IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 7, Issue 3, Ver. I (May. - Jun. 2017), PP 01-07
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

An Area Efficient FFT Structures Design by Sharing Arithmetic Units

Pradnya Zode[1], Dr.A.Y.Deshmukh[2]
[1](*Research Scholar, Department of Electronics Engineering. G.H. Raisoni College of engineering, Nagpur, Maharashtra, India)*
[2](*Professor, Department of Electronics Engineering. G.H. Raisoni College of engineering, Nagpur, Maharashtra, India)*

Abstract: *Low power consumption has become apparent need in the area of VLSI digital signal processing. This gives rise to the need of minimization of silicon area which can be done by folding algorithm. As silicon area decreases power consumption of a circuit decreases. Folding transformation is a technique which reduces silicon chip area by combining various arithmetic operations into one operation by time scheduling technique. It is applied on iterative with appropriate folding set. This paper presents an approach to design fast Fourier transform (FFT) architectures using folding transformation. Proposed work is focused on design of efficient VLSI architecture for FFT using radix-3 algorithm which aims at reducing mainly area which results in reduction of power consumption and hardware complexity. A proper procedure for designing FFT architectures using folding transformation and register minimization techniques is projected. Results show that numbers of adders are reduced by 37.5 % and multipliers by 33.33% without changing characteristics of FFT algorithms.*

Keywords: *Data flow graph, Fast Fourier transform, Folding transformation technique, Parallel-pipelined, Radix-3 algorithm.*

I. Introduction

Digital Signal processing (DSP) is an area of science and engineering which has developed quickly over last few years [1,2]. Very Large Scale Integration (VLSI) is very popular and interesting integrated circuit (IC) technology in the area of electronics engineering for more than last 3-4 decades [3]. The importance of DSP systems have expanded significantly because of their wide class of applications in electronic design which includes video compression, digital set-top box television & , cable modems, DVDs players, portable video systems/computers, multimedia & wireless communications, digital radio, digital still & network cameras, speech processing, transmission systems, radar imaging, acoustic beam-formers, GPS, biomedical signal processing, video conferencing systems, digital telephony, sonar, just to name a few [1,2,4,5,6,7,8].

The power consumption of VLSI devices has gaining attention in recent years, largely because of the use of battery operated portable products goes on increasing [3]. Power consumption has become the uncertain barrier for the applications of portable communication and computing DSP systems. In such systems DSP devices are of more interest. These DSP devices are used widely for complex functions such as data compression, and speech processing [1]. The increasing requirements for portable systems to incorporate complex functions has led to increased demand for low-power DSP devices.The increasing importance of portable electronics and consumer-oriented devices has become a fundamental driving factor in the design of new computational elements in CMOS VLSI systems on a chip [7].

In DSP architectures, the folding algorithm is used to systematically determine the control circuits where multiple algorithm operations are time multiplexed to a single functional unit. This folding technique can be used for synthesis of DSP architectures that can be operated using single or multiple clocks. In synthesizing DSP architectures, it is important to minimize the silicon area of the integrated circuits, which is achieved by reducing the number of functional units (such as multipliers and adders), registers, multiplexers, and interconnection wires. For example, executing N addition operations using a single adder [2]. This means executing multiple operations on a single functional unit and the number of functional units in the implementation is reduced, thus resulting with low silicon area in integrated circuits [2].In this paper folding algorithm is applied on fast Fourier transform (FFT) algorithms. The FFT algorithm is one of the fundamental processing elements in any DSP system [1,2]. The FFT are widely used in various areas such as telecommunications, speech & image processing and filters, medical electronics and seismic processing, frequency analysis of the signals generated by vibration sensors, spectrum analyzers, etc [1,2,4,5,6,7,8]. FFT is used as one of the key component in OFDM-based wide-band communication systems [9] and wireless mobile terminals which demands high-speed and low-power FFT designs [10].The requirement for low-power FFT architectures for telecommunication systems in portable form is becoming more and more important.

II. Folding Transformation

The folding transformation is introduced by K.K. Parhi which is described in detail in [7,11,12]. The folding transformation is a technique used to minimise the silicon area of an Integrated circuit by time multiplexing multiple operations to a signal functional units like multipliers, adders, registers, multiplexers and interconnection wires [7]. The folding transformation technique can be used for synthesis of DSP architectures that can be operated using single or multiple clocks [7]. A brief review of folding transformation is given here before applying it to any DSP algorithm.

The folding transformation technique is explained by the Fig. 1 (a) and (b) [7].

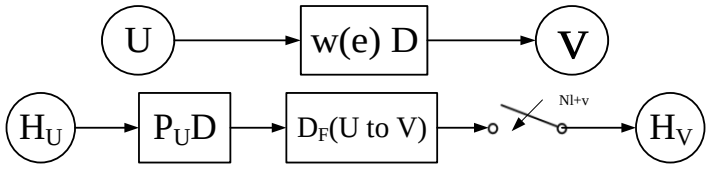

Figure 1: (a) and (b) The basic of folding transformation technique [7].

Fig. 1 (a) shows an edge e from the source node U to the destination node V with weight D i.e. the edge e have w(e) delay value on it. Figure 2.1 (b) shows the corresponding folded edge. The functional source unit of node U is H_U which is pipelined by P_U stages must pass through delays is given by equation 1 [7]:

$$D_F\left(U \xrightarrow{e} U\right) = Nw(e) - P_U + v - u \quad (1)$$

The edge switched at destination node V whose functional unit is H_V at instance Nl+v, where N is a folding factor. Folding factor is defined as the number of algorithmic operations folded to or executed by a single functional unit in hardware and is denoted by N while u and v are the folding orders of nodes U and V respectively [7]. The folding order of a node is the time partition to which the node is scheduled to execute in hardware. A folding set, S, is defined as an ordered set of operations executed by the same functional unit [7]. Each folding set contains N entries, some of which may be null operations. The operation in the i^{th} position within the folding set which goes from 0 to N-1 is executed by the functional unit during time position i. For a complete folded system to be possible, $D_F(U \xrightarrow{e} V) \geq 0$ must hold for all of the edges in the DFG. Once valid folding sets have been assigned, retiming can be used to satisfy this property or determine that the folding sets are not feasible [7].

III. Retiming Transformation

Retiming is a transformation technique used to change the location delay elements in a circuit without affecting the input-output characteristics of the circuit [7,13,14]. Retiming has many applications in synchronous digital circuit design. These applications include reducing the clock period of the circuit, reducing the number of the registers in the circuit, reducing the power consumption of the circuit and logic synthesis [7]. Cutset retiming is used in this thesis which is the special case of retiming. A cutset is a set of edges that can remove from the graph to create two disconnected subgraphs [7]. Cutset retiming only affects the weights of the edges in the cutset. If the two disconnected subgraph are denoted by S1 and S2, then using cutset retiming d delays can add to the edges going from S1 to S2 and removing d delays from each edge going from S2 to S1.Sometimes pipelining is also used which is a special case of cutset retiming. In this case there are no edges from subgraph S2 to S1 i.e. Pipelining applies to the graph without loops.

IV. Folding Transformation Technique

The complete procedure for folding transformation technique is described stepwise below [7].

1. Selection of appropriate folding order and accordingly folding set.
2. Write the folding equation for each and every edge shown in the DFG.
3. If needed perform retiming for folding.
4. Rewrite the folding Equation.
5. Use the folding equations to construct Lifetime table.
6. Draw the Lifetime chart and find out the minimum number of registers.
7. Carry out the data allocation using forward and backward register allocation technique.
8. Draw the folded architecture.

V. Fast Fourier Transform

The discrete Fourier transform (DFT) plays a central role in the implementation of various DSP algorithms and systems [1,16]. It transforms a signal from the time domain into the frequency domain, providing information about the spectrum of the signal [1,2]. The computation of an N-point DFT using direct methods requires N^2 arithmetic number of operations. The Fast Fourier Transform (FFT) is simply a mathematical algorithm to speed up the DFT calculation by reducing the necessary number of multiplications and additions [1,2,4,15]. The FFT algorithms are based on the principle of decomposing the computation of DFT into sequences of smaller DFTs. DFT is identical to samples of the Fourier transform at equally spaced frequencies. Consequently, computation of the N-point DFT corresponds to the computation of N samples of the Fourier transform at N equally spaced frequencies $\omega k = 2\pi k/N$. The FFT utilizes some clever algorithms to do the same thing as the DFT, but in much less time. While a DFT computation of length *N* takes N^2 arithmetic operations, a FFT for the same length takes only *NlogN* operation [1,2,4,5]. FFT was first discussed by J. W.Cooley and J. W. Tukey in the 1960s and was in fact a rediscovery of an idea of the German mathematician Karl Friedrich Gauss (1777 – 1855). This is the most common FFT algorithm called as the Cooley-Tukey algorithm, named after J.W.Cooley and John Tukey [16]. These scientists are given credit for bringing the FFT to the world from their paper: "An algorithm for the machine calculation of complex Fourier Series," *Mathematics Computation*, Vol. 19, 1965, pp 297-301 [16]. This publication of the Cooley-Tukey Fast Fourier transform (FFT) algorithm in 1965 has opened a new area in DSP by reducing the order of complexity of computational tasks in DFT and convolution from N^2 to N $\log_2$ N, where N is the problem OR transform size. The Cooley and Tukey FFT algorithm is a turning point to computation of DFT. FFTs are used to speed up the computations of DFTs [2].
FFT plays an important role in modern digital signal communications such as digital video broadcasting and OFDM systems [1,2]. Because the FFT is everywhere, different methods for calculating FFT of a signal have been studied extensively, and continue to be an active research topic.The FFT algorithm decomposes a DFT of length N into two half length DFTs of size *N*/2. The DFT is split into even index terms and odd index terms. The decomposition process is repeated *M = logrN* times, where r is the radix. A number of FFT architectures have been proposed in the literature. Based on the Cooley-Tukey (CT) decomposition [1,2], many FFT algorithms, such as radix-2 [1,2], radix-3 [4], radix-2^2 [2], radix-2^3[5,6], split-radix as well as radix-4 algorithms, have been proposed using the complex mathematical relationship to reduce the hardware complexity. These algorithms are based on decomposing an N-point DFT recursively into smaller DFTs, leading to a reduction of the computational complexity. Most FFT algorithms and architectures have focused on power-of-two size DFTs. High-speed and low-power FFT design is the need of time. Hence, it is important to design high-performance and low-power FFT for various applications.

The FFT operates by decomposing an *N* point time domain signal into *N* time domain signals each composed of a single point. The second step is to calculate the *N* frequency spectra corresponding to these *N* time domain signals. Lastly, the *N* spectra are synthesized into a single frequency spectrum. The basic computational step of the FFT algorithm is a butterfly [1,2]. The most commonly used decomposition methods of FFT algorithm are decimation-in-time (DIT) and decimation-in-frequency (DIF) [1,2,15] . Both of these rely on the recursive decomposition of an *N* point transform into 2 (*N*/2) point transforms. The method is particularly simple if *N* is divisible by 2 and if *N* is a regular power of 2, the decomposition can be applied repeatedly until the trivial '1 point' transform is reached. DIT is based on decomposing the input sequence into smaller sub-sequences while in DIF the output sequence is decomposes into smaller sub-sequences [1,2].

VI. Radix-3 FFT Algorithm: Introduction

The common applications of DSP uses radix-2 FFT algorithms where N i.e. FFT sizes are power of 2. In this paper, FFT algorithms where N is not power of 2 are considered which works for other input sizes. It is always beneficial to use different radix FFT algorithms depends on the application with different input sample sizes N. When N is power of 3 FFT algorithm based on radix-3 can be implemented. For example considered N = 9 = 3^2 then input sequence samples can be divided into 3 sets of sequences where each sequence is of length N/3 [2].
The DFT of the finite duration sequence {x(n)} where $0 \le n \le N-1$ can be expressed as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

Where k=0, 1, 2, ----, N-1.

$$\text{And } W = e^{-j(2\pi/N)}.$$

The radix-3 DIT FFT algorithm can be derived by considering 3 input samples as

$$X[k] = \sum_{n=0}^{\frac{N}{3}-1} x[3n]W_N^{k3n} + \sum_{n=0}^{\frac{N}{3}-1} x[3n+1]W_N^{k(3n+1)} + \sum_{n=0}^{\frac{N}{3}-1} x[3n+2]W_N^{k(3n+2)}$$

$$X[k] = \sum_{n=0}^{\frac{N}{3}-1} x[3n]W_N^{3nk} + W_N^k \sum_{n=0}^{\frac{N}{3}-1} x[3n+1]W_N^{3nk} + W_N^{2k} \sum_{n=0}^{\frac{N}{3}-1} x[3n+2]W_N^{3nk}$$

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

$$X[k] = P(k) + W_N^k Q(k) + W_N^{2k} R(k)$$

The butterfly structure for radix-3 FFT algorithm is shown in Fig. 2.

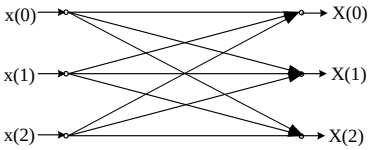

Figure 2: Radix-3 FFT algorithm [2].

VII. Folding Of 9-Point DIT FFT Using Radix-3 Algorithm.

The signal flow graph for N = 9 using decimation-in-time using radix-3 FFT algorithm is shown in Fig. 3. The DFG for the 9-point DIT FFT using radix-3 algorithm is shown in Fig. 4. In Fig. 4 A0, A1, A2, B0, B1, B2 are nodes which represents the 3-point butterfly structures shown in Fig 2.

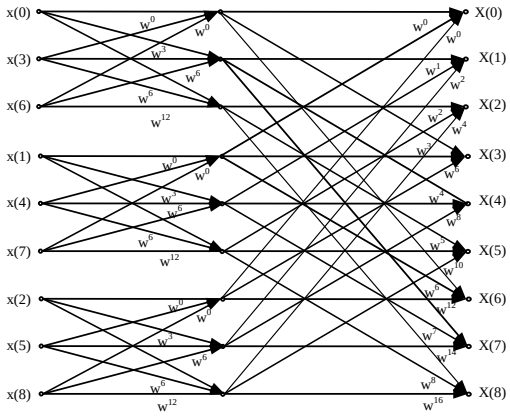

Figure 3: 9-point DIT FFT using radix-3 algorithm.

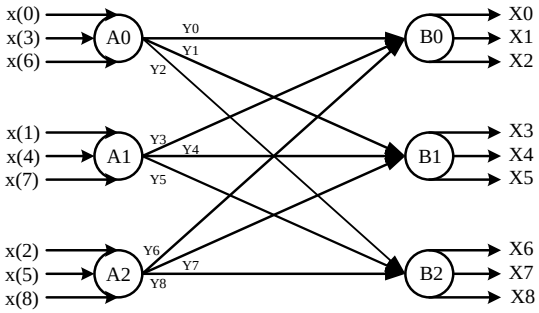

Figure 4: DFG of 9-point DIT FFT using radix-3 algorithm

.

Step 1: Select Folding order N=8 and folding sets are as follows

A={A0,ϕ,ϕ,A1,ϕ,ϕ,A2,ϕ} B={B0,ϕ,ϕ,B1,ϕ,ϕ,B2,ϕ}

Where symbol ϕ shows the null operation i.e. for the folding order of ϕ symbol the arithmetic unit will not perform any operation.

Step 2: The Folding Equations for DFG in Fig. 4 using folding equation

$D_F (U \rightarrow V) = N\ w\ (e) - P_U + v - u$

In the above folding edge equation P_U is the of the source node U. Here for simplification, assumed that the P_A, $P_B = 0$ i.e. all these nodes don't have pipelining stage

- $D_F (A_0 \rightarrow B_0) = 8\ (0) - 0 + 0 - 0 = 0$
- $D_F (A_0 \rightarrow B_1) = 8\ (0) - 0 + 3 - 0 = 3$
- $D_F (A_0 \rightarrow B_2) = 8\ (0) - 0 + 6 - 0 = 6$
- $D_F (A_1 \rightarrow B_0) = 8\ (0) - 0 + 0 - 3 = -3$
- $D_F (A_1 \rightarrow B_1) = 8\ (0) - 0 + 3 - 3 = 0$
- $D_F (A_1 \rightarrow B_2) = 8\ (0) - 0 + 6 - 3 = 3$
- $D_F (A_2 \rightarrow B_0) = 8\ (0) - 0 + 0 - 6 = -6$
- $D_F (A_2 \rightarrow B_1) = 8\ (0) - 0 + 3 - 6 = -3$
- $D_F (A_2 \rightarrow B_2) = 8\ (0) - 0 + 6 - 6 = 0$

Step 3: If needed perform retiming for folding.
The cutset retiming is used here which is a special case of retiming. In the cutest retiming apply cutest to the edges in the DFG with all are in forward direction. Then one delay or latch is to be placed on each and every edge under cutest. The DFG with cutset and after cutest retiming are shown below in Fig. 5 and 6 respectively.

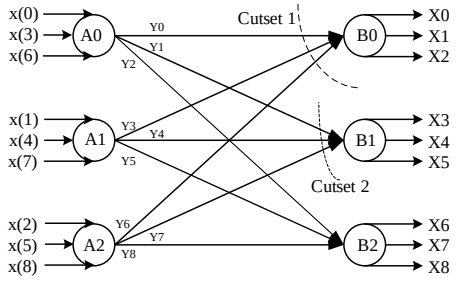

Figure 5: DFG of 9-point DIT FFT using radix-3 algorithm with cutset.

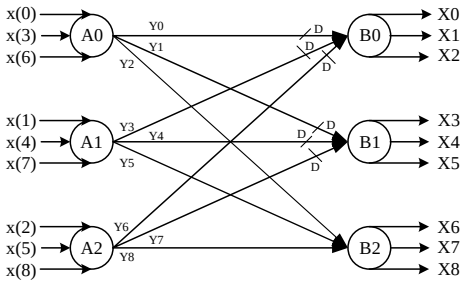

Figure 6: Retimed DFG of the 9-point DIT FFT using radix-3 algorithm.

Register Minimization Technique
Performing folding transformation technique on any DSP algorithm reduces the number of function unit for example adder and multipliers in the folded architecture but at the same time increases the number of registers in the folded architecture. To avoid the folded architecture with excessive number of registers techniques are proposed by [7,11,12] to design the folded architecture with minimum number of registers and to allocate the data to these registers which is achieved by performing the following steps.

Step 4: Rewrite the folding Equation.
For proper folding none of the edge of the DFG should have negative delay value.

- $D_F(A_0 \rightarrow B_0) = 8(1) - 0 + 0 - 0 = 8$
- $D_F(A_0 \rightarrow B_1) = 8(1) - 0 + 3 - 0 = 11$
- $D_F(A_0 \rightarrow B_2) = 8(0) - 0 + 6 - 0 = 6$
- $D_F(A_1 \rightarrow B_0) = 8(1) - 0 + 0 - 3 = 5$
- $D_F(A_1 \rightarrow B_1) = 8(1) - 0 + 3 - 3 = 8$
- $D_F(A_1 \rightarrow B_2) = 8(0) - 0 + 6 - 3 = 3$
- $D_F(A_2 \rightarrow B_0) = 8(1) - 0 + 0 - 6 = 2$
- $D_F(A_2 \rightarrow B_1) = 8(1) - 0 + 3 - 6 = 5$
- $D_F(A_2 \rightarrow B_2) = 8(0) - 0 + 6 - 6 = 0$

Step 5: Lifetime table is shown in Table 1 where T_{input} is the input clock cycle for each node and T_{output} is the clock cycle from where output can be read.
Step 6: Draw the Lifetime chart and find out the minimum number of registers.
The linear lifetime chart is shown in Fig. 7 is based on [7], where each horizontal line represents a clock cycle and each vertical line represents the lifetime of a variable. From the lifetime chart, the minimum numbers of registers that can be used to implement the architecture of FFT is the maximum number of live variables at any clock cycle [7]. The minimum number of registers to implement the FFT is maximum number of live variables =max {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} = 9.

Table 1 Lifetime table for 9-point DIT FFT

Inputs	Lifetime	Outputs	Lifetime
x0	0->6	y0	0->0
x1	1->7	y1	0->3
x2	2->8	y2	0->6
x3	3->6	y3	3->0
x4	4->7	y4	3->3
x5	5->8	y5	3->6
x6	6->6	y6	6->0
x7	7->7	y7	6->3
x8	8->8	y8	6->6

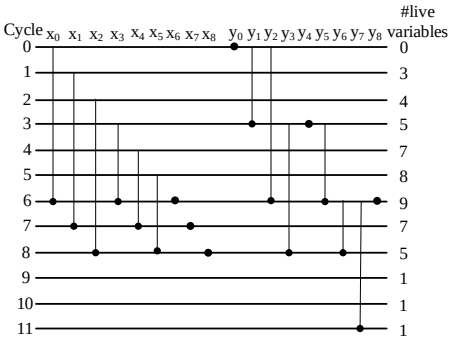

Figure 7: Lifetime chart for 9-point DIT FFT using radix-3 algorithm.

Step 7: Carry out data allocation using forward and backward register allocation technique.
The registers are denoted as R1, R2, R3, R4, R5, R6, R7, R8, R9 in the allocation table shown in Fig. 8. The variables are allocated forward register first and then to an appropriate backward register, thus the name "forward-backward" [7].

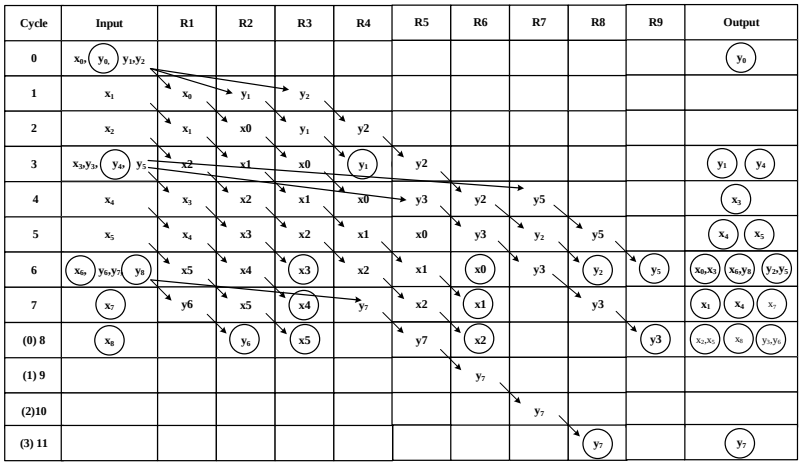

Figure 8: Register allocation table for 9-point DIT FFT using radix-3 algorithm.

Step 8: Draw the folded architecture.
Folded Architecture of 9-point DIT FFT using radix-3 algorithm is shown in Fig 9. In this BF I and BF II are 3-point butterfly units. BF I will perform operations of A0 to A2 & BF II will perform operations of B0 to B2 3-point butterfly units respectively from DFG of 9-point DIT FFT using radix-3 algorithm shown in Fig 4.

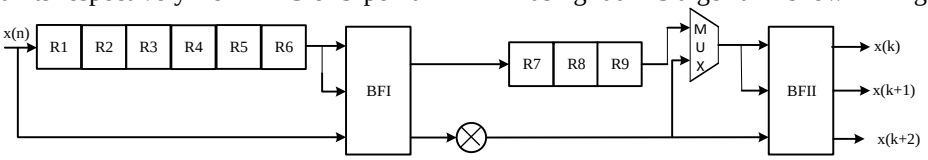

Figure 9: Folded Architecture of 9-point DIT FFT using radix-3 algorithm.

VIII. Folding Of 9-Point DIF FFT Using Radix-3 Algorithm.

Signal flow graph for N = 9 using decimation-in-frequency using radix-3 FFT algorithm is shown in Fig. 10.

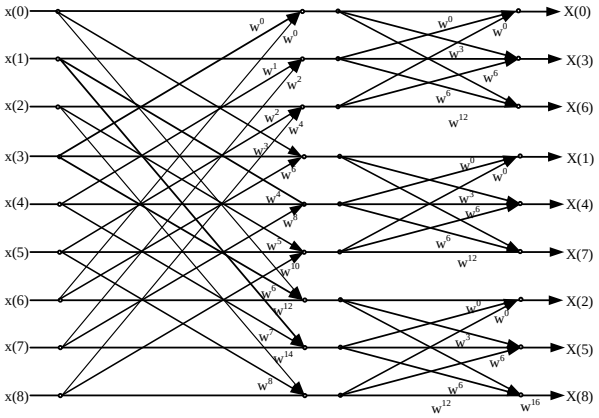

Figure 10: 9-point DIF FFT using radix-3 algorithm.

Step 8: Draw the folded architecture.
Folded Architecture of 9-point DIF FFT using radix-3 algorithm is shown in Fig 11.

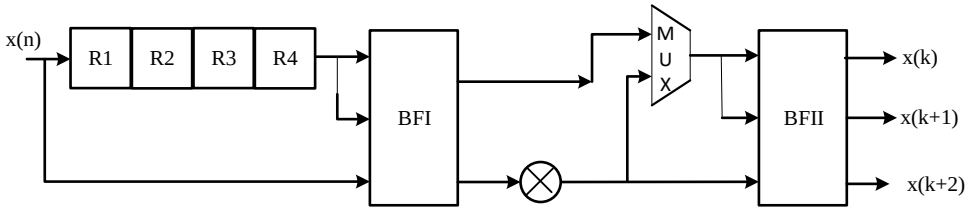

Figure 11: Folded Architecture of 9-point DIT FFT using radix-3 algorithm.

IX. Implementation Of Proposed Work And Results

Table 2 shows comparison between conventional and folded 9-point DIT & DIF FFT using radix-3 algorithm in terms of number of butterfly units.

Table 2 Comparison between conventional and folded 9-point DIT and DIF FFT

	No. of Butterfly units
Conventional	6
Folded	2

The original and folded FFT algorithm are modelled in Verilog HDL and synthesized using ISE-Xilinx tool. Both the designs were synthesized using Vertex-5 FPGA and the results are shown in the table 3.

Table 3 Synthesis result of conventional and folded 9-point DIT and DIF FFT

	DIT FFT		DIF FFT	
Parameters	Conventional	Folded	Conventional	Folded
Power	0.208 mW	0.66mW	0.220 mW	0.69 mW
Delay	200ns	200ns	200ns	200ns

X. Conclusion

VLSI DSP is very fast growing technology in field of today's wireless communication. Most common algorithms used in DSP are Fast Fourier Transforms. The major purpose of this manuscript was to design efficient structures of FFT algorithms in terms of area, power and speed. For this the different transformation techniques were performed on the data flow graphs of FFT algorithms like pipelining, parallel processing, folding and retiming. Folding transformation technique is used to reduce the silicon area for any algorithm. And retiming is used to reduce the number of delays in the circuit so that the speed of the circuit will increase. For any DSP algorithms multiplier and adders are the significant blocks. Therefore folded structures were designed with minimum number of butterfly units. When input samples are not in the range of power of 2 it is very much beneficial to use the different radix algorithms. Therefore folded structure for 9-point DIT and DIF FFT using radix-3 algorithms is designed. The folded structure was efficient in power consumption, area and performance. The power consumption can be reduced from 37% up to 50% in the proposed folded architectures. These folded structures are very suitable for applications in implantable or portable devices due to their low area and power consumption.

References

[1] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principle, Algorithms, and Applications*, Third edition, Pearson Education 2004.

[2] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, PHI Learning Private Limited, 2011.

[3] N.Weste, D.Harris and A.Banerjee, *CMOS VLSI Design: A Circuits and Systems Perspective*, Third edition, Pearson Education 2009.

[4] L. R. Rabiner and B. Gold, *Theory and application of Digital Signal Processing*, PHI Learning Private Limited, 2012.

[5] R.Woods, J.Mcallister, G.Lightbody and Ying Yi, *FPGA-based implementation of Signal Processing Systems*, John Wiley & Sons, 2008.

[6] S.S. Bhattacharyya, Ed F. Deprettere, R. Leupers and J.Takala, *Handbook of Signal Processing Systems*, Springer Science+Business Media, LLC 2010.

[7] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, John Wiley & Sons, 1999.

[8] T. Arslan, A.T. Erdogan and D.H. Horrocks, "Low power design for DSP: methodologies and techniques" *ELSEVIER, Microelectronics Journal* 27(1996) 731-744.

[9] S.-N Tang, J. Tsai, T.-Y. Chang, "A 2.4-GS/s FFT Processor for OFDM-Based WPAN Applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol.57, no.6, pp.451-455, June 2010.

[10] T. Cho, H. Lee, J. Park, C. Park, "A high-speed low-complexity modified radix-2^5 FFT processor for gigabit WPAN applications," *ISCAS*, pp. 1259-1262, 2011.

[11] K. K. Parhi, C. Y. Wang and A.P. Brown, "Synthesis of control circuits in folded pipelined DSP architectures*", IEEE Journal Solid-State Circuits*, Vol. 27, No. I, January 1992.

[12] Tracy C. Denk and Keshab K. Parhi "Synthesis of Folded Pipelined Architectures for Multirate DSP Algorithms" *IEEE Transactions On Very Large Scale Integration (VLSI) Systems*, Vol. 6, No. 4, December 1998.

[13] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, pp. 5–35, 1991.

[14] C. Leiserson, F. Rose, and J. B. Saxe, "Optimizing synchronous circuitry by retiming," in Proc. 3rd Caltech Conf. VLSI, 1983, pp. 87–116.

[15] Ramirez, R. W., *The FFT Fundamentals and Concepts*, Englewood Cliffs, NJ: Prentice-Hall, 1985.

[16] Cooley, J. W., and J. W. Tukey. "An Algorithm for Machine Calculation of Complex Fourier Series." *Math Computation* (April 1965), Vol. 19, pp. 297-301.