HARDWARE IMPLEMENTATION OF FFT/IFFT ALGORITHMS INCORPORATING EFFICIENT COMPUTATIONAL ELEMENTS

E.Konguvel[1], M. Kannan[2]
Department of Electronics Engineering,
Madras Institute of Technology Campus, Anna University,
Chennai – 600044, India.
konguart08@gmail.com[1], mkannan@annauniv.edu[2]

***ABSTRACT* – Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) computation involves a quite large number of complex multiplications and complex additions. Optimizing the FFT processing elements in terms of complex multiplication reduces area and power consumption. In this work, complex multipliers in the FFT processors are replaced by area and power efficient approximate multipliers. In image and signal processing applications which can tolerate minimum error, accurate computing units are always not necessary. Accurate computing units can be replaced with approximate computing units. Approximate computing can decrease the design complexity with an increase in area and power efficiency. In this paper, approximate 8- and 16-bit multipliers are designed and implemented in radix-2 butterfly unit which is the crucial computational component in FFT/IFFT processing. The designed FFT/IFFT processing units are analyzed, synthesized and simulated in Altera Cyclone II EP2C35F672C6 FPGA device. Experimental results shows that the proposed 16-point FFT architecture incorporating approximate complex multiplier achieves an area efficiency of about 33.47% and power efficiency of 1.8% when compared to accurate 16-point FFT processor. The 8 point and 16 point Decimation-In-Time (DIT) – FFT incorporating approximate computational elements operates at a speed of 26.69Gbps and 46.20Gbps respectively.**

Keywords— FFT, IFFT, Approximate computing, FPGA implementation.

I. INTRODUCTION

A comparative study on efficient FFT/IFFT algorithms, architectures and significance of length of the data sequences for FFT/IFFT computations on corresponding applications were discussed [1]. Several real-time implementation strategies and complexities during run-time environment of FFT/IFFT processing units were stated in [2 - 4]. The implementation of a complex multiplier plays a major role in the butterfly element which is the integral part of FFT/IFFT processing algorithms. Multiplication is the critical computation compared to addition because switching activity of multipliers are high compared to other data path units of any processing architecture. Hence, it is extremely essential to implement a complex multiplier that operates in low power and of high efficiency. It is also obvious that precise computation of a complex multiplier consumes more power than approximate computing [5]. Approximate computation will decrease the design complexity with an increase in performance and decrease in power consumption with a minimum error that would be tolerant based on the specific application. The accuracy of this approximate complex multiplier can also be increased by using optimal error correction schemes [6].

Many approximation techniques are available in the literature to improve power and energy efficiency of complex multipliers. The most common techniques are truncation, voltage over scaling and simplification of logic complexity by modifying the truth table. Truncation helps in complexity reduction by eliminating partial product term matrix lower parts in the complex multiplier [7 - 8]. This truncation may result in an error. Voltage over scaling is lowering the supply voltage below $V_{dd} - V_{critical}$ which may result in transient circuit timing errors [9]. Selection of approximate arithmetic architecture is very important in voltage over scaling since different hardware implementation of same arithmetic function responds differently [10]. An approximate 2X2 multiplier with tunable error characteristics (3.32% of error) with an average power savings of 31.78% ~ 45.40% when compared with precise 2X2 multiplier is proposed [11]. In designing fast multiplier, compressors have been used widely to speed up the partial product reduction stage. Two designs of approximate compressor have been proposed where exact full adder cells are replaced by the approximate full adder cells [12]. However, this is not efficient because the error rate of this compressor is more than 53%. The use of m x m multiplier to perform n x n multiplication have been proposed. It takes m consecutive bits of an n-bit operand, either starting from MSB or ending at LSB and apply two segments that include leading ones from two operands (SSM) to an m x m multiplier [13]. The partial products are decomposed into two major units and processed in parallel to reduce the delay in fixed-width multipliers. This fixed-width multiplier with column bypassing technique is optimal for low power error tolerant applications [14]. Various approximate complex multipliers (wallace, array and dadda) are designed by partial product perforation technique. This technique is to perforate any two rows from the original partial products generated. The perforation skips the generation of partial product and decreases the number of operands to be accumulated, hence reducing power consumption of about 50% [15].

The multi-bit adders in digital signal architecture are designed by using the imprecise full adder cells that results in a power savings of 69% when compared to that of a design with precise full adder cells [16]. The usage of compressors and compressor-adders in complex multipliers reduces the power consumption and has good area efficiency as well [17]. Different sizes of approximate compressors were used in building the multiplier using an algorithm that allocates the compressors with minimum error [18]. Modified approximate compressors are used in order to design a low-power and high-speed multiplier with minimum error values [19]. The proposed approximate complex multiplier design is used in building the FFT computational units with an average error of 2.5% to 3.5% that consumes low power with less area utilization. The proposed approximate complex multiplier, radix-2 FFT butterfly module, 8 point and 16 point FFT algorithms are synthesized and simulated in Altera Quartus II simulator tool and implemented in Altera Cyclone II EP2C35F672C6 FPGA device.

The organization of this paper is as follows. A brief introduction to approximate multipliers and its structures are discussed in section II. The implementation of approximate

multipliers in FFT/IFFT computations are given in section III. Comparative analysis and results of approximate with precise 8 and 16 points FFT computations are discussed in section IV. Concluding remarks are stated in section V.

II. APPROXIMATE MULTIPLIERS

Any arithmetic processing can be performed on an approximate basis. Implementation of multiplier includes three phases, generation of partial products, partial product reduction and final addition. Power consumption, critical path delay and circuit complexity are dominated by the partial product reduction stage. Many techniques have been proposed to reduce the critical path in the multipliers. The most commonly used technique for partial product reduction is that use of compressors.

The compressor is made of full adders or half adders to count the number of one's in the input. Lower order compressors consumes lesser area. Hence we use 4-2 and 5-3 compressor for approximating altered partial products.

In this work, an approximate complex 16-bit multiplier is designed and it is used in building FFT computational elements. The original generated partial products are altered to propagate and generate signals using following equation.

$$P_{m,n} = A_{m,n} + A_{n,m} \quad (1)$$

$$G_{m,n} = A_{m,n} \,.\, A_{n,m} \quad (2)$$

This propagate signals are approximated using approximate half adders, approximate full adders, approximate 4:2 compressor adders. The approximation is applied using simple OR gate for generate signals. The first stage of partial product reduction using approximation is shown in Figure 1. The second stage of reduction uses twelve approximate full adder, four 4:2 approximate compressor adder, eleven 5:3 approximate compressor adder circuits.

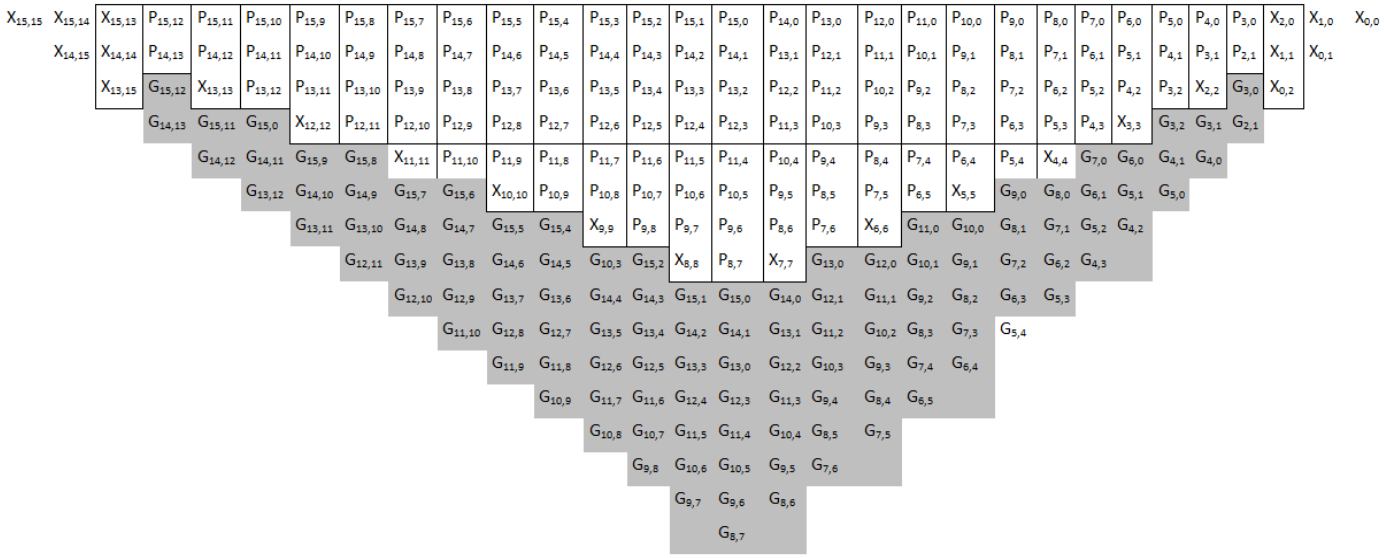

Fig. 1. Partial product reduction using approximation

A. *Approximate half adder:*

In a precise Half Adder (HA), XOR gate is used to calculate *"Sum"*. But XOR gates consumes more area and power [5]. So, XOR gate of the precise half adder is replaced with OR gate for approximation. The logic difference between precise and approximate half adder is shown in the table I. The following equation (3) & (4) illustrates the approximate half adder circuit.

$$\text{Sum} = A + B \quad (3)$$

$$\text{Carry} = A \,.\, B \quad (4)$$

TABLE I. TRUTH TABLE OF HALF ADDER.

Input		Precise HA		Approx. HA	
A	***B***	***Sum***	***Carry***	***Sum***	***Carry***
0	0	0	0	0	0
0	1	1	0	1	0
1	0	1	0	1	0
1	1	0	1	1	1

B. *Approximate full adder:*

To calculate the sum and carry of Full Adder (FA) three XOR gates are necessary. For the approximation of full adder, one XOR gate is replaced with OR gate in sum computation. The logic difference between precise and approximate full adder is shown in the table II. The following equations (5) – (7) illustrates the approximate full adder circuit.

$$X = A + B \quad (5)$$

$$\text{Sum} = X \text{ xor } C \quad (6)$$

$$\text{Carry} = X \,.\, C \quad (7)$$

TABLE II. TRUTH TABLE OF FULL ADDER.

Input			Precise FA		Approx. FA	
A	***B***	***C***	***Sum***	***Carry***	***Sum***	***Carry***
0	0	0	0	0	0	0
0	0	1	1	0	1	0
0	1	0	1	0	1	0
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	1
1	1	0	0	1	1	0
1	1	1	1	1	0	1

C. *Approximate 4:2 and 5:3 compressor:*

Compressors and compressor-adders are the fundamental building blocks of multipliers to accumulate generated partial products. Compressor-adders are used in the second stage of multiplier architecture to reduce the number of partial products and also to reduce the gate count and critical path delay. The use of approximate compressors in the least significant bits decreases power consumption and circuit area. The logic differentiation between precise and approximate 4:2 compressor-adder is shown in the table III. The following equations (8) – (11) illustrates the approximate 4:3 compressor-adder circuits.

$$X = A \,.\, B \quad (8)$$

$$Y = C \,.\, D \quad (9)$$

$$\text{Sum} = (A \text{ XOR } B) + (C \text{ XOR } D) + X \,.\, Y \quad (10)$$

$$\text{Carry} = X + Y \quad (11)$$

In 5:3 compressor-adder, five input bits are summed up to produce three output bits. This compressor will be used in the second stage of partial product reduction stage. The logic differentiation between precise and approximate 5:3 compressor-adder is shown in the table III. The following equations (12) – (14) illustrates the 5:3 approximate compressor adder circuits. The outputs S1 and S2 will remain same for precise as well as approximate 5:3 compressor-adders. But the output S3 in the precise 5:3 compressor-adder calculation is replaced by S3' in approximate 5:3 compressor-adder which is given in equation (15).

$$S1 = A \text{ XOR } B \text{ XOR } C \text{ XOR } D \text{ XOR } E \quad (12)$$

$$S2 = C \text{ XOR } D \quad (13)$$

$$S3 = A \,.\, (\sim(A \text{ XOR } B)) + B \,.\, (A \text{ XOR } B) \,.\, (C \,.\, (\sim(A \text{ XOR } B \text{ XOR } C \text{ XOR } D))) + D \,.\, (A \text{ XOR } B \text{ XOR } C \text{ XOR } D) \quad (14)$$

$$S3' = C \,.\, D \quad (15)$$

TABLE III. TRUTH TABLE OF 4:2 COMPRESSOR ADDER.

Input				Precise 4:2		Approx. 4:2	
A	*B*	*C*	*D*	*Sum*	*Carry*	*Sum*	*Carry*
0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0
0	0	1	0	1	0	1	0
0	0	1	1	0	1	0	1
0	1	0	0	1	0	1	0
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	0
0	1	1	1	1	1	1	1
1	0	0	0	1	0	1	0
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	0
1	0	1	1	1	1	1	1
1	1	0	0	0	1	0	1
1	1	0	1	1	1	1	1
1	1	1	0	1	1	1	1
1	1	1	1	0	0	1	1

III. FFT/IFFT ALGORITHM

A. *Preliminaries :*

The basic principle behind this FFT algorithm is to decompose the input sequence of length N into smaller Discrete Fourier Transform (DFT) sequences. Let x(n) be an N-point sequence, where N is assumed to be a power of 2. The DFT X(k) and its inverse (IDFT) x(n) of an N-point sequence can be mathematically given as,

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}, 0 \le k \le N-1 \quad (16)$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi kn/N}, 0 \le n \le N-1 \quad (17)$$

The exponential term specified in equation (16) and (17) represents the twiddle factor needed for FFT and IFFT computations respectively. The decomposition can be classified as Decimation-In-Frequency (DIF) and Decimation-In-Time (DIT), depending upon the partition that takes place from input and output data points respectively. For real-valued application DIT algorithm is preferable as it involves less number of computation when compared to DIF. Depending upon the application any of the algorithms can be used. Approximate multiplier is used in the radix-2 DIT butterfly used in this work.

B. *Radix 2 DIT Butterfly :*

The basic radix-2 butterfly algorithm for DIT-FFT is shown in figure 2. In figure 2, A and B indicate the complex input from preceding stage while C and D indicate the complex output of the present stage (or complex input to the subsequent stage). The twiddle factors W_N are defined as the co-efficients which are used to compute results from the preceding stage and to form inputs to the subsequent stages of FFT algorithm.

TABLE. IV. TRUTH TABLE OF 5:3 COMPRESSOR ADDER.

Input					Precise 5:3			Approx. 5:3
A	*B*	*C*	*D*	*E*	*S1*	*S2*	*S3*	*S3'*
0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0
0	0	0	1	0	1	0	0	0
0	0	0	1	1	0	1	0	0
0	0	1	0	0	1	0	0	0
0	0	1	0	1	0	1	0	0
0	0	1	1	0	0	1	0	0
0	0	1	1	1	1	1	0	0
0	1	0	0	0	1	0	0	0
0	1	0	0	1	0	1	0	0
0	1	0	1	0	0	1	0	0
0	1	0	1	1	1	1	0	0
0	1	1	0	0	0	1	0	1
0	1	1	0	1	1	1	0	1
0	1	1	1	0	1	1	0	1
0	1	1	1	1	0	0	1	1
1	0	0	0	0	1	0	0	0
1	0	0	0	1	0	1	0	0
1	0	0	1	0	0	1	0	0
1	0	0	1	1	1	1	0	0
1	0	1	0	0	0	1	0	0
1	0	1	0	1	1	1	0	0
1	0	1	1	0	1	1	0	0
1	0	1	1	1	0	0	1	0
1	1	0	0	0	0	1	0	0
1	1	0	0	1	1	1	0	0
1	1	0	1	0	1	1	0	0
1	1	0	1	1	1	0	1	0
1	1	1	0	0	1	1	0	1
1	1	1	0	1	1	0	1	1
1	1	1	1	0	1	0	1	1
1	1	1	1	1	1	0	1	1

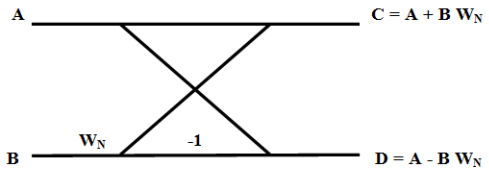

Fig. 2. Radix-2 butterfly structure.

From figure 2., C and D can be written as

$$C_r + jC_j = (A_r + jA_j) + (B_r + jB_j)(W_r + jW_j) \quad (18)$$

$$D_r + jD_j = (A_r + jA_j) - (B_r + jB_j)(W_r + jW_j) \quad (19)$$

Rewrite equation (18) – (19) as,

$$C_r = A_r + B_rW_r - B_jW_j \quad (20)$$

$$C_j = A_j + W_jB_r + W_rB_j \quad (21)$$

$$D_r = A_r - B_rW_r + B_jW_j \quad (22)$$

$$C_j = A_j - W_jB_r - W_rB_j \quad (23)$$

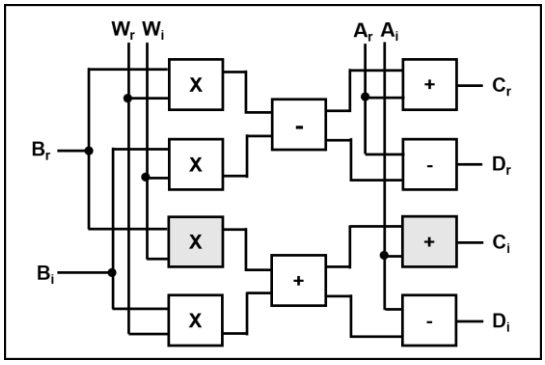

Fig. 3. Radix 2 butterfly structure with multipliers.

From the equations (20) – (23), the radix-2 butterfly structure for DIT-FFT computation is shown as figure 3. One radix-2 butterfly operation requires four complex multiplications, three complex additions and three complex subtractions. Complex multiplications requires more area and power when comparing with complex additions and complex subtractions. So in conventional radix-2 butterfly structures, the complex precise multiplications are replaced by the approximate multiplications which requires reduced area utilization as well as reduced power consumption.

C. *16 Point DIT FFT:*

In DIT-FFT algorithm input bits are bit-reversed and output bits are natural in order. Twiddle factors for every stages are accessed from the look-up-tables. The twiddle factors are computed using binary scaling technique. Binary scaling technique is widely used in digital arithmetic processing to perform a floating point multiplication using the integers. The flow graph for 16-point FFT algorithm is shown in figure 4.

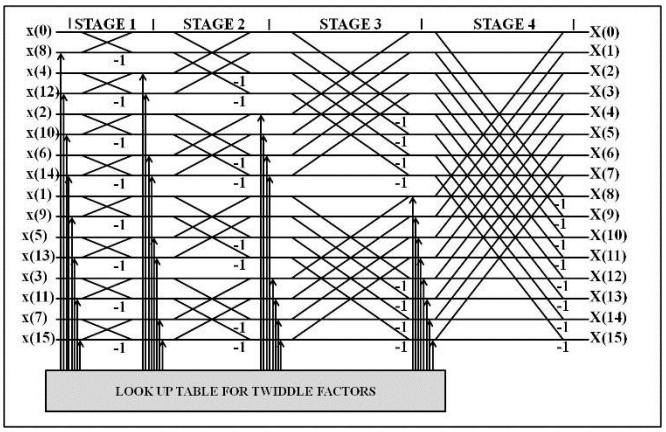

Fig. 4. Flow diagram of FFT 16 point.

IV. RESULTS AND DISCUSSIONS

The approximate 16-bit complex multiplier and radix-2 butterfly architecture incorporating the approximate complex multiplier using Verilog hardware description language (IEEE 1364) and implemented in Altera EP2C35F672C6 Cyclone II FPGA device. A precise 16-bit complex multiplier is also designed and the results were compared with the approximate 16-bit complex multiplier in terms of area utilization, delay and power consumption. The 8 point and 16 point DIT-FFT incorporating the approximate complex multiplier are also designed and results were compared with that of conventional approaches. The experimental results obtained using Altera Quartus simulator were presented and discussed in the succeeding sub-sections.

A. *Synthesis results of approximate 16-bit multiplier:*

The approximate 16-bit complex multiplier occupies 424 logical elements out of 33216 which achieves an area utilization of 1.27%. From the table V, it is concluded that 28.97% of logical elements have been reduced in the approximate multipliers when compared to that of 16-bit precise multipliers.

TABLE. V. SYNTHESIS RESULTS OF APPROX. 16-BIT MULTIPLIER.

Parameter	Precise 16-bit	Approx. 16-bit
Logical Elements	597	424
FPGA Area Utilization %	1.79	1.27
Delay (ns)	63.74	23.09
Total Thermal power (mW)	228.61	225.32
Core Dynamic power (mW)	5.88	4.30
Core Static power (mW)	80.32	80.31
I/O Thermal power (mW)	142.41	140.71

B. *Timing and Power analysis of approximate multiplier:*

The power play analyser and the classic timing analyser tools are used to analyse the power consumption and timing parameters. Approximate 16-bit complex multiplier achieves a worst case propagation delay of 23.09ns. The approximate multiplier consumes a total thermal power of 225.32mW which includes a core dynamic power of 4.30mW, core static power of 80.31mW and I/O thermal power of 140.71mW. When compared to accurate multiplier, 1.43% of total thermal power has been reduced in approximate multiplier.

C. *Synthesis results of radix-2 using proposed multiplier:*

The RTL schematic of radix–2 butterfly using the proposed approximate 16-bit multiplier is shown in the figure 5. The radix-2 structure occupies 677 logical elements, a total combinational function of 673 and 64 dedicated logic registers out of 33216 total logical elements. The table VI illustrates the synthesis results of radix-2 butterfly incorporating precise 16-bit complex multiplier and approximate 16-bit complex multiplier. It is evident that 17.68% of logical elements have been reduced in the radix-2 butterfly that has approximate multiplier.

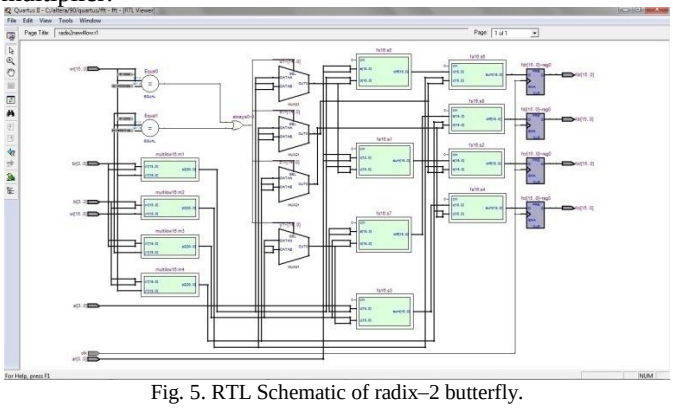

Fig. 5. RTL Schematic of radix–2 butterfly.

TABLE. VI. SYNTHESIS RESULTS OF RADIX-2 BUTTERFLY.

Parameter	Radix-2 (With precise mul.)	Radix-2 (With approx. mul.)
Logical Elements	821	677
FPGA Area Utilization %	2.46	2.03
Delay (ns)	26.29	22.56
Total Thermal power (mW)	334.83	311.88
Core Dynamic power (mW)	15.27	20.13
Core Static power (mW)	80.68	80.60
I/O Thermal power (mW)	238.87	211.15

D. *Timing and Power Analysis of radix-2 using proposed multiplier:*

Radix-2 butterfly with approximate multiplier design achieves a worst case setup time of 22.56ns, combinational delay of 8.708ns. It consumes a total thermal power of 311.88mW which includes a core dynamic power of 20.13mW, core static power of 80.60mW and I/O thermal power of 211.15mW. When compared to radix-2 structure with precise multiplier, 1.43% of total thermal power has been reduced in radix-2 structure with approximate multiplier.

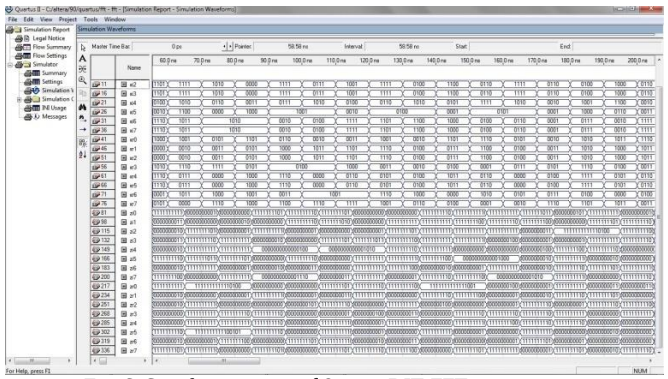

Fig. 6. Simulation report of 8 point DIT-FFT computation.

E. Synthesis Result of 8 & 16 point DIT-FFT computation:

The 8 point DIT-FFT with approximate multiplier occupies 1549 logical elements, 1489 combinational function out of 33216 total elements. It is shown in the table VII that 30% of logical elements have been reduced when compared with the 8 point DIT-FFT with precise multiplier. The 16 point DIT-FFT with approximate multiplier occupies 2721 logical elements, 2663 combinational function out of 33216 total logical elements. It is also shown in the table VII that 33.47% of logical elements have been reduced when compared with the 16 point DIT-FFT with precise multiplier.

The real and imaginary parts of the input sequence and twiddle factor for all the stages of DIT-FFT have been assigned in the waveform editor and simulated. The simulation results for 8 point DIT-FFT and 16 point DIT-FFT were shown in the figure 6 and 7 respectively.

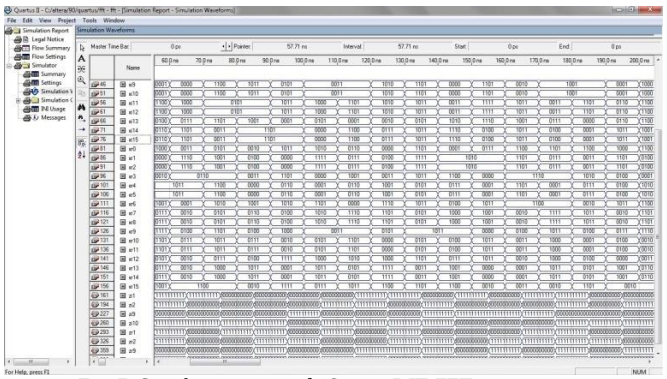

Fig. 7. Simulation report of 16 point DIT-FFT computation.

F. Timing and Power Analysis of 8 & 16 point DIT-FFT computation:

The 8 point DIT-FFT with approximate computational units achieves delay of 12.03ns at 83.15MHz of operating speed whereas conventional architectures have a delay of 17.95ns at 17.88MHz of operating speed. The 8 point DIT-FFT with approximate computational units consumes a total thermal power of 392.91mW whereas conventional architectures consumes 387.88mW of total power.

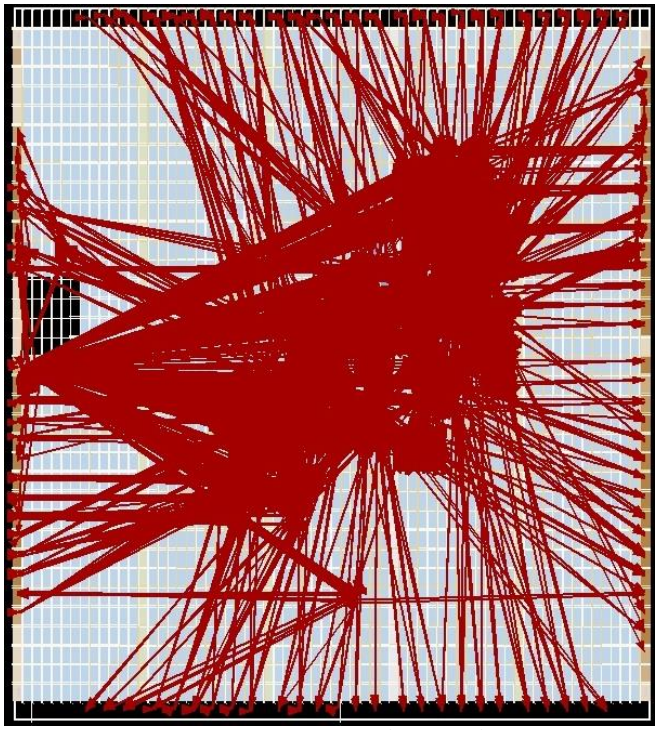

Fig. 8. Chip planner view with fan in and fan out.

The 16 point DIT-FFT with approximate computational units achieves delay of 13.86ns at 72.17MHz of operating speed whereas conventional architectures have a delay of 20.13ns at 49.68MHz of operating speed. The 16 point DIT-FFT with approximate computational units consumes a total thermal power of 417.26mW whereas conventional architectures consumes 424.84mW of total power. The chip planner view of 16 point DIT-FFT with approximate computational elements is given in the figure 8. The chip planner view provides a visual display of logic utilization, fan-ins and fan-outs of the proposed system.

TABLE. VII. SYNTHESIS RESULTS OF 8 AND 16 POINT DIT-FFT

Parameter	8 point DIT-FFT		16 point DIT-FFT	
	Precise	*Approx.*	*Precise*	*Approx.*
Logical Elements	2213	1549	4088	2721
FPGA area utilization (%)	6.66	4.66	12.31	8.19
Combinational Slices	2148	1489	3994	2663
Dedicated Registers	488	496	718	618
Pin Count	321	321	385	385
Delay (ns)	17.95	12.03	20.13	13.86
Power Dissipation (mW)	392.91	387.88	424.84	417.26
Frequency (MHz)	55.70	83.15	49.68	72.17
Throughput (Gbps)	17.88	26.69	31.84	46.20

V. CONCLUSION & FUTURE SCOPE

In this work, an area efficient low-power radix-2 butterfly incorporating approximate computational elements is used in building 8 point and 16 point DIT-FFT algorithms. The experimental results show that the usage of approximate complex multipliers have reduced the power consumption and a significant reduction in the number of logic slices required to perform the 8 and 16 point DIT-FFT computation. The proposed 8 point DIT-FFT achieves an area utilization of 4.66% consuming 387.88 mW of power at a maximum throughput of 26.69 Gbps. Similarly, the proposed 16 point DIT-FFT achieves an area utilization of 8.19% consuming 417.26 mW of power at a maximum throughput of 46.20 Gbps. The 8 point DIT-FFT incorporating approximate elements achieves an area efficiency of 30%, power efficiency of 1.3% and throughput efficiency of 49.27% when compared with the conventional architectures. Similarly, the 16 point DIT-FFT incorporating approximate elements achieves an area efficiency of 33.47%, power efficiency of 1.8% and throughput efficiency of 45.10% when compared with the conventional architectures. Implementation of higher points FFT/IFFT architectures incorporating approximate complex multiplier design with self-error correction schemes are left for future work.

REFERENCES

1. E. Konguvel and M. Kannan, "A Survey on FFT/IFFT Processors for Next Generation Telecommunication Systems," *Journal of Circuits Systems and Computers*, Vol. 27. No. 3, March 2018.
2. Amjadha, A., E. Konguvel, and J. Raja. "Design of Multipath Delay Commutator Architecture based FFT Processor for 4th Generation System." *International Journal of Computer Applications (0975–8887)*, Vol. 89, No. 12, March 2014.
3. M. Kannan, E. Konguvel, J. Madhumitha, M. Mohamed Kamil Ashiq and K. Abhishek, "FPGA implementation of FFT architecture using modified Radix-4 algorithm," Asian J. Res. Soc. Sci. Humanit. 7 (2017) 47–57.
4. Manuel, Betsy Rose, E. Konguvel, and M. Kannan. "An area efficient high speed optimized FFT algorithm." *Signal Processing, Communication and Networking (ICSCN), 2017 Fourth International Conference on*. IEEE, 2017.

5. Suganthi Venkatachalam and Seok-Bum Ko, “Design of Power and Area Efficient Approximate Multipliers” in *IEEE Transactions On Very Large Scale Integration (VLSI) Systems*, Vol. 25, No. 5, May 2017.
6. C.-H. Lin and C. Lin, “High accuracy approximate multiplier with error correction,” in *Proc. IEEE 31st Int. Conf. Comput. Design*, Sep. 2013, pp. 33–38.
7. E. J. King and E. E. Swartzlander, “Data-dependent truncation scheme for parallel multipliers,” in *Proc. Conf. Rec. 31st Asilomar Conf. Signals, Syst. Comput.*, Nov. 1998, pp. 1178–1182.
8. M. J. Schulte and E. E. Swartzlander, “Truncated multiplication with correction constant,” in *Proc. 6th VLSI Signal Process.*, Oct. 1993, pp. 388–396.
9. R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, “MACACO: Modeling and analysis of circuits for approximate computing,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2011, pp. 667–673.
10. Y. Liu, T. Zhang, and K. K. Parhi, “Computation error analysis in digital signal processing systems with overscaled supply voltage,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 4, pp. 517–526, Apr. 2010.
11. P. Kulkarni, P. Gupta, and M. D. Ercegovac, “Trading accuracy for power in a multiplier architecture,” *J. Low Power Electron.*, vol. 7, no. 4, pp. 490_501, Dec. 2011.
12. A. Momeni, J. Han, P. Montuschi and F. Lombardi, “Design and analysis of Approximate compressors for multiplication,” *IEEE Trans. Comput.*, Vol. 64, No. 4, pp 984 – 994., Apr. 2015.
13. S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, “Energy-efficient approximate multiplication for digital signal processing and classification applications,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 6, pp. 1180–1184, Jun. 2015.
14. S. Balamurugan and P. S. Mallick, “Fixed-width multiplier circuits using column bypassing and decompositon logic techniques,” *Int. J. Elect. Eng. Inform.*, vol. 7, no. 4, pp. 655_664, Dec. 2015.
15. G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, “Design-efficient approximate multiplication circuits through partial product perforation,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 10, pp. 3105–3117, Oct. 2016.
16. V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, “Low Power digital signal processing using approximate adders,” *IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems*, Vol.32, no. 1, pp. 124 – 137, Jan 2013.
17. Y. Bansal and C. Madhu, ``A novel high-speed approach for 16_16 vedic multiplication with compressor adders," *Comput. Elect. Eng.*, vol. 49, pp. 39_49, Jan. 2016.
18. D. Esposito, A. G. M. Strollo, E. Napoli, D. De Caro and N. Petra, "Approximate Multipliers Based on New Approximate Compressors," in *IEEE Transactions on Circuits and Systems I: Regular Papers.* doi: 10.1109/TCSI.2018.2839266.
19. T. Yang, T. Ukezono and T. Sato, "Low-Power and High-Speed Approximate Multiplier Design with a Tree Compressor," *2017 IEEE International Conference on Computer Design (ICCD)*, Boston, MA, 2017, pp. 89-96.