Fixed-Point Analysis and Parameter Optimization of the Radix-2^k Pipelined FFT Processor

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

***Abstract*—The radix-2^k algorithm plays a crucial role in the pipelined implementation of fast Fourier transform (FFT). This paper presents a fixed-point analysis and hardware evaluation of radix-2^k FFT under the framework of the single-path delay feedback (SDF) and multi-path delay commutator (MDC) pipelined structure. The investigation is carried out with variable operating word-lengths to ensure the generality. Furthermore, the main streams to fulfill FFT coefficients weighting, namely, the approach using complex multipliers and the one adopting memoryless CORDIC units, are both considered in the analysis. Based on these derivations, a joint optimization of radix-2^k algorithm and operating word-length is discussed to achieve a reasonable trade-off between computational accuracy and hardware expenditure. Simulations and experiments indicates that the derived SQNR is reliable to unfold the quantization effects of fixed-point radix-2^k FFT. In addition, the proposed joint optimization strategy is capable of providing better solutions to implement the radix-2^k FFT processor efficiently.**

***Index Terms*—Fast Fourier transform (FFT), fixed-point accuracy, multi-path delay commutator (MDC), radix-2^k algorithm, single-path delay feedback (SDF).**

I. Introduction

THE discrete Fourier transform (DFT) is an essential tool in digital signal processing. It has also seen broader usage in modern digital communications due to the adoption of orthogonal frequency division multiplexing (OFDM) technique in leading communication standards. Starting with the pioneering Cooley-Tukey algorithm [1], plenty of fast Fourier transform (FFT) schemes have been proposed to boost the efficiency of DFT computing, including the radix-r [2], split-radix [2] and the extensively-used radix-r^k algorithm [3].

When implementing the FFT algorithm using fixed-point arithmetic, there is an inherent accuracy problem due to the finite word-length used. Traditionally, the studies focused on evaluating the accuracy of the FFT module using integer multipliers. In specific, [4], [5] considered the quantization loss for radix-2 FFT computing using identical word-length throughout the process, while similar work was done for split-radix in [6]. In these literatures, the quantization errors were modeled as a uniformly-distributed white noise, while the reasonability is verified later in [7]. To achieve a better trade-off between the hardware consumption and the computational accuracy, [8]–[11] further investigated quantization property of the radix-r or split-radix FFT module with varying operating word-length, based on which a simple criterion of word-length optimization was proposed to yield comparable accuracy with fewer bit budget. Recently, the coordinate rotation digital computer (CORDIC) serves as a feasible alternative to the integer multiplier in FFT processors. In this scenario, the overall quantization noise was rethought in [12]–[14] by integrating the noise incurred in the precision-limited CORDIC operations to the analytical model and afterwards, the research findings were attached to the optimization of FFT architectures.

Compared to other ingenious schemes, the radix-2^k algorithm is more hardware-friendly in constructing the pipelined FFT structures, since small radix results in a simple butterfly unit, and the order k contributes to facilitate the twiddle factor multiplications. Numerous of state-of-the-art pipelined processors, whether adopting the feed forward approach [15]–[18] or the feedback scheme [19]–[25], give preference to the radix-2^k computing strategy. Nevertheless, the existing fixed-point analysis of radix-2^k scheme reveals insufficiency to cope with the issues in practical applications, which has motivated the first part of research in this paper. To ensure the generality of the conclusions, we concentrate on the analysis of *extended* radix-2^k algorithm with varying operating word-length, i.e., both the algorithm order k and the bit-width are configurable for the computational stages of the pipelined FFT processor.

In addition to the accuracy, memory cost is also an important metric to evaluate the FFT module. For radix-2^k architectures equipped with integer multipliers, [15]–[23] provide a rough estimate of memory expenditure without considering the cost of twiddle factors storage. This work has been extended in [3], [24], [25] by incorporating the memories used for storing twiddle factors into the overall cost. Based on these existing studies, we further generalize the issue to the extended radix-2^k FFT scenario, where the multi-path delay commutator (MDC) [15] and single path delay feedback (SDF) [19] serve as the representative to cover the topics related to the feed-forward and feed-back radix-2^k pipelined processor. It is noteworthy that the computational precision is interrelated with the memory occupation. However, boosting the accuracy runs contrary to de-

Manuscript received September 29, 2014; revised February 14, 2015 and April 23, 2015; accepted June 07, 2015. Date of publication June 19, 2015; date of current version August 13, 2015. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Joseph Cavallaro. This work was supported in part by this lab the Natural Science Foundation of China (NSFC) under Grants 91338105.

J. Wang, C. Xiong, and K. Zhang 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).

J. Wei is with the School of Electronic Science and Engineering, National University of Defense Technology, Changsha, 410073, China, and also with the Science and Technology on Information Transmission and Dissemination in Communication Networks Lab (e-mail: 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/TSP.2015.2447500

clining the memory cost. Taking this fact into account, an optimization strategy to strike a reasonable balance between these often conflicting objectives is proposed, which serves as another contribution of this paper.

The rest of this paper is organized as follows. The introduction of extended radix-2^k algorithm is presented in Section II using the matricial factorization approach. Afterwards, Section III provides a fixed-point analysis and hardware evaluation for the extended radix-2^k algorithm with varying operating word-length. Based on these efforts, Section IV discusses the joint optimization of radix-2^k algorithm and the word-length to achieve a reasonable trade-off between the computational precision and the hardware efficiency, which is followed by the simulation analysis and experiment verifications. Finally, we conclude the work in Section V.

Notation: Boldfaced letters are used for vectors or matrices. $\mathbf{I}_m$ represents the m-square identity matrix, $\mathbf{1}_m$ and $\mathbf{0}_m$ denote the m-dimension all-one and all-zero row vector, respectively. Transpose, conjugate transpose, inverse and trace of a matrix $\mathbf{A}$ are represented by $\mathbf{A}^{\mathrm{T}}$, $\mathbf{A}^{\mathrm{H}}$, $\mathbf{A}^{-1}$ and $tr(\mathbf{A})$, respectively. $\mathbf{A}\otimes\mathbf{B}$ returns the Kronecker product of two matrices and $\prod_{i=1}^{m}\mathbf{A}_i=(\ldots((\mathbf{A}_1\cdot\mathbf{A}_2)\cdot\mathbf{A}_3)\cdot\ldots\cdot\mathbf{A}_m)$. $\mathbb{E}[\cdot]$ denotes the statistical expectation. $\|\mathbf{x}\|_1$ computes the ℓ_1-norm of a vector $\mathbf{x}$ and the "power function" is defined as $\mathcal{P}(\mathbf{x})=\mathbb{E}\left[tr(\mathbf{x}\mathbf{x}^{\mathrm{H}})\right]$. $\mathbb{N}$ are the set of natural numbers including the zero element and $\mathbb{Z}_+$ denote the set of positive integers. $\mathbb{N}^m$ and $\mathbb{Z}_+^m$ represent the space of m-dimension vectors with natural number and positive integer entries, respectively. The operation $\mathrm{mod}(x,2)$ is denoted as $\underline{x}$ for simplicity. Finally, the sign function $\mathrm{sgn}(\cdot)$ equals -1 when the operand is negative, and it equals 0 when the operand is zero.

II. Metricial Representation of the Extended Radix-2^k Algorithm

The DFT can be expressed metrically as

$$\mathbf{y}=\mathbf{T}_N\mathbf{x}_0, \tag{1}$$

where $N=2^L$ and $\mathbf{x}_0=(x_0\,x_1\cdots x_{N-1})^{\mathrm{T}}$. The *$N$-point DFT matrix* $\mathbf{T}_N$ is the size N square matrix with entries $(\mathbf{T}_N)_{u,v}=e^{-j2\pi uv/N}$, $u,v\in\{0,\ldots,N-1\}$. Assume that the 2^L-point DFT is implemented by M cascaded computation stages adopting radix-2^{k_1}, radix-2^{k_2}, $\cdots$, radix-2^{k_M} algorithm respectively, $\mathbf{T}_N$ can be factorized as [3]

$$\mathbf{T}_N=\mathbf{\Lambda}_N\cdot\prod_{i=0}^{M-1}\mathbf{H}\left(c_{M-i},k_{M-i}\right), \tag{2}$$

where the *reordering matrix* $\mathbf{\Lambda}_N$ represents the N-point bit-reversed reordering operation. Denote $S\times S$ stride permutation matrix by $\mathbf{G}_S$, whose effect on an S-dimension vector is

$$\mathbf{G}_S\cdot(z_0\cdots z_{S-1})^{\mathrm{T}}=\left(z_0\;z_{S/2}\;z_1\;z_{S/2+1}\cdots z_{S/2-1}\;z_{S-1}\right)^{\mathrm{T}},$$

then $\mathbf{\Lambda}_N$ can be written as [3]

$$\mathbf{\Lambda}_N=\prod_{l=0}^{L-1}\left(\mathbf{I}_{2^l}\otimes\mathbf{G}_{N/2^l}\right). \tag{3}$$

The parameter c_m in (2) is given by

$$c_m=\sum_{i=1}^{m}k_i,\,m\in\{1,\ldots,M\}, \tag{4}$$

which is essentially the segmental accumulation of algorithm order k_i. $m=M$ leads to $c_M=\sum_{i=1}^{M}k_i=L$. The term $\mathbf{H}\left(c_m,k_m\right)$ in (2) describes the mth computation stage using radix-2^{k_m} algorithm. For the determined c_m, the underlying operations of $\mathbf{H}\left(c_m,k_m\right)$ vary with k_m. When $k_m>1$ and k_m is odd,

$$\begin{aligned}&\mathbf{H}(c_m,k_m)\\&=\mathbf{M}_3(c_m-k_m,k_m)\mathbf{B}_{\mathrm{F}}(c_m-1)\\&\quad\cdot\prod_{i=1}^{(k_m-1)/2}[\mathbf{M}_2(c_m-2i-1,2i+1)\mathbf{B}_{\mathrm{F}}(c_m-2i)\\&\quad\cdot\mathbf{M}_1(c_m-2i-1)\mathbf{B}_{\mathrm{F}}(c_m-2i-1)],\end{aligned} \tag{5}$$

where the *butterfly matrix* $\mathbf{B}_{\mathrm{F}}(u)$ applies the in-place radix-2 butterfly operations to the N-dimension sequence. It can be expressed in the form [3]

$$\mathbf{B}_{\mathrm{F}}(u)=\left(\mathbf{I}_{2^u}\otimes\mathbf{G}_{N/2^u}\right)^{-1}\left(\mathbf{I}_{N/2}\otimes\mathbf{T}_2\right)\left(\mathbf{I}_{2^u}\otimes\mathbf{G}_{N/2^u}\right) \tag{6}$$

with $\mathbf{T}_2$ denoting the 2-point DFT computation. In (5), the N-diagonal *multiplication matrices* $\mathbf{M}_1(u),\mathbf{M}_2(u,v)$ and $\mathbf{M}_3(u,v)$ weight the butterfly outputs with proper twiddle factors. To provide a further description of these coefficient-weighting matrices, let the *twiddle-factor matrix* $\mathbf{W}_S^{(u)}$ be

$$\mathbf{W}_S^{(u)}=\mathrm{quasidiag}\left(\mathbf{I}_{S/u}\;\boldsymbol{w}_{S/u}\;\boldsymbol{w}_{S/u}^2\cdots\boldsymbol{w}_{S/u}^{u-1}\right), \tag{7}$$

where $u\le S$ and both of the parameters are a power of two, $\boldsymbol{w}_{S/u}=\mathrm{diag}\left(1\;e^{-j2\pi/S}\;e^{-j2\cdot2\pi/S}\cdots e^{-j(S/u-1)\cdot2\pi/S}\right)$. In this way $\mathbf{M}_1(u),\mathbf{M}_2(u,v)$ and $\mathbf{M}_3(u,v)$ in (5) can be generated through permuting and extending the entries of the corresponding twiddle-factor matrices, as shown in (8a)–(8c) at the bottom of the page.

$$\mathbf{M}_1(u)=\mathbf{I}_{2^u}\otimes\mathbf{W}_4^{(2)}\otimes\mathbf{I}_{N/2^{u+2}}, \tag{8a}$$

$$\mathbf{M}_2(u,v)=\mathbf{I}_{2^u}\otimes\left((\mathbf{G}_4\otimes\mathbf{I}_{2^{v-2}})\cdot\mathbf{W}_{2^v}^{(4)}\cdot(\mathbf{G}_4\otimes\mathbf{I}_{2^{v-2}})\right)\otimes\mathbf{I}_{N/2^{u+v}}, \tag{8b}$$

$$\mathbf{M}_3(u,v)=\mathbf{I}_{2^u}\otimes\left(\prod_{i=1}^{v}\left(\mathbf{G}_{2^i}\otimes\mathbf{I}_{N/2^{u+i}}\right)\cdot\mathbf{W}_{N/2^u}^{(2^v)}\cdot\prod_{i=1}^{v}\left(\mathbf{G}_{2^i}\otimes\mathbf{I}_{N/2^{u+i}}\right)\right). \tag{8c}$$

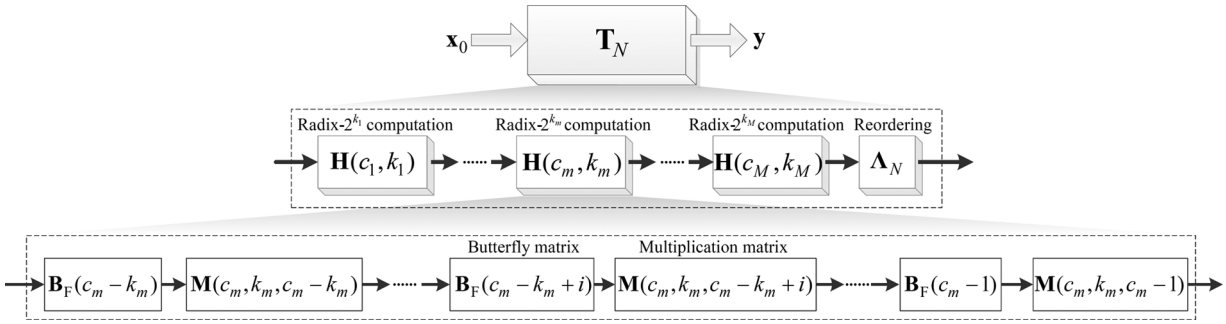


Fig. 1. Matricial representation of the DFT transform matrix and the radix-2^{k_m} computing scheme.

Similar to (5), the internal operation of $\mathbf{H}(c_m,k_m)$ for even k_m greater than 2 is

$$\begin{aligned}&\mathbf{H}(c_m,k_m)\\&=\mathbf{M}_3(c_m-k_m,k_m)\mathbf{B}_\mathrm{F}(c_m-1)\cdot\mathbf{M}_1(c_m-2)\mathbf{B}_\mathrm{F}(c_m-2)\\&\quad\cdot\prod_{i=1}^{k_m/2-1}[\mathbf{M}_2(c_m-2i-2,2i+2)\mathbf{B}_\mathrm{F}(c_m-2i-1)\\&\quad\cdot\mathbf{M}_1(c_m-2i-2)\mathbf{B}_\mathrm{F}(c_m-2i-2)].\end{aligned}\tag{9}$$

In particular, $k_m=1$ leads to $(k_m-1)/2=0$. On this occasion (5) requires to be revised as

$$\mathbf{H}(c_m,1)=\mathbf{M}_3(c_m-1,1)\cdot\mathbf{B}_\mathrm{F}(c_m-1).\tag{10}$$

In the same way, $k_m=2$ would simplify (9) to

$$\begin{aligned}\mathbf{H}(c_m,2)=\mathbf{M}_3(c_m-2,2)\cdot\mathbf{B}_\mathrm{F}(c_m-1)\\\cdot\mathbf{M}_1(c_m-2)\cdot\mathbf{B}_\mathrm{F}(c_m-2).\end{aligned}\tag{11}$$

It can be concluded from (5) and (9)–(11) that for arbitrary $k_m\in\mathbb{N}_+$, $\mathbf{H}(c_m,k_m)$ consists of k_m butterfly matrices and the same amount of multiplication matrices. Consequently, these equations can be integrated as the form

$$\mathbf{H}(c_m,k_m)=\prod_{i=1}^{k_m}\mathbf{M}(c_m,k_m,c_m-i)\cdot\mathbf{B}_\mathrm{F}(c_m-i),\tag{12}$$

where the term $\mathbf{M}(c_m,k_m,c_m-i), i\in\{1,2,\ldots,k_m\}$ in (12) is connected with the defined multiplication matrices in (8a)–(8c) through the following criterion

$$\begin{aligned}&\mathbf{M}(c_m,k_m,c_m-i)\\&=\begin{cases}\mathbf{M}_1(c_m-i), & \text{if } i\in\Psi_1(k_m),k_m\geqslant 2\\ \mathbf{M}_2(c_m-i-1,i+1), & \text{if } i\in\Psi_2(k_m),k_m\geqslant 3\\ \mathbf{M}_3(c_m-k_m,k_m), & \text{if } i=1\end{cases}.\end{aligned}\tag{13}$$

The set $\Psi_1(k_m)$ and $\Psi_2(k_m)$ in (13) are defined as

$$\Psi_1(k_m)=\left\{2+\underline{k_m},4+\underline{k_m},\ldots,k_m\right\}\quad(k_m\geqslant 2),\tag{14a}$$

$$\Psi_2(k_m)=\left\{3-\underline{k_m},5-\underline{k_m},\ldots,k_m-1\right\}\quad(k_m\geqslant 3)\tag{14b}$$

with $\underline{k_m}=\mathrm{mod}(k_m,2)$ in the expressions. Substitute (12) into (2) to obtain

$$\begin{aligned}\mathbf{T}_N=\mathbf{\Lambda}_N\cdot\prod_{i=0}^{M-1}\prod_{j=1}^{k_{M-i}}[\mathbf{M}(c_{M-i},k_{M-i},c_{M-i}-j)\\\cdot\mathbf{B}_\mathrm{F}(c_{M-i}-j)].\end{aligned}\tag{15}$$

As shown, $\mathbf{T}_N$ can be resolved into $\sum_{i=1}^{M}k_i=L$ butterfly matrices alternating with the same amount of multiplication matrices. Equations (12) and (15) provide a matrical representation of the radix-2^{k_m} computing stage and the DFT transform matrix, respectively. The foregoing derivation is summarized briefly in Fig. 1. Moreover, the introduced reordering matrix $\mathbf{\Lambda}_N$, butterfly matrix $\mathbf{B}_\mathrm{F}(u)$ and multiplication matrices $\mathbf{M}_1(u),\mathbf{M}_2(u,v),\mathbf{M}_3(u,v)$ offer some useful properties to simplify the later discussion. Here we summary these features as the following proposition:

Proposition 1: $\mathbf{\Lambda}_N$, $\mathbf{M}_1(u),\mathbf{M}_2(u,v)$, $\mathbf{M}_3(u,v)$ and the normalized butterfly matrix $\mathbf{B}_\mathrm{F}(u)/\sqrt{2}$ are all unitary matrices, i.e., $\mathbf{B}_\mathrm{F}(u)\mathbf{B}_\mathrm{F}^\mathrm{H}(u)=2\mathbf{I}_N$, $\mathbf{\Lambda}_N\mathbf{\Lambda}_N^\mathrm{H}=\mathbf{M}_1(u)\mathbf{M}_1^\mathrm{H}(u)=\mathbf{M}_2(u,v)\mathbf{M}_2^\mathrm{H}(u,v)=\mathbf{M}_3(u,v)\mathbf{M}_3^\mathrm{H}(u,v)=\mathbf{I}_N$.

Proof: See Appendix B. ■

III. Fixed-Point Analysis and Hardware Evaluation for the Pipelined Radix-2^k Structure

When translating $\mathbf{T}_N$ into fixed-point radix-2^k structures, the accuracy and hardware consumption account for the main consideration. These two important issues will be involved in this section with an $N=2^L$-point FFT, whose relevant structure consists of M stages adopting radix-2^{k_1}, radix-2^{k_2}, $\cdots$, radix-2^{k_M} algorithm, respectively.

A. Fixed-Point Analysis of the Extended Radix-2^k FFT

According to (15), the single parameter of $\mathbf{B}_\mathrm{F}(\cdot)$ varies from 0 to $L-1$ and is irrelevant to the concrete $k_m, m=1,\ldots M$. As the term $\mathbf{B}_\mathrm{F}(l)$ correlates to the $l+1$th butterfly unit in the hardware, the feature of $\mathbf{B}_\mathrm{F}(l)$ indicates that the operations of butterfly units are not affected by the selection of the radix-2^k algorithm. In terms of the quantization property of the butterfly units, the operating word-length plays a crucial role. Assume that the word-length (for both real and imaginary part) of the initial input and final output are w_ini and w_fin, respectively. Denote the *operating word-length vector* $\mathbf{w}=(w_0\ w_1\cdots w_{L-1})$, where the $l+1$th butterfly unit processes the data using w_l-bit representations and the word-length remains unchanged when the stream passes through the complex multipliers. In addition, $w_\mathrm{ini}\leqslant w_0\leqslant\cdots\leqslant w_{L-1}=w_\mathrm{fin}$ to maintain a non-decreasing computing accuracy throughout the process. $\forall j\in\{0,\ldots,L-1\}$, if $w_j-w_{j-1}>0$ (w_{j-1} is w_ini exactly for $j=0$), then the $j+1$th, $\ldots$, $j+w_j-w_{j-1}$th butterfly unit do not require to scale the inputs. Based on this fact, the scaling operations executed in butterfly units can be correlated with the specified operating

word-length. Let $\boldsymbol{\delta} = (\delta_0\ \delta_1 \cdots \delta_{L-1})$, where $\delta_l \in \{0, 1\}$ is determined as follows

$$\delta_l = \begin{cases} 1 + \mathrm{sgn}(w_{\mathrm{ini}} - w_l), & \text{if } l = 0 \\ 1 + \mathrm{sgn}\left(w_{\mathrm{ini}} - w_l + l - \sum_{j=0}^{l-1} \delta_j\right), & \text{else} \end{cases} \tag{16}$$

$\delta_l = 1$ indicates the input of the $l+1$th butterfly unit should be scaled by $1/2$ to avoid the potential overflow. Conversely, $\delta_l = 0$ suggests the scaling operation is not essential. Let $\tilde{\mathbf{x}}_l$ and $\tilde{\mathbf{y}}_l$ be the fixed-point input and output of the $l+1$th butterfly unit, the internal arithmetic considering the potential scaling operation can be expressed as

$$\tilde{\mathbf{y}}_l = \tilde{\mathbf{B}}_{\mathrm{F}}(l)\tilde{\mathbf{x}}_l = \left(\frac{1}{2}\right)^{\delta_l} \mathbf{B}_{\mathrm{F}}(l)\tilde{\mathbf{x}}_l + \delta_l \mathbf{B}_{\mathrm{F}}(l)\boldsymbol{\epsilon}_{s,l}, \tag{17}$$

where the term $\tilde{\mathbf{B}}_{\mathrm{F}}(\cdot)$ can be viewed as the implemented version of $\mathbf{B}_{\mathrm{F}}(\cdot)$. In practical systems, the scaling of samples is fulfilled by a 1-bit right shift along with the rounding operation, which incurs the zero-mean *roundoff noise* $\boldsymbol{\epsilon}_{s,l}$. Moreover, the entries of $\boldsymbol{\epsilon}_{s,l}$ are assumed to be independent identically distributed with variance $\sigma_{s,l}^2$. Therefore, the power of $\boldsymbol{\epsilon}_{s,l}$ can be obtained using the power function

$$\mathcal{P}(\boldsymbol{\epsilon}_{s,l}) = tr\left\{\mathbb{E}(\boldsymbol{\epsilon}_{s,l}\boldsymbol{\epsilon}_{s,l}^{\mathrm{H}})\right\} = N\sigma_{s,l}^2. \tag{18}$$

In (15), the multiplication matrix $\mathbf{M}(c_m, k_m, l)$ serves as the intermediary between $\mathbf{B}_{\mathrm{F}}(l)$ and $\mathbf{B}_{\mathrm{F}}(l+1)$, where the subscript $m = \arg\min_{m\in\{1,\ldots,M\}}\{c_m > l\}$. In the hardware circuit, correspondingly, the $l+1$th butterfly unit is connected with the $l+2$th butterfly unit through the operation

$$\tilde{\mathbf{x}}_{l+1} = \tilde{\mathbf{M}}(c_m, k_m, l)\tilde{\mathbf{y}}_l = \mathbf{M}(c_m, k_m, l)\tilde{\mathbf{y}}_l + \boldsymbol{\epsilon}_{c,l}, \tag{19}$$

where $\tilde{\mathbf{x}}_{l+1}$ represents the input of $l+2$th butterfly unit, while $\tilde{\mathbf{y}}_l$ is the output of the $l+1$th one. $\tilde{\mathbf{M}}(\cdot,\cdot,\cdot)$ in (19) is the fixed-pointed version of $\mathbf{M}(\cdot,\cdot,\cdot)$. The operation results in *truncation noise* $\boldsymbol{\epsilon}_{c,l}$ as the fixed-point multiplications suffer from truncations. More precisely, the complex multiplication would contaminate the FFT output on condition that the applied coefficient belongs to non-trivial twiddle factors. For trivial twiddle factors ± 1 and $\pm j$, the relevant results do not bring in additional noise. The analysis reveals the fact that $\boldsymbol{\epsilon}_{c,l}$ contains zero entries, which differs from the roundoff noise $\boldsymbol{\epsilon}_{s,l}$. To determine the power of truncation noise, the number of non-trivial twiddle factors in the multiplication matrices should be given precedence.

Attention is firstly allotted to the twiddle factor matrix $\mathbf{W}_S^{(u)}$ in (7), where non-trivial twiddle factors account for the amount

$$\lambda_0 = \begin{cases} 0, & \text{if } S = u \text{ or } u = 1 \\ S - S/u - u, & \text{else} \end{cases}. \tag{20}$$

Based on this conclusion, $\lambda_1(u)$, $\lambda_2(u, v)$ and $\lambda_3(u, v)$, which respectively corresponds to the number of non-trivial twiddle factors in $\mathbf{M}_1(u)$, $\mathbf{M}_2(u, v)$ and $\mathbf{M}_3(u, v)$, can be determined by further considering the replication feature of Kronecker product. According to (8a)–(8c), we have

$$\lambda_1(u) \equiv 0, \tag{21}$$

$$\lambda_2(u, v) = \begin{cases} 0, & \text{if } v < 3 \\ N\left(\frac{3}{4} - \frac{4}{2^v}\right), & \text{else} \end{cases} \tag{22}$$

$$\lambda_3(u, v) = \begin{cases} 0, & \text{if } N = 2^{u+v} \\ N\left(1 - \frac{1}{2^v}\right) - 2^{u+v}, & \text{else} \end{cases}. \tag{23}$$

From (13), the amount of non-trivial twiddle factors included in $\mathbf{M}(c_m, k_m, l)$ can be calculated by replacing the u, v in (21)–(23) with concrete parameters. That is

$$\begin{aligned} \lambda(c_m, k_m, l) &= \lambda(c_m, k_m, c_m - i) \\ &= \begin{cases} N\left(\frac{3}{4} - \frac{1}{2^{i-1}}\right), & \text{if } i \in \Psi_2(k_m) \text{ and } k_m \geqslant 3 \\ N\left(1 - \frac{1}{2^{k_m}}\right) - 2^{c_m}, & \text{if } i = 1 \text{ and } m < M \\ 0, & \text{else} \end{cases} \end{aligned} \tag{24}$$

where $l \in \{c_m - k_m, \ldots, c_m - 1\}$ and $\Psi_2(k_m)$ is defined in (14b). **In the following derivations,** $\lambda(c_m, k_m, l)$ **is abbreviated to** $\lambda(l)$. The hidden parameters c_m and k_m can be determined uniquely for the given l, since the subscript m satisfies $m = \arg\min_{m\in\{1,\ldots,M\}}\{c_m > l\}$. Similar to the assumption of $\boldsymbol{\epsilon}_{s,l}$, the non-zero entries in $\boldsymbol{\epsilon}_{c,l}$ are considered to be i.i.d. variables with variance $\sigma_{c,l}^2$. In this way,

$$\mathcal{P}(\boldsymbol{\epsilon}_{c,l}) = tr\left\{\mathbb{E}(\boldsymbol{\epsilon}_{c,l}\boldsymbol{\epsilon}_{c,l}^{\mathrm{H}})\right\} = \lambda(l)\sigma_{c,l}^2. \tag{25}$$

Taking the quantization operations described in (17) and (19) into account, the noise-contaminated radix-2^{k_m} computation denoted by $\tilde{\mathbf{H}}(c_m, k_m)$ is shown in Fig. 2. Let $\tilde{\mathbf{x}}_{c_m-k_m} = \mathbf{x}_{c_m-k_m} + \mathbf{n}_{c_m-k_m}$ be the input of $\tilde{\mathbf{B}}_{\mathrm{F}}(c_m - k_m)$, where $\mathbf{x}_{c_m-k_m}$ is non-contaminated signal term while $\mathbf{n}_{c_m-k_m}$ represents the overall quantization noise. Then from (12), the output of $\tilde{\mathbf{H}}(c_m, k_m)$ denoted by $\tilde{\mathbf{x}}_{c_m}$ can be written as (26) at the bottom of the page.

$$\begin{aligned} \tilde{\mathbf{x}}_{c_m} &= \left[\prod_{i=1}^{k_m} \tilde{\mathbf{M}}(c_m, k_m, c_m - i)\tilde{\mathbf{B}}_{\mathrm{F}}(c_m - i)\right] \cdot \tilde{\mathbf{x}}_{c_m-k_m} \\ &= \left(\frac{1}{2}\right)^{\sum_{u=c_m-k_m}^{c_m-1} \delta_u} \left[\prod_{i=1}^{k_m} \mathbf{M}(c_m, k_m, c_m - i)\mathbf{B}_{\mathrm{F}}(c_m - i)\right] \cdot (\mathbf{x}_{c_m-k_m} + \mathbf{n}_{c_m-k_m}) \\ &\quad + \sum_{l=c_m-k_m}^{c_m-2} \left(\frac{1}{2}\right)^{\sum_{u=l+1}^{c_m-1} \delta_u} \left[\prod_{j=1}^{c_m-l-1} \mathbf{M}(c_m, k_m, c_m - j)\mathbf{B}_{\mathrm{F}}(c_m - j)\right] \cdot [\delta_l \mathbf{M}(c_m, k_m, l)\,\mathbf{B}_{\mathrm{F}}(l)\boldsymbol{\epsilon}_{s,l} + \boldsymbol{\epsilon}_{c,l}] \\ &\quad + \delta_{c_m-1}\mathbf{M}(c_m, k_m, c_m - 1)\mathbf{B}_{\mathrm{F}}(c_m - 1)\boldsymbol{\epsilon}_{s,c_m-1} + \boldsymbol{\epsilon}_{c,c_m-1}, \end{aligned} \tag{26}$$

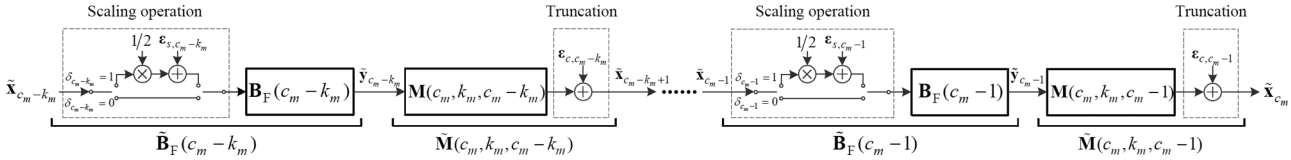


Fig. 2. Internal operations of radix-2^{k_m} computing stage when considering the quantization operations.

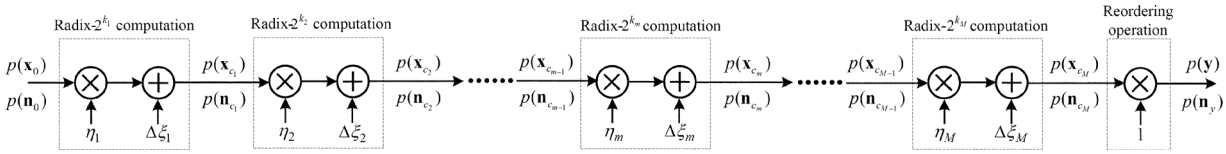


Fig. 3. Equivalent power propagating model of the mixed radix-2^k FFT processor.

Analogous to $\tilde{\mathbf{x}}_{c_m-k_m}$, $\tilde{\mathbf{x}}_{c_m}$ is also composed of the signal term $\mathbf{x}_{c_m}$ and the noise term $\mathbf{n}_{c_m}$, where (26) offers a fine-grained illustration the two components, respectively. By applying Proposition 1, (18) and (25) to (26), the powers of $\mathbf{x}_{c_m}$ and $\mathbf{n}_{c_m}$ are

$$\begin{aligned}\mathcal{P}(\mathbf{x}_{c_m}) &= \eta_m \cdot \mathcal{P}(\mathbf{x}_{c_m-k_m}) \\ &= \left(\frac{1}{4}\right)^{\sum_{u=c_m-k_m}^{c_m-1}\delta_u} 2^{k_m}\mathcal{P}(\mathbf{x}_{c_m-k_m}), \qquad (27a)\\ \mathcal{P}(\mathbf{n}_{c_m}) &= \eta_m \cdot \mathcal{P}(\mathbf{n}_{c_m-k_m}) + \Delta\xi_m \\ &= \left(\frac{1}{4}\right)^{\sum_{u=c_m-k_m}^{c_m-1}\delta_u} 2^{k_m}\cdot \mathcal{P}(\mathbf{n}_{c_m-k_m}) \\ &\quad + \sum_{l=c_m-k_m}^{c_m-2}\left(\frac{1}{4}\right)^{\sum_{u=l+1}^{c_m-1}\delta_u} 2^{c_m-l-1}\left[2\delta_l N\sigma_{s,l}^2+\lambda(l)\sigma_{c,l}^2\right] \\ &\quad + 2\delta_{c_m-1}N\sigma_{s,c_m-1}^2 + \lambda(c_m-1)\sigma_{c,c_m-1}^2. \qquad (27b)\end{aligned}$$

In the derivation $tr\{\mathbb{E}(\varepsilon_{s,l_1}\varepsilon_{s,l_2}^{\mathrm{H}})\} = tr\{\mathbb{E}(\varepsilon_{c,l_1}\varepsilon_{c,l_2}^{\mathrm{H}})\} = 0$ for $l_1 \neq l_2$, since the quantization errors stemming from different modules are assumed to be independent. Equation (27a) and (27b) indicate that $\tilde{\mathbf{H}}(c_m, k_m)$ weakens all the input signals with an attenuation coefficient $\eta_m = (1/4)^{\sum_{u=c_m-k_m}^{c_m-1}\delta_u}\cdot 2^{k_m}$. Meanwhile, it injects additional noise $\Delta\xi_m$ into the intermediate results during the execution.

From the perspective of power propagation, (27a) and (27b) establish an recurrence relation between $\tilde{\mathbf{H}}(c_m, k_m)$ and its adjacent computation stages $\tilde{\mathbf{H}}(c_{m-1}, k_{m-1})$, $\tilde{\mathbf{H}}(c_{m+1}, k_{m+1})$. This recurrence relation is clearly illustrated in Fig. 3, which contributes to conduct a power analysis of the FFT output $\tilde{\mathbf{y}} = \mathbf{y} + \mathbf{n}_y$. Let $\tilde{\mathbf{x}}_0 = \mathbf{x}_0 + \mathbf{n}_0$ be the initial fixed-point FFT input, where the entries of noise term $\mathbf{n}_0$ are modeled as zero-mean i.i.d. variables with variance $\sigma_{n_0}^2$, hence $\mathcal{P}(\mathbf{n}_0) = N\sigma_{n_0}^2$. Then at the output terminal, the power of valid output $\mathbf{y}$ and the quantization noise $\mathbf{n}_y$ can be derived as

$$\mathcal{P}(\mathbf{y}) = \left(\prod_{m=1}^{M}\eta_m\right)\cdot\mathcal{P}(\mathbf{x}_0) = \left(\frac{1}{4}\right)^{\sum_{u=0}^{L-1}\delta_u} N\cdot\mathcal{P}(\mathbf{x}_0), \qquad (28a)$$

$$\begin{aligned}\mathcal{P}(\mathbf{n}_y) &= \left(\prod_{m=1}^{M}\eta_m\right)\mathcal{P}(\mathbf{n}_0) + \sum_{i=2}^{M}\left(\prod_{m=i}^{M}\eta_m\right)\Delta\xi_{i-1} + \Delta\xi_M \\ &= \left(\frac{1}{4}\right)^{\sum_{u=0}^{L-1}\delta_u} N^2\sigma_{n_0}^2 + \sum_{l=0}^{L-1}\left(\frac{1}{4}\right)^{\sum_{u=0}^{L-1}\delta_u-\sum_{t=0}^{l}\delta_t}\cdot \\ &\quad 2^{L-l-1}\left[2N\delta_l\sigma_{s,l}^2 + \lambda(l)\sigma_{c,l}^2\right]. \qquad (28b)\end{aligned}$$

Therefore, the signal to quantization noise ratio (SQNR) of FFT results is

$$\begin{aligned}\vartheta &= \frac{\mathcal{P}(\mathbf{y})}{\mathcal{P}(\mathbf{n}_y)} \\ &= \frac{\mathcal{P}(\mathbf{x}_0)}{N\sigma_{n_0}^2 + \sum_{l=0}^{L-1}2^{2\sum_{u=0}^{l}\delta_u-l-1}\left(2N\delta_l\sigma_{s,l}^2+\lambda(l)\sigma_{c,l}^2\right)}. \qquad (29)\end{aligned}$$

To make (29) more practical, the complex quantization errors involved are considered to have i.i.d. real and imaginary components. Specifically, if the quantized data is represented by 1 sign bit along with $w-1$ data bits, then the real part of truncation noise is assumed to be normally distributed in $[-2^{-w}, 2^{-w}]$, so does the imaginary part. By contrast, the real and imaginary part of scaling errors are modeled as discrete random variables, which take values from the set $\{0, 2^{-w}, -2^{-w}\}$ with corresponding probabilities $1/2$, $1/4$ and $1/4$. Considering the foregoing assumptions, $\sigma_{n_0}^2$ and $\sigma_{s,l}^2$ in (29) can be further expressed as

$$\sigma_{n_0}^2 = \frac{2}{3}\cdot 2^{-2w_{\mathrm{ini}}}, \sigma_{s,l}^2 = 2^{-2w_l}.$$

The value of $\sigma_{c,l}^2$ is closely related to the hardware solution of twiddle-factor multiplication. When integer multipliers are selected to fulfill the task, $\sigma_{c,l}^2 = \frac{2}{3}\cdot 2^{-2w_l}$ for the w_l-bit-represented butterfly output. In contrast, when the memoryless CORDIC cores are used, the newly-introduced quantization noise has been analyzed in Appendix C. From (53) in Appendix C, $\sigma_{c,l}^2$ ought to be revised as follows in this case:

$$\begin{aligned}\sigma_{c,l}^2 &= \frac{2}{3}\cdot 2^{-2w_l}\cdot S^2\sum_{t=1}^{T_c-1}(1+2^{-t})\prod_{j=t+1}^{T_c}(1+2^{-2j}) \\ &\quad + \frac{2}{3}\cdot 2^{-2w_l}\cdot\left(3+2^{-3}+2^{-6}+2^{-T_c}\right) \\ &\quad + 2^{l+1-2T_c-2\sum_{u=0}^{l}\delta_u}\cdot\frac{\mathcal{P}(\mathbf{x}_0)}{N}, \qquad (30)\end{aligned}$$

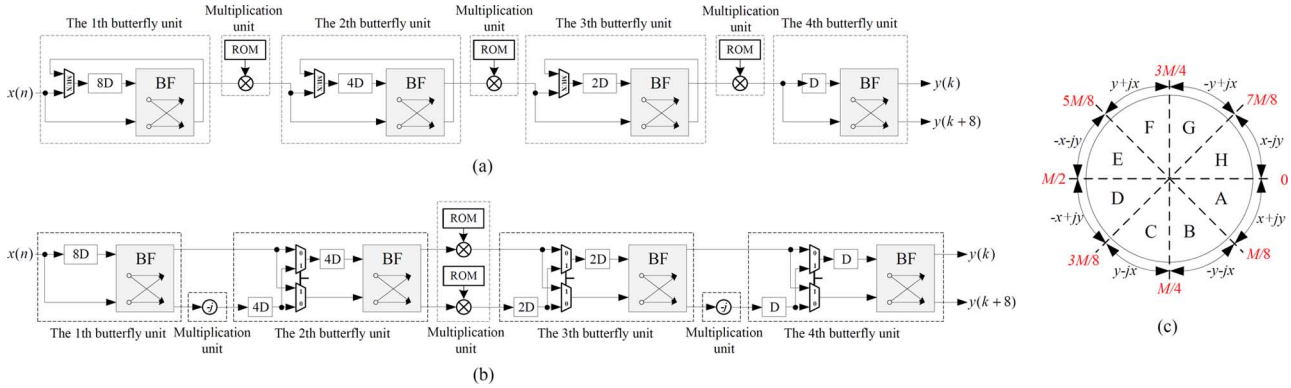


Fig. 4. Hardware structures of SDF and MDC pipelines with an example of 16-point FFT computation. (a) The SDF pipeline using radix-2 algorithm; (b) The MDC pipeline using radix-2^2 algorithm; (c) The coefficient distribution in a complex plane.

where $S = \prod_{t=1}^{T_c} \cos(\tan^{-1}(2^{-t})) \approx 1 - 2^{-3} - 2^{-6}$, T_c represents the number of non-trivial sub-rotators in the CORDIC module.

Denoting the *algorithm vector* as $\mathbf{k} = (k_1 k_2 \cdots k_M)$, where the mth element corresponds to the mth computation stage utilizing radix-2^{k_m} algorithm. For the given $\mathbf{k}$, operating word-length vector $\mathbf{w} = (w_0 w_1 \cdots w_{L-1})$, FFT length $N = 2^L$ and input word-length w_{ini}, the SQNR ϑ for the FFT module using integer multipliers is calculated as follows:

Algorithm 1: The Computation of SQNR for the Extended Radix-2^k FFT Module Using Integer Multipliers

1: For given $\mathbf{k}$, $\mathbf{w}$ and w_{ini}, compute $\mathbf{c} = (c_1\, c_2 \cdots c_M)$ and $\boldsymbol{\delta} = (\delta_0 \delta_1 \cdots \delta_{L-1})$ using (4) and (16), respectively; calculate $\sigma_{n_0}^2$, $\sigma_{s,l}^2$ and $\sigma_{c,l}^2$ according to $\mathbf{w}$ and w_{ini}.
2: Determine $\boldsymbol{\lambda} = (\lambda(0)\lambda(1) \cdots \lambda(L-1))$ through (24), where $\lambda(l)$ is the abbreviation of $\lambda(c_{u_l}, k_{u_l}, l)$, the subscript $u_l = \arg\min_{u_l \in \{1,\ldots,M\}} \{c_{u_l} > l\}$.
3: Substitute $\boldsymbol{\delta}$, $\boldsymbol{\lambda}$, $\sigma_{n_0}^2$, $\sigma_{s,l}^2$ and $\sigma_{c,l}^2$ into (29) to calculate the SQNR ϑ.

If $\sigma_{c,l}^2$ is computed using (30) in the first step, Algorithm 1 will become available to determine the SQNR of the FFT processor adopting CORDIC modules.

B. Hardware Evaluation of the Pipelined Radix-2^k Processor

In this part, the MDC and the SDF structures serve as the representative to expand on the discussion. These two approaches have been extensively used in the pipelined implementation of FFT. Fig. 4 illustrates the corresponding hardware structures with an example of the 16-point FFT. Attention is firstly allotted to the memory consumption. For a $N = 2^L$-point radix-2^k FFT, generally, the relevant pipelined structure consists of L butterfly units to perform the operations of $\mathbf{B}_{\text{F}}(0), \ldots, \mathbf{B}_{\text{F}}(L-1)$, respectively. In SDF pipeline, the lth $(l = 1, 2, \ldots, L)$ butterfly unit is equipped with a $N/2^l$-unit memory bank. Owing to the feedback interconnection, the memories are shared by input samples and intermediate results. By contrast, the $2, 3, \ldots, L$th butterfly unit in MDC pipeline utilize commutators to rearrange the samples, hence these modules consume twice memories as many as their SDF counterparts. Let R_b be the memory resources used for rearranging data streams in butterfly units, then

$$R_b = \begin{cases} \sum_{l=0}^{L-1} 2^{L-l} w_l, & \text{for SDF scheme} \\ N w_0 + \sum_{l=1}^{L-1} 2^{L-l+1} w_l, & \text{for MDC scheme} \end{cases}. \quad (31)$$

It is worth noting that R_b depends exclusively on the operating word-length vector $\mathbf{w}$. For the FFT processor using memoryless CORDIC units, R_b is exactly the overall memory cost. While for the FFT processor equipped with integer multipliers, additional memories are required to store twiddle factors. This part of storage burden can be relieved with the assist of the circular symmetry in the complex plane [27]. As shown in Fig. 4(c), only twiddle factors belonging to the angle range $[0, \pi/4]$ are required to be stored while the rest can be easily generated by conjunction or swapping the real and imaginary part of a complex coefficient. Thus for the twiddle factor matrix $\mathbf{W}_S^{(u)}$ in (7), the coefficients $e^{-j2\pi k/S}$, $k = 0, 1, \ldots S/8$ account for the storage expenditure. Assume that the real and imaginary part of twiddle factors are both in w_T-bit representation, then the memory cost of $\mathbf{W}_S^{(u)}$ is

$$R_0 = \begin{cases} 0, & if\ S < 8\ or\ S = u\ or\ u = 1 \\ (S/8 + 1) \cdot 2w_{\text{T}} \approx S/4 \cdot w_{\text{T}}, & else \end{cases} \quad (32)$$

which is proportional to S, the resolution of twiddle factors. $R_0 = 0$ suggests $\mathbf{W}_S^{(u)}$ is composed only of trivial twiddle factors, i.e., ± 1 and $\pm j$. In this case the multiplications can be fulfilled without multipliers, thus the relevant memory cost can be avoided.

The discussion can be extended to $\mathbf{M}_1(u)$, $\mathbf{M}_2(u, v)$ and $\mathbf{M}_3(u, v)$ in (8a)–(8c). Note that the twiddle factors included in the multiplication matrix come from the corresponding twiddle factor matrix, the memory expenses for $\mathbf{M}_1(u)$, $\mathbf{M}_2(u, v)$ and $\mathbf{M}_3(u, v)$ can be obtained through (32). That is

$$R_1(u) \equiv 0, \quad (33)$$

$$R_2(u, v) = \begin{cases} 2^{v-2} w_{\text{T}}, & \text{if } v \geqslant 3 \\ 0, & \text{else} \end{cases} \quad (34)$$

$$R_3(u, v) = \begin{cases} \frac{N \cdot w_{\text{T}}}{2^{u+2}}, & \text{if } N \geqslant 2^{u+3} \text{ and } N > 2^{u+v} \\ 0, & \text{else} \end{cases}. \quad (35)$$

By replacing the parameter u, v in (33)–(35) with the concrete values given in (13), the occupied memories corresponding to

TABLE I
MULTIPLIERS REQUIREMENT OF $\mathbf{H}(c_m, k_m)$ IN THE SDF PIPELINE OR THE MDC PIPELINE

k_m \ c_m	L	$L-1$	$L-2$	$L-3$	$< L-3$
1	$\mathbf{f}_m = [0\ 0\ 0]$, $\tilde{\mathbf{f}}_m = [0\ 0\ 0]$	$\mathbf{f}_m = [0\ 0\ 0]$, $\tilde{\mathbf{f}}_m = [0\ 0\ 0]$	$\mathbf{f}_m = [1\ 0\ 0]$, $\tilde{\mathbf{f}}_m = [1\ 0\ 0]$	$\mathbf{f}_m = [0\ 1\ 0]$, $\tilde{\mathbf{f}}_m = [0\ 1\ 0]$	$\mathbf{f}_m = [0\ 0\ 1]$, $\tilde{\mathbf{f}}_m = [0\ 0\ 1]$
2	$\mathbf{f}_m = [0\ 0\ 0]$, $\tilde{\mathbf{f}}_m = [0\ 0\ 0]$	$\mathbf{f}_m = [1\ 0\ 0]$, $\tilde{\mathbf{f}}_m = [2\ 0\ 0]$	$\mathbf{f}_m = [0\ 1\ 0]$, $\tilde{\mathbf{f}}_m = [0\ 2\ 0]$	$\mathbf{f}_m = [0\ 0\ 1]$, $\tilde{\mathbf{f}}_m = [0\ 0\ 2]$	
3	$\mathbf{f}_m = [1\ 0\ 0]$, $\tilde{\mathbf{f}}_m = [1\ 0\ 0]$	$\mathbf{f}_m = [1\ 1\ 0]$, $\tilde{\mathbf{f}}_m = [1\ 2\ 0]$	$\mathbf{f}_m = [1\ 0\ 1]$, $\tilde{\mathbf{f}}_m = [1\ 0\ 2]$		
4	$\mathbf{f}_m = [0\ 1\ 0]$, $\tilde{\mathbf{f}}_m = [0\ 2\ 0]$	$\mathbf{f}_m = [0\ 1\ 1]$, $\tilde{\mathbf{f}}_m = [0\ 2\ 2]$			
> 4 and odd	$\mathbf{f}_m = [1\ 0\ \lfloor (k_m-3)/2 \rfloor]$, $\tilde{\mathbf{f}}_m = [1\ 0\ 2\ \lfloor (k_m-3)/2 \rfloor]$	$\mathbf{f}_m = [1\ 0\ \lfloor (k_m-3)/2 \rfloor + 1]$, $\tilde{\mathbf{f}}_m = [1\ 0\ 2\ \lfloor (k_m-3)/2 \rfloor + 2]$			
> 4 and even	$\mathbf{f}_m = [0\ 1\ \lfloor (k_m-3)/2 \rfloor]$, $\tilde{\mathbf{f}}_m = [0\ 2\ 2\ \lfloor (k_m-3)/2 \rfloor]$	$\mathbf{f}_m = [0\ 1\ \lfloor (k_m-3)/2 \rfloor + 1]$, $\tilde{\mathbf{f}}_m = [0\ 2\ 2\ \lfloor (k_m-3)/2 \rfloor + 2]$			

For $\mathbf{M}(c_m, k_m, l)$, $l \in \{c_m - k_m, \ldots, c_m - 1\}$ can be derived as follows

$$R(c_m, k_m, l) = R(c_m, k_m, c_m - i) = \begin{cases} 2^{i-1} \cdot w_{\mathrm{T}}, & \text{if } i \in \Psi_2(k_m) \text{ and } k_m \geqslant 3 \\ \frac{N \cdot w_{\mathrm{T}}}{2^{c_m - k_m + 2}}, & \text{if } i = 1, c_m - k_m \leqslant L-3 \text{ and } m < M \\ 0, & \text{else} \end{cases} \tag{36}$$

where $\Psi_2(k_m)$ is defined in (14b). Similar to $\lambda(c_m, k_m, l)$ in (24), $R(c_m, k_m, l)$ **is abbreviated to** $R(l)$ **in the following derivation**.

Equation (36) enables us to provide an accurate analysis of the memory consumption for storing twiddle factors in the pipelined FFT processor. The discussion begins with the SDF structure. It should be noted that each $\mathbf{M}(c_m, k_m, l)$ is implemented by a single complex multiplier assisted by a memory bank to store the relevant coefficients. Hence, $R_m\big|_{\mathrm{SDF}}$, the memory expense of twiddle factors storage in SDF pipeline is given by

$$R_m\big|_{\mathrm{SDF}} = \sum_{l=0}^{L-1} R(l) = \sum_{m \in \Gamma_1} \sum_{i \in \Psi_2(k_m)} 2^{i-1} w_{\mathrm{T}} + \sum_{m \in \Gamma_2} \frac{N \cdot w_{\mathrm{T}}}{2^{c_m - k_m + 2}}, \tag{37}$$

where Γ_1 and Γ_2 are defined as $\Gamma_1 = \{m \in \mathbb{N} | k_m \geqslant 3\}$, $\Gamma_2 = \{m \in \mathbb{N} | c_m - k_m \leqslant L-3 \text{ and } m < M\}$.

In terms of MDC structures, commutators provide two independent connections of adjacent stages, making the discussion differ from the SDF case. As butterfly units export the additions and subtractions through two links simultaneously, $\mathbf{M}(c_m, k_m, l)$ would be mapped to two multipliers to weight the two data streams individually. In general, this change may double the memory cost associated with twiddle factors storage, since each multiplier ought to be equipped with an exclusive ROM unit. Nevertheless, some particular configurations provided by Proposition 2 may invalidate this statement:

Proposition 2: For $\mathbf{M}_2(u, v)$ and $\mathbf{M}_3(u, v)$ including non-trivial coefficients, $\mathbf{M}_2(u, v)$ weights all the additions of butterfly outputs by trivial coefficients if and only if $v = 3$; $\mathbf{M}_3(u, v)$ shows the same property provided $v = 1$.

Proof: See Appendix D. ■

Proposition 2 indicates that in the MDC pipeline, $\mathbf{M}_2(u, v)$ would consume the same hardware resources as it is in the SDF architecture if $v = 3$, so does $\mathbf{M}_3(u, v)$ in the case of $v = 1$. Taking the concrete parameter definitions given in (13) into account, $v = 3$ in $\mathbf{M}_2(u, v)$ leads to $i = 2$ with $i \in \Psi_2(k_m)$. Consequently, k_m ought be an odd number no less than 3. In summary, the memory requirement to store twiddle factors in MDC pipelines can be concluded as

$$R_m\big|_{\mathrm{MDC}} = \sum_{m \in \Gamma_1} \sum_{i \in \Psi_2(k_m)} 2^i w_{\mathrm{T}} - \sum_{m \in \Gamma_1} \underline{k_m} \cdot 2 w_{\mathrm{T}} + \sum_{m \in \Gamma_2} \frac{N \cdot w_{\mathrm{T}}}{2^{c_m - k_m + 1}} - \sum_{m \in \Gamma_2 \cap \Gamma_3} \frac{N \cdot w_{\mathrm{T}}}{2^{c_m - k_m + 2}} \tag{38}$$

with $\mathbf{\Gamma}_3 = \{m \in \mathbb{N} | k_m = 1\}$, $\underline{k_m} = \mathrm{mod}(k_m, 2)$.

Next, we investigate the computing resources consumption, namely, the number of multipliers (CORDIC cores) and adders in radix-2^k pipelined processor. For $\mathbf{H}(c_m, k_m)$ defined in (9), the relevant requirement of complex multipliers is summarized in Table I, where $\mathbf{f}_m$ and $\tilde{\mathbf{f}}_m$ correspond to the cost in SDF pipeline and MDC pipeline, respectively. The three elements of $\mathbf{f}_m$ or $\tilde{\mathbf{f}}_m$ separately represent the number of $\mathbf{W}_8^{(u)}$ multipliers, $\mathbf{W}_{16}^{(u)}$ multipliers and other general multipliers. The $\mathbf{W}_8^{(u)}$ multipliers and $\mathbf{W}_{16}^{(u)}$ multipliers are considered individually here as they have diverse hardware complexities compared to the general ones. From Table I the overall multipliers cost can be obtained using

$$\mathbf{f}_{\mathrm{mul}} = \begin{cases} \sum_{m=1}^{M} \mathbf{f}_m, & \text{for SDF scheme} \\ \sum_{m=1}^{M} \tilde{\mathbf{f}}_m, & \text{for MDC scheme} \end{cases}. \tag{39}$$

The foregoing complex multipliers can be replaced uniformly with CORDIC modules. Thus the relevant consumption of CORDIC cores is

$$n_{\mathrm{cordic}} = \|\mathbf{f}_{\mathrm{mul}}\|_1 \tag{40}$$

where $\|\cdot\|_1$ is the ℓ_1-norm of a vector.

The realization of butterfly units accounts for the occupation of adders in FFT processors. Regardless of the configuration of $\mathbf{k}$, an N-point FFT processor includes $L = \log_2 N$ radix-2 butterfly stages for both SDF structure and MDC structure. Therefore, the number of complex adders is given by

$$n_{\text{adder}} = 2\log_2 N = 2L = 2c_M. \tag{41}$$

We summarize the foregoing discussion on hardware consumption of the extended radix-2^k FFT module as follows:

Algorithm 2: Hardware Cost Of The Extended Radix-2^k FFT Module Using Integer Multipliers

1: For given $\mathbf{k}$, $\mathbf{w}$, compute $\mathbf{c} = (c_1 c_2 \cdots c_M)$ using (4); then determine $\Gamma_1 = \{m \in \mathbb{N} \big| k_m \geqslant 3\}$, $\Gamma_2 = \{m \in \mathbb{N} \big| c_m - k_m \leqslant L-3 \text{ and } m < M\}$ and $\Gamma_3 = \{m \in \mathbb{N} \big| k_m = 1\}$; for each $m \in \Gamma_1$, find $\Psi_2(k_m)$ from (14b).
2: Determine R_b from (31); for SDF structure, compute R_m using (37); for MDC structure, determine R_m from (38); then the overall memory cost is $R_b + R_m$.
3: Find the multipliers requirement from (39), where the entries of $\mathbf{f}_{\text{mul}}$ represent the cost of $\mathbf{W}_8^{(u)}$ multipliers, $\mathbf{W}_{16}^{(u)}$ multipliers and other general multipliers, respectively.
4: Find the occupation of complex adders from (41).

When the FFT module is provided with CORDIC units, R_b is exactly the overall memory requirement, while the amount of CORDIC units is described in (40).

Finally, it can be found from (15) that $\mathbf{k}$ is only connected with the arrangement of coefficient-weighting units, namely, the locations of complex multipliers or CORDIC units in the circuit. Note the computing delay of pipelined FFT processor comes primarily from butterfly operations, the change of $\mathbf{k}$ will not affect this metric markedly. When eliminating the latency of coefficient weighting, N clock cycles are essential for both SDF and MDC pipelined processor to fulfill the N-point FFT and generate the bit-reversed output under arbitrary $\mathbf{k}$.

IV. Applications and Simulations

A. Optimization of the FFT Processor

In practical FFT modules, the computing accuracy is always in conflict with the hardware expense. The tension can be relieved by jointly optimizing the algorithm vector $\mathbf{k}$ and operating word-length vector $\mathbf{w}$. In this part we select the memory cost as the representative of hardware expense to unfold the kernel. Based on the actual demands, the optimization problems can be divided into two categories:

1) Memory Priority: Minimum the memory consumption while SQNR is no less than the given threshold T_{SQNR}:

$$\begin{aligned} &\min_{\mathbf{w},\mathbf{k}} R \\ &s.t.\ \vartheta \geqslant T_{\text{SQNR}}; \\ &\quad w_{\text{ini}} \leqslant w_0 \leqslant \cdots \leqslant w_{L-1} = w_{\text{fin}}, \mathbf{w} \in \mathbb{N}^L; \\ &\quad \|\mathbf{k}\|_1 = L, \mathbf{k} \in \mathbb{Z}_+^M \text{ and } M \in \{1, \ldots L\}. \end{aligned} \tag{42}$$

Algorithm 3: SA Based Heuristic Searching Strategy to Find the Optimum Solution of $\mathbf{k}$ and $\mathbf{w}$

Initialization: Discretionarily select $\mathbf{w}$ and $\mathbf{k}$ satisfying the second and the third constraint; Let $t = T_s$, $\mathbf{w}_{\text{opt}} = \mathbf{w}$, $\mathbf{k}_{\text{opt}} = \mathbf{k}$, $\psi_{\text{c}} = \psi_{\text{opt}} = +\infty$;
1: **for** $n_{\text{out}} = 0$; $n_{\text{out}} < N_{\text{out}}$; $n_{\text{out}} = n_{\text{out}} + 1$ **do**
2: **for** $n_{\text{in}} = 0$; $n_{\text{in}} < N_{\text{in}}$; $n_{\text{in}} = n_{\text{in}} + 1$ **do**
3: $\mathbf{w}_t = f_{nb}(\mathbf{w})$, $\mathbf{k}_t = g_{nb}(\mathbf{k})$;
4: **if** $\mathbf{w}_t$, $\mathbf{k}_t$ satisfy the first constriction **then**
5: Calculate $f_{\text{obj}}(\mathbf{w}_t, \mathbf{k}_t)$ and generate a random variable $\tau \sim U[0,1)$;
6: **if** $\tau \leqslant \min\left(1, e^{[\psi_{\text{c}} - f_{\text{obj}}(\mathbf{w}_t, \mathbf{k}_t)]/t}\right)$ **then**
7: $\mathbf{w} = \mathbf{w}_t$, $\mathbf{k} = \mathbf{k}_t$, $\psi_{\text{c}} = f_{\text{obj}}(\mathbf{w}_t, \mathbf{k}_t)$;
8: **end if**
9: **if** $f_{\text{obj}}(\mathbf{w}_t, \mathbf{k}_t) \leqslant \psi_{\text{opt}}$ **then**
10: $\mathbf{w}_{\text{opt}} = \mathbf{w}_t$, $\mathbf{k}_{\text{opt}} = \mathbf{k}_t$, $\psi_{\text{opt}} = f_{\text{obj}}(\mathbf{w}_t, \mathbf{k}_t)$;
11: **end if**
12: **end if**
13: **end for**
14: $t = \beta \cdot t$;
15: **if** $n_{\text{out}} = N_{\text{out}} - 1$ **or** $\mathbf{w}_{\text{opt}}, \mathbf{k}_{\text{opt}}$ remain unchanged in two successive temperatures **then**
16: Return $\mathbf{w}_{\text{opt}}$ and $\mathbf{k}_{\text{opt}}$;
17: **end if**
18: **end for**

2) Precision Priority: Maximum SQNR while the memory cost does not exceeds the constraint R_c:

$$\begin{aligned} &\min_{\mathbf{w},\mathbf{k}} -\vartheta \\ &s.t.\ R \leqslant R_c; \\ &\quad w_{\text{ini}} \leqslant w_0 \leqslant \cdots \leqslant w_{L-1} = w_{\text{fin}}, \mathbf{w} \in \mathbb{N}^L; \\ &\quad \|\mathbf{k}\|_1 = L, \mathbf{k} \in \mathbb{Z}_+^M \text{ and } M \in \{1, \ldots L\}. \end{aligned} \tag{43}$$

Here the maximization of SQNR ϑ is realized through minimizing $-\vartheta$ equivalently.

Equations (42) and (43) are both nonlinear integer programming problems. The second and the third constraint set a feasible region Υ roughly, within which the number of candidates is $n_s = \binom{L + w_{\text{fin}} - w_{\text{ini}} - 1}{L-1} \sum_{l=0}^{L-1} \binom{L-1}{l}$. If w_{ini} is close to w_{fin} and the FFT size is medium, n_s is acceptable and the exhaustive search can be executed in Υ by further considering the first constraint. However, larger $w_{\text{fin}} - w_{\text{ini}}$ and L would render the exhaustive search ineffective. On this occasion, we employ the Simulated Annealing (SA) algorithm [28] to find the optimum solution. In Algorithm 3, we provide a summary of SA-based heuristic searching scheme rather than discussing the details. Before starting the search process, the initial temperature T_s, attenuation factor of temperature β, the number of inner loops N_{in} and outer loops N_{out} should be assigned. Function $f_{\text{obj}}(\cdot,\cdot)$ represents the objective function of the problem, $f_{nb}(\cdot)$ and $g_{nb}(\cdot)$ define a potential one-step transition in the solution space, where the internal procedure is il-

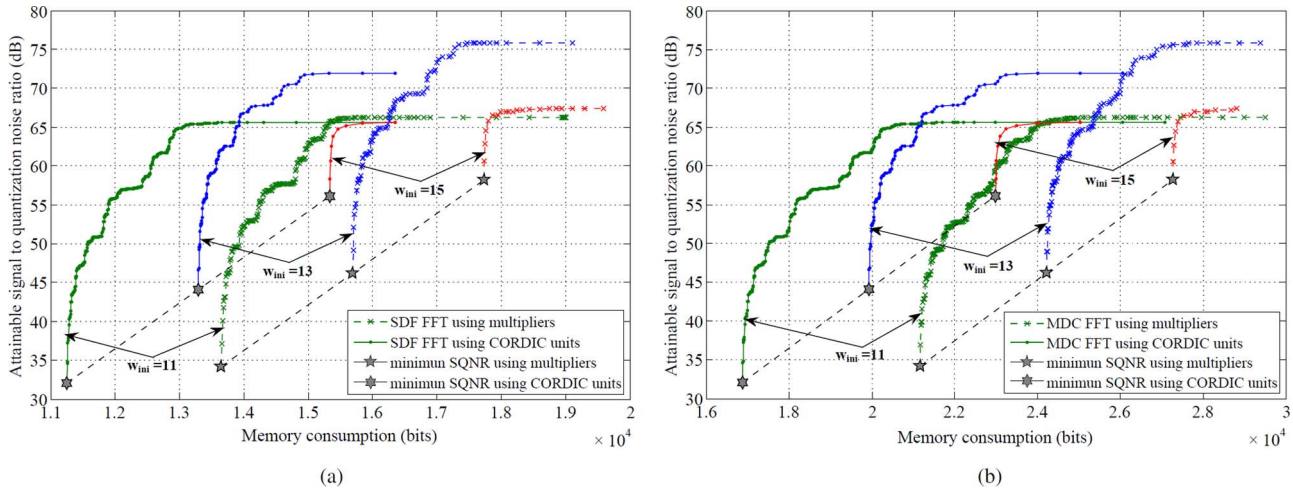


Fig. 5. The attainable SQNR vs. memory consumption. ($N = 512$, $w_{\text{fin}} = 16$, $w_{\text{T}} = 16$ for complex multipliers and $T_c = 13$ for CORDIC units). (a) the SDF structure (b) the MDC structure.

lustrated in Algorithm 4 and Algorithm 5, respectively. $\mathbf{w}_{\text{opt}}$ and $\mathbf{k}_{\text{opt}}$ record best solutions ever found and serve as the final output. Since $f_{nb}(\cdot)$ and $g_{nb}(\cdot)$ restrict the search space to the region Υ, the SA-based search process is capable to converge to the optimum point before traversing the whole space.

Algorithm 4: The Internal Procedure of $\mathrm{w}_t = f_{nb}(\mathbf{w})$ to Generate a Neighboring Solution of $\mathbf{w}$

1: Let $\mathbf{\Delta w} = (\Delta w_0\ \Delta w_1 \cdots \Delta w_{L-1})$ with $\Delta w_0 = w_0 - w_{\text{ini}}$ and $\Delta w_i = w_i - w_{i-1}$, $i = 1, \ldots, L-1$;
2: Randomly select u_1 from $\mathbf{U}_1 = \{i \in \mathbb{N} | \Delta w_i > 0\}$. Then randomly choose u_2 from $\mathbf{U}_2 = \{0, 1, \ldots, L-1\} \backslash \{u_1\}$. Replace Δw_{u_1} and Δw_{u_2} in $\mathbf{\Delta w}$ with $\Delta w_{u_1} - 1$ and $\Delta w_{u_2} + 1$, respectively. Denote this modified $\mathbf{\Delta w}$ as $\mathbf{\Delta \tilde{w}} = (\Delta \tilde{w}_0\ \Delta \tilde{w}_1 \cdots \Delta \tilde{w}_{L-1})$.
3: The entries of output vector $\mathbf{w}_t = (w_{t,0}\ w_{t,1} \cdots w_{t,L-1})$ are determined recursively as: $w_{t,0} = w_{\text{ini}} + \Delta \tilde{w}_0$, $w_{t,i} = w_{t,i-1} + \Delta \tilde{w}_i$ for $i = 1, \ldots, L-1$.

Algorithm 5: The Internal Procedure of $\mathbf{k}_t = g_{nb}(\mathbf{k})$ to Generate a Neighboring Solution of $\mathbf{k}$

1: Let $\rho = \dim\{\mathbf{k}\}$, expand $\mathbf{k}$ to a L-dimension vector as $\mathbf{k}_e = (\mathbf{k}\ \mathbf{0}_{L-\rho}) = (k_{e,0}\ k_{e,1} \cdots k_{e,L-1})$;
2: Randomly select $\rho/2$ elements from the set $\mathbf{E}_1 = \{0, \ldots, \rho-1\}$ and $\mathbf{E}_2 = \{0, \ldots, L-1\}$ to construct $\mathbf{E}_+$ and $\mathbf{E}_S$, respectively;
3: Let $\tilde{\mathbf{k}} = (\tilde{k}_0\ \tilde{k}_1 \cdots \tilde{k}_{L-1})$, whose entries are calculated as $\tilde{k}_j\big|_{j\in\mathbf{E}_+} = k_{e,j}\big|_{j\in\mathbf{E}_+} - 1$, $\tilde{k}_j\big|_{j\in\mathbf{E}_S} = k_{e,j}\big|_{j\in\mathbf{E}_S} + 1$, $\tilde{k}_j\big|_{j\notin\mathbf{E}_S\cup\mathbf{E}_+} = k_{e,j}\big|_{j\notin\mathbf{E}_S\cup\mathbf{E}_+}$;
4: Remove all the zero entries from $\tilde{\mathbf{k}}$ to obtain $\mathbf{k}_t$.

B. Analysis of Simulation Results

In the simulation, the entries of input vector $\mathbf{x}_0$ are selected as the normally distributed complex white noise, whose real and imaginary part are independent and normally distributed in $[-1, 1]$, then $\mathcal{P}(\mathbf{x}_0) = tr\left\{\mathbb{E}(\mathbf{x}_0\mathbf{x}_0^{\text{H}})\right\} = 2N/3$ and the SQNR defined in (29) is concreted into the form

$$\vartheta = \frac{2N/3}{N\sigma_{n_0}^2 + \sum_{l=0}^{L-1} 2^{2\sum_{u=0}^{l}\delta_u - l - 1}\left(2N\delta_l\sigma_{s,l}^2 + \lambda(l)\sigma_{c,l}^2\right)}. \tag{44}$$

Substitute (44) into (43), we can determine the attainable SQNR under given memory constraint. This issue is considered in Fig. 5 with $N = 512$ and $w_{\text{fin}} = 16$. In addition, $w_{\text{T}} = 16$ for complex multipliers and $T_c = 13$ for CORDIC units so that they have the similar angular resolution. By comparing the results in Figs. 5(a) and 5(b), it can be found that the SDF hardware scheme is more memory-efficient than its MDC counterpart to reach the specified SQNR performance.

Another meaningful property shown in Fig. 5 is that the minimum SQNR points, whose corresponding x-coordinates are exactly the minimum memory costs under certain N, w_{ini} and $w_{\text{fin}}(w_{\text{ini}} < w_{\text{fin}})$, are collinear for changing w_{ini}. The explanation for this phenomenon is given as follows. For FFT module with integer multipliers, the overall memory consumption is composed of R_b and R_m, where R_b depends exclusively on $\mathbf{w}$ while R_m is connected only to $\mathbf{k}$. Therefore,

1) To minimize R_m, these investigated points ought to share the identical $\mathbf{k}$;
2) To minimize R_b, $\mathbf{w}$ should be in the form $\mathbf{w} = \{w_{\text{ini}} \cdots w_{\text{ini}}\ w_{\text{fin}}\}$. In this case $\boldsymbol{\delta} = (\mathbf{1}_{L-1}0)$ from (16).

Thus when w_{ini} experiences a 1-bit increase, the minimum memory cost will correspondingly rise by ΔR. This increment comes from the change of R_b. From (31) we have

$$\Delta R = \begin{cases} \sum_{l=0}^{L-2} 2^{L-l} = 2N - 4, & \text{for SDF scheme} \\ \sum_{l=1}^{L-2} 2^{L-l+1} + N = 3N - 8, & \text{for MDC scheme.} \end{cases}$$

Since ΔR is irrelevant to R_m, the expression is also suitable for the FFT module utilizing CORDIC units. In addition, when $\mathbf{w} = \{w_{\text{ini}} \cdots w_{\text{ini}}\ w_{\text{fin}}\}$, a 1-bit increase of w_{ini} would shrink $\sigma_{n_0}^2$ and $\sigma_{s,l}^2$, $\sigma_{c,l}^2$, $l = 0, 1, \ldots, L-2$ to a quarter. Thus from (44) the gain of SQNR $\Delta\vartheta \approx 6$ dB. Based on the foregoing discussion, it can be confirmed that these investigated points are collinear with the slope $\Delta\vartheta/\Delta R$.

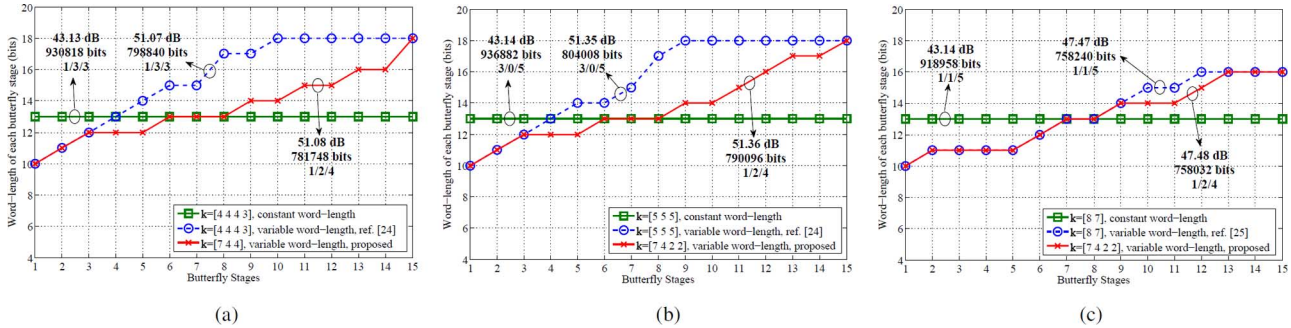


Fig. 6. Word-length configuration of SDF processor using complex multipliers to minimize memory cost under the SQNR constraint, where $x_1/x_2/x_3$ indicates the amount of $\mathbf{W}_8^{(u)}$ multipliers, $\mathbf{W}_{16}^{(u)}$ multipliers and general multipliers are x_1, x_2 and x_3, respectively ($N = 32768$). (a) $w_{\rm ini} = 10$, $w_{\rm fin} = 18$, $w_{\rm T} = 9$ (b) $w_{\rm ini} = 10$, $w_{\rm fin} = 18$, $w_{\rm T} = 10$ (c) $w_{\rm ini} = 10$, $w_{\rm fin} = 16$, $w_{\rm T} = 8$.

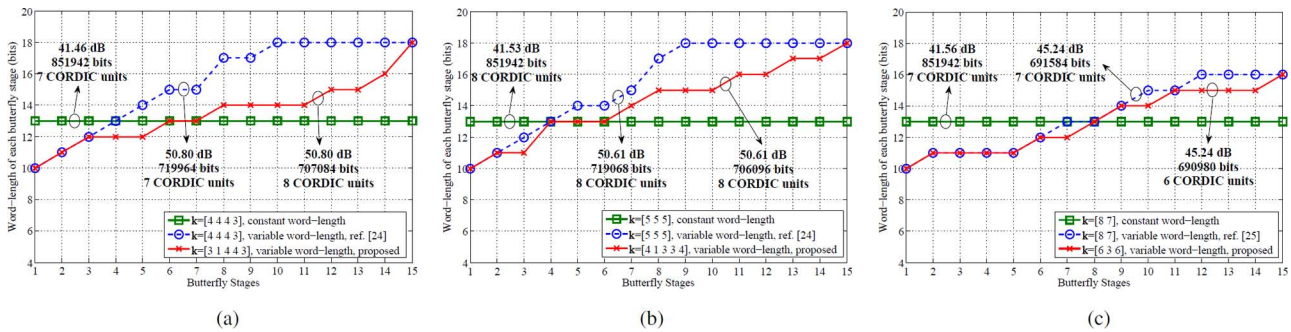


Fig. 7. Word-length configuration of SDF processor using CORDIC units to minimize memory cost under the SQNR constraint ($N = 32768$). (a) $w_{\rm ini} = 10$, $w_{\rm fin} = 18$, $T_c = 13$ (b) $w_{\rm ini} = 10$, $w_{\rm fin} = 18$, $T_c = 13$ (c) $w_{\rm ini} = 10$, $w_{\rm fin} = 16$, $T_c = 13$.

On the other hand, the saturated SQNR of each curve, which formulates the best accuracy performance under certain N, $w_{\rm ini}$ and $w_{\rm fin}$, is also noteworthy. First, for concrete N and $w_{\rm fin}$, the increase of $w_{\rm ini}$ is not invariably beneficial to boost the saturated SQNR. This is because $w_{\rm ini}$ correlates to both noise variances $\sigma_{n_0}^2$, $\sigma_{s,l}^2$, $\sigma_{c,l}^2$ and $\boldsymbol{\delta}$. The co-effect of these variables results in the phenomenon. Second, for the fixed N and $w_{\rm ini}$, $w_{\rm fin}$, the FFT module using complex multipliers possesses higher saturated SQNR than its counterpart adopting CORDIC units, since the CORDIC unit introduces more quantization noise than complex multipliers in the rotations.

Compared to the constant word-length realization, a joint optimization of $\mathbf{k}$ and $\mathbf{w}$ using (42) is able to bring considerable area-reduction to the circuit without loss of accuracy. Fig. 6 supports this conclusion with a 32768-point SDF FFT using complex multipliers. Similarly, this issue is rethought in Fig. 7 for the FFT module adopting CORDIC units. Apart from the optimized solutions we proposed, the feasible schemes offered by [24] and [25] serve as references here, where $\mathbf{k}$ and $\mathbf{w}$ are jointly optimized under the constraint $\mathbf{k} = [k \cdot \mathbf{1}_n \; l]$ if the FFT size $N = 2^{kn+l}$. We relax the constraint to $\|\mathbf{k}\|_1 = \log_2 N$, making the selection of $\mathbf{k}$ more flexible. This extension, as revealed from Figs. 6 and 7, enables the FFT processor to gain further memory-decline.

C. Experimental Verifications

We implement a 512-point SDF FFT processor in the field programmable gate array (FPGA) to validate whether (29) could provide a reliable estimation of the actual SQNR. Radix-2^2 and radix-2^3 algorithms serve as the representative in the test and the SQNR performance is regulated through changing the operating word-length vector $\mathbf{w}$. When conducting the experiment, the samples participate in the fixed-point computation in the FPGA and the floating-point computation in the computer simultaneously. By this means the quantization noise can be determined by comparing the two sets of results.

According to Fig. 8, when the FFT processor is equipped with complex multipliers, (29) may always overestimate the SQNR values. This is because the roundoff errors belonging to different stages are not completely independent as we have supposed. In contrast, for the FFT processor using CORDIC units, (29) will underestimates the SQNR sometimes. The reason lies in the fact that (52) in Appendix C is essentially an upper bound of angle approximation error variance. As a result, $\sigma_{c,l}^2$ obtained from (30) will exceeds the actual value in certain applications. Note the bias of SQNR estimation is less than 1.5 dB throughout the experiment, we think (29) is available for practical applications.

Finally, we investigate the optimization design of radix-2^k FFT processor under the framework of OFDM transmission. The utilized OFDM system is described in Fig. 8, whose configuration is listed in Table II. At the transmitter, the FFT module executes IFFT algorithm to generate the OFDM signal. Correspondingly, the receiver utilizes the FFT core to fulfill the demodulation. In the experiment, the FFT module at transmitter is implemented by IP soft cores, while the other one belonging to the receiver acts as the optimization objective. We implement the tested radix-2^k schemes using multiplier-based SDF structure, where $N = 512$, $w_{\rm ini} = 12$, $w_{\rm fin} = 16$ and $w_{\rm T} = 16$.

For the specified memory constraint $R_c = 16$ Kb, we can jointly determine $\mathbf{k}$ and $\mathbf{w}$ through (43) to maximize SQNR.

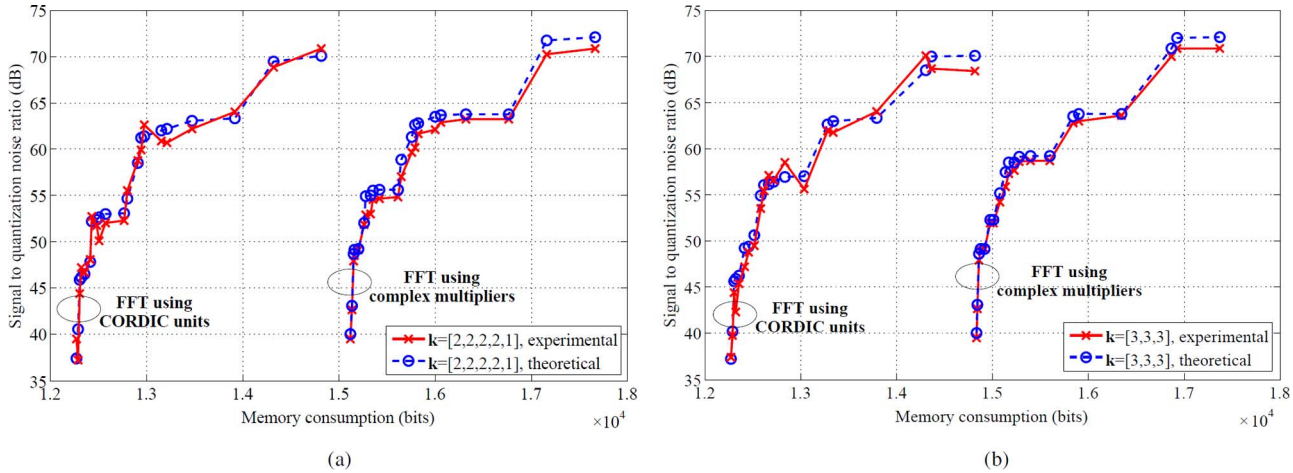


Fig. 8. The comparison of theoretical SQNR values and experimental results ($N = 512$, $w_{\text{ini}} = 12$, $w_{\text{fin}} = 16$, $w_{\text{T}} = 16$ for complex multipliers and $T_c = 13$ for CORDIC units). (a) radix-2^2 FFT (b) radix-2^3 FFT.

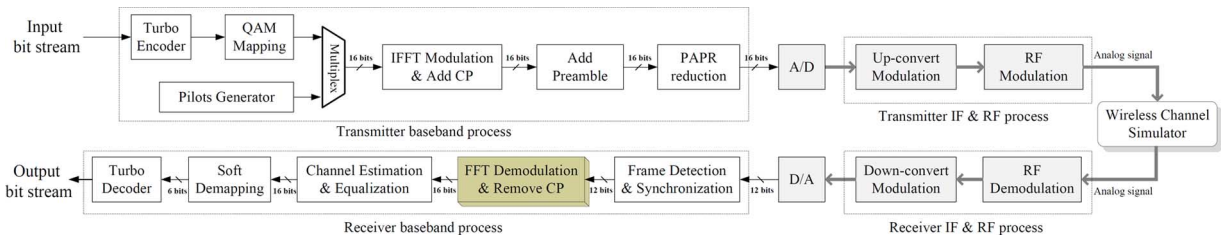


Fig. 9. Block diagram of the adopted OFDM communication system.

TABLE II
CONFIGURATION OF THE OFDM SYSTEM

Code Rate	4/5
Modulation Type	QPSK, 16QAM, 64QAM
Number of Subcarriers	512
Ratio of Cyclic Prefix	1/4, 128 subcarriers
Pilot Structure	block type with 1/5-density
Channel Model	ITU-VA@100km/h

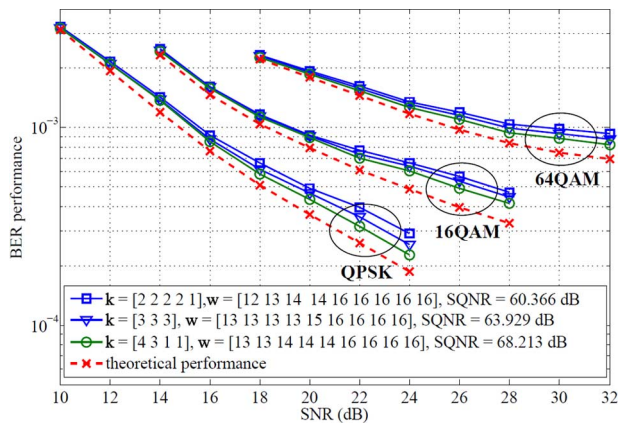


Fig. 10. BER comparison of joint optimization of $\mathbf{k}$, $\mathbf{w}$ and individual optimization of $\mathbf{w}$ under given $\mathbf{k}$ (memory constraint $R_c = 16$ Kb).

On the other hand, $\mathbf{w}$ will be optimized individually when the radix-2^k algorithm has been selected by designers. Since radix-2^2 algorithm and radix-2^3 algorithm are popular solutions to deal with the medium-size FFT, the two schemes will serve as references in subsequent discussion, where $\mathbf{k} = [2\ 2\ 2\ 2\ 1]$ for the radix-2^2 512-point FFT and $\mathbf{k} = [3\ 3\ 3]$ for the radix-2^3 case. From the theoretical SQNR values presented in Fig. 10, the joint optimization of $\mathbf{k}$, $\mathbf{w}$ is able to gain better computing accuracy than the individual optimization of $\mathbf{w}$ for given radix-2^k scheme. This SQNR superiority can be converted into the enhancement of bit-error-rate (BER) performance at high SNR region. While at low SNR region, since the quantization errors introduced by the fixed-point FFT becomes secondary compared to the channel noise, the communication quality depends primarily on the channel.

For the specified BER performance, it is usually desirable to meet the requirement with minimum hardware expense. This scenario is discussed in Table III, where the minimization of occupied memories is emphasized. As shown, by jointly optimizing $\mathbf{k}$ and $\mathbf{w}$, the proposed scheme ($\mathbf{k} = [4\ 2\ 1\ 1\ 1]$, $\mathbf{w} = [13\ 13\ 14\ 15\ 15\ 16\ 16\ 16\ 16]$) is more memory-efficient than the individual optimization of $\mathbf{w}$ to satisfy the given BER constraint. Moreover, we also measure the computing delay and maximum throughput of the schemes listed in Table III. It is revealed from the results that the proposed scheme has similar performance as the references in these two aspects, which suggests that the joint optimization of $\mathbf{k}$ and $\mathbf{w}$ will not pose a threat to the computing delay or throughput.

TABLE III
DIFFERENT CONFIGURATIONS OF RADIX-2^k SDF FFT PROCESSOR TO MEET THE SPECIFIED BER REQUIREMENT

Algorithm Vector	**Word-length Vector**	**Memory** (*bits*)	**Computing Delay**† (*clock cycles*)	**Throughput**‡ (*Msamples/s*)	**Theoretical SQNR** (*dB*)	**BER Performance @SNR=30 dB**
$\mathbf{k} = [4\ 2\ 1\ 1\ 1]$	$\mathbf{w} = [13\ 13\ 14\ 15\ 15\ 16\ 16\ 16\ 16]$	16096	531	326	69.360	0.939×10^{-4}
$\mathbf{k} = [3\ 3\ 3]$	$\mathbf{w} = [14\ 14\ 14\ 14\ 14\ 15\ 15\ 15\ 16]$	16900	527	334	69.179	0.941×10^{-4}
$\mathbf{k} = [2\ 2\ 2\ 2\ 1]$	$\mathbf{w} = [14\ 14\ 14\ 14\ 14\ 16\ 16\ 16\ 16]$	17216	531	319	69.813	0.935×10^{-4}

† The computing delay is measured using the ModelSim simulation tool, thus the results are in clock cycles.
‡ The maximum throughput is determined from the implementation report generated by ISE 12.4, where the target FPGA is Xilinx XC6VLX240T. In the OFDM system, these FFT modules operate with the 16 Msamples/s constant throughput.

V. CONCLUSION

In this paper, we presented a fixed-point analysis of the extended radix-2^k FFT with variable operating word-length. The derived SQNR was able to provide a reliable estimation of the actual value. Moreover, we mapped the top-level computing scheme to the pipelined SDF and MDC structure and evaluated the relevant hardware cost. Based on these efforts, algorithm selection and operating word-length configuration were jointly optimized to strike a reasonable balance between computational accuracy and hardware burden, where memory expense acted as the representative in the discussion. Experiments verified that the proposed joint optimization strategy was more effective than existing approaches for the low-cost and high-accuracy design of radix-2^k FFT.

APPENDIX A
SOME PROPERTIES OF KRONECKER PRODUCT AND STRIDE PERMUTATIONS MATRIX

This appendix summaries the features of Kronecker product and stride permutations matrix, while [29] provides a detailed illustration of these lemmas and corollaries.

Lemma 1: Let $\mathbf{G}_S$ be a stride permutations matrix, then $\mathbf{G}_S$ is invertible and $\mathbf{G}_S^{-1} = \mathbf{G}_S^{\mathrm{T}} = \mathbf{G}_S^{\mathrm{H}}$.

Lemma 2: For given $\mathbf{A}_{m\times n}, \mathbf{B}_{p\times q}$, $(\mathbf{A}\otimes\mathbf{B})^{\mathrm{H}} = \mathbf{A}^{\mathrm{H}}\otimes\mathbf{B}^{\mathrm{H}}$. In particular, $(\mathbf{A}\otimes\mathbf{B})^{-1} = \mathbf{A}^{-1}\otimes\mathbf{B}^{-1}$ on condition that $\mathbf{A}$ and $\mathbf{B}$ are both invertible square matrices.

Corollary 1: Let $\mathbf{G}_S$ and $\mathbf{I}_R$ be a S-square stride permutations matrix and R-square identity matrix, respectively, then

$$(\mathbf{I}_R \otimes \mathbf{G}_S)^{-1} = (\mathbf{I}_R \otimes \mathbf{G}_S)^{\mathrm{H}}, \tag{45a}$$
$$(\mathbf{G}_S \otimes \mathbf{I}_R)^{-1} = (\mathbf{G}_S \otimes \mathbf{I}_R)^{\mathrm{H}}. \tag{45b}$$

More generally,

$$\left(\prod_{i=1}^{n}(\mathbf{I}_{R_i}\otimes\mathbf{G}_{S_i})\right)^{-1} = \left(\prod_{i=1}^{n}(\mathbf{I}_{R_i}\otimes\mathbf{G}_{S_i})\right)^{\mathrm{H}}, \tag{46a}$$
$$\left(\prod_{i=1}^{n}(\mathbf{G}_{S_i}\otimes\mathbf{I}_{R_i})\right)^{-1} = \left(\prod_{i=1}^{n}(\mathbf{G}_{S_i}\otimes\mathbf{I}_{R_i})\right)^{\mathrm{H}} \tag{46b}$$

provided $R_1S_1 = R_2S_2 = \cdots = R_nS_n$.

Lemma 3: For given $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n$ and $\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_n$, it holds

$$\left(\bigotimes_{i=1}^{n}\mathbf{A}_i\right)\left(\bigotimes_{i=1}^{n}\mathbf{B}_i\right) = \bigotimes_{i=1}^{n}\mathbf{A}_i\mathbf{B}_i. \tag{47}$$

APPENDIX B
PROOF OF PROPOSITION 1 INTRODUCED IN SECTION II

For $\mathbf{\Lambda}_N$ displayed in (3), let $R_i = 2^i$ and $S_i = N/2^i$, then from (46a) we have $\mathbf{\Lambda}_N^{-1} = \mathbf{\Lambda}_N^{\mathrm{H}}$, hence $\mathbf{\Lambda}_N\mathbf{\Lambda}_N^{\mathrm{H}} = \mathbf{I}_N$.

In terms of $\mathbf{M}_1(u)$, we have

$$\begin{aligned}&\mathbf{M}_1(u)\mathbf{M}_1^{\mathrm{H}}(u)\\ &= \left(\mathbf{I}_{2^u}\otimes\mathbf{W}_4^{(2)}\otimes\mathbf{I}_{N/2^{u+2}}\right)\cdot\left(\mathbf{I}_{2^u}\otimes\mathbf{W}_4^{(2)}\otimes\mathbf{I}_{N/2^{u+2}}\right)^{\mathrm{H}}\\ &= \mathbf{I}_{2^u}\otimes\left(\mathbf{W}_4^{(2)}\cdot\left(\mathbf{W}_4^{(2)}\right)^{\mathrm{H}}\right)\otimes\mathbf{I}_{N/2^{u+2}} = \mathbf{I}_N,\end{aligned}$$

where Lemma 3 is applied in the second step. Similarly, by adopting Lemma 3 to $\mathbf{M}_2(u,v)\mathbf{M}_2^{\mathrm{H}}(u,v)$ we obtain

$$\mathbf{M}_2(u,v)\mathbf{M}_2^{\mathrm{H}}(u,v) = \mathbf{I}_{2^u}\otimes\mathbf{\Gamma}\otimes\mathbf{I}_{N/2^{u+v}},$$
where
$$\begin{aligned}\mathbf{\Gamma} &= (\mathbf{G}_4\otimes\mathbf{I}_{2^{v-2}})\cdot\mathbf{W}_{2^v}^{(4)}\cdot(\mathbf{G}_4\otimes\mathbf{I}_{2^{v-2}})\\ &\quad\cdot\left[(\mathbf{G}_4\otimes\mathbf{I}_{2^{v-2}})\cdot\mathbf{W}_{2^v}^{(4)}\cdot(\mathbf{G}_4\otimes\mathbf{I}_{2^{v-2}})\right]^{\mathrm{H}}\\ &= (\mathbf{G}_4\otimes\mathbf{I}_{2^{v-2}})\cdot\mathbf{W}_{2^{v+1}}^{(4)}\cdot\left(\mathbf{W}_{2^v}^{(4)}\right)^{\mathrm{H}}\cdot(\mathbf{G}_4\otimes\mathbf{I}_{2^{v-2}})^{\mathrm{H}}\\ &= (\mathbf{G}_4\otimes\mathbf{I}_{2^{v-2}})\cdot(\mathbf{G}_4\otimes\mathbf{I}_{2^{v-2}})^{\mathrm{H}} = \mathbf{I}_{2^v}.\end{aligned}$$

Therefore, $\mathbf{M}_2(u,v)\mathbf{M}_2^{\mathrm{H}}(u,v) = \mathbf{I}_N$. Let

$$\mathbf{\Omega} = \prod_{i=1}^{v}(\mathbf{G}_{2^i}\otimes\mathbf{I}_{N/2^{u+i}}),$$

then from (46b) we obtain $\mathbf{\Omega}\mathbf{\Omega}^{\mathrm{H}} = \mathbf{I}_{N/2^u}$. Thus

$$\begin{aligned}&\mathbf{M}_3(u,v)\mathbf{M}_3^{\mathrm{H}}(u,v)\\ &= \mathbf{I}_{2^u}\otimes\left[\left(\mathbf{\Omega}\cdot\mathbf{W}_{N/N2^u}^{(2^v)}\cdot\mathbf{\Omega}\right)\left(\mathbf{\Omega}\cdot\mathbf{W}_{N/2^u}^{(2^u)}\cdot\mathbf{\Omega}\right)^{\mathrm{H}}\right]\\ &= \mathbf{I}_{2^u}\otimes\mathbf{I}_{N/2^u} = \mathbf{I}_N.\end{aligned}$$

Finally, $\mathbf{B}_{\mathrm{F}}(u)\mathbf{B}_{\mathrm{F}}^{\mathrm{H}}(u)$ can be simplified with the assist of (45b), that is

$$\begin{aligned}&\mathbf{B}_{\mathrm{F}}(u)\mathbf{B}_{\mathrm{F}}^{\mathrm{H}}(u)\\ &= (\mathbf{I}_{2^u}\otimes\mathbf{G}_{N/2^u})^{-1}(\mathbf{I}_{N/2}\otimes\mathbf{T}_2)(\mathbf{I}_{N/2}\otimes\mathbf{T}_2)^{\mathrm{H}}(\mathbf{I}_{2^u}\otimes\mathbf{G}_{N/2^u})\\ &= (\mathbf{I}_{2^u}\otimes\mathbf{G}_{N/2^u})^{-1}(\mathbf{I}_{N/2}\otimes(\mathbf{T}_2\mathbf{T}_2^{\mathrm{H}}))\cdot(\mathbf{I}_{2^u}\otimes\mathbf{G}_{N/2^u})\\ &= 2(\mathbf{I}_{2^u}\otimes\mathbf{G}_{N/2^u})^{-1}\cdot\mathbf{I}_N\cdot(\mathbf{I}_{2^u}\otimes\mathbf{G}_{N/2^u}) = 2\mathbf{I}_N.\end{aligned}$$

APPENDIX C
ERROR ANALYSIS OF THE MEMORYLESS CORDIC MODULE

Let $x = x_{\rm re} + j \cdot x_{\rm im}$ and $y = y_{\rm re} + j \cdot y_{\rm im}$ be the complex numbers, then the twiddle factor multiplication $y = x \cdot e^{j\theta}$ can be considered as a rotation of a 2-dimension vector:

$$\mathbf{y}_{\rm out} = \begin{bmatrix} y_{\rm re} \\ y_{\rm im} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_{\rm re} \\ x_{\rm im} \end{bmatrix} = \mathbf{C}(\theta)\mathbf{x}_{\rm in}.$$

The memoryless CORDIC scheme [26] decomposes the desired angle θ into a sum of predefined angles as

$$\theta = \mu_1\pi + \mu_2\frac{\pi}{2} + \sum_{t=1}^{T_c} v_t \tan^{-1}(2^{-t}) + \delta_\theta, \tag{48}$$

where $\mu_1, \mu_2 \in \{0, 1\}$, $v_t \in \{-1, 1\}$. Then the rotation matrix $\mathbf{C}(\theta)$ can be approximated as

$$\tilde{\mathbf{C}}(\theta) = S(1-2\mu_1)\cdot\mathbf{U}\prod_{t=1}^{T_c}\mathbf{\Theta}_t \tag{49}$$

with $S = \prod_{t=1}^{T_c}\cos\left(\tan^{-1}(2^{-t})\right) \approx 1 - 2^{-3} - 2^{-6}$ and

$$\mathbf{U} = \begin{bmatrix} 1-\mu_2 & -\mu_2 \\ \mu_2 & 1-\mu_2 \end{bmatrix}, \mathbf{\Theta}_t = \begin{bmatrix} 1 & -v_t 2^{-t} \\ v_t 2^{-t} & 1 \end{bmatrix}.$$

The π- and $\pi/2$-rotator involve no quantization errors in the fixed-point implementation. For the remaining T_c sub-rotators in CORDIC module, the t-bit ($t \in \{1, \ldots, T_c\}$) right-shift operations performed by the tth sub-rotator introduces roundoff noise $\mathbf{e}_t = [e_{t,\rm re}\ e_{t,\rm im}]^{\rm T}$ to the circuit. After T_c sub-rotations are accomplished, the obtained data should be multiplied with S to counteract the intrinsic gain of rotations, which produces additional noise $\mathbf{e}_s = [e_{s,\rm re} e_{s,\rm im}]^{\rm T}$ due to the 3-bit and 6-bit right-shift operations. Thus the output of scaling operation can be expressed as follows considering the foregoing roundoff effects:

$$\begin{aligned}\mathbf{y}_s = {} & S(1-2\mu_1)\mathbf{U}\sum_{t=1}^{T_c}\mathbf{\Theta}_t(\mathbf{x}_{\rm in} + \boldsymbol{\varepsilon}_{\rm in}) + \\ & S(1-2\mu_1)\mathbf{U}\sum_{t=1}^{T_c-1}\prod_{j=t+1}^{T_c}\mathbf{\Theta}_j\mathbf{e}_t + \mathbf{e}_{T_c} + \mathbf{e}_s,\end{aligned} \tag{50}$$

where $\mathbf{y}_s = [y_{s,\rm re} y_{s,\rm im}]^{\rm T} \in \mathbb{R}^2$, $\boldsymbol{\varepsilon}_{\rm in} = [\varepsilon_{\rm re}\varepsilon_{\rm im}]^{\rm T}$ is the noise superimposed on the input sample. Assuming that $\mathbf{e}_s$ is independent of $\mathbf{e}_t$, $t \in \{1, \ldots T_c\}$ and $tr\{\mathbb{E}(\mathbf{e}_{t_1}\mathbf{e}_{t_2}^{\rm T})\} = 0$ for $t_1 \neq t_2$. When $x_{\rm re}$, $x_{\rm im}$ are both represented using 1 sign bit along with $w-1$ data bits, it can be verified that $tr\{\mathbb{E}(\mathbf{e}_t\mathbf{e}_t^T)\} = 2/3 \cdot (1 + 2^{-t})2^{-2w}$, hence

$$\begin{aligned} & tr\{\mathbb{E}(\mathbf{y}_s\mathbf{y}_s^{\rm T})\} \\ = {} & tr\left\{\mathbb{E}(\mathbf{x}_{\rm in}\mathbf{x}_{\rm in}^{\rm T})\cdot S^2\mathbf{U}\sum_{t=1}^{T_c}\mathbf{\Theta}_t\left(\mathbf{U}\sum_{t=1}^{T_c}\mathbf{\Theta}_t\right)^{\rm T}\right\} + \\ & tr\left\{\mathbb{E}(\varepsilon_{\rm in}\varepsilon_{\rm in}^{\rm T})\cdot S^2\mathbf{U}\sum_{t=1}^{T_c}\mathbf{\Theta}_t\left(\mathbf{U}\sum_{t=1}^{T_c}\mathbf{\Theta}_t\right)^{\rm T}\right\} + \\ & tr\left\{S^2\sum_{t=1}^{T_c-1}\mathbb{E}(\mathbf{e}_t\mathbf{e}_t^{\rm T})\cdot\mathbf{U}\prod_{j=t+1}^{T_c}\mathbf{\Theta}_j\left(\mathbf{U}\prod_{j=t+1}^{T_c}\mathbf{\Theta}_j\right)^{\rm T}\right\} + \\ & tr\{\mathbb{E}(\mathbf{e}_s\mathbf{e}_s^{\rm T})\} + tr\{\mathbb{E}(\mathbf{e}_{T_c}\mathbf{e}_{T_c}^{\rm T})\}. \end{aligned}$$

Let $\varepsilon = \varepsilon_{\rm re} + j\cdot\varepsilon_{\rm im} \in \mathbb{C}$. We further take $\mathbf{U}\mathbf{U}^{\rm T} = \mathbf{I}_2$ and $\mathbf{\Theta}_t\mathbf{\Theta}_t^{\rm T} = (1 + 2^{-2t})\cdot\mathbf{I}_2$ into account, it yields

$$\begin{aligned} & tr\{\mathbb{E}(\mathbf{y}_s\mathbf{y}_s^{\rm T})\} \\ = {} & \mathbb{E}(|x|^2) + \mathbb{E}(|\varepsilon|^2) + \frac{2}{3}\cdot 2^{-2w}\left(3 + 2^{-3} + 2^{-6} + 2^{-T_c}\right) \\ & + \frac{2}{3}\cdot 2^{-2w}\cdot S^2\sum_{t=1}^{T_c-1}(1 + 2^{-t})\prod_{j=t+1}^{T_c}(1 + 2^{-2j}). \end{aligned} \tag{51}$$

In terms of δ_θ, the residual error of angle approximation in (48), it also brings inaccuracy $e_\theta \in \mathbb{C}$ to the result. Let $x_{\rm ini}$ be the complex input of FFT and x be the output sample of the lth butterfly unit, then from [30] we have

$$\mathbb{E}(|e_\theta|^2) \approx tr\{\mathbb{E}(\mathbf{y}_s\mathbf{y}_s^{\rm T})\}\left(\tan^{-1}(2^{-T_c})\right)^2 \approx \mathbb{E}(|x|^2)2^{-2T_c}. \tag{52}$$

Considering both (51) and (52), we arrive at the following conclusions: i) the investigated CORDIC module provides an unit gain to the input $(x_{\rm re} + \varepsilon_{\rm re}) + j\cdot(x_{\rm im} + \varepsilon_{\rm im})$, in this respect it coincides with the complex multiplier; ii) when the CORDIC module rotates the input by a non-trivial angle, the operation will introduce additional noise to the output. The newly-generated noise power is given by

$$\begin{aligned}\sigma^2 = {} & \frac{2^{-2w+1}}{3}S^2\sum_{t=1}^{T_c-1}(1 + 2^{-t})\prod_{j=t+1}^{T_c}(1 + 2^{-2j}) + \\ & \frac{2^{-2w+1}}{3}(3 + 2^{-3} + 2^{-6} + 2^{-T_c}) + \mathbb{E}(|x|^2)2^{-2T_c}.\end{aligned} \tag{53}$$

APPENDIX D
PROOF OF PROPOSITION 2 INTRODUCED IN SECTION III

Note $\mathbf{M}_2(u, v)\mathbf{B}_{\rm F}(u)$ can be rewritten as follows from (6) and (8b):

$$\mathbf{M}_2(u, v)\mathbf{B}_{\rm F}(u) = \mathbf{I}_{2^u} \otimes (\mathbf{W}\cdot\mathbf{B}) \tag{54}$$

with

$$\mathbf{W} = \left((\mathbf{G}_4 \otimes \mathbf{I}_{2^{v-2}})\mathbf{W}_{2^v}^{(4)}(\mathbf{G}_4 \otimes \mathbf{I}_{2^{v-2}})\right) \otimes \mathbf{I}_{N/2^{u+v}}, \tag{55a}$$

$$\mathbf{B} = \mathbf{G}_{N/2^u}^{-1}\left(\mathbf{I}_{N/2^{u+1}} \otimes \mathbf{T}_2\right)\mathbf{G}_{N/2^u}. \tag{55b}$$

For a given $N/2^u$-dimension input $\mathbf{x}_b$, $\mathbf{y}_b = \mathbf{B}\mathbf{x}_b$ allocate the first $N/2^{u+1}$ positions to addition results, while subtractions serve as the last $N/2^{u+1}$ entries. Therefore, to weight all the additions by trivial twiddle factors, the first $N/2^{u+1}$ entries located at the main diagonal of $\mathbf{W}$ should be trivial coefficients. However, the prerequisite is the amount of non-trivial twiddle factors in $\mathbf{W}$ is no less than $N/2^{u+1}$, this yields

$$(2^v - 2^{v-2} - 4)\cdot N/2^{u+v} \leqslant N/2^{u+1}, \tag{56}$$

where $2^v - 2^{v-2} - 4$ is the number of trivial factors in $\mathbf{W}_{2^v}^{(4)}$, which is obtained from (20). Note $\mathbf{M}_2(u, v)$ includes non-trivial

twiddle factors, thus $v \geq 3$ according to (34). In this way, the candidates satisfying (56) are locked onto $v = 3, 4$, and

$$\mathbf{W}\big|_{v=3} = \mathrm{diag}\left(1\ 1\ 1\ -j\ 1\ e^{-j\pi/4}\ 1\ e^{-j3\pi/4}\right) \otimes \mathbf{I}_{N/2^{u+3}},$$
$$\mathbf{W}\big|_{v=4} = \mathrm{diag}\big(1\ 1\ 1\ 1\ 1\ e^{-j\pi/4}\ -j\ e^{-j3\pi/4}\ 1\ e^{-j\pi/8}$$
$$e^{-j\pi4}\ e^{-j3\pi/8}\ 1\ e^{-j3\pi/8} e^{-j3\pi/4} e^{-j9\pi/8}\big) \otimes \mathbf{I}_{N/2^{u+4}}.$$

It can be verified that $\mathbf{W}\big|_{v=3}$ satisfies the constraint while $\mathbf{W}\big|_{v=4}$ does not. Thus from (54), $\mathbf{M}_2(u, v)$ meet the requirement if and only if $v = 3$.

In terms of $\mathbf{M}_3(u, v)\mathbf{B}_{\mathrm{F}}(u)$, we have,

$$\mathbf{M}_3(u, v)\mathbf{B}_{\mathrm{F}}(u) = \mathbf{I}_{2^u} \otimes (\mathbf{W}' \cdot \mathbf{B}), \tag{57}$$

where

$$\mathbf{W}' = \prod_{i=1}^{v} \left(\mathbf{G}_{2^i} \otimes \mathbf{I}_{N/2^{u+i}}\right) \cdot \mathbf{W}_{N/2^u}^{(2^v)} \prod_{i=1}^{v} \left(\mathbf{G}_{2^i} \otimes \mathbf{I}_{N/2^{u+i}}\right). \tag{58}$$

Similar to (56), $\mathbf{W}'$ should firstly satisfies

$$N/2^u - N/2^{u+v} - 2^v \leqslant N/2^{u+1},$$

or

$$\frac{1}{2^{v-1}} + 2\left(\frac{2^{u+v}}{N}\right) \geqslant 1 \tag{59}$$

equivalently. As $\mathbf{M}_3(u, v)$ contains non-trivial coefficients, the selection of u, v should firstly guarantee $N/2^u/ > 2^v$ and $N/2^u \geqslant 8$ from (35). In this case, $v = 1$ meet the constraint in (59). This yields

$$\mathbf{W}'\big|_{v=1} = \mathbf{W}_{N/2^u}^{(2)} = \mathrm{quasidiag}\left(\mathbf{I}_{N/2^{u+1}}, \boldsymbol{w}_{N/2^{u+1}}\right). \tag{60}$$

Clearly, the first $N/2^{u+1}$ entries located at the main diagonal of $\mathbf{W}'\big|_{v=1}$ are trivial coefficients. Moreover, (59) also holds for the situation when $v \geqslant 2$ and $N = 2^{u+v+1}$. and

$$\mathbf{W}'\big|_{v\geqslant 2, N=2^{u+v+1}}$$
$$= \prod_{i=1}^{v} (\mathbf{G}_{2^i} \otimes \mathbf{I}_{2^{v+1-i}}) \cdot \mathbf{W}_{2^{v+1}}^{(2^v)} \cdot \prod_{i=1}^{v} (\mathbf{G}_{2^i} \otimes \mathbf{I}_{2^{v+1-i}}). \tag{61}$$

It can be verified that trivial and non-trivial coefficients are permuted alternately in the main diagonal of $\mathbf{W}_{2^{v+1}}^{(2^v)}$ for arbitrary $v \geq 2$. Unfortunately, this property is inherited by $\mathbf{W}'$ in spite of the additional transformations in (61). As a result, $\mathbf{M}_3(u, v)$ weight all the additions in butterfly outputs by non-trivial twiddle factors if and only if $v = 1$.

REFERENCES

[1] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, pp. 297–301, Apr. 1965.

[2] K. R. Rao, D. N. Kim, and J. J. Hwang, *Fast Fourier Transform: Algorithms and Applications*. New York, NY, USA: Springer, 2010.

[3] A. Cortés, I. Vélez, 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.

[4] T. Thong and B. Liu, "Fixed-point fast Fourier transform error analysis," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 24, no. 6, pp. 563–573, Jun. 1976.

[5] P. Kabal and B. Sayar, "Performance of fixed-point FFT's: rounding and scaling considerations," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Jun. 1986, vol. 11, pp. 221–224.

[6] D. James, "Quantization errors in the fast Fourier transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 23, no. 3, pp. 277–283, Mar. 1975.

[7] V. Palfi and I. Kollar, "Roundoff errors in fixed-point FFT," in *Proc. IEEE Int. Symp. Intell. Signal Process. (WISP)*, 2009, pp. 87–91.

[8] W. -H. Chang and T. Q. Nguyen, "On the fixed-point accuracy analysis of FFT algorithms," *IEEE Trans. Signal Process.*, vol. 56, no. 10, pp. 4673–4682, Oct. 2008.

[9] O. Sarbishei and K. Radecka, "Analysis of mean-square-error (MSE) for fixed-point FFT units," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2011, pp. 1732–1735.

[10] M. R. Mohammadnia and L. Shannon, "Minimizing the error: A study of the implementation of an integer split-radix FFT on an FPGA for medical imaging," in *Proc. Int. Conf. Field-Programmable Tech. (FPT)*, 2012, pp. 360–367.

[11] C. -J. Wei, S. -M. Liu, S.-J. Chen, and Y. -H. Hu, "Optimal fixed-point fast Fourier transform," in *Proc. IEEE Workshop. Signal Process. Syst. (SiPS)*, 2013, pp. 377–382.

[12] M. Bekooij, J. Huisken, and K. Nowak, "Numerical accuracy of fast Fourier transforms with CORDIC arithmetic," *J. VLSI Signal Process.*, vol. 25, no. 2, pp. 187–193, Feb. 2000.

[13] O. Sarbishei and K. Radecka, "On the fixed-point accuracy analysis and optimization of FFT units with CORDIC multipliers," in *Proc. IEEE Symp. Comput. Arithmetic (ARITH)*, 2011, pp. 62–69.

[14] P. S. Yoon and Y. Y. Jun, "Fixed-point analysis and parameter selections of MSR-CORDIC with applications to FFT designs," *IEEE Trans. Signal Process.*, vol. 60, no. 12, pp. 6245–6256, Dec. 2012.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] M. Turrillas, A. Cortés, J. F. Sevillano, I. Vélez, C. Oria, A. Irizar, and V. Baena, "Comaprison of area-efficient FFT algorithms for DVB-T2 receivers," *Electron. Lett.*, vol. 46, no. 15, pp. 1088–1089, Jul. 2010.

[25] M. Turrillas, A. Cortés, I. Vélez, J. F. Sevillano, and A. Irizar, "An area-efficient radix-2^8 FFT algorithm for DVB-T2 receivers," *Microelectron. J.*, vol. 45, no. 10, pp. 1311–1318, Oct. 2014.

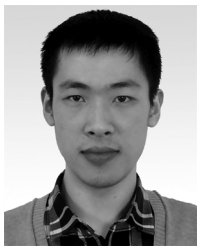
[26] M. Garrido and J. Grajal, "Efficient memoryless CORDIC for FFT computation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2007, vol. 2, pp. II-113–II-116.

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

[28] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Sci.*, vol. 220, no. 4598, pp. 671–680, 1983.

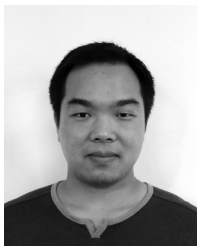
[29] R. A. Horn and R. C. Johnson, *Topics in Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1991.

[30] Y. H. Hu, "The quantization effects of the CORDIC algorithm," *IEEE Trans. Signal Process.*, vol. 40, no. 4, pp. 834–844, Apr. 1992.



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

His research interests are in the area of 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, P.R. China, in 2003 and his Ph.D. degree in Information and communication engineering from the National University of Defense Technology (NUDT), Changsha, P.R. China, in 2009. Since March 2010, he has been with NUDT, Changsha, China, where he is currently a lecturer.

His 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 M.S. degree in information and communication engineering from the National University of Defense Technology (NUDT), Changsha, P.R. China, in 2011 and 2013, respectively. She is currently pursuing the Ph.D. degree with the School of Electronic Science and Engineering, NUDT.

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

Jibo Wei (M'04) received his B.S. degree and M.S. degree from the National University of Defense Technology (NUDT), Changsha, P.R. China, in 1989 and 1992, respectively, and the Ph.D. degree from Southeast University, Nanjing, P.R. China, in 1998, all in electronic engineering. He is currently the director and a professor of the Department of Communication Engineering of NUDT. His research interests include design and optimization of communication systems and signal processing in communications, more specially, the areas of MIMO, Multicarrier transmission, cooperative communication, and cognative network.

Prof. Wei is the member of the IEEE Communication Society and also the member of the IEEE VTS. He also works as one of the editors of *Journal on Communications* and the senior member of China Institute of Communications and Electronics respectively.