Efficient VLSI Architectures of Split-Radix FFT using New Distributed Arithmetic

Ansuman DiptiSankar Das, Abhishek Mankar, N Prasad, K. K. Mahapatra, Ayas Kanta Swain

Abstract*—Fast Fourier transform (FFT) has become ubiquitous in many engineering applications. Efficient algorithms are being designed to improve the architecture of FFT. Among the different proposed algorithms, split-radix FFT has shown considerable improvement in terms of reducing hardware complexity of the architecture compared to radix-2 and radix-4 FFT algorithms. New distributed arithmetic (NEDA) is one of the most used techniques in implementing multiplier-less architectures of many digital systems. This paper proposes efficient multiplier-less VLSI architectures of split-radix FFT algorithm using NEDA. As the architecture does not contain any multiplier block, reduction in terms of power, speed, and area can greatly be observed. One of the proposed architectures is designed by considering all the inputs at a time and the other is designed by considering 4 inputs at a time, the total number of inputs in both cases being 32. The proposed designs are designed using both FPGA as well as ASIC design flows. 180nm process technology is used for ASIC implementation. The results show the improvements of proposed designs compared to other architectures.

***Index Terms*—Split-radix, FFT, VLSI, NEDA, multiplier-less, FPGA, ASIC.**

I. INTRODUCTION

Fast Fourier Transform (FFT) has become ubiquitous in many engineering applications [1]. High-speed FFT architectures are necessary to implement several communication systems, signal processing systems, etc. [2] – [4]. The FFT blocks are also used in mechanical engineering and civil engineering applications [5] – [6]. FFT has been considered as the most efficient way of implementing the discrete Fourier transform (DFT) and it was first implemented in 1965 [7]. The efficiency of the FFT algorithm lies in its reduced number of arithmetic operations. DFT has the order of $O(N^2)$ arithmetic operations whereas FFT has the order of $O(N \log N)$ arithmetic operations. If the architecture is designed for complex inputs, the number of arithmetic operations becomes approximately double when compared to those which are designed for real inputs.

Manuscript received on March, 2013.

Ansuman DiptiSankar Das, Dept. Of ECE, NIT Rourkela, India.
Abhishek Mankar, Dept. Of ECE, NIT Rourkela, India.
N Prasad,Dept. Of ECE, NIT Rourkela, India.
K. K. Mahapatra, Professor, Dept. Of ECE, NIT Rourkela, India.
Ayas Kanta Swain, Asst. Professor, Dept. Of ECE, NIT Rourkela, India.

One of the disadvantages of conventional FFT architectures is the presence of multiplier blocks, which has increased hardware, increased power consumption and reduced operating frequency. The basic FFT design is based on radix-2 butterfly block, which was proposed by Cooley-Tukey [7]. Recent advances in the algorithm include FFT architectures based on higher and split-radix such as radix-4, radix-8, radix-2^k, etc. [8] – [12].

Split-radix FFT is one of the FFT algorithms that use combination of different radix FFT. Split-radix FFT algorithm combines simplicity of radix-2 FFT with less computational complexity radix-4 FFT. The advantage of split-radix FFT is that it has considerably fewer number of arithmetic computations compared to that of radix-4 and radix-2 FFT. Split-radix also has several other advantages such as regular structure, no reordering of internal signals except for outputs, etc. Since it mostly uses radix-2 block in its architecture, it is possible to implement split-radix FFT for inputs of kind 2^k, k being an integer.

Distributed Arithmetic (DA) was invented about 30 years ago and has since seen widespread applications in area of VLSI implementation of DSP algorithms [13]. DA has become one of the most efficient tools in implementation of multiply and accumulate (MAC) unit in several DSP systems. Most of the applications, for example discrete cosine transform (DCT), discrete wavelet transform (DWT) calculation, are commonly implemented using DA based approach as they all are hardware intensive with multipliers and MAC units. MAC unit is implemented using DA by pre-computing all possible products and then storing them in a read only memory (ROM). In simple words, DA computes the inner product of two multi-dimensional vectors. Thus, increase in the number of dimensions increases the memory requirement to store all the obtained products. This is due to the reason that, increase in number of dimensions increases the number of obtained partial products. The elimination increased memory requirement is possible only if one or both of the inputs has a fixed set of coefficients. This method is commonly known as NEw Distributed Arithmetic (NEDA) [14]. Thus, using NEDA, distribution of arithmetic is done on the coefficient values instead of doing on the inputs. This results in memory-less DA architecture of the implemented systems. Conventional NEDA based architectures are bit-serial in nature. Depending on the application and requirement, they can be designed as digit-serial or bit-parallel architectures. Thus, NEDA is classified under the family of shift-add algorithms. VLSI implementation of NEDA becomes simpler if the constant coefficients have magnitudes those are less than one.

DSP system design techniques such as folding, pipelining have always improved performance of the systems in terms of hardware, latency, frequency, etc. In DSP architectures, systematic control circuits are determined by using the folding transformation.

In folding technique, time multiplexing of algorithm operations is done, by reducing to a single functional unit. Thus, in DSP systems, folding technique provides a means of trading time for area. Conventional folding technique can be used to reduce the number of hardware functional units by a factor of N at the expense of increasing the computation time or multiplexing time by a factor of N [15].This technique also helps in data allocation in the required registers. To avoid excess amount of registers that are generated in these architectures while folding, there are techniques to minimise the number of registers needed to implement DSP architectures through folding.

In the following sections, first we present a brief overview of split-radix FFT and NEDA. Then, we propose multiplier-less VLSI architectures of split-radix using NEDA. Later, we give the FPGA and ASIC implementation summary of proposed designs. Next, we compare the proposed architectures with the existing ones. Finally, we conclude the paper with mentioning possible further improvements.

II. OVERVIEW OF SPLIT-RADIX FFT AND NEDA

A. Split-radix FFT

While calculating FFT using Radix-2 method, it can be concluded that the even-numbered points and the odd-numbered points are computed independently. This leads to the possibility of using different computational methods for different independent parts of the algorithm which will reduce computational complexity. Split-radix algorithm uses the above method by combining the simplicity of radix-2 algorithm and lesser computational complexity of radix-4 algorithm, achieving the lowest number of arithmetic operation count to compute DFT of power-of-two sizes *N*. Split-radix method recursively expresses DFT of length *N* in terms of one smaller DFT of length *N*/2 and two smaller DFTs of length *N*/4. Split-radix is only applicable when *N* is a multiple of 4, but we can combine this with other FFT algorithms.

The *N-point* DFT of a sequence $x(n)$ is given by

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

$$k = 0,1, \dots \dots, N-1 \tag{1}$$

Where $W_N^{nk} = e^{-j2\pi nk/N}$ is known as the twiddle factor.

The algorithm for the fast and less complexity computation of the DFT by Split-radix (SRFFT) was developed by Duhamel and Hollmann [16], [17] for data sequences having a length *N* that is an integer power of 2. According to them, the even-numbered samples of the *N*-point DFT can be calculated by

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

$$k = 0,1, \dots \dots, \frac{N}{2} - 1 \tag{2}$$

Those even-numbered DFT points can be calculated without any additional multiplications. So, radix-2 algorithm is sufficient for the above calculation. The odd-numbered samples $X(2k+1)$ requires an additional multiplication of W_N^n . To implement this, radix-4 algorithm is used for its lesser computational complexity.

Using radix-4 algorithm for the odd –numbered samples of the *N*-point DFT, the following N/4-point DFT$_s$ are obtained.

$$X(4k+1) = \sum_{n=0}^{\frac{N}{4}-1} [\left\{x(n) - x\left(n + \frac{N}{2}\right)\right\} - j\left\{x\left(n + \frac{N}{4}\right) - x\left(n + \frac{3N}{4}\right)\right\}] W_N^n W_{N/4}^{nk}$$

$$k = 0,1, \dots \dots, \frac{N}{4} - 1 \tag{3}$$

And

$$X(4k+3) = \sum_{n=0}^{\frac{N}{4}-1} [\left\{x(n) - x\left(n + \frac{N}{2}\right)\right\} + j\left\{x\left(n + \frac{N}{4}\right) - x\left(n + \frac{3N}{4}\right)\right\}] W_N^{3n} W_{N/4}^{nk}$$

$$k = 0,1, \dots \dots, \frac{N}{4} - 1 \tag{4}$$

Hence, the *N*-point DFT now has been decomposed into one *N/2*-point DFT without phase factor and another two *N/4*-point DFTs with phase factor. Figure 1 shows the split-radix butterfly unit.

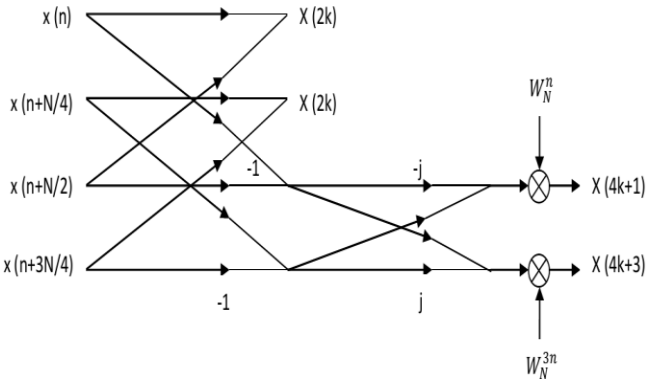

Fig. 1. Split-radix butterfly unit

B. New Distributed Arithmetic (NEDA)

NEw Distributed Arithmetic (NEDA) technique is being used in many digital signal processing systems that require MAC unit as their computational block. Transforms such as FFT, DCT, etc. have many multipliers that in turn require more hardware. Implementation of such transforms using NEDA improves performance of the system in terms of area, speed and power. The mathematical derivation of NEDA is discussed below.

Inner product calculation of two sequences can be represented as

$$Z = \sum_{i=1}^{k} C_i X_i \tag{5}$$

Where C_i are constant fixed coefficients and X_i are varying inputs. Matrix representation of equation (5) may be given as

$$Z = [C_1 \quad C_2 \quad \cdots \quad C_k] \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{bmatrix} \tag{6}$$

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

$$C_i = -C_i^M 2^M + \sum_{k=N}^{M-1} C_i^k \, 2^k \tag{7}$$

Where $C_i = 0 \; or \; 1, \; k = N, N+1, \ldots, M$ and C_i^M is the sign bit and C_i^N is the least significant bit. Substituting equation (7) in equation (6) results in the following matrix product which is modelled according to the required design of FFT.

$$Z = [-2^0 \quad 2^{-1} \quad \cdots \quad 2^{-12}] \begin{bmatrix} C_1^0 & \cdots & C_k^0 \\ \vdots & \ddots & \vdots \\ C_1^{12} & \cdots & C_k^{12} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{bmatrix} \tag{8}$$

The matrix containing C_i^k is a sparse matrix, which means the values are either 1 or 0. The number of rows in C matrix defines the precision of fixed coefficients used. Equation (8) is rearranged as shown below.

$$Z = [-2^0 \quad 2^{-1} \quad \cdots \quad 2^{-12}] \begin{bmatrix} W_0 \\ W_1 \\ \vdots \\ W_{12} \end{bmatrix} \tag{9}$$

Where $$\begin{bmatrix} W_0 \\ W_1 \\ \vdots \\ W_{12} \end{bmatrix} = \begin{bmatrix} C_1^0 & \cdots & C_k^0 \\ \vdots & \ddots & \vdots \\ C_1^{12} & \cdots & C_k^{12} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{bmatrix} \tag{10}$$

In each row, the W matrix consists of sums of the inputs depending on the coefficient values. An example that shows the NEDA operations is discussed below. Consider to evaluate the value of equation (11).

$$Y = \begin{bmatrix} \cos\frac{\pi}{8} & \cos\frac{\pi}{4} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \tag{11}$$

Equation (11) can be expressed in the form of equation (8) as shown in equation (12).

$$Y = [-2^0 \quad 2^{-1} \quad \cdots \quad 2^{-12}] \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \tag{12}$$

Equation (12) may be rewritten as

$$Y = [-2^0 \quad 2^{-1} \quad \cdots \quad 2^{-12}] \begin{bmatrix} 0 \\ X_1 + X_2 \\ X_1 \\ X_1 + X_2 \\ X_2 \\ X_1 \\ X_1 + X_2 \\ 0 \\ X_2 \\ X_1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{13}$$

Applying precise shifting, we rewrite equation (13) as

$$Y = [2^{-1} \quad 2^{-2} \quad 2^{-3} \quad 2^{-4} \quad 2^{-5} \quad 2^{-6} \quad 2^{-8} \quad 2^{-9}] \begin{bmatrix} X_1 + X_2 \\ X_1 \\ X_1 + X_2 \\ X_2 \\ X_1 \\ X_1 + X_2 \\ X_2 \\ X_1 \end{bmatrix} \tag{14}$$

Thus implementing equation (14) further reduces number of adders compared to implement equation (13). Multiplication with $2^{-i}, i \in Z^+$ can be realized with the help of arithmetic shifters. In equation (14), 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 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.

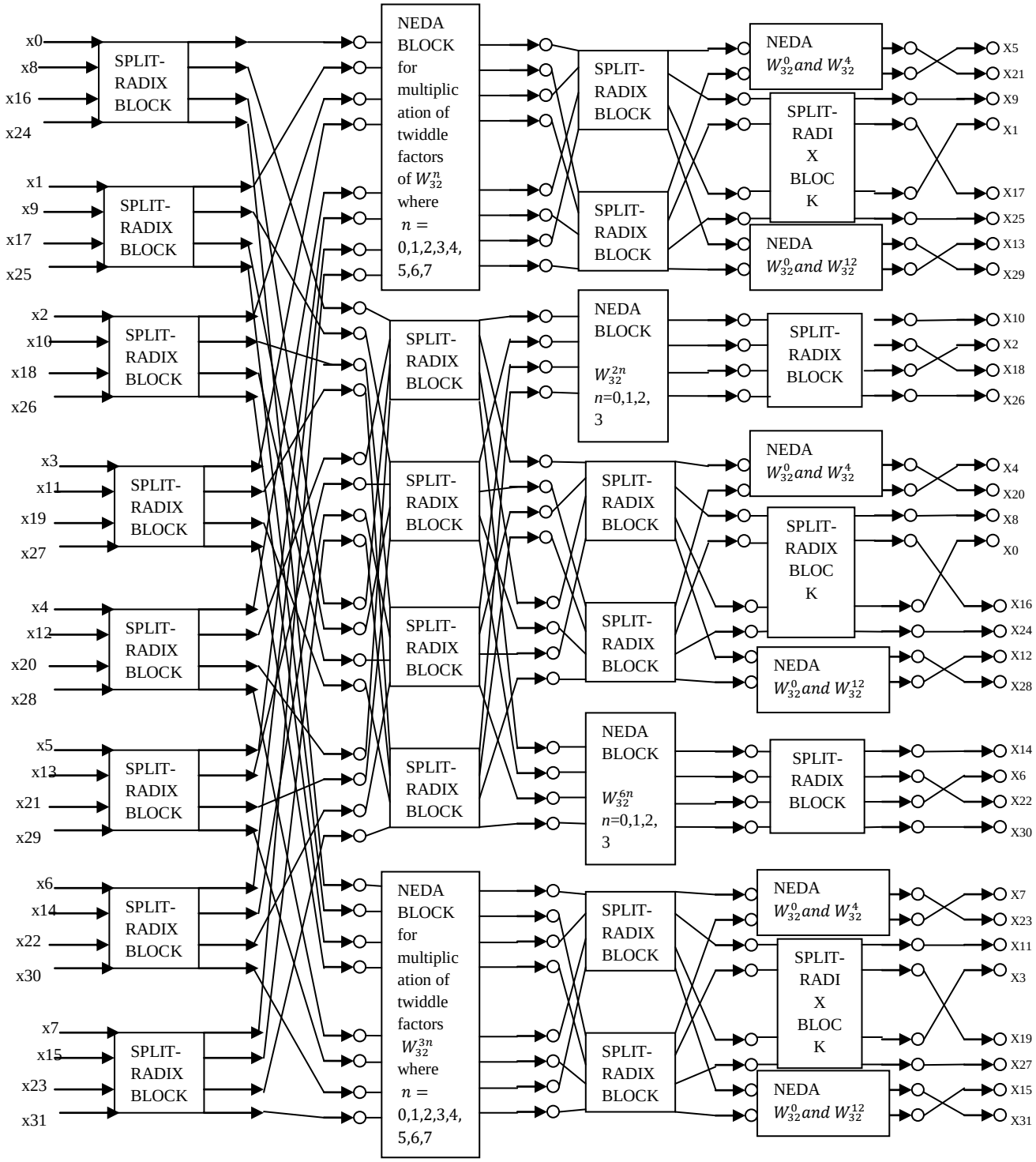

Fig. 2. Proposed architecture – I of 32-point split-radix FFT

III. PROPOSED DESIGNS

A. Proposed Architecture – I

A 32-point complex split-radix FFT has been proposed in this paper. 32 complex inputs have been taken with a precession of 16 bits, in parallel. The number of stages to calculate the final output is 5. The inputs are taken in normal order and the outputs are in bit-reversal order. The even-numbered samples have been implemented by radix-2 FFTalgorithm and the odd-numbered samples have been implemented using radix-4 FFT algorithm. The twiddle factor multiplications have been implemented using NEDA technique. The proposed architecture – I is shown in figure 2. In stage-I, eight radix-4 butterfly modules have been used. The inputs to each radix-4 butterfly present in stage-I are $x(n), x\left(n+\frac{N}{4}\right), x\left(n+\frac{N}{2}\right), x(n+\frac{3N}{4})$ where $0 \le n \le \frac{N}{4}-1$ respectively. The first output of each split-radix butterfly present in stage-I are represented by $S1(0), S1(1), \ldots\ldots, S1(7)$ respectively. The second output of each split-radix butterfly of stage-I are represented by $S1(8), S1(9), \ldots\ldots, S1(15)$ respectively. Similarly the third and fourth output of each split-radix butterfly of stage-I are represented as $S1(16), S1(17), \ldots, S1(23)$

and $S1(24), S1(26), \ldots, S1(31)$ respectively.

In stage-II, the samples $S1(0), S1(1), \ldots\ldots, S1(7)$ are multiplied by twiddle factor of W_N^n and the samples $S1(24), S1(26), \ldots\ldots, S1(31)$ are multiplied by twiddle factor of W_N^{3n} where N=32 and $0 \le n \le \frac{N}{4} - 1$ respectively. Those inner product calculations have been done by NEDA technique to achieve a multiplier-less architecture. The rest of stage-I samples are fed to four split-radix butterfly units and the outputs are given to stage-III. In stage-III, the samples $\{S2(0), S2(1), \ldots\ldots, S2(7)\}$, $\{S2(12), S2(13), \ldots\ldots, S2(19)\}$, $\{S2(24), S2(25), \ldots, S2(31)$ are fed to six split-radix butterfly units and the outputs are given to stage-IV respectively. The other samples of stage-III are multiplied by twiddle factor of W_N^{2n} and W_N^{6n} where N=32 and $0 \le n \le \frac{N}{8} - 1$ respectfully.

In stage-IV, five more split-radix butterfly units have been used and the inputs and outputs of those are clearly shown in figure. The twiddle factor that is to be multiplied in stage-IV whenever required is W_N^{4n} and W_N^{12n} where N=32 and $0 \le n \le \frac{N}{16} - 1$. The final stage (stage-V) uses only radix-2 butterfly units whenever required. The twiddle factor to be multiplied in stage-V is W_N^0 since $0 \le n \le \frac{N}{32} - 1$ that is n=0. The NEDA technique has been used here whenever there is a need for the calculation of inner products. We got the final output at the end of stage-V. Figure 3 shows the split-radix butterfly used in the proposed architectures.

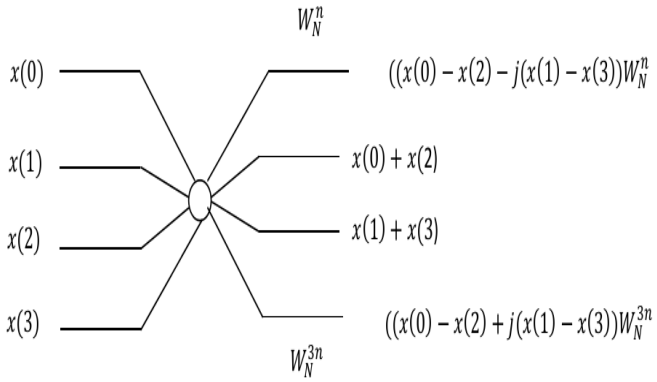

Fig. 3. Split-radix butterfly used in proposed designs

B. Proposed Architecture – II

The draw-back of the proposed architecture – I lies in its huge number of input-output pins, which makes the design less implementable both on FPGAs as well as an ASIC. To overcome the above draw-back, an intelligent way of implementing the split-radix FFT is done through folding.

The proposed architecture – II, shown in figure 4, takes 4 inputs at a time which sums up to 8 clock cycles to read all the 32 inputs. For every clock cycle, the outputs of the first stage split-radix block are stored in registers and this process continues till all 32 outputs are stored. Later, the stored outputs are processed for second stage computations which consist of either NEDA blocks or split-radix blocks. The outputs of second stage split-radix blocks are stored in 16 registers for further processing. The outputs of second stage NEDA blocks and some outputs of second stage split-radix blocks are given to third stage split-radix blocks. The remaining outputs of second stage split-radix blocks are given to NEDA blocks of third stage. Some outputs of third stage split-radix blocks are given to fourth stage NEDA blocks. The remaining outputs of third stage split-radix blocks along with third stage NEDA blocks are given to fourth stage split-radix blocks.

The outputs of fourth stage NEDA blocks and some outputs of fourth stage split-radix blocks are fed to fifth stage radix-2 blocks. Rest of the outputs of fourth stage split-radix blocks are directly mapped to outputs.

TABLE I. DATAFLOW TABLE FOR INPUT-OUTPUTS OF PROPOSED ARCHITECTURE – II

Clock cycle	Inputs	Outputs
1	x0,x8,x16,x24	
2	x1,x9,x17,x25	P0,P8,P16,P24
3	x2,x10,x18,x26	P1,P9,P17,P25
4	x3,x11,x19,x27	P2,P10,P18,P26
5	x4,x12,x20,x28	P3,P11,P19,P27
6	x5,x13,x21,x29	P4,P12,P20,P28
7	x6,x14,x22,x30	P5,P13,P21,P29
8	x7,x15,x23,x31	P6,P14,P22,P30
9		P7,P15,P23,P31
10	P8,P12,P16,P20	
11	P9,P13,P17,P21	Q8,Q12,Q16,Q20
12	P10,P14,P18,P22	Q9,Q13,Q17,Q21
13	P11,P15,P19,P23	Q10,Q14,Q18,Q22
14		Q11,Q15,Q19,Q23
15	W0,W2,W4,W6	
16	W1,W3,W5,W7	S0,S2,S4,S6
17	Q12,Q14,Q16,Q18	S1,S3,S5,S7
18	Q13,Q15,Q17,Q19	R12,R14,R16,R18
19	W8,W10,W12,W14	R13,R15,R17,R19
20	W9,W11,W13,W15	S8,S10,S12,S14
21		S9,S11,S13,S15
22	S2,S3,S4,S5	
23	T8,T9,T10,T11	Y9,U3,U4,Y25
24	R14,R15,R16,R17	Y10,V9,V10,Y26
25	T20,T21,T22,T23	Y8,U15,U16,Y24
26	S10,S11,S12,S13	Y14,V21,V22,Y30
27		Y11,U11,U12,Y27
28	L0,L1,U3,U4	Y5,Y21,Y1,Y17
29	L6,L7,V9,V10	Y13,Y29,Y2,Y18
30	L12,L13,U15,U16	Y4,Y20,Y0,Y16
31	L18,L19,V21,V22	Y12,Y28,Y6,Y22
32	L8,L9,U11,U12	Y7,Y23,Y3,Y19
33	L14,L15,0,0	Y15,Y31,0,0

In table I, the internal signals W0 to W15 are obtained after multiplying the signals P0 to P7 and P24 to P31 with their respective twiddle factors of second stage. Similarly, the

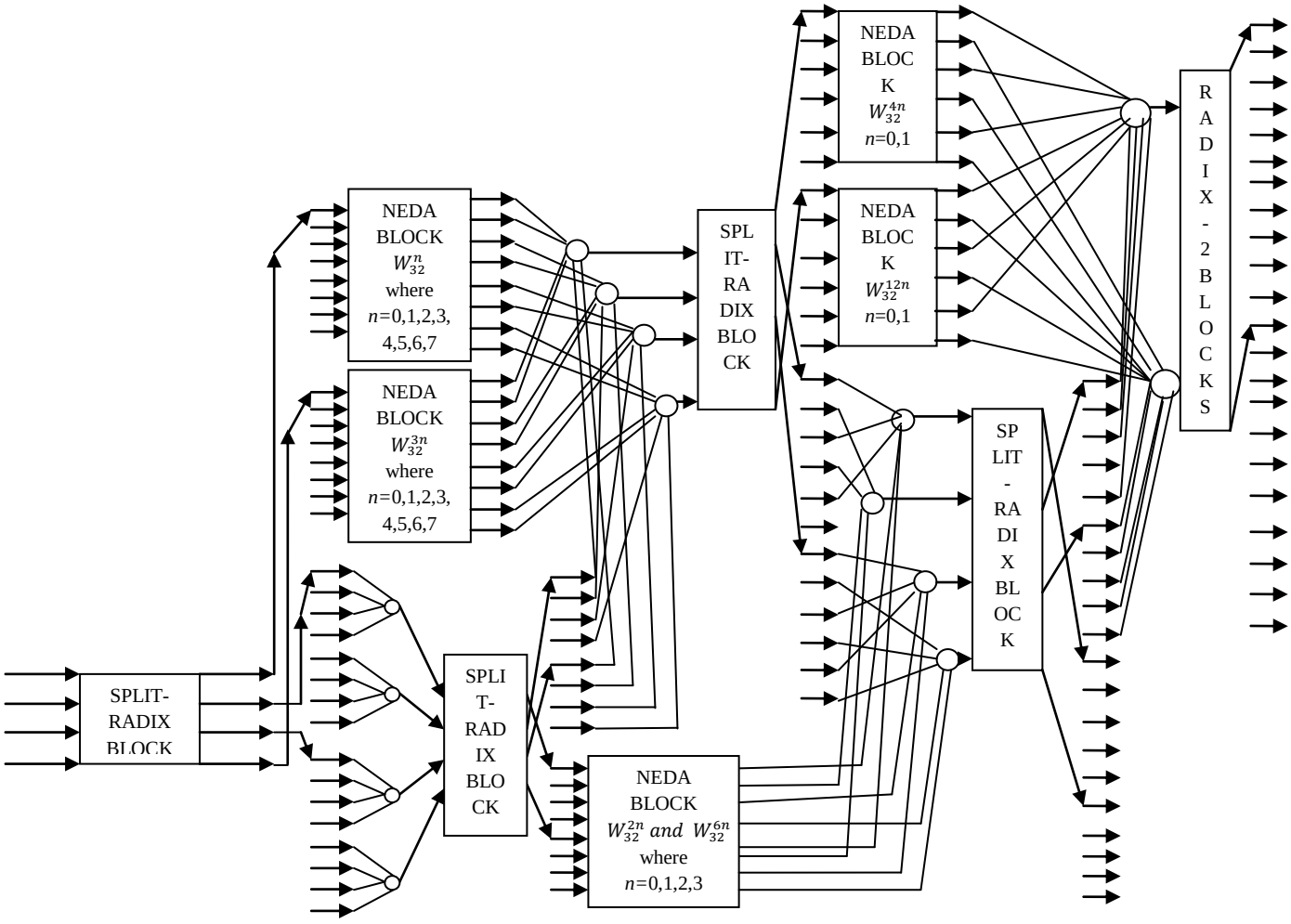

Fig. 4. Proposed architecture – II, of 32-point split-radix FFT

signals T8, T9, T10, T11, T20, T21, T22 and T23 are obtained after multiplying the signals Q8, Q9, Q10, Q11, Q20, Q21, Q22 and Q23 with their corresponding twiddle factors of third stage. Finally, the signals L0, L1, L6, L7, L12, L13, L18, L19, L8, L9, L14 and L15 are obtained after multiplying the signals S0, S1, S6, S7, R12, R13, R18, R19, S8, S9, S14 and S15 with their twiddle factors of fourth stage respectively. The twiddle factors have been performed using NEDA blocks at respective stages. The outputs of the proposed architecture start coming from the 23rd clock cycle till 33rd clock cycle in bit-reversal order.

IV. FPGA AND ASIC IMPLEMENTATION SUMMARY

The proposed architectures have been implemented using Xilinx ISE as well as Altera Quartus II, wherever applicable. The proposed architecture – I can operate at a maximum frequency of 100.368 MHz on Xilinx Virtex-5 FPGAs. The outputs of proposed architecture – I are obtained after 45 ns, which results in its latency, in parallel. But, as the number of IOBs is too high to accommodate, we go for proposed architecture – II. Table II shows the FPGA device utilization summary of proposed architecture – II. The power has been calculated using Xilinx XPower Analyzer.

TABLE II. FPGA DEVICE UTILIZATION SUMMARY OF PROPOSED ARCHITECTURE – II

FPGA device: XC5VLX330T-2FF1738	Proposed Architecture – II	
	Used	Utilization
Number of occupied slices	2426	51840 (4%)
Number of slice registers	5010	207360 (2%)
Number of slice LUTs	7099	207360 (3%)
Frequency	527.329 MHz	
Dynamic Power at maximum frequency	0.40262 W	

Table III shows the comparison results of the proposed architecture – II, with the architecture mentioned in [18]. The comparison has been made using Altera Quartus II and its Cyclone II family of FPGA. From table III, it is clear that, the proposed architecture gives better results in terms of speed, power and area.

Table IV shows the ASIC implementation of the proposed architectures in 0.18µm process technology using Synopsys DC for logic synthesis and Cadence SoC Encounter for physical design. The process technology that has been followed to carryout physical design of the proposed

architectures is UMC 0.18µm mixed mode generic core.

TABLE III. COMPARISON OF PROPOSED ARCHITECTURE – II USING ALTERA CYCLONE II FAMILY OF FPGA

FPGA comparison results using Altera Cyclone II	[18]	Proposed Architecture – II
Number of inputs	32	32
Combinational functions	1442	14304
Logic registers	857	1123
18x18 multipliers	4	0
Memory	2(1K)	0
Execution time (µs)	7.995	0.14457
Frequency (MHz)	100	210.97
Device	EP2C35	EP2C70

TABLE IV. ASIC IMPLEMENTATION RESULTS OF PROPOSED ARCHITECTURES USING SYNOPSYS DC AND CADENCE SOC ENCOUNTER

ASIC implementation results using Synopsys DC	Process technology: 0.18µm	
	Proposed Architecture – I	Proposed Architecture – II
Total cell area	1063769.421537	803245.469974
Total dynamic power	84.1841 mW	14.9286 mW
Add-sub width	16 bits	16 bits
Slack at 100 MHz	3.68 ns	6.62 ns

The physical design of proposed architectures has been made in such a way that the timing constraints are met after both placement as well as routing. The layouts are shown in figure 5 and figure 6. The core utilization of proposed designs has been set to 0.8 to avoid congestion while routing. The proposed architectures have been routed using Nano route. The slack achieved for proposed architecture – I at 100 MHz clock is 3.68 ns and for proposed architecture – II is 6.62 ns. From table IV it is clear proposed architecture – II gives better results in terms of area and power compared to proposed architecture – I.

Fig. 5. Physical Layout of proposed architecture – I

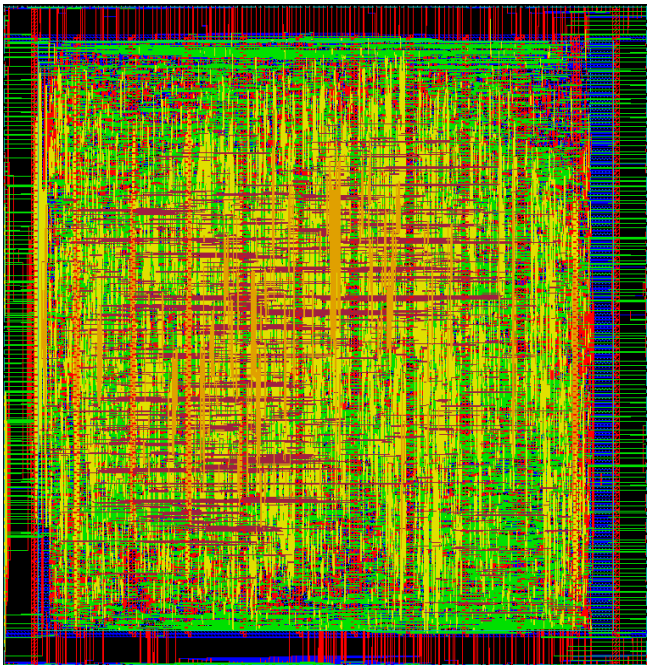

Fig. 6. Physical layout of proposed architecture – II

V. CONCLUSIONS

This paper has reported two novel and efficient architectures of split-radix FFT using NEDA. Both proposed architectures are designed for complex inputs with a data width of 16 bits, maintained constant all along. The simulation outputs of proposed architectures have not shown much deviation from numerical values. The proposed architectures are multiplier-less as well as memory-less ones. Proposed architecture – I is implemented as a fully dedicated architecture that takes all inputs in parallel and it has less delay of 4 clock cycles. But, proposed architecture – I has huge number of input-output pins;

this drawback has been overcome in the later proposed architecture. Proposed architecture – II is implemented using folding which is folded so as to take 4 inputs at a time. Both the proposed architectures are implemented sequentially which results in a form of pipelining. The data flow of proposed architecture – II is clearly mentioned in table II. Proposed architecture – II gives a maximum frequency of 527.329 MHz on Xilinx Virtex-5 FPGA and 210.97 MHz on Altera Cyclone II EP2C70 FPGA, thus showing its applicability in communication systems. There is a huge decrement in power of proposed architecture – II when compared. ASIC implementation of proposed architectures has been done using Synopsys and Cadence tools.

REFERENCES

1. P. Duhamel and M. Vetterli, "Fast Fourier Transforms: A Tutorial Review and A State of The Art," *IEEE Signal Processing Society*, vol. 4, no. 19, 1990, pp. 259 – 299.
2. Y.-W. Lin, H.-Y. Liu, and C.-Y. Lee, "A 1-GS/s FFT/IFFT processor for UWB applications," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 8, Aug. 2005, pp. 1726 – 1735.
3. S.-N. Tang, J.-W. Tsai, and T.-Y. Chang, "A 2.4-GS/s FFT Processor for OFDM-Based WPAN Applications," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 57, no. 6, Jun. 2010, pp. 451 – 455.
4. J ohn G. Proakis, Dimitris G. Manolakis, "Digital Signal Processing: Principles, Algorithms, and Applications", Prentice- Hall, 1998.
5. Z. Ismail, N. H. Ramli, Z. Ibrahim, T. A. Majid, G. Sundaraj, and W. H. W. Badaruzzaman, "Design Wind Speeds using Fast Fourier Transform: A Case Study," *Computational Intelligence in Control*, Idea Group Publishing, 2012, ch. XVII.
6. Robert Frey, "The FFT Analyzer in Mechanical Engineering Education," *Sound and Vibration: Instrumentation Reference Issue*, Feb. 1999, pp. 1 – 3.
7. James W. Cooley and John W. Tukey, "An Algorithm for Machine Calculation of Complex Fourier Series," *Mathematics of Computation*, vol. 19, 1965, pp. 297 – 301.
8. Mario Garrido, J. Grajal, M. A. Sánchez, and Oscar Gustafsson, "Pipelined Radix-2^k Feedforward FFT Architectures," *IEEE Trans. VLSI Syst.*, vol. 21, no. 1, Jan. 2013, pp. 23 – 32.
9. Y. Chen, Y. Tsao, Y. Wei, C. Lin, and C. Lee, "An indexed- scaling pipelined FFT processor for OFDM-based WPAN applications," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 55, no. 2, Feb. 2008, pp. 146–150.
10. M. Shin and H. Lee, "A high-speed four-parallel radix-2^4 FFT processor for UWB applications," *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2008, pp. 960–963.
11. F. Arguello and E. Zapata, "Constant geometry split-radix algorithms," *Journal of VLSI Signal Processing*, 1995.
12. Steven G. Johnson and Matteo Frigo, "A Modified Split-Radix FFT with Fewer Arithmetic Operations," *IEEE Trans. Signal Processing*, vol. 55, no. 1, Jan. 2007, pp. 111 – 119.
13. Stanley A. White, "Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review," *IEEE ASSP Magazine*, vol. 6, no. 3, Jul. 1989, pp. 4 – 19.
14. Wendi Pan, Ahmed Shams, and Magdy A. Bayoumi, "NEDA: A NEw Distributed Arithmetic Architecture and its Application to One Dimensional Discrete Cosine Transform," *Proc. IEEE Workshop on Signal Processing Syst.*, Oct. 1999, pp. 159 – 168.
15. Keshab K. Parhi, "VLSI Digital Signal Processing Systems: Design and Implementation", Wiley, 1999.
16. P. Duhamel and H. Hollmann, "Split-radix FFT algorithm," *Electron. Lett.*, vol. 20, no. 1, Jan. 1984, pp. 14 – 16.
17. P. Duhamel, "Implementation of split-radix FFT algorithms for complex, real, and real-symmetric data," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, Apr. 1986, pp. 285 – 295.
18. Cynthia Watanabe, Carlos Silva, and Joel Muñoz, "Implementation of Split-Radix Fast Fourier Transform on FPGA," *Proc. Programmable Logic Conference*, vol. 6, Mar. 2010, pp. 167 – 170.

AUTHORS PROFILE

Ansuman DiptiSankar Das was born in Balasore, India, in 1986. He received his B. Tech degree in electronics and telecommunication engineering form BPUT, Odisha, in 2007. He's currently pursuing his M. Tech in VLSI Design and Embedded Systems at National Institute of Technology Rourkela, India. His current areas of interest are VLSI architectures for digital signal processing and design of real-time embedded systems.

Abhishek Mankar was born in Munger, India, in 1987. He received his B. Tech degree in electronics and communication engineering from WBUT, Kolkata, in 2010. He's currently pursuing his M. Tech in VLSI Design and Embedded Systems at National Institute of Technology Rourkela, India. His current areas of interest are FSM based VLSI designs, high performance VLSI architectures using NEDA.

N Prasad was born in Anantapur, India, in 1990. He received his B. Tech degree in electronics and communication engineering from JNTU Hyderabad, in 2011. He's currently pursuing his M. Tech in VLSI Design and Embedded Systems at National Institute of Technology Rourkela, India. His current areas of interest are VLSI system architectures, design implementation and applications of CORDIC, multiplier-less VLSI system designs.

Kamalakanta Mahapatra received his B. Tech degree (with honors) from the Regional Engineering College (currently, the National Institute of Technology), Calicut, India, in 1985, M. Sc. (Engg.) degree from the Regional Engineering College (currently, the National Institute of Technology Rourkela), Rourkela, India, in 1989, and Ph. D. degree from the Indian Institute of Technology, Kanpur, India, in 2000.

Currently, he is with the National Institute of Technology Rourkela as a professor in the Electronics and Communication Engineering department. His research interests include power electronics, embedded computing, real-time systems, and very large scale integration design. Dr. Mahapatra is a fellow of the Institution of Engineers (India) in the Electronics and Communication division.

Ayas Kanta Swain received his B. Tech degree from IGIT, Sarang, Odisha, India, in 2001, M. Tech (research) degree from the National Institute of Technology Rourkela, in 2010. He is currently an assistant professor in department of Electronics and Communication Engineering at the National Institute of Technology Rourkela where he is also pursuing his Ph. D. degree. His current areas of interest are VLSI Design, Embedded Systems, system on chip designs, and network on chip designs.