AN ENERGY AND AREA EFFICIENT ARITHMETIC UNITS OF RADIX-4 AND RADIX-8 BUTTERFLY STRUCTURE USING SPLIT RADIX FFT

V.Vasudhevan[1], A.Hajara[2], R.Gomathy[3], N.Nivethaa[4]

V.Vasudhevan[1],Asst. Prof, Panimalar Engineerimg College, Tamil Nadu, India
A.Hajara[2], Final yr, Panimalar Engineerimg College, Tamil Nadu, India
R.Gomathy[3], Final yr, Panimalar Engineerimg College, Tamil Nadu, India
N.Nivethaa[4], Final yr, Panimalar Engineerimg College, Tamil Nadu, India

ABSTRACT

During this paper, we tend to gift a category of high-radix butterfly parts that utilize (m, n)counters to exchange the adders in typical realization of butterfly parts. With these butterfly parts, we tend to cut back the hardware quality, delay time, and also power consumption.

Keyword : - *Decimation in time (DIT), Fast fourier transform (FFT), VHDL, Radix-2, Radix-4, Radix-8, Split-Radix FFT.*

1. INTRODUCTION

FFT has been the foremost necessary algorithms in digital signal process. Within the recent years, FFT has been applied in the fashionable communication systems because of its potency for OFDM (Orthogonal Frequency Division Multiplex) implementation. Several applications, like xDSL modems, HDTV, mobile radio terminals, use FFT processor as a key part.

The butterfly elements are the essential building blocks in the associate FFT processor. Since FFT processors employing a radix-4 design has fewer multiplications than the radix-2 design, the radix-4[5][6] architectures are usually used for FFT processors. Higher radix tend to cut back the operation rate, arithmetic work, and hence the power consumption. Economical style of high-radix butterfly parts is so necessary.

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

2. EXISTING SYSTEM
RADIX-2

The implementation of FFT can be done in both Decimation-InTime FFT (DITFFT) [1] and Decimation-In-Frequency FFT (DIFFFT) algorithm according to our convenience. Both the algorithm has same computational complexity but differ in input and output computational arrangement. The name Radix-2[1] is called because of its base equal to 2 and its representation as 2^M, where M represents the 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:

$=\sum_{n(even)} x(n)W_N^{kn}+\sum_{n(odd)} x(n)W_N^{kn}$ (1)

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

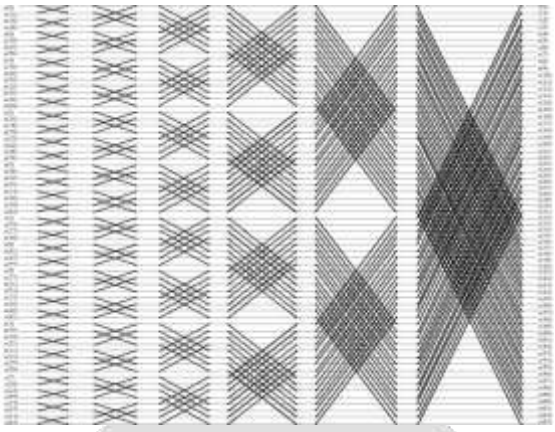


Figure 1: signal flow graph of 64 point Radix-2 FFT

3. PROPOSED SYSTEM

RADIX-4

Radix-4 algorithm decomposes a N-point DFT into a four (N/4)-point DFT. This algorithm requires only half as many stages as radix-2 algorithm requires. Number of stages in radix-4 is reduced and are $\log_4 N$.
This algorithm involves $(3N/8)\cdot \log_2 N$ complex multiplication and $(3N/2)\cdot \log_2 N$ complex additions. N/4 butterflies are used in each $(\log_2 N)/2$ stages, which is one quarter the number of butterflies in a radix-2 FFT. The radix-4 butterfly is consequently larger and more complicated when compared to the conventional radix-2 butterfly. Addressing of data and twiddle factors are more complex. The overall number of operations is lower and consumes less time.

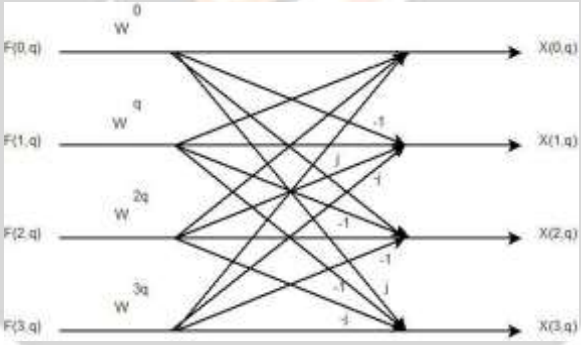


Fig 2: A Simple Radix 4 DIF FFT algorithm

3.1 ARCHITECTURE OF RADIX-4 FFT BUTTERFLY

For an N-point sequence, the radix-4 FFT[7] algorithm takes 4 data points at a time from memory, performing the butterfly computation and returns the result to the memory. This procedure is repeated many times, i.e., $((N\log_4 N)/4)$ times in the computation of an N-point DFT. Therefore, the memory requirement is an essential factor for the FFT. processor design. The requirement of memory size is 2N[4] for the input sequence which is complex number and 2N for the output sequence. Hence the capacity of memory for an N-point FFT processor is 4N.

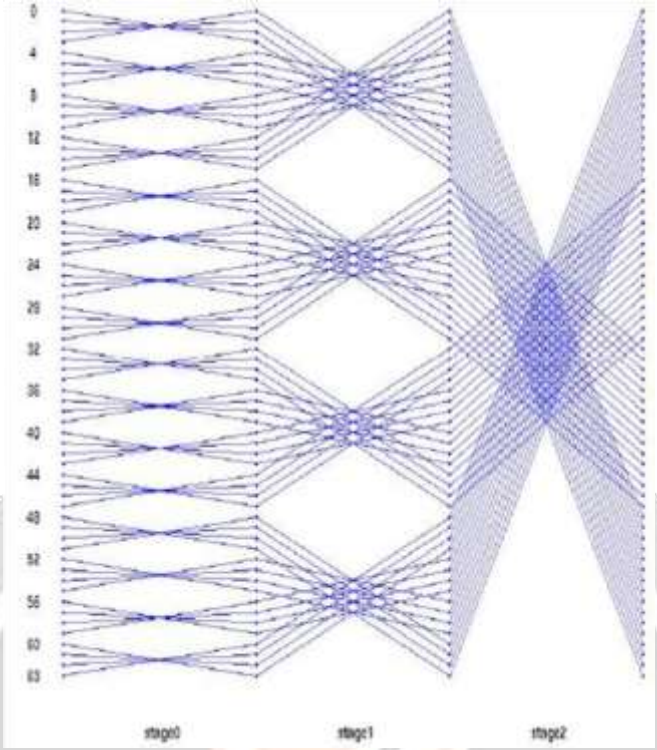


Fig 2.1: 64point DIT FFT in Radix-4

4. RADIX-8

Radix-8[6] is another FFT algorithm which was purposely used to improve the speed of functioning by reducing the computation, delay and power; this can be obtained by changing base 4 to 8. For a same number if the base increases the power will gradually reduce.

In this tecnique the number of stages are reduced to 75% since N=8^2(N=8^m) which means only 2 stages are involved. The following will explain the functioning of Radix-8 algorithm and how the computational complexity is reduced.

4.1 FUNCTIONING OF RADIX-8 ALGORITHM

By using the FFT algorithm the computational complexity is reduced to$Nlog_r^N$, where r represents the Radix-r FFT[7]. The Radix-r FFT can be easily derived from DFT by decomposing the N point DFT into a set of recursively related r-point transform and x(n) in powers of r. In Radix-8 algorithm r is 8. The DIT Radix-8 FFT recursively partitions a DFT into an eight quarter-length DFTs of groups of every eighth sample. The outputs of these shorter FFTs are reused in order to compute many outputs, which greatly reduces the total computational cost.

The Radix-8 Decimation-In-Time (DIT)[5] and Decimation-In Frequency(DIF) 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[4] algorithm rearranges the DFT equation into eight parts and then sums the overall groups of every eighth discrete time index where 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]. 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

$$W_N^k = e^{-(\frac{2\Pi k}{N})}, W_N^{2k}, W_N^{3k}, \ldots\ldots, W_N^{7k}.$$

The following equations illustrate Radix-8 DIT-FFT[7], 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.

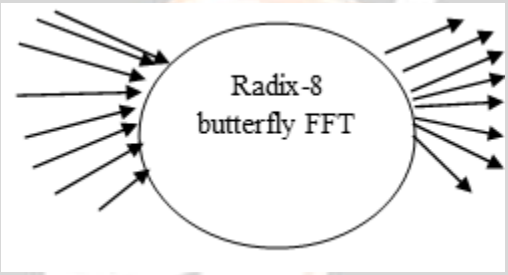


Fig 3: Basic structure of Radix-8 FFT.

Figure 4 depicts a 64-point Radix-8 FFT[8] using the butterfly symbol shown in figure 2 to represent mathematical operations.

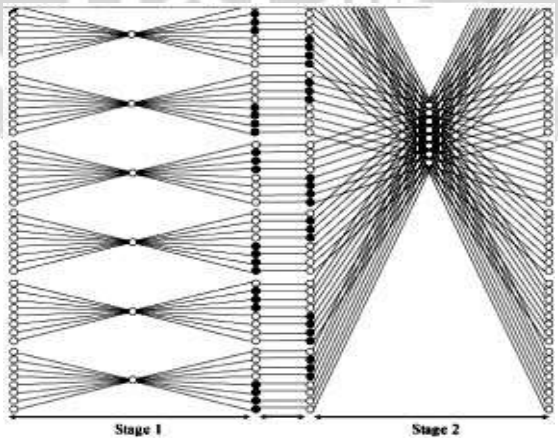
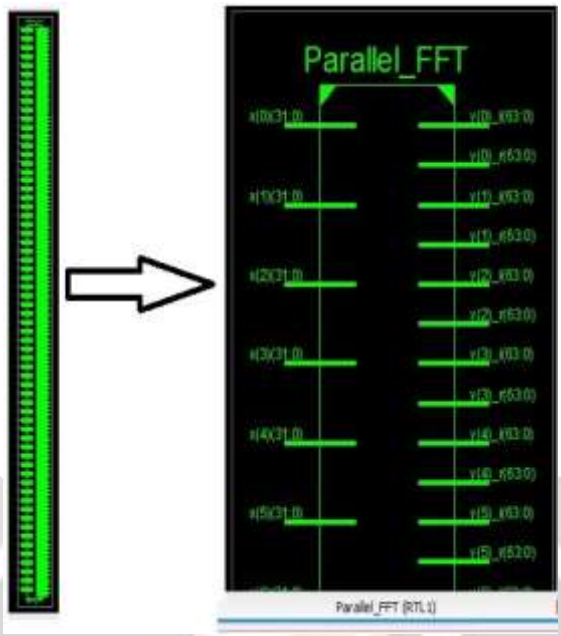


Figure 4:-Signal flow graph of 64 point FFT using Radix-8

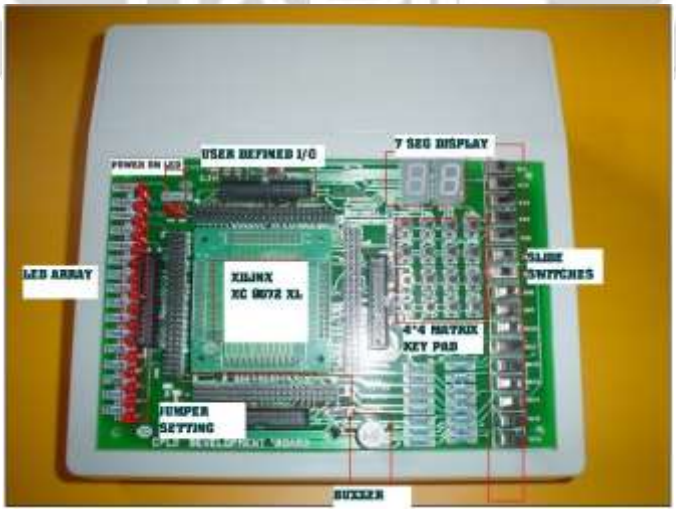
The above diagram represents the 64-point Radix-8[3] DITFFT butterfly diagram in which we have only 2 stages, the internal RTL views are as below:



5. HARDWARE DESCRIPTION

Programmable Logic Devices (PLDs) offer wide range of logic capacity, features, speed and voltage characteristics and these devices can be changed at any time to perform any number of functions. The concept is to have a few PLD[2] blocks or macro cells on a single device with general purpose interconnect in between. Simple logic paths can be implemented within a single block. More sophisticated logic will require multiple blocks and use the general purpose interconnect in between to make these connections. CPLDs are great at handling wide and complex gating at blistering speeds. e.g., 5ns which is equivalent to 200MHz. the timing model for CPLD[1] is easy to calculate, so even before design it can calculate the in to output speeds. CPLDs enable ease of design, lower development costs, more product revenue for money, and the opportunity to speed your products to market, etc.

5.1 CPLD LAYOUT BOARD



6. INSTALLATION

6.1 SOFTWARE

Use ISE Webpack 6.1 provided along with this CPLD KIT. Use the 12 digit Key for installing ISE which is given on the CD cover

6.2 HARDWARE

- Connect the 3-pin power supply cable to the 230V power supply.
- Connect the parallel port (LPT1) from PC to the parallel port connector at the back side of the KIT.

7. CONCLUSION

From the experiments it is observed that radix-2 occupies more area, delay and also consumes more power when compared to radix-4 and radix-8

8. 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. B.Parhami, Computer Arithmetic, Algorithms and Hardware Designs, 1999.
3. James W.Cooley and John W.Tukey, An Algorithm for the Machine Calculation of Complex Fourier Series.
4. 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.
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