies. The proposed system is based on multiple radix algorithms.

B. RADIX-2

Radix algorithm is defined by their base i.e. if base is equals to 2 then it is known as radix-2. The Radix-2 algorithm decompose an N-point DFT into four (N/2)-point DFT. Radix-2 algorithm requires Double stages as radix-4 requires. No. of stages requires in radix 2 are log2N. N/4 butterflies are used in each of (log2N)/2 stages, which is one quarter the number of butterflies in a radix-2 FFT. The radix-4 butterfly is consequently larger and more complicated than a radix-2 butterfly.

The name Radix-2[1][2] is called due to its base is equals to 2 and the representation is 2M, where M represents the index/stage and its value is always a positive integer. The representation of DITFFT is as follows. In Radix-2 algorithm the DFT computation sequence is split into even and odd-half parts. The Radix-2 DITFFT is derived by rewriting the equation shown bellow.

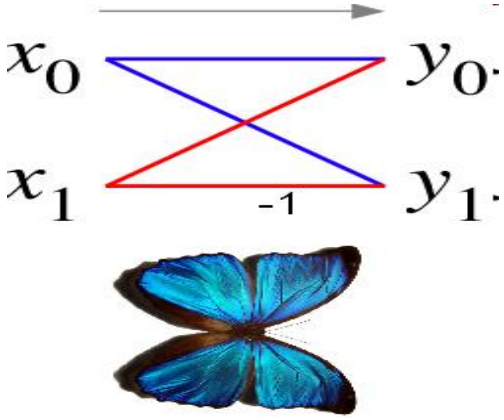


Figure:1 Butterfly Diagram

C. RADIX-4

The radix algorithm is which have base equal to 4 called as radix-4. Radix-4 have (3N/8)· Log2N complex multiplication and (3N/2) · log2N complex addition. The Radix-4 algorithm decompose an N-point DFT into four (N/4)-point DFT. Radix-4 algorithm requires only half as many stages as radix-2 requires. No. of stages in radix 4 are log4N. N/4 butterflies are used in each of (log2N)/2 stages, which is one quarter the number of butterflies in a radix-2 FFT. The radix-4 butterfly is consequently larger and more complicated than a radix-2 butterfly. Addressing of data and twiddle factors is more complex. Radix-4 FFT requires fewer calculations than a radix-2 FFT. Radix-4 FFT is significantly faster than radix-2 FFT. A radix-4 FFT combines two stages of a radix-2 FFT into one, so that half as many stages are required. The overall number of operations is lower.

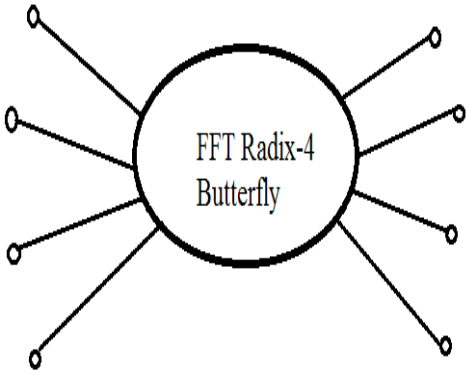


Figure 2:- Basic structure of Radix-4 FFT

D. RADIX-8

The Another type of Radix algorithm is Radix-8 which have advantages over Radix-2 and Radix-4. Radix-8[7][17] is another FFT algorithm which was surveyed to improve the speed of functioning by reducing the computation; this can be obtained by changing to base to 8.For a same number if base increases the power will reduce. In this project the number of stages are reduced to 75% since $N=8^2$ ($N=8^M$) that is; only 2 stages. The following will explains the functioning of Radix-8 and how the computational complexity is reduced. By using the FFT algorithm the computational complexity reduces to , where r represents the Radix-r FFT[2]. The Radix-r FFT can easily derived from DFT by decomposing the N point DFT into a set of recursively related r-point transform and x(n) is powers of r. In Radix-8 algorithm the r is 8.The DIT Radix-8 FFT recursively partitions a DFT into eight quarter-length DFTs of groups of every eighth sample. The outputs of these shorter FFTs are reused to compute many outputs, which greatly reduce the total computational cost.

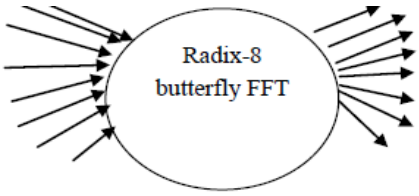


Figure 3:- Basic structure of Radix-4 FFT

IV. SIMULATION RESULTS

The 32 point radix-4 DIT FFT and 64 point radix-8 DIT FFT 16 bit are synthesized in Spartan 3E starter board as the evaluation development board. The family is Spartan 3E the device used is xc3s50E , the package isPQ208 and the speed is -5. The top level source is HDL, the synthesis tool is XST (VHDL), the simulator is ISIM (VHDL). For the 32 point the 16 bit binary numbers in x r[31:0] and x i[31:0] are given as the input of real and imaginary parts and for the 64 point the 16 bit binary numbers in x r[63:0] and x i[63:0] are given as the input of real and imaginary parts after the transformation using VHDL coding the output of real and imaginary part are obtain in y r[63:0] and y i[63:0]. The transformation is done with the help of butterfly diagram. The result obtained after simulation in the Isim window is shown figure 4

oldthesis Project Status (11/10/2016 - 12:11:29)			
Project File:	thesisoid.xise	**Parser Errors:**	No Errors
Module Name:	E_FFT_32	**Implementation State:**	Synthesized
Target Device:	xc3s50-5pq208	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	1031 Warnings (1031 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values) [-]			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	0	768	0%
Number of bonded IOBs	2050	124	1653%

Detailed Reports [-]					
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Thu Nov 10 12:11:28 2016	0	1031 Warnings (1031 new)	0
Translation Report					
Map Report					
Place and Route Report					

Figure 4:hardware utilization of 32 point DITFFT by Radix-4

```
   Source:            STAGE5:output11_real<15> (PAD)
   Destination:       output11_real<15> (PAD)

   Data Path: STAGE5:output11_real<15> to output11_real<15>
                                 Gate     Net
     Cell:in->out      fanout   Delay    Delay  Logical Name (Net Name)
     ----------------------------------------  ------------
     E_FFT16:output11_real<15>    1    0.000   0.681  STAGE5 (output11_real_15_OBUF)
      OBUF:I->O                   4.909          output11_real_15_OBUF (output11_real<15>
     ----------------------------------------
     Total                       5.590ns (4.909ns logic, 0.681ns route)
                                         (87.8% logic, 12.2% route)

=========================================================================


Total REAL time to Xst completion: 19.00 secs
Total CPU time to Xst completion: 18.18 secs

-->

Total memory usage is 280360 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings : 1031 (   0 filtered)
Number of infos    :    0 (   0 filtered)
```

Figure5: Delay time of 32 point DITFFT by Radix-4

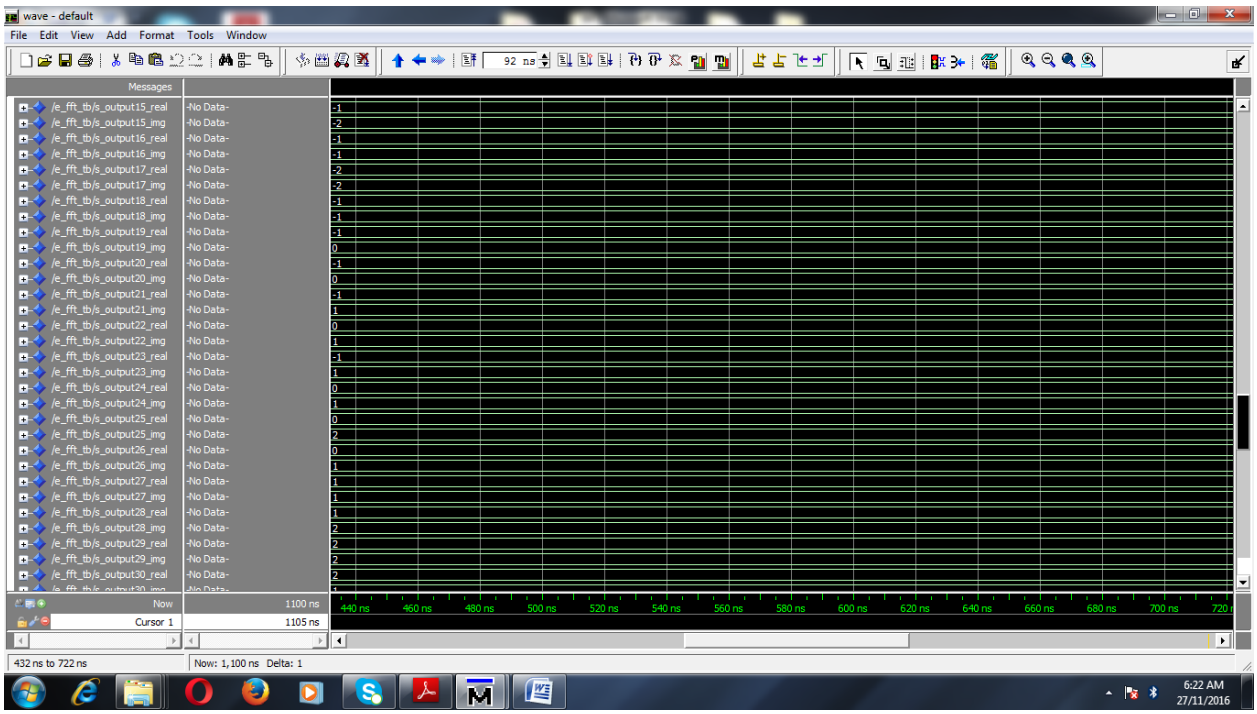


Figure 4.8:- Simulation of 32 point FFT cont...

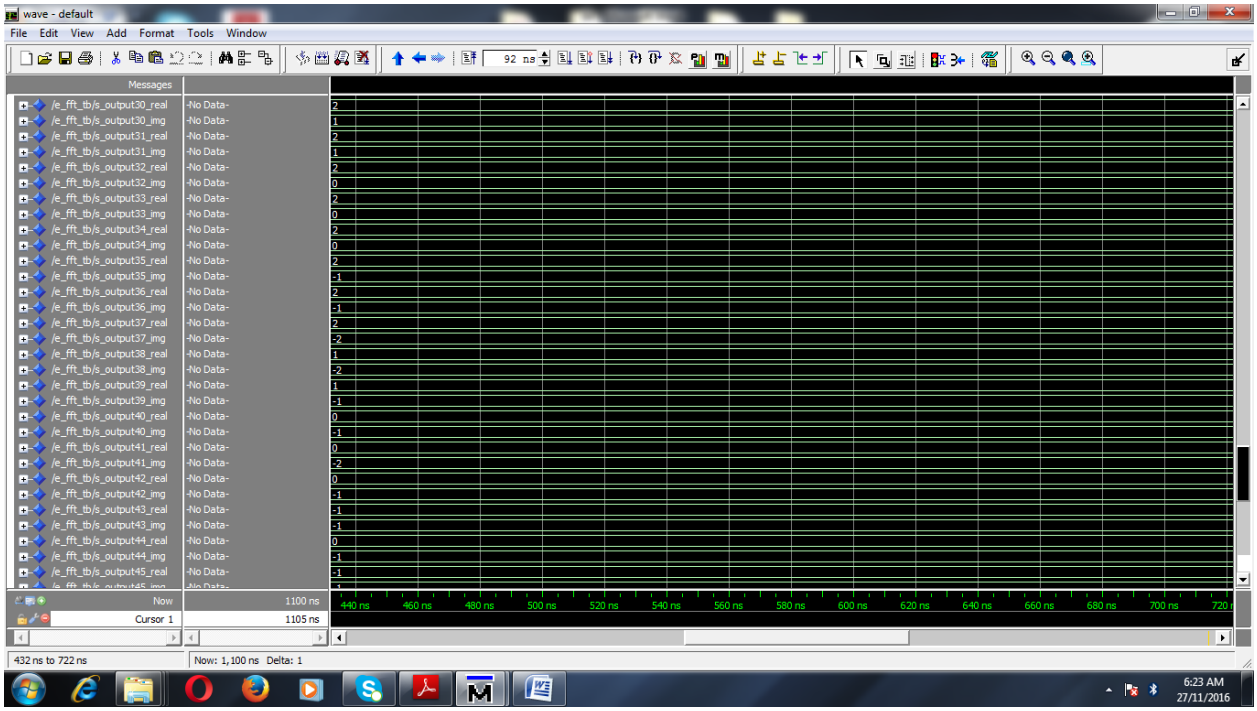


Figure 4.9:- Simulation of 32 point FFT cont...

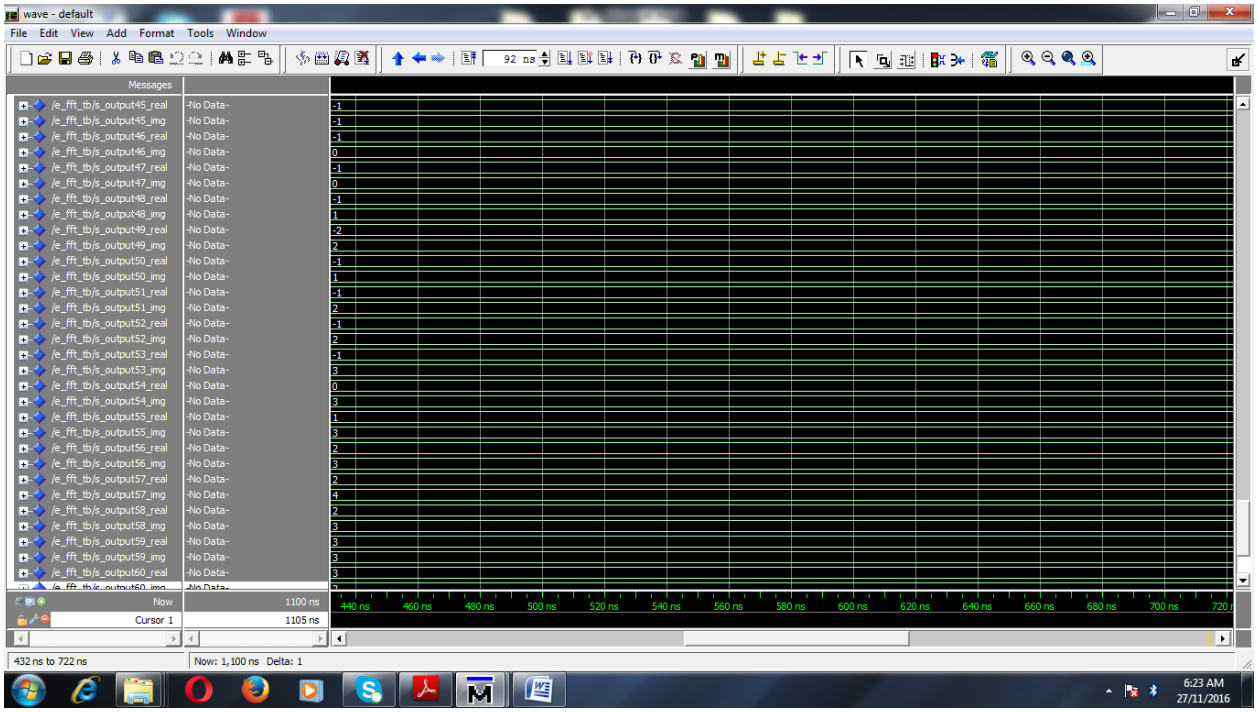


Figure 4.10:- Simulation of 32 point FFT cont...

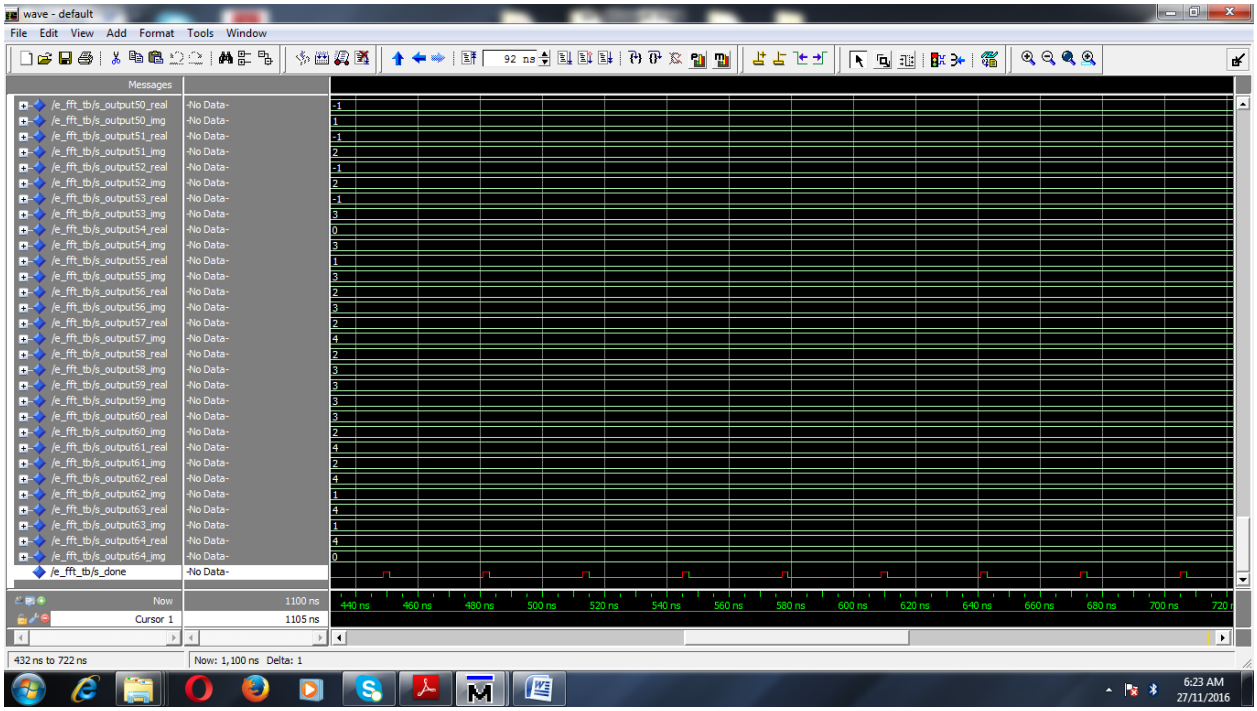


Figure 4.11:- Simulation of 32 point FFT

Table1: comparisons of 32 point FFFT

	32 Point DITFFT By multiple Radix								
	Radix-2			Radix-4			Split-Radix		
Logic Utilization	Available	used	Utilization	Available	Used	Utilization	Available	used	Utilization
No. of Slice LUTs	46560	18253	39%	768	0	0 %	46560	22120	47%
No. of Fully Used LUTFFT	18253	0	0%	----	---	---	22120	0	0 %
No. of Bonded IOBs	240	2560	1066%	124	2050	1653%	240	2560	1066%
No. of DSP48 EIs	288	148	51	---	---	----	288	160	55%

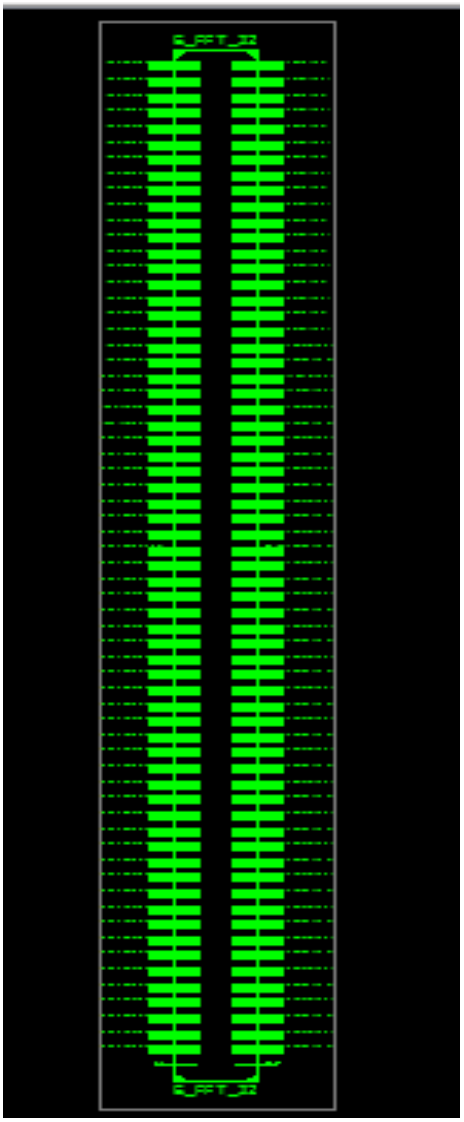


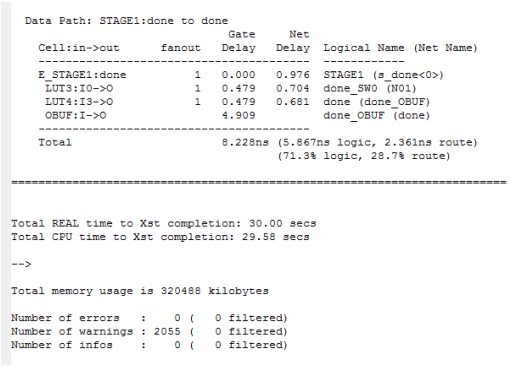
Figure 6 :- RTL View of 32 point DITFFT by Radix-4

oldsynopsis Project Status (11/10/2016 - 12:30:13)			
Project File:	oldsyno.xise	**Parser Errors:**	No Errors
Module Name:	E_FFT_64	**Implementation State:**	Synthesized
Target Device:	xc3s50-5pq208	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	2055 Warnings (2055 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values) [-]			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	1	768	0%
Number of 4 input LUTs	2	1536	0%
Number of bonded IOBs	4098	124	3304%

Detailed Reports [-]					
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Thu Nov 10 12:30:12 2016	0	2055 Warnings (2055 new)	0
Translation Report					
Map Report					

Figure 7 Hardware utilization of 64 point DITFFT by Radix-8



```
Data Path: STAGE1:done to done
                                   Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
  E_STAGE1:done          1   0.000   0.976  STAGE1 (s_done<0>)
   LUT3:I0->O            1   0.479   0.704  done_SW0 (N01)
   LUT4:I3->O            1   0.479   0.681  done (done_OBUF)
   OBUF:I->O                 4.909          done_OBUF (done)
  ----------------------------------------
  Total                      8.228ns (5.867ns logic, 2.361ns route)
                                     (71.3% logic, 28.7% route)

=========================================================================


Total REAL time to Xst completion: 30.00 secs
Total CPU time to Xst completion: 29.58 secs

-->

Total memory usage is 320488 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings : 2055 (   0 filtered)
Number of infos    :    0 (   0 filtered)
```

Figure 8 Delay time of 64 point DITFFT by Radix-8

V. CONCLUSIONS

In this Project, the design of 32 and 64 point FFT using Radix-4 and Radix-8 algorithms are performed, and the performance analysis with all the three algorithms are done using Minimum Delay (ns) as parameter and their simulation results are shown by Xilinx synthesis tool .The test bench wave forms are displayed by using Xilinx ISE Design Suite 14.7. Further, the performance analysis can also be done by taking various parameters into consideration for different or same number of points.

ACKNOWLEDGEMENTS

We thankful to incalculably our management for outspreading their support in providing us substructure and allowing us to use them in the successful completion of our research paper.

REFERENCES

[1] Chen-Fong Hsian, Yaun Chen, Chen-Yi Lee, "A Generalized Mixed- Radix Algorithm For Memory Based FFT Processer", IEEE Trans. Circuit System II, express briefs, Volume 57 no.1, PP. 26-30, January 2010.

[2] Wen-Chang Yeh, Chein-Wei Jen, "High-Speed and Low-Power Split-Radix FFT ", IEEE Transactions On Signal Processing, Volume 51, NO. 3, PP. 866-874, March 2003.

[3] V. Venkata Lakshmi Dadala,CH. Satya Naresh, R. Anil Kumar "Butterfly Design for Radix-4k DIT FFT", Volume 3, Issue 10, PP. 1348-1353 October - 2014.

[4] K. Sarath Chandra Varma, Ch. Kranthi Kumar, K. Sravan kumar, D. Sharat Chandra varma, N. Rajasekha , " Design and Simulation of 64-Point FFT Using Radix-4 Algorithm for OFDM Applications " ,Volume 10, Issue 2, PP. 35-40, Mar - Apr. 2015.

[5] K. Sreekanth Yadav, V. Charishma, Neelima koppal, "Design And Simulation Of 64 Point FFT Using Radix- 4 Algorithm For FPGA Implementation", Volume4 Issue2 pp.109-113, 2013.

[6] K. Sowjanya, B. Leele Kumari "Design And Performance Analysis Of 32 And 64 Point FFT Using Radix-2 Algorithm", AECE-IRAJ International Conference", PP. 55-58, 14th July 2013.

[7] Siva Kumar Palaniappan, Tun Zainal Azni Zulkifli "Design of 16-point Radix-4 Fast Fourier Transform in 0.18µm CMOS Technology" Volume 4 Issue8 PP. 570-575, 2007.

[8] Asmita Haveliya "Design and Simulation of 32-Point FFT Using Radix-2 Algorithm for FPGA Implementation" Second International Conference on Advanced Computing & Communication Technologies PP. 168-171, 2012.

[9] Kasina Madhusudhana Rao, V. Ravi Tejesvi, AnanthaRao," Verilog Implementation of 32 Point FFT Using Radix-2 Algorithm on FPGA Technology", (IOSR-JECE) Volume 9, Issue 1, Ver. II, PP. 40-43, Jan. 2014.

[10] Soda Karan G, Brundavani P, "FPGA Implementation of 256-Bit, 64-Point DIT-FFT Using Radix-4 Algorithm" Volume 3, Issue 9, PP. 127-133, September 2013.

[11] Remya Ramachandran, Vanmathi k. "Simulation Of Radix-2 Fast Fourier Transform Using Xilinx", (IJCSE) Volume 3 No.02 PP. 125-130, Mar 2014.

[12] K. Sowjanya,, Leela Kumari Balivada " Design and Performance Analysis of 32 and 64 Point FFT using Multiple Radix Algorithms " ,Volume78– No.1, PP. 25-29,September 2013.

[13] Amaresh Kumar, U. N. Tripathi , Roopak Kumar Verma, Manish Mishra "64 Point Radix-4 FFT Butterfly Realization Using FPGA" (IJEIT) Volume 4, Issue 4, PP.57-60, October 2014.

[14] Shruti Khandelwal Pankaj Gulhane "Design Of Radix-4 FFT In VHDL Using Simulink", Volume 2 Issue 2, PP. 49-56,Ferbruary, 2013.

