Systematic Design of Original and Modified Mastrovito Multipliers for General Irreducible Polynomials

Tong Zhang, *Student Member*, *IEEE*, and Keshab K. Parhi, *Fellow*, *IEEE*

Abstract—This paper considers the design of bit-parallel dedicated finite field multipliers using standard basis. An explicit algorithm is proposed for efficient construction of Mastrovito product matrix, based on which we present a systematic design of Mastrovito multiplier applicable to $GF(2^m)$ generated by an arbitrary irreducible polynomial. This design effectively exploits the spatial correlation of elements in Mastrovito product matrix to reduce the complexity. Using a similar methodology, we propose a systematic design of modified Mastrovito multiplier, which is suitable for $GF(2^m)$ generated by high-Hamming weight irreducible polynomials. For both original and modified Mastrovito multipliers, the developed multiplier architectures are highly modular, which is desirable for VLSI hardware implementation. Applying the proposed algorithm and design approach, we study the Mastrovito multipliers for several special irreducible polynomials, such as trinomial and equally-spaced-polynomial, and the obtained complexity results match the best known results. Moreover, we have discovered several new special irreducible polynomials which also lead to low-complexity Mastrovito multipliers.

Index Terms—Finite (or Galois) field, standard basis, multiplication, irreducible polynomials, complexity, VLSI architecture, Toeplitz matrix.

1 Introduction

Efficient hardware implementations of finite (or Galois) field $GF(2^m)$ arithmetic units are highly desirable for many applications in error-correcting coding and cryptography [1], [2], [3], [4]. Among the $GF(2^m)$ arithmetic operations, multiplication is the most basic and important building block in many applications. A number of efficient $GF(2^m)$ multiplication approaches and architectures have been proposed in which different basis representations of field elements are used, such as *standard basis*, *dual basis*, and *normal basis*. Standard basis is more promising in the sense that it gives designers more freedom on irreducible polynomial selection and hardware optimization. For detailed discussions of these three basis representations, readers are referred to [3], [5]. In this paper, we are interested in the design of bit-parallel $GF(2^m)$ multipliers using standard basis.

The standard basis multiplication involves two steps: *polynomial multiplication* and *modulo reduction*. Let $f(x)$ be the irreducible polynomial generating $GF(2^m)$, $c(x)$ be the product of $a(x)$ and $b(x)$, where $a(x)$, $b(x)$, $c(x) \in GF(2^m)$. The finite field multiplication is performed as $c(x) = (a(x) \cdot b(x)) \bmod f(x)$. An efficient dedicated bit-parallel multiplier was proposed by Mastrovito [6] in which a *product matrix* $\mathbf{M}$ is introduced to combine the above two steps together. Thus, the multiplication is carried out by $\mathbf{c} = \mathbf{M} \cdot \mathbf{b}$, where $\mathbf{c}$ and $\mathbf{b}$ represent the coefficient vectors of $c(x)$ and $b(x)$, respectively. Given irreducible polynomial $f(x)$, each entry in matrix $\mathbf{M}$ is obtained by Xoring certain coefficients of $a(x)$. Many entries in matrix $\mathbf{M}$ can be computed efficiently by sharing some common items, e.g., two entries $\mathbf{M}(i_1, j_1) = a_0 + a_2 + a_3$ and $\mathbf{M}(i_2, j_2) = a_0 + a_2 + a_4$ can be computed using three XOR gates by sharing the common item $a_0 + a_2$. This method is called *subexpression sharing* [7]. Mastrovito multipliers using two special irreducible polynomials, *trinomial* and *equally-spaced-polynomial* (ESP), have been studied by many researchers for their low-complexity implementations [8], [9], [10], [11], [12], [13], [14]. The essence of all these works is to find an architecture to exploit subexpression sharing efficiently based on the specific irreducible polynomials. It has been shown in [8] that Mastrovito multiplier using irreducible trinomial $x^m + x^n + 1$ only requires $(m^2 - 1)$ XOR gates and m^2 AND gates. By generalizing the approach of [8], Halbutogullari and Koç [14] discovered that the space complexity of Mastrovito multiplier using irreducible ESP $x^m + x^{tr} + \cdots + x^r + 1$, where $(t + 1)r = m$, can be reduced to $(m^2 - r)$ XOR gates and m^2 AND gates. Furthermore, [14] presents a new formulation of the Mastrovito product matrix for an arbitrary irreducible polynomial $x^m + x^{n_k} + \cdots + x^{n_1} + 1$, where the space complexity is given as m^2 AND gates and $(m-1)(m+k-1) + \sum_{j \in \mathcal{N}}(m - 1 - j)$ XOR gates, $\mathcal{N} \subset \{0, 1, \cdots, m-2\}$. However, [14] fails to find a method to explicitly compute the set $\mathcal{N}$, which makes its result less practicable in general.

In general cases, the complexity of computing product matrix is proportional to the Hamming weight of the irreducible polynomial $f(x)$ (denoted as *pwt*). So, the Mastrovito multiplier is good only when low-Hamming weight irreducible polynomials are used. A modified

• *The authors are with the Department of Electrical and Computer Engineering, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455. E-mail: {tzhang, parhi}@ece.umn.edu.*

Manuscript received 27 Oct. 2000; revised 7 Feb. 2001; accepted 3 Apr. 2001. For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 113073.

Mastrovito multiplier was proposed by Song and Parhi [15], which has a complexity proportional to $(m-1-pwt)$. Its basic idea is to use the complementary irreducible polynomial for the computation of matrix $\mathbf{M}$ with appropriate compensation. Such a multiplication scheme is efficient when $GF(2^m)$ is generated by high-Hamming weight irreducible polynomial.

Generally, for large m, efficient design for both original and modified Mastrovito multipliers becomes rather difficult and a systematic design approach is highly desirable. In this paper, we generalize the approach of [8] in a different way compared with [14]. We propose a theorem and an algorithm (which can explicitly compute set $\mathcal{N}$ in [14]) for the construction of product matrix in the original Mastrovito multiplier, based on which we develop an efficient systematic design of the original Mastrovito multiplier. This design effectively exploits subexpression sharing in the computation of product matrix to reduce the complexity. Using a similar methodology, we also develop a systematic design of modified Mastrovito multiplier. For both original and modified Mastrovito multipliers, explicit algorithms and architectures are presented and the complexities are given in detail. Applying our proposed design approach, we study the irreducible trinomial and ESP and the complexity results match the results in [8] and [14]. Meanwhile, another computation approach for trinomial case is proposed to make a trade-off between space complexity and delay. Moreover, with the aid of the proposed algorithm, we discover several new irreducible polynomials leading to low-complexity original Mastrovito multipliers, which is especially desirable when neither an irreducible trinomial nor an irreducible ESP exists.

This paper is organized as follows: We introduce the notation of this paper and the fundamentals of finite field and Mastrovito multiplier in Section 2. In Section 3, we propose a theorem and an algorithm for the construction of product matrix, based on which a systematic design approach for original Mastrovito multiplier is developed. In Section 4, using a similar design methodology, we present a systematic design approach for modified Mastrovito multiplier. Efficient computation approaches for several special irreducible polynomials are discussed in Section 5. This paper is an extended version of [16].

2 Notation and Preliminaries

Since several arithmetic operations pertaining to matrices and vectors will be extensively used throughout this paper, we first introduce the following notation: Column vectors and matrices are represented by small and capital boldfaced characters, respectively. Matlab matrix notations are used, e.g., $\mathbf{Z}(i,:)$, $\mathbf{Z}(:,j)$, 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. The operations of shift by feeding zero are represented by corresponding arrows, e.g., $\mathbf{v}[\downarrow 2]$, $\mathbf{U}[\rightarrow 1]$, and $\mathbf{U}[\downarrow 1]$ represent down shift of vector $\mathbf{v}$ by two positions, right shift of matrix $\mathbf{U}$ by one column, and down shift of matrix $\mathbf{U}$ by one row, respectively, which are explicitly given as:

$$\begin{aligned}\mathbf{v}[\downarrow 2] &= [0,0,v_0,\cdots,v_{n-2}]^T\\ \mathbf{U}[\rightarrow 1] &= [\mathbf{o},\mathbf{U}(:,1),\cdots,\mathbf{U}(:,m-1)]\\ \mathbf{U}[\downarrow 1] &= [\mathbf{o},\mathbf{U}(1,:)^T,\cdots,\mathbf{U}(m-1,:)^T]^T,\end{aligned}$$

where $\mathbf{o}$ represents the zero column vector. Furthermore, we note that the AND and XOR gates considered in this paper are all 2-input gates, whose delays are denoted as T_A and T_X, respectively.

Finite field $GF(2^m)$ contains 2^m elements and can be viewed as an m-dimensional vector space over $GF(2)$, which only has two elements, 0 and 1. With the standard basis $\{1,x,x^2,\cdots,x^{m-1}\}$, the elements of the finite field $GF(2^m)$ can be represented as polynomials of degree $m-1$ as follows:

$$\begin{aligned}GF(2^m) &= \{a(x)|a(x)\\ &= a_{m-1}x^{m-1}+a_{m-2}x^{m-2}+\cdots+a_1x+a_0,\ a_i\in GF(2)\}.\end{aligned}$$

Such polynomial representation is generally used for finite field arithmetic operations, where addition is carried out by polynomial addition over $GF(2^m)$ using bit-independent XOR operations.

Finite field multiplication using standard basis is carried out by polynomial multiplication and modulo operation. Let $f(x)=x^m+f_{m-1}x^{m-1}+\cdots+f_1x+1$ be the irreducible polynomial generating $GF(2^m)$, and $c(x)$ be the product of $a(x)$ and $b(x)$, where $a(x),b(x),c(x)\in GF(2^m)$. The polynomial multiplication, $d(x)=a(x)\cdot b(x)$, can performed as

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

where $\mathbf{b}=[b_0,b_1,\cdots,b_{m-1}]^T$ and $\mathbf{d}=[d_0,d_1,\cdots,d_{2m-2}]^T$ are the coefficient vectors of $b(x)$ and $d(x)$, respectively, and matrix $\mathbf{A}$ is given as

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

Next, we perform the modulo reduction

$$c(x)=d(x)\bmod f(x),$$

which can be expressed as

$$\begin{aligned}c(x) &= \sum_{k=0}^{2m-2} d_kx^k \bmod f(x)\\ &= \sum_{k=0}^{m-1} d_kx^k+\sum_{k=m}^{2m-2} d_k\Big(x^k \bmod f(x)\Big).\end{aligned} \tag{2}$$

For $m\le k\le 2m-2$, if we denote $x^k \bmod f(x)$ as $u^{(l)}(x)$, where $l=k-m+1$, we have

$$u^{(l)}(x)=\begin{cases} x^m\ mod\ f(x), & l=1,\\ u^{(l-1)}(x)\cdot x \bmod f(x), & l=2,3,\cdots,m-1.\end{cases}$$

Since the irreducible polynomial is

$$f(x) = x^m + f_{m-1}x^{m-1} + \cdots + f_1 x + f_0,$$

where $f_0 = 1$, we get

$$u_j^{(1)} = f_j, \quad 0 \le j \le m-1,$$
$$u_j^{(l)} = \begin{cases} u_{j-1}^{(l-1)} + u_{m-1}^{(l-1)} f_j, & 1 \le j \le m-1 \\ u_{m-1}^{(l-1)}, & j = 0, \end{cases} \tag{3}$$

where $u_j^{(l-1)}$ and $u_j^{(l)}$ are the coefficients of $u^{(l-1)}(x)$ and $u^{(l)}(x)$, respectively. Let $\mathbf{u}^{(l)}$ denote the coefficient vector of $u^{(l)}(x)$, then, using vector notation, (3) can be rewritten as

$$\mathbf{u}^{(l)} = \begin{cases} \mathbf{f}, & l = 1 \\ \mathbf{u}^{(l-1)}[\downarrow 1] + u_{m-1}^{(l-1)} \cdot \mathbf{f}, & 2 \le l \le m-1, \end{cases} \tag{4}$$

where $\mathbf{f} = [1, f_1, \cdots, f_{m-1}]^T$. Let $\mathbf{s}$ denote vector

$$[d_0, d_1, \cdots, d_{m-1}]^T,$$

which consists of the first m entries of $\mathbf{d}$, we can rewrite (2) in matrix notation as follows:

$$\mathbf{c} = \mathbf{I}_{m\times m} \cdot \mathbf{s} + \sum_{l=1}^{m-1} d_{l+m-1} \cdot \mathbf{u}^{(l)}$$
$$= \left[\mathbf{I}_{m\times m}, \mathbf{U}\right] \cdot \mathbf{d} = \left[\mathbf{I}_{m\times m}, \mathbf{U}\right] \cdot \mathbf{A} \cdot \mathbf{b},$$

where $\mathbf{I}_{m\times m}$ represent $m \times m$ identity matrix and matrix $\mathbf{U} = [\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \cdots, \mathbf{u}^{(m-1)}]$. Note that all successive columns in matrix $\mathbf{U}$ have the recursive relation as in (4). Define

$$\mathbf{M} = \left[\mathbf{I}_{m\times m}, \mathbf{U}\right] \cdot \mathbf{A}. \tag{5}$$

Thus, the $GF(2^m)$ multiplication can be carried out by $\mathbf{c} = \mathbf{M} \cdot \mathbf{b}$, which is the well-known Mastrovito multiplication scheme, and the matrix $\mathbf{M}$ is called product matrix.

3 Mastrovito Multiplier

In this section, we first introduce a theorem and an algorithm pertaining to the construction of product matrix $\mathbf{M}$. Then, we develop an approach to compute $\mathbf{M}$ by adding a series of Toeplitz matrices, through which subexpression sharing could be extensively exploited to reduce the XOR complexity. Accordingly, a highly modular multiplier architecture is presented.

3.1 Proposed Theorem and Algorithm

In Section 2, we have shown that product matrix $\mathbf{M}$ can be expressed as the product of two independent matrices, $[\mathbf{I}_{m\times m}, \mathbf{U}]$ and $\mathbf{A}$, and there exists a recursive relation between successive columns in matrix $\mathbf{U}$, as described in (4). Through the following theorem and algorithm, we will see that matrix $\mathbf{U}$ also can be constructed by adding a series of Toeplitz matrices.

Theorem 3.1. *For the computation of matrix U in (5), we can always construct a set $\mathcal{N} \subset \{0, 1, \cdots, m-2\}$ and*

$$\mathbf{U} = \sum_{n\in\mathcal{N}} \mathbf{F}[\rightarrow n], \tag{6}$$

where the Toeplitz matrix $\mathbf{F} = [\mathbf{f}, \mathbf{f}[\downarrow 1], \cdots, \mathbf{f}[\downarrow m-2]]$.

Let the irreducible polynomial be

$$f(x) = x^m + x^{k_s} + \cdots + x^{k_1} + 1,$$

where $m > k_s > \cdots > k_1 > 1$, the set $\mathcal{N}$ in Theorem 3.1 can be explicitly constructed with the aid of a *weighted tree* generated based on $f(x)$. The complete construction approach is described by the following algorithm:

Algorithm 3.1.

Input: The parameters of irreducible polynomial: m, $k_1, \cdots, k_s$;

Output: set $\mathcal{N} \subset \{0, 1, \cdots, m-2\}$.

Procedure:

1. Generate a weighted tree D according to the following properties:
   - Each node d_j in D has at most s child nodes and each edge has the weight $w \in \{(m-k_i), 1 \le i \le s\}$;
   - Let d_1 denote the root and $h(d_1, d_j)$ denote the weight of path from d_1 to d_j, where $h(d_1, d_1) = 0$, we have $\forall\, d_j$, if $\exists\, r \in \{(m-k_i), 1 \le i \le s\}$ and $(h(d_1, d_j) + r) < m-1$, then d_j always has one child node d_l and the weight of edge between d_j and d_l is r;
   - $\forall d_j \in D$, $h(d_1, d_j) < m-1$.
2. Construct multiset $\mathcal{H} = \{h(d_1, d_j),\ \forall d_j \in D\}$ and set $\mathcal{N} = \emptyset$;
3. For $0 \le j \le m-2$, do
   a. create multiset $\mathcal{S}_j = \emptyset$;
   b. $\forall\, h \in \mathcal{H}$, if $h = j$, then insert h into $\mathcal{S}_j$;
   c. if $(|\mathcal{S}_j| \ mod\ 2) = 1$, then insert j into the set $\mathcal{N}$.

Here, we note that a *multiset* is like a *set*, except that repeated elements are allowed, and $|\mathcal{S}_j|$ represents the order of $\mathcal{S}_j$. A proof of Theorem 3.1 is given in Appendix A, in which Algorithm 3.1 is developed. From the above algorithm, we know that the least two elements in $\mathcal{N}$ are always 0 and $(m-k_s)$ and we have $|\mathcal{N}| \le k_s$.

Example 3.1. Consider the multiplication of $a(x) = x^4 + x^2$ and $b(x) = x^3 + x^2 + 1$ over $GF(2^5)$ with the underlying irreducible polynomial $f(x) = x^5 + x^4 + x^3 + x^2 + 1$. We have $\{(m-k_i), 1 \le i \le s\} = \{1, 2, 3\}$. Applying Algorithm 3.1, we generate the tree D. as shown in Fig. 1, and get the multiset $\mathcal{H} = \{0, 1, 2, 2, 3, 3, 3, 3\}$. Thus, we have

$$\mathcal{S}_0 = \{0\} \Rightarrow |\mathcal{S}_0| = 1; \quad \mathcal{S}_1 = \{1\} \Rightarrow |\mathcal{S}_1| = 1;$$
$$\mathcal{S}_2 = \{2, 2\} \Rightarrow |\mathcal{S}_2| = 2; \quad \mathcal{S}_3 = \{3, 3, 3, 3\} \Rightarrow |\mathcal{S}_3| = 4.$$

Since $|\mathcal{S}_2| \bmod 2 = |\mathcal{S}_3| \bmod 2 = 0$, we get $\mathcal{N} = \{0, 1\}$. Therefore, we perform the multiplication as

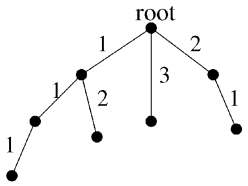

Fig. 1. Tree structure.

$$\mathbf{c} = \mathbf{M} \cdot \mathbf{b} = \Big(\Big[\mathbf{I}_{5\times 5}, \sum_{n\in\mathcal{N}} \mathbf{F}[\rightarrow n] \Big] \cdot \mathbf{A} \Big) \cdot \mathbf{b} =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

where

$$F = \Big[\mathbf{f}, \mathbf{f}[\downarrow 1], \cdots, \mathbf{f}[\downarrow m-2] \Big] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}.$$

Thus, we have $c(x) = (a(x)b(x)) \mod f(x) = x^4$.

3.2 Multiplier Architecture

Applying Theorem 3.1 and Algorithm 3.1, in this section, we develop an efficient computation approach for the product matrix $\mathbf{M}$ and present the corresponding low-complexity Mastrovito multiplier architecture. In the following, we frequently use two special matrices: the *Toeplitz* matrix and the *upper-triangular Toeplitz* matrix. It is well-known that the sum of two upper-triangular Toeplitz matrices (or one Toeplitz matrix and one upper-triangular Toeplitz matrix) can be obtained by only computing the sum of two row vectors. This property will be used to exploit the subexpression sharing in the construction of product matrix $\mathbf{M}$ to reduce the entire XOR complexity.

Given a general irreducible polynomial $f(x) = x^m + x^{k_s} + \cdots + x^{k_1} + 1$, we express its coefficient vector as $\mathbf{f} = \mathbf{e}_1 + \mathbf{e}_{k_1+1} + \cdots + \mathbf{e}_{k_s+1}$, where $\mathbf{e}_i$ is the m-dimensional ith *canonical unit* vector:

$$\mathbf{e}_i = [\underbrace{0, \cdots, 0}_{i-1}, 1, \underbrace{0, \cdots, 0}_{m-i}]^T.$$

Define

$$\mathbf{E}_j = \Big[\mathbf{e}_j, \mathbf{e}_j[\downarrow 1], \cdots, \mathbf{e}_j[\downarrow m-2] \Big]. \tag{7}$$

We can write the Toeplitz matrix $\mathbf{F}$ in Theorem 3.1 as $\sum_{i=0}^{s} \mathbf{E}_{k_i+1}$, where $k_0 = 0$, and have

$$\mathbf{U} = \sum_{n\in\mathcal{N}} \mathbf{F}[\rightarrow n] = \sum_{i=0}^{s} \sum_{n\in\mathcal{N}} \mathbf{E}_{k_i+1}[\rightarrow n]. \tag{8}$$

Moreover, according to (1), we write $\mathbf{A}$ in block matrix form as

$$\mathbf{A} = [\mathbf{A}_s^T, \mathbf{A}_t^T]^T, \tag{9}$$

where $\mathbf{A}_s$ is an $m \times m$ *lower-triangular* Toeplitz matrix and $\mathbf{A}_t$ is an $(m-1) \times m$ *upper-triangular* Toeplitz matrix:

$$\begin{aligned} \mathbf{A}_s &= \begin{bmatrix} a_0 & 0 & \cdots & 0 \\ a_1 & a_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1} & a_{m-2} & \cdots & a_0 \end{bmatrix}, \\ \mathbf{A}_t &= \begin{bmatrix} 0 & a_{m-1} & \cdots & a_2 & a_1 \\ 0 & 0 & \cdots & a_3 & a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{m-1} \end{bmatrix}. \end{aligned} \tag{10}$$

Substituting (6), (8), and (9) into (5), we get

$$\mathbf{M} = \mathbf{A}_s + \sum_{i=0}^{s} \sum_{n\in\mathcal{N}} \Big(\mathbf{E}_{k_i+1}[\rightarrow n] \cdot \mathbf{A}_t \Big). \tag{11}$$

Based on the special structure of $\mathbf{E}_j$ as defined in (7), it can be easily proven that

$$\mathbf{E}_j[\rightarrow n] \cdot \mathbf{A}_t = \Big(\tilde{\mathbf{A}}_t[\rightarrow n] \Big)[\downarrow (j-1)], \tag{12}$$

where $\tilde{\mathbf{A}}_t = [\mathbf{A}_t^T, \mathbf{o}]^T$ and $\mathbf{o}$ is an m-dimensional zero column vector. Thus, if we denote $\sum_{n\in\mathcal{N}} (\tilde{\mathbf{A}}_t[\rightarrow n])$ as $\mathbf{S}$, (11) can be rewritten as

$$\begin{aligned} \mathbf{M} &= \mathbf{A}_s + \sum_{i=0}^{s} \Big(\sum_{n\in\mathcal{N}} \tilde{\mathbf{A}}_t[\rightarrow n] \Big)[\downarrow k_i] \\ &= \mathbf{A}_s + \mathbf{S} + \sum_{i=1}^{s} \Big(\mathbf{S}[\downarrow k_i] \Big). \end{aligned} \tag{13}$$

Because each $\tilde{\mathbf{A}}_t[\rightarrow n]$ is an upper-triangular Toeplitz matrix, we easily know that matrix $\mathbf{S}$ is also an upper-triangular Toeplitz matrix with the following form:

$$\mathbf{S} = \begin{bmatrix} 0 & s_{m-1} & \cdots & s_1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_{m-1} \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{14}$$

and computing

$$\mathbf{S}(1,:) = \sum_{n\in\mathcal{N}} \Big(\tilde{\mathbf{A}}_t(1,:)[\rightarrow n] \Big) = \sum_{n\in\mathcal{N}} \Big(\mathbf{A}_t(1,:)[\rightarrow n] \Big) \tag{15}$$

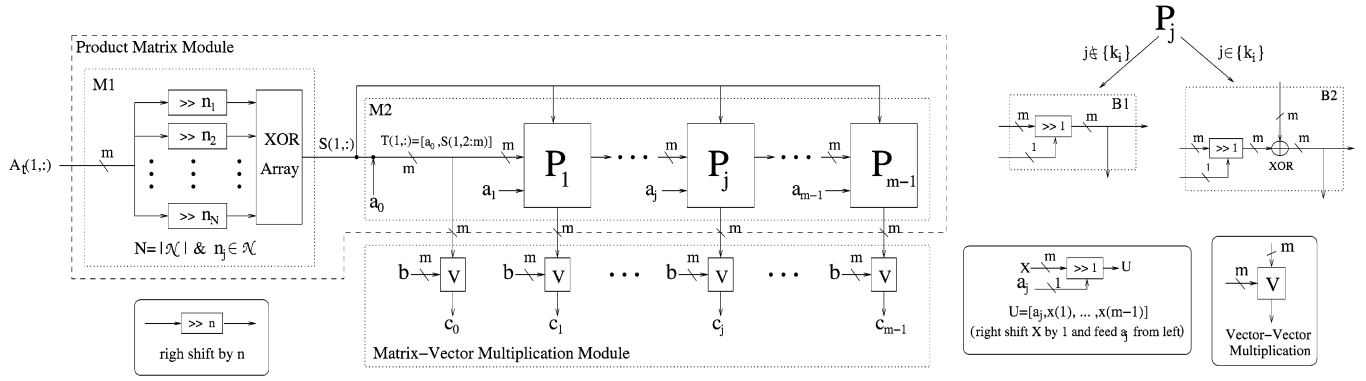

Fig. 2. General Mastrovito Multiplier architecture.

is sufficient to completely construct matrix $\mathbf{S}$. Since the first entry of $\mathbf{A}_t(1,:)$ is zero, the first $n+1$ entries of $\mathbf{A}_t(1,:)[\rightarrow n]$ are also zeroes. Recalling that $n=0$ is the least element in $\mathcal{N}$, we get the XOR complexity for computing $\mathbf{S}(1,:)$ based on (15) is

$$\sum_{(n\in\mathcal{N})\&(n\neq 0)}(m-n-1)=\sum_{n\in\mathcal{N}}(m-n-1)-(m-1) \quad (16)$$

and if binary tree structure is used, the delay will be $\lceil \log_2 |\mathcal{N}| \rceil T_X$.

After having obtained $\mathbf{S}$, we use (13) to compute the product matrix $\mathbf{M}$. From (10) and (14), we know that the addition of $\mathbf{A}_s$ and $\mathbf{S}$ does not need any XOR gates and the matrix $\mathbf{A}_s+\mathbf{S}$, denoted by $\mathbf{T}$, is a Toeplitz matrix. Therefore, (13) can be rewritten as

$$\mathbf{M}=\mathbf{T}+\mathbf{S}[\downarrow k_1]+\cdots+\mathbf{S}[\downarrow k_s]. \quad (17)$$

In order to compute (17) efficiently, we introduce the following observation:

Observation 3.1. Given an $m\times m$ upper-triangular Toeplitz matrix $\mathbf{S}$ and $m\times m$ matrix $\mathbf{P}$ whose last $(m-l)$ rows form a Toeplitz submatrix. If $m>j\geq l$, then computing the sum of vector $\mathbf{P}(j+1,:)$ and $\mathbf{S}(1,:)$ is sufficient to construct the sum of $\mathbf{P}$ and $\mathbf{S}[\downarrow j]$ and the last $(m-j)$ rows of the sum matrix still form a Toeplitz submatrix.

Applying Observation 3.1, we compute $\mathbf{M}$ based on (17) with linear tree structure as follows:

Algorithm 3.2.

1. Initially, set $\mathbf{Q}_0=\mathbf{T}$;
2. For $1\leq i\leq s$, construct $\mathbf{Q}_i=\mathbf{Q}_{i-1}+\mathbf{S}[\downarrow k_i]$ by computing $\mathbf{Q}_i(k_i+1,:)=\mathbf{Q}_{i-1}(k_i+1,:)+\mathbf{S}(1,:)$;
3. Finally, set $\mathbf{M}=\mathbf{Q}_s$.

In the above algorithm, for each i, $\mathbf{Q}_{i-1}(k_i+1,:)+\mathbf{S}(1,:)$ requires $(m-1)$ XOR gates, so we need $s(m-1)$ XOR gates to compute $\mathbf{M}$ with the delay of sT_X. In order to obtain the result $c(x)$ via $\mathbf{c}=\mathbf{M}\cdot\mathbf{b}$, we also need m^2 AND gates and $m(m-1)$ XOR gates to complete this matrix-vector multiplication. Its delay will be $T_A+\lceil \log_2 m\rceil T_X$ if binary tree structure is used.

Based on the above computation approach, we develop the dedicated Mastrovito multiplier architecture as shown in Fig. 2. Given irreducible polynomial $f(x)=x^m+x^{k_s}+\cdots+x^{k_1}+1$, set $\mathcal{N}$ is constructed using Algorithm 3.1. Product Matrix Module computes matrix $\mathbf{M}$ and consists of two blocks: M1 and M2. Block M1 generates the vector $\mathbf{S}(1,:)$ by computing $\sum_{n\in\mathcal{N}}(\mathbf{A}_t(1,:)[\rightarrow n])$. Supplied with $\mathbf{S}(1,:)$, block M2 computes the product matrix $\mathbf{M}$ using Algorithm 3.2. M2 contains $(m-1)$ $\mathrm{P_j}$ blocks, each one generates one row vector of $\mathbf{M}$. If $j\in\{k_i, 1\leq i\leq s\}$, then $\mathrm{P_j}$ is identical to block B2; otherwise, it is identical to block B1. The Matrix-Vector Multiplication Module computes $\mathbf{M}\cdot\mathbf{b}$ and consists of m identical V blocks. The block V computes the inner-product of two vectors of length m. The total complexity of the proposed original Mastrovito multiplier for general irreducible polynomial is given by

- XOR Complexity:

$$(m+s-1)(m-1)+\sum_{n\in\mathcal{N}}(m-n-1),$$

- AND Complexity: m^2,
- Delay: $T_A+(s+\lceil \log_2 |\mathcal{N}|\rceil+\lceil \log_2 m\rceil)T_X$.

It needs to be pointed out that, for a given irreducible polynomial, there likely exist some common items in the computation of (15) and (17). By sharing these common items, we may further optimize the hardware architecture of product matrix module to some extent. Thus, the multiplier architecture shown in Fig. 2 may need some corresponding modifications and the above XOR complexity value is actually an upper bound for general cases.

4 Modified Mastrovito Multiplier

Using a similar design methodology as in Section 3, in this section, we develop a systematic design approach for the modified Mastrovito multiplier [15], which is preferable if high-Hamming weight irreducible polynomial is used.

4.1 Proposed Theorem and Algorithm

For a polynomial $f(x)=f_mx^m+f_{m-1}x^{m-1}+\cdots+f_1x+f_0$, we define its *complementary polynomial* as

$$p(x)=(1+f_m)x^m+(1+f_{m-1})x^{m-1}+\cdots+(1+f_1)x+(1+f_0).$$

According to [15], we have the following theorem to construct matrix $\mathbf{U}$ in (5) using the complementary irreducible polynomial:

Theorem 4.1. *Given an irreducible polynomial*

$$f(x) = x^m + x^{k_s} + \cdots + x^{k_1} + 1,$$

its complementary polynomial can be written as

$$p(x) = x^{t_{m-s-1}} + \cdots + x^{t_1},$$

where $m > t_{m-s-1} > \cdots > t_1 > 0$. *Let* $\mathbf{p}$ *represent the coefficient vector of* $p(x)$, *then we can obtain the matrix* $\mathbf{U}$ *in (5) by adding two matrices* $\mathbf{V}$ *and* $\mathbf{Q}$, *where* $\mathbf{V}$ *is constructed recursively as*

$$\mathbf{V}(:,i) = \begin{cases} \mathbf{p}, & i = 1 \\ \mathbf{p}[\downarrow 1] + \Big(\mathbf{V}(m,1) + 1\Big) \cdot \mathbf{p} + e_1, & i = 2 \\ \mathbf{V}(:,i-1)[\downarrow 1] + \mathbf{V}(m,i-2) \cdot \mathbf{e}_1 & \\ + \Big(\mathbf{V}(m,i-2) + \mathbf{V}(m,i-1)\Big) \cdot \mathbf{p}, & 3 \le i \le m-1, \end{cases}$$

where $\mathbf{e}_1$ *is the first canonical unit vector and*

$$\mathbf{Q} = \mathbf{w} \cdot \Big(\mathbf{e}_1^T + \mathbf{V}(m,:)[\rightarrow 1]\Big),$$

where

$$\mathbf{w} = [\underbrace{1, \cdots, 1}_{m}]^T.$$

In the following, we propose a theorem and an algorithm to construct matrix $\mathbf{V}$ in Theorem 4.1 by adding a series of Toeplitz matrices together.

Theorem 4.2. *For the computation of matrix* $\mathbf{V}$ *in Theorem 4.1, we can always find two sets* $\mathcal{L} \subset \{0, 1, \cdots, m-2\}$ *and* $\mathcal{J} \subset \{1, \cdots, m-2\}$, *and*

$$V = \sum_{l \in \mathcal{L}} \mathbf{P}[\rightarrow l] + \sum_{j \in \mathcal{J}} \mathbf{E}_1[\rightarrow j],$$

where $\mathbf{E}_1$ *is defined in (7) and*

$$\mathbf{P} = \Big[\mathbf{p}, \mathbf{p}[\downarrow 1], \cdots, \mathbf{p}[\downarrow m-2]\Big].$$

The sets $\mathcal{L}$ and $\mathcal{J}$ in Theorem 4.2 are constructed by the following algorithm:

Algorithm 4.1.

Input: The parameters of complementary polynomial: m, $t_1, \cdots, t_{m-s-1}$;

Output: set $\mathcal{L} \subset \{0, 1, \cdots, m-2\}$ and $\mathcal{J} \subset \{1, \cdots, m-2\}$

Procedure:

1. Generate a weighted tree D according to the following properties:
   - The root d_1 always has one child node d_2 connected by an edge with weight $w = 1$. Besides d_2, root d_1 has at most $2(m-s-1)$ other child nodes, where the weight of edge $w \in \{(m-t_i), (m-t_i+1), 1 \le i \le (m-s-1)\}$;
   - $\forall\, d_j \neq d_1$, it has at most $2(m-s-1)$ child nodes, where the weight of edge $w \in \{(m-t_i), (m-t_i+1), 1 \le i \le (m-s-1)\}$;
   - Let $h(d_1, d_j)$ denote the weight of path from d_1 to d_j, where $h(d_1, d_1) = 0$, we have $\forall\, d_j$, if $\exists\, r \in \{(m-t_i), (m-t_i+1), 1 \le i \le (m-s-1)\}$ and $(h(d_1, d_j) + r) < m-1$, then d_j always has a child node d_l and the weight of edge between d_j and d_l is r;
   - $\forall d_j \in D$, $h(d_1, d_j) < m-1$.
2. Construct a subset, denoted as $\tilde{D}$, of all nodes in D such that it contains each node which is connected with its parent node by an edge with the weight $z \in \{(m-t_i+1), 1 \le i \le (m-s-1)\}$;
3. Construct multisets $\mathcal{H} = \{h(d_1, d_j),\ \forall d_j \in D\}$ and $\mathcal{G} = \{1, h(d_1, d_i), \forall\, d_i \in \tilde{D}\}$ and sets $\mathcal{L} = \mathcal{J} = \emptyset$;
4. For $0 \le j \le m-2$, do
   a. create $\mathcal{S}_j = \mathcal{T}_j = \emptyset$;
   b. $\forall\, h \in \mathcal{H}$, if $h = j$, then insert h into $\mathcal{S}_j$; $\forall\, g \in \mathcal{G}$, if $g = j$, then insert g into $\mathcal{T}_j$;
   c. if $(|\mathcal{S}_j|\ mod\ 2) = 1$, then insert j into the set $\mathcal{L}$; if $(|\mathcal{T}_j|\ mod\ 2) = 1$, then insert j into the set $\mathcal{J}$.

A proof of Theorem 4.1 is given in Appendix B in which Algorithm 4.1 is developed.

Example 4.1. Consider the construction of matrix $\mathbf{U}$ when the high Hamming weight irreducible polynomial $f(x) = x^7 + x^6 + x^5 + x^3 + x^2 + x + 1$ is being used. Its complementary polynomial is $p(x) = x^4$. We have $\{(m-t_i), (m-t_i+1), 1 \le i \le (m-s-1)\} = \{3, 4\}$. Applying Algorithm 4.1, we generate the tree D as shown in Fig. 3. From this tree, we get $\mathcal{H} = \{0, 1, 3, 4, 4, 5\}$ and $\mathcal{G} = \{1, 4, 5\}$. Thus, we obtain $\mathcal{L} = \{0, 1, 3, 5\}$ and $\mathcal{J} = \{1, 4, 5\}$. So, matrix $\mathbf{V}$ is computed as

$$\begin{aligned} \mathbf{V} &= \sum_{l \in \mathcal{L}} \mathbf{P}[\rightarrow l] + \sum_{j \in \mathcal{J}} \mathbf{E}_1[\rightarrow j] \\ &= \begin{bmatrix} 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 1&1&0&1&0&1 \\ 0&1&1&0&1&0 \\ 0&0&1&1&0&1 \end{bmatrix} + \begin{bmatrix} 0&1&0&0&1&1 \\ 0&0&1&0&0&1 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{bmatrix} \\ &= \begin{bmatrix} 0&1&0&0&1&1 \\ 0&0&1&0&0&1 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 1&1&0&1&0&0 \\ 0&1&1&0&1&0 \\ 0&0&1&1&0&1 \end{bmatrix}. \end{aligned}$$

Since $\mathbf{V}(7,:) = [0\ 0\ 1\ 1\ 0\ 1]$, we have $\mathbf{V}(7,:)[\rightarrow 1] = [0\ 0\ 0\ 1\ 1\ 0]$ and

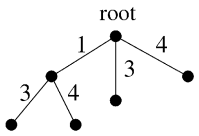

Fig. 3. Tree structure.

$$\mathbf{Q} = \mathbf{w} \cdot \left(\mathbf{e}_1^T + \mathbf{V}(7,:)[\rightarrow 1]\right) = \mathbf{w} \cdot [1\ 0\ 0\ 1\ 1\ 0].$$

Finally, we have

$$\mathbf{U} = \mathbf{V} + \mathbf{Q} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

4.2 Multiplier Architecture

In this section, based on the above theorem and algorithm, we develop an efficient computation approach and corresponding multiplier architecture for the modified Mastrovito multiplication scheme.

According to (5), (9), and Theorem 4.1, we have

$$\begin{aligned} \mathbf{M} &= [\mathbf{I}_{m\times m}, \mathbf{U}] \cdot \mathbf{A} = \mathbf{A}_s + \mathbf{U} \cdot \mathbf{A}_t \\ &= \underbrace{\mathbf{A}_s + \mathbf{V} \cdot \mathbf{A}_t}_{M_1} + \underbrace{\mathbf{Q} \cdot \mathbf{A}_t}_{M_2}, \end{aligned} \tag{18}$$

and the modified Mastrovito multiplication is carried out as

$$\mathbf{c} = \mathbf{M} \cdot \mathbf{b} = \mathbf{M}_1 \cdot \mathbf{b} + \mathbf{M}_2 \cdot \mathbf{b}.$$

In the following, we develop efficient approaches for computing $\mathbf{M}_1$ and $\mathbf{M}_2$, respectively. Let's begin with $\mathbf{M}_1$. Recall that the complementary polynomial is $p(x) = x^{t_{m-s-1}} + \cdots + x^{t_1}$, thus the matrix $\mathbf{P}$ in Theorem 4.2 can be written as

$$\mathbf{P} = \sum_{i=1}^{m-s-1} \mathbf{E}_{t_i+1}, \tag{19}$$

where $\mathbf{E}_i$ is defined in (7). Applying (12) and (19), we have

$$\begin{aligned} \mathbf{V} \cdot \mathbf{A}_t &= \Big(\sum_{l\in\mathcal{L}} \mathbf{P}[\rightarrow l] + \sum_{j\in\mathcal{J}} \mathbf{E}_1[\rightarrow j]\Big) \cdot \mathbf{A}_t \\ &= \sum_{i=1}^{m-s-1} \Big(\sum_{l\in\mathcal{L}} \tilde{\mathbf{A}}_t[\rightarrow l]\Big)[\downarrow t_i] + \sum_{j\in\mathcal{J}} \tilde{\mathbf{A}}_t[\rightarrow j], \end{aligned}$$

where $\tilde{\mathbf{A}}_t = [\mathbf{A}_t^T, \mathbf{o}]^T$ and $\mathbf{o}$ is an m-dimensional zero column vector. Denote $\sum_{j\in\mathcal{J}} \tilde{\mathbf{A}}_t[\rightarrow j]$ and $\sum_{l\in\mathcal{L}} \tilde{\mathbf{A}}_t[\rightarrow l]$ as $\tilde{\mathbf{S}}_1$ and $\tilde{\mathbf{S}}_2$, respectively, we have

$$\mathbf{M}_1 = \mathbf{A}_s + \tilde{\mathbf{S}}_1 + \sum_{i=1}^{m-s-1} \tilde{\mathbf{S}}_2[\downarrow t_i]. \tag{20}$$

Since each $\tilde{\mathbf{A}}_t[\rightarrow n]$ is an upper-triangular Toeplitz matrix, we know that both $\tilde{\mathbf{S}}_1$ and $\tilde{\mathbf{S}}_2$ are also upper-triangular Toeplitz matrices, and computing

$$\begin{aligned} \tilde{\mathbf{S}}_1(:,1) &= \sum_{j\in\mathcal{J}} \Big(\mathbf{A}_t(1,:)[\rightarrow j]\Big) \\ \tilde{\mathbf{S}}_2(:,1) &= \sum_{l\in\mathcal{L}} \Big(\mathbf{A}_t(1,:)[\rightarrow l]\Big) \end{aligned} \tag{21}$$

is sufficient to construct $\tilde{\mathbf{S}}_1$ and $\tilde{\mathbf{S}}_2$. Since the least element in $\mathcal{L}$ is always 0, similar to (16), we obtain the XOR complexities of computing $\tilde{\mathbf{S}}_1$ and $\tilde{\mathbf{S}}_2$ as

$$\sum_{j\in\mathcal{J}} (m-j-1) - (m - \min(\mathcal{J}) - 1) \text{ and}$$
$$\sum_{l\in\mathcal{L}} (m-l-1) - (m-1)$$

with the delay of $\lceil \log_2 |\mathcal{J}| \rceil T_X$ and $\lceil \log_2 |\mathcal{L}| \rceil T_X$, respectively. Similarly to Algorithm 3.2, for the computation of $\mathbf{M}_1$ using (20), we have the following algorithm:

Algorithm 4.2.

1. Initially, set $\mathbf{Q}_0 = \mathbf{A}_s + \tilde{\mathbf{S}}_1$;
2. For $1 \le i \le (m-s-1)$, construct $\mathbf{Q}_i = \mathbf{Q}_{i-1} + \tilde{\mathbf{S}}_2[\downarrow t_i]$ by computing
$$\mathbf{Q}_i(t_i+1,:) = \mathbf{Q}_{i-1}(t_i+1,:) + \tilde{\mathbf{S}}_2(1,:);$$
3. Finally, set $\mathbf{M}_1 = \mathbf{Q}_{m-s-1}$.

In the above algorithm, the construction of Toeplitz matrix $\mathbf{Q}_0$ doesn't need any gates. For $1 \le i \le (m-s-1)$, each step needs $(m-1)$ XOR gates, so the XOR complexity and delay of the above algorithm is $(m-s-1)(m-1)$ and $(m-s-1)T_X$, respectively.

Based on the above discussion, we conclude that the matrix $\mathbf{M}_1$ can be computed with the following procedure:

Procedure 4.1.

1. Given a high-Hamming weight irreducible polynomial $f(x) = x^m + x^{k_s} + \cdots + x^{k_1} + 1$, get its complementary polynomial $p(x) = x^{t_{m-s-1}} + \cdots + x^{t_1}$ and construct two sets $\mathcal{L}$ and $\mathcal{J}$ using Algorithm 4.1;
2. Construct $\tilde{\mathbf{S}}_1$ and $\tilde{\mathbf{S}}_2$ using (21), if we combine $\mathcal{L}$ and $\mathcal{J}$ together to form a new multiset $\mathcal{L}^*$, then the complexity of this step is

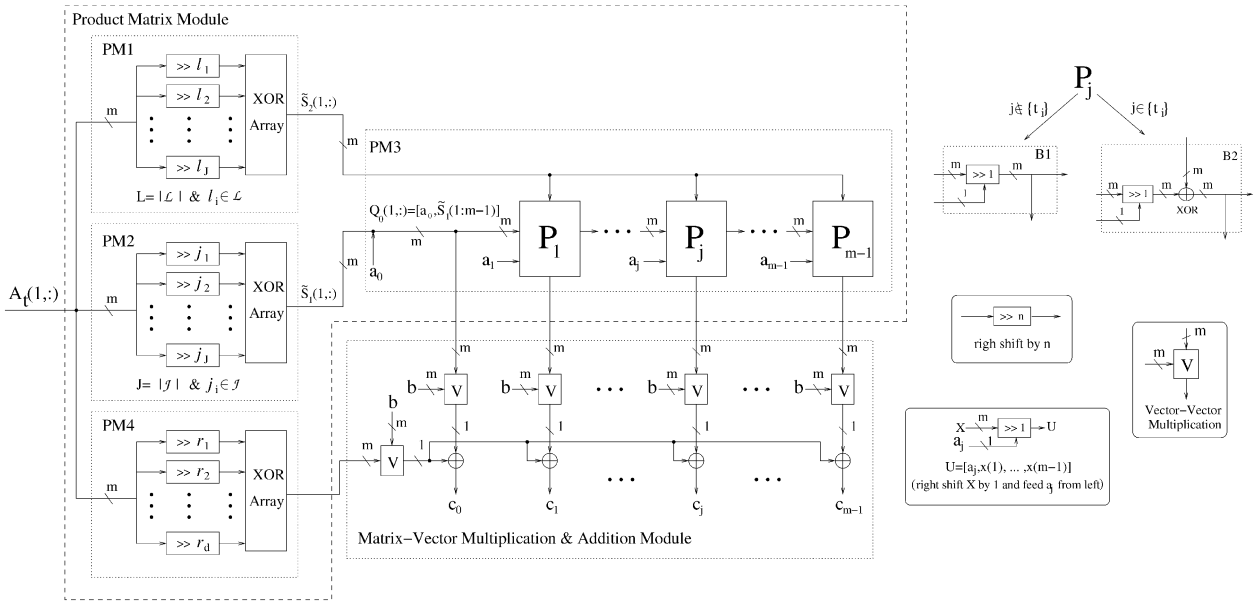

Fig. 4. General Modified Mastrovito Multiplier architecture.

- XOR complexity:

$$\sum_{l\in\mathcal{L}^*}(m-l-1)-2(m-1)+\min(\mathcal{J});$$

- Delay: $\lceil \log_2 D \rceil T_X$, where $D = max(|\mathcal{L}|, |\mathcal{J}|)$;

3. Then, use Algorithm 4.2 to compute matrix $\mathbf{M}_1$. For this step, we have

- XOR complexity: $(m-s-1)(m-1)$;
- Delay: $(m-s-1)T_X$.

Next, let's consider the computation of $\mathbf{M}_2$. Denote the vector $\mathbf{e}_1^T + \mathbf{V}(m,:)[\rightarrow 1]$ as $\mathbf{q}^T$. From Theorem 4.1, we know that all row vectors in $\mathbf{Q}$ are equal to $\mathbf{q}^T$. Thus, each row vector in $\mathbf{M}_2$ is equal to $\mathbf{q}^T \cdot \mathbf{A}_t$. Without loss of generality, suppose $\mathbf{q}^T$ has d nonzero entries whose positions are $r_d, \cdots, r_1$, respectively, where $r_d > \cdots > r_1$, then we have

$$\mathbf{q}^T \cdot \mathbf{A}_t = \sum_{i=1}^{d} \mathbf{A}_t(r_i,:). \tag{22}$$

Since $\mathbf{A}_t$ is an upper-triangular Toeplitz matrix, we have $\mathbf{A}_t(r_i,:) = \mathbf{A}_t(1,:)[\rightarrow (r_i-1)]$, thus (22) can be rewritten as

$$\mathbf{q}^T \cdot \mathbf{A}_t = \sum_{i=1}^{d} \mathbf{A}_t(1,:)[\rightarrow (r_i-1)]. \tag{23}$$

Therefore, the computation of $\mathbf{M}_2$ only needs $\sum_{i=2}^{d}(m-r_i)$ XOR gates with the delay of $\lceil \log_2 d \rceil T_X$.

In the above, we have developed the approaches for computing $\mathbf{M}_1$ and $\mathbf{M}_2$. In order to complete the $GF(2^m)$ multiplication, we only need to compute

$$\mathbf{c} = \mathbf{M} \cdot \mathbf{b} = \mathbf{M}_1 \cdot \mathbf{b} + \mathbf{M}_2 \cdot \mathbf{b}.$$

The matrix-vector multiplication $\mathbf{M}_1 \cdot \mathbf{b}$ requires $m(m-1)$ XOR and m^2 AND gates with the delay of $T_A + \lceil \log_2 m \rceil T_X$. Since all row vectors in $\mathbf{M}_2$ are identical and the first r_1 elements in each row are zeros, the matrix-vector multiplication $\mathbf{M}_2 \cdot \mathbf{b}$ only requires $(m - r_1 - 1)$ XOR and $(m - r_1)$ AND gates with the delay of $T_A + \lceil \log_2 (m-r_1) \rceil T_X$. The addition of $\mathbf{M}_1 \cdot \mathbf{b}$ and $\mathbf{M}_2 \cdot \mathbf{b}$ needs m XOR gates with the delay of T_X.

Based on the above computation approach, we develop the modified Mastrovito multiplier architecture as shown in Fig. 4. The set $\mathcal{L}$ and $\mathcal{J}$ are constructed using Algorithm 4.1. Product Matrix Module computes the two matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ and consists of four blocks: PM1, PM2, PM3, and PM4. Blocks PM1 and PM2 generate the vector $\tilde{\mathbf{S}}_2(1,:)$ and $\tilde{\mathbf{S}}_1(1,:)$, respectively. Block PM3 computes the matrix $\mathbf{M}_1$ using Algorithm 4.2. PM3 contains $(m-1)$ $\mathrm{P_j}$ blocks, each one generates one row vector of $\mathbf{M}$. If $j \in \{t_i, 1 \le i \le m-s-1\}$, then $\mathrm{P_j}$ is identical to block B2; otherwise, it is identical to block B1. Block PM4 computes the vector $\mathbf{q}^T \cdot \mathbf{A}_t$, the row vector of matrix $\mathbf{M}_2$, where $r_1, \cdots, r_d$ represent the position of the nonzero elements in $\mathbf{q}^T$. The Matrix-Vector Multiplication & Addition Module computes $\mathbf{M}_1 \cdot \mathbf{b} + \mathbf{M}_2 \cdot \mathbf{b}$. In this architecture, the operation of $\mathbf{M}_1 \cdot \mathbf{b}$ and $\mathbf{M}_2 \cdot \mathbf{b}$ can be performed in parallel. If their delays are denoted by $T_A + m_1 T_X$ and $T_A + m_2 T_X$, the total delay of the modified Mastrovito multiplier will be $T_A + (1 + \max(m_1, m_2))T_X$. Therefore, the total complexity of modified Mastrovito multiplier is

- XOR complexity:

$$(2m-s-2)(m-1) + \sum_{l\in\mathcal{L}^*}(m-l-1) + \sum_{i=1}^{d}(m-r_i) + \min(\mathcal{J}),$$

- AND complexity: $m^2 + m - r_1$,
- Delay: $T_A + (1 + \max(m_1, m_2))T_X$,

where multiset $\mathcal{L}^*$ is the combination of $\mathcal{L}$ and $\mathcal{J}$, d is the Hamming weight of $\mathbf{q}^T$, and r_i represents the index of nonzero element of $\mathbf{q}^T$, and

$$m_1 = m - s - 1 + \lceil \log_2 max((|\mathcal{L}|, |\mathcal{J}|)) \rceil + \lceil \log_2 m \rceil,$$
$$m_2 = \lceil \log_2 d \rceil + \lceil \log_2 (m - r_1) \rceil.$$

We note that, for given high Hamming weight irreducible polynomial, further hardware optimization is possible by sharing common items during computation of $\mathbf{M}_1$ and $\mathbf{M}_2$. So, the above XOR complexity result is also an upper bound for general cases.

5 Special Irreducible Polynomials

In Section 3, we presented an efficient computation approach of original Mastrovito multiplication for general cases and pointed out that further simplification can be achieved for specific irreducible polynomials by further exploiting subexpression sharing. In this section, we show that by applying our proposed explicit algorithm for the construction of set $\mathcal{N}$, we can easily obtain efficient multiplication schemes with further reduced complexity for several special irreducible polynomials.

5.1 $\frac{m}{2} \geq k_s$

In the following, we show that, for irreducible polynomial $f(x) = x^m + x^{k_s} + \cdots + x^{k_1} + 1$, where $\frac{m}{2} \geq k_s$, we can reduce the XOR complexity of the Mastrovito multiplier by computing $\mathbf{S}(1,:)$ in (15) using *linear tree* structure instead of binary tree. Since $\frac{m}{2} \geq k_s$, we easily have

$$\mathcal{N} = \{0, m - k_s, m - k_{s-1}, \cdots, m - k_1\}$$

and $|\mathcal{N}| = s + 1$. Thus, (15) can be simplified as

$$\mathbf{S}(1,:) = A_t(1,:) + \sum_{i=1}^{s} \Big(A_t(1,:)[\rightarrow (m - k_i)] \Big). \tag{24}$$

Using linear tree structure, we compute $\mathbf{S}(1,:)$, according to (24), as follows:

Algorithm 5.1.

1. Initially, set $\mathbf{v}_0^T = \mathbf{A}_t(1,:)$;
2. For $1 \leq i \leq s$, compute
$$\mathbf{v}_i^T = \mathbf{v}_{i-1}^T + \mathbf{A}_t(1,:)[\rightarrow (m - k_i)];$$
3. Finally, set $\mathbf{S}(1,:) = \mathbf{v}_s^T$.

The XOR complexity of the above algorithm is

$$\sum_{n \in \mathcal{N}} (m - n - 1) - (m - 1) = \sum_{i=1}^{s} (k_i - 1), \tag{25}$$

with the delay of sT_X. Compared with using binary tree, the XOR complexity doesn't change, but delay increases from $\lceil \log_2 (s+1) \rceil T_X$ to sT_X. Next, we will prove that, as compensation for the increased delay, the XOR complexity of computing $\mathbf{M}$ using Algorithm 3.2 can be reduced from $s(m-1)$ to $\sum_{i=1}^{s}(m - k_i)$ by sharing the intermediate results obtained in Algorithm 5.1.

Proof. In Algorithm 3.2, based on Observation 3.1, we construct each matrix $\mathbf{Q}_i$ by only computing

$$\mathbf{Q}_i(k_i + 1,:) = \mathbf{Q}_{i-1}(k_i + 1,:) + \mathbf{S}(1,:). \tag{26}$$

Moreover, since the last $m - k_i$ rows of each matrix $\mathbf{Q}_i$ form a Toeplitz submatrix, we have

$$\mathbf{Q}_{i-1}(k_i + 1,:) = \mathbf{Q}_{i-1}(k_{i-1} + 1,:)[\rightarrow (k_i - k_{i-1})]. \tag{27}$$

Therefore, based on (26), (27), and the fact that $\mathbf{Q}_0 = \mathbf{T} = \mathbf{S} + \mathbf{A}_s$, by induction we have

$$\begin{aligned} \mathbf{Q}_i(k_i + 1,:) = &\sum_{j=1}^{i} \Big(\mathbf{S}(1,:)[\rightarrow (k_i - k_j)] \Big) \\ &+ \mathbf{S}(1,:)[\rightarrow k_i] + \mathbf{A}_s(k_i + 1,:). \end{aligned} \tag{28}$$

From (10), we have

$$\mathbf{A}_s(k_i + 1,:) = [\underbrace{a_{k_i}, a_{k_{i-1}}, \cdots, a_1}_{k_i}, a_0, 0, \cdots, 0],$$
$$\mathbf{A}_t(1,:) = [\underbrace{0, a_{m-1}, \cdots, a_{k_i+1}}_{m-k_i}, a_{k_i}, \cdots, a_1].$$

Thus,

$$\mathbf{A}_s(k_i + 1,:) = \mathbf{A}_t(1,:)[\leftarrow (m - k_i)] + \mathbf{e}_{k_i+1}^T a_0, \tag{29}$$

where $\mathbf{e}_{k_i+1}$ is the $(k_i + 1)$th canonical unit vector. Therefore, (28) becomes

$$\begin{aligned} \mathbf{Q}_i(k_i + 1,:) = &\sum_{j=1}^{i} \Big(\mathbf{S}(1,:)[\rightarrow (k_i - k_j)] \Big) + \mathbf{S}(1,:)[\rightarrow k_i] \\ &+ \mathbf{A}_t(1,:)[\leftarrow (m - k_i)] + \mathbf{e}_{k_i+1}^T a_0. \end{aligned} \tag{30}$$

Since each $\mathbf{Q}_i(k_i + 1,:)$ is an m-dimensional row vector, the first k_i entries of $\mathbf{Q}_i(k_i + 1,:)$ are identical to the last k_i entries of $\mathbf{Q}_i(k_i + 1,:)[\rightarrow (m - k_i)]$. From (14), we also know that $\forall\, q \geq (m-1)$, $\mathbf{S}(1,:)[\rightarrow q]$ is a zero row vector. Therefore, according to (30), the last k_i entries of $\mathbf{Q}_i(k_i + 1,:)[\rightarrow (m - k_i)]$ are identical to the last k_i entries of $\tilde{\mathbf{q}}_i^T$, where $\tilde{\mathbf{q}}_i^T$ is defined as

$$\tilde{\mathbf{q}}_i^T = \sum_{j=1}^{i} \mathbf{S}(1,:)[\rightarrow (m - k_j)] + \mathbf{A}_t(1,:). \tag{31}$$

Substituting (24) into (31), we have

$$\begin{aligned} \tilde{\mathbf{q}}_i^T = &\sum_{j=1}^{i} \mathbf{A}_t(1,:)[\rightarrow (m - k_j)] \\ &+ \sum_{i=1}^{s} \sum_{j=1}^{i} \mathbf{A}_t(1,:)[\rightarrow ((m - k_j) + (m - k_i))] + \mathbf{A}_t(1,:). \end{aligned} \tag{32}$$

Since $\frac{m}{2} \geq k_s$, we have $\forall\, i, j \leq s$, $(m - k_i) + (m - k_j) \geq m$ and $\mathbf{A}_t(1,:)[\rightarrow ((m - k_j) + (m - k_i))]$ is actually a zero row vector. So, we can rewrite (32) as

$$\tilde{\mathbf{q}}_i^T = \sum_{j=1}^{i} \mathbf{A}_t(1,:)[\rightarrow (m - k_j)] + \mathbf{A}_t(1,:). \tag{33}$$

We note that $\tilde{\mathbf{q}}_i^T$ is identical to $\mathbf{v}_i^T$, which we have obtained in Algorithm 5.1. So, the first k_i entries of $\mathbf{Q}_i(k_i + 1,:)$ are equal to the last k_i entries of the

intermediate result $\mathbf{v}_i^T$ in Algorithm 5.1. Thus, in Algorithm 3.2, for each $1 \le i \le s$, we only need to compute the last $(m-k_i)$ entries of $\mathbf{Q}_i(k_i+1,:)$, which requires $(m-k_i)$ XOR gates, and the total XOR complexity of Algorithm 3.2 is $\sum_{i=1}^{s}(m-k_i)$ with the delay of sT_X. □

Therefore, the entire XOR complexity of computing product matrix $\mathbf{M}$ is

$$\sum_{i=1}^{s}(k_i-1)+\sum_{i=1}^{s}(m-k_i)=s(m-1)$$

with the delay of $2sT_X$. Moreover, the matrix-vector multiplication $\mathbf{M}\cdot\mathbf{b}$ requires $m(m-1)$ XOR and m^2 AND gates with the delay of $T_A+\lceil\log_2 m\rceil T_X$. Thus, for an irreducible polynomial in which $\frac{m}{2} \ge k_s$, if the linear tree structure is used to compute $\mathbf{S}(1,:)$, the entire complexity of Mastrovito multiplier is

- XOR complexity: $(m+s)(m-1)$,
- AND complexity: m^2,
- Delay: $T_A+(2s+\lceil\log_2 m\rceil)T_X$.

5.2 Trinomial

If $GF(2^m)$ is generated by irreducible trinomial $f(x)=x^m+x^n+1$, we have

$$\mathcal{N}=\left\{l(m-n), 0\le l\le\left\lfloor\frac{m-2}{m-n}\right\rfloor\right\}.$$

Therefore, we get $k=\lfloor\frac{m-2}{mn-}\rfloor$, (15) and (13) can be simplified as

$$\mathbf{S}(1,:)=\sum_{i=0}^{k}\Big(\mathbf{A}_t(1,:)[\rightarrow i(m-n)]\Big), \tag{34}$$

$$\mathbf{M}=\mathbf{A}_s+\mathbf{S}+\mathbf{S}[\downarrow n]. \tag{35}$$

Obviously, computing

$$\mathbf{M}(n+1,:)=\mathbf{A}_s(n+1,:)+\mathbf{S}(n+1,:)+\mathbf{S}(1,:) \tag{36}$$

is sufficient to construct $\mathbf{M}$ in (35). According to the complexity results presented in Section 3, we know that the XOR complexities of computing (34) and (36) are $\sum_{i=1}^{k}(m-i(m-n)-1)$ and $(m-1)$, respectively. In the following, we will see that the above complexity values can be further reduced. First, we show that $(m-n)$, instead of $(m-1)$, XOR gates are sufficient to complete the computation in (36). From (29), we have

$$\mathbf{A}_s(n+1,:)=\mathbf{A}_t(1,:)[\leftarrow(m-n)]+\mathbf{e}_{n+1}^T a_0$$

and, since $\mathbf{S}$ is an upper-triangular Toeplitz matrix, its $(n+1)$th row can be written as

$$\begin{aligned}\mathbf{S}(n+1,:)&=\mathbf{S}(1,:)[\rightarrow n]\\&=\Big(\sum_{i=0}^{k}\big(\mathbf{A}_t(1,:)[\rightarrow i(m-n)]\big)\Big)[\rightarrow n]\\&=\mathbf{A}_t(1,:)[\rightarrow n].\end{aligned}$$

So, (36) can be rewritten as

$$\begin{aligned}\mathbf{M}(n+1,:)=\mathbf{A}_t(1,:)[\leftarrow(m-n)]+\mathbf{e}_{n+1}^T a_0\\+\mathbf{A}_t(1,:)[\rightarrow n]+\mathbf{S}(1,:).\end{aligned} \tag{37}$$

Let's consider the addition of $\mathbf{A}_t(1,:)[\leftarrow(m-n)]$ and $\mathbf{S}(1,:)$ in (37). Because the last $(m-n)$ entries of $\mathbf{A}_t(1,:)[\leftarrow(m-n)]$ are zeros, we only need to compute the first n entries of $\mathbf{A}_t(1,:)[\leftarrow(m-n)]+\mathbf{S}(1,:)$. Obviously, the first n entries of $\mathbf{A}_t(1,:)[\leftarrow(m-n)]+\mathbf{S}(1,:)$ are equal to the last n entries of $\mathbf{A}_t(1,:)+\mathbf{S}(1,:)[\rightarrow(m-n)]$ and we have

$$\begin{aligned}&\mathbf{A}_t(1,:)+\mathbf{S}(1,:)[\rightarrow(m-n)]\\&=\mathbf{A}_t(1,:)+\Big(\sum_{i=0}^{k}\mathbf{A}_t(1,:)[\rightarrow i(m-n)]\Big)[\rightarrow(m-n)]\\&=\sum_{i=0}^{k}\Big(\mathbf{A}_t(1,:)[\rightarrow i(m-n)]\Big)+\mathbf{A}_t(1,:)[\rightarrow(k+1)(m-n)]\\&=\mathbf{S}(1,:)+\mathbf{A}_t(1,:)[\rightarrow(k+1)(m-n)].\end{aligned}$$

Since $k=\lfloor\frac{m-2}{m-n}\rfloor$, we have $(k+1)(m-n)\ge(m-1)$. Thus, the item $\mathbf{A}_t(1,:)[\rightarrow(k+1)(m-n)]$ is actually a zero vector and the first n entries of $\mathbf{A}_t(1,:)[\leftarrow(m-n)]+\mathbf{S}(1,:)$ are identical to the last n entries of $\mathbf{S}(1,:)$. Therefore, the addition of $\mathbf{A}_t(1,:)[\leftarrow(m-n)]$ and $\mathbf{S}(1,:)$ does not need any XOR gates. Furthermore, in (37), the sum of the other two items,

$$\mathbf{e}_{n+1}^T a_0+\mathbf{A}_t(1,:)[\rightarrow n]=[\underbrace{0,\cdots,0}_{n},\underbrace{a_0,a_{m-1},\cdots,a_{n+1}}_{m-n}],$$

has $(m-n)$ nonzero entries. Thus, we conclude that the computation of (37) only needs $(m-n)$ XOR gates.

The XOR complexity of computing (34) can be reduced by using either the *linear tree* or *hybrid tree* method, which will lead to different trade-off between XOR complexity and delay.

5.2.1 Linear Tree

If (34) is computed using linear tree structure as $\sum_{i=k}^{0}\mathbf{A}_t(1,:)[\rightarrow i(m-n)]$, we achieve the lowest XOR complexity with the delay increasing linearly with k. This approach has been thoroughly studied in [8], in which it was proven that $(n-1)$ XOR gates are sufficient to compute (34) with the delay of kT_X. We have known that the computation of (35) needs $(m-n)$ XOR gates with delay of $1T_X$. Thus, the total complexity of the Mastrovito multiplier in [8] was obtained as:

- XOR complexity: m^2-1,
- AND complexity: m^2,
- Delay: $T_A+(k+1+\lceil\log_2 m\rceil)T_X$, $k=\lfloor\frac{m-2}{m-n}\rfloor$.

Especially, it's pointed out in [8] that if $n=\frac{m}{2}$, then the XOR complexity can be further reduced from (m^2-1) to $(m^2-\frac{m}{2})$, with the delay reduced by $1T_X$.

5.2.2 Hybrid Tree

We have known that using linear tree structure to compute (34) will lead to a very low XOR complexity with the delay increasing linearly with k. However, when k is very large, the delay may be intolerable for applications requiring high speed. In the following, we present another approach for computing (34), where the XOR complexity also can be

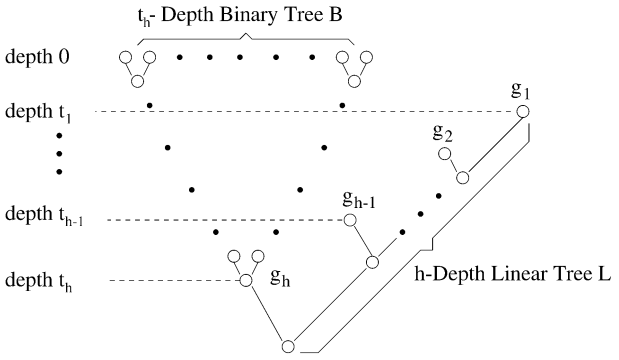

Fig. 5. Hybrid tree computation scheme.

reduced by exploiting subexpression sharing and the delay increases linearly with $\lfloor \log_2(k+1) \rfloor$.

Let the Hamming weight of $k+1$ be h and $\lfloor \log_2(k+1) \rfloor = t_h$, then $k+1$ can always be written as $k+1 = 2^{t_h} + 2^{t_{h-1}} + \cdots + 2^{t_1}$, where $t_h > t_{h-1} > \cdots > t_1$. If we denote $\mathbf{A}_t(1,:)[\rightarrow i(m-n)]$ as $\mathbf{A}_t(1,:)^{(i)}$, (34) can be rewritten as

$$\begin{aligned}\sum_{i=0}^{k}\left(\mathbf{A}_t(1,:)^{(i)}\right) &= \sum_{i=0}^{w_h-1}\left(\mathbf{A}_t(1,:)^{(i)}\right) + \sum_{i=w_h}^{w_{h-1}-1}\left(\mathbf{A}_t(1,:)^{(i)}\right) \\ &\quad + \cdots + \sum_{i=w_2}^{w_1-1}\left(\mathbf{A}_t(1,:)^{(i)}\right) \\ &= \mathbf{g}_h^T + \mathbf{g}_{h-1}^T + \cdots + \mathbf{g}_1^T,\end{aligned} \tag{38}$$

where

$$w_j = \sum_{i=j}^{h} 2^{t_i},\ 1 \le j \le h. \tag{39}$$

We compute the vector $\mathbf{g}_h^T$ using a binary tree B with the height of t_h as shown in Fig. 5, where each node represents a m-dimensional vector. At each depth j , there are 2^{t_h-j} nodes with the following values:

$$\sum_{i=0}^{2^j-1}\mathbf{A}_t(1,:)^{(i)}, \sum_{i=2^j}^{2^{(j+1)}-1}\mathbf{A}_t(1,:)^{(i)}, \cdots, \sum_{i=2^{t_h}-2^j}^{2^{t_h}-1}\mathbf{A}_t(1,:)^{(i)}. \tag{40}$$

In (40), all other items can be directly obtained by right shifting the first item $\sum_{i=0}^{2^j-1}\mathbf{A}_t(1,:)^{(i)}$ by $l \cdot 2^j$ columns, where $1 \le l \le 2^{t_h-j}-1$. So, only computing the first node at each depth is sufficient to construct the whole binary tree. The first node at depth j is computed by adding the first two nodes at depth $j-1$, where $(m - 2^{j-1}(m-n) - 1)$ XOR gates are required. Thus, the XOR complexity of computing $\mathbf{g}_h^T$ is

$$\begin{aligned}&\sum_{j=1}^{t_h}\left(m - 2^{j-1}(m-n) - 1\right) \\ &= (m-1)t_h - (2^{t_h}-1)(m-n).\end{aligned} \tag{41}$$

Moreover, each vector $\mathbf{g}_i^T$ in (38), where $1 \le i \le h-1$, can be obtained by right shifting the first node at depth t_i in the binary tree B. Thus, we compute $\sum_{i=h}^{1}\mathbf{g}_i^T$ using linear tree L and obtain a hybrid tree to compute (38) as shown in Fig. 5. From (38), we know that there are only $(m-1-w_{i+1}(m-n))$ nonzero entries in vector $\mathbf{g}_i^T$. Therefore, it requires

$$\begin{aligned}&\sum_{i=1}^{h-1}\Big(m-1-w_{i+1}(m-n)\Big) \\ &= (h-1)(m-1) + \sum_{i=2}^{h} w_i(m-n)\end{aligned} \tag{42}$$

XOR gates to compute the linear tree L. Combining (41) and (42) gives the total XOR complexity for the computation of (38) as

$$(t_h + h - 1)(m-1) + \left(\sum_{i=2}^{h} w_i - 2^{t_h} + 1\right)(m-n)$$

and the delay is $(t_h+1)T_X$. We have known that computing (35) requires $(m-n)$ XOR gates with the delay of $1T_X$. Therefore, in trinomial cases, if the above hybrid tree structure is employed to compute (34), the total computation complexity of Mastrovito multiplier will be

- XOR complexity:

$$(m + t_h + h - 1)(m-1) + (\sum_{i=2}^{h} w_i - 2^{t_h} + 2)(m-n),$$

- AND complexity: m^2,
- Delay: $T_A + (t_h + 2 + \lceil \log_2 m \rceil)T_X$,

where $t_h = \lfloor \log_2(k+1) \rfloor$, h is the Hamming weight of $(k+1)$, and w_i is defined in (39).

5.3 Pentanomial

A polynomial $f(x) = x^m + x^{k_s} + \cdots + x^{k_1} + 1$ is called *pentanomial* if $s = 3$. For general irreducible pentanomials, it's impossible to write the set $\mathcal{N}$ in a simple form, e.g., as in the trinomial case, and we have to use the general approach presented in Section 3 to design the product matrix module and perform the possible hardware optimization for the dedicated irreducible pentanomial.

In the following, we present two special irreducible pentanomials for which the set $\mathcal{N}$ has a simple form and the complexity of corresponding Mastrovito multipliers can be easily obtained.

5.3.1 Special Case 1

For irreducible pentanomials where $\frac{m}{2} \ge k_3$, it follows from the analysis in Section 5.1 that if linear tree structure is used to compute $\mathbf{S}(1,:)$, the total complexity of the Mastrovito multiplier is

- XOR complexity: $(m+3)(m-1)$,
- AND complexity: m^2,
- Delay: $T_A + (6 + \lceil \log_2 m \rceil)T_X$.

5.3.2 Special Case 2

For such irreducible pentanomials that

$$m - k_3 = k_3 - k_2 = k_2 - k_1 = r,$$

applying Algorithm 3.1, we have

$$\mathcal{N} = \left\{ n = 4l \cdot r, n = (4l+1) \cdot r,\ 0 \le l \le \left\lfloor \frac{d}{4} \right\rfloor \right\}, \tag{43}$$

where $d = \lfloor \frac{m-2}{r} \rfloor$. So, (15) can be simplified as

$$\mathbf{S}(1,:) = \sum_{l=0}^{\lfloor \frac{d}{4} \rfloor} \left(\mathbf{g}^T[\rightarrow 4l \cdot r] \right), \tag{44}$$

where

$$\mathbf{g}^T = \mathbf{A}_t(1,:) + \mathbf{A}_t(1,:)[\rightarrow r]. \tag{45}$$

The vector $\mathbf{g}^T$ is computed using $(m-r-1)$ XOR gates with the delay of $1T_X$. If we use linear tree to compute (44) as $\sum_{l=\lfloor \frac{d}{4} \rfloor}^{0} \mathbf{g}^T[\rightarrow 4l \cdot r]$, then, applying the results in [8], it only requires $(m-4r-1)$ XOR gates with the delay of $\lfloor \frac{d}{4} \rfloor T_X$. Therefore, the total XOR complexity of computing $\mathbf{S}(1,:)$ is $(2m-5r-2)$ with the delay of $(\lfloor \frac{d}{4} \rfloor + 1)T_X$. Furthermore, using Algorithm 3.2, we need $3(m-1)$ XOR gates to compute $\mathbf{M}$ with the delay of $3T_X$. Thus, in this case, the total complexity of the Mastrovito multiplier is given by

- XOR complexity: $(m+3)(m-1) + (2m-5r-2)$,
- AND complexity: m^2,
- Delay: $T_A + (\lfloor \frac{d}{4} \rfloor + 4 + \lceil \log_2 m \rceil)T_X$.

5.4 ESP

A polynomial $f(x) = 1 + x^r + x^{2r} + \cdots + x^{tr} + x^m$, where $(t+1)r = m$, is called an ESP (equally-spaced-polynomial). An ESP with $r = 1$ is usually referred as an AOP (all-one-polynomial). For irreducible ESP, applying Algorithm 3.1, we have $\mathcal{N} = \{0, r\}$, based on which it can be shown that matrix $\mathbf{U}$ in (5) always has the following form:

$$\mathbf{U} = \mathbf{L} + \mathbf{E}_1[\rightarrow r],$$

where $\mathbf{E}_1$ is defined in (7) and

$$\mathbf{L} = \begin{bmatrix} \mathbf{I}_{r\times r} & | & \\ \vdots & | & \mathbf{O}_{m\times m-1-r} \\ \mathbf{I}_{r\times r} & | & \end{bmatrix},$$

where $\mathbf{O}_{i\times j}$ and $\mathbf{I}_{i\times i}$ represent $i \times j$ zero matrix and $i \times i$ identity matrix, respectively. Therefore, the product matrix $\mathbf{M}$ can be computed as

$$\begin{aligned} \mathbf{M} &= \mathbf{A}_s + \mathbf{U} \cdot \mathbf{A}_t \\ &= \mathbf{A}_s + \mathbf{L} \cdot \mathbf{A}_t + \mathbf{E}_1[\rightarrow r] \cdot \mathbf{A}_t. \end{aligned}$$

Let $\mathbf{Q}_1$ denote $\mathbf{L} \cdot \mathbf{A}_t$, we have

$$\mathbf{Q}_1 = \mathbf{L} \cdot \mathbf{A}_t = \left[\mathbf{P}^T, \mathbf{P}^T, \cdots, \mathbf{P}^T \right]^T, \tag{46}$$

where $r \times m$ matrix $\mathbf{P} = \mathbf{A}_t(1:r,:)$. Let $\mathbf{Q}_2$ denote $\mathbf{A}_s + \mathbf{E}_1[\rightarrow r] \cdot \mathbf{A}_t$, we have

$$\mathbf{Q}_2 = \mathbf{A}_s + \mathbf{E}_1[\rightarrow r] \cdot \mathbf{A}_t = \mathbf{A}_s + \tilde{\mathbf{A}}_t[\rightarrow r], \tag{47}$$

where $\tilde{\mathbf{A}}_t = [\mathbf{A}_t^T, o]^T$. Obviously, $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are obtained without any computation. We perform the $GF(2^m)$ multiplication as

$$\mathbf{c} = \mathbf{M} \cdot \mathbf{b} = \mathbf{Q}_1 \cdot \mathbf{b} + \mathbf{Q}_2 \cdot \mathbf{b}.$$

According to (46), we know that only computing $\mathbf{P} \cdot \mathbf{b}$ is sufficient to obtain the result of $\mathbf{Q}_1 \cdot \mathbf{b}$. Since $\mathbf{P} = \mathbf{A}_t(1:r,:)$, we have

$$\mathbf{P}(i,:) = \mathbf{A}_t(i,:) = \mathbf{A}_t(1,:)[\rightarrow (i-1)],$$

which shows that the first i entries in $\mathbf{P}(i,:)$ are zeros. Therefore we get the complexity of computing $\mathbf{Q}_1 \cdot \mathbf{b}$ as

$$\begin{aligned} \#ofAND &= \sum_{i=1}^{r} (m-i) = mr - \frac{r^2+r}{2}, \\ \#ofXOR &= \sum_{i=1}^{r} (m-i-1) = mr - \frac{r^2+3r}{2}, \end{aligned}$$

and the delay is $T_A + \lceil \log_2 (m-1) \rceil T_X$.

From the definition of $\mathbf{A}_s$ and $\mathbf{A}_t$ in (10), we know that each row vector in $\mathbf{Q}_2(1:m-r,:)$ contains r zero entries and each row vector $\mathbf{Q}_2(m-r+i,:)$ contains $(r-i)$ zeros, where $1 \le i \le r$. Therefore, in the computation of $\mathbf{Q}_2 \cdot \mathbf{b}$, the numbers of AND and XOR gates are

$$\begin{aligned} \#ofAND &= (m-r)(m-r) + \sum_{i=1}^{r} (m-r+i) \\ &= m^2 - mr + \frac{r^2+r}{2} \\ \#ofXOR &= (m-r-1)(m-r) + \sum_{i=1}^{r} (m-r+i-1) \\ &= m^2 - mr + \frac{r^2+r}{2} - m, \end{aligned}$$

and its delay is $T_A + \lceil \log_2 m \rceil T_X$. Obviously, $\mathbf{Q}_1 \cdot \mathbf{b}$ and $\mathbf{Q}_2 \cdot \mathbf{b}$ can be computed in parallel. At last, we also need m XOR gates to add $\mathbf{Q}_1 \cdot \mathbf{b}$ and $\mathbf{Q}_2 \cdot \mathbf{b}$ together to get the final result $\mathbf{c}$. Therefore, we get the total complexity as follows:

- XOR complexity: $m^2 - r$,
- AND complexity: m^2,
- Delay: $T_A + (1 + \lceil \log_2 m \rceil)T_X$.

Here, we note that the above XOR complexity result is identical to that obtained in [14].

6 Conclusions

In this paper, we have presented a systematic design approach for Mastrovito multipliers. The complexity results are m^2 AND gates and at most $\sum_{n\in\mathcal{N}}(m-n-1) - (m-1)$ XOR gates. We note that, although the complexity results appear the same as those presented in [14], we propose an explicit algorithm to compute the set $\mathcal{N}$, which makes our design really practical. We have extended this design approach to the modified Mastrovito multiplication scheme, which is suitable for high-Hamming weight irreducible polynomials. For both original and modified Mastrovito multipliers with general irreducible polynomials, the developed computation approach effectively exploits subexpression sharing and the complexity analyses are given in detail. The corresponding hardware architectures for both cases are highly modular. Meanwhile, in this paper, we have studied several special irreducible polynomials. For trinomials and ESPs, the complexity results

match the best known results achieved in [8] and [14]. We also present another computation approach for trinomials to provide a trade-off between XOR complexity and delay. Moreover, several other special irreducible polynomials, which also lead to low-complexity implementation, have been discovered and corresponding complexities are given. Finally, we note that, with the explicit algorithms and design procedures, all the proposed efficient design schemes can be easily employed by VLSI automation design tools for dedicated bit-parallel $GF(2^m)$ multiplier design.

APPENDIX A
PROOF OF THEOREM 3.1

In the following, we prove Theorem 3.1 in two steps, through which Algorithm 3.1 is developed: 1) First, we will show that the matrix $\mathbf{U}$ is equal to the sum of a series of matrices constructed by the following procedure:

Procedure A-1.

1. Initially, set $i = n = 1$ and create a *matrix multiset* $\mathcal{W} = \{\mathbf{U}_1\}$. We define
$$\mathbf{U}_1 = [\underbrace{\mathbf{f}, \mathbf{o}, \cdots, \mathbf{o}}_{m-1}],$$
where $\mathbf{o}$ is the m-dimensional zero column vector;
2. $\forall\, \mathbf{U}_l \in \mathcal{W}$, shift down its ith column by one position to serve as its $(i+1)$th column:
$$\mathbf{U}_l(:, i+1) = \mathbf{U}_l(:, i)[\downarrow 1].$$
3. $\forall\, \mathbf{U}_l \in \mathcal{W}$, if the last entry of $\mathbf{U}_l$'s ith column vector, $\mathbf{U}_l(m, i)$, is 1, then {Increase n by 1, create $\mathbf{U}_l$'s child matrix $\mathbf{U}_n$ as
$$\mathbf{U}_n = [\underbrace{\mathbf{o}, \cdots, \mathbf{o}}_{i}, \mathbf{f}, \underbrace{\mathbf{o}, \cdots, \mathbf{o}}_{m-i-2}]$$
and insert $\mathbf{U}_n$ into $\mathcal{W}$ ($\mathbf{U}_l$ is called the parent matrix of $\mathbf{U}_n$)};
4. Increase i by 1, if $i = m-1$, procedure terminates, else return to Step 2.

Using the above procedure, we obtain a multiset $\mathcal{W}$ containing N matrices $\mathbf{U}_1, \cdots, \mathbf{U}_N$, where $N = |\mathcal{W}|$ is the order of $\mathcal{W}$. Each element matrix is m by $(m-1)$ and has the form:
$$\mathbf{U}_i = \Big[\underbrace{\mathbf{o}, \cdots, \mathbf{o}}_{r_i}, \underbrace{\mathbf{f}, \mathbf{f}[\downarrow 1], \cdots, \mathbf{f}[\downarrow m-2-r_i]}_{m-1-r_i}\Big].$$

In the following, we prove by induction that the sum of all $\mathbf{U}_i$s in $\mathcal{W}$ is identical to the matrix $\mathbf{U}$:
$$\mathbf{U}(:, l) = \sum_{i=1}^{N} \mathbf{U}_i(:, l), \quad 1 \le l \le m-1.$$

When $i = 1$, from Procedure A-1, we have $\mathbf{U}_1(:, 1) = f$ and $\mathbf{U}_i(:, 1) = \mathbf{o}$, $\forall\, i > 1$. From (4), we also have $\mathbf{U}(:, 1) = \mathbf{f}$. So, we get
$$\mathbf{U}(:, 1) = \sum_{i=1}^{N} \mathbf{U}_i(:, 1).$$

Assume, for $l \ge 1$, we have
$$\mathbf{U}(:, l) = \sum_{i=1}^{N} \mathbf{U}_i(:, l). \tag{A.1}$$

Applying the recursive relation between the successive columns of $\mathbf{U}$ as shown in (4), we compute $\mathbf{U}(:, l+1)$ as follows:
$$\begin{aligned}\mathbf{U}(:, l+1) &= \mathbf{U}(:, l)[\downarrow 1] + \mathbf{U}(m, l) \cdot \mathbf{f} \\ &= \Big(\sum_{i=1}^{N} \mathbf{U}_i(:, l)\Big)[\downarrow 1] + \Big(\sum_{i=1}^{N} \mathbf{U}_i(m, l)\Big) \cdot \mathbf{f} \\ &= \sum_{i=1}^{N} \Big(\mathbf{U}_i(:, l)[\downarrow 1]\Big) + \sum_{i=1}^{N} \Big(\mathbf{U}_i(m, l) \cdot \mathbf{f}\Big).\end{aligned} \tag{A.2}$$

According to Procedure A-1, we know that there always exist two integers $1 \le r \le t \le N$ and

- if $i \le r$, then matrix $\mathbf{U}_i$ is created before the lth iteration in the procedure and
$$\mathbf{U}_i(:, l)[\downarrow 1] = \mathbf{U}_i(:, l+1);$$
- if $r < i < t$, then $\mathbf{U}_i$ is created at the lth iteration, $\mathbf{U}_i(:, l)[\downarrow 1]$ is zero vector, and $\mathbf{U}_i(:, l+1) = \mathbf{f}$;
- if $t \le i$, then $\mathbf{U}_i$ is created after the lth iteration, both $\mathbf{U}_i(:, l)[\downarrow 1]$ and $\mathbf{U}_i(:, l+1)$ are zero vectors.

We also know that each $\mathbf{U}_i(m, l) = 1$ corresponds to a matrix created at the lth iteration. Thus, we have
$$\sum_{i=1}^{N} \Big(\mathbf{U}_i(:, l)[\downarrow 1]\Big) = \sum_{i=1}^{r} \mathbf{U}_i(:, l+1) + \sum_{i=t}^{N} \mathbf{U}_i(:, l+1),$$
$$\sum_{i=1}^{N} \Big(\mathbf{U}_i(m, l) \cdot \mathbf{f}\Big) = \sum_{i=r+1}^{t-1} \mathbf{U}_i(:, l+1).$$

Substituting the above two equations into (A.2), we get
$$\begin{aligned}\mathbf{U}(:, l+1) &= \Big(\sum_{i=1}^{r} \mathbf{U}_i(:, l+1) + \sum_{i=t}^{N} \mathbf{U}_i(:, l+1)\Big) \\ &\quad + \sum_{i=s+1}^{t-1} \mathbf{U}_i(:, l+1) \\ &= \sum_{i=1}^{N} \mathbf{U}_i(:, l+1).\end{aligned} \tag{A.3}$$

Thus, we have derived (A.3) from the assumption (A.1). Therefore, we can conclude that the sum of all matrices in $\mathcal{W}$ is equal to matrix $\mathbf{U}$.

2) Next, we introduce another method to construct multiset $\mathcal{W}$ with the aid of a weighted tree. Without loss of generality, let the irreducible polynomial be
$$f(x) = x^m + x^{k_s} + \cdots + x^{k_1} + 1,$$
where $m > k_s > \cdots > k_1 > 1$. Thus, for $0 < j < m$, only when $j \in \{(m - k_i), 1 \le i \le s\}$, the last entry of $\mathbf{f}[\downarrow (j-1)]$ is 1. From Procedure A-1, we have

$$\mathbf{U}_1 = \Big[\mathbf{f}, \mathbf{f}[\downarrow 1], \cdots, \mathbf{f}[\downarrow m-2]\Big]$$

and, except $\mathbf{U}_1$, each other matrix $\mathbf{U}_i$ in $\mathcal{W}$ can be constructed by shifting right its parent matrix by w columns, where $w \in \{(m-k_i), 1 \le i \le s\}$. Thus, we can construct a *weighted tree D* in which each node d_i represents the matrix $\mathbf{U}_i$ in $\mathcal{W}$ and the weight of each edge represents the number of columns by which the parent matrix right shifts to generate its child matrix. According to Procedure A-1, it can be shown that this tree is uniquely determined by the following property:

Property A-1.

1. *Each node d_j in D has at most s child nodes and each edge has the weight $w \in \{(m-k_i), 1 \le i \le s\}$.*
2. *Let d_1 denote the root and $h(d_1, d_j)$ denote the weight of path from d_1 to d_j, where $h(d_1, d_1) = 0$, we have $\forall\, d_j$, if $\exists\, r \in \{(m-k_i), 1 \le i \le s\}$ and $(h(d_1, d_j) + r) < m-1$, then d_j always has a child node d_l and the weight of edge between d_j and d_l is r.*
3. *$\forall d_j \in D$, $h(d_1, d_j) < m-1$.*

Since $\mathbf{U}_1$ is identical to the matrix $\mathbf{F}$ in Theorem 3.1, we can construct $\mathcal{W}$ as

$$\mathcal{W} = \{\mathbf{F}[\rightarrow h], \forall h \in \mathcal{H}\},$$

where $\mathcal{H} = \{h(d_1, d_i), \forall\, d_i \in D\}$. Obviously, the multiset $\mathcal{H}$ may contain some repeated elements, which means $\mathcal{W}$ may contain some identical matrices. Because here the addition is logic XOR, the sum of two identical matrices is actually a zero matrix. So, we can remove those repeated elements in *pairs* from multiset $\mathcal{H}$ using the following algorithm:

Algorithm A-1.

1. Initially, set $\mathcal{N} = \emptyset$.
2. For $0 \le j \le m-2$, do
   - create $\mathcal{S}_j = \emptyset$,
   - $\forall\, h \in \mathcal{H}$, if $h = j$, then insert h into $\mathcal{S}_j$.
3. If $(|\mathcal{S}_j| \bmod 2) = 1$, then insert j into the set $\mathcal{N}$.

Using the above algorithm, we construct a new set $\mathcal{N} \subset \{0, 1, \cdots, m-2\}$ and

$$\sum_{n \in \mathcal{N}} \mathbf{F}[\rightarrow n] = \sum_{h \in \mathcal{H}} \mathbf{F}[\rightarrow h] = \mathbf{U}.$$

We note that combining Property A-1 and Algorithm A-1 just produces Algorithm 3.1 in Section 3.1. □

APPENDIX B
PROOF OF THEOREM 4.2

Similarly to the proof of Theorem 3.1, we prove Theorem 4.2 in two steps, through which Algorithm 4.1 is developed: 1) First, we use the following procedure to construct two matrix multiset $\mathcal{W}$ and $\mathcal{Z}$, where the sum of all the matrices in these two multisets is equal to matrix $\mathbf{V}$.

Procedure B-1.

1. Initially, set $i = 2$, $n_1 = 2$, and $n_2 = 1$. Create two multisets $\mathcal{W} = \{\mathbf{W}_1, \mathbf{W}_2\}$ and $\mathcal{Z} = \{\mathbf{Z}_1\}$:

$$\mathbf{W}_1 = [\underbrace{\mathbf{p}, \mathbf{p}[\downarrow 1], \mathbf{o}, \cdots, \mathbf{o}}_{m-1}],$$
$$\mathbf{W}_2 = [\underbrace{\mathbf{o}, \mathbf{p}, \mathbf{o}, \cdots, \mathbf{o}}_{m-1}],$$
$$\mathbf{Z}_1 = [\underbrace{\mathbf{o}, \mathbf{e}_1, \mathbf{o}, \cdots, \mathbf{o}}_{m-1}],$$

   where $\mathbf{p}$ represents the coefficient vector of complementary irreducible polynomial $p(x)$ and $\mathbf{e}_1$ is the first canonical unit vector.
2. If $\mathbf{W}_1(m, 1) = 1$, then {Increase n_1 by 1, create $\mathbf{W}_1$'s child matrix $\mathbf{W}_{n_1} = \mathbf{W}_2$ and insert $\mathbf{W}_{n_1}$ into $\mathcal{W}$};
3. $\forall\, \mathbf{W}_l \in \mathcal{W}$ and $\forall\, \mathbf{Z}_l \in \mathcal{Z}$, do

$$\mathbf{W}_l(:, i+1) = \mathbf{W}_l(:, i)[\downarrow 1], \mathbf{Z}_l(:, i+1) = \mathbf{Z}_l(:, i)[\downarrow 1].$$

4. $\forall\, \mathbf{W}_l \in \mathcal{W}$, if $\mathbf{W}_l(m, i-1) = 1$, then {Increase both n_1 and n_2 by 1, create $\mathbf{W}_l$'s child matrix $\mathbf{W}_{n_1}$ and $\mathbf{Z}_{n_2}$ as

$$\mathbf{W}_{n_1} = [\underbrace{\mathbf{o}, \cdots, \mathbf{o}}_{i}, \mathbf{p}, \underbrace{\mathbf{o}, \cdots, \mathbf{o}}_{m-i-2}],$$
$$\mathbf{Z}_{n_2} = [\underbrace{\mathbf{o}, \cdots, \mathbf{o}}_{i}, \mathbf{e}_1, \underbrace{\mathbf{o}, \cdots, \mathbf{o}}_{m-i-2}]$$

   and insert $\mathbf{W}_{n_1}$ and $\mathbf{Z}_{n_2}$ into $\mathcal{W}$ and $\mathcal{Z}$, respectively};
5. $\forall\, \mathbf{W}_l \in \mathcal{W}$, if $\mathbf{W}_l(m, i) = 1$, then {Increase n_1 by 1, create $\mathbf{W}_l$'s child matrix $\mathbf{W}_{n_1}$ as

$$\mathbf{W}_{n_1} = [\underbrace{\mathbf{o}, \cdots, \mathbf{o}}_{i}, \mathbf{p}, \underbrace{\mathbf{o}, \cdots, \mathbf{o}}_{m-i-2}]$$

   and insert $\mathbf{W}_{n_1}$ into $\mathcal{W}$};
6. Increase i by 1, if $i = m-1$, procedure terminates, else return to Step 3.

Using the above procedure, we get two multisets $\mathcal{W}$ and $\mathcal{Z}$. Similarly to the proof of Theorem 3.1, based on the recursive relation of successive column vectors in matrix $\mathbf{V}$ as shown in Theorem 4.1, it can be proven by induction that matrix $\mathbf{V}$ is identical to the sum of all matrices in $\mathcal{W}$ and $\mathcal{Z}$:

$$\mathbf{V} = \sum_{i=1}^{|\mathcal{W}|} \mathbf{W}_i + \sum_{j=1}^{|\mathcal{Z}|} \mathbf{Z}_j. \tag{B.1}$$

2) Next, we introduce another method to construct multiset $\mathcal{W}$ and $\mathcal{Z}$ with the aid of a weighted tree. Since the complementary polynomial is $p(x) = x^{t_{m-s-1}} + \cdots + x^{t_1}$, the last entry of $\mathbf{p}[\downarrow (j-1)]$ or $\mathbf{p}[\downarrow (j-2)]$ is 1 only when $j \in \{(m-t_i), (m-t_i+1), 1 \le i \le (m-s-1)\}$. According to Procedure B-1, each child matrix in multiset $\mathcal{W}$ can be obtained by right shifting its parent matrix by $w \in \{(m-t_i), (m-t_i+1), 1 \le i \le (m-s-1)\}$ columns and the whole construction process begins from two matrices: $\mathbf{W}_1$ and $\mathbf{W}_2 = \mathbf{W}_1[\rightarrow 1]$. Therefore, we can construct a *weighted tree D* in which every node d_i represents the matrix $\mathbf{W}_i$ in $\mathcal{W}$ and the weight of each edge represents the number of columns by which the parent matrix right shifts to generate its child matrix. According to Procedure B-1, it can be shown that the tree D is uniquely determined by the following property:

Property B-1.

1. *The root d_1, representing matrix $\mathbf{W}_1$, always has one child node d_2 (representing $\mathbf{W}_2$) connected by an edge with weight $w = 1$. Besides d_2, root d_1 has at most $2(m - s - 1)$ other child nodes, where the weight of edge*

   $$w \in \{(m - t_i), (m - t_i + 1), 1 \leq i \leq (m - s - 1)\}.$$

2. *$\forall\, d_j \neq d_1$, it has at most $2(m - s - 1)$ child nodes, where the weight of edge*

   $$w \in \{(m - t_i), (m - t_i + 1), 1 \leq i \leq (m - s - 1)\}.$$

3. *Let $h(d_1, d_j)$ denote the weight of path from d_1 to d_j, where $h(d_1, d_1) = 0$, we have $\forall\, d_j$, if $\exists\, r \in \{(m - t_i), (m - t_i + 1), 1 \leq i \leq (m - s - 1)\}$ and $(h(d_1, d_j) + r) < m - 1$, then d_j always has a child node d_l and the weight of edge between d_j and d_l is r.*
4. *$\forall d_j \in D$, $h(d_1, d_j) < m - 1$.*

Since $\mathbf{W}_1$ is identical to the matrix $\mathbf{P}$ in Theorem 4.2, we can construct the multiset $\mathcal{W}$ as follows:

$$\mathcal{W} = \{\mathbf{P}[\rightarrow h], \forall h \in \mathcal{H}\},$$

where $\mathcal{H} = \{h(d_1, d_i), \forall\, d_i \in D\}$. Moreover, the above weighted tree D also can be used to represent all the element matrices in multiset $\mathcal{Z}$. Let root d_1 represent matrix $\mathbf{Z}_0 = \mathbf{E}_1$, each other node d_i represents a matrix generated through shifting $\mathbf{Z}_0$ right by $h(d_1, d_i)$ columns. According to Procedure B-1, if we introduce such a subset of all nodes in D, denoted as $\tilde{D}$, that it contains each node which is connected with its parent node by an edge with the weight $z \in \{(m - t_i + 1), 1 \leq i \leq (m - s - 1)\}$, we have

$$\mathcal{Z} = \{\mathbf{E}_1[\rightarrow g], \forall g \in \mathcal{G}\},$$

where $\mathcal{G} = \{1, h(d_1, d_i), \forall\, d_i \in \tilde{D}\}$. Therefore, (B.1) can be rewritten as

$$\mathbf{V} = \sum_{i \in \mathcal{H}} \mathbf{W}_i + \sum_{j \in \mathcal{G}} \mathbf{Z}_j.$$

Similarly to the discussion in Appendix A, using Algorithm A-1, we can remove those repeated elements in pairs from multiset $\mathcal{H}$ and $\mathcal{G}$ and obtain $\mathcal{L} \subset \{0, 1, \cdots, m - 2\}$ and $\mathcal{J} \subset \{1, \cdots, m - 2\}$, respectively, and

$$\mathbf{V} = \sum_{l \in \mathcal{L}} \mathbf{P}[\rightarrow l] + \sum_{j \in \mathcal{J}} \mathbf{E}_1[\rightarrow j].$$

Summarizing the above approach of constructing $\mathcal{L}$ and $\mathcal{J}$, we get Algorithm 4.1 in Section 4.1. □

REFERENCES

[1] S.A. Vanstone and P.C. Oorschot, *An Introduction to Error Correcting Codes with Applications.* Kluwer, 1989.
[2] A.J. Menezes, *Handbook of Applied Cryptography.* CRC, 1997.
[3] A.J. Menezes, *Applications of Finite Fields.* Kluwer Academic, 1994.
[4] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and Their Applications.* Cambridge Univ. Press, 1994.
[5] D. Jungnickel, *Finite Fields: Structure and Arithmetics.* Wissenschaftsverlag, 1993.
[6] E.D. Mastrovito, "VLSI Designs for Multiplication over Finite Fields GF(2^m)," *Proc. Sixth Int'l Conf. Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes (AAECC-6),* pp. 297-309, July 1988.
[7] K.K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation.* John Wiley & Sons, 1999.
[8] B. Sunar and Ç.K. Koç, "Mastrovito Multiplier for All Trinomials," *IEEE Trans. Computers,* vol. 48, no. 5, pp. 522-527, May 1999.
[9] T. Itoh and S. Tsujii, "Structure of Parallel Multipliers for a Class of Finite Fields GF(2^m)," *Information and Computations,* vol. 83, pp. 21-40, 1989.
[10] M.A. Hasan, M.Z. Wang, and V.K. Bhargava, "Modular Construction of Low Complexity Parallel Multipliers for a Class of Finite Fields GF(2^m)," *IEEE Trans. Computers,* vol. 41, no. 8, pp. 962-971 Aug. 1992.
[11] Ç.K. Koç and B. Sunar, "Low-Complexity Bit-Parallel Canonical and Normal Basis Multipliers for a Class of Finite Fields," *IEEE Trans. Computers,* vol. 47, no. 3, pp. 353-356, Mar. 1998.
[12] H. Wu and M.A. Hasan, "Low-Complexity Bit-Parallel Multipliers for a Class of Finite Fields," *IEEE Trans. Computers,* vol. 47, no. 8, pp. 883-887, Aug. 1998.
[13] A. Halbutogullari and Ç.K. Koç, "Mastrovito Multiplier for General Irreducible Polynomials," *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes,* pp. 498-507, 1999.
[14] A. Halbutogullari and Ç.K. Koç, "Mastrovito Multiplier for General Irreducible Polynomials," *IEEE Trans. Computers,* vol. 49, no. 5, pp. 503-518, May 2000.
[15] L. Song and K.K. Parhi, "Low Complexity Modified Mastrovito Multipliers over Finite Fields GF(2^m)," *Proc. IEEE Int'l Symp. Circuits and Systems,* vol. 1, pp. 508-512, May 1999.
[16] T. Zhang and K.K. Parhi, "Systematic Design Approach of Mastrovito Multipliers over GF(2^m)," *Proc. IEEE Workshop Signal Processing Systems (SiPS),* pp. 507-516, Oct. 2000.
[17] G.H. Golub and C.F. Van Loan, *Matrix Computations,* third ed. Johns Hopkins Univ. Press, 1996.

Tong Zhang (S'98) received the BS and MS degrees in electrical engineering from the Xian Jiaotong University, Xian, China, in 1995 and 1998, respectively. He is pursuing the PhD degree in electrical engineering at the University of Minnesota. His current research interests include design of VLSI architectures and circuits for digital signal processing and communication systems, with the emphasis on error-correcting coding and finite field arithmetic. He is a student member of the IEEE.

Keshab K. Parhi (S'85-M'88-SM'91-F'96) received the BTech, MSEE, and PhD degrees from the Indian Institute of Technology, Kharagpur (India) (1982), the University of Pennsylvania, Philadelphia (1984), and the University of California at Berkeley (1988), respectively. He is a distinguished McKnight University Professor of Electrical and Computer Engineering at the University of Minnesota, Minneapolis, where he also holds the Edgar F. Johnson Professorship. His research interests include all aspects of VLSI implementations of broadband access networks. He is currently working on VLSI adaptive digital filters, equalizers and beamformers, error control coders and cryptography architectures, low-power digital systems, and computer arithmetic. He has published more than 320 papers in these areas. He has authored the text book *VLSI Digital Signal Processing Systems* (Wiley, 1999) and coedited the reference book *Digital Signal Processing for Multimedia Systems* (Dekker, 1999). He received the 2001 W.R.G. Baker prize paper award from the IEEE, a Golden Jubilee medal from the IEEE Circuits and Systems Society in 1999, a 1996 Design Automation Conference best paper award, the 1994 Darlington and the 1993 Guillemin-Cauer best paper awards from the IEEE Circuits and Systems Society, the 1991 paper award from the IEEE Signal Processing Society, the 1991 Browder Thompson prize paper award from the IEEE, and the 1992 Young Investigator Award of the US National Science Foundation.

Dr. Parhi has served on editorial boards of the *IEEE Transactions on Circuits and Systems*, the *IEEE Transactions on Signal Processing*, the *IEEE Transactions on Circuits and Sytems-Part II: Analog and Digital Signal Processing*, the *IEEE Transactions on VLSI Systems*, and the *IEEE Signal Processing Letters*. He is an editor of the *Journal of VLSI Signal Processing*. He is the guest editor of a special issue of the *IEEE Transactions on Signal Processing*, two special issues of the *Journal of VLSI Signal Processing* and a special issue of the *Journal of Analog Integrated Circuits and Signal Processing*. He served as technical program cochair of the 1995 IEEE Workshop on Signal Processing and the 1996 ASAP conference, and is the general chair of the 2002 IEEE Workshop on Signal Processing Systems. He serves on numerous technical program committees and was a distinguished lecturer of the IEEE Circuits and Systems Society (1994-1999).