A Mixed-Decimation MDF Architecture for Radix-2^k Parallel FFT

Jian Wang, Chunlin Xiong, Kangli Zhang, and Jibo Wei, *Member, IEEE*

***Abstract*— This paper presents a mixed-decimation multipath delay feedback (M^2 DF) approach for the radix-2^k fast Fourier transform. We employ the principle of folding transformation to derive the proposed architecture, which activates the idle period of arithmetic modules in multipath delay feedback (MDF) architectures by integrating the decimation-in-time operations into the decimation-in-frequency-operated computing units. Furthermore, we compare the proposed design with other efficient schemes, namely, the MDF and the multipath delay commutator (MDC) scheme theoretically and experimentally. Relying on the obtained expressions and statistics, it can be concluded that the M^2DF design serves as an efficient alternative to the MDF scheme, since it achieves improved efficiency in the utilization of arithmetic resources without deteriorating the superiorities of feedback structures. In addition, the recommended design performs better in memory requirement and computing delay compared with the MDC approach.**

***Index Terms*— Decimation-in-frequency (DIF), decimation-in-time (DIT), fast Fourier transform (FFT), multipath delay feedback (MDF), pipelined-parallel architecture.**

I. Introduction

BEING an efficient algorithm for discrete Fourier transform (DFT) computation, fast Fourier transform (FFT) has seen broader usage in the field of digital signal processing. It also plays an increasing important role in modern digital communications, since orthogonal frequency division multiplexing technique has been adopted in leading communication standards, including wireless LAN and long-term evolution. Therefore, an efficient implementation of FFT has attracted much attention and hardware designers have put forward various schemes to achieve reasonable tradeoffs between area and performance. By shortening the critical path in the signal diagram, pipelined architectures have an inherent advantage over other efficient hardware structures in providing high throughputs. To cope with the gradual increase of real-time business, plenty of published works have focused on designing and optimizing the pipelined structures [1]–[21].

In the serial-input-serial-output (SISO) scenario, single-path delay commutator (SDC) [1] structure is one of the most classical approaches to perform the pipelined FFT computation. To reduce the memory banks in SDC pipelines, single-path delay feedback (SDF) architecture is proposed in [2], which is characterized by the feedback connections in the circuits. These hardware schemes can be combined with the radix-2 [2], [3], radix-4 [4], and especially radix-2^k algorithm [5]–[8] to execute the DFT operation. Compared with the radix-4 approach, the radix-2^k pipeline is equipped with simpler butterfly units while making a better utilization of complex multipliers than the typical radix-2 scheme. Thus, radix-2^k algorithm acts as an effective alternative to the conventional computation methods from the perspective of hardware design.

Admittedly, the extension of communication service has stimulated a dramatic rise of throughput requirements. This advance in real-time business has additionally rendered the hardware schemes designed for SISO applications obsolete to a certain extent. On this occasion, multipath delay commutator (MDC) [9]–[13] and multipath delay feedback (MDF) [14]–[21], which serve as the upgrade of SDC and SDF, respectively, are proposed to calculate the FFT when several samples of the same sequence are received in parallel. In general, the MDF structure is composed of multiple interconnected SDF paths, and each path is responsible for managing one of the parallel input streams. This design contributes to the inheritance of utilizing registers efficiently at the expense of squandering the arithmetic components, particularly the butterfly units. By contrast, the MDC approach paves the way for boosting the hardware efficiency of arithmetic units (AUs), while additional memories have to be consumed for either reordering the samples or folding the streams, which additionally leads to an increase of computing delay.

Whether feed forward structures represented by the MDC approach or feedback structures, such as the SDF and the MDF design, they afford feasible solutions to strike a balance between the consumption of hardware resources and the reachable performance. As we move forward the discussion, the hardware resources can be further divided into two categories: 1) arithmetic resources associated with logical or arithmetic operations and 2) memory resources referred to as memory units, which are responsible for caching samples. Due to the outstanding performance in the efficient use of memory resources, the MDF scheme has been a tremendous success in a variety of applications [17]–[21]. Underneath the triumph, the underutilization of arithmetic resources is still a stubborn problem for feedback design and has not been resolved satisfactorily. The objective of this paper is to

Manuscript received June 22, 2013; revised March 19, 2014, September 6, 2014, November 22, 2014; accepted January 27, 2015. Date of publication February 26, 2015; date of current version December 24, 2015. This work was supported by the National Natural Science Foundation of China under Grant 61101096 and Grant 61101098.

The authors are with the School of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, China (e-mail: wangjian710108@126.com; xchlzju@nudt.edu.cn; zkl8855@163.com; wjbhw@nudt.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.2015.2402207

make a breakthrough in this respect while maintaining the advantages of feedback structures. To this end, the theory of folding transformation described in [22] and first adopted to derive the pipelined FFT architectures in [11] is employed to derive the proposed scheme, namely, the mixed-decimation MDF (M^{2}DF) architecture. The kernel of M^{2}DF design lies in scheduling the decimation-in-time (DIT) operations onto the decimation-in-frequency (DIF)-operated basic blocks. By this means, we mobilize the idle period of arithmetic modules in MDF architectures and thereby gain a considerable reduction of arithmetic resources to make up for the deficiency of the feedback FFT modules.

The rest of this paper is organized as follows. In Section II, we introduce the top-level architecture of the pipelined-parallel FFT processor based on the matrix factorization of radix-2^k algorithm. Section III expands on the M^{2}DF architectures using folding transformation. Theoretical and experimental comparisons of the M^{2}DF scheme and existing approaches are related in Section IV. Finally, the conclusion is drawn in Section V.

II. Top-Level Design of Pipelined Parallel DFT Processor

The matrix factorization of DFT unfolds the relations between computing algorithms and hardware structures. In terms of the extensively used radix-2^k algorithm, Cortes *et al.* [23] provide a comprehensive research on the matricial notation and specific VLSI implementation in SISO applications. In this section, we introduce the top-level design of parallel radix-2^k FFT processor by drawing on the research findings of [23]. Consider an N-point DFT computation, the input stream $\mathbf{x} = [x_0, x_1, \ldots, x_{N-1}]^T$ and output stream $\mathbf{y} = [y_0, y_1, \ldots, y_{N-1}]^T$ can be correlated through

$$\mathbf{y} = \mathbf{T}_N \cdot \mathbf{x} \tag{1}$$

where $\mathbf{T}_N$ is the N-point DFT transform matrix with elements

$$(\mathbf{T}_N)_{m,n} = e^{\frac{-j2\pi mn}{N}} \tag{2}$$

$m, n \in \{0, 1, \ldots, N-1\}$. Based on the principle of Cooley–Tukey algorithm, the relationship between $\mathbf{x}$ and $\mathbf{y}$ can be described alternatively in the form [23]

$$\mathbf{y} = \mathbf{P}_N^{(P)} \cdot [\mathbf{I}_S \otimes \mathbf{T}_P] \cdot \mathbf{D}_N^{(S)} \cdot [\mathbf{T}_S \otimes \mathbf{I}_P] \cdot \mathbf{x} \tag{3}$$

where $S = N/P$ and both of the parameters are assumed to be a power of two in the following discussion. The Kronecker product, denoted by $\otimes$, is an operation on two matrices of an arbitrary size resulting in a block matrix. Given an $m \times n$ matrix $\mathbf{A}$ and a $p \times q$ matrix $\mathbf{B}$, $\mathbf{A} \otimes \mathbf{B}$ returns an $mp \times nq$ block matrix

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{0,0}\mathbf{B} & \cdots & a_{0,n-1}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m-1,0}\mathbf{B} & \cdots & a_{m-1,n-1}\mathbf{B} \end{bmatrix}.$$

In addition, $\mathbf{I}_r$ in (3) is the identity matrix of size r; $\mathbf{D}_N^{(S)}$ is a diagonal matrix given by [23]

$$\mathbf{D}_N^{(S)} = \text{quasidiag}\{[\mathbf{I}_P, \mathbf{d}_P, (\mathbf{d}_P)^2, \ldots, (\mathbf{d}_P)^{S-1}]\} \tag{4}$$

with $\mathbf{d}_P = \text{diag}\{[1, e^{-j2\pi/N}, \ldots, e^{-j(P-1)\cdot 2\pi/N}]\}$; $\mathbf{P}_N^{(r)}$ is referred to as the stride-by-r permutation matrix, whose effect on a vector $\mathbf{z} = [z_0, z_1, \ldots, z_{N-1}]^T$ can be expressed as

$$\mathbf{P}_N^{(r)}\mathbf{z} = \text{vec}_{N/r,r}\left\{[\text{unvec}_{r,N/r}(\mathbf{z})]^T\right\}. \tag{5}$$

The function $\text{unvec}_{r,N/r}(\mathbf{z})$ in (5) is used for converting the vectorial operand into an $r \times N/r$ matrix $\mathbf{Z}$ with entries $(\mathbf{Z})_{m,n} = z_{m+n\cdot r}$, $m \in \{0, 1, \ldots, r-1\}$, $n \in \{0, 1, \ldots, N/r-1\}$. Conversely, $\text{vec}_{N/r,r}(\cdot)$ executes the relevant inverse operations to regain a vector. Applying $\text{unvec}_{P,S}(\cdot)$ to (3), it yields

$$\mathbf{Y} = \mathbf{T}_P \cdot \mathbf{D} \odot \mathbf{X} \cdot \mathbf{T}_S \tag{6}$$

with

$$\mathbf{X} = \text{unvec}_{P,S}(\mathbf{x}) = [\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{P-1}]^T = \begin{bmatrix} x_0 & x_P & \cdots & x_{(S-1)P} \\ x_1 & x_{P+1} & \cdots & x_{(S-1)P+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{P-1} & x_{2P-1} & \cdots & x_{SP-1} \end{bmatrix} \tag{7a}$$

$$\mathbf{Y} = \text{unvec}_{P,S}\left[(\mathbf{P}_N^{(P)})^{-1}\mathbf{y}\right] = [\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{P-1}]^T = \begin{bmatrix} y_0 & y_1 & \cdots & y_{S-1} \\ y_S & y_{S+1} & \cdots & y_{2S-1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{(P-1)S} & y_{(P-1)S+1} & \cdots & y_{PS-1} \end{bmatrix} \tag{7b}$$

and $(\mathbf{D})_{m,n} = e^{-j2\pi mn/N}$, $m \in \{0, 1, \ldots, P-1\}$, $n \in \{0, 1, \ldots, S-1\}$. Moreover, the symbol $\odot$ in (6) stands for the Hadamard product. For two matrices $\mathbf{A}$, $\mathbf{B}$ of the same dimension, $\mathbf{A} \odot \mathbf{B}$ is a matrix of the identical dimension as the operands, with elements given by

$$(\mathbf{A} \odot \mathbf{B})_{u,v} = (\mathbf{A})_{u,v} \cdot (\mathbf{B})_{u,v}.$$

Actually, (6) offers a matricial illustration of the P-parallel FFT, where $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{P-1}$ and $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{P-1}$ serve as the P-branch input and output, respectively. For the sake of perusing a hardware-friendly scheme to implement (6), $\mathbf{T}_S$ requires to be represented using smaller radices. Assuming that $S = 2^{n_S} = 2^{kn_k+l}$, then $\mathbf{T}_S$ can be decomposed as follows from the derivations in [23]:

$$\mathbf{T}_S = \mathbf{Q}_S \cdot \mathbf{H}(l, kn_k) \prod_{m=0}^{n_k-1} \mathbf{H}(k, km) \tag{8}$$

where $\mathbf{H}(u, v)$ represents the operation adopting radix-2^u FFT algorithm. To make the expression more explicit, let

$$\mathbf{B}_F(v) = \mathbf{I}_{2^v} \otimes \left[\mathbf{P}_{S/2^v}^{(2)} \cdot (\mathbf{I}_{S/2^{v+1}} \otimes \mathbf{T}_2) \cdot \left(\mathbf{P}_{S/2^v}^{(2)}\right)^{-1}\right] \tag{9}$$

be the radix-2 butterfly matrix. When $\mathbf{B}_F(v)$ operates on a S-dimension vector $\mathbf{x}_F$, the $m \cdot S/2^v + n$th and the $(m + 1/2) \cdot S/2^v + n$th ($m = 0, \ldots, 2^v - 1$, $n = 0, \ldots, S/2^{v+1} - 1$) entry of $\mathbf{x}_F$ are combined to execute the radix-2 butterfly transform $\mathbf{T}_2$. Moreover, denote S-square diagonal

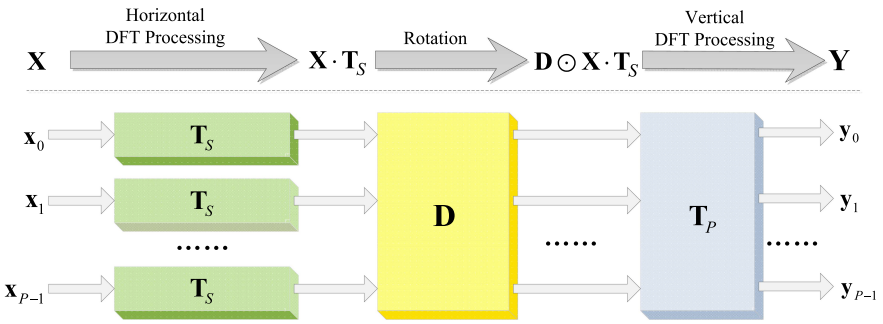


Fig. 1. Top-level design of proposed pipelined-parallel hardware structure.

matrix $\mathbf{M}(u, v)$ as coefficient-weighting matrix, which multiplies the results of butterfly operations with proper twiddle factors, then

$$\mathbf{H}(u,v) = \begin{cases} \prod_{m=v}^{v+u-1} \mathbf{M}(u,m)\mathbf{B}_F(m), & \text{if } u \neq 0 \\ \mathbf{I}_S, & \text{if } u = 0. \end{cases} \tag{10}$$

As can be observed in (10), the radix-2^u computing unit $\mathbf{H}(u, v)$ is further resolved into u butterfly matrices alternating with the same amount of coefficient-weighting matrices. When dealing with a serial input data stream, as pointed out in [23], the SDF elementary structure is a feasible choice to implement $\mathbf{B}_F(v)$ and $\mathbf{M}(u, v)$. Therefore, $\mathbf{H}(l, kn_k)\prod_{m=0}^{n_k-1}\mathbf{H}(k, km)$ in (8) can be realized using an SDF pipeline to compute S-point DFT serially. For the remaining term $\mathbf{Q}_S$ in (8), it corresponds to the bit-reversed ordering

$$\mathbf{Q}_S = \prod_{m=0}^{n_S-1}\left[\mathbf{I}_{2^m} \otimes \mathbf{P}_{S/2^m}^{(S/2^{m+1})}\right]. \tag{11}$$

It can be inferred from (11) that streams ordering is irrelevant to k and l. This character equips the bit-reversed reordering component with great generality among radix-2^k pipelines.

The derivations in (6) and (8) allow us to map the DFT transform to a pipelined-parallel hardware structure, where the top-level design is illustrated in Fig. 1. As shown, the parallel processing includes three phases.

1) *Phase 1* (*Horizontal DFT Processing*): During this phase the operator $\mathbf{T}_S$ acts on P-parallel data streams $\mathbf{x}_0, \ldots, \mathbf{x}_{P-1}$ independently to obtain $\mathbf{X} \cdot \mathbf{T}_S$. According to (8), the R2^{k}SDF pipelined architecture in series with a bit-reversed reordering unit is a feasible way to implement $\mathbf{T}_S$ efficiently.
2) *Phase 2* (*Rotation*): The uth ($u = 0, \ldots, S-1$) output of Phase 1 in the vth ($v = 0, \ldots, P-1$) stream is multiplied by the corresponding twiddle factor $e^{-j2\pi uv/N}$ to calculate $\mathbf{D} \odot \mathbf{X} \cdot \mathbf{T}_S$.
3) *Phase 3* (*Vertical DFT Processing*): P samples within each clock cycle are processed simultaneously through a parallel-input-parallel-output implementation of $\mathbf{T}_P$. Parallel output streams $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{P-1}$ are generated after the execution.

In addition to the pipelined implementation of $\mathbf{T}_S$ in Phase 1, the three phases interpreted above also construct a pipelined framework. Therefore, this proposed architecture will give the system access to a high throughput.

III. Design of Mixed-Decimation MDF Architecture

In this section, we first apply the theory of folding transformation to derive the folding matrices associated with DIF- and DIT-operated SDF processor. With the assist of this theoretical basis, the operations in SDF pipeline are rescheduled to reverse the underutilization of arithmetic modules, which is accomplished by integrating DIT operations into DIF-operated basic blocks. Finally, this optimization strategy is used to implement the top-level design in Section II, generating the proposed M^{2}DF structure with improved hardware efficiency.

A. Analyzing the SDF Scheme Using Folding Transformation

The folding transformation provides a systematic technique for designing control circuits for hardware where several algorithm operations are time multiplexed on a single computing device [22]. Consider an eight-point radix-2^k DIF computation, the algorithm can be presented using a data flow graph shown in Fig. 2(a), where the nodes represent computations and the directed edges represent data paths. As shown, the flow graph consists of three stages and four operations are executed within each stage. When x_i, $i = 0, \ldots, 7$ arrives in serial, multiple operations in each stage can be time multiplexed to a single computing unit without any collisions, and the control circuits are determined systematically by folding transformation. In this way, the FFT flow graph in Fig. 2(a) can be transformed into a pipelined form shown in Fig. 2(b), where operations $A_0, \ldots, A_3$, $B_0, \ldots, B_3$, and $C_0, \ldots, C_3$ are performed in the computing unit U_A, U_B, and U_C, respectively.

The multiple operations included in a computing module are arranged by folding sets. A folding set is an ordered set of operations executed by the same computing unit. Apart from the operations associated with the nodes in the data flow graph, generally, there are also null operations in the folding set. The number of operations folded into a folding set is called the folding factor (denoted by R in subsequent discussion). Correspondingly, the computing unit works in cycles with a period taking up R time partitions, and the operation in the rth ($r = 0, \ldots, R-1$) position is executed

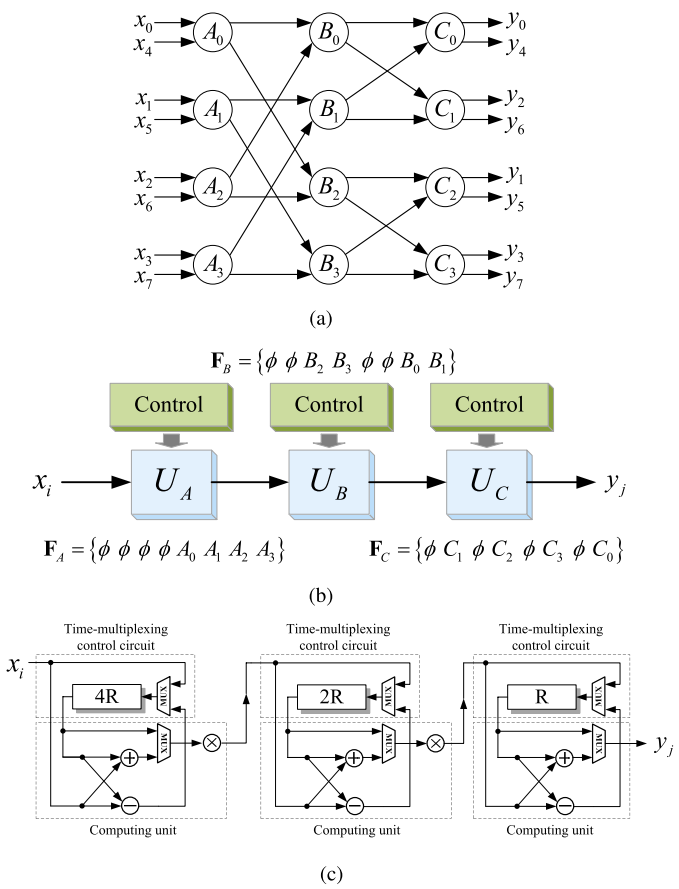


Fig. 2. (a) Data flow graph of an eight-point DIF DFT using a radix-2^k algorithm. (b) Data flow graph can be converted into a pipelined version through folding transformation. (c) Optimized hardware scheme to implement the pipelined flow graph. Moreover, the shift register unit marked nR indicates n positions are available in the device to cache the complex inputs.

during the rth time partition. For example, consider a folding set $\mathbf{F} = \{\phi\ A_0\ \phi\ A_1\ \phi\ A_2\ \phi\ A_3\}$ with $R = 8$, the computing unit operates A_0, A_1, A_2, and A_3 in the first, third, fifth, and seventh time partitions, respectively, while keeping idle in remaining intervals.

When folding transformation is utilized to derive the circuit from a data flow graph, the accurate determination of folding sets serves as the cornerstone. In particular, the folding factor depends on the periodicity of operations executed by a computing unit, while the arrangement of elements within a folding set is linked closely to specific hardware structures. In terms of an N-point SDF processor using a radix-2^k DIF FFT algorithm, it constitutes of $\log_2 N$ computing modules. The lth ($l = 0, 1, \ldots, \log_2 N - 1$) module alternates idle condition and operating status with an interval, including $N/2^{l+1}$ time instances. Taking this priori knowledge into account, the folding sets corresponding to U_A, U_B, and U_C in Fig. 2(b) can be written preliminarily as

$$\check{\mathbf{F}}_A = \{\phi\ \ \phi\ \ \phi\ \ \phi\ \ A_0\ \ A_1\ \ A_2\ \ A_3\} \quad (12a)$$
$$\check{\mathbf{F}}_B = \{\phi\ \ \phi\ \ B_0\ \ B_1\ \ \phi\ \ \phi\ \ B_2\ \ B_3\} \quad (12b)$$
$$\check{\mathbf{F}}_C = \{\phi\ \ C_0\ \ \phi\ \ C_1\ \ \phi\ \ C_2\ \ \phi\ \ C_3\} \quad (12c)$$

with the folding factor $R = 8$.

Equations (12a)–(12c) incorporate the features of SDF computing units into the derivation of folding sets. Nevertheless, they fail to consider the precedence of operations associated with different modules. As can be found in Fig. 2(b), when the data stream flows serially to U_A, the newly generated intermediate results would arrive at U_B and U_C with a delay of four and six time partitions, respectively, after which the modules are able to perform the operations formulated by (12b) and (12c). In view of this causality constraint, the mth ($m = 0, \ldots, R-1$) element in $\check{\mathbf{F}}_B$ ought to be aligned with the $[m+4]_R$th entry of $\check{\mathbf{F}}_A$ ($[\cdot]_x$ returns the modulus of x). Likewise, the nth ($n = 0, \ldots, R-1$) element in $\check{\mathbf{F}}_C$ requires to match the $[n+6]_R$th position of $\check{\mathbf{F}}_A$. This yields modified folding sets

$$\mathbf{F}_A = \{\phi\ \ \phi\ \ \phi\ \ \phi\ \ A_0\ \ A_1\ \ A_2\ \ A_3\} \quad (13a)$$
$$\mathbf{F}_B = \{\phi\ \ \phi\ \ B_2\ \ B_3\ \ \phi\ \ \phi\ \ B_0\ \ B_1\} \quad (13b)$$
$$\mathbf{F}_C = \{\phi\ \ C_1\ \ \phi\ \ C_2\ \ \phi\ \ C_3\ \ \phi\ \ C_0\} \quad (13c)$$

where $\mathbf{F}_A$ inherits $\check{\mathbf{F}}_A$ directly. By contrast, $\mathbf{F}_B$, $\mathbf{F}_C$ can be obtained by cyclic right shifting the elements in $\check{\mathbf{F}}_B$, $\check{\mathbf{F}}_C$ for four and six positions, respectively. As folding sets reveal limitations when analyzing the entire hardware structure, we recommend an alternative analytical tool named folding matrix to cover this shortage. Denote an $n_f \times R$ matrix

$$\mathbf{F}_{\mathrm{DIF}} = \begin{bmatrix} \phi & \phi & \phi & \phi & A_0 & A_1 & A_2 & A_3 \\ \phi & \phi & B_2 & B_3 & \phi & \phi & B_0 & B_1 \\ \phi & C_1 & \phi & C_2 & \phi & C_3 & \phi & C_0 \end{bmatrix} \quad (14)$$

as the folding matrix corresponding to (12a)–(12c), where n_f equals the amount of computing stages in the folded flow graph, while the rows of $\mathbf{F}_{\mathrm{DIF}}$ corresponds to different folding sets. Therefore, the entry $(\mathbf{F}_{\mathrm{DIF}})_{m,n}$ ($m = 0, \ldots, n_f - 1$, $n = 0, \ldots, R-1$) represents the operation executed by the mth computing unit at the $lR + n$th ($l = 1, 2, \ldots$) time instance.

When obtaining the folding sets associated with U_A, U_B, and U_C, the supporting circuits used to fulfill the time multiplexing of computing units can be obtained through folding transformation. To boost the hardware efficiency, the implementation scheme can be further optimized by introducing the register minimization technique, which eventually generates the SDF hardware structure presented in Fig. 2(c). The shift register unit marked nR in the graph indicates that n positions are available in the component to cache the complex inputs.

Similar processes can be applied to analyze the DIT data flow graph shown in Fig. 3(a), which is essentially a transposition of its DIF counterpart. Assuming that the input samples arrive in bit-reversed order, the folding sets considering the property of SDF computing are

$$\check{\mathbf{G}}_{\tilde{A}} = \{\phi\ \ \tilde{A}_0\ \ \phi\ \ \tilde{A}_1\ \ \phi\ \ \tilde{A}_2\ \ \phi\ \ \tilde{A}_3\} \quad (15a)$$
$$\check{\mathbf{G}}_{\tilde{B}} = \{\phi\ \ \phi\ \ \tilde{B}_0\ \ \tilde{B}_1\ \ \phi\ \ \phi\ \ \tilde{B}_2\ \ \tilde{B}_3\} \quad (15b)$$
$$\check{\mathbf{G}}_{\tilde{C}} = \{\phi\ \ \phi\ \ \phi\ \ \phi\ \ \tilde{C}_0\ \ \tilde{C}_1\ \ \tilde{C}_2\ \ \tilde{C}_3\} \quad (15c)$$

with the folding factor $R = 8$. Note that the arrival of freshly generated results at $U_{\tilde{B}}$ and $U_{\tilde{C}}$ involve additional delay of one and three time partitions, thus by cyclic right shifting the

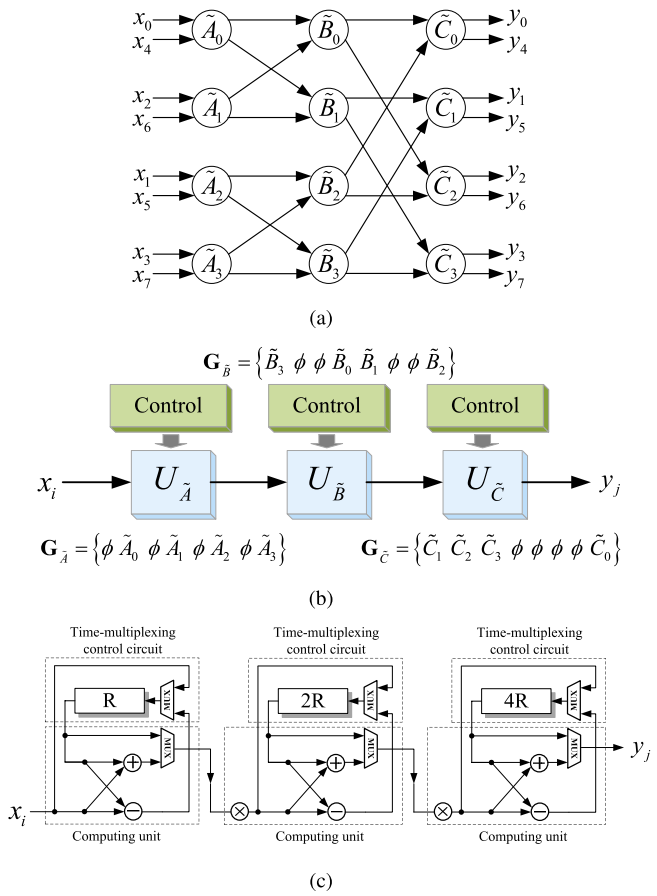


Fig. 3. (a) Data flow graph of an eight-point DIT DFT using a radix-2^k algorithm. (b) Data flow graph can be converted into a pipelined version through folding transformation. (c) Optimized hardware scheme to implement the pipelined flow graph.

elements in $\check{\mathbf{G}}_{\tilde{B}}$ for one space and $\check{\mathbf{G}}_{\tilde{C}}$ for three spaces, we obtain

$$\mathbf{G}_{\tilde{A}} = \{\phi \quad \tilde{A}_0 \quad \phi \quad \tilde{A}_1 \quad \phi \quad \tilde{A}_2 \quad \phi \quad \tilde{A}_3\} \tag{16a}$$

$$\mathbf{G}_{\tilde{B}} = \{\tilde{B}_3 \quad \phi \quad \phi \quad \tilde{B}_0 \quad \tilde{B}_1 \quad \phi \quad \phi \quad \tilde{B}_2\} \tag{16b}$$

$$\mathbf{G}_{\tilde{C}} = \{\tilde{C}_1 \quad \tilde{C}_2 \quad \tilde{C}_3 \quad \phi \quad \phi \quad \phi \quad \phi \quad \tilde{C}_0\}. \tag{16c}$$

Accordingly, the $n_f \times R$ folding matrix corresponding to the DIT data flow graph is expressed as

$$\mathbf{G}_{\mathrm{DIT}} = \begin{bmatrix} \phi & \tilde{A}_0 & \phi & \tilde{A}_1 & \phi & \tilde{A}_2 & \phi & \tilde{A}_3 \\ \tilde{B}_3 & \phi & \phi & \tilde{B}_0 & \tilde{B}_1 & \phi & \phi & \tilde{B}_2 \\ \tilde{C}_1 & \tilde{C}_2 & \tilde{C}_3 & \phi & \phi & \phi & \phi & \tilde{C}_0 \end{bmatrix}. \tag{17}$$

The entry $(\mathbf{G}_{\mathrm{DIT}})_{m,n}$ $(m = 0, \ldots, n_f - 1,\ n = 0, \ldots, R-1)$ also represents the operation executed by the mth computing module at the $lR + n$th $(l = 1, 2, \ldots)$ time instance. With the assist of folding transformation and register minimization techniques, the hardware scheme can be derived, which is shown in Fig. 3(c).

B. Rescheduling the Operations in SDF Pipelines

The folding matrix given in (14) and (17) conveys the fact that whether designers adopt the DIF approach or DIT scheme to construct the SDF circuit, the existence of null operations in folding sets will always degrade the efficiency of arithmetic components, leading to an approximate 50% utilization of complex adders merely. To address this issue, we reschedule the operations in SDF pipelines to activate the idle intervals of computing units. As the folding matrix acts as the simplified representation of circuit status, the rearrangement of operations can be achieved equivalently by modifying the folding matrices. To be specific, the modification is accomplished through the following three steps.

1) Cyclic right shift the columns of $\mathbf{G}_{\mathrm{DIT}}$ for $S_R = 1$ space, it generates

$$(\mathbf{G}_{\mathrm{DIT}})_1 = \begin{bmatrix} \tilde{A}_3 & \phi & \tilde{A}_0 & \phi & \tilde{A}_1 & \phi & \tilde{A}_2 & \phi \\ \tilde{B}_2 & \tilde{B}_3 & \phi & \phi & \tilde{B}_0 & \tilde{B}_1 & \phi & \phi \\ \tilde{C}_0 & \tilde{C}_1 & \tilde{C}_2 & \tilde{C}_3 & \phi & \phi & \phi & \phi \end{bmatrix} \tag{18}$$

where the subscript outside the brackets denotes the cyclic right shift operation. Note the transformation introduces an additional S_R-unit delay to the folding sets $\mathbf{G}_{\tilde{A}}$, $\mathbf{G}_{\tilde{B}}$, and $\mathbf{G}_{\tilde{C}}$ simultaneously, thus the entry $[(\mathbf{G}_{\mathrm{DIT}})_1]_{m,n}$ $(m = 0, \ldots, n_f - 1,\ n = 0, \ldots, R-1)$ represents the operation executed by the mth computing module at the $lR + n - 1$th $(l = 1, 2, \ldots)$ time instance, differing from the definitions of $\mathbf{G}_{\mathrm{DIT}}$. To fulfill the operations formulated by (18), a latency unit providing a S_R-clock-cycle delay for input samples should be integrated into the circuit associated with $\mathbf{G}_{\mathrm{DIT}}$.

2) Flip $(\mathbf{G}_{\mathrm{DIT}})_1$ vertically to obtain

$$(\mathbf{G}_{\mathrm{DIT}})_1^{\mathrm{F}} = \begin{bmatrix} \tilde{C}_0 & \tilde{C}_1 & \tilde{C}_2 & \tilde{C}_3 & \phi & \phi & \phi & \phi \\ \tilde{B}_2 & \tilde{B}_3 & \phi & \phi & \tilde{B}_0 & \tilde{B}_1 & \phi & \phi \\ \tilde{A}_3 & \phi & \tilde{A}_0 & \phi & \tilde{A}_1 & \phi & \tilde{A}_2 & \phi \end{bmatrix} \tag{19}$$

where the added superscript stands for the vertical flip. As the rows of folding matrix are connected with specific computing units, the modification of current step would redefine this mapping relation, i.e., the mth $(m = 0, \ldots, n_f - 1)$ row of $(\mathbf{G}_{\mathrm{DIT}})_1^F$ describes the time-multiplexing scheme of the $n_f - m - 1$th computing unit, rather than the mth module as the original $\mathbf{G}_{\mathrm{DIT}}$.

3) Overlay $\mathbf{F}_{\mathrm{DIF}}$ with $(\mathbf{G}_{\mathrm{DIT}})_1^F$. Thus, we can obtain

$$\begin{aligned} \mathbf{F} &= \mathbf{F}_{\mathrm{DIF}} + (\mathbf{G}_{\mathrm{DIT}})_1^{\mathrm{F}} \\ &= \begin{bmatrix} \tilde{C}_0 & \tilde{C}_1 & \tilde{C}_2 & \tilde{C}_3 & A_0 & A_1 & A_2 & A_3 \\ \tilde{B}_2 & \tilde{B}_3 & B_2 & B_3 & \tilde{B}_0 & \tilde{B}_1 & B_0 & B_1 \\ \tilde{A}_3 & C_1 & \tilde{A}_0 & C_2 & \tilde{A}_1 & C_3 & \tilde{A}_2 & C_0 \end{bmatrix}. \end{aligned} \tag{20}$$

Clearly, the superposition eliminates the preexisting null operations, which suggests that the synthesis of DIT SDF pipeline and DIF SDF pipeline is able to fully utilize the arithmetic modules. With respect to the hardware implementation, as shown in Fig. 4, the mth $(m = 0, \ldots, n_f - 1)$ computing unit in the DIF SDF pipeline should be integrated with the $n_f - m - 1$th one in the DIT counterpart due to the vertical flip operation in the second step. In addition, the DIT-processing stream ought to experience a S_R-clock-cycle delay up front

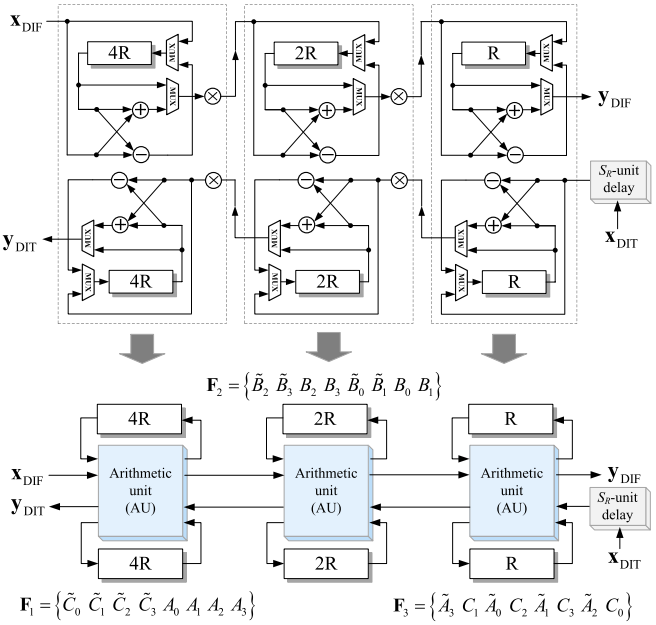


Fig. 4. Integration of the DIF and DIT pipelined processor to fully utilize the computing units.

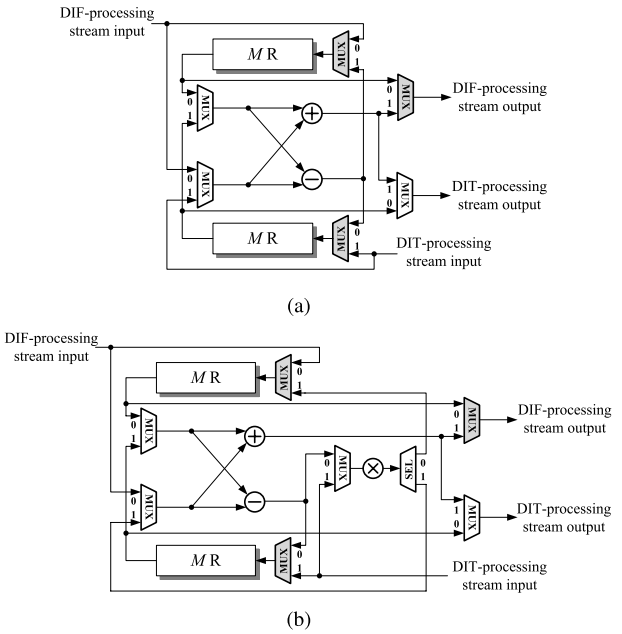


Fig. 5. Underlying structures of AUs. (a) Type I structure, complex adders are shared only. (b) Type II structure, both complex adders and complex multipliers are shared.

when considering the cyclic shift operation in the first step. It is also worth noting that S_R takes nonunique value because of the cyclic feature of folding sets. Apart from $S_R = 1$ in the previous discussion, there are a set of candidates

$$S_R = nR + 1, \quad n \in \mathbb{N} \tag{21}$$

with R represents the folding factor, which is equal to the FFT size in serial computing.

In terms of the underlying design of arithmetic modules, the conventional SDF elementary structure should be upgraded to undertake both DIF processing and DIT processing tasks. According to the utilization efficiency of complex multipliers, two kinds of hardware structures referred to as Type I and Type II are proposed in Fig. 5. For Type I architecture in Fig. 5(a), adders are shared by two streams, while multipliers (not exhibited) are provided individually. Type II architecture is shown in Fig. 5(b), where both adders and multipliers are reused to further improve the hardware efficiency. Thus, Type II is suitable for the situation when data streams occupy the complex multipliers with a 50% utilization ratio or less. Moreover, the multiplexors and selectors in the circuit can be divided into two categories (white for Category I and gray for Category II in Fig. 5). When arithmetic modules are in service, the components within each category share the identical logic signal. The control scheme, which is synchronized with the DIF-computing stream, is summarized as follows.

1) For the first M samples of the stream, the components belonging to Category I are controlled by logic 1, while others in Category II are controlled by logic 0.
2) During the next M samples, the control signals should be inverted.

The scheme above is suitable for both Type I and Type II design and is $2M$-clock-cycle periodic, where M is the length of shift register sets in the arithmetic module.

C. Design of the M^{2}DF Hardware Structure

For the top-level design shown in Fig. 1, the rescheduling strategy proposed in Section III-B is beneficial to optimize the underlying scheme, which results in the proposed M^{2}DF structure. We begin with two-parallel architecture to unfold the kernel. Taking a 32-point DFT computation, for example, the corresponding M^{2}DF design is shown in Fig. 6. During the horizontal DFT processing, as shown, $\mathbf{x}_0$ and $\mathbf{x}_1$ execute DIF and DIT scheme, respectively, to calculate $\mathbf{x}_0\mathbf{T}_S$ and $\mathbf{x}_1\mathbf{T}_S$. By drawing on the discussion in Section III-B, the pipelined circuit constructed by the modified AUs in Fig. 5 is able to fulfill this task, which promotes the arithmetic resources to full utilization. In addition, as input samples require to be rearranged as bit reversal before participating in DIT computation, the latency unit in Fig. 4 is replaced with a reordering module, which permutes the samples with a delay of $S_R = S + 1$ clock cycles. After the rotation, the parallel streams are connected to a simple radix-2 butterfly unit to execute the vertical DFT processing.

As for the four-parallel and eight-parallel applications, Fig. 7 shows the hardware structures using a 64-point DFT. As shown, the circuit used to perform the horizontal DFT is essentially a parallel extension of the counterpart in the two-parallel structure. By contrast, the hardware scheme for vertical DFT depends closely on the parallelism. For four-parallel computing in Fig. 7(a), four radix-2 butterfly units along with requisite rotation devices operate coordinately to provide a throughput of four samples per clock cycle. In the eight-parallel scenario,

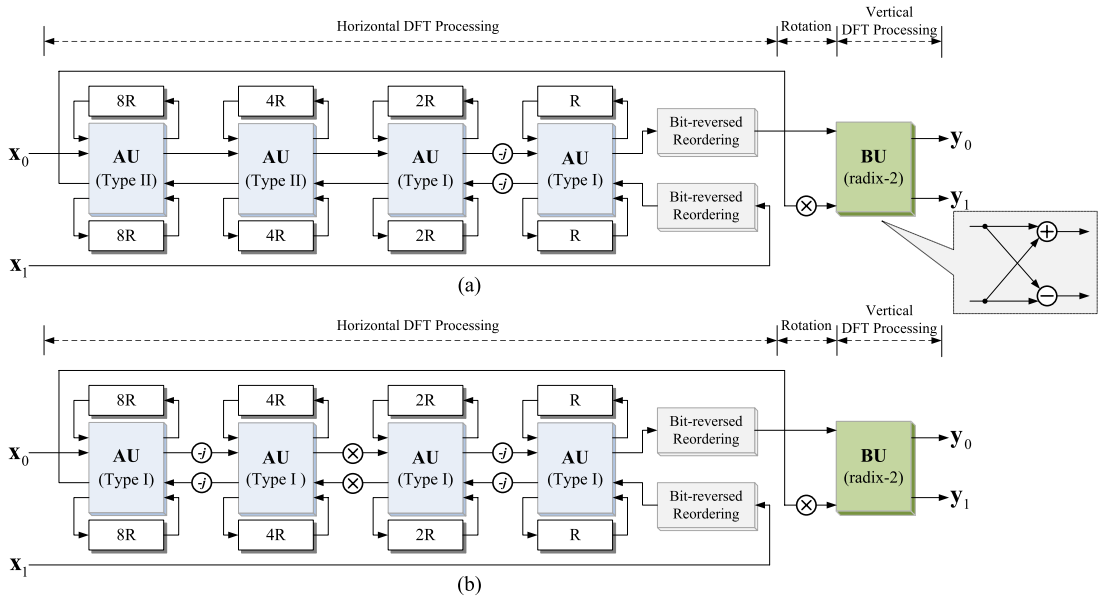


Fig. 6. Proposed two-parallel M^{2}DF architectures for the computation of 32-point DFT. (a) Use radix-2 algorithm in the horizontal DFT processing phase. (b) Use radix-2^2 algorithm in the horizontal DFT processing phase.

by means of rearranging the interconnections of the eight-point radix-2 data flow graph, the multiple stages included in the vertical DFT circuit can be implemented using a fixed structure, which contributes to reduce the hardware complexity. Furthermore, note each multiplier in the vertical DFT unit is related to a single known coefficient, they can be mapped to the low-complexity constant multipliers.

The M^{2}DF structure can be readily applied to the P-parallel radix-2^k computation, where even P is emphasized here. The exponent k will only affect the arrangement of complex multipliers in the horizontal DFT circuit. On the other hand, the parallelism P plays an important role in the realization of vertical DFT unit, as this module is responsible for a P-parallel P-point DFT. In terms of the rotation, $P-1$ complex multipliers are sufficient to complete the relevant task, which are irrelevant to the selection of radix-2^k algorithm.

IV. Comparison and Evaluation

In this section, we first evaluate the hardware expense of M^{2}DF structures. Afterward, we compare the M^{2}DF scheme with existing approaches theoretically and experimentally.

A. Evaluation of the Proposed M^{2}DF Structure

In general, an FFT processor can be decomposed into three parts: 1) the arithmetic component; 2) the control circuit; and 3) the reordering module. The arithmetic component is primarily composed of complex adders and multipliers to execute the butterfly operations and twiddle factor multiplications. Moreover, the multipliers can be further categorized into the general ones with varied twiddle factors and constant ones using fixed coefficients. This fine-grained partition takes account of the diverse hardware complexity between general complex multipliers and constant ones. In this way, the numbers of complex adders λ_a, general complex multipliers λ_c^g and constant complex multipliers account for the primary consumption of arithmetic resources in an FFT module.

Memory expenditure comes from the realization of control circuit and reordering module. The control circuit is responsible for the rearrangement of data streams to coordinate with the arithmetic computations. In pipelined structures, shift register sets are widely used to implement the control circuit. On the other hand, the reordering module permutes the samples to obtain the desired parallel output $\text{unvec}_{P,S}(\mathbf{y})$. To this end, $\mathbf{Y}$ in (7b) should be gained firstly, after which the permutation network corresponding to the matrix $\mathbf{P}_N^{(P)}$ is applied to bridge the gap between $\mathbf{Y}$ and $\text{unvec}_{P,S}(\mathbf{y})$. It is worth noting that the hardware design of $\mathbf{P}_N^{(P)}$ could be identical for the FFT processors with the same parallelism, while the circuit to obtain $\mathbf{Y}$ is linked closely to concrete FFT schemes. Thus, in the following analysis, the memory cost to complete the first-phase reordering is considered as the metric to evaluate the hardware efficiency. Furthermore, some FFT processors do not utilize $\text{unvec}_{P,S}(\mathbf{x})$ directly, e.g., Garrido *et al.* [12] convert $\text{unvec}_{P,S}(\mathbf{x})$ into $[\text{unvec}_{S,P}(\mathbf{x})]^T$ to launch the FFT. This additional memory requirement is also considered here.

For the M^{2}DF design, $P/2 \cdot \log_2 N$ radix-2 butterfly units, which consist of $\lambda_a = P \cdot \log_2 N$ complex adders are an integral part of the P-parallel, N-point DFT computation. The general complex multipliers are existed in the horizontal DFT and rotation units, where the selection of radix-2^k algorithm makes a difference to the overall consumption

$$\lambda_c^g = \begin{cases} \frac{P}{2} \cdot \log_2\left(\frac{N}{P}\right) - 1, & \text{for radix-2} \\ P \cdot \lceil \log_{2^k}\left(\frac{N}{P}\right) \rceil - 1, & \text{for radix-}2^k (k > 1). \end{cases} \tag{22}$$

It can be found from (22) that high radices are beneficial to reduce the general complex multipliers at the expense of increasing the number of constant complex multipliers. Apart from the $P\lfloor \log_{2^k}(N/P) \rfloor$ constant multipliers used to implement the horizontal DFT unit, additional ones will be

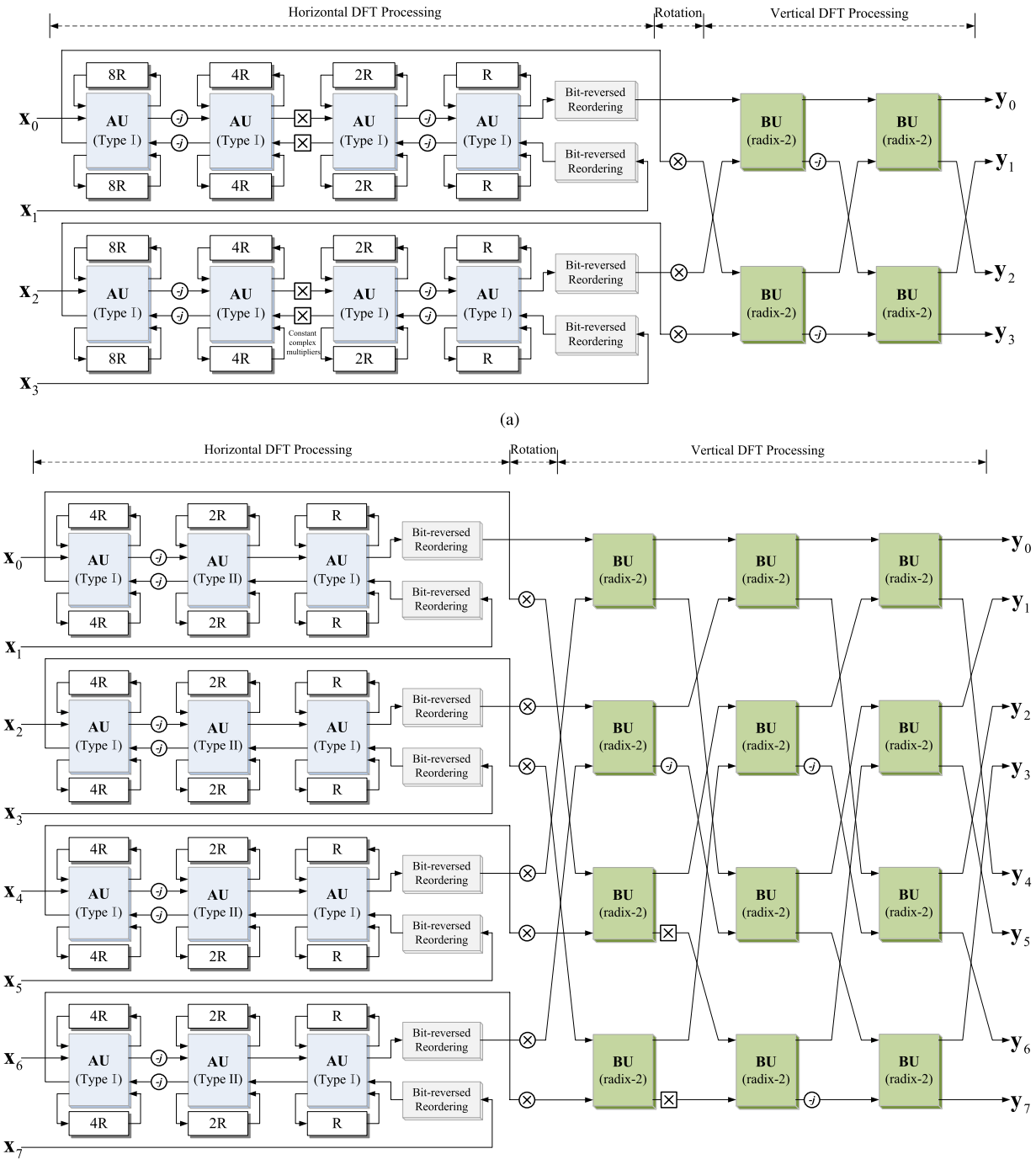


Fig. 7. Proposed M^{2}DF architectures for the computation of 64-point DFT. (a) Four-parallel computation, use radix-2^4 algorithm in the horizontal DFT processing phase. (b) Eight-parallel computation, use radix-2^3 algorithm in the horizontal DFT processing phase.

consumed by vertical DFT unit when $P > 8$. In terms of the memory consumption, $N - P$ registers applied to implement control circuit, together with the additional $N + P$ ones to perform the reordering operation are considered in the following comparison.

Table I compares the proposed structures with other efficient approaches in the two-parallel, four-parallel, and eight-parallel scenario, respectively. In spite of the differences among underlying design, we suppose the input and output parallel streams of different kinds of DFT processors follow the definitions in (7a) and (7b). As shown in the table, one of the remarkable features of the M^{2}DF structures is embodied in the effective use of complex adders. For conventional MDF schemes, the utilization ratio of adders is $\sim$50%, and the parallelization cannot improve this ratio. By contrast, the proposed design achieves a full utilization and reduces the number of adders by half compared with the MDF counterparts.

On the other hand, the figures in Table I suggest that the M^{2}DF structures are more memory efficient than the MDC counterpart. This is essentially a reflection of the

TABLE I
COMPARISON OF THE PROPOSED ARCHITECTURES TO OTHER APPROACHES FOR THE COMPUTATION OF AN N-POINT FFT

Hardware Scheme	Arithmetic components consumption ($n = \log_2 N$)				Memory consumption		Performance	
	Complex Multipliers			*Complex*	*Data*	*Streams*	*Latency*	*Throughput*
	General	*Constant*	*Overall* †	*Adders*	*Reordering*	*Folding*	*(Clk.)*	*(Samp. / Clk.)*
2-PARALLEL ARCHITECTURES								
R2MDC, [11]	$n-2$	0	$n-2$	$2n$	$5N/8-3$	$3N/2-2$	$11N/8-4$	2
R2^{2}MDC, [12]	$2\lceil n/2\rceil-2$	0	$n-2$	$2n$	$2N$	$N-2$	$3N/2-1$	2
R2^{2}MDF, [17]	$2\lceil (n-1)/2\rceil-1$	0	$n-2$	$4n$	N	$N-2$	$N-1$	2
R2^{4}MDF, [14]	$2\lceil n/4\rceil-2$	$2\lfloor n/4\rfloor$	$n-2$	$4n$	N	$N-2$	$N-1$	2
R2M^{2}DF, proposed	$n-2$	0	$n-2$	$2n$	$N+2$	$N-2$	N	2
R2^{2}M^{2}DF, proposed	$2\lceil (n-1)/2\rceil-1$	0	$n-2$	$2n$	$N+2$	$N-2$	N	2
R2^{3}M^{2}DF, proposed	$2\lceil (n-1)/3\rceil-1$	$\lfloor (n-1)/3\rfloor$	$n-2$	$2n$	$N+2$	$N-2$	N	2
R2^{4}M^{2}DF, proposed	$2\lceil (n-1)/4\rceil-1$	$2\lfloor (n-1)/4\rfloor$	$n-2$	$2n$	$N+2$	$N-2$	N	2
4-PARALLEL ARCHITECTURES								
R2^{2}MDC, [12]	$3\lceil n/2\rceil-3$	0	$3n/2-3$	$4n$	$2N$	$N-4$	$3N/4-1$	4
R4MDC, [13]	$3\lceil n/2\rceil-3$	0	$3n/2-3$	$4n$	N	$8N/3-4$	$7N/12-1$	4
R2^{4}MDF, [16]	$4\lceil n/4\rceil-4$	$4\lfloor n/4\rfloor$	$2n-4$	$8n$	N	$N-4$	$N/2-1$	4
R2^{4}MDF, [15]	$4\lceil n/4\rceil-4$	$4\lfloor n/4\rfloor$	$2n-4$	$8n$	N	$N-4$	$N/2-1$	4
R2M^{2}DF, proposed	$2n-5$	0	$2n-5$	$4n$	$N+4$	$N-4$	$N/2$	4
R2^{2}M^{2}DF, proposed	$4\lceil (n-2)/2\rceil-1$	0	$2n-5$	$4n$	$N+4$	$N-4$	$N/2$	4
R2^{3}M^{2}DF, proposed	$4\lceil (n-2)/3\rceil-1$	$2\lfloor (n-2)/3\rfloor$	$2n-5$	$4n$	$N+4$	$N-4$	$N/2$	4
R2^{4}M^{2}DF, proposed	$4\lceil (n-2)/4\rceil-1$	$4\lfloor (n-2)/4\rfloor$	$2n-5$	$4n$	$N+4$	$N-4$	$N/2$	4
8-PARALLEL ARCHITECTURES								
R2^{4}MDC, [12]	$8\lceil n/4\rceil-8$	$6\lfloor n/4\rfloor$	$7n/2-8$	$8n$	$2N$	$N-8$	$3N/8-1$	8
R2^{4}MDF, [18]	$8\lceil n/4\rceil-8$	$8\lfloor n/4\rfloor$	$4n-8$	$16n$	N	$N-8$	$N/4-1$	8
R2^{4}MDF, [19]	$8\lceil (n-3)/4\rceil-1$	$8\lfloor (n-3)/4\rfloor+2$	$4n-11$	$16n$	N	$N-8$	$N/4-1$	8
R2^{5}MDF, [21]	$8\lceil n/5\rceil-8$	$16\lfloor n/5\rfloor$	$24n/5-8$	$16n$	N	$N-8$	$N/4-1$	8
R2M^{2}DF, proposed	$4n-13$	2	$4n-11$	$8n$	$N+8$	$N-8$	$N/4$	8
R2^{2}M^{2}DF, proposed	$8\lceil (n-3)/2\rceil-1$	2	$4n-11$	$8n$	$N+8$	$N-8$	$N/4$	8
R2^{3}M^{2}DF, proposed	$8\lceil (n-3)/3\rceil-1$	$4\lfloor (n-3)/3\rfloor+2$	$4n-11$	$8n$	$N+8$	$N-8$	$N/4$	8
R2^{4}M^{2}DF, proposed	$8\lceil (n-3)/4\rceil-1$	$8\lfloor (n-3)/4\rfloor+2$	$4n-11$	$8n$	$N+8$	$N-8$	$N/4$	8

† To simplify the expressions, $\lfloor x\rfloor \approx x$, $\lceil x\rceil \approx x$ have been adopted in the determination of overall multipliers.

inherent advantage of feedback FFT architectures. The MDC structures, as revealed from the design in [11]–[13], will spend additional memories either on the reordering of samples or on the construction of control circuit to fold the parallel streams.

Furthermore, owing to the optimized usage of multipliers in the arithmetic components, the proposed M^{2}DF structures are ahead of the MDF design in terms of the efficient utilization of multipliers. Nevertheless, since the improvement is premised on the preservation of low memory requirement and short computing delay, the MDC approach will surpass the proposed method at this metric in certain applications. In this respect, the M^{2}DF design and the MDC scheme have their own merits.

B. Analysis of Experimental Results

The proposed M^{2}DF processors have been implemented in Xilinx Virtex-6 field-programmable gate array (FPGA), XC6VLX240T-3FF784 using the programming software ISE 12.4. Furthermore, the MDF approaches introduced in [2] and [5], together with the MDC schemes proposed in [11] and [12] are realized in the same platform to serve as references. We test the proposed design and other architectures in the two-parallel, four-parallel, and eight-parallel scenario, respectively, and the radix-2, radix-2^2, and radix-2^3 algorithm are considered in each case with a word length of 16 bits. The experimental results are listed in Table II, where the occupation of slices, DSP48E blocks and block RAMs is evaluated after the place-and-route operation, while the computing latency in clock cycles is obtained from the simulations in ModelSim SE 6.5c. The maximum reachable clock frequency, which is applied to determine throughputs in the table, is mentioned in the ISE implementation report with automatic timing constraints generated by the software.

In the test, we utilize IP soft cores to implement both general complex multipliers and constant ones, and each component is constituted of three DSP48E units. In consequence, the total amount of multipliers can be determined precisely via scaling the occupied DSP48Es in Table II with 3, which agree with the theoretical requirements presented in Table I. Some features on the consumption of multipliers are embodied in the statistics. First, compared with the MDF scheme, the edge of M^{2}DF structure will become more distinct when k takes odd values for radix-2^k computing. In such cases, there are considerable multipliers experiencing a 50% utilization ratio

TABLE II
HARDWARE COST AND PERFORMANCE OF THE P-PARALLEL N-POINT FFT PROCESSORS FOR 16 bits

Configurations†	Structures	Consumption of Slices			Consumption of RAMs (18k×1)			System Performance	
		Slice LUTs	*Slice registers*	*DSP48E*	*Streams folding*	*Data reordering*	*Overall*	*Latency (clock cycles)*	*Throughput (Msamples/s)*
Radix-2, 2/512	M^{2}DF	1683	1998	21	2	2	5	530	610
	MDF, [2]	2256	2164	39	2	2	5	526	630
	MDC, [11]	1907	2085	21	3	2	6	718	624
Radix-2, 4/1024	M^{2}DF	3968	4710	45	4	4	10	534	1220
	MDF, [2]	5226	5140	81	4	4	10	528	1276
	MDC, [11]	4450	4852	48	6	4	12	722	1248
Radix-2, 8/2048	M^{2}DF	8844	9601	99	8	8	19	538	2440
	MDF, [2]	12035	10834	171	8	8	19	530	2520
	MDC, [11]	9813	9972	108	12	8	23	726	2496
Radix-2^2, 2/512	M^{2}DF	1572	1843	21	2	2	5	526	610
	MDF, [5]	2118	2026	21	2	2	5	520	638
	MDC, [12]	1836	1985	21	2	4	7	776	624
Radix-2^2, 4/1024	M^{2}DF	3671	4359	45	4	4	10	530	1220
	MDF, [5]	4854	4785	45	4	4	10	522	1276
	MDC, [12]	4169	4412	39	4	8	14	780	1248
Radix-2^2, 8/2048	M^{2}DF	8197	9008	99	8	8	19	534	2440
	MDF, [5]	11135	9892	99	8	8	19	524	2552
	MDC, [12]	9353	9586	84	8	16	27	784	2496
Radix-2^3, 2/512	M^{2}DF	1640	1888	21	2	2	5	534	584
	MDF, [5]	2270	2143	27	2	2	5	526	590
	MDC, [12]	1956	2079	21	2	4	7	780	580
Radix-2^3, 4/1024	M^{2}DF	3796	4432	45	4	4	10	538	1168
	MDF, [5]	5151	5028	57	4	4	10	528	1180
	MDC, [12]	4479	4683	48	4	8	14	784	1160
Radix-2^3, 8/2048	M^{2}DF	8589	9243	99	8	8	19	542	2336
	MDF, [5]	11705	10325	123	8	8	19	530	2360
	MDC, [12]	9816	9980	87	8	16	27	788	2320

† "Radix-2, 2/512" specifies the FFT processor adopt radix-2 algorithm to compte 512-point DFT with a parallelism of 2. This definition can be extended to the other items in the first column.

in radix-2^k stages with feedback-form implementation, which pave a way for the multiplier multiplexing in M^{2}DF structure. Next, the MDC and M^{2}DF scheme have their own superiorities at this metric, and the winner is attached closely to the concrete parallelism P and the factor of radix size k. On the occasion of $P \geqslant 2^k$, the MDC design will show a better performance since this prerequisite equips the multipliers in the MDC circuit with full utilization.

On the other hand, area expense is measured by the occupation of block RAMs. Corresponding to the metrics in Table I, the memories associated with streams folding and data reordering are listed individually in Table II. As shown, these records are in favor of the conclusion that either the M^{2}DF or MDF scheme has an advantage over its MDC competitor in area consumption. In addition, it is worth noting that actual memory costs presented in Table II exceed the relevant theoretical values throughout the test, this is because block memory cores do not achieve a full utilization, i.e., only a part of cells in each memory bank participate in the FFT computing.

Furthermore, since slice lookup tables (LUTs) serve as the arithmetic resources in FPGA, the relevant test results in Table II are further attached to concrete modules in the FFT processor to support a fine-grained analysis, as presented graphically in Fig. 8. These bar charts suggest that butterfly units account for a substantial cost of slice LUTs in contrast with other components. Thus, a decline in the number of butterfly units, as has been accomplished by the proposed M^{2}DF scheme, is qualified to bring in a considerable reduction of arithmetic resources. Another feature can be extracted from the graphs is that the MDF design requires approximate twofold as many LUT resources as other schemes to construct butterfly units. Note that the butterfly unit is composed primarily of complex adders, from this perspective the aforementioned character evidences the theoretical adder consumptions in Table I indirectly.

Finally, it can be found the experimental latencies differ from the theoretical values shown in Table I, and the reason consists of both sides: 1) the practical complex multipliers need several clock cycles to obtain the correct results, which leads to a rise of overall latencies and 2) some additional registers are embedded in the circuit to shorten the critical paths, thus the group delay of FFT processors involves a further increase. With respect to the throughputs, both the

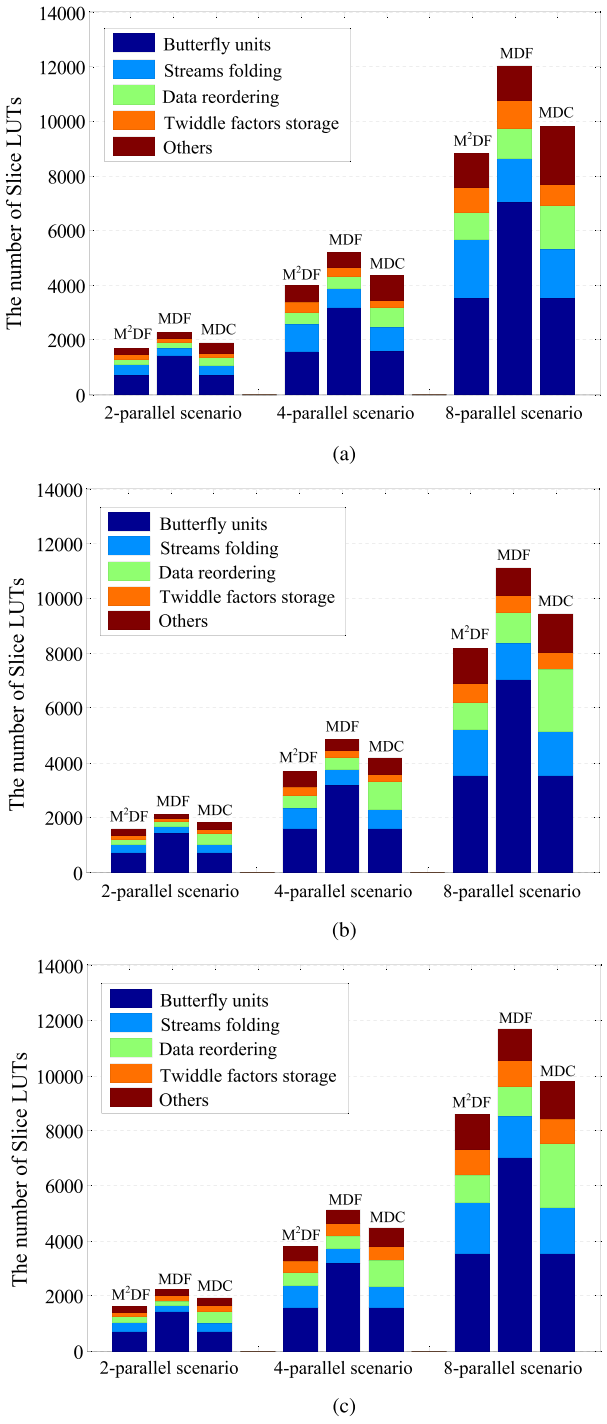


Fig. 8. Correlating the consumed slice LUTs presented in Table II to specific modules of the parallel FFT processor. (a) Radix-2 FFT. (b) Radix-2^2 FFT. (c) Radix-2^3 FFT.

proposed schemes and the references have similar performance throughout the various configurations. This feature suggests that the critical path in M^{2}DF is comparable with that belonging to MDF or MDC circuits.

V. Conclusion

In spite of the low memory requirement and short computing delay, conventional MDF schemes suffer from the inefficient use of adders and multipliers in practical applications. This paper recommends an M^{2}DF structure to grapple with this obstacle, which eliminates the standby time of arithmetic modules in feedback architectures by integrating DIT operations into the DIF-operated computing units. According to the theoretical analysis and experimental results, the M^{2}DF design inherits the strengths of feedback structures while significantly curbing the overexploitation of arithmetic resources. This outstanding feature enables the M^{2}DF structure to be an efficient alternative to the MDF scheme. On the other hand, the M^{2}DF and the MDC scheme consume the same amount of adders, while they have their own merits in multiplier overhead. However, when the computing delay and memory resources are primary concerns, the M^{2}DF approach will be more hardware friendly than the MDC scheme.

References

[1] G. Bi and E. V. Jones, "A pipelined FFT processor for word-sequential data," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 12, pp. 1982–1985, Dec. 1989.

[2] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementations," *IEEE Trans. Comput.*, vol. C-33, no. 5, pp. 414–426, May 1984.

[3] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT processor," in *Proc. IEEE Custom Integr. Circuits Conf.*, May 1998, pp. 131–134.

[4] C.-C. Wang, J.-M. Huang, and H.-C. Cheng, "A 2K/8K mode small-area FFT processor for OFDM demodulation of DVB-T receivers," *IEEE Trans. Consum. Electron.*, vol. 51, no. 1, pp. 28–32, Feb. 2005.

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

[6] L. Jia, Y. Gao, J. Isoaho, and H. Tenhunen, "A new VLSI-oriented FFT algorithm and implementation," in *Proc. 11th Annu. IEEE Int. Adv. Syst. Integr. Circuits Conf.*, Rochester, NY, USA, Sep. 1998, pp. 337–341.

[7] T. Lenart and V. Öwall, "A 2048 complex point FFT processor using a novel data scaling approach," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Bangkok, Thailand, May 2003, pp. IV-45–IV-48.

[8] W.-C. Yeh and C.-W. Jen, "High-speed and low-power split-radix FFT," *IEEE Trans. Signal Process.*, vol. 51, no. 3, pp. 864–874, Mar. 2003.

[9] C. Cheng and K. K. Parhi, "High-throughput VLSI architecture for FFT computation," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 54, no. 10, pp. 863–867, Oct. 2007.

[10] Y.-N. Chang, "An efficient VLSI architecture for normal I/O order pipeline FFT design," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55, no. 12, pp. 1234–1238, Dec. 2008.

[11] M. Ayinala, M. Brown, and K. K. Parhi, "Pipelined parallel FFT architectures via folding transformation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 6, pp. 1068–1081, Jun. 2012.

[12] M. Garrido, J. Grajal, M. A. Sanchez, and O. Gustafsson, "Pipelined radix-2^k feedforward FFT architectures," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 1, pp. 23–32, Jan. 2013.

[13] K.-J. Yang, S.-H. Tsai, and G. C. H. Huang, "MDC FFT/IFFT processor with variable length for MIMO-OFDM systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 4, pp. 720–731, Apr. 2013.

[14] J. Lee, H. Lee, S.-I. Cho, and S.-S. Choi, "A high-speed, low-complexity radix-2^4 FFT processor for MB-OFDM UWB systems," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2006, pp. 210–213.

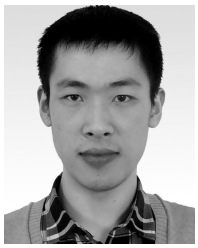
[15] S.-I. Cho, K.-M. Kang, and S.-S. Choi, "Implemention of 128-point fast Fourier transform processor for UWB systems," in *Proc. Int. Wireless Commun. Mobile Comput. Conf.*, Aug. 2008, pp. 210–213.

[16] H. Liu and H. Lee, "A high performance four-parallel 128/64-point radix-2^4 FFT/IFFT processor for MIMO-OFDM systems," in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, Nov./Dec. 2008, pp. 834–837.

[17] N. Li and N. P. Van Der Meijs, "A radix-2^2 based parallel pipeline FFT processor for MB-OFDM UWB system," in *Proc. IEEE Int. SOC Conf.*, Sep. 2009, pp. 383–386.

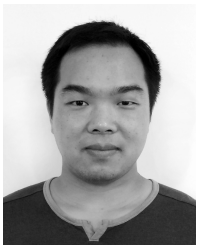
[18] S.-N. Tang, J.-W. Tsai, and T.-Y. Chang, "A 2.4-GS/s FFT processor for OFDM-based WPAN applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 6, pp. 451–455, Jun. 2010.

[19] 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.
[20] 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.
[21] T. Cho and H. Lee, "A high-speed low-complexity radix-2^5 modified FFT processor for high rate WPAN applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 1, pp. 187–191, Jan. 2013.
[22] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. Hoboken, NJ, USA: Wiley, 1999.
[23] A. Cortes, I. Velez, and J. F. Sevillano, "Radix r^k FFTs: Matrical representation and SDC/SDF pipeline implementation," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2824–2839, Jul. 2009.



Jian Wang received the B.S. degree in communication engineering from the National University of Defense Technology, Changsha, China, in 2012, where he is currently pursuing the Ph.D. degree with the School of Electronic Science and Engineering.

His current research interests include special purpose VLSI processor architectures with emphasis on VLSI design for the kernel modules in communication systems.



Chunlin Xiong received the B.S. degree in communication engineering from Zhejiang University, Hangzhou, China, in 2003, and the Ph.D. degree in information and communication engineering from the National University of Defense Technology (NUDT), Changsha, China, in 2009.

He has been with NUDT since 2010, where he is currently a Lecturer. His current research interests include the design of digital VLSI circuits and systems, error-correction coding, and signal processing for wireless communications.

Kangli Zhang received the B.S. degree in communication engineering and the M.S. degree in information and communication engineering from the National University of Defense Technology (NUDT), Changsha, China, in 2011 and 2013, respectively, where she is currently pursuing the Ph.D. degree with the School of Electronic Science and Engineering.

Her current research interests include signal processing and the design of VLSI circuits and systems.

Jibo Wei (M'04) received the B.S. and M.S. degrees from the National University of Defense Technology (NUDT), Changsha, China, in 1989 and 1992, respectively, and the Ph.D. degree from Southeast University, Nanjing, China, in 1998, all in electronic engineering.

He is currently the Director and a Professor with the Department of Communication Engineering, NUDT. His current research interests include design and optimization of communication systems and signal processing in communications, in particular, multiple input and multiple output, multicarrier transmission, cooperative communication, and cognative network.

Prof. Wei is the member of the IEEE Communication Society and the IEEE Vehicular Technology Society. He is also the Senior Member of the China Institute of Communications and Electronics, and an Editor of the *Journal on Communications*.