Annals of Telecommunications
https://doi.org/10.1007/s12243-019-00742-6

VLSI implementation of an area and energy efficient FFT/IFFT core for MIMO-OFDM applications

Konguvel Elango[1] · Kannan Muniandi[1]

Received: 25 January 2019 / Accepted: 1 December 2019

Abstract
This research article presents an implementation of high-performance Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) core for multiple input multiple output-orthogonal frequency division multiplexing (MIMO-OFDM)-based applications. The radix-2 butterflies are implemented using arithmetic optimization technique which reduces the number of complex multipliers involved. High-performance approximate multipliers with negligible error rate are used to eliminate the power-consuming complex multipliers in the radix-2 butterflies. The FFT/IFFT prototype using the proposed high-performance butterflies are implemented using Altera Quartus EP2C35F672C6 Field Programmable Gate Array (FPGA) which yields 40% of improved logic utilization, 33% of improved timing parameters, and 14% of improved throughput rate. The proposed optimized radix-2-based FFT/IFFT core was also implemented in 45-nm CMOS technology library, using Cadence tools, which occupies an area of 143.135 mm^2 and consumes a power of 9.10 mW with a maximum throughput of 48.44 Gbps. Similarly, the high-performance approximate complex multiplier-based optimized FFT/IFFT core occupies an area of 64.811 mm^2 and consumes a power of 6.18 mW with a maximum throughput of 76.44 Gbps.

Keywords FFT · IFFT · Decimation In Time (DIT) · Approximate multipliers · MIMO-OFDM

1 Introduction

Fast Fourier Transform (FFT) and its Inverse (IFFT) play a vital role in telecommunication based applications such as optical OFDM, massive/cooperative/multi-user multi input multi output (MIMO), and MIMO in 5G. The contemporary improvements in telecommunication technologies require several features and standards that to be supported in a single integrated MIMO OFDM chip. Particularly, since the FFT (and IFFT) is one of the principal components in OFDM baseband processor, it is essential to develop a low power-area-delay fast Fourier transform processor that supports varying FFT/IFFT sizes (data points) for MIMO-OFDM based telecommunication systems.

✉ Konguvel Elango
konguart08@gmail.com

Kannan Muniandi
mkannan@annauniv.edu

[1] Department of Electronics Engineering, Madras Institute of Technology Campus, Anna University, Chennai 600004, India

A wide range of architectures and optimized algorithms for efficient implementation of FFT/IFFT computational units have been proposed in the literature over the past few decades, which can be categorized into sequential (memory-based) and pipelined. Although sequential architectures provide low-delay solution, it requires more area and consumes more power for the implementation of complex multipliers present in the basic butterfly units. Multipath delay commutator (MDC) and single-path delay feedback (SDF), which are the major pipelined FFT algorithms, provide high-throughput in spite of that it occupies more hardware resources.

The Decimation-In-Time (DIT) as well as Decimation-In-Frequency (DIF) butterfly structures, which forms the basic unit of FFT (IFFT), comprises of complex adders and complex multipliers as the key components. Real-time implementation of complex multiplication is of high complex when compared with that of complex addition. Hence, optimization in terms of complex multiplication or arithmetic flow in DIT/DIF butterfly structure or both enhance the efficiency of real-time implementation of FFT/IFFT computations.

Adder-compressors used in the radix-2 basic DIT butterfly structure to implement FFT algorithm that consumes low power is presented [1]. Partial column radix-2 and radix 2/4 butterflies are used in designing energy-efficient FFT

Published online: 17 December 2019

computational unit [2]. Radix-$2^{m\text{-bits}}$ encoding scheme was used to reduce the partial product lines in the multiplication so as to design a low power FFT processor [3]. Distributed arithmetic based approach is proposed in order to perform multiplier less FFT architecture [4]. Mixed radix butterflies (radix-2 and radix-2/4) were used to implement the FFT which has less memory access than conventional algorithm [5]. Split radix FFT architecture which has shared memory architecture with clock gating was presented in [6]. Radix-$2/2^2/3$-based partial cached FFT processor was designed for low-energy 3rd Generation Partnership Project (GPP) – Long-Term Evolution (LTE) applications [7]. A mixed pipelined/cached 128- to 1024 point FFT using power-aware twiddle factor multiplication was designed for Orthogonal Frequency Division Multiple – Access (OFDMA) [8]. To minimize the complexity in twiddle factor complex multiplication, a radix-$2^4/2^2/2^3$-based MDF FFT architecture is proposed which is optimal for IEEE 802.16e applications [9]. A 2.4 GS/s eight-data path-pipelined FFT processor was presented for Wireless Personal Area Network (WPAN) applications [10]. Mixed Generalized High Radix (GHR) FFT algorithm uses 2D and 1D FFT factorization methods to reduce the area utilization. Computational speed and hardware efficiency are improved, since this method endures eight radices and 34 different FFT lengths [11]. 1536-point FFT computation is achieved with variable length FFT processors [12] that suffers high latency and increased hardware utilization. A 16-bit 64-point sequential algorithm-based 1-dimensional (1D) FFT architecture results in an area efficient, high-speed processor suitable for Wireless Local Area Networks (WLAN) [13]. A low-power 64-point pipeline FFT processor based on radix-4^3 butterflies, achieving 25% clock rate minimization than traditional FFT architectures, targeting IEEE 802.11a/g applications, is proposed [14].

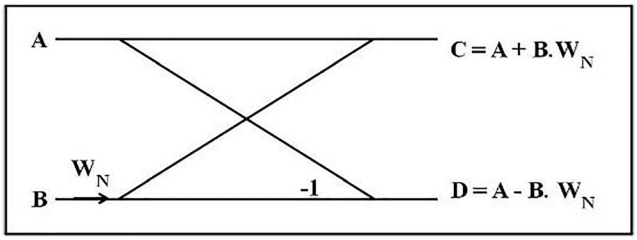

Fig. 1 Radix-2 DIT butterfly

Designing a power and energy-efficient MIMO-OFDM integrated chip is foremost concern in this advanced telecommunication technological era. Approximate arithmetic computation appears to be the effective solution for the systems that exhibits an intrinsic error tolerance. The computational errors arising because of approximation can be considered as trade-off for the significant gains in power and area.

Approximate multipliers are focused in this work rather than approximate adders since multiplication involves much complexity. In approximate complex multiplication, the approximation can be applied in the partial product accumulation [15] and also in partial product generation stages or in both [16]. Approximate 4–2 compressors are used in partial product generation of 8 × 8 Dadda multiplier ends in non-zero outputs for zero inputs which affects the mean relative error (MRE) [17]. A Karnaugh-map entry is modified in a 2 × 2 complex multiplier, which is used to build 4 × 4 and 8 × 8 complex multipliers [18]. An approximate 4:2 counter design is used in building the blocks of Wallace tree multiplier that has a maximum error % of 13.76 [19]. The usage of compressors and compressor-adders in complex multipliers reduces the power consumption and has good area efficiency as well [20]. Different sizes of approximate compressors were used in building the multiplier using an algorithm that allocates the compressors with minimum error [21]. Modified approximate

Table 1 Computational complexity for the direct DFT computation and FFT algorithm

Number of data points N	Direct computation		FFT algorithm	
	Complex multiplications N^2	Complex additions N(N-1)	Complex multiplications $(N/2)\log_2 N$	Complex additions $N\log_2 N$
8	64	56	12	24
16	256	240	32	64
32	1024	992	80	160
64	4096	4032	192	384
128	16,384	16,256	448	896
256	65,536	65,280	1024	2048
512	262,144	261,632	2304	4608
1024	1,048,576	1,047,552	5120	10,240
2048	4,194,304	4,192,256	11,264	22,528
4096	16,777,216	16,773,120	24,576	49,152
8192	67,108,864	67,100,672	53,248	106,496

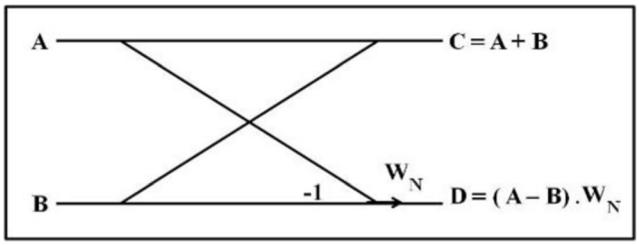

Fig. 2 Radix-2 DIF butterfly

compressors are used in order to design a low-power and high-speed multiplier with minimum error values [22].

In this paper, two major contributions are discussed. First, 8- and 16-point DIT based fast Fourier transform (FFT) algorithm based on arithmetic optimization is proposed. Second, 8- and 16-point DIT based FFT algorithm using approximate complex multiplication is proposed.

The organization of the paper is as follows. A brief introduction to FFT algorithms is given in Section 2. Arithmetic optimization-based DIT-FFT computations are discussed in Section 3. Approximate complex multiplier-based radix-2 DIT butterfly is discussed in Section 4. Comparative result analysis and discussions of the proposed FFT/IFFT processor designs are given in the Section 5 and concluding remarks and future suggestions are detailed in Section 6.

2 FFT—preliminaries

The Discrete Fourier Transform (DFT) and its Inverse (IDFT) of N-point discrete time data sequence can be given as,

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

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi kn/N}, 0 \le n \le N-1 \quad (2)$$

The DFT and IDFT of any given discrete time sequence of length N are $X(k)$ and $x(n)$ respectively. From the above Eqs. (1) and (2), it is clear that n and k are nth and kth samples of N data points, where N may vary from 8 to 8192 points which is subjected to the specific application. The exponential term given in Eqs. (1) and (2) represents the twiddle factor needed for stage-wise FFT/IFFT computations, which are the equally spaced time/frequency samples. Mathematically, twiddle factor W_N can be represented as,

$$W_N = e^{-j2\pi/N} \quad (3)$$

The direct implementation of DFT (or IDFT) requires N^2 complex multiplications and N(N-1) complex additions, where N is the length of data sequence. Divide and Conquer approach, FFT and IFFT, proposed by Cooley and Tukey [23], considering the periodicity and symmetric properties of twiddle factor W_N, reduced the number of complex multiplications to $(N/2)\log_2 N$ and number of complex additions to $N\log_2 N$. Table 1 provides the comparison of computational complexity for direct DFT (or IDFT) computation and FFT (or IFFT) algorithm for various data length N.

By Cooley and Tukey method, the length of input data sequences, N is sub-divided into r_1, r_2, r_3,… and so on, where "r" is so-called the radix of FFT. The FFT algorithm with $r = 2$, is referred as radix-2 FFT algorithm, which is the most prevalent one. But, radices ranging from 2 to 10 are also used rarely based on its requirement. In single-radix FFT algorithms, the size of input sequence N must be the power of radix value. For example, with radix-8, length of data points N in the FFT should be a power of 8. But, in mixed-radix or

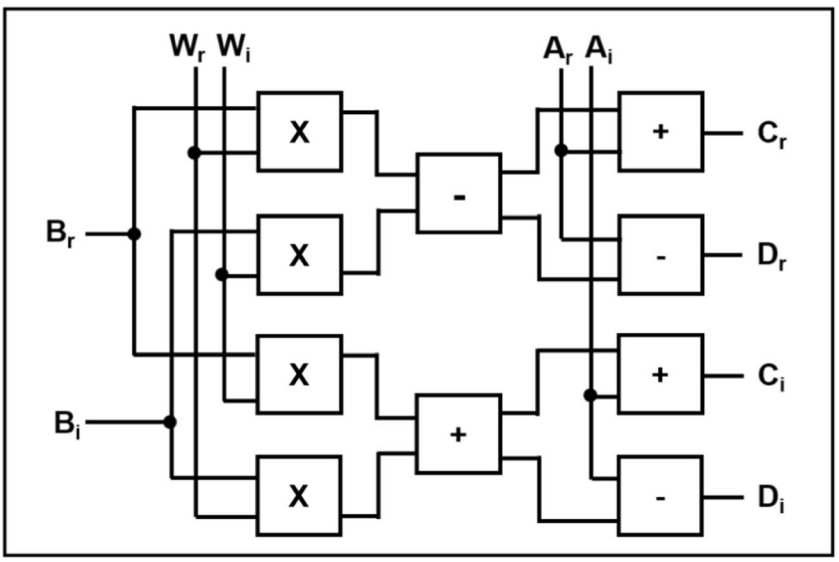

Fig. 3 Conventional radix-2 DIT butterfly structure

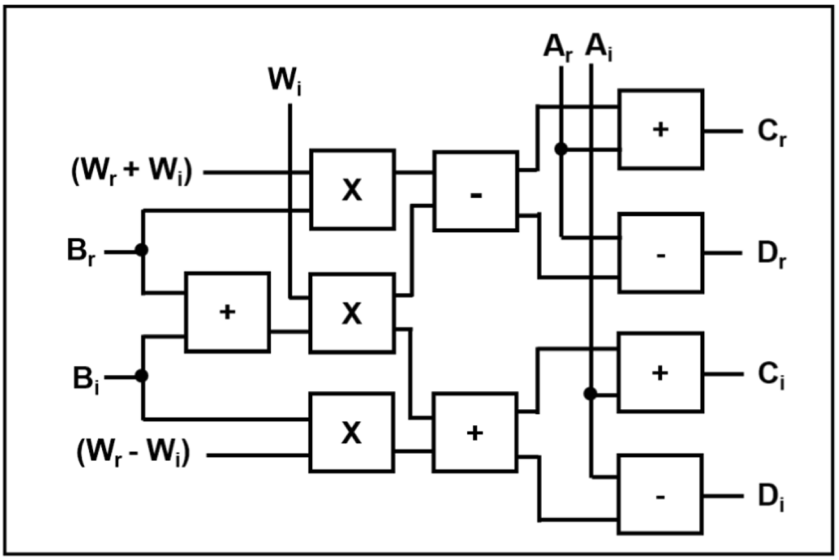

Fig. 4 Optimized radix-2 DIT FFT butterfly structure

split-radix FFT algorithm, non-prime FFT length can be decomposed into several prime factors. For example, an FFT of length 1000 can be decomposed in 6 stages using radices of 2 and 5 since $1000 = 2 \times 2 \times 2 \times 5 \times 5 \times 5$ or 4 stages using radices of 2, 5, and 10 since $1000 = 2 \times 5 \times 10 \times 10$. This kind of basic r-point computation is termed as the butterfly unit.

The decomposition of FFT length can be broadly 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. The basic radix-2 butterfly diagram for DIT (Fig. 1) and DIF (Fig. 2) computations are shown below. It can be shown that A and B denote the complex output from previous stage (or complex input to the present stage) whereas C and D denote the complex output of the present stage (or complex input to the next stage). The twiddle factor is denoted as W_N which is given in Eq. (3).

3 Arithmetic optimization for radix-2 butterfly

In FFT/IFFT computation, the butterfly structure plays a vital role since it involves a significant number of complex additions and complex multiplications. Therefore, optimizing the butterfly structure in any means in turn reduces the area utilization and power consumption of the entire FFT/IFFT computation. The arithmetic optimization in radix-2 DIT butterfly reduces the computational complexity of a single butterfly. The optimized radix-2 butterfly is used in building the 8- point and 16-point DIT FFT algorithm. From Fig. 1, radix-2 DIT butterfly algorithm can be re-drawn with all the arithmetic components involved (complex additions, complex subtractions and complex multiplications) as in Fig. 3.

The conventional radix-2 DIT butterfly consists of 4 complex multiplier and 6 complex adder/subtractor

Table 2 Computational complexity for conventional and optimized FFT algorithm

FFT length N	Number of stages $\log_2 N$	Number of butterflies per stage N2	Total number of butterflies	Number of multipliers	
				Conventional FFT	Optimized FFT
8	3	4	12	48	36
16	4	8	32	128	96
32	5	16	80	320	240
64	6	32	192	768	576
128	7	64	448	1792	1344
256	8	128	1024	4096	3072
512	9	256	2304	9216	6912
1024	10	512	5120	20,480	15,360
2048	11	1024	11,264	45,056	33,792
4096	12	2048	24,576	98,304	73,728
8192	13	4096	53,248	212,992	159,744

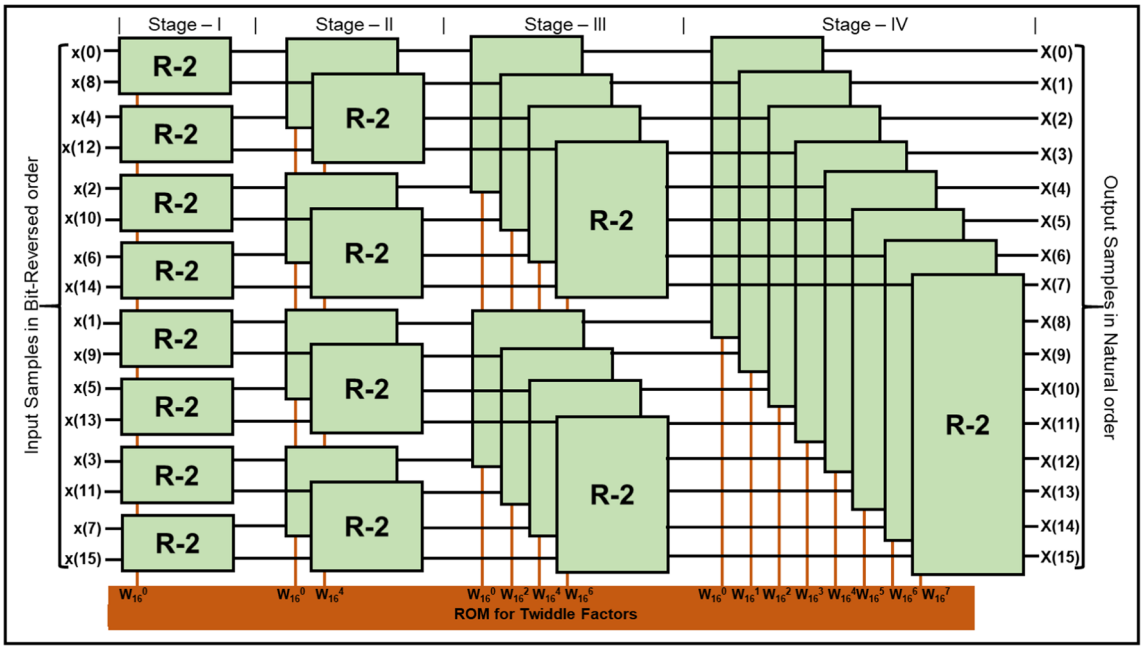

Fig. 5 16-point DIT-FFT architecture

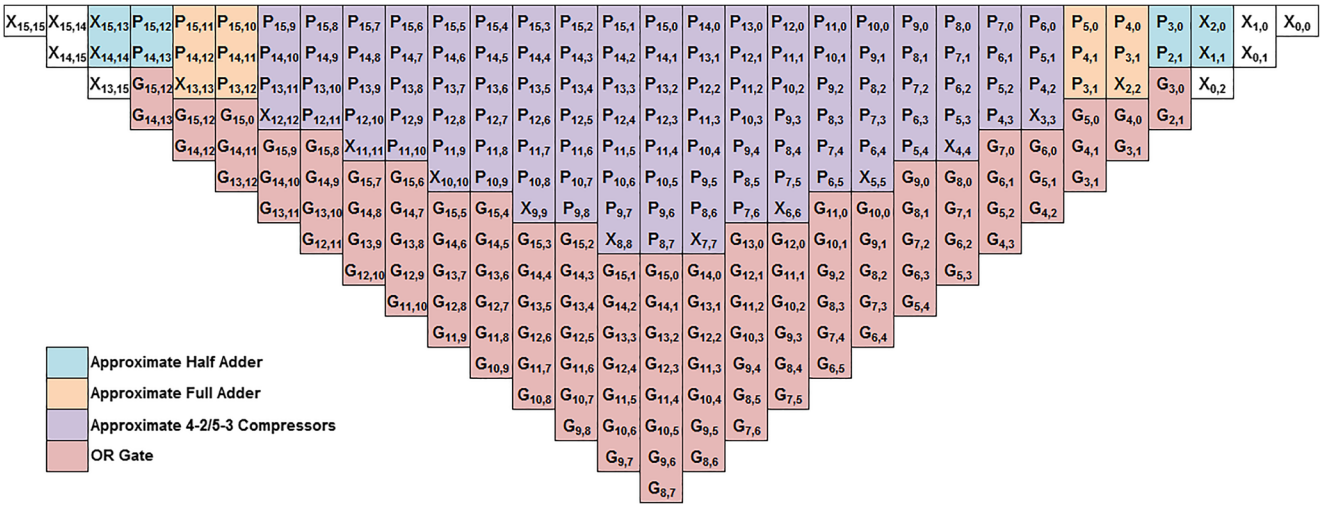

Fig. 6 Partial product reduction using approximation

Table 3 Accurate and approximated half and full adder logic table

Inputs			Accurate Half Adder		Approximate Half Adder		Accurate Full Adder		Approximate Full Adder	
A	B	C	Sum	Carry	Sum_{HA}	$Carry_{HA}$	Sum	Carry	Sum_{FA}	$Carry_{FA}$
0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0
0	1	1	0	1	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

*Altered truth table entries

modules. Mathematically, the outputs C and D can be written as,

$$(C_r + iC_i) = (A_r + iA_i) + (B_r + iB_i)\,(W_r + iW_i) \quad (4)$$

$$(D_r + iD_i) = (A_r + iA_i) - (B_r + iB_i)\,(W_r + iW_i) \quad (5)$$

On expanding the above equations with real and imaginary terms,

$$(C_r + iC_i) = A_r + B_rW_r - B_iW_i + iA_i + iB_rW_i + iB_iW_r \quad (6)$$

$$(D_r + iD_i) = A_r - B_rW_r + B_iW_i + iA_i - iB_rW_i - iB_iW_r \quad (7)$$

The real and imaginary terms can be detached, in order for FPGA or Application Specific Integrated Circuit (ASIC) implementation, Eqs. (6) and (7) become

$$C_r = A_r + B_rW_r - B_iW_i \quad (8)$$

$$C_i = A_i + B_rW_i + B_iW_r \quad (9)$$

$$D_r = A_r - B_rW_r + B_iW_i \quad (10)$$

$$D_i = A_i - B_rW_i - B_iW_r \quad (11)$$

The complex arithmetic operations involved in Eqs. (8) to (11) is shown in Fig. 3. The arithmetic optimization can be applied to the above four Eqs. (8), (9), (10) and (11) in order to

Table 4 Accurate and approximated 4–2 and 5–3 compressor-adders logic table

Inputs					Accurate 4–2 Compressor		Approximate 4–2 Compressor		Accurate 5–3 Compressor			Approx. 5–3 Com.
A	B	C	D	E	Sum	Carry	Sum_{4-2}	$Carry_{4-2}$	$Sum1_{5-3}$	$Sum2_{5-3}$	$Sum3_{5-3}$	$Sum3'_{5-3}$
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	1	0	1	0	0	0
0	0	0	1	0	1	0	1	0	1	0	0	0
0	0	0	1	1	0	1	0	1	0	1	0	0
0	0	1	0	0	1	0	1	0	1	0	0	0
0	0	1	0	1	0	1	1*	0*	0	1	0	0
0	0	1	1	0	0	1	1*	0*	0	1	0	0
0	0	1	1	1	1	1	1	1	1	1	0	0
0	1	0	0	0	1	0	1	0	1	0	0	0
0	1	0	0	1	0	1	1*	0*	0	1	0	0
0	1	0	1	0	0	1	1*	0*	0	1	0	0
0	1	0	1	1	1	1	1	1	1	1	0	0
0	1	1	0	0	0	1	0	1	0	1	0	1*
0	1	1	0	1	1	1	1	1	1	1	0	1*
0	1	1	1	0	1	1	1	1	1	1	0	1*
0	1	1	1	1	0	0	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

*Altered truth table entries

get the repeated terms with a common multiplication factor. By adding and subtracting B_rW_i to Eqs. (8) and (10) and B_iW_i to (9) and (11), the Eqs. (8) to (11) can be rewritten as,

$$C_r = A_r + B_rW_r - B_iW_i + B_rW_i - B_rW_i \tag{12}$$

$$C_i = A_i + B_rW_i + B_iW_r + B_iW_i - B_iW_i \tag{13}$$

$$D_r = A_r - B_rW_r + B_iW_i + B_rW_i - B_rW_i \tag{14}$$

$$D_i = A_i - B_rW_i - B_iW_r + B_rW_i - B_rW_i \tag{15}$$

By rearranging the repeated terms, Eqs. (12) to (15) can be rewritten as,

$$C_r = A_r + B_r(W_r + W_i) - W_i(B_r + B_i) \tag{16}$$

$$C_i = A_i + B_i(W_r - W_i) + W_i(B_r + B_i) \tag{17}$$

$$D_r = A_r - B_r(W_r + W_i) + W_i(B_r + B_i) \tag{18}$$

$$D_i = A_i - B_i(W_r - W_i) - W_i(B_r + B_i) \tag{19}$$

The term $(B_r + B_i)$ appears in all the equations from (16) to (19), which can be considered as shared adder for all the four computation. The terms $(W_r + W_i)$ and $(W_r - W_i)$ involves none of the arithmetic computation since the twiddle factor values are constant that can be stored in Read Only Memory (ROM). For various higher length FFT implementations, this twiddle factor values can be shared from ROM since it exhibits symmetricity and periodicity properties. The arithmetic computations involved in Eqs. (16) to (19) is shown in Fig. 4.

From Fig. 4, the optimized radix-2 DIT butterfly structure has 3 complex multipliers and 7 complex adder/subtractor modules. When compared with conventional radix-2 DIT method, one multiplier has been reduced at the cost of an addition of one complex adder in the optimized radix-2 DIT butterfly structure. Even though only one complex multiplier is reduced in optimized radix-2 structure, implementation of higher points FFT/IFFT will have a significant reduction in computational complexity. The computational complexity of conventional and optimized radix-2 DIT butterfly for various FFT lengths are shown in Table 2. By using the arithmetically optimized basic radix-2 butterfly modules in implementing the FFT/IFFT core, it is evident that 25% of area utilized by complex multipliers can be reduced. This optimized radix-2 butterfly is used to implement the 16 point DIT FFT which is shown in Fig. 5. The optimized radix-2 butterflies are represented as "R-2" and the twiddle factors needed for butterfly computations in each stage are stored and accessed from ROM.

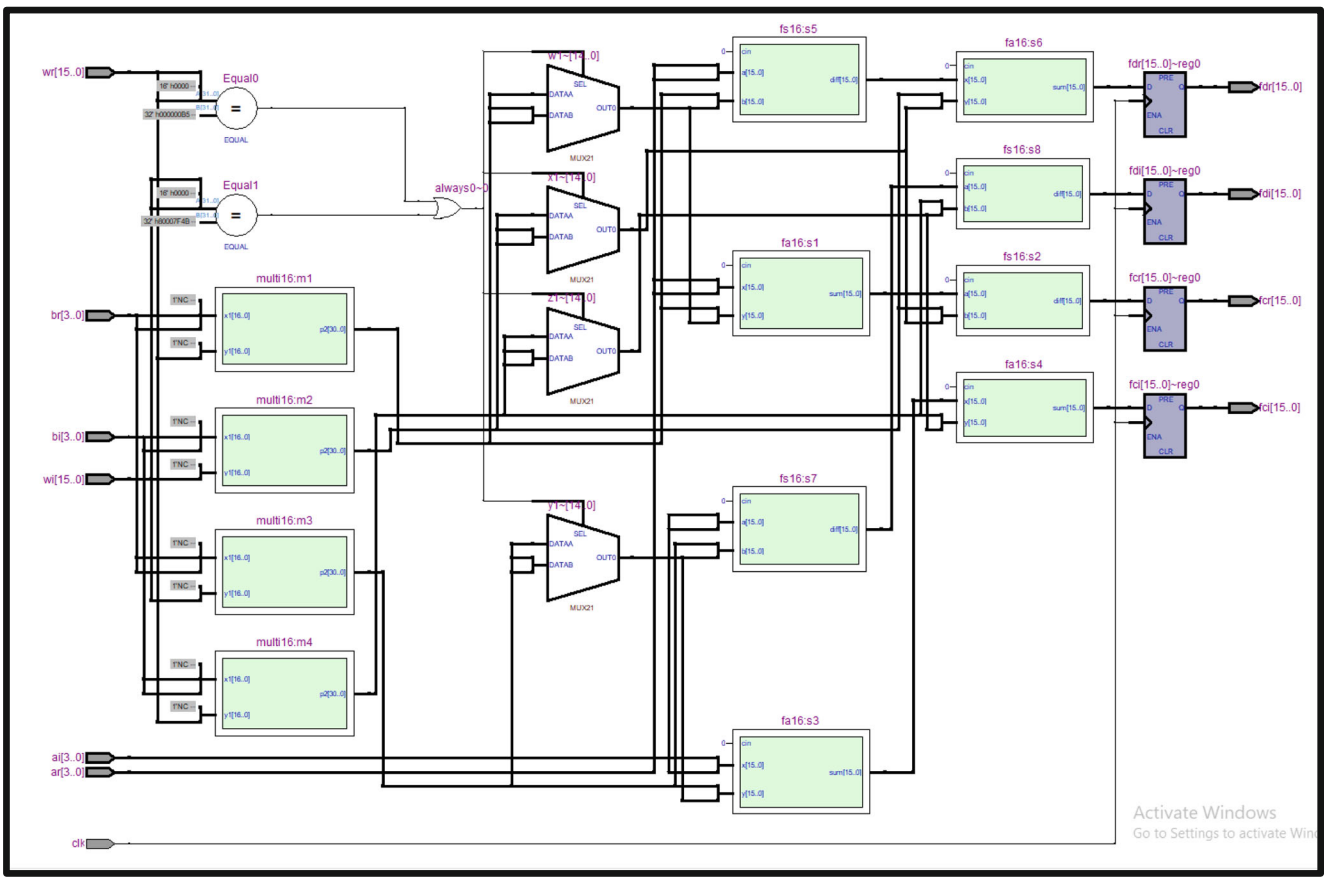

Fig. 7 RTL schematic of conventional radix-2 butterfly structure

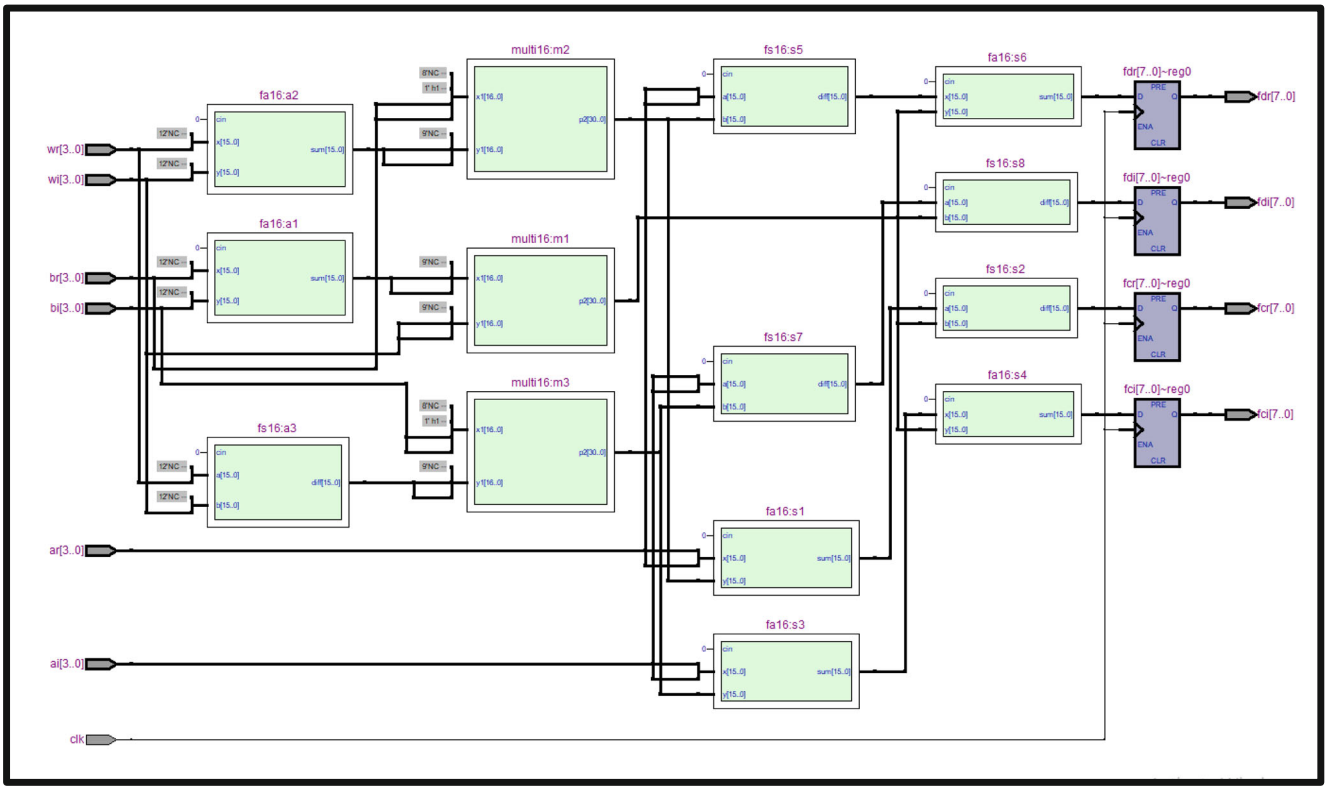

Fig. 8 RTL schematic of optimized radix-2 butterfly structure

4 Approximate complex multipliers for FFT/IFFT computation

Any real or complex arithmetic processing can be performed on an approximation. An approximate complex 16-bit multiplier is designed and it is used in building the preliminary blocks for FFT/IFFT computations. Implementation of a complex multiplication module includes three phases: generation of partial products, partial product reduction and final addition. Dynamic power consumption, path delay, and circuit complexity are dominated by the partial product reduction phase. Many techniques have been proposed to reduce the path delay in the implementation of complex multiplication, in which, usage of compressors plays a critical function. The formerly generated partial products are altered to propagate and generate signals which are shown in the following equations.

$$P_{m,n} = A_{m,n} + A_{n,m} \tag{20}$$

$$G_{m,n} = A_{m,n}.A_{n,m} \tag{21}$$

The propagate signals (Eq. 20) are approximated using approximate half adders, approximate full adders and approximate 4–2 compressor adders. The compressors are constructed using full adders or half adders in order to count the number of one's in the input. Here, 4–2 and 5–3 compressors are utilized for approximating altered partial products, since lower order compressors consume minimum area. The approximation is applied using simple OR gate for generate signals (Eq. 21). The first stage of partial product reduction using

Table 5 Computational complexity for multipliers and radix-2 butterfly FPGA implementation

Parameter	16-bit complex multiplier		Conventional butterfly	Optimized butterfly	Conventional butterfly	Optimized butterfly
Multiplier type	Conventional accurate	Proposed approximate	Conventional accurate	Conventional accurate	Proposed approximate	Proposed approximate
Logic slices	597	424	818	221	673	162
Registers	–	–	64	32	64	32
Utilization rate	1.80%	1.28%	2.46%	0.67%	2.03%	0.49%
Delay (ns)	67.07	23.28	28.22	14.38	26.75	16.25
Power (mW)	10.88	8.57	32.95	10.53	28.39	9.90

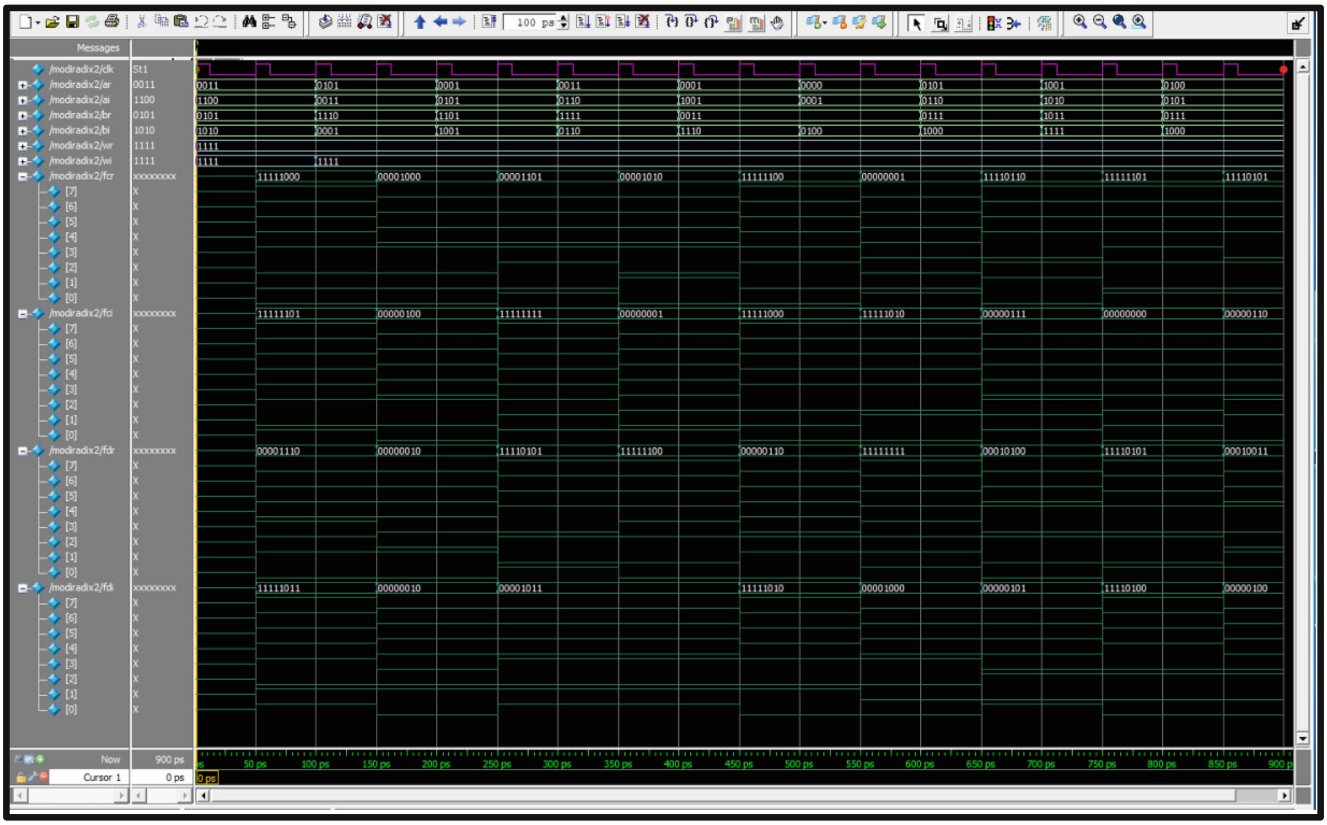

Fig. 9 Simulation waveforms for optimized radix-2 butterfly

approximation technique for 16-bit multiplier is shown in Fig. 6. The second stage of reduction uses twelve approximate full adder, four 4–2 approximate compressor, and eleven 5–3 approximate compressor circuits.

In an accurate half adder, XOR gate is used to calculate Sum_{HA}. XOR gate of the accurate half adder is replaced with OR gate in approximation, since area utilization of the XOR gate is higher. To calculate the Sum_{FA} and $Carry_{FA}$ of full adder, 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 because of this replacement in approximate half adder and full adder circuit operations are shown in Table 3 as * (asterisk symbol) and illustrated mathematically through the following equations,

$$Sum_{HA} = B + C \tag{22}$$

$$Carry_{HA} = B \cdot C \tag{23}$$

$$Sum_{FA} = (A + B)\ xor\ C \tag{25}$$

$$Carry_{FA} = (A + B) \cdot C \tag{26}$$

Table 6 Computational complexity for 8-point FFT structure FPGA implementation

Parameter Multiplier type	Conventional Accurate 16-Bit	Optimized	Conventional Approximate 16-Bit	Optimized
Logic slices	2207	1864	1622	1565
Registers	496	496	488	456
Utilization rate	6.64%	5.61%	4.88%	4.71%
Delay (ns)	11.58	8.54	10.29	8.42
Power (mW)	76.44	70.38	73.02	70.17
Frequency (MHz)	49.37	53.37	70.18	75.97
Throughput (Gbps)	12.64	13.66	17.97	19.45

Fig. 7 Computational complexity for 16-point FFT structure FPGA implementation

Parameter Multiplier type	Conventional Accurate 16-bit	Optimized	Conventional Approximate 16-bit	Optimized
Logic slices	6306	2751	4089	2495
Registers	962	618	718	572
Utilization rate	18.98%	8.28%	23.31%	7.51%
Delay (ns)	12.72	8.62	11.34	7.92
Power (mW)	168.49	106.65	118.78	87.32
Frequency (MHz)	43.50	47.30	64.21	74.65
Throughput (Gbps)	44.54	48.44	65.75	76.44

Compressors and compressor-adders are the primary building blocks of the proposed complex multipliers to accumulate the generated partial products. Compressor-adders are used in the second stage of complex multiplication to reduce the number of partial products so as to reduce the gate count and critical path delay. The utilization of approximate compressors in the least significant bits decreases power consumption and circuit area because of less switching rate than most significant bits. In 5–3 compressor-adder circuit, 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 Sum_{4-2} and $Carry_{4-2}$ represent the sum and carry outputs for the 4–2 compressor module which are expressed in Eqs. (27)–(28). Similarly, $Sum1_{5-3}$, $Sum2_{5-3}$, and $Sum3_{5-3}$ represent the accurate outputs for the 5–3 compressor-adder circuit expressed in the Eqs. (29)–(31). $Sum3'_{5-3}$ represents the output for the approximate 5–3 compressor-adder circuit shown in the Eq. (32). The outputs $Sum1_{5-3}$ and $Sum2_{5-3}$ will remain same for the accurate as well as approximate 5–3 compressor-adder computations. But the output $Sum3_{5-3}$ in the accurate 5–3 compressor-adder computation is replaced by $Sum3'_{5-3}$ in the approximate 5–3 compressor-adder computation. The logic differentiation between accurate and approximate, 4–2 and 5–3 compressor-adders, is shown as * (asterisk symbol) in Table 4.

$$Sum_{4-2} = (B\ xor\ C) + (D\ xor\ E) + (B \cdot C) \cdot (D \cdot E) \quad (27)$$

$$Carry_{4-2} = (B \cdot C) + (D \cdot E) \quad (28)$$

$$Sum1_{5-3} = A\ xor\ B\ xor\ C\ xor\ D\ xor\ E \quad (29)$$

$$Sum2_{5-3} = C\ xor\ D \quad (30)$$

$$Sum3_{5-3} = A \cdot (\sim(A\ xor\ B)) + B \cdot (A\ xor\ B) \cdot (C \cdot (\sim(A\ xor\ B\ xor\ C\ xor\ D\ ))) + D \cdot (A\ xor\ B\ xor\ C\ xor\ D) \quad (31)$$

$$Sum3'_{5-3} = C \cdot D \quad (32)$$

5 Implementation results and discussions

The proposed 8- and 16-point FFT/IFFT algorithms are implemented using conventional as well as optimized radix-2 FFT algorithm. The designed 16-bit approximate complex multiplier is used in both the conventional radix-2 algorithm (Fig. 3) and optimized radix-2 algorithm (Fig. 4). The performance measures of conventional and optimized FFT algorithms are also compared with that of FFT algorithm (conventional and optimized) designed using approximate 16-bit complex multiplier.

The conventional FFT algorithm, arithmetically optimized FFT algorithm and FFT algorithm (conventional and optimized) using approximate multipliers, for 8 and 16 points are designed, synthesized, simulated, and tested in 45-nm technology library using Cadence® NCLaunch, SimVision and Encounter RTL Compiler. The proposed FFT/IFFT designs are also analyzed, synthesized, and simulated in Altera® Quartus-II EP2C35F672C6 FPGA device using Verilog Hardware Description Language (HDL). The experimental results obtained and the analysis made using Cadence and Altera simulators are discussed in detail as follows.

The Register Transfer Level (RTL) description of the conventional radix-2 butterfly structure and arithmetically optimized radix-2 butterfly structure are presented in Figs. 7 and 8 which shows the adder and complex multiplier modules present. The clock enabled registers are used at each output port to throw the output bits to the successive modules.

The synthesis and compilation summary of the conventional and optimized radix-2 structures with accurate and approximate complex multipliers are presented in Table 5. The designed radix-2 butterflies using accurate and approximate multipliers are implemented in EP2C35F672C6 device which has a total of 33,216 combinational functions. The utilization rate for conventional and optimized radix-2 butterflies using accurate complex multiplier are 2.46% and 0.67% respectively with an improvement factor of 3.67%. Similarly, the utilization rates for conventional and optimized radix-2 butterflies using approximate complex multiplier are 2.03% and 0.49% with an improvement factor of 4.14%. It is evident from the

Table 8 ASIC implementation of different 8- and 16-point FFT architecture

FFT Points	Method	Multiplier type	Cells	Area (μm^2)	Power (mW)	Delay (ps)
8-point	Conventional	Accurate	16,232	94,687	4.97	406
	Optimized	16-bit	16,068	93,255	3.71	381
	Conventional	Approximate	3486	36,038	4.18	398
	Optimized	16-bit	2542	28,010	3.61	370
16-point	Conventional	Accurate	30,085	161,401	13.91	407
	Optimized	16-bit	24,488	143,135	9.10	364
	Conventional	Approximate	11,970	120,847	8.19	342
	Optimized	16-bit	7338	64,811	6.18	308

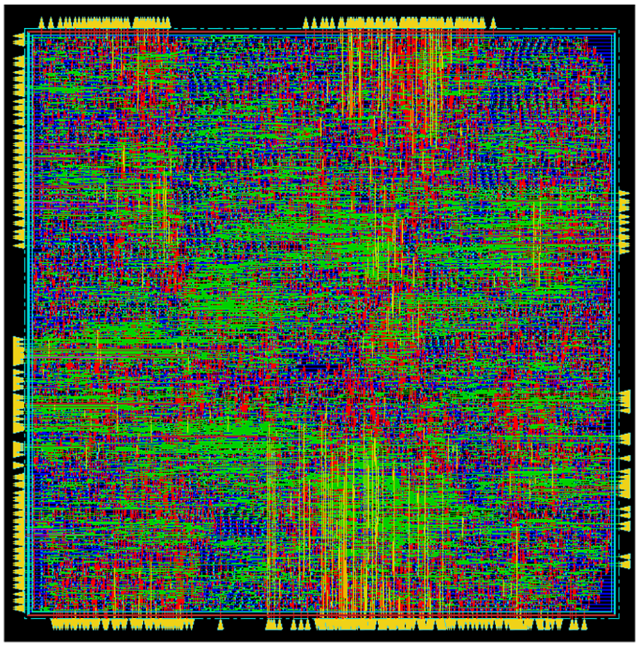

Fig. 10 Layout of the FFT core

table that parameters pertaining to delay and power are improved at least by a factor of 1.65% and 1.48% respectively for optimized radix-2 butterfly structure using approximate multiplier. The simulation waveforms of the proposed optimized radix-2 butterfly (for first stage), which has the input–output signals, clock signal and twiddle factor inputs, obtained from Altera ModelSim tool are shown in Fig. 9.

The synthesis and compilation summary of the conventional and optimized 8-point FFT architecture using 16-bit accurate and approximate complex multipliers are presented in Table 6. It is evident from the table that FFT algorithm using arithmetically optimized radix-2 butterfly have improved performance than with the conventional radix-2 butterfly in terms of number of logic slices, delay, power consumption and operating frequency. The optimized butterfly improves the FPGA utilization rate by 1.18%. The time delay for conventional FFT and optimized FFT are 11.58 ns and 8.54 ns respectively. Similarly, dynamic power consumption in FPGA for the conventional and proposed optimized FFT algorithm are 76.44 mW and 0.38 mW respectively. The conventional and optimized FFT algorithm using the proposed 16-bit complex approximate multiplier also have improved performance metrics when compared to that of using accurate multipliers.

The synthesis and compilation summary of the conventional and optimized 16-point FFT architecture using 16-bit accurate and approximate multipliers are presented in Table 7. It is apparent from the table that 16-point with optimized radix-2 butterfly have improved performances than 8-point, as discussed in the Section 3. The throughput for the designed module in the FPGA device is calculated based on the frequency of operation, number of output paths, and number of output bits per output path. The error calculation of the approximate computations is done using the Matlab software and found to be lesser than 2% when compared with the accurate computations.

ASIC implementation results for the conventional and optimized 8-point FFT architecture using 16-bit accurate and approximate complex multipliers in 45-nm technology library

Table 9 Comparison among different FFT processors

FFT Processor	Architecture	Process (nm)	FFT size	Frequency (MHz)	Normalized area (mm^2)	Normalized energy (nJ)
S.Yeng [7]	MDF	180	128~2048	35	0.94	1.28
C.M.Chen [8]	Cached FFT	180	128~1024	51	2.05	2.06
M.S.Patil [9]	SDF	180	128~2048	40	2.21	1.39
S.N.Tang [10]	Multi-Data Scaling	90	2048	300	2.265	0.83
J.Chen [11]	GHR	180	128~2048	122.88	2.44	4.12
C.Yu [12]	SDF	90	128~2048	40	2.813	0.313
K.J.Yang [29]	MDC	90	128~2048	40	6.05	5.16
T.Cho [30]	MR-Pipelined	90	512	310	6.093	1.31
S.J.Huang [31]	Cascaded Pipeline/Parallel	90	512	324	7.265	0.82
J.Raja [13]	Cached FFT	180	64	251	9.98	53.85
S.J.Huang [32]	Memory Base	90	512	324	19.21	14.41
T.Ahamed [33]	Feed-Forward	65	512	330	21.41	1.84
S.Kala [14]	R4-SDC	130	64	5	24.8	2.11
Proposed (accurate multi.)	Optimized radix-2	45	8~16	47.30	0.6	0.38
Proposed (approx. multi.)	Optimized radix-2	45	8~16	74.65	1.43	0.67

using standard cells with a supply voltage (V_{DD}) of 1.02 V are presented in Table 8. The backend physical design up to the layout of the proposed FFT/IFFT core with nine metal layers and one poly process is shown in Fig. 10.

The proposed 8-point and 16-point, conventional and optimized, with 16-bit approximate multipliers have improved performance in terms of area, power, and delay which are the essential parameters to be considered for chip design. A factor of 2.5 to 3.5% of area utilization improvement is achieved with the proposed approximate multiplier design for the 8- and 16-point FFT computations. About 1.2 to 1.6% improvement in power utilization and 1.1 to 1.3% improvement in delay calculation are also achieved with the proposed 8- and 16-points FFT computations using the approximate complex multipliers.

Normalized area, energy, and throughput are the significant parameters to evaluate the performance of semi-customed ASIC implementation of the designed various FFT/IFFT architectures and comparison with the existing architectures. Several methods [22, 24, 25] have been proposed in the literature to measure the normalized area and energy for FFT/IFFT processors, but these evaluations were carried out for the same number of FFT/IFFT data points (N). For different FFT/IFFT sizes (N), different simulation parameters and technology library used, the evaluation proposed in [7, 26–28], is optimal for the comparison, that can be given as,

$$Area_{Normalized} = \frac{Area}{N \times \left(\frac{Tech}{0.18}\right)^2} \tag{33}$$

$$Energy_{Normalized} = \frac{Power \times Exec.Time}{N \times \left(\frac{V_{DD}}{1.8}\right)^2} \tag{34}$$

Table 9 compares the operation frequency, normalized area and normalized energy of the different existing FFT/IFFT architectures with the proposed architectures. Even though different FFT sizes are considered for comparison, the area and power parameter are normalized by using the Eqs. (33) and (34). It is obvious from the table that the normalized area and energy are very lesser when comparing with the other architectures. This is because the proposed optimized radix-2 butterfly structure reduces the multiplier count when compared with the other existing architectures. When approximation is used in the complex multiplier, it can also be claimed that the area utilization and power consumption is also reduced.

6 Conclusion

In this paper, we have presented the design of 8- and 16-point FFT/IFFT core using arithmetically optimized radix-2 butterfly units and complex approximate multipliers. The computational radix-2 butterflies are area efficient, low power, and high speed which accomplishes the implementation of high performance 8- and 16-point FFT/IFFT processing core. The number of complex multipliers in a single radix-2 butterfly is reduced to three units using arithmetic optimization technique. The accurate complex multipliers in conventional as well as optimized radix-2 butterflies are replaced with area efficient approximate complex multipliers. The proposed optimized radix-2 butterfly based 16-point FFT core occupies an area of 143.135 mm^2, consumes a power of 9.10 mW with a maximum throughput of 48.44 Gbps. Similarly, the approximate radix-2 butterfly based optimized 16-point FFT core occupies an area of 64.811 mm^2, consumes a power of 6.18 mW with a maximum throughput of 76.44 Gbps. Therefore, we conclude that the proposed FFT/IFFT designs ensure a good trade-off among area utilization, power consumption, and operating throughput which is more appropriate for telecommunication based applications such as optical OFDM, massive/cooperative/multi-user MIMO and MIMO in 5G. As a future suggestion, one could implement the proposed optimized radix-2 butterfly units for higher points FFT/IFFT computations which would be expected to have improved performance than the existing architectures.

References

1. Fonseca M, Costa E, Martins J (2011) Implementation of pipelined butterflies from Radix-2 FFT with decimation in time algorithm using efficient adder compressors. Proceedings of 2nd IEEE Latin American Symposium on Circuits and Systems (LASCAS)
2. Takala T, Punkka K (2005) Butterfly unit supporting Radix-4 and Radix-2 FFT. Proceedings of the 2005 International TICSP Workshop on Spectral Methods and Multirate Signal Processing, SMMSP 2005 30:47–54
3. Costa E, Monteiro J, Bampi S (2003) Gray encoded arithmetic operators applied to FFT and FIR dedicated datapaths. In: 12th International Conference on Very Large Scale Integration (VLSI-SoC), pp 307–312
4. Laguri N, Anusudha K (2014) VLSI implementation of efficient split radix FFT based on distributed arithmetic. In: IEEE Intonference on Green Computing Communication and Electrical Engineering (ICGCCEE), pp 1–5
5. Lin J, Chung H (2013) The split-radix fast Fourier transforms with radix-4 butterfly units. In: IEEE Signal and Information Processing Association Annual Summit and Conference (APSIPA), pp 1–5
6. Qian Z, Nasiri N, Segal O, Margala M (2014) FPGA implementation of low-power split-radix FFT processors. In: 24th IEEE International Conference on Field Programmable Logic and Applications (FPL), pp 1–2
7. Sheng-Yeng K-T, Chao-Ming, Yuan-Hao (2010) Energy-efficient 128~2048/1536-point FFT processor with resource block mapping for 3GPP-LTE system. In: The 2010 International Conference on Green Circuits and Systems, Shanghai, pp 14–17
8. C. Chen, C. Hung and Y. Huang, An energy-efficient partial FFT processor for the OFDMA communication system in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 57, no. 2, pp. 136–140, 2010

9. Patil MS, Chhatbar TD, Darji AD (2010) An area efficient and low power implementation of 2048 point FFT/IFFT processor for mobile WiMAX. In: 2010 International Conference on Signal Processing and Communications (SPCOM), Bangalore, pp 1–4
10. S. Tang, J. Tsai and T. Chang, "A 2.4-GS/s FFT processor for OFDM-based WPAN applications," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 57, no. 6, pp. 451–455, 2010
11. J. Chen, J. Hu, S. Lee and G. E. Sobelman, "Hardware efficient mixed Radix-25/16/9 FFT for LTE systems," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 2, pp. 221–229, 2015
12. C. Yu and M. Yen, "Area-efficient 128- to 2048/1536-point pipeline FFT processor for LTE and Mobile WiMAX systems," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 9, pp. 1793–1800, 2015
13. Raja J, Mangaiyarkarasi P, Moorthi K (2015) Area efficient low power high performance cached FFT processor for MIMO OFDM application. Int J Appl Eng Res 10:11853–11868
14. Kala S, Nalesh S, Nandy SK, Narayan R (2013) Design of a low power 64 point FFT architecture for WLAN applications. In: 2013 25th International Conference on Microelectronics (ICM), Beirut, pp 1–4
15. Liu W, Qian L, Wang C, Jiang H, Han J, Lombardi F (Aug. 2017) Design of approximate radix-4 booth multipliers for error-tolerant computing. IEEE Trans Comput 66(8):1435–1441
16. Momeni A, Han J, Montuschi P, Lombardi F (Apr. 2015) Design and analysis of approximate compressors for multiplication. IEEE Trans Comput 64(4):984–994
17. Kulkarni P, Gupta P, Ercegovac MD (2011) Trading accuracy for power in a multiplier architecture. J Low Power Electron 7(4):490–501
18. Lin C-H, Lin C (2013) High accuracy approximate multiplier with error correction. In: Proc. IEEE 31st Int. Conf. Comput. Design, pp 33–38
19. Bansal Y, Madhu C (2016) A novel high-speed approach for 16_16 vedic multiplication with compressor adders. Comput. Elect. Eng 49:39–49
20. Esposito D, Strollo AGM, Napoli E, De Caro D, Petra N Approximate multipliers based on new approximate compressors. In: IEEE Transactions on Circuits and Systems I: Regular Papers. https://doi.org/10.1109/TCSI.2018.2839266
21. Yang T, Ukezono T, Sato T (2017) Low-Power and High-Speed Approximate Multiplier Design with a Tree Compressor. In: 2017 IEEE international conference on computer design (ICCD), Boston, MA, pp 89–96
22. Lin YT, Tsai PY, Chiueh TD (2005) Low-power variable-length fast fourier transform processor. IEEE Proc Comput Digit Tech 152: 499–506
23. Cooley J, Tukey J (1965) An algorithm for the machine calculation of the complex fourier series. Mathematical Computation 19:297–301
24. Chiueh TD, Tsai PY (2007) OFDM baseband receiver design for wireless communications. Wiley, New York
25. Chen CM, Hung CC, Huang YH (2010) An energy-efficient partial FFT processor for the OFDMA communication system. IEEE Trans Circuits Syst II Exp Briefs 57:136–140
26. Konguvel E, Kannan M (2018) A survey on FFT/IFFT processors for next generation telecommunication systems. Journal of Circuits, Systems and Computers 27(03):1830001
27. Konguvel E, Kannan M (2019) Hardware implementation of FFT/IFFT algorithms incorporating efficient computational elements. Journal of Electrical Engineering & Technology, Springer, 2093–7423 04(4):1717–1721
28. Manuel BR, Konguvel E, Kannan M (2017) An Area Efficient High Speed Optimized FFT algorithm. In: Proc. of 2017 4thIEEE International Conference on Signal Processing, Communications and Networking (ICSCN'17), Chennai, pp 1–5, 16–18 March
29. K. Yang, S. Tsai and G. C. H. Chuang, "MDC FFT/IFFT processor with variable length for MIMO-OFDM systems," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 4, pp. 720–731, 2013
30. Cho T, Lee H (2013) A high-speed low-complexity modified radix-25 FFT processor for high rate WPAN applications. IEEE Trans Very Large Scale Integr 21:187–191
31. S. Huang and S. Chen, "A high-throughput Radix-16 FFT processor with parallel and normal input/output ordering for IEEE 802.15.3c systems," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 59, no. 8, pp. 1752–1765, 2012
32. Huang S, Chen S (2010) A green FFT processor with 2.5-GS/s for IEEE 802.15.3c (WPANs). In: The 2010 International Conference on Green Circuits and Systems, Shanghai, pp 9–13
33. Ahmed T, Garrido M, Gustafsson O (2011) A 512-point 8-parallel pipelined feedforward FFT for WPAN. In: 2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR), Pacific Grove, CA, pp 981–984