A Mixed Decimation MDC Architecture for Radix 2^2 4-Parallel FFT

[1]Syeda Husna Saba, [2]Dr.Ayesha Naaz
[1]Student, [2]Professor
[1]Electronics and Communication Engineering,
[1]Muffakham Jah College Of Engineering And Technology, Hyderabad, India

Abstract: This paper presents mixed decimation multipath delay Commutator (M2DC) for radix 2^2 4 parallel Fast Fourier Transform. Generally, the algorithms used are either Decimation In Time (DIT) or Decimation In Frequency (DIF) FFT in which additional clocks are required for reordering input and output bits which increases latency. Hence the concept of mixed decimation i.e. integrating DIT and DIF is proposed for 16 points and the 4 parallel radix 2^2 feed forward (MDC) architecture is implemented which increases the throughput. Therefore the proposed work provides higher throughput and lower latency.

***IndexTerms* – Decimation in Time, Decimation in Frequency, Mixed Decimation, Multipath Delay Commutator.**

I. INTRODUCTION

Fast Fourier Transform (FFT) is dependably an acknowledged subject for investigate from past numerous years for various applications in computerized framework. FFT is a standard calculation as it is computationally proficient over Discrete Fourier Transform (DFT) which is generally utilized as a part of different utilizations of computerized radar, biomedical, wireless communication etc. As FFT has huge size of computational necessities, it uses large area and consumes more power when actualized on hardware equipment. In this manner, a productive and precise execution of FFT processor is constantly favored. Advancement of digital systems is continuously replacing the old analog systems. Digital systems are the backbone of modern day organizations, products, processes and services and their quality is increasingly subject to these systems. Applications of present day digital systems include communication systems, traffic systems, control systems, weather forecasting systems, internet and so forth. Nowadays, computerized frameworks are set up to go up against the necessities of the most requesting DSP applications, which force strong conditions, for example, little silicon region, clock recurrence, high throughput, diminished power utilization, latency and constant calculations. So as to satisfy these necessities, advanced frameworks are being actualized on Application Specific Integrated Circuits (ASICs) and Field Programmable Gate Arrays (FPGAs) gadgets. .

Therefore, an efficient implementation of FFT has attracted much attention and hardware designers have put forward various schemes to achieve reasonable tradeoffs between area and performance. By shortening the critical path in the signal diagram, pipelined architectures have an inherent advantage over other efficient hardware structures in providing high throughputs. To cope with the gradual increase of real-time business, plenty of published works have focused on designing and optimizing the pipelined structures. To meet the requirements of real time applications such as high performance, hardware designers have tried to implement efficient architectures for computation of the FFT. Therefore, pipelined hardware architectures are used as they provide high throughputs and low latencies which are suitable for real time, providing reasonably low area and power consumption.

The types of available pipelined architectures: feedback (FB) and feed forward (FF). Contrary, feedback architectures are characterized by their feedback loops, i.e., some outputs of the butterflies are fed back at the same stage. FB architectures are differentiated into single-path delay feedback (SDF) [4], which processes the continuous flow of one sample per clock cycle and (MDF) or parallel feedback [7] that process several samples in parallel and the FF, also known as multi-path delay commutator (MDC) [2], do not contain feedback loops and each stage passes the processed data to the next stage. Several samples in parallel can be processed in these architectures. In current real-time applications, the FFT has to be calculated at very high throughput rates, even in the range of Giga samples per second. These high-performance requirements appear in applications such as orthogonal frequency division multiplexing (OFDM), and ultra wideband (UWB).

The two main challenges can be differentiated. Initial is to calculate the FFT of multiple independent data sequences. For this, all FFT processors can share the rotation memory in order to reduce the hardware. Designs that manage a variable number of sequences can also be obtained. The second challenge is to calculate the FFT when several samples of the same sequence are received in parallel. This must be done when the required throughput is higher than the clock frequency of the device. This way it is needful to go to FFT architectures that serve several samples in parallel. As a result, parallel feedback architectures, which had not been considered for several decades, have become very popular in the last few years. Conversely, not very much attention has been paid to feed forward (MDC) architectures. In this work it is presented the radix- 2^2 feed forward FFT architectures. Radix- 2^2 FFT architectures for 4 parallel samples are presented. These architectures are shown to be more hardware-efficient than previous feed forward and parallel feedback architectures in the literature. This makes them very relevant for the computation of the FFT in the most demanding applications.

II. DISCRETE FOURIER TRANSFORM

The DFT is the most imperative discrete transform, used to perform Fourier examination in numerous down to practical applications. In DSP, the capacity is any amount or flag that changes after some time, for example, the weight of a sound wave, a radio signal, or day by day temperature readings, tested over a limited time interim. In picture handling, the examples can be the estimations of pixels along a line or segment of a picture. The DFT is likewise used to productively tackle fractional differential conditions, and to perform different activities, for example, convolutions or increasing huge whole numbers. DFT assumes a key part in an extensive variety of signal processing applications since it can be utilized as a numerical instrument to depict the

connection between the time-area and recurrence space portrayals of discrete signs. The DFT X[k] of a sequence x[n] of N terms is defined as:

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

where the sequence x[n] is viewed as N consecutive samples, x[nT], of a continuous signal x(t), and ω = e-j2π/n is the twiddle factor. Similar to DFT there exist inverse DFT (IDFT) which maps a frequency sequence to its corresponding time sequence. The IDFT finds its usefulness in circular convolution where the inverse DFT of the product of two-time sequences corresponds to circularly convolving the two-time sequences.

III FAST FOURIER TRANSFORM:

The Fourier transform is one of the most commonly applied tools used for altering a function form the time domain to the frequency domain. For digital signal processing (DSP), the discrete Fourier transform (DFT) algorithm is used extensively for Fourier analysis, though never computed directly because of its complexity. Instead a collection of efficient algorithms where developed by Cooley and Tukey to speed up the DFT computations considerably. The Fast Fourier transform (FFT) provides a "divide and conquer" methods to estimate the complex DFT algorithms.

FFT is a method developed in 1965 by James q. Cooley and John w. Tukey. FFT is defined as an efficient class of computational algorithms (or a method) for computing the discrete Fourier transform (DFT) efficiently of the sequence of n numbers FFT is the fast implementations of DFT which relies on mathematical simplifications from simple complex number arithmetic to group theory and number theory and classifications of the input sequence to achieve their performance gain. It increases DFTs operational efficiency 1~2 orders of magnitude.

3.1 Decimation In Time Fast Fourier Transform (DIT FFT):

The equation DFT X[k] of a sequence x[n] of N terms is defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^k \qquad (3.1)$$

Algorithms which decompose the x[n] sequences into successive smaller sequences are called Decimation in time (DIT) algorithms. The main principle of DIT FFT algorithm is illustrated as follows, when $N=2^v$.

Here X(k) is computed by dividing x[n] into two N/2 sequences i.e. even and odd indexed sequences. Rewrite the equation (3.1) as follows

$$X(k) = \sum even\ \text{x[n]} W_N^{nk} + \sum odd\ \text{x[n]} W_N^{nk} \qquad (3.2)$$

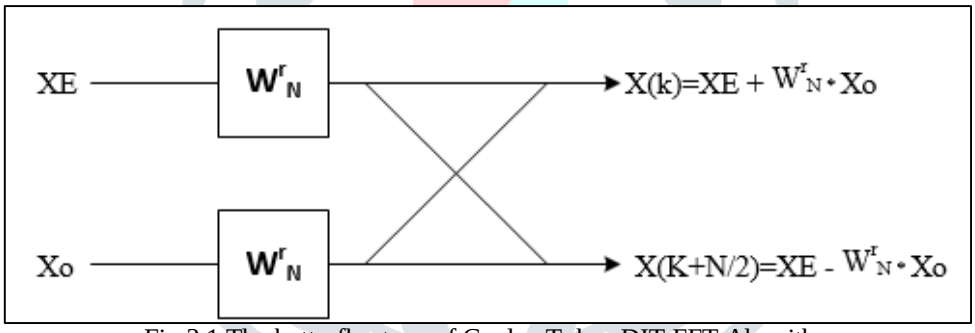

Fig 3.1 The butterfly stage of Cooley Tukey DIT FFT Algorithm

In a similar bundle, three operations were characterized all together to play out the required FFT activities. Here the main work of addition includes adding to that of two complex numbers, this operation includes the real and complex parts independently, exploiting the complex information write as complex numbers segments are in various registers and the subtraction work subtracts two complex numbers and works as the addition.

The last characterized work was multiplication, this capacity increases two complex numbers. For the multiplication work, the way that the augmentation of two 16-bit twofold numbers brings about a 32 bit paired number and that the twiddle factor (W_N) is a fragmentary number considered. At that point, the duplication comes about considered were from 29 to 14 bits with a specific end goal to keep up the 16 bit portrayal and keep working with whole number. Now substitute *n=2m*for even and *n=2m+1* for odd indices of x[n] sequence as

$$X(k) = \sum_{m=0}^{\frac{N}{2}-1} x[2m]\, W_N^{nk} + \sum_{m=0}^{\frac{N}{2}-1} x[2m+1]\, W_N^{(2m+1)k} \qquad (3.3)$$

Equation (3.3) is rewritten as

$$X(k) = \sum_{m=0}^{\frac{N}{2}-1} x[2m]\, (W_N^2)^{mk} + \sum_{m=0}^{\frac{N}{2}-1} x[2m+1]\, (W_{N/2})^{mk} \qquad (3.4)$$

Finally, the equation 3.3 can be rewritten as,

$$X(k) = \sum_{m=0}^{\frac{N}{2}-1} x[2m]\, (W_{N/2})^{mk} + \sum_{m=0}^{\frac{N}{2}-1} x[2m+1]\, (W_{N/2})^{mk} \qquad (3.5)$$

3.2 Decimation In Frequency Fast Fourier Transform (DIF FFT):

Decimation in Frequency (DIF) algorithms are built on structuring the DFT computation by decomposing *X[k]* i.e. output into smaller sub sequences. For DIF algorithms, reconsider N as a power of 2 and individual computation of even and odd indexed samples.

Since, the DFT *X[k]* for an input sequence x[n] is defined by the following equation.

$$X(K) = \sum_{n=0}^{N-1} x(n) W_N^k \qquad \text{K=0,1,2,….N-1} \qquad (3.6)$$

And even indexed samples are

$$X(2K) = \sum_{n=0}^{N-1} x(n) W_N^{(2n)k} \qquad K=0,1,\ldots.N/2-1 \qquad (3.7)$$

Rewrite equation (3.7) as follows

$$X(2K) = \sum_{n=0}^{\frac{N}{2}-1} x[n]\, W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x[n+N/2]\, W_N^{\left(n+\frac{N}{2}\right)(2nk)} \qquad (3.8)$$

The previous equation can also be expressed as

$$X(2K) = \sum_{n=0}^{\frac{N}{2}-1} x[n]\, W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x[n+N/2]\, W_N^{\left(n+\frac{N}{2}\right)(2k)} \qquad (3.9)$$

Since W_N^{2nk} is periodic, we have

$$W_N^{2k(n+N/2)} = W_N^{2nk} W_N^{nk} = W_N^{2nk} \qquad (3.10)$$

And $W_{N^2} = W_{N/2}$ (3.11)

By using equation (3.9) and (3.10), the equation (3.7) can be written as

$$X(2K) = \sum_{n=0}^{N-1} (x(n) + x[n+N/2]) W_{N/2}^{kn} \qquad (3.12)$$

Equation (3.12) represents the N/2 point DFT by summation of first and second halves of the input sequence.

Now, consider the odd indexed samples:

$$X(2K+1) = \sum_{n=0}^{N-1} x(n) W_N^{(2k+1)n} \quad k=0,1,2,\ldots N-1 \qquad (3.13)$$

Equation (3.13) can be re written as follows

$$X(2K+1) = \sum_{n=0}^{N-1} x(n) W_N^{(2k+1)n} + \sum_{n=N/2}^{N-1} x(n) W_N^{(2k+1)n} \qquad (3.14)$$

Then the second half of equation (3.14) can be expressed as

$$\sum_{n=\frac{N}{2}}^{N-1} x(n) W_N^{(2k+1)n} = \sum_{n=0}^{\frac{N}{2}-1} x\left(n+\tfrac{N}{2}\right) W_N^{(2k+1)\left(n+\frac{N}{2}\right)}$$

$$= W_N^{(2k+1)n/2} \sum_{n=0}^{N/2-1} x(n+N/2) W_N^{(2k+1)n} \qquad (3.15)$$

Since $W_{N^{N/2(2k)}} = 1$, $W_{N^{N/2}} = -1$,

$$W_N^{N/2(2k)} \sum_{n=0}^{N/2-1} x(n+N/2) W_N^{(2k+1)n} = -\sum_{n=0}^{N/2-1} x(n+N/2) W_N^{(2k+1)n} \qquad (3.16)$$

By using equation (3.16) in equation (3.14),

$$X(2K+1) = \sum_{n=0}^{N/2-1} (x[n] - x[n+N/2]) W_N^{(2k+1)n} \qquad (3.17)$$

And

$$X(2K+1) = \sum_{n=0}^{N/2-1} (x[n] - x[n+N/2]) W_{N/2}^{kn} W_N^n \qquad (3.18)$$

Hence the mathematical form of DIF FFT and DIT FFT is shown above and it also represents the even and odd samples of FFT.

3.3 Mixed decimation (DIT&DIF) algorithm:

Since its introduction by Cooley and Tukey, the FFT algorithm has taken various configurations, including the familiar decimation-in-time (DIT), decimation-in frequency (DIF), iso geometric configuration, etc. As the computation proceeds through the successive stages in a DIT algorithm, the number of, a non trivial multiplication increases. The first two stages are free of multiplications, while the last stage has most. In contrast, the number of nontrivial multiplications decreases as the computation progresses through the stages in the DIF algorithm. Recently, K. Nakayama combined the DIT and DIF algorithms into a computationally more efficient new algorithm which also preserves the simple butterfly structure. In the next section we present a derivation of this Mixed Decimation FFT algorithm (MDFFT), which is somewhat more straightforward and slightly more general than Nakayama's original development.

The most commonly accepted approach to compute a two dimensional (2D) DFT is to first transform all rows and then all columns (row-column algorithm). E. I-Ioyer and W. Berry and D. Harris et al. proposed a more efficient approach by decimating rows and columns simultaneously. For later references, we shall call this the Simultaneous Decimation FFT (SDFFT). For the radix 2 case the saving in real multiplications over the row-column algorithm is 25% for square arrays and up to 23% for rectangular arrays of reasonable size. In Section III, we derive a new 2D FFT algorithm, by decimating rows and columns simultaneously and by employing both DIT and DIF at the same time. The new algorithm is called Mixed Simultaneous Decimation FFT (MSDFFT). It exhibits asymptotically the same arithmetic complexity as SDFFT, but for arrays of reasonable size it gives a saving in real multiplications of 25 to 30%. In section IV we compare this algorithm to some commonly used 2D DFT algorithms. Besides the row column and the SDFFT's, we examine the 2D DFT algorithm which is obtained by applying the approach of Rader and Brenner and the 2D Winograd Fourier Transform Algorithm (WFTA)[6]. Although asymptotically the Rader-Brenner algorithm is more efficient than MSDFFT, this is not true for moderate array sizes. It also has high round-off error. For some array sizes, the 2D WFTA requires less real multiplications than the MSDFFT, but for some others the opposite is true. The advantage of MSDFFT is its simple structure since it uses the butterfly. If the DIF and DIT FFT algorithms are combined, the input data and output data sequence can occur in the natural order or in an order with simple rule, for example, even and odd. Fig. 3.3 shows the 8-point Radix-2 Mixed DIF/DIT FFT algorithm. Since the final stage is computed based on Radix-2 DIT algorithm, output sequence occurs in normal order, and the input sequences are broken into simple even and odd sequences as shown in Fig. 3.3. The shaded box in Fig. 3.3 represents 4-point DFT. Since the 4-point DFT is computed based on DIF FFT algorithm, the order of input data sequences does not change any more. From Fig.3.3, the transition from DIF to DIT is required since the output sequence of 4-point DFT with Radix-2 DIF FFT is different from the input sequence of 2 point DFT with Radix-2 DIT FFT used at the final stage. This transition can be done by changing the sequence as shown in Fig. 3.3 and it can be implemented as commutator with storage register or memory.

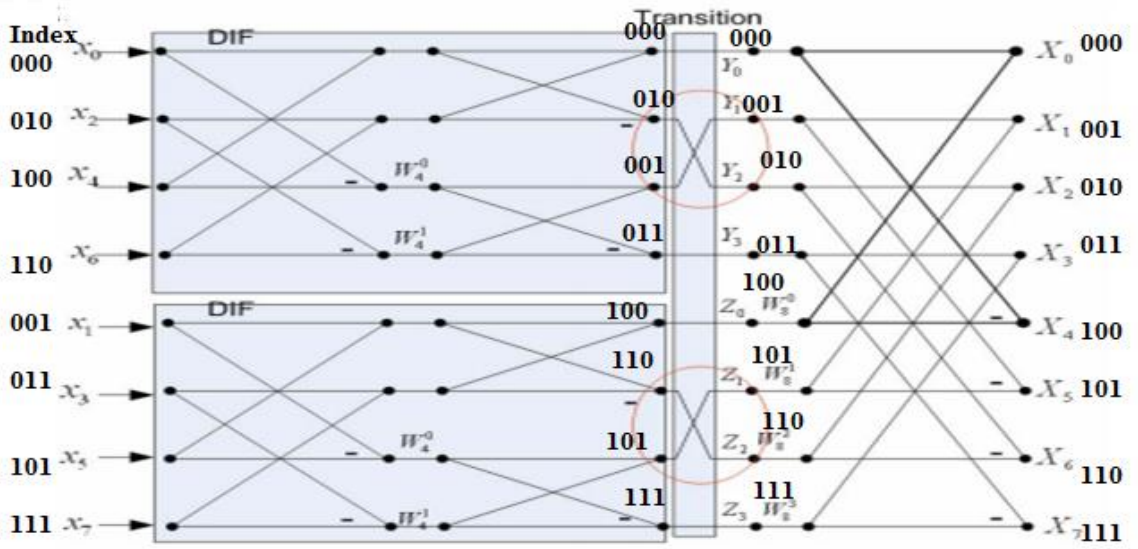

Fig 3.3 8-point Radix-2 Mixed DIF/DIT FFT algorithm

IV Various FFT architectures:

There are various architectures for FFT implementation as shown in the list below but among them the mostly used architecture is pipelined architecture suitable for their advantages of better throughput and latency.

(i) Single-memory architecture
(ii) Dual-memory architecture
(iii) Cached memory architecture
(iv) Array architecture
(v) Pipelined architecture
(vi) Parallel architecture have been used

4.1 Pipelined FFT Architecture:

Pipelined FFT models are otherwise called streaming structures and exploit parallel preparing between the pipeline stages. Pipelined architectures comprise of efficient structure, area productive, nearly basic control unit and have high throughput. One of their significant favourable circumstances is to process continuous flow of information with no extra hardware in contrast with memory-based models, which influences it to cost effective too. In addition, it is additionally conceivable to in-wrinkle the check recurrence in pipelined designs with option of enlist, which abbreviate the basic way and increment the throughput. Pipelined models are more adaptable when changes of variable lengths are to be registered around the SDF models due to the objective applications for this work. With a specific end goal to build the throughput and in addition to diminish the chip-region, SDF pipelined FFT structures are frequently utilized. SDF FFT models have an input circle at each pipeline organize [16]. SDF models comprise of a butterfly preparing unit, an input memory, i.e., FIFO, memory component at each phase to store the twiddle factor coefficients and a complex multiplier in each stage.

4.2 Feedforward FFT Architectures

Feedforward FFT models are likewise alluded as; Multi-way Delay Commutator (MDC) pipelined FFT designs. MDC FFT design is the most straight-forward usage of FFT calculation. These structures don't have any input circles; in this manner, the FFT informant information is prepared by the FFT butterflies and rotators, and after that sustained to the following phase of MDC design. In this way, usage proportion of butterflies in this design is 100%. This design can process a few examples in parallel, which implies MDC structures have higher throughput than SDF models.

V Design Of Proposed 16 Point Mixed Decimation FFT:

The same scheme applied for the above 8-point DFT is used. Final stage is computed based on Radix-2 DIT FFT algorithm. It results in two 8-point DFTs of even-indexed and odd-indexed time samples. Each one of the two 8-point DFTs is broken into eight 4-point DFTs and four 8-point DFTs based on the DIF FFT algorithm. We use the mixed DIT/DIF algorithm for 16-point FFT. Fig. shows the signal flow graph (SFG) of the 16-point FFT algorithm.

The figure shown is the proposed 16 point Mixed DIF/DIT algorithm. Initially in the stage1 the 16 point is divided into two even indexed and odd indexed samples and then these are further divided into 4 four point DIF FFTs then after implementing DIF FFT till stage 2 then the indexes are transitioned to obtain the output as required for DIT FFT. This implementation provides the indexes which are not bit reversed and hence it achieves the advantages of mixed decimation concept and reduces the latency. This signal flow graph is implemented on pipelined Feed forward architecture to achieve higher throughput.

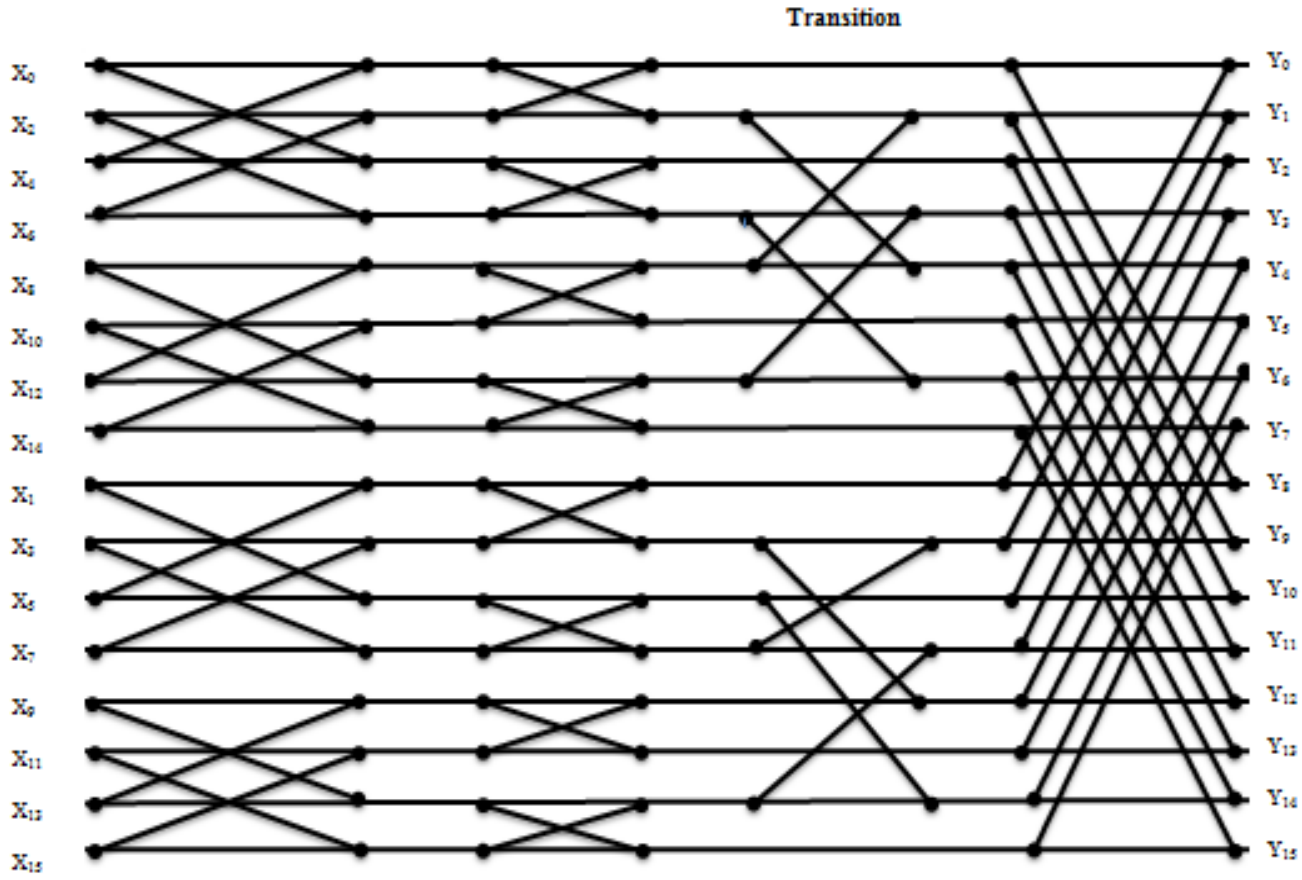

Fig 5.1 Proposed 16 point mixed decimation FFT

VI RADIX 2^2 FEEDFORWARD ARCHITECTURE

This section presents the radix- 2^2 feed forward architectures. A 16-point 4-parallel radix- feed forward FFT architecture is explained in depth in order to clarify the approach and show how to analyze the architectures. Fig. shows a 16-point 4-parallel radix- feed forward FFT architecture. The design is created of radix-2 butterflies (R2), non-trivial rotators, trivial rotators, which are diamond- shaped, and shuffling structures, which consist of buffers and multiplexers. The lengths of the buffers are indicated by a number. The design processes four samples in parallel during a continuous flow.

The order of the data at the different stages is shown at the bottom of the figure by their indices, together with the bits that correspond to these indices. In the horizontal, indexed samples arrive at the same terminal at different time instants, whereas samples in the vertical arrive at the same time at different terminals. Finally, samples flow from left to right. Thus, indexed samples (0, 8, 4, 12) arrive in parallel at the inputs of the circuit at the first clock cycle, whereas indexed samples (12, 13, 14, 15) arrive at consecutive clock cycles at the lower input terminal. Taking the previous issues under consideration, the architecture can be analyzed as follows. The rotation reminiscences of the circuit store the coefficients of the flow graph. In the architecture shown in Fig. 3 the sample with index is the third one that arrives at the lower edge of the second stage. Thirdly, the buffers and multiplexers carry out data shuffling.

These circuits have already been used in previous pipelined FFT architectures [4], and shows how they work. For the first clock cycles the multiplexers are set to “0”, being the length of the buffers. Thus, the first samples from the upper path are stored in the output buffer and the first samples from the lower path are stored in the input buffer. Next, the multiplexer changes to “1”, so set passes to the output buffer and set is stored in the input buffer. At an equivalent time, sets and are provided in parallel at the output. When the multiplexer commutes again to “0”, sets and are provided in parallel. As a result, sets and are interchanged. Finally, the control of the circuit is very simple: As the multiplexers commute every clock cycles and is a power of two, the control signals of the multiplexers are directly obtained from the bits of a counter. The proposed architectures can process a continuous flow of data. The throughput in samples per clock cycle is equal to the number of samples in parallel, whereas the latency is proportional to the size of the FFT divided by the number of parallel samples. Thus, the most suitable architecture for a given application can be selected by considering the throughput and latency that the application demands. Indeed, the number of parallel samples can be increased arbitrarily, which assures that the most demanding requirements are met. Finally, the memory size doesn't increase with the quantity of parallel samples.

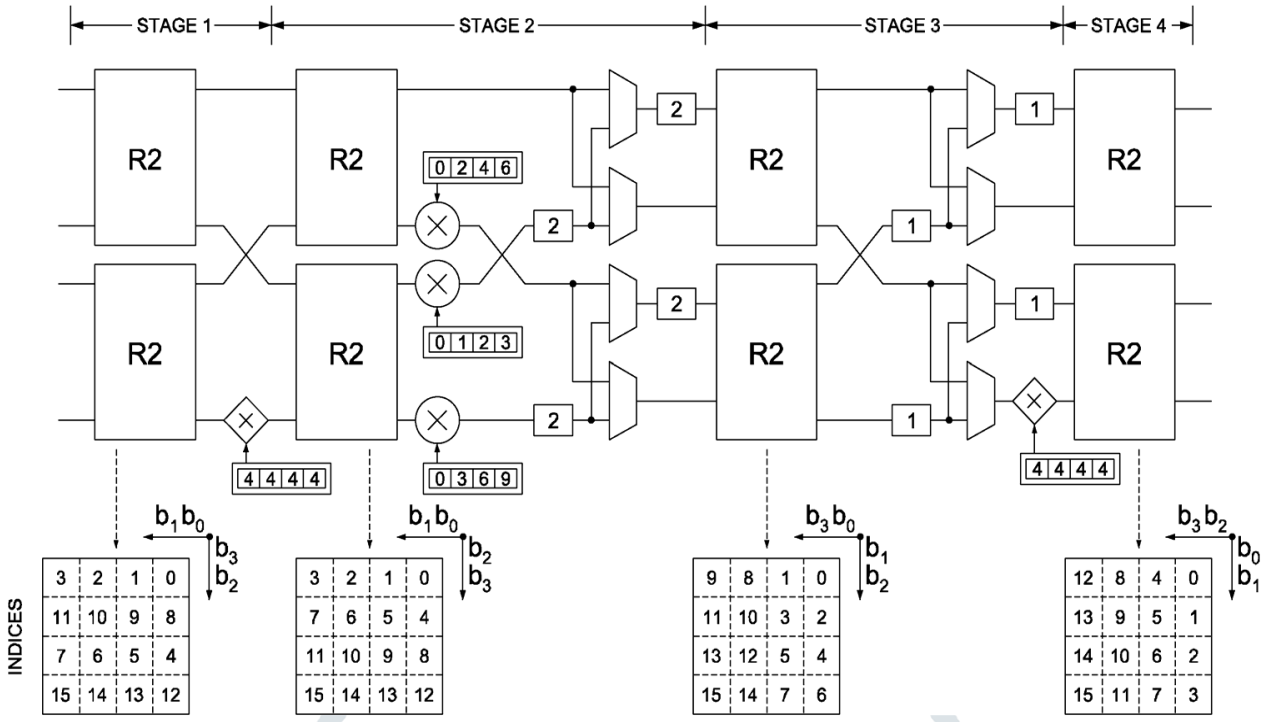

Figure 5.2 Parallel 16 Point Radix 2^2 feed forward architecture

VII. RESULTS AND DISCUSSION:

In electronic/communication systems, throughput refers to rate at which output data is produced. Higher the throughput, more productive is the system. In most of the cases, it is measured as time difference between two consecutive outputs (nth and n+1th). Latency is the time taken by a system to produce output after input is applied. It is a measure of delay response of a design.

Higher the latency value, slower is the system. In synchronous designs, it is measured in terms of number of clock cycles. In combinational designs, latency is basically propagation delay of circuit. In non-pipelined designs, latency improvement is major area of concern. Both latency and throughput are inter-related. It is desired to have maximum throughput and minimum latency. Throughput can be calculated using the formula

Throughput = no. of bits * highest clock period.

Where highest clock period = 1/ latency.

The latency obtained is 1.095 ns

And throughput is 14.16 GS/s.

Table 7.1 Comparison of normal DIT algorithm with mixed decimation FFT.

FFT	No.of Slice LUTs	Latency
8 point DIT FFT	339 out of 46560	5.032 ns
8 point Mixed Decimation FFT	341 out of 46560	4.557 ns
Proposed 16 point Mixed Decimation FFT	1461 out of 46560	7.557 ns

Table 7.1 shows the comparison of 8 point and 16 point mixed decimation FFT algorithm with normal DIT, it can be observed that mixed decimation concept reduces latency. Table 7.2 shows the combination of feed forward along with mixed decimation reduces latency and achieve higher throughput.

Table 7.2 Comparison of previous work with proposed work

TYPE OF PIPELINED FFT	LATENCY	THROUGHPUT
Multipath Delay Commutator (MDC) [5]	0.026 µs	1831 MS/s
Proposed Mixed Decimation Multipath Delay Commutator (M2DC)	1.095 NS	14.16 GS/S

VIII CONCLUSION AND FUTURE SCOPE

In this paper, 16 point mixed decimation DIF/DIT FFT is proposed. This algorithm is implemented on feed forward architecture for 4 parallel radix 2^2 FFT. The algorithm has output generated in normal order which helps in reduction of latency and the architecture used i.e. MDC or feed forward architecture which helps in achieving higher throughput in the range of GS/s.

The proposed architecture achieves lower latency and higher throughput for 4 parallel 16 point M2DC (mixed decimation multipath delay Commutator) when compared to only MDC architectures. This work can be further extended for any number of parallel samples and for any number of points.

IX ACKNOWLEDGMENT

The author would like to thank Dr. Ayesha Naaz for her valuable suggestions about the presentation of this work.

REFERENCES

[1] M. G. KIM, S. K. SHIN AND M. H. SUNWOO, "NEW PARALLEL MDC FFT PROCESSOR WITH EFFICIENT SCHEDULING SCHEME," 2014 IEEE ASIA PACIFIC CONFERENCE ON CIRCUITS AND SYSTEMS (APCCAS), ISHIGAKI, 2014, PP. 667-670.

[2] M. Ayinala and K. K. Parhi, "Parallel - pipelined radix-2^2 FFT architecture for real valued signals," *2010 Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, 2010, pp. 1274-1278.

[3] Suganya V and Paramasivam C, "Parallel pipelined FFT architecture for real valued signals," *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Chennai, 2016, pp. 2146-2149.

[4] F. Qureshi and J. Takala, "New Identical Radix-2^k Fast Fourier Transform Algorithms," *2016 IEEE International Workshop on Signal Processing Systems (SiPS)*, Dallas, TX, 2016, pp. 195-200.

[5] M. Garrido, J. Grajal, M. A. Sanchez and O. Gustafsson, "Pipelined Radix-2^k Feedforward FFT Architectures," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 1, pp. 23-32, Jan. 2013.

[6] C. Caraiscos and Bede Liu, "Two dimensional DFT using mixed time and frequency decimations," *ICASSP '82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Paris, France, 1982, pp. 24-27.

[7] K. Nakayama, "An improved fast Fourier transform algorithm using mixed frequency and time decimations," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 2, pp. 290-292, Feb. 1988.

[8] S. Lee and S. Park, "Modified SDF Architecture for Mixed DIF/DIT FFT," 2007 IEEE International Symposium on Circuits and Systems, New Orleans, LA, 2007, pp. 25902593.

[9] Reshma K J1, Prof. Ebin M Manuel2, "An Area Efficient Mixed Decimation MDF Architecture for Radix 2^2 Parallel FFT",International Research Journal of Engineering and Technology (IRJET).

[10] J. Wang, C. Xiong, K. Zhang and J. Wei, "A Mixed-Decimation MDF Architecture for Radix-2^k Parallel FFT," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 1, pp. 67-78, Jan. 2016.