

Design and Implementation of Radix-4 FFT Using Retiming Approach

Jalaja S¹, Sunita kulkarni²

VTU Research Scholar, Assistant Professor Department of ECE, BIT, Bangalore

M. Tech, Department of ECE, BIT, Bangalore

jalajabit@gmail.com, Srkulkarni93@gmail.com

Abstract: FFTs efficiently calculate the Discrete Fourier Transform (DFT). Calculating DFT using the FFT reduces the number of arithmetic operations compared to direct computation method. FFTs are found very useful in the fields like image processing signal analysis filtering etc. In this paper we have proposed 16 bit complex Radix-4 FFT using New Distributed Arithmetic (NEDA) algorithm. NEDA replaces complex multiplication operation by using adders and shifters. NEDA is multiplication less and ROM less unit so minimizes the hardware requirements. Adders are main blocks in NEDA so any changes in adder blocks affect the NEDA operation. We are making use of modified carry select adder using BEC instead of carry select adder in the NEDA blocks. This proposed design achieves less area and less power than CSLA and improves the system performance. ModelSim is used to simulate the design and Xilinx ISE 14.7 project navigator is used for synthesis. The proposed design is also implemented in ASIC design flow. The modified radix-4 retiming architecture is more efficient in terms of power compared to the conventional design.

Keywords: Fast Fourier Transform (FFT), New Distributed Arithmetic (NEDA), Distributed Arithmetic (DA), Carry Select Adder (CSLA), Retimed Radix 4, Binary to Excess one converter (BEC).

1. INTRODUCTION

FFT algorithms uses the properties like phase factor and periodicity for efficient calculation of DFT. There are 2 methods for calculating FFT.

- Divide and Conquer
- Linear filtering

Since the discovery of FFT many algorithms like Radix 2, Radix 4, and Radix 8 have been developed. These FFT algorithms can be developed either in decimation in time or decimation in frequency. An FFT algorithm greatly reduces the complexity in computation compared to direct computations. The computation time dependence on length of DFT as length increases time decreases and FFT requires very small storage. Because of FFT real time implementation of DFT is made possible.

The arithmetic operations significantly consume more power which results in overall system power consumption. We know that FFT has least number of arithmetic operations hence suitable for low power FFT processors. The intermediate computations are done using NEDA which is a multiplier less architecture. NEDA eliminates the use of ROM required by Distributed Arithmetic (DA) structures by using additional adders, shifters and multiplexer.

Adders are used as part of arithmetic logic units as well as different part of processor, where they are used to figure out the addresses, table indices and other applications. Some different uses of adders are in multiplier and accumulator units (MAC). Adders are also used in multiplier, in high speed integrated circuits and digital signal processing to execute the various algorithms like FFT, IIR and FIR.

For ripple carry adder, the output carry of previous adder becomes input to next full adder. Carry travels long path hence overall delay increases. This delay is called carry propagation delay. Because of this, the speed of adder reduces. Carry select adders are used in most of the data processing processor to enhance the speed of arithmetic operation. The structure of this CSLA can be further modified to reduce the area, delay and power. For the same modified structure we apply the retiming approach to achieve low power.

1.1 RELATED WORK

Radix-4 butterfly structure consists of four inputs and four outputs. This algorithm is based on four point butterfly units. This four point butterfly structure can be implemented without the help of multipliers by using four two-point butterfly units. The length of the FFT is 4^N where N is the number of stages. The stage of radix-4 is half of the stage of radix-2. In radix-4 algorithm N-point DFT is successively decomposed into four $\frac{N}{4}$ point DFTs, sixteen $\frac{N}{16}$ point DFTs and so on. The basic equations for radix-4 algorithm can be obtained as follows:

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

$$X[k] = \sum_{n=0}^{\frac{N}{4}-1} X(n)W_N^{nk} + \sum_{n=\frac{N}{4}}^{\frac{2N}{4}-1} X(n)W_N^{nk} + \sum_{n=\frac{3N}{4}}^{\frac{4N}{4}-1} X(n)W_N^{nk} + \sum_{n=\frac{5N}{4}}^{N-1} X(n)W_N^{nk}$$

$$X[K] = \sum_{n=0}^{\frac{N}{4}-1} X(n)W_N^{nk} + W_N^{nk/4} \sum_{n=0}^{\frac{N}{4}-1} X(n + \frac{N}{4})W_N^{nk} + W_N^{nk/2} \sum_{n=0}^{\frac{N}{4}-1} X(n + \frac{N}{2})W_N^{nk} + W_N^{3nk/4} \sum_{n=0}^{\frac{N}{4}-1} X(n + \frac{3N}{4})W_N^{nk}$$

The three twiddle factor coefficients can be expressed as:

$$W_N^{(\frac{N}{4})K} = [\cos(\frac{\pi}{2}) - j \sin(\frac{\pi}{2})]^k = (-j)^k$$

$$W_N^{(\frac{N}{2})K} = [\cos(\pi) - j \sin(\pi)]^k = (-1)^k$$

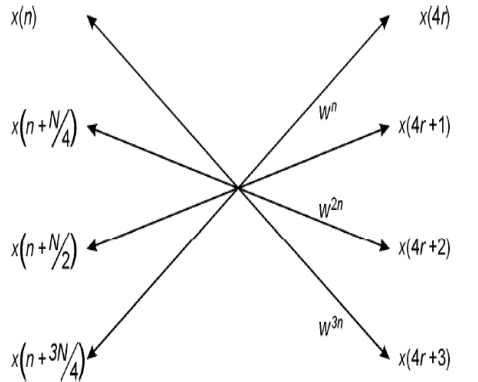
$$W_N^{(\frac{3N}{4})K} = [\cos(\frac{3\pi}{2}) - j \sin(\frac{3\pi}{2})]^k = j^k$$


Fig 1: Butterfly structure

2. MATHEMATICAL DERIVATION OF NEDA

To implement the sum of product term we require ROM, adders and shifters in Distributed Arithmetic technique. In NEDA we use only adders and shifters thus eliminating ROM. Inner product computation of any two sequences can be represented using summation as

$$Z = \sum_{i=1}^K R_i X_i \dots\dots\dots (1)$$

Where R_i are constant coefficients and X_i are varying inputs. Matrix representation of equation (1) can be given as:

$$Z = [R_1 \ R_2 \ \dots\dots\dots R_K] \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_K \end{bmatrix} \dots\dots\dots (2)$$

Considering both R_i and X_i in 2's complement they can be expressed in the form

$$R_i = -R_i^M 2^M + \sum_{K=N}^{M-1} R_i^K 2^K \dots\dots\dots (3)$$

Where $R_i = 0$ or 1 , $K = N, N+1 \dots M$ and R_i^M is the sign bit and R_i^N is the least significant bit. Substituting (3) in (2) results the following matrix product which is modeled according to the required design of FFT.

$$Z = [-2^0 \ 2^{-1} \ \dots \ 2^{-12}] \begin{bmatrix} R_1^0 & \dots & R_K^0 \\ \vdots & \ddots & \vdots \\ R_1^{12} & \dots & R_K^{12} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_K \end{bmatrix} \dots \dots (4)$$

The matrix containing R_i^K is a sparse matrix, which means the values are either 1 or 0. The number of rows in R matrix defines the precision of fixed coefficients used. Equation (4) is rearranged as shown below.

$$Z = [-2^0 \ 2^{-1} \ \dots \ 2^{-12}] \begin{bmatrix} W_0 \\ W_1 \\ \vdots \\ W_{12} \end{bmatrix} \dots \dots (5)$$

Where

$$\begin{bmatrix} W_0 \\ W_1 \\ \vdots \\ W_{12} \end{bmatrix} = \begin{bmatrix} R_1^0 & \dots & R_K^0 \\ \vdots & \ddots & \vdots \\ R_1^{12} & \dots & R_K^{12} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_K \end{bmatrix} \dots \dots (6)$$

In each row, the matrix consists of sums of the inputs depending on the coefficient values. An example that shows the NEDA operations is discussed below.

$$Y = [\cos \frac{\pi}{8} \ \cos \frac{\pi}{4}] \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \dots \dots (7)$$

This can be expressed in the form of equation (4) as

$$Y = [-2^0 \ 2^{-1} \ \dots \ 2^{-12}] \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \dots \dots (8)$$

Equation (8) can be written as

$$Y = [-2^0 \ 2^{-1} \ \dots \dots \dots \ 2^{-12}] \begin{bmatrix} 0 \\ X_1 + X_2 \\ X_1 \\ X_1 + X_2 \\ X_2 \\ X_1 \\ X_1 + X_2 \\ 0 \\ X_2 \\ X_1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \dots \dots \dots (9)$$

Applying precise shifting, we rewrite equation (9) as:

$$Y = [2^{-1} \ 2^{-2} \ 2^{-3} \ 2^{-4} \ 2^{-5} \ 2^{-6} \ 2^{-8} \ 2^{-9}] \begin{bmatrix} X_1 + X_2 \\ X_1 \\ X_1 + X_2 \\ X_2 \\ X_1 \\ X_1 + X_2 \\ X_2 \\ X_1 \end{bmatrix} \dots \dots \dots (10)$$

Thus implementing equation (10) further reduces number of adders compared to implement equation (9). Multiplication with 2^{-i} , can be realized with the help of arithmetic shifters. In equation (10), the first row of matrix shifts right by 1 bit, second row by 2 bits and so on. More precisely, the shifts carried out are arithmetic right shifts. The output Y can be realized as a column matrix when we need the partial products. Thus, NEDA based architecture designs have less critical path compared to traditional MAC units without multipliers as well as memory. The fig (2) shows the architectural implementation of NEDA block with retiming approach.

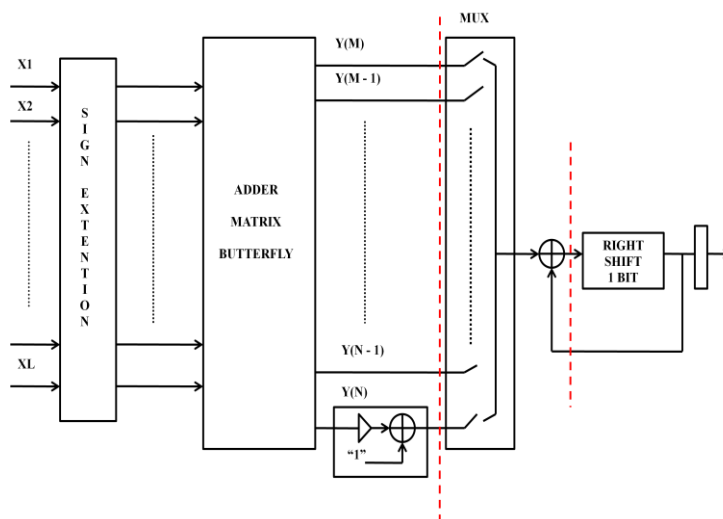


Fig 2: Architecture implementation of NEDA with retiming

3. PROPOSED WORK FOR RADIX-4 FFT USING RETIMED NEDA BLOCK

We know that design with high speed, area efficient and power efficient data paths are most interested and researched topics in VLSI design systems. Fig (3) explains the proposed architecture with retimed NEDA block. For complete architecture the longest path is identified and the cutset dotted line is drawn to adopt retiming technique. As a proof one example is shown in red dotted line in fig (3). In this paper, we have proposed the implementation of 16- point complex Radix-4 FFT using retimed NEDA block. Complex multiplications required during the process have been implemented by using NEDA. According to the radix-4 algorithm, to implement 16-point FFT, eight radix-4 butterflies are required. Four radix-4 butterflies are used in the first Stage and the other four are used in the second stage. The input is taken in normal order and the output in bit reversal order. Totally 9 NEDA blocks are used for complex computations. The inputs are $P(n), P(n+4), P(n+8), P(n+12)$ where n is 0,1,2 and 3 for 1st, 2nd, 3rd and 4th butterfly blocks respectively. The twiddle factors are $W_{16}^0, W_{16}^1, W_{16}^2, W_{16}^3$ for butterflies in first stage.

The outputs of first stage are multiplied with twiddle factors for these computation complex multiplications that consist of twiddle factor retimed NEDA blocks are used. The first radix-4 butterfly in the second stage takes the first output of the 4 radix-4 butterfly blocks used in the first stage. The second radix-4 butterfly in the second stage takes the second output of the 4 radix-4 butterfly blocks followed by the retimed NEDA block. This process continues for the rest radix-4 butterfly blocks present in the second stage. There is no need of using any NEDA block after second stage as the twiddle factor W_{16}^0 that is 1 is multiplied to the outputs of the second stage.

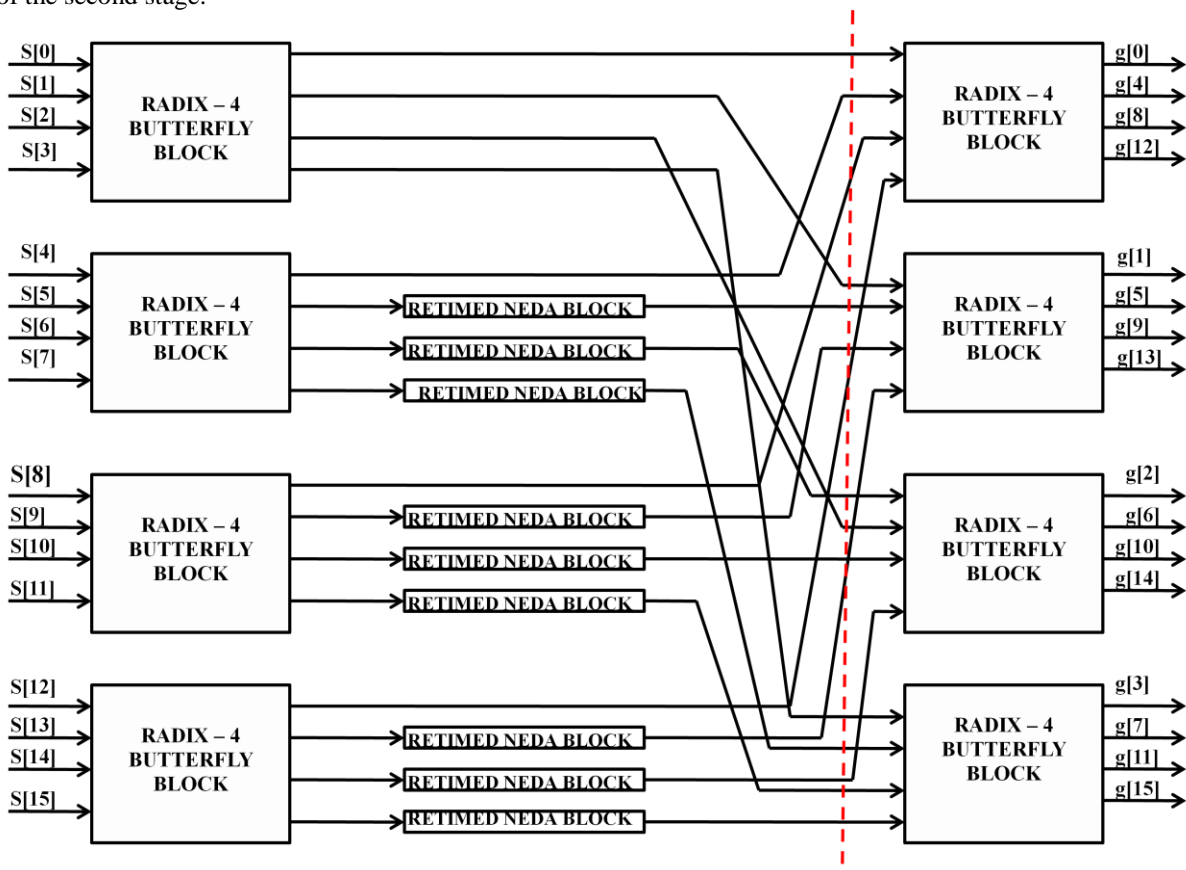


Fig 3: Schematic of proposed architecture with retimed NEDA block

In most of the adders the speed is reduced due to the propagation of carry. Now a day's CSLA are used in most of the computational systems to avoid the problem of carry propagation delay. These CSLA produces multiple carry and select one to generate sum. The computation of SRFFT using CSLA consumes more area because CSLA uses multiple pair of RCA to produce partial sum and carry by considering carry input $c_{in} = 0$ and $c_{in} = 1$, then final sum and carry are selected by mux.

We can make use of BEC instead of RCA with $c_{in} = 1$ in the structure of CSLA to reduce area and power. The regular CSLA uses two RCA whereas modified CSLA uses BEC instead of RCA with $c_{in} = 1$. The advantage of this BEC structure is it uses less number of logic gates than full adder structures. To replace N-bit RCA, N+1 bit BEC is required. The figure 4 below shows the structure of modified CSLA.

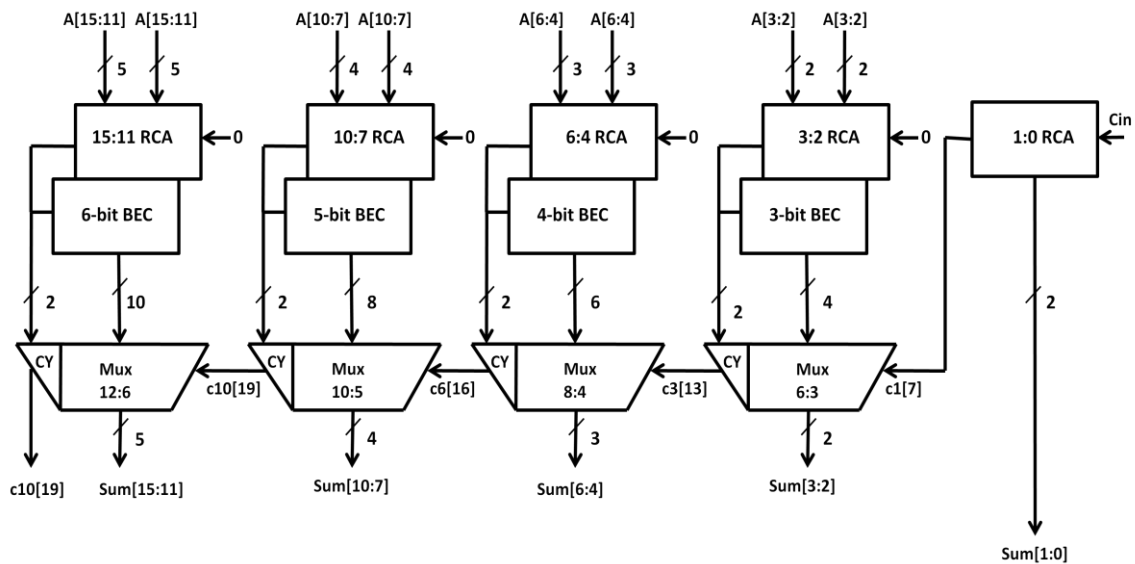


Fig 4: Schematic of proposed modified CSLA

4. RESULTS AND DISCUSSION

The synthesis report obtained from Xilinx ISE 14.7 for Spartan FPGA is shown in following tables. Table's give the device utilization summary obtained for Spartan 6 FPGA for FFT with NEDA using CSLA and FFT with NEDA using Modified CSLA. Device utilization is less for NEDA using modified CSLA. The proposed design is simulated by using ModelSim and waveform is as shown in fig 5.

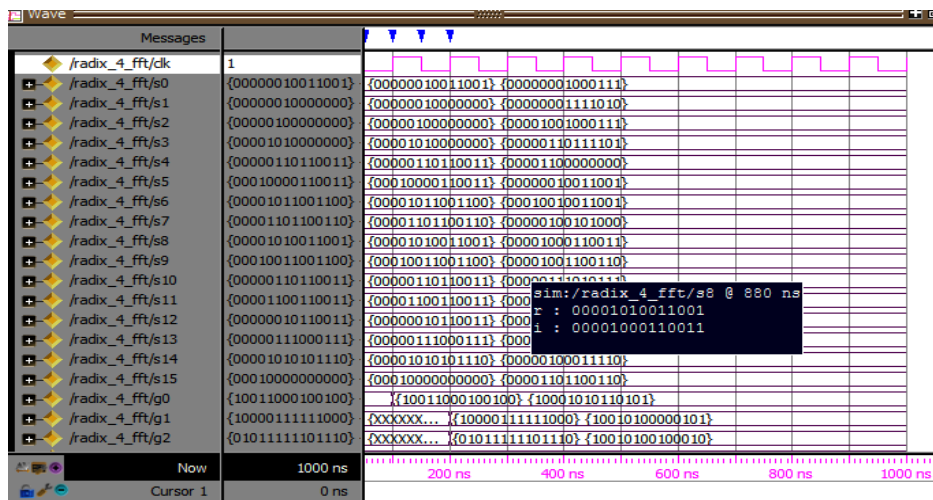


Fig 5: Simulation waveform

Table 1: Number of adders and multiplication required for different process of 16 bits.

No of Bits	Real adders			Real Multiplications		
	Radix 2	Radix 4	Proposed Design with NEDA	Radix 2	Radix 4	Proposed Design with NEDA
16	152	148	246	24	20	0

Table 2: Device utilization summary obtained for Spartan 6 using CSLA

Logic Utilization	Used	Available	Utilization
Number of slice register	4,720	126,576	3%
Number of slice LUTs	5,618	63,288	8%
Number used as logic	5,594	63,288	8%
Number of occupied slices	1,944	15,822	12%

Table 3: Device utilization summary obtained for Spartan 6 using Modified CSLA

Logic Utilization	Used	Available	Utilization
Number of slice register	2,283	126,576	1%
Number of slice LUTs	3,442	63,288	5%
Number used as logic	3,413	63,288	5%
Number of occupied slices	1,192	15,822	7%

Table 4: Number of cell count, area and Power using ASIC implementation

ASIC Implementation	Radix 4[11]	Conventional design with modified CSLA	Proposed design with Retiming approach
Area	125623	129056	140435
Total Power(mW)	8.623	3.38	2.942

5. CONCLUSION

Radix 4 complex 16 point FFT core using NEDA with modified CSLA is designed. It is a ROM-less and multiplier-less method. The proposed design is efficient in terms of hardware and power consumption. New Distributed Arithmetic (NEDA) technique is being used in many digital signal processing systems that require MAC unit. We are applying retiming approach for proposed design. The proposed modified Radix-4 structure reduces the total power consumption both in case of with and without retiming approach. ASIC synthesis results show that compared with the conventional design with modified CSLA, the proposed design with and without retiming approach achieves over 13 % as well as 66% lower power consumption when computing a 16-point complex-valued transform.

REFERENCES

- [1] Alban Ferizi, Bernhard Hoehner, Melanie Jung, Georg Fischer, and Alexander Koelpin, "Design and Implementation of a Fixed-Point Radix- 4 FFT Optimized for Local Positioning in Wireless Sensor Networks," Intl. Multi-Conf. Syst. Signals and Devices, pp. 1 – 4, Mar. 2012.
- [2] Anumol B. Chennattucherry, Diego James, " A Novel Approach to Reduce Area and Power for FFT Implementation", Proc. Intl. Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol.2, pp 130-138.
- [3] Asmitha Haveliya, Amity University Lucknow, India "Design And Simulation Of 32- Point FFT Using Radix-2 Algorithm For FPGA Implementation" Second International Conference on Advanced Computing & Communication Technologies, 2012
- [4] B. Ramkumar and Harish M Kittur, " Low-Power and Area-Efficient Carry Select Adder", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, VOL. 2, NO. 2, PP. 371-375, Feb.2012.
- [5] C. González-Concejero, V. Rodellar, " A portable hardware design of a FFT algorithm", Latin American Applied Research, 37:78-82, 2007.

- [6] D.Rajveerappa and K.Umapathy, Low-Power and High Speed 128-Point Pipeline FFT/IFFT processor for OFDM Applications, International Journal of Computer Sciences Issues, Vol.9, Issue 2, No1, March 2012.
- [7] J.Wang and L.A.Ronningen, An Implementation of Radix-4 FFT Architecture on FPGAs, Journal of Clean Energy Technologies, Vol. 2, No. 1, January 2014
- [8] Li Wenqi, Wang Xuan, and Sun Xiangran, "Design of Fixed-Point High- Performance FFT Processor," Intl. Conf. Edu. Tech. and Comput., vol. 5, pp. 139 – 143, Jun. 2010.
- [9] M.Garrido, K.K.Parhi and J.Grajal, A Pipelined FFT Architecture for Real-Valued Signals, IEEE Transactions on Circuits and Systems –I: Regular Papers, Vol.56, No.12, December 2009
- [10] M. Hasan, T. Arshan, and J.S. Thomson, "A novel coefficient ordering based low power pipelined radix-4 FFT processor for wireless LAN applications", IEEE Trans. of Consumer Electronics: 128-134,2003.
- [11] P.Augusta Sophy,R. Srinivasan,J. Raja,M. Avinash, "Analysis and design of low power radix-4 FFT processor using pipelined architecture" International Conference on Computing and Communications Technologies (ICCCT), 2015.
- [12] Samir Palnitkar, Verilog HDL- A Guide to Digital Design and Synthesis, IEEE 1364-2001 Compliant. 232 2015
- [13] S Salivahanan, A.Vallavaraj and C.Gnanapriya, Digital Signal Processing.
- [14] Stanley A.White, "Applications of Distributed Arithmetic to Digital Signal processing:A Tutorial Review," IEEE ASSP Magazine, vol. 6,no. 3, pp. 4 – 19.
- [15] X.Xiao, E.Oruklu and J.Saniie, Low Power Memory Addressing Scheme for Fast Fourier Transform Processors, IEEE 2009