A High-Flexible Low-Latency Memory-Based FFT Processor for 4G, WLAN, and Future 5G

Shaohan Liu and Dake Liu, *Senior Member, IEEE*

***Abstract*— A high-throughput programmable fast Fourier transform (FFT) processor is designed supporting 16- to 4096-point FFTs and 12- to 2400-point discrete Fourier transforms (DFTs) for 4G, wireless local area network, and future 5G. A 16-path data parallel memory-based architecture is selected as a tradeoff between throughput and cost. To implement a hardware-efficient high-speed processor, several improvements are provided. To maximally reuse the hardware resource, a reconfigurable butterfly unit is proposed to support computing including eight radix-2 in parallel, four radix-3/4 in parallel, two radix-5/8 in parallel, and a radix-16 in one clock cycle. Twiddle factor multipliers using different schemes are optimized and compared, wherein modified coordinate rotation digital computer scheme is finally implemented to minimize the hardware cost while supporting both FFTs and DFTs. An optimized conflict-free data access scheme is also proposed to support multiple butterflies at any radices. The processor is designed as a general IP and can be implemented using a processor synthesizer (application-specific instruction-set processor designer). The electronic design automation synthesis result based on a 65-nm technology shows that the processor area is 1.46 mm^2. The processor supports 972 MS/s 4096-point FFT at 250 MHz with a power consumption of 68.64 mW and a signal-to-quantization-noise ratio of 66.1 dB. The proposed processor has better-normalized throughput per area unit than the state-of-the-art available designs.**

***Index Terms*— Application-specific instruction-set processor designer, coordinate rotation digital computer (CORDIC), domain-specific architecture (DSA), fast Fourier transform (FFT), orthogonal frequency-division multiplexing (OFDM).**

I. Introduction

FAST Fourier transform (FFT) is a compute-intensive algorithm in the physical layer of an orthogonal frequency-division multiplexing (OFDM) system to convert data between time domain and frequency domain. Many OFDM systems such as 4G LTE/LTE-A [1] and wireless local area network (WLAN) [2], [3] require power-of-two FFTs. LTE uplink precoding requires nonpower-of-two discrete Fourier transforms (DFTs) from 12 to 2400. In the upcoming 5G [4] (the fifth generation mobile communication), FFT is still an essential algorithm for all of the waveform candidates, and the FFT computation speed should be high enough to support the high data rate of 5G. Therefore, in the future multimode base station, the FFT processor should support diverse DFTs and high-speed FFTs.

Many high-speed FFT processors [5]–[13] have been proposed for power-of-two FFTs. However, there are a limited number of processors supporting nonpower-of-two DFTs. Processor in [14] adds an extra radix-3 unit to support the 1536-point DFT in 4G LTE. The single-path delay feedback (SDF) architecture in [15] supports 48 2^m3^n points using 6T-RC processing element and section-based twiddle factor (TF) generator (STFG). SDF processor in [16] supports 46 $2^m3^n5^k$ points using a single-table approximation method (STAM) for TF generation. However, the throughput is restricted to $1\times$ clock rate because of the limitation of the single-path pipelined architecture. Processors in [17] and [18] support 128- to 2048-point FFTs and 12- to 1296-point DFTs and use the prime factor algorithm (PFA) to minimize the number of TF multiplication. However, the data access of PFA cannot be in parallel for data I/O (read in input and write out result). The throughput is thus restricted to $1\times$ clock rate and cannot meet the high-speed requirement for future 5G. There are still challenges in designing a low-power high-speed yet flexible processor for DFTs and FFTs.

To design a high-speed processor supporting DFTs and FFTs, several aspects should be considered, including: 1) butterfly unit(s) supporting 2-, 3-, 5-, and higher radices; 2) TF multiplication scheme with hardware efficiency, and 3) conflict-free data access scheme that supports multiple butterfly units for 2-, 3-, 5-, and higher radices as well as minimizes the memory usage for nonpower-of-two DFTs.

In this paper, the contributions are to propose a memory-based FFT processor supporting 54 modes including 16-4096 point FFTs and 12-2400 point DFTs for 4G, WLAN, and future 5G and propose improvements on critical parts in butterfly unit, TF multiplier, and data access scheme. By reusing the hardware resources under timing constraint, we propose a reconfigurable butterfly unit for eight radix-2 in parallel, four radix-3/4 in parallel, two radix-5/8 in parallel, and a radix-16 in one clock cycle. TF multipliers using different schemes are optimized to support both FFTs and DFTs. The proposed coordinate rotation digital computer (CORDIC) scheme is found to be the most hardware efficient scheme for this design. Furthermore, we add bit-reverse operation to the basic add-based data access scheme and support multiple butterfly units at any radix. Compared with the

Manuscript received March 13, 2018; revised August 8, 2018; accepted October 27, 2018. Date of publication November 27, 2018; date of current version February 22, 2019. This work was supported by the National High Technical Research and Development Program of China (863 Program) under Grant 2014AA01A705. *(Corresponding author: Dake Liu.)*

The authors are with the School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China (e-mail: shaohanliu@bit.edu.cn; dake@bit.edu.cn).

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/TVLSI.2018.2879675

TABLE I
SYSTEM PARAMETERS FOR WLAN, 4G, AND FUTURE 5G

Standard	Waveform	FFT sizes	Modulation	Maximum Bandwidth per Stream	Streams
WLAN(802.11n [2]/ac [3])	OFDM	64-512	up to 256QAM	160MHz(512 point)	8
4G LTE/LTE-A [1]	OFDM	128-2048/1536 12-2400	up to 256QAM	30.72MHz(2048 point)	8
Recommended 5G [19]	OFDM	up to 4096	up to 256QAM	122.88MHz	128

state-of-the-art designs, the proposed processor supports the most kinds of transform sizes with the highest normalized throughput per area (Nor.Thrpt/area) (MS/s/mm^2).

This paper is organized as follows. The design considerations are analyzed in Section II. Section III illustrates the mixed-radix algorithm. Section IV provides the architecture of the FFT processor and describes the details of the butterfly unit and the TF multipliers. Section V provides the conflict-free memory access scheme. The implement results are discussed in Section VI. Section VII gives a conclusion.

II. DESIGN CONSIDERATIONS

A. Requirements of 4G, WLAN, and Future 5G

A baseband system should support multiple modes including 2G, 3G, 4G, WLAN, and future 5G [20]. FFT processor is an indispensable component in 4G, WLAN, and future 5G. 4G and WLAN require FFTs from 64 to 2048 and DFTs from 12 to 2400. For future 5G, we use the parameters [19] recommended by China Mobile Communications Corporation. The system parameters are listed in Table I.

The precision requirement can be determined by modulation complexity in Table I plus an extra dynamic range. High quadratic-amplitude modulation requires high data precision.

The requirement on the maximum throughput can be calculated by the maximum bandwidth multiplying the number of streams that one processor handles. In this paper, we assume that one FFT processor supports at most 8-stream 4096-point FFT computations. The requirement on throughput (RqOnThrpt) is computed as follows, where the cyclic prefix samples are not removed. Therefore, the RqOnThrpt is overestimated. The hardware design based on this requirement can provide sufficient performance for 5G

$$\text{RqOnThrpt} = 8 \times 122.88 \text{ MHz} = 983.04 \text{ MS/s}. \quad (1)$$

In actual situation, the latency could be another major constraint for the maximum throughput. From base station implementation perspective, the FFT latency should be negligibly small compared with the user plane latency of 4 ms [19]. In this design, the requirement on latency (RqOnLtcy) for 8-stream 4096-point FFT computations is much smaller than 4 ms and is sufficient for 5G

$$\text{RqOnLtcy} = \frac{8 \times 4096}{\text{RqOnThrpt}} = 33.34 \text{ us} \ll 4 \text{ ms}. \quad (2)$$

B. Design Space Exploration

There are many architectures that support high-throughput FFT processing. However, only SDF [15], [16], MDF [6], and memory-based [17], [18] architectures support diverse nonpower-of-two-point DFTs. The throughput of SDF architecture is restricted to 1 $\times$ clock rate and is hard to meet the requirement in 1. The multipath delay feedback (MDF) design in [6] supports only one DFT size (1536-point), and the supported range of DFT sizes is hard to extend. Therefore, in this paper, the proposed processor is optimized on the basis of the memory-based architecture with general radix-r butterfly unit(s).

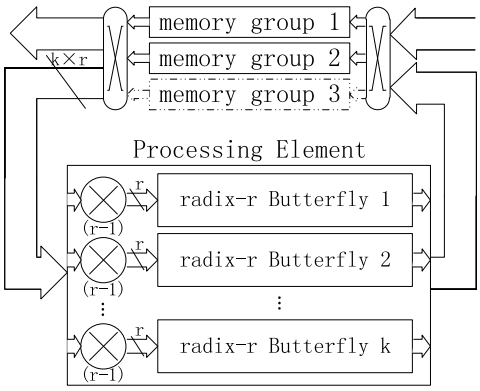


Fig. 1. Architecture of the general memory-based FFT processor.

Compared with pipelined butterfly unit [5], [8], general radix-r butterfly unit is easier to support additional 3- and 5-radix by reusing the adders and multipliers. Therefore, in this design, the reference architecture is set as memory-based architecture with general radix-r butterfly unit(s). The proposed processor is optimized based on the reference architecture.

For an FFT processor that has k radix-r butterfly units in Fig. 1, an N-point FFT can be decomposed into several cascaded radix-r stages. The number of computation stages is

$$\text{No.Stage} = \lceil \log_2 N / \log_2 r \rceil \quad (3)$$

in which $\lceil\rceil$ means the rounding-up operation. The number of clocks in each stage is

$$\text{Clks}_{\text{Stage}} = N/(k \times r). \quad (4)$$

The clock consumption of an N-point FFT should be under the constraint of the throughput requirement as shown in the following equation:

$$\text{Clks}_{\text{Stage}} \times \text{No.Stage} + \text{Clks}_{\text{Overhead}} \leq (N/\text{RqOnThrpt}) \times R \quad (5)$$

where R is the clock frequency and is set to 250 MHz in this design. $\text{Clks}_{\text{Overhead}}$ is the noncomputing clocks including possible I/O cycles (regardless of hazard overhead).

TABLE II
HARDWARE COST OF ARCHITECTURE CANDIDATES

	Continuous flow	r	k	R2	Mul	Mem	Cost
1	Y	8	2	24	4+14	3×4096	2640
2		16	1	32	8+15		2970
3	N	8	4	48	8+28	$\mathbf{2 \times 4096}$	3280
4		**16**	**1**	**32**	**8+15**		**2470**
5		32	1	80	28+31		4750

1) Continuous-Flow Processor: For continuous-flow FFT processor that requires no I/O cycles, the memory size is $3 \times N$ (single port), which corresponds to three memory groups in Fig. 1. Using $N = 4096$ and (1), (5) can be simplified as follows:

$$\frac{4096}{k \times r} \times \left\lceil \frac{\log_2 4096}{\log_2 r} \right\rceil + 0 \le \frac{4096 \times 250 \text{ MHz}}{983.04 \text{ MS/s}}. \quad (6)$$

2) Noncontinuous-Flow Processor: For noncontinuous-flow FFT processors, an extra I/O stage is needed between two adjacent FFTs for reading input of the next FFT and writing result of the previous FFT. The memory size is $2 \times N$ (single port), which corresponds to two memory groups in Fig. 1. Using $N = 4096$ and (1), (5) can be simplified as follows:

$$\frac{4096}{k \times r} \times \left\lceil \frac{\log_2 4096}{\log_2 r} \right\rceil + \frac{4096}{k \times r} \le \frac{4096 \times 250 \text{ MHz}}{983.04 \text{ MS/s}}. \quad (7)$$

Based on (6) and (7), the candidate architectures are listed in Table II. Parameter k is power-of-two considering the feasibility of conflict-free data access scheme. R2 means radix-2 units. Mul means the complex multipliers, including nontrivial multipliers and TF multipliers. Nontrivial multipliers refer to the multipliers of which the multiplicands are not -1 or $\pm j$ in the butterfly unit. As a coarse estimate, the hardware cost is calculated as follows:

$$\text{Cost} = a \times \text{R2} + b \times \text{Mul} + c \times \frac{\text{Mem}}{4096} \quad (8)$$

where a, b, and c are the weight parameters of R2, Mul, and Mem. In this design, the values of these parameters are obtained according to the hardware cost of the radix-2 unit, complex multiplier, and 4096 $\times$ (2 $\times$ bitwidth) single-port static random-access memory. a, b, and c are finally set to 10, 50, and 500 as the silicon cost factors.

According to Table II, the memory-based architecture with a radix-16 butterfly unit (the fourth candidate) is the most hardware efficient architecture. Optimizations are made in Sections III and IV to support additional DFTs, reduce hardware cost, and promote data path utilization.

III. MIXED-RADIX ALGORITHM

The DFT is defined as

$$\begin{cases} X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} \\ W_N^{nk} = \exp\left(-j \frac{2\pi nk}{N}\right) \end{cases} \quad (9)$$

where $x[n]$ and $X[k]$ are the input and output sequences, and the DFT size is N. If N is not a prime number, the original DFT can be decomposed into cascaded smaller size radices using the mixed-radix algorithm. Suppose that an N-point DFT can be decomposed into S-cascaded small-size DFT stages as shown in the following equation:

$$\begin{cases} N = \prod_{j=1}^{S} N_j \\ n = \sum_{m=1}^{S-1} \left(\prod_{n=m+1}^{S} (N_n) n_m \right) + n_S \\ k = k_1 + \sum_{m=2}^{S} \left(\prod_{n=1}^{m-1} (N_n) k_m \right) \end{cases} \quad (10)$$

where $N_i, i = 1, 2, \ldots S$ is the radix number of the ith stage, and $n_i \in [0, 1, \ldots, N_i - 1)$ and $k_i \in [0, 1, \ldots, N_i - 1)$ are the corresponding sequence numbers of the input and output data in the ith stage, $i = 1, 2, \ldots S$. In this design, we factorize DFTs to radix-2, -3, -4, -5, -8, and -16 points. There are

$$N_i \in \{2, 3, 4, 5, 8, 16\}. \quad (11)$$

In the ith stage, the radix-N_i butterfly is calculated as follows:

$$x_i[k_i] = \begin{cases} \sum_{n_i=0}^{N_i-1} x[n_i] W_{N_i}^{n_i k_i} & (i = 1) \\ \sum_{n_i=0}^{N_i-1} \left(W_{\prod_{j=1}^{i} N_j}^{\left(k_1 + \sum_{m=2}^{i-1} \left(\prod_{n=1}^{m-1} (N_n) k_m \right)\right) n_i} x_{i-1}[n_i] W_{N_i}^{n_i k_i} \right) & (\text{else}). \end{cases} \quad (12)$$

IV. PROCESSOR ARCHITECTURE

In Section II, the memory-based architecture with a radix-16 butterfly unit is chosen as the reference architecture. In this section, the reference architecture is modified and optimized to support DFT/FFT sizes with high throughput and reduced hardware cost.

As shown in Fig. 2, the proposed FFT processor consists of: 1) a ping-pong data memory including two 16-bank 28-bit single-port memory groups (28-bit data width including 14-bit real part and 14-bit image part); 2) a block floating point (BFP) [5] memory that stores the exponents of the data in

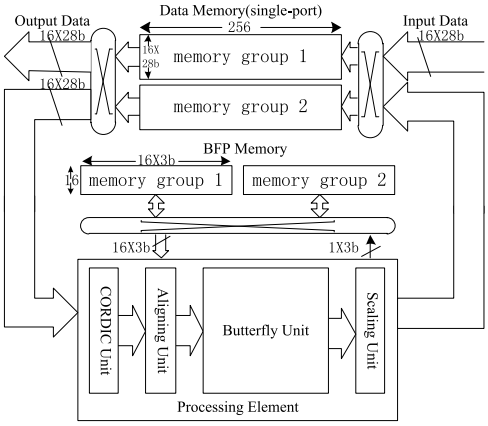


Fig. 2. Structure of the proposed FFT processor (addressing is hidden).

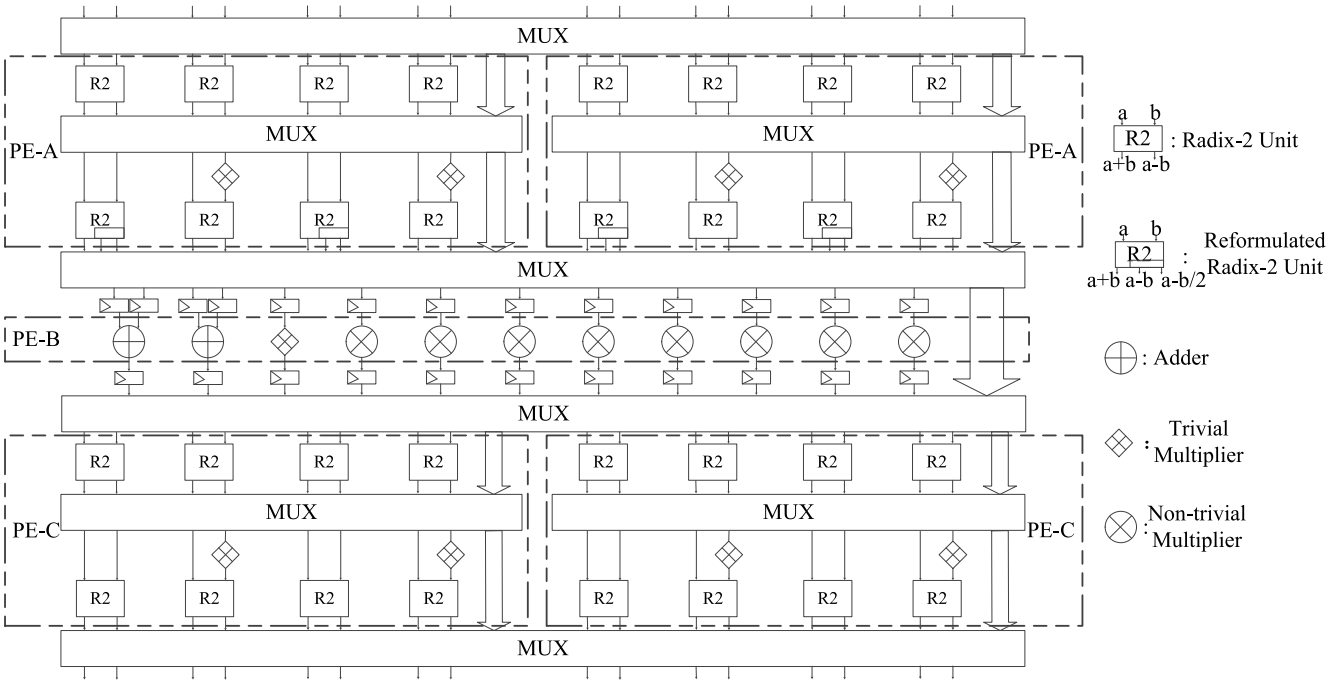


Fig. 3. Butterfly unit in processing element.

BFP format; and 3) a processing element including a CORDIC unit, an aligning unit, a butterfly unit, and a scaling unit. The CORDIC unit conducts TF multiplications. The aligning unit and the scaling unit conduct aligning operations and scaling operations, respectively, for BFP operations.

A. Butterfly Unit

To support different radices, multiple butterfly units have been proposed. A unified butterfly unit in [21] supports radix-2, -3, -4, -5, and -7 butterfly operations by reusing the hardware adders and multipliers. The enhanced delay element matrix unit in [17] supports radix-2, -3, -4, -5, -8, -9, -16, and -25 butterfly operations by using the 2-D DFT factorization method. The high radix small butterfly (HRSB) unit in [18] supports radix-2, -3, -4, -5, -8, -9, -12, -15, -16, and -25 butterfly operations by using a two-stage multipath delay commutator unit.

According to the design space exploration in Section II, the butterfly unit we proposed is based on the radix-16 butterfly unit and is reformulated to support 2-, 3-, 4-, 5-, 8-, and 16-point radices. As shown in Fig. 3, the butterfly unit contains two processing element (PE)-A units, a PE-B unit, two PE-C units, and several switch networks. The butterfly unit supports one radix-16, two radix-5/8 in parallel, four radix-3/4 in parallel, or eight radix-2 operations in parallel.

In order to maximally reuse the hardware resources, the radix-16 DFT is divided into two cascaded radix-4 DFTs as shown in the following equation:

$$X_{16}(k_1,k_2)=\sum_{n_2=0}^{3}\underbrace{\left[\underbrace{W_{16}^{n_2k_1}}_{\mathrm{B}}\sum_{n_1=0}^{3}\underbrace{\left(x_{16}(n_1,n_2)W_4^{n_1k_1}\right)}_{\mathrm{A}}W_4^{n_2k_2}\right]}_{\mathrm{C}} \tag{13}$$

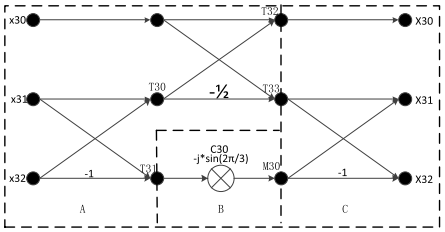


Fig. 4. SFG for three-point DFT algorithm.

where A, B, and C are implemented in the PE-A, PE-B, and PE-C hardware units in Fig. 3 separately.

In a similar way, a radix-8 DFT is reformulated by a radix-4 DFT cascading a radix-2 DFT as shown in the following equation:

$$X_8(k_1,k_2)=\sum_{n_2=0}^{1}\underbrace{\left[\underbrace{W_8^{n_2k_1}}_{\mathrm{B}}\sum_{n_1=0}^{3}\underbrace{\left(x_8(n_1,n_2)W_4^{n_1k_1}\right)}_{\mathrm{A}}W_4^{n_2k_2}\right]}_{\mathrm{C}}. \tag{14}$$

For radix-3 and radix-5, the Winograd Fourier transform algorithm is used to minimize the number of multipliers. The signal flow graphs (SFGs) of radix-3 and radix-5 are divided into three parts as shown in Figs. 4 and 5.

In Fig. 3, a PE-A unit supports two radix-4 or four radix-2 operations for radix-2, -4, -8, and -16 FFTs and also supports one part A operation for the radix-5 DFT or two part A operations for the radix-3 DFT. A PE-C unit supports two radix-4 operations for the radix-16 FFT or four radix-2 operations for the radix-8 FFT and also supports one part C operation for the radix-5 DFT or two part C operations for the radix-3 DFT. The PE-B unit contains all of the nontrivial multipliers in

TABLE III
REUSED MULTIPLIERS IN PE-B

Radix	Mul1	Mul2	Mul3	Mul4	Mul5	Mul6	Mul7	Mul8	Mul9
Radix-16	$W_{16}^{1\times1}$	$W_{16}^{1\times2}$	$W_{16}^{1\times3}$	$W_{16}^{2\times1}$	$-j$	$W_{16}^{2\times3}$	$W_{16}^{3\times1}$	$W_{16}^{3\times2}$	$W_{16}^{3\times3}$
Radix-8	1	1	1	$\left(W_8^1\right)^0$	$(-j)^0$	$\left(W_8^3\right)^0$	$\left(W_8^1\right)^1$	$(-j)^1$	$\left(W_8^3\right)^1$
Radix-3	$C30^0$	1	$C30^1$	1	1	$C30^2$	1	1	$C30^3$
Radix-5	$C53^0$	$C50^0$	$C53^1$	$C52^0$	1	$C51^0$	$C52^1$	$C50^1$	$C51^1$

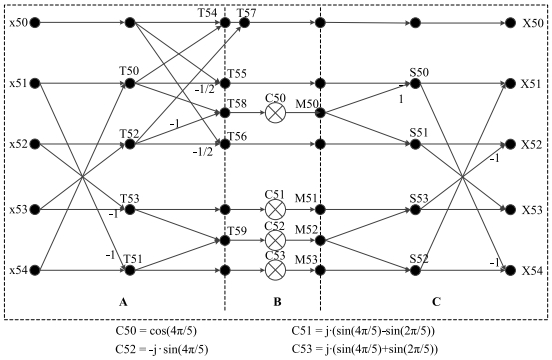


Fig. 5. SFG for five-point DFT algorithm.

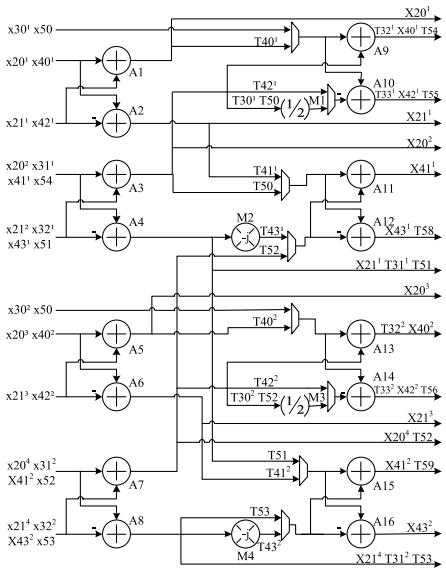


Fig. 6. Hardware implement of PE-A.

radix-3, -5, -8, and -16 FFTs and contains the adders in part B of radix-5 computation.

1) PE-A: The hardware implementation of PE-A unit is shown in Fig. 6. The PE-A unit contains 4 trivial multipliers (M1 to M4) and 16 complex adders (A1 to A16). Trivial multipliers refer to the multipliers of which the multiplicands are 1, 0.5, or $-j$. $x50-x54$ and $T50-T56$, $T58$, $T59$ signals are used for radix-5 computation as shown in Fig. 5. $x30-x32$ and $T30-T33$ signals are used in radix-3 computation as shown in Fig. 4. $x20$, $x21$ and $X20$, $X21$ are signals corresponding to the input and output data in radix-2 computation. $x40-x43$, $T40-T43$, and $X40-X43$ are signals corresponding to the input, intermediate, and output data in radix-4 computation. For the radix-2, -3, or -4 computations, PE-A supports parallel butterfly operations. Superscripts in Fig. 6 indicate the index number of butterflies.

2) PE-B: The PE-B unit contains nine complex multipliers (one trivial multiplier and eight nontrivial multipliers). For part B of radix-5 (Fig. 5), a complex adder is required to compute T57. Therefore, two additional complex adders are required to support two parallel radix-5 operations.

The nontrivial multiplications of different radices are listed in (13), (14), and Figs. 4 and 5. There are 1, 4, 2, and 8 nontrivial multiplications in the radix-3, -5, -8, and -16 butterfly operations. Considering that one radix-16, two radix-5/8, and four radix-3 butterflies are supported in parallel, the number of nontrivial multiplications in radix-3, -5, -8, and -16 butterfly operations is 4, 8, 4, and 8. In order to reuse the nontrivial multipliers in PE-B, canonical signed digit representation and common subexpression sharing methods [5] are used. The reuse scheme is listed in Table III. The first row depicts each multiplier, and each column specifies multiplier functions from Mul1 to Mul9. Signal $C30$ is defined in Fig. 4, and signals $C50-C53$ are defined in Fig. 5. For radix-3, -5, or -8 computation, PE-B supports parallel butterfly operations. Superscripts in Table III indicate the index number of butterflies.

3) PE-C: The hardware implementation of PE-C is shown in Fig. 7. There are 16 complex adders (A1 to A16) and two trivial multipliers (M1 and M2) in one PE-C unit. The naming rule of radix-2 and radix-4 signals is the same way as that in PE-A (Fig. 6). For radix-3 and radix-5, the signals have been defined in part C in Figs. 4 and 5.

4) Summary and Comparison: The proposed butterfly unit consists of 32 pipeline registers, 66 complex adders, 8 nontrivial multipliers, and switch networks. The switch networks in Fig. 3 contain 19 2-1MUXes, 22 3-1MUXes, 15 4-1MUXes, and 5 5-1MUXes. These MUXs are controlled according to (13), (14), Figs. 4 and 5, and Table III.

Table IV compares the proposed butterfly unit with the butterfly units in [17] and [18] and reveals that the proposed unit achieves much lower computation cycle cost with slightly higher hardware cost. In Table IV, R3, R5, and R16 mean the radix-3, radix-5, and radix-16.

B. Twiddle Factor Multiplier

TF multiplier is an important module in FFT processor and consumes significant hardware resources. Traditional $(1/8)$ symmetry algorithm [22] uses a $(N/8)$ ROM table to store the TFs and a complex multiplier to process the multiplications. The algorithm also supports the FFT points that are the exact

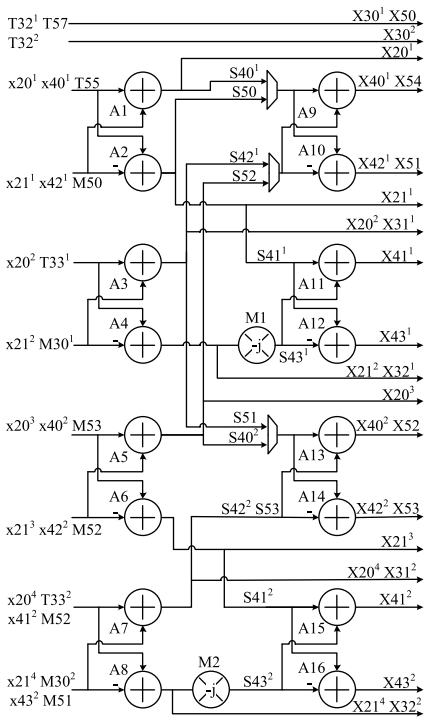


Fig. 7. Hardware implement of PE-C.

TABLE IV
COMPARISON OF BUTTERFLY UNIT

	Reg	Adder	Mul	Radix	Computation Cycles		
					R3	R5	R16
Proposed	32	66	8	2/4/8/16 3/5	0.25*	0.5	1
EDEM [17]	25	34	10	2/4/8/16 3/5/9/25	0.5	1	4
HRSB [18]	20	26	10	2/3/4/5/8 9/12/15 16/25	1	1	4

* 0.25 means 4 radix-3 FFTs are computed in one cycle

TABLE V
11 FFT/DFT POINTS THAT ARE ALIQUANT OF EACH OTHER

4096	2400	2304	2160	1944	1920
1800	1728	1536	1440	1296	-

divisors of N. However, if there is another M-point FFT (M is an aliquant of N), an extra $(M/8)$ ROM table is needed to store the TFs of M-point FFT. Therefore, the total size of ROM table is $((N + M/8))$, and all of the FFT sizes that are the exact divisors of N (or M) are supported. Suppose N and M are divisible by 8.

This design supports 54 different FFT/DFT sizes. Regardless of the sizes that are the exact divisors of another size, there are still 11 sizes of which the TFs should be stored in a ROM table. The 11 sizes ($\{N^1, \ldots N^{11}\}$) are listed in Table V.

To support all the above-mentioned sizes, $(1/8)$ symmetry algorithm requires $\sum_{i=1}^{11} (N^i/8) = 2828$ table entries. The huge ROM table imposes significant hardware cost.

To reduce the ROM size in the TF multiplier, many ROM-based schemes are proposed, such as ROM sharing scheme [23], ROM partition scheme [15], [24], memoryless rotator [5], [25], and trigonometric approximation [6]. CORDIC [26] is another kind of TF multiplier, such as radix-2 memoryless CORDIC [27], mixed scaling rotation-CORDIC [28], [29], P-CORDIC [30], [31], and radix-4 CORDIC [32]. However, only a few of the above-mentioned schemes, including ROM partition scheme and CORDIC scheme, support nonpower-of-two DFT points. In this paper, we design two TF multipliers using the ROM partition scheme and CORDIC scheme separately and select and implement the one with less hardware cost.

1) TF Multiplier Using ROM Partition Scheme: By partitioning the ROM table into two small ROM tables, the ROM partition scheme can significantly reduce the number of table entries at the cost of an extra complex multiplier. For power-of-two FFT sizes, the number of table entries can be reduced from $(N/8)$ to $2\sqrt{(N/8)}$ [24]. For nonpower-of-two DFT points, the number of table entries can be reduced from N to $U + V$ $(N = UV)$ [15].

According to the fact that all of the 11 sizes in Table V are divisible by 8, we integrate the schemes in [24] and [15] and propose a ROM partition scheme that reduces the number of table entries to $(U/8) + V$ $(N = UV)$.

The general form of TF is

$$\mathrm{TF} = W_N^k = \exp\left(-j\frac{2\pi k}{N}\right). \tag{15}$$

Suppose that $N\%8 = 0$ ($\%$ means the remainder operation). We can reformulate (15) in terms of a multiplication of two sublevel TFs as shown in the following equation:

$$\begin{cases} N = U \times V \\ N\%8 = U\%8 = 0 \\ W_N^k = W_N^{Vk_1+k_2} = W_U^{k_1} \times W_N^{k_2} \\ k_1 \in [0, U), \quad k_2 \in [0, V). \end{cases} \tag{16}$$

The first sublevel TF ($W_U^{k_1}$) can be stored in a $(U/8)$ ROM table (coarse ROM) using $(1/8)$ symmetry algorithm [22]. The second sublevel TF ($W_N^{k_2}$) can be stored in a V ROM table (fine ROM).

To support multiple FFT/DFT sizes that are aliquant of each other, the ROM tables should store TFs with a set of (U, V) pairs. There should have a (U, V) pair that satisfies $(U \times V)\%N^i == 0$ for $N^i \in \{N^1, \ldots N^{11}\}$. In this design, we use the following (U, V) pairs:

$$\begin{cases} U = 128 \\ V \in \{12, 18, 32, 135, 243, 225\}. \end{cases} \tag{17}$$

The total number of table entries is $(U/8) + \sum V = 681$.

Fig. 8 shows the hardware architecture of the TF multiplier using the ROM partition scheme.

2) TF Multiplier Using CORDIC Scheme: Instead of loading TFs from the ROM table, CORDIC scheme rotates the input data directly according to the rotation angle in radians $\theta_0 = (2\pi/N) \times k$ [see in particular (15)]. If N is a power of 2, scheme [27] can be used to omit the procedure of

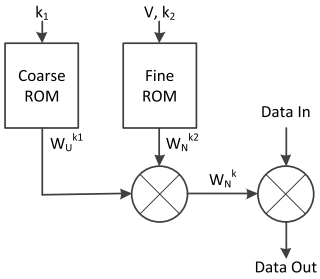


Fig. 8. Hardware architecture of TF multiplier using ROM partition scheme.

multiplying coef $= (2\pi/N)$. However, if N is not power of 2, the multiplication is indispensable

$$\begin{cases} \theta_0 = \text{coef} \times k \\ \text{coef} = \dfrac{2\pi}{N}. \end{cases} \tag{18}$$

To support multiple FFT/DFT points ($\{N^i\}$), a ROM table is necessary to store the values of coef $= (2\pi/N^i)$. In this design, the dynamic range of $\{N^i\}$ is from 12 to 4096.

Suppose that in the view of binary bit field, the rotation angle in radians $[\theta_0 \in [0, 2\pi)]$ is represented using 3 integer bits and $T_\theta - 3$ fractional bits. The mean quantization error below LSB of θ_0 is $\text{biterror}_\theta = (1/2)^{T_\theta - 2}$. Considering that the TF index (k) is an integer that has no quantization error, the mean quantization error of coefficient should be

$$\text{biterror}_c = \left(\frac{1}{2}\right)^{T_C - 2} \le \frac{(1/2)^{T_\theta - 2}}{k} \tag{19}$$

in which T_C is the bitwidth of coefficient (3 integer bits and $T_C - 3$ fractional bits). Formula (19) can be rewritten as follows:

$$\begin{cases} T_C - 2 \ge T_\theta - 2 + \log_2(k) \\ k \in [0, N_{\max}). \end{cases} \tag{20}$$

The value of T_C can be set as the minimal integer of $T_\theta + \log_2(k)$ as shown in the following equation:

$$T_C = T_\theta + \lceil \log_2 N_{\max} \rceil \tag{21}$$

where $\lceil\ \rceil$ means the rounding-up operation. Suppose that $T_\theta = 14$ and $N_{\max} = 4096$. The bitwidth of coefficient should be $14 + 12 = 26$ bits, which results in large ROM size and high hardware cost of the multiplier in (18).

To reduce the bitwidth of coefficient, we can represent the coefficient using floating point. Formula (18) can be changed as follows:

$$\begin{cases} \theta_0 = \text{coef}_{\text{man}} \times k \times 2^{\text{coef}}_{\text{exp}} \\ \text{coef}_{\text{exp}} = \left\lceil \log_2 \left(\dfrac{2\pi}{N}\right) \right\rceil \\ \text{coef}_{\text{man}} = \dfrac{2\pi}{N} / 2^{\text{coef}}_{\text{exp}} \in [0.5, 1)\,. \end{cases} \tag{22}$$

The coef_{man} is the mantissa of coefficient and has 0 integer bit and T'_C fractional bits. The coef_{exp} is an integer and has

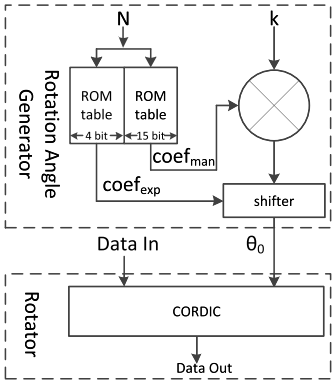


Fig. 9. Hardware architecture of TF multiplier using CORDIC scheme.

no quantization error. Formulas (19)–(21) are recomputed in the following equations, respectively,

$$\left(\frac{1}{2}\right)^{T'_C+1} \times 2^{\text{coef}}_{\text{exp}} \le \frac{(1/2)^{T_\theta - 2}}{k} \tag{23}$$

$$\begin{cases} T'_C \ge T_\theta - 3 + \text{coef}_{\text{exp}} + \log_2 k \\ \text{coef}_{\text{exp}} = \left\lceil \log_2 \left(\dfrac{2\pi}{N}\right) \right\rceil \le \lceil \log_2 2\pi \rceil - \lceil \log_2 N \rceil + 1 \\ 0 \le k < N \end{cases} \tag{24}$$

$$\begin{aligned} T'_C &= \lceil T_\theta - 3 + \lceil \log_2 2\pi \rceil - \lceil \log_2 N \rceil + 1 + \log_2 N \rceil \\ &= T_\theta + 1. \end{aligned} \tag{25}$$

The bitwidth of coef_{man} is $T_\theta + 1$ bits, which is $\lceil \log_2 N_{\max} \rceil - 1$ bits less than the former scheme. Using $T_\theta = 14$, the bitwidth of coef_{man} is 15 bits. According to the dynamic range of $\{N^i\}$, the coef_{exp} ranges from -9 to 0 and can be represented using a 4-bit signal.

The hardware architecture of the TF multiplier using CORDIC scheme is shown in Fig. 9. The coef_{man} and $2^{\text{coef}_{\text{exp}}}$ are stored in ROM tables, respectively. The number of table entries is 54, which equals the number of FFT/DFT sizes. The multiplication with $2^{\text{coef}_{\text{exp}}}$ is implemented using a shifter. The CORDIC unit in Fig. 9 uses the radix-4 CORDIC [32] and contains a seven-stage pipelined microrotation unit and a scaling unit that including two real multipliers (RMs).

3) Comparison: Suppose that a complex multiplier consists of four RMs and two real adders (RA). The hardware cost of the above-mentioned schemes is compared in Table VI using the Semiconductor Manufacturing International Corporation (SMIC) 65-nm CMOS technology.

From Table VI, we can conclude that in this design, the hardware cost of the proposed CORDIC scheme is lower than that of other schemes. The TF multipliers in this design are implemented using the proposed CORDIC scheme.

V. Conflict-Free Parallel Data Access Scheme

Despite the data access schemes for PFA, there are two major kinds of data access schemes for FFT processor: XOR-

TABLE VI
COMPARISON OF DIFFERENT TF MULTIPLIERS

Scheme	RA	RM	Mem	Area (um^2)	Ratio	SQNR (dB)[*1]
$\frac{1}{8}$ symmetry [22]	2	4	2828	67730	100%	82.7
STFG [15]	4	8	793	41548	61.3%	77.9
STAM [16]	2	5	512	27031	39.9%	82.7[*2]
ROM partition[3][24]	4	8	48	24325	35.9	77.9
Proposed ROM partition	4	8	681	35551	52.5%	77.9
Proposed CORDIC	14	3	54	20858	30.8%	80.6

[*1] The word lengths of input and output are set as 14 bits and 15 bits separately.
[*2] The precision (SQNR) may decrease when computing non-power-of-two-point DFTs.
[*3] The ROM partition scheme in [24] only supports power-of-two sizes.

based scheme [33]–[36] and add-based scheme [18], [37]. Both schemes use distributed memory with multiple memory sub-banks. The addressing of XOR-based scheme is implemented using XOR gates, and the addressing of add-based scheme is implemented using adders. The XOR-based addressing circuit requires lower gate count, but higher memory cost for nonpower-of-two-point DFTs. For example, a 1944-point DFT requires 8192-word ($\times 2$) memory in [33]. Traditional add-based scheme requires no extra memory for DFTs, but it does not support multiple butterflies. Furthermore, the address generation scheme in [38] only supports r^n transform sizes and cannot support $2^m 3^n 5^k$ sizes in Table I. In this paper, we add bit-reverse operation to the add-based method and support multiple butterfly units at any radix.

The mixed-radix algorithm is shown in (10). An N-point DFT/FFT is decomposed into S-cascaded stages, of which the radices are $N_1, N_2, \ldots, N_S$. Meanwhile, the index number n is decomposed into $n_1, n_2, \ldots, n_S$. The basic add-based scheme is shown in the following equation, which has been proven in [37]:

$$\begin{cases} \mathbf{bank} = \left(\sum_{j=1}^{S} \mathbf{n_j} \right) \bmod N_1 \\ \mathbf{address} = \sum_{m=2}^{S-1} \left(\prod_{n=m+1}^{S} (N_n) \mathbf{n_m} \right) + \mathbf{n_S} \\ N_1 = R_{\max} \end{cases} \tag{26}$$

where $R_{\max}$ is the maximum radix number that the processor supports, and the number of memory banks equals $R_{\max}$. The variables presented in bold type are vectors, and the data in the same vector are accessed simultaneously. Formula (26) supports only one radix-r butterfly computation. Thus, the vector length equals N_i in the ith stage. The data in the same vector has $n_i \in \{0, 1, \ldots, N_i - 1\}$ and the same $n_j (j \neq i)$. However, a butterfly unit usually supports multiple butterflies in parallel. In this design, a modified add-based scheme is proposed to support multiple butterflies at any radix.

The binary representation of $n_i (1 \leq \mathrm{i} \leq \mathrm{S})$ in (26) is

$$\begin{cases} n_i = d_T d_{T-1} \ldots d_1 \\ T = \log_2 L \geq \log_2 R_{\max} \\ d_j = 0 \left(j > \log_2 N_i \right) \end{cases} \tag{27}$$

where n_i is a T-bit binary number, $d_T d_{T-1} \ldots d_1$ are the binary bits of n_i (d_T is the MSB). L is the data parallelism of the processing element and is a power of two number that is equal to or larger than $R_{\max}$. The bit-reverse operation is defined as follows:

$$n_i^{br} = d_1 d_2 \ldots d_T. \tag{28}$$

Formula (26) can be changed as follows to support multiple butterflies:

$$\begin{cases} \mathbf{bank} = \left(\mathbf{n_1^{br}} + \sum_{j=2}^{S} \mathbf{n_j} \right) \bmod L \\ \mathbf{address} = \prod_{n=3}^{S} (N_n) \mathbf{n_2'} + \sum_{m=3}^{S-1} \left(\prod_{n=m+1}^{S} (N_n) \mathbf{n_m} \right) + \mathbf{n_S} \\ \mathbf{n_2'} = \mathbf{n_2} \gg \left\lfloor \log_2 \frac{L}{N_1} \right\rfloor \end{cases} \tag{29}$$

where $\lfloor\rfloor$ means the rounding-down operation, and $\gg$ is the logic right shift operator. We can define that

$$N_2' = \begin{cases} 1 & (\mathrm{N_1 N_2} \leq L) \\ N_2 \gg \left\lfloor \log_2 \frac{L}{N_1} \right\rfloor & (\text{else}). \end{cases} \tag{30}$$

In this way, the memory size that is used for an N-point DFT/FFT is

$$\text{memorysize} = L \times N_2' \times \prod_{j=3}^{S} N_j \tag{31}$$

which is equal to or slightly larger than N.

In the first stage, if

$$\lceil \log_2 N_1 \rceil == \log_2 L \tag{32}$$

the L-path data parallel processing element supports only one radix-N_1 butterfly computation. Therefore, one radix-N_1 butterfly is computed in each clock cycle. The data accessed simultaneously, called a block of data, have different n_1 and the same $n_2, n_3, \ldots n_S$. The **bank** in (29) can be simplified as follows:

$$\mathbf{bank} = \left(\mathbf{n_1^{br}} + c \right) \bmod L \tag{33}$$

where $\mathbf{n_1^{br}}$ is a $(N_1 \times 1)$-vector that has N_1 different elements, and c is a scalar. Obviously, the N_1 elements in **bank** are different from each other and are stored in different memory banks. Therefore, data can be accessed without conflict.

In the first stage, if

$$\lceil \log_2 N_1 \rceil < \log_2 L \tag{34}$$

for each element of $\mathbf{n_1}$, the MSBs of $\mathbf{n_1}$ is 0.

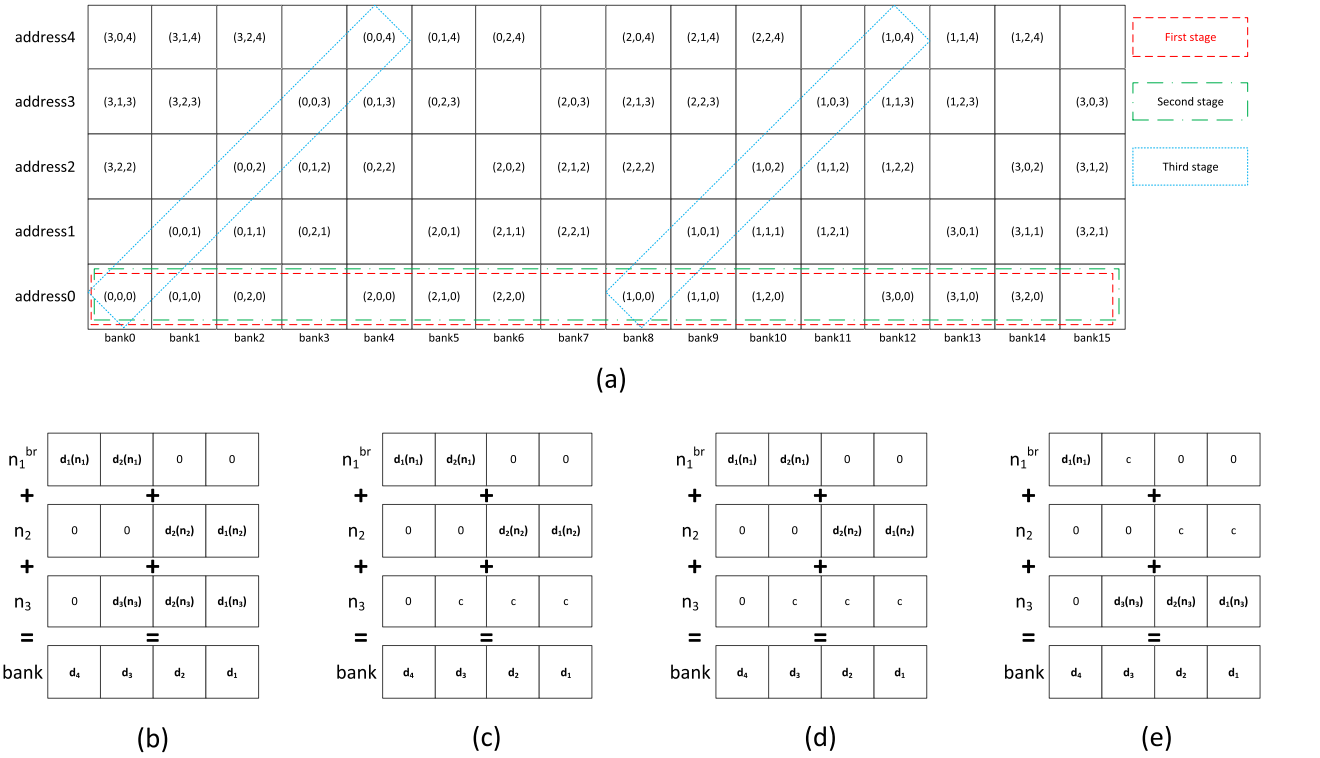


Fig. 10. Data allocation (n_1, n_2, n_3) and index mapping (binary bit field) of a 60-point DFT ($N_1 = 4$, $N_2 = 3$, $N_3 = 5$). (a) Data allocation (n_1, n_2, n_3). (b) Index mapping ($\mathbf{d}_i$ is vector bit and c is scalar bit). (c) Index mapping (first stage). (d) Index mapping (second stage). (e) Index mapping (third stage).

Then, $\mathbf{n_1}$ has

$$\begin{cases} n_1 = d_T d_{T-1} \dots d_1 \\ d_j = 0 (j > M) \\ M = \lceil \log_2 N_1 \rceil. \end{cases} \tag{35}$$

$\mathbf{n_1}$ occupies M LSBs of the total T bits. There are $(T - M)$ bits left. Therefore, it supports 2^{T-M} radix-N_1 butterflies at most.

Define N_2'' and $\mathbf{n_2''}$ as

$$\begin{aligned} N_2'' &= \begin{cases} N_2 (N_2 \le 2^{T-M}) \\ 2^{T-M} (\text{else}) \end{cases} \\ \mathbf{n_2''} &= \begin{cases} \mathbf{n_2} (N_2 \le 2^{T-M}) \\ \mathbf{n_2} \bmod 2^{T-M} (\text{else}). \end{cases} \end{aligned} \tag{36}$$

A data block has different n_1 and n_2'' and the same $n_2', n_3, \dots, n_S$. The **bank** in (29) can be simplified as follows:

$$\mathbf{bank} = \left(\mathbf{n_1^{br}} + \mathbf{n_2''} + c\right) \bmod L \tag{37}$$

where **bank**, $\mathbf{n_1^{br}}$, and $\mathbf{n_2''}$ are $((2^{T-M} \times N_1) \times 1)$-vectors, and c is a scalar. We can see that n_1^{br} and n_2'' have no overlap in binary representation, so that the elements in **bank** are different from each other, and data in the same data block can be accessed without conflict as shown in the following equation:

$$\begin{cases} n_1^{br} = d_1^{n_1} d_2^{n_1} \cdots d_M^{n_1} 0 \cdots 0 \\ n_2'' = 0 \cdots 0 d_{T-M}^{n_2} \cdots d_2^{n_2} d_1^{n_2} \\ n_1^{br} + n_2'' = d_1^{n_1} \cdots d_M^{n_1} d_{T-M}^{n_2} \cdots d_1^{n_2}. \end{cases} \tag{38}$$

Similarly, in the ith stage

$$\begin{cases} \mathbf{bank} = \left(\left(\mathbf{n_1''}\right)^{\mathbf{br}} + \mathbf{n_i} + c\right) \bmod L \\ \mathbf{n_1''} = \begin{cases} \mathbf{n_1} (N_1 N_i \le L) \\ \mathbf{0} (2 \times N_i \ge L) \\ \mathbf{n_1} \bmod 2^{T-M} (\text{else}) \end{cases} \\ M = \lceil \log_2 N_i \rceil \end{cases} \tag{39}$$

where **bank**, $\mathbf{n_1''}$, and $\mathbf{n_i}$ are $((2^{T-M} \times N_i) \times 1)$-vectors, and c is a scalar.

Take a 60-point DFT as an example. Suppose that $L = 16$, $N_1 = 4$, $N_2 = 3$, and $N_3 = 5$. The data allocation (n_1, n_2, n_3) and index mapping of the 60-point DFT are shown in Fig. 10. In the first stage, three radix-4 butterflies are computed simultaneously. Data in the same data block have $n_1 \in \{0, 1, 2, 3, 0, 1, 2, 3, 0, 1, 2, 3\}$, $n_2 \in \{0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2\}$, and a scalar n_3. In the second stage, four radix-3 butterflies are computed simultaneously. Data in the same data block have $n_1 \in \{0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3\}$, $n_2 \in \{0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2\}$, and a scalar n_3. In the third stage, two radix-5 butterflies are computed simultaneously. Data in the same data block have $n_1 \in \{0, 0, 0, 0, 0, 1, 1, 1, 1, 1\}$ (or $n_1 \in \{2, 2, 2, 2, 2, 3, 3, 3, 3, 3\}$), $n_3 \in \{0, 1, 2, 3, 4, 0, 1, 2, 3, 4\}$, and a scalar n_2.

In the data output stage, the data access mode is the same as that in the first computation stage that is discussed earlier. In another word, the data parallelism equals N_1 if the condition in (32) is met and equals $N_1 \times N_2''$ if the condition in (34) is met.

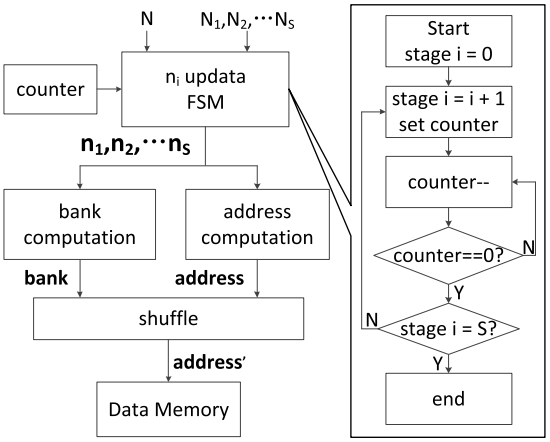


Fig. 11. Hardware implementation of conflict-free parallel data access scheme.

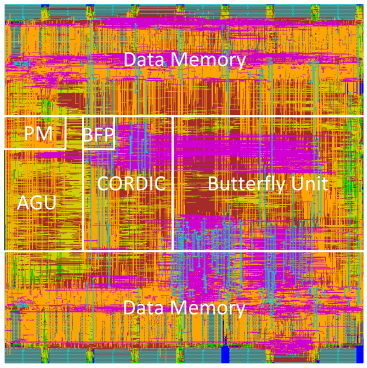


Fig. 12. Layout view of proposed processor.

There are still restrictions in (29) for parallel data input. In the data input stage, the data parallelism supported in (29) is N_S and is restricted when N_S is small (e.g., $N_S = 3$). To solve this problem, the bit-reverse operation in (28) can be used to reverse the bits of n_S in **bank** computation in (29). Using (40), the data access mode in the data input stage is similar to that in the data output stage

$$\begin{cases} \mathbf{bank} = \begin{cases} \left(n_1^{br} + n_2\right) \bmod L & (S = 2) \\ \left(n_1^{br} + \sum_{j=2}^{S-1} n_j + n_S^{br}\right) \bmod L & (S > 2) \end{cases} \\ \mathbf{address} = \prod_{n=3}^{S} (N_n)\mathbf{n}_2' + \sum_{m=3}^{S-1}\left(\prod_{n=m+1}^{S} (N_n)\mathbf{n_m}\right) + \mathbf{n_S} \\ \mathbf{n}_2' = \mathbf{n}_2 \gg \left\lfloor \log_2 \frac{L}{N_1} \right\rfloor. \end{cases} \tag{40}$$

The hardware implementation of the proposed scheme is shown in Fig. 11. The finite-state machine updates the ($\mathbf{n_1}$, $\mathbf{n_2}$, $\cdots$, $\mathbf{n_S}$) in every clock cycle. Then, the **bank** and **address** are generated according to (29). The ith elements in **bank** and **address** are the bank and address of the ith input–output of the processing element. After the shuffle operation, the **address** (in the order of data path) is transformed to **address**′ (in the order of memory banks). The ith element in **address**′ is the address of data stored in the ith memory bank.

VI. Implementation and Discussion

The processor architecture is synthesized using application-specific instruction-set processor (ASIP) designer tool suite [41]. As a retargetable compiler, simulation, and hardware generation environment, the ASIP designer is used to generate software development toolchain and RTL code automatically.

The instruction set of the processor includes six computation instructions that correspond to radix-2, -3, -4, -5, -8, and -16 butterfly operations, two configuration instructions that configure the point and radices of each stage, two I/O instructions, and some other control instructions. The layout view is shown in Fig. 12.

A. Cycle Cost

In order to decrease the hardware cost, $3N \times (\text{bitwidth} \times 2)$ single-port data memory is used in this design. An additional I/O stage is built between two adjacent FFTs.

The cycle costs for DFTs and FFTs are given in Table VII. Mem is the memory size, and CC is the number of computation cycles. IOC is the number of I/O cycles. Latency is the total cycle cost considering CC, IOC, and extra cycles considering hazard overhead. To eliminate potential RAW hazard, 10 no operation performed (NOP) cycles are inserted between adjacent two computation stages. Considering the potential RAW hazard between the last computation stage and data output stage, 10 NOP cycles are inserted in the data output stage. Thus, the IOC is defined as follows:

$$\text{IOC} = \max\left(\text{InputCycles}, \text{OutputCycles} + 10\right). \tag{41}$$

For comparison, the parameters of generalized high-radix (GHR) processor [17] and HRSB processor [18] are also listed. The proposed processor supports more transform sizes and has higher throughput.

In Table VII, the throughput of computing 4096-point FFT is ((4096/1054) $\times$ 250 MHz $\approx$ 972 MS/s), which is slightly smaller than the RqOnThrpt (983 MS/s) in (1). However, considering the cyclic prefix samples that are removed before FFT, the throughput is sufficient for the design requirement.

B. Hardware Cost

We have implemented the proposed design in a 65-nm CMOS technology. When computing 4096-point FFT, the power consumption is 68.64 mW during processing and is 30.34 mW during I/O transfer. A summary of the results is given in Tables VIII and IX. Several designs are listed for comparison, including a 64- to 4096-point FFT processor [8], two memory-based DFT processors [17], [18] using PFA algorithm, and two SDF DFT processors [15], [16].

Compared with the high-throughput FFT processor [8] and the 1 $\times$ R throughput DFT processors [15]–[18], the proposed processor supports approximate 4 $\times$ R throughput for

TABLE VII
REQUIRED LATENCY FOR DFT COMPUTATION

size	Proposed					GHR [17]			HRSB [18]	
	Factorization	Mem	CC[*1]	IOC[*2]	Latency[*3]	Factorization	CC[*1]	Latency[*4]	Factorization	CC[*1]
12	4x3	16	2	11	23	3x4	11	22	-	-
144	16x3x3	144	33	19	72	9x16	61	72	-	-
384	16x8x3	384	80	34	134	16x8x3	256	267	-	-
480	16x3x2x5	480	148	48	226	16x2x3x5	416	427	8x12x5	376
648	8x3x3x3x3	864	288	72	400	9x9x4x2	594	605	-	-
864	16x2x3x3x3	864	348	96	484	16x2x9x3	864	875	9x8x12	792
972	4x3x3x3x3x3	1296	513	108	671	4x9x9x3	1053	1064	9x9x12	972
1080	8x3x3x3x5	1440	504	144	688	8x9x3x5	1026	1037	8x9x15	990
1200	16x3x5x5	1200	439	144	613	16x3x25	740	751	-	-
1296	16x3x3x3x3	1296	549	144	733	16x9x9	1188	1199	-	-
2048	16x16x8	2048	384	138	542	-	-	-	8x8x8x4	2048
4096	16x16x16	4096	768	266	1054	-	-	-	-	-

[*1] CC means the number of computation cycles in an FFT computation.
[*2] IOC means the number of data input and output cycles.
[*3] The latency is the total cycle cost including CC, IOC, and extra cycles caused by RAW hazard.
[*4] The latency in [17] only contains CC and extra cycles. The IOC is hidden because of continuous-flow operation.

TABLE VIII
HARDWARE COMPARISON WITH OTHER MEMORY-BASED PROCESSORS

Architecture		Radix	Throughput	RA[*1]	RM[*1]	Mem size	Mem bank
Proposed		2/3/4/5/8/16	3.89xR(4096) 1.95xR(1200)	358	45	2N(Single-Port)	16
TVLSI2015 [17]		2/3/4/5/8/9 16/25	1xR	108	44	2N(Dual-Port)	7
TVLSI2017 [18]	FFT core	2^3	1xR	32	24	2N(Dual-Port)	4
	DFT core	2/3/4/5/8/9 12/15/16/25	1xR	72	56	3N(Dual-Port)	5
ITCAS2014 [8]		$2/2^6$	1.9xR(4096)	112+24x42%[*2]	32+48x42%[*2]	2N(Single-Port)	4[*3]

[*1] RA and RM are real adder and real multiplier separately. Suppose that a complex multiplier consists of 4 RMs and 2 RAs.
[*2] The constant multiplier in [8] accounts for 42% of a general complex multiplier.
[*3] The main memory in [8] is the 4-bank single-port SRAM with four-word data width.

TABLE IX
COMPARISON OF FFT PROCESSORS

	Proposed	ITCAS2014 [8]	TVLSI2015 [17]	TVLSI2017 [18]		ITCAS2017 [15]	ITCAS2018 [16]
Architecture	Memory-based	Memory-based	Memory-based	Memory-based		SDF	SDF
Technology	SMIC 65nm	TSMC 180nm	SMIC 180nm	SMIC 55nm		TSMC 90nm	TSMC 40nm
Vdd	1.2V	1.8V	1.8V	1.08V		-[*1]	0.99V
Size	16-4096 12-2400	64-4096	128-2048 12-1296	12-1296	128-2048	2-2187 (2^m3^n)	4-2048 ($2^m3^n5^k$)
Frequency	250MHz	80MHz	122.88MHz	122.88MHz		188.67MHz	500MHz
Bit Width	14bit	12bit	16bit	16bit		16bit	14bit
Precision (SQNR)	66.1dB(4096) 64.3dB(1200)	48.4dB(2048)	23dB (25dB input)	-		-	35.84dB(2048) 44.03dB(1200)
Area	1.21x1.21mm^2	4.8mm^2	5x5mm^2	1.063mm^2	0.615mm^2	1.664mm^2	0.6x0.6mm^2
Power	68.64mW	156.2mW	320mW	26.3mW	19.2mW	35.2mW	48.46mW
Throughput (R:clock rate)	3.89xR(4096) 1.95xR(1200)	2xR(4096)	1xR	1xR		1xR	1xR
Nor.Thrpt/area[*2] (MS/s/mm^2)	2.65xR(4096) 1.48xR(1200)	2.61xR(4096)	0.35xR(1200)	0.81xR(1200)	1.39xR(2048)	1.38xR(2^m3^n)	1.05xR($2^m3^n5^k$)
Nor.FFTs/energy[*2] (FFTs/μJ)	4.07(4096) 2.07(1200)	1.27(4096)	0.66(1200)	0.94(1200)	1.28(2048)	2.17(2^m3^n)	1.08($2^m3^n5^k$)

[*1] The voltage is not mentioned in [15]. There we suppose Vdd=1.2V.
[*2] Nor.Thrpt/area [7] (normalized throughput per area) and Nor.FFTs/energy [7], [39], [40] (normalized FFTs per energy unit) shall be the higher the better.

FFT computations and approximate $2 \times R$ throughput for DFT computations with negligible hardware overhead and meets the requirements in Table I.

To compare the area and energy efficiency, Nor.Thrpt/area [7] and normalized FFTs per energy unit (Nor.FFTs/energy) [7], [39], [40] are used to alleviate the influences of different technologies (Tech), voltages (V_{DD}), and word lengths (WL). Nor.Thrpt/area (42) indicates that how many samples are delivered per area unit (MS/s/(mm)^2). Nor.FFTs/energy (43) indicates that how many FFTs are

computed per energy unit (FFTs/μJ)

$$\begin{aligned}&\text{Nor.Thrpt/area}\\&= \left(\frac{\text{Tech}}{65\ \text{nm}}\right)^2\\&\quad\times\left[\frac{2}{3}\left(\frac{WL}{14}\right)+\frac{1}{3}\left(\frac{WL}{14}\right)^2\right]\times\frac{\text{Throughput}}{\text{area}\ (\text{mm}^2)}\end{aligned}\tag{42}$$

$$\begin{aligned}&\text{Nor.FFTs/energy}\\&= \frac{\left(\frac{\text{Tech}}{65\ \text{nm}}\right)\times\left(\frac{V_{\text{DD}}}{1.2}\right)^2\times\left(\frac{2}{3}\left(\frac{WL}{14}\right)+\frac{1}{3}\left(\frac{WL}{14}\right)^2\right)}{\text{energy}\times 10^6}\\&\quad\times\frac{\text{size}}{4096}\end{aligned}\tag{43}$$

where energy is defined as

$$\text{energy}=\begin{cases}\text{Power}\times\text{CC}\times\dfrac{1}{R}(\text{continuous flow processor})\\(\text{Power}_{\text{CC}}\times\text{CC}+\text{Power}_{\text{I/O}}\times\text{IOC})\times\dfrac{1}{R}(\text{else})\end{cases}\tag{44}$$

where R is the clock frequency, CC is the number of computation cycles, and IOC is the number of I/O cycles.

It can be seen from Table IX that the proposed processor supports the most kinds of transform sizes with the highest Signal to Quantization Noise Ratio (SQNR) and performs the best in the Nor.Thrpt/area.

VII. Conclusion

In this paper, a high-speed programmable memory-based FFT processor is proposed to support 12- to 2400-point DFTs and 16- to 4096-point FFTs for 4G, WLAN, and future 5G. By reusing the hardware resources, a 16-path data parallel butterfly unit is proposed, supporting eight radix-2 in parallel, four radix-3/4 in parallel, two radix-5/8 in parallel, and a radix-16 in one clock cycle. A CORDIC scheme is proposed to support both FFTs and DFTs with high hardware efficiency. An optimized add-based data access scheme is also proposed to support multiple butterfly units at any radix. ASIP designer is used to synthesize this design from high level. The processor is synthesized using SMIC 65-nm technology and provides a high throughput of 972 MS/s (4096-point) at 250 MHz with a power consumption of 68.64 mW and an SQNR of 66.1 dB (4096). The Nor.Thrpt/area of our design is better than that of the state-of-the-art available designs.

Acknowledgment

The authors would like to thank Synopsys for their support in the use of ASIP Designer, which is used as a high-level synthesizer.

References

[1] *Evolved Universal Terrestrial Radio Access (E-UTRA); LTE Physical Channels and Modulation*, document 3GPP TS 36.211, 2012.

[2] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput*, IEEE Standard 802.11n-2009, 2009.

[3] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications–Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz*, IEEE Standard 802.11ac-2013, 2013.

[4] *IMT Vision—Framework and Overall Objectives of the Future Development of IMT for 2020 and Beyond*, document M.2083, 2015.

[5] S.-J. Huang and S.-G. Chen, "A high-throughput radix-16 FFT processor with parallel and normal input/output ordering for IEEE 802.15.3c systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 8, pp. 1752–1765, Aug. 2012.

[6] C.-H. Yang, T.-H. Yu, and D. Markovic, "Power and area minimization of reconfigurable FFT processors: A 3GPP-LTE example," *IEEE J. Solid-State Circuits*, vol. 47, no. 3, pp. 757–768, Mar. 2012.

[7] S.-N. Tang, C.-H. Liao, and T.-Y. Chang, "An area- and energy-efficient multimode FFT processor for WPAN/WLAN/WMAN systems," *IEEE J. Solid-State Circuits*, vol. 47, no. 6, pp. 1419–1435, Jun. 2012.

[8] S.-N. Tang, F.-C. Jan, H.-W. Cheng, C.-K. Lin, and G.-Z. Wu, "Multimode memory-based FFT processor for wireless display FD-OCT medical systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 12, pp. 3394–3406, Dec. 2014.

[9] M. Dali, A. Guessoum, R. M. Gibson, A. Amira, and N. Ramzan, "Efficient FPGA implementation of high-throughput mixed radix multipath delay commutator FFT processor for MIMO-OFDM," *Adv. Electr. Comput. Eng.*, vol. 17, no. 1, pp. 27–38, Feb. 2017.

[10] 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, Express Briefs*, vol. 57, no. 6, pp. 451–455, Jun. 2010.

[11] P. Wang, J. McAllister, and Y. Wu, "Software defined FFT architecture for IEEE 802.11ac," in *Proc. IEEE Global Conf. Signal Inf. Process.*, Dec. 2013, pp. 1246–1249.

[12] H. Abdoli, H. Nikmehr, N. Movahedinia, and F. de Dinechin, "Improving energy efficiency of OFDM using adaptive precision reconfigurable FFT," *Circuits Syst. Signal Process.*, vol. 36, no. 7, pp. 2742–2766, Jul. 2017.

[13] T. B. Nguyen and H. Lee, "High-throughput low-complexity mixed-radix FFT processor using a dual-path shared complex constant multiplier," *J. Semicond. Technol. Sci.*, vol. 17, no. 1, pp. 101–109, Feb. 2017.

[14] C. Yu and M.-H. Yen, "Area-efficient 128-to 2048/1536-point pipeline FFT processor for LTE and mobile WiMAX systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 9, pp. 1793–1800, Sep. 2015.

[15] X.-Y. Shih, Y.-Q. Liu, and H.-R. Chou, "48-mode reconfigurable design of SDF FFT hardware architecture using radix-3^2 and radix-2^3 design approaches," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 6, pp. 1456–1467, Jun. 2017.

[16] X.-Y. Shih, H.-R. Chou, and Y.-Q. Liu, "VLSI design and implementation of reconfigurable 46-mode combined-radix-based FFT hardware architecture for 3GPP-LTE applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 1, pp. 118–129, Jan. 2018.

[17] J. Chen, J. Hu, S. Lee, and G. E. Sobelman, "Hardware efficient mixed radix-25/16/9 FFT for LTE systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 2, pp. 221–229, Feb. 2015.

[18] K.-F. Xia, B. Wu, T. Xiong, and T.-C. Ye, "A memory-based FFT processor design with generalized efficient conflict-free address schemes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 6, pp. 1919–1929, Jun. 2017.

[19] *Guideline for 3.5GHz 5G System Prototype and Trial (Version 1.0)*, Mobile World Congress, Barcelona, Spain, 2017.

[20] D. Liu, "Baseband ASIP design for SDR," *China Commun.*, vol. 12, no. 7, pp. 60–72, Jul. 2015.

[21] F. Qureshi, M. Garrido, and O. Gustafsson, "Unified architecture for 2, 3, 4, 5, and 7-point DFTs based on Winograd Fourier transform algorithm," *Electron. Lett.*, vol. 49, no. 5, pp. 348–349, Feb. 2013.

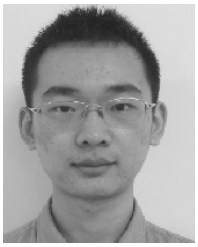
[22] M. Hasan and T. Arslan, "Scheme for reducing size of coefficient memory in FFT processor," *Electron. Lett.*, vol. 38, no. 4, pp. 163–164, Feb. 2002.

[23] P.-C. Jui, C.-L. Wey, and M.-T. Shiue, "Low-cost parallel FFT processors with conflict-free ROM-based twiddle factor generator for DVB-T2 applications," in *Proc. IEEE 56th Int. (MWSCAS)*, Aug. 2013, pp. 1003–1006.

[24] H.-J. Kang, B.-D. Yang, and J.-Y. Lee, "Low complexity twiddle factor multiplication with ROM partitioning in FFT processor," *Electron. Lett.*, vol. 49, no. 9, pp. 589–591, Apr. 2013.

[25] S.-J. Huang and S.-G. Chen, "A new memoryless and low-latency FFT rotator architecture," in *Proc. ISIC*, Dec. 2014, pp. 180–183.

[26] A. M. Despain, "Fourier transform computers using CORDIC iterations," *IEEE Trans. Comput.*, vol. C-23, no. 10, pp. 993–1001, Oct. 1974.
[27] M. Garrido and J. Grajal, "Efficient memoryless CORDIC for FFT computation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, vol. 2, Apr. 2007, pp. II-113–II-116.
[28] C.-H. Lin and A.-Y. Wu, "Mixed-scaling-rotation CORDIC (MSR-CORDIC) algorithm and architecture for high-performance vector rotational DSP applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 11, pp. 2385–2396, Nov. 2005.
[29] S. Y. Park and Y. J. Yu, "Fixed-point analysis and parameter selections of MSR-CORDIC with applications to FFT designs," *IEEE Trans. Signal Process.*, vol. 60, no. 12, pp. 6245–6256, Dec. 2012.
[30] M. Kuhlmann and K. K. Parhi, "P-CORDIC: A precomputation based rotation CORDIC algorithm," *EURASIP J. Appl. Signal Process.*, vol. 2002, no. 1, pp. 936–943, Sep. 2002.
[31] B. Lakshmi and A. S. Dhar, "High speed architectural implementation of CORDIC algorithm," in *Proc. IEEE Region 10 Conf. (TENCON)*, Nov. 2008, pp. 1–5.
[32] E. Antelo, J. Villalba, J. D. Bruguera, and E. L. Zapata, "High performance rotation architectures based on the radix-4 CORDIC algorithm," *IEEE Trans. Comput.*, vol. 46, no. 8, pp. 855–870, Aug. 1997.
[33] S. Richardson D. Marković, A. Danowitz, J. Brunhaver, and M. Horowitz, "Building conflict-free FFT schedules," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 4, pp. 1146–1155, Apr. 2015.
[34] 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.*, vol. 19, no. 12, pp. 2290–2302, Dec. 2011.
[35] Q.-J. Xing, Z.-G. Ma, and Y.-K. Xu, "A novel conflict-free parallel memory access scheme for FFT processors," *IEEE Trans. Circuits Syst. II, Express Briefs*, vol. 64, no. 11, pp. 1347–1351, Nov. 2017.
[36] H.-F. Luo, Y.-J. Liu, and M.-D. Shieh, "Efficient memory-addressing algorithms for FFT processor design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 10, pp. 2162–2172, Oct. 2015.
[37] C.-F. Hsiao, Y. Chen, and C.-Y. Lee, "A generalized mixed-radix algorithm for memory-based FFT processors," *IEEE Trans. Circuits Syst. II, Express Briefs*, vol. 57, no. 1, pp. 26–30, Jan. 2010.
[38] J.-C. Kuo, C.-H. Wen, C.-H. Lin, and A.-Y. Wu, "Vlsi design of a variable-length FFT/IFFT processor for OFDM-based communication systems," *EURASIP J. Appl. Signal Process.*, vol. 2003, pp. 1306–1316, Jan. 2003, doi: 10.1155/S1110865703309060.
[39] B. M. Baas, "A low-power, high-performance, 1024-Point FFT processor," *IEEE J. Solid-State Circuits*, vol. 34, no. 3, pp. 380–387, Mar. 1999.
[40] Y. Chen, Y.-W. Lin, Y.-C. Tsao, and C.-Y. Lee, "A 2.4-Gsample/s DVFS FFT processor for MIMO OFDM communication systems," *IEEE J. Solid-State Circuits*, vol. 43, no. 5, pp. 1260–1273, May 2008.
[41] Synopsys. *ASIP Designer*. Accessed: Apr. 2017. [Online]. Available: https://www.synopsys.com/dw/ipdir.php?ds=asip-designer



Shaohan Liu received the B.S. degree from the Beijing Institute of Technology, Beijing, China, in 2014, where he is currently working toward the Ph.D. degree in information and electronics.

His current research interests include application-specific instruction-set processors design, baseband signal processing, and fast Fourier transform accelerator design.

Dake Liu (SM'08) received the Tech.Dr. degree from Linköping University, Linköping, Sweden, in 1995.

He was involved in the design of communication systems and radio frequency CMOS integrated circuits. He is currently a Professor and the Head of the Institute of Application Specific Instruction Set Processors (ASIP), Beijing Institute of Technology, Beijing, China, and also a Professor at the Computer Engineering Division, Department of Electrical Engineering, Linköping University. He is also the Co-Founder and the Chief Technology Officer of Freehand DSP AB Ltd., Stockholm, Sweden, and the Co-Founder of Coresonic AB, Linköping, which was acquired by MediaTek, Hsinchu, Taiwan. He is a Professor of the China Recruitment Program of Global Experts (1000-plan). He has authored over 200 papers on journals and international conferences. He holds five U.S. patents. His current research interests include high-performance low-power ASIP and integration of on-chip multiprocessors for communications and media digital signal processing.