IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 18, 2018, accepted July 17, 2018, date of publication July 25, 2018, date of current version August 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2859451

N-Term Karatsuba Algorithm and Its Application to Multiplier Designs for Special Trinomials

YIN LI, YU ZHANG, XIAOLI GUO, AND CHUANDA QI
Department of Computer Science and Technology, Xinyang Normal University, Xinyang 464000, China
Corresponding author: Yin Li (yunfeiyangli@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61402393 and Grant 61601396 and in part by the Shanghai Key Laboratory of Integrated Administration Technologies for Information Security under Grant AGK201607.

ABSTRACT In this paper, we propose a new type of non-recursive Mastrovito multiplier for GF(2^m) using an n-term Karatsuba algorithm (KA), where GF(2^m) is defined by an irreducible trinomial, x^m+x^k+1, $m = nk$. We show that such a type of trinomial combined with the n-term KA can fully exploit the spatial correlation of entries in related Mastrovito product matrices and lead to a low-complexity architecture. The optimal parameter n is further studied. As the main contribution of this paper, the lower bound of the space complexity of our proposal is about $O(m^2/2) + m^{3/2}$). Meanwhile, the time complexity matches the best Karatsuba multiplier known to date. To the best of our knowledge, it is the first time that Karatsuba-based multiplier has reached such a space complexity bound while maintaining a relatively low time delay.

INDEX TERMS N-term Karatsuba algorithm, specific trinomials, bit-parallel multiplier.

I. INTRODUCTION

The finite field $GF(2^m)$ arithmetic has many applications in cryptography and error-correcting code [1], [2]. For instance, one of the most important applications of $GF(2^m)$ is the elliptic cure cryptosystem (ECC) [3]. Among the $GF(2^m)$ arithmetic operations, multiplication is of most importance because other costly operations such as exponentiation and inversion can be carried out by iterative multiplications. Therefore, it is necessary to design highly efficient multipliers for $GF(2^m)$ multiplication.

The choices of the field basis and irreducible polynomials are crucial to multiplier design. Compared with other bases, polynomial basis (PB) is more promising in the sense of flexibility in irreducible polynomial selection and hardware optimization [9]. Moreover, some variations of polynomial basis, e.g., shifted polynomial basis (SPB) [5], [13] and generalized polynomial basis (GPB) [10], are proposed as well to optimize the multiplier architecture further. Among these irreducible polynomials in use, irreducible trinomial is one of the most common considerations. During recent years, many bit-parallel multiplier using PB have been proposed for $GF(2^m)$ generated with an irreducible trinomial [7], [12], [16], [17], [27].

Generally speaking, the PB multiplication consists of two steps: polynomial multiplication and modulo reduction. The polynomial multiplication can be optimized using a divide-and-conquer algorithm such as Karatsuba algorithm (KA) [4], [18]. Such an algorithm saves coefficient multiplications at the cost of extra additions compared to the school-book method. Thus, it can be easily adopted to design efficient $GF(2^m)$ multipliers. Specifically, there exists a class of Karatsuba based multipliers, named as non-recursive Karatsuba multiplier, only apply KA once in the polynomial multiplication and obtain a trade-off between the space and time complexities. During recent years, several non-recursive Karatsuba multipliers have been proposed for various type of irreducible polynomials [14], [15], [21], [24], [28]. On one hand, such multipliers cost several more XOR gates delay compared with the fastest bit-parallel multiplier known to date [6], where no divide-and-conquer algorithm is applied. On the other hand, the space complexities of these multipliers are roughly reduced by 1/4.

Empirically, non-recursive Karatsuba multipliers focusing on specific irreducible polynomials usually have better space and time complexity than the ones for general polynomials. Such polynomials include equally-spaced trinomial (EST) [28], all-one polynomial (AOP) [15], etc. Recently, we explore another special form of trinomial $x^m + x^{\frac{m}{3}} + 1$ combined with a three-term Karatsuba algorithm to obtain an efficient bit-parallel multiplier [25]. The proposed multiplier roughly costs 2/3 circuit gates of the fastest multipliers, while its time delay matches the best

known Karatsuba multiplier. In this study, we take inspiration from our previous scheme and investigate the construction of the similar type of multipliers. Consider the $GF(2^m)$ multiplication defined by an irreducible trinomial $x^m + x^k + 1$, where $m = nk, n \geq 2$. We name this type of trinomial as n-spaced trinomial. Obviously, this type of trinomial is EST if $n = 2$. Shou *et al.* [26] have already investigated the development of the bit-parallel multiplier for this trinomial using a n-term Karatsuba algorithm. But their scheme requires 3 more XOR gate delays compared with the fastest one. In this paper, we apply a n-term Karatsuba algorithm along with the shifted polynomial basis (SPB) to simplify the field multiplication. Mastrovito approach is utilized for polynomial reduction. It is demonstrated that the corresponding Mastrovito matrices for different parts of the field multiplication have relatively simpler forms, which lead to an efficient architecture. Moreover, we also give the explicit formulae with respect to the space and time complexity of the corresponding multipliers. As a result, the lower bound of our proposal costs approximately $O(\frac{m^2}{2} + m^{3/2})$ circuit gates compared with the fastest bit-parallel multipliers, while its time delay matches the Karatsuba based multipliers known to date.

The rest of this paper is organized as follows: In Section 2, we briefly review the n-term Karatsuba algorithm, the SPB representation and some pertinent notations. Then, we present a new bit-parallel multiplier architecture for n-spaced trinomial in Section 3. After that, a small example is given. Section 4 presents a comparison between the proposed multiplier and some others. More discussion about the optimal parameter is also given. Finally, some conclusions are drawn.

II. PRELIMINARY

In this section, we briefly review some important notations and related algorithms that used throughout this paper.

A. IRREDUCIBLE n-SPACED TRINOMIAL

We first consider the existence of the irreducible trinomial $x^m + x^k + 1, m = nk$ which are used to define the finite field $GF(2^m)$. The following lemma is useful.

Lemma 1 [2]: Let $f_1(x), f_2(x), \cdots, f_N(x)$ be all the distinct monic irreducible polynomial over $\mathbb{F}_p$ of degree m and order e. Let $t \geq 2$ be an integer whose prime factors divide e but not $\frac{p^m-1}{e}$. Assume also that $p^m \equiv 1 \bmod 4$ if $t \equiv 0 \bmod 4$. Then $f_1(x^t), f_2(x^t), \cdots, f_N(x^t)$ are all the distinct monic irreducible polynomials in $\mathbb{F}_p[x]$ of degree $m \cdot t$ and order $t \cdot e$.

Lemma 1 provides a way to construct an irreducible trinomial of higher degree, i.e., $x^{nk} + x^k + 1$, from the known irreducible trinomial $x^n + x + 1$. If a trinomial $x^n + x + 1$ is irreducible over $\mathbb{F}_2$, one can find an integer k that satisfies the above condition, to construct an irreducible trinomial $x^{nk}+x^k+1$. For example, it is easy to check that both x^3+x+1 and $x^4 + x + 1$ are irreducible. Meanwhile, their orders are 7 and 15, respectively. It follows that $x^{3k} + x^k + 1$ ($k = 7^i$) and $x^{4k} + x^k + 1$ ($k = 3^i \times 5^j$, $i, j \geq 0$) are all irreducible.

B. SHIFTED POLYNOMIAL BASIS

The shifted polynomial basis (SPB) [13] actually is a variation of the polynomial basis. This notion is originally applied in the field $GF(2^m)$ generated with irreducible trinomials, and then pentanomials [5]. In this study, we consider the field $GF(2^m)$ generated by a n-spaced trinomial $f(x) = x^{nk} + x^k + 1$. Let x be a root of $f(x)$, and the set $M = \{x^{nk-1}, \cdots, x, 1\}$ constitutes a polynomial basis (PB). Then, the SPB can be obtained by multiplying the set M by a certain exponentiation of x:

Definition 2 [13]: Let v be an integer and the ordered set $M = \{x^{nk-1}, \cdots, x, 1\}$ be a polynomial basis of $GF(2^m)$ over $\mathbb{F}_2$. The ordered set $x^{-v}M := \{x^{i-v} | 0 \leq i \leq nk-1\}$ is called the shifted polynomial basis with respect to M.

Under SPB representation, the field multiplication can be performed as:

$$C(x)x^{-v} = A(x)x^{-v} \cdot B(x)x^{-v} \bmod f(x).$$

If the parameter v is properly selected, the field multiplication using SPB representation is simpler than that using PB representation, especially for the field define by irreducible trinomial or some type of pentanomials [5]. This characteristic directly lead to a more efficient Mastrovito multiplier which has lower time complexity compared with classic one using PB. Furthermore, it has been proved that the optimal value of v is k or $k-1$ for trinomials [13]. To construct an efficient multiplier for n-spaced trinomials, we choose $v = k$ and use this denotation thereafter.

C. n-TERM KARATSUBA ALGORITHM

The classic Karatsuba algorithm multiplies two 2-term polynomials using three scalar multiplications at the cost of one extra addition. Then, Weimerskirch and Paar [8] proposed a slightly generalized algorithm for the polynomial multiplication with arbitrary degree. This algorithm has the same idea as the classic one. We denote such an algorithm as n-term KA ($n > 2$). Provide that there are two polynomials of degree $n-1$ over $\mathbb{F}_2$:

$$A(x) = \sum_{i=0}^{n-1} a_i x^i, \quad B(x) = \sum_{i=0}^{n-1} b_i x^i.$$

The n-term KA for polynomial multiplication AB is as follows:

- Compute for each $i = 0, \cdots, n-1$,

$$E_i = a_i b_i.$$

- Compute for each $i = 1, \cdots, 2n-3$ and for all s, t with $s + t = i$ and $n > t > s \geq 0$,

$$E_{s,t} = (a_s + a_t)(b_s + b_t).$$

- The coefficients of $D(x) = A(x)B(x) = \sum_{i=0}^{2n-2} d_i x^i$ can be computed as

$$d_0 = E_0,$$
$$d_{2n-2} = E_{n-1},$$
$$d_i = \begin{cases} \sum_{\substack{s+t=i,\\ n>t>s\geq 0}} E_{s,t} + \sum_{\substack{s+t=i,\\ n>t>s\geq 0}} (E_s + E_t) \text{ (odd } i), \\ \sum_{\substack{s+t=i,\\ n>t>s\geq 0}} E_{s,t} + \sum_{\substack{s+t=i,\\ n>t>s\geq 0}} (E_s + E_t) + E_{i/2} \text{ (even } i), \end{cases}$$

where $i = 1, 2, \cdots, 2n-3$.

The correctness proof about above formulae can be found in [8]. Merge the similar items for E_i, $(i = 0, 1, \cdots, n-1)$, $D(x)$ is rewritten as:

$$D(x) = E_{n-1}(x^{2n-2} + \cdots + x^{n-1}) + E_{n-2}(x^{2n-3} + \cdots + x^{n-2}) + \cdots + E_0(x^{n-1} + \cdots + 1) + \sum_{i=1}^{2n-3}\Big(\sum_{\substack{s+t=i,\\ n>t>s\geq 0}} E_{s,t}\Big)x^i. \quad (1)$$

One can easily check that the above formula costs about $O(\frac{n^2}{2})$ coefficient multiplications and $O(\frac{5n^2}{2})$ additions. Compared with classic KA, the n-term KA saves more coefficient multiplications at the expense of more coefficient additions. Besides Weimerskirch and Paar's algorithm, Fan *et al.* [19] and Montgomery [20] proposed more alternative Karatsuba-like formulae. Their formulae aim to decrease as many coefficient multiplications as possible. These formulations usually contain complicated linear combinations of the coefficients, which will lead more gates delay for the bit-parallel architecture. Thus, we prefer to utilize the above algorithm to develop bit-parallel multiplier.

In Section 3, we investigate the construction of non-recursive Karatsuba algorithm using n-term KA for the n-spaced trinomial. Our main strategy is analogous to that in [24], which combines Mastrovito approach and n-term KA. Therefore, some notations pertaining to matrices and vectors are used as well. Note that these notations have already been presented in [9] and [24]. $\mathbf{Z}(i, :)$, $\mathbf{Z}(:, j)$ and $\mathbf{Z}(i, j)$ represent the ith row vector, jth column vector, and the entry with position (i, j) in matrix $\mathbf{Z}$, respectively. $\mathbf{Z}[\circlearrowleft i]$ represents cyclic shift of $\mathbf{Z}$ by upper i rows. $\mathbf{Z}[\Downarrow i]$ represents appending i zero vectors to the top of $\mathbf{Z}$.

III. EFFICIENT MULTIPLIER BASED ON n-TERM KARATSUBA ALGORITHM

In this section, we present an efficient non-recursive Karatsuba multiplier for n-spaced trinomial $x^{nk} + x^k + 1$ using SPB representation. We firstly investigate the structure of the product matrix for polynomial multiplication based on n-term KA. Then, reduced matrices are calculated using Mastrovito approach. Accordingly, we propose the related multiplier architecture. It is shown that corresponding matrix-vector multiplications can be implemented efficiently for n-spaced trinomial. The space and time complexity of the corresponding multiplier is also discussed.

Provide that the finite field $GF(2^m)$ is generated with an irreducible trinomial $x^m + x^k + 1$, $m = nk$, the field elements are represented using SPB. Applying n-term KA as presented previously, we partition two arbitrary field elements $A = \sum_{i=0}^{m-1} a_i x^{i-k}$, $B = \sum_{i=0}^{m-1} b_i x^{i-k}$ into n parts with each part consisting of k bits. More explicitly,

$$A = A_{n-1}x^{(n-2)k} + A_{n-2}x^{(n-3)k} + \cdots + A_1 + A_0 x^{-k},$$
$$B = B_{n-1}x^{(n-2)k} + B_{n-2}x^{(n-3)k} + \cdots + B_1 + B_0 x^{-k},$$

where $A_i = \sum_{j=0}^{k-1} a_{j+(i-1)k} x^j$, $B_i = \sum_{j=0}^{k-1} b_{j+(i-1)k} x^j$, for $i = 0, 1, \cdots, n-1$.

Then, we multiply A and B using the n-term Karatsuba algorithm presented in Section 2 and do following transformation:

$$AB = \Big(E_{n-1}\cdot x^{(n-2)k} + E_{n-2}\cdot x^{(n-3)k} + \cdots + E_1 + E_0\cdot x^{-k}\Big)\cdot h(x) + \sum_{i=1}^{2n-3}\Big(\sum_{\substack{s+t=i,\\ n>t>s\geq 0}} E_{s,t}\Big)x^{ik-2k}, \quad (2)$$

where $h(x) = x^{(n-2)k} + x^{(n-1)k} + \cdots + 1 + x^{-k}$, $E_i = A_i B_i$ $(i = 0, 1, \cdots, n-1)$ and $E_{s,t} = (A_s + A_t)(B_s + B_t)$. We partition the above expression into two parts, i.e.,

$$S_1 = (A_{n-1}B_{n-1}x^{(n-2)k} + \cdots + A_1B_1 + A_0B_0x^{-k})h(x),$$
$$S_2 = \sum_{i=1}^{2n-3}\Big(\sum_{\substack{s+t=i,\\ n>s>t\geq 0}} E_{s,t}\Big)x^{ik-2k},$$

and compute them independently. Thus, the field multiplication $C = AB \bmod f(x)$ now is rewritten as

$$C = (S_1 + S_2) \bmod f(x).$$

In order to apply Mastrovito approach, we have to rewrite both S_1 and S_2 into matrix-vector forms and then reduce those matrices. Please note that $m = nk$ and thus corresponding product matrices are more complicated than those presented in [24] and [25]. The following subsections give the details.

A. COMPUTATION OF S_1 MODULO $f(x)$

Since

$$S_1 = (A_{n-1}B_{n-1}x^{(n-2)k} + \cdots + A_1B_1 + A_0B_0x^{-k})h(x) = A_{n-1}h(x)B_{n-1}x^{(n-2)k} + \cdots + A_0h(x)B_0x^{-k},$$

it is clear that S_1 in fact consists of n parts, each of which can be recognized as a shift of $A_ih(x)B_i$, for $i = 0, 1, \cdots, n-1$. Through constructing the matrix-vector form of $A_ih(x)B_i$, $i = 0, 1, \cdots, n-1$, we can develop the matrix-vector form of S_1. It is noted that

$$A_ih(x)B_i = (A_ix^{(n-2)k} + \cdots + A_i + A_ix^{-k})\cdot B_i.$$

Such an expression can be written as big matrix-vector multiplication derived from the matrix-vector form of A_iB_i. Let $\mathbf{A}_i$

represents the multiplication matrix related to $A_i h(x)$ and $\mathbf{b}_i$ represents the coefficient vector of $B_i(x)$. Then, $A_i h(x) B_i = \mathbf{A}_i \cdot \mathbf{b}_i$, where

$$\mathbf{A}_i = \begin{matrix} -k \\ \vdots \\ \vdots \\ nk-1 \end{matrix} \begin{bmatrix} \mathbf{A}_{i,L} \\ \left.\begin{matrix} \mathbf{A}_{i,L} + \mathbf{A}_{i,H} \\ \vdots \\ \mathbf{A}_{i,L} + \mathbf{A}_{i,H} \end{matrix}\right\} n-1 \\ \mathbf{A}_{i,H} \end{bmatrix}.$$

The labels on the left side indicate the exponent of indeterminate x for each row in $\mathbf{A}_i$, which range from $-k$ to $nk-1$. However, we check that the degrees of x in $A_i h(x) B_i$ are actually in the range $[-k, nk-2]$. But the last row of $\mathbf{A}_i$ is $\mathbf{0}$, which does not affect the result. The matrices $\mathbf{A}_{i,H}$ and $\mathbf{A}_{i,L}$ are both $k \times k$ triangular Toeplitz matrix, i.e.,

$$\mathbf{A}_{i,L} = \begin{bmatrix} a_{ik+0} & 0 & \cdots & 0 \\ a_{ik+1} & a_{ik+0} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{ik+k-1} & a_{ik+k-2} & \cdots & a_{ik+0} \end{bmatrix},$$

and

$$\mathbf{A}_{i,H} = \begin{bmatrix} 0 & a_{ik+k-1} & \cdots & a_{ik+1} \\ 0 & 0 & \cdots & a_{ik+2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{ik+k-1} \\ 0 & 0 & \cdots & 0 \end{bmatrix},$$

for $i = 0, 1, \cdots, n-1$.

Accordingly, these n submatrix-vector multiplications can constitute a bigger matrix-vector multiplication pertaining to S_1, denoted by $\mathbf{A}_{S_1} \cdot \mathbf{b}$. More explicitly,

$$\begin{aligned} S_1 &= \mathbf{A}_{S_1} \cdot \mathbf{b} = \mathbf{A}_{S_1} \cdot [\mathbf{b}_0, \mathbf{b}_1, \cdots, \mathbf{b}_{n-1}]^T \\ &= \begin{bmatrix} \mathbf{A}_0 & \mathbf{0}_{k\times k} & \cdots & \mathbf{0}_{k\times k} \\ \mathbf{0}_{k\times k} & \mathbf{A}_1 & \ddots & \mathbf{0}_{(m-2k)\times k} \\ \mathbf{0}_{(m-2k)\times k} & \mathbf{0}_{(m-2k)\times k} & \cdots & \mathbf{A}_{n-1} \end{bmatrix} \\ &\quad \times \begin{bmatrix} \mathbf{b}_0, \\ \vdots \\ \mathbf{b}_{n-1} \end{bmatrix}. \end{aligned} \tag{3}$$

For simplicity, we do not write the degree labels of the product matrix here. Notice that $\deg(A_i B_i h) = nk-2$, $i = 0, 1, \cdots, n-1$, we have $\deg(S_1) = nk-2+(n-2)k = 2m-2k-2$. One can check that the degrees of the terms of S_1 are in the range $[-2k, 2m-2k-2]$. Based on Mastrovito scheme, S_1 needs a further reduction by $f(x)$. The following reduction rule is applied:

$$\begin{cases} x^i = x^{m+i} + x^{i+k}, \text{ for } i = -2k, \cdots, -k-1; \\ x^i = x^{i-m} + x^{i-m+k}, \text{ for } i = m-k, m-k+1, \\ \qquad \cdots, 2m-2k-2. \end{cases} \tag{4}$$

The reduction can be regarded as the construction of the Mastrovito matrix from $\mathbf{A}_{S_1}$ according to (4). Let $\mathbf{M}_{S_1}$ denote the Mastrovito matrix related to S_1. In order to analyze the organization of $\mathbf{M}_{S_1}$, we introduce a lemma, which is the key step toward the development of the multiplier architecture.

Lemma 3: Provide that $\mathbf{A}$ *is an arbitrary* $(2m-1) \times m$ *matrix and* $\mathbf{b}$ *is a* $m \times 1$ *vector over* $\mathbb{F}_2$. *The Mastrovito matrix* $\mathbf{M}$ *related to* $\mathbf{A} \cdot \mathbf{b}$ *modulo* $x^m + x^k + 1$ *using (4) can be obtained as follows:*

$$\mathbf{M} = \mathbf{M}_1 + \mathbf{M}_2,$$

where

$$\begin{aligned} \mathbf{M}_1 = [&\mathbf{A}(1,:)^T + \mathbf{A}(m+1,:)^T, \mathbf{A}(2,:)^T + \mathbf{A}(m+2,:)^T, \\ &\cdots, \mathbf{A}(2m-1,:)^T + \mathbf{A}(m-1,:)^T, \mathbf{A}(m,:)^T][\circlearrowleft k], \end{aligned} \tag{5}$$

and

$$\begin{aligned} \mathbf{M}_2 = [&\mathbf{A}(1,:)^T, \cdots, \mathbf{A}(k,:)^T, \mathbf{A}(k+m+1,:)^T, \\ &\cdots, \mathbf{A}(2m-1,:)^T, \mathbf{0}]. \end{aligned} \tag{6}$$

Proof: We notice that the product matrix $\mathbf{A}$ here includes $2m-1$ rows with each row corresponding the degree from $-2k$ to $2m-2k-2$. Clearly, the first k rows and the last $m-k-1$ rows correspond to the term degrees that are out of the range $[-k, m-k-1]$. Based on (4), the reduction steps consist of reducing the row $\{-2k, -2k+1, \cdots, -k-1\}$ by adding them to the row $\{-k, \cdots, -1\}$ and $\{m-2k, \cdots, m-k-1\}$, and reducing the row $\{m-k, \cdots, 2m-2k-2\}$ by adding them to the row $\{0, \cdots, m-k-2\}$ and $\{-k, \cdots, m-2k-2\}$. The explicit reduction process follows the same line as the proof of Observation 3.1, [24]. Then, we partition these rows into two categories, let

$$\begin{aligned} \mathbf{M}_1 = [&\mathbf{A}(k+m+1,:)^T + \mathbf{A}(k+1,:)^T, \cdots, \\ &\mathbf{A}(2m-1,:)^T + \mathbf{A}(m-1,:)^T, \mathbf{A}(m,:)^T, \mathbf{A}(1,:)^T \\ &+ \mathbf{A}(m+1,:)^T \cdots, \mathbf{A}(k,:)^T + \mathbf{A}(k+m,:)^T], \end{aligned}$$

and

$$\begin{aligned} \mathbf{M}_2 = [&\mathbf{A}(1,:)^T, \cdots, \mathbf{A}(k,:)^T, \mathbf{A}(k+m+1,:)^T, \\ &\cdots, \mathbf{A}(2m-1,:)^T, \mathbf{0}]. \end{aligned}$$

We compare the row number and obtain the result immediately. □

Based on Lemma 3, we immediately give the following proposition with respect to the structure of $\mathbf{M}_{S_1}$.

Proposition 4: The Mastrovito matrix $\mathbf{M}_{S_1}$ *can be constructed as*

$$\mathbf{M}_{S_1} = \mathbf{M}_{S_1,1} + \mathbf{M}_{S_1,2},$$

where

$$\mathbf{M}_{S_1,1} = \begin{bmatrix} \mathbf{A}_{0,L} + \mathbf{A}_{0,H} & \mathbf{A}_{1,L} + \mathbf{A}_{1,H} & \cdots & \mathbf{A}_{n-1,L} + \mathbf{A}_{n-1,H} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{0,L} + \mathbf{A}_{0,H} & \mathbf{A}_{1,L} + \mathbf{A}_{1,H} & \cdots & \mathbf{A}_{n-1,L} + \mathbf{A}_{n-1,H} \end{bmatrix},$$

and

$$\mathbf{M}_{S_1,2} = \begin{bmatrix} \mathbf{A}_{0,L} & \mathbf{0}_{k\times k} & \mathbf{0}_{k\times k} & \cdots & \mathbf{0}_{k\times k} \\ \mathbf{0}_{k\times k} & \mathbf{A}_{1,H} & \mathbf{A}_{2,L}+\mathbf{A}_{2,H} & \cdots & \mathbf{A}_{n-1,L}+\mathbf{A}_{n-1,H} \\ \mathbf{0}_{k\times k} & \mathbf{0}_{k\times k} & \mathbf{A}_{2,H} & \cdots & \mathbf{A}_{n-1,L}+\mathbf{A}_{n-1,H} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{k\times k} & \mathbf{0}_{k\times k} & \mathbf{0}_{k\times k} & \cdots & \mathbf{A}_{n-1,H} \end{bmatrix}.$$

Proof: The proof is the same as the proof of Lemma 3. We directly get this conclusion by substituting $\mathbf{A}$ by $\mathbf{A}_{S_1}$. □

It is noted that there are some overlapped terms between $\mathbf{M}_{S_1,1}$ and $\mathbf{M}_{S_1,2}$. By adding these two matrices together, we can obtain the explicit form of $\mathbf{M}_{S_1}$, which is in (7), as shown at the bottom of this page. Moreover, the matrix-vector multiplication $S_1 = \mathbf{M}_{S_1} \cdot \mathbf{b}$ can be computed according to the strategy used in [25] and overlapped terms are considered reusing to save more logic gates.

1) DETAILED COMPUTATION OF S_1 MODULO $f(x)$

(i) Perform $2n$ row-vector products

$$\mathbf{A}_{0,L} * \mathbf{b}_0, \mathbf{A}_{0,H} * \mathbf{b}_0, \mathbf{A}_{1,L} * \mathbf{b}_1, \mathbf{A}_{1,H} * \mathbf{b}_1, \cdots \quad \mathbf{A}_{n-1,L} * \mathbf{b}_{n-1}, \mathbf{A}_{n-1,H} * \mathbf{b}_{n-1}, \quad (8)$$

in parallel. The symbol "*" represent only row-vector product related to $\mathbf{A}_{i,L}$ (or $\mathbf{A}_{i,H}$) and $\mathbf{b}_i$, $i = 0, 1, \cdots, n-1$. For instance, $\mathbf{A}_{0,H} * \mathbf{b}_0$ represents computing the products

$$[\mathbf{A}_{0,H}(i,1) \cdot b_0, \cdots, \mathbf{A}_{0,H}(i,k) \cdot b_{k-1}],$$

for $i = 1, 2, \cdots, k$ in parallel.

(ii) Compute

$$\mathbf{A}_{0,L}\mathbf{b}_0 + \mathbf{A}_{0,H}\mathbf{b}_0, \cdots, \mathbf{A}_{n-1,L}\mathbf{b}_{n-1} + \mathbf{A}_{n-1,H}\mathbf{b}_{n-1}$$

using binary XOR trees in parallel. Meanwhile, $\mathbf{A}_{0,H}\mathbf{b}_0, \mathbf{A}_{1,L}\mathbf{b}_1, \cdots, \mathbf{A}_{n-1,L}\mathbf{b}_{n-1}$ are computed using *sub-expression sharing* technique.

(iii) Sum up all the n entries of each row using binary XOR tree to obtain the final result.

Remark: It is clear that the row-vector products in (8) contain all the possible row-vector products in (7). Only nk^2 AND gates are required to compute these expressions.

In addition, $\mathbf{A}_{i,L}\mathbf{b}_i + \mathbf{A}_{i,H}\mathbf{b}_i$, $(i = 0, \cdots, n-1)$ contain all the terms of $\mathbf{A}_{i,L}\mathbf{b}_i$ or $\mathbf{A}_{i,H}\mathbf{b}_i$. These expressions can be computed in parallel and more logic gates can be saved using *sub-expression sharing* for binary tree. Such an approach has already been studied in [24]. The authors have shown that if two binary XOR trees share t common items, only $t - W(t)$ XOR gates can be saved, where $W(t)$ is the Hamming weight of t. It is easy to check that the j-th row $(j = 1, 2, \cdots, k)$ of $\mathbf{A}_{i,L}\mathbf{b}_i$ shares j terms with $\mathbf{A}_{i,L}\mathbf{b}_i + \mathbf{A}_{i,L}\mathbf{b}_i$ for $i = 1, 2, \cdots, n-1$. Meanwhile, the j-th row of $\mathbf{A}_{i,L}\mathbf{b}_i$ includes j terms and originally requires $j-1$ XOR gates for binary XOR tree. Minus the saved XOR gates, we can see that number of required XOR actually is $j - 1 - (j - W(j)) = W(j) - 1$. Specifically, the k-th row of $\mathbf{A}_{i,L}\mathbf{b}_i$ is identical to that of $\mathbf{A}_{i,L}\mathbf{b}_i + \mathbf{A}_{i,L}\mathbf{b}_i$, no XOR gates is needed here. Based on similar approach, we can calculate the real number of XOR gates for the j-th row of $\mathbf{A}_{0,H}\mathbf{b}_0$ is $W(k-j) - 1$ for $j = 1, 2, \cdots, k-1$.

TABLE 1. Space and time complexities of S_1 mod $f(x)$.

Operation	# AND	#XOR	Delay
$\mathbf{A}_{i,L}*\mathbf{b}_i, \mathbf{A}_{i,H}*\mathbf{b}_i$	nk^2	-	T_A
$\mathbf{A}_{i,L}\mathbf{b}_i + \mathbf{A}_{i,H}\mathbf{b}_i$ $(i=0,1,\cdots,n-1)$	-	$n(k^2-k)$	$\lceil \log_2 k \rceil T_X$
$\mathbf{A}_{0,H}\mathbf{b}_0, \mathbf{A}_{i,L}\mathbf{b}_i$ $(i=1,2,\cdots,n-1)$	-	$n\sum_{i=1}^{k-1} W(i)$ $+n-nk$	
S_1	-	$m+kW(n-2)$ $+k\sum_{i=0}^{n-2} W(i)$	$\lceil \log_2 n \rceil T_X$

Table 1 summarizes the space and time complexity of S_1 mod $f(x)$ for all the steps. One can notice that after calculation of the row-vector products in (8), each row of $\mathbf{A}_{i,L} * \mathbf{b}_i + \mathbf{A}_{i,H} * \mathbf{b}_i$ consists of k terms. Thus, the inner product of $\mathbf{A}_{i,L}\mathbf{b}_i + \mathbf{A}_{i,H}\mathbf{b}_i$ will be obtained using a binary XOR tree with a delay of $\lceil \log_2 k \rceil T_X$. Finally, we have to perform additions among the n entries to obtain the coefficient vector with respect to S_1. More partial additions can be saved using the same sub-expression sharing. For simplicity, we put the details to the Appendix A.

2) AN EXAMPLE OF S_1 mod $f(x)$

Firstly we have an irreducible 4-spaced trinomial $x^4 + x + 1$ over $\mathbb{F}_2$. Then, we can construct another irreducible 4-spaced trinomial of higher degree according to Lemma 4, i.e., $x^{12} + x^3 + 1$.

$$S_1 \bmod f(x) = \mathbf{M}_{S_1} \cdot \mathbf{b} \times \begin{bmatrix} \mathbf{A}_{0,H} & \mathbf{A}_{1,L}+\mathbf{A}_{1,H} & \mathbf{A}_{2,L}+\mathbf{A}_{2,H} & \cdots & \mathbf{A}_{n-2,L}+\mathbf{A}_{n-2,H} & \mathbf{A}_{n-1,L}+\mathbf{A}_{n-1,H} \\ \mathbf{A}_{0,L}+\mathbf{A}_{0,H} & \mathbf{A}_{1,L} & \mathbf{0}_{k\times k} & \cdots & \mathbf{0}_{k\times k} & \mathbf{0}_{k\times k} \\ \mathbf{A}_{0,L}+\mathbf{A}_{0,H} & \mathbf{A}_{1,L}+\mathbf{A}_{1,H} & \mathbf{A}_{2,L} & \cdots & \mathbf{0}_{k\times k} & \mathbf{0}_{k\times k} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{A}_{0,L}+\mathbf{A}_{0,H} & \mathbf{A}_{1,L}+\mathbf{A}_{1,H} & \mathbf{A}_{2,L}+\mathbf{A}_{2,H} & \cdots & \mathbf{A}_{n-2,L} & \mathbf{0}_{k\times k} \\ \mathbf{A}_{0,L}+\mathbf{A}_{0,H} & \mathbf{A}_{1,L}+\mathbf{A}_{1,H} & \mathbf{A}_{2,L}+\mathbf{A}_{2,H} & \cdots & \mathbf{A}_{n-2,L}+\mathbf{A}_{n-2,H} & \mathbf{A}_{n-1,L} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_{n-2} \\ \mathbf{b}_{n-1} \end{bmatrix}. \quad (7)$$

$$\begin{bmatrix} \mathbf{A}_{0,H} & \mathbf{A}_{1,L}+\mathbf{A}_{1,H} & \mathbf{A}_{2,L}+\mathbf{A}_{2,H} & \mathbf{A}_{3,L}+\mathbf{A}_{3H} \\ \mathbf{A}_{0,L}+\mathbf{A}_{0,H} & \mathbf{A}_{1,L} & \mathbf{0}_{k\times k} & \mathbf{0}_{k\times k} \\ \mathbf{A}_{0,L}+\mathbf{A}_{0,H} & \mathbf{A}_{1,L}+\mathbf{A}_{1,H} & \mathbf{A}_{2,L} & \mathbf{0}_{k\times k} \\ \mathbf{A}_{0,L}+\mathbf{A}_{0,H} & \mathbf{A}_{1,L}+\mathbf{A}_{1,H} & \mathbf{A}_{2,L}+\mathbf{A}_{2,H} & \mathbf{A}_{3,L} \end{bmatrix}$$

$$= \left[\begin{array}{ccc:ccc:ccc:ccc} 0 & a_2 & a_1 & a_3 & a_5 & a_4 & a_6 & a_8 & a_7 & a_9 & a_{11} & a_{10} \\ 0 & 0 & a_2 & a_4 & a_3 & a_5 & a_7 & a_6 & a_8 & a_{10} & a_9 & a_{11} \\ 0 & 0 & 0 & a_5 & a_4 & a_3 & a_8 & a_7 & a_6 & a_{11} & a_{10} & a_9 \\ \hdashline a_0 & a_2 & a_1 & a_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_1 & a_0 & a_2 & a_4 & a_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_2 & a_1 & a_0 & a_5 & a_4 & a_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hdashline a_0 & a_2 & a_1 & a_3 & a_5 & a_4 & a_6 & 0 & 0 & 0 & 0 & 0 \\ a_1 & a_0 & a_2 & a_4 & a_3 & a_5 & a_7 & a_6 & 0 & 0 & 0 & 0 \\ a_2 & a_1 & a_0 & a_5 & a_4 & a_3 & a_8 & a_7 & a_6 & 0 & 0 & 0 \\ \hdashline a_0 & a_2 & a_1 & a_3 & a_5 & a_4 & a_6 & a_8 & a_7 & a_9 & 0 & 0 \\ a_1 & a_0 & a_2 & a_4 & a_3 & a_5 & a_7 & a_6 & a_8 & a_{10} & a_9 & 0 \\ a_2 & a_1 & a_0 & a_5 & a_4 & a_3 & a_8 & a_7 & a_6 & a_{11} & a_{10} & a_9 \end{array}\right].$$

Consider the field multiplication using SPB representation over $GF(2^{12})$ defined by the previous trinomial. We have the SPB parameter $k = 3$ and SPB is defined as $\{x^{-3}, x^{-2}, \cdots, x^7, x^8\}$. Assume that $A = \sum_{i=0}^{11} a_i x^{i-3}$ and $B = \sum_{i=0}^{11} b_i x^{i-3}$ are two elements in $GF(2^{12})$. A, B are partitioned as

$$A = A_3x^6 + A_2x^3 + A_1 + A_0x^{-3},$$
$$B = B_3x^6 + B_2x^3 + B_1 + B_0x^{-3},$$

where $A_i = a_{2+3i}x^2 + a_{1+3i}x + a_{0+3i}$, $B_i = b_{2+3i}x^2 + b_{1+3i}x + b_{0+3i}$, for $i = 0, 1, 2, 3$.

Based on equation (2) and previous description, it is obviously that $AB = S_1 + S_2$ and the explicit form of S_1 and S_2 are as follows:

$$S_1 = A_3B_3hx^6 + A_2B_2hx^3 + A_1B_1h + A_0B_0hx^{-3},$$
$$S_2 = E_{3,2}x^9 + E_{3,1}x^6 + (E_{3,0} + E_{2,1})x^3 + E_{2,0} + E_{1,0}x^{-3},$$

where $h(x) = x^6 + x^3 + 1 + x^{-3}$ and $E_{s,t} = (A_s + A_t)(B_s + B_t)$ for $3 \geq s > t \geq 0$. Let

$$\mathbf{A}_{i,L} = \begin{bmatrix} a_{3i+0} & 0 & 0 \\ a_{3i+1} & a_{3i+0} & 0 \\ a_{3i+2} & a_{3i+1} & a_{3i+0} \end{bmatrix},$$

and

$$\mathbf{A}_{i,H} = \begin{bmatrix} 0 & a_{3i+2} & a_{3i+1} \\ 0 & 0 & a_{3i+2} \\ 0 & 0 & 0 \end{bmatrix}.$$

Accordingly, it is easy to compute the matrices $\mathbf{A}_{S_1}$, $\mathbf{M}_{S_1,1}$ and $\mathbf{M}_{S_1,2}$, which are presented in the appendix. The Mastrovito matrix related to $S_1 \bmod f(x)$ is as shown at the top of this page.

Therefore, one can check that the exact number of logic gates requried by every step of $S_1 \bmod f(x)$:

- Computation of $\mathbf{A}_{0,L} * \mathbf{b}_0, \mathbf{A}_{0,H} * \mathbf{b}_0, \cdots, \mathbf{A}_{3,L} * \mathbf{b}_3, \mathbf{A}_{3,H} * \mathbf{b}_3$ requires 36 AND gates with one T_A gate delay.
- Computation of $\mathbf{A}_{0,L}\mathbf{b}_0 + \mathbf{A}_{0,H}\mathbf{b}_0, \cdots, \mathbf{A}_{3,L}\mathbf{b}_3 + \mathbf{A}_{3,H}\mathbf{b}_3$ costs 24 XOR gates in all. Meanwhile, no XOR gates are needed for the computation of $\mathbf{A}_{0,H}\mathbf{b}_0, \mathbf{A}_{1,L}\mathbf{b}_1, \mathbf{A}_{2,L}\mathbf{b}_2, \mathbf{A}_{3,L}\mathbf{b}_3$ using *sub-expression sharing*, as the binary XOR tree for these expressions can be embedded into those of $\mathbf{A}_{i,L}\mathbf{b}_i + \mathbf{A}_{i,H}\mathbf{b}_i$ for $i = 0, 1, 2, 3$. These operations requires $2T_X$ delay in parallel.
- The final additions among 4 entries of each row costs 21 XOR gates using the trick presented in the appendix, which cost another $2T_X$ delay in parallel.

As a result, the calculation of $\mathbf{M}_{S_1,1} \cdot \mathbf{b}$ totally requires 36 AND gates and 45 XOR gates, with $T_A + 4T_X$ gate delay. This result meets the complexity formulae shown in Table 1.

B. COMPUTATION OF S_2 MODULO $x^{nk} + x^k + 1$

We then consider the computation of $S_2 \bmod f(x)$ in details. Note that

$$S_2 = \sum_{i=1}^{2n-3} \Big(\sum_{\substack{s+t=i, \\ n>s>t\geq 0}} E_{s,t} \Big) x^{ik-2k},$$

and $E_{s,t} = (A_s + A_t)(B_s + B_t), (n > s > t \geq 0)$ consist of k bits. Each of these expressions can be recognized as

a small matrix-vector multiplication. Let $\sum_{i=0}^{k-1} u_i^{(s,t)} x^i$ and $\sum_{i=0}^{k-1} v_i^{(s,t)} x^i$ denote the result of $A_s + A_t$ and $B_s + B_t$, respectively. We have $E_{s,t} = \mathbf{U}_{s,t} \cdot \mathbf{v}_{s,t}$, where $\mathbf{U}_{s,t}$ is the product matrix constructed from $\sum_{i=0}^{k-1} u_i^{(s,t)} x^i$ and $\mathbf{v}_{s,t}$ is the coefficient vector $[v_0^{(s,t)}, v_1^{(s,t)}, \cdots, v_{k-1}^{(s,t)}]^T$, i.e.,

$$E_{s,t} = \mathbf{U}_{s,t} \cdot \mathbf{v}_{s,t} = \begin{bmatrix} u_0^{(s,t)} & 0 & \cdots & 0 & 0 \\ u_1^{(s,t)} & u_0^{(s,t)} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ u_{k-2}^{(s,t)} & u_{k-3}^{(s,t)} & \cdots & u_0^{(s,t)} & 0 \\ u_{k-1}^{(s,t)} & u_{k-2}^{(s,t)} & \cdots & u_1^{(s,t)} & u_0^{(s,t)} \\ 0 & u_{k-1}^{(s,t)} & \cdots & u_2^{(s,t)} & u_1^{(s,t)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & u_{k-1}^{(s,t)} & u_{k-2}^{(s,t)} \\ 0 & 0 & \cdots & 0 & u_{k-1}^{(s,t)} \end{bmatrix} \cdot \begin{bmatrix} v_0^{(s,t)} \\ v_1^{(s,t)} \\ \vdots \\ v_{k-2}^{(s,t)} \\ v_{k-1}^{(s,t)} \end{bmatrix}. \tag{9}$$

It is noted that these matrix-vector multiplications are independent and thus can be implemented in parallel. However, S_2 contains $\binom{n}{2}$ different expressions in all, each of which has a different degree. In order to simplify the reduction process, we first classify these expressions into several categories, where the expressions in the same category can constitute a bigger matrix-vector multiplication. Then we can perform a reduction with each category. In addition, the classification has already been studied in [23]. Here, we can utilize the result directly. Let

$$S(n) = S_2 \cdot x^{2k} = \sum_{i=1}^{2n-3} \Big(\sum_{\substack{s+t=i, \\ n>s>t\geq 0}} E_{s,t} \Big) x^{ik}.$$

The classification lemma is as follows:

Lemma 5 [23]: $S(n)$ can be expressed as the plus of $g_1 x^{(2\lambda-1)k}, g_2 x^{(2\lambda-3)k}, \cdots, g_\lambda x^k$ for $\lambda = \frac{n}{2}$ (n is even) or $\lambda = \frac{n-1}{2}$ (n is odd), where

$$\begin{aligned} g_1 &= C_{n-2}^{(1)} x^{(n-2)k} + C_{n-3}^{(1)} x^{(n-3)k} + \cdots + C_0^{(1)}, \\ g_2 &= C_{n-2}^{(2)} x^{(n-2)k} + C_{n-3}^{(2)} x^{(n-3)k} + \cdots + C_0^{(2)}, \\ &\vdots \\ g_{\frac{n}{2}} &= C_{n-2}^{(\frac{n}{2})} x^{(n-2)k} + C_{n-3}^{(\frac{n}{2})} x^{(n-3)k} + \cdots + C_0^{(\frac{n}{2})}, \end{aligned}$$

or

$$\begin{aligned} g_1 &= C_{n-1}^{(1)} x^{(n-1)k} + C_{n-2}^{(1)} x^{(n-2)k} + \cdots + C_0^{(1)}, \\ g_2 &= C_{n-1}^{(2)} x^{(n-1)k} + C_{n-2}^{(2)} x^{(n-2)k} + \cdots + C_0^{(2)}, \\ &\vdots \\ g_{\frac{n-1}{2}} &= C_{n-1}^{(\frac{n-1}{2})} x^{(n-1)k} + C_{n-2}^{(\frac{n-1}{2})} x^{(n-2)k} + \cdots + C_0^{(\frac{n-1}{2})}, \end{aligned}$$

where $C_j^{(i)} \in \{E_{s,t}\}$, $n > s > t \geq 0$.

Proof: See [23, Sec. 3.2]. □

Based on the above lemma, it is obvious that S_2 can be partitioned into λ parts and all these parts are independent. More explicitly, $S_2 = g_1 x^{(2\lambda-3)k} + g_2 x^{(2\lambda-5)k} + \cdots + g_\lambda x^{-k}$. Obviously, $g_1, g_2, \cdots, g_\lambda$ contain all the nonzero terms of S_2, where the number of such terms equals $(n-2)k + 2k - 2 + 1 = m - 1$ terms if n is even or $(n-1)k + 2k - 2 + 1 = m + k - 1$ if n is odd. We can first compute these expressions in parallel, then, perform reductions related to $g_1 x^{(2\lambda-3)k}, g_2 x^{(2\lambda-5)k}, \cdots, g_\lambda x^{-k}$.

1) DETAILED COMPUTATION OF $S_2 \bmod f(x)$

(i) Perform bitwise addition $A_s + A_t, B_s + B_t, (n > s > t \geq 0)$ in parallel.
(ii) Perform $\binom{n}{2}$ matrix-vector bitwise multiplications, i.e, $\mathbf{E}_{s,t} = \mathbf{U}_{s,t} * \mathbf{v}_{s,t}$ in parallel.
(iii) Classify these $\binom{n}{2}$ matrices $\mathbf{E}_{s,t}$ into λ parts according to Lemma 5 and constitute the small matrices of the same category into λ big matrices $\mathbf{E}_{g_1}, \cdots, \mathbf{E}_{g_\lambda}$, which correspond to $g_1, g_2, \cdots, g_\lambda$.
(iv) Add all the entries of the same row in $\mathbf{E}_{g_1}, \cdots, \mathbf{E}_{g_\lambda}$ using binary XOR tree, and obtain the coefficients of $g_1, g_2, \cdots, g_\lambda$.
(v) Perform reduction for $g_1 x^{(2\lambda-3)k}, g_2 x^{(2\lambda-5)k}, \cdots, g_\lambda x^{-k}$ modulo $f(x)$ using (4).
(vi) Add all these results binary XOR tree to obtain the $S_2 \bmod f(x)$.

Remark: According to (9), it is clear that after performing bitwise multiplication, $\mathbf{E}_{s,t}$ are all $(2k-1) \times k$ matrices. When we classify these matrices and constitute them to λ big matrices, one can check that the number of entries for each row of $\mathbf{E}_{g_1}, \cdots, \mathbf{E}_{g_\lambda}$ is equal to k. Thus, the coefficients of $g_1, g_2, \cdots, g_\lambda$ will be obtained with $\lceil \log_2 k \rceil T_X$ delay. Whereafter, we can perform the modular reduction for $g_1 x^{(2\lambda-3)k}, g_2 x^{(2\lambda-5)k}, \cdots, g_\lambda x^{-k}$. Such reductions also rely on equation (4). We have following observations for the computation of $S_2 \bmod f(x)$.

Observation 6: To compute $g_1 x^{(2\lambda-3)k}, g_2 x^{(2\lambda-5)k}, \cdots, g_\lambda x^{-k}$ modulo $f(x)$, we only need to reduce these expressions at most once.

Proof: Apparently, the minimal and maximal degrees of the terms in $g_1 x^{(2\lambda-3)k}, g_2 x^{(2\lambda-5)k}, \cdots, g_\lambda x^{-k}$ are $-k$ and $2m - 3k - 2$, respectively. Apply reducing formulae of (4), we have

$$\begin{aligned} x^{m-k} &= x^0 + x^{-k}, \\ x^{m-k+1} &= x^1 + x^{-k+1}, \\ &\vdots \\ x^{2m-3k-2} &= x^{m-2k-2} + x^{m-3k-2}. \end{aligned}$$

The exponents of x in the right side now are all in the range $[-k, m-k-1]$, no further reduction is needed. □

Observation 7: When the modular reduction and addition are combined, Step (v) and (vi) can be calculated with at most $\lceil \log_2 n \rceil T_X$ delay.

Proof: We know $g_1 x^{(2\lambda-3)k}, g_2 x^{(2\lambda-5)k}, \cdots, g_\lambda x^{-k}$ modulo $f(x)$ only need to reduced once. But, g_i contains different number of nonzero terms according to the parity of n, which lead to different reduction formulations.

For simplicity, we only consider the case of odd n here and put the analysis about other case in Appendix.

If n is odd, we have $\lambda = \frac{n-1}{2}$, and the degree of g_i is $nk + k - 2$. Let $g_i = \sum_{j=0}^{nk+k-2} h_j^{(i)} x^j$. Then,

$$g_i x^{(n-2i-2)k} = \sum_{j=0}^{2ik+k-1} h_j^{(i)} x^{j+(n-2i-2)k} + \sum_{j=2ik+k}^{nk-1} h_j^{(i)} x^{j+(n-2i-2)k} + \sum_{j=nk}^{nk+k-2} h_j^{(i)} x^{j+(n-2i-2)k},$$

for $i = 1, 2, \cdots, \frac{n-1}{2}$. When we reduce above expression modulo $f(x) = x^{nk} + x^k + 1$, only two parts are needed to be reduced. Then,

$$\sum_{j=2ik+k}^{nk-1} h_j^{(i)} x^{j+(n-2i-2)k} = \sum_{j=2ik+k}^{nk-1} h_j^{(i)} (x^{j-2ik-2k} + x^{j-2ik-k}),$$
$$\sum_{j=nk}^{nk+k-2} h_j^{(i)} x^{j+(n-2i-2)k} = \sum_{j=nk}^{nk+k-2} h_j^{(i)} (x^{j-2ik-2k} + x^{j-2ik-k}).$$

By combining non-overlapped parts of above expressions, the result of $g_i x^{(n-2i-2)k} \bmod f(x)$ is given by

$$g_i x^{(n-2i-2)k} \bmod f(x) = p_1^{(i)}(x) + p_2^{(i)}(x) + p_3^{(i)}(x),$$

where

$$p_1^{(i)} = \sum_{j=0}^{2ik+k-1} h_j^{(i)} x^{j+nk-2ik-2k} + \sum_{j=2ik+k}^{nk-1} h_j^{(i)} x^{j-2ik-2k},$$
$$p_2^{(i)} = \sum_{j=2ik+k}^{nk+k-2} h_j^{(i)} x^{j-2ik-k},$$
$$p_3^{(i)} = \sum_{j=nk}^{nk+k-2} h_j^{(i)} x^{j-2ik-2k}.$$

Moreover, it is noted that the term degrees of $p_3^{(i)}$ are in the range $[nk - 2ik - 2k, nk - 2ik - k - 2]$. One can check that these ranges are $[-k, -2], [k, 2k - 2], \cdots, [(n-4)k, (n-3)k - 2]$. Therefore, there is no overlapped term among all the $p_3^{(i)}$, which cost no XOR gates to add them up. Denoted by r the addition of $p_3^{(1)}, p_3^{(2)}, \cdots, p_3^{(\frac{n-1}{2})}$.

Consequently, to obtain $S_2 \bmod f(x)$, we only need to add $p_1^{(1)}, p_2^{(1)}, \cdots, p_1^{(\frac{n-1}{2})}, p_2^{(\frac{n-1}{2})}$ and r in parallel, which cost $\lceil \log_2 n \rceil$ XOR gate delay. We directly conclude the observation. □

We next analyze the space and time complexity related to S_2. Firstly, $2k \cdot \binom{n}{2} = (n^2 - n)k$ XOR gates are needed for pre-computation of $A_s + A_t$ and $B_s + B_t, (n > t > s \geq 0)$ in Step (i), which cost one T_X in parallel. Then, the $\binom{n}{2}$ matrix-vector bitwise multiplications in Step (ii) cost $k^2 \cdot \binom{n}{2} = (n^2 - n)k^2/2$ AND gates with T_A gate delay.

The classification in Step (iii) does not cost any logic gates. Step (iv) includes adding all the entries of the same row in $\mathbf{E}_{g_1}, \cdots, \mathbf{E}_{g_\lambda}$. Since these matrices are determined by $g_1, g_2, \cdots, g_\lambda$, the required XOR gates varies according to parity of n. If n is even, each of $g_1, g_2, \cdots, g_{\frac{n}{2}}$ consists of $n-1$ sub-polynomials. That is to say, $\mathbf{E}_{g_i}, (i = 1, 2, \cdots, \frac{n}{2}$ corresponds a combination of $n-1$ matrices $\mathbf{E}_{s,t}$. Thus the coefficient computation for each g_i costs $nk^2 - k^2 - m + 1$ XOR gates with $\lceil \log_2 k \rceil T_X$ delay. If n is odd, $g_1, g_2, \cdots, g_{\frac{n-1}{2}}$ consists of n sub-polynomials. Similarly, it costs $nk^2 - k - m + 1$ XOR gates for each g_i with $\lceil \log_2 k \rceil T_X$ delay.

Step (v) and (vi) follow the description in Observation 3.2.2. We only add n (or $n-1$) vectors together to obtain $S_2 \bmod f(x)$. The space and time complexity for all the steps is stated in Table 2.

TABLE 2. Space and time complexities of $S_2 \bmod f(x)$.

Operation	# AND	#XOR	Time delay
$A_s + A_t$	-	$\frac{(n^2-n)k}{2}$	T_X
$B_s + B_t$	-	$\frac{(n^2-n)k}{2}$	
$\mathbf{U}_{s,t} * \mathbf{v}_{s,t}$	$\frac{(n^2-n)k^2}{2}$	-	T_A
Step (iv)	-	$\frac{m^2-km-mn+n}{2}$	$\lceil \log_2 k \rceil T_X$
Step (v)(vi)	-	$\frac{3nm}{4} - \frac{3m}{2} - \frac{n}{2} + 1$	$\lceil \log_2 (n-1) \rceil T_X$
Step (iv)*	-	$\frac{m^2-km-mn+k+n-1}{2}$	$\lceil \log_2 k \rceil T_X$
Step (v)(vi)*	-	$\frac{3nm}{4} - \frac{3m}{2} - \frac{k}{4} - n + 1$	$\lceil \log_2 n \rceil T_X$
* represents the case of odd n.			

2) AN EXAMPLE OF $S_2 \bmod f(x)$

To illustrate our classification and reduction strategy, we give a small example here. Consider S_2 presented in former example. According to Lemma 5, S_1 can be rewritten as

$$S_1 = g_1 x^3 + g_2 x^{-3},$$

where $g_1 = E_{3,2}x^6 + E_{3,1}x^3 + E_{3,0}$, $g_2 = E_{2,1}x^6 + E_{2,0}x^3 + E_{1,0}$. Let $A_s + A_t = \sum_{i=0}^{2} u^{(s,t)} x^i$ and $B_s + B_t = \sum_{i=0}^{2} v^{(s,t)} x^i$ for $3 \geq s > t \geq 0$. The explicit form of $\mathbf{E}_{g_1}$ is given by

$$\mathbf{E}_{g_1} = \left[\begin{array}{ccc}
u_0^{(3,0)}v_0^{(3,0)} & 0 & 0 \\
u_1^{(3,0)}v_0^{(3,0)} & u_0^{(3,0)}v_1^{(3,0)} & 0 \\
u_2^{(3,0)}v_0^{(3,0)} & u_1^{(3,0)}v_1^{(3,0)} & u_0^{(3,0)}v_2^{(3,0)} \\
\hline
u_0^{(3,1)}v_0^{(3,1)} & u_2^{(3,0)}v_1^{(3,0)} & u_1^{(3,0)}v_2^{(3,0)} \\
u_1^{(3,1)}v_0^{(3,1)} & u_0^{(3,1)}v_1^{(3,1)} & u_2^{(3,0)}v_2^{(3,0)} \\
u_2^{(3,1)}v_0^{(3,1)} & u_1^{(3,1)}v_1^{(3,1)} & u_0^{(3,1)}v_2^{(3,1)} \\
\hline
u_0^{(3,2)}v_0^{(3,2)} & u_2^{(3,1)}v_1^{(3,1)} & u_1^{(3,1)}v_2^{(3,1)} \\
u_1^{(3,2)}v_0^{(3,2)} & u_0^{(3,2)}v_1^{(3,2)} & u_2^{(3,1)}v_2^{(3,1)} \\
u_2^{(3,2)}v_0^{(3,2)} & u_1^{(3,2)}v_1^{(3,2)} & u_0^{(3,2)}v_2^{(3,2)} \\
\hline
0 & u_2^{(3,2)}v_1^{(3,2)} & u_1^{(3,2)}v_2^{(3,2)} \\
0 & 0 & u_2^{(3,2)}v_2^{(3,2)} \\
0 & 0 & 0
\end{array}\right].$$

The organization of $\mathbf{E}_{g_2}$ is almost the same as $\mathbf{E}_{g_1}$. It is easy to see that the computation of g_1, g_2 in Step (iv) cost 32 XOR gates with $2T_X$ delay. In addition, 17 more XOR gates are needed as well for Step (v) and (vi) with $2T_X$ delay. Combined with the number of logic gates required in Step (i), (ii), it totally requires 54 AND and 85 XOR gates for $S_2 \bmod f(x)$, with $T_A + 5T_X$ delay.

C. THEORETIC COMPLEXITY

After the computation of S_1 and S_2 modulo $f(x)$, other m XOR gates are needed to add two results together. From Table 1 and 2, it is clear that the delay of $S_2 \bmod f(x)$ cost one more T_X than $S_1 \bmod f(x)$. Thus, in parallel implementation of S_1, S_2 modulo $f(x)$, the delay is $T_A + (1 + \lceil \log_2 k \rceil + \lceil \log_2 n \rceil)T_X$ (or $T_A + (1 + \lceil \log_2 k \rceil + \lceil \log_2 (n-1) \rceil)T_X$ for even n). Plus one more T_X that cost in the final addition, we obtain the time complexity of our proposed architecture as

$$\text{Time delay} = T_A + (2 + \lceil \log_2 k \rceil + \lceil \log_2 n \rceil)T_X.$$

The space complexity is

$$\begin{aligned}
\#\,\text{AND} &= \frac{m^2}{2} + \frac{mk}{2}, \\
\#\,\text{XOR} &= \frac{m^2}{2} + \frac{mk}{2} + \frac{5mn}{4} + n\sum_{i=1}^{k-1} W(i) + n \\
&\quad + k\sum_{i=1}^{n-2} W(i) + kW(n-2) - \frac{5m}{2} + 1, \\
\#\,\text{XOR}^* &= \frac{m^2}{2} + \frac{mk}{2} + \frac{5mn}{4} + n\sum_{i=1}^{k-1} W(i) + \frac{n}{2} \\
&\quad + \frac{k}{4} + k\sum_{i=1}^{n-2} W(i) + kW(n-2) - \frac{5m}{2} + \frac{1}{2}.
\end{aligned} \tag{10}$$

The symbol "*" represent the case of odd n. The formulation for the number of XOR varies according to the parity of n. We note that these formulae contain sums of hamming weights related to $k-1$ or $n-2$. In fact, the expression $\sum_{i=0}^{\delta} W(i)$ can be roughly written as $\frac{\delta}{2}\log_2 \delta$ [22], where δ is a nonzero integer. Thus, the hamming weight formulations related to n roughly equal $O(m \log_2 n)$, while the formulations related to k are roughly equal to $O(m \log_2 k)$. Omit the linear parts, the number of required XOR gates can be rewritten as:

$$\frac{m^2}{2} + \frac{mk}{2} + \frac{5mn}{4} + O(m \log_2 k) + O(m \log_2 n). \tag{11}$$

The above formula reveals the lower bound of the space complexity of our proposal. Based on (10) and (11), it is obvious that with the increase of the parameter n, the number of required AND gates is decreasing. If $n = m$, #AND achieves its lower bound, i.e., $\frac{m^2+m}{2}$. But at this time, the number of required XOR gates is more than $\frac{7m^2}{4}$. Therefore, the optimal parameter n should be the one that minimizes both the number of XOR and AND gates. We combine the two formulations with respect to #AND and #XOR, define a function:

$$M(n) = m^2 + mk + \frac{5mn}{4}.^{1}$$

Please note that $m = nk$. Obviously, $M(n) = m^2 + \frac{m^2}{n} + \frac{5mn}{4}$. When $\frac{m^2}{n} = \frac{5mn}{4}$, namely, $n = 2(\frac{m}{5})^{1/2}$, we obtain the minimal value of $M(n)$, which indicate the best asymptotic space complexity of our proposal. In this case, we see that k is almost equal to n. The space complexity is

$$\begin{aligned}
\#\,\text{AND} &= \frac{m^2}{2} + \frac{\sqrt{5}m^{3/2}}{4}, \\
\#\,\text{XOR} &= \frac{m^2}{2} + \frac{3\sqrt{5}m^{3/2}}{4} + O(m \log_2 k).
\end{aligned}$$

Figure 1 shows the space complexity tendency with the increase of n. It is clear that n could not always increase. Combined with the lowest asymptotic space complexity analysis, we can see that our proposal is more suitable for $x^{nk} + x^k + 1$, where n is smaller than k.

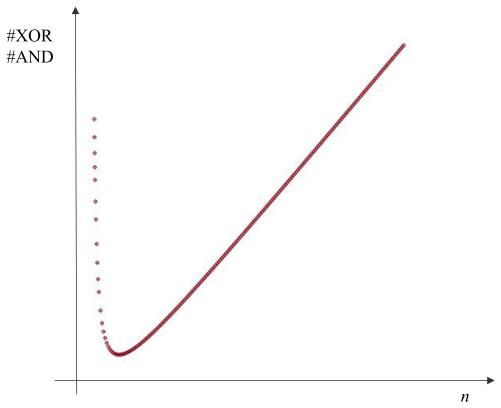

FIGURE 1. Space complexity tendency with increase of n.

IV. SPEEDUP STRATEGY

As shown in previous section, the time delay of our proposal is $T_A + (2 + \lceil \log_2 k \rceil + \lceil \log_2 n \rceil)T_X$. Since

$$\lceil \log_2 k \rceil + \lceil \log_2 n \rceil \le \lceil \log_2 m \rceil + 1,$$

the upper bound of the delay is $T_A + (3 + \lceil \log_2 m \rceil)T_X$. This result is worse than the multiplier using classic Karatsuba algorithm. The main reason is the delay of S_2 is bigger than that of S_1. Indeed, we can add the intermediate values in advance during the computation process of S_1, S_2 to speed up the whole architecture. For better comprehension, we define some additional notations.

- $\mathbf{q}_{S_1,0}, \mathbf{q}_{S_1,1}, \cdots, \mathbf{q}_{S_1,n-1}$ represent the coordinate vectors of $\mathbf{M}_{S_1}(:, i \sim ik) \cdot \mathbf{b}_{i+1}$ in (7) after the computation of $\mathbf{A}_{i,L}\mathbf{b}_i + \mathbf{A}_{i,H}\mathbf{b}_i$, $i = 0, 1, \cdots, n-1$.

[1] Here, we assume that the XOR and AND consist of the same number of transistors. In practical application, one can modify this function by multiplying different weight factors.

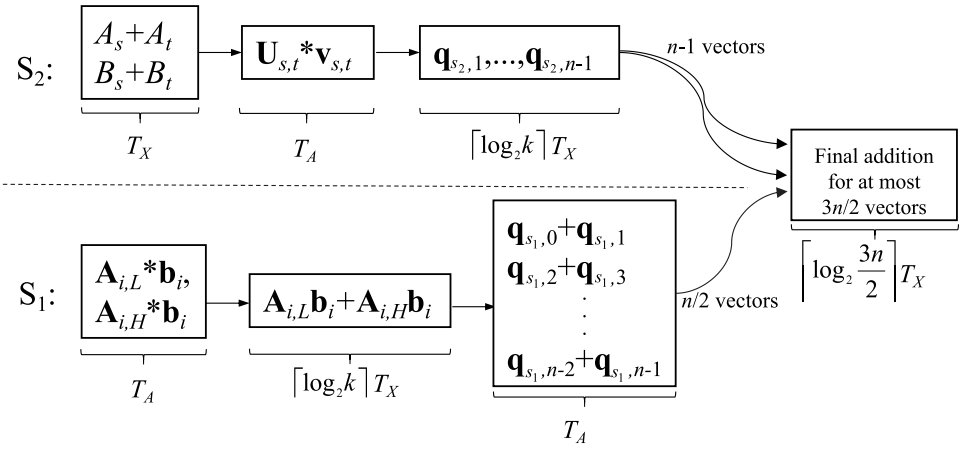

FIGURE 2. Speedup strategy related to our architecture.

TABLE 3. Comparison of bit-parallel multipliers for $GF(2^m)$ generated with $x^m + x^k + 1$, $(m = nk)$.

Multiplier	# AND	#XOR	Time delay
Sunar [7]	m^2	m^2-1	$T_A + (2 + \lceil \log_2 m \rceil)T_X$
Wu [16]	m^2	m^2-1	$T_A + (2 + \lceil \log_2 m \rceil)T_X$
Wu [17]	m^2	m^2-1	$T_A + (2 + \lceil \log_2 m \rceil)T_X$
Fan [13]	m^2	m^2-1	$T_A + \lceil \log_2(2n-1)k \rceil T_X$
Elia [14]	$\frac{3m^2}{4}$	$\frac{3m^2}{4} + \frac{13m}{3} - \frac{23}{4}$	$T_A + (3 + \lceil \log_2 m \rceil)T_X$
Négre [27]	m^2	$\frac{23m^2}{18} - \frac{3m}{2}$	$T_A + \lceil \log_2(2n-1)k \rceil T_X$
Fan [11] Type-A	$m^2 - \frac{m}{3}$	$m^2 - \frac{m}{3}$	$T_A + \lceil \log_2(\max((3n-3)k, (2n-2)k+2^v)) \rceil T_X$
Type-B	$m^2 - \frac{m}{3}$	$m^2 - \frac{2m}{3} + \frac{m}{3} \cdot W(\frac{m}{3})$	$T_A + \lceil \log_2((3n-3)k-1) \rceil T_X$
Li [24]	$\frac{3m^2+2m-1}{4}$	$\frac{3m^2}{4} + \frac{m}{2} + O(m \log_2 m)$	$T_A + (1 + \lceil \log_2(2n-1)k \rceil)T_X$
Li [25] $(x^{3k} + x^k + 1)$	$\frac{2m^2}{3}$	$\frac{2m^2}{3} + \frac{7m}{3} - 1$	$T_A + \lceil \log_2 \frac{8m}{3} \rceil)T_X$
This paper	$\frac{m^2}{2} + \frac{mk}{2}$	$\frac{m^2}{2} + \frac{mk}{2} + \frac{5mn}{4} + O(m \log_2 k)$	$T_A + (\lceil \log_2 k \rceil + \lceil \log_2 3n \rceil)T_X$
where $2^{v-1} < \frac{m}{3} \leq 2^v$ and $W(*)$ is the hamming weight of the number *			

- $\mathbf{q}_{S_2,0}, \mathbf{q}_{S_2,1}, \cdots, \mathbf{q}_{S_2,n-1}$ represent the coordinate vectors corresponding to the polynomials $p_1^{(1)}, p_2^{(1)}, \cdots, p_1^{(\frac{n-1}{2})}, p_2^{(\frac{n-1}{2})}$ and r^1 after we compute the entries additions of Step (v).

For example,

$$\mathbf{q}_{S_1,0} = [\mathbf{A}_{0,H}\mathbf{b}_0, (\mathbf{A}_{0,L}+\mathbf{A}_{0,H})\mathbf{b}_0, \cdots, (\mathbf{A}_{0,L}+\mathbf{A}_{0,H})\mathbf{b}_0]^T,$$
$$\mathbf{q}_{S_2,0} = [h_{3k}^{(1)}, \cdots, h_{nk-1}^{(1)}, h_0^{(1)}, \cdots, h_{3k-1}^{(1)}]^T.$$

According to Table 1 and 2, it is easy to see that the computation of $\mathbf{q}_{S_1,0}, \mathbf{q}_{S_1,1}, \cdots, \mathbf{q}_{S_1,n-1}$ cost $T_A + \lceil \log_2 k \rceil T_X$, while $\mathbf{q}_{S_2,0}, \mathbf{q}_{S_2,1}, \cdots, \mathbf{q}_{S_2,n-1}$ cost $T_A + (1 + \lceil \log_2 k \rceil)T_X$.

Our speedup strategy is adding these vector $\mathbf{q}_{S_1,i}$ and $\mathbf{q}_{S_2,i}$ directly before completing S_1 and S_2. Since the computation of $\mathbf{q}_{S_2,i}$ cost one more T_X than $\mathbf{q}_{S_1,i}$, we can perform one more addition for each two vectors, i.e., $\mathbf{q}_{S_1,i} + \mathbf{q}_{S_1,i+1}$ for $i = 0, 2, \cdots, n-2$ (or $i = 0, 2, \cdots, n-3$ if n is odd). After this addition, we obtain $\lceil \frac{n}{2} \rceil$ column vectors. Plus n (or $n-1$) coordinate vectors $\mathbf{q}_{S_2,0}, \mathbf{q}_{S_2,1}, \cdots, \mathbf{q}_{S_2,n-1}$, there are at most $\lceil \frac{3n}{2} \rceil$ vectors need to be added, which requires only $\lceil \log_2 \lceil \frac{3n}{2} \rceil \rceil T_X$. The computation sequence of our architecture is arranged as shown in Fig.1.

As a result, the whole time delay is

$$T_A + (1 + \lceil \log_2 k \rceil + \lceil \log_2 \lceil \frac{3n}{2} \rceil \rceil)T_X.$$

Furthermore, based on a related Lemma of [11], we have $1 + \lceil \log_2 \lceil \frac{3n}{2} \rceil \rceil = \lceil \log_2 3n \rceil$. Thus, the time delay formulation can be simplified as

$$T_A + (\lceil \log_2 k \rceil + \lceil \log_2 3n \rceil)T_X.$$

[1] If n is even, there only $n-1$ coordinate vectors corresponding to $p_1^{(1)}, p_2^{(1)}, \cdots, p_1^{(\frac{n}{2}-1)}, p_2^{(\frac{n}{2}-1)}, g_{\frac{n}{2}} x^{-k}$.

V. COMPARISON AND DISCUSSION

According to the descriptions in the previous section, it is clear that the time delay of our proposal using speedup strategy is $T_A+(\lceil\log_2 k\rceil+\lceil\log_2 3n\rceil)T_X$. However, it is especially attractive if

$$\lceil\log_2 k\rceil + \lceil\log_2 3n\rceil = \lceil\log_2 3n\cdot k\rceil = \lceil\log_2 3m\rceil. \quad (12)$$

At this time, the corresponding time delay is $T_A + \lceil\log_2 3m\rceil)T_X$, which approximately equals the fastest 2-term Karatsuba based multiplier [24]. In fact, we have checked all the irreducible $x^{nk}+x^k+1, k>1$ of degree $m=nk\in[100, 1023]$ over $\mathbb{F}_2$, and found about 54% such trinomials satisfy (12), and the rest of them requires at most one T_X than than the fastest Karatsuba multiplier so far.

Table 3 gives a comparison of different implementations of bit-parallel multipliers in the fields generated by trinomials $x^m+x^k+1, m=nk$. More explicitly, we omit the expression $O(m\log_2 n)$ in (11), as n is usually smaller than k shown in Section 3.3. Based on this table, it is easy to see that our proposal has better space complexity while maintains relatively low time complexity. The best of our result only costs about $\frac{m^2}{2}+O(m^{3/2})$ circuit gates compared with the previous architectures without using a divide-and-conquer algorithm. On the other hand, the time complexity of the proposed multiplier is very closed to the fastest result utilizing classic Karatsuba algorithm.

VI. CONCLUSION

In this paper, we investigate the application of a n-term Karatsuba algorithm and develop a new type of bit-parallel multiplier for a class of irreducible trinomials. The proposed architecture shows that specific type of trinomials combined with Karatsuba algorithm variations can reduce the space complexity further compared with classic Karatsuba multipliers. We next work on the construction of n-term Karatsuba multiplier for general trinomials.

APPENDIX A
THE SUB-EXPRESSION SHARING FOR ENTRIES ADDITION IN S_1

Let $\mathbf{P}_i$ denote the coordinate vector of $\mathbf{A}_{i,L}\mathbf{b}_i+\mathbf{A}_{i,H}\mathbf{b}_i$ and $\mathbf{P}'_i$ denote the coordinate vector of $\mathbf{A}_{i,L}\mathbf{b}_i$ (or $\mathbf{A}_{i,H}\mathbf{b}_i$ for $i=0$). Clearly, both $\mathbf{P}_i$ and $\mathbf{P}'_i$ are $k\times 1$ vectors. Therefore, (7) can be rewritten as:

$$\mathbf{M}_{S_1}\cdot\mathbf{b}=\begin{bmatrix}\mathbf{P}'_0 & \mathbf{P}_1 & \mathbf{P}_2 & \cdots & \mathbf{P}_{n-2} & \mathbf{P}_{n-1}\\ \mathbf{P}_0 & \mathbf{P}'_1 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0}\\ \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}'_2 & \cdots & \mathbf{0} & \mathbf{0}\\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\ \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \cdots & \mathbf{P}'_{n-2} & \mathbf{0}\\ \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \cdots & \mathbf{P}_{n-2} & \mathbf{P}'_{n-1}\end{bmatrix}.$$

So we only need to compute entries additions for k intermediate coordinate vectors

$$\mathbf{P}_0+\mathbf{P}_1+\cdots+\mathbf{P}_{n-2}+\mathbf{P}'_{n-1} \quad (13)$$

TABLE 4. Saved XOR gates about the entries addition.

Matrix rows	Overlapped terms	Saved #XOR
$1\sim k$	$(n-2)k$	$((n-2)-W(n-2))k$
$k+1\sim 2k$	k	$1-W(1))k$
$2k+1\sim 3k$	$2k$	$(2-W(2))k$
$\vdots$	$\vdots$	$\vdots$
$(n-2)k+1\sim(n-1)k$	$(n-2)k$	$((n-2)-W(n-2))k$
$(n-1)k+1\sim m$	nk	$(n-1)k$

and all the entries additions can be computed through reusing these values. Table 4 indicates the overlapped values and the number of saved XOR gates.

Note that the additions between these vectors without *sub-expression sharing* require $2(n-1)k-k\sum_{i=1}^{n-2} i$ XOR gates. By subtracting the number of saved XOR gates, the number of required XOR gates actually is

$$m+k\sum_{i=1}^{n-2}W(i)+kW(n-2).$$

APPENDIX B
RELATED MATRICES OF THE EXAMPLE IN SECTION 3.1.2

As we know the form of $\mathbf{A}_{i,L}$ and $\mathbf{A}_{i,H}$, it is easy to obtain the explicit formulae with respect to $\mathbf{A}_i$ ($i=0, 1, 2, 3$), and $\mathbf{A}_{S_1}$.

For the size of the above matrix, we do not present the line number in the left side. One should note that the rows of $\mathbf{A}_{S_1}$ correspond the term degree $[-6, 17]$.

$$\mathbf{A}_i=\begin{array}{c}-3\\-2\\-1\\ \\0\\1\\2\\ \\3\\4\\5\\ \\6\\7\\8\\ \\9\\10\\11\end{array}\begin{bmatrix}a_{3i+0} & 0 & 0\\ a_{3i+1} & a_{3i+0} & 0\\ a_{3i+2} & a_{3i+1} & a_{3i+0}\\ \hline a_{3i+0} & a_{3i+2} & a_{3i+1}\\ a_{3i+1} & a_{3i+0} & a_{3i+2}\\ a_{3i+2} & a_{3i+1} & a_{3i+0}\\ \hline a_{3i+0} & a_{3i+2} & a_{3i+1}\\ a_{3i+1} & a_{3i+0} & a_{3i+2}\\ a_{3i+2} & a_{3i+1} & a_{3i+0}\\ \hline a_{3i+0} & a_{3i+2} & a_{3i+1}\\ a_{3i+1} & a_{3i+0} & a_{3i+2}\\ a_{3i+2} & a_{3i+1} & a_{3i+0}\\ \hline 0 & a_{3i+2} & a_{3i+1}\\ 0 & 0 & a_{3i+2}\\ 0 & 0 & 0\end{bmatrix},$$

and $\mathbf{A}_{S_1}$, as shown at the top of the next page.

$$\mathbf{A}_{S_1} = \left[\begin{array}{ccc:ccc:ccc:ccc}
a_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_1 & a_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_2 & a_1 & a_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
a_0 & a_2 & a_1 & a_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_1 & a_0 & a_2 & a_4 & a_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_2 & a_1 & a_0 & a_5 & a_4 & a_3 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
a_0 & a_2 & a_1 & a_3 & a_5 & a_4 & a_6 & 0 & 0 & 0 & 0 & 0 \\
a_1 & a_0 & a_2 & a_4 & a_3 & a_5 & a_7 & a_6 & 0 & 0 & 0 & 0 \\
a_2 & a_1 & a_0 & a_5 & a_4 & a_3 & a_8 & a_7 & a_6 & 0 & 0 & 0 \\
\hdashline
a_0 & a_2 & a_1 & a_3 & a_5 & a_4 & a_6 & a_8 & a_7 & a_9 & 0 & 0 \\
a_1 & a_0 & a_2 & a_4 & a_3 & a_5 & a_7 & a_6 & a_8 & a_{10} & a_9 & 0 \\
a_2 & a_1 & a_0 & a_5 & a_4 & a_3 & a_8 & a_7 & a_6 & a_{11} & a_{10} & a_9 \\
\hdashline
0 & a_2 & a_1 & a_3 & a_5 & a_4 & a_6 & a_8 & a_7 & a_9 & a_{11} & a_{10} \\
0 & 0 & a_2 & a_4 & a_3 & a_5 & a_7 & a_6 & a_8 & a_{10} & a_9 & a_{11} \\
0 & 0 & 0 & a_5 & a_4 & a_3 & a_8 & a_7 & a_6 & a_{11} & a_{10} & a_9 \\
\hdashline
0 & 0 & 0 & 0 & a_5 & a_4 & a_6 & a_8 & a_7 & a_9 & a_{11} & a_{10} \\
0 & 0 & 0 & 0 & 0 & a_5 & a_7 & a_6 & a_8 & a_{10} & a_9 & a_{11} \\
0 & 0 & 0 & 0 & 0 & 0 & a_8 & a_7 & a_6 & a_{11} & a_{10} & a_9 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 0 & a_8 & a_7 & a_9 & a_{11} & a_{10} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_8 & a_{10} & a_9 & a_{11} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{11} & a_{10} & a_9 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{11} & a_{10} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{11} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right].$$

After reduction process, the explicit form of $\mathbf{M}_{S_1,1}$ and $\mathbf{M}_{S_1,2}$ are presented as shown at the top of the next page.

APPENDIX C
PROOF OF OBSERVATION FOR EVEN *n*

If n is even, we have $\lambda = \frac{n}{2}$, and the degree of g_i is $nk-2$. Let $g_i = \sum_{j=0}^{nk-2} h_j^{(i)} x^j$. Then

$$g_i x^{(n-2i-1)k} = \sum_{j=0}^{2ik-1} h_j^{(i)} x^{j+(n-2i-1)k} + \sum_{j=2ik}^{nk-2} h_j^{(i)} x^{j+(n-2i-1)k},$$

for $i = 1, 2, \cdots, \frac{n}{2}-1$. Similar with case of odd n, only one part of the above expression needs reduction by $f(x)$. We have

$$\sum_{j=2ik}^{nk-2} h_j^{(i)} x^{j+(n-2i-1)k} = \sum_{j=2ik}^{nk-2} h_j^{(i)} (x^{j-2ik-k} + x^{j-2ik}).$$

We note that if $i = \frac{n}{2}$, all the term degrees of $g_{\frac{n}{2}} x^{-k}$ are in the range $[-k, nk-k-1]$. No further reduction is needed.

By combining non-overlapped parts of above expressions, the result of $g_i x^{(n-2i-1)k} \bmod f(x)$ is given by

$$g_{\frac{n}{2}} x^{-k} \bmod f(x) = g_{\frac{n}{2}} x^{-k}$$
$$g_i x^{(n-2i-1)k} \bmod f(x) = p_1^{(i)}(x) + p_2^{(i)}(x),$$

where

$$p_1^{(i)} = \sum_{j=0}^{2ik-1} h_j^{(i)} x^{j+nk-2ik-k} + \sum_{j=2ik}^{nk-1} h_j^{(i)} x^{j-2ik-k},$$
$$p_2^{(i)} = \sum_{j=2ik}^{nk-2} h_j^{(i)} x^{j-2ik},$$

for $i = 1, 2, \cdots, \frac{n}{2}-1$. Therefore, in this case, to obtain $S_2 \bmod f(x)$, we only need to add $p_1^{(1)}, p_2^{(1)}, \cdots, p_1^{(\frac{n}{2}-1)}$,

$$
\mathbf{M}_{S_1,1} = \left[\begin{array}{ccc|ccc|ccc|ccc}
a_0 & a_2 & a_1 & a_3 & a_5 & a_4 & a_6 & a_8 & a_7 & a_9 & a_{11} & a_{10} \\
a_1 & a_0 & a_2 & a_4 & a_3 & a_5 & a_7 & a_6 & a_8 & a_{10} & a_9 & a_{11} \\
a_2 & a_1 & a_0 & a_5 & a_4 & a_3 & a_8 & a_7 & a_6 & a_{11} & a_{10} & a_9 \\
\hline
a_0 & a_2 & a_1 & a_3 & a_5 & a_4 & a_6 & a_8 & a_7 & a_9 & a_{11} & a_{10} \\
a_1 & a_0 & a_2 & a_4 & a_3 & a_5 & a_7 & a_6 & a_8 & a_{10} & a_9 & a_{11} \\
a_2 & a_1 & a_0 & a_5 & a_4 & a_3 & a_8 & a_7 & a_6 & a_{11} & a_{10} & a_9 \\
\hline
a_0 & a_2 & a_1 & a_3 & a_5 & a_4 & a_6 & a_8 & a_7 & a_9 & a_{11} & a_{10} \\
a_1 & a_0 & a_2 & a_4 & a_3 & a_5 & a_7 & a_6 & a_8 & a_{10} & a_9 & a_{11} \\
a_2 & a_1 & a_0 & a_5 & a_4 & a_3 & a_8 & a_7 & a_6 & a_{11} & a_{10} & a_9 \\
\hline
a_0 & a_2 & a_1 & a_3 & a_5 & a_4 & a_6 & a_8 & a_7 & a_9 & a_{11} & a_{10} \\
a_1 & a_0 & a_2 & a_4 & a_3 & a_5 & a_7 & a_6 & a_8 & a_{10} & a_9 & a_{11} \\
a_2 & a_1 & a_0 & a_5 & a_4 & a_3 & a_8 & a_7 & a_6 & a_{11} & a_{10} & a_9
\end{array}\right],
$$

$$
\mathbf{M}_{S_1,2} = \left[\begin{array}{ccc|ccc|ccc|ccc}
a_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_1 & a_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a_2 & a_1 & a_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & a_5 & a_4 & a_6 & a_8 & a_7 & a_9 & a_{11} & a_{10} \\
0 & 0 & 0 & 0 & 0 & a_5 & a_7 & a_6 & a_8 & a_{10} & a_9 & a_{11} \\
0 & 0 & 0 & 0 & 0 & 0 & a_8 & a_7 & a_6 & a_{11} & a_{10} & a_9 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & a_8 & a_7 & a_9 & a_{11} & a_{10} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_8 & a_{10} & a_9 & a_{11} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{11} & a_{10} & a_9 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{11} & a_{10} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{11} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right].
$$

$p_2^{(\frac{n}{2}-1)}$ and $g_{\frac{n}{2}}x^{-k}$ in parallel, which cost $\lceil \log_2(n-1) \rceil$ XOR gate delay.

REFERENCES

[1] A. J. Menezes, I. F. Blake, X. Gao, R. C. Mullin, S. A. Vanstone, and T. Yaghoobian, *Applications of Finite Fields*. Norwell, MA, USA: Kluwer, 1993.

[2] R. Lidl and H. Niederreiter, *Finite Fields*. New York, NY, USA: Cambridge Univ. Press, 1996.

[3] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography* (London Mathematical Society Lecture Note Series), vol. 265. Cambridge, U.K.: Cambridge Univ. Press, 1999.

[4] A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers on automata," *Sov. Phys. Doklady*, vol. 7, no. 7, pp. 595–596, 1963.

[5] H. Fan and M. A. Hasan, "Fast bit parallel-shifted polynomial basis multipliers in $GF(2^n)$," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 12, pp. 2606–2615, Dec. 2006.

[6] H. Fan and M. A. Hasan, "A survey of some recent bit-parallel $GF(2^n)$ multipliers," *Finite Fields Appl.*, vol. 32, pp. 5–43, Mar. 2015.

[7] B. Sunar and C. K. Koc, "Mastrovito multiplier for all trinomials," *IEEE Trans. Comput.*, vol. 48, no. 5, pp. 522–527, May 1999.

[8] A. Weimerskirch and C. Paar, "Generalizations of the Karatsuba algorithm for efficient implementations," Cryptol. ePrint Arch., Tech. Rep. 2006/224, 2006. [Online]. Available: http://eprint.iacr.org/

[9] T. Zhang and K. K. Parhi, "Systematic design of original and modified Mastrovito multipliers for general irreducible polynomials," *IEEE Trans. Comput.*, vol. 50, no. 7, pp. 734–749, Jul. 2001.

[10] A. Cilardo, "Fast parallel $GF(2^m)$ polynomial multiplication for all degrees," *IEEE Trans. Comput.*, vol. 62, no. 5, pp. 929–943, May 2013.

[11] H. Fan, "A chinese remainder theorem approach to bit-parallel $GF(2^n)$ polynomial basis multipliers for irreducible trinomials," *IEEE Trans. Comput.*, vol. 65, no. 2, pp. 343–352, Feb. 2016.

[12] A. Hariri and A. Reyhani-Masoleh, "Bit-serial and bit-parallel Montgomery multiplication and squaring over $GF(2^m)$," *IEEE Trans. Comput.*, vol. 58, no. 10, pp. 1332–1345, Oct. 2009.

[13] H. Fan and Y. Dai, "Fast bit-parallel $GF(2^n)$ multiplier for all trinomials," *IEEE Trans. Comput.*, vol. 54, no. 4, pp. 485–490, Apr. 2005.

[14] M. Elia, M. Leone, and C. Visentin, "Low complexity bit-parallel multipliers for $GF(2^m)$ with generator polynomial $x^m + x^k + 1$," *Electron. Lett.*, vol. 35, no. 7, pp. 551–552, Apr. 1999.

[15] K.-Y. Chang, D. Hong, and H.-S. Cho, "Low complexity bit-parallel multiplier for GF(2^m) defined by all-one polynomials using redundant representation," *IEEE Trans. Comput.*, vol. 54, no. 12, pp. 1628–1630, Dec. 2005.

[16] H. Wu, "Bit-parallel finite field multiplier and squarer using polynomial basis," *IEEE Trans. Comput.*, vol. 51, no. 7, pp. 750–758, Jul. 2002.
[17] H. Wu, "Montgomery multiplier and squarer for a class of finite fields," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 521–529, May 2002.
[18] H. Fan, J. Sun, M. Gu, and K.-Y. Lam, "Overlap-free Karatsuba-Ofman polynomial multiplication algorithms," *IET Inf. Secur.*, vol. 4, no. 1, pp. 8–14, Mar. 2010.
[19] H. Fan, M. Gu, J. Sun, and K.-Y. Lam, "Obtaining more Karatsuba-like formulae over the binary field," *IET Inf. Secur.*, vol. 6, no. 1, pp. 14–19, Mar. 2012.
[20] P. L. Montgomery, "Five, six, and seven-term Karatsuba-like formulae," *IEEE Trans. Comput.*, vol. 54, no. 3, pp. 362–369, Mar. 2005.
[21] Y. Li, G.-L. Chen, and J. Li, "Speedup of bit-parallel Karatsuba multiplier in $GF(2^m)$ generated by trinomials," *Inf. Process. Lett.*, vol. 111, no. 8, pp. 390–394, Jan. 2011.
[22] Y. Li and Y. Chen, "New bit-parallel Montgomery multiplier for trinomials using squaring operation," *Integr., VLSI J.*, vol. 52, pp. 142–155, Jan. 2016.
[23] X.-N. Xie, G.-L. Chen, and Y. Li, "Novel bit-parallel multiplier for $GF(2^m)$ defined by all-one polynomial using generalized Karatsuba algorithm," *Inf. Process. Lett.*, vol. 114, no. 3, pp. 140–146, Oct. 2014.
[24] Y. Li, X. Ma, Y. Zhang, and C. Qi, "Mastrovito form of non-recursive Karatsuba multiplier for all trinomials," *IEEE Trans. Comput.*, vol. 66, no. 9, pp. 1573–1584, Sep. 2017.
[25] Y. Li, Y. Zhang, and X. Guo, "Efficient nonrecursive bit-parallel Karatsuba multiplier for a special class of trinomials," *VLSI Des.*, vol. 2018, Jan. 2018, Art. no. 9269157, doi: 10.1155/2018/9269157.
[26] G. Shou, Z. Mao, Y. Hu, Z. Guo, and Z. Qian, "Low complexity architecture of bit parallel multipliers for GF(2^m)," *Electron. Lett.*, vol. 46, no. 19, pp. 1326–1327, Sep. 2010.
[27] C. Négre, "Efficient parallel multiplier in shifted polynomial basis," *J. Syst. Archit.*, vol. 53, nos. 2–3, pp. 109–116, Feb. 2007.
[28] H. Shen and Y. Jin, "Low complexity bit parallel multiplier for GF(2^m) generated by equally-spaced trinomials," *Inf. Process. Lett.*, vol. 107, no. 6, pp. 211–215, Mar. 2008.

YIN LI received the B.Sc. degree in information engineering and the M.Sc. degree in cryptography from Information Engineering University, Zhenzhou, in 2004 and 2007, respectively, and the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, in 2011. He held a post-doctoral position at the Department of Computer Science, Ben-Gurion University of the Negev, Israel. He is currently a Lecturer with the Department of Computer Science and Technology, Xinyang Normal University, Xinyang, Henan, China. His current research interests include algorithm and architectures for computation in finite field, computer arithmetic, and secure cloud computing.

YU ZHANG received the B.Sc. degree from the Henan University of Economics and Law in 2008 and the Ph.D. degree from the Huazhong University of Science and Technology in 2015. Since 2016, he has been with the Department of Computer Science and Information Technology, Xinyang Normal University, where he is currently a Lecturer. His major interests include information security, cryptography, and information retrieval.

XIAOLI GUO received the B.Sc. degree in computer science from Xinyang Normal University in 2017, where she is currently pursuing the M.D. degree with the Department of Computer Science and Technology. Her research interests include computer algebra and cryptography.

CHUANDA QI received the B.Sc. degree in mathematics from Xinyang Normal University, Xinyang, in 1985 and the Ph.D. degree in cryptography from Information Engineering University in 2007. He is currently a Professor with the Department of Computer Science and Technology, Xinyang Normal University. His research interests include cryptography, complexity theory, and mathematical logic.