A High-Throughput Radix-16 FFT Processor With Parallel and Normal Input/Output Ordering for IEEE 802.15.3c Systems

Shen-Jui Huang, *Student Member, IEEE*, and Sau-Gee Chen, *Member, IEEE*

***Abstract*—This paper presents a high-throughput FFT processor for IEEE 802.15.3c (WPANs) standard. To meet the throughput requirement of 2.59 Giga-samples/s, radix-16 FFT algorithm is adopted and reformulated to an efficient form so that the required number of butterfly stages is reduced. Specifically, the radix-16 butterfly processing element consists of two cascaded parallel/pipelined radix-4 butterfly units. It facilitates low-complexity realization of radix-16 butterfly operation and high operation speed due to its optimized pipelined structure. Besides, a new three-stage multiplier for twiddle factor multiplication is also proposed, which has lower area and power consumption than conventional complex multipliers. Moreover, a conflict-free multibank memory addressing scheme is devised to support up to 16-way parallel and normal-order data input/output. Without needing to reorder the input/output data, this scheme helps a high-throughput design result. Equipped with those new performance-boosting techniques, overall the proposed radix-16 FFT processor is area-efficient with high data processing rate and hardware utilization efficiency. The EDA synthesis results show that whole FFT processor area is $0.93\ \mathrm{mm}^2$, and the power consumption is 42 mW with 90 nm process. The SQNR performance is 57 dB with 12-bit wordlength implementation.**

***Index Terms*—Fast Fourier transform (FFT), non-conflict memory addressing scheme, OFDM, radix-16 FFT, WPANs.**

I. Introduction

Since the recent decade, the increasing demand for real-time and high-rate multimedia services has been pushing the birth of high-rate wireless communication systems. Ultra wideband (UWB) communication system, for example, can deliver data rates up to 480 Mb/s at a short distance range of $2\ \mathrm{m} \sim 10\ \mathrm{m}$. However, it is not enough to support high data rate applications of more than 1 Gbps such as high-definition (HD) streaming content downloads, HD video on demand, home theater, and etc. To meet the application demands, IEEE 802.15.3c-2009 standard for high-rate Wireless Personal Area Networks (WPANs) [1] was ratified recently. In the standard, there are three PHY modes, i.e., Single Carrier mode (SC PHY), High Speed Interface mode (HSI PHY), and Audio/Visual mode (AV PHY). Except SC PHY, both HSI PHY and AV PHY are based on OFDM modulations. As is well known, Fast Fourier transform (FFT) operation is one of the key operations for OFDM-based communication systems. Besides, for SC PHY mode, FFT operations are widely employed for effective channel equalization.

Designs of efficient FFT architectures have been actively investigated since last decades. Generally, FFT architectures can be divided into two different categories: pipelined structures (including the single-path delay feedback (SDF) architectures [2], [3], the multi-path delay commutator (MDC) [3], [4], and multi-path delay feedback (MDF) architectures [12], [18], [23]), and memory-based architectures [5]–[8], including cached memory architecture [9]. Pipelined architectures have the advantage of high throughput, but demand high area cost especially for long-length FFTs. Memory-based FFT architectures usually contain one butterfly processing element (PE), memory banks and control logics. Though they have low area costs, their throughputs are often limited by the available number of PEs and memory access bandwidth. Since 802.15.3c standard provides extremely high data rates up to 2.592 Giga-samples/s for HSI PHY, one needs to finish 512-point FFT operations within 222.2 ns. FFT processors with such high throughput are rarely seen in the literature, due to the stringent specification. The FFT processors in [10], [11] are targeted either for WLAN or DVB-T applications with throughput around tens of Mega-samples/s, while the designs in [12], [13], [18] are targeted for UWB application of throughput 409.6 Mega-samples/s. There are gigabit FFT processors [13], [19], [23] for WPAN applications. Although the work in [13] can support 802.15.3c application, its 2048-point FFT architectre is not specifically tailored for 512-point FFT operation needed by 802.15.3c systems. The work in [23], similar to [13], is also a MDF pipelined architecture for WPAN applications. Though it has high throughput, its outputs are not in normal order which requires extra re-ordering buffer for its following stage's operation, particularly for frequency-domain channel equalization operations generally conducted in OFDM systems, like the one [21] our proposed FFT processor is integrated into. The memory-based FFT processor [19] is specifically designed for 802.15.3c applications. Despite the fact that it provides normal-order output, it contains too many pipelined PEs which are not area and power efficent enough.

Since generally a radix-r FFT algorithm for N-point FFT requires $(N/r)\log_r N$ FFT butterfly operations, higher-radix FFT algorithms can complete FFT operations with smaller numbers of FFT stages than the lower-radix algorithms.

Manuscript received March 21, 2011; revised August 23, 2011; accepted November 21, 2011. Date of publication January 10, 2012; date of current version July 24, 2012. This work is supported in part by National Science Council, Taiwan under Grants NSC 99-2220-E-009-016 and NSC 99-2219-E-009-00. This paper was recommended by Associate Editor A. Strollo.

The authors are with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: shenray.ee95g@g2.nctu.edu.tw; sgchen@mail.nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2011.2180430

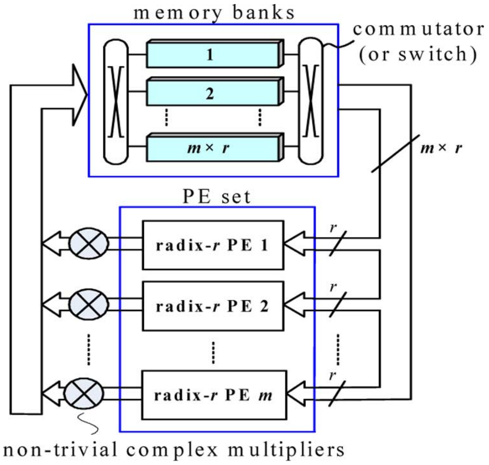


Fig. 1. Architecture of a general memory-based FFT processor.

TABLE I
THE HARDWARE RESOURCE REQUIREMENTS OF MEMORY-BASED FFT PROCESSOR IN REALIZING 512-POINT FFT OF 802.15.3C, VERSUS FFT RADICES

radix (r)	No. of BO_r				Min. m	Min. no. of complex multiplier	
	BO_2	BO_4	BO_8	BO_{16}		non-trivial	trivial
4	256	128×4			16	48	0
8	0	0	64×3		4	28	12
16	256	0	0	32×2	2	30	18

†BO_r : radix-r Butterfly Operation

Another benefit of higher-radix FFT operations over the lower-radix ones is that higher memory access bandwidth can be more conveniently provided in hardware implementations, as the bandwidth is proportional to r. An improved radix-16 algorithm is proposed in [14] which can reduce the numbers of twiddle factor operations and lookup-table accesses. A radix-16 algorithm suitable for multiply-and-add instruction is proposed in [15]. However, those algorithms are mainly devised for the execution of general-purpose processors.

Owing to the mentioned advantages of a high-radix memory-based FFT design, this work will design a high-performance radix-16 FFT processor which satisfies the mentioned throughput requirement of 802.15.3c applications. However, since generally a radix-16 butterfly unit is more complicated and less flexible than lower-radix ones, this work reformulates conventional radix-16 FFT algorithm so as to facilitate efficient and optimized pipelined realization of a radix-16 PE with high computing power and speed. Further, several performance-enhancement techniques are applied to the whole FFT processor design, including an efficient multiplier structure for twiddle factor multiplication, schemes of conflict-free memory access and normal-order FFT output generations.

The rest of this article is organized as follows. In Section II, design concerns for 802.15.3c FFT processor are examined. In Section III, a reformulated radix-16 algorithm and its butterfly structure are analyzed. In Section IV, a high-throughput and high-speed FFT processor architecture is introduced. Also, a three-stage multiplier for twiddle factor multiplication is presented. In Section V, a new conflict-free memory addressing scheme is presented. In Section VI, block-floating point (BFP) operations are designed and implemented to achieve high signal-to-quantization-noise ration (SQNR). The issue of continuous-flow FFT operation is also addressed. Implementation results are discussed in Section VII, and a conclusion is made in Section VIII.

II. DESIGN CONSIDERATION FOR THE 802.15.3C FFT PROCESSOR

In designing an FFT processor, one can first determine the required specifications of the target FFT processor as a function of FFT radix r, and then decide the most suitable radix. Consider a general radix-r memory-based FFT architecture as shown in Fig. 1, which consists of m parallel radix-r PEs and $m \times r$ memory banks (for simultaneous data access), in order to provide high throughput operations. It is also assumed that m parallel FFT butterfly operations can be finished in one clock cycle.

Based on these conditions, one can get the following simple design constraint for finishing the 512-point FFT operation within one OFDM symbol period T

$$\left(\frac{512}{r}\right) \times \frac{\lceil \log_r 512 \rceil}{m} \leq (R \times T) \tag{1}$$

where R is the operating clock rate of the target FFT processor, and T is equal to 222.2 ns (including 24.69 ns guard interval), as specified in IEEE 802.15.3c standard [1]. The term $(512/r)$ represents the number of radix-r butterflies per FFT stage, and the term $\lceil \log_r 512 \rceil$ is the number of FFT stages, where $\lceil x \rceil$ represents the ceiling function of x. The left-hand side of the inequality corresponds to the number of radix-r butterfly operations per PE needs to compute per OFDM symbol, while the right-hand side represents the available number of clock cycles within an OFDM symbol time. The constraint provides a design guideline for the target FFT processor. One can increase m value to reduce clock rate, at the cost of increased area, and vice versa. It is a trade-off among speed, area, power and implementation feasibility. In the proposed design, the operating clock rate for hardware realization is assumed one-eighth the sample rate, i.e., 2.592 GHz $\times\ 1/8 = 324$ MHz and $m = 8$. As a result, the available number of clock cycles is 72.

Based on the design constraint (1), Table I lists the required numbers of FFT stages, butterfly operations (BO), minimum number m of PEs and complex multipliers, versus FFT radix. There are two kinds of complex multipliers involved in the computation: non-trivial and trivial complex multipliers. The non-trivial multipliers are used for twiddle factor multiplications between FFT stages, while the trivial ones are incurred when large r is used, such as the multiplications with W_8^i in radix-8 FFT algorithms, where i is an integer, and $W_8^i = e^{-j2\pi i/8}$. As shown, the design based on radix-4 FFT algorithm needs to process four radix-4 stages and one radix-2 stage. Also, it requires 16 radix-4 PEs, and total 48 non-trivial complex multipliers. The radix-8 design needs to execute three radix-8 FFT stages with four radix-8 PEs, 28 non-trivial and 12 trivial complex multipliers. For radix-16 design, although the required two radix-16 PEs leads to higher area cost than the radix-8 design, we will reformulate conventional radix-16 FFT algorithm so that the effective number of FFT stages is reduced from three to two, and the number of radix-16 PEs can be reduced from two to one (subject to constraint (1)). Thus, a high-throughput and more efficient radix-16 FFT processor design than the radix-8 or radix-16 designs can be achieved, as will be explained in the following sections.

III. A Reformulated Radix-16 FFT Algorithm and its Butterfly Structures

To fully exploit the advantage of a high-radix FFT algorithm (particularly the radix-16 algorithm in this work), conventional FFT algorithm is reformulated in a form suitable for efficient realization of all the function components and overall FFT PE architecture, as will be detailed in the following.

A. Conventional Radix-16 Algorithm

Given an N-point discrete Fourier transform (DFT)

$$X(k)=\sum_{n=0}^{N-1} x(n)W_N^{kn}, \quad k=0,\ldots,N-1 \tag{2}$$

where $x(n)$ and $X(k)$ denote the input and output of the DFT, respectively, and W_N^{kn} is equal to $e^{-j2\pi kn/N}$. Let

$$\begin{aligned} N=512,\ n=32n_1+n_2,\ k=16k_2+k_1,\\ n_1,\ k_1=0,1,\ldots,15; \quad n_2,\ k_2=0,1,\ldots,31. \end{aligned} \tag{3}$$

A radix-16 decimation-in-frequency (DIF) FFT algorithm can be derived based on the following first-stage decomposition by substituting (3) into (2) as

$$\begin{aligned} &X(16k_2+k_1)\\ &=\sum_{n_2=0}^{31}\sum_{n_1=0}^{15} x(32n_1+n_2)W_{512}^{(16k_2+k_1)(32n_1+n_2)}\\ &=\sum_{n_2=0}^{31}\left\{\underbrace{\sum_{n_1=0}^{15} x(32n_1+n_2)W_{16}^{n_1k_1}}_{16\text{-}point\ DFT\ of\ the\ 1st\ stage}\quad \underbrace{W_{512}^{n_2k_1}}_{\substack{\text{twiddle factor}\\ \text{of stage 1}}}\right\}W_{32}^{n_2k_2} \end{aligned} \tag{4}$$

$$=\sum_{n_2=0}^{31}\left\{BO_{16}^{(1,n_2)}(k_1)\right\}W_{32}^{n_2k_2} \tag{5}$$

where $BO_{16}^{(1,n_2)}(k_1)$ represents the k_1-th output of the first-stage's n_2-th radix-16 butterfly operation. Equation (4) can be executed starting from the 32 inner (first-stage) 16-point butterfly operations, followed by 16 outer (second-stage) 32-point DFTs. Next, let n_2 and k_2 be further defined as $n_2=2n_2'+n_3$ and $k_2=16k_3+k_2'$, $n_2', k_2'=0,1,\ldots,15$, then radix-16 decomposition is again applied to the second DFT stage, so that one can get the following 512-point FFT in mixed-radix form, i.e., two radix-16 stages and one radix-2 stage

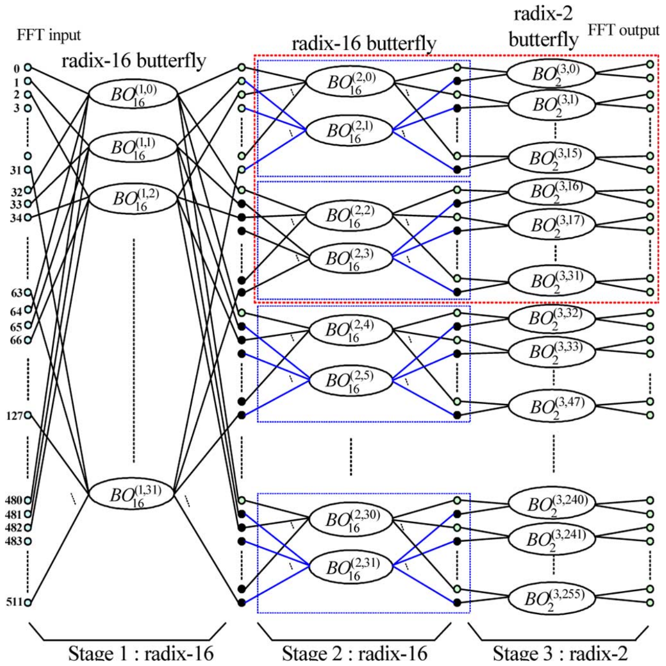


Fig. 2. SFG for radix-16 512-point FFT algorithm.

$$\begin{aligned} &X(16(k_2'+16k_3)+k_1)\\ &=\sum_{n_3=0}^{1}\sum_{n_2'=0}^{15}\left\{BO_{16}^{(1,2n_2'+n_3)}(k_1)\right\}W_{32}^{(2n_2'+n_3)(16k_3+k_2')}\\ &=\sum_{n_3=0}^{1}\left\{\underbrace{\sum_{n_2'=0}^{15} BO_{16}^{(1,2n_2'+n_3)}(k_1)W_{16}^{n_2'k_2'}}_{16\text{-}point\ DFT\ of\ the\ 2nd\ stage}\quad \underbrace{W_{32}^{n_3k_2'}}_{\substack{\text{twiddle factor}\\ \text{of stage 2}}}\right\}W_2^{n_3k_3} \end{aligned} \tag{6}$$

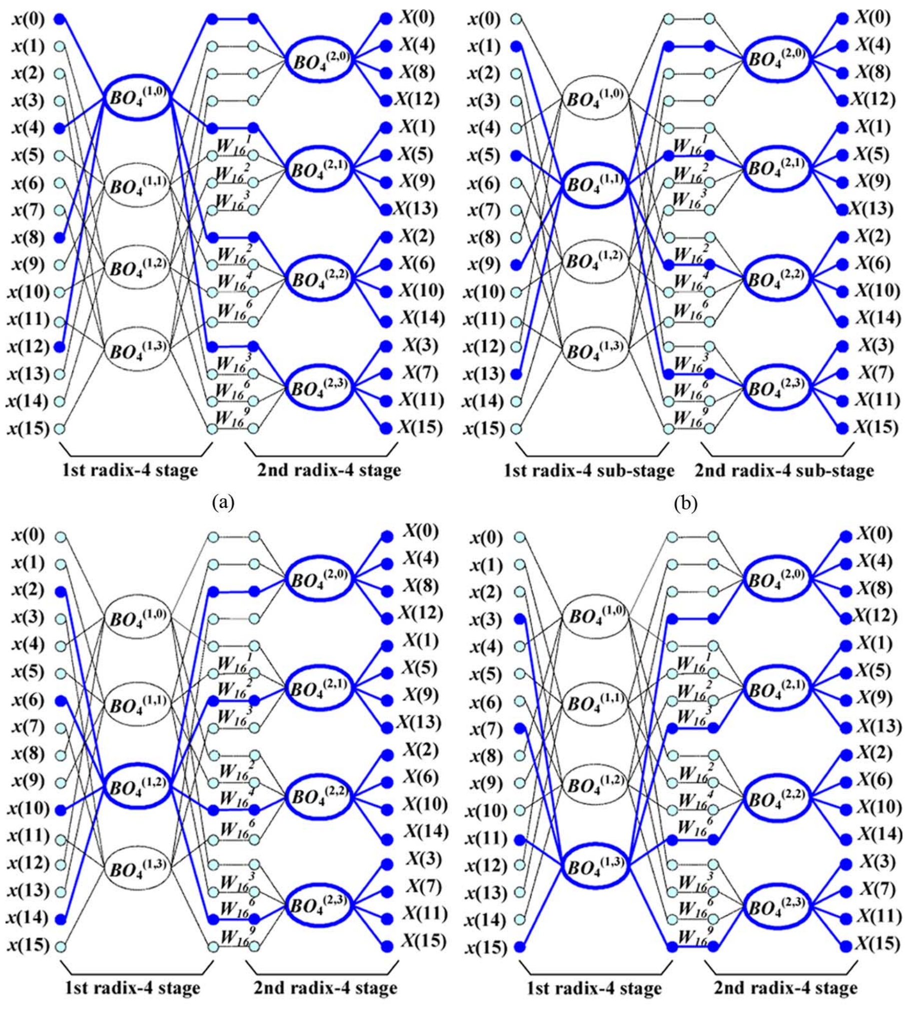
$$=\underbrace{\sum_{n_3=0}^{1}\left\{BO_{16}^{(2,2k_1+n_3)}(k_2')\right\}W_2^{n_3k_3}}_{2\text{-}point\ DFT\ of\ the\ 3rd\ stage} \tag{7}$$

$$=BO_2^{(3,16k_1+k_2')}(k_3). \tag{8}$$

In (7), the term $BO_{16}^{(2,2k_1+n_3)}(k_2')$ represents the k_2'-th output of the second-stage's $(2k_1+n_3)$-th radix-16 butterfly operation, while the term $BO_2^{(3,16k_1+k_2')}(k_3)$ in (8) represents the k_3-th output of the third-stage's radix-2 butterfly operation. The signal-flow graph (SFG) of the 512-point FFT in $(16k_1+k_2')$-th mixed-radix form is shown in Fig. 2.

B. Reformulated Radix-16 Algorithm

For further reducing the high implementation cost of a conventional radix-16 butterfly unit, one can let $n_1=4m_1+m_2$, $k_1=4l_2+l_1, m_1, m_2, l_1, l_2, \in\{0,1,2,3\}$, then the first-stage radix-16 butterfly operations can be reformulated as two radix-4



(a) (b) (c) (d)

Fig. 3. Execution order of radix-4 butterfly operations of the reformulated 16-point butterfly operation (a) $t_{cnt} = 0$. (b) $t_{cnt} = 1$. (c) $t_{cnt} = 2$. (d) $t_{cnt} = 3$.

sub-stages as shown in (9)–(11) at the bottom of the page, where $BO_4^{(1,m_2)}(l_1)$ represents the l_1-th output of the first-sub-stage's m_2-th radix-4 butterfly operation, and $BO_4^{(2,l_1)}(l_2)$ represents the l_2-th output of the second-sub-stage's l_1-th radix-4 butterfly operation. The same process can be applied to the second-stage's radix-16 butterfly operations. The reformulation leads to two simpler cascaded radix-4 butterfly sub-stages than the original radix-16 butterfly stage, because the coefficients of 4-point DFT are all trivial values of $\{1/j/-1/-j\}$. Consequently, a radix-16 butterfly can be efficiently executed in a four-cycle period. During the period, the partial results of all 16 output samples will be accumulated simultaneously. Fig. 3(a)–(d) il-

$$
\begin{aligned}
&BO_{16}^{(1,n_2)}(k_1)\\
&\quad = \sum_{m_2=0}^{3}\sum_{m_1=0}^{3} x(32(4m_1+m_2)+n_2)W_{16}^{(4m_1+m_2)(4l_2+l_1)}W_{512}^{n_2(4l_2+l_1)}\\
&\quad = \underbrace{\sum_{m_2=0}^{3}\underbrace{\left\{\sum_{m_1=0}^{3} x(32(4m_1+m_2)+n_2)W_4^{m_1 l_1}W_{16}^{m_2 l_1}\right\}}_{\text{1st radix-4 substage}} W_4^{m0_2 l_2}\, W_{512}^{n_2(4l_2+l_1)}}_{\text{2nd radix-4 substage}} \qquad (9)
\end{aligned}
$$

$$
= \sum_{m_2=0}^{3}\left\{BO_4^{(1,m_2)}(l_1)\right\}W_4^{m_2 l_2}W_{512}^{n_2(4l_2+l_1)} \qquad (10)
$$

$$
= BO_4^{(2,l_1)}(l_2)W_{512}^{n_2(4l_2+l_1)} \qquad (11)
$$

lustrate the detailed accumulation flow. At the first cycle (i.e., cycle time counter $t_{cnt} = 0$), the first input data set of samples $x(0), x(4), x(8), x(12)$ come in simultaneously, and the butterfly operation $BO_4^{(1,0)}$ of the first radix-4 sub-stage is executed. After the multiplication with corresponding twiddle factors (all equal to one at this cycle time), its four results are then respectively fed to $BO_4^{(2,0)}$, $BO_4^{(2,1)}$, $BO_4^{(2,2)}$, and $BO_4^{(2,3)}$ of the second radix-4 sub-stage for partial butterfly operations of the second radix-4 stage and accumulation of the first partial results of $X(0) \sim X(15)$. In the figure, those active paths are shown in bold blue solid lines, while inactive paths are shown in thin dashed lines. Next, at cycle time $t_{cnt} = 1$, similar process is repeated for $BO_4^{(1,1)}$, and the second partial results of $X(0) \sim X(15)$ are accumulated simultaneously in the same cycle time. This process is continued until all the four butterfly operations ($BO_4^{(1,0)} \sim BO_4^{(1,3)}$) and their associated twiddle factor multiplications and the second radix-4 sub-stage operations are executed.

Since one radix-16 butterfly operation consumes four clock cycles, four independent radix-16 butterfly operations will be executed in parallel in order to meet the throughput requirement of 802.15.3c standard. As a result, the whole operation time for the 512-point FFT can be greatly reduced. Moreover, the reformulated radix-16 512-point FFT operation can be finished within only two radix-16 FFT stages, instead of three FFT stages (i.e., two radix-16 stages and one radix-2 stage) required by conventional radix-16 algorithms, as explained in the following. First, consider the butterfly operations executed on the top-right corner of Fig. 2. Since the four radix-16 butterfly operations $BO_{16}^{(2,0)}$, $BO_{16}^{(2,1)}$, $BO_{16}^{(2,2)}$ and $BO_{16}^{(2,3)}$ are executed simultaneously, outputs from butterfly operations $BO_{16}^{(2,0)}$ and $BO_{16}^{(2,1)}$ can be joined together to do the succeeding radix-2 butterfly operations immediately, and similarly for the outputs from $BO_{16}^{(2,2)}$ and $BO_{16}^{(2,3)}$. On the other hand, for conventional radix-16 algorithms, the final radix-2 stage cannot be executed until all the second-stage's radix-16 butterfly operations have been executed. In the following sections, an area-efficient and high-throughput FFT processor for 802.15.3c system based on the reformulated radix-16 algorithm will be detailed.

IV. High Throughput FFT Architecture

The proposed high-throughput FFT processor architecture is depicted in Fig. 4. It consists of the following main parts, together with their specific novelties and advantages. (i) A memory unit composed of 16 dual-port memory banks, which facilitates 16-way parallel data access. (ii) A memory bank index and address generation unit (BAGU), which generates conflict-free and in-place memory bank indexes and address for the radix-16 FFT operation. (iii) Four commutator blocks located in front of the input side and after the output side of the memory, provide efficient data routing mechanism which is governed by the BAGU signals. (iv) A scaling unit (SU) coordinates controlled scaling operations for block floating-point (BFP) operations, which generates higher signal-to-quantization noise ratio (SQNR) than the existing designs. (v) The kernel processing engine, which is a high-performance computing engine for radix-16 butterfly operations. It contains the following optimized subblocks: four radix-16 PEs (i.e., PE_R16 0 through PE_R16 3), two sets of radix-2 PEs (each set contains four radix-2 PEs), and four sets of complex multipliers (each contains four complex multipliers) for twiddle factor multiplications. Those multipliers are optimized with the help of common-subexpression sharing technique and a new twiddle-factor multiplication scheme. All the function units inside the kernel processing engine are detailed as follows, while the mentioned function units (i) to (iv) will be discussed in the next section.

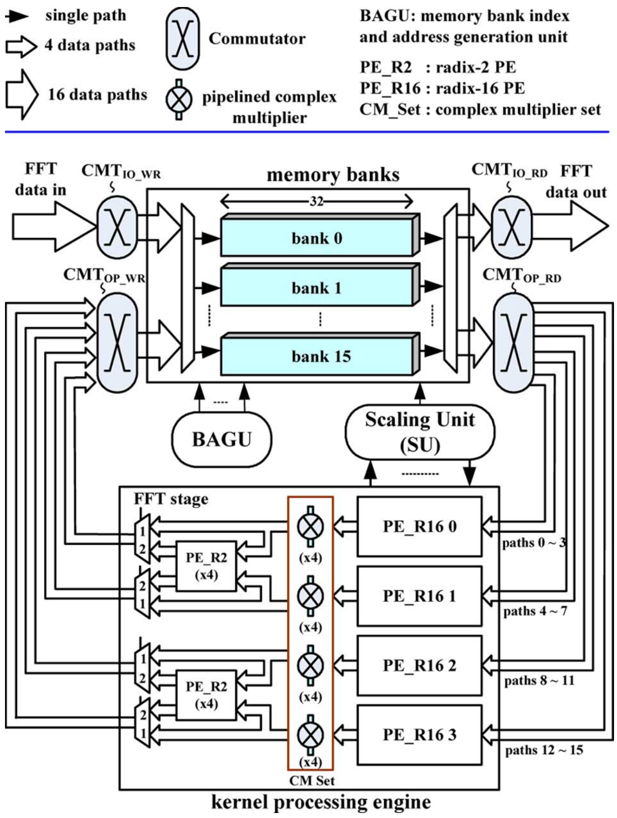


Fig. 4. Block diagram of the proposed FFT processor architecture.

A. $PE\_R16\ 0 \sim PE\_R16\ 3$

The PE_R16 unit, as shown in Fig. 5, is designed specifically for the reformulated radix-16 algorithm described by (9)–(11). The operation begins by reading 16 samples from memory banks and then passing them to proper PE_R16's input sides through the commutator $\mathrm{CMT_{OP\_RD}}$. Though PE_R16 is based on radix-4^2 decomposition, it is different from the conventional radix-4^2 architecture in [24]. For the proposed architecture, since each of the four PE_R16 units executes one radix-16 butterfly operation within four clock cycles simultaneously, the total throughput remains 16 samples/cycle. In addition, it helps to finish the 512-point FFT of 802.15.3c within two FFT stages because the final stage's radix-2 operations can be performed right after the second stage's radix-16 operations. A PE_R16 contains the following components: PE_R4_S1, CM_W_16 set, PE_R4_S2, LU, $\mathrm{MUX\_16}_4$, SAU, and DSSU. Pipeline registers are inserted between PE_R4_S1 and $\mathrm{CM\_}W_{16}$ to achieve high operation speed. The BFP-related function units SAU and

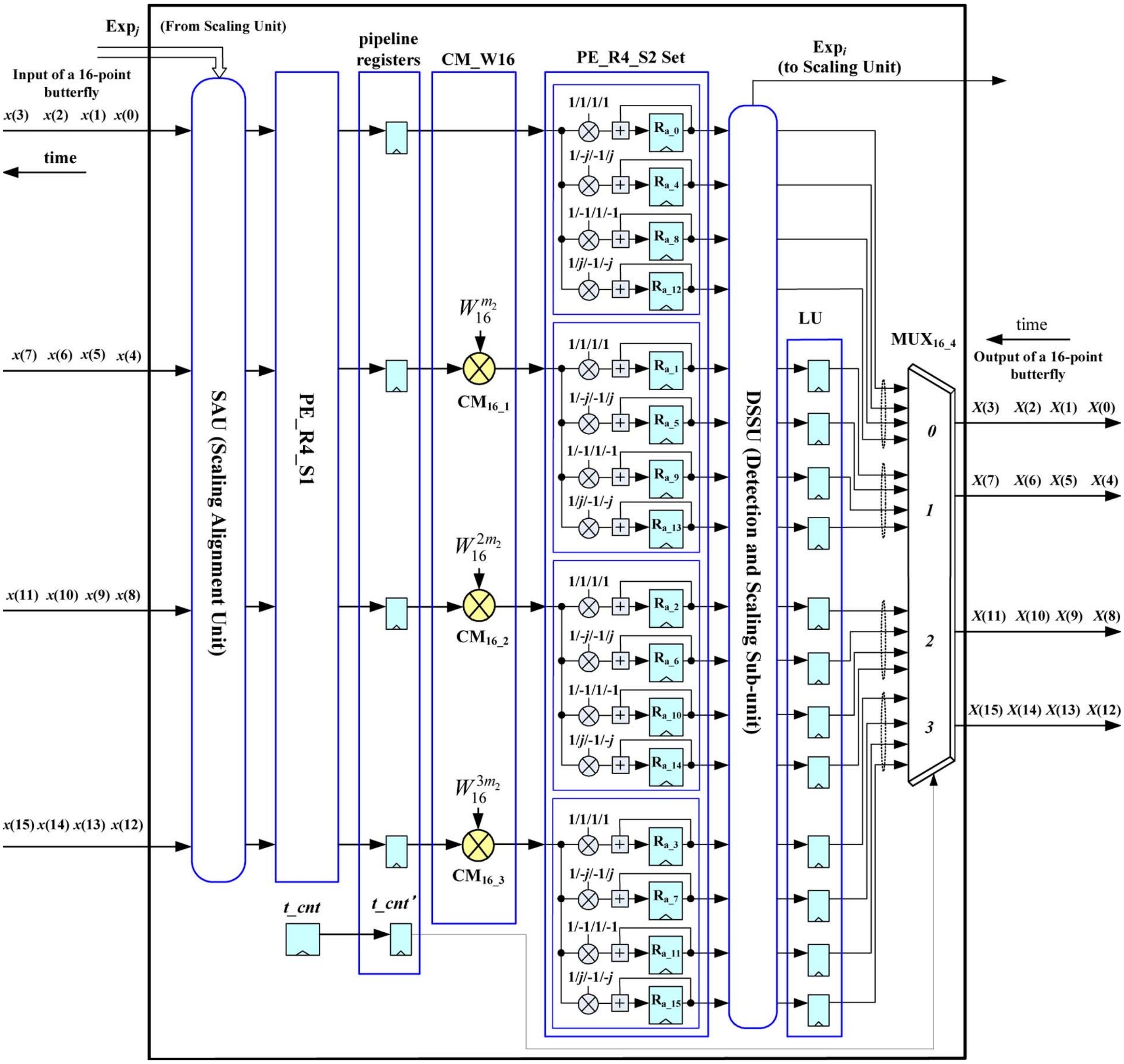


Fig. 5. Architecture of PE_R16.

DSSU will be detailed together with SU in Section VI, while all the other components are explained below.

1) PE_R4_S1: It is responsible for the first radix-4 stage operation, i.e., $BO_4^{(1,0)} \sim BO_4^{(1,3)}$ in Fig. 3. By further applying radix-2^2 decomposition to a 4-point DFT, PE_R4_S1 can be constructed with four radix-2 PEs.

2) CM_W_{16} Set: It is an optimized complex multiplier set which executes the multiplications with constant values of $W_{16}^{m_2 l_1}$, $m_2, l_1 \in \{0, 1, 2, 3\}$ in (9). Since the multiplication term of the first data link (i.e., $l_1 = 0$) is equal to one, only three complex multipliers (i.e., $\mathrm{CM}_{16\_1} \sim \mathrm{CM}_{16\_3}$ for $l_1 = 1, 2$ and 3, respectively) are required. For efficient implementation of these multipliers, two schemes are considered. First, by utilizing symmetry property of complex sinusoidal functions, both the real part $\mathrm{Re}\{x\}$ and imaginary part $\mathrm{Im}\{x\}$ of any number x of those factors can be derived from only three terms, i.e., $\mathrm{Re}\{W_{16}^1\}$, $\mathrm{Re}\{W_{16}^2\}$, and $\mathrm{Re}\{W_{16}^3\}$. For example, $\mathrm{Re}\{W_{16}^9\}$, and $\mathrm{Im}\{W_{16}^9\}$ are equal to $-\mathrm{Re}\{W_{16}^1\}$ and $\mathrm{Re}\{W_{16}^3\}$, respectively. Second, canonical signed digit (CSD) representation and common sub-expression sharing methods are exploited to reduce area cost. Table II lists the CSD representations of $\mathrm{Re}\{W_{16}^1\}$, $\mathrm{Re}\{W_{16}^2\}$, and $\mathrm{Re}\{W_{16}^3\}$. The common term "101" (as enclosed by those red ellipse) between $\mathrm{Re}\{W_{16}^1\}$ and $\mathrm{Re}\{W_{16}^2\}$ are used to construct the common subexpression block. With proper shifting and combinations, one can construct $\mathrm{CM}_{16\_1} \sim \mathrm{CM}_{16\_3}$ with low hardware cost. The architecture of CM_{16_1} is shown as an example in Fig. 6, while CM_{16_2}, CM_{16_3} are similar to CM_{16_1}. From the synthesis results, the total area cost of CM_W$_{16}$ set is very close to that of a conventional complex multiplier.

TABLE II
CSD REPRESENTATION OF THREE BASIC TWIDDLE FACTORS

CSD representation: (12 bits)												
$\mathrm{Re}\{W_{16}{}^1\}$	1	0	0	0	-1	0	-1	0	0	1	0	0
$\mathrm{Re}\{W_{16}{}^2\}$	1	0	-1	0	-1	0	1	0	1	0	0	0
$\mathrm{Re}\{W_{16}{}^3\}$	0	1	0	-1	0	0	0	1	0	0	0	0

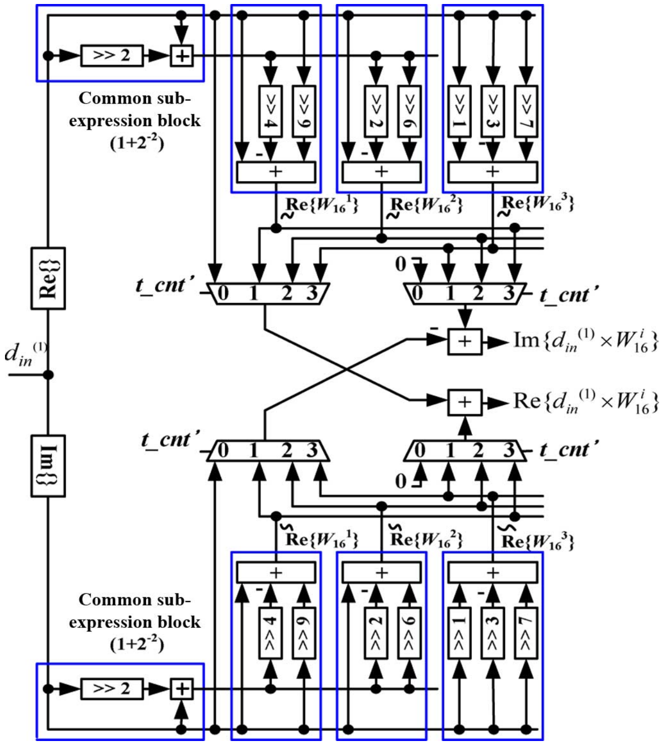


Fig. 6. Low-complexity realizations of CM_{16_1}.

3) PE_R4_S2 Set: It contains four sub-blocks which are responsible for the second radix-4 stage's butterfly operations, i.e., $BO_4^{(2,0)} \sim BO_4^{(2,3)}$. For each PE_R4_S2, its input data arrive sequentially (as indicated in Fig. 3), and four registers are used for accumulating the partial results. Overall there are 16 registers (i.e., $R_{\mathrm{a\_0}} \sim R_{\mathrm{a\_15}}$). In front of each accumulator, the four-point DFT coefficients $\{1, j, -1, -j\}$ are dynamically selected for proper partial result generation in each clock cycle. After every four cycles, the 16-point DFT results will be ready.

4) LU: Since at each clock cycle, only four DFT results of total 16 DFT results inside a PE_R16 are scheduled to send out for twiddle factor multiplication, 12 extra registers are necessary for temporary buffering of the DFT results, otherwise these values will be over-written by the incoming data of the succeeding butterfly operations. Note that the four DFT results stored in $R_{\mathrm{a\_0}}$, $R_{\mathrm{a\_4}}$, $R_{\mathrm{a\_8}}$, and $R_{\mathrm{a\_12}}$, can be output immediately without buffering.

5) MUX 16_4: The multiplexer selects four data samples out of its 16 input data at each clock cycle. The selection mechanism is based on a simple in-place memory addressing scheme, which means that a DFT output $X(i)$ will be written back to the same memory location storing the DFT input data with the same index (i.e., $x(i)$).

B. PE_R2 Set

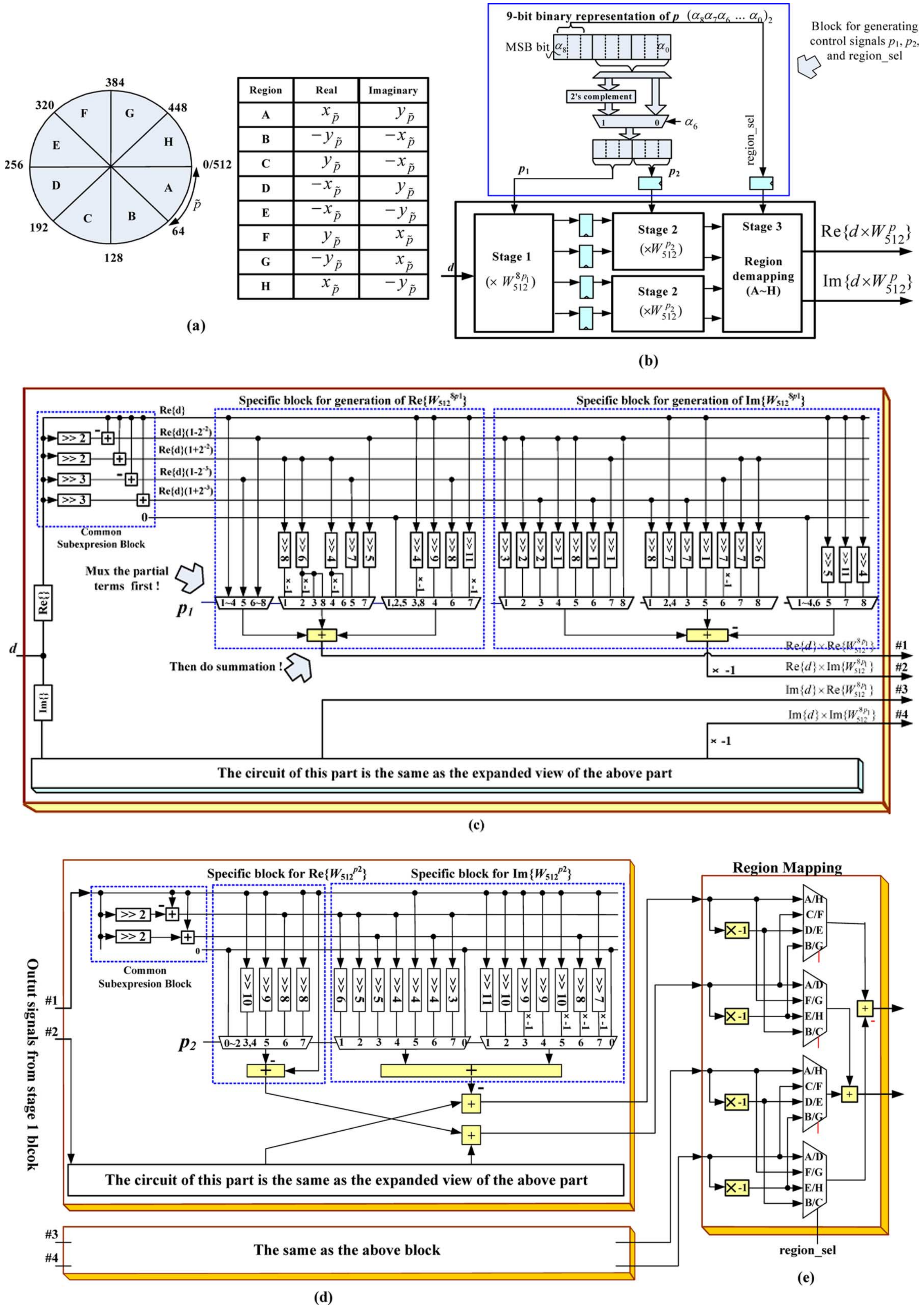
PE_R2 set is responsible for the final-stage's radix-2 butterfly operations. It consists of 8 radix-2 PEs. As mentioned earlier in Section III, to meet the stringent throughput requirement, the final radix-2 stage is executed immediately after its preceding radix-16 stage (i.e., the second FFT stage), without feeding back the results of PE-R16 units to memory.

C. Complex Multiplier Set (CM Set) and a Novel Twiddle Factor Multiplication Scheme

A shown in Fig. 4, each set of 16 output data from the four PE_R16 units will be simultaneously multiplied by proper twiddle factors, i.e., the $W_{512}^{n_2 k_1}$ terms in (4), for the first FFT stage or the $W_{32}^{n_3 k_2'}$ terms in (6) for the second FFT stage. To reduce the multiplier area, a new three-stage multiplier structure for twiddle factor multiplication is proposed as follows. Consider the multiplication operation of a term d with a twiddle factor W_{512}^{p}, $p = 0 \sim 511$. First, by utilizing 1/8 symmetry property, the exponent p is mapped to a number $\tilde{p}$ in a smaller region A (as shown in Fig. 7(a)) than before, so that $d \times W_{512}^{\tilde{p}}$ is considered instead. Besides, in order to further reduce the number of the twiddle factors and simplify the associated multiplication operations, $\tilde{p}$ is rewritten as $\tilde{p} = 8p_1 + p_2$, $p_1 = 0 \ldots 8$, $p_2 = 0 \ldots 7$, so that can be decomposed into the following two-stage multiplication operation:

$$\begin{aligned} d \times W_{512}^{\tilde{p}} &= d \times W_{512}^{(8p_1+p_2)} = d \times \underbrace{W_{512}^{8p_1}}_{\text{stage 1}} \times \underbrace{W_{512}^{p_2}}_{\text{stage 2}} \\ &= d \times (\mathrm{Re}\{W_{512}^{8p_1}\} + j\,\mathrm{Im}\{W_{512}^{8p_1}\}) \\ &\quad \times (\mathrm{Re}\{W_{512}^{p_2}\} + j\,\mathrm{Im}\{W_{512}^{p_2}\}) \\ &= (\mathrm{Re}\{d\}\mathrm{Re}\{W_{512}^{8p_1}\}\mathrm{Re}\{W_{512}^{p_2}\} \\ &\quad - \mathrm{Re}\{d\}\mathrm{Im}\{W_{512}^{8p_1}\}\mathrm{Im}\{W_{512}^{p_2}\}) \\ &\quad + j\,(\mathrm{Re}\{d\}\mathrm{Re}\{W_{512}^{8p_1}\}\mathrm{Im}\{W_{512}^{p_2}\} \\ &\quad + \mathrm{Re}\{d\}\mathrm{Im}\{W_{512}^{8p_1}\}\mathrm{Re}\{W_{512}^{p_2}\}) \\ &\quad + j\,(\mathrm{Im}\{d\}\mathrm{Re}\{W_{512}^{8p_1}\}\mathrm{Re}\{W_{512}^{p_2}\} \\ &\quad - \mathrm{Im}\{d\}\mathrm{Im}\{W_{512}^{8p_1}\}\mathrm{Im}\{W_{512}^{p_2}\}) \\ &\quad - (\mathrm{Im}\{d\}\mathrm{Re}\{W_{512}^{8p_1}\}\mathrm{Im}\{W_{512}^{p_2}\} \\ &\quad + \mathrm{Im}\{d\}\mathrm{Im}\{W_{512}^{8p_1}\}\mathrm{Re}\{W_{512}^{p_2}\}). \end{aligned} \tag{12}$$

By doing so, equivalently the 64 twiddle factors are further reduced to 16 different values. The corresponding block diagram of a complex twiddle-factor multiplier is shown in Fig. 7(b). Furthermore, based on the CSD representations of the 16 parameters listed in Table III, multiplier-less realization of the first two stages in the figure can be achieved by optimizing the common sub-expression eliminations of the 16 parameters. Similar to the scheme used in $\mathrm{CM\_W}_{16}$ set, the four terms "101", "10–1", "1001", "100–1", as enclosed by red or blue ellipses in Table III, are used to construct the common sub-expression block. The optimized multiplier-less stage-1 and stage-2 architectures are shown in Figs. 7(c) and 7(d), respectively. The proposed multiplier-less architecture is area efficient. From synthesis results, it achieves 24% area reduction of conventional complex multipliers. Unlike the constant multiplier schemes in [12], [22], [23], which are mostly limited to $\mathrm{W}_{16}/\mathrm{W}_{32}/\mathrm{W}_{64}$ multiplications, the proposed scheme can be applied to all the constant multiplications in the entire design. In addition, there is no need to allocate memory for storing any twiddle factors. The control information p_1, p_2, and region_index are generated easily in a low-complexity submodule on the fly, as shown at the top-left corner of Fig. 7(b). From synthesis result, the area of these control signals is roughly 1.38% of a complex multiplier


384
320
448
256
0/512
192
64
128
F
G
E
H
D
A
C
B
$\tilde{p}$
9-bit binary representation of p $(\alpha_8\alpha_7\alpha_6 \ldots \alpha_0)_2$
MSB bit α_8
α_0
2's complement
1
0
α_6
p_1
p_2
region_sel
Block for generating control signals p_1, p_2, and region_sel
d
Stage 1 $(\times W_{512}^{8p_1})$
Stage 2 $(\times W_{512}^{p_2})$
Stage 2 $(\times W_{512}^{p_2})$
Stage 3 Region demapping (A~H)
$\mathrm{Re}\{d\times W_{512}^{p}\}$
$\mathrm{Im}\{d\times W_{512}^{p}\}$
Specific block for generation of $\mathrm{Re}\{W_{512}^{8p1}\}$
Specific block for generation of $\mathrm{Im}\{W_{512}^{8p1}\}$
Re{d}
Re{d}(1-2^-2)
Re{d}(1+2^-2)
Re{d}(1-2^-3)
Re{d}(1+2^-3)
0
>>2
>>3
Common Subexpresion Block
Mux the partial terms first !
Then do summation !
p_1
Re{}
Im{}
× -1
$\mathrm{Re}\{d\}\times\mathrm{Re}\{W_{512}^{8p_1}\}$ #1
$\mathrm{Re}\{d\}\times\mathrm{Im}\{W_{512}^{8p_1}\}$ #2
$\mathrm{Im}\{d\}\times\mathrm{Re}\{W_{512}^{8p_1}\}$ #3
$\mathrm{Im}\{d\}\times\mathrm{Im}\{W_{512}^{8p_1}\}$ #4
The circuit of this part is the same as the expanded view of the above part
Specific block for $\mathrm{Re}\{W_{512}^{p2}\}$
Specific block for $\mathrm{Im}\{W_{512}^{p2}\}$
p_2
Outut signals from stage 1 blcok
#1
#2
#3
#4
The same as the above block
Region Mapping
A/H
C/F
D/E
B/G
A/D
F/G
E/H
B/C
× -1
region_sel

Region	Real	Imaginary
A	$x_{\tilde p}$	$y_{\tilde p}$
B	$-y_{\tilde p}$	$-x_{\tilde p}$
C	$y_{\tilde p}$	$-x_{\tilde p}$
D	$-x_{\tilde p}$	$y_{\tilde p}$
E	$-x_{\tilde p}$	$-y_{\tilde p}$
F	$y_{\tilde p}$	$x_{\tilde p}$
G	$-y_{\tilde p}$	$x_{\tilde p}$
H	$x_{\tilde p}$	$-y_{\tilde p}$

(a)

(b)

(c)

(d)

(e)

Fig. 7. (a) Eight-Region symmetric division and mapping of twiddle factors. (b) Block diagram of three-stage realization of a complex multiplier (for twiddle factor). (c)–(e) Detailed structure of the three-stage realization of a complex multiplie.

in average. Note that for high-speed operation, pipelined registers are inserted between stage-1 and stage-2 blocks. After the two-stage operations, the third stage de-maps the result back to the correct form.

TABLE III
12-BIT CSD REPRESENTATIONS OF 16 BASIC VALUES FOR COMPOSING TWIDDLE FACTORS

p_1	$\mathrm{Re}\{W_{512}^{8p_1}\}$												$-\mathrm{Im}\{W_{512}^{8p_1}\}$											
1	1	0	0	0	0	0	0	0	-1	0	-1	0	0	0	0	1	0	-1	0	0	1	0	0	1
2	1	0	0	0	0	0	-1	0	-1	0	0	0	0	0	1	0	-1	0	0	1	0	0	0	0
3	1	0	0	0	-1	0	1	0	1	0	0	0	0	0	1	0	0	1	0	1	0	0	1	0
4	1	0	0	0	-1	0	-1	0	0	1	0	0	0	1	0	-1	0	0	0	1	0	0	0	0
5	1	0	0	-1	0	0	0	1	0	0	-1	0	0	1	0	0	0	-1	0	0	1	0	-1	0
6	1	0	-1	0	1	0	1	0	1	0	0	-1	0	1	0	0	1	0	0	-1	0	0	1	0
7	1	0	-1	0	0	1	0	-1	0	0	0	-1	0	1	0	1	0	0	0	1	0	1	0	-1
8	1	0	-1	0	-1	0	1	0	1	0	0	0	1	0	-1	0	-1	0	1	0	1	0	0	0
p_2	$\mathrm{Re}\{W_{512}^{p_2}\}$												$-\mathrm{Im}\{W_{512}^{p_2}\}$											
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	-1	0	0	1
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	-1	0	0	1	0
3	1	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	1	0	1	0	-1	0	0
4	1	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	1	0	-1	0	0	1	0	0
5	1	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	1	0	0	0	0	0	-1	0
6	1	0	0	0	0	0	0	0	-1	0	1	0	0	0	0	0	1	0	1	0	-1	0	-1	0
7	1	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	1	0	-1	0	-1	0	0	0	0

V. NEW CONFLICT-FREE NORMAL-ORDER OUTPUT MEMORY ADDRESSING SCHEME

To avoid possible conflicts in simultaneously reading (or writing) 16 data from (or to) the memory banks during FFT operations, a proper memory addressing scheme is necessary. The well-known non-conflict memory addressing schemes [5], [7] are only applicable to radix-2 FFT algorithm. Although the addressing scheme in [6] is for general radix-r FFT operations, its FFT size should be a power-of-r number. Besides, those schemes are only limited to single-PE architecture. On the other hand, the radix-2 addressing scheme for multiple PEs [16] is relatively inefficient compared with higher-radix schemes and is not suitable for 802.15.3c application. In the following, we will detail a new memory efficient addressing scheme suitable for the proposed multiple-PE FFT architecture.

The proposed scheme has three special features. First, it ensures conflict-free FFT butterfly executions during the entire FFT operation. Second, it supports parallel data outputs with normal ordering. This feature is always desirable for providing immediate and normal-order FFT outputs to the succeeding functional blocks, such as channel estimator for timely operations. Thirdly, like many other designs, the in-place FFT computation strategy is also adopted for low memory overhead consideration. The scheme is described as follows.

A. Optimized Input Data Loading for Conflict-free FFT Operation

In the beginning of FFT operation, input data $x(n)$, $n = 0, 1, \ldots, 511$, are loaded into the 16 memory banks according to the following mapping functions for later conflict-free data access. Let $n = (b_8b_7b_6b_5b_4b_3b_2b_1b_0)_2$, $b_i \in \{0,1\}$, be the binary representation of input sample index n. Then, $x(n)$ is routed to the following memory address $Addr[x(n)]$ inside the memory bank $Bank[x(n)]$:

$$\begin{aligned} Bank[x(n)] = {} & (8(b_5 \oplus b_1) + 4b_0 + 4b_8 \\ & + ((b_8b_7)_2 + 2b_6 + (b_4b_3)_2 \\ & + (b_2b_1)_2 + (2b_5 - 1) \\ & \times (b_5 \oplus b_1))\mathrm{mod}(4))\mathrm{mod}(16) \qquad (13) \\ Addr[x(n)] = {} & (b_8b_7b_6b_4b_3)_2 \qquad (14) \end{aligned}$$

TABLE IV
DATA EXECUTION ARRANGEMENT AND DISTRIBUTION IN THE FOUR RADIX-16 PES FOR ENTIRE FFT OPERATIONS

	PE_R16 0				PE_R16 1				PE_R16 2				PE_R16 3			
Stage 1	0	128	256	384	1	129	257	385	2	130	258	386	3	131	259	387
	32	160	288	416	33	161	289	417	34	162	290	418	35	163	291	419
	64	192	320	448	65	193	321	449	66	194	322	450	67	195	323	451
	96	224	352	480	97	225	353	481	98	226	354	482	99	227	355	483
	4	132	260	388	5	133	261	389	6	134	262	390	7	135	263	391
	36	164	292	420	37	165	293	421	38	166	294	422	39	167	295	423
	68	196	324	452	69	197	325	453	70	198	326	454	71	199	327	455
	100	228	356	484	101	229	357	485	102	230	358	486	103	231	359	487
	≈				≈				≈				≈			
	28	156	284	412	29	157	285	413	30	158	286	414	31	159	287	415
	60	188	316	444	61	189	317	445	62	190	318	446	63	191	319	447
	92	220	348	476	93	221	349	477	94	222	350	478	95	223	351	479
	124	252	380	508	125	253	381	509	126	254	382	510	127	255	383	511
Stage 2	0	8	16	24	1	9	17	25	32	40	48	56	33	41	49	57
	2	10	18	26	3	11	19	27	34	42	50	58	35	43	51	59
	4	12	20	28	5	13	21	29	36	44	52	60	37	45	53	61
	6	14	22	30	7	15	23	31	38	46	54	62	39	47	55	63
	64	72	80	88	65	73	81	89	96	104	112	120	97	105	113	121
	66	74	82	90	67	75	83	91	98	106	114	122	99	107	115	123
	68	76	84	92	69	77	85	93	100	108	116	124	101	109	117	125
	70	78	86	94	71	79	87	95	102	110	118	126	103	111	119	127
	≈				≈				≈				≈			
	448	456	464	472	449	457	465	473	480	488	496	504	481	489	497	505
	450	458	466	474	451	459	467	475	482	490	498	506	483	491	499	507
	452	460	468	476	453	461	469	477	484	492	500	508	485	493	501	509
time	454	462	470	478	455	463	471	479	486	494	502	510	487	495	503	511

where symbol "$\oplus$" means the "Exclusive OR" operation, while "mod" represents the modulo operation. The derivation of the above formulas is based on the following considerations:

1) Conflict-Free Memory Access: Table IV lists the partial execution arrangement and the associated data distribution of the first and the second radix-16 stages' butterfly operations. In the tables, the four vertical charts are respectively associated with the four radix-16 PEs. The 16 entries in the same row of the charts indicate those data to be input concurrently to the corresponding PEs, and therefore should be put in different memory banks. Moreover, for the first FFT stage, all those four consecutively indexed input data in each row should be stored in different memory banks, and so are those input data whose indexes differ by multiples of 128 (due to factor b_7 in (13)). On the other hand, for the second FFT stage, in every row of each chart, those input data whose indexes differ by multiples

TABLE V
DETAILED ALLOCATION OF INPUT DATA IN MEMORY BANKS

addr \ bank	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	34	4	38	1	35	5	39	2	32	6	36	3	33	7	37
1	46	8	42	12	47	9	43	13	44	10	40	14	45	11	41	15
2	20	54	16	50	21	55	17	51	22	52	18	48	23	53	19	49
3	58	28	62	24	59	29	63	25	56	30	60	26	57	31	61	27
4	68	102	64	98	69	103	65	99	70	100	66	96	71	101	67	97
5	106	76	110	72	107	77	111	73	104	78	108	74	105	79	109	75
6	80	114	84	118	81	115	85	119	82	112	86	116	83	113	87	117
7	126	88	122	92	127	89	123	93	124	90	120	94	125	91	121	95
8	166	128	162	132	167	129	163	133	164	130	160	134	165	131	161	135
9	140	174	136	170	141	175	137	171	142	172	138	168	143	173	139	169
10	178	148	182	144	179	149	183	145	176	150	180	146	177	151	181	147
11	152	186	156	190	153	187	157	191	154	184	158	188	155	185	159	189
12	226	196	230	192	227	197	231	193	224	198	228	194	225	199	229	195
13	200	234	204	238	201	235	205	239	202	232	206	236	203	233	207	237
14	246	208	242	212	247	209	243	213	244	210	240	214	245	211	241	215
15	220	254	216	250	221	255	217	251	222	252	218	248	223	253	219	249
16	263	293	259	289	260	294	256	290	261	295	257	291	262	292	258	288
≈				≈						≈						
30	497	471	501	467	498	468	502	464	499	469	503	465	496	470	500	466
31	475	505	479	509	472	506	476	510	473	507	477	511	474	504	478	508

of 8 should also be stored in different memory banks. Note that after each radix-16 butterfly operation is done, the results will be written back to their corresponding in-place memory locations through commutators. Based on (13) and (14), allocation of the input data samples are optimized as shown in Table V, which results in conflict-free data accesses and uninterrupted butterfly operations.

2) Parallel and Normal-Order Output Capability: this is one of the key contributions of the proposed design. When a 512-point FFT computation is completed, each set of 16 contiguous $X(k)s$ will be distributed in 16 different memory banks due to the following two factors in (13).

$2b_6$: When $b_6 = 1$, since it corresponds to data index increase of 64, this factor contributes to the increase of the bank index $Bank[x(n+64)]$ of $x(n+64)$ by 2 (under mod 4 operation) over that of $x(n)$, inside each set of 128 contiguous data (i.e., $x(0) \sim x(127)$, $x(128) \sim x(255)$, $x(256) \sim x(383)$, $x(384) \sim x(511)$). For examples, as highlighted in Table V, $Bank[x(0)] = 0$ and $Bank[x(64)] = 2$, while $Bank[x(4)] = 2$ and $Bank[x(68)] = 0$.

$4b_8$: When $b_8 = 1$, since it corresponds to index increase of 256, this factor contributes to the increase of $Bank[x(n+256)]$ value by 4 (under mod 16 operation) over $Bank[x(n)]$, in addition to the bank index value increase of $2b_8 = 2$ contributed from the term $(b_8b_7)_2$ in (13). For example, if $Bank[x(0)] = 0$, then $Bank[x(256)] = 0 + 2 + 4 = 6$, as also highlighted in Table V.

Since radix-16 algorithm and in-place memory addressing scheme are adopted, a quad-bit digit-reverse mapping between $X(k)$, $k = (k_8k_7k_6\ldots k_0)_2$, and $x(n)$ can be described by the function of $k = (k_8k_7k_6k_5k_4k_3k_2k_1k_0)_2 = (b_0b_4b_3b_2b_1b_8b_7b_6b_5)_2$. Table VI lists the partial mapping patterns for all $X(k)$s. It can be verified that the all the 16 contiguous FFT output data $X(16m) \sim X(16m+15)$, $m = 0$ to 31, are in different memory banks. This feature achieves high output rate of $X(k)$ (16 samples/cycle), which facilitates an efficient integrated system design.

B. Bank Index and Address Generation Unit (BAGU) and Commutators

During FFT operations, first a 5-bit counter *slot_cnt* (counts from $0 \sim 31$) is used to help generating the corresponding locations of the four PE_R16s' inputs according to the arrangements in Table IV. The four PE_R16s' input data paths are labeled with path indexes $0, 1, \ldots, 15$, as shown in Fig. 4. Then the 16 memory bank indexes $(bank\_p_0 \sim bank\_p_{15})$ and 16 addresses $(addr\_p_0 \sim addr\_p_{15})$ of the 16 parallel inputs to the four radix-16 PEs are generated according to (13) and (14), respectively. Next, a path decoder will decode the bank index and generate the selection signals $sel\_bank_0 \sim sel\_bank_{15}$ for the 16 memory banks. Together with $addr\_p_0 \sim addr\_p_{15}$, they form the memory read addresses for the 16 memory bank during FFT operations. A simplified logic block for path decoder and the detailed interconnections among path decoder, PE inputs/outputs, 16 memory banks, and commutators are shown in Fig. 8. Also, by adopting the in-place memory access scheme and pipelining the read addresses in a shift register of length 7, the memory write addresses can be easily obtained 7 clock cycles after the read addresses are generated. Meanwhile, commutator $\mathrm{CMT_{OP\_WR}}$ will route the 16 PE outputs to proper memory banks according to the control signals obtained from the delayed selection signals (i.e., $sel\_bank_0 \sim sel\_bank_{15}$) by 7 cycles. On the other hand, commutator $\mathrm{CMT_{OP\_RD}}$ will route the 16 memory bank outputs to proper radix-16 PE inputs according to the signals obtained from the delayed $bank\_p_0 \sim bank\_p_{15}$ signals by one cycle. Commutator $\mathrm{CMT_{IO\_WR}}$ is responsible for routing the input data to proper memory bank during FFT input period, while $\mathrm{CMT_{IO\_RD}}$ is responsible for routing the memory output data to proper output destinations. A counter is used to help generating the input sample index of each input path at each clock cycle, followed by the generations of bank and address indexes of each path. On the other hand, during FFT output period, a counter is used to help generating the output index (i.e., k of $X(k)$) of each output path, then each k should be converted to its corresponding index n first before generating the bank and memory indices. Since the remaining processes are similar to those for regular FFT operations, their discussions are omitted.

VI. IMPLEMENTATION ISSUES

A. Block Floating-Point Scheme and SQNR Performance

For hardware implementation of the proposed FFT processor, a proper wordlength should be determined first. In order to obtain high SQNR, block floating-point (BFP) approach is often adopted [11]. In this work, block size of 16 is adopted for matching with the proposed radix-16 based architecture. As mentioned before, SU (shown in Fig. 4) coordinates the overall scaling operations. Fig. 5 illustrates that after each set of 16-point DFT is done, DSSU will detect the maximum magnitude of data values stored in $R_{\mathrm{a\_0}} \sim R_{\mathrm{a\_15}}$. Then, a scaling value (i.e., an exponent value) is determined and kept in SU. Meanwhile, the values stored in $R_{\mathrm{a\_0}} \sim R_{\mathrm{a\_15}}$ will be scaled before being outputting to complex multipliers. The scheme can effectively reduce the area cost of complex multipliers, because its wordlength is maintained the same as

TABLE VI
THE PARTIAL LOCATION DISTRIBUTION OF $X(k)$ IN THE MEMORY BANKS

***X*(*k*)**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Bank	0	9	2	11	1	10	3	8	6	15	4	13	7	12	5	14
Addr	0	0	4	4	8	8	12	12	16	16	20	20	24	24	28	28
***X*(*k*)**	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Bank	8	1	10	3	9	2	11	0	14	7	12	5	15	4	13	6
Addr	0	0	4	4	8	8	12	12	16	16	20	20	24	24	28	28
				≀≀								≀≀				
***X*(*k*)**	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511
Bank	13	6	15	4	14	7	12	5	3	8	1	10	0	9	2	11
Addr	3	3	7	7	11	11	15	15	19	19	23	23	27	27	31	31

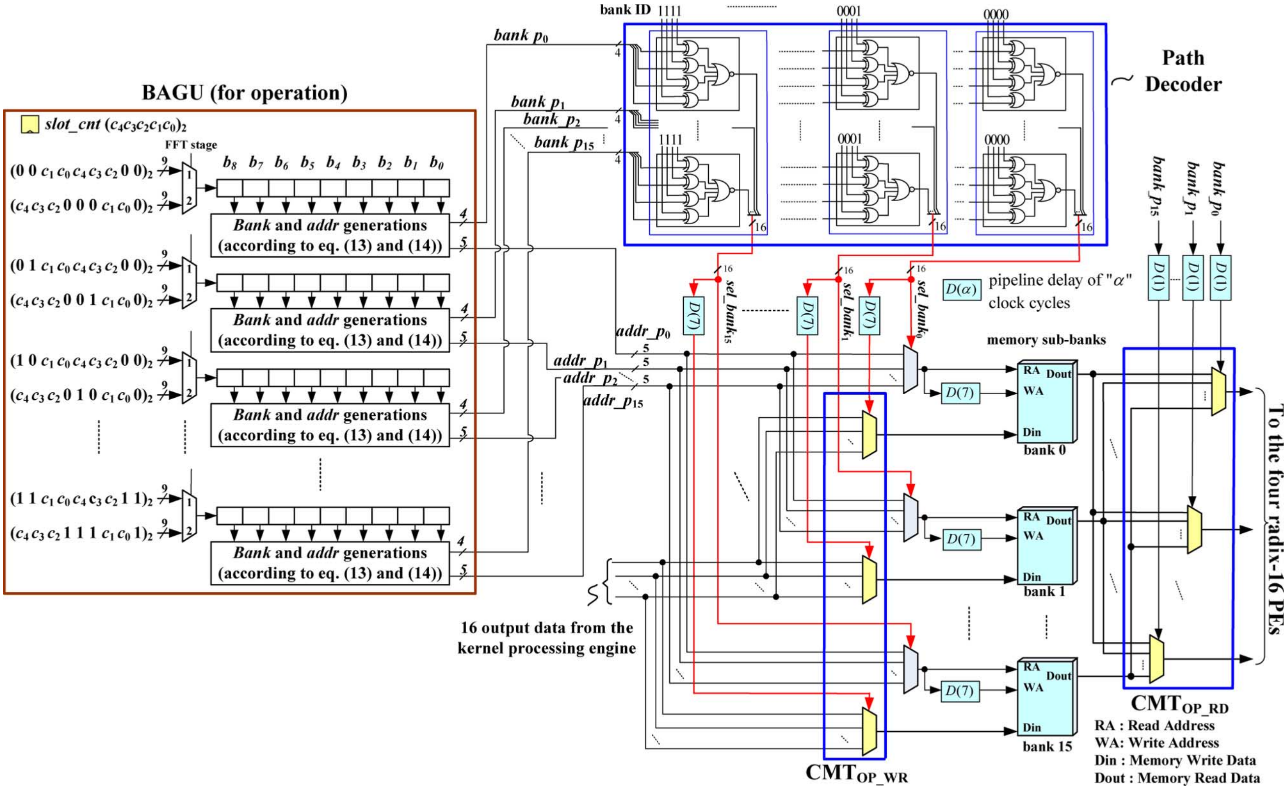

BAGU (for operation)
slot_cnt ($c_4c_3c_2c_1c_0$)$_2$
FFT stage
$(0\ 0\ c_1\ c_0\ c_4\ c_3\ c_2\ 0\ 0)_2$
$(c_4\ c_3\ c_2\ 0\ 0\ 0\ c_1\ c_0\ 0)_2$
$(0\ 1\ c_1\ c_0\ c_4\ c_3\ c_2\ 0\ 0)_2$
$(c_4\ c_3\ c_2\ 0\ 0\ 1\ c_1\ c_0\ 0)_2$
$(1\ 0\ c_1\ c_0\ c_4\ c_3\ c_2\ 0\ 0)_2$
$(c_4\ c_3\ c_2\ 0\ 1\ 0\ c_1\ c_0\ 0)_2$
$(1\ 1\ c_1\ c_0\ c_4\ c_3\ c_2\ 1\ 1)_2$
$(c_4\ c_3\ c_2\ 1\ 1\ 1\ c_1\ c_0\ 1)_2$
b_8 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0
Bank and addr generations (according to eq. (13) and (14))
bank ID 1111 0001 0000
bank p_0
bank p_1
bank p_2
bank p_{15}
Path Decoder
sel_bank$_{15}$
sel_bank$_1$
sel_bank$_0$
$D(\alpha)$ pipeline delay of "α" clock cycles
$D(7)$
$D(1)$
addr_p_0
addr_p_1
addr_p_2
addr_p_{15}
memory sub-banks
RA WA Din Dout
bank 0
bank 1
bank 15
16 output data from the kernel processing engine
CMT$_{\text{OP\_WR}}$
CMT$_{\text{OP\_RD}}$
To the four radix-16 PEs
RA : Read Address
WA: Write Address
Din : Memory Write Data
Dout : Memory Read Data

Fig. 8. Detailed architectures of BAGU, Path Decoder and Commutators.

that of PE_R16s' inputs. At the second radix-16 FFT stage, the inputs to each PE_R16 should be aligned through SAU. The SQNR performance versus wordlength is shown in Fig. 9. With the radix-16 BFP approach, better SQNR performance can be achieved compared to per-FFT-stage fixed scaling method (i.e., scale by 4 bits for each radix-16 stage). Since the proposed FFT processor is also employed to do FFT operations for channel equalizations [21], for enough SQNR consideration and without loss of generality 12-bit wordlength is chosen in our implementations to achieve 57 dB SQNR results.

B. Continuous-Flow Operation

The continuous-flow operation is an important issue for FFT processor design when integrating a FFT processor into the whole system. Though the addressing scheme in [6] only needs a memory size of $2N$ for maintaining continuous-flow operation, it can't support mixed-radix operation. A continuous-flow mixed-radix FFT processor with a novel in-place scheme was proposed in [17]. However, it can't meet the high-throughput requirement of 802.15.3c, because it is based on low-radix FFT algorithm (i.e., rdix-4/2). Regardless of

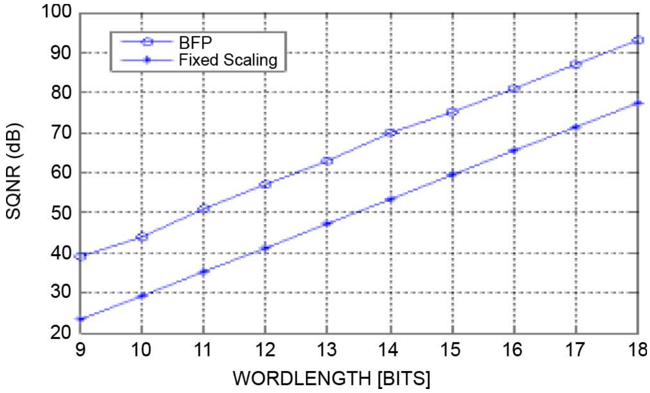


Fig. 9. SQNR performance versus wordlength.

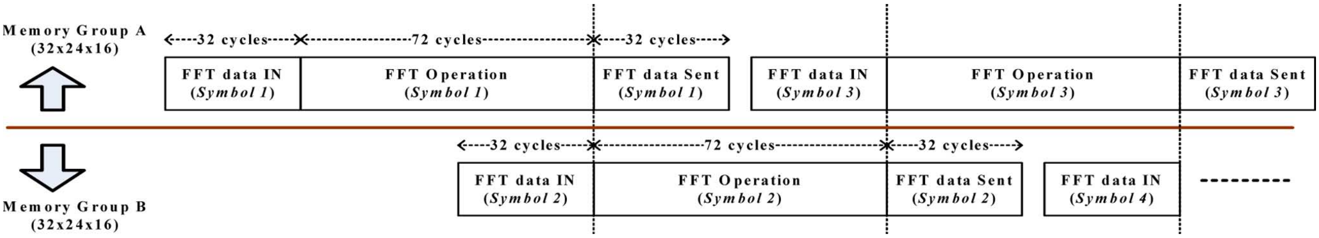


Fig. 10. FFT input/output and operation scheduling for continuous-flow operations.

TABLE VII
THE COMPARISONS OF SEVERAL HIGH-THROUGHPUT FFT PROCESSORS FOR 802.15.3C (A) CBM IS THE ABBREVIATION OF COMPLEX BOOTH MULTIPLIER (B) CCM IS THE ABBREVIATION OF COMPLEX CONSTANT MULTIPLIER

		This work	[19]	[12] (modified)	[13]	[23]
FFT architecture		memory based	memory based	MDF	MDF	MDF
FFT algorithm		reformulated radix-16	radix-16	radix-2^3	radix-2^4	modified radix-2^5
FFT size		512	512	512	2048	512
Data path type		8/16-way	16-way	8-way	8-way	8-way
Area analysis	No. of complex register	512+128	512+672	512	2040	512
	No. of complex adders	112	256	120	152	120
	No. of CBM[a]	16*0.76	32	16	8+8*0.55	8
	No. of CCM[b] for W_8	0	0	24	2	8
	No. of CCM for W_{16}	12	0	0	16	8
	No. of CCM for W_{32}	0	0	0	0	8
	Twiddle factor ROM size	0	-	20480 bits (with 10 bits)	-	10240 bits (with 10 bits)
	Reorder buffer needed	No	No	yes	yes	yes
wordlength (bits)		12	12	-	9	14
SQNR (dB)		57	-	-	32.8	41
Process (μm)		0.09	0.09	-	0.09	.09
Area (mm^2)		0.93	2.46	-	1.16	0.9
Power (mW)		42 @324 MHz	103.5 @324MHz	-	159 @300MHz	-
Throughput rate (R : clock rate)		8R	8R	8R	8R	8R

a: CBM is the abbreviation of complex booth multiplier; b: CCM is the abbreviation of complex constant multiplier

OFDM or single-carrier baseband receivers, generally FFT processors are followed by frequency-domain equalization operations. For very high-throughput applications like 802.15.3c systems, in addition to continuous-flow operation, parallel and normal-order outputs from FFT to equalizers are necessary. By adding another size-N memory block, the proposed FFT processor can support continuous-flow operations, in addition to providing parallel and normal-order output. Due to the high input/output capability, only $2N$ memory size is needed during continuous-flow operation. The timing diagram of the proposed FFT processor's operation scheduling is shown in Fig. 10. At beginning, the FFT input data from previous stage is stored in memory group A. Since the proposed design provides 16-way parallel data input/output, it takes only 32 clock cycles to load the data into memory. Then FFT operation is performed on Symbol 1 within 72 clock cycles. When FFT operation is done, FFT output data is sent out to next stage within 32 clock cycles. Similar process for memory group B is shown in the lower half of Fig. 10. Hence, the proposed memory addressing scheme can also support continuous-flow operations with low memory requirement and without delaying input and output data.

VII. IMPLEMENTATION RESULTS AND DISCUSSIONS

Table VII shows the performance comparisons of the proposed FFT processor and several FFT processors [13], [19], [23] for 802.15.3c application, plus one 8-parallel MDF architecture modified from the design in [12]. The design in [19] is also a memory-based architecture based on radix-16 FFT algortihm. However, since it is not well designed, it requires much more complex multipliers, complex adders, and complex registers than the proposed design. Hence, the area and power performances are significantly improved in the current design. The design in [13] is a pipelined 8-parallel MDF architecture based on radix-2^4 algorithm. Although it is claimed for 802.15.3c application, it is specifically designed for 2048-point FFT, but not optimized in terms of area and power for 512-point FFT required by 802.15.3c systems. The design in [23] based on a modified radix-2^5 algorithm is also a pipelined 8-parallel MDF architecture. Table VII shows the area analysis for the compared designs. Since the design in [13] is not optimized for 512-point FFT operations so that it is hard to do a fair comparison with the proposed design. As shown in Table VII, the design in [23] can reduce the area cost for complex multipliers, compared to the design in [13], due to its heavy use of constant multipliers. Although the number of complex multipliers of the proposed design is larger than that of [23], it has fewer complex adders, and it does not need the memory space for storing twiddle factors. Moreover, our design implements a shorter wordlength of 12 bits than 14 bits of [23]. By taking all these factors into account, it can be roughly concluded that the two designs have comparable area sizes.

The main differences of the proposed design and the above designs are explained as follows. First, the proposed design provides normal-order FFT output. The feature of normal-order FFT output eliminates the extra re-ordering buffer needed for MDF pipelined architectures. To our best knowledge, currently there are no MDF pipelined FFT processors with normal-order output capability in the literatures. Even in continous-flow mode, the proposed design only requires a size-$2N$ memory space, while MDF pipelined architecures require a memory space of size close to $3N$, including a memory space of size N inside PE, and $2N$ for alternating re-ordering buffer and output buffer. Second, the proposed design has better SQNR performance than those of [13], and [23], as shown in Table VII. The total execution time of one 512-point FFT is 72 clock cycles which meets the specification requirement. The VLSI implementation of the proposed FFT processor with 12-bit wordlength has been accomplished by using UMC 90-nm process. The area of the whole FFT processor is $0.93\ \mathrm{mm}^2$, and the power consumption is 42 mW. The hardware utilization of the proposed design is close to 100%. The layout view of the proposed design is shown in Fig. 11.

VIII. Conclusion

In this work, a new radix-16 FFT processor for 802.15.3c system has been proposed. The proposed architecture achieves high throughput rate by employing several performance-enhancement techniques, including a reformulated radix-16 algorithm realized with multiple memory banks and PEs, a novel conflict-free memory addressing scheme, and a new twiddle factor multiplier structure. Also, high operation speed is obtained by devising an efficient pipelined PE structure, while the new twiddle factor multiplier has low hardware complexity and power consumption. Synthesis results show that the proposed FFT processor can provide up to 2.59 GS/s throughput, while only dissipates 42 mW. The proposed FFT architecture can also be modified and extended to support other longer FFT sizes.

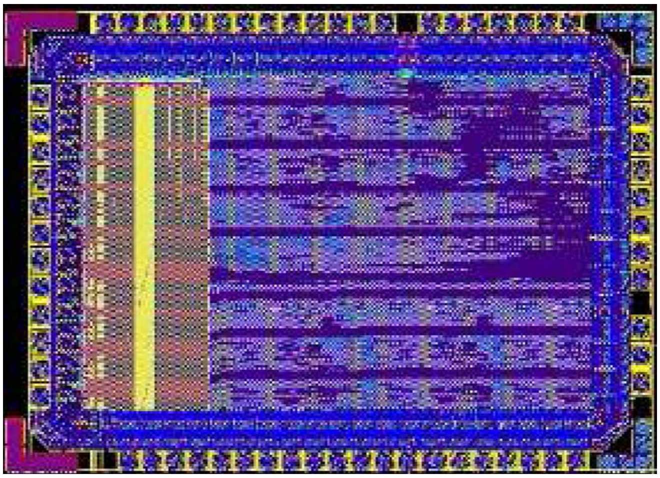


Fig. 11. Layout view of the proposed FFT processor for 802.15.3c.

References

[1] IEEE 802.15.3c-2009: Part 15.3: *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs)*.

[2] S. Johansson, S. He, and P. Nilsson, “Wordlength optimization of a pipelined FFT processor,” in *Proc. 42nd Midwest Symp. Circuits Syst.*., 1999, pp. 501–503.

[3] S. He and M. Torkelson, “Designing pipeline FFT processor for OFDM (de)modulation,” in *Proc. URSI Int. Symp. Signals, Syst., Electron.*, 1998, pp. 257–262.

[4] Y. N. Chang and K. Parhi, “An efficient pipelined FFT architecture,” *IEEE Trans. Circuit Syst. II, Analog Digit. Signal Process.*, vol. 50, no. 6, pp. 322–325, Jun. 2003.

[5] D. Cohen, “Simplified control scheme of FFT hardware,” *IEEE Trans. Signal Process.*, vol. ASSP-24, no. 12, pp. 577–579, Dec. 1976.

[6] L. G. Johnson, “Conflict free memory addressing for dedicated FFT hardware,” *IEEE Trans. Circuit Syst. II, Analog Digit. Signal Process.*, vol. 39, no. 5, pp. 312–316, May 1992.

[7] Y. Ma, “An effective memory addressing scheme for FFT processors,” *IEEE Trans. Signal Process.*, vol. 47, no. 3, pp. 907–911, Mar. 1999.

[8] Y. Ma and L. Wanhammar, “A hardware efficient control of memory addressing for high-performance FFT processors,” *IEEE Trans. Signal Process.*, vol. 48, no. 3, pp. 917–921, Mar. 2000.

[9] B. M. Bass, “A low-power, high-performance, 1024-point FFT processor,” *IEEE J. Solid-State Circuits*, vol. 34, no. 4, pp. 380–387, Mar. 1999.

[10] J. C. Kuo, C. H. Wen, and A. Y. Wu, “Implementation of a programmable 64 $\sim$ 2048-point FFT/IFFT processor for OFDM-based communication systems,” in *Proc. 2003 Int. Symp. Circuit Syst.*, May 2003, pp. 121–124.

[11] Y. W. Lin, H. Y. Liu, and C. Y. Lee, “A dynamic scaling FFT processor for DVB-T applications,” *IEEE J. Solid-State Circuits*, vol. 39, no. 11, pp. 2005–2013, Nov. 2004.

[12] Y. W. Lin, H. Y. Liu, and C. Y. Lee, “A 1-GS/s FFT/IFFT processor for UWB applications,” *IEEE J. Solid-State Circuits*, vol. 40, no. 8, pp. 1726–1735, Aug. 2005.

[13] 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. 6, no. 57, pp. 451–455, Jun. 2010.

[14] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, “An improved radix-16 FFT algorithm,” in *Proc. Can. Conf. Electr. Comput. Eng.*, May 2004, vol. 2, pp. 1089–1092.

[15] D. Takahashi, “A radix-16 FFT algorithm suitable for multiply-add instruction based on goedecker method,” in *Proc. Int. Conf. Multimedia Expo*, 2003, pp. 845–848.

[16] J. Baek and K. Choi, “New address generation scheme for memory-based FFT processor using multiple radix-2 butterflies,” in *Proc. Int. SoC Design Conf.*, 2008, vol. 1, pp. 273–276.

[17] B. G. Jo and M. H. Sunwoo, “New continuous-flow mixed-radix (CFMR) FFT processor using novel in-place strategy,” *IEEE Trans. Circuit Syst. I, Reg. Papers*, vol. 52, no. 5, pp. 911–919, May 2005.

[18] M. Shin and H. Lee, “A high-speed four-parallel radix-24 FFT/IFFT processor for UWB applications,” in *Proc. IEEE Int. Symp. Circuits Syst.*, 2008, pp. 960–963.

[19] S. J. Huang and S.-G. Chen, “A green FFT processor with 2.5-GS/s for IEEE 802.15.3c (WPANs),” in *IEEE Int. Conf. Green Circuit Syst.*, Jun. 2010, pp. 9–13.

[20] P.-Y. Tsai and C.-Y. Lin, “A generalized conflict-free memory addressing scheme for continuous-flow parallel-processing FFT processors with rescheduling,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, no. 99, pp. 1–13, 2010.

[21] F.-C. Yeh, T.-Y. Liu, T.-C. Wei, W.-C. Liu, and S.-J. Jou, “A SC/OFDM dual mode frequency-domain equalizer for 60 GHz multi-Gbps wireless transmission,” *Proc. VLSI Design, Autom. Test*, pp. 1–4, 2011.

[22] F.-L. Yuan, Y.-H. Lin, C.-F. Wu, M.-T. Shiue, and C.-K. Wang, “A 256-Point dataflow scheduling 2 × 2 MIMO FFT FFT/IFFT processor for ieee 802.16 WMAN,” in *Proc. IEEE Asian Solid-State Circuits Conf.*, Nov. 2008, pp. 309–312.

[23] T. Cho and H. Lee, “A high-speed low-complexity modified radix-25 FFT processor for gigabit WPAN applications,” in *Proc. IEEE Int. Symp. Circuits Syst.*, 2011, pp. 1259–1262.

[24] J.-Y. Oh and M.-S. Lim, “Area and power efficient pipeline FFT algorithm,” in *Proc. IEEE Workshop Signal Process. Syst. Des. Implement.*, 2005, pp. 520–525.

Shen-Jui Huang was born in Nantou, Taiwan, in 1972. He received the B.S. degree from the Department of Electrical Engineering, National Central University, Jungli, Taiwan, in 1995, and the M.S. degree from the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, in 1997. He is currently pursuing the Ph.D. degree at the Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan.

He joined the ISSC Corp. in 2000, as a digital design engineer for the Bluetooth/WiFi chip development. In 2009, he joined the Mobile Device Corp. for Wimax chip development. Currently, he is a senior engineer in Mediatek Corp., Hsinchu, Taiwan. His research interests include baseband signal processing, reconfigurable hardware architecture, and encryption/authentication circuit design.

Sau-Gee Chen received the B.S. degree from National Tsing Hua University, Taiwan, in 1978, and the M.S. and Ph.D. degrees in electrical engineering from the State University of New York at Buffalo, NY, in 1984 and 1988, respectively.

Currently, he is a Professor at the Department of Electronics Engineering, and Director of Honors Program, College of Electrical and Computer Engineering/College of Computer Science, National Chiao Tung University, Taiwan. He was Associate Dean, Office of International Affairs, during March–July, 2011, and was the Director of Institute of Electronics from 2003 to 2006, at the same organization. His research interests include digital communication, multi-media computing, digital signal processing, and VLSI signal processing. He has published more than 100 conference and journal papers, and holds 11 US and Taiwan patents.

Prof. Chen served as an Associate Editor of IEEE Transactions on Circuits and Systems—I: Regular Papers during 2004–2006. He was an officer of IEEE Taipei Chapter for two terms.