Design and verification of high speed 2^2 radix FFT module for DSP processor

Astha singh, SRIT, Jabalpur
Prof. Dewyanshu Rao, SRIT, Jabalpur

Abstract: The FFT processor design with vedic multiplier and new semi-pipelined Fast Fourier transform (SPFFT) with modified multiplication arrangement for modern communication systems provides an efficient way for computation of FFT with better area utilizing hardware architecture. Previously, the radix-2^2 had been used only for single path delay feedback architectures. Later with many types of research works the radix 2^2 was extended to multi-path delay commutator (MDC) architectures. This work presents area optimization of SPFFT architecture. This architecture is provided for parallelism value 4 and 16 sample points and the area of proposed SPFFT is compared with other SPFFT (feed forward) architectures using the same synthesis tool and FPGA.

Keywords: HDL- Hardware Descriptive Language, SOC- System On Chip,FPGA- Field Programmable Gate Array, RTL- Register Transfer Level

I-INTRODUCTION

The Fast Fourier Transform (FFT) is a widely used transform algorithm in signal processing applications, which is primarily a computational tool, used to efficiently calculate the Discrete Fourier transform (DFT) and its inverse using digital computers. Since its introduction by Cooley and Tukey [1], FFT has been the mainstay for spectral analysis of digital signals. Spectral analysis is extensively used in communication systems, signal processing, image processing, bio-robotics, intelligent maintenance and almost every branch of science and engineering [2–4], making FFT one of the most widely used algorithms on digital devices. With the advent of smart phones and hand held media and entertainment devices, the performance and cost of FFT processors has an ever greater significance. With the remarkable progress in the very large scale integration (VLSI) circuit technology, many complex circuits, unthinkable yesterday have become easily realizable today. Algorithms that seemed impossible to implement, now have attractive implementation possibilities for the future. This means that not only the conventional computer arithmetic methods, but also the unconventional ones are worth investigation in new designs.

Author	Topic	Outcome
Ajmal S A et al [1]	FPGA based area optimized parallel pipelined Radix-22 feed forward FFT architecture	197 slice used of vertex FPGA with their new parallel pipelined radix-22 feed forward Fast Fourier transform (PPFFT) architecture
Ngoc-Hung Nguyen et al [2]	An FPGA-Based Implementation of a Pipelined FFT Processor for High-Speed Signal Processing Applications	50, 866 slices for 1024-point FFT with their pipelined FFT processor based on the radix-2 decimation-in-frequency (R2DIF) algorithm using the single-path delay feedback (SDF) pipelined architecture
Asmita Haveliya et al [3]	Design and Simulation of 32-Point FFT Using Radix-2 Algorithm for FPGA Implementation	2,016 slices for 32 points FFT, Fast Fourier Transform (FFT), based on Decimation-In-Time (DIT) domain, Radix-2 algorithm,

Table 1 literature work summary

The FFT computation which we are modifying in our thesis work actually consists of complex multiplication. Along with this when FFT computations are carried out it shows that it needs 4 multipliers along with 1 adder and 1 subtractor. Vedic multiplication is although one of the fastest methods of multiplication but it consumes much area which turns out to be a drawback.

Asmita Haveliya et al [3] use Fast Fourier Transform (FFT), based on Decimation-In-Time (DIT) however DIF is more suitable and need less computation. Ngoc-Hung Nguyen et al [2] use pipelined FFT processor based on the radix-2 decimation-in-frequency (R2DIF) algorithm using the single-path delay feedback (SDF) pipelined architecture, their method was significantly good but single path feedback need extra time for analysis, yes they save the time using pipeline but feedback need extra time which reduces overall timing performance as can be expected by any pipeline architecture. Ajmal S A et al [1] Use parallel pipelined radix-22 feed forward Fast Fourier transform (PPFFT) architecture, this type of architecture are very good for speed enhancement but on the other hand due to extra pipelining this type of architecture requires lots of addition registers which increase the area demand and cost also the power consumption. They did not concern about the power and area need.

These multiplications addition and subtraction are time and area consuming processes in it. Hence we have come out with a new idea for eliminating this problem. For this we have merged two types of FFT's, namely Area optimized and Pipelined FFT and proposed a new FFT called FSM based FFT. Here we have replaced one multiplier out of the 4 as explained above with one adder and one subtraction. So now we have total three multiplier, two adders and three subtractions. Along with this we have also added a new approach of addition which is called Wallace method of addition. This method helps in giving the results in much less time and hence helps in speed enhancement. How this will affect the area and speed is explained in detail in methodology but this approach is a good deal to deal with area and speed problem.

II-METHODOLOGY

The FFT processor design with vedic multiplier and new semi-pipelined Fast Fourier transform (SPFFT) with modified multiplication arrangement for modern communication systems provides an efficient way for computation of FFT with better area utilizing hardware architecture. Previously, the radix-2^2 had been used only for single path delay feedback architectures. Later with many types of research works the radix 2^2 was extended to multi-path delay commutator (MDC) architectures. This work presents area optimization of SPFFT architecture. This architecture is provided for parallelism value 4 and 16 sample points and the area of proposed SPFFT is compared with other SPFFT (feed forward) architectures using the same synthesis tool and FPGA.

We are proposing a new approach to design FFT algorithm for that we made significant logics on the basic butterfly and reduced the total number of multiplications, as we knew that basically we need total four digital multiplications, one addition and one subtraction for performing multiplication between two complex number's, our method has reduced these four digital multiplications into only three digital multiplications on the cost of one extra addition and two extra subtractions, still it is a fair deal if we concern about area and speed. We also have proposal to have different categories of butterflies and FSM based calculation to manage these categories.

It is been proved that Vedic multiplication is the fastest multiplication approach but there are some other multiplication techniques which are better are better than Vedic multiplication in terms of chip area. We have come up with the idea to merge two different multiplication techniques Vedic and Wallace and these gives us a fast and area efficient multiplication approach. Our proposed FFT algorithm is already efficient in respect of area and speed, all we need to optimize it further is to use our proposed Vedic cum Wallace multiplication technique on proposed FFT algorithm.

DIF-FFT algorithm: The basic DIF butterfly is explained in figure 2, here we have two inputs A & B and one twiddle factor W. The outputs are (A+B) and (A-B)*W.

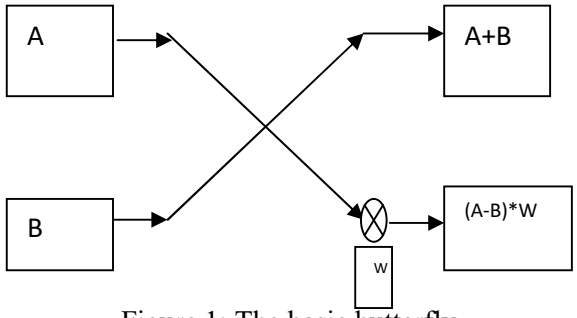

Figure 1: The basic butterfly

Let say we have radix two & 8 point FFT. Figure 6 explains the process of 3 stage pipelining. Figure explained pipelined FFT where they achieved 3 stages pipelining for 8 point FFT between different stages, as we knew pipelining makes fast computation so ref. [2] has achieved fast FFT using pipelining.

In an 8 point FFT processor without pipelining it takes four clock cycles to generate the output for one set of input and in the next clock cycle the next set of input is applied for processing. Where as in pipelined FFT processor as soon as one set of input is transferred to second stage a new set of input is applied to the first stage during the same clock cycle, due to which number of clock cycles is reduced.

Proposed Memory based FFT: Let say we have radix two & N point FFT. We knew that each stage required N/2 butterflies. Also we aware that total m*(N/2) butterflies where m is the total stages. Memory based FFT[1] shown in figure 8, suggest if we have the combination of N/2 butterflies then we can perform the FFT operations in 'm' number of cycles.

This approach required a complex Control Unit for proper arranging the output from the previous stage and feed that into the next stage. It is also known as feedback algorithm.

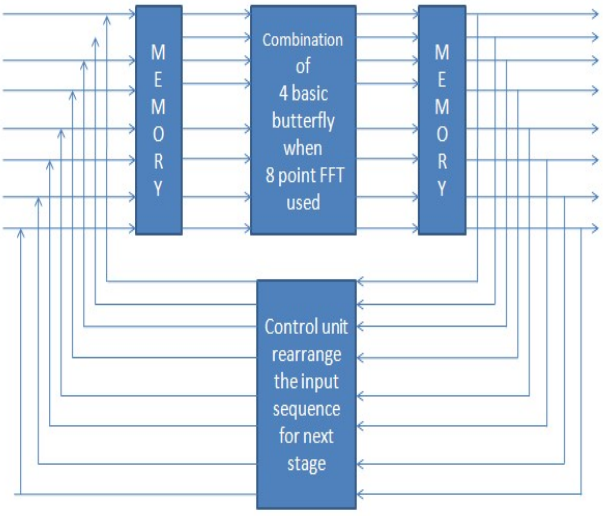

Figure 2: Memory based FFT

Pipelined FFT [2] suggests pipelining but no hardware reduction as compare to basic conventional one. It uses pipelining and complete hardware that is 12 butterflies. It is fastest with more hardware. Area efficient FFT [3] suggest us to use one basic butterfly and reuse it 12 times, but there we suppose to wait 12 clock cycles for the output and then-after new input can be given to it. It is slowest but need less hardware. Memory based FFT[1] use four butterflies three time means need hardware of four butter fly but need only three clock to have desired output. Memory based FFT is have somewhat in between Area efficient FFT and Pipelined FFT it is faster than Area efficient FFT and required less hardware then Pipelined FFT. We are using the same approach that Memory based FFT [1] has used.

But our designed basic butterfly has two major difference over Memory based FFT. That is:-

New complex multiplication technique approach. Combination of four basic butterfly and FSM based Control unit to control them.

Complex multiplication: Consider a input Z1=(x1+iy1) and Z2=(x2 + iy2). If we take simple multiplication of these inputs they give the outputs as-

Real Part(R) = (x1x2-y1y2) & Imaginary Part(I) = (x1y2+y1x2)

On implementing it requires four multipliers and one adder and one subtraction.

But if we multiply the two inputs by the proposed approach we will get the outputs as

Real Part(R) = x1(x2+y2)-y2(x1+y1) & Imaginary part(I) = x1(x2+y2)-x2(x1-y1)

If we add above two terms (R and I) it gives the same value as simple multiplication. But implementation of R and I requires three multipliers and two adder and three subtractions (term x1(x1+x2) is counted once because it is repeating in real and imaginary part), so one multiplier is reduced on cost of one adder and two subtraction.

Proposed complex multiplication need one extra adder and two extra subtractions on the cost of one reduced multiplier.

As we knew a 16 bit adder need 16 Full adder and 16 bit subtraction need 16 Full adder with 16 XOR gates.

But one 16 bit multiplier needs 16x16=256 AND gate and 32x15=480 Full adder (for conventional multiplication) and this can be reducing maximum up-to 75% of conventional requirement if we use advance multiplication techniques (like Wallace, Vedic, booth etc.).

Still one adder and two subtractions is a better deal instead of using one 16 bit multiplication.

64 point				8 points			
Conventional		proposed		Conventional		Proposed	
Multiply	Add/sub	Multiply	Add/sub	Multiply	Add/sub	Multiply	Add/sub
6*32*4= 768	6*32*2= 384	6*32*3= 576	6*32*5= 960	3*4*4= 48	3*4*2= 24	3*4*3= 36	3*4*5= 60

Let if we need to multiply z1=3.25+3j and z2=7.5+1.17 j then
R=3.25(7.5+1.17)-1.17(3.25+3) => 3.25(8.67)-1.17(3.25)=> 28.1775-3.8025=> 20.865
I=3.25(7.5+1.17)-7.5(3.25-3) => 3.25(8.67)-7.5(.25)=> 28.1775-1,875=> 26.3025
Let's have the above example in binary as in our proposed signed complex multiplier design
X1(3.25) => 000000000011.0100
X2(7.5) => 000000000111.1000
Y1(3) => 000000000011.0000
Y2(1.1875)=> 000000000001.0011
X2+Y2(8.6875) = 000000001000.1011
X1+Y1(6.25)= 000000000110.0100
X1-Y1(0.25)= 000000000000.0100
{X1*(X2+Y2)}(28.234375) => 00011100.00111100
{Y2*(X1+Y1)}(7.421875)=> 00000111.01101100
{X2*(X1-Y1)}(1.875)=> 00000001.11100000
{X1*(X2+Y2) – Y2*(X1+Y1)}(20.8125) => 00010100.11010000
{ X1*(X2+Y2) – X2*(X1-Y1)}(26.359375)=> 00011010.01011100
So our final Real number is R= 20.8125 and imaginary number is I= 26.359375

Proposed Butterflies: As we know the twiddle factor is a complex number and we need complex floating signed multiplication only to multiply with twiddle factor, also it is permanent for each butterfly. We have identified those butterflies in which twiddle factor has only real or imaginary value, and designed different butterfly for them. So we have categorized three types of butterflies.

Type-1: If twiddle factor has real and imaginary both (W_N^k = x1+iy1) then-

Let two inputs are A = x2+iy2 and B = x3+iy3

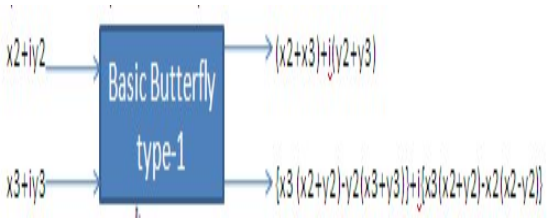

Figure 3(a): Type-1 butterfly

Type-2: If twiddle factor has only real value (W_N^k = 1) then-

Let two inputs are A = x2+iy2 and B = x3+iy3

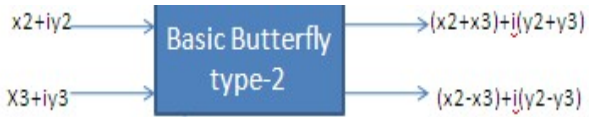

Figure 3(b): Type-2 butterfly

Type-3: If twiddle factor has only imaginary (W_N^k = -i) then-

Let two inputs are A = x2+iy2 and B = x3+iy3

Figure 3(c): Type-3 butterfly

Above it is been observed that the butterfly of type-2 & type-3 does not required multiplication, so we have a complex control unit to identify butterfly type and use different type of butterflies.

For high throughput systems, pipelined architecture is a good choice, and it is also an ideal method to implement high-speed long-size FFT owing to its regular structure and simple control. The performance of pipelined FFT processor can be improved by optimizing the structure and saving hardware resources. The block diagram of our proposed FFT processor is illustrated in Fig.1. It consists of four essential units. Control unit, Butterfly unit (BU), which has three-stage pipelined structure, carries out the complex multiplication. RAMs are used to store and output data.

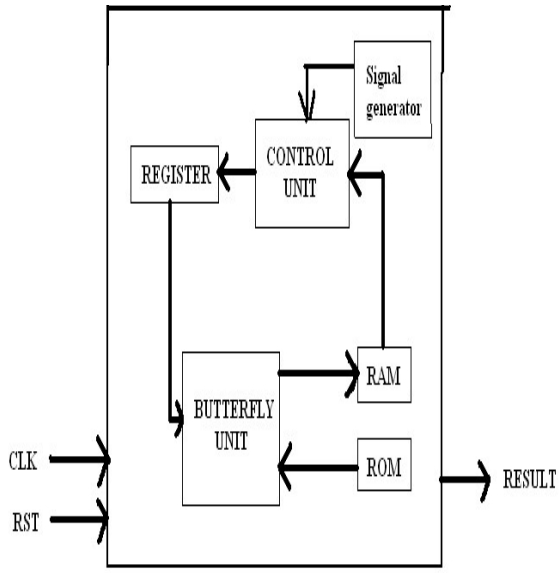

Figure 4: The pipelined FFT architecture

Timing signal Generator: To achieve pipelined FFT algorithm we suppose to have a timing control for changing different states of FSM. We have come up with this isolated Timing signal generator after having problem with synchronization in the same control module.

The RAM Unit: We need RAM of size 8x8 in case of 8 point FFT & RAM of size 64x8 for 64 point FFT. We required RAM for storage of last butterfly stage outputs, ones we have our output we suppose to feed that back into the butterfly. Control unit is there for read the data from RAM and after done sort of calculation it put that into the butterfly stage through Registers.

The ROM unit: we have two different purposes for the ROM first to store the input data at the time of simulation, we have FPGA with limited I/O ports and in case of 64 point FFT, if we have 16 bit word length we required total 64x16=1024 Pins for input only. We also using ROM for storing the twiddle factor which we know is permanent and specific for specific butterfly.

The Registers: It same as the RAM and we just need it store the output from the control unit and the new stage gets the input from it. Registers used in this system stores the output of the control unit and supplies this data to the Butterfly Unit when needed.

Control Unit: Control unit, which generates all control signals for the whole system, is responsible for operation control of the processor.

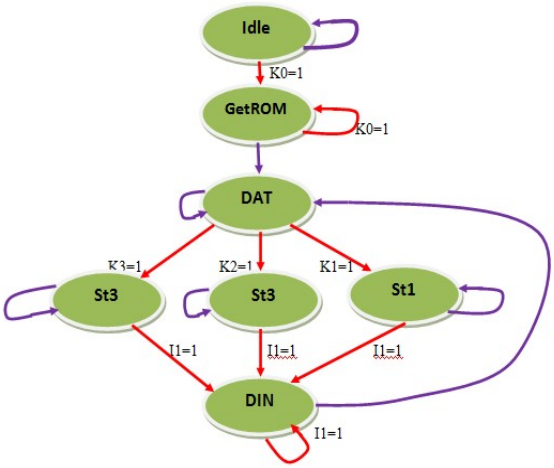

Figure 5. Control Unit FSM

*Red lines are IF conditions, Purple lines are the ELSE conditions

Figure 5 shows the FSM for 8 point FFT. The input signals are from the timing and control unit. The main work of the control unit is to control and proper arrangement of inputs and outputs it also perform the correct type of butterfly selection for the correct sequence of the Butterflies (sequence is fixed as per the FFT algorithm). An 8-bit signal controls the whole FFT processor. And this signal generates two parameters, write_en and read_en, to control the system. It also generates signals to select data from RAM, each of which is made up of 8 32-bit registers. The BU and the remaining parts are controlled as well. This control unit harmonizes all steps of the FFT processor based on a 7-bit counter. To achive synchronization in control unit we have used Melay type FSM (finite state machine) is explained below the table:

III-RESULTS

Table explained below shows the result for our FFT module for different inputs points. Here we can observe the effect on area & speed because of changes in the inputs

	Platform Used Vertex 4 xc2vp100-6ff1696				
Synthesis Results for	**4-point FFT**	**8-point FFT**	**16 point FFT**	**32 point FFT**	**64 point FFT**
Number of slices	183	368	694	1357	2664
Number of slice Flip-Flop	307	616	1178	2302	4551
Number of 4 input LUT	256	518	969	1892	3695
Number of Bounded IOB	28	66	130	258	514
Number of MULT 18x18s	2	3	6	12	24
Number of GCLK	1	1	1	1	1
Total Equivalent GATE Count	10890	21,434	42,037	83,369	165783
Time Delay	4.071 ns	4.35 ns	4.724 ns	4.891 ns	5.036 ns

Table 2: summary of synthesized results for different inputs

From the table above we can observe the results obtain after synthesis of proposed FFT processor for different FFT points (number of inputs), Slice, LUT, GATE count, No. of MULT and slice flip flop can be consider as area report of the design, time delay is the propagation time between input and output it basically shows the speed of the design.

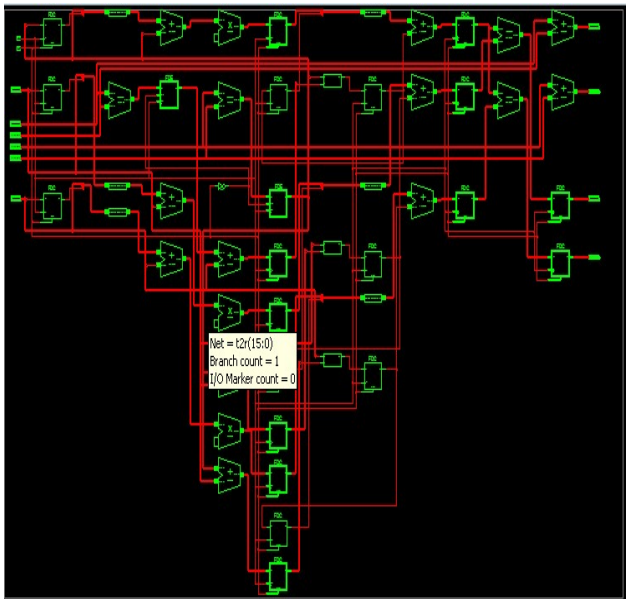

Figure 6- Internal modules RTL of proposed FFT

BASSSIC Project Status

Project File:	basssic.ise	**Current State:**	Programming File Generated
Module Name:	fft	**• Errors:**	No Errors
Target Device:	xc4vlx15-12ff668	**• Warnings:**	134 Warnings
Product Version:	ISE 9.2i	**• Updated:**	Sat Jun 1 20:30:18 2013

BASSSIC Partition Summary

No partition information was found.

Device Utilization Summary

Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	183	12,288	1%	
Number of 4 input LUTs	243	12,288	1%	
Logic Distribution				
Number of occupied Slices	150	6,144	2%	
Number of Slices containing only related logic	150	150	100%	
Number of Slices containing unrelated logic	0	150	0%	
Total Number of 4 input LUTs	243	12,288	1%	
Number of bonded IOBs	162	320	50%	
Number of BUFG/BUFGCTRLs	1	32	3%	
Number used as BUFGs	1			
Number used as BUFGCTRLs	0			

Figure 7 Design utilization summary of 8 point FFT

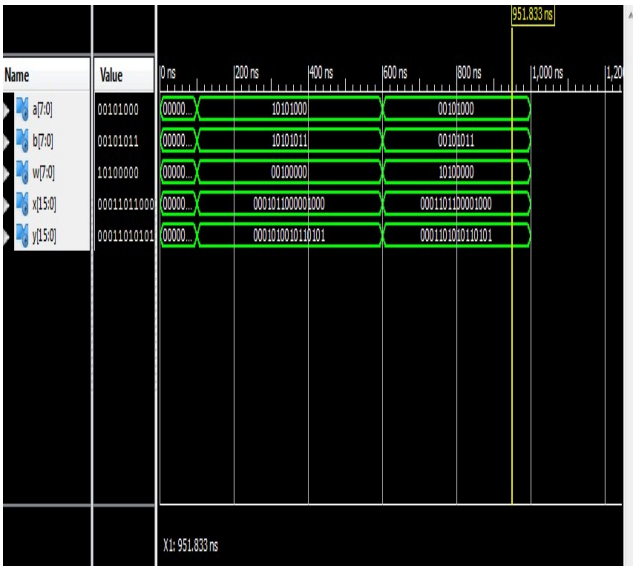

Figure 8: FFT top simulation results Waveform

Author	Outcome
Ajmal S A et al [1]	197 slice used of vertex FPGA with their new parallel pipelined radix-22 feed forward Fast Fourier transform (PPFFT) architecture
Ngoc-Hung Nguyen et al [2]	50, 866 slices for 1024-point FFT with their pipelined FFT processor based on the radix-2 decimation-in-frequency (R2DIF) algorithm using the single-path delay feedback (SDF) pipelined architecture
Asmita Haveliya et al [3]	2,016 slices for 32 points FFT, Fast Fourier Transform (FFT), based on Decimation-In-Time (DIT) domain, Radix-2 algorithm,
Proposed FFT processor results	183 vertex FPGA slice for the 4 –point FFT design, 1357 Vertex FPGA slice for the 32-point FFT design And 2664 Vertex FPGA slice for the 64- point FFT design, means 42,624 slices for the 1024 point FFT design

Table 3 Comparative results

Form the comparative results it can be observe that proposed work requires less number of slices of Vertex FPGA as compare to available works.

IV-CONCLUSION

Fast Fourier Transform (FFT) processor is widely used in different applications, such as: WLAN, Image process, spectrum measurements, Radar and multimedia communication services, FFT blocks are used in OFDM links such as very high speed digital subscriber line (VDSL), Digital Audio Broadcasting (DAB) systems and microwave portable links, It can also be used to design the speed efficient Infinite Impulse Response Filter, Finite Impulse Response Filter and Wireless communication.

In field of DSP, Transformation is prime requirement here we are proposing to design an IP (Intellectual Property) for the FFT. VLSI is all about Area, Power and speed, in our proposed design we have reduced the Area and got high frequency of FFT operations. The FFT processor design with vedic multiplier and new semi-pipelined Fast Fourier transform (SPFFT) with modified multiplication arrangement for modern communication systems provides an efficient way for computation of FFT with better area utilizing hardware architecture. We have gone through around 20 literature papers and after their thorough study we have come up with an idea as explained in thesis. We have tried to implement our module as per our proposed design. Form the comparative results it can be observe that proposed work requires less number of slices of Vertex FPGA as compare to available works.

We are concluding our thesis that our designed FFT module is very much area optimized and it has higher speed of computation with all sort of operations which FFT can perform.

REFERENCES

[1] Ajmal S A, S L Gangadharaiah, FPGA based area optimized parallel pipelined Radix-2^2 feed forward FFT architecture, IEEE International Conference On Recent Trends In Electronics Information Communication Technology, May 20-21, 2016, India, ISBN: 978-1-5090-0774-5/16, 2016 IEEE

[2] Ngoc-Hung Nguyen, Sheraz Ali Khan, Cheol-Hong Kim, and Jong-Myon Kim, An FPGA-Based Implementation of a Pipelined FFT Processor for High-Speed Signal Processing Applications, Springer International Publishing AG 2017 S. Wong et al. (Eds.): ARC 2017, LNCS 10216, pp. 81–89, 2017. DOI: 10.1007/978-3-319-56258-2_8

[3] Asmita Haveliya, *Design and Simulation of 32-Point FFT Using Radix-2 Algorithm for FPGA Implementation,* 2012 Second International Conference on Advanced Computing & Communication Technologies 978-0-7695-4640-7/12, 2012 IEEE, DOI 10.1109/ACCT.2012.43

[4] S.S.Kerur, Prakash Narchi, Jayashree C N, Harish M Kittur and Girish V A "Implementation of Vedic Multiplier for Digital Signal Processing" International Conference on VLSI, Communication & Instrumentation (ICVCI) 2011.

[5] G. Shafirulla, M. Subbareddy, " Design of high speed FFT processor based on FPGA", Vol.2, Issue 3, May- June 2012, pp 657-660, International Journal of Modern Engineering Research (IJMER) , ISSN: 2249-6645.

[6] Anvesh kumar, Ashish raman, "Low power, High Speed ALU by Vedic Mathematics" published in National Conference organised by NIT, hamirpur 2009.

[7] Chen-Fong Hsiao, Yuan Chen, Chen-Yi Lee, "A Generalized Mixed-Radix Algorithm for Memory-Based FFT Processors", pp 1549-7747 © 2010 IEEE.

[8] M. Mohamed Ismail, M.J.S Rangachar, Ch. D. V. Paradesi Rao, "An Area Efficient Mixed-Radix 4-2 Butterfly with Bit Reversal for OFDM Applications", European Journal of Scientific Research, ISSN 1450-216X Vol.40 No.4 (2010), pp.515-521, © EuroJournals Publishing, Inc. 2010.

[9] M. Ramalatha, Senior Member, IEE*E,* K. Deena Dayalan, Member*,* IEE*E,* P. Dharani, Member, IEE*E,* S. Deborah Priya, Member*,* IEE*E,"* High Speed Energy Efficient ALU Design using Vedic Multiplication Techniques" ACTEA 2009, July 15-17, 2009 Zouk Mosbeh, Lebanon, 978-1-4244-3834-1/09/ © 2009 IEEE.

[10] Xin Xiao, Erdal Oruklu and Jafar Saniie, "Fast Memory Addressing Scheme for Radix-4 FFT Implementation", 978-1-4244-3355-1/09/$25.00©2009 IEEE.

[11] Hen-Geul Yeh, Gerald Truong, "Speed and Area Analysis of Memory Based FFT Processors in a FPGA"*,* 1-4244-0697-8/07/.00 © 2007 IEEE.

[12] Himanshu Thapliyal and Hamid R Arabnia, "A time area power efficient multiplier and square architecture based on Ancient Indian Vedic mathematics", department of computer science. The University of Georgia, 415 graduate studies research centre Athens, Georgia 30602-7404, U.S.A.

[13] Shripad Kulkarni, "Discrete Fourier Transform (DFT) by using Vedic Mathematics", report, vedicmathsindia.blogspot.com, 2007.