Design and Performance Analysis of 32 and 64 Point FFT using Multiple Radix Algorithms

K.Sowjanya
Department of E.C.E, UCEK
JNTUK, Kakinada
Andhra Pradesh, India.

Leela Kumari Balivada
Department of E.C.E, UCEK
JNTUK, Kakinada
Andhra Pradesh, India.

ABSTRACT
Always technical designers choice includes algorithms, flowcharts, programming etc., and end users requires the given input and application output. Based upon this view, this paper focus on the advancement of the Fast Fourier Transform (FFT), by doing design and observing the performance analysis of the 32 FFT[1] and 64 point FFT, using Radix-2, Radix-8[17] and Split Radix algorithm. The algorithm is developed by Decimation-In-Time (DIT) of the Fast Fourier Transform (FFT), using VHDL as a design entity and synthesis are performed in Xilinx ISE Design Suite 13.2 version. Using synthesis results performance analysis is done between 32 and 64 point Fast Fourier Transform (FFT)[16] in terms of speed and computational complexity.

General Terms
Decimation -In-Time (DIT), Fast Fourier Transform (FFT), VHDL

Keywords
Radix-2, Radix-8, Split-Radix, Synthesis.

1. INTRODUCTION
Currently in the field of signal processing for communications, there is a rapid development evolving programming formats and in algorithms which act as a key in designing a system. Fourier Transform, Discrete Fourier Transform (DFT), Fast Fourier Transform (FFT)[3][7][16] are basis for many signal processing and communication based applications. It is the analysis of the signal in time and frequency domain. From the Fourier Transform of the discrete signals we can develop Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT).Fast Fourier Transform (FFT) is an efficient way to compute DFT.

In digital signal processing frequency analysis of discrete signal can easily performed. In order to obtain the performance of such analysis we have to transform the time domain signal into frequency domain signal. Fast Fourier Transform (FFT) does not do the transformation, but just perform computations to evaluate Discrete Fourier Transform (DFT)[4] efficiently. The original computations for N-point DFT sequence requires N^2 multiplications and N(N-1) additions.

In the year 1965 J.W.Cooley and J.W.Tukey[6] developed FFT algorithm to reduce the computations of the Discrete Fourier Transform (DFT) from N^2 to $\frac{N}{2}\log_2 N$multiplications and N(N-1) to $N\log_2 N$ additions.

The FFT algorithm is also known as Cooley-Tukey algorithm. The pioneering work of J.W.Cooley and J.W.Tukey[6] several algorithms are further developed to reduce the computational complexity, which includes Radix-2,[2]Radix-4,Radix-8,Split-Radix.All these algorithms are developed on one method ,that is, Divide and Conquer method.

Fast Fourier Transform (FFT) is based on decomposition and breaking the transform into smaller sequences and at last again combining into one transform. This paper proposes design of 32 and 64 point FFT and observing performance analysis of 32 and 64 point FFT using different Radix algorithms. By using VHDL[13] as a design entity the program is synthesized in Xilinx ISE Design Suite 13.2 version. A DFT decomposes sequence of values into different frequency components which is useful in many fields like noise reduction, Digital Video Broadcasting (DVB), but computing the sequence directly from the definition is often too slow to be practical.

The computation of DFT involves the multiplication of twiddle factor which is in matrix form by a complex-valued input vector. The N-point Discrete Fourier Transform (DFT) X(k) of an N-point sequence x(n) is by definition:

$$X(k)=\sum_{n=0}^{N-1} x(n)e^{-(\frac{i2\pi nk}{N})} \quad 0\leq n\leq k \qquad (1)$$

2. RADIX-2
The implementation of FFT can be done in Decimation-In-Time FFT (DITFFT) [1][9][17] and Decimation-In-Frequency FFT (DIFFFT) algorithm. Both of the algorithm has same computational complexity but differ in input and output computational arrangement. The name Radix-2[1][2] is called due to its base is equals to 2 and the representation is 2^M, 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 (1) as:

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

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

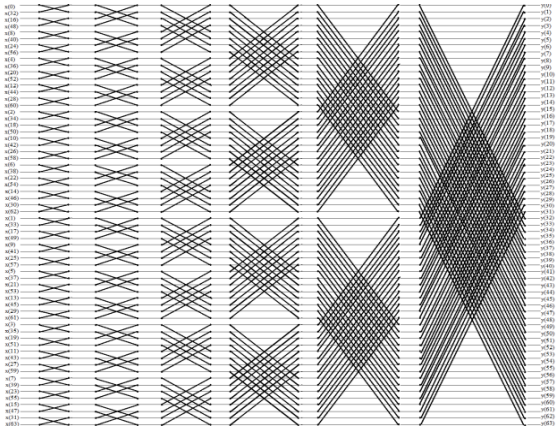

Figure 1: signal flow graph of 64 point Radix-2 FFT

To find the number of butterfly stages required to compute N length sequence is M=log_2^N, and $\frac{N}{2}$ butterfly operations are computed in each stage. Therefore there are 5 butterfly stages and 16 butterfly operations for each stage are computed to produce 32 point FFT [1]. Similarly, 6 butterfly stages and 32 butterfly operations for each stage are computed to produce 64 point FFT.Figure1 shows the signal flow graph representation of 64 point FFT. To compute the DIT-FFT sequence the given input should be in shuffled (bit reversal) order and output represents in normal order.

3. RADIX-8

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 paper 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.

3.1. Functioning of Radix-8 Algorithm

By using the FFT algorithm the computational complexity reduces to $Nlog_r^N$, where r represents the Radix-r FFT[7]. 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.

The Radix-8 Decimation-In-Time and Decimation-In-Frequency Fast Fourier Transform (FFTs) gain their speed by reusing the results of smaller, intermediate computations to compute multiple DFT frequency outputs. The Radix-8 Decimation-In-Time algorithm rearranges the DFT equation into eight parts: sums over all groups of every eighth discrete-time index n=[0,8,16,.....N-8], n=[1,9,17,...N-7],n=[2,10,18,....N-6], n=[3,11,19.....N-5],n=[4,12,20,...N-4], n=[5,13,21....N-3],n=[6,14,18....N-2], n=[7,15,19....N-1],(This works only when the FFT length is multiple of eight). Simply, as in the Radix-2 DITFFT, further mathematical manipulation shows that the length-N DFT can be computed as the sum of the outputs of four length-n/8 DFTs, of the even-indexed and odd-indexed discrete-time samples, respectively, where all the time samples are multiplied by so-called twiddle factors W_N^k=$e^{-(\frac{2i\prod k}{N})}$,$W_N^{2k}$, W_N^{3k},……, W_N^{7k}.

The following equations illustrate Radix-8 DIT-FFT, which the N-point input sequence splits into eighth subsequence's, x(8n), x(8n+1), x(8n+2) ,...x(8n+7), n=0,1,...$\frac{N}{8}-1$. Equation (1) can be written as follows:

$$X(k)=\sum_{m=0}^{\frac{N}{8}-1} x(8m)W_N^{k(8m)}+\sum_{m=0}^{\frac{N}{8}-1} x(8m+1)W_N^{k(8m+1)}+\sum_{m=0}^{\frac{N}{8}-1} x(8m+)W_N^{k(8m+2)}+\sum_{m=0}^{\frac{N}{8}-1} x(8m+3)W_N^{k(8m+3)}+\sum_{m=0}^{\frac{N}{8}-1} x(8m+4)W_N^{k(8m+4)}+\sum_{m=0}^{\frac{N}{8}-1} x(8m+)W_N^{k(8m+5)}+\sum_{m=0}^{\frac{N}{8}-1} x(8m+6)W_N^{k(8m+6)}+\sum_{m=0}^{\frac{N}{8}-1} x(8m+7)W_N^{k(8m+7)}$$

$$=DFT_{N/8}^{[X(8n)]}+W_N^k DFT_{N/8}^{[X(8n+1)]}+W_N^k DFT_{N/8}^{[X(8n+2)]}+W_N^k DFT_{N/8}^{[X(8n+3)]}+W_N^k DFT_{N/8}^{[X(8n+4)]}+W_N^k DFT_{N/8}^{[X(8n+5)]}+W_N^k DFT_{N/8}^{[X(8n+6)]}+W_N^k DFT_{N/8}^{[X(8n+7)]} \quad (3)$$

The basic operation of Radix-8 butterfly is shown in the following figure.

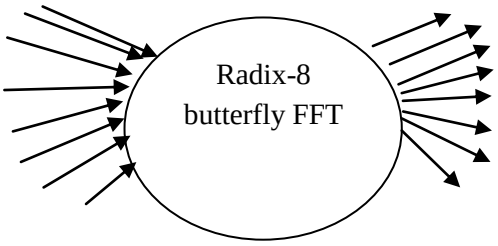

Figure 2: Basic structure of Radix-8 FFT.

Figure 3 depicts a 64-point Radix-8 FFT using the butterfly symbol shown in figure 2 to represent mathematical operations.

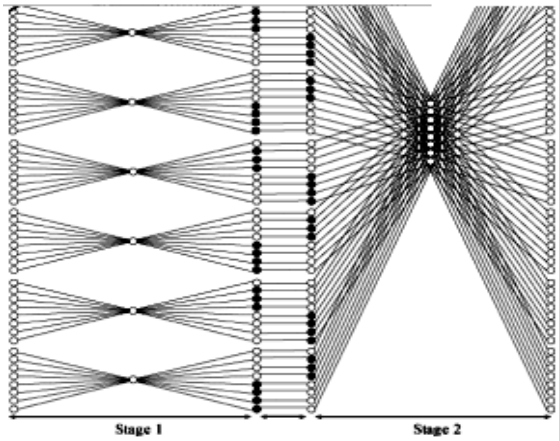

Figure 3:-Signal flow graph of 64 point FFT using Radix-8

The above diagram represents 64-point Radix-8 DITFFT butterfly diagram in which we have only 2 stages, the internal RTL views are as shown below:

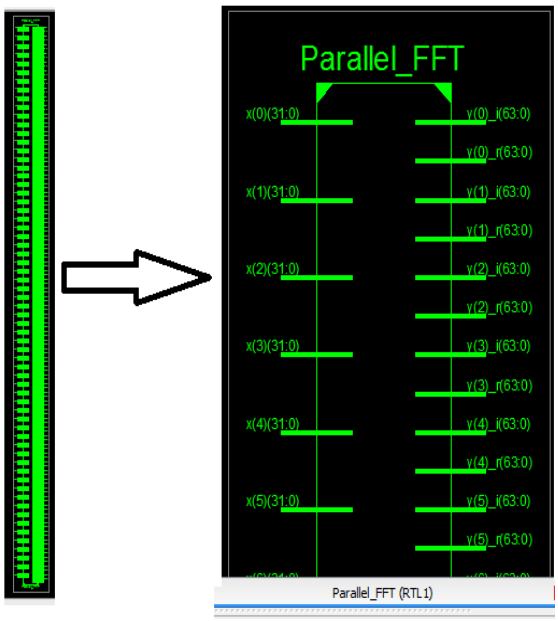

Figure 4:- RTL view for 64 Point FFT

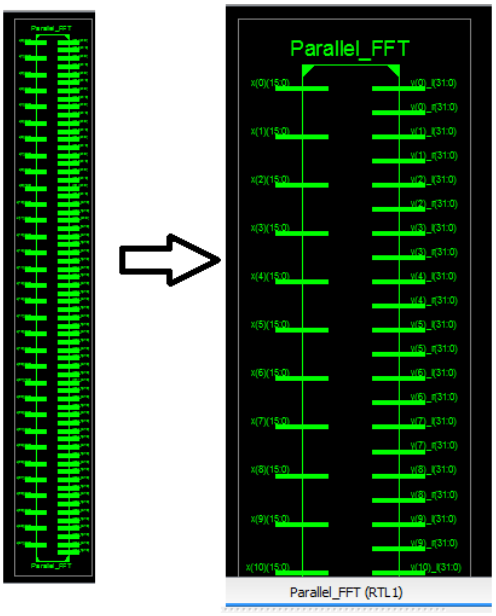

Figure 5:-RTL view for 32 Point FFT

4. SPLIT-RADIX

The split-Radix [10] FFT investigates whether the computational cost of the FFT can be further reduced by combining two or more different radices, resulting in a Split-Radix algorithm. In, as with Radix-2 FFT case, the DFT operation is split into odd-half and even-half parts. In this paper the Split-Radix algorithm is of 32 point is split as 8/4.The even components are decomposed into 8k and odd components are decomposed into 4k+1 frequency components. Repeating this process for half and quarter length DFTs gives the Split-Radix FFT algorithm.

$X(k)= \sum_{n=0}^{N-1} x(n)e^{-\left(i\frac{2\pi nk}{N}\right)}$ 0<n<k

$=\sum_{m=0}^{\frac{N}{8}-1} x(8m)W_N^{k(8m)}+\sum_{m=0}^{\frac{N}{4}-1} x(4m+1)W_N^{K(4m+1)}$

$=\sum_{m=0}^{\frac{N}{8}-1} x(8m)W_N^{k(8m)}+W_N^k \sum_{m=0}^{\frac{N}{4}-1} x(4m)W_N^{K(4m)}$ (4)

By decimating the input vector in time index into two subsets as expressed in equation (5),(6) and calculating with twiddle factors $W_{\frac{N}{8}} = W_N^8, W_{\frac{N}{4}}=W_N^4$.

$x_{even}(n) = x(8m)$ m=0,1,....$\frac{N}{8}-1$ (5)

$x_{odd}(n)=x(4m+1)$ m=0,1,....$\frac{N}{4}-1$ (6)

After these problem subsets are solved by the Split-Radix algorithms recursively, the solution to the original sequence of length N can be obtained according to equation (4). The RTL view of 32 point Split-Radix is shown in below figure 5.

5. SOFTWARE USED

The goal of the VHDL [13] synthesis is to create a design that implements the required functionality and matches the designer's constraints in speed, area and power. The 32 Point FFT[1][14] and 64 point FFT Proposed in this paper is been simulated using Xilinx ISE Design Suite 13.2 with device family as Vertex 6 lower power (XC6VLX130TL),Package FF784 and with speed -1L.

6. SIMULATION RESULTS

The vertex 6 device utilization summary for 32 and 64 point is shown below table 1 and 2.

Table 1: Device utilization summary for 32 point FFT using Radix-2 Algorithm.

Device Utilization Summary for 32 Point FFT using Radix-2 Algorithm			
Logic Utilization	Available	used	Utilization
No. of Slice LUTs	46560	18253	39%
No. of Fully Used LUTFFT Pairs	18253	0	0%
No. of Bonded IOBs	240	2560	1066%
No.of DSP48EIs	288	148	51%

Table 2: Device utilization summary for 32 point FFT using Split-Radix Algorithm.

Device utilization summary for 32 point FFT using Split – Radix Algorithm.			
Logic Utilization	Available	Used	Utilization
No. of Slice LUTs	46560	22120	47%
No. of Fully Used LUTFFT pairs	22120	0	0%
No. Of Bonded IOBs	240	2560	1066%
No. Of DSP48EIs	288	160	55%

Table 3: Time delay of 32 point FFT

32 Point		
Parameter	Radix-2	Split-Radix
Minimum Delay(ns)	21.526	21.153
Total Real Time to XST Completion	162.00ns	137.00ns
Total Real Time to CPU Completion	162.03ns	136.30ns

Table 4: Time delay of 64 point FFT

64 Point	
Parameter	Minimum Delay(ns)
Radix-2	39.618
Radix-8	19.973
Split-Radix	24.795

The performance analysis of 32 and 64 point FFT are shown in below figure.

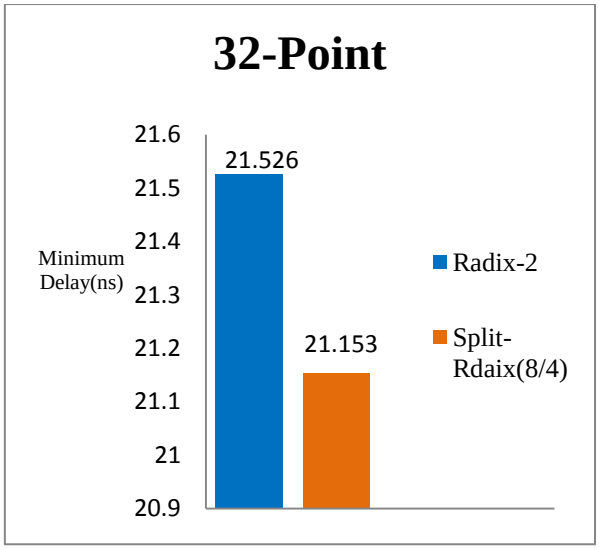

Figure 6: Performance analysis of 32 point FFT

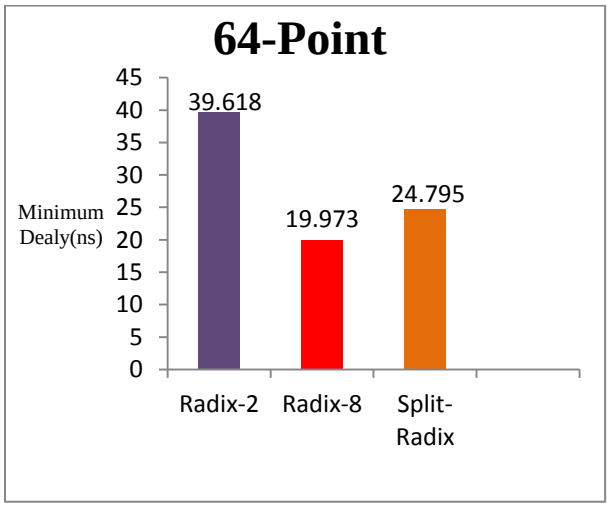

Figure 7: Performance analysis of 64 point FFT

7. CONCLUSION

In this Paper, the design of 32 and 64 point FFT using Radix-2, Radix-8, and Split-Radix algorithms are performed, and the performance analysis with all the three algorithms are done using Minimum Delay(ns) as parameter and their synthesis and simulation results are shown by Xilinx synthesis tool on vertex .The test bench wave forms are displayed by using Xilinx ISE Design Suite 13.2. Further, the performance analysis can also be done by taking various parameters into consideration for different or same number of points.

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 M.I.Eladawy, "Efficient FPGA implementation of FFT/IFFT Processor", international journal of circuits and signal processing,Issue3,Volume3,2009.

[4] Alam V.Oppenhem, Ronald W.Schaler with John R.back , Discrete Time Signal Processing, second Edition.

[5] B.Parhami, Computer Arithmetic, Algorithms and Hardware Designs, 1999.

[6] James W.Cooley and John W.Tukey, An Algorithm for the Machine Calculation of Complex Fourier Series.

[7] Saad Bouguezel, M.Omair Ahmad, "Improved Radix-4 and Radix-8 Algorithms", IEEE Department of Electrical and Computer Engineering Concordia University 1455 de Maisenneuve Blvd west Montreal, P.q.,Canada.

[8] Ali saidi, "Decimation in Frequency FFT Algorithm", Motorola Applied Research, Paging and Wireless Data Group Boynton Beach.

[9] Wei-Hsin chang and Truong Q.Nguyen fellow IEEE, "On the Fixed point Accuracy Analysis of FFT Algorithms",IEEE Transactions ON Signal Processing, Vol. 56, No. 10, Oct 2008.

[10] Jesus Gracia, Juan A. Michell, Gustavo Ruiz, Angel M. Burón, Dept. de Electrónica y Computadores, Facultad de Ciencias, Univ. de Cantabria, Avda, "FPGA realization of a Split Radix FFT processor", Los Castros s/n, 39005 Santander, SPAIN.

[11] HardwareDescriptionLanguage.URL: http://en.wikipedia.org/wiki/Hardware_description_language.

[12] Very High Speed Integrated Circuit Hardware Description Language URL: http://electrosofts.com/vhdl/

[13] Peter J. Ashenden, The Designer's Guide to VHDL, Second Edition.

[14] N. Weste, M. Bickerstaff, T. Arivoli, P.J. Ryan, J. W. Dalton, D.J.Skellern", and T.M. Percivalt "A 50Mhz 16Point-FFT processor for WLAN applications".

[15] Rizalafande Che Ismail and Razaidi Hussin "High Performance Complex Number Multiplier Using Booth-Wallace Algorithm" School of Microelectronic Engineering Kolej University Kejuruteraan Utara Malaysia.

[16] Bergland, G. D. "A Guided Tour of the Fast Fourier Transform." *IEEE Spectrum* **6**, 41-52, July 1969.

[17] "Design of a radix-8/4/2 FFT processor for OFDM Systems", Jungmin Park ,Computer Engineering ,Iowa State University.