# Polynomial Basis Multiplication over GF($2^m$)

Serdar S. Erdem[1], Tuğrul Yanık[2], Çetin K. Koç[3]
[1] *Elektronik Bölümü, Gebze Yüksek Teknoloji Enstitüsü, Turkey, serdem@gyte.edu.tr*

[2] *Computer Engineering Dep., Fatih University, Istanbul, Turkey, tyanik@fatih.edu.tr*

[3] *Information Security Research Center, Istanbul Commerce University, Istanbul, Turkey, koc@cryptocode.net*

April 15, 2006

**Abstract.** In this paper, we describe, analyze and compare various $GF(2^m)$ multipliers. Particularly, we investigate the standard modular multiplication, the Montgomery multiplication, and the matrix-vector multiplication techniques.

**Keywords:** Finite fields, binary fields, computer arithmetic, modular multiplication, modular reduction.

**Mathematics Subject Classification (2000):** 12-XX, 68-XX, 94-XX.

## 1. Introduction

Efficient implementations of the basic arithmetic operations in finite fields GF($2^m$) are desired for the applications of cryptography and coding theory [Menezes94, Menezes97].

The elements in GF($2^m$) can be represented in various bases. The choice of basis used to represent field elements has a significant impact on the performance of the field arithmetic. The multiplication methods that use polynomial basis representations are very efficient in comparison to the best methods for multiplication using the other basis representations.

In the literature, there is a large number of papers dealing with the hardware and the software implementations of the polynomial basis multiplication. Polynomial basis multiplication is based on two main arithmetic operations over the binary polynomials: polynomial multiplication and reduction modulo an irreducible polynomial.

An efficient bit parallel multiplier was first proposed by Mastrovito. In this work, the so-called Mastrovito matrix is constructed from the coefficients of the first multiplicand and the irreducible polynomial defining the field. Then, the polynomial multiplication and modulo reduction steps are performed together using this matrix. [Sunar1999] thoroughly studied the Mastrovito multiplier for the irreducible trino-

mials, which constitute a class of irreducible polynomials with low hamming weight. Irreducible polynomials with special structures and low hamming weights have been used in many papers [IItoh89, Hasan92, Wu98] to design efficient finite field multipliers. [Halbutoğulları00] has generalized the Mastrovito multiplier in [Sunar1999] for arbitrary irreducible polynomials. [Zhang01] proposed a practical and systematic design approach for this general Mastrovito multiplier.

[Wu02] showed that non-Mastrovito multipliers using direct modular reduction also give compatible performance, and proposed efficient non-Mastrovito multipliers for irreducible trinomials. [Reyhani04] proposes to use a reduction matrix to derive a new formulation for polynomial basis multiplication. A generalized architecture for the multiplier and optimizations for special irreducible polynomials are proposed.

In [Koç98], the Montgomery multiplication method in $GF(2^m)$ is proposed. The method works for an arbitrary irreducible polynomial. In [Wu01], the method in [Koç98] is optimized for irreducible trinomials.

The remainder of the paper is organized as follows: In Section 2, the representation and multiplication of GF($2^m$) elements in the polynomial basis is illustrated. In Section 3, we explain how the polynomials representing the field elements can be multiplied, and then reduced modulo an irreducible polynomial. We calculate the space and the time complexities of the standard polynomial multiplication, the modulo reduction operation, and the complete field multiplication for equally spaced irreducible polynomials and general irreducible polynomials with $r$ nonzero terms ($r$-nomials).

In Section 4, we explain how the field multiplications can be computed in two ways by using matrix vector operations. In the first method the polynomial multiplication is performed by any method. Then, the polynomial product is reduced with a reduction matrix. In the second method, the polynomial multiplication and modular reduction are performed in a single step by using the so-called Mastrovito Matrix. In Section 5, we explain the Montgomery multiplication method where the finite field is constructed with an arbitrary irreducible polynomial and a special case where the finite field is constructed with an irreducible trinomial. And in Section 6, we give a conclusion where we analyze and compare the strengths and weaknesses of the methods we mentioned in our previous sections.

## 2. Polynomial Basis Multiplication in GF($2^m$)

In this section, we illustrate the representation and multiplication of GF($2^m$) elements in the polynomial basis.

The finite field GF($2^m$) is an extension field of GF(2) and constitutes a dimension $m$ vector space over it. The finite field GF(2) has only the elements 0 and 1. In this binary field, the addition and the subtraction are defined as XOR operation while the multiplication is defined as AND operation. Let $x \in \mathrm{GF}(2^m)$ and be a root of the degree $m$ irreducible polynomial over GF(2)

$$\omega(x) = x^m + \omega_{m-1}x^{m-1} + \cdots \omega_1 x + \omega_0 = 0 \ . \tag{1}$$

Then, the following set constitutes the polynomial basis in GF($2^m$):

$$\left\{1, x, \ldots, x^{m-1}\right\} \ .$$

With polynomial basis, GF($2^m$) elements can be represented as degree $m-1$ polynomials as follows:

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

where the coefficients $a_i$ are the polynomial basis coordinates in GF(2).

When the elements of GF($2^m$) are represented as polynomials over GF(2), their addition and subtraction are equivalent to the coefficient-wise XOR, denoted by "+" in this paper. Also, because of the equation (1), all arithmetic operations in GF($2^m$) are performed modulo the irreducible polynomial $\omega(x)$ chosen to construct the field. Let $a(x)$ and $b(x)$ be two field elements and $c(x)$ be their product. Then,

$$c(x) = a(x)b(x) \mod \omega(x) \ . \tag{2}$$

Thus, the polynomial basis multiplication has two steps: polynomial multiplication and reduction modulo an irreducible polynomial.

### 2.1. Polynomial Multiplication

Let $d(x) = a(x)b(x)$ be the product of the polynomials representing the field elements. $d(x)$ is the degree $2m-2$ polynomial

$$d(x) = a(x)b(x) = \left(\sum_{i=0}^{m-1} a_i x^i\right)\left(\sum_{j=0}^{m-1} b_j x^j\right) = \sum_{k=0}^{2m-2} d_k x^k \ ,$$

where

$$d_k = \sum_{i+j=k} a_i b_j, \quad 0 \le i, j \le m-1, \quad 0 \le k \le 2m-2 \ .$$

## 2.2. Modular Reduction

In the modular reduction $c(x) = d(x) \mod \omega(x)$, the degree $2m-2$ polynomial $d(x)$ is reduced by the degree $m$ irreducible polynomial $\omega(x)$ iteratively. The partial remainder after each reduction can be computed by the following iteration:

$$\begin{aligned} &d^{(2m-2)}(x) = d(x) \ , \\ &d^{(k-1)}(x) = d^{(k)}(x) + \omega(x) d_k^{(k)} x^{k-m}, \qquad m \le k \le 2m-2 \ , \end{aligned} \tag{3}$$

Here, $d^{(k)}(x)$ is a partial remainder of degree $k$ and $d^{(m-1)}(x) = c(x)$. The iteration in (3) reduces $d^{(k)}(x)$ from degree $k$ to $k-1$, since adding (coefficientwise XORing) $d^{(k)}(x)$ with the polynomial

$$\begin{aligned} \omega(x) d_k^{(k)} x^{k-m} &= \left( x^m + \sum_{i=0}^{m-1} \omega_i x^i \right) d_k^{(k)} x^{k-m} \\ &= d_k^{(k)} x^k + \sum_{i=0}^{m-1} d_k^{(k)} \omega_i x^{i+k-m} \\ &= d_k^{(k)} x^k + \sum_{i=k-m}^{k-1} d_k^{(k)} \omega_{i-(k-m)} x^i \end{aligned}$$

cancels its term with the order $k$.

The choice of the irreducible polynomial $\omega(x)$ may ease the modular reduction. Sparse irreducible polynomials having fewer nonzero terms are usually preferred for efficiency. A degree $m$ irreducible polynomial over GF(2) which has $r$ nonzero terms are in the form

$$x^m + x^{m_1} + x^{m_2} + \cdots + x^{m_{r-3}} + x^{m_{r-2}} + 1 \ .$$

Here, $r > 1$ must be an odd number such as 3 and 5. The sparse polynomials with three or five nonzero terms as shown below are called trinomial and pentanomial respectively:

$$\begin{aligned} &x^m + x^{m_1} + 1, \\ &x^m + x^{m_1} + x^{m_2} + x^{m_3} + 1 \ . \end{aligned}$$

Equally spaced irreducible polynomials are another choice for efficient modular reduction. An equally spaced polynomial (ESP) is in the form

$$x^{ns} + x^{(n-1)s} + \cdots + x^s + 1 \ , \tag{4}$$

where $ns = m$.

## 3. Direct Multiplication and Reduction

Straightforwardly, the polynomials representing the field elements can be multiplied, and then reduced modulo an irreducible polynomial to compute their product in GF($2^m$). In this section, we calculate the space and the time complexities of the standard polynomial multiplication, the modulo reduction operation, and the complete field multiplication. We find the complexity of the modulo reduction operation for equally spaced irreducible polynomials and the general irreducible polynomials with $r$ nonzero terms ($r$-nomials). At the end of the section, we present the tabulated results of the complexity calculations.

### 3.1. Polynomial Multiplication

The following proposition and its corollary give the space and time complexities of the standard polynomial multiplication.

*PROPOSITION 1.* Let $a(x)$ and $b(x)$ be a degree $m-1$ polynomials. The computation $d(x) = a(x)b(x)$ with the standard polynomial multiplication requires

- $m^2$ coefficient multiplications (ANDs),
- $(m-1)^2$ coefficient additions (XORs).

Also, the time delay for the computation of the coefficient $d_k$ of $d(x)$ is $T_A + \lceil \log_2 \min(k+1, 2m-1-k) \rceil T_X$ where $T_A$ and $T_X$ are the time delays for the coefficient multiplication (AND) and the coefficient addition (XOR) operations respectively.

*COROLLARY 1.* The computation $d(x) = a(x)b(x)$ with the standard multiplication incurs the time complexity $T_A + \lceil \log_2 m \rceil T_X$.

*Proof 1.* In standard polynomial multiplication, each coefficient of $a(x)$ needs to be multiplied with each coefficient of $b(x)$. Thus, the number of the required coefficent multiplications is $m^2$.

The coefficients of $d(x) = a(x)b(x)$ are

$$\begin{aligned} d_k &= \textstyle\sum_{i=0}^{k} a_i b_{k-i}, && 0 \le k \le m-1 \ , \\ d_k &= \textstyle\sum_{i=k-m+1}^{m-1} a_i b_{k-i}, && m \le k \le 2m-2 \ . \end{aligned}$$

The number of the required additions above can be given as

$$\begin{aligned} k - 0 &= k, && 0 \le k \le m-1 \ , \\ m-1-(k-m+1) &= 2m-2-k, && m \le k \le 2m-2 \ . \end{aligned}$$

If an XOR tree is used for the addition, the time delay for the computation of $d_k = \sum_{i+j=k} a_i b_j$ will be

$$\begin{aligned} &T_A + \lceil \log_2(k+1) \rceil T_X, && 0 \le k \le m-1 \ , \\ &T_A + \lceil \log_2(2m-2-k+1) \rceil T_X, && m \le k \le 2m-2 \ . \end{aligned}$$

Then, the time delay needed to compute $d_k$ is

$$T_A + \lceil \log_2 \min(k+1, 2m-1-k) \rceil T_X, \qquad 0 \le k \le 2m-2 \ .$$

Also, the total number of coefficent additions is

$$\begin{aligned} \sum_{k=0}^{m-1} k + \sum_{k=m}^{2m-2} 2m-2-k &= \sum_{k=0}^{m-1} k + \sum_{k=0}^{m-2} k \\ &= m(m-1)/2 + (m-1)(m-2)/2 \\ &= (m-1)^2 \ . \end{aligned}$$

□

### 3.2. Modular Reduction

The modular reduction can be performed by the iteration in (3) for any irreducible polynomial. However, if one uses an equally spaced irreducible polynomial or takes the advantage of the zero terms of a general irreducible polynomial, one can employ a more efficient reduction scheme.

#### 3.2.1. *Equally Spaced Irreducible Polynomials*

Because the terms of this kind of polynomials are equally spaced, many term cancellations occur in the modular reduction, leading a decrease in the space and time complexities. The following proposition and its proof illustrate the reduction with equally spaced irreducible polynomials.

*PROPOSITION 2.* Let $d(x)$ be a degree $2m-2$ polynomial and $\omega(x)$ be a degree $m = ns$ equally spaced polynomial given by (4). Then, the computation $d(x) \bmod \omega(x)$ requires $2m - s - 1$ coefficient additions (XORs) and incurs the delay $2T_X$ where $T_X$ is the time delay for the coefficient addition.

*Proof 2.* One can easily verify that

$$x^{m+s} \bmod \omega(x) = x^{ns+s} \bmod \omega(x) = 1$$

for an irreducible polynomial $\omega(x)$ given by (4). Thus, the term $d_i x^i$ of the $d(x)$ can be reduced to the term $d_i x^{i-m-s}$ where $i - m - s \ge 0$. Then, the first part of the modular reduction is to compute

$$d_{i-m-s} = d_{i-m-s} + d_i \qquad i = m+s, m+s+1, \ldots, 2m-2 \ .$$

This part takes $m-s-1$ coefficient additions and one $T_X$ delay. The second part is to reduce the terms with the order $i+m=i+ns$ for $i=m,m+1,\ldots,m+s$ as follows:

$$d_{i+js}=d_{i+js}+d_{i+ns} \qquad j=0,1,\ldots,n-1\ .$$

This part takes $m$ coefficient additions and one $T_X$ delay. The total coefficient additions are $2m-s-1$. Two parts of the modular reduction can be performed parallel. Thus, the total delay of $2T_X$ is required to compute the two parts and combine their results. □

3.2.2. *Polynomials with r Nonzero Terms (r-nomials)*
Exploiting the zero terms of the polynomial used in reduction, one can simplify the iteration equation in (3). Let an irreducible polynomial $\omega(x)$ with $r$ nonzero terms be given by

$$\omega(x)=\left(x^m+\sum_{j=1}^{r-1}x^{m_j}\right)\ , \tag{5}$$

where $m>m_1>m_2>\cdots>m_{r-2}>m_{r-1}=0$. Then, the iteration equation in (3) can be modified as follows:

$$\begin{aligned} &d^{(2m-2)}(x)=d(x),\\ &d^{(l)}(x)=d^{(k)}(x)+\omega(x)\sum_{i=l+1}^{k}d_i^{(k)}x^{i-m}, \qquad m\le k\le 2m-2\ , \end{aligned} \tag{6}$$

where $k>l\ge k-(m-m_1)$ and $l\ge m-1$. The iteration in (6) reduces $d^{(k)}(x)$ from degree $k$ to $l$, because adding (coefficientwise XORing) $d^{(k)}(x)$ with the polynomial

$$\begin{aligned} \omega(x)\sum_{i=l+1}^{k}d_i^{(k)}x^{i-m} &= \left(x^m+\sum_{j=1}^{r-1}x^{m_j}\right)\sum_{i=l+1}^{k}d_i^{(k)}x^{i-m}\\ &=\sum_{i=l+1}^{k}d_i^{(k)}x^{i}+\sum_{j=1}^{r-1}x^{m_j}\sum_{i=l+1}^{k}d_i^{(k)}x^{i-m}\\ &=\sum_{i=l+1}^{k}d_i^{(k)}x^{i}+\sum_{j=1}^{r-1}\sum_{i=l+1}^{k}d_i^{(k)}x^{i-(m-m_j)} \end{aligned} \tag{7}$$

cancels its terms with the orders $k,k-1,\ldots,l+2,l+1$ as long as $l\ge k-(m-m_1)\ge k-(m-m_j)$.

Let $\iota$ be the minimum number of iterations required to reduce $d(x)$ to degree $m-1$. Then, $2m-2-\iota\max(k-l)=2m-2-\iota(m-m_1)<m$. As a result,

$$\iota=\left\lfloor\frac{m-2}{m-m_1}\right\rfloor+1\ . \tag{8}$$

Now, we give the upper bounds for the space and the time complexities of the modular reduction by the following propositions:

*PROPOSITION 3.* Let $\omega(x)$ be a degree $m$ binary polynomial with $r$ nonzero coefficients. Also, let $d(x)$ be a degree $n$ and $d'(x) \equiv d(x) \bmod \omega(x)$ be a degree $n'$ polynomial where $m-1 \leq n' < n$. Computing $d'(x)$ from $d(x)$ requires at most $(r-1)(n-n')$ coefficient additions (XORs).

*COROLLARY 2.* [Wu02] Let $d(x)$ be a degree $2m-2$ polynomial. The computation $d(x) \bmod \omega(x)$ requires at most $(r-1)(m-1)$ coefficient additions (XORs).

*Proof 3.* Equation (7) is written for a polynomial $\omega(x)$ with $r$ binary coefficients. If the reduction iteration in (6) is rewritten by using (7),

$$\begin{aligned} d^{(l)}(x) &= d^{(k)}(x) + \sum_{i=l+1}^{k} d_i^{(k)} x^i + \sum_{j=1}^{r-1} \sum_{i=l+1}^{k} d_i^{(k)} x^{i-(m-m_j)} \\ &= \sum_{i=0}^{l} d_i^{(k)} x^i + \sum_{j=1}^{r-1} \sum_{i=l+1}^{k} d_i^{(k)} x^{i-(m-m_j)} \ , \end{aligned}$$

where $d^{(k)}(x)$ denote the degree $k$ polynomial obtained by reducing $d(x)$ modulo $\omega(x)$. It is easy to see that at most $(k-l)(r-1)$ coefficient additions are required to find $d^{(l)}(x)$ from the coefficients of $d^{(k)}(x)$. Let $k_0, k_1, \ldots, k_\iota$ be the integers such that $k_0 = n$, $k_\iota = n'$, and $k_i > k_{i+1} \geq k_i - (m - m_1)$ for $k < \iota$. Also, let

$$d^{(k_i)}(x) \equiv d(x) \bmod \omega(x)$$

denote a partial remainder of $d(x)$ with degree $k_i$. Then, $d^{(k_0)}(x) = d(x)$, $d^{(k_\iota)}(x) = d'(x)$, and $d^{(k_{i+1})}(x)$ can be computed from $d^{(k_i)}(x)$ by (6). Thus, $d'(x)$ can be found by computing $d^{(k_i)}(x)$ for $i = 1, 2, \ldots, \iota$ iteratively. The maximum number of the coefficient additions for this computation can be given as

$$\sum_{i=0}^{\iota-1} (k_i - k_{i+1})(r-1) = (k_0 - k_\iota)(r-1) = (n-n')(r-1) \ .$$

□

*PROPOSITION 4.* Let $d(x)$ be a degree $2m-2$ polynomial and $\omega(x)$ be a degree $m$ binary polynomial with $r$ nonzero coefficients. Also, let the order of the nonzero terms of $\omega(x)$ satisfy that $m_1 > m_2 >$

$\ldots > m_{r-2} > m_{r-1} = 0$ as in (5). Then, an upper bound for the time complexity of the computation $d(x) \bmod \omega(x)$ can be given by

$$\left( \left\lfloor \frac{m-2}{m-m_1} \right\rfloor + 1 \right) \lceil \log_2(|U|+1) \rceil T_X \ ,$$

where $U \subseteq \{1, 2, \ldots, r-1\}$ is the largest set or one of the largest sets in the form $\{u, v | m - m_1 > |m_u - m_v|\}$ and $T_X$ is the time delay for the coefficient addition (XOR).

Also, the definition of $U$ implies that it is a set of consecutive integers, i.e. $U = \{u_{min}, u_{min}+1, \ldots, u_{max}-1, u_{max}\}$ where $u_{\min}$ and $u_{\max}$ are the minimum and maximum elements of $U$ respectively.

*COROLLARY 3.* The set $U$ for some special cases is as follows:

- $U = \{1, 2, \ldots, r-1\}$, when $m_1 < \frac{m}{2}$.
- $U = \{1, 2, \ldots, r-2\}$ or $U = \{2, 3, \ldots, r-1\}$, when $m_1 = \frac{m}{2}$.
- $U = \{1\}$ or $U = \{2\}$ or $\ldots$ or $U = \{r-1\}$, when $m_1 = m-1$.

Then, we have the following:

- If $m_1 < \frac{m}{2}$, $|U| = r-1$ and the delay is at most $2\lceil \log_2 r \rceil T_X$.
- If $m_1 = \frac{m}{2}$, $|U| = r-2$ and the delay is at most $2\lceil \log_2 (r-1) \rceil T_X$.
- If $m_1 = m-1$, $|U| = 1$ and the delay is at most $(m-1)T_X$.

*Proof 4.* As can be understood from Proof 3, the reduction iteration in (6) can be given as follows.

$$d^{(l)}(x) = \sum_{i=0}^{l} d_i^{(k)} x^i + \sum_{j=1}^{r-1} \sum_{i=l+1}^{k} d_i^{(k)} x^{i-(m-m_j)} \ ,$$

where $\omega(x)$ is a degree $m$ polynomial with the $r$ binary coefficients as in (5). To compute $d^{(l)}(x)$, we need to compute

$$h(x) = \sum_{j=1}^{r-1} \sum_{i=l+1}^{k} d_i^{(k)} x^{i-(m-m_j)} \ .$$

Note that the $n$th coefficient of $h(x)$ can be given by

$$h_n = \sum_{\forall j \in J(n)} d_{n+(m-m_j)}^{(k)} \ ,$$

where $J(n) = \{j | k \geq n + (m - m_j) \geq l + 1\}$. Note that the definition of the set $J(n)$ implies that $k - l > |m_{j1} - m_{j2}|$ for $\forall j1, j2 \in J(n)$. Let $U$ be the largest or one of the largest sets in the form $\{u, v | k - l > |m_u - m_v|\}$. Then, $|J(n)| \leq |U|$. Let $u_{\min}$ and $u_{\max}$ be the minimum and maximum elements of $U$ respectively. Because $m_j > m_{j+1}$,

$$|m_{u_{\max}} - m_{u_{\min}}| \geq |m_u - m_v|$$

for any $u, v$ pair such that $u_{\max} \geq u > v \geq u_{\min}$. Thus, $U$ is a consecutive set of integers and $|U| = u_{\max} - u_{\min} + 1$. Since the reduction iteration in (6) is defined for $l \geq k - (m - m_1)$, we can set $k - l = m - m_1$ and redefine the form of $U$ as follows:

$$\{u, v | m - m_1 > |m_u - m_v|\}.$$

Note that $|J(n)|$ coefficient additions are required to compute $h_n$. Also, $h(x)$ must be added with another polynomial to find $d^{(l)}(x)$. Thus, the number of the coefficient additions to compute the $n$th coefficient of $d^{(l)}(x)$ are at most $|J(n)| + 1 \leq |U| + 1$. Then, the time complexity of a single iteration is below $\lceil \log_2 (|U| + 1) \rceil T_X$. This must be multiplied by the number of iterations given by (8) to find upper bound for a complete modular reduction. □

*PROPOSITION 5.* Let $d(x)$ be of degree $2m - 2$. The upper bound for the time complexity of the computation $d(x) \bmod \omega(x)$ equals to $2\lceil \log_2(r - 1) \rceil T_X$ where $\omega(x)$ is given by (5) and $m/2 \geq m_j$ for $\forall j$.

*Proof 5.* Corollary 3 gives the upper bound for the cases $m_1 = m/2$ and $m_1 < m/2$ as $2 \lceil \log_2 (r - 1) \rceil T_X$ and $2 \lceil \log_2 r \rceil T_X$ respectively.

However, the time complexity for the case $m_1 < m/2$ can be as low as $2 \lceil \log_2 (r - 1) \rceil T_X$. When $m_1 < m/2$, the iteration in (6) must be used two times as can be calculated from (8). The degree $2m - 2$ polynomial $d(x)$ is reduced into first a degree $2m - 2 - (m - m_1) = m + m_1 - 2$ polynomial, then a degree $m - 1$ polynomial. As can be understood from Proof 4, if a degree $k$ polynomial is reduced to a degree $l$ polynomial, at most $|J(n)| + 1$ coefficient additions are needed to compute the $n$th coefficient of the reduced polynomial where $J(n) = \{j | k \geq n + m - m_j \geq l + 1\}$.

In the first reduction, $J(n) = \{j | 2m - 2 \geq n + m - m_j \geq m + m_1 - 1\}$. Note that, if $n \geq m$, $m_j \geq 2$. Because $m_{r-1} = 0$, $J(n)$ can include only the integers $1, 2, \ldots, r - 2$ but not $r - 1$. Thus, $|J(n)| \leq r - 2$ for $n \geq m$. Then, the time complexity of the computation of the terms with the order $n \geq m$ in the first reduction is at most $\lceil \log_2 (|J(n)| + 1) \rceil T_X = \lceil \log_2 (r - 1) \rceil T_X$. Since the terms with the order $n < m$ do not need to

be reduced, the second reduction can start before their computation. Thus, they do not contribute the time complexity.

In the second reduction, $J(n) = \{j | m + m_1 - 2 \geq n + m - m_j \geq m\}$. Then, $J(n) = \{j | m_1 - 2 \geq n - m_j \geq 0\} = \{u, v | m_1 - 2 \geq |m_u - m_v|\}$. Note that $J(n) \neq \{1, 2, \ldots, r-1\}$. $J(n)$ must be smaller and $|J(n)| \leq r - 2$ Then, the time complexity of the second reduction is at most $\lceil \log_2 (|J(n)| + 1) \rceil T_X = \lceil \log_2 (r-1) \rceil T_X$. As a result the overall time complexity for the case $m_1 < m/2$ is at most $2 \lceil \log_2 (r-1) \rceil T_X$. □

### 3.3. Complexity of Field Multiplication

As mentioned before, the field multiplication consists of the polynomial multiplication and the modular reduction. The space and time complexities of the polynomial multiplication given by Proposition 1 and Corollary 1 are tabulated in Table I.

Table I. The complexity of the polynomial multiplication.

| Coefficient Multiplication (AND) | Coefficient Addition (XOR) | Time Complexity |
|---|---|---|
| $m^2$ | $(m-1)^2$ | $T_A + \lceil \log_2 m \rceil T_X$ |

The number of coefficient additions required for modular reduction is given in Proposition 2 and Corollary 2 for different types of irreducible polynomials. Table II tabulates these results in the first column. Table II also gives the number of the coefficient additions required for the field multiplication in the second column, adding the entries of the first column with $(m-1)^2$, the number of coefficient additions required for the polynomial multiplication.

Table II. The number of the coefficient additions (XORs)

| Irreducible Polynomials | Modular Reduction | Field Multiplication |
|---|---|---|
| Equally Spaced | $2m - s - 1$ | $m^2 - s$ |
| Trinomial | $2m - 2$ | $m^2 - 1$ |
| Pentanomial | $4m - 4$ | $m^2 + 2m - 3$ |
| $r$-nomial | $(m-1)(r-1)$ | $(m-1)(m+r-2)$ |

Table III tabulates the upper bounds of the modular reduction delay for different types of irreducible polynomials, given in Proposition 2 and Proposition 5. Also, this table presents the total field delay. However,

Table III. The time complexity

| Irreducible Polynomial | Modular Reduction | Field Multiplication |
|---|---|---|
| Equally Spaced | $2T_X$ | $T_A + (\lceil \log_2 m \rceil + 2)\, T_X$ |
| Trinomial* | $2T_X$ | $T_A + (\lceil \log_2 m \rceil + 2)\, T_X$ |
| Pentanomial* | $4T_X$ | $T_A + (\lceil \log_2 m \rceil + 4)\, T_X$ |
| $r$-nomial* | $2\lceil \log_2(r-1) \rceil\, T_X$ | $T_A + (\lceil \log_2 m \rceil + 2\lceil \log_2(r-1) \rceil)\, T_X$ |

*Assuming the order of the nonzero terms $m_j \leq m/2$, except the term $x^m$

the time complexities for trinomial, pentanomial, and $r$-nomials are valid, if these polynomials have no term with an order larger than $m/2$ except the term with the order $m$. Note that this is not a severe constraint in practice.

The time complexity of the field multiplication can be further reduced by cleverly combining the polynomial multiplication and modular reduction operations as shown in [Wu02]. However, the decrease in the delay will be insignificant for any practical application.

## 4. Matrix Vector Product Techniques

The GF($2^m$) multiplication given by (2) can be described in terms of matrix-vector operations. There are mainly two different approaches based on matrix vector operations to compute a field product:

1. The polynomial multiplication part is performed by any method. Then, the resulting product is reduced by using a reduction matrix.
2. The polynomial multiplication and modular reduction parts are performed in a single step by using the so-called Mastrovito matrix.

Let $a(x)$ and $b(x)$ denote two degree $m$ polynomials representing the elements in GF($2^m$). Let $c(x) = a(x)b(x) \bmod \omega(x)$ denote their field product. The coefficient vectors of these polynomials are given by

$$\mathbf{a} = [a_0, a_1, \cdots, a_{m-1}]^T$$
$$\mathbf{b} = [b_0, b_1, \cdots, b_{m-1}]^T$$
$$\mathbf{c} = [c_0, c_1, \cdots, c_{m-1}]^T.$$

Also, let us define the polynomials

$$\begin{aligned} d(x) &= a(x)b(x) = d_0 + d_1 x + \cdots + d_{2m-2} x^{2m-2} \ , \\ d^{(L)}(x) &= d_0 + d_1 x + \cdots + d_{m-1} x^{m-1} \ , \\ d^{(H)}(x) &= d_m + d_{m+1} x + \cdots + d_{2m-2} x^{m-2} \ . \end{aligned} \tag{9}$$

The coefficient vectors representing these polynomials are

$$\begin{aligned} \mathbf{d} &= [d_0, d_1, \cdots, d_{2m-2}]^T \ , \\ \mathbf{d}^{(L)} &= [d_0, d_1, \cdots, d_{m-1}]^T \ , \\ \mathbf{d}^{(H)} &= [d_m, d_{m+1}, \cdots, d_{2m-2}]^T \ . \end{aligned}$$

The work in [Reyhani04] reduces the polynomial multiplication $d(x)$ using an $(m \times m-1)$ reduction matrix $\mathbf{Q}$ to obtain the field product $c(x)$ as below:

$$\mathbf{c} = \mathbf{d}^{(L)} + \mathbf{Q} \cdot \mathbf{d}^{(H)} \ . \tag{10}$$

The papers [Sunar1999,Halbutoğulları00,Zhang01] follow the Mastrovito multiplication scheme below:

$$\mathbf{c} = \mathbf{M} \cdot \mathbf{b} \ , \tag{11}$$

where $\mathbf{M}$ is the $(m \times m)$ Mastrovito matrix whose entries are the function of the coefficients of $a(x)$ and $\omega(x)$. The Mastrovito matrix $\mathbf{M}$ is related to the reduction matrix $\mathbf{Q}$ by

$$\mathbf{M} = \mathbf{L} + \mathbf{Q} \cdot \mathbf{U} \ , \tag{12}$$

where $\mathbf{L}$ and $\mathbf{U}$ are the following $(m \times m)$ and $(m-1 \times m)$ matrices:

$$\begin{aligned} \mathbf{L} &= \begin{bmatrix} a_0 & 0 & 0 & \cdots & 0 & 0 \\ a_1 & a_0 & 0 & \cdots & 0 & 0 \\ a_2 & a_1 & a_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m-2} & a_{m-3} & a_{m-4} & \cdots & a_0 & 0 \\ a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_1 & a_0 \end{bmatrix} \ , \\ \mathbf{U} &= \begin{bmatrix} 0 & a_{m-1} & a_{m-2} & \cdots & a_2 & a_1 \\ 0 & 0 & a_{m-1} & \cdots & a_3 & a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{m-1} & a_{m-2} \\ 0 & 0 & 0 & \cdots & 0 & a_{m-1} \end{bmatrix} \ . \end{aligned} \tag{13}$$

This is because $d(x) = a(x)b(x)$ can be given in the vector notation by

$$\mathbf{d} = \begin{bmatrix} \mathbf{d}^{(L)} \\ \mathbf{d}^{(H)} \end{bmatrix} = \begin{bmatrix} \mathbf{L} \cdot \mathbf{b} \\ \mathbf{U} \cdot \mathbf{b} \end{bmatrix} .$$

Then, $\mathbf{c} = \mathbf{d}^{(L)} + \mathbf{Q} \cdot \mathbf{d}^{(H)} = \mathbf{L} \cdot \mathbf{b} + \mathbf{Q} \cdot \mathbf{U} \cdot \mathbf{b} = (\mathbf{L} + \mathbf{Q} \cdot \mathbf{U}) \cdot \mathbf{b} = \mathbf{M} \cdot \mathbf{b}$.

The Mastrovito and the reduction matrices are studied thoroughly in [Reyhani04] and [Zhang01] for various types of irreducible polynomials. Before investigating these matrices, we give the common notation used in their analysis.

*NOTATION 1.* The down shift of a vector by $i$ elements, the down shift of a matrix by $i$ rows, and the right shift of a matrix by $i$ columns are denoted as follows respectively:

$$\begin{aligned}
\mathbf{a}[\downarrow i] &= [\underbrace{0, \ldots, 0}_{i}, a_0, a_1, \ldots, a_{m-1-i}]^T , \\
\mathbf{A}[\downarrow i] &= [\underbrace{\mathbf{0}, \ldots, \mathbf{0}}_{i}, \mathbf{Arow}_0, \mathbf{Arow}_1, \ldots, \mathbf{Arow}_{m-1-i}]^T , \\
\mathbf{A}[\rightarrow i] &= [\underbrace{\mathbf{0}, \ldots, \mathbf{0}}_{i}, \mathbf{Acol}_0, \mathbf{Acol}_1, \ldots, \mathbf{Acol}_{n-1-i}] .
\end{aligned}$$

Here, $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{Arow}_i$ is its $i$th row, and $\mathbf{Acol}_i$ is its $i$th column. After the shifts, the emptied positions are filled by zeros.

### 4.1. Reduction Matrix

The reduction matrix can be defined in terms of the irreducible polynomial $\omega(x)$ used to construct the field. Let $\omega_i$ and $q_{i,j}$ denote the coefficients of $\omega(x)$ and the entries of the reduction matrix $\mathbf{Q}$ respectively. Also, let $\mathbf{q}_j = [q_{0,j}, q_{1,j}, \ldots, q_{m-1,j}]^T$ denote the column $j$ of the reduction matrix $\mathbf{Q}$. Then,

$$\mathbf{q}_j = \begin{cases} \boldsymbol{\omega} & j = 0 , \\ \mathbf{q}_{j-1}[\downarrow 1] + q_{m-1,j-1}\boldsymbol{\omega} & j = 1, \ldots, m-2 , \end{cases} \tag{14}$$

where $\boldsymbol{\omega} = [1, \omega_1, \ldots, \omega_{m-1}]^T$. In polynomial notation, the column $\mathbf{q}_j$ corresponds $q_j(x) = q_{0,j} + q_{1,j}x + \ldots + q_{m-1,j}x^{m-1}$ and Equation (14) is equivalent to

$$q_j(x) = \begin{cases} x^m \bmod \omega(x) & j = 0 , \\ xq_{j-1}(x) \bmod \omega(x) & j = 1, \ldots, m-2 , \end{cases}$$

i.e., $q_j(x) = x^{m+j} \mod \omega(x)$. Equation (14) can be proven by analyzing the modular reduction

$$\begin{aligned} c(x) &= d(x) \mod \omega(x) \\ &= \textstyle\sum_{j=0}^{m-1} d_j x^j + \sum_{j=0}^{m-2} d_{j+m} \left(x^{j+m} \mod \omega(x)\right) \ , \\ &= d^{(L)}(x) + \textstyle\sum_{j=0}^{m-2} d_j^{(H)} q_j\left(x\right) \end{aligned}$$

where $d^{(L)}(x)$ and $d^{(H)}(x)$ are as given in (9). In vector notation, this is equivalent to the reduction equation in (10) as seen below:

$$\mathbf{c} = \mathbf{d}^{(L)} + \sum_{j=0}^{m-2} d_j^{(H)} \mathbf{q}_j = \mathbf{d}^{(L)} + \mathbf{Q} \cdot \mathbf{d}^{(H)} \ .$$

The general form of the reduction matrix is given in [Zhang01] as

$$\mathbf{Q} = \sum_{n \in \mathcal{N}} \hat{\mathbf{Q}}[\rightarrow n] \ , \tag{15}$$

where $\hat{\mathbf{Q}} = [\boldsymbol{\omega}, \boldsymbol{\omega}[\downarrow 1], \ldots, \boldsymbol{\omega}[\downarrow m-2]]$ and $\mathcal{N} \subset \{0, 1, \ldots, m-2\}$ is an appropriate set. This result is predictable from Equation (14). This equation tells that $\mathbf{q}_0 = \boldsymbol{\omega}$ and $\mathbf{q}_{j>0} = \mathbf{q}_{j-1}[\downarrow 1] + \alpha\boldsymbol{\omega}$ where $\alpha$ is either 1 or 0. Let $\mathbf{q}_n = \mathbf{q}_{n-1}[\downarrow 1] + \boldsymbol{\omega}$ for some $n > 0$. Then, $\mathbf{q}_{n+j}$ contains the additive term $\boldsymbol{\omega}[\downarrow j]$ for $j \geq 0$, and thus $\mathbf{Q}$ contains the additive term $\hat{\mathbf{Q}}[\rightarrow n]$.

Let the field GF($2^m$) be constructed with a degree $m$ binary polynomial $\omega(x)$ with $r$ nonzero terms as in (5). Let $m_k$ for $k = 1, 2, \ldots, r-1$ denote the orders of its nonzero terms such that $m > m_1 > m_2 > \cdots > m_{r-2} > m_{r-1} = 0$. Then,

$$\hat{\mathbf{Q}} = \sum_{k=1}^{r-1} \mathbf{I}_{m\times(m-1)}[\downarrow m_k] \ , \tag{16}$$

where

$$\mathbf{I}_{m\times(m-1)} = \begin{bmatrix} \mathbf{I}_{(m-1)\times(m-1)} \\ \mathbf{0}_{1\times(m-1)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{bmatrix} \ .$$

This is because the matrix $\hat{\mathbf{Q}} = [\boldsymbol{\omega}, \boldsymbol{\omega}[\downarrow 1], \ldots, \boldsymbol{\omega}[\downarrow m-2]]$ where $\boldsymbol{\omega}$ is the coefficient vector of the irreducible polynomial $\omega(x)$. Note that if $i = m_k$, the coefficients $\omega_i = 1$. Otherwise, they are zero. Each nonzero $\omega_i$ generates $\mathbf{I}_{m\times(m-1)}[\downarrow i]$. Thus, $\hat{\mathbf{Q}}$ is a sum of the terms in the form $\mathbf{I}_{m\times(m-1)}[\downarrow m_k]$.

Using (15) and (16), the reduction equation (10) can be written as

$$\begin{aligned}\mathbf{c} &= \mathbf{d}^{(L)} + \textstyle\sum_{n\in\mathcal{N}} \hat{\mathbf{Q}}[\rightarrow n]\cdot\mathbf{d}^{(H)} \\ &= \mathbf{d}^{(L)} + \textstyle\sum_{n\in\mathcal{N}}\sum_{k=1}^{r-1} \mathbf{I}_{m\times(m-1)}[\downarrow m_k][\rightarrow n]\cdot\mathbf{d}^{(H)}.\end{aligned}$$

Then, the reduction equation (10) becomes

$$\mathbf{c} = \mathbf{d}^{(L)} + \sum_{k=1}^{r-1} \mathbf{s}[\downarrow m_k] \ , \tag{17}$$

where

$$\mathbf{s} = \sum_{n\in\mathcal{N}} \mathbf{I}_{m\times(m-1)}[\rightarrow n]\mathbf{d}^{(H)} \ . \tag{18}$$

Up to this point, we illustrate the reduction matrix method. Now, we will find out its complexity with a series of proposition.

*PROPOSITION 6.* Let $\omega(x)$ be a degree $m$ binary polynomial with $r$ nonzero terms. Let $m_k$ for $k = 1, 2, \ldots, r-1$ denote the orders of the nonzero terms except the term $x^m$. Then, if $(m+1)/2 \geq m_k$, the set used to construct the reduction matrix for $\omega(x)$ as shown in (15) is

$$\mathcal{N} = \{0\} \cup \{m - m_k : \ 1 \leq k \leq r-2\}.$$

*Proof 6.* See Equation (14). If the reduction matrix entry in the last row $q_{m-1,j-1} = 1$, $\boldsymbol{\omega}$ is added to column $j$ of the reduction matrix $\mathbf{Q}$. Note that the set $\mathcal{N}$ consists of the columns to which $\boldsymbol{\omega}$ is added because each of the copies of $\boldsymbol{\omega}$ vector will generate a shifted copy of $\hat{\mathbf{Q}}$. Then, $\mathcal{N} \supset \{0\}$ because $\mathbf{q}_0 = \boldsymbol{\omega}$ according to (14). Thus,

$$\mathbf{Q} = \hat{\mathbf{Q}} + \sum_{n\in\mathcal{N}-\{0\}} \hat{\mathbf{Q}}[\rightarrow n] \ .$$

Note that, due to the definition in (16), the last row entries $\hat{q}_{m-1,j-1} = 1$ where $j \in \{m - m_k : \ 1 \leq k \leq r-2\}$. Because $\mathbf{Q}$ contains a copy of $\hat{\mathbf{Q}}$, $q_{m-1,j-1} = 1$ where $j \in \{m - m_k : \ 1 \leq k \leq r-2\}$. As a result,

$$\mathbf{Q} = \hat{\mathbf{Q}} + \sum_{k=1}^{r-2} \hat{\mathbf{Q}}[\rightarrow (m - m_k)]$$

and $\mathcal{N} \supset \{m - m_k : \ 1 \leq k \leq r-2\}$. Note that $(m+1)/2 \geq m_k$ implies $2(m-m_k) \geq m-1 > m-2$. Thus, $\hat{\mathbf{Q}}[\rightarrow (m-m_k)]$ does not contribute the nonzero entries in the last row of the reduction matrix and the set $\mathcal{N}$ only includes $\{0\} \cup \{m - m_k : \ 1 \leq k \leq r-2\}$.

□

*PROPOSITION 7.* Let $\omega(x)$ be a degree $m$ binary polynomial with $r$ nonzero terms. Let $m_k$ for $k = 1, 2, \ldots, r-1$ denote the orders of the nonzero terms except the term $x^m$. Then, if $(m+1)/2 \geq m_k$, the reduction matrix method described by (17) and (18) incurs

- $(r-1)(m-1)+1$ coefficient additions, and
- $\lceil \log_2 r \rceil + \lceil \log_2 (r-1) \rceil$ delay.

*Proof 7.* $\mathcal{N} = \{0\} \cup \{m - m_k : 1 \leq k \leq r-2\}$ because $(m+1)/2 \geq m_k$. Then, (17) becomes

$$\begin{aligned} \mathbf{s} &= \textstyle\sum_{k=1}^{r-1} \mathbf{I}_{m\times(m-1)}[\rightarrow (m-m_k)]\mathbf{d}^{(H)} \\ &= \textstyle\sum_{k=1}^{r-1} \left[\mathbf{d}^{(H)}[\uparrow (m-m_k)], 0\right]^T \end{aligned} \quad ,$$

where $[\uparrow (m-m_k)]$ denotes an up shift by $(m-m_k)$. Then, the number of coefficient additions needed for this computation is

$$\sum_{k=1}^{r-2} (m - (m-m_k) - 1) = -r + 2 + \sum_{k=1}^{r-2} m_k \ .$$

The associated time delay is $\lceil \log_2 (r-1) \rceil$. On the other hand, the number of coefficient additions needed for (18) is

$$\sum_{k=1}^{r-1} (m - m_k) = (r-1)\, m - \sum_{k=1}^{r-1} m_k = (r-1)\, m - \sum_{k=1}^{r-2} m_k \ .$$

The associated time delay is $\lceil \log_2 r \rceil$. The total number of the coefficient additions is $(r-1)(m-1)+1$ and the total delay is $\lceil \log_2 r \rceil + \lceil \log_2 (r-1) \rceil$.

□

*PROPOSITION 8.* Let $\omega(x)$ be an equally spaced irreducible polynomial as in (4) where $ns = m$. Then, the reduction matrix is

$$\mathbf{Q} = \begin{bmatrix} \mathbf{I}_{s\times s} & \mathbf{I}_{(m-s-1)\times(m-s-1)} \\ \mathbf{I}_{s\times s} & \\ \vdots & \\ \mathbf{I}_{s\times s} & \mathbf{0}_{(s+1)\times(m-s-1)} \end{bmatrix}$$

and the reduction matrix method described by (10) incurs

- $2m - s - 1$ coefficient additions, and

– $2T_X$ delay.

*Proof 8.* For an equally spaced polynomial, the order of the nonzero terms $m_k = ks$ where $k = 0, 1, \ldots n-1$. Thus, Equation (16) becomes

$$\hat{\mathbf{Q}} = \sum_{k=1}^{n-1} \mathbf{I}_{m\times(m-1)}[\downarrow ks] \ .$$

Because $\mathbf{q}_0 = \boldsymbol{\omega}$ according to (14), $\mathcal{N} \supset \{0\}$. Thus,

$$\mathbf{Q} = \hat{\mathbf{Q}} + \sum_{n\in\mathcal{N}-\{0\}} \hat{\mathbf{Q}}[\rightarrow n] \ .$$

Note that $\hat{\mathbf{Q}}$ has 1 in the row $m-1$ and the column $s-1$. So does the matrix $\mathbf{R}$. Thus, the column $s$ of $\mathbf{R}$ is added with $\boldsymbol{\omega}$. Then,

$$\mathbf{Q} = \hat{\mathbf{Q}} + \hat{\mathbf{Q}}[\rightarrow s] + \sum_{n\in\mathcal{N}-\{0,s\}} \boldsymbol{\Omega}[\rightarrow n] \ .$$

Note that $\hat{\mathbf{Q}} + \hat{\mathbf{Q}}[\rightarrow s]$ equals to the reduction matrix given in the proposition. Also, there is no other nonzero entries in the last row of this matrix else than the column 0 and $s$. Thus, the reduction matrix must be as given in the proposition.

As can be understood from (10), for every nonzero entry of $\mathbf{Q}$, a coefficient of $d^{(L)}(x)$ is added with a coefficient of $d^{(H)}(x)$. Thus, the number of the required coefficient additions equals to the hamming weight of $\mathbf{Q}$, $2m-s-1$. Moreover, let $\theta = \max\{H(\mathbf{Qrow}_j) : \ 0 \le j \le m-1\}$. Then, maximum $\theta$ coefficients of $d^{(H)}(x)$ are added together to compute a single coefficient of the result. The corresponding coefficient of $d^{(L)}(x)$ must also be added to this sum. The addition of these $\lceil \log_2(\theta+1) \rceil T_X = \lceil \log_2 3 \rceil T_X = 2T_X$ coefficients causes the maximum delay.

□

## 4.2. Mastrovito Matrix

The Mastrovito matrix $\mathbf{M}$ can be computed from the reduction matrix $\mathbf{Q}$ as shown in (12). Using the equations (15) and (16), we obtain

$$\mathbf{M} = \mathbf{L} + \sum_{n\in\mathcal{N}} \hat{\mathbf{Q}}[\rightarrow n] \cdot \mathbf{U} = \mathbf{L} + \sum_{n\in\mathcal{N}} \sum_{k=1}^{r-1} \mathbf{I}_{m\times(m-1)}[\downarrow m_k][\rightarrow n] \cdot \mathbf{U}$$

for a set $\mathcal{N} \subset \{0, 1, \ldots, m-2\}$. Here, $m > m_1 > m_2 > \cdots > m_{r-2} > m_{r-1} = 0$ are the orders of the nonzero terms of the irreducible

polynomial $\omega(x)$ used to construct the field. Let us define

$$\mathbf{S} = \sum_{n \in \mathcal{N}} \hat{\mathbf{U}}[\rightarrow n]$$

where $\hat{\mathbf{U}} = [\mathbf{U}, \mathbf{0}_{1\times m}]^T$. Then, the Mastrovito matrix can be given by

$$\mathbf{M} = \mathbf{L} + \sum_{k=1}^{r-1} \mathbf{S}[\downarrow m_k] = \mathbf{L} + \mathbf{S} + \sum_{k=1}^{r-2} \mathbf{S}[\downarrow m_k] \ . \tag{19}$$

Note that $\mathbf{L}$ and $\mathbf{U}$ given by (13) as well as $\hat{\mathbf{U}} = [\mathbf{U}, \mathbf{0}_{1\times m}]^T$ are upper-triangular Toeplitz matrices. Then, the matrix $\mathbf{S}$ is also an upper-triangular Toeplitz matrix in the form

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

and it can be completely constructed by just computing

$$\mathbf{Srow}_0 = \sum_{n \in \mathcal{N}} \hat{\mathbf{U}}\mathbf{row}_0[\rightarrow n] = \sum_{n \in \mathcal{N}} \mathbf{U}\mathbf{row}_0[\rightarrow n] \ ,$$

where $\mathbf{Srow}_0$, $\hat{\mathbf{U}}\mathbf{row}_0$, and $\mathbf{U}\mathbf{row}_0$ denote the row 0 of the matrices $\mathbf{S}$, $\hat{\mathbf{U}}$, and $\mathbf{U}$ respectively.

The following proposition gives the complexity of the Mastrovito method for $r$-nomials.

*PROPOSITION 9.* [Zhang01]
Let $\omega(x)$ be a degree $m$ binary polynomial with $r$ nonzero terms. Let $m_k$ for $k = 1, 2, \ldots, r-1$ denote the orders of the nonzero terms except the term $x^m$. Then, if $m/2 \leq m_k$, the Mastrovito matrix method described by (19) and (11) can be computed by

- $(m-1)(m+r-2)$ coefficient additions,
- $m^2$ coefficient multiplications, and
- $T_A + (2r - 4 + \lceil \log_2 m \rceil) T_X$ delay.

The following proposition gives the complexity of the Mastrovito method for equally spaced polynomials.

*PROPOSITION 10.* [Zhang01]
Let $\omega(x)$ be a degree $m = ns$ equally spaced polynomial as in (4). Then, the Mastrovito matrix method described by (19) and (11) can be computed by

- $m^2 - s$ coefficient additions,
- $m^2$ coefficient multiplications, and
- $T_A + (1 + \lceil \log_2 m \rceil) T_X$ delay.

### 4.3. Complexity of Field Multiplication

The number of coefficient additions required for modular reduction performed by the reduction matrix is given in Proposition 7 and 8 for different types of irreducible polynomials. Table IV tabulates these results in the first column. Table IV also gives the number of the coefficient additions required for the field multiplication in the second column, adding the entries of the first column with $(m-1)^2$, the number of coefficient additions required for the polynomial multiplication.

Table IV. The number of the coefficient additions (XORs) (reduction matrix)

| Irreducible Polynomials | Modular Reduction | Field Multiplication |
|---|---|---|
| Equally Spaced | $2m - s - 1$ | $m^2 - s$ |
| $r$-nomial* | $(m-1)(r-1) + 1$ | $(m-1)(m+r-2) + 1$ |

*Assuming the order of the nonzero terms $m_j \leq (m+1)/2$, except the term $x^m$

Table V tabulates the modular reduction delay for different types of irreducible polynomials, given in the propositions 7 and 8. Also, this table presents the total field delay.

Table V. The time complexity (reduction matrix)

| Irreducible Polynomial | Modular Reduction | Field Multiplication |
|---|---|---|
| Equally Spaced | $2T_X$ | $T_A + (\lceil \log_2 m \rceil + 2)\, T_X$ |
| $r$-nomial* | $\big(\lceil \log_2 r \rceil + \lceil \log_2 (r-1) \rceil\big) T_X$ | $T_A + \big(\lceil \log_2 m \rceil + \lceil \log_2 r \rceil + \lceil \log_2 (r-1) \rceil\big) T_X$ |

*Assuming the order of the nonzero terms $m_j \leq (m+1)/2$, except the term $x^m$

The time complexity of the field multiplication can be further reduced by cleverly combining the polynomial multiplication and modu-

lar reduction operations as shown in [Reyhani04]. However, the decrease in the delay will be insignificant for any practical application.

The complexities for the Mastrovito Matrix Method are given in the propositions 9 and 10. The results are tabulated in Table VI and VII

Table VI. The number of the coefficient additions (XORs) (Mastrovito)

| Irreducible Polynomials | Field Multiplication |
|---|---|
| Equally Spaced $r$-nomial* | $m^2 - s$ $(m-1)(m+r-2)$ |
| *Assuming the order of the nonzero terms $m_j \leq m/2$, except the term $x^m$ | |

Table VII. The time complexity (Mastrovito)

| Irreducible Polynomial | Field Multiplication |
|---|---|
| Equally Spaced $r$-nomial* | $T_A + (\lceil \log_2 m \rceil + 1)\, T_X$ $T_A + (\lceil \log_2 m \rceil + 2r - 4) T_X$ |
| *Assuming the order of the nonzero terms $m_j \leq m/2$, except the term $x^m$ | |

One can conclude that direct field multiplication, reduction matrix, and Mastrovito matrix methods require the same number of coefficient additions and coefficient multiplications except that the reduction matrix method requires one extra coefficient addition. Also, note that the worst delay belongs the Mastrovito matrix method.

## 5. Montgomery Multiplication

In this section we explain the Montgomery multiplication method in GF($2^m$). Let the arbitrary irreducible polynomial in (1) define the field GF($2^m$). Instead of equation (2), the Montgomery multiplication calculates

$$c(x) = a(x)b(x)r^{-1}(x) \mod \omega(x) \tag{20}$$

where $r(x)$ is a fixed element and $gcd(r(x),\ \omega(x)) = 1$.

Because of Bezout's identity, one can find two polynomials $r^{-1}(x)$ and $\omega^{'}(x)$ such that

$$r(x)r^{-1}(x) + \omega(x)\omega^{'}(x) = 1 \tag{21}$$

where $r^{-1}(x)$ is the inverse of $r(x)$ modulo $\omega(x)$. These two polynomials can be calculated with the extended Euclidean algorithm. Koç and Acar [Koç98] selected $r(x) = x^m$ for high performance modular reduction in the Montgomery multiplication algorithm, which can be given as follows:

**Montgomery Modular Multiplication Algorithm**

| | |
|---|---|
| Inputs: | $a(x), b(x), r(x), \ \omega^{'}(x)$ |
| Output: | $c(x) = a(x)b(x)r^{-1}(x) \mod \omega(x)$ |
| Step 1: | $t(x) = a(x)b(x)$ |
| Step 2: | $u(x) = t(x)\ \omega^{'}(x) \mod r(x)$ |
| Step 3: | $c(x) = [t(x) + u(x)\ \omega(x)]/r(x)$ |

To prove the correctness of this algorithm we note that Step 2 implies that there exists a polynomial

$$u(x) = t(x)\ \omega^{'}(x) + h(x)r(x)\ . \tag{22}$$

We write $c(x)$ in Step 3 by using (22) as follows:

$$\begin{aligned} c(x) &= \tfrac{1}{r(x)}[t(x) + t(x)\ \omega^{'}(x)\ \omega(x) + h(x)r(x)\ \omega(x)] \\ &= \tfrac{1}{r(x)}[t(x)(1 + \omega^{'}(x)\ \omega(x)) + h(x)r(x)\ \omega(x)]\ . \end{aligned}$$

From (21), we can write $1 + \omega(x)\omega^{'}(x) = r(x)r^{-1}(x)$ and substitute it into our last expression

$$\begin{aligned} c(x) &= \tfrac{1}{r(x)}[t(x)r(x)r^{-1}(x) + h(x)r(x)\ \omega(x)] \\ &= t(x)r^{-1}(x) + h(x)\ \omega(x) \\ &= a(x)b(x)r^{-1} \mod \omega(x)\ . \end{aligned}$$

The degree of $c(x)$ can be verified from Step 3 as follows:

$$\begin{aligned} deg[c(x)] &\leq max\{deg[t(x)], deg[u(x)] + deg[\omega(x)]\} - deg[r(x)] \\ &\leq max\{2m-2, deg[r(x)] - 1 + m\} - deg[r(x)] \\ &\leq max\{2m - 2 - deg[r(x)], m-1\}\ . \end{aligned}$$

Then, it can be concluded that $deg[c(x)] \leq m-1$, if $deg[r(x)] \geq m-1$. If we choose $r(x) = x^m$, the result $c(x)$ will be of degree $m-1$ at most.

We can calculate the space and time complexities of each step in the Montgomery algorithm. In Step 1, we have $m^2$ coefficient multiplications (ANDs), $(m-1)^2$ coefficient additions (XORs), and $T_A + \lceil \log_2 m \rceil T_X$ delay as stated in Proposition 1 and Corollary 1.

In Step 2, the modular multiplication result $u(x)$ is at most of degree $m-1$ and has the coefficients $u_k = \sum_{i=0}^{k} a_i b_{k-i}$. Computing $u_k$ for $k = 0, 1, \ldots, m-1$ requires $m(m+1)/2$ coefficient multiplications and $m(m-1)/2$ coefficient additions. Computing $u_{m-1}$ causes the longest delay $T_A + \lceil \log_2 m \rceil T_X$.

In Step 3, the product $u(x)\omega(x)$ is a degree $2m-1$ polynomial. When this product is added to $t(x)$, the terms lower than $x^m$ cancel out and the result can be divided by $r(x)$ without any remainder. This is how the Montgomery method reduces the product. Because of the cancellations, we do not need to compute the terms of $c(x) = t(x) + u(x)\omega(x)$ lower than $x^m$ but the higher terms from order $m$ to $2m-1$. Computing the coefficient $c_k = t_k + \sum_{i=k-m+1}^{m-1} u_i \omega_{k-i}$ for $k = m, m+1, \ldots, 2m-1$ requires $2m-1-k$ coefficient additions and $2m-1-k$ coefficient multiplications. Then, the numbers of coefficient additions and multiplications required to compute $c(x)$ are both $m(m-1)/2$ . Computing the coefficient $c_m$ causes the maximum delay $T_A + \lceil \log_2 (m-1) \rceil T_X$.

Adding up all three steps, we observe that we need $2m^2$ coefficient multiplications (ANDs) and $2m^2 - 3m - 1$ coefficient additions (XORs) and the total time complexity is $3T_A + (2\lceil \log_2 m \rceil + \lceil \log_2 (m-1) \rceil) T_X$.

### 5.1. Reducing the Precision of Montgomery Multiplier

The Montgomery multiplication can be realized in an iterative way where the polynomials are represented with multiple subpolynomials. Let $n$ be the number of terms of our subpolynomials and $m = sn$ where $s$ is the number of subpolynomials. We represent our polynomial $a(x)$ as

$$\begin{aligned} a(x) &= \textstyle\sum_{i=0}^{s-1} A_i(x) x^{ni} \\ &= A_{s-1}(x) x^{n(s-1)} + A_{s-2}(x) x^{n(s-2)} + \cdots + A_1(x) x^n + A_0(x) \ , \end{aligned}$$

where $A_i(x)$ is a polynomial of length $n$ such that

$$\begin{aligned} A_i(x) &= \textstyle\sum_{j=0}^{n-1} a_{ni+j} x^j \\ &= a_{ni+n-1} x^{n-1} + a_{ni+n-2} x^{n-2} + \cdots + a_{ni+1} x + a_{ni} \ . \end{aligned}$$

We set $r(x) = x^n$. Let $C_0$ and $\Omega_0$ the last $n$ terms of $c(x)$ and $\omega(x)$, respectively. $\Omega_0^{'}(x)$ is equal to $\Omega_0^{-1}(x) \mod x^n$ because

$$\begin{aligned} x^{sn} x^{-sn} + \omega(x)\omega^{'}(x) &= 1 \ (\mathrm{mod} x^n) \\ \Omega_0 \Omega_0^{'} &= 1 \ (\mathrm{mod} x^n) \ . \end{aligned}$$

The iterative Montgomery algorithm can be realized as below:

**Iterative Montgomery Modular Multiplication Algorithm**

Inputs: $a(x), b(x),\ \omega(x),\ \Omega_0^{'}(x)$
Output: $c(x) = a(x)b(x)x^{-m} \mod \omega(x)$
Step 1: $c(x) = a(x)b(x)$
Step 2: $for\ i = 0\ to\ s-1\ do$
Step 3: $M(x) = C_0(x)\ \Omega_0^{'}(x)(\mathrm{mod} x^n)$
Step 4: $c(x) = [c(x) + M(x)\ \omega(x)]/x^n$

The algorithm requires the computation of the $n$-length polynomial $\Omega_0^{'}(x)$ instead of the entire polynomial $\omega^{'}(x)$. This computation is explained in detail by Koç and Acar. In order to divide our $c(x)$ with $x^n$ we add a multiple of $\omega(x)$ to $c(x)$ so that the least significant $n$ coefficients of $c(x)$ become zero.

### 5.2. Montgomery with Irreducible Trinomials

Wu proposes a variant of the Montgomery multiplication algorithm which uses $r(x) = x^k$ for some $k < m$ instead of $r(x) = x^m$ [Wu01]. However, this algorithm restricts the irreducible polynomial defining the field to a trinomial $\omega(x) = x^m + x^k + 1$ while Koç and Acar's approach allow an arbitrary irreducible polynomial. This algorithm introduces a final reduction step to the Montgomery multiplication algorithm which can be expressed as

Step 4: $if\ deg[c(x)] > m-1,\ then\ c(x) = c(x) \mod \omega(x).$

Although this final step seems to increase the complexity, the $u(x)$ term calculated at Step 2 has a lower degree bound which can reduce the complexity in Step 3. We will explain this approach in some detail.

From the Extended Euclidean algorithm we obtain $r^{-1}(x) = x^{m-k} + 1$ and $\omega^{'}(x) = 1$ that satisfies (21). The first step is exactly the same as the Montgomery Algorithm. We need $m^2$ coefficient multiplications (ANDs) and $(m-1)^2$ coefficient additions (XORs) to calculate $t(x)$. The time complexity is $T_A + \lceil \log_2 m \rceil T_X$. In the second step we calculate

$$\begin{aligned} u(x) &= t(x)\ \omega^{'}(x) \mod r(x) \\ &= t_0 + t_1 x + t_2 x^2 + \cdots + t_{2m-2} x^{2m-2} \mod x^k \\ &= t_0 + t_1 x + t_2 x^2 + \cdots + t_{k-1} x^{k-1}\ . \end{aligned} \tag{23}$$

To calculate Step 3, we partition $t(x)$ as follows:

$$t(x) = t_L(x) + x^k t_M(x) + x^{m+k} t_H(x)\ , \tag{24}$$

where

$$t_L(x) = t_0 + t_1 x + t_2 x^2 + \cdots + t_{k-1} x^{k-1} \ , \tag{25}$$

$$t_M(x) = t_k + t_{k+1} x + \cdots + t_{m+k-1} x^{m-1} \ , \tag{26}$$

and

$$t_H(x) = t_{m+k} + t_{m+k+1} x + \cdots + t_{2m-2} x^{m-k-2} \ . \tag{27}$$

By comparing (23) with (25) we observe $u(x) = t_L(x)$. Substituting $t(x)$ and $u(x)$ we can rewrite Step 3 as follows:

$$\begin{aligned} c(x) &= [t(x) + u(x)\ \omega(x)]/r(x) \\ &= [t_L(x) + x^k t_M(x) + x^{m+k} t_H(x) + t_L(x)(x^m + x^k + 1)]/r(x) \\ &= (x^{m-k} + 1) t_L(x) + t_M(x) + x^m t_H(x) \ . \end{aligned} \tag{28}$$

By Step 4, we obtain

$$\begin{aligned} c(x) &= c(x) \bmod \ \omega(x) \\ &= [(x^{m-k} + 1) t_L(x) + t_M(x) + x^m t_H(x)] \ \bmod \ \omega(x) \\ &= (x^{m-k} + 1) t_L(x) + t_M(x) + (x^k + 1) t_H(x) \ . \end{aligned} \tag{29}$$

We know the degrees of the polynomials in (29) and observe that none of the products will produce a polynomial with a higher degree than $m - 1$. The five polynomials which we obtain from (29) are

$$t_L(x) = t_0 + t_1 x + t_2 x^2 + \cdots + t_{k-1} x^{k-1} \ , \tag{30}$$

$$x^{m-k} t_L(x) = t_0 x^{m-k} + t_1 x^{m-k+1} + \cdots + t_{k-1} x^{m-1} \ , \tag{31}$$

$$t_M(x) = t_k + t_{k+1} x + \cdots + t_{m+k-1} x^{m-1} \ , \tag{32}$$

$$t_H(x) = t_{m+k} + t_{m+k+1} x + \cdots + t_{2m-2} x^{m-k-2} \ , \tag{33}$$

$$x^k t_H(x) = t_{m+k} x^k + t_{m+k+1} x^{k+1} + \cdots + t_{2m-2} x^{m-2} \ . \tag{34}$$

The field multiplication $c(x) = \sum_{i=0}^{m-1} c_i x^i$ is the addition of the five polynomials above. (32) contributes to all the coefficient of $c(x)$. To compute $c(x)$, we add the other four polynomials to the polynomial in (32). The polynomials in (30) and (31) are each added to the polynomial in(32) with $k$ coefficient additions. Also, the polynomials in (33) and (34) are each added to the sum with $m - k - 1$ additions. This will add up to $2m - 2$ coefficient additions (XORs). Notice that the polynomials in (30) and (31) can be added simultaneously to the sum because they don't have terms in common. Based on this fact the time complexity of this summation step will be $(\log_2 4) T_X = 2T_X$. To realize Wu's method, we need $m^2$ coefficient multiplications (ANDs) and $m^2 - 1$ coefficient additions in total. And, the total time complexity is not greater than $T_A + (\lceil \log_2 m \rceil + 2) T_X$.

## 6. Conclusions

In this paper, we describe several GF($2^m$) parallel multipliers using the polynomial basis. There are several methods to perform the two parts of the field multiplication: polynomial multiplication and modular reduction.

In section 3, we analyze the standard polynomial multiplication and the direct modular reduction. We give the details of the direct modular reduction for equally spaced irreducible polynomials and irreducible polynomials with a fixed number of nonzero terms. Coefficient additions and coefficient multiplications for the polynomial multiplication is $\mathcal{O}(m^2)$ while the delay is $\mathcal{O}(\log_2 m)$. Coefficient additions for the modular reduction is $\mathcal{O}(r.m)$ while the delay is $\mathcal{O}(\log_2 r)$ where $r$ is the number of the nonzero terms of the irreducible polynomial used in the modular reduction. Equally spaced polynomials which have a special structure yield somehow better space and time complexities in the modular reduction.

In Section 4, we mention matrix vector product techniques covering much space in the literature. The high order terms of a polynomial product can be reduced by multiplying them with a reduction matrix or multiplication and reduction can be performed together through a Mastrovito matrix multiplication. We show the relation between the reduction matrix and the Mastrovito matrix. The analysis of the reduction matrix method reveals that its complexity is almost the same as direct modular reduction. On the other hand, the analysis of the Mastrovito matrix reveals that its space complexity can be the same as previous methods but $r$, the number of the nonzero terms of the irreducible polynomial, is a linear term instead of $\mathcal{O}(\log_2 r)$ term in its time complexity. This is a disadvantage though very efficient Mastrovito multipliers are implemented for low hamming weight irreducible polynomials such as trinomials [Sunar1999] where $r$ is small. Equally spaced polynomials give the better performances than general $r$-term polynomials for all methods. Also, using irreducible polynomials with various special structures, one can improve the performance of the modular reduction. However, this improvement is not more than $m$ in the space complexity and 1 or 2 gate delay in the time complexity. Because the total space complexity is quadratic and the total time complexity is logarithmic, these are minor improvements.

In Section 5, we explain the general Montgomery multiplication technique which works for an arbitrary irreducible polynomial. The space complexity seems to be bearable while the time delay is high. By changing the precision of Montgomery multiplication we change the scope of parallelism in the implementations. This is a property that the

other methods do not have. Using trinomials in this method, brings the time and space complexity down to values that are comparable to the previous methods.

## References


Halbutoğulları, A. and Koç, Ç. K.: Mastrovito Multipliers for General Irreducible Polynomials, *IEEE Transactions on Computers*, vol. 49 no. 5, pp. 503-518 May 2000.

Hasan, M. A., Wang, M.Z., and Bhargava, V. K.: Modular Construction of Low Complexity Parallel Multipliers for a class of Finite Fields GF($2^m$), *IEEE Transactions on Computers*, vol 41 no 8, pp. 962-971 May 1992.

Itoh, T. and Tsujii, S.: Structure of Parallel Multipliers for a class of Finite Fields GF($2^m$) *Informations and Computations*, vol. 83, no. 8, pp. 21-40, 1989.

Koç, Ç. K. and Acar, T.: Montgomery Multiplication in $GF(2^k)$, *Designs, Codes and Cryptography*, 14(1), 57-69 April 1998.

Lidl, R. and Niederreiter, H.: *Introduction to Finite Fields and Their Applications*, Cambridge Univ. Press, 1994.

Mastrovito, E.D.: VLSI Designs for Multiplication over Finite Fields over GF($2^m$), *Proc. Sixth Int'l Conf. Applied Algebra, Algebraic Algorithms, and Error Correcting Codes (AAECC-6)*, pp. 297-309, July 1988.

Menezes, A. J.: *Applications of Finite Fields*, Kluwer Academic, 1994.

Menezes, A. J.: *Handbook of Applied Cryptography*, CRC, 1997.

Reyhani-Masoleh, A. and Hasan, M. A.: Low Complexity Bit-Parallel Architectures for Polynomial Basis Multiplication over GF($2^m$), *IEEE Transactions on Computers*, vol 53 no 8, 945-959 August 2004.

Sunar, B. and Koç, Ç. K.: Mastrovito Multiplier for All Trinomials, *IEEE Transactions on Computers*, vol 48 no 5, pp. 522-527 May 1999.

Wu, H. and Hasan, M. A.: Low-Complexity Bit-Parallel Multipliers for a class of Finite Fields, *IEEE Transactions on Computers*, vol 47 no. 8, pp. 883-887 Aug. 1998.

Wu, H. Montgomery Multiplier and Squarer for a Class of Finite Fields, *IEEE Transactions on Computers*, vol 51 no 5, pp. 521-529 May 2002.

Wu, H. Bit-Parallel Finite Field Multiplier and Squarer Using Polynomial Basis, *IEEE Transactions on Computers*, vol 51 no 7, pp. 750-758 July 2002.

Zhang, T. and Parhi, K. K.: Systematic Design of Original and Modified Mastrovito Multipliers for General Irreducible Polynomials, *IEEE Transactions on Computers*, vol 50 no 7, 734-749 July 2001.