A Fast Algorithm With Less Operations for Length-$N = q \times 2^m$ DFTs

Kenli Li, *Member, IEEE*, Weihua Zheng, and Keqin Li, *Fellow, IEEE*

***Abstract*—Discrete Fourier transform (DFT) is widely used in almost all fields of science and engineering. Fast Fourier transform (FFT) is an efficient tool for computing DFT. In this paper, we present a fast Fourier transform (FFT) algorithm for computing length-$q \times 2^m$ DFTs. The algorithm transforms all q-points sub-DFTs into three parts. In the second part, the operations of sub-transformation contain only multiplications by real constant factors. By transformation, length-2^m-scaled DFTs (SDFT) are obtained. An extension of scaled radix-2/8 FFT (SR28FFT) is presented for computing these SDFTs, in which, the real constant factors of SDFTs are attached to the coefficients of sub-DFTs to simplify multiplication operations. The proposed algorithm achieves reduction of arithmetic complexity over the related algorithms. It can achieve a further reduction of arithmetic complexity for computing a length-$N = q \times 2^m$ IDFT by $2N - 4m$ real multiplications. In addition, the proposed algorithm is applied to real-data FFT, and is extended to 6^m DFTs.**

***Index Terms*—Fast Fourier transform (FFT), inverse DFT (IDFT), quantization error, radix-2/8 FFT, scaled DFT (SDFT).**

I. Introduction

THE discrete Fourier transform (DFT) is an important tool for many applications in scientific and engineering fields [1]–[3]. DFT is generally implemented with an efficient technique called fast Fourier transform (FFT). Since the introduction of Cooley and Tukey's algorithm in 1965 [4], considerable research has been conducted resulting in a large number of algorithms for length-2^m DFTs [5]–[8]. One of these algorithms, split-radix FFT (SRFFT), achieved a reduction of order of magnitude of computational complexity over Cooley and Tukey's algorithm. The algorithm was presented by Yavne in 1968 [9] and subsequently rediscovered simultaneously by various authors [5], [10], [11] in 1984. Grigoryan and Agaian [12] presented another algorithm, that reduces the number of arithmetic operations for DFTs of lengths greater than 256 at the expense of more complicated structure compared with SRFFT; however, it is computationally less efficient for smaller lengths. The record of computational complexity, set by the radix-2/4 FFT for computing powers-of-two DFTs, was broken by Johnson and Frigo in 2007 [13]. Subsequently, the lowest operation count for powers-of-two FFTs was obtained by Zheng *et al.* in 2014 [14]. Another contribution of [14] is that a method of high comprehensive properties was proposed and the method can efficiently compute a power-of-two DFT with less arithmetic operations, higher computation accuracy, and less accesses to the lookup table of twiddle factors over SRFFT.

The radix-2/4 FFT, the radix-2/8 FFT, the MSRFFT, and the scaled radix-2/8 FFT were mainly proposed for length-2^m DFTs. In order to explore the performance of these approaches in the other DFTs, many works have been made towards extending them to other length DFT. Grigoryan and Agaian [15] gave an algorithm which extends Cooley-Tukey algorithm to composite length DFTs. The idea of the radix-2/4 FFT proposed in [5] has been extended to the length-p^m [16] and length-$q \times 2^m$ DFTs [17]. The algorithm in [17] decomposes a length-N DFT into one length-$N/2$ DFT and two length-$N/4$ DFTs and repeats successively the process until the size is reduced to a q-point or a $2q$-point DFT. It has been shown that this algorithm optimizes the number of arithmetic operations compared with other reported algorithms [16], [18]–[20]. The idea of radix-2/8 FFT has also been extended to the length-$q \times 2^m$ DFTs [21]. It has been shown that the algorithm in [21] optimizes the number of data transfers, address generations, and twiddle factor evaluations or accesses to the lookup table. The MSRFFT and Cooley-Tukey transform have also been extended to length-$q \times 2^m$ [22], and achieved a reduction of arithmetic complexity over the algorithms in [17], [21]. However, until now, no attempt has been made in developing an FFT algorithm that explores the performance of the scaled radix-2/8 FFT (SR28FFT) algorithm in length-$q \times 2^m$ DFTs.

Applications in which lengths of DFTs are $q \times 2^m$ are arising recently. The following are several examples.

1) In the applications of orthogonal frequency division multiplexing (OFDM) demodulation and modern microscopy, the sequence lengths may be non-powers-of-two [21].
2) FFT is an efficient tool to compute window modified discrete cosine transform (MDCT). The MPEG audio coding standard uses dynamically window MDCT to achieve high quality performance. In layer III of MDCT-I and

Manuscript received July 17, 2014; revised October 14, 2014; accepted November 26, 2014. Date of publication December 10, 2014; date of current version January 07, 2015. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Hing Cheung So. This research was partially funded by the Key Program of National Natural Science Foundation of China (Grant No. 61133005, 61432005), and the National Natural Science Foundation of China (Grant Nos. 61070057, 61173013, 61472136, 61370095, 61472124), Ministry of Education of China (Grant No. 708066), the Ph.D. Programs Foundation of Ministry of Education of China (20100161110019), Project supported by the National Science Foundation for Distinguished Young Scholars of Hunan (12JJ1011).

K. Li is with the College of Information Science and Engineering, Hunan University, Changsha 410082, China (e-mail: lkl@hnu.edu.cn).

W. Zheng is with the College of Information Science and Engineering, Hunan University, Changsha 410082, China (e-mail: zhengdavid@hnu.edu.cn).

K. Li is with the College of Information Science and Engineering, Hunan University, Changsha 410082, China, and also with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lkq@hnu.edu.cn; lik@newpaltz.edu).

Digital Object Identifier 10.1109/TSP.2014.2379678

MDCT-II, the length of data blocks is $N \neq 2^m$. The layer III specifies a longer block ($N = 36$) and a shorter block ($N = 12$) [23].

3) A 1536-points FFT processor is used in Third Generation Partnership Project Long-Term Evolution [24].

Therefore, the purpose of this paper is to develop a new algorithm for computing a length-$N = q \times 2^m$ DFT, reducing the number of operations and the quantization errors. The algorithm divides a length-$N = q \times 2^m$ DFT into q DFTs of length-2^m in terms of radix-q FFT algorithm. The most outer decomposition and assemblage can be partitioned into three subparts. In the intermediate sub-part, every input is only multiplied by a real constant factor. The real constant factors and sub-DFTs of length-2^m can be combined as length-2^m SDFTs. These SDFTs of length-2^m are computed with the algorithm that is based on the SR28FFT algorithm, which is presented in Section III. Hence, a further improvement in arithmetic complexity and quantization error is obtained.

The contributions of this paper are listed as follows:

- The proposed algorithm has lower arithmetic complexities and smaller quantization errors than algorithms in [17], [21], [22] for length-$N = q \times 2^m$ DFT.
- We achieve a further reduction of arithmetic complexity for computing a length-$N = q \times 2^m$ IDFT by $2N - 4m$ real multiplications.
- The proposed algorithm is extended to length-$N = 6^m$ DFT.
- The proposed algorithm is applied to real-data FFT.

The remainder of the paper is organized as follows. Section II presents a complete derivation of the proposed algorithm. Section III introduces an algorithm to compute the sub-SDFTs of length-2^m. Section IV analyze the performance of the proposed algorithm by analyzing computational complexity and quantization error and comparing them with the existing algorithms. The algorithm extended to 6^m DFTs and its computational complexity are presented in Section V. An example, i.e., the computation of 5×2^m DFTs, is illustrated in Section VI in detail. Section VII introduces the implementation of IDFT with the proposed algorithm. Section VIII discusses real-data FFT. Section IX gives conclusions.

II. The Proposed Algorithm

Given a length-$N = q \times 2^m$ sequence x_n, where q is a small odd, its DFT is also a length-N sequence defined by

$$X_k = \sum_{n=0}^{N-1} x_n w_N^{nk}, \quad k = 0, 1, \ldots, N-1, \tag{1}$$

where $j = \sqrt{-1}$, and $w_N = e^{-j2\pi/N}$. Eq. (1) can be expressed in matrix form as follows,

$$X = \mathbf{F}_N x, \tag{2}$$

where x, X are the vector expression of input sequence and the vector expression of output sequence respectively, and $\mathbf{F}_N$ is the coefficient matrix of the DFT.

For a length-$N = q \times 2^m$ DFT, we now consider its decomposition according to radix-q FFT algorithm. All small odd q DFTs can be decomposed into three parts [25], [26], in which the first "pre-weave" stage and the final "post-weave" stage consist only of additions, negations, and multiplications by j. The middle stage contains only multiplication by real coefficients. Wang [25] discussed three types of basic sparse matrices $\mathbf{B_1}$, $\mathbf{B_2}$, and $\mathbf{B_3}$ which often appear in the calculations of small prime DFTs including 9-points DFT, and presented a method to decompose matrices of small prime DFTs into these three types of matrices. In what follows the coefficient matrix of a length-$N = q \times 2^m$ DFT is decomposed and is factorized accordingly. Let index n in Eq. (1) be

$$n = (n_0 q + n_1 N/q) \mod N, \tag{3}$$

where $n_0 \in [0, N/q)$ and $n_1 \in [0, q)$. Let y_{n_0,n_1} represent $x_{(n_0 q + n_1 N/q)|N}$, $\mathbf{Y}$ is a two dimensional vector expression of the inputs of the DFT, and $\mathbf{Y}$ is defined as follows:

$$\mathbf{Y} = \begin{bmatrix} y_{0,0} & y_{N/q,0} & \cdots & y_{N-N/q,0} \\ y_{0,q} & y_{N/q,q} & \cdots & y_{N-N/q,q} \\ \cdots & \cdots & \cdots & \cdots \\ y_{0,N-q} & y_{N/q,N-q} & \cdots & y_{N-N/q,N-q} \end{bmatrix}. \tag{4}$$

Let q and N/q be two decomposition factors of k, such that $k = k_0 q + k_1 N/q$. The DFT in (1) provides transform decomposition as follows:

$$\begin{aligned} X_k &= \sum_{n_0=0}^{N/q-1} \sum_{n_1=0}^{q-1} y_{n_0,n_1} w_N^{(n_0 q + n_1 N/q)k} \\ &= \sum_{n_1=0}^{q-1} \sum_{n_0=0}^{N/q-1} y_{n_0,n_1} w_{N/q}^{n_0 (q k_0)} w_q^{n_1 (N/q k_1)}. \end{aligned} \tag{5}$$

As mentioned above, the coefficient matrix $\mathbf{F}_q$ can be factorized into a product of three matrices. Let $\mathbf{L}_q$, $\mathbf{M}_q$, and $\mathbf{R}_q$ be these three matrices, where $\mathbf{L}_q$, $\mathbf{M}_q$, and $\mathbf{R}_q$ are q order matrices and $\mathbf{M}_q$ is a diagonal matrix. We have $\mathbf{F}_q = \mathbf{L}_q \mathbf{M}_q \mathbf{R}_q$, and (5) can be expressed as follows,

$$\begin{aligned} X &= \mathbf{F}_q (\mathbf{F}_{N/q} \mathbf{Y})^T \\ &= \mathbf{L}_q \mathbf{M}_q \mathbf{R}_q (\mathbf{F}_{N/q} \mathbf{Y})^T \\ &= \mathbf{L}_q \mathbf{M}_q \left(\mathbf{F}_{N/q} (\mathbf{R}_q \mathbf{Y}^T)^T \right)^T, \end{aligned} \tag{6}$$

where ^{T} is the operator of matrix transposition, and $\mathbf{M}_q (\mathbf{F}_{N/q} (\mathbf{R}_q \mathbf{Y}^T)^T)^T$ are SDFTs.

For a length-$N = 3 \times 2^m$ DFT, the sub-DFT of length-3 can be divided as

$$\mathbf{R_3} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{A_3}, \tag{7}$$

$$\mathbf{M_3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{3}{2} & 0 \\ 0 & 0 & -j s_3^1 \end{bmatrix}, \tag{8}$$

$$\mathbf{L_3} = \mathbf{A_3} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{9}$$

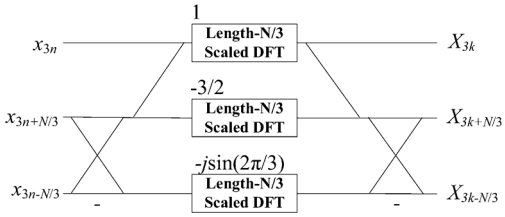

Fig. 1. Flow graph of length-3×2^m DFT.

where $s_q^n = \sin 2\pi n/q$, and

$$\mathbf{A}_q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & I_{M-1} & \bar{I}_{M-1} \\ 0 & \bar{I}_{M-1} & -I_{M-1} \end{bmatrix}, \tag{10}$$

with $M = (q-1)/2$, I_{M-1} is the $M-1$ order identity matrix, and $\bar{I}_{M-1}$ is the $M-1$ order anti-diagonal identity matrix. Fig. 1 shows the signal flow graph of the length-$N = 3 \times 2^m$ DFT. The diagonal elements in $\mathbf{M_3}$ are attached to length-$N/3$ sub-DFTs and combined as five length 2^m SDFTs with scaling factors 1, $-5/4$, $(c_1 - c_2)/2$, $-js_1$, and $-js_1$ respectively.

For a length-$N = 5 \times 2^m$ DFT, the sub-transformations of length-5 can be decomposed as

$$\mathbf{R_5} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \mathbf{I_3} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathbf{H_2} & 0 \\ 0 & 0 & \mathbf{I_2} \end{bmatrix} \mathbf{A_5}, \tag{11}$$

$$\mathbf{M_5} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & -\frac{5}{4} & 0 & 0 & 0 \\ 0 & 0 & \frac{c_5^1 - c_5^2}{2} & 0 & 0 \\ 0 & 0 & 0 & -js_5^1 & 0 \\ 0 & 0 & 0 & 0 & -js_5^1 \end{bmatrix}, \tag{12}$$

$$\mathbf{L_5} = \mathbf{A_5} \begin{bmatrix} \mathbf{A_3} & 0 & 0 \\ 0 & -1 & s_5^2/s_5^1 \\ 0 & s_5^2/s_5^1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & \mathbf{I_3} \end{bmatrix}, \tag{13}$$

where $c_q^k = \cos 2k\pi/q$, and

$$\mathbf{H_2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Fig. 2 shows the flow graph of the size-5×2^m DFT. The diagonal elements in $\mathbf{M_5}$ are attached to the length-$N/5$ sub-DFTs and combined as SDFTs.

For a length-$N = 7 \times 2^m$ DFT, the sub-transformations of length-7 can be decomposed as

$$\mathbf{R_7} = \mathbf{H_{71}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathbf{H_{31}} & 0 \\ 0 & 0 & \bar{\mathbf{I}}_3\mathbf{H_{31}} \end{bmatrix} \mathbf{A}_7, \tag{14}$$

$$\mathbf{M_7} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_0 - 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -jb_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -jb_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -jb_3 \end{bmatrix}, \tag{15}$$

$$\mathbf{L_7} = \mathbf{A_7} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathbf{H_{32}} & 0 \\ 0 & 0 & \bar{\mathbf{I}}_3\mathbf{H_{32}} \end{bmatrix} \mathbf{H_{72}}, \tag{16}$$

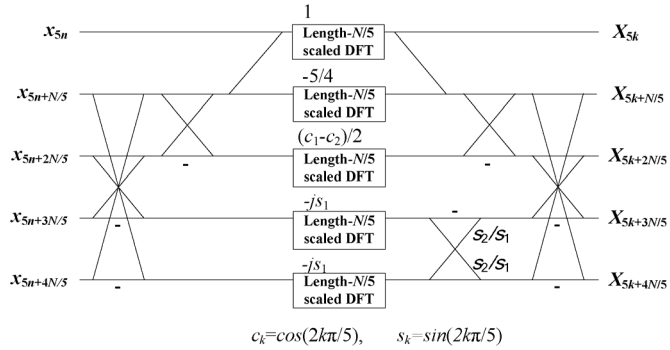

Fig. 2. Flow graph of length-5×2^m DFT.

where $c_q^k = \cos 2k\pi/q$,

$$\mathbf{H_{71}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{17}$$

$$\mathbf{H_{72}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{a_2}{a_3} & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -\frac{a_1}{a_3} & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{b_1}{b_3} & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -\frac{b_2}{b_3} \end{bmatrix}, \tag{18}$$

$$\mathbf{H_{31}} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & -1 & 1 \\ -1 & 1 & 1 \end{bmatrix}, \tag{19}$$

$$\mathbf{H_{32}} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{20}$$

Fig. 3 shows the flow graph of the size-7×2^m DFT. The diagonal elements in $\mathbf{M_7}$ are attached to the length-$N/7$ sub-DFTs and combined as SDFTs.

The integer 9 is a composite number. However, a 9-points DFT is evaluated with 9 treated as a prime other than a composite number in [25]. So, for a length-$N = 9 \times 2^m$ DFT, its 9-points sub-DFT is decomposed as follows,

$$\mathbf{R}_9 = \mathbf{H_{92}} \begin{bmatrix} \mathbf{I_3} & 0 & 0 \\ 0 & \mathbf{H_{31}} & 0 \\ 0 & 0 & \bar{\mathbf{I}}_3\mathbf{H_{31}} \end{bmatrix} \begin{bmatrix} \mathbf{A_3} & 0 & 0 \\ 0 & \mathbf{I_3} & \bar{\mathbf{I}}_3 \\ 0 & \bar{\mathbf{I}}_3 & -\mathbf{I_3} \end{bmatrix}, \tag{21}$$

$$\mathbf{M_9} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{3}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -js_3^1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_9^4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_9^4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -js_3^1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -js_9^4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -js_9^4 \end{bmatrix}, \tag{22}$$

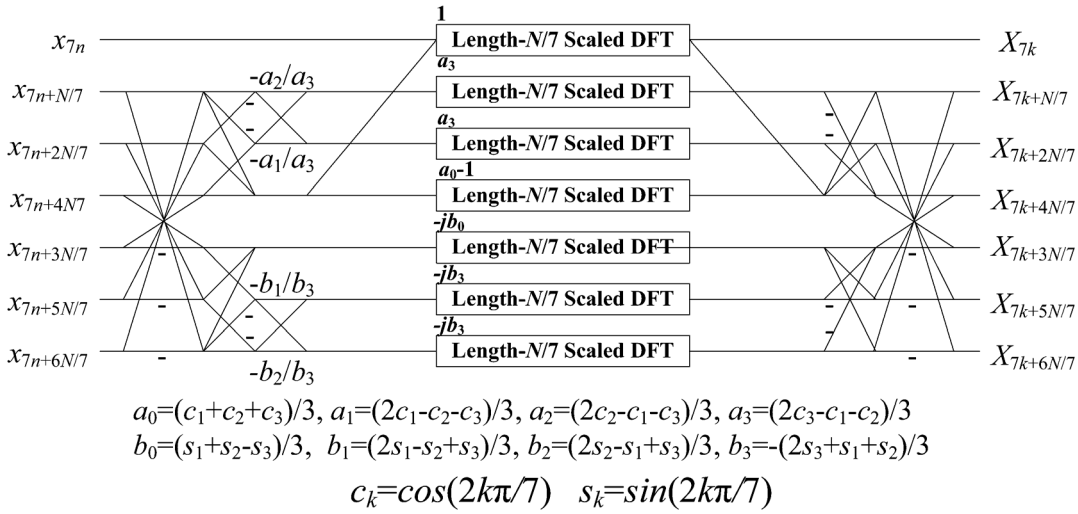

Fig. 3. Flow graph of length-7×2^m DFT.

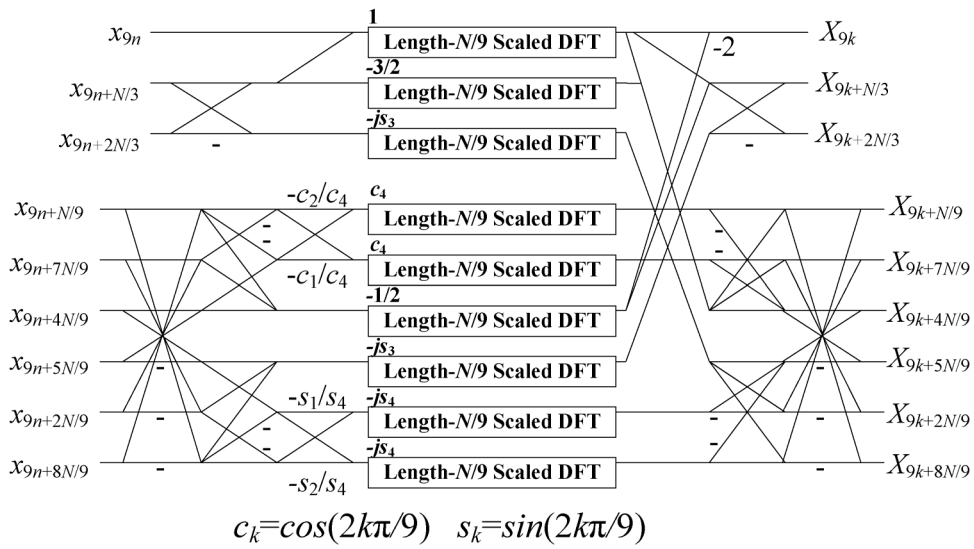

Fig. 4. Flow graph of length-9×2^m DFT.

$$\mathbf{L_9} = \begin{bmatrix} \mathbf{A_3} & 0 & 0 \\ 0 & \mathbf{I_3} & \bar{\mathbf{I}}_\mathbf{3} \\ 0 & \bar{\mathbf{I}}_\mathbf{3} & -\mathbf{I_3} \end{bmatrix} \begin{bmatrix} \mathbf{I_3} & 0 & 0 \\ 0 & \mathbf{H_{32}} & 0 \\ 0 & 0 & \bar{\mathbf{I}}_\mathbf{3}\mathbf{H_{32}} \end{bmatrix} \mathbf{H_{91}}, \quad (23)$$

where the input matrix is the transposition of $[x_0\ x_3\ x_6\ x_1\ x_7\ x_4\ x_5\ x_2\ x_8]$ and

$$\mathbf{H_{91}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (24)$$

$$\mathbf{H_{92}} = \begin{bmatrix} \mathbf{I_3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{c_9^2}{c_9^4} & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -\frac{c_9^1}{c_9^4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{s_9^1}{s_9^4} & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & \frac{s_9^2}{s_9^4} \end{bmatrix}, \quad (25)$$

Fig. 4 shows the flow graph of the length-9×2^m DFT. The diagonal elements in $\mathbf{M_9}$ are attached to length-$N/9$ sub-DFTs and combined as SDFTs.

We assume that q_1 and q_2 are two odd numbers and co-prime with each other. For a DFT of length-$q = q_1 \times q_2$ sub-DFT, we can use (6) to yield a transformation of three stages, in which q, q_1, and q_2 correspond respectively to N, q, and 2^m in (6).

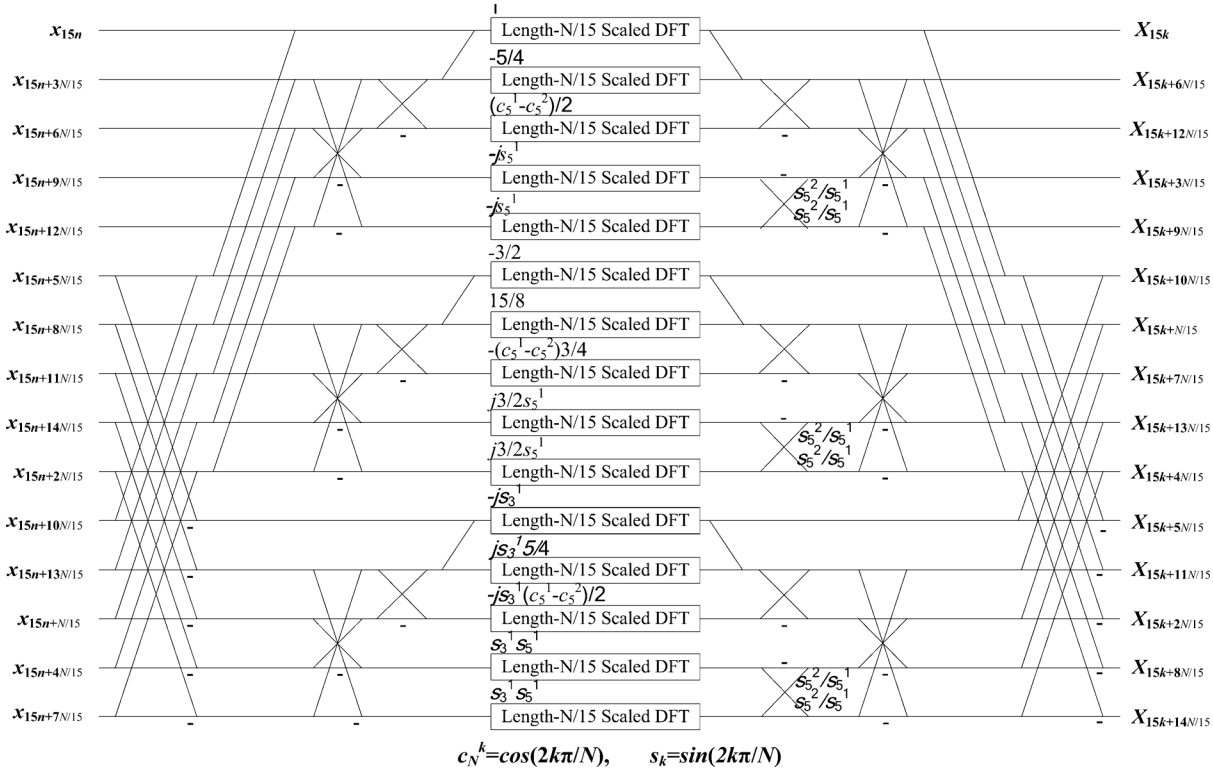

Fig. 5. Flow graph of length-15×2^m DFT.

In this way, we can perform the following decomposition for a length-$N = 15 \times 2^m$ DFT:

$$\mathbf{R_{15}} = \begin{bmatrix} I_6 & 0 \\ 0 & \mathbf{A_9} \end{bmatrix} \begin{bmatrix} I_5 & I_5 & 0 \\ 0 & I_5 & 0 \\ 0 & 0 & I_5 \end{bmatrix} \begin{bmatrix} \mathbf{R_5} & 0 & 0 \\ 0 & \mathbf{R_5} & 0 \\ 0 & 0 & \mathbf{R_5} \end{bmatrix}, \quad (26)$$

$$\mathbf{M_{15}} = \begin{bmatrix} \mathbf{M_5} & 0 & 0 \\ 0 & -\frac{3}{2}\mathbf{M_5} & 0 \\ 0 & 0 & -js_3^1\mathbf{M_5} \end{bmatrix}, \quad (27)$$

$$\mathbf{L_{15}} = \begin{bmatrix} I_6 & 0 \\ 0 & \mathbf{A_9} \end{bmatrix} \begin{bmatrix} I_5 & 0 & 0 \\ I_5 & I_5 & 0 \\ 0 & 0 & I_5 \end{bmatrix} \begin{bmatrix} \mathbf{L_5} & 0 & 0 \\ 0 & \mathbf{L_5} & 0 \\ 0 & 0 & \mathbf{L_5} \end{bmatrix}. \quad (28)$$

Fig. 5 shows the flow graph of the length-$N = 15 \times 2^m$ DFT. The diagonal elements in $\mathbf{M_{15}}$ are attached to length-$N/15$ sub-DFTs and combined as SDFTs.

Suppose that q is a composite number and has two divisors, q_1 and q_2. The two divisors are odd numbers and not co-prime with each other, e.g., $q = 3 \times 9$. For a length-$q = q_1 \times q_2$ sub-DFT defined in (1), assume that q be $n_0 q_1 + n_1$. The sub-DFT can be decomposed by using (6) and Cooley-Tukey algorithm [4]. The special case of $q = 3^l$ is described in Section V in detail.

To summarize, the proposed algorithm is presented for computing a length-$N = q \times 2^m$ DFT. This solution scheme falls into three stages. Firstly, the DFT in (1) is divided into q length-N/q sub-DFTs, and by changing the order of inner and outer sub-transformations the scaled sub-DFTs of length-2^m are obtained. Then, the scaled sub-DFTs are evaluated with the algorithm which is based on the SR28FFT algorithm and is presented in the next subsection. Finally, N/q sets of q results with the same index in different sub-DFTs are combined into the final outputs. The first decomposition stage and the final assemblage stage correspond to "pre-weave" and "post-weave" of q-points DFTs respectively.

III. Computation of SDFTs

In order to efficiently compute a DFT of length-$N = 2^m$, two algorithms was presented by Zheng *et al.* [14] for computing powers-of-two sub-DFTs with rotating factors. The two algorithms are based on radix-2/8 FFT, and named scaled radix-2/8 FFT-1 (SR28FFT-1) and SR28FFT-2. As an auxiliary algorithm, the goal of SR28FFT-1 is to assist other algorithms with more high comprehensive properties. The algorithms which use SR28FFT-1 to compute their sub-DFTs, can optimize computational complexity, computation accuracy, and coefficient evaluations. SR28FFT-2 is also an auxiliary algorithm. However, SR28FFT-2 neglects the improvement of other performances, and focuses on reducing the number of operations. Based on SR28FFT-1 and SR28FFT-2, two algorithms in [14] were proposed for efficiently computing a length-$N = 2^m$ DFT. While the two algorithms use SR28FFT-2 for computing their sub-DFTs with rotating factors, and whose arithmetic complexity are less than that of MSRFFT [13].

The most basic method for all FFT algorithms is "divide-and-conquer" approach [27]. The idea of "divide-and-conquer" approach is to map the original DFT into sub-DFTs to save the cost of evaluating the original DFT. This map can also be applied to sub-DFTs, thus obtaining a reduction of order of magnitude of computational complexity. The fundamental difference among algorithms lies in the ways of decomposition in which DFT is divided into smaller sub-DFTs. Another efficient method

for all FFT algorithms is the scaled DFT (SDFT). The arithmetic complexity issue has been well studied [28]. Only by varying decomposition, it is difficult to further reduce arithmetic complexity. The further improvement in arithmetic complexity requires efficient techniques like SDFT. However, SDFT is often used only as a trick in many algorithms [16], [17], [21], and its influence on the complexity does not gain enough attention. In reality, SDFT plays an important pole in some algorithms. For modified SRFFT (MSRFFT) presented in [13] and scaled radix-2/8 FFT (SR28FFT) presented in [14], the realization of the lower operation count relies on the applications of SDFT rather than recursive modifications.

We now consider using radix-2/4 FFT to compute SDFTs, and using SR28FFT-2 to compute the sub-DFTs generated by using the decomposition of radix-2/4 FFT in the computation of SDFTs. Assume that the constant factor of SDFTs is r, and a SDFT of length-$N = 2^m$ is defined as

$$X_k = r \sum_{n=0}^{N-1} x_n w_N^{nk}, \quad k = 0, 1, \ldots, N-1. \tag{29}$$

The SDFT in (29), by performing the butterflies of radix-2/4 FFT, is decomposed into a length $N/2$ sub-SDFT:

$$X(2k) = r \sum_{n=0}^{N/2-1} u(n) w_{N/2}^{nk}, \; k = 0, 1, \ldots, (N/2-1), \tag{30}$$

for the even-indexed terms, and two length-$N/4$ sub-DFTs:

$$X(4k+1) = \sum_{n=0}^{N/4-1} a_e(n) w_N^n r\varsigma(N/4, n) w_{N/4}^{nk}, \quad k = 0, 1, \ldots, (N/4-1), \tag{31}$$

and

$$X(4k+3) = \sum_{n=0}^{N/4-1} a_o(n) w_N^{3n} r\varsigma(N/4, n) w_{N/4}^{nk}, \quad k = 0, 1, \ldots, (N/4-1), \tag{32}$$

for the odd-indexed terms. The factor $\varsigma(N/4, n)$ in (31) and (32) is defined as

$$\varsigma(N, n) = s_1(N, n) s_2(N, n), \quad n = 0, 1, \ldots, N-1, \tag{33}$$

where

$$s_1(N, n) = \begin{cases} 1, & N \le 4, \\ s_1(N/8, n), & 0 < (n|N/4) < N/8, \\ s_1(N/8, n) w_8^1, & N/8 \le (n|N/4), \end{cases} \tag{34}$$

and

$$s_2(N, n) = \begin{cases} 1, & N \le 8, \\ s_2(N/8, n) \cos\left(2\pi (n|N/8)/N\right), & \text{otherwise.} \end{cases} \tag{35}$$

The sequence $u(n)$ in (30), $a_e(n)$ in (31), and $a_o(n)$ in (32) can be expressed in a matrix form

$$\begin{bmatrix} u(n) \\ v(n) \end{bmatrix} = \mathbf{H_2} \begin{bmatrix} x(n) \\ x(N/2+n) \end{bmatrix}, \tag{36}$$

and

$$\begin{bmatrix} a_e(n) \\ a_o(n) \end{bmatrix} = \mathbf{H_2} \begin{bmatrix} v(n) \\ -jv(N/2+n) \end{bmatrix}. \tag{37}$$

The sub-DFT in (30) will be decomposed through Eqs. (30)–(32). The sub-DFTs in (31) and (32) will be decomposed by using the SR28FFT-2 algorithm.

We now summarize the proposed scheme for computing length-$N = 2^m$ SDFTs. The SDFT is decomposed into a length-$N/2$ sub-SDFT and two length-$N/4$ sub-DFTs. The length-$N/2$ sub-SDFT is decomposed recursively until the size is reduced to a 2-points DFT or a 1-point DFT. The two length-$N/4$ sub-DFTs are implemented with the SR28FFT algorithm. In the proposed algorithm, to optimize arithmetic complexity, the SR28FFT-2 algorithm is used for computing the length-$N/4$ sub-DFTs. However, to obtain high comprehensive performance, the SR28FFT-1 algorithm is a choice.

IV. Performance Analysis

In this section, we consider the performance of the proposed algorithm by analyzing its computational complexity and comparing it with the related algorithms reported in [17], [21], [22]. The analysis and comparison of the computation complexity are focused on the number of floating-point operations required. We assume that a common complex multiplication requires four real multiplications and two real additions, and a complex multiplication by 2, 4, 1/2, 3/2, 5/4, or 15/8 requires two real multiplications.

A. Computational Complexity

Let M_N and A_N be the number of real multiplications and the number of additions required for computing a length-N DFT respectively, M_N^S be the number of real multiplications for calculating a SDFT of length-N with the algorithm presented in Section III.

The computation of the proposed algorithm for a length-$N = q \times 2^m$ DFT consists of three stages: the first decomposition stage, the second stage of the computation of SDFTs, and the final assemblage stage. The arithmetic complexity of the decomposition and assemblage stages for computing a length-$N = 3 \times 2^m$ DFT can be computed from the signal flowgraph in Fig. 1, whereas that of a length-$N = 5 \times 2^m$ DFT can be obtained by analyzing the signal flowgraph in Fig. 2. The required arithmetic complexity for computing a length-$N = 7 \times 2^m$ DFT can be calculated through Fig. 3. The required operation count for computing a length-$N = 9 \times 2^m$ DFT can be obtained through Fig. 4. Finally, the operations count of a length-$N = 15 \times 2^m$ DFT can be evaluated from Fig. 5. These arithmetic complexity are given in Table I.

The length-$N = 2^m$ SDFT are evaluated with the algorithm presented in Section III. From the analysis of [14], we know that the algorithm presented in Section III requires the same real additions as that of radix-2/4 FFT. In other words, for computing a SDFT of length-$N = 2^m$, the algorithm presented in Section III requires

$$A_N^S = 8/3Nm - 16/9N + 2 - (-1)^m 2/9. \tag{38}$$

TABLE I
ARITHMETIC COMPLEXITY IN THE DECOMPOSITION STAGE AND THE ASSEMBLAGE STAGE

N	One time			N/q times		
	M_q	A_q	Total	M_N	A_N	Total
3×2^m	0	12	12	0	$4N$	$4N$
5×2^m	4	32	36	$4N/5$	$32N/5$	$36N/5$
7×2^m	8	68	76	$8N/7$	$68N/7$	$76N/7$
9×2^m	10	80	90	$10N/9$	$80N/9$	$10N$
15×2^m	12	156	168	$4N/5$	$32N/5$	$36N/5$

real additions. As compared with the NR24FFT algorithm presented in [14] (while the algorithm uses SR28FFT-2 for its sub-DFTs with rotating factors), the algorithm presented in Section III requires $4m$ extra real multiplications for computing a length-$N = 2^m$ SDFT, where $m \neq 0$ and the scaling factor is not equal to 1. It is clear that the number of real multiplications required by the algorithm in Section III can be expressed as follows:

$$\begin{aligned} M_N^S =& 1.083485821Nm - 2.578858356N \\ & - 0.00000540457805(-1)^m N \\ & - 0.000132812(-1)^j N \\ & - 0.000637938(-1)^k N + 0.000188196(-1)^l N \\ & + 1.024357683m + 0.814546249(-1)^m m \\ & - 0.47534031(-1)^j m + 0.704109766(-1)^k m \\ & + 0.998929971(-1)^l m + 5.567285901 \\ & - 7.279909886(-1)^m + 3.31879978(-1)^j \\ & - 1.447097859(-1)^k - 4.616173247(-1)^l, \\ & \qquad for\ \ N \geq 8, \end{aligned} \quad (39)$$

where $m = \log_2^N$, $j = ((m|3) == 0)$, $k = (((m+3)|4) == 0)$, and $l = (((m+3)|5) == 0)$. (Note: the integer value of the boolean value "true" is 1; the integer value of the boolean value "false" is 0).

Overall, a DFT of length-$N = q \times 2^m$ is implemented with the proposed algorithm, requiring

$$M_N = N/qM_q + qM_{N/q}^S - 4m, \quad (40)$$

real multiplications, and

$$\begin{aligned} A_N =& N/qA_q + qA_{N/q}^S \\ =& \frac{8}{3}Nm - \frac{16}{9}N + 2q - \frac{2}{9}q(-1)^m + N/qA_q, \end{aligned} \quad (41)$$

real additions, where the values of N/qM_q and N/qA_q can be obtained from Table I.

In order to carry out a complete comparison of the arithmetic complexity of the proposed algorithm with that of the existing algorithms, Figs. 6–9 give four comparisons in terms of the ratio of computational complexity to $N \log_2^N$. The algorithms compared contain SRFFT, MSRFFT, the algorithms reported in [17], [21], and our previous work [22]. It can be seen that, for q=3, 5, 9, and 15, the savings of arithmetic operations over the algorithms in [17], [21], [22] are achieved by the proposed algorithm, and more computationally efficiency is also obtained over SRFFT and MSRFFT. The reduction of arithmetic operations

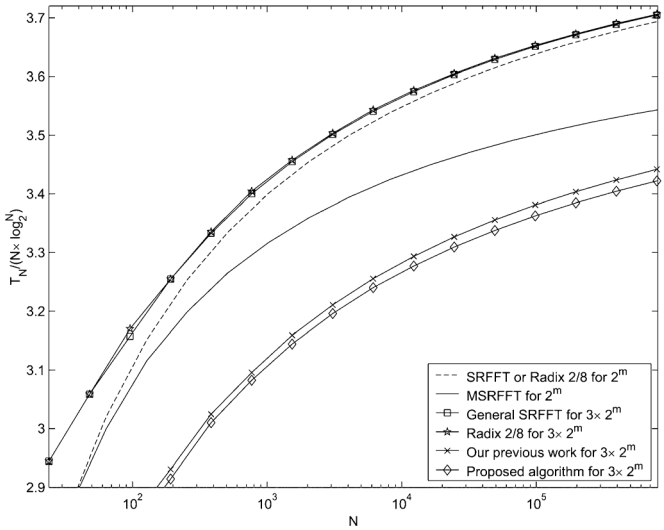

Fig. 6. Ratio of computational complexity for computing length-$N = 3 \times 2^m$ to $N \log_2^N$.

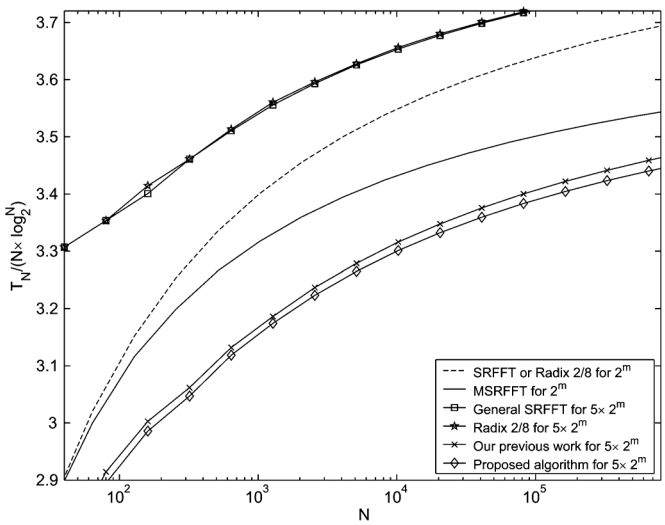

Fig. 7. Ratio of computational complexity for computing length-$N = 5 \times 2^m$ to $N \log_2^N$.

over our previous work [22] derives from the use of SR28FFT-2.

As compared with our another work in [29], for computing a DFT of length-$N = 3 \times 2^m$ or -$N = 9 \times 2^m$, when its length is smaller, the proposed algorithm requires the same number of arithmetic operations as that in [29]. However, when its length is longer, the proposed algorithm requires less arithmetic operations than that in [29].

B. An Alternative With High Comprehensive Performance

Of course, in order to obtain high comprehensive properties, and optimize arithmetic operations, coefficient evaluations, computation accuracy, we can use the extracting factor $s(N/4, n)$ to replace the factor $\varsigma(N/4, n)$ in (31) and (32), and then use SR28FFT-1 to compute the length-$N/4$ sub-DFTs in (31) and (32). The new extracting factor $s(N, n)$ is defined as

$$s(N,n) = \begin{cases} 1, & 0 < (n|N/4) < N/8; \\ w_8^1, & N/8 \leq (n|N/4). \end{cases} \quad (42)$$

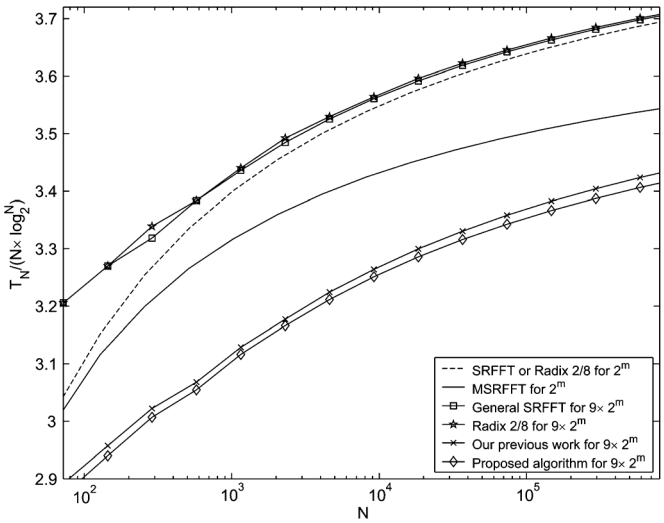

Fig. 8. Ratio of computational complexity for computing length-$N = 9 \times 2^m$ to $N \log_2^N$.

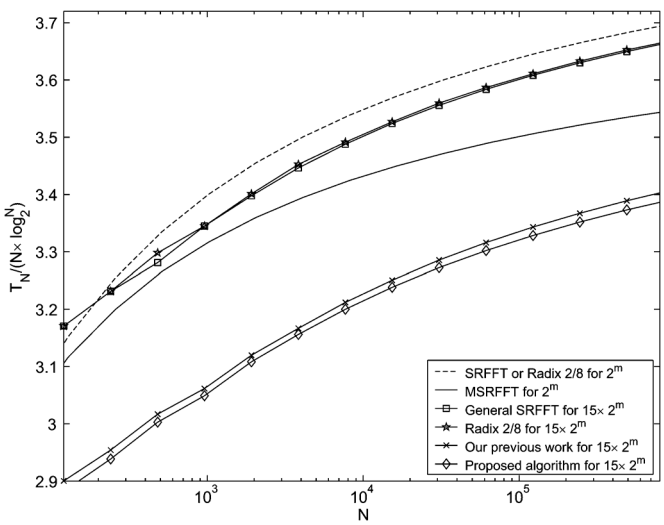

Fig. 9. Ratio of computational complexity for computing length-$N = 15 \times 2^m$ to $N \log_2^N$.

V. Extension to 6^m DFTs

Ref. [25] provided approaches for 3- and 9-points DFTs, but did not provide a scheme for 3^l DFTs, where l is an integer greater than 2. In this section, the proposed algorithm will be extended to 6^m DFTs. Assume that a length-$N = 6^m$ DFT is defined in (1). The DFT in (1) is divided as

$$\begin{aligned} X_{k_1 N/3+k_0} = & \sum_{n=0}^{N/3-1} x_{3n} w_{N/3}^{nk_0} \\ & + w_N^{k_1 N/3+k_0} \sum_{n=0}^{N/3-1} x_{3n+1} w_{N/3}^{nk_0} \\ & + w_N^{-(k_1 N/3+k_0)} \sum_{n=0}^{N/3-1} x_{3n-1} w_{N/3}^{nk_0}, \\ & k_0 = 0, 1, \ldots, N/3-1, \\ & k_1 = 0, 1, 2, \end{aligned} \tag{43}$$

TABLE II
Computational Complexity for Length-$N = 3^m$ DFTs

N	M_N	A_N	Total
3	0	12	16
9	10	80	90
27	98	380	478
81	506	1568	2074
243	2162	5996	8158
729	8426	21872	30298
2187	31106	77276	108382

by Suzuki's radix-3 FFT algorithm. The sub-DFTs of length-$N/3$ will be decomposed recursively through (43) until the sizes are reached to length-9×2^m DFTs. The resulting length-9×2^m sub-DFTs will be evaluated with the proposed algorithm in Section II.

The decomposition in (43) is derived from the Suzuki's radix-3 FFT algorithm [20]. However, two little modifications are made in this section. First, the decomposition is applied to powers-of-six DFTs. Second, by comparing their computational complexity, one know that Wang's algorithm [25] implements a 9-points DFT with more efficiency than Suzuki's radix-3 FFT algorithm [20]. Thus, the sub-DFTs of length-9×2^m, which should have been recursively decomposed until the sizes are reached to 3×2^m, are evaluated with the proposed algorithm presented in Section II.

Through a similar modification, other algorithms, that are developed for powers-of-three DFTs, can also be used to implement a length-6^m DFT. For example, the radix-3/9 FFT in [16] can be modified to recursively decompose a DFT of length-6^m until the sizes of the resulting sub-DFTs are all reached to 9×2^m and 3×2^m. The sub-DFTs of length-9×2^m and -3×2^m are evaluated with the proposed algorithm in Section II.

We now consider the computational complexity of the algorithm extended to 6^m DFTs. The algorithm extended to the length-$N = 6^m$ DFT is a mixture of the radix-3 FFT and the radix-9 FFT. The extended mixed-radix FFT recursively decomposes the DFT into three length-$N/3$ sub-DFTs by performing one special radix-3 FFT butterfly of $k_0 = 0$ and $N/3-1$ general radix-3 butterflies till the size of the sub-DFTs reaches 9×2^m. The special butterfly requires 4 real multiplications and 12 real additions. Each general radix-3 butterfly requires 8 real multiplications and 16 real additions. Hence, the proposed mixed-radix FFT requires

$$\begin{cases} A_N = 3 \times A_{N/3} + (N/3-1) \times 8 + 4, \\ M_N = 3 \times M_{N/3} + (N/3-1) \times 16 + 12, \end{cases} \tag{44}$$

for computing the length-$N = 6^m$ DFT. Table II shows the computational complexity of the extended algorithm for length-$N = 3^m$ DFTs, where $M_3 = 0$, $A_3 = 12$, $M_9 = 10$, and $A_9 = 80$. The computational complexity can be used in (40) and (41) to compute the computational complexity of the extended algorithm for the length-6^m DFTs.

Replacing $M_3 = 0$ and $M_9 = 10$ by $M_3 = 4$ and $M_9 = 26$, we can obtain the computational complexity of the length-3^m DFTs under general condition in Table III. From the table, we can see that extended mixed-radix FFT reduces slightly the

TABLE III
COMPUTATIONAL COMPLEXITY FOR LENGTH-$N = 3^m$ DFTS

N	Radix-3 FFT in [20]			Radix-3/9 FFT in [16]			Proposed Mixed-Radix		
	M_N	A_N	Total	M_N	A_N	Total	M_N	A_N	Total
3	4	12	16	4	12	16	4	12	16
9	32	80	112	24	84	108	26	80	106
27	164	380	544	156	396	552	146	380	526
81	704	1568	2272	660	1640	2300	650	1568	2218
243	2756	5996	8752	2750	6268	9018	2594	5996	8590
729	10208	21872	32080	10264	22900	33164	9722	21872	31594
2187	36452	77276	113728	37440	80908	118348	34994	77276	112270

TABLE IV
COMPUTATIONAL COMPLEXITY FOR 6^m DFTS

N	Our previous work in [29]			Proposed Algorithm					
				Using Modified Suzuki's Radix-3 FFT			Using Radix-3/9 FFT in [16]		
	M_N	A_N	Total	M_N	A_N	Total	M_N	A_N	Total
6	4	36	40	4	36	40	8	36	44
36	96	464	560	80	464	544	80	464	544
216	1132	4364	5496	1044	4444	5488	1060	4444	5504
1296	10488	35824	46312	10488	36752	47240	9656	36752	46408
7776	85932	273164	359096	90276	282268	372544	83812	282268	366080
46656	656648	1989472	2646120	710400	2064656	2775056	646400	2064656	2711056
279936	4799036	14043932	18842968	5302644	14623996	19926640	4832884	14623996	19456880
1679616	34197016	96922192	131119208	38077896	101162384	139240280	34482632	101162384	135645016

number of operations as compared to the radix-3 FFT [20] and radix-3/9 FFT [16].

Table IV shows the computational complexity of the proposed algorithm and our previous work in [29]. In order to compare them under the same conditions, we assume that a complex multiplication by 2, 1/2, or 3/2 does not require any real operations. From this table, one can see that, although two slight modifications of the proposed algorithm have been made for 6^m DFTs, the proposed algorithm requires more real operations than the algorithm in [29] for computing 6^m DFTs (except for length-36 and length-216). The situation when the modified Sukuzi's radix-3 FFT is replaced by the radix-3/9 FFT algorithm in [16] is similar to that of the modified Sukuzi's radix-3 FFT.

VI. AN EXAMPLE

We now take the length-$N = 5 \times 2^m$ DFT as an example. The length-$N = 5 \times 2^m$ DFT is defined in (1). Its matrix definition is given in (2). At the beginning of computation, the input sequence x_n in (2) should be transformed into the two-dimension matrix representation. Let index n in (2) be

$$n = n_0 5 + n_1 N/5, \tag{45}$$

where $n_0 \in [0, N/5)$ and $n_1 \in [0, 5)$. Let y_{n_0,n_1} represent $x_{5n_0+n_1N/5}$. The two-dimension vector y is the inputs of the DFT, and is defined as

$$y = \begin{bmatrix} x_0 & x_{N/5} & \cdots & x_{N-N/5} \\ x_5 & x_{5+N/5} & \cdots & x_{5+N-N/5} \\ \cdots & \cdots & \cdots & \cdots \\ x_{N-5} & x_{N-5+N/5} & \cdots & x_{2N-5-N/5} \end{bmatrix}. \tag{46}$$

Let 5 and $N/5$ be two decomposition factors of k, such that $k = 5k_0 + k_1 N/5$. Now, one can provide the expression of the DFT in (1) in the two-dimension form,

$$X_k = \sum_{n_1=0}^{4} \sum_{n_0=0}^{N/5-1} y_{n_0,n_1} w_{N/5}^{n_0(5k_0)} w_5^{n_1(N/5k_1)}, \tag{47}$$

for (1). The coefficient matrix $\mathbf{F_5}$ can be factorized into a product of three matrices, $\mathbf{L_5}$, $\mathbf{M_5}$, and $\mathbf{R_5}$, i.e., $\mathbf{F_5} = \mathbf{L_5}\mathbf{M_5}\mathbf{R_5}$. So, (47) can be expressed as

$$X = \mathbf{L_5}\mathbf{M_5}\left(\mathbf{F}_{N/5}(\mathbf{R_5}y^T)^T\right)^T, \tag{48}$$

where $\mathbf{L_5}$, $\mathbf{M_5}$, $\mathbf{R_5}$ are defined in (11)–(15) in Section II. According to (48), one can compute easily length-$N-5\times 2^m$ DFT. Specifically, $\mathbf{R_5}$ is first multiplied by the input sequence y^T. The sequence of length-N y^T is decomposed into 5 length-2^m sub-sequences. $\mathbf{M_5}(\mathbf{F}_{N/5}(\mathbf{R_5}y^T)^T)^T$ are then computed with scaled MSRFFT algorithm. Finally, $\mathbf{L_5}$ multiply the results of the SDFTs, and the final outputs are obtained.

VII. THE COMPUTATION OF IDFT

The inverse DFT (IDFT) has nearly the same definition as DFT, except for the normalizing factor $1/N$ and conjugate coefficients. An IDFT can be implemented with a forward FFT algorithm followed by a permutation, and can also be implemented with a simple forward three steps algorithm [30]. The advantage of the forward algorithms is that there is no requirement to rewrite a program for IDFT. The calculations involved in normalizing factor are often ignored in the operation count of IDFT. In fact, the calculations of normalization of IDFT require $2N$ extra multiplications for computing a size-N IDFT, which needs to cost quite a lot processing time.

Generally speaking, the computation of IDFT requires $2N$ more real multiplications than that of DFT. In the proposed al-

gorithm, IDFT requires only $4q\log_2^N$ extra real multiplications. The IDFT of (1) can be defined by

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k w_N^{-nk}. \tag{49}$$

Except for the factor $1/N$ and coefficient w_N^{-nk}, the above definition is the same as that of DFT. An IDFT can be viewed as a SDFT with normalizing factor $1/N$. Because we are often interested only the input/output sequence, the factor $1/N$ of IDFT is ignored. Sometimes $1/N$ is distributed between DFT and IDFT to increase the symmetry, i.e.,

$$\begin{cases} X_k = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} x_n w_N^{nk} & \text{(50a)} \\ x_n = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} X_k w_N^{-nk}. & \text{(50b)} \end{cases}$$

Like most of applications, we adopt (49) as the definition of IDFT in the following discussions (for the definition in (50b), the only difference is the normalizing factors). For forward computation, (49) can be expressed in matrix form as follows [30],

$$x = j\left[\frac{1}{N}\mathbf{F}_N(jX^*)\right]^*. \tag{51}$$

Eq. (51) can be evaluated by three steps. (1) Exchange the real and imaginary parts of X_k. 2) Perform a forward DFT computation. 3) Exchange the real and imaginary parts of the results of the second step.

In the first and third steps, the exchanges do not require any arithmetic operations. In the second step, the proposed algorithm is implemented for computing a length-$N = q \times 2^m$ IDFT. The forward computation of IDFT is the same as that of DFT, except for the scaling factors which is equal to the value of the original scaling factors of DFT divided by N. Hence, the number of operations of scaled MSRFFT for computing IDFT are identical to that for computing DFT if the scaling factor of the original DFT is not equal to 1. The algorithm presented in Section III for computing an IDFT requires $4m$ extra real multiplications when the scaling factor of the original DFT is equal to 1. Compared with the arithmetic complexity given in (40) and (41), the arithmetic complexity of the proposed algorithm for computing a length-$N = q \times 2^m$ IDFT only requires $4m$ extra real multiplications.

VIII. The Real-Data DFT

The transformation on real-data DFT obeys the symmetry rule: $X_{N-k} = X_k^*$, where X_0 and $X_{N/2}$ are real numbers. These rules will be obeyed by subsequent sub-transforms.

As far as the proposed algorithm are concerned, the decomposing stage contains additions and subtracts but not complex multiplications. Hence, the results are all real data. For the subsequent transform of SDFTs, the results obey the symmetry rule mentioned above as well. Thus, the decomposing stage requires half the number of operations required on complex data. For a length-$N = 2^m$ real data scaled/unscaled DFT, by eliminating redundant operations, the algorithm presented in Section III requires half of real operations required for complex data signals minus $N - 2$.

Except for the two special cases of $n = 0$ and $n = N/2q$, the assembling stage requires half the number of operations required on complex data by eliminating the redundant calculations. However, for each of the two special cases of $n = 0$ and $n = N/2q$, the q signals are all the outputs of the computation stage of SDFTs. There are $(q-1)/2$ signals which are real data. There are $(q-1)/2$ signals which are all imaginary data. The numbers of the required additions are half of those for complex signals minus $q - 1$, since the addition of a real and an imaginary does not require any real operation.

Overall, when the proposed algorithm computes a DFT of length-$N = q \times 2^m$ on real-data, the number of real additions is half of that required for complex signals minus $N - 2$, and the number of real multiplications required is half of that of complex signals.

IX. Conclusions

This paper presented a new algorithm for computing the DFT of length-$N = q \times 2^m$. The algorithm is an extension of the SR28FFT algorithm. The substantial reduction of arithmetic complexity can be obtained over the algorithms in [17], [21], [22]. Higher computational efficiency is also achieved over SRFFT and MSRFFT algorithms. Especially for computing a length-$N = q \times 2^m$ IDFT, only $4m$ extra real multiplications are required, i.e., the reduction of operations reaches 10%. In addition, the algorithm is extended to 6^m DFTS, and is applied to real-data FFT algorithm. Particularly, the length-2^m sub-DFTs can be evaluated with other algorithms, such as radix-2^k FFT and radix-$2/2^k$ FFT [31] etc., for more regularity and less accesses to lookup table. Considering the actual hardware design, the structure of the proposed algorithm, that length-2^m sub-DFTs are placed in the intermediate parts of size q DFTs combining length-2^m SDFTs, has smaller quantization error than the related algorithms.

Acknowledgment

We are thankful to the anonymous reviewers for their constructive comments, which have helped us to improve the quality of the manuscript.

References

[1] M. Frigo and S. Johnson, "The design and implementation of FFTW3," *Proc. IEEE*, vol. 93, no. 2, pp. 216–231, 2005.
[2] K. Katoh, K. Kuma, H. Toh, and T. Miyata, "MAFFT version 5: Improvement in accuracy of multiple sequence alignment," *Nucleic Acids Res.*, vol. 33, no. 2, pp. 511–518, 2005.
[3] Y. Sheng, T. Yapo, and B. Cutler, "Global illumination compensation for spatially augmented reality," *Comput. Graph. Forum*, vol. 29, no. 2, pp. 387–396, 2010, Wiley Online Library.
[4] J. Cooley and J. Tukey, "An algorithm for the machine calculation of complex fourier series," *Math. Comput.*, vol. 19, no. 90, pp. 297–301, 1965.
[5] P. Duhamel and H. Hollmann, "Split-radix FFT algorithm," *Electron. Lett.*, vol. 20, pp. 14–16, Jan. 1984.
[6] G. Bi, G. Li, and X. Li, "A unified expression for split-radix DFT algorithms," in *Proc. IEEE Int. Conf. Commun., Circuits, Syst. (ICCCAS)*, 2010, pp. 323–326.

[7] I. Kamar and Y. Elcherif, "Conjugate pair fast fourier transform," *Electron. Lett.*, vol. 25, no. 5, pp. 324–325, Apr. 1989.
[8] D. Takahashi, "An extended split-radix FFT algorithm," *IEEE Signal Process. Lett.*, vol. 8, no. 5, pp. 145–147, 2001.
[9] R. Yavne, "An economical method for calculating the discrete Fourier transform," in *Proc. AFIPS Fall Joint Comput. Conf.*, 1968, vol. 33, pp. 115–125.
[10] M. Vetterli and H. J. Nussbaumer, "Simple FFT and dct algorithms with reduced number of operations," *Signal Process.*, vol. 6, no. 4, pp. 267–278, 1984.
[11] J. Martens, "Recursive cyclotomic factorization-a new algorithm for calculating the discrete fourier transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 4, pp. 750–761, 1984.
[12] A. Grigoryan and S. Agaian, "Split manageable efficient algorithm for fourier and hadamard transforms," *IEEE Trans. Signal Process.*, vol. 48, no. 1, pp. 172–183, 2000.
[13] S. G. Johnson and M. Frigo, "A modified split-radix FFT with fewer arithmetic operations," *IEEE Trans. Signal Process.*, vol. 55, no. 1, pp. 111–119, 2007.
[14] W. Zheng, K. Li, and K. Li, "Scaled radix-2/8 algorithm for efficient computation of length- $N = 2^m$ DFTs," *IEEE Trans. Signal Process.*, vol. 62, pp. 2492–2503, May 2014.
[15] A. M. Grigoryan and S. S. Agaian, *Multidimensional Discrete Unitary Transforms: Representation: Partitioning, Algorithms*. Boca Raton, FL, USA: CRC Press, 2003.
[16] M. Vetterli and P. Duhamel, "Split-radix algorithms for length- p^m dft's," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 1, pp. 57–64, 1989.
[17] G. Bi and Y. Chen, "Fast DFT algorithms for length $n = q \times 2^m$," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 45, no. 6, pp. 685–690, 1998.
[18] R. Singleton, "An algorithm for computing the mixed radix fast Fourier transform," *IEEE Trans. Audio Electroacoust.*, vol. 17, no. 2, pp. 93–103, 1969.
[19] D. Kolba and T. Parks, "A prime factor FFT algorithm using high-speed convolution," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 25, no. 4, pp. 281–294, 1977.
[20] Y. Suzuki, T. Sone, and K. Kido, "A new FFT algorithm of radix 3, 6, 12," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-34, no. 2, pp. 380–383, Apr. 1986.
[21] S. Bouguezel, M. Ahmad, and M. Swamy, "A new radix-2/8 FFT algorithm for length- $q \times 2^m$ DFTs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 9, pp. 1723–1732, Sep. 2004.
[22] W. Zheng, L. Kenli, and L. Keqin, "A fast algorithm based on SRFFT for length $n = q \times 2^m$ DFTs," *IEEE Trans. Circuits Syst. II*, vol. 61, pp. 110–114, Feb. 2014.
[23] H. Shu, X. Bao, C. Toumoulin, and L. Luo, "Radix-3 algorithm for the fast computation of forward and inverse MDCT," *IEEE Signal Process. Lett.*, vol. 14, no. 2, pp. 93–96, 2007.
[24] C. Hsiao, Y. Chen, and C. Lee, "A generalized mixed-radix algorithm for memory-based FFT processors," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 57, no. 1, pp. 26–30, 2010.
[25] Z. Wang, "A prime factor fast W transform algorithm," *IEEE Trans. Signal Process.*, vol. 40, no. 9, pp. 2361–2368, 1992.
[26] M. Macleod, "Multiplierless Winograd and prime factor FFT implementation," *IEEE Signal Process. Lett.*, vol. 11, no. 9, pp. 740–743, 2004.
[27] I. Good, "The interaction algorithm and practical Fourier analysis," *J. Roy. Statist. Soc. Series B (Methodol.)*, pp. 361–372, 1958.
[28] P. Duhamel and M. Vetterli, "Fast fourier transforms: A tutorial review and a state of the art," *Signal Process.*, vol. 19, no. 4, pp. 259–299, 1990.
[29] W. Zheng and K. Li, "Split radix algorithm for length 6^m DFT," *IEEE Signal Process. Lett.*, vol. 20, pp. 713–716, Jul. 2013.
[30] P. Duhamel, B. Piron, and J. Etcheto, "On computing the inverse DFT," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 36, no. 2, pp. 285–286, 1988.
[31] S. Bouguezel, M. Ahmad, and M. Swamy, "A general class of split-radix FFT algorithms for the computation of the DFT of length-2^m," *IEEE Trans. Signal Process.*, vol. 55, no. 8, pp. 4127–4138, Aug. 2007.

Kenli Li (M'14) received the Ph.D. degree in computer science from Huazhong University of Science and Technology, China, in 2003. He was a visiting scholar at University of Illinois at Urbana-Champaign from 2004 to 2005. He is currently a full professor of computer science and technology at Hunan University and associate director of National Supercomputing Center in Changsha. His major research includes parallel computing, grid and cloud computing, and DNA computing. He has published more than 110 papers in international conferences and journals, such as IEEE-TC, IEEE-TPDS, JPDC, ICPP, CCGrid. He is an outstanding member of CCF.

Weihua Zheng received his Master's degree from the National University of Defense Technology, China, in 2010. He is currently working towards the Ph.D. degree at Hunan University of China. His research interests include fast Fourier transform, multiscale analysis, audio signal processing, image processing, parallel computing, natural language understanding, machine learning, knowledge representation, artificial intelligence. He has published some papers in IEEE TRANSACTIONS ON SIGNAL PROCESSING, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEM II, *Signal Processing*, and IEEE SIGNAL PROCESSING LETTERS.

Keqin Li (F'15) is a SUNY Distinguished Professor of computer science. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things, and cyber-physical systems. He has published over 320 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently or has served on the editorial boards of TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CLOUD COMPUTING, and *Journal of Parallel and Distributed Computing*.