A Memory-Based FFT Processor Design With Generalized Efficient Conflict-Free Address Schemes

Kai-Feng Xia, Bin Wu, Tao Xiong, and Tian-Chun Ye

***Abstract*— This paper presents the design and implementation of memory-based fast Fourier transform (FFT) processors with generalized efficient, conflict-free address schemes. We unified the conflict-free address schemes of three different FFT lengths, including the single-power points, the common nonsingle-power points, and the nonsingle-power points applied with a prime factor algorithm. Though the three cases differ in terms of decomposition, they are all compatible with memory-based architecture by the way of the proposed address schemes. Moreover, the decomposition algorithm utilizes a method, named high-radix–small-butterfly (HRSB), to decrease the computation cycles and eliminate the complexity of the processing engine. In addition, an efficient index generator, a simplified multipath delay commutator engine, and a unified Winograd Fourier transform algorithm butterfly core were also designed. We designed two FFT examples in long-term evolution system to verify the availability of the address scheme, including a 2^n (128–2048)-point FFT unit and a 35 different point (12–1296) DFT unit. Compared with previous works with similar address schemes, this paper supports more generalized lengths and achieves more flexible throughput.**

***Index Terms*— Conflict-free address scheme, long-term evolution (LTE), memory-based fast Fourier transform (FFT) processor, prime factor algorithm (PFA), Winograd algorithm.**

I. Introduction

THE fast Fourier transform (FFT) is a commonly used algorithm in digital signal processing areas, such as imaging applications and communication systems. Image processing requires computation sizes as high as 2^{22} [1], whereas communication systems apply several computation lengths simultaneously, such as the sizes of 128–2048 in 3GPP long-term evolution (LTE) systems [2]. Consequently, long and variable-size FFT applications are becoming popular.

FFT processors can be categorized into two architectures: pipeline- and memory-based architectures. Memory-based FFTs pass the data multiple times through a single butterfly processing element (PE) or set of PEs, with several memory banks to hold intermediate results, processing the data recursively regardless of computation length. Memory-based FFTs achieve better hardware efficiency, compared with pipeline ones. Nevertheless, the throughput of memory-based FFTs is usually restricted by the butterfly radix and concurrent data access contentions.

Conflict-free address schemes for concurrent data access from different memory banks become an essential problem. A parity bit check method for one or more radix-2 PEs is first introduced in [3] and [4]. An in-place strategy from [5] reduces the total memory storage to minimum N. In [6], a mixed radix-$4/2$ in-place scheme makes the input and output bits symmetric, and then, the conflict-free scheme is extended to a mixed-radix algorithm. In [7], a multiple radix-2 PE scheme is demonstrated to increase the throughput of FFT processors. However, the methods in [6] and [7] are only suitable for power-of-two point FFTs. In [8], a single- and multiple-PE method for arbitrary radix-b algorithm is discussed. However, it requires memory in every stage.

In [9]–[11], a multipath delay commutator (MDC) architecture with high radix is used to replace the complex PE in conventional memory-based FFTs. This method provides low-power dissipation, high data rates, good computational efficiency, and refined length flexibility. However, none of them provides a detailed conflict-free address scheme. In [12], a generalized conflict-free address scheme with single or multiple radix-2^q MDC architectures for 2^n-point FFTs is illustrated. However, this scheme does not extend the principle to arbitrary-length FFTs and more general decomposition algorithms. Generalized mixed-radix algorithms that support both traditional 2^n-point and prime-sized FFTs are proposed in [13] and [14]. Although [13] and [14] obtain the memory bank and address rules according to the decomposition algorithms, they do not state the data distribution procedure clearly, especially when prime factor algorithm (PFA) is applied within the FFT. Moreover, they cannot fully support the continuous-flow working mode.

This paper presents a memory-based FFT processor design methodology with a generalized conflict-free address scheme for arbitrary-length FFTs. We unify the conflict-free address schemes of three different FFT lengths, including the single-power point (SPP) FFTs, the common nonsingle-power point (NSPP) FFTs, and the NSPP FFTs applied with the PFA, to the same address generation format. The memory bank index and the internal address are all generated by

Manuscript received April 22, 2016; revised October 23, 2016, December 7, 2016, and January 21, 2017; accepted January 31, 2017. Date of publication March 3, 2017; date of current version May 22, 2017. This work was supported by the National Major Science and Technology Program of China under Grant 2014ZX03001011-002. This paper was presented at the IEEE International Symposium on Circuits and Systems Proceedings, Montreal, Canada, 2016.

The authors are with the Institute of Microelectronics, Chinese Academy of Sciences, Beijing 100029, China (e-mail: xiakaifeng@ime.ac.cn; wubin@ime.ac.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.2017.2666820

modulo and multiplication operations of the decomposition digits. Moreover, a decomposition algorithm, named high-radix–small-butterfly (HRSB), utilizes high-radix PE to reduce the total number of computation cycles and small butterfly units (BUs) to eliminate the complexity of the computation engine. Thus, the FFTs can complete N-point computations within N cycles to support the continuous-flow mode with low complexity. In the previous methods [13], [14], two successive identical symbols produce two slightly different results, because the factorization changes after each transform. To avoid this problem, we apply the same factorization to every data symbol. Furthermore, an efficient index generator, a simplified configurable MDC unit, and a unified Winograd Fourier transform algorithm (WFTA) butterfly core for point-2, 3, 4, 5 DFTs are designed for the prime-factor FFTs. We designed two FFT examples in LTE systems, including a 2^n-point (128–2048) FFT unit and a DFT unit with 35 different points (12–1296). The techniques proposed previously can be extended to arbitrary-length FFTs.

The remainder of this paper is organized as follows. Section II reviews the background information related to this paper. In Section III, the address schemes for arbitrary-length SPP FFTs, NSPP FFTs, and NSPP FFTs with the PFA are illustrated, respectively. In Section IV, the details of the FFT architectures for LTE systems are presented. The implementation results and the performance comparisons are analyzed in Section V. The conclusion is given in Section VI.

II. Background

A. Methods to Support Continuous-Flow Memory-Based FFTs

Memory-based FFTs usually include three stages: the input sampling, intermediate computations, and the output reordering. The continuous-flow mode should be available only if the computation time is not longer than the input/output time. Fig. 1 is a typical parallel-PE memory-based FFT adopting the in-place strategy, which consists of L parallel radix-r PEs and two P-bank memory groups. Consequently, the continuous-flow memory-based FFT should satisfy

$$\frac{N}{r \cdot L} \cdot \lceil \log_r N \rceil \leq N \tag{1}$$

where $\lceil \log_r N \rceil$ is the recursive computation stages. The total number of parallel paths of the PEs is $P = r \cdot L$. As a result, increasing the radix r or the PE parallelism L represents two basic methods of satisfying (1).

B. Index Mapping Methods of the PFA and CFA

The definition of a discrete Fourier transform (DFT) is given by $C(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}$. In [15] and [16], the method for decomposing one-dimensional DFTs into multidimensional DFTs is demonstrated. If the length N is decomposed into $N_1 N_2$, the index mapping can be expressed as follows:

$$\begin{aligned} n &= \langle K_1 n_1 + K_2 n_2 \rangle_N, \quad n_1, k_1 = 0, 1, \ldots, N_1 - 1 \\ k &= \langle K_3 k_1 + K_4 k_2 \rangle_N, \quad n_2, k_2 = 0, 1, \ldots, N_2 - 1 \end{aligned} \tag{2}$$

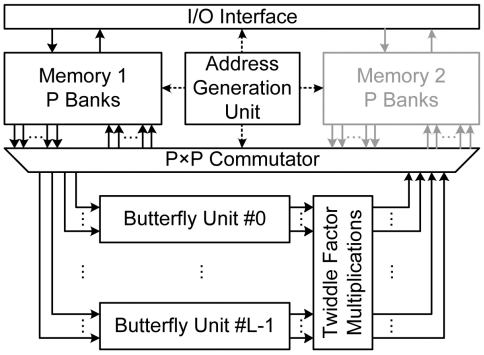

Fig. 1. Architecture of a multiple-PE, memory-based FFT processor [12].

where $\langle \cdot \rangle$ denotes modulo-N operation. If N_1 and N_2 are relatively prime, a Ruritanian map for the input n and a Chinese remainder theorem (CRT) map for the output k are used. Therefore

$$\begin{aligned} K_1 &= N_2; \quad K_3 = N_2 N_2^{-1}, \text{ so } \langle N_2 N_2^{-1} \rangle_{N_1} = 1 \\ K_2 &= N_1; \quad K_4 = N_1 N_1^{-1}, \text{ so } \langle N_1 N_1^{-1} \rangle_{N_2} = 1. \end{aligned} \tag{3}$$

Then, the definition of the DFT can be

$$\begin{aligned} C(k_1, k_2) &= \sum_{n=0}^{N-1} x(n_1, n_2) W_N^{N_1 N_1 N_1^{-1} n_2 k_2} W_N^{N_2 N_2 N_2^{-1} n_1 k_1} \\ &= \sum_{n=0}^{N-1} x(n_1, n_2) W_{N_2}^{N_1 N_1^{-1} n_2 k_2} W_{N_1}^{N_2 N_2^{-1} n_1 k_1} \\ &= \sum_{n_2=0}^{N_2-1} \left\{ \sum_{n_1=0}^{N_1-1} x(n_1, n_2) W_{N_1}^{n_1 k_1} \right\} W_{N_2}^{n_2 k_2}. \end{aligned} \tag{4}$$

This is one of the index mapping methods of the PFA. On the other hand, if N_1 and N_2 are not relatively prime, we can obtain

$$K_1 = N_2, \quad K_2 = 1, \ K_3 = 1, \ K_4 = N_1. \tag{5}$$

Consequently, the DFT is given as

$$\begin{aligned} C(k_1, k_2) &= \sum_{n=0}^{N-1} x(n_1, n_2) W_N^{N_1 n_2 k_2} W_N^{N_2 n_1 k_1} W_N^{n_2 k_1} \\ &= \sum_{n_2=0}^{N_2-1} \left\{ \sum_{n_1=0}^{N_1-1} x(n_1, n_2) W_{N_1}^{n_1 k_1} W_N^{n_2 k_1} \right\} W_{N_2}^{n_2 k_2}. \end{aligned} \tag{6}$$

This is the famous Cooley–Tukey algorithm, which is also called the common-factor algorithm (CFA).

Both the PFA and the CFA decompose the N-point DFT into N_2 DFTs with N_1-point and N_1 DFTs with N_2-point successively. They both utilize an input index map for computation and an output reorder map for natural outputting. The mapping method of the PFA is more complicated than that of the CFA. However, there are twiddle factors $W_N^{n_2 k_1}$ between the two-level DFTs for the CFA but not for the PFA. The mapping methods of (3) and (5) can both be extended to dimensions greater than two.

C. Conflict-Free Address Method Proposed by Tsai and Lin

A generalized conflict-free memory address scheme for memory-based FFTs with parallel arithmetic processing units is presented in [12]. Assume that the $N = 2^m$ FFT employs a mixed-radix algorithm with the maximum radix-2^q MDC units. The FFT is computed in total S stages, where $S = \lceil m/q \rceil$ and $\lceil \cdot \rceil$ is the ceiling function. At stage s, the adopted radix is 2^{q_s}, with $q_s \leq q$, and the number of radix-2^{q_s} BUs at current stage is 2^{m-q_s}. $b_s^{\mathcal{I}}, b_s^{\mathcal{O}}$ denote the different bits of the upper and lower binary indices for the input and output data of the radix-2^{q_s} MDC at stage s. Consequently, $U_s^{\mathcal{M}} = \{b_s^{\mathcal{I}}, b_s^{\mathcal{O}}\}$ represents a set containing joint constraints in stage s for nonconflict data access. Since in-place strategy is applied, address representation of the even symbol for the same memory is in the reversed binary format of the odd symbol. Thus, bit constraints for forward and reversed address representations, $U_s^{\mathcal{F}}$ and $U_s^{\mathcal{R}}$, in one stage should be considered concurrently. EXCLUSIVE OR (XOR) operations of all the decimation constraints from stage 0 to $S-1$ can be used to obtain the memory bank index.

For simplicity, we use 128-point FFT to illustrate. If we apply radix-2^3 algorithm, the 128-point FFT can be factorized into three stages: $2 \times 2^3 \times 2^3$. The forward and reversed constraints in each stage can be given by

$$\begin{aligned} U_0^{\mathcal{F}} &= \{b[6]\}, \quad U_1^{\mathcal{F}} = \{b[5], b[3]\} \\ U_2^{\mathcal{F}} &= \{b[2], b[0]\}, \quad U_0^{\mathcal{R}} = \{b[0]\} \\ U_1^{\mathcal{R}} &= \{b[1], b[3]\}, \quad U_2^{\mathcal{R}} = \{b[4], b[6]\}. \end{aligned} \tag{7}$$

There are three disjoint sets of constraints, i.e., $G_0 = U_0^{\mathcal{F}} \cup U_2^{\mathcal{R}}$, $G_1 = U_0^{\mathcal{R}} \cup U_2^{\mathcal{F}}$, and $G_2 = U_1^{\mathcal{F}} \cup U_1^{\mathcal{R}}$. If there is only one MDC PE, the bank mapping function B is the combined XOR operation of the three constraints

$$B(b) = \text{XOR}\{G_0, G_1, G_2\}. \tag{8}$$

The physical address in the internal memory bank for each index can be derived by eliminating a bit, in the constraint sets, of the binary index. One of the physical address assignments can be $A(b) = \sum_{i=1}^{6} b[i]2^i$.

However, for a computation engine with P parallel MDC PEs, if the constraint that distinguishes the upper and lower paths of a radix-2 butterfly is in one set, the constraints that differentiate the memory banks for different MDC PEs must be in other sets in any stage s. As a result, the 128-point FFT can include at most eight parallel memory banks and four radix-2^3 MDC PEs. Consequently, the bank mapping function $B_P(b)$ for the eight banks is

$$\begin{cases} B_P[d] = \text{XOR}(G_d), \quad d = 0, 1, 2 \\ B_P(b) = \sum_{d=0}^{2} B_P[d]2^d. \end{cases} \tag{9}$$

Correspondingly, the physical addresses in each memory bank can be obtained by dropping $P+1$ bits of the binary index, one from each constraint set. One format of the addresses can be $A_P(b) = \sum_{i=5}^{6} b[i]2^i + \sum_{j=2}^{3} b[j]2^j$.

III. Continuous-Flow Conflict-Free Address Scheme for Different Computation Lengths

In this section, we will propose the continuous-flow conflict-free address schemes for different FFT computation lengths, including SPP FFTs, common NSPP FFTs, and NSPP FFTs applied with PFA. Although these three cases apply different decomposition algorithms, their conflict-free address methods can be unified by a single scheme. The address schemes for the three cases build progressively from one to the next.

A. Address Scheme for Single-Power Point FFTs

The conflict-free address scheme for SPP FFTs presented by Tsai and Lin [12] is illustrated only for 2^n-point FFTs. We can extend the method to arbitrary-length single-power r^n-point FFTs by replacing the XOR operations in (8) and (9) with r-modulo additions. Assume that the $N = r^m$-point FFT employs a mixed radix with the maximum radix-r^q MDC units. The parallelism of the MDC PE is $P = r^p$, and P must divide r^{m-q}, the number of butterflies at each stage. As a result, formation (1) can be transformed into

$$\frac{N}{r \cdot r^p} \cdot \lceil \log_{r^q} r^m \rceil \leq N \rightarrow \left\lceil \frac{m}{q} \right\rceil \leq r^{p+1}. \tag{10}$$

However, as the throughput of SPP FFTs is associated with the PE number closely, it is critical to explore the relationship between the number of constraint sets and the applied algorithm under different cases.

1) Only Increasing the PE Parallelism or the Butterfly Radix: If we only increase the parallelism of the PE P, while keeping the butterfly radix as the simplest radix-r, then $q = 1$, and (10) will become $m \leq r^{p+1}$. For any stage s, the r parallel data indices are only different in the $(m-s-1)$th digit. For the m-level forward address representation, there are m different constraint sets and vice versa

$$U_s^{\mathcal{F}} = d[m-s-1], \quad U_s^{\mathcal{R}} = d[s]. \tag{11}$$

As a result, there are in total m disjoint constraint sets for the radix-r FFT. Since there are r^p different radix-r BUs, it requires at least $p+1$ separate constraint sets, p for MDC units and one for r paths in one MDC, to distribute the data to different memory banks. Because the number of separated constraint sets in this case is m, $p+1 \leq m$ must be satisfied. Therefore, the parallelism p must satisfy the following restriction:

$$\log_r m \leq p+1 \leq m. \tag{12}$$

If we only increase the radix of the BU and keep the PE parallelism to one, then (10) will be $\lceil m/q \rceil \leq r$. Because there is only one PE, the number of constraint sets needed to distribute the data to different memory banks is simply one. Regardless of the value of m, there will be a q to satisfy the above two conditions. For 2^{22}-point FFTs in [1], if radix-2 is employed, the parallelism p must range from 4 to 21 to support the continuous-flow mode, i.e., at least 16 parallel radix-2 PEs must be applied. However, if there is only one BU, the butterfly radix must be at least radix-2^{12}. However, the hardware consumption of the butterfly engine will be huge

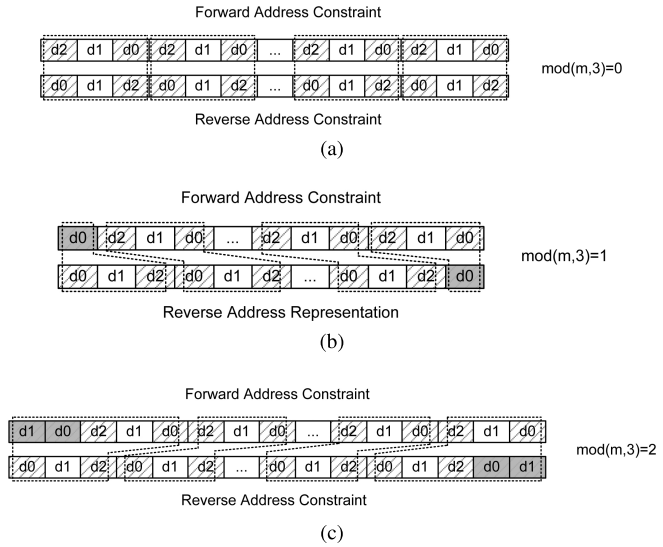

Fig. 2. Forward and reversed address constraint representations for different lengths. (a) $\mathrm{mod}(m,3)=0$. (b) $\mathrm{mod}(m,3)=1$. (c) $\mathrm{mod}(m,3)=2$.

only if the PE parallelism or the butterfly radix becomes excessive.

2) Increasing the PE Parallelism and the Butterfly Radix Simultaneously: The most efficient way to realize (10) is to use the high-radix algorithm as well as the parallel-PE techniques. If there are in total K separate constraint sets for all the stages, there must be $p+1 \le K$. For a hardware-efficient MDC unit, q will appropriately be 3 or 4. We will use $q=3$ for discussion. There are three different m types, including $\mathrm{mod}(m,3)=0$, $\mathrm{mod}(m,3)=1$ and $\mathrm{mod}(m,3)=2$, which have different constraint set numbers.

According to the method in [12], the different digits of the concurrent data for the input and output of the forward address representation of radix-r^3 are d_0 and d_2 in the digit group (d_0,d_1,d_2). d_0 and d_2 denote the address constraint in one stage. The radix-r^3 algorithm is utilized in every stage in the $\mathrm{mod}(m,3)=0$ type. As shown in Fig. 2(a), the digits with a shadow in a dotted block region represent one separate constraint set. The reversed address representation is to reverse the digit position of the forward. As a result, the forward address constraint representation is the same as that of the reversed. Therefore, there are $(m/3)$ different sets for the $\mathrm{mod}(m,3)=0$ type. For the $\mathrm{mod}(m,3)=1$ and $\mathrm{mod}(m,3)=2$ types, the first stage is not radix-r^3 but instead is radix-r or radix-r^2. Thus, the forward and reversed address constraint representations are not the same, as shown in Fig. 2(b) and (c). There are $\lceil (m/3) \rceil$ and $\lceil (m/3) \rceil - 1$ different constraint groups for $\mathrm{mod}(m,3)=1$ and $\mathrm{mod}(m,3)=2$, respectively. Therefore, the parallelism of PE should satisfy

$$\begin{cases} \log_r \lceil \frac{m}{3} \rceil \le p+1 \le \lceil \frac{m}{3} \rceil, & \mathrm{mod}(m,3)=0,1 \\ \log_r \lceil \frac{m}{3} \rceil \le p+1 \le \lceil \frac{m}{3} \rceil - 1, & \mathrm{mod}(m,3)=2. \end{cases} \tag{13}$$

For 2^{22}-point FFTs, the parallelism should satisfy $2 \le p \le 7$ if the radix-2^3 algorithm is employed, i.e., at least four radix-2^3 MDC units will be required.

As a result, the number of constraint sets can be deduced by the specific algorithms and FFT lengths. Because the constraint set number is associated with the PE parallelism, the achievable throughput of the SPP FFT is also decided by the applied algorithms and FFT sizes together.

B. Address Scheme for Nonsingle-Power Point FFTs

For arbitrary NSPP FFTs, a mixed-radix algorithm is usually employed. Assume that the FFT is of size $N = N_1 N_2 N_3$, $N_3 \ge N_1$, $N_3 \ge N_2$, $n_1 = 0,\ldots,N_1-1$, $n_2 = 0,\ldots,N_2-1$, $n_3 = 0,\ldots,N_3-1$. N_1, N_2, and N_3 are common decomposition components. We still apply the Cooley–Tukey algorithm in this decomposition. If the decomposition components are not equal with each other, the forward and reversed address representations are asymmetric. Thus, the in-place strategy will be invalidated and the address method for SPP FFT cannot be used directly. However, the r-modulo operation to generate the memory bank index in SPP FFT can be transplanted into this situation. For the first-level DFT applied radix-N_1, the N_1 parallel data are only different in the first digit n_1 while keeping the other digits the same. It is the same for parallel data in digits n_2 and n_3, when computing in the second and third levels with radix-N_2 and radix-N_3, respectively. Because N_3 is the largest factor, the memory bank number must be set to N_3 or larger. Otherwise, there will be data contentions at the third level. If there is only one BU, similar to the conflict-free address scheme for SPP FFTs in (8), the bank address of the mixed-radix algorithm can be calculated by the N_3-modulo operation of all the digits

$$\mathrm{bank} = \langle n_1 + n_2 + n_3 \rangle_{N_3}. \tag{14}$$

The physical address in each memory bank is found by deleting the digit of n_3 in the digital representation $[n_1,n_2,n_3]$. Thus, the address can be expressed as

$$\mathrm{address} = n_1 N_2 + n_2. \tag{15}$$

The physical bank and address assignment in [13] has the same address format as (14) and (15); however, they are obtained through another method.

The multiple-PE strategy can also be used in the NSPP FFTs. Similar to the SPP FFTs, three independent decimation constraint sets exist for $N_1 N_2 N_3$. If there are P BUs for the first-stage computation, P must divide the current stage's BU number $N_2 N_3$. Because n_1 is used to distinguish the parallel data of radix-N_1 BU, the digits to distribute the data to different BUs must be n_2 or n_3. Take $24 = 2 \times 3 \times 4$-point FFT for example. The memory bank and address are generated by (14) and (15). If there are two radix-2 PEs for the first stage, the memory bank can still be 4. The first binary bit of digit 4 can be used to differentiate the two PEs. However, this rule is only available when the total path number PN_1 is not larger than the bank number N_3. Thus, it is hard to extend multiple-PE bank generation rule in (9) to common NSPP FFTs.

As mentioned in [13], if the in-place strategy is used to minimize the memory storage, a reversed decomposition order of the former symbol should be applied to the latter. If the decomposition of the former symbol is $N_1 \times N_2 \times N_3$, the latter must be $N_3 \times N_2 \times N_1$ to satisfy the in-place strategy. However, this will lead to two slightly different results in the

fixed-point model, just as the decomposition algorithms are different. As a result, a same decomposition manner should be applied to all the data symbols, and the input data $x[i]$ cannot be placed in the location of the output data $X[i]$ of the previous symbol. Therefore, an additional memory group will inevitably be used to reorder the output results. Compared with the SPP FFTs with only $2N$ memory storage in Fig. 1, there must be $3N$ memory units in the NSPP FFTs. Two for the ping-pong switch of the input sampling and one for the output reordering.

In the NSPP FFTs, using high-radix BU is a more efficient way than parallel-PE technique to decrease the computation cycles and to increase the throughput, because the former method has less address controlling complexity than the latter. If the decomposition of a 120-point FFT is $2 \times 3 \times 4 \times 5$, then the total computation time of this method is $60+40+30+24 = 154$ cycles. We can combine some trivial factors together and make the decompositions compact, such as by using $4 \times 5 \times 6$ or $3 \times 5 \times 8$. Then, the computation time should reduce to $30 + 24 + 20 = 74$ cycles and $40 + 24 + 15 = 79$ cycles, respectively. This mechanism is the same as that in SPP FFTs, only increasing the butterfly radix while keeping the BU number to one. We can apply high-radix algorithm for the BU, and small-connected MDC BUs to implement the processing engine. The radix-8 BU can be implemented by a connected radix-2–4 MDC unit, and then, the computation time is $40 + 24 + 30 = 94$ cycles. This is the concept of the proposed decomposition scheme named HRSB that will be introduced in Section IV-B.

C. Address Scheme for Nonsingle-Power Point FFTs Applied With PFA

When PFA is employed to reduce the twiddle factor multiplications, the address scheme for NSPP FFTs in Section III-B will meet some problems. Because the index mapping of PFA is quite different with CFA, the data distribution rule is also distinct. Take $12 = 3 \times 4$-point FFT as an example. When CFA is applied, the decomposition factors n_1 and n_2 of index n can be easily obtained to generated the memory bank and address. The address scheme for 12-point FFT is

$$\text{bank} = \langle n_1 + n_2 \rangle_4, \quad \text{address} = n_1. \tag{16}$$

Fig. 3 is the two-level dataflow graph of 12-point FFT, in which n is the input number index and (b, a) denotes the corresponding memory bank and address. However, if the PFA is applied and data are still stored in the same position as that in the CFA, data contentions will appear in stage 1 as shown in Fig. 3(a). Some data will originate from the same memory bank. Thus, a new data distribution manner should be explored. The input PFA index mapping of 12-point FFT is $n = \langle 4n_1 + 3n_2 \rangle_{12}$. The input index n corresponds to the decomposition components n_1 and n_2 one by one. Hence, n_1 and n_2 can be used to generate the memory bank and address as (16). When the input index n is stored at the position generated by n_1 and n_2, data conflicts in each stage will be avoided as shown in Fig. 3(b). In order to obtain n_1 and n_2 to generate the address scheme when n is known, the inverse mapping rule of $n = \langle 4n_1 + 3n_2 \rangle_{12}$ must be obtained.

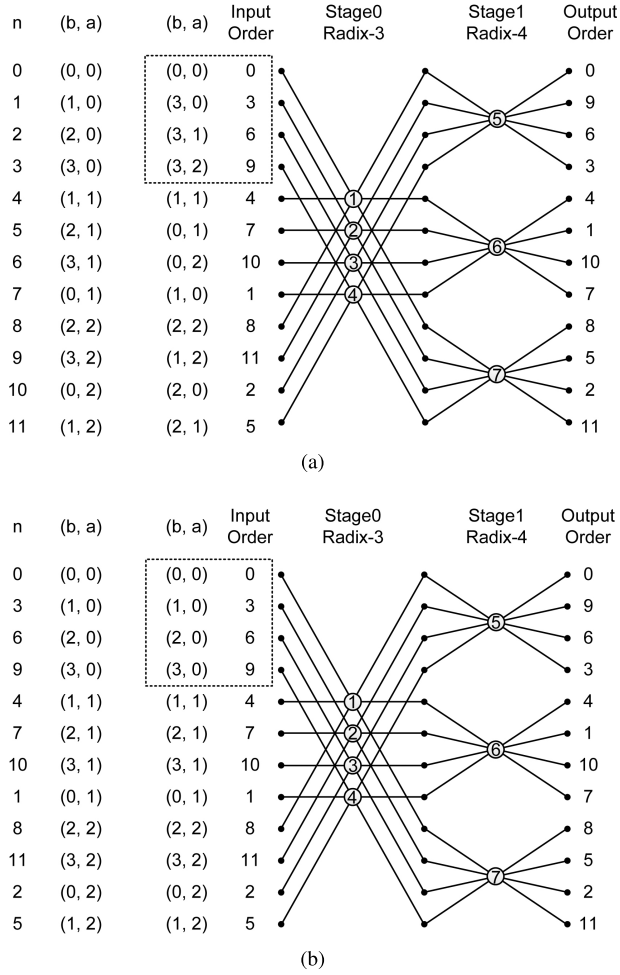

Fig. 3. Dataflow graph of the 12-point DFT combined with the PFA (a) with bank conflict and (b) without bank conflict.

Subsequently, it is essential to explore the common rule of the inverse PFA mapping. For both PFA and Cooley–Tukey algorithms, the multidimensional input and output mappings can be derived from the two-dimensional mappings in Section II-B. Assume that DFT length N is factorized into m factors, $N = N_1 N_2 \ldots N_m$. For CFA, the input and output index mapping is

$$\begin{aligned} n &= \sum_{i=1}^{m-1} n_i (N_{i+1}, \ldots, N_m) + n_m \\ k &= \sum_{i=2}^{m} k_i (N_1, \ldots, N_{i-1}) + k_1. \end{aligned} \tag{17}$$

In (17), $n_i, k_i = 0, \ldots, N_i - 1$. n_i and k_i are the m input and output factors of the decomposition algorithm. For PFA, the m factors are mutually prime, then the input and output index mapping is

$$n = \left\langle \sum_{i=1}^{m} n_i M_i \right\rangle_N \qquad k = \left\langle \sum_{i=1}^{m} k_i a_i M_i \right\rangle_N. \tag{18}$$

In (18), $M_i = N/N_i$, $\langle a_i M_i \rangle_{N_i} = 1$ should be satisfied. M_i and $a_i M_i$ are index mapping parameters for the input and output,

respectively. For simplicity, a_i is usually set to the smallest integer that makes $\langle a_i M_i \rangle_{N_i} = 1$ satisfied. When only CFA is applied, it is easy to obtain the decomposition components n_i by divisions stage by stage, if index n is given. However, for the PFA, it is more complicated. The input and output inverse PFA index mapping rule can be derived in [16] and [17], and they are expressed as

$$n_i = \langle n a_i \rangle_{N_i} \qquad k_i = \langle k \rangle_{N_i}. \tag{19}$$

In (19), the parameter a_i has the same meaning with that in (18). When all the decomposition factors n_i are acquired, the general memory bank indices and addresses are able to generated as that in common NSPP FFTs. Assume that N_m is the largest decomposition component, thus, the memory bank number is set to N_m. The memory bank and address generation rule is

$$\text{bank} = \left\langle \sum_{i=1}^{m} n_i \right\rangle_{N_m} \qquad \text{address} = \sum_{i=1}^{m-1} n_i (N_{i+1}, \ldots, N_{m-1}). \tag{20}$$

This address generation rule can also be adopted by the output index mapping for reordering by changing the decomposition components n_i to k_i. For the input index 1 in 12-point FFT, if CFA is applied, the factors n_1 and n_2 are 0 and 1, respectively. The memory bank and address (b, a) generated by (20) is $(1, 0)$. When PFA is applied, n_1 and n_2 can be obtained by $n_1 = \langle n \rangle_3 = 1$, $n_2 = \langle 3n \rangle_4 = 3$. Consequently, the bank and address (b, a) of index 1 for this case is $(0, 1)$.

However, if there are prime factors and common factors in the decomposition concurrently, both the index mapping rule for CFA and PFA should be applied. For $N = 2 \times 3 \times 4 \times 5$, the factors 3, 4, and 5 are prime with each other, while 2 and 4 are not prime mutually. The PFA index mapping rule cannot be used directly. Thus, we can change the decomposition into $N = 3 \times 5 \times 8$, and 8 can further be decomposed into 2×4 by CFA. That is, the PFA is used between the prime factors, while CFA is applied in the internal nonprime factors. Then, the input and output index mapping is as

$$\begin{aligned} n &= \langle 40n_1 + 24n_2 + 15(4n_3 + n_4) \rangle_{120} \\ k &= \langle 40k_1 + 96k_2 + 105(k_3 + 2k_4) \rangle_{120}. \end{aligned} \tag{21}$$

For the inverse index mapping of (21), the mutually prime factors n_1, n_2, and $n_3' = 4n_3 + n_4$ should be obtained by (19) first, and then, the large prime factor n_3' can be further factorized into small nonprime factors n_3 and n_4 by CFA. All the factorized factors, n_1, n_2, n_3, and n_4, can be used to generated the memory bank and address according to (20). Moreover, when large factor is used in the decomposition, the HRSB scheme can be used to implement the high-radix BU as stated in Section III-B.

IV. FFT Architectures in LTE Systems

There are two different FFT modules in LTE system: one is the 2^n mode ($n = 7$–11) FFT, and the other is the 35 different-length NSPP DFT, whose transform sizes range from 12 to 1296.

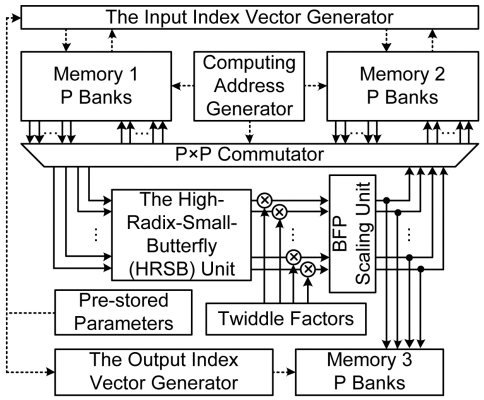

Fig. 4. Block diagram of the proposed FFT processor architecture.

The architectures of the 2^n-point FFT and the NSPP DFT are similar. Only some details are different. The entire FFT processor architecture is shown in Fig. 4. It consists of the following main parts: an input and an output index vector generator, a computation address generator, three different memory bank groups, a PE unit applied with the HRSB scheme, some commutators between the memory and the PE, some prestored twiddle factors, and FFT size parameters, and an exponent scaling unit. The dashed lines represent the control signals while the real lines denote the flowing data. The input index vector generator distributes the input data to different memory banks without data conflicts, and the output one reorders the output data to a natural sequence. The computation address generator obtains all the concurrent data of each cycle and stores back the intermediate results. Memory groups 1 and 2 are in a ping-pong mode to hold two continuous data symbols in input sampling, and memory group 3 is used to output the computed data in right order. Three memory groups only exist in NSPP FFTs. For SPP FFTs, as in-place strategy is available, only two memory groups are enough. The HRSB unit is the kernel processing engine. The commutators located between the memories and HRSB unit provide efficient data routing mechanism which is controlled by the computation address generator. The twiddle factors and the FFT size parameters are all prestored in register files, which are used to configure the FFT working modes. The exponent scaling unit controls scaling operations for block floating-point, which can increase the signal-to-quantization-noise ratio and reduce the memory storage.

The 2^n-point FFT is operated in continuous-flow mode with the in-place strategy using only two parallel radix-$2/2^2/2^3$ MDC units. For instance, to execute a 128-point FFT, only one MDC unit is active in stage 0, and two radix-2^3 MDC units are activated in the remaining stages. The computation can be completed within 128 clock cycles. The input and output index vector generators in the 2^n-point FFT are merged to one. The only difference is that the binary representation of the index for the input is in a forward manner, while it is in a reversed manner for the output. The architectures of the index vector generator and the radix-2^3 MDC unit are detailed in [12].

The length of the NSPP DFT can be generally expressed as

$$N = 2^p 3^q 5^r, \quad p = 2, \ldots, 8; \ q = 1, \ldots, 5; \ q = 1, 2. \quad (22)$$

Because the factors 2, 3, and 5 are mutually prime, [14] simply factorizes N into three different prime parts, namely, 2^p, 3^q, and 5^r using the PFA, and applies the CFA within different parts. However, this factorization method cannot completely satisfy the continuous-flow mode. For example, the 972-point DFT is decomposed into $4 \times 3 \times 9 \times 9$, in [14]. The nine-point DFTs are computed using a radix-3 delay element matrix within three cycles, and radix-4 and radix-3 DFTs can both be computed in one cycle. Therefore, the total computation cycle is $243 + 324 + 324 + 324 = 1215$, which is more than the computation length. Moreover, it suffers a much more complicated data routing scheme.

In this paper, we present a new factorization method that will make the continuous-flow mode completely available and simultaneously ensure the effectiveness of the conflict-free address scheme. The DFT length N is first decomposed into three prime parts as in [14]. If the number of computation cycles is not available to support the continuous-flow mode, some trivial factors are combined together to make the decomposition more compact. The proposed new factorization method consists of the following steps.

i) Obtain each value of p, q, and r.
ii) If $p > 4$, 2^p is decomposed into 2^4 and 2^{p-4}, except that 2^5 is composed of 2^3 and 2^2; otherwise, there is only one level for 2^p. For 3^q, if $q = 5$, there will be three levels, 3^2, 3^2, and 3; 3^2 and 3^{q-2} for $q = 3, 4$; and one level for $q = 1, 2$. And only one level for 5^r.
iii) Calculate the computation time based on the above decomposition. If the decomposition levels for 2^p, 3^q, and 5^r are i, j, and k, the computation time of the complete decomposition is $T = i \cdot (N/4) + j \cdot (N/3) + k \cdot (N/5)$.
iv) If the computation time T is still larger than N, some trivial factors in each part will be combined. However, the 2^4, 3^2, and 5^2 factors should remain unchanged.
v) Calculate the new computation time. If it still does not satisfy the time restriction, step to iv) again.

According to steps 1–3, 972-point FFTs are decomposed into $4 \times 3 \times 9 \times 9$. However, the computation time, as mentioned above, does not satisfy the time requirements. We then combine the trivial factors 4 and 3 together to reduce the computation time further, and obtain the composition $9 \times 9 \times 12$. The high-radix butterfly, such as radix-9 and radix-12, can be implemented by a multistage MDC unit that will make the entire computation time satisfy the time restriction. Although this method will make the PFA incompatible to some lengths, this situation only exists in lengths of 864 and 972.

A. Index Vector Generator

An index vector generator is used to distribute the input data to proper memory positions, and reorder the output data to natural sequence. As shown in Fig. 5(a), the index vector generator can be designed hierarchically. The index counter counts in natural order, and the mutually prime factors n_i of the input index n are obtained using the inverse PFA mapping in (19). According to (19), the inverse PFA mapping methods of the input and output to obtain the prime factors are different. Then, some of the large prime factors should be further decomposed to trivial factors by the CFA. At last, all the trivial factors are used to obtain the memory bank and address according to (20). Take the 1296-point FFT for example. The index counter counts from 0 to 1295, and then the prime factors of each index, ranging from $[0, \ldots, 80]$ and $[0, \ldots, 15]$, are obtained through (19). The value of the prime factor 81 can further be divided into 9×9. The three stage factors, 9, 9, and 16, are all implemented by two level trivial factors 3 or 4. We use four memory banks for this FFT length, and the memory bank and address can be obtained through the trivial factors using the method in (20). Moreover, both the input and output inverse PFA mappings in (19) use N-modulo operations. We can use an $n + 1$-level compare-select tree structure to obtain the last modulo results, as shown in Fig. 5(b). In addition, the configurable information, such as the bank numbers and the parameters of the PFA mapping for different FFT sizes, are prestored in register files.

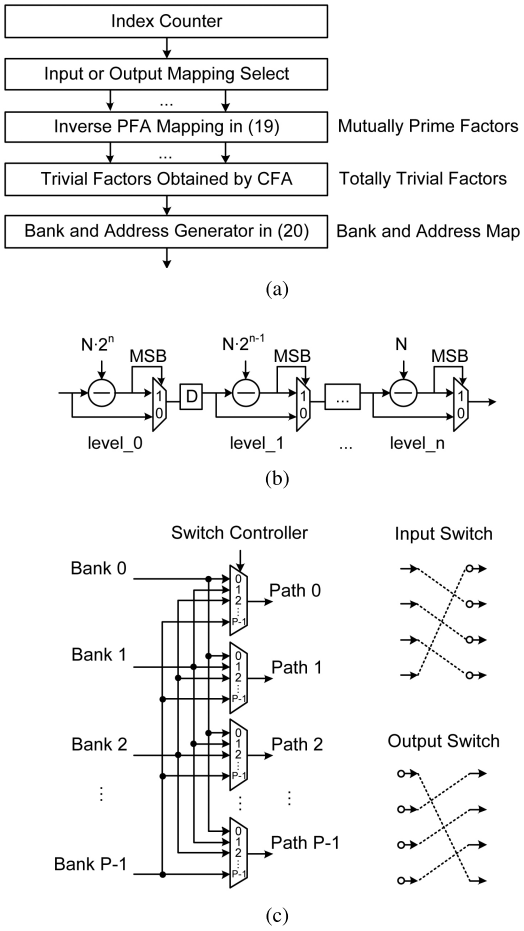

Fig. 5. (a) Index vector generator. (b) $n + 1$-level N-modulo operation. (c) Commutator architecture between the memory banks and PE.

In the computation mode, P concurrent data indices of the PE are obtained according to the computation address generator. Then, the bank indices and memory addresses of

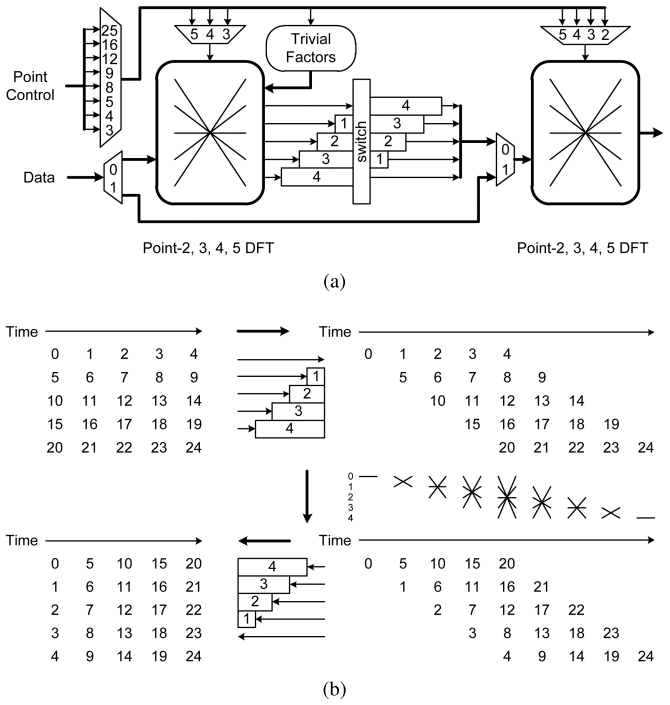

Fig. 6. Proposed HRSB architecture. (a) Architecture of the two-stage MDC unit. (b) Time scheme of the radix-25 BU.

each data are computed through the proposed address scheme in (20). Since the indices are different in one digit n_s of all the decomposition components, only one bank index is needed to be computed while others can be easily derived from it. When data are read out from different memory banks, they switch to the right data paths of the PE through the input commutator as shown in Fig. 5(c). The switching pattern of the commutator is controlled by the bank indices above. When the data have been calculated in the PE, all the data are sent back to the place where they are read out. Thus, the output commutator of the PE has the reversed switching pattern of the input commutator.

B. Butterfly Engine Applys HRSB Scheme

1) HRSB Architecture for Radix-8/9/12/15/16/25 Butterflies: The high-radix butterflies according to the decomposition algorithm, such as radix-8/9/12/15/16/25, are implemented by the HRSB scheme, i.e., a two-stage MDC unit. As shown in Fig. 6(a), the HRSB architecture contains two-stage unified radix-2/3/4/5 butterfly cores, some different-length delay elements, a switch block, trivial twiddle factors between the two butterflies, and some point control multiplexers. The two small BUs are connected in a pipeline manner to construct a one-level computation stage. The time operation of the two-stage radix-25 BU is shown in Fig. 6(b). In the first BU stage, 5 concurrent data are sent into the first BU. After being computed, the data will subsequently be sent to the delay element and switch unit to change the interval of the parallel data for the second BU. Finally, the computed data will return back to the in-place memory bank locations after reordering. Although the time flow graphs of radix-8, 12, 15 BUs are not regular as the radix-9, 16, 25 BUs, the procedures are similar. When radix-8 butterfly is applied in the HRSB architecture, as radix-4 is used in the first stage, there must be two groups of radix-2 data in the second stage to keep up the processing speed.

TABLE I
NUMBER OF COMPUTATION CYCLES OF DIFFERENT BUTTERFLIES

Radix Size	Radix-25 Radix-15	Radix-16,Radix-12	Radix-9
Computation Cycles	5	4	3
Radix Size	Radix-8	Radix-5 Radix-4,Radix-3	[a]Radix-2
Computation Cycles	2	1	0.5

[a]Two groups of radix-2 FFTs are computed in one cycle.

When the computations are completed in the first stage butterfly, it can immediately be calculated by the second stage. Thus, using the HRSB scheme, the radix-8, 9, 12, 15, 16, 25 DFTs can be completed in 2, 3, 4, 5, 4, and 5 cycles, respectively, as shown in Table I.

Moreover, the HRSB architecture can also be configured to bypass the first BU to obtain only radix-2/3/4/5 butterflies. As a result, the HRSB architecture can combine all the required computation butterflies into one PE. For the 972-point FFTs, as the decomposition manner is $9\times9\times12$ and the butterfly engine is implemented by HRSB architecture, the entire computation time is $2\times3\times(972/9)+4\times(972/12)=972$. Thus, the computation cycles of FFTs that apply the HRSB architecture can be acquired by $T=\sum_{i=1}^{m}T_i\cdot N/N_i$, where m is the computation stages, N_i is one stage's butterfly radix, and T_i is the butterfly's computation cycles. Although PFA is applied in DFT, twiddle factors remain between the common-factor components. The twiddle factors are stored in register files to be obtained by the computation indices of the PE. By utilizing the $\pi/8$ symmetry property of trigonometric functions, total 330 twiddle factors are prestored for the DFT unit.

2) Unified WFTA Butterfly Core for Radix-2/3/4/5 Butterflies: In the butterfly engine, the small radix-2/3/4/5 butterflies are implemented by a unified WFTA core. The WFTA for prime-factor DFTs can be written as

$$[X(0),\ldots,X(N-1)]^T=O\times M\times I\times[x(0),\ldots,x(N-1)]^T \tag{23}$$

where I represents the preaddition between the inputs, M is a diagonal matrix of coefficient multiplication, and O is the post-addition of the results after multiplication [16]. In [18], a reconfigurable 2-, 3-, 4-, 5-, 7-point WFTA butterfly core for memory-based FFTs is proposed. The complexity of the butterfly core is equal to that of the 7-point DFT in terms of adders and multipliers plus some multiplexers. To be adopted in our implementation, the butterfly core should complete two radix-2 butterflies or one radix-3/4/5 butterfly within one cycle in reconfigurable mode.

As in [18], we can set the input of the butterfly core to zero and the multiplication coefficients to zero or one to reduce the number of multiplexers. The unified architecture

TABLE II
CONFIGURATION INFORMATION OF THE UNIFIED WFTA CORE

Size	Input Mapping					Output Mapping					Multiplication Coefficients[a]					Control
	y_0	y_1	y_2	y_3	y_4	Y_0	Y_1	Y_2	Y_3	Y_4	C_0	C_1	C_2	C_3	C_4	$s_0s_1s_2s_3$
2^b	0	$x(0)$	$x(1)$	0	0	$X(0)$	$X(1)$	–	$X'(0)$	$X'(1)$	C_{30}	0	$-C_{31}$	0	0	1021
3	$x(0)$	$x(1)$	$x(2)$	0	0	$X(0)$	$X(1)$	$X(2)$	–	–	0	0	-1	0	0	00--
4	0	$x(0)$	$x(2)$	$x(1)$	$x(3)$	$X(0)$	$X(2)$	–	$X(1)$	$X(3)$	0	-1	0	1	C_{41}	1110
5	$x(0)$	$x(2)$	$x(3)$	$x(1)$	$x(4)$	$X(0)$	$X(1)$	$X(4)$	$X(2)$	$X(3)$	C_{50}	C_{51}	C_{52}	C_{53}	C_{54}	0000

[a]The detail multiplication coefficients can be found in [18].
[b]There are two radix-2 input groups for the radix-2 mode: y_1,y_2 and y_5,y_6 are the two-group inputs, and Y_0,Y_1 and Y_3,Y_4 are the two-group outputs.

TABLE III
COMPARISON OF DIFFERENT SINGLE-POWER POINT FFTS

	Algorithm	Parallelism	Process. Cycles (Continuous-flow)	Mem. Size	Mem. Bank	Com. Mul.	Con. Mul.[a]	Com. Add.
Xiao [19]	Radix-2	2	$N(\log_2 N)/2$ (No)	N	2	1	-	2
Luo [20]	Radix-2	2	$N(\log_2 N)/2$ (No)	N	1	1	-	2
Huang [21]	Radix-4^2	16	$N(\log_2 N)/64$ (Yes)	$2N$	16	$16*0.76$	12	112
Garrido [22]	Radix-4	4	$N(\log_2 N)/8$ (No)	N	4	3	-	8
This Work	Radix-2^3	4	$N(\log_2 N)/12$ (Yes)	$2N$	4	4	4	12

[a]The multiplication of trivial twiddle factors such as W_8 and W_{16}.

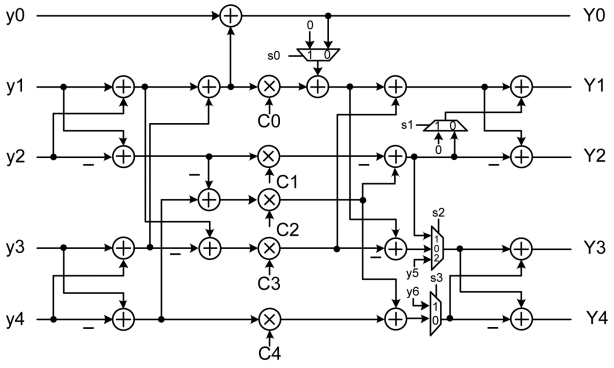

Fig. 7. Proposed unified BU for two-, three-, four-, and five-point DFTs.

is shown in Fig. 7, and is almost the same as the five-point WFTA architecture with only three additional single-gate multiplexers and one 3 : 1 multiplexer. The unified WFTA core has a much simpler architecture than the multi-radix structure in [14], which uses more multiplexers. The input and output relations, the required coefficients and the control signal of the multiplexers are all shown in Table II, where the dash means an irrelevant condition. The middle coefficient multiplications can be implemented by constant multipliers that are constructed of shifters and adders.

V. HARDWARE IMPLEMENTATION RESULTS AND PERFORMANCE COMPARISON

The proposed address scheme can support arbitrary-lenth FFT computations, including SPP FFTs and NSPP FFTs. For SPP FFTs, the proposed address scheme adjusts the decomposition algorithm and the parallelism of PE to modify the throughput. We compare several SPP FFTs that have different conflict-free address schemes with our 2^n-point FFT in Table III. The applied algorithm and data parallelism decide the amount of resources of the architecture and the processing time directly. Radix-2 designs [19], [20] lead to simple conflict-free control mechanisms and less arithmetic units, however, the processing time is always too long to be executed in high-throughput applications. For high-radix designs [21], [22], the processing time is faster, while the arithmetic units are more consumptive. Compared with other FFTs, the radix-2^k algorithm in our design is a tradeoff between throughput and hardware resources. The throughput and the hardware units can be adjusted by the applied algorithm and data parallelism concurrently, no matter using in high or low throughput situations.

For NSPP FFTs, we compare the hardware resources of several DFTs with different address schemes in Table IV. Compared with the schemes in [13], [14], [23], and [24], our method can fully satisfy the continuous-flow mode for all the 35 prime-factor lengths in LTE system. Even the most aggressive competitor [14] is too slow to satisfy a DFT of size 972. The proposed method requires fewer computation cycles than the others. Hence, our FFT owns the highest processing speed. Though our method consumes more memory than [14], it can unify the ultimate results, which reduces the difficulty to evaluate the correctness of the FFTs. There are 13 complex adders, 5 constant multipliers for coefficient multiplications, and 5 complex multipliers for twiddle factor multiplications in the unified butterfly core. Thus, the resources in the PE are double. The PE in [14] occupies almost the same hardware as ours. However, it uses more control multiplexers and buffers, and the controlling mechanism is more complicated. In conclusion, our DFT is the most hardware-efficient one

$$\text{Norm. Area} = \frac{\text{Area}}{(L_{\min}/55\text{nm})^2}$$
$$\text{Norm. Energy} = \frac{\text{Power} \times T_{\text{clock}} \times N_{\text{execute}}}{N_{\text{FFT}}(\text{Volt.}/1.08\text{V})^2(L_{\min}/55\text{nm})}. \quad (24)$$

We implemented the two FFT units in LTE system using the SMIC 55 nm technology. A summary of the results and a performance comparison with [14] are given in Table V. The

TABLE IV
PERFORMANCE COMPARISON OF THE DFTS WITH DIFFERENT ADDRESS SCHEMES

	DFT Size	Altera [23]	Xilinx [24]	Hsiao [13]	Chen [14]	This Work
Computation Cycles	480	2400	1496	616	416	376 ($8 \times 12 \times 5$)
	864	5184	2842	1512	864	792 ($9 \times 8 \times 12$)
	972	5832	3346	1863	1053	972 ($9 \times 9 \times 12$)
	1080	6480	3703	1836	1026	990 ($8 \times 9 \times 15$)
Memory Size[a]		$3N$	$4N$	$2N$	$2N$	$3N$
Throughput Rate		$1/6$	$1/3$	$1/2$	1^b	1
PE	Com. Mul.	-	21	-	10	10
	Com. Add.	-	35	-	34	26
	Con. Mul.	-	0	-	10	10

[a] [13] and [14] cannot obtain a unified fix-point result.
[b]Continuous-flow mode cannot be available in some particular FFT points, such as 972.

TABLE V
IMPLEMENTATION PERFORMANCE AND COMPARISON WITH [14]

	This Work		Chen [14]	
Architecture	Memory-based		Memory-based	
FFT Size	12∼1296	128∼2048	12∼1296	128∼2048
Clock	122.88 MHz		122.88 MHz	
RAM Banks	5	4	7	5
Rate	1×		1×	
Continuous	Yes		No	
Memory-size	$3N$	$2N$	N	N
Wordlength	16 bits		16 bits	
Technology	55 nm@1.08 V		0.18 μm@1.8 V	
Gate Count	340 K	136 K	482 K	316 K
Core Area (Norm.)	1.063 mm^2 (1.063)	0.615 mm^2 (0.615)	5×5 mm^2 (2.3341)	
Core Power (Norm.)	26.3 mW (24.3)	19.2 mW (19.2)	320 mW (35.2)	

normalized area and normalized energy are defined as (24), where N_{execute} is the number of clock cycles required to perform the FFT, N_{FFT} is the FFT size, T_{clock} is the clock period, and L_{min} is the used technology [2]. The two FFT units are both evaluated at the maximum size, i.e., 1296 and 2048 points. This paper includes the same design parameter with [14]. However, the normalized area and power are both smaller. Our method has a simpler address control scheme than [14]. Reference [14] applies seven memory banks for the DFT unit and five banks for the FFT unit, whereas our method only requires five and four banks, respectively. According to [13], more memory banks will lead to greater area overhead and power dissipation. Moreover, the HRSB architecture and the unified WFTA core in the DFT unit are more hardware-efficient than the delay element matrix and the multi-radix structure in [14]. The 128–2048-point FFT is an SPP FFT unit, while the 12–1296-point FFT is an NSPP one. As stated in Section IV, memory consumption is $2N$ in SPP FFTs, and is $3N$ in NSPP FFTs. Actually, [14] uses only N memory amount in the design of two FFT units, which cannot fully support the continuously-flow mode. Although our design uses more memory, it achieves better area and power performance, which further verifies that our method provides greater hardware efficiency. The FFT and DFT units occupy 0.615- and 1.063-mm^2 core area, 19.2- and 26.3-mW power consumption, respectively, at 122.88-MHz clock frequency.

VI. CONCLUSION

We present memory-based FFT implementations with generalized efficient conflict-free address schemes. Address schemes for different FFT lengths are integrated in this paper to support FFT processing for various systems. The memory bank and address can be generated by modulo and multiplication operations of the decomposition digits. For both SPP and NSPP FFTs, high-radix algorithm and parallel-processing technique can be used to increase the throughput. And the address scheme for FFTs applied with PFA is explored. Moreover, a decomposition method, named HRSB, is designed to suit the high-radix algorithm. Full hardware architectures for the FFTs in LTE systems are illustrated, including the index vector generator, the butterfly engine, and the unified WTFA core. The implementation results and comparisons are also presented.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their careful review and precious comments.

REFERENCES

[1] C.-L. Yu, K. Irick, C. Chakrabarti, and V. Narayanan, "Multidimensional DFT IP generator for FPGA platforms," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 4, pp. 755–764, Apr. 2011.
[2] 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.
[3] D. Cohen, "Simplified control of FFT hardware," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 24, no. 6, pp. 577–579, Dec. 1976.
[4] M. C. Pease, "Organization of large scale Fourier processors," *J. ACM*, vol. 16, no. 3, pp. 474–482, Jul. 1969.
[5] L. G. Johnson, "Conflict free memory addressing for dedicated FFT hardware," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 39, no. 5, pp. 312–316, May 1992.
[6] B. G. Jo and M. H. Sunwoo, "New continuous-flow mixed-radix (CFMR) FFT processor using novel in-place strategy," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 5, pp. 911–919, May 2005.
[7] 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.*, vol. 1. Nov. 2008, pp. I-273–I-276.
[8] D. Reisis and N. Vlassopoulos, "Conflict-free parallel memory accessing techniques for FFT architectures," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 11, pp. 3438–3447, Dec. 2008.
[9] K. H. Chen and Y. S. Li, "A multi-radix FFT processor using pipeline in memory-based architecture (PIMA) FOR DVB-T/H systems," in *Proc. Int. Conf. Mixed Design Integr. Circuits Syst.*, Jun. 2008, pp. 549–553.

[10] S.-Y. Lee, C.-C. Chen, C.-C. Lee, and C.-J. Cheng, "A low-power VLSI architecture for a shared-memory FFT processor with a mixed-radix algorithm and a simple memory control scheme," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2006, pp. 157–160.
[11] P.-Y. Tsai, T.-H. Lee, and T.-D. Chiueh, "Power-efficient continuous-flow memory-based FFT processor for WiMax OFDM mode," in *Proc. Int. Symp. Intell. Signal Process. Commun.*, Dec. 2006, pp. 622–625.
[12] 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.
[13] 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.
[14] 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.
[15] C. Burrus, "Index mappings for multidimensional formulation of the DFT and convolution," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 25, no. 3, pp. 239–242, Jun. 1977.
[16] D. Kolba and T. Parks, "A prime factor FFT algorithm using high-speed convolution," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 25, no. 4, pp. 281–294, Apr. 1977.
[17] C. Temperton, "A generalized prime factor FFT algorithm for any $N = 2^p 3^q 5^r$," *SIAM J. Sci. Statist. Comput.*, vol. 13, no. 3, pp. 676–686, Mar. 1992.
[18] 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, May 2013.
[19] X. Xiao, E. Oruklu, and J. Saniie, "An efficient FFT engine with reduced addressing logic," *IEEE Trans. Circuits Syst. II*, vol. 55, no. 11, pp. 1149-–1153, Nov. 2008.
[20] 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.
[21] 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.
[22] M. Garrido, M. Á. Sánchez, M. L. López-Vallejo, and J. Grajal, "A 4096-point radix-4 memory-based FFT using DSP slices," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 1, pp. 375–379, Jan. 2017. [Online]. Available: http://ieeexplore.ieee.org
[23] Altera. (2013). *DFT/IDFT Reference Design*. [Online]. Available: http://www.altera.com
[24] Xilinx. (2014). *LogiCORE IP Discrete Fourier Transform V4.0*. [Online]. Available: http://www.xilinx.com

Kai-Feng Xia received the B.S. degree from Wuhan University, Wuhan, China, in 2010. He is currently pursuing the Ph.D. degree in electronic engineering with the Institute of Microelectronics of Chinese Academy of Sciences, Beijing, China.

His current research interests include VLSI architectures for communication and signal processing systems.

Bin Wu received the B.S. degree from Xi Hua University, Chengdu, China, in 1999, the M.S. degree from Chongqing University, Chongqing, China, in 2002, and the Ph.D. degree from the Institute of Microelectronics of Chinese Academy of Sciences, Beijing, China, in 2011.

He is currently a Professor with the Institute of Microelectronics of Chinese Academy of Sciences. His current research interests include high-performance/lower power VLSI designs for broadband wire-line and wireless communication systems.

Tao Xiong received the B.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2013. He is currently pursuing the M.S. degree with the Institute of Microelectronics of Chinese Academy of Sciences, Beijing, China.

His current research interests include low-power VLSI designs.

Tian-Chun Ye received the B.S. degree from Fudan University, Shanghai, China, in 1986.

He has been a Professor with the Institute of Microelectronics of Chinese Academy of Sciences (IMECAS), Beijing, China, since 1997. He is currently the Director of IMECAS and the China Internet of Things Development Center, Wuxi, China. His current research interests include semiconductor device and IC fabrication processes and equipment technologies.