Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 02, February-2016

FPGA Based Design and Simulation of 32- Point FFT Through Radix-2 DIT Algorith

Sudhanshu Mohan Khare
M.Tech (perusing), Dept. of ECE
Laxmi Naraian College of Technology,
Bhopal, India

M. Zahid Alam
Associate Professor, Dept. of ECE,
Laxmi Naraian College of Technology,
Bhopal, India

***Abstract*— Digital signal processing needs the conversion from time domain to frequency domain. For discrete sequences, this conversion is done through Discrete Fourier Transform but it is numerically inefficient. Thus an efficient algorithm is developed to compute DFT and it is known as Fast Fourier transform (FFT). This paper concentrates on FPGA based design synthesis and simulation of the Fast Fourier Transform (FFT) through Decimation-In- Time (DIT) with Radix-2 algorithm. VHDL is used as design entity and XST (VHDL) of Xilinx Design Suite ISE 14.7 is used for synthesis purpose. Simulation is done with the help of ISim of Xilinx 14.7. In this paper XC6VLX75T device is used with vertex6 family and FF484 package. Synthesis results show that the design is better in terms of area as well as speed.**

Keywords*— *Fast Fourier Transform, Discrete Fourier Transform, VHDL, FPGA, Radix, Butterfly, DIT.

I. INTRODUCTION

This paper represents a FPGA based design of 32-points FFT by using Decimation in time, radix-2 algorithm. The paper concentrates on the design of FFT for a FPGA kit. In this project coding is done by using VHDL and simulation is done with the help of ISim of Xilinx Design Suite, ISE 14.7. This paper proposes design of 32 point FFT by using VHDL as a design entity and it is synthesized in XST (VHDL) of Xilinx ISE Design Suite 14.7 version. The project is focused on a design which is efficient in terms of area as well as speed for computing a 32 point FFT. In this project 16 bit numbers are provided as inputs in which 8 bits are used to represent integer part and next 8 bits are used to represent decimal part.

Transforms are generally used to convert a function from time domain to frequency domain without any loss of information, similarly Fourier Transform also converts a function from the time domain to the frequency domain. In this project the FFT is implemented for a 32-points sequence with the help of Decimation In Time algorithm with radix-2. The synthesis of 32-points FFT is divided into eight stages, first stage is used to supply inputs where second stage consists of a finite state machine (FSM), and next five stages are used for calculation of radix-2 DIT FFT through butterfly operations and last stage is used to store the outputs of 32 point FFT i.e. 32-point sequences which are now converted into frequency domain. Before understanding the implementation part one need to have in depth knowledge of DFT and FFT.

II. DFT

The Discrete Fourier Transform is a very important mathematical tool which is used for discrete-time signal-processing. It is used to convert time domain sequence to frequency domain. DFT computes the Z- transform for evenly spaced points around a unit circle for the given sequence. If the given sequence is of finite duration then the DFT is used as the transform. DFT finds its application in digital signal processing such as linear filtering, spectrum analysis etc. DFT is very important discrete transform to perform Fourier analysis in several applications.

Let a finite duration sequence x[n] of length N, its DFT can be given as:

$$X(K) = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk/N}$$

Where, k = 0, 1, 2, 3, ……N-1

Here W_N is known as Twiddle Factor and it can be expressed as the complex value phase factor, which is equals to Nth root of unity and this twiddle factor is expressed as:

$$W_N = e^{-j2\pi/N}$$

Hence X (k) can be written as:

$$X(K) = \sum_{n=0}^{N-1} x[n]\, W_N^{nk} \quad ; \qquad 0 \le n \le N\text{-}1$$

DFT involves a lot of calculations and therefore it requires a time efficient algorithm. It can be observed that for each value of k, direct computation of X(k) involves N complex multiplications it means 4N real multiplications and N-1 complex additions i.e. 4N-2 real additions. Hence, for the calculations of all N values of the DFT requires N^2 complex multiplications and N^2-N complex additions.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 02, February-2016

Instead of using DFT if we will use FFT algorithm such as Decimation in Time (DIT) FFT with radix-2 algorithm, then the number of complex multiplications and additions will be reduced to (N/2) $\log_2 N$ and $N\log_2 N$ respectively to compute the DFT of a given complex sequence x[n]. Therefore we can say that the Fast Fourier Transform is a computationally and numerically efficient way to compute the Discrete Fourier Transform.

III. FFT

The Fast Fourier transforms (FFT) are the numerically efficient algorithms to compute the Discrete Fourier Transform (DFT). FFT algorithms are based on the concept of divide and conquer approach and it is done by decomposing the computation of DFT into smaller sequences of DFTs. This approach is useful in many areas but calculating it directly from the definition is often very slow to be practical. The FFT is used in so many applications where the frequency-domain representation of a signal has to be processed. In the field of communications the FFT is important because of its use in orthogonal frequency division multiplexing (OFDM) systems. Also FFT is the most popular technique of digital spectrum analysis. Cooley and Tukey have first introduced the concept of FFT to describe a significant computational reduction by making effective use of symmetry and periodicity properties of the twiddle factors. These properties can be given as:

Symmetry property: $W_N^{k+N/2} = -W_N^k$

Periodicity property: $W_N^{k+N} = W_N^k$

The FFT is an algorithms used for the efficient computation of the DFT. DFT and FFT are the most popular signal processing tools. FFT computes the DFT in an efficient manner and provides exactly the same result as obtained by evaluating the DFT definition.

Mathematically FFT can be calculated through DFT:

$$X(K) = \sum_{n=0}^{N-1} x[n]\, W_N^{nk} \quad ; \ 0 \le n \le N\text{-}1$$

Here W_N^{nk} is called twiddle factor and it is given by:

$$W_N^{nk} = e^{-j\left(\frac{2\pi}{N}\right)nk}$$

Fast Fourier Transform (FFT) is based on decomposition and breaking it into smaller sequences and at the end again combining into one transform. The FFT algorithms can be classified as Decimation In Time (DIT) and Decimation In Frequency (DIF) algorithms. In decimation in time the sequence in time domain is divided into smaller sequences and the DFT of these sequences are combined in a certain pattern to obtain the required DFT. In the decimation in frequency approach, the frequency samples of the DFT are decomposed into smaller and smaller subsequences in a certain manner. DIT and DIF algorithms are further divided into Radix-2, Radix-4, Radix-8, Split-Radix etc. All these algorithms are based on the single method, that is, Divide and Conquer method.

IV. DIT RADIX – 2 FFT

FFT can be implemented based on Decimation-In-Time (DIT-FFT) and Decimation-In-Frequency (DIF-FFT) algorithm. Both the algorithms are having same computational complexity but they are different in input and output computational arrangement. The name Radix-2 is called due to its base is equals to 2 and the representation is 2^M, here M represents the index/stage and its value is always equals to a positive integer. For example 32 point FFT is divided into 5 stages and can be written as 2^5, where 2 represents its Radix and therefore this algorithm is called as "Radix-2 DIT-FFT algorithm". In Radix-2 DIT algorithm the sequence splits into two sequences consisting of even numbered values and odd numbered values of the input sequence x(n). The Radix-2 DIT-FFT can be expressed mathematically as:

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

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

In this way an N-point FFT can be divided into two N/2 -point DFTs, for example 32 point DFT can be divided into two 16 points DFT. Now this N/2 point DFT can also be divided into two N/4 point DFTs and so on. In our case 16-points DFT can be divided into two 8-points DTFs, then these 8-point DFTs can be divided into two 4-points DFTs and at last these 4-points DFTs can be finally divided into two 2-points DFTs. Therefore a 32-points FFT can be computed in 5 stages where each stage consists of 16 butterfly operations. Butterfly operation in a Radix-2 algorithm is a portion of the computation of FFT that provides the Fourier transform of two point sequence in a simplified manner. The name "butterfly" arises from the shape of the signal-flow diagram in the radix-2 case. Generally the term "butterfly" appears in the context of the Cooley–Tukey DIT-FFT algorithms. In these algorithms it recursively breaks down a DFT of composite size $n = r^m$ into r smaller transforms of size m where r is called as the "radix" of the transform. These smaller DFTs are then combined with the help of size-r butterflies, which in our case are size-2 butterflies and these themselves are DFTs of size r and it is performed m times on corresponding outputs of the sub-transforms and it is pre-multiplied by roots of unity which is also known as twiddle factors. This is the procedure of "decimation in time (DIT) FFT" algorithms.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 02, February-2016

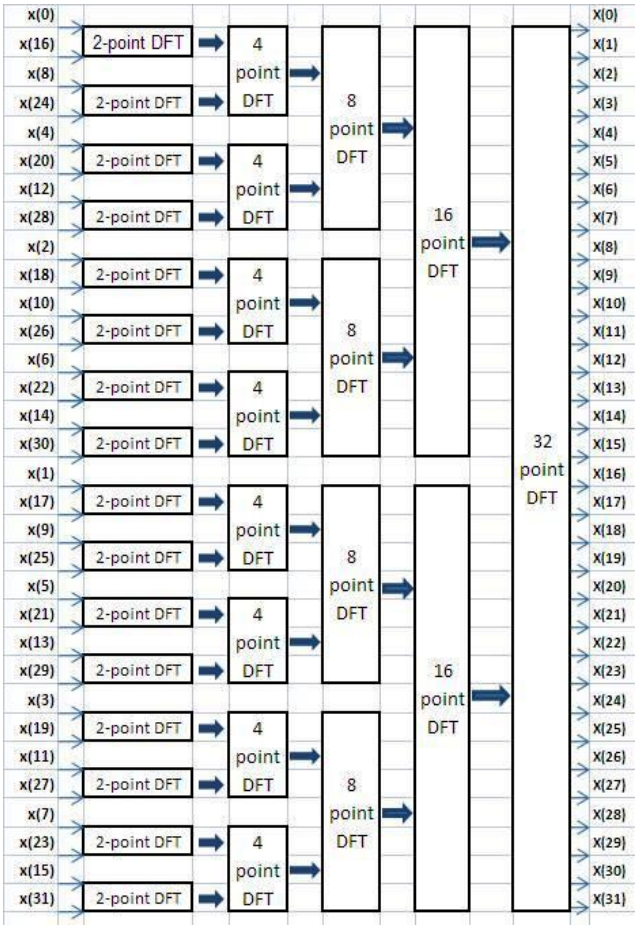

Fig.1:- Block Diagram of a 32 Point DIT FFT with Radix-2

Butterfly is a DFT of size-2 that takes two inputs (*A*, *B*); where *A* and *B* are corresponding outputs of the two sub-transforms and it provides two outputs (*C*, *D*). Here *W* represents respective twiddle factor. When successively applied in DIT-FFT algorithms until the shorter and shorter DFTs reach length-2, the result is the radix-2 DIT FFT algorithm.

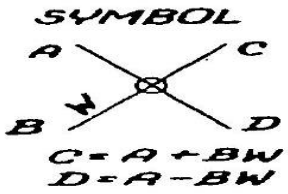

Fig.2:- Basic Butterfly Operation

The radix-2 algorithm is the simplest FFT algorithm with decimation in time. The decimation-in-time (DIT) radix-2 FFT recursively divides a DFT into two half-length DFTs of the even-sequences and odd-sequences of time samples. The outputs of the shorter FFTs are reused to compute many outputs; therefore it is reducing the total computational cost and the delay time.

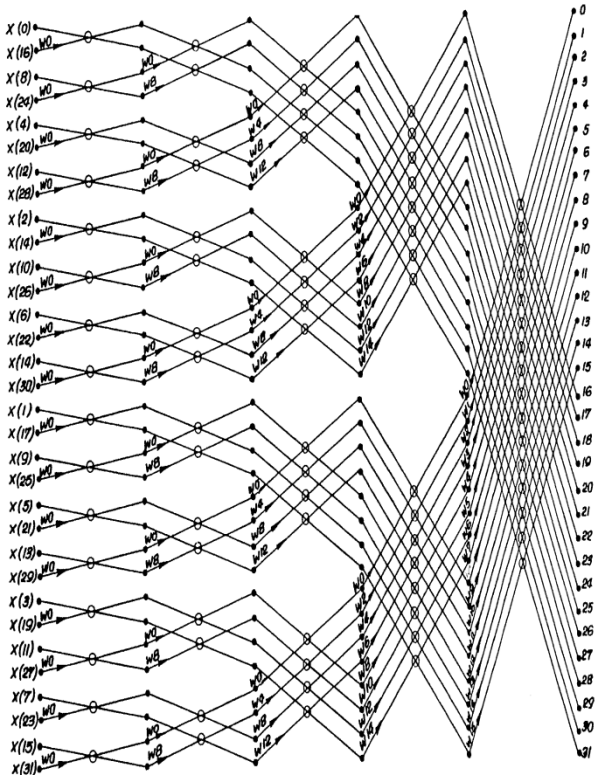

Fig.3:- Signal Flow Graph of a 32 Point DIT FFT with Radix-2

On the basis of above discussion we can say that a 32 point FFT can be implemented via DIT-FFT with Radix-2 algorithm by using 16 butterflies per stage and 5 such stages are required to implement a sequence of 32 points. Hence we must use 80 butterfly operations to design a 32 point FFT.

V. ARCHITECTURE & SYNTHESIS

The architecture of the Fast Fourier Transform is based on its working principle. The hardware architecture may be divided into two categories, one is memory based architecture and the other is pipelined based architecture. Memory based architecture is further divided into four types: single memory architecture, dual memory architecture, array – architecture, and cached memory architecture.

In this project the dual memory based architecture is used in which the sequence is divided into groups and it is stored separately in two different RAMs. The 32-points FFT is performed on a sequence which consist 32 values and the position of each value is represented by 'k'. The binary representation of the factor 'k' needs 5 bits as it needs to represent 32 numbers. For example the position of 1st sequence is represented by 00001 and the position of 16th sequence is represented by 10000.
If we will observe the respective positions of sequences in a butterfly operation then it can be concluded that every butterfly operation consists two sequences and the position of these two sequences have a common pattern. In each butterfly operation the position of 1st sequence is of lower

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 02, February-2016

MSB (most significant bit) and the position of 2nd sequence is of higher MSB. The position of these sequences can be obtained by simply dividing the 32 sequences into two parts in which 1st part will consists all the sequences having positions of lower MSB and 2nd part will consists all the sequences having positions of higher MSB. Therefore RAM-1 will consists all the sequences with their respective positions of lower MSB i.e. positions 0,1,2,3,……13,14,15 and RAM-2 consists all the sequences with their respective positions of higher MSB i.e. 16,17,18,……29,30,31.

The FFT is based on division of odd and even numbered sequences and therefore we need to divide our time domain sequence into two parts one must contains all the even numbered sequences and the other one should consists of all the odd numbered sequences. This division of even and odd numbered sequences must be continued till we get the pairs of sequences. To simplify this we can use "Bit Reversal" method in which all the bits of the position term 'K' are reversed to get the correct order of the sequences.

Generally bit reversal is used in a FFT processor to get the input sequence in correct manner, but it is not a good idea to implement bit reversal on FPGA if we have to achieve minimum delay and hardware. Therefore in my project I have avoided bit reversal and the correct sequence of FFT is achieved with the help of two RAMs in which sequences of lower MSB and higher MSB positions are stored. These RAMs will serially supply the input sequences and the correct sequence will be achieved with the help of a unique pattern. This concept is shown below in table number 1.

Table.1:- Unique pattern to get the sequence for a 32 Point FFT

RAM-1 :	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RAM-2 :	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
I/P of Butterfly:	1	9	5	13	3	11	7	15	2	10	6	14	4	12	8	16
unique sequence to get next butterfly	0	8	-4	8	-10	8	-4	8	-13	8	4	8	-10	8	4	8

For a 32 point FFT we have to use 16 different twiddle factors which are complex in nature and must be calculated before applying to the FFT. The table number 2 given below shows the calculated values of twiddle factors $\boldsymbol{W_{32}^{k}}$ for a 32 point FFT.

Table.2:- Twiddle factors for a 32 Point FFT

K	COS(nk/16)	sin(nk/16)	Twiddle Factor (W^{k}_{32})
0	1.000	0.000	1
1	0.981	0.195	0.980785312758822-0.195090159353512i
2	0.924	0.383	0.923879659446842-0.382683125915406i
3	0.831	0.556	0.831469888725349-0.555569819323419i
4	0.707	0.707	0.707107250279226-0.707106312093558i
5	0.556	0.831	0.555570922512935-0.831469151597416i
6	0.383	0.924	0.382684351713887-0.923879151704 01i
7	0.195	0.981	0.195091460654289-0.980785053913331i
8	0.000	1.000	1.3267948966775-06-0.99999999912i
9	-0.195	0.981	-0.195088858052393-0.980785571602587i
10	-0.383	0.924	-0.382681900116252-0.923880167188047i
11	-0.556	0.831	-0.555568716132925-0.831470625851818i
12	-0.707	0.707	-0.707105373906644-0.707108188463 65i
13	-0.831	0.556	-0.831468414468019-0.555572025701473i
14	-0.924	0.383	-0.923878643959552-0.382685577511694i
15	-0.981	0.195	-0.980784795066114-0.195092761954721i

RTL schematics of the synthesized FFT are shown in fig.4 and internal view of the RTL schematic is shown in fig.5 which represents different stages of 32-point FFT. Fig .6 and fig.7 shows respectively the RTL and its internal view of a butterfly operation which uses twiddle factors to compute FFT.

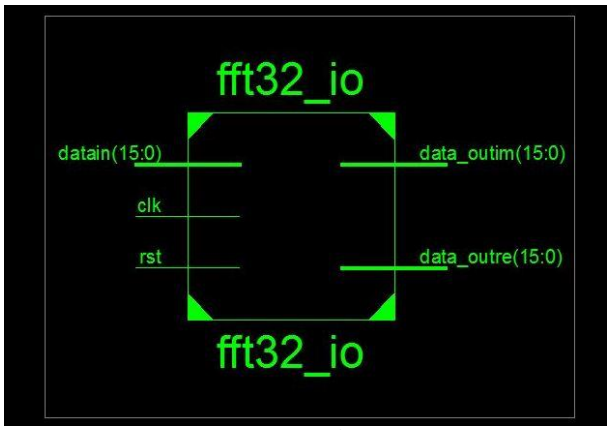

Fig.4:- RTL schematic of a 32 Point DIT FFT

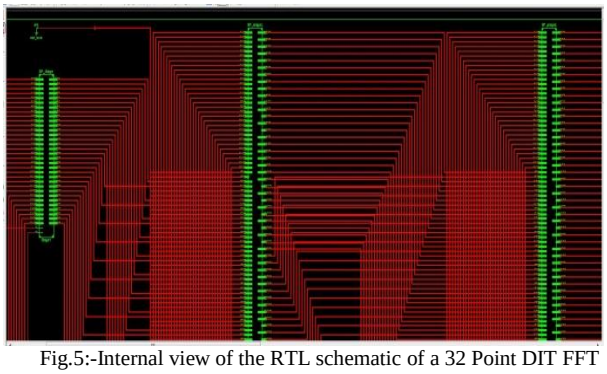

Fig.5:-Internal view of the RTL schematic of a 32 Point DIT FFT

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 02, February-2016

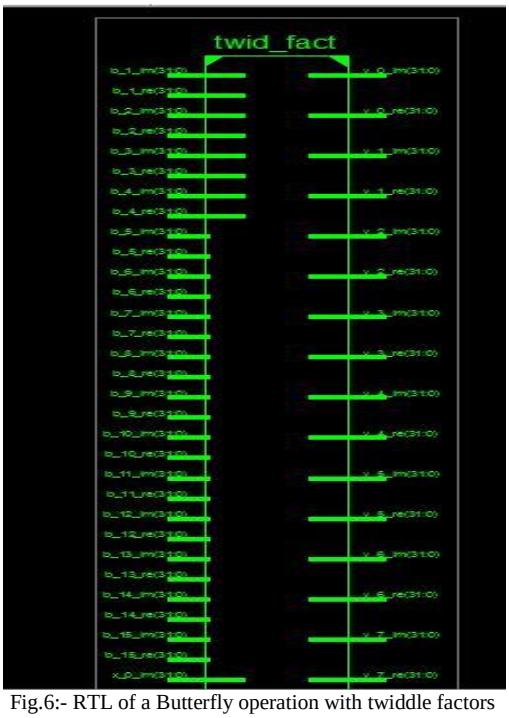

Fig.6:- RTL of a Butterfly operation with twiddle factors

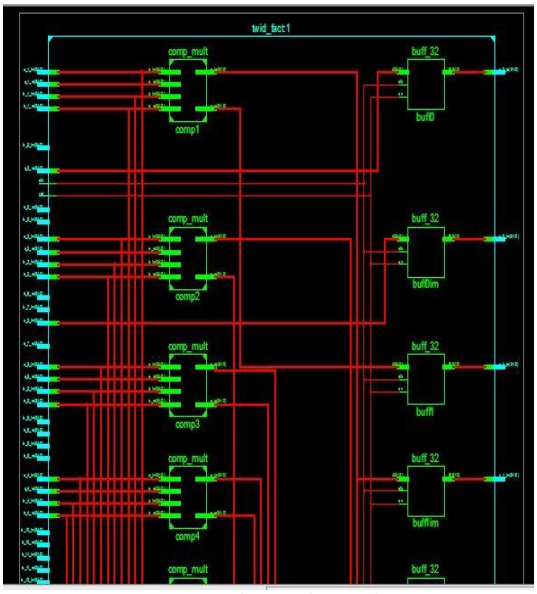

Fig.7:- Internal view of RTL of Butterfly operations

V. SIMULATION & RESULTS

The simulation of this project is done with the help of ISim of Xilinx ISE 14.7 in which 32-point input sequence is provided with the help of an input block and twiddle factors are also provided by calculating and converting them into their 16-bit binary equivalents. The following figures shows the waveforms obtained after simulation.

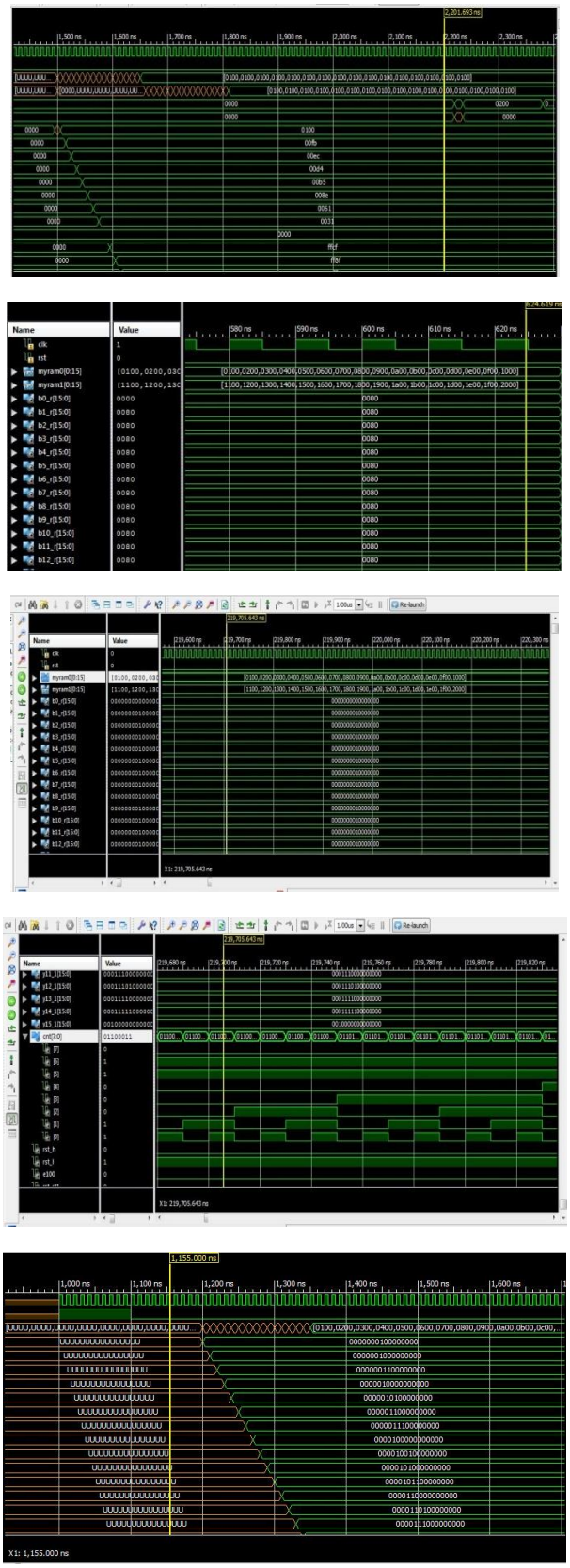

Fig.8:- Simulation waveforms for a 32-point FFT

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 02, February-2016

VI. CONCLUSION

Simulation of the FFT and results obtained from its detailed synthesis report shows that the proposed design of 32-point DIT FFT with radix-2 is very efficient in terms of area as well as it is also efficient in terms of speed. The minimum delay time it required is 2.330 ns per stage. The table-4 given below shows the device utilization summary and the fig.9 shows the timing summary of the top module implementation.

Table.3:- Device Utilization Summary

Logic Utilization	Used	Available	Utilization
Numbers of fully used LUT-FF pairs	5865	24092	24%
Number of bonded IOBs	50	240	20%
Number of slice registers	8042	93120	8%
Number of Slice LUTs	21915	46560	47%
Number used as logics	21915	46560	47%

```
Timing Summary:
---------------
Speed Grade: -1

   Minimum period: 2.330ns (Maximum Frequency: 429.185MHz)
   Minimum input arrival time before clock: 1.387ns
   Maximum output required time after clock: 0.789ns
   Maximum combinational path delay: No path found

Timing Details:
---------------
All values displayed in nanoseconds (ns)
```

Fig.9:- Timing Summary of the implementation of a 32-point FFT

REFERENCES

[1] 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.

[2] K. Sowjanya and Leela Kumari , "Design and performance analysis of 32 and 64 point fft using radix-2 algorithm", 1K. SOWJANYA, 2B. Proceedings of AECE-IRAJ International Conference, 14th July 2013, Tirupati, India, ISBN: 978-81-927147-9-0

[3] Manoj Verma and Ravi Sindal, "Simulation and Analysis of DIT FFT Algorithm for Spartan 3 FPGA", 2013 International Conference on Communication Systems and Network Technologies

[4] Shaik qadeer, Md. Zafar Ali Khan, Syed Abdul Sattar and Ahmed, "A radix-2 DIT FFT with reduced arithmetic complexity", 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI).

[5] Z. Lukac and M. Temerinac , "Analysis of some methods For maintaining accuracy in implementation of FFT on fixed point DSP", Proc. EEE Conference, Montenegro, 2005 September pp. 28 -30.

[6] R. Roa, D. N. Kim, "Fast Fourier Transform: Algorithms And Applications," Springer Publications, UK, 20IO

[7] Truong Nguyen, and Wei-Hsin Chang , "integer FFT with optimized coincident sets," EEE Inter. Confe. on, Acoustics, Speech and Signal Processing, 2007, ICASSP 2007.

[8] Johnson, S.G. ; Frigo, M, " A modified split radix FFT with fewer arithmetic operations," Signal Processing, EEE Transactions, year 2007, volume 55, issue I.

[9] Remya Ramachandran Department of EEE Hindustan College of Engineering and Technology, "Simulation of radix-2 fast fourier transform using xilinx", Remya Ramachandran et al. / International Journal of Computer Science Engineering (IJCSE)

[10] T.S. Ghouse basha1, Peerla Sabeena sulthana, "Design And Simulation Of Pipelined Radix-2k Feed-Forward FFT Architectures", International journal of innovative research in electrical, electronics, instrumentation and control engineering, Vol. 2, Issue 9, September 2014.

[11] Afreen Fatima, "Designing and Simulation of 32 Point Fft Using Radix-2 Algorithm for Fpga", IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE) e-ISSN: 2278-1676,p-ISSN: 2320-3331, Volume 9, Issue 1 Ver. III (Jan. 2014), PP 42-50 www.iosrjournals.org

[12] Alan V. Oppenheim, Ronald W. Schafer with John R. Buck, Discrete Time Signal Processing, Second Edition.