DESIGN AND PERFORMANCE ANALYSIS OF 32 AND 64 POINT FFT USING RADIX-2 ALGORITHM

[1]K. SOWJANYA, [2]B. LEELE KUMARI

Electronics and Communication dept, UCEK., JNT University, Kakinada, Andhra Pradesh, India

Abstract: Fast Fourier Transform is an algorithm used to compute Discrete Fourier Transform (DFT) of a finite series. This Paper Proposes the performance analysis of 32 and 64 point FFT using RADIX-2 Algorithm and it concentrate on Decimation-In-Time Domain (DIT) of the Fast Fourier Transform (FFT). Here we use Xilinx Design Suite 13.2 Version, by using VHDL as a Design Entity and the Synthesis Result are Stimulated on Vertex Kit.FFT Computation Technique is used in Wide Range of its Mathematics, Auto Correlation, Data Compression, Pattern Recognition etc. The Synthesis Results Shows the Comparison of 32and 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, VHDL.

I. INTRODUCTION

The Fourier Transform Decomposes a Wave form-basically any Real world wave form into Sinusoids. It is possible to generalize the Fourier transform on discrete structures such as Finite Groups. The Efficient Computation Of such structures, by fast Fourier transform, is essential for high speed computing.FFT algorithms are commonly employed to compute DFTs, but there is a clear distinction is that "DFT" refers to a Mathematical transformation, regardless of how it is computed, whereas "FFT" refers to a specific families of a algorithms for computing DFTs. Fast Fourier Transform (FFT) is developed by Cooley and Tukey in1965.Highly efficient procedure for computing the DFT of a finite series and requires less number of computations than that of direct evaluation of DFT. Fast Fourier transform (FFT) is based on decomposition and breaking the sequence into smaller sequences and combining them to get total sequence.

This paper Proposes and concentrate on the design of 32 and 64 point FFT and its performance analysis. By using VHDL as a design entity the synthesis and stimulation is done on Xilinx ISE Design Suite 13.2.A DFT Decomposes a sequence of values into components of different frequencies. This operation is useful in many fields but computing it directly from the definition is often too slow to be practical. An FFT is a way to compute the same result more quickly: Computing a DFT of N-points, takes O(N2) Arithmetical operations, while an FFT can compute the same DFT in only O(NlogN) operations. FFTs can decomposed using DFTs of even and odd points, which is called a decimationin-time (DIT) FFT, or they can decomposed using another approach which is called a Decimation-infrequency(DIF) FFT.

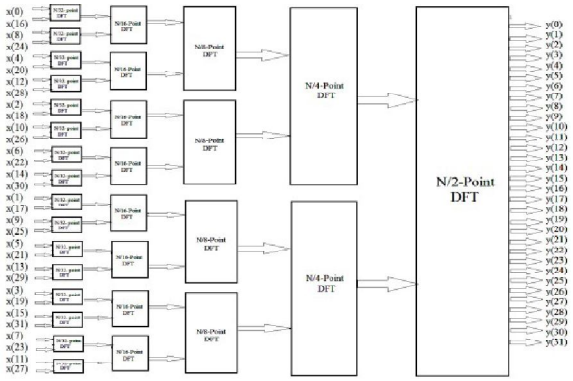

Figure 1:-Block diagram of 32 point radix-2 FFT

Computation of the end point DFT corresponds of computation of N samples of Fourier transform at N equally spaced frequencies. Consider the input x(n) of length N is a complex data sequence, its DFT X(k) is also complex data sequence of length N which is defined as

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N}. \quad (1)$$

Where k =0, 1 ...N-1, $\omega = e^{2\pi i/N}$ is known as Twiddle factor.

Twiddle Factor coefficients are used to combine the results from the previous stage to form inputs to the next stage.

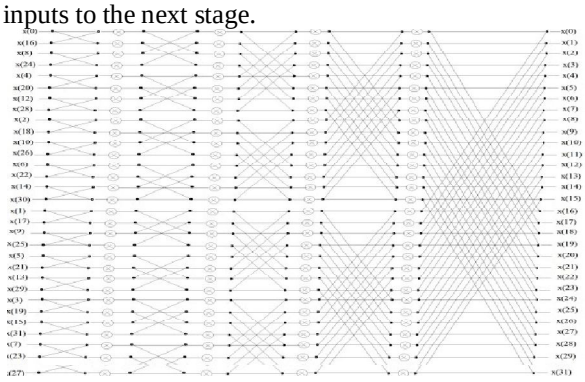

Figure 4:-FFT Algorithm of 64 point using Radix-2

To evaluate all N values of X (k), the number of complex multiplications and additions required for DFT are N2 and N (N-1) and for FFT those values are reduced to (N/2) log2N and Nlog2N to compute the input sequence x(n).

II. RADIX-2

The Radix indicates the size of FFT decomposition. In this paper Radix is 2 which is single-Radix FFT. For single Radix FFTs, transform size must be choose according to the power of Radix. Here we use 32and 64 sizes, which is 25 and 26.

The Radix-2 Decimation-in-Time FFT (DIT-FFT) is applied to the two Points N/2 DFT's. To find the number of butterfly stages required to compute N length sequence can be M=log2N, and N/2 butterfly operations are computed in each stage. In this paper, there are 5 butterfly stages and 16 butterfly operations are computed to produce 32 Point FFT. Similarly, 6 butterfly stages and 32 butterfly operations are computed to produce 64 Point FFT. Fig 1and Fig 3 shows the butterfly stages whereas, Fig 2 and Fig 4 shows the butterfly diagram of each and every stage.
In DIT-FFT the given input sequence is in shuffled order and the output sequence is in natural order. By using Bit-Reversal input sequence gets shuffled.

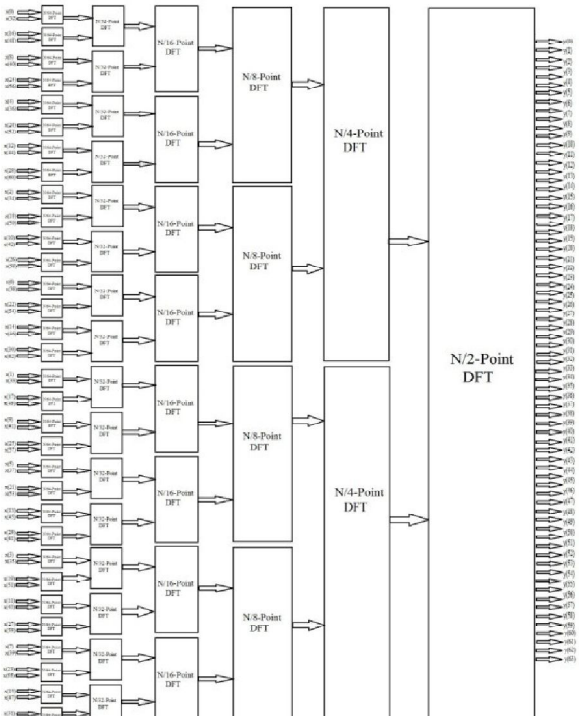

Figure 3:-Block diagram of 64 point radix-2 FFT

The Radix-2 decimation in time FFT is the basic form of Cooley-Tukey implementation algorithms. Radix-2 first computes the DFT of the even index inputs and the odd index inputs and then combines the two results to produce the entire DFT sequence. The basic computation block in the FFT is butterfly in which the two inputs are combined to give two outputs.

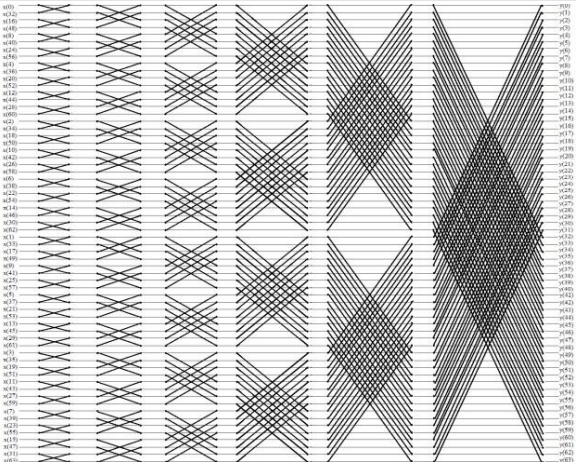

Figure 4:-FFT Algorithm of 64 point using Radix-2

The FFT operation of butterfly diagram is shown in the below figure, and the powers of the twiddle factors associated in butterflies are in natural order.

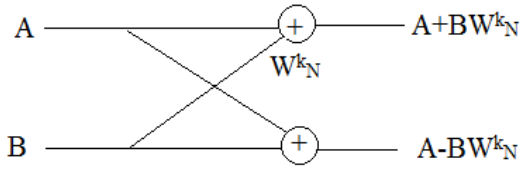

Figure 5:-Basic Butterfly Diagram

The twiddle factor exponent k of each stage is calculated by using below equation

K=Nt/2m where t=0, 1, 2,.....2m-1

III. IMPLEMENTATION PROCEDURE

The implementation procedure of 32 and 64-bit radix-2 FFT algorithm using VHDL has following steps:-
1. Select an N-point sequence, where N is equivalent to 32 and 64.
2. Sort the samples in bit-reversed order. 3. Apply the first stage butterfly using adjacent pairs of numbers.
4. Apply the second stage butterfly using pairs that are separated by 2.
5. Apply the third stage butterfly using numbers that are separated by 4.
6. Similarly apply the fourth and fifth stage butterfly using numbers that are separated by 8 and 16.
7. Continue butterfly the numbers in the buffer until we get separation of our sequence upto length N/2.
8. The final output values at the last stage will get in the normal order that is from 0 to N.

IV. SOFTWARE USED

The goal of the VHDL synthesis is to create a design that implements the required functionality and matches the designer's constraints in speed, area and power. The 32 and 64 point FFT proposed in this paper is been simulated using Xilinx ISE Design Suite 13.2 with the device family as vertex6 lower

power. The summary of the device description is as follows.

Family	VETREX 6 Lower Power
Device	XC6VLX130TL
Package	FF784
Speed	-1L

Table 1:-Design Properties

V. SOFTWARE SIMULATION

The RTL view of butterfly structure obtained after simulation of 32 and 64 point FFT is shown in below figure. Figure

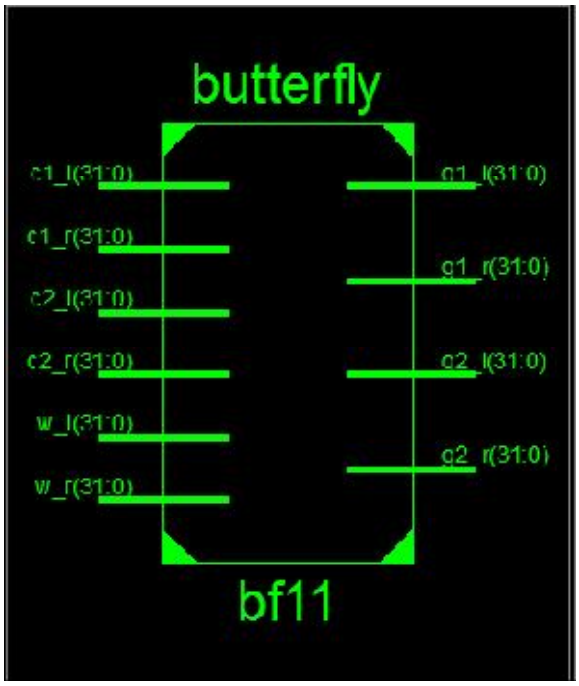

Figure 6:-RTL view of 32 point FFT butterfly component

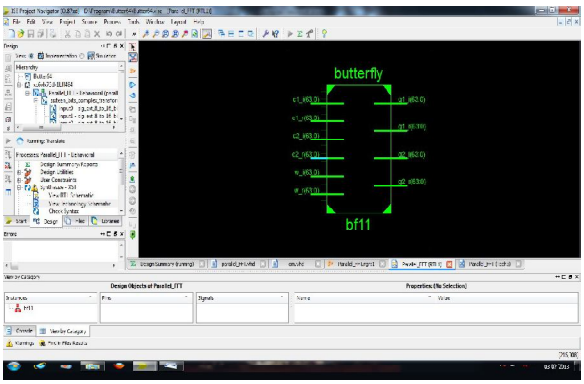

Figure 7:-RTL view of 64 point FFT butterfly component

The Vertex 6 device utilization summary of 32 and 64 point are in shown in below table 2 and 3. Table

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of slice LUTs	18428	46560	39%
Number of fully used LUT-FFT pairs	0	18428	0%
Number of Bonded IOBs	2560	240	1066%
Number of DSP48EIs	148	288	51%

Table 2:- Device Utilization Summary for 64 Point FFT

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of slice LUTs	130530	46560	280%
Number of fully used LUT-FFT pairs	0	130530	0%
Number of Bonded IOBs	10240	240	4266%
Number of DSP48EIs	112	288	38%

Table 3:- Device Utilization Summary for 64 Point FFT

In simulation results of 32 and 64 point FFT the input x is a signed vector and output y is complex numbers, in which both x and y are represented in binary format. The results are shown in below figure.

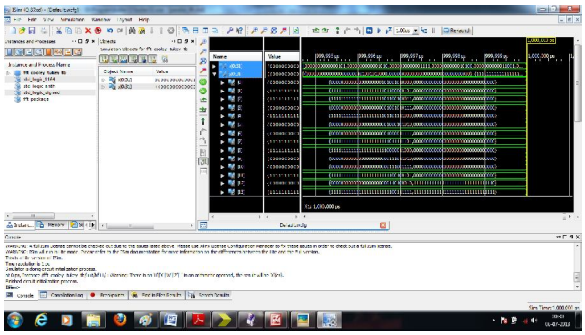

Figure 8:-Synthesis result for 32 point FFT

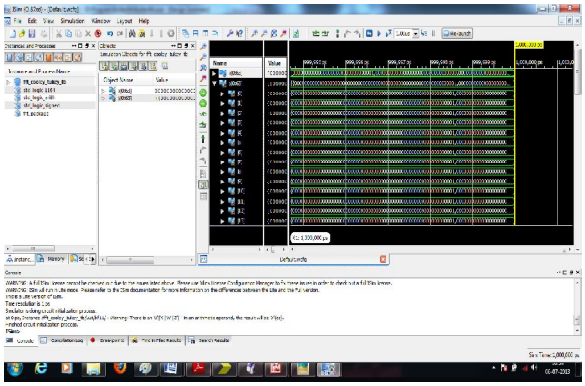

Figure 9:-Synthesis result for 64 point FFT

The minimum delay, Total Real Time to XST Completion and Total CPU Time to XST Completion of 32 and 64 Point FFT using Radix-2 algorithm are tabulated.

Parameters	32 Point	64 Point
Min.Delay	21.522ns	30.412ns
Total REAL Time to XST Completion	152.00secs	1876.00secs
Total CPU Time to XST Completion	152.82secs	1876.40secs

Table 4:- Time Delay of 32 and 64 Point FFT

The Performance analysis of 32 and 64 Point FFT are shown in figure 9.

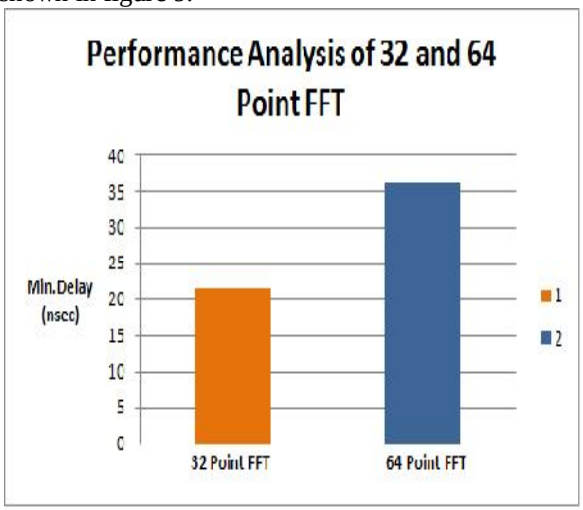

Figure 9:-Performance analysis

CONCLUSION

In this paper, we have proposed 32 and 64 point FFT design using Radix-2 algorithm, and their simulation and synthesis are done by Xilinx Synthesis Tool on Vertex. The test bench waveforms are displayed by using Xilinx ISE Design Suite 13.2.The Performance analysis of 32 and 64 Point FFT are represented by using the parameter, Minimum delay .The Performance analysis can also be done between single Radix and Split-Radix FFT algorithms by taking various parameters into consideration.

REFERENCES

[1] Asmita Haveliya, "Design and simulation of 32-point FFT using Radix-2 Algorithm for FPGA Implmentation",2012 second International conference on Advanced Computing and Communication Technologies.

[2] Sneha N.Kherde, Meghana Hasamnis, "Efficient Design and Implementation of FFT", International Journal of Engineering Science and Technology (IJEST), ISSN: 0975-5462 NCICT Special Issue Feb 2011.

[3] Ahmed Saeed, M.Elabably, G.Abdelfadeel, and N.I. Eladway. "Efficient Implementation of FFT/IFFT Processor", International Journal of Circuits, Systems and signal Processing, Issue 3, volume 3, 2009.

[4] Saad Bouguezel, M.Omar Ahmad, "Improved radix-4 and Radix-8 FFT Algorithms"IEEE department of Electrical and Computer Engineering, Concardio University, 1455 de Maisonneuve Blvd.West Monteral.P.Q.canada.

[5] Ali Saidi, "Decimation-in-Time-Frequency FFT Algorithm" Motorola Applied Research, Paging and Wireless Data Group Boynton Beach.

[6] Vassil S.Dimitrov, Kimmo U.JarVinen, and Jithra Adikari, "Area-Efficient Multipliers Based on Multiple-Radix Representations"IEEE Transactions on Computers, Vol 60, February 2011.

[7] Alan V.Oppenhem, Ronald W.Schaler with John R.Back, Discrete Time Signal Processing, Second Edition. [8] B.Parhami, Computer Arithmetic, Algorithms and Hardware Designs, 1999.

★★★