Composite Cyclotomic Fourier Transforms With Reduced Complexities

Xuebin Wu, Meghanad Wagh, Ning Chen, *Member, IEEE*, Ying Wang, *Member, IEEE*, and Zhiyuan Yan, *Senior Member, IEEE*

***Abstract*—Discrete Fourier transforms (DFTs) over finite fields have widespread applications in digital communication and storage systems. Hence, reducing the computational complexities of DFTs is of great significance. Recently proposed cyclotomic fast Fourier transforms (CFFTs) are promising due to their low multiplicative complexities. Unfortunately, there are two issues with CFFTs: (1) they rely on efficient short cyclic convolution algorithms, which have not been sufficiently investigated in the literature and (2) they have very high additive complexities when directly implemented. To address both issues, we make three main contributions in this paper. First, for any odd prime p, we reformulate a p-point cyclic convolution as the product of a $(p-1)\times(p-1)$ Toeplitz matrix and a vector, which has well-known efficient algorithms, leading to efficient bilinear algorithms for p-point cyclic convolutions. Second, to address the high additive complexities of CFFTs, we propose composite cyclotomic Fourier transforms (CCFTs). In comparison to previously proposed fast Fourier transforms, our CCFTs achieve lower overall complexities for moderate to long lengths and the improvement significantly increases as the length grows. Third, our efficient algorithms for p-point cyclic convolutions and CCFTs allow us to obtain longer DFTs over larger fields, e.g., the 2047-point DFT over $\mathrm{GF}(2^{11})$ and 4095-point DFT over $\mathrm{GF}(2^{12})$, which are first efficient DFTs of such lengths to the best of our knowledge. Finally, our CCFTs are also advantageous for hardware implementations due to their modular structure.**

***Index Terms*—Cooley–Tukey algorithm, cyclotomic fast Fourier transforms, discrete Fourier transforms, finite fields, prime-factor algorithm.**

I. Introduction

DISCRETE Fourier transforms (DFTs) over finite fields [1] have widespread applications in error correction coding, which in turn is used in all digital communication and storage systems. For instance, both syndrome computation and Chien search in the syndrome based decoder of Reed–Solomon codes [2] and [3], a family of widely used error control codes, can be formulated as polynomial evaluations and hence can be implemented efficiently using DFTs over finite fields. Directly implementing an N-point DFT requires $O(N^2)$ multiplications and $O(N^2)$ additions and becomes costly when N is large. Hence, reducing the computational complexities of DFTs is of great significance. Recently, efficient long DFTs have become particularly important as increasingly longer error control codes are chosen for digital communication and storage systems. For example, Reed–Solomon codes over $\mathrm{GF}(2^{12})$ and with block length of several thousands are considered for hard drive [4] and tape storage [5] as well as optical communication systems [6] to achieve better error performance; the syndrome based decoder of such codes requires DFTs of lengths up to 4095 over $\mathrm{GF}(2^{12})$. In addition to complexity, a modular structure is desirable for efficient hardware implementations of DFTs.

In the literature, fast Fourier transforms (FFTs) based on the prime-factor algorithm [7] and the Cooley-Tukey algorithm [8] have been proposed for DFTs over the complex field. When FFTs based on the prime-factor algorithm are adapted to DFTs over finite fields [9], they still have high multiplicative complexities. In contrast, recently proposed cyclotomic FFTs (CFFTs) are promising since they have significantly lower multiplicative complexities [10] and [11]. However, CFFTs have two issues. First, they rely on efficient algorithms for short cyclic convolutions, which do not always exist. For instance, CFFTs over $\mathrm{GF}(2^{11})$ would require efficient algorithms for 11-point cyclic convolutions. Previous works (see, for example, [10]–[12]) have not investigated CFFTs over $\mathrm{GF}(2^{11})$ partially due to the lack of efficient 11-point cyclic convolutions in the literature. Second, CFFTs have very high additive complexities when directly implemented, which can be reduced by techniques such as the common subexpression elimination (CSE) algorithm (see, for example, [12]–[15]). In particular, the CSE algorithm in [12] is effective for reducing the additive complexities of CFFTs over $\mathrm{GF}(2^l)$ for $l \leq 10$. Although the CSE algorithm has a polynomial complexity [12, Sec.III-F], its time and memory requirements limit its effectiveness for long DFTs. Due to these two issues, CFFTs over $\mathrm{GF}(2^{11})$ and $\mathrm{GF}(2^{12})$ have not been investigated in the literature.

In this paper, we address both aforementioned issues. The main contributions of our paper are as follows.

- For an odd prime p, we reformulate a p-point cyclic convolution over characteristic-2 finite fields as the product of a $(p-1)\times(p-1)$ Toeplitz matrix and a vector. Since $p-1$ is composite, this product can be readily obtained by multi-dimensional technologies from well-known Toeplitz matrix vector products (TMVPs) of very small sizes [16]–[20]. In comparison to other ad hoc techniques based on TMVPs, our reformulation achieves lower multiplicative complexity, especially for small to moderate p

Manuscript received May 16, 2010; revised October 25, 2010; accepted December 27, 2010. Date of publication January 17, 2011; date of current version April 13, 2011. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Warren J. Gross. This work was supported in part by NSF under Grant ECCS-0925890 and in part by a grant from Thales Communications, Inc. The material in this paper was presented in part at the IEEE Workshop on Signal Processing Systems, Cupertino, CA, October 2010.

X. Wu, M. Wagh, and Z. Yan are with the Department of Electrical and Computer Engineering, Lehigh University, PA 18015 USA (e-mail: xuw207@lehigh.edu; mdw0@lehigh.edu; yan@lehigh.edu).

N. Chen was with the Department of Electrical and Computer Engineering, Lehigh University, Bethlehem, PA 18015 USA. He is now with the SandForce, Inc., Saratoga, CA 95070 USA (e-mail: nchen@sandforce.com).

Y. Wang is with the New Jersey Research Center of QUALCOMM, Inc., Bridgewater, NJ 08807 USA (e-mail: aywang11@gmail.com).

Digital Object Identifier 10.1109/TSP.2011.2106778

and leads to efficient bilinear algorithms for p-point cyclic convolutions over characteristic-2 finite fields. Our reformulation can be readily extended to the real and complex fields as well as more general finite fields. Furthermore, by multi-dimensional technologies, we can also obtain efficient algorithms for p^n-point cyclic convolutions. These algorithms are also keys to long CFFTs.

- Due to the high additive complexities of CFFTs, we propose composite cyclotomic Fourier transforms (CCFTs), which are generalization of CFFTs. When the length N of a DFT is factored, i.e., $N = N_1 \times N_2$, the CCFT uses N_1- and N_2-point CFFTs as sub-DFTs via the prime-factor and Cooley-Tukey algorithms. Thus, CFFTs are simply a special case of our CCFTs, corresponding to the trivial factorization, i.e., $N = 1 \times N$. This generalization reduces overall complexities in three ways. First, this divide-and-conquer strategy itself leads to lower complexities. Second, the moderate lengths of the sub-DFTs enable us to apply complexity-reducing techniques such as the CSE algorithm in [12] more effectively. Third, when the length N admits different factorizations, the one with the lowest complexity is selected. In the end, while an N-point CCFT may have a higher multiplicative complexity than an N-point CFFT, the former achieves a lower overall complexity for long DFTs because of its significantly lower additive complexity. Moreover, when N is composite, an N-point CCFT has a modular structure, which is suitable for efficient hardware implementations. Our CCFTs provide a systematic approach to designing long DFTs with low complexity.
- Our efficient algorithms for p-point cyclic convolutions and CCFTs allow us to obtain longer DFTs over larger fields. For example, we propose CFFTs over $\mathrm{GF}(2^{11})$, which are unavailable in the literature heretofore partially due to the lack of efficient 11-point cyclic convolution algorithms. Our 2047-point DFTs over $\mathrm{GF}(2^{11})$ and 4095-point DFTs over $\mathrm{GF}(2^{12})$ are also first efficient DFTs of such lengths to the best of our knowledge and they are promising for emerging communication systems.

Our work in this paper extends and improves previous works [10] and [12] on CFFTs over finite fields of characteristic-2 in several ways. First, previously proposed CFFTs focus on $(2^l - 1)$-point CFFTs over $\mathrm{GF}(2^l)$ for $l \leq 10$. In contrast, our CCFTs allow us to derive long DFTs with low complexity over larger fields. Our approach can be applied to any finite field, but we present CCFTs over $\mathrm{GF}(2^{11})$ and $\mathrm{GF}(2^{12})$ due to their significance in applications. Furthermore, our work investigates N-point CFFTs over $\mathrm{GF}(2^l)$ for any N that divides $2^l - 1$, i.e., $N|2^l - 1$. Second, our CCFTs achieve lower overall complexities than **all** previously proposed FFTs for moderate to long lengths and the improvement significantly increases as the length grows.

The rest of the paper is organized as follows. Section II briefly reviews the necessary background of this paper, such as the CFFT, the prime-factor algorithm, the Cooley-Tukey algorithm and the CSE algorithm. We propose an efficient bilinear algorithm for p-point cyclic convolutions over $\mathrm{GF}(2^l)$ in Section III. We then use an 11-point cyclic convolution algorithm to construct 2047-point CFFT over $\mathrm{GF}(2^{11})$ in Section V. We also propose our CCFTs and compare their complexities with previously proposed FFTs in Section V. Concluding remarks are provided in Section VI.

II. Background

A. Cyclotomic Fast Fourier Transforms

In this paper, we consider DFTs over finite fields $\mathrm{GF}(2^l)$. Let $\alpha \in \mathrm{GF}(2^l)$ be an element with order N, which implies that $N|2^l - 1$ (otherwise α does not exist). Given an N-dimensional column vector $\mathbf{f} = (f_0, f_1, \ldots, f_{N-1})^T$ over $\mathrm{GF}(2^l)$, the DFT of $\mathbf{f}$ is given by $\mathbf{F} = (F_0, F_1, \ldots, F_{N-1})^T$, where

$$F_j = \sum_{i=0}^{N-1} f_i \alpha^{ij}. \tag{1}$$

If we define $f(x) = \sum_{i=0}^{N-1} f_i x^i$, we have $F_j = f(\alpha^j)$. Directly computing the DFT requires $O(N^2)$ multiplications and $O(N^2)$ additions and is impractical for large Ns. Cyclotomic FFTs (CFFTs) [10] and [11] can reduce the multiplicative complexities greatly.

We first partition the integer set $\{0, 1, \ldots, N-1\}$ into m cyclotomic cosets modulo N with respect to $\mathrm{GF}(2)$ [3]: $C_{s_0}, C_{s_1}, \ldots, C_{s_{m-1}}$, where $C_{s_k} = \{2^0 s_k, 2^1 s_k, \ldots, 2^{m_k-1} s_k\} \pmod{N}$ and $s_k = 2^{m_k} s_k \pmod{N}$. A polynomial $L(x) = \sum_i l_i x^{2^i}$, where $l_i \in \mathrm{GF}(2^l)$, is called a *linearized polynomial* over $\mathrm{GF}(2^l)$, since it has a linear property $L(x+y) = L(x) + L(y)$ for $x, y \in \mathrm{GF}(2^l)$. With the help of cyclotomic cosets, $f(x)$ can be decomposed as a sum of linearized polynomials

$$f(x) = \sum_{k=0}^{m-1} L_k(x^{s_k}),$$

$$L_k(x) = \sum_{j=0}^{m_k-1} f_{s_k 2^j \bmod N} x^{2^j}.$$

Therefore, $F_j = \sum_{k=0}^{m-1} L_k(\alpha^{j s_k})$ and each $\alpha^{j s_k}$ lies in the subfield $\mathrm{GF}(2^{m_k}) \subseteq \mathrm{GF}(2^l)$.

Using a normal basis $\{\gamma_k^{2^0}, \gamma_k^{2^1}, \ldots, \gamma_k^{2^{m_k-1}}\}$ of $\mathrm{GF}(2^{m_k})$ over $\mathrm{GF}(2)$, $\alpha^{j s_k}$ can be expressed by $\sum_{i=0}^{m_k-1} a_{i,j,k} \gamma_k^{2^i}$, where $a_{i,j,k} \in \{0, 1\}$. By the linear property of $L_i(x)$'s, $F_j = \sum_{k=0}^{m-1} \sum_{i=0}^{m_k-1} a_{i,j,k} L_k(\gamma_k^{2^i})$. Written in the matrix form, the DFT of $\mathbf{f}$ is given by $\mathbf{F} = \mathbf{A}\mathbf{L}\mathbf{\Pi}\mathbf{f}$, where $\mathbf{A}$ is an $N \times N$ binary matrix constructed from the binary coefficients $a_{i,j,k}$, $\mathbf{\Pi}$ is an $N \times N$ permutation matrix, $\mathbf{L} = \mathrm{diag}(\mathbf{L}_0, \mathbf{L}_1, \ldots, \mathbf{L}_{m-1})$ is a block diagonal matrix, and $\mathbf{L}_k$'s are $m_k \times m_k$ square matrices. The permutation matrix $\mathbf{\Pi}$ reorders the vector $\mathbf{f}$ into $\mathbf{f}' = (\mathbf{f}_0'^T, \mathbf{f}_1'^T, \ldots, \mathbf{f}_{m-1}'^T)^T$ and $\mathbf{f}_k' = (f_{s_k 2^0 \bmod N}, f_{s_k 2^1 \bmod N}, \ldots, f_{s_k 2^{m_k-1} \bmod N})^T$.

Though the idea of cyclotomic decomposition dates back to [21], the normal basis representation is a key step in reducing the multiplicative complexity of DFTs [10]. Since $\gamma_k^{2^{m_k}} = \gamma_k$, the kth block $\mathbf{L}_k$ of $\mathbf{L}$ is actually a circulant matrix, which is given by

$$\mathbf{L}_k = \begin{bmatrix} \gamma_k^{2^0} & \gamma_k^{2^1} & \cdots & \gamma_k^{2^{m_k-1}} \\ \gamma_k^{2^1} & \gamma_k^{2^2} & \cdots & \gamma_k^{2^0} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_k^{2^{m_k-1}} & \gamma_k^{2^0} & \cdots & \gamma_k^{2^{m_k-2}} \end{bmatrix}.$$

Hence, the multiplication between $\mathbf{L}_k$ and $\mathbf{f}'_k$ can be formulated as an m_k-point cyclic convolution between $\mathbf{b}_k = (\gamma_k^{2^0}, \gamma_k^{2^{m_k-1}}, \gamma_k^{2^{m_k-2}} \ldots, \gamma_k^{2^1})^T$ and $\mathbf{f}'_k$. Since m_k is usually small, we can use efficient bilinear form algorithms [1] for short cyclic convolutions to compute $\mathbf{L}_k\mathbf{f}'_k$. Those bilinear form algorithms have the following form:

$$\mathbf{L}_k\mathbf{f}'_k = \mathbf{b}_k \otimes \mathbf{f}'_k = \mathbf{Q}_k(\mathbf{R}_k\mathbf{b}_k \cdot \mathbf{P}_k\mathbf{f}'_k) = \mathbf{Q}_k(\mathbf{c}_k \cdot \mathbf{P}_k\mathbf{f}'_k)$$

where $\mathbf{P}_k$, $\mathbf{Q}_k$, and $\mathbf{R}_k$ are all binary matrices, $\mathbf{c}_k = \mathbf{R}_k\mathbf{b}_k$ is a precomputed constant vector and $\cdot$ denotes a component-wise multiplication between two vectors. Combining all the matrices, we get

$$\mathbf{F} = \mathbf{A}\mathbf{Q}(\mathbf{c} \cdot \mathbf{P}\mathbf{f}') \tag{2}$$

where $\mathbf{Q} = \mathrm{diag}(\mathbf{Q}_0, \mathbf{Q}_1, \ldots, \mathbf{Q}_{m-1})$, $\mathbf{P} = \mathrm{diag}(\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_{m-1})$ and $\mathbf{c} = (\mathbf{c}_0^T, \mathbf{c}_1^T, \ldots, \mathbf{c}_{m-1}^T)^T$.

The multiplications required by (2) are due to the component-wise multiplication between $\mathbf{c}$ and $\mathbf{P}\mathbf{f}'$ and the additions required by (2) are for multiplications between binary matrices and vectors. The direct implementation of CFFT in (2) requires much fewer multiplications than the direct implementation of DFT, at the expense of a very high additive complexity.

B. Common Subexpression Elimination

Given an $N \times M$ binary matrix $\mathbf{M}$ and an M-dimensional vector $\mathbf{x}$ over a field $\mathbb{F}$. The matrix vector multiplication $\mathbf{M}\mathbf{x}$ can be done by additions over $\mathbb{F}$ only, the number of which is denoted by $\mathcal{C}(\mathbf{M})$ since the complexity is determined by $\mathbf{M}$, when $\mathbf{x}$ is arbitrary. The problem of determining the minimal number of additions, denoted by $\mathcal{C}_{\mathrm{opt}}(\mathbf{M})$, has been shown to be NP-complete [22]. Instead, different common subexpression elimination algorithms (see, e.g., [13]–[15]) have been proposed to reduce $\mathcal{C}(\mathbf{M})$. The CSE algorithm proposed in [12] takes advantage of the *differential savings* and *recursive savings* and can greatly reduce the number of additions in calculating $\mathbf{M}\mathbf{x}$, although the reduced additive complexity, denoted by $\mathcal{C}_{\mathrm{CSE}}(\mathbf{M})$, is not guaranteed to be the minimum. Like other CSE algorithms, the CSE algorithm in [12] is randomized and the reduction results of different runs are not necessarily the same. Therefore, in practice, a better result can be obtained by first running the CSE algorithm many times and then selecting the smallest number of additions. The CSE algorithm in [12] greatly reduces the additive and overall complexities of CFFTs with length up to 1023, but it is much more difficult to reduce the additive complexity of longer CFFTs. This is because though the CSE algorithm in [12] has a polynomial complexity (it is shown that its complexity is $O(N^4 + N^3M^3)$), the runtime and memory requirements become prohibitive when M and N are very large, which occurs for long CFFTs.

C. Prime-Factor and Cooley-Tukey Algorithms

Both the prime-factor algorithm and the Cooley-Tukey algorithm first decompose an N-point DFT into shorter sub-DFTs and then construct the N-point DFT from the sub-DFTs [1]. The prime-factor algorithm requires that the length N has at least two co-prime factors, i.e., there exist two co-prime numbers N_1 and N_2 such that $N = N_1N_2$. For an integer $i \in \{0, 1, \ldots, N-1\}$, there is a unique integer pair (i_1, i_2) such that $0 \le i_1 \le N_1 - 1$, $0 \le i_2 \le N_2 - 1$ and $i = i_1N_2 + i_2N_1 \pmod{N}$, since N_1 and N_2 are co-prime. For any integer $j \in \{0, 1, \ldots, N-1\}$, let $j_1 = j \pmod{N_1}$, $j_2 = j \pmod{N_2}$, where $0 \le j_1 \le N_1 - 1$ and $0 \le j_2 \le N_2 - 1$. By the Chinese remainder theorem, (j_1, j_2) uniquely determines j and j can be represented by $j = j_1N_2^{-1}N_2 + j_2N_1^{-1}N_1 \pmod{N}$, where $N_2^{-1}N_2 = 1 \pmod{N_1}$ and $N_1^{-1}N_1 = 1 \pmod{N_2}$. Substituting the above representation of i and j in (1), we get $\alpha^{ij} = (\alpha^{N_2})^{i_1j_1}(\alpha^{N_1})^{i_2j_2}$, where α^{N_2} and α^{N_1} are the N_1th root and the N_2th root of 1, respectively. Therefore, (1) becomes

$$F_j = \underbrace{\sum_{i_1=0}^{N_1-1} \left(\overbrace{\sum_{i_2=0}^{N_2-1} f_{i_1N_2+i_2N_1} \alpha^{N_1i_2j_2}}^{N_2-\text{point DFT}} \right) \alpha^{N_2i_1j_1}}_{N_1-\text{point DFT}}. \tag{3}$$

In this way, the N-point DFT is obtained by using N_1- and N_2-point sub-DFTs. The N-point DFT result is derived by first carrying out N_1 N_2-point DFTs and N_2 N_1-point DFTs and then combining the results according to the representation of j. The prime-factor algorithm can also be applied to N_1- and N_2-point DFTs if they have co-prime factors.

The Cooley-Tukey algorithm has a different decomposition strategy from the prime-factor algorithm. Let $N = N_1N_2$, where N_1 and N_2 do not have to be co-prime. Let $i = i_1 + i_2N_1$, where $0 \le i_1 \le N_1 - 1$ and $0 \le i_2 \le N_2 - 1$ and $j = j_1N_2 + j_2$, where $0 \le j_1 \le N_1 - 1$ and $0 \le j_2 \le N_2 - 1$. Then (1) becomes

$$F_j = \underbrace{\sum_{i_1=0}^{N_1-1} \left(\overbrace{\sum_{i_2=0}^{N_2-1} f_{i_1+i_2N_1} \alpha^{N_1i_2j_2}}^{N_2-\text{point DFT}} \right) \alpha^{i_1j_2}\alpha^{N_2i_1j_1}}_{N_1-\text{point DFT}}. \tag{4}$$

In this way, the Cooley-Tukey algorithm also decomposes the N-point DFT into N_1- and N_2-point DFTs. However, compared with (3), (4) has an extra term $\alpha^{i_1j_2}$, which is called a *twiddle factor* and incurs additional multiplicative complexity. The Cooley-Tukey algorithm can be used for arbitrary non-prime length N, including the prime powers to which case the prime-factor algorithm cannot be applied. The Cooley-Tukey algorithm is very suitable if N has a lot of small factors: for example, a 2^n-point DFT by the Cooley-Tukey algorithm requires $O(n \cdot 2^n)$ multiplications.

III. p-Point Cyclic Convolutions Over $\mathrm{GF}(2^l)$

The lengths of cyclic convolutions involved in N-point CFFTs over $\mathrm{GF}(2^l)$ are the same as the sizes of the cyclotomic cosets modulo N with respect to $\mathrm{GF}(2)$ and they are usually much smaller than N, the length of the CFFTs. Efficient algorithms for such short cyclic convolutions play an essential role in the multiplicative complexity reduction of CFFTs.

Despite their significance, there are no general algorithms for efficient cyclic convolutions of arbitrary length over finite fields. Of course, efficient ad hoc algorithms for 2- to 9-point cyclic convolutions can be found in the literature (4- and 8-point can be found in [23] and [24], and the rest can be found in

[1] and [2]). Furthermore, cyclic convolutions with composite lengths can be constructed with multi-dimensional technologies described in [1]. For instance, a 10-point cyclic convolution algorithm can be constructed based on 2- and 5-point algorithms, while a 12-point cyclic convolution algorithm is constructed based on 3- and 4-point algorithms. However, an efficient algorithm for cyclic convolutions of larger prime lengths (for example, 11- or 13-point) is not available in the open literature. We can implement these cyclic convolutions via the convolution theorem. Although the DFT and the inverse DFT can be implemented by the Winograd algorithm [25] or the Rader algorithm [26], this approach remains inefficient, especially for small to moderate lengths. In [27], strategies to derive cyclic convolution algorithms directly over any finite field $\mathrm{GF}(q^m)$ were developed. Unfortunately, these methods are applicable only to lengths $q^m - 1$ or their factors.

Herein, for an odd prime p, we propose to reformulate a p-point cyclic convolution as the product of a $(p-1) \times (p-1)$ Toeplitz matrix and a vector. Since $p-1$ is composite, this product can be readily obtained by multi-dimensional technologies from well-known Toeplitz matrix vector products (TMVPs) of very small sizes, leading to efficient bilinear algorithms for p-point cyclic convolutions. Since these cyclic convolutions will be used for CFFTs over $\mathrm{GF}(2^l)$, we focus on cyclic convolutions over $\mathrm{GF}(2^l)$. However, our reformulation can be readily extended to the real and complex fields as well as more general finite fields. Furthermore, by multi-dimensional technologies, we can also obtain efficient algorithms for p^n-point cyclic convolutions. These algorithms are also key to long CFFTs.

For a p-dimensional vector $\mathbf{x} = (x_0, x_1, \ldots, x_{p-1})^T$ over some field $\mathrm{GF}(2^l)$, where p is any odd prime integer, we consider its corresponding polynomial $X(w) = \sum_{i=0}^{p-1} x_i w^i$. Assuming that the p-point cyclic convolution of two vectors $\mathbf{x}$ and $\mathbf{y}$ is $\mathbf{z}$, all of which are p-dimensional vectors over $\mathrm{GF}(2^l)$, their corresponding polynomials are related by [1]

$$Z(w) = X(w)Y(w) \pmod{w^p + 1}. \tag{5}$$

Note that $w^p+1 = (w+1)(w^{p-1}+w^{p-2}+\cdots+1)$ and $w+1$ and $w^{p-1} + w^{p-2} + \cdots + 1$ are co-prime in $\mathrm{GF}(2^l)$. Hence, by the Chinese remainder theorem, $Z(w)$ can be uniquely determined by Z_0 and $Z'(w) = \sum_{i=0}^{p-2} Z'_i w^i$, where

$$\begin{aligned} Z_0 &= Z(w) \pmod{w+1}, \\ Z'(w) &= Z(w) \pmod{w^{p-1} + w^{p-2} + \cdots + 1}. \end{aligned} \tag{6}$$

It is easy to see that $Z_0 = \sum_{i=0}^{p-1} z_i$, $Z'_i = z_i + z_{p-1}$ and the vector $\mathbf{Z}^\dagger = (Z_0, Z'_0, Z'_1, \ldots, Z'_{p-2}) = (Z_0, \mathbf{Z}'^T)^T$ can be derived by multiplying the vector $\mathbf{z}$ with a $p \times p$ matrix $\mathbf{B}$ with structure

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ & & & 1 \\ & \mathbf{I}_{p-1} & & \vdots \\ & & & 1 \end{bmatrix}$$

where $\mathbf{I}_{p-1}$ is a $(p-1) \times (p-1)$ identity matrix. That is, $\mathbf{Z}^\dagger = \mathbf{B}\mathbf{z}$. Simillarly, we have $\mathbf{X}^\dagger = \mathbf{B}\mathbf{x}$ and $\mathbf{Y}^\dagger = \mathbf{B}\mathbf{y}$.

From (5) and (6), it is easy to see that $Z_0 = X_0 Y_0$ and

$$Z'(w) = X'(w)Y'(w) \pmod{w^{p-1} + w^{p-2} + \cdots + 1}. \tag{7}$$

Therefore, to compute the p-point cyclic convolution $\mathbf{z}$ of $\mathbf{x}$ and $\mathbf{y}$, we first compute $\mathbf{X}^\dagger = \mathbf{B}\mathbf{x}$ and $\mathbf{Y}^\dagger = \mathbf{B}\mathbf{y}$, then compute $\mathbf{Z}^\dagger$ from $\mathbf{X}^\dagger$ and $\mathbf{Y}^\dagger$ and finally, $\mathbf{z} = \mathbf{B}^{-1}\mathbf{Z}^\dagger$.

From (7), the polynomial product can be computed as

$$\begin{aligned} &X'(w)Y'(w) \\ &\quad = \sum_{k=0}^{p-2} \sum_{j=0}^{p-2} \left(Y'_{k-j} + Y'_{k-j+p} + Y'_{p-1-j}\right) X'_j w^k \\ &\qquad (\mathrm{mod}\ w^{p-1} + w^{p-2} + \cdots + 1) \end{aligned} \tag{8}$$

and hence the vector $\mathbf{Z}'$ can be computed through a matrix product $\mathbf{Z}' = \mathbf{M}\mathbf{X}'$, where the elements of matrix $\mathbf{M}$ are

$$M_{k,j} = Y'_{k-j} + Y'_{k-j+p} + Y'_{p-1-j}. \tag{9}$$

Note that in (8) and (9), Y'_i is considered as zero outside its valid range, i.e., $Y'_i = 0$ if $i < 0$ or $i > p-2$.

We can check that $\mathbf{B}$ is an invertible matrix and $\mathbf{B}^{-1}$ is given by

$$\mathbf{B}^{-1} = \begin{bmatrix} 1 & \mathbf{A}_1 \\ \mathbf{A}_2 & \mathbf{A}_3 \end{bmatrix}$$

where the length-$(p-1)$ row vector $\mathbf{A}_1 = (0, 1, 1, \ldots, 1)$, the length-$(p-1)$ column vector $\mathbf{A}_2 = (1, 1, \ldots, 1)^T$ and the $(p-1) \times (p-1)$ matrix $\mathbf{A}_3$ has 0 on the first upper diagonal and 1 everywhere else.

Now consider the resulted cyclic convolution $\mathbf{z} = (z_0, \mathbf{z}'^T)^T$ as the product of $\mathbf{B}^{-1}$ and $\mathbf{Z}^\dagger$:

$$\mathbf{z} = \begin{bmatrix} z_0 \\ \mathbf{z}' \end{bmatrix} = \mathbf{B}^{-1} \begin{bmatrix} Z_0 \\ \mathbf{Z}' \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{A}_1 \\ \mathbf{A}_2 & \mathbf{A}_3 \end{bmatrix} \begin{bmatrix} Z_0 \\ \mathbf{Z}' \end{bmatrix}.$$

Values of z_0 and $\mathbf{z}'$ can be computed as $z_0 = Z_0 + \mathbf{A}_1\mathbf{Z}'$ and $\mathbf{z}' = \mathbf{A}_2 Z_0 + \mathbf{A}_3\mathbf{Z}'$. Note that $\mathbf{A}_1$ and $\mathbf{A}_3$ are related as $\mathbf{A}_1 = (1, 1, \ldots, 1)\ \mathbf{A}_3$. This implies that the sum of the components of $\mathbf{A}_3\mathbf{Z}'$ gives $\mathbf{A}_1\mathbf{Z}'$. Furthermore, $\mathbf{A}_2$ contains only 1's. Thus, the computation of z_0 and $\mathbf{z}'$ reduces to

$$\begin{aligned} z_0 &= Z_0 + \sum(\mathbf{A}_3\mathbf{Z}') \\ \mathbf{z}' &= [Z_0, Z_0, \ldots, Z_0]^T + \mathbf{A}_3\mathbf{Z}'. \end{aligned} \tag{10}$$

Equation (10) shows that multiplying a vector with $\mathbf{B}^{-1}$ needs only the evaluation of $\mathbf{A}_3\mathbf{Z}'$.

Since $\mathbf{Z}' = \mathbf{M}\mathbf{X}'$, one needs to compute $\mathbf{R}\mathbf{X}'$ where the $(p-1) \times (p-1)$ matrix $\mathbf{R} = \mathbf{A}_3\mathbf{M}$. We now show that $\mathbf{R}$ is a Toeplitz matrix. From the structure of $\mathbf{A}_3$, we have

$$R_{i,j} = M_{i+1,j} + \sum_{k=0}^{p-2} M_{k,j}. \tag{11}$$

From (9), using appropriate ranges for the three terms we get

$$\sum_{k=0}^{p-2} M_{k,j} = Y'_{p-1-j} + \sum_{s=0}^{p-2} Y'_s. \tag{12}$$

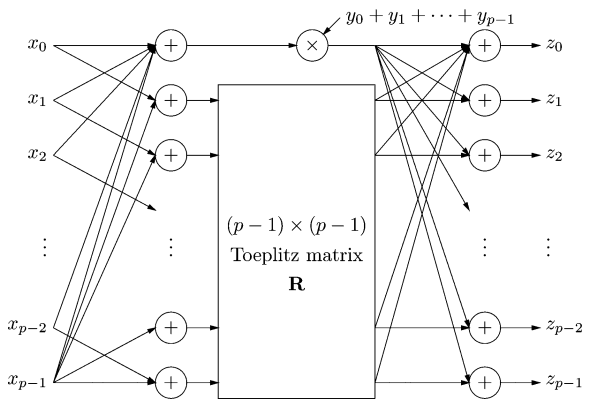


Fig. 1. An efficient algorithm for p-point cyclic convolutions.

Finally, combining (9), (11) and (12) gives

$$R_{i,j} = Y'_{i-j+1} + Y'_{i-j+p+1} + \sum_{s=0}^{p-2} Y'_s. \tag{13}$$

Since $R_{i,j}$ is a function of only $i-j$, $\mathbf{R}$ is a Toeplitz matrix. Recall that Y'_i is zero if its index is outside the valid range of 0 to $p-2$. Thus, in (13), at most one of the first two terms is valid for any combination of i and j.

Fig. 1 illustrates our algorithm for p-point cyclic convolutions, which rely on the implementation of $\mathbf{RX'}$. Direct implementation of $\mathbf{RX'}$ requires $(p-1)^2$ multiplications, but we can reduce it since $\mathbf{R}$ is a Toeplitz matrix. For any odd prime $p > 3$, $p-1$ is composite and $\mathbf{RX'}$ can be obtained by using multi-dimensional technologies from TMVPs of smaller sizes [16]–[20]. For example, CFFTs over $\mathrm{GF}(2^{11})$, $\mathrm{GF}(2^{13})$, $\mathrm{GF}(2^{17})$ and $\mathrm{GF}(2^{19})$, involve 11-, 13-, 17- and 19-point cyclic convolutions, respectively. Using our reformulations, these cyclic convolutions can be obtained from TMVPs of size 2×2, 3×3 (see, e.g., [20]) and 5×5 (provided in the Appendix). Hence, our reformulation leads to efficient cyclic convolution algorithms for any odd prime $p \leq 19$, which are sufficient for CFFTs over characteristic-2 fields as large as $\mathrm{GF}(2^{19})$.

This reformulation is also applicable to a prime greater than 19, where $p-1$ may have a prime factor p' greater than five. In this case, one can use two ad hoc techniques to proceed. First, one can break a $p' \times p'$ matrix into blocks and treat them separately. Second, one can extend the $p' \times p'$ matrix to a larger matrix so that it remains a Toeplitz matrix and its size becomes composite again. The complexity of the cyclic convolution algorithm obtained through this reformulation is much smaller than that of the direct implementation. For example, we can first extend the $(p-1) \times (p-1)$ Toeplitz matrix to a $2^{\lceil \log_2(p-1) \rceil} \times 2^{\lceil \log_2(p-1) \rceil}$ matrix and its product with a vector requires fewer than $3^{\lceil \log_2(p-1) \rceil}$ multiplications if we use the two-way split method described in [20].

We note that a p-point cyclic convolution can be formulated as a $p \times p$ circulant matrix vector product. Since a circulant matrix is a special case of Toeplitz matrix, one can of course apply the two ad hoc techniques described above to this $p \times p$ circulant matrix directly. However, since our reformulation turns a p-point cyclic convolution into a $(p-1) \times (p-1)$ TMVP, which directly benefits from multi-dimensional technologies at the expense of only one extra multiplication, we believe our reformulation will lead to a lower multiplicative complexity. We cannot prove this analytically, but will illustrate this point below with an example.

We also remark that our reformulation leads to bilinear algorithms for cyclic convolutions, which can be implemented efficiently since the pre- and post-addition matrices are all binary.

A. Example: 11-Point Convolution Algorithm Over $\mathrm{GF}(2^l)$

To illustrate the advantages of our reformulation above, we derive our efficient 11-point cyclic convolution algorithm over $\mathrm{GF}(2^l)$ and compare its multiplicative complexity with some other approaches. By using 2×2 and 5×5 TMVPs, we obtain an 11-point cyclic convolution algorithm $\mathbf{z} = \mathbf{Q}^{(11)}(\mathbf{R}^{(11)}\mathbf{y} \cdot \mathbf{P}^{(11)}\mathbf{x})$, where the matrices $\mathbf{Q}^{(11)}$, $\mathbf{P}^{(11)}$ and $\mathbf{R}^{(11)}$ are derived from a 10×10 TMVP, which is then decomposed into 2×2 and 5×5 TMVPs via multi-dimensional technologies. Since the 10×10 TMVP requires 42 multiplications, our 11-point cyclic convolution requires 43 multiplications. Details about the matrices can be found in [28].

Let us compare this multiplicative complexity with the two aforementioned ad hoc techniques. First, we can partition the 11×11 circulant matrix into a 10×10 Toeplitz matrix, a 10×1 column vector, a 1×10 row vector and a single element and then apply multi-dimensional technologies to the 10×10 TMVP. In addition to the 10×10 TMVP, this approach requires 21 extra multiplications, as opposed to the one in our approach. Second, we can extend the 11×11 circulant matrix to a 12×12 Toeplitz matrix and then apply multi-dimensional technologies to it. A 12×12 TMVP requires $54 = 3 \times 3 \times 6$ multiplications. Taking into account that we pad a zero to the 11×1 vector and that the last element of the TMVP is not needed, two multiplications can be saved and we need 52 multiplications in total (note that this total multiplicative complexity is the same regardless of the decomposition order of 12). We can also extend the 11×11 circulant matrix to a 15×15 Toeplitz matrix or a 16×16 one, which require 66 and 60 multiplications, respectively. Our reformulation is more efficient than these ad hoc techniques in terms of the multiplicative complexity. This is because our reformulation turns a p-point cyclic convolution into a $(p-1) \times (p-1)$ TMVP, which directly benefits from multi-dimensional technologies at the expense of only one extra multiplication.

We also compare our result with the implementation via the convolution theorem, i.e., first multiplying the DFTs of the two vectors component-wisely and then computing the inverse DFT of the resulting vector. If we use the Rader's algorithm [26] to implement the DFT and inverse DFT, it needs 101 multiplications in total. Hence, this approach is less efficient than ours.

By using the CSE algorithm in [12], our 11-point cyclic convolution algorithm requires 43 multiplications and 164 additions. When we use this algorithm in CFFTs over $\mathrm{GF}(2^{11})$, one of the two inputs is known in advance. The computation further reduces to 42 multiplications since one of the multiplication has an operand of one and 120 additions because the additions involving the known input can be pre-computed.

IV. Long Cyclotomic Fourier Transforms

A. 2047-Point CFFT Over $\mathrm{GF}(2^{11})$

The efficient algorithm for 11-point cyclic convolutions we designed in Section III-A is the key to the CFFTs over $\mathrm{GF}(2^{11})$. A direct implementation of a 2047-point CFFT with this cyclic convolution algorithm requires 7812 multiplications and 2130248 additions. The prohibitively high additive complexity is dominated by the multiplication between the 2047×2047 binary matrix $\mathbf{A}$ (see Section II-A) and a 2047-dimensional vector, which requires 2095280 additions. Unfortunately, if we use the CSE algorithm in [12] to reduce its additive complexity, the time complexity of the CSE algorithm itself is too high (it may need months to finish).

Due to the high time complexity of the CSE algorithm in [12], we have tried a simplified CSE algorithm with limited success. In the original CSE algorithm in [12], only one of the patterns with the greatest recursive savings is selected and removed in each round of iterations. Instead of selecting only one pattern, our simplified CSE algorithm has a reduced time complexity as it removes multiple patterns at one time. The simplified CSE algorithm with a reduced time complexity allows us to reduce the additive complexity for the 2047-point CFFT to 529720 additions, about one fourth of that for the direct implementation. Despite this improvement, the effectiveness of this simplified CSE algorithm is rather limited.

B. Difficulty With Long CFFTs

Consider an N-point CFFT over $\mathrm{GF}(2^l)$. Let $C_{s_0}, C_{s_1}, \ldots, C_{s_{m-1}}$ be the m cyclotomic cosets modulo N over $\mathrm{GF}(2)$ and $|C_{s_k}| = m_k$. Suppose an m_k-point cyclic convolution can be done with $\mathcal{M}(m_k)$ multiplications and hence directly implementing the N-point DFT with the CFFT requires $\sum_{k=0}^{m-1} \mathcal{M}(m_k)$ multiplications and $\mathcal{C}(\mathbf{AQ}) + \mathcal{C}(\mathbf{P})$ additions, where $\mathcal{C}(\cdot)$ denotes the number of additions we need to evaluate the product of a binary matrix and a vector. The multiplicative complexity can be further reduced because we can pre-compute the vector $\mathbf{c}$ in (2) and some of its elements may be one. Then the CSE algorithm can be applied to the matrices $\mathbf{AQ}$ and $\mathbf{P}$ to reduce $\mathcal{C}(\mathbf{AQ})$ and $\mathcal{C}(\mathbf{P})$ to $\mathcal{C}_{\mathrm{CSE}}(\mathbf{AQ})$ and $\mathcal{C}_{\mathrm{CSE}}(\mathbf{P})$, respectively. Since $\mathbf{P} = \mathrm{diag}(\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_{m-1})$ is a block diagonal matrix, we have $\mathcal{C}_{\mathrm{CSE}}(\mathbf{P}) = \sum_{i=0}^{m-1} \mathcal{C}_{\mathrm{CSE}}(\mathbf{P}_i)$. That is, we can reduce the additive complexity of each $\mathbf{P}_i$ to get a better result of $\mathcal{C}(\mathbf{P})$. Since the size of $\mathbf{P}_i$ is much smaller than that of $\mathbf{P}$, it allows us to run the CSE algorithm many times to achieve a smaller additive complexity. However, the matrix $\mathbf{AQ}$ is not a block diagonal matrix and therefore we have to apply the CSE algorithm directly to $\mathbf{AQ}$. When the size of $\mathbf{AQ}$ is large, the CSE algorithm in [12] requires a lot of time and memory and hence it is impractical for extremely long DFTs. As mentioned above, it would take months for the CSE algorithm in [12] to reduce the additive complexity of 2047-point CFFT over $\mathrm{GF}(2^{11})$, let alone 4095-point CFFTs over $\mathrm{GF}(2^{12})$. The prohibitively high time complexity of the CSE algorithm in [12] and the limited effectiveness of the simplified CSE algorithm motivate our composite cyclotomic Fourier transforms.

V. Composite Cyclotomic Fourier Transforms

A. Composite Cyclotomic Fourier Transforms

Instead of simplifying the CSE algorithm or designing other low complexity optimization algorithms, we propose composite cyclotomic Fourier transforms (CCFTs) by first decomposing a long DFT into shorter sub-DFTs, via the prime-factor or Cooley-Tukey algorithm and then implementing the sub-DFTs by CFFTs. Note that both the decompositions require only that α is a primitive Nth root of 1, hence they can be extended to finite fields easily. When N is prime, our CCFTs reduce to CFFTs. When N is composite, we first decompose the DFT into shorter sub-DFTs and then combine the sub-DFT results according to (3) or (4). The shorter sub-DFTs are implemented by CFFTs to reduce their multiplicative complexities and then we use the CSE algorithm in [12] to reduce their additive complexities. Finally, when N has multiple factors, the factorization can be carried out recursively.

Suppose the length of the DFT is composite, i.e., $N = N_1 N_2$. Either the prime-factor algorithm or the Cooley-Tukey algorithm can be used to decompose the N-point DFT into sub-DFTs when N_1 and N_2 are co-prime. When N_1 and N_2 are not co-prime, only the Cooley-Tukey algorithm can be used. It is easy to show that if N_1 and N_2 are co-prime, the prime-factor and Cooley-Tukey algorithms lead to the same additive complexity for CCFTs, but the Cooley-Tukey algorithm results in a higher multiplicative complexity due to the extra multiplications of twiddle factors. Hence, the prime-factor algorithm is better than the Cooley-Tukey algorithm in this case and the Cooley-Tukey algorithm is used only if the prime-factor algorithm cannot be applied.

We denote the multiplicative and additive complexities of an N-point DFT by $\mathcal{K}^{\mathrm{mult}}(N)$ and $\mathcal{K}^{\mathrm{add}}(N)$, respectively and the algorithm used to implement this DFT is specified in the subscription of $\mathcal{K}$. Assuming that $N = \prod_{i=1}^{s} N_i$ and the total number of non-unitary twiddle factors required by the Cooley-Tukey algorithm decompositions is denoted by T, the complexity of this decomposition is given by

$$\mathcal{K}^{\mathrm{add}}_{\mathrm{CCFT}}(N) = \sum_{i=1}^{s} \frac{N}{N_i} \mathcal{K}^{\mathrm{add}}_{\mathrm{CFFT}}(N_i) \tag{14}$$

$$\mathcal{K}^{\mathrm{mult}}_{\mathrm{CCFT}}(N) = \sum_{i=1}^{s} \frac{N}{N_i} \mathcal{K}^{\mathrm{mult}}_{\mathrm{CFFT}}(N_i) + T. \tag{15}$$

For $N|2^l - 1$, $4 \le l \le 12$, there is at most one pair of N_i's that are not co-prime in the decomposition of N, say N_1 and N_2, without loss of generality. In this case, $T = N/(N_1 - 1)(N_2 - 1)(N_1 N_2)$. If all the elements in the decomposition of N are co-prime to each other, then $T = 0$.

The decomposition allows our CCFTs to achieve lower complexities for several reasons. First, this divide-and-conquer strategy is used in many fast Fourier transforms. If we assume CFFTs have quadratic additive complexities with their length N when directly implemented (this assumption is at least supported by the additive complexities of the CFFTs without CSE in Table IV), the CCFT decomposition reduces the additive complexity from $O(N^2)$ to $O(N \sum_{i=1}^{s} N_i)$. Second, the lengths of the sub-DFTs are much shorter, which enables us to apply several powerful but complicated techniques to

TABLE I
COMPLEXITIES OF SHORT CYCLIC CONVOLUTIONS OVER $\mathrm{GF}(2^l)$

L	mult.	additive complexities		
		$\mathcal{C}_{\mathrm{CSE}}(\mathbf{Q}^{(L)})$	$\mathcal{C}_{\mathrm{CSE}}(\mathbf{P}^{(L)})$	total
2	1	2	1	3
3	3	5	4	9
4	5	9	4	13
5	9	16	10	26
6	10	21	11	32
7	12	24	23	47
8	19	35	16	51
9	18	40	31	71
10	28	52	31	83
11	42	76	44	120
12	32	53	34	87

reduce the complexities of the sub-DFTs. For example, it takes much less time and memory to apply the CSE algorithm in [12] to the sub-DFTs and thus we can run it multiple times to get a better reduction result. Third, when the length of the DFT admits different factorizations (for example, $2^6 - 1 = 63 = 3 \times 21 = 9 \times 7$), we choose the decomposition(s) with the lowest complexity.

B. Complexity Reduction

We reduce the additive and the overall complexities of our CCFTs in three steps. First, we reduce the complexities of short cyclic convolutions. Second, we use these short cyclic convolutions to construct CFFTs of moderate lengths. Third, we use CFFTs of moderate lengths as sub-DFTs to construct our CCFTs.

1) Complexity Reduction of Short Cyclic Convolutions: Efficient short cyclic convolution algorithms, such as the p-point reformulations we proposed in Section III, are the keys to the multiplicative complexity reduction of CFFTs and our CCFTs. Suppose an L-point cyclic convolution $\mathbf{b}^{(L)} \otimes \mathbf{a}^{(L)}$ is calculated with the bilinear form $\mathbf{Q}^{(L)}(\mathbf{R}^{(L)}\mathbf{b}^{(L)} \cdot \mathbf{P}^{(L)}\mathbf{a}^{(L)})$. Since $\mathbf{b}^{(L)}$ is the normal basis in our CCFTs, $\mathbf{R}^{(L)}\mathbf{b}^{(L)}$ can be precomputed to reduce the multiplicative complexity. We apply the CSE algorithm in [12] to reduce the additive complexities in the multiplication with binary matrices $\mathbf{Q}^{(L)}$ and $\mathbf{P}^{(L)}$. The complexity reduction results $\mathcal{C}_{\mathrm{CSE}}(\mathbf{Q}^{(L)})$, $\mathcal{C}_{\mathrm{CSE}}(\mathbf{P}^{(L)})$, the total additive complexity $\mathcal{C}_{\mathrm{CSE}}(\mathbf{Q}^{(L)}) + \mathcal{C}_{\mathrm{CSE}}(\mathbf{P}^{(L)})$ and the multiplicative complexities are listed in Table I. Note that the complexity of the 11-point cyclic convolution is derived from our reformulation introduced in Section III.

2) Additive Complexity Reduction of CFFTs With Moderate Lengths: Blocks of CFFTs with moderate lengths are used to build our CCFTs. Their moderate lengths allow us to use multiple techniques to reduce their additive complexities.

- First, for any CFFT, we run the CSE algorithm in [12] multiple times and then choose the best results.
- Second, for each CFFT in (2), we may reduce $\mathcal{C}(\mathbf{AQ})$ together as a whole, or reduce $\mathcal{C}(\mathbf{A})$ and $\mathcal{C}(\mathbf{Q})$ separately. Since $(\mathbf{AQ})\mathbf{v} = \mathbf{A}(\mathbf{Qv})$, $\mathcal{C}_{\mathrm{opt}}(\mathbf{AQ}) \leq \mathcal{C}_{\mathrm{opt}}(\mathbf{A}) + \mathcal{C}_{\mathrm{opt}}(\mathbf{Q})$. However, this property may not hold for the CSE algorithm because it may not find the optimal solutions. Furthermore, we may benefit from reducing $\mathcal{C}(\mathbf{A})$ and $\mathcal{C}(\mathbf{Q})$ separately for the following reasons. First, $\mathbf{Q}$ has a block diagonal structure, which is similar as $\mathbf{P}$ and we can find a better reduction result for $\mathcal{C}(\mathbf{Q})$. Second, $\mathbf{AQ}$ has much more columns than $\mathbf{A}$ and hence the CSE algorithm requires less memory and time to reduce $\mathbf{A}$ than to reduce $\mathbf{AQ}$.
- Third, there is flexibility in terms of normal bases used to construct the matrix $\mathbf{A}$ in (2) and this flexibility can be used to further reduce the additive complexity of any CFFT. For each cyclotomic coset, a normal basis is needed. A normal basis is not unique in finite fields and any normal basis can be used in the construction of the matrix $\mathbf{A}$, leading to the same multiplicative complexity. But different normal bases result in different $\mathbf{A}$s and hence different additive complexities due to $\mathbf{A}$. There are several options regarding the normal basis. One can simply choose a fixed normal basis for all cyclotomic cosets of the same size as in [12]. A more ideal option is to enumerate all possible normal bases and their corresponding $\mathbf{A}$s and to select the smallest additive complexity. However, when the underlying field is large, the number of possible normal bases is very large and hence it becomes infeasible to enumerate all possible constructions. Thus, in this paper we use a compromise of these two options: for each cyclotomic coset we choose a normal basis at random and the combination of random normal bases leads to $\mathbf{A}$; we minimize the complexity over as many combinations as complexity permits. We refer to this as a random normal basis option.

We emphasize that all three techniques require multiple runs of the CSE algorithm. Since the time and memory requirements of the CSE algorithm grow with the length of DFT, the moderate length of the sub-DFTs is the key enabler of these techniques. Though the CSE algorithm may be costly for moderate length CFFTs, it is a one-time task.

For any $k \leq 320$ so that $k|2^l - 1$ $(4 \leq l \leq 12)$, the multiplicative and additive complexities of the k-point CFFT are shown in Table II. Table II shows four different schemes to reduce the additive complexity for CFFTs. Schemes A and B both use the fixed normal basis option in the construction of the matrix $\mathbf{A}$, while schemes C and D are based on the random normal basis option. Schemes A and C reduce $\mathcal{C}(\mathbf{A})$ and $\mathcal{C}(\mathbf{Q})$ separately, while schemes B and D reduce $\mathcal{C}(\mathbf{AQ})$ as a whole. For smaller CFFTs, we typically minimize the complexity over hundreds of combinations of normal bases and fewer combinations for longer CFFTs. In Table II, the smallest additive complexities are in a boldface font. We observe that the random normal basis option offers further additive complexity reduction in most of the cases. However, since the fixed normal basis is not necessarily one of the combinations, in some cases the fixed normal basis option outperforms the random normal basis option. Also, sometimes applying the CSE to $\mathbf{AQ}$ together as a whole leads to lower complexity and in some cases it is better to apply the CSE to $\mathbf{A}$ and $\mathbf{Q}$ separately.

3) Construction of CCFTs Using Moderate-Length CFFTs as Sub-DFTs: We use the CFFTs with moderate lengths in Table II as sub-DFTs to construct our CCFTs. With (14) and (15), the computational complexities of our CCFTs over $\mathrm{GF}(2^l)$ $(4 \leq l \leq 12)$ with non-prime lengths can be calculated. The results are summarized in Table III, where the factorizations in parentheses are not co-prime and the Cooley-Tukey algorithm is used in these cases. We have tried all the decompositions with lengths smaller than 320 and the decompositions with the smallest overall complexities are listed in Table III. Note that for each sub-DFT, the scheme with the smallest additive complexity

TABLE II
THE COMPLEXITIES OF THE CFFTS WHOSE LENGTHS ARE LESS THAN 320 AND ARE FACTORS OF $2^l - 1$ FOR $1 \leq l \leq 12$

N	l	mult.	additive complexities			
			A	B	C	D
3	2	1	**6**	**6**	**6**	**6**
5	4	5	20	**16**	20	**16**
7	3	6	31	**24**	31	**24**
9	6	11	51	**48**	51	**48**
11	10	28	109	102	102	**84**
13	12	32	125	100	110	**91**
15	4	16	87	**74**	87	**74**
17	8	38	153	163	**151**	153
21	6	27	167	179	**147**	153
23	11	84	335	407	**323**	357
31	5	54	354	**299**	335	350
33	10	85	413	440	**404**	434
35	12	75	406	303	358	**299**
39	12	97	502	425	472	**391**
45	12	90	481	415	498	**414**
51	8	115	**641**	755	676	739
63	6	97	798	**759**	806	1031
65	12	165	1092	**901**	1114	915
73	9	144	1498	1567	**1447**	1526
85	8	195	1601	1816	**1589**	1810
89	11	336	**2085**	4326	2247	3973
91	12	230	1668	1431	1596	**1421**
93	10	223	1772	1939	**1736**	1788
105	12	234	1762	1481	1776	**1333**
117	12	299	2304	2028	2366	**1947**
195	12	496	4900	4230	4942	**4166**
273	12	699	8064	**7217**	8082	7223
315	12	752	8965	**8032**	9899	8099

TABLE III
THE SMALLEST COMPLEXITY OF OUR N-POINT CCFTS OVER $\mathrm{GF}(2^l)$ FOR COMPOSITE N AND $N|2^l - 1$ FOR $4 \leq l \leq 12$ (WE ASSUME THE SUB-DFTS ARE SHORTER THAN 320)

l	N	Decomposition	mult.	add.	total
4	15	1×15	16	74	186
6	9	(3×3)	10	36	146
	21	3×7	25	114	389
	63	$(3 \times 3) \times 7$	124	468	1832
8	51	1×51	115	641	2366
	85	1×85	195	1590	4515
	255	3×85	670	5277	15327
9	511	7×73	1446	11881	36463
10	33	1×33	85	404	2019
	93	3×31	193	1083	4750
	341	1×341	922	15184	32702
	1023	33×31	4417	22391	106314
11	2047	23×89	15204	76702	395986
12	35	5×7	65	232	1727
	39	1×39	97	391	2622
	45	(3×15)	91	312	2405
	65	1×65	165	902	4697
	91	1×93	230	1421	6711
	105	7×15	202	878	5524
	117	1×117	299	1947	8824
	195	3×65	560	3093	15973
	273	3×91	781	4809	22772
	315	5×63	800	4803	23203
	455	7×65	1545	7867	43402
	585	5×117	2080	11607	59447
	819	7×117	2795	16437	80722
	1365	7×195	4642	33842	140608
	4095	65×63	16700	106098	490198

listed in Table II is used in the CCFT implementation to reduce the total additive complexity. We also note that all DFT lengths in Table III are composite. The prime lengths are omitted because in these cases, a CCFT reduces to a CFFT, which can be found in Table II.

Since some lengths of the DFTs have different decompositions, it is possible that one decomposition has a smaller additive complexity but a larger multiplicative complexity than another one. Therefore, we need a metric to compare the overall complexities between different decompositions. In this paper, we follow our previous work [12] and assume that the complexity of a multiplication over $\mathrm{GF}(2^l)$ is $2l-1$ times of that of an addition over the same field and the total complexity of a DFT is a weighted sum of the additive and multiplicative complexities, i.e., $\mathrm{total} = (2l-1) \times \mathrm{mult} + \mathrm{add}$. This assumption is based on both the software and hardware implementation considerations [12]. Table III lists the decompositions with the smallest overall complexities.

Table III provide complexities of all N-point DFTs over $\mathrm{GF}(2^l)$ when $N|2^l-1$ and $4 \leq l \leq 12$. Note that the decomposition corresponding to $1 \times N$ is merely the N-point CFFT over $\mathrm{GF}(2^l)$. We have used the simplified CSE algorithm described in Section IV-A to reduce the complexity of the 2047-point CFFTs over $\mathrm{GF}(2^{11})$ and applied the CSE algorithm in [12] to the other CFFTs. Thus, we have expanded the results of [12], where only the (2^l-1)-point CFFTs over $\mathrm{GF}(2^l)$ were given. We also observe that for some short lengths (see, for example, $N = 15$, 33, or 65), the N-point CFFTs lead to the lowest complexity for the N-point CCFTs. For the DFTs longer than 320, i.e., 511-point CFFTs over $\mathrm{GF}(2^9)$, 341-point CFFTs over $\mathrm{GF}(2^{10})$ and 455-, 585-, 819-, and 1365-point CFFTs over $\mathrm{GF}(2^{12})$, the time complexity of the CSE algorithm in [12] is still considerable. Thus, we cannot minimize their complexities using schemes A, B, C, and D and hence they are not listed in Table II.

Though the twiddle factors in the Cooley-Tukey decomposition incur extra multiplicative complexity, Table III show that the Cooley-Tukey decomposition reduces the total complexity of our CCFTs in some cases (the decompositions in parentheses). For example, a 9-point CFFT requires 11 multiplications and 48 additions and a 3×3 CCFT based on the Cooley-Tukey decomposition requires 10 multiplications and 36 additions. Despite the twiddle factors, the CCFT based on the Cooley-Tukey decomposition have lower multiplicative and additive complexities, because we can take advantage of the low complexity of the 3-point DFT.

C. Complexity Comparison and Analysis

We compare the complexities of our CCFTs with those of previously proposed FFTs in the literature in Table IV. For each length, the lowest total complexity is in boldface font. In Table IV, our CCFTs achieve the lowest complexities for $N \geq 255$. Although the algorithm in [29] is proved asymptotically fast, the complexities of our CCFTs are only a fraction of those in [29] and the advantage grows as the length increases. Although the FFTs in [9] are also based on the prime-factor algorithm, our CCFTs achieve lower complexities for two reasons. First, since our CCFTs use CFFTs as the sub-DFTs, the multiplicative complexities of our CCFTs are greatly reduced compared with the FFTs in [9]. For example, the multiplicative complexity of our 511-point CCFT is only one fourth of the prime-factor algorithm in [9]. Furthermore, using the powerful CSE algorithm in [12], the additive complexities of our CCFTs are also greatly reduced. Compared with the CFFTs, our CCFTs have somewhat higher multiplicative complexities, but this is

TABLE IV
COMPARISON OF THE COMPLEXITIES OF OUR N-POINT CCFTS WITH FFTS AVAILABLE IN THE LITERATURE

N	Field	Wang and Zhu [29]			Trung et al. [9]			CFFT					CCFT		
		mult.	add.	total	mult.	add.	total	mult.	w/o CSE		w/ CSE [12]		mult.	add.	total
									add.	total	add.	total			
15	GF(2^4)	41	97	384	–	–	–	16	201	313	74	**186**	20	78	218
63	GF(2^6)	801	801	9612	–	–	–	97	2527	3594	759	**1826**	124	468	1832
255	GF(2^8)	1665	5377	30352	1135	3887	20902	586	34783	43573	6736	15526	670	5277	**15327**
511	GF(2^9)	13313	13313	239634	6516	17506	128278	1014	141710	158948	23130	40368	1446	11881	**36463**
1023	GF(2^{10})	32257	32257	645140	5915	30547	142932	2827	536093	589806	75360	129073	4417	22391	**106314**
2047	GF(2^{11})	78601	78601	1689622	–	–	–	7812	2130248	2294300	–	–	15204	76702	**395986**
4095	GF(2^{12})	180225	180225	4325400	–	–	–	10832	8434414	8683550	–	–	16700	106098	**490198**

more than made up by their reduced additive complexities. The additive complexities of our CCFTs are only a small fraction of those of CFFTs when directly implemented. Compared with the CFFTs with reduced additive complexities in [12], our CCFTs still have much smaller additive complexities due to their decomposition structure for $N \geq 63$. For example, the additive complexities of our CCFT is only about half of that of the CFFT for $N = 511$ and one third for $N = 1023$. Due to the significant reduction of the additive complexities, the total complexities of our CCFTs with $N \geq 255$ are lower than those of CFFTs. In comparison to CFFTs, the improvement by our CCFTs also grows as the length increases.

For the prime-length DFTs, such as the 31-point DFT over $\mathrm{GF}(2^5)$, 127-point DFT over $\mathrm{GF}(2^7)$ and 8191-point DFT over $\mathrm{GF}(2^{13})$, our CCFTs reduce to the CFFTs and they have the same computational complexities.

In the end, we remark that our CCFT is built from shorter DFTs and short DFTs can be used as modules in the hardware design. We may either pipeline them to achieve a high speed, or reuse them to save the chip area. This modular structure is favorable to hardware implementations.

VI. CONCLUSION

For any odd prime integer p, we reformulate a p-point cyclic convolution as a $(p-1) \times (p-1)$ Toeplitz matrix vector product, leading to efficient cyclic convolution algorithms. Based on this reformulation, we have obtained an efficient 11-point cyclic convolution algorithm and derived the CFFTs over $\mathrm{GF}(2^{11})$. We have also proposed a novel composite cyclotomic Fourier transform algorithm that leads to lower complexities through decomposing long DFTs into shorter ones using the prime-factor or Cooley-Tukey algorithm. Our CCFTs over $\mathrm{GF}(2^l)$ $(4 \leq l \leq 12)$ have lower complexities than previously known FFTs over finite fields. They also have a modular structure, which is desirable in hardware implementations.

APPENDIX
5×5 TOEPLITZ MATRIX VECTOR PRODUCT OVER $\mathrm{GF}(2^l)$

An $n \times n$ TMVP over $\mathrm{GF}(2^l)$ as

$$\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} r_{n-1} & r_n & \cdots & r_{2n-2} \\ r_{n-2} & r_{n-1} & \cdots & r_{2n-3} \\ \vdots & \vdots & \ddots & \vdots \\ r_0 & r_1 & \cdots & r_{n-1} \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{bmatrix}$$

can be computed with bilinear algorithm $\mathbf{E}^{(n)}(\mathbf{G}^{(n)}\mathbf{r} \cdot \mathbf{H}^{(n)}\mathbf{v})$, where $\mathbf{r} = (r_0, r_1, \ldots, r_{2n-2})^T$, $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})^T$ and $\mathbf{E}^{(n)}$, $\mathbf{G}^{(n)}$ and $\mathbf{H}^{(n)}$ are all binary matrices.

For $n = 5$,

$$\mathbf{G}^{(5)} = \begin{bmatrix} 1&1&1&1&1&0&0&0&0 \\ 0&1&1&1&1&1&0&0&0 \\ 0&0&1&1&1&1&1&0&0 \\ 0&0&0&1&1&1&1&1&0 \\ 0&0&0&0&1&1&1&1&1 \\ 0&1&0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0 \\ 0&0&0&1&1&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&1&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&1&0&0&1&0 \\ 0&0&0&0&1&0&0&0&0 \end{bmatrix}, \quad \mathbf{H}^{(5)} = \begin{bmatrix} 1&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \\ 1&1&0&0&0 \\ 1&0&1&0&0 \\ 1&0&0&1&0 \\ 0&1&1&0&0 \\ 0&1&0&0&1 \\ 0&0&1&1&0 \\ 0&0&1&0&1 \\ 0&0&0&1&1 \\ 1&1&0&1&1 \end{bmatrix},$$

and

$$\mathbf{E}^{(5)} = \begin{bmatrix} 0&0&0&0&1&0&0&0&0&1&0&1&1&1 \\ 0&0&0&1&0&0&0&1&0&0&1&0&1&1 \\ 0&0&1&0&0&0&1&0&1&0&1&1&0&0 \\ 0&1&0&0&0&1&0&0&1&1&0&0&0&1 \\ 1&0&0&0&0&1&1&1&0&0&0&0&0&1 \end{bmatrix}.$$

REFERENCES

[1] R. E. Blahut, *Fast Algorithms for Digital Signal Processing*. Reading, MA: Addison-Wesley, 1985.

[2] R. E. Blahut, *Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley, 1984.

[3] S. B. Wicker, *Error Control Systems for Digital Communications and Storage*. Upper Saddle River, NJ: Prentice-Hall, 1995.

[4] "Hard disk drive long data sector," White Paper, Apr. 20, 2007 [Online]. Available: http://www.idema.org/

[5] Y. Han, W. E. Ryan, and R. Wesel, "Dual-mode decoding of product codes with application to tape storage," in *Proc. IEEE Global Telecommun. Conf.*, Nov. 2, 2005, vol. 3, pp. 1255–1260.

[6] T. Buerner, R. Dohmen, A. Zottmann, M. Saeger, and A. J. van Wijngaarden, "On a high-speed Reed–Solomon CODEC architecture for 43 Gb/s optical transmission systems," in *Proc. 24th Int. Conf. Microelectronics*, May 16–19, 2004, vol. 2, pp. 743–746.

[7] I. J. Good, "The interaction algorithm and practical Fourier analysis," *J. Roy. Stat. Soc. Series B (Methodol.)*, vol. 20, no. 2, pp. 361–372, 1958.

[8] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, no. 90, pp. 297–301, 1965.

[9] T. K. Truong, P. D. Chen, L. J. Wang, Y. Chang, and I. S. Reed, "Fast, prime factor, discrete Fourier transform algorithms over $\mathrm{GF}(2^m)$ for $8 \leq m \leq 10$," *Inf. Sci.*, vol. 176, no. 1, pp. 1–26, 2006.

[10] P. V. Trifonov and S. V. Fedorenko, "A method for fast computation of the Fourier transform over a finite field," *Probl. Inf. Transm.*, vol. 39, no. 3, pp. 231–238, 2003.

[11] S. V. Fedorenko, "A method for computation of the discrete Fourier transform over a finite field," *Probl. Inf. Transm.*, vol. 42, pp. 139–151, 2006.
[12] N. Chen and Z. Yan, "Cyclotomic FFTs with reduced additive complexities based on a novel common subexpression elimination algorithm," *IEEE Trans. Signal Process.*, vol. 57, no. 3, pp. 1010–1020, Mar. 2009.
[13] P. Trifonov, "Matrix-vector multiplication via erasure decoding," in *Proc. XI Int. Symp. Probl. Redundancy in Inf. Control Syst.*, Jul. 2007, pp. 104–108.
[14] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms Reading*. Reading, MA: Addison-Wesley, 1974.
[15] O. Gustafsson and M. Olofsson, "Complexity reduction of constant matrix computations over the binary field," in *Proc. 1st Int. Workshop Arithmetic of Finite Fields (WAIFI)*, Berlin, Germany, 2007, pp. 103–115.
[16] J. C. Allwright, "Real factorisation of noncyclic-convolution operators with application to fast convolution," *Electron. Lett.*, vol. 7, no. 24, pp. 718–719, 1971.
[17] D. A. Pitassi, "Fast convolution using the Walsh transforms," *IEEE Trans. Electromagn. Compat.*, pp. 130–133, 1971.
[18] R. Agarwal and C. Burrus, "Fast one-dimensional digital convolution by multidimensional techniques," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 22, no. 1, pp. 1–10, 1974.
[19] S. Winograd, *Arithmetic Complexity of Computations*. Philadelphia, PA: SIAM, 1980.
[20] H. Fan and M. A. Hasan, "A new approach to subquadratic space complexity parallel multipliers for extended binary fields," *IEEE Trans. Comput.*, vol. 56, pp. 224–233, 2007.
[21] T. G. Zakharova, "Fourier transform evaluation in fields of characteristic 2," *Probl. Peredachi Inf.*, vol. 28, pp. 62–77, 1992.
[22] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, ser. Books in Mathematical Sciences. San Francisco, CA: Freeman, 1979.
[23] V. Afanasyev and I. Grushko, "FFT algorithms for the fields $\mathrm{GF}(2^m)$," in *Pomekhoustoichivoe Kodirovanie i Nadezhnost' EVM (Error-Resistant Coding and Reliability of Computers)* (in Russian). Moscow: Nauka, 1987, pp. 33–55.
[24] N. Churkov and S. Fedorenko, "A method for construction of composite length cyclic convolutions over finite fields," in *Proc. 32nd Conf. Nedelya Nauki Sankt-Peterburgskogo Gosudarstvennogo Polytechnicheskogo universiteta (Scientific Week of Saint-Petersberg State Polytechnic Univ.)*, 2004, vol. 5, pp. 180–181.
[25] S. Winograd, "On computing the discrete Fourier transforms," *Proc. Nat. Acad. Sci. USA*, vol. 73, no. 4, pp. 1005–1006, Apr. 1976.
[26] C. M. Rader, "Discrete Fourier transforms when the number of data samples is prime," *Proc. IEEE*, vol. 56, pp. 1107–1108, Jun. 1968.
[27] M. Wagh and S. Morgera, "A new structured design method for convolutions over finite fields, Part I," *IEEE Trans. Inf. Theory*, vol. 29, no. 4, pp. 583–595, Jul. 1983.
[28] X. Wu, M. D. Wagh, N. Chen, Z. Yan, and Y. Wang, "Composite cyclotomic Fourier transforms with reduced complexities," 2010 [Online]. Available: http://arxiv.org/abs/1007.1213
[29] Y. Wang and X. Zhu, "A fast algorithm for the Fourier transform over finite fields and its VLSI implementation," *IEEE J. Sel. Areas Commun.*, vol. 6, pp. 572–577, Apr. 1988.

Xuebin Wu received both the B.E. and M.E. degrees in electrical engineering from Tsinghua University, Beijing, China, in 2004 and 2007, respectively. He is currently working towards the Ph.D. degree in electrical engineering at Lehigh University, Bethlehem, PA.

His research interest lies in channel coding technologies and VLSI architecture and circuit design for digital signal processing and communication systems.

Meghanad Wagh received the B.Tech. and Ph.D. degrees in electrical engineering from the Indian Institute of Technology, Bombay, India.

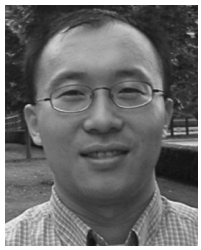
He is currently an Associate Professor of computer engineering at Lehigh University, Bethlehem, PA, where he also serves as the co-director of the computer engineering program. He is a consultant to a number of industries. His prior appointments include a postdoctoral fellowship at Concordia University, Montreal, QC, Canada, and an assistant professorship at the Old Dominion University, Norfolk, VA. His research interests include digital signal processing algorithms, parallel architectures, and design of high-speed nano-electronic circuits.

Ning Chen (S'06–M'10) received the B.E. and M.E. degrees from Tsinghua University, Beijing, China, in 2001 and 2004, respectively and the Ph.D. degree from Lehigh University, in 2010, all in electrical engineering.

From 2009 to 2010, he was with the Enterprise Storage Division, PMC-Sierra, Allentown, PA. Currently, he is with SandForce, Inc., Saratoga, CA. His research interests are in the VLSI design and implementation of digital signal processing and communication systems.

Ying Wang (S'00-M'06) received the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign in 2006.

Currently, she is a Staff Engineer with the New Jersey Research Center of QUALCOMM Corporate Research and Development, Bridgewater, NJ. Her research interests include wireless communications, statistical signal processing, multimedia security and forensics and detection and estimation theory.



Zhiyuan Yan (S'00-M'03-SM'08) received the B.E. degree in electronic engineering from Tsinghua University, Beijing, China, in 1995 and the M.S. and Ph.D. degrees, both in electrical engineering, from the University of Illinois, Urbana, in 1999 and 2003, respectively.

During summer 2000 and 2002, He was a Research Intern with Nokia Research Center, Irving, Texas. He joined the Electrical and Computer Engineering Department of Lehigh University, Bethlehem, Pennsylvania, as an Assistant Professor in August 2003. His current research interests are in coding theory, cryptography, wireless communications and VLSI implementations of communication and signal processing systems and he has published over 70 technical papers in refereed journals and conference proceedings.

Dr. Yan served as a Technical Program Committee Co-Chair and a General Co-Chair for ACM Great Lakes Symposium on VLSI in 2007 and 2008, respectively. He has been an Associate Editor for the IEEE COMMUNICATIONS LETTERS since 2008 and is an Associate Editor for the *Journal of Signal Processing Systems*. He is a member of the IEEE Information Theory, Communications, Signal Processing and Computer Societies. He is also a recipient of the prestigious National Science Foundation (NSF) Faculty Early Career Development (CAREER) award in 2011.