Systematic Design Approach of Mastrovito Multipliers over $GF(2^m)$

Tong Zhang and Keshab K. Parhi
Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN, USA
E-mail:{tzhang, parhi}@ece.umn.edu *

Abstract — **This paper considers the design of low-complexity bit-parallel dedicated finite field multiplier. A systematic design approach of Mastrovito multiplier is proposed, which is applicable to $GF(2^m)$ generated by an arbitrary irreducible polynomial. This approach extensively exploits the spatial correlation of matrix elements in Mastrovito multiplication to reduce the complexity. The developed general Mastrovito multiplier is highly modular, which is desirable for VLSI hardware implementation. Meanwhile, the presented approach can be used to develop efficient Mastrovito multipliers for several special irreducible polynomials, such as trinomial and equally-spaced-polynomial, and further find some other special irreducible polynomials which can also lead to low-complexity multipliers.**

1 INTRODUCTION

Finite fields $GF(2^m)$ have received a lot of attention because of their applications in cryptography, error-control coding and computer algebra. 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 efficient in the sense that it gives designers more freedom on irreducible polynomial selection and hardware optimization. In this paper, we are interested in the design of bit-parallel $GF(2^m)$ multiplier using standard basis.

The standard basis multiplication involves two steps: polynomial multiplication and modular reduction. An efficient dedicated bit-parallel multiplier was proposed by Mastrovito [3], where a product matrix is introduced to combine these two steps together. Mastrovito multipliers using two special irreducible polynomials, *trinomial* and *equally-spaced-polynomial* (ESP), have been studied by many researchers in order to reduce the space complexity [4]-[7]. It has been shown in [4] 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 [4], [7] has found that the space complexity for Mastrovito multiplier using irreducible ESP $x^m + x^{tr} + \cdots + x^r + 1$, where $(t + 1)r = m$, can be reduced to

* This research was supported by the Army Research Office by grant number DA/DAAG55-98-1-0315.

$(m^2 - r)$ XOR gates and m^2 AND gates. Furthermore, [7] presents a new formulation of the Mastrovito multiplication 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 $(m-1)(m+k)+\sum_{j\in\mathcal{N}}(m-1-j)$ XOR gates, where $\mathcal{N} \subset \{0, 1, \cdots, m-2\}$. However, [7] fails to find a method to compute the set $\mathcal{N}$, which makes its result less practicable.

In this paper, we generalize the approach of [4] in a different way compared with [7]. We propose a theorem and algorithm (which can compute the $\mathcal{N}$ in [7]) about the construction of reduction matrix **R**, based on which we develop an efficient design approach of Mastrovito multiplier for an arbitrary irreducible polynomial. This approach exploits the spatial correlation of matrix elements in Mastrovito multiplication to reduce the complexity. Explicit algorithm and architecture are presented and the complexity analysis is given in detail. For irreducible trinomial and ESP, this design approach provides alternative methods to those of [4] and [7] which give the same complexity values. Furthermore, this proposed approach can be used to find some other special irreducible polynomials which will also lead to low-complexity multiplier, which is especially desirable when neither an irreducible trinomial nor an irreducible ESP exists.

This paper is organized as follows. We introduce the fundamentals of finite field and Mastrovito multiplier and the notation of this paper in Section 2. Section 3 proposes a theorem and algorithm about the computation of reduction matrix **R**, based on which a systematic design approach of Mastrovito multiplier and corresponding architecture are developed in Section 4. Applying the proposed algorithm, Section 5 presents two new special irreducible polynomials which also lead to low-complexity multipliers.

2 PRELIMINARIES AND NOTATION

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

$$GF(2^m) = \{a | a = a_{m-1}\alpha^{m-1} + a_{m-2}\alpha^{m-2} + \cdots + a_1\alpha + a_0,\ a_i \in GF(2)\}$$

where α is the root of irreducible polynomial $f(x)$ of degree m over $GF(2^m)$. Finite field addition is carried out as polynomial addition over $GF(2^m)$ using bit-independent XOR operation.

Let $a(\alpha), b(\alpha) \in GF(2^m)$, and $f(x)$ be the irreducible polynomial generating $GF(2^m)$. In order to compute the multiplication $c(\alpha) = (a(\alpha) \cdot b(\alpha)) \mathrm{mod} f(x)$, we first compute the product polynomial $d(\alpha) = a(\alpha) \cdot b(\alpha)$ and then reduce $d(\alpha)$ using

$f(x)$ to get result $c(\alpha)$. We can compute coefficients of $d(\alpha)$ as follows:

$$\begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{m-1} \\ d_m \\ d_{m+1} \\ \vdots \\ d_{2m-2} \end{bmatrix} = \begin{bmatrix} a_0 & 0 & \cdots & 0 & 0 \\ a_1 & a_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m-2} & a_{m-3} & \cdots & a_0 & 0 \\ a_{m-1} & a_{m-2} & \cdots & a_1 & a_0 \\ 0 & a_{m-1} & \cdots & a_2 & a_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{m-1} & a_{m-2} \\ 0 & 0 & \cdots & 0 & a_{m-1} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{m-1} \end{bmatrix} \tag{1}$$

where the $(2m-1) \times m$ matrix in (1) is denoted by $\mathbf{A}$. Then we perform modular reduction on $d(\alpha)$ to get $c(\alpha)$. We can prove that this reduction can be expressed as $\mathbf{c} = \mathbf{R} \cdot \mathbf{d}$, where $\mathbf{c}$ and $\mathbf{d}$ represent the coefficient vectors of $c(\alpha)$ and $d(\alpha)$, respectively. The $m \times (2m-1)$ matrix $\mathbf{R}$ is called reduction matrix and its entries are solely dependent on the coefficients of irreducible polynomial $f(x)$. Thus we can get the result $c(\alpha)$ via product $\mathbf{c} = \mathbf{R} \cdot \mathbf{A} \cdot \mathbf{b}$. The product of $\mathbf{R}$ and $\mathbf{A}$ is just the so-called Mastrovito multiplication matrix $\mathbf{M}$. Therefore, by introducing $\mathbf{M} = \mathbf{R} \cdot \mathbf{A}$, the $GF(2^m)$ multiplication can be done as $\mathbf{c} = \mathbf{M} \cdot \mathbf{b}$.

In what follows, column vector and matrix are represented by small and capital boldfaced characters, respectively. Matlab notations of matrix and vector are used, e.g., **Z**(i,:), **Z**(:,j) and **Z**(i,j) represent the i-th row vector, j-th column vector and the entry with position (i,j) in matrix **Z**, respectively, and **v**(i) represents i-th entry in vector **v**. The operations of shift by feeding zero are represented by corresponding arrows, e.g., $\mathbf{v}[\downarrow 2]$ and $\mathbf{U}[\rightarrow 1]$ represent down shift of vector **v** by 2 positions and right shift of matrix **U** by 1 column, respectively, which are explicitly given as:

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

where **o** represents the zero column vector.

Finally, we note that the AND and XOR gates considered in this paper are all 2-input, whose delays are denoted as T_A and T_X, respectively.

3 PROPOSED THEOREM AND ALGORITHM

In this section, we will propose a theorem and algorithm about the construction of reduction matrix **R**:

Theorem 3.1 *Let $f(x) = x^m + f_{m-1}x^{m-1} + \cdots + f_1 x + 1$ be the irreducible polynomial generating $GF(2^m)$. We can generate a set $\mathcal{N} \subset \{0, 1, \cdots, m-2\}$, and construct the reduction matrix* **R** *as*

$$\mathbf{R} = \left[\mathbf{I}_{m \times m}, \sum_{n \in \mathcal{N}} \mathbf{F}[\rightarrow n]\right] \tag{2}$$

where $\mathbf{I}_{m\times m}$ *represents* $m \times m$ *identity matrix,* $\mathbf{F} = [\mathbf{f}, \mathbf{f}[\downarrow 1], \cdots, \mathbf{f}[\downarrow m-2]]$ *and* $\mathbf{f} = [1, f_1, \cdots, f_{m-1}]^T$ *is the coefficient vector of* $f(x)$. ■

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 obtained 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 which has 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 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 set* $\mathcal{H} = \{h(d_1, d_j),\ \forall d_j \in D\}$ *and* $\mathcal{N} = \{\varnothing\}$;

3. *For* $0 \le j \le m - 2$, *do*

   (a) *create* $\mathcal{S}_j = \{\varnothing\}$;

   (b) $\forall\, h \in \mathcal{H}$, *if* $h = j$, *then insert* h *into* $\mathcal{S}_j$;

   (c) *if* $(|\mathcal{S}_j| \bmod 2) = 1$, *then insert* j *into the set* $\mathcal{N}$. ■

Here $|\mathcal{S}_j|$ stands for the order of set $\mathcal{S}_j$. From 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 construction of* $\mathbf{R}$ *when the irreducible polynomial* $f(x) = x^5 + x^4 + x^3 + x^2 + 1$ *is being used. We have* $\{(m - k_i), 1 \le i \le s\} = \{1, 2, 3\}$. *Applying Algorithm 3.1, we first generate the tree D as shown in Fig. 1 and get the set* $\mathcal{H} = \{0, 1, 2, 2, 3, 3, 3, 3\}$. *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 matrix* $\mathbf{R}$ *can be computed as*

$$\mathbf{R} = \left[\mathbf{I}_{5\times 5}, \sum_{n\in\mathcal{N}} \mathbf{F}[\rightarrow n]\right] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

where

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \qquad \blacksquare$$

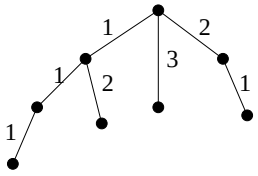

Figure 1: Tree structure

4 GENERAL IRREDUCIBLE POLYNOMIALS

Given a general irreducible polynomial $f(x) = x^m + x^{k_s} + \cdots + x^{k_1} + 1$, its coefficient vector can be expressed 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 i-th *canonical* vector:

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

Define

$$\mathbf{E}_j = \Big[\mathbf{e}_j, \mathbf{e}_j[\downarrow 1], \cdots, \mathbf{e}_j[\downarrow m-2]\Big]. \tag{3}$$

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

$$\sum_{n \in \mathcal{N}} \mathbf{F} = \sum_{i=0}^{s} \sum_{n \in \mathcal{N}} \mathbf{E}_{k_i+1}[\rightarrow n]. \tag{4}$$

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

$$\mathbf{A} = [\mathbf{A}_s^T, \mathbf{A}_t^T]^T \tag{5}$$

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:

$$\mathbf{A}_s = \begin{bmatrix} a_0 & 0 & \cdots & 0 \\ a_1 & a_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1} & a_{m-2} & \cdots & a_0 \end{bmatrix}, \ \mathbf{A}_t = \begin{bmatrix} 0 & a_{m-1} & \cdots & a_1 \\ 0 & 0 & \cdots & a_2 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{m-1} \end{bmatrix} \tag{6}$$

Substituting (2), (4) and (5) into $\mathbf{M} = \mathbf{R} \cdot \mathbf{A}$, we get

$$\mathbf{M} = \mathbf{A}_s + \sum_{i=0}^{s} \sum_{n \in \mathcal{N}} \Big(\mathbf{E}_{k_i+1}[\rightarrow n] \cdot \mathbf{A}_t \Big). \tag{7}$$

Based on the definition of $\mathbf{E}_j$ in (3), it can be proved that

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

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 $\sum_{n \in \mathcal{N}} \Big(\tilde{\mathbf{A}}_t[\rightarrow n] \Big)$ as $\mathbf{S}$, (7) can be rewritten as

$$\mathbf{M} \quad = \quad \mathbf{A}_s + \sum_{i=0}^{s} \Big(\sum_{n \in \mathcal{N}} \tilde{\mathbf{A}}_t[\rightarrow n] \Big)[\downarrow k_i] = \mathbf{A}_s + \mathbf{S} + \sum_{i=1}^{s} \Big(\mathbf{S}[\downarrow k_i] \Big). \tag{8}$$

Because each $\tilde{\mathbf{A}}_t[\rightarrow n]$ is an upper-triangular Toeplitz matrix, from [8], we easily know that matrix $\mathbf{S}$ is also an upper-triangular Toeplitz matrix, and computing

$$\mathbf{S}(1,:) = \sum_{n \in \mathcal{N}} \Big(\tilde{\mathbf{A}}_t(1,:)[\rightarrow n] \Big) = \sum_{n \in \mathcal{N}} \Big(\mathbf{A}_t(1,:)[\rightarrow n] \Big) \tag{9}$$

is sufficient to construct matrix $\mathbf{S}$ which has the following form

$$\mathbf{S} \quad = \quad \begin{bmatrix} 0 & s_{m-1} & \cdots & s_1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_{m-1} \\ 0 & 0 & \cdots & 0 \end{bmatrix}. \tag{10}$$

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. Recall that $n = 0$ is the least element in $\mathcal{N}$, we get the XOR complexity of computing (9) is

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

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 can use (8) to compute the product matrix $\mathbf{M}$. From (6) and (10), 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, (8) can be rewritten as

$$\mathbf{M} = \mathbf{T} + \mathbf{S}[\downarrow k_1] + \cdots + \mathbf{S}[\downarrow k_s]. \tag{11}$$

In order to compute (11) efficiently, we first introduce the following theorem:

Theorem 4.1 *Given an upper-triangular matrix* $\mathbf{S}$ *and* $m \times m$ *matrix* $\mathbf{P}$ *whose last* $(m-l)$ *rows form a Toeplitz sub-matrix. 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 sub-matrix.* ■

Applying Theorem 4.1, (11) can be computed using linear tree structure as follows:

Algorithm 4.1

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 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. To find the result $c(\alpha)$ via the product $\mathbf{c} = \mathbf{M} \cdot \mathbf{b}$, we also need m^2 AND gates and $m(m-1)$ XOR gates. The delay of this matrix-vector multiplication will be $T_A + \lceil \log_2 m \rceil T_X$ if binary tree structure is used.

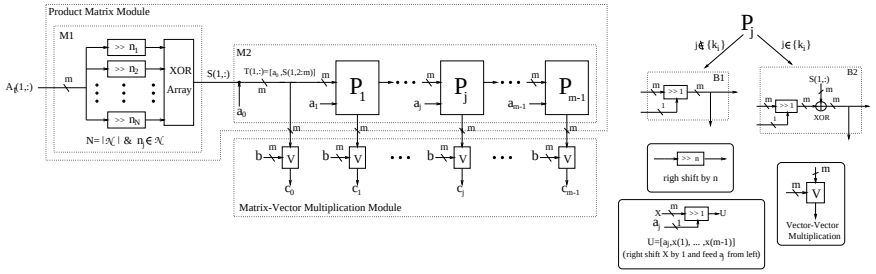

Figure 2: General Mastrovito Multiplier Architecture

Based on above computation approach, we develop the architecture of the dedicated Mastrovito multiplier as shown in Fig. 2. Set $\mathcal{N}$ is computed 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 $\sum_{n \in \mathcal{N}} \Big(\mathbf{A}_t(1, :)[\rightarrow n] \Big)$. Supplied with $\mathbf{S}(1, :)$, Block $M2$ computes the product matrix $\mathbf{M}$ using Algorithm 4.1. $M2$ contains $(m - 1)$ P_j blocks, each one generates one row vector of $\mathbf{M}$. If $j \in \{k_i, 1 \leq i \leq s\}$ then P_j is identical with block $B2$, otherwise it is identical with 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 complexities of the proposed Mastrovito multiplier for general underlying irreducible polynomial are given as:

- XOR Complexity: $(m + s - 1)(m - 1) + \sum_{n \in \mathcal{N}} (m - n - 1)$;
- AND Complexity: m^2;
- Delay: $T_A + \Big(s + \lceil \log_2 |\mathcal{N}| \rceil + \lceil \log_2 m \rceil \Big) T_X$.

It needs to be pointed out that for given irreducible polynomial, there likely exist some common items in the computation of (9) and (11). 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. Therefore, above XOR complexity value is actually an upper bound for general cases.

5 SPECIAL IRRIDUCIBLE POLYNOMIALS

For Mastrovito multiplier using special irreducible polynomial, such as trinomial and ESP, the corresponding $\mathcal{N}$ usually has a very simple or regular form which will lead to a low-complexity multiplier architecture. For example, applying Algorithm 3.1, we have

$$\mathcal{N} = \{l(m-n), 0 \leq l \leq \lfloor \frac{m-2}{m-n} \rfloor\}$$

if trinomial is used, and $\mathcal{N} = \{0, r\}$ if ESP is used. Based on the form of $\mathcal{N}$ and the design approach developed in last section, we can easily derive the same low-complexity multiplier architectures as proposed in [4] and [7]. Moreover, for those cases where neither an irreducible trinomial nor an irreducible ESP exists, using Algorithm 3.1 we can find some other special irreducible polynomials which will also lead to a low-complexity multiplier architecture. Below we briefly describe two such special irreducible polynomials as examples:

Example 5.1

For the irreducible polynomial $f(x) = x^m + x^{k_s} + \cdots + x^{k_1} + 1$, if $\lceil \frac{m}{2} \rceil \geq k_s$, applying Algorithm 3.1, we immediately have

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

and $|\mathcal{N}| = s+1$. Thus (9) can be simplified as

$$\mathbf{S}(1,:) \quad = \quad \mathbf{A}_t(1,:) + \sum_{i=1}^{s} \Big(\mathbf{A}_t(1,:)[\rightarrow (m-k_i)] \Big). \tag{12}$$

If we first compute (12) using linear tree structure and then compute matrix **M** *using Algorithm 4.1, we can prove that the XOR complexity of computing matrix* **M** *can be reduced significantly by sharing common items which are generated during the computation of (12), and the complexity of result multiplier will reduce to:*

- *XOR Complexity:* $(m+s)(m-1)$*;*
- *AND Complexity:* m^2*;*
- *Delay:* $T_A + \Big(2s + \lceil \log_2 m \rceil\Big) T_X$.

Example 5.2

For irreducible pentanomial $f(x) = x^m + x^{k_3} + x^{k_2} + x^{k_1} + 1$, *if* $m - k_3 = k_3 - k_2 = k_2 - k_1 = r$, *let* $\lfloor \frac{m-2}{r} \rfloor = d$, *the set* $\mathcal{N}$ *is obtained as:*

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

So (9) can be simplified as

$$\mathbf{S}(1,:) \quad = \quad \sum_{l=0}^{\lfloor \frac{d}{4} \rfloor} \left(\mathbf{g}^T [\rightarrow 4l \cdot r] \right) \tag{13}$$

where $\mathbf{g}^T = \mathbf{A}_t(1,:) + \mathbf{A}_t(1,:)[\rightarrow r]$ *and can be computed using* $(m - r - 1)$ *XOR gates with the delay of* $1T_X$. *Applying the result in [4], we can compute (13) using* $(m - 4r - 1)$ *XOR gates with the delay of* $\lfloor \frac{d}{4} \rfloor T_X$. *Furthermore, we need* $3(m-1)$ *XOR gates to compute* $\mathbf{M}$ *using Algorithm 4.1, with the delay of* $3T_X$. *Therefore, in this case, the total complexity of Mastrovito multiplier is*

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

6 CONCLUSIONS

In this paper, we present a systematic design approach of Mastrovito multiplier. The result architecture of Mastrovito multiplier for general irreducible polynomial is highly modular and requires m^2 AND gates and at most $\sum_{n \in \mathcal{N}} (m-n-1) - (m-1)$ XOR gates. It's noted that although the complexity results appear the same as those achieved in [7], the presented design uses an entirely different construction approach compared with [7], and more important, an explicit algorithm has been proposed to compute set $\mathcal{N}$ which makes our design really practicable. Meanwhile, the presented approach can be used for special irreducible polynomial cases, such as trinomial and ESP, and the complexity results can match the results achieved in [4] and [7]. We also use this approach to find two new special irreducible polynomials which will lead to low-complexity multipliers. In addition, with the explicit algorithms and design procedures, this proposed approach can be easily used in VLSI automation design tools for dedicated bit-parallel $GF(2^m)$ multiplier design.

References

[1] A. J. Menezes et. al, *Applications of Finite Fields,* Kluwer, 1993.

[2] R. Lidl and H. Niederreiter, *Finite Fields,* Cambridge University Press, 2 edition, 1997.

[3] E. D. Mastrovito, "VLSI designs for multiplication over finite fields GF(2^m)", *in Proc. 6th International Conference on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes (AAECC-6)*, pp. 297–309, Rome, July 1988.

[4] B. Sunar and Ç. K. Koç, "Mastrovito multiplier for all trinomials", *IEEE Trans. on Computers*, vol. 48, pp. 522–527, May 1999.

[5] 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. on Computers*, vol. 41, pp. 962–971, Aug. 1992.

[6] Ç. K. Koç and B. Sunar, "Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields", *IEEE Trans. on Computers*, vol. 47, pp. 353–356, March 1998.

[7] A. Halbutoğulları and Ç. K. Koç, "Mastrovito multiplier for general irreducible polynomials", *IEEE Trans. on Computers*, vol. 49, pp. 503–518, May 2000.

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 3rd edition, 1996.

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