ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015

Survey Report for Radix 2, Radix 4, Radix 8 FFT Algorithms

K. Jayaram[1], C. Arun[2]

Assistant Professor, PhD Scholar, St.Peter`s University, Avadi, Chennai, Tamil Nadu, India[1]

Professor, Department of ECE, R.M.K College of Engineering & Technology, Tamil Nadu, India[2]

ABSTRACT: FFT mainly involved important role in electronics & communication field. The various types of FFT radix algorithm have analyzed and is to be modified in future. As a result of its exhaustive computational necessities, it occupies large area and consumes high power if implemented in hardware. Efficient algorithms are developed to improve its architecture.In this paper, we have conversed about FFT Radix 2 algorithm, FFT Radix 4 algorithm and FFT Radix 8 algorithm. For this detailed analysis, power consumption, hardware, memory requirement and throughout of each algorithm have distinguished.

KEYWORDS: DITFFT & DIFFFT Algorithms, Radix 2,4 & 8 Fast Fourier Transform (FFT)

I.INTRODUCTION

Fast Fourier Transform (FFT) is a commonly used technique for the computation of Discrete Fourier Transform (DFT). DFT computations are required in the fields like filtering, spectral analysis etc. to calculate the frequency spectrum . However, direct computation of Discrete Fourier Transform (DFT) requires on the order of N^2 operations where N is the transform size.FFT is used in digital video broadcasting and OFDM systems. The FFT algorithm, first explained by Cooley and Tukey, opened a new area in digital signal processing by reducing the order of complexity of DFT from N^2 to $N \log_2 N$ .There are a number of different 'Fast Fourier Transform' (FFT) algorithms that enable the calculations much faster than a DFT. FFT algorithms used for quick calculation of discrete Fourier transform of a data vector. FFT is used to speed up the DFT, it reduces the computation time required to compute a discrete Fourier transform and improves the performance by factor 100 or more over direct evaluation of DFT.

In Cooley-Tukey radix-2 algorithm, the N point DFT is subdivided into two (N/2) point DFTs and then (N/2) point DFT is recursively divided into smaller DFTs until a two point DFT, whose butterfly is just an addition and a subtraction of input complex numbers. It is the best suitable algorithm for a number N, which is a power of 2. Higher radix algorithms such as radix-4, radix-8, etc can be employed to reduce the complex multiplications but the butterfly structure becomes complex with the multiple input complex adders.

In recent years, there has been some research in the design of multi-path pipelined FFT architecture that provides a high throughput. Many FFT processor architectures are introduced in order to utilize the OFDM transmission, such as a Single path Delay Commutator (SDC), Multipath Delay Commutator (MDC), Single path Delay Feedback (SDF), and Multipath Delay Feedback (MDF). Among the various FFT architectures, the MDF architecture is frequently used as a solution to provide a throughput rate of more than 1 GS/s. In order to reduce the area and power consumption,several FFT algorithms and dynamic scaling schemes have been proposed.

II. LITERATURE SURVEY

2.1 RADIX 2 FFT ALGORITHM

Fast Fourier Transform (FFT) is an efficient method for computing Discrete Fourier Transform (DFT). The DFT of a signal time domain signal x(n) is given in equation.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \ , k = 0,1,2,\ldots,N-1$$ ---------- 1

In the case of the radix-2 Cooley–Tukey algorithm, butterfly is basic block which is shown in Fig. 1.

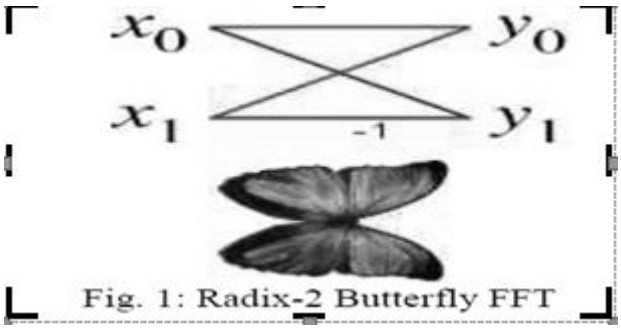

Fig. 1: Radix-2 Butterfly FFT

Figure 1 shows that butterfly is simply a DFT of size-2 that takes two inputs ($x0$, $x1$) (corresponding outputs of the two sub-transforms) and gives two outputs ($y0$, $y1$) by the formula (not includeing twiddle factors):

$$y_0 = x_0 + x_1$$
$$y_1 = x_0 - x_1 \text{ ----------2}$$

If one draws the data-flow diagram for this pair of operations,the (x0, x1) to (y0, y1) lines cross and resemble the wings of a butterfly, hence the name (see also the illustration at right).More specifically, a decimation-in-time FFT algorithm on n = 2 p inputs with respect to a primitive n^{th} root of unity relies on O (n log n) butterflies of the form:

$$y_0 = x_0 + x_1 w^K$$
$$y_1 = x_0 - x_1 w^K \text{ ---------- 3}$$

2.2 4 POINT RADIX 2 FFT

4-point transform can be reduced to two 2-point transforms: one for even elements, one for odd elements. The odd one will be multiplied by W4k. Diagrammatically; this can be represented as two levels of butterflies.

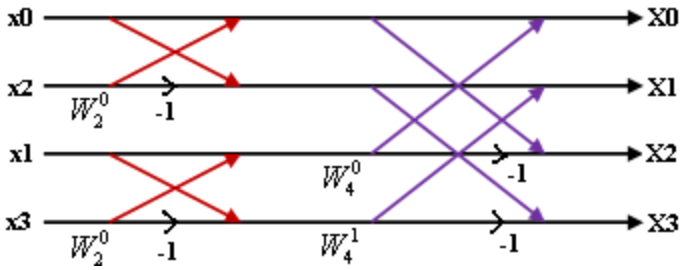

Fig 2: 4 point Radix 2 FFT

The Figure 2 shows that 4 point Radix 2 FFT butterfly diagram to compute the DFT points. The radix 2 algorithms are the simplest FFT algorithms. The decimation-in-time (DIT) radix 2 FFT recursively partitions a DFT into two half-length DFTs of the even-indexed and odd-indexed time samples. The outputs of these shorter FFTs are reused to compute many outputs, thus greatly reducing the total computational cost. The radix 2 decimation-in-time and

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015

decimation-in-frequency fast Fourier transforms (FFTs) are the simplest FFT algorithms. Like all FFTs, they gain their speed by reusing the results of smaller, intermediate computations to compute multiple DFT frequency outputs.

2.2. RADIX 4 FFT ALGORITHM

Radix-4 is another FFT algorithm which was surveyed to improve the speed of functioning by reducing the computation; this can be obtained by change the base to 4. For a same number if base increases the power/index will decreases. For radix-4 the number of stages are reduced to 50% since N=43 (N=4M) i.e. only 3 stages. Radix-4 is having four inputs and four outputs and it follows in-place algorithm. The radix-4 DIT-FFT recursively partitions a DFT into four quarter-length DFTs of groups of every fourth time sample. The outputs of these shorter FFTs are reused to compute many outputs, thus greatly reducing the total computational cost. The radix-4 FFTs require only 75% as many complex multiplications as the radix-2 FFTs. The radix-4 decimation-in-time and decimation-in-frequency Fast Fourier transforms (FFTs) gain their speed by reusing the results of smaller, intermediate computations to compute multiple DFT frequency outputs.

In Radix-4 FFT Algorithms N point input signal are decompose in to following equation. The equations are derived in following manner.

$$X(4k)=\sum_{n=0}^{N/4-1}[x(n)+x(n+\frac{N}{4})+x(n+\frac{N}{2})+x(n+\frac{3N}{4})]W_N^0 W_{N/4}^{kn} \quad \text{----------} \quad (4)$$

$$X(4k+1)=\sum_{n=0}^{N/4-1}[x(n)-jx(n+\frac{N}{4})-x(n+\frac{N}{2})+jx(n+\frac{3N}{4})]W_N^n W_{N/4}^{kn} \quad \text{----------} \quad (5)$$

$$X(4k+2)=\sum_{n=0}^{N/4-1}[x(n)-x(n+\frac{N}{4})+x(n+\frac{N}{2})-x(n+\frac{3N}{4})]W_N^{2n} W_{N/4}^{kn} \quad \text{----------} \quad (6)$$

$$X(4k+3)=\sum_{n=0}^{N/4-1}[x(n)+jx(n+\frac{N}{4})-x(n+\frac{N}{2})-jx(n+\frac{3N}{4})]W_N^{3n} W_{N/4}^{kn} \quad \text{----------} \quad (7)$$

Where k = 0, 1 ----N/4 – 1 and used the property $W^{4kn}N = W^{kn}$ N/4 ,X (4r), X (4r+1), X (4r+2), and X (4r+3) are N/4-point DFTs. Every N/4 output is a sum of four input samples all multiplied by -1, j, or +1 –j. Twiddle Factors are multiplied by above sum. Every N/4-sample DFTs is divided into four N/16- sample DFTs. Every N/16 DFT is distributed further in four N/64-Pt.and so on.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015

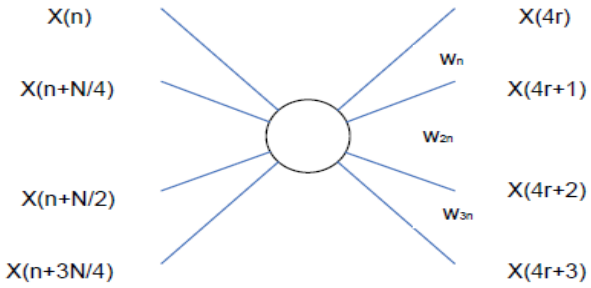

Fig 3: Radix 4 FFT algorithm

A basic radix-4 Processing element (Butterfly) is represented in Figure 3. These are the expression of Radix 4 FFT algorithms. The radix 4 Butterfly contains 3 complex multiplications and 12 complex additions N/4 butterfly involves in each stage and number of stage is log4N for N-point sequence. Therefore, the number of complex multiplications is 3N/4log4N and number of complex additions is 12N/log4N. In comparisonof radix 2 FFT, number of complex multiplications are reduce by 25% but number of complex additions are increased by 50%.

2.3. RADIX 8 FFT ALGORITHM

Radix 8 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. the number of stages are reduced to 75% since $N=8^2$ that is; only 2 stages. By using the FFT algorithm the computational complexity reduces to where r represents the Radix-r FFT. 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).

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,... . Equation can be written as follows:

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

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

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015

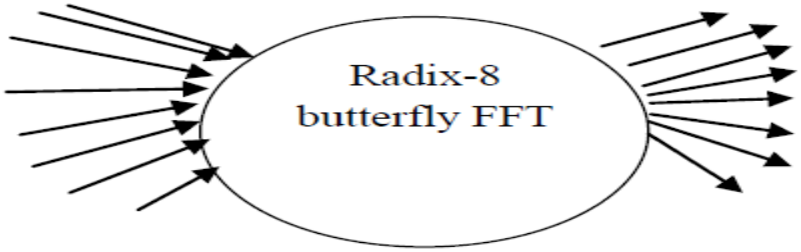

Figure 4: Basic Radix 8 butterfly FFT.

The basic operation of Radix 8 butterfly is shown in the following figure. The figure 4 shows the computation of Radix 8 FFT can be achieved by applying the algorithm in efficient way.

III. COMPARISON OF DIFFERENT RADIX ALGORITHM

The comparison of different radix algorithms is done based on the hardware requirements. Here Radix 2,Radix 4 and Radix algorithms is analysed.

Operations	Radix 2	Radix 4	Radix 8
Real addition	1032	976	972
Real multiplication	264	208	204
Total	1296	1184	1176

Table 1.Comparison table of Different Radix Algorithms

From the Table 1,the comparison is made between the previous pipelined architectures and the proposed ones for the case of computing an N-point complex FFT. The comparison is made in terms of required number of complex multipliers, adders, delay elements, throughput and control complexity.From this analysis,the requirement of hardware is very minimum for Radix 8 algorithm.

IV. CONCLUSION

In this Paper, the design FFT using Radix-2, Radix-4, and Radix-8 algorithms are performed, and the performance analysis with all the three algorithms are done using Minimum Delay as parameter and their synthesis .Further, the performance analysis can also be done by taking various parameters into consideration for different or same number of points. In future the Radix 8 FFT architecture can be designed for real valued signals.

REFERENCES

[1]. Sudha Kiran G , Brundavani P " FPGA Implementation of 256-Bit, 64-Point DIT-FFT Using Radix-4 Algorithm " International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 9, September 2013.
[2]. K.Sowjanya,Leela Kumari Balivada," Design and Performance Analysis of 32 and 64 Point FFT using Multiple Radix Algorithms", International Journal of Computer Applications (0975 – 8887) Volume78– No.1, September 2013.
[3]. 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.
[4]. 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.
[5]. 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.
[6]. Shousheng He and Mats Torkelson, "Designing Pipeline FFT Processor for OFDM (de)Modulation", IEEE Signals, Systems, and Electronics, Sep. 1998,pp. 257-262.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015

[7]. Shousheng He and Mats Torkelson, "A New Approach to Pipeline FFT Processor", IEEE Parallel Processing Symposium, April. 1996, pp.776-780.
[8]. C. Sidney Burrus, "Index Mapping for Multidimensional Formulation of the DFT and Convolution", IEEE Trans. Acoust., Speech, and Signal Processing, Vol. ASSP-25, June. 1977, pp. 239-242.
[9]. Lihong Jia, Yonghong GAO, Jouni Isoaho, and Hannu Tenhunen, "A New VLSI-Oriented FFT Algorithm and Implementation", IEEE ASIC Conf., Sep.1998, pp. 337-341.
[10]. Martin Vetterli and Pierre Duhamel, "Split-Radix Algorithms for LengthptmDFT's", IEEE Trans. Acoust, Speech, and Signal Processing, Vol. 37, No.1, Jan. 1989, pp. 57-64.
[11]. Daisuke Takahashi, "An Extended Split-Radix FFT Algorithm", IEEE Signal Processing Letters, Vol. 8, No. 5, May. 2001, pp. 145-147.
[12]. Y.T. Lin, P.Y. Tsai, and T.D. Chiueh, "Low-power variable-length fast Fourier transform processor", IEE Proc. Comput. Digit. Tech, Vol. 152, No. 4, July.2005, pp. 499-506.