International Journal of Advanced Information in Engineering Technology (IJAIET) ISSN: 2454-6933
Vol.3, No.6, June 2016

Design and simulation of 16 point radix 4 complex Fast Fourier Transform algorithm for efficient FPGA implementation using NEDA

Sangeetha Vijayan
M Tech(pursuing),Dept. of electronics,
Saintgits College of Engineering,Pathamuttom,Kottayam

Marie James
Assistant Professor,Dept. of electronics,
Saintgits College of Engineering,Pathamuttom,Kottayam

Abstract- **Transforms like Fast Fourier Transforms (FFT),Discrete Cosine Transforms are a major block in communication systems. This paper reports architecture of complex FFT core using new distributed arithmetic (NEDA) algorithm. New Distributed Arithmetic (NEDA) is one of the techniques to implement many digital signal processing systems that require multiply and accumulate units The advantage of the proposed architecture is that the entire transform can be implemented using adders and shifters only, thus minimising the hardware requirement compared to other architectures. It focuses on the development of the Fast Fourier Transform (FFT) algorithm, based on Decimation-In- Time (DIT) domain, called Radix-4 DIT-FFT algorithm.This paper proposes FPGA implementation of a 16 point radix-4 complex FFT core using NEDA with modified CSLA.The proposed design is simulated by using ModelSim and synthesised by Xilinx ISE project navigator The synthesis results show that the computation for calculating the 16 point FFT is efficient in terms of area and power using the proposed method.**

Keywords—Fast Fourier Transform (FFT), FPGA, New Distributed Arithmetic (NEDA), radix-4

I. INTRODUCTION

Today's electronic systems mostly run on batteries thus making the designs to be hardware efficient and powerefficient. Application areas such as digital signal processing, communications, etc. employ digital systems which carryout complex functionalities. Hardware efficient and power efficient architectures for these systems are most required to achieve maximum performance. Fast Fourier Transform (FFT) is one of the most efficientways to implement Discrete Fourier Transform (DFT) due to its reduced usage of arithmetic units. DFT is one of those primary tools that are used for the frequency analysis of discrete time signals and to represent a discrete time sequence in frequency domain using its spectrum samples. The analysis (forward) and synthesis (inverse) equations of an N point FFT are given below.

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

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

As evident from equation (1), thebasis of both synthesis and analysis equations remains samevthus increasing the scope of the architecture to both analyzevand synthesize. Due to increased employability of FFT invmodern electronic systems, higher radix FFTs such as radix –4, radix – 8, radix – 2^K, split radix, etc. are designed forimproved timing and reduced hardware. The basic difference of the mentioned methods lies in the structure of their butterfly units.

II. NEDA

Distributed Arithmetic (DA) has become an efficient tool to implement multiply and accumulate (MAC) unit in many digital signal processing (DSP) system. It eliminates the need of a multiplier that is used as a part of MAC unit. DA implements MAC unit by pre-computing all possible products and by storing them using a read only memory (ROM). Usage of ROM can be eliminated if one set of the inputs has a fixed value. This is done by distributing the coefficients to the inputs of the unit. This approach is called NEw Distributed Arithmetic (NEDA). Thus, using NEDA, any MAC like unit can be implemented just by using adders and shifters. Architecture designs in use either DA approach or CORDIC unit approach to implement FFT, which require ROM as en essential unit in the design. The proposed approach is based on NEDA, which does not require any ROM thus making the design to have reduced hardware. The distribution of the coefficients is done optimally to further reduce the redundant hardware units. The organization of rest of the paper is as follows. Section II briefly overviews NEDA, section III elucidates the proposed design, section IV discusses the results and performance, and section V concludes the work done in this paper. New Distributed Arithmetic (NEDA) technique is being used in many digital signal processing systems that require MAC unit. Transforms such as FFT, DCT, etc. have many multiplications that in turn require a number of multipliers. Implementation of such transforms using NEDA improves performance of the system in terms of speed, power and area. The mathematical derivation of NEDA is discussed as follows. Inner product calculation of two sequences may be represented as

$$Z = \sum_{i=1}^{k} C_i X_i \quad (3)$$

Where Ci are constant coefficients and Xi are varying inputs. Matrix representation of equation (2) may be given as

$$Z = \begin{bmatrix} C_1 & C_2 & \cdots & C_k \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{bmatrix} \quad (4)$$

Considering both Ci and Xi in 2's complement format, they may be expressed in the form

$$C_i = -C_i^M 2^M + \sum_{k=N}^{M-1} C_i^k 2^k \quad (5)$$

Where Ci= 0 or 1, k=N,N+1, … , M and Ci ^{M} is the sign bit and Ci ^{N} is the least significant bit. Substituting equation (4) in equation (3) results in the following matrix product which is modelled according to the required design.

$$Z = \begin{bmatrix} -2^0 & 2^{-1} & \cdots & 2^{-12} \end{bmatrix} \begin{bmatrix} C_1^0 & \cdots & C_k^0 \\ \vdots & \ddots & \vdots \\ C_1^{12} & \cdots & C_k^{12} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{bmatrix} \quad (6)$$

The matrix containing Ci^{k}is a sparse matrix, which meansthe values are either 0 or 1. The number of rows in C matrixdefines the precision of fixed coefficients. Equation (5) isrearranged as shown below.

$$Z = \begin{bmatrix} -2^0 & 2^{-1} & \cdots & 2^{-12} \end{bmatrix} \begin{bmatrix} W_0 \\ W_1 \\ \vdots \\ W_{12} \end{bmatrix} \quad (7)$$

$$\begin{bmatrix} W_0 \\ W_1 \\ \vdots \\ W_{12} \end{bmatrix} = \begin{bmatrix} C_1^0 & \cdots & C_k^0 \\ \vdots & \ddots & \vdots \\ C_1^{12} & \cdots & C_k^{12} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{bmatrix} \quad (8)$$

The W matrix consists of sums of the inputs depending on the coefficient values, in each row. An example that shows the NEDA operations is discussed below. Consider to evaluate the value of equation (8).

$$Y = \begin{bmatrix} \cos\frac{\pi}{8} & \cos\frac{\pi}{4} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \quad (9)$$

Equation (8) can be expressed in the form of equation (5) as shown in equation (9).

$$Y = \begin{bmatrix} -2^0 & 2^{-1} & \cdots & 2^{-12} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \quad (10)$$

Equation (9) may be rewritten as

$$Y = \begin{bmatrix} -2^0 & 2^{-1} & \cdots & 2^{-12} \end{bmatrix} \begin{bmatrix} 0 \\ X_1 + X_2 \\ X_1 \\ X_1 + X_2 \\ X_2 \\ X_1 \\ X_1 + X_2 \\ 0 \\ X_2 \\ X_1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (12)$$

Applying precise shifting, we rewrite equation (10) as

$$Y = \begin{bmatrix} 2^{-1} & 2^{-2} & 2^{-3} & 2^{-4} & 2^{-5} & 2^{-6} & 2^{-8} & 2^{-9} \end{bmatrix} \begin{bmatrix} X_1 + X_2 \\ X_1 \\ X_1 + X_2 \\ X_2 \\ X_1 \\ X_1 + X_2 \\ X_2 \\ X_1 \end{bmatrix} \quad (13)$$

Thus implementing equation (11) further reduces number of adders compared to implement equation (10). Multiplication with 2^i, ibelongs to Z^+ can be realized with the help of shifters. In equation (11), the first row of X 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 if we need the partial products. Thus NEDA based architecture designs have less critical path compared to traditional MAC units.

III. PROPOSED RADIX 4 ALGORITHM

The Discrete Fourier Transform (DFT) plays a significantly important role in many applications of digital signal processing. Basically, it has been applied in a wide range of fields such as linear filtering, spectrum analysis, digital video broadcasting and orthogonal frequency demodulation multiplexing (OFDM). The rapidly increasing demand of OFDMbased applications, including modern wireless telecommunication such as LAN, needs real-time high speed computation in Fast Fourier Transform algorithm. This has made the design of FFT processor a critical requirement for the up coming wireless technology.

With the advent of this requirement, the study of high performance VLSI FFT architecture is likewise of increasing importance. Many different hardware architectures have been proposed for the implementation of FFT algorithms. The main concern of the design approach will be power and architectural size. Among various FFT algorithms, radix-2 FFT with Cooley-Turkey algorithm, is very popular because it makes efficient use of symmetry . Several architectures have been proposed based on Cooley- Turkey algorithm to further reduce the computation complexity, including radix-4, radix-2, and split-radix. Basically, this Fast Fourier Transform algorithm use Divide-and-Conquer approach to divide the computation recursively and then extract as many common twiddle factors as possible. The number of required real additions and multiplications is usually used to compare the efficiency of different FFT algorithms. In terms of the multiplicative comparison, the split-radix FFT is computationally better to all the other algorithms because it has most trivial multiplications[3]. Eventually, this algorithm has a drawback because of irregular structure that leads this algorithm not suitable for implementation on digital signal processors. Structural regularity is also important in implementation of FFT algorithms on dedicated chips such as in ASIC (Application Specific Integrated Chip). Hence, radix-2 and radix-4 FFT algorithm are preferable in terms of speed and accuracy.

In this paper, we have proposed the implementation of 16- point complex FFT using radix-4 method. 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-4butterflies are required. Four radix-4 butterflies are used in the first stage and the other four being used in the second/final stage. The input is taken in normal order and the output in bit-reversal order. The output of each radix-4 butterfly is multiplied by the respective twiddle factors [7]. In the shown block diagram, the first stage consists of four radix-4 butterflies. The inputs to the butterflies are x(n), x(n+4), x(n+8), x(n+12) where n is 0 for first butterfly, 1 for second butterfly, 2 for the third butterfly, and 3 for the last butterfly, all of first stage. The twiddle factors are given by W_{16}^{0}, W_{16}^{q}, W_{16}^{2q},W_{16}^{3q} where *q* is 0 for first butterfly, 1 for second butterfly, 2 for third butterfly, and 3 for the last butterfly, all of first stage. The outputs of first stage are multiplied with respective twiddle factors and are given as inputs to the second stage.As proposed in our design, the complex twiddle multiplications required at the stage-1 output have been implemented by using NEDA blocks..

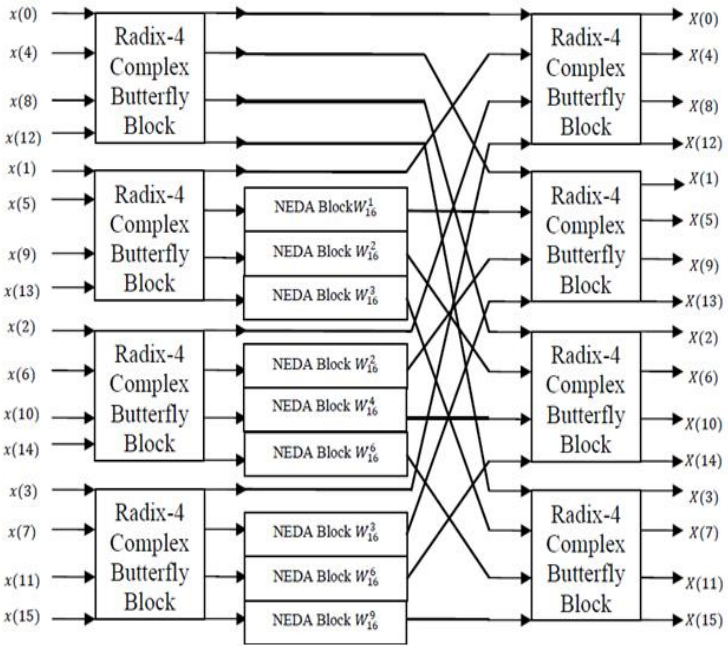

Fig. 1. Block diagram of the proposed architecture

NEDA blocks are required at the output of first stage of the 16 point FFT processor. In the second stage, 4 more radix-4 butterfly 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 NEDA block (if required). 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 □□□ □ that is 1 is multiplied to the outputs of the second stage. The final output comes in a bit-reversal order. The advantage of using radix-4 algorithm is that it retains the simplicity of radix-2 algorithm and gives the output with lesser complexity [8]. The NEDA block shown in the proposed block diagramdoes the complex multiplication of the output of the first stage and the respective twiddle factor. The twiddle factor values used here are as follows.

$$W_{16}^{1} = \cos \pi/8 - j \sin \pi/8 = 0.9238 - j0.3826$$
$$W_{16}^{2} = \cos \pi/4 - j \sin \pi/4 = 0.7071 - j0.7071$$
$$W_{16}^{3} = \cos 3\pi/8 - j \sin 3\pi/8 = 0.3826 - j0.9238$$
$$W_{16}^{4} = \cos 4\pi/8 - j \sin 4\pi/8 = 0 - j$$
$$W_{16}^{6} = \cos 3\pi/4 - j \sin 3\pi/4 = -0.7071 - j0.7071$$
$$W_{16}^{9} = \cos 9\pi/8 - j \sin 9\pi/8 = 0.9238 - j0.3826 \quad (14)$$

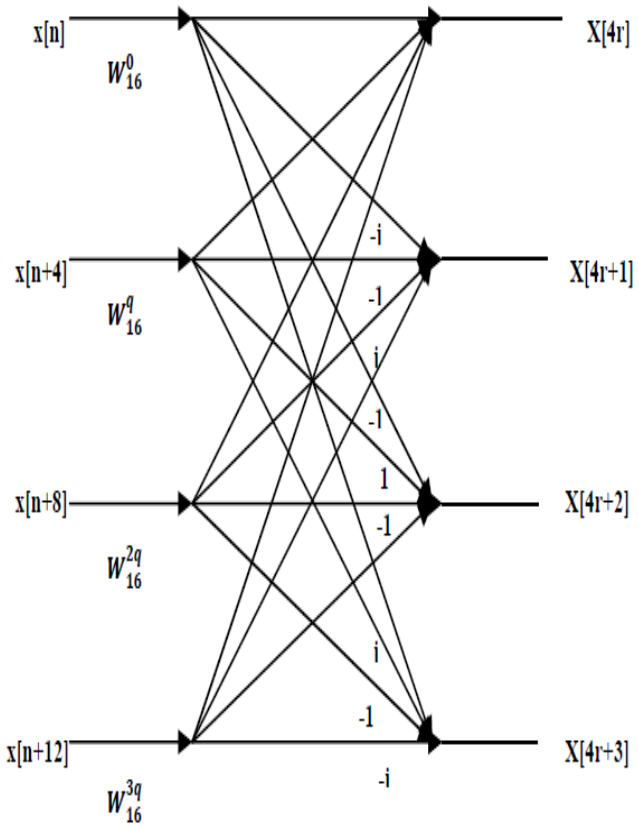

Fig. 2. Radix-4 butterfly structure

IV. NEDA USING CARRY SELECT ADDER AND MODIFIED CARRY SELECT ADDER

Design of area- and power-efficient high-speed data path logic systems are one of the most substantial areas of research in VLSI system design. In digital adders, the speed of addition is limited by the time required to propagate a carry through the adder. The sum for each bit position in an elementary adder is generated sequentially only after the previous bit position has been summed and a carry propagated into the next position.The CSLA is used in many computational systems to alleviate the problem of carry propagation delay by independently generating multiple carries and then select a carry to generate the sum . However, the CSLA is not area efficient because it uses multiple pairs of Ripple Carry Adders (RCA) to generate partial sum and carry by considering carry input Cin=0 and Cin=1, then the final sum and carry are selected by the multiplexers (mux).

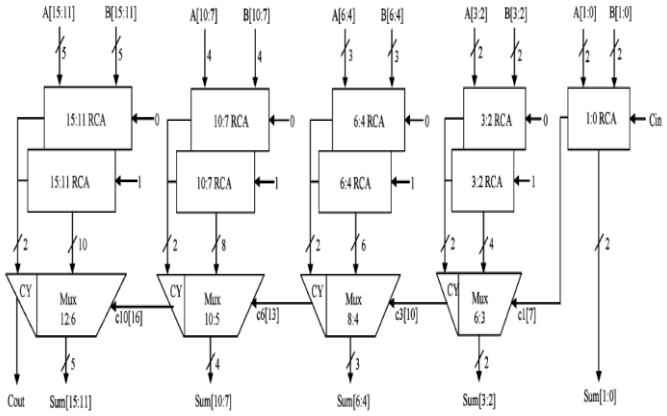

Fig.3 Structure of 16-b SQRT CLSA

The structure of the 16-b regular SQRT CSLA is shown in Fig. 3. It has five groups of different size RCA.

Binary to Excess-1 Converter(BEC) is used instead of RCA with Cin=1 in the regular CSLA to achieve lower area and power consumption. The main advantage of this BEC logic comes from the lesser number of logic gates than the n-bit Full Adder (FA) structure.

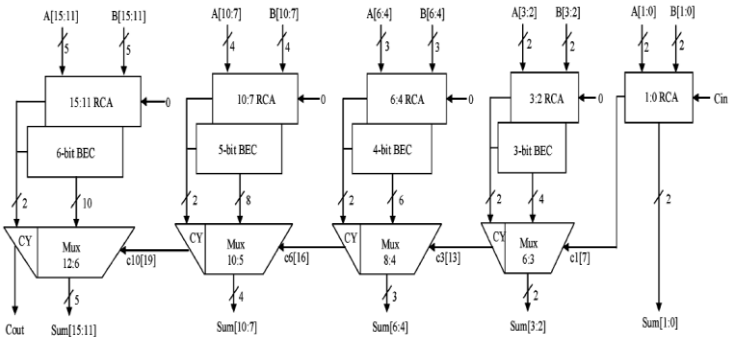

Fig..4.Modified 16-b SQRT CSLA. The parallel RCA with Cin=1 is replaced with BEC.

The structure of the proposed 16-b SQRT CSLA using BEC for RCA with Cin=1 to optimize the area and power is shown in Fig. 4

V. RESULTS AND DISCUSSION

The proposed design is simulated by using ModelSim and synthesised by Xilinx ISE project navigator.

	USED		AVAILABLE	UTILISATION	
	NEDA using CSLA	NEDA using Modified CSLA		NEDA using CSLA	NEDA using Modified CSLA
No. of slice LUTs	6471	4364	474,240	1%	1%
No. of slice registers	5192	2808	948,480	1%	1%
No. used as logic	6439	4332	474,240	1%	1%
No. of occupied slices	2924	1751	118,560	2%	1%

TABLE I Device utilisation summary for Virtex 6 FPGA

TABLE II Power and delay for Virtex 6 FPGA

	NEDA using CSLA	**NEDA using modified CSLA**
TOTAL POWER(W)	4.792	4.462
DELAY(ns)	7.022	9.612

VI. CONCLUSION

Radix 4 complex 16 point FFT core using NEDA with modified CSLA is designed. It is ROM less and multiplier less method.The proposed design is efficient in terms of hardware and power consumption. Fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform and its inverse. A DFT decomposes a sequence of values to the components of different frequencies. This operation is useful in many fields but computing it directly from the definition is often too slow to be practical.

Using NEDA, any MAC like unit can be implemented just by using adders and shifters. Architecture designs in use either DA approach or CORDIC unit approach to implement FFT, which require ROM as en essential unit in the design. The proposed approach is based on NEDA, which does not require any ROM thus making the design to have reduced hardware.The proposed design is simulated by using ModelSim and synthesised by Xilinx ISE project navigator The synthesis results show that the computation for calculating the 16 point FFT is efficient in terms of area and power using the proposed method.

REFERENCES

[1] AsmithaHaveliya, Amity University Lucknow, India "Design And Simulation Of 32-Point FFT Using Radix-2 Algorithm For FPGA Implementation" 2012 Second International Conference on Advanced Computing & Communication Technologies

[2] W. Hussain, F. Garzia, J. Nurmi, "Evaluation of Radix-2 and Radix-4 FFT Processing on a Reconfigurable Platform", *13th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems*, 2010

[3] M. Hasan and T. Arslan, "Implementation of low power fft processor cores using a novel order-base processing scheme," in Proc. IEEE Circuits Devices Syst., June 2003, vol. 150, pp. 149–154.

[4] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline fft processors for vlsi implementations," IEEE Trans. Comput., vol. 33, no. 5, pp. 414– 426, May 1984.

[5] J. W. Cooley and J. W. Tukey, "An Algorithm for Machine Calculation of Complex Fourier Series," Math.Comput., vol. 19, pp. 297–301, Apr. 1965

[6] Douglas L. Jones "Decimation-in-Time (DIT) Radix-2 FFT Algorithms" Connexions module: m12016

[7] Very High Speed Integrated Circuit Hardware Description Language.URL: http://electrosofts.com/vhdl/

[8] C. González-Concejero, V. Rodellar," A portable hardware design of a FFT algorithm", Latin American Applied Research,37:78-82, 2007

[9] C. S. Burrus and T. W. Parks,"DFT/FFT and Convolution Algorithms", New York,NY : John Wiley,1985