Mastrovito form of Karatsuba Multiplier for All Trinomials

Yin Li, Xingpo Ma, Yu Zhang and Chuanda Qi

Abstract—We present a Matrix-vector form of Karatsuba multiplication over $GF(2^m)$ generated by an irreducible trinomial. Based on shifted polynomial basis (SPB), two Mastrovito matrices for different Karatsuba multiplication parts are studied. Then related multiplier architecture is proposed. This design effectively exploits the overlapped entries of the Mastrovito matrices to reduce the space complexity even further. We show that this new type of Karatsuba multiplier is only one T_X slower than the fastest bit-parallel multiplier for all trinomials, where T_X is the delay of one 2-input XOR gate. Meanwhile its space complexity is roughly reduced by $O(\frac{m^2}{4})$ logic gates. Compared with previously proposed bit-parallel Karatsuba multipliers, it is the first time to achieve such time delay bound, while maintain nearly the same space complexity.

Index Terms—Karatsuba multiplier, Mastrovito, shifted polynomial basis, trinomials.

1 Introduction

Efficient hardware implementation of multiplication over $GF(2^m)$ is one of the main topics studied during recent years, as it is frequently required in many areas such as coding theory and public key cryptography [17], [18]. A number of algorithms for efficient $GF(2^m)$ multiplication have been proposed, one of the most attractive approach is the Karatsuba algorithm (KA) [1]. This algorithm was originally used in the digit number multiplication and can be directly shifted to polynomial multiplication [2]. For polynomial multiplication, the key idea of the KA saves coefficient multiplications at the cost of extra coefficient additions.When using polynomial basis (PB) representation, the $GF(2^m)$ multiplication consists of polynomial multiplication and modular reduction. Let $f(x)$ be the irreducible polynomial that defines $GF(2^m)$, $A(x), B(x)$ be two arbitrary elements of $GF(2^m)$. Then the field multiplication is defined as $C(x) = A(x) \cdot B(x) \bmod f(x)$. Since the KA can optimize the polynomial multiplication, it can be easily adopted to design high efficient $GF(2^m)$ multipliers, either for sub-quadratic complexity multipliers (cost $O(m^\delta)$ circuit gates for implementation with $1 < \delta < 2$) [3], [4] or hybrid quadratic complexity multipliers (cost $O(m^2)$ circuit gates) [10], [11]. The hybrid quadratic multipliers firstly perform a few iterations of KA to reduce the whole space complexities, and then perform quadratic multiplication algorithm over the smaller input operands to achieve relatively higher speed. Therefore, this kind of multipliers provided a trade-off between the time and space complexities. However, the classic KA is applicable for optimizing the polynomial multiplication and unfeasible for the modular reduction. Meanwhile, the extra additions also lead the hybrid quadratic multiplier slower than the ordinary quadratic $GF(2^m)$ multipliers [12], [13], [22], [23], [24]. For example, the classic Karatsuba multiplier proposed by Elia [10] is at least 2 T_X slower than fastest bit-parallel multiplier [13], where T_X is the delay of one 2-input XOR gate. Until now, there are several schemes proposed to accelerate the hybrid multipliers, e.g., [9], [15], [21]. These schemes are utilizing either an equally-spaced trinomial (EST) or a modified KA.

Mastrovito [5] provided a novel approach that transforms the field multiplication into a matrix-vector multiplication. A product matrix $\mathbf{M}$ is introduced to combine the polynomial multiplication and modular reduction together. Thus the field multiplication is carried out by $\mathbf{c} = \mathbf{M} \cdot \mathbf{b}$, where $\mathbf{c}$, $\mathbf{b}$ are the coefficient vectors of $C(x)$ and $B(x)$ and $\mathbf{M}$ is constructed from $A(x)$ and $f(x)$ presented previously. Empirically, the Mastrovito multiplier is generally faster than other type of multipliers, but the construction of $\mathbf{M}$ is the implementation bottleneck of this type of multipliers. In [13], Fan and Hasan proposed a new Mastrovito multiplier based on shifted polynomial basis (SPB) which simplified construction of $\mathbf{M}$. This type of multiplier is considered as the fastest bit-parallel multiplier so far. This architecture contains m^2 AND and m^2-1 XOR gates with time delay of $T_A+(1+\lceil \log_2 m \rceil)T_X$ (for good field it is equal to $T_A + \lceil \log_2 m \rceil T_X$), where T_A is the delay of one 2-input AND gate. In this paper, we apply the idea of Mastrovito to KA and describe a new architecture for Karatsuba quadratic multiplier for all irreducible trinomials. Explicit formulae of $\mathbf{M}$ with respect to KA are presented, based on which we build a Mastrovito form of Karatsuba multiplier using SPB. Our scheme fully takes advantage of the Mastrovito and Karatsuba algorithm and achieves faster implementation speed. It is argued that this new multiplier is only one T_X slower than the fastest multipliers for trinomials. Meanwhile, the space complexity of the proposed mul-

• *Yin Li, Xingpo Ma, Yu Zhang are with Department of Computer Science and Technology, Chuanda Qi is with the Department of Mathematics, Xinyang Normal University, Henan, P.R.China, 464000. email: yunfeiyangli@gmail.com (Yin Li).*

tiplier is roughly 3/4 of those multipliers. To the best of our knowledge, it is the first time to achieve such time delay bound compared with previously proposed bit-parallel Karatsuba multipliers.

The remainder of this paper is organized as follows: in section 2, we first briefly introduce some basic concepts and recall the Karatsuba and Mastrovito algorithms. Then, based on combination of these two algorithms, a new type of bit-parallel multiplier architecture is proposed in the following section. Section 4 presents the comparison between the proposed multiplier and some others. The last section summarizes the results and draws some conclusions.

2 NOTATION AND PRELIMINARY

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

Consider a binary extension field generated with an irreducible trinomial $GF(2^m) \cong \mathbb{F}_2[x]/(f(x))$ where $f(x) = x^m + x^k + 1$. Let x be a root of $f(x)$, then the set $M = \{x^{m-1}, \cdots, x, 1\}$ constitute a polynomial basis (PB). The shifted polynomial basis (SPB) proposed by Fan and Dai [12] was derived from polynomial basis. It can be obtained by multiplying the set M by a certain exponentiation of x:

Definition 1 *[12] Let v be an integer and the ordered set $M = \{x^{m-1}, \cdots, x, 1\}$ be a polynomial basis of $GF(2^m)$ over $\mathbb{F}_2$. The ordered set $x^{-v}M := \{x^{i-v} | 0 \le i \le m-1\}$ is called the shifted polynomial basis(SPB) with respect to M.*

It is easy to check that the field multiplication using SPB is nearly the same as that using PB except a certain parameter:

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

The advantage of SPB over PB is that it can simplify the modular reduction if v is properly chosen. For trinomial $x^m + x^k + 1$, it has been proved that the optimal value of v here is k or $k-1$ [12]. In this study, we choose that v equals k and use this denotation thereafter.

The Karatsuba algorithm (KA) optimized the polynomial multiplication $D(x) = A(x) \cdot B(x)$ by partitioning each polynomial into two halves.

$$\begin{aligned} AB =& (A_H x^n + A_L) \cdot (B_H x^n + B_L) \\ =& A_H B_H x^{2n} + [(A_H + A_L)(B_H + B_L) \\ & + A_H B_H + A_L B_L]\, x^n + A_L B_L, \end{aligned} \quad (1)$$

where $n = m/2$, A_H, A_L and B_H, B_L are two halves of $A(x)$ and $B(x)$, respectively. When m is odd, the formula is almost the same as (1). We note that the addition and subtraction are the same in $GF(2^m)$. The above expression saves one partial multiplications at the cost of three extra partial additions. For VLSI implementation of expression (1), it leads more XOR gate delay than the ordinary polynomial multiplication.

Indeed, polynomial multiplication $D(x) = A(x) \cdot B(x)$ can be implemented as a matrix-vector multiplication

$$\mathbf{d} = \mathbf{A} \cdot \mathbf{b},$$

where $\mathbf{b} = [b_0, b_1, \cdots, b_{m-1}]^T$ and $\mathbf{d} = [d_0, d_1, \cdots, d_{m-1}]^T$ are the coefficient vectors of $B(x)$ and $D(x)$, respectively. The matrix $\mathbf{A}$ is given by

$$\mathbf{A} = \begin{bmatrix} a_0 & 0 & 0 & \cdots & 0 & 0 \\ a_1 & a_0 & 0 & \cdots & 0 & 0 \\ a_2 & a_1 & a_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m-2} & a_{m-3} & a_{m-4} & \cdots & a_0 & 0 \\ a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_1 & a_0 \\ 0 & a_{m-1} & a_{m-2} & \cdots & a_2 & a_1 \\ 0 & 0 & a_{m-1} & \cdots & a_3 & a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{m-1} & a_{m-2} \\ 0 & 0 & 0 & \cdots & 0 & a_{m-1} \end{bmatrix}. \quad (2)$$

Then, we can perform the modular reduction $C(x) = D(x) \bmod f(x)$. The Mastrovito algorithm [5] provides a way to combine the polynomial multiplication and modular reduction into a matrix-vector multiplication, i.e.,

$$\mathbf{c} = \mathbf{M} \cdot \mathbf{b}.$$

The matrix $\mathbf{M}$ is called *product matrix* which is constructed from $\mathbf{A}$ and $f(x)$, and $\mathbf{c} = [c_0, c_1, \cdots, c_{m-1}]^T$ is the coefficient vector of $C(x)$. The main problem of Mastrovito algorithm is that its implementation relays on the organization of $\mathbf{M}$. Several algorithms are given to construct $\mathbf{M}$ efficiently or make it simpler, some of them are presented in the literatures [12], [13], [22], [23], [24], [25].

In the following section, we will use Mastrovito approach to speed up KA and describe a new architecture for bit-parallel Karatsuba multiplier. We first introduce some notations pertaining to matrices and vectors which are already presented in [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. $\mathbf{Z}[\uparrow i], \mathbf{Z}[\downarrow i]$ represent up and down shift of matrix $\mathbf{Z}$ by i rows and feed the vacancies with zero.

Besides, two extra types of operations are also introduced: $\mathbf{Z}[\circlearrowleft i]$ represent cyclic shift of $\mathbf{Z}$ by upper i rows. $\mathbf{Z}[\upuparrows i]$ and $\mathbf{Z}[\downdownarrows i]$ represent appending i zero vectors to the top and bottom of $\mathbf{Z}$, respectively. For example, $\mathbf{Z}[\circlearrowleft 2]$, $\mathbf{Z}[\upuparrows 1]$ and $\mathbf{Z}[\downdownarrows 2]$ are given by:

$$\begin{aligned} &\mathbf{Z}[\circlearrowleft 2] = [\mathbf{Z}(3,:)^T, \cdots, \mathbf{Z}(m,:)^T, \mathbf{Z}(1,:)^T, \mathbf{Z}(2,:)^T]^T, \\ &\mathbf{Z}[\upuparrows 1] = [\mathbf{Z}(1,:)^T, \cdots, \mathbf{Z}(m,:)^T, \mathbf{0}]^T, \\ &\mathbf{Z}[\downdownarrows 2] = [\mathbf{0}, \mathbf{0}, \mathbf{Z}(1,:)^T, \cdots, \mathbf{Z}(m,:)^T]^T. \end{aligned}$$

3 MASTROVITO FORM OF KARATSUBA MULTIPLIER

In this section, we firstly introduce a matrix form of Karatsuba algorithm for $GF(2^m)$ multiplication using

SPB representation. Then, we develop an efficient approach to calculate the product matrix, based on modified *sub-expression sharing* [5]. Accordingly, a fast bit-parallel Karatsuba multiplier architecture is proposed.

3.1 Matrix form of Karatsuba algorithm

Let $f(x) = x^m + x^k + 1$ be an irreducible trinomial generating the finite field $GF(2^m)$. Provided that $A, B \in GF(2^m)$ are two arbitrary elements in SPB representation, namely,

$$A = x^{-k}\sum_{i=0}^{m-1} a_i x^i,\ B = x^{-k}\sum_{i=0}^{m-1} b_i x^i.$$

The field multiplication consists of performing polynomial multiplication with parameter x^{-k} and then reducing the product modulo $f(x)$, i.e.,

$$\begin{aligned} C =& A \cdot B \bmod f(x) \\ =& x^{-2k}\cdot\left(\sum_{i=0}^{m-1} a_i x^i\right)\cdot\left(\sum_{i=0}^{m-1} b_i x^i\right) \bmod f(x) \\ =& x^{-k}\sum_{i=0}^{m-1} c_i x^i. \end{aligned} \tag{3}$$

We partition A and B into two halves and apply KA to (3). Two cases are considered:

Case 1, m **is even.** Let $m = 2n$ and

$$A = (A_2x^n + A_1)x^{-k},\ B = (B_2x^n + B_1)x^{-k},$$

where $A_1 = \sum_{i=0}^{n-1} a_i x^i, A_2 = \sum_{i=0}^{n-1} a_{i+n}x^i, B_1 = \sum_{i=0}^{n-1} b_i x^i, B_2 = \sum_{i=0}^{n-1} b_{i+n}x^i$. Then,

$$\begin{aligned} C =& A \cdot B \bmod f(x) \\ =& (A_2x^n + A_1)x^{-k}\cdot(B_2x^n + B_1)x^{-k} \bmod f(x) \\ =& \left(A_2B_2x^{2n} + (A_2B_1 + A_1B_2)x^n + A_1B_1\right)x^{-2k} \bmod f(x) \\ =& \left[A_2B_2x^{2n} + A_1B_1 + (A_2B_2 + A_1B_1)x^n\right. \\ & \left.+(A_1 + A_2)(B_1 + B_2)x^n\right]x^{-2k} \bmod f(x) \\ =& \left[(A_2x^{2n} + A_2x^n)B_2 + (A_1x^n + A_1)B_1\right. \\ & \left.+(A_1 + A_2)(B_1 + B_2)x^n\right]x^{-2k} \bmod f(x) \end{aligned} \tag{4}$$

Case 2, m **is odd.** Let $m = 2n+1$ and

$$A = (A_2x^n + A_1)x^{-k},\ B = (B_2x^{n+1} + B_1)x^{-k},$$

where $A_1 = \sum_{i=0}^{n-1} a_i x^i, A_2 = \sum_{i=0}^{n} a_{i+n}x^i, B_1 = \sum_{i=0}^{n} b_i x^i, B_2 = \sum_{i=0}^{n-1} b_{i+n+1}x^i$. Then,

$$\begin{aligned} C =& A \cdot B \bmod f(x) \\ =& (A_2x^n + A_1)\cdot(B_2x^{n+1} + B_1)x^{-2k} \bmod f(x) \\ =& \left(A_2B_2x^{2n+1} + (A_2B_1 + A_1B_2x)x^n + A_1B_1\right)x^{-2k} \bmod f(x) \\ =& \left[A_2B_2x^{2n+1} + A_1B_1 + (A_2B_2x + A_1B_1)x^n\right. \\ & \left.+(A_1 + A_2)(B_1 + B_2x)x^n\right]x^{-2k} \bmod f(x) \\ =& \left[(A_2x^n + A_2)B_2x^{n+1} + (A_1x^n + A_1)B_1\right. \\ & \left.+(A_1 + A_2)(B_1 + B_2x)x^n\right]x^{-2k} \bmod f(x) \end{aligned} \tag{5}$$

In order to compute (4) and (5), we use following notations:

$$S_1 = \begin{cases} (A_2x^n + A_2)B_2x^n + (A_1x^n + A_1)B_1, & (m \text{ is even}), \\ (A_2x^n + A_2)B_2x^{n+1} + (A_1x^n + A_1)B_1, & (m \text{ is odd}), \end{cases}$$

$$S_2 = \begin{cases} (A_1 + A_2)(B_1 + B_2), & (m \text{ is even}), \\ (A_1 + A_2)(B_1 + B_2x), & (m \text{ is odd}). \end{cases}$$

Therefore, the field multiplication is given by

$$C = S_1x^{-2k} + S_2x^{n-2k} \bmod f(x).$$

Let $U = \sum_{i=0}^{n-1} u_i x^i$ and $V = \sum_{i=0}^{n-1} v_i x^i$ be the results of $A_1 + A_2$ and $B_1 + B_2$, respectively[1]. According to previous description presented in Section 2, we know that polynomial multiplication $(A_2x^n + A_2)B_2x^i (i = n, n+1)$, $(A_1x^n + A_1)B_1$ and $S_2 = UV$ can be rewritten as matrix-vector form. In addition, multiplying by x^{-2k} or x^{n-2k} only affects the products degree and does not change their coefficients. Thus, we write $S_2x^{n-2k} = \mathbf{U'}\cdot\mathbf{v}$ where $\mathbf{U'}$ is slightly different from the multiplicative matrix with respect to S_2. For example, if m is even, the form of $\mathbf{U'}$ is as follows:

$$\mathbf{U'} = \begin{array}{c} n-2k \\ n-2k+1 \\ n-2k+2 \\ \vdots \\ 2n-2k-2 \\ 2n-2k-1 \\ 2n-2k \\ 2n-2k+1 \\ \vdots \\ 3n-2k-3 \\ 3n-2k-2 \end{array} \begin{bmatrix} u_0 & 0 & \cdots & 0 & 0 \\ u_1 & u_0 & \cdots & 0 & 0 \\ u_2 & u_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ u_{n-2} & u_{n-3} & \cdots & u_0 & 0 \\ u_{n-1} & u_{n-2} & \cdots & u_1 & u_0 \\ 0 & u_{n-1} & \cdots & u_2 & u_1 \\ 0 & 0 & \cdots & u_3 & u_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & u_{n-1} & u_{n-2} \\ 0 & 0 & \cdots & 0 & u_{n-1} \end{bmatrix}.$$

Different with the matrix form in (2), there are labels in the left side that indicate the exponent of indeterminate x for each line. Notice that S_1x^{-2k} consist of two sub-polynomial multiplications which correspond two matrix-vector multiplications. However, these multiplications can be implemented by only one matrix-vector multiplication. More explicitly, $S_1x^{-2k} = \mathbf{A'}\cdot\mathbf{b}$, where $\mathbf{A'}$ is constructed from $A_2x^n + A_2, A_1x^n + A_1$ and labeled by $\{-2k, -2k+1, \cdots, 2m-2-2k\}$ for the coefficients degree. The organization of $\mathbf{A'}$ varied according the parity of m. Two cases are considered:

Case 1: m **is even.**

$$\begin{aligned} S_1x^{-2k} &= \mathbf{A'}\cdot\mathbf{b} \\ &= \begin{bmatrix} \mathbf{A}_{L1}, & \mathbf{0}_{n\times n} \\ \mathbf{A}_{L1} + \mathbf{A}_{L2}, & \mathbf{A}_{H1} \\ \mathbf{A}_{L2}, & \mathbf{A}_{H1} + \mathbf{A}_{H2} \\ \mathbf{0}_{n\times n}, & \mathbf{A}_{H2} \end{bmatrix}\cdot\begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}. \end{aligned} \tag{6}$$

where $\mathbf{b}_1, \mathbf{b}_2$ represent the coefficient vectors of B_1, B_2, $\mathbf{0}_{n\times n}$ is an $n \times n$ zero matrix, $\mathbf{A}_{L1}$ and $\mathbf{A}_{H1}$ are $n \times n$

1. when m is odd, $U = A_1 + A_2 = \sum_{i=0}^{n} u_i x^i$ and $V = B_1 + B_2x = \sum_{i=0}^{n} v_i x^i$

lower-triangular Toeplitz matrices, $\mathbf{A}_{L2}$ and $\mathbf{A}_{H2}$ are $n \times n$ upper-triangular Toeplitz matrices[2].

$$\mathbf{A}_{L1} = \begin{bmatrix} a_0 & 0 & \cdots & 0 \\ a_1 & a_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \cdots & a_0 \end{bmatrix},$$

$$\mathbf{A}_{L2} = \begin{bmatrix} 0 & a_{n-1} & \cdots & a_2 & a_1 \\ 0 & 0 & \cdots & a_3 & a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{n-1} \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix},$$

$$\mathbf{A}_{H1} = \begin{bmatrix} a_n & 0 & \cdots & 0 \\ a_{n+1} & a_n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1} & a_{m-2} & \cdots & a_n \end{bmatrix},$$

$$\mathbf{A}_{H2} = \begin{bmatrix} 0 & a_{m-1} & \cdots & a_{n+2} & a_{n+1} \\ 0 & 0 & \cdots & a_{n+3} & a_{n+2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{m-1} \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}.$$

Case 2: m **is odd.**

$$\begin{aligned} S_1 x^{-2k} &= \mathbf{A'} \cdot \mathbf{b} \\ &= \begin{bmatrix} \mathbf{A}_{L1}, & \mathbf{0}_{(n+1)\times n} \\ \mathbf{A}_{L1} + \mathbf{A}_{L2}, & \mathbf{A}_{H1} \\ \mathbf{A}_{L2}, & \mathbf{A}_{H1} + \mathbf{A}_{H2} \\ \mathbf{0}_{(n+1)\times(n+1)}, & \mathbf{A}_{H2} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}. \end{aligned} \tag{7}$$

where $\mathbf{0}_{(n+1)\times n}$ and $\mathbf{0}_{(n+1)\times(n+1)}$ represent a $(n+1) \times n$ zero matrix and a $(n+1) \times (n+1)$ zero matrix, respectively. $\mathbf{A}_{L1}$ is a $n \times (n+1)$ lower-triangular Toeplitz matrix, $\mathbf{A}_{H1}$ is a $n \times n$ lower-triangular Toeplitz matrix, $\mathbf{A}_{L2}$ is a $n \times (n+1)$ upper-triangular Toeplitz matrix and $\mathbf{A}_{H2}$ is a $n \times n$ upper-triangular Toeplitz matrix.

$$\mathbf{A}_{L1} = \begin{bmatrix} a_0 & 0 & \cdots & 0 & 0 \\ a_1 & a_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1} & a_{n-2} & \cdots & a_0 & 0 \end{bmatrix},$$

$$\mathbf{A}_{L2} = \begin{bmatrix} 0 & a_{n-1} & \cdots & a_1 & a_0 \\ 0 & 0 & \cdots & a_2 & a_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{n-1} \end{bmatrix},$$

$$\mathbf{A}_{H1} = \begin{bmatrix} a_n & 0 & \cdots & 0 \\ a_{n+1} & a_n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-2} & a_{m-3} & \cdots & a_n \end{bmatrix},$$

$$\mathbf{A}_{H2} = \begin{bmatrix} a_{m-1} & \cdots & a_{n+2} & a_{n+1} \\ 0 & \cdots & a_{n+3} & a_{n+2} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & a_{m-1} \end{bmatrix}.$$

Similarly, the organization of $\mathbf{U'}$ is easier than that of $\mathbf{A'}$, i.e., $\mathbf{U'} = [\mathbf{U}_1^T, \mathbf{U}_2^T]^T$, where

$$\mathbf{U}_1 = \begin{bmatrix} u_0 & 0 & \cdots & 0 \\ u_1 & u_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ u_{t-1} & u_{t-2} & \cdots & u_0 \end{bmatrix},$$

and

$$\mathbf{U}_2 = \begin{bmatrix} 0 & u_{t-1} & \cdots & u_2 & u_1 \\ 0 & 0 & \cdots & u_3 & u_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & u_{t-1} \end{bmatrix}.$$

Here, if m is even, $t = n = \frac{m}{2}$, else if m is odd, $t = n + 1 = \frac{m+1}{2}$.

Example 3.1. Consider the field multiplication using SPB representation over $GF(2^5)$ with the underlying irreducible trinomial $x^5 + x^2 + 1$. The parameter $k = 2$ and SPB is defined as $\{x^{-2}, x^{-1}, 1, x, x^2\}$. Assume that $A = \sum_{i=0}^{4} a_i x^{i-2}$ and $B = \sum_{i=0}^{4} b_i x^{i-2}$ are two elements in $GF(2^5)$, we partition A, B as $A = A_2 + A_1 x^{-2}, B = B_2 + B_1 x^{-2}$, where

$$A_1 = a_1 x + a_0,\ A_2 = a_4 x^2 + a_3 x + a_2,$$
$$B_1 = b_2 x^2 + b_1 x + b_0,\ B_2 = b_4 x + b_3.$$

According to equation (5), then

$$\begin{aligned} A \cdot B &= [(A_2 x^2 + A_2) B_2 x^3 + (A_1 x^2 + A_1) B_1 \\ &\quad + (A_2 + A_1)(B_2 x + B_1) x^2] x^{-4} \\ &= S_1 x^{-4} + S_2 x^{-2}. \end{aligned}$$

Therefore, the matrices $\mathbf{A}'$ and $\mathbf{U}'$ are given by:

$$\mathbf{A}' = \begin{array}{r} -4 \\ -3 \\ -2 \\ -1 \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{bmatrix} a_0 & 0 & 0 & 0 & 0 \\ a_1 & a_0 & 0 & 0 & 0 \\ a_0 & a_1 & a_0 & 0 & 0 \\ a_1 & a_0 & a_1 & a_2 & 0 \\ 0 & a_1 & a_0 & a_3 & a_2 \\ 0 & 0 & a_1 & a_2+a_4 & a_3 \\ 0 & 0 & 0 & a_3 & a_2+a_4 \\ 0 & 0 & 0 & a_4 & a_3 \\ 0 & 0 & 0 & 0 & a_4 \end{bmatrix}, \tag{8}$$

and

$$\mathbf{U}' = \begin{array}{r} -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{array} \begin{bmatrix} u_0 & 0 & 0 \\ u_1 & u_0 & 0 \\ u_2 & u_1 & u_0 \\ 0 & u_2 & u_1 \\ 0 & 0 & u_2 \end{bmatrix}, \tag{9}$$

where $u_2 = a_4, u_1 = a_3 + a_1, u_0 = a_0 + a_2$. We have $S_1 x^{-4} = \mathbf{A}' \cdot \mathbf{b}, S_2 x^{-2} = \mathbf{U}' \cdot \mathbf{v}$.

2. Please note that the matrix in the right side actually contains $4n = 2m$ rows, but the last row is $\mathbf{0}$, which does not affect the result.

3.2 Reduction process

It is easy to check that the products S_1x^{-2k}, S_2x^{n-2k} contain the terms of degrees out of the range $[-k, m-k-1]$. To compute the field multiplication, we have to perform the reduction operation for these two expressions. According to Mastrovito scheme, the reduction operation can be regarded as the construction of product matrices from $\mathbf{A'}$ and $\mathbf{U'}$ using the equation $x^m = x^k + 1$. Denoted by $\mathbf{M}_A$ and $\mathbf{M}_U$ the product matrices of $S_1x^{-2k} \bmod f(x)$ and $S_2x^{n-2k} \bmod f(x)$, respectively. In the following, we investigate the construction details for these two matrices.

3.2.1 Construction of $\mathbf{M}_A$

Note that the matrix $\mathbf{A'}$ contains $2m-2$ rows, each of which corresponds to the polynomial degree from $-2k$ to $2m-2-2k$, we can reduce the rows labeled by $\{-2k, -2k+1, \cdots, -k-1\}$ (the first k rows), and by $\{m-k, m-k+1, \cdots, 2m-2-k\}$ (the last $m-k-1$ rows) of the matrix $\mathbf{A'}$ to obtain $\mathbf{M}_A$. According to the form of $f(x) = x^m + x^k + 1$ and the SPB definition, we mainly utilize following equations:

$$\begin{cases} x^i = x^{m+i} + x^{i+k}, \text{ for } i = -2k, \cdots, -k-1; \\ x^i = x^{i-m+k} + x^{i-m}, \text{ for } i = m-k, m-k+1, \\ \qquad\qquad\qquad\qquad\qquad \cdots, 2m-2k-2. \end{cases} \tag{10}$$

Corresponding to (10), the computation of $\mathbf{M}_A$ consists of adding the rows of $\mathbf{A'}$ labeled by i to those rows labeled by $m+i, i+k$ (or $i-m+k, i-m$). More explicitly, we eliminate 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 eliminate the row $m-k, \cdots, 2m-2k-2$ by adding them to the row $0, \cdots, m-k-2$ and $-k, \cdots, m-2k-2$. These operations can be implemented by matrix addition. We will utilize following three $m \times m$ matrices:

$$\begin{aligned} \mathbf{A}_U =& [\mathbf{A'}(1,:)^T, \cdots, \mathbf{A'}(k,:)^T, \underbrace{\mathbf{0}, \cdots, \mathbf{0}}_{m-k}]^T, \\ \mathbf{A}_M =& [\mathbf{A'}(k+1,:)^T, \cdots, \mathbf{A'}(k+m,:)^T]^T, \\ \mathbf{A}_L =& [\underbrace{\mathbf{0}, \cdots, \mathbf{0}}_{k+1}, \mathbf{A'}(k+m+1,:)^T, \cdots, \mathbf{A'}(2m-1,:)^T]^T. \end{aligned}$$

Obviously, the nonzero parts of $\mathbf{A}_U$, $\mathbf{A}_L$ are the rows of $\mathbf{A'}$ whose labels are out of the range $[-k, m-k-1]$. Then, the product matrix M_A is obtained as follows:

$$\begin{aligned} \mathbf{M}_A =& \mathbf{A}_M + \mathbf{A}_U + \mathbf{A}_U[\downarrow (m-k)] \\ &+ \mathbf{A}_L[\uparrow 1] + \mathbf{A}_L[\uparrow (k+1)]. \end{aligned} \tag{11}$$

The structure of $\mathbf{M}_A$ highly influences the efficiency of our multiplier, thus, we make several important observations about the construction of $\mathbf{M}_A$.

Observation 3.1. If m is even, $\mathbf{A}_M + \mathbf{A}_U[\downarrow (m-k)] + \mathbf{A}_L[\uparrow (k+1)]$ is

$$\begin{bmatrix} \mathbf{A}_{L1} + \mathbf{A}_{L2}, & \mathbf{A}_{H1} + \mathbf{A}_{H2} \\ \mathbf{A}_{L1} + \mathbf{A}_{L2}, & \mathbf{A}_{H1} + \mathbf{A}_{H2} \end{bmatrix} [\circlearrowleft k]. \tag{12}$$

If m is odd, such matrix can be rewritten as

$$\left[\begin{pmatrix} \mathbf{A'}_{L1} + \mathbf{A'}_{L2} \\ \mathbf{A}_{L1} + \mathbf{A}_{L2} \end{pmatrix} [\circlearrowleft 1], \quad \begin{matrix} \mathbf{A}_{H1} + \mathbf{A}_{H2} \\ \mathbf{A'}_{H1} + \mathbf{A'}_{H2} \end{matrix} \right] [\circlearrowleft k]. \tag{13}$$

where $\mathbf{A'}_{L1} = \mathbf{A}_{L1}[\downdownarrows 1]$, $\mathbf{A'}_{H1} = \mathbf{A}_{H1}[\downdownarrows 1]$ and $\mathbf{A'}_{L2} = \mathbf{A}_{L2}[\upuparrows 1]$, $\mathbf{A'}_{H2} = \mathbf{A}_{H2}[\upuparrows 1]$.

Observation 3.2. Random matrix-vector multiplication $\mathbf{A}[\circlearrowleft i] \cdot \mathbf{b}$ over $\mathbb{F}_2$ costs the same logic gates as $\mathbf{A} \cdot \mathbf{b}$.

The proof of observation 3.2 is direct, and the proof of observation 3.1 can be found in the appendix. Furthermore, it is also clear that

$$\begin{aligned} \mathbf{A}_U + \mathbf{A}_L[\uparrow 1] =& [\mathbf{A'}(1,:)^T, \cdots, \mathbf{A'}(k,:)^T, \mathbf{A'}(k+m+1,:)^T, \\ & \cdots, \mathbf{A'}(2m-1,:)^T, \mathbf{0}] \end{aligned} \tag{14}$$

Denoted by $\mathbf{M}_{A,1}$ the matrix presented either in (12) or (13), and by $\mathbf{M}_{A,2}$ the matrix presented in (14). Combined with Observation 3.1, (11) can be simplified as:

$$\mathbf{M}_A = \mathbf{M}_{A,1} + \mathbf{M}_{A,2}.$$

Therefore, we can utilize the same strategy presented in [19], where $S_1x^{-2k} \bmod f(x)$ can be implemented as:

$$\begin{aligned} S_1x^{-2k} \bmod f(x) &= \textstyle\sum_{i=0}^{m-1} s_i x^i \\ &= \mathbf{M}_A \cdot \mathbf{b} \\ &= \mathbf{M}_{A,1} \cdot \mathbf{b} + \mathbf{M}_{A,2} \cdot \mathbf{b} \\ &= [\mathbf{M}_{A,1}, \mathbf{M}_{A,2}] \cdot [\mathbf{b}, \mathbf{b}]^T. \end{aligned} \tag{15}$$

We implement the above expression by following steps:

- Perform line-vector products
$$\begin{aligned} &[\mathbf{M}_{A,1}(i,1) \cdot b_0, \cdots, \mathbf{M}_{A,1}(i,m) \cdot b_{m-1}, \\ &\quad \mathbf{M}_{A,2}(i,1) \cdot b_0, \cdots, \mathbf{M}_{A,2}(i,m) \cdot b_{m-1}], \end{aligned} \tag{16}$$
$i = 1, 2, \cdots, m$ in parallel.
- Sum up all the $2m$ entries of each row using binary XOR tree, i.e.,
$$s_{i-1} = \sum_{j=1}^{m} \mathbf{M}_{A,1}(i,j) \cdot b_{j-1} + \sum_{j=1}^{m} \mathbf{M}_{A,2}(i,j) \cdot b_{j-1},$$
$i = 1, 2, \cdots, m$.

Special case $m = 2k$. When m is even and $m = 2k$ ($k = n$), we can obtain the simplest form of $\mathbf{M}_{A,1}$ and $\mathbf{M}_{A,2}$, where

$$\mathbf{M}_{A,1} = \begin{bmatrix} \mathbf{A}_{L1} + \mathbf{A}_{L2}, & \mathbf{A}_{H1} + \mathbf{A}_{H2} \\ \mathbf{A}_{L1} + \mathbf{A}_{L2}, & \mathbf{A}_{H1} + \mathbf{A}_{H2} \end{bmatrix},$$

and

$$\mathbf{M}_{A,2} = \begin{bmatrix} \mathbf{A}_{L1}, & \mathbf{0}_{n\times n} \\ \mathbf{0}_{n\times n}, & \mathbf{A}_{H2} \end{bmatrix}.$$

By swapping and combining some overlapped entries, expression (15) now can be rewritten as:

$$\begin{aligned} \mathbf{M}_{A,1} \cdot \mathbf{b} + \mathbf{M}_{A,2} \cdot \mathbf{b} &= \begin{bmatrix} \mathbf{A}_{L2}, & \mathbf{A}_{H1} \\ \mathbf{A}_{L2}, & \mathbf{A}_{H1} \end{bmatrix} \cdot \mathbf{b} \\ &+ \begin{bmatrix} \mathbf{0}_{n\times n}, & \mathbf{A}_{H2} \\ \mathbf{A}_{L1}, & \mathbf{0}_{n\times n} \end{bmatrix} \cdot \mathbf{b}. \end{aligned} \tag{17}$$

In this case, we just compute two submatrix-vector multiplication and add them up to obtain $S_1x^{-2k} \bmod f(x)$.

3.2.2 Construction of $\mathbf{M}_U$

Analogous with the construction procedure of $\mathbf{M}_A$, the reduction form of $\mathbf{U}'$ can also be obtained by shifting of $\mathbf{U}'$ and adding specific rows of $\mathbf{U}'$ to itself. The form of $\mathbf{M}_U$ varied according to the values of m and k. There are six cases need to be considered:

1) m even, $m > 2k+2$;
2) m even, $m = 2k$ or $m = 2k+2$;
3) m even, $m < 2k$;
4) m odd, $m > 2k+1$;
5) m odd, $m = 2k+1$;
6) m odd, $m < 2k+1$.

Case 1 and 4: Note that the following equation holds with respect to these two cases:

$$\begin{aligned} &n-2k > -k+1, && m \text{ is even and } m > 2k+2,\\ &n-2k > -k, && m \text{ is odd and } m > 2k+1. \end{aligned}$$

We only need to reduce the rows which are labeled with $x^{m-k}, \cdots, x^{m+n-2k-2} (x^{m+n-2k-1}$ for odd $m)$ of $\mathbf{U}'$. The equation $x^i = x^{i-m+k} + x^{i-m}, i = m-k, \cdots, m+n-2k-2$ is used for reduction. Then the product matrix $\mathbf{M}_U$ is partitioned as $\mathbf{M}_U = \mathbf{M}_{U,1} + \mathbf{M}_{U,2}$, where

$$\begin{aligned} &\mathbf{M}_{U,1} =\\ &\quad [\mathbf{U}_2(k+1,:)^T, \cdots, \mathbf{U}_2(n-1,:)^T, \mathbf{0}, \mathbf{U}_1(1,:)^T, \cdots,\\ &\qquad \mathbf{U}_1(n,:)^T, \mathbf{U}_2(1,:)^T, \cdots, \mathbf{U}_2(k,:)^T]^T, (m \text{ even}),\\ &\quad [\mathbf{U}_2(k+1,:)^T, \cdots, \mathbf{U}_2(n,:)^T, \mathbf{U}_1(1,:)^T, \cdots,\\ &\qquad \mathbf{U}_1(n+1,:)^T, \mathbf{U}_2(1,:)^T, \cdots, \mathbf{U}_2(k,:)^T]^T, (m \text{ odd}), \end{aligned}$$

and

$$\begin{aligned} &\mathbf{M}_{U,2} =\\ &[\underbrace{\mathbf{0}, \cdots, \mathbf{0}}_{k}, \mathbf{U}_2(k+1,:)^T, \cdots, \mathbf{U}_2(n-1,:)^T, \underbrace{\mathbf{0}, \cdots, \mathbf{0}}_{n+1}]^T, (m \text{ even}),\\ &[\underbrace{\mathbf{0}, \cdots, \mathbf{0}}_{k}, \mathbf{U}_2(k+1,:)^T, \cdots, \mathbf{U}_2(n,:)^T, \underbrace{\mathbf{0}, \cdots, \mathbf{0}}_{n+1}]^T, (m \text{ odd}). \end{aligned}$$

Case 2 and Case 5: In these two case, one can check that $S_2 x^{n-2k} \bmod f(x) = S_2 x^{n-2k}$, thus it requires no reduction for the matrix $\mathbf{U}'$ and the product matrix $\mathbf{M}_U$ is equal to $\mathbf{U}'$ itself.

Case 3 and Case 6: Since $m < 2k$ (or $m < 2k+1$), we have $n < k$ for both even and odd m. The following formula holds:

$$\begin{aligned} &m+n-2k-2 < m-k-2, && m \text{ is even and } m < 2k,\\ &m+n-2k-1 < m-k-1, && m \text{ is odd and } m < 2k+1. \end{aligned}$$

Therefore, we only need to reduce the rows which are labeled with $x^{n-2k}, \cdots, x^{-k-1}$ of $\mathbf{U}'$. The equation $x^i = x^{m+i} + x^{k+i}, i = n-2k, \cdots, -k-1$ is used for reduction. Analogous to Case 1 and 4, the product matrix $\mathbf{M}_U = \mathbf{M}_{U,1} + \mathbf{M}_{U,2}$, where

$$\begin{aligned} &\mathbf{M}_{U,1} =\\ &\quad [\mathbf{U}_1(k-n+1,:)^T, \cdots, \mathbf{U}_1(n,:)^T, \mathbf{U}_2(1,:)^T, \cdots,\\ &\qquad \mathbf{U}_2(n-1,:)^T, \mathbf{0}, \mathbf{U}_1(1,:)^T, \cdots, \mathbf{U}_1(k-n,:)^T]^T, (m \text{ even}),\\ &\quad [\mathbf{U}_1(k-n+1,:)^T, \cdots, \mathbf{U}_1(n+1,:)^T, \mathbf{U}_2(1,:)^T, \cdots,\\ &\qquad \mathbf{U}_2(n+1,:)^T, \mathbf{U}_1(1,:)^T, \cdots, \mathbf{U}_1(k-n,:)^T]^T, (m \text{ odd}), \end{aligned}$$

and

$$\mathbf{M}_{U,2} = [\underbrace{\mathbf{0}, \cdots, \mathbf{0}}_{n}, \mathbf{U}_1(1,:)^T, \cdots, \mathbf{U}_1(k-n,:)^T, \underbrace{\mathbf{0}, \cdots, \mathbf{0}}_{m-k}]^T.$$

3.3 Complexity Analysis

3.3.1 Complexity analysis for $S_1 x^{-2k} \bmod f(x)$

Applying Observation 3.1 and 3.2, we have following theorem.

Theorem 1 *The m line-vector products of (16) only requires $\frac{m^2}{2}$ AND gates for even m and $\frac{m^2-1}{2}$ AND gates for odd m.*

Proof According to (6), (7) and (15), it is clear that the non-zero entries of $\mathbf{M}_{A,2}$ are included in four sub-matrices $\mathbf{A}_{L1}, \mathbf{A}_{L2}$, $\mathbf{A}_{H1}, \mathbf{A}_{H2}$. From (10) and (11), we know $\mathbf{M}_{A,1}$ contains all the non-zero entries of these four matrices. Therefore, computing the line-vector products of $\mathbf{M}_{A,1} \cdot \mathbf{b}$ is sufficient to obtain all the line-vector products for $\mathbf{M}_A \cdot \mathbf{b}$. If m is even, it requires only $n^2 + n^2 = \frac{m^2}{2}$ AND gates to compute these products. If m is odd, it requires $(n+1) \cdot n + (n+1) \cdot n = \frac{m^2-1}{2}$ AND gates. □

Next, we investigate the number of XOR gates needed in (15). Notice that $\mathbf{M}_{A,1}$ and $\mathbf{M}_{A,2}$ shares some common entries. If these common entries multiply the same vector entries during the computation of line-vector products presented in (16), the results stay the same. When adding up all the result of each row, some XOR gates can be saved by sharing the common items. This technique is so called *sub-expression sharing* [7]. However, when we use binary XOR tree to sum up the products of each row, the common items between two binary trees should be utilized carefully. For example, two expressions $c_1 = z_0 + z_1 + z_2 + z_3$ and $c_2 = z_0 + z_2 + z_3 + z_4$ share three common items, but only one XOR will be saved when using binary tree. The computation details are as follows:

$$c_1 = \underbrace{[\underbrace{[z_0 + z_2]}_{\text{I}} + \underbrace{[z_1 + z_3]}_{\text{I}}]}_{\text{II}}, \quad c_2 = \underbrace{[\underbrace{[z_0 + z_2]}_{\text{I}} + \underbrace{[z_3 + z_4]}_{\text{I}}]}_{\text{II}}.$$

The sub-expressions labeled by the same Roman number are calculated simultaneously. It is clear that the common item z_3 cannot save one more XOR gates unless the binary tree is not used.

In [6], [16], the authors have shown that if two binary XOR trees share k common items, then $k - W(k)$ XOR gates can be saved, where $W(k)$ is the Hamming weight of the binary representation of integer k. Obviously, one can check that c_1, c_2 of above expression can save only $3 - W(3) = 1$ XOR gate. Thus, we use the similar trick to save XOR gates in the accumulating processes. Firstly, we utilize n intermediate values $P_0, P_1, \cdots, P_{n-1}$ ($n+1$ values for odd m), where

$$[P_0, \cdots, P_{n-1}]^T = [\mathbf{A}_{L1} + \mathbf{A}_{L2}, \mathbf{A}_{H1} + \mathbf{A}_{H2}] \cdot \mathbf{b},$$

if m is even, or

$$[P_0, \cdots, P_n]^T = \left[\begin{array}{c} \mathbf{A}_{L1} + \mathbf{A}_{L2} \\ (\mathbf{A'}_{L1} + \mathbf{A'}_{L2})[1, :] \end{array}, \mathbf{A'}_{H1} + \mathbf{A'}_{H2} \right] \cdot \mathbf{b},$$

if m is odd.

Please note that both $\mathbf{M}_{A,1}$ and $\mathbf{M}_{A,2}$ shares some common entries with the matrices appeared in above formulae. The computation of the coefficients of $S_1 x^{-2k} \bmod f(x)$ can use these n (or $n+1$) intermediate values. For example, since $s_0 = \mathbf{M}_{A,1}(1:) \cdot \mathbf{b} + \mathbf{M}_{A,2}(1:) \cdot \mathbf{b}$, when m is even and $0 < k < n$, $\mathbf{M}_{A,1}(1:) \cdot \mathbf{b}$ has m items overlapped with P_{n-k} and $\mathbf{M}_{A,2}(1:) \cdot \mathbf{b}$ has 1 terms overlapped with P_0. Notice that only 1 common items is shared by s_0 and P_0, no XOR gates ($1 - W(1) = 0$) will saved according to previous assertion. Thus the computation of s_0 actually cost $m - 1 - (m - W(m)) + 1 = W(m)$ XOR gates. Table 1 indicates the explicit number of XOR gates required by each s_i if m is even and $0 < k < n$. For simplicity purpose, we divide each coefficient s_i into two parts and indicate their overlapped values, independently. One extra XOR gate for addition between $\mathbf{M}_{A,1}(i:) \cdot \mathbf{b}$ and $\mathbf{M}_{A,2}(i:) \cdot \mathbf{b}$ ($i = 1, 2, \cdots, m-1$) of each row is already counted into the second column. One can find more details in the appendix.

In this case, notice that the computation of the intermediate values $P_0, P_1, \cdots P_{n-1}$ requires $(m-1)n = \frac{m^2 - m}{2}$ XOR gates, plus the number of XOR gates presented in Table 1, we can obtain the explicit number of XOR gates required by $S_1 x^{-2k} \bmod f(x)$:

$$\#\text{XOR: } \frac{m^2 - 3m}{2} + \sum_{i=1}^{k} W(i) + \sum_{i=1}^{m-k-1} W(i) + mW(m).$$

The circuit delay of $S_1 x^{-2k} \bmod f(x)$ equals the depth of the biggest XOR tree. According to (12) and (13), it is easy to check that

$$\text{Delay: } T_A + \lceil \log_2(2m - k - 1) \rceil T_X.$$

Note that the number of #AND gates for $S_1 x^{-2k} \bmod f(x)$ is already given by theorem 1. The space and time complexity of other cases are summarized in table 2.

Special case $m = 2k$. According to (17), the first submatrix has its upper $\frac{m}{2}$ rows equal its lower $\frac{m}{2}$ rows, thus we only need to compute $[\mathbf{A}_{L2}, \mathbf{A}_{H1}] \cdot \mathbf{b}$. In addition, both $\mathbf{A}_{L2}$ and $\mathbf{A}_{H1}$ are triangular matrices, one can easily check that each row of $[\mathbf{A}_{L2}, \mathbf{A}_{H1}]$ consists of at most $\frac{m}{2}$ nonzero entries. Thus such matrix-vector multiplication cost $\frac{m^2}{4}$ AND gates and $\frac{m^2 - 2m}{4}$ XOR gates with delay of $T_A + \lceil \log_2(\frac{m}{2}) \rceil T_X$. The computation of the second submatrix-vector multiplication is similar, it totally requires $\frac{m^2}{4}$ AND gates and $\frac{m^2}{4} - m + 1$ XOR gates with delay of $T_A + \lceil \log_2(\frac{m}{2}) \rceil T_X$. Finally, $m-1$ XOR gates are needed to add these two results up which lead one more T_X delay.

Example 3.2. Consider the reduction of $\mathbf{A}'$ presented in Example 3.1. The construction of $\mathbf{M}_A$ is based on following equation:

$$\begin{cases} x^i = x^{5+i} + x^{i+2}, \text{ for } i = -4, -3; \\ x^i = x^{i-3} + x^{i-5}, \text{ for } i = 3, 4. \end{cases}$$

Then $\mathbf{M}_A = \mathbf{M}_{A,1} + \mathbf{M}_{A,2}$ where

$$\mathbf{M}_{A,1} = \begin{array}{c} -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{array} \left[\begin{array}{ccc|cc} a_0 & a_1 & a_0 & a_4 & a_3 \\ a_1 & a_0 & a_1 & a_2 & a_4 \\ \hline 0 & a_1 & a_0 & a_3 & a_2 \\ a_0 & 0 & a_1 & a_2 + a_4 & a_3 \\ a_1 & a_0 & 0 & a_3 & a_2 + a_4 \end{array} \right],$$

and

$$\mathbf{M}_{A,2} = \begin{array}{c} -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{array} \left[\begin{array}{ccc|cc} a_0 & 0 & 0 & 0 & 0 \\ a_1 & a_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_4 & a_3 \\ 0 & 0 & 0 & 0 & a_4 \\ 0 & 0 & 0 & 0 & 0 \end{array} \right].$$

Then $S_1 x^{-4} \bmod x^5 + x^2 + 1 = [\mathbf{M}_{A,1}, \mathbf{M}_{A,2}] \cdot [\mathbf{b}, \mathbf{b}]^T$. Firstly, it is easy to see that $\frac{5^2 - 1}{2} = 12$ AND gates are needed for the bitwise multiplication. Then we evaluate the number of required XOR gates. According to the description of Subsection 3.3, we use three intermediate values:

$$\begin{aligned} P_0 &= a_1 b_1 + a_0 b_2 + a_4 b_3 + a_3 b_4 + a_0 b_0, \\ P_1 &= a_1 b_2 + a_2 b_3 + a_4 b_4 + a_1 b_0 + a_0 b_1, \\ P_2 &= a_3 b_3 + a_2 b_4 + a_1 b_1 + a_0 b_2. \end{aligned}$$

As show in Table 8, each vector of $\mathbf{M}_{A,1} \cdot \mathbf{b}$ and $\mathbf{M}_{A,2} \cdot \mathbf{b}$ shares some common items of P_0, P_1, P_0 and certain number of XOR gates will be saved. For example, the binary tree of $\mathbf{M}_{A,1}(4,:) \cdot \mathbf{b} + \mathbf{M}_{A,2}(4,:) \cdot \mathbf{b}$ is as follows:

$$\underbrace{[\underline{a_1 b_2 + a_2 b_3}] + [a_4 b_4}_{P_1} + \underbrace{a_4 b_3] + [\underline{a_3 b_4 + a_0 b_0}]}_{P_0}, \tag{18}$$

while the binary tree of $\mathbf{M}_{A,1}(5,:) \cdot \mathbf{b} + \mathbf{M}_{A,2}(5,:) \cdot \mathbf{b}$ is:

$$\underbrace{[\underline{a_3 b_3 + a_2 b_4}]}_{P_2} + \underbrace{a_4 b_4 + [\underline{a_1 b_0 + a_0 b_1}]}_{P_1}. \tag{19}$$

Additions in the square brackets of the above expressions are compute simultaneously, and the underlined ones can be saved as we compute them in P_0, P_1, P_2. One can check that (18) and (19) require $W(3) + W(2) - 1 + W(1) = 3$ and $W(3) + W(2) - 1 = 2$ XOR gates, respectively. Consequently, we totally need 18 [3] XOR gates for $S_1 x^{-4} \bmod x^5 + x^2 + 1$ with delay $T_A + (\lceil \log_2 7 \rceil T_X) = T_A + 3T_X$.

3. More XOR gates can be reused in this example, but we prefer to use the general formulae presented in Table 9.

TABLE 1
The overlapped values, m even, $0 < k < n$

first part	Overlapped	#XOR	second part	Overlapped	#XOR
$\mathbf{M}_{A,1}(1:)\cdot\mathbf{b}$	P_k	$W(m)-1$	$\mathbf{M}_{A,2}(1:)\cdot\mathbf{b}$	P_0	$W(1)$
$\mathbf{M}_{A,1}(2:)\cdot\mathbf{b}$	P_{k+1}	$W(m)-1$	$\mathbf{M}_{A,2}(2:)\cdot\mathbf{b}$	P_1	$W(2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{M}_{A,1}(n-k:)\cdot\mathbf{b}$	P_{n-1}	$W(m)-1$	$\mathbf{M}_{A,2}(k:)\cdot\mathbf{b}$	P_{k-1}	$W(k)$
$\mathbf{M}_{A,1}(n-k+1:)\cdot\mathbf{b}$	P_0	$W(m)-1$	$\mathbf{M}_{A,2}(k+1:)\cdot\mathbf{b}$	P_{k+1}	$W(m-k-1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{M}_{A,1}(m-k:)\cdot\mathbf{b}$	P_{n-1}	$W(m)-1$	$\mathbf{M}_{A,2}(n-1:)\cdot\mathbf{b}$	P_{n-1}	$W(n+1)$
$\mathbf{M}_{A,1}(m-k+1:)\cdot\mathbf{b}$	P_0	$W(m)-1$	$\mathbf{M}_{A,2}(n)\cdot\mathbf{b}$	P_0	$W(n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{M}_{A,1}(m:)\cdot\mathbf{b}$	P_{k-1}	$W(m)-1$	$\mathbf{M}_{A,2}(m-1:)\cdot\mathbf{b}$	P_{n-1}	$W(1)$

TABLE 2
The space and time complexity of $S_1x^{-2k} \bmod f(x)$

case	#AND	#XOR	Delay
m even, $m < 2k$	$\frac{m^2}{2}$	$\frac{m^2-3m}{2} + \sum_{i=1}^{k} W(i) + \sum_{i=1}^{m-k-1} W(i) + mW(m)$	$T_A + (\lceil\log_2(m+k)\rceil)T_X$
m even, $m = 2k$	$\frac{m^2}{2}$	$\frac{m^2-m}{2}$	$T_A + (1+\lceil\log_2(\frac{m}{2})\rceil)T_X$
m odd, $m \geq 2k+1$ $n = \frac{m-1}{2}$	$\frac{m^2-1}{2}$	$\frac{m^2-2m-1}{2} + (n+1)W(m) + nW(n+1) + (m-k-1)W(n)$ $\sum_{i=1}^{k} W(i) + \sum_{i=1}^{n} W(i) + \sum_{i=1}^{n-k} W(i)$	$T_A + \lceil\log_2(2m-k-1)\rceil)T_X$
m odd, $m < 2k+1$ $n = \frac{m-1}{2}$	$\frac{m^2-1}{2}$	$\frac{m^2-2m-1}{2} + (n+1)W(m) + nW(n+1) + kW(n)$ $\sum_{i=1}^{m-k-1} W(i) + \sum_{i=1}^{n} W(i) + \sum_{i=1}^{k-n} W(i)$	$T_A + (\lceil\log_2(m+k)\rceil)T_X$

3.3.2 *Complexity analysis for* $S_2x^{n-2k} \bmod f(x)$

Apparently, we compute $S_2x^{n-2k} \bmod f(x)$ as follows:

$$\begin{aligned} S_2x^{n-2k} \bmod f(x) &= \mathbf{M}_U \cdot \mathbf{v} \\ &= \mathbf{M}_{U,1}\cdot\mathbf{v} + \mathbf{M}_{U,2}\cdot\mathbf{v} \\ &= [\mathbf{M}_{U,1}, \mathbf{M}_{U,2}] \cdot [\mathbf{v},\mathbf{v}]^T \end{aligned} \tag{20}$$

The computation of $S_2x^{n-2k} \bmod f(x)$ consists of the precomputation of U, V and matrix-vector multiplication presented as above. Firstly, $2n$ XOR gates are needed for precomputation of U, V which cost one T_X in parallel. Then, note that there are some common items between $\mathbf{M}_{U,1}$ and $\mathbf{M}_{U,2}$, the matrix-vector multiplication $[\mathbf{M}_{U,1}, \mathbf{M}_{U,2}] \cdot [\mathbf{v},\mathbf{v}]^T$ follows the same line of the computing strategy presented in subsection 3.3.1. One T_A is needed for bitwise parallel multiplication and the required number of T_X is equal to the depth of biggest binary trees related to non-zero entries of each row of $[\mathbf{M}_{U,1}\cdot\mathbf{v}, \mathbf{M}_{U,2}\cdot\mathbf{v}]$, which is varied according to m and k. For simplicity, we only evaluate the upper bound of the number of T_X for some cases. The space and time complexity of (20) is summarized in table 3.

Example 3.3. Consider the reduction of $\mathbf{U}'$ in Example 3.1. Since $5 = 2\cdot 2+1$, there is no reduction needed here. We have $\mathbf{M}_U = \mathbf{U}'$. In addition, the delay of $\mathbf{M}_U \cdot \mathbf{v}$ is $T_A + \lceil\log_2 3\rceil T_X = T_A + 2T_X$. Plus one T_X for computation U, V, the delay of $S_2x^{-2} \bmod x^5+x^2+1$ is $T_A + 3T_X$ which is no bigger than that of $S_1x^{-4} \bmod x^5+x^2+1$.

4 Complexity and Comparison

4.1 Theoretic complexity

Based on the delay of the two expressions presented in Table 2 and 3, we immediately have following proposition.

Proposition 1 *Let T_{s_1} and T_{s_2} denote the delay of S_1x^{-2k} and S_2x^{n-2k} modulo $f(x)$, respectively. Then T_{s_1} is no bigger than T_{s_2}.*

Proof According to circuit delay expressions in Table 2 and 3, it is clear that both T_{s_1} and T_{s_2} contain 1 T_A. We only need to compare the coefficients of T_X in these tables. Still consider the six cases indicated in subsection 3.2.2.

Case 1: $1 + \lceil\log_2(m-k-1)\rceil \leq \lceil\log_2(2m-k-1)\rceil$,
Case 2: $1 + \lceil\log_2(\frac{m}{2})\rceil \leq \lceil\log_2 m\rceil$,
Case 3: $1 + \lceil\log_2(k)\rceil \leq \lceil\log_2(m+k)\rceil$,
Case 4: $1 + \lceil\log_2(m-k-1)\rceil \leq \lceil\log_2(2m-k)\rceil$,
Case 5: $1 + \lceil\log_2(\frac{m-1}{2})\rceil \leq \lceil\log_2(m+k)\rceil$,
Case 6: $1 + \lceil\log_2(k)\rceil \leq \lceil\log_2(m+k)\rceil$.

One can directly check that T_{s_2} is smaller or at most equal to T_{s_1}, which conclude the proposition. $\square$

Proposition 1 ensure that $S_1x^{-2k} \bmod f(x)$ and $S_2x^{n-2k} \bmod f(x)$ can be implemented simultaneously, and the final time delay is T_{s_1}. Then we add up these

TABLE 3
The space and time complexity of $S_2 x^{n-2k} \bmod f(x)$

case	#AND	#XOR	Delay
1	$\frac{m^2}{4}$	$\frac{m^2}{4}+\sum_{i=1}^{\frac{m}{2}-k-1}W(i)+1$	$< T_A+(1+\lceil\log_2(m-k-1)\rceil)T_X$
2	$\frac{m^2}{4}$	$\frac{m^2}{4}+1$	$T_A+(1+\lceil\log_2(\frac{m}{2})\rceil)T_X$
3	$\frac{m^2}{4}$	$\frac{m^2}{4}+\sum_{i=1}^{k-\frac{m}{2}}W(i)+1$	$< T_A+(1+\lceil\log_2 k\rceil)T_X$
4	$\frac{m^2+2m+1}{4}$	$\frac{m^2+2m-3}{4}+\sum_{i=1}^{\frac{m-1}{2}-k}W(i)$	$< T_A+(1+\lceil\log_2(m-k-1)\rceil)T_X$
5	$\frac{m^2+2m+1}{4}$	$\frac{m^2+2m-3}{4}$	$T_A+(1+\lceil\log_2(\frac{m-1}{2})\rceil)T_X$
6	$\frac{m^2+2m+1}{4}$	$\frac{m^2+2m-3}{4}+\sum_{i=1}^{k-\frac{m-1}{2}}W(i)$	$< T_A+(1+\lceil\log_2 k\rceil)T_X$

values to obtain the ultimate result which requires m XOR gates and one T_X in parallel. As a result, we obtain the total space complexity of the proposed multiplier by summing up all these related expressions.

If m is even:

$$\begin{aligned}&\#\text{AND}: \tfrac{3m^2}{4},\\ &\#\text{XOR}: \tfrac{3m^2}{4}-\tfrac{m}{2}+O(m\log_2 m)^*,\\ &\text{Delay}: \begin{cases} T_A+(1+\lceil\log_2(2m-k-1)\rceil)\,T_X,\,(m>2k),\\ T_A+(1+\lceil\log_2(m+k)\rceil)\,T_X,\,(m<2k).\end{cases}\end{aligned} \tag{21}$$

If m is odd:

$$\begin{aligned}&\#\text{AND}: \tfrac{3m^2+2m-1}{4},\\ &\#\text{XOR}: \tfrac{3m^2}{4}+\tfrac{m}{2}+O(m\log_2 m)^*,\\ &\text{Delay}: \begin{cases} T_A+(1+\lceil\log_2(2m-k-1)\rceil)\,T_X,\,(m\geq 2k+1),\\ T_A+(1+\lceil\log_2(m+k)\rceil)\,T_X,\,(m<2k+1).\end{cases}\end{aligned} \tag{22}$$

We note that the expressions for number of XOR gates in Table 2 and 3 contain the sum of hamming weights related to certain integer σ, denoted by $\sum_{i=1}^{\sigma}W(i)$. This expression can be roughly written as $\frac{\sigma}{2}\log_2\sigma$ [6], [16]. Notice that such $\sigma \leq m$, thus we use the expression with $O(m\log_2 m)^*$ instead to make it simpler. If m is even and $m=2k$, the explicit formula with respect to the space and time complexities are given in Table 5.

4.2 Comparison

As the most important contribution of this study, the time delay of our multiplier is summarized in following table.

TABLE 4
Time delay for SPB multiplier using $f(x)=x^m+x^k+1$

$m<2k$	$T_A+(1+\lceil\log_2(m+k)\rceil)\,T_X$
$m=2k$	$T_A+(1+\lceil\log_2 m\rceil)\,T_X$
$2k\leq m-1$	$T_A+(1+\lceil\log_2(2m-k-1)\rceil)\,T_X$

Compared with the fastest bit-parallel multiplier [13], [14], our proposal only requires one more T_X. In addition, it is especially attractive if the corresponding circuit delay is $T_A+(1+\lceil\log_2 m\rceil)T_X$, this happens only if m,k satisfy:

$$\begin{aligned}\lceil\log_2(m+k)\rceil &= \lceil\log_2 m\rceil,\ m<2k,\\ \lceil\log_2(2m-k-1)\rceil &= \lceil\log_2 m\rceil,\ 2k\leq m-1.\end{aligned}$$

In fact, for the range $100\leq m\leq 1023$ with cryptographic interests, there are 1405 irreducible trinomials and 457 trinomials satisfy the above condition.

In Table 5, we give a comparison of several different bit-parallel multipliers for irreducible trinomials. All these multipliers are using PB representations except particular description. It is clear that our scheme is faster than other Karatsuba-based multiplier and still has roughly 25% logic gates gain. In [9], [15], [21], the authors investigated the speedup of Karatsuba multiplier independently. None of them had given such a precise time bound for all the trinomials. To the best of our knowledge, this is the first time to show that Karatsuba-based multiplier can always be only $1T_X$ slower than the fastest bit-parallel multipliers which were previously proposed.

5 Conclusion

In this paper, we have constructed a matrix-vector form of Karatsuba algorithm, based on which a novel bit-parallel $GF(2^m)$ multiplier is proposed. Mastrovito scheme and shifted polynomial basis are combined together to reduce the gates delay. New sub-expression sharing approach is utilized to exploit common items sharing efficiently. As a result, it is argue that our proposal matches the fastest Karatsuba-based multipliers and is only one T_X slower than the fastest quadratic multipliers where no divide-and-conquer algorithm is applied.

Furthermore, we note that, based on the similarity between SPB and Montgomery multiplier, the proposed efficient design schemes can be easily move to design bit-parallel $GF(2^m)$ Montgomery multiplier. Finally, the space and time trade-off enable our scheme to apply in acceleration of scalar multiplication under some area constraint platforms. We next work on Mastrovito-Karatsuba multiplier for pentanomials.

TABLE 5
Comparison of Some Bit-Parallel Multipliers for Irreducible Trinomials

Multiplier	# AND	# XOR	Time delay
$x^m + x^k + 1, 1 < k \leq \frac{m-1}{2}$			
Montgomery [27], school-book [26]	m^2	$m^2 - 1$	$T_A + (2 + \lceil \log_2 m \rceil) T_X$
Mastrovito [22] [23] [24]	m^2	$m^2 - 1$	$T_A + (2 + \lceil \log_2 m \rceil) T_X$
Mastrovito [25]	m^2	$m^2 - 1$	$T_A + (\lceil \log_2 (2m + 2k - 3) \rceil) T_X$
SPB Mastrovito [13]	m^2	$m^2 - 1$	$T_A + \lceil \log_2 (2m - k - 1) \rceil T_X$
Montgomery [14]	m^2	$m^2 - 1$	$T_A + \lceil \log_2 (2m - k - 1) \rceil T_X$
Karatsuba [10]	$\frac{3m^2+2m-1}{4}$	$\frac{3m^2}{4} + 4m + k - \frac{23}{4}$ (m odd)	$T_A + (3 + \lceil \log_2 (m - 1) \rceil) T_X$
	$\frac{3m^2}{4}$	$\frac{3m^2}{4} + \frac{5m}{2} + k - 4$ (m even)	
Modified Karatsuba [15]	$\frac{m^2}{2} + (m-k)^2$	$\frac{m^2}{2} + (m-k)^2 + 2k$	$T_A + (2 + \lceil \log_2 (m - 1) \rceil) T_X$
Modified Karatsuba [9]	$m^2 - k^2$	$m^2 + k - k^2 - 1 (1 < k < \frac{m}{3})$	$\leq T_A + (2 + \lceil \log_2 m \rceil) T_X$
		$m^2 + 4k - k^2 - m - 1 (\frac{m}{3} \leq k < \frac{m-1}{2})$	
		$m^2 + 2k - k^2 (k = \frac{m-1}{2})$	
Montgomery squaring [16]	$\frac{3m^2+2m-1}{4}$	$\frac{3m^2}{4} + O(m \log_2 m)$ (m odd)	$\leq T_A + (3 + \lceil \log_2 m \rceil) T_X$
	$\frac{3m^2}{4}$	$\frac{3m^2}{4} + O(m \log_2 m)$ (m even)	$T_A + (2 + \lceil \log_2 m \rceil) T_X$
Chinese Remainder Theorem [29]	Δ	$\Delta + 3k - m$ (Type-A)	$T_A + \lceil \log_2 (\Theta) \rceil T_X$
	Δ	$\Delta + 2k - m + kW(k)$ (Type-B)	$T_A + \lceil \log_2 (3m - 3k - 1) \rceil T_X$
SPB Mastrovito-Karatsuba	$\frac{3m^2+2m-1}{4}$	$\frac{3m^2}{4} + \frac{m}{2} + O(m \log_2 m)$ (m odd)	$T_A + (1 + \lceil \log_2 (2m - k - 1) \rceil) T_X$
	$\frac{3m^2}{4}$	$\frac{3m^2}{4} - \frac{m}{2} + O(m \log_2 m)$ (m even)	
where $\Delta = m^2 + \frac{(m-k)(m-1-3k)}{2}$ ($\frac{m-1}{3} \leq k < \frac{m}{2}$, $2^{v-1} < k \leq 2^v$), $\Theta = \max(3m-3k-1, 2m-2k+2^v)$			
$x^m + x^k + 1, m = 2k$			
Mastrovito [22] [24]	m^2	$m^2 - \frac{m}{2}$	$T_A + (1 + \lceil \log_2 (m - 1) \rceil) T_X$
Montgomery [27], school-book [26]	m^2	$m^2 - \frac{m}{2}$	$T_A + (1 + \lceil \log_2 (m - 1) \rceil) T_X$
Mastrovito [25]	m^2	$m^2 - \frac{m}{2}$	$T_A + (\lceil \log_2 (\frac{3m}{2}) \rceil) T_X$
SPB Mastrovito [13]	m^2	$m^2 - \frac{m}{2}$	$T_A + \lceil \log_2 (\frac{3m}{2}) \rceil T_X$
SPB Karatsuba [21]	$\frac{3m^2}{4}$	$\frac{3m^2}{4} + m + 1$	$T_A + (1 + \lceil \log_2 (m - 1) \rceil) T_X$
Modified Karatsuba [9]	$\frac{3m^2}{4}$	$\frac{3m^2}{4} + m - 1$	$T_A + (1 + \lceil \log_2 (m + 2) \rceil) T_X$
Chinese Remainder Theorem [29]	$\frac{7m^2-2m}{8}$	$\frac{7m^2+2m}{8}$ (Type-A)	$T_A + \lceil \log_2 (2m) \rceil T_X$
	$\frac{7m^2-2m}{8}$	$\frac{7m^2-2m}{8} + kW(k)$ (Type-B)	$T_A + \lceil \log_2 (\frac{3m}{2}) \rceil T_X$
SPB Mastrovito-Karatsuba	$\frac{3m^2}{4}$	$\frac{3m^2}{4} + \frac{m}{2} + 1$	$T_A + (1 + \lceil \log_2 m \rceil) T_X$
$x^m + x^k + 1, m < 2k$			
Mastrovito [22] [24]	m^2	$m^2 - 1$	$T_A + (\lceil \log_2 (2^r m) \rceil) T_X$
Mastrovito [25]	m^2	$m^2 - 1$	$T_A + (\lceil \log_2 (\Theta) \rceil) T_X$
SPB Mastrovito [13]	m^2	$m^2 - 1$	$T_A + \lceil \log_2 (m + k) \rceil T_X$
Montgomery [14]	m^2	$m^2 - 1$	$T_A + \lceil \log_2 (m + k) \rceil T_X$
SPB Mastrovito-Karatsuba	$\frac{3m^2+2m-1}{4}$	$\frac{3m^2}{4} + \frac{m}{2} + O(m \log_2 m)$ (m odd)	$T_A + (1 + \lceil \log_2 (m + k) \rceil) T_X$
	$\frac{3m^2}{4}$	$\frac{3m^2}{4} - \frac{m}{2} + O(m \log_2 m)$ (m even)	
where $r = \lceil \frac{m-1}{m-k} \rceil$, $\Theta = 2^r \big(m-1-(r-2)(m-k)\big) - 2(m-k) + 1$			

APPENDIX A
PROOF OF OBSERVATION 3.1

Proof According to the definitions of $\mathbf{A}_U, \mathbf{A}_M$ and $\mathbf{A}_L$, we have

$$\mathbf{A}_U[\downarrow (m-k)] = [\underbrace{\mathbf{0}, \cdots, \mathbf{0}}_{m-k}, \mathbf{A'}(1,:)^T, \cdots, \mathbf{A'}(k,:)^T]^T,$$

$$\mathbf{A}_M = [\mathbf{A'}(k+1,:)^T, \cdots, \mathbf{A'}(k+m,:)^T]^T,$$

$$\mathbf{A}_L[\uparrow (k+1)] = [\mathbf{A'}(k+m+1,:)^T, \cdots, \mathbf{A'}(2m-1,:)^T, \underbrace{\mathbf{0}, \cdots, \mathbf{0}}_{k+1}]^T.$$

Then

$$\mathbf{A}_U[\downarrow (m-k)] + \mathbf{A}_L[\uparrow (k+1)] = [\mathbf{A'}(k+m+1,:)^T, \cdots, \mathbf{A'}(2m-1,:)^T \\ \mathbf{0}, \mathbf{A'}(1,:)^T, \cdots, \mathbf{A'}(k,:)^T]^T.$$

Thus $\mathbf{A}_M + \mathbf{A}_U[\downarrow (m-k)] + \mathbf{A}_L[\uparrow (k+1)]$ is given by:

$$[\mathbf{A'}(k+m+1,:)^T + \mathbf{A'}(k+1,:)^T, \cdots, \mathbf{A'}(2m-1,:)^T + \mathbf{A'}(m-1,:)^T, \\ \mathbf{A'}(m,:)^T, \mathbf{A'}(1,:)^T + \mathbf{A'}(m+1,:)^T \cdots, \mathbf{A'}(k,:)^T + \mathbf{A'}(k+m,:)^T].$$

In addition, this matrix can also be rewritten as $\mathbf{E}[\circlearrowleft k]$, where

$$\mathbf{E} = [\mathbf{A'}(1,:)^T + \mathbf{A'}(m+1,:)^T \cdots, \mathbf{A'}(k,:)^T + \mathbf{A'}(k+m,:)^T, \\ \mathbf{A'}(k+1,:)^T + \mathbf{A'}(k+m+1,:)^T, \cdots, \mathbf{A'}(m-1,:)^T + \mathbf{A'}(2m-1,:)^T, \\ \mathbf{A'}(m,:)^T].$$

It is clear that **E** is equal to the upper m rows of $\mathbf{A'}$ plus its lower $m-1$ rows. We can obtain the formulae (12) and (13) immediately. □

APPENDIX B

Table 6-10 give the overlapped values for $S_1 x^{-2k} \bmod f(x)$ of other cases.

ACKNOWLEDGMENTS

The first author is supported by the National Natural Science Foundation of China (Grant no. 61402393).

REFERENCES

[1] A. Karatsuba and Yu. Ofman. Multiplication of Multidigit Numbers on Automata. Soviet Physics-Doklady (English translation), vol. 7, no. 7, pp. 595-596, 1963.

[2] Joachim Von Zur Gathen and Jurgen Gerhard. 2003. Modern Computer Algebra (2 ed.). Cambridge University Press, New York, NY, USA.

[3] H. Fan, J. Sun, M. Gu, and K.-Y. Lam. Overlap-free Karatsuba-Ofman polynomial multiplication algorithms. *Information Security, IET*, vol.4, no.1, pp.8-14, March 2010.

[4] A. Weimerskirch, and C. Paar, Generalizations of the Karatsuba Algorithm for Efficient Implementations. *Cryptology ePrint Archive, Report 2006/224*, http://eprint.iacr.org/

[5] E.D. Mastrovito. VLSI Architectures for Computation in Galois Fields. PhD thesis, Linköping University, Department of Electrical Engineering, Linköping, Sweden, 1991.

[6] Yiyang Chen. On Space-Time Trade-Off for Montgomery Multipliers over Finite Fields. MD thesis, Department of Computer Science and operational Research, Montreal University, Montreal, Canada. 2015. https://papyrus.bib.umontreal.ca/xmlui/bitstream/handle/1866/12571/Chen_Yiyang_2015_memoire.pdf

[7] K.K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. John Wiley & Sons, 1999.

[8] Ku-Young Chang, Dowon Hong, and Hyun-Sook Cho. Low complexity bit-parallel multiplier for $GF(2^m)$ defined by all-one polynomials using redundant representation. *IEEE Trans. Comput.*, 54(12):1628–1630, 2005.

[9] Young In Cho, Nam Su Chang, Chang Han Kim, Young-Ho Park, and Seokhie Hong. New bit parallel multiplier with low space complexity for all irreducible trinomials over $gf(2^n)$. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 20(10):1903–1908, Oct 2012.

[10] M. Elia, M. Leone, and C. Visentin. Low complexity bit-parallel multipliers for $GF(2^m)$ with generator polynomial $x^m + x^k + 1$. *Electronic Letters*, 35(7):551–552, 1999.

[11] M. Leone. A New Low Complexity Parallel Multiplier for a Class of Finite Fields. *Proc. Cryptographic Hardware and Embedded Systems (CHES 2001)*, LNCS 2162, pp. 160-170, 2001.

[12] Haining Fan and Yiqi Dai. Fast bit-parallel $gf(2^n)$ multiplier for all trinomials. *IEEE Trans. Comput.*, 54(4):485–490, 2005.

[13] Haining Fan and M.A. Hasan. Fast bit parallel-shifted polynomial basis multipliers in gf(2n). *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 53(12):2606–2615, Dec 2006.

[14] A. Hariri and A. Reyhani-Masoleh, Bit-serial and bit-parallel montgomery multiplication and squaring over $GF(2^m)$. *IEEE Transactions on Computers*, 58(10):1332–1345, 2009.

[15] Yin Li, Gong liang Chen, and Jian hua Li. Speedup of bit-parallel karatsuba multiplier in $GF(2^m)$ generated by trinomials. *Information Processing Letters*, 111(8):390 – 394, 2011.

[16] Yin Li, Yiyang Chen. New bit-parallel Montgomery multiplier for trinomials using squaring operation. *Integration, the VLSI Journal*, Volume 52, 142–155, January 2016.

[17] Rudolf Lidl and Harald Niederreiter. *Introduction to finite fields and their applications*. Cambridge University Press, New York, NY, USA, 1994.

[18] Rudolf Lidl and Harald Niederreiter. *Finite Fields*. Cambridge University Press, New York, NY, USA, 1996.

[19] Christophe Negre. Efficient parallel multiplier in shifted polynomial basis. *J. Syst. Archit.*, 53(2-3):109–116, 2007.

[20] Francisco Rodríguez-Henríquez and Çetin Kaya Koç. Parallel multipliers based on special irreducible pentanomials. *IEEE Trans. Comput.*, 52(12):1535–1542, 2003.

[21] Haibin Shen and Yier Jin. Low complexity bit parallel multiplier for $GF(2^m)$ generated by equally-spaced trinomials. *Inf. Process. Lett.*, 107(6):211–215, 2008.

[22] B. Sunar and Ç.K. Koç, Mastrovito multiplier for all trinomials, IEEE Transactions on Computers, 48(5) (1999) 522–527.

[23] A. Halbutogullari and Ç.K. Koç, Mastrovito multiplier for general irreducible polynomials, IEEE Transactions on Computers, 49(5) (May 2000) 503–518.

[24] T. Zhang and K.K. Parhi, Systematic design of original and modified mastrovito multipliers for general irreducible polynomials, IEEE Transactions on Computers, 50(7) (July 2001) 734–749.

[25] N. Petra, D. De Caro, and A.G.M. Strollo, "A novel architecture for galois fields $GF(2^m)$ multipliers based on mastrovito scheme," *IEEE Trans. Computers*, vol. 56, no. 11, pp. 1470–1483, November 2007.

[26] Huapeng Wu. Bit-parallel finite field multiplier and squarer using polynomial basis. *IEEE Trans. Comput.*, 51(7):750–758, 2002.

[27] Huapeng Wu. Montgomery multiplier and squarer for a class of finite fields. *IEEE Trans. Comput.*, 51(5):521–529, 2002.

[28] H. Fan and M.A. Hasan, A survey of some recent bit-parallel multipliers, Finite Fields and Their Applications, 32 (2015) 5–43.

[29] H. Fan, A Chinese Remainder Theorem Approach to Bit-Parallel $GF(2^n)$ Polynomial Basis Multipliers for Irreducible Trinomials, IEEE Trans. Comput. 65(2): 343–352, February 2016.

[30] Recommended Elliptic Curves for Federal Government Use, http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf, July, 1999.

TABLE 6
The overlapped values, m even, $n < k < m$

first part	Overlapped	#XOR	second part	Overlapped	#XOR
$\mathbf{M}_{A,1}(1:)\cdot\mathbf{b}$	P_{k-n}	$W(m)-1$	$\mathbf{M}_{A,2}(1:)\cdot\mathbf{b}$	P_0	$W(1)$
$\mathbf{M}_{A,1}(2:)\cdot\mathbf{b}$	P_{k-n+1}	$W(m)-1$	$\mathbf{M}_{A,2}(2:)\cdot\mathbf{b}$	P_1	$W(2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{M}_{A,1}(m-k:)\cdot\mathbf{b}$	P_{n-1}	$W(m)-1$	$\mathbf{M}_{A,2}(n:)\cdot\mathbf{b}$	P_{n-1}	$W(n)$
$\mathbf{M}_{A,1}(m-k+1:)\cdot\mathbf{b}$	P_0	$W(m)-1$	$\mathbf{M}_{A,2}(n+1:)\cdot\mathbf{b}$	P_0	$W(n+1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{M}_{A,1}(m+n-k:)\cdot\mathbf{b}$	P_{n-1}	$W(m)-1$	$\mathbf{M}_{A,2}(k:)\cdot\mathbf{b}$	P_{k-n-1}	$W(k)$
$\mathbf{M}_{A,1}(m+n-k+1:)\cdot\mathbf{b}$	P_0	$W(m)-1$	$\mathbf{M}_{A,2}(k+1:)\cdot\mathbf{b}$	P_{k-n+1}	$W(m-k-1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{M}_{A,1}(m:)\cdot\mathbf{b}$	P_{k-n-1}	$W(m)-1$	$\mathbf{M}_{A,2}(m-1:)\cdot\mathbf{b}$	P_{n-1}	$W(1)$

TABLE 7
The overlapped values, m odd, $0 < k \leq n-1$

first part	Overlapped	#XOR	second part	Overlapped	#XOR
$\mathbf{M}_{A,1}(1:)\cdot\mathbf{b}$	P_k, P_{k+1}	$W(n+1)+W(n)-1$	$\mathbf{M}_{A,2}(1:)\cdot\mathbf{b}$	P_0	$W(1)$
$\mathbf{M}_{A,1}(2:)\cdot\mathbf{b}$	P_{k+1}, P_{k+2}	$W(n+1)+W(n)-1$	$\mathbf{M}_{A,2}(2:)\cdot\mathbf{b}$	P_1	$W(2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{M}_{A,1}(n-k:)\cdot\mathbf{b}$	P_{n-1}, P_n	$W(n+1)+W(n)-1$	$\mathbf{M}_{A,2}(k:)\cdot\mathbf{b}$	P_{k-1}	$W(k)$
$\mathbf{M}_{A,1}(n-k+1:)\cdot\mathbf{b}$	P_0	$W(m)-1$	$\mathbf{M}_{A,2}(k+1:)\cdot\mathbf{b}$	P_k, P_{k+1}	$W(n-k)+W(n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{M}_{A,1}(m-k:)\cdot\mathbf{b}$	P_{n-1}	$W(m)-1$	$\mathbf{M}_{A,2}(n:)\cdot\mathbf{b}$	P_{n-1}, P_n	$W(1)+W(n)$
$\mathbf{M}_{A,1}(m-k+1:)\cdot\mathbf{b}$	P_0, P_1	$W(n+1)+W(n)-1$	$\mathbf{M}_{A,2}(n+1)\cdot\mathbf{b}$	P_0	$W(n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{M}_{A,1}(m:)\cdot\mathbf{b}$	P_{k-1}, P_k	$W(n+1)+W(n)-1$	$\mathbf{M}_{A,2}(m-1:)\cdot\mathbf{b}$	P_n	$W(1)$

TABLE 8
The overlapped values, m odd, $n < k \leq m-1$

first part	Overlapped	#XOR	second part	Overlapped	#XOR
$\mathbf{M}_{A,1}(1:)\cdot\mathbf{b}$	P_{k-n}	$W(m)-1$	$\mathbf{M}_{A,2}(1:)\cdot\mathbf{b}$	P_0	$W(1)$
$\mathbf{M}_{A,1}(2:)\cdot\mathbf{b}$	P_{k-n+1}	$W(m)-1$	$\mathbf{M}_{A,2}(2:)\cdot\mathbf{b}$	P_1	$W(2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{M}_{A,1}(m-k:)\cdot\mathbf{b}$	P_n	$W(m)-1$	$\mathbf{M}_{A,2}(n:)\cdot\mathbf{b}$	P_{n-1}	$W(n)$
$\mathbf{M}_{A,1}(m-k+1:)\cdot\mathbf{b}$	P_0, P_1	$W(n+1)+W(n)-1$	$\mathbf{M}_{A,2}(n+1:)\cdot\mathbf{b}$	P_0	$W(n)+W(1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{M}_{A,1}(m+n-k:)\cdot\mathbf{b}$	P_{n-1}, P_n	$W(n+1)+W(n)-1$	$\mathbf{M}_{A,2}(k:)\cdot\mathbf{b}$	P_{k-n-1}	$W(n)+W(k-n)$
$\mathbf{M}_{A,1}(m+n-k+1:)\cdot\mathbf{b}$	P_0	$W(m)-1$	$\mathbf{M}_{A,2}(k+1)\cdot\mathbf{b}$	P_{k-n+1}	$W(m-k-1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{M}_{A,1}(m:)\cdot\mathbf{b}$	P_{k-n-1}	$W(m)-1$	$\mathbf{M}_{A,2}(m-1:)\cdot\mathbf{b}$	P_n	$W(1)$

TABLE 9
The overlapped values, m odd, $k = n$

first part	Overlapped	#XOR	second part	Overlapped	#XOR
$\mathbf{M}_{A,1}(1:)\cdot\mathbf{b}$	P_0	$W(m)-1$	$\mathbf{M}_{A,2}(1:)\cdot\mathbf{b}$	P_0	$W(1)$
$\mathbf{M}_{A,1}(2:)\cdot\mathbf{b}$	P_1	$W(m)-1$	$\mathbf{M}_{A,2}(2:)\cdot\mathbf{b}$	P_1	$W(2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{M}_{A,1}(n+1:)\cdot\mathbf{b}$	P_n	$W(m)-1$	$\mathbf{M}_{A,2}(n:)\cdot\mathbf{b}$	P_{n-1}	$W(n)$
$\mathbf{M}_{A,1}(n+2:)\cdot\mathbf{b}$	P_0, P_1	$W(n+1)+W(n)-1$	$\mathbf{M}_{A,2}(n+1:)\cdot\mathbf{b}$	P_0	$W(n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{M}_{A,1}(m-1:)\cdot\mathbf{b}$	P_{n-2}, P_{n-1}	$W(n+1)+W(n)-1$	$\mathbf{M}_{A,2}(m-2:)\cdot\mathbf{b}$	P_{n-2}	$W(2)$
$\mathbf{M}_{A,1}(m:)\cdot\mathbf{b}$	P_{n-1}, P_n	$W(n+1)+W(n)-1$	$\mathbf{M}_{A,2}(m-1:)\cdot\mathbf{b}$	P_{n-1}	$W(1)$