

Mastrovito Multiplier for General Irreducible Polynomials ^{*†}

A. Halbutoğulları[‡] and Ç. K. Koç
Electrical & Computer Engineering
Oregon State University
Corvallis, Oregon 97331

June 26, 2000

Abstract

We present a new formulation of the Mastrovito multiplication matrix for the field $GF(2^m)$ generated by an arbitrary irreducible polynomial. We study in detail several specific types of irreducible polynomials, e.g., trinomials, all-one-polynomials, and equally-spaced-polynomials, and obtain the time and space complexity of these designs. Particular examples, illustrating the properties of the proposed architecture, are also given. The complexity results established in this paper match the best complexity results known to date. The most important new result is the space complexity of the Mastrovito multiplier for an equally-spaced-polynomial, which is found as $(m^2 - \Delta)$ XOR gates and m^2 AND gates, where Δ is the spacing factor.

Index Terms: Finite fields, multiplication, polynomial basis, complexity, irreducible polynomials.

1 Introduction

Efficient hardware implementations of the arithmetic operations in the Galois field $GF(2^m)$ are frequently desired in coding theory, computer algebra, and public-key cryptography [10, 9, 6]. The measure of efficiency is the number of gates (XOR and AND) and also the total gate delay of the circuit. The representation of the field elements have crucial role in the efficiency of the architectures for the arithmetic operations. For example, the well-known Massey-Omura [11] algorithm uses the normal basis representation, where the squaring of a field element is equivalent to a cyclic shift in its binary representation. Efficient bit-parallel algorithms for the multiplication operation in the canonical basis representation, which have much less space and time complexity than the Massey-Omura multiplier, have also been proposed.

The standard (polynomial) basis multiplication requires a polynomial modular multiplication followed by a modular reduction. In practice, these two steps can be combined. A novel method of multiplication is proposed by Mastrovito [7, 8], where a matrix product representation of the multiplication operation is used. The Mastrovito multiplier using the special generating trinomial $x^m + x + 1$ is shown to require $(m^2 - 1)$ XOR gates and m^2 AND gates [7, 8, 12, 13]. Furthermore, Sunar and Koç [14] has shown that when the irreducible trinomial is of the general form $x^m + x^n + 1$ for $n = 1, 2, \dots, m - 1$ and $m \neq 2n$, then the Mastrovito multiplier still requires $(m^2 - 1)$ XOR gates and m^2 AND gates. However, the required number of XOR gates is reduced to $(m^2 - \frac{m}{2})$ for the trinomial $x^m + x^{\frac{m}{2}} + 1$ where m is even [14].

In this paper, we generalize the approach of [14] in several different ways. We describe a method of construction for the Mastrovito multiplier for a general irreducible polynomial. We give detailed space and time analysis of the proposed method for several types of irreducible polynomials. In each case, the method proposed in this paper gives complexity results matching the best known results up to date.

The most important result of the this paper is in the case equally-spaced-polynomial (ESP), i.e., a polynomial of the form

$$p(x) = x^{k\Delta} + x^{(k-1)\Delta} + \dots + x^\Delta + 1, \quad (1)$$

^{*}IEEE Transactions on Computers, 49(5):503–518, May 2000.

[†]This research is supported in part by Intel Corporation and Secured Information Technology, Inc.

[‡]Present Address: i2 Technologies, 565 Technology Square, 9th Floor, Cambridge, MA 02139.

where $k\Delta = m$. The proposed Mastrovito multiplier for an ESP requires $(m^2 - \Delta)$ XOR gates and m^2 AND gates. For $k = 2$, the ESP reduces to the equally-spaced-trinomial (EST) $x^m + x^{\frac{m}{2}} + 1$, and for $\Delta = 1$, it reduces to the all-one-polynomial (AOP). Our method requires $(m^2 - \frac{m}{2})$ XOR and m^2 AND gates for the trinomial of the form $x^m + x^{\frac{m}{2}} + 1$ for an even m , matching the result in [14]. Furthermore, our proposed architecture requires $(m^2 - 1)$ XOR gates and m^2 AND gates when the irreducible polynomial is an AOP. This result matches the best known space complexity result to date for the canonical basis multiplication based on an irreducible AOP, as given in [5]. Their architecture requires $(m^2 - 1)$ XOR gates and m^2 AND gates, and has the lowest space complexity among similar bit-parallel multipliers [7, 4, 3].

We first introduce the fundamentals of the Mastrovito multiplier and the notation of this paper in §2. The architecture of the Mastrovito multiplier for a general irreducible polynomial is described in §3, in which we also give detailed complexity analysis and an example design. This section is followed by the architectural details of the multipliers based on binomials, trinomials, ESPs, and AOPs in §4, 5, 6, and 7, respectively. For each case we give a detailed complexity analysis and a design example. Finally, the conclusions of this study are summarized in §8.

2 Notation & Preliminaries

Let $p(x)$ be the irreducible polynomial generating the Galois field $GF(2^m)$. In order to compute the multiplication $c(x) = a(x)b(x) \bmod p(x)$ in $GF(2^m)$, where $a(x), b(x), c(x) \in GF(2^m)$, we need to first compute the product polynomial

$$d(x) = a(x)b(x) = \left(\sum_{i=0}^{m-1} a_i x^i \right) \left(\sum_{i=0}^{m-1} b_i x^i \right) \quad (2)$$

and then reduce $d(x)$ using $p(x)$ to find the result $c(x) \in GF(2^m)$. We can compute coefficients of $d(x)$ using following matrix-vector product:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_{m-2} \\ d_{m-1} \\ d_m \\ d_{m+1} \\ \vdots \\ d_{2m-3} \\ d_{2m-2} \end{bmatrix} = \begin{bmatrix} a_0 & 0 & 0 & \cdots & 0 & 0 \\ a_1 & a_0 & 0 & \cdots & 0 & 0 \\ a_2 & a_1 & a_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m-2} & a_{m-3} & a_{m-4} & \cdots & a_0 & 0 \\ a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_1 & a_0 \\ 0 & a_{m-1} & a_{m-2} & \cdots & a_2 & a_1 \\ 0 & 0 & a_{m-1} & \cdots & a_3 & a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{m-1} & a_{m-2} \\ 0 & 0 & 0 & \cdots & 0 & a_{m-1} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{m-2} \\ b_{m-1} \end{bmatrix}. \quad (3)$$

We will denote this multiplication matrix by \mathbf{M} and its rows denoted by \mathbf{M}_i , where $i = 0, 1, \dots, 2m - 2$. Note that the entries of \mathbf{M} solely consist of coefficients of $a(x)$. We also define the $m \times m$ submatrix $\mathbf{U}^{(0)}$ of \mathbf{M} as the first m rows of \mathbf{M} , and the $(m - 1) \times m$ submatrix $\mathbf{L}^{(0)}$ of \mathbf{M} as the last $(m - 1)$ rows of \mathbf{M} , i.e.,

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

$$\mathbf{L}^{(0)} = \begin{bmatrix} 0 & a_{m-1} & a_{m-2} & \cdots & a_2 & a_1 \\ 0 & 0 & a_{m-1} & \cdots & a_3 & a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{m-1} & a_{m-2} \\ 0 & 0 & 0 & \cdots & 0 & a_{m-1} \end{bmatrix}. \quad (5)$$

We use the superscripts to denote the step numbers during the reduction process, for example, $\mathbf{L}^{(0)}$, $\mathbf{L}^{(1)}$, etc. The superscript f indicates the final form of the matrix, for example, $\mathbf{L}^{(f)}$. The rows in the submatrix $\mathbf{L}^{(0)}$ of matrix \mathbf{M} is reduced using the irreducible polynomial $p(x)$, so that, at the end of reduction, $\mathbf{L}^{(f)}$ becomes the zero matrix. During the reduction process, the rows of \mathbf{L} are added to the rows with lower indices according to the irreducible polynomial. During the reduction of a single row, this row is added to certain other rows. We call all of these rows the children of the reduced row. The final submatrix $\mathbf{U}^{(f)}$ is equal to the so-called Mastrovito matrix \mathbf{Z} , which is multiplied by the column vector \mathbf{b} to produce the result as $\mathbf{c} = \mathbf{Z}\mathbf{b}$.

We use \parallel to represent the concatenation of two vectors. For example, the vectors $\mathbf{V} = [v_n, v_{n-1}, \dots, v_0]$ and $\mathbf{W} = [w_n, w_{n-1}, \dots, w_0]$ can be concatenated to form a new vector of length $2(n+1)$ as follows:

$$\mathbf{V} \parallel \mathbf{W} = [v_n \ v_{n-1} \ \cdots \ v_1 \ v_0 \ w_n \ w_{n-1} \ \cdots \ w_1 \ w_0].$$

In general, two vectors need not be of equal length in order to be concatenated. Also during the reduction, the vectors are shifted to the right or left, where the empty locations are filled with zeros. We use the right and left arrows to represent the right and left shifts. For example, $(\mathbf{V} \rightarrow \mathbf{3})$ and $(\mathbf{V} \leftarrow \mathbf{2})$ represent right and left shifts of the vector \mathbf{V} by 3 and 2 positions, respectively, which are explicitly given as

$$\begin{aligned} (\mathbf{V} \rightarrow \mathbf{3}) &= [0 \ 0 \ 0 \ v_n \ v_{n-1} \ \cdots \ v_6 \ v_5 \ v_4 \ v_3], \\ (\mathbf{V} \leftarrow \mathbf{2}) &= [v_{n-2} \ v_{n-3} \ v_{n-4} \ v_{n-5} \ v_{n-6} \ \cdots \ v_1 \ v_0 \ 0 \ 0]. \end{aligned}$$

Furthermore, at certain steps of the reduction, vectors are used to form matrices. For example, to form a matrix using the last $(n-1)$ entries of the above vectors, the following notation is adopted:

$$\begin{bmatrix} \mathbf{V} \\ (\mathbf{V} \rightarrow \mathbf{3}) \\ (\mathbf{V} \leftarrow \mathbf{2}) \end{bmatrix}_{3 \times (n-1)} = \begin{bmatrix} v_{n-2} & v_{n-3} & v_{n-4} & v_{n-5} & \cdots & v_3 & v_2 & v_1 & v_0 \\ 0 & v_n & v_{n-1} & v_{n-2} & \cdots & v_6 & v_5 & v_4 & v_3 \\ v_{n-4} & v_{n-5} & v_{n-6} & v_{n-7} & \cdots & v_2 & v_1 & 0 & 0 \end{bmatrix}.$$

As seen above, although the original vectors are longer, only the last $(n-1)$ entries are used, and the rest is discarded.

During the reduction operation, we frequently encounter certain special matrices. A particular matrix type is the Toeplitz matrix, which is a matrix whose entries are constant along each diagonal. It is well-known that the sum of two Toeplitz matrices is also a Toeplitz matrix [1]. This property will be used to establish a recursion.

Finally, we note that the gates used in the proposed design are 2-input AND and XOR gates, whose delays are denoted by T_A and T_X , respectively.

3 General Polynomials

We start with the most general form of an irreducible polynomial as

$$p(x) = x^{n_k} + x^{n_{k-1}} + \cdots + x^{n_1} + x^{n_0}, \quad (6)$$

where n_i for $i = 0, 1, 2, \dots, k$ are positive integers with the property

$$m = n_k > n_{k-1} > \cdots > n_1 > n_0 = 0.$$

The difference between the highest two orders, i.e., $n_k - n_{k-1} = m - n_{k-1}$ will be denoted by Δ .

In the following, we first summarize the general outline of the reduction process, and then, propose a method to obtain the same result more efficiently.

When the irreducible polynomial (6) is used to reduce the rows of $\mathbf{L}^{(0)}$, each row will have k children. The one corresponding to the constant term $x^{n_0} = 1$ is guaranteed to be added to a row in \mathbf{U} , but the others might be added to the rows of \mathbf{L} , and will need to be reduced further. To simplify the notation and observe the regularity, we use k additional matrices. The children produced due to the reductions corresponding to the x^{n_i} term will be added to the $m \times m$ matrix $\mathbf{Xi}^{(1)}$, for $i = 0, 1, \dots, (k-1)$. The children that fall back into the submatrix \mathbf{L} are stored in $\mathbf{L}^{(1)}$, which are to be reduced later.

By introducing the \mathbf{Xi} matrices, we preserve the matrix $\mathbf{U}^{(0)}$ during the reduction. At the end of the first step, i.e., when every row of matrix $\mathbf{L}^{(0)}$ is reduced exactly once, the following matrices will be produced:

$$\mathbf{U}^{(1)} = \mathbf{U}^{(0)} + \mathbf{X0}^{(1)} + \mathbf{X1}^{(1)} + \dots + \mathbf{X}(k-1)^{(1)}, \quad (7)$$

where

$$\mathbf{Xi}^{(1)} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & n_i - 1 \\ 0 & a_{m-1} & a_{m-2} & \cdots & a_{n_i} & \cdots & a_2 & a_1 & n_i \\ 0 & 0 & a_{m-1} & \cdots & a_{n_i+1} & \cdots & a_3 & a_2 & n_i + 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{m-1} & \cdots & a_{m-n_i+1} & a_{m-n_i} & m-1 \end{bmatrix} \quad (8)$$

for $i = 0, 1, \dots, (k-1)$. The part of matrix \mathbf{M} , which is to be further reduced after the first step, will be

$$\mathbf{L}^{(1)} = \begin{bmatrix} 0 & \cdots & 0 & l_{m-1}^{(1)} & l_{m-2}^{(1)} & \cdots & l_{\Delta+2}^{(1)} & l_{\Delta+1}^{(1)} & 0 \\ 0 & \cdots & 0 & 0 & l_{m-1}^{(1)} & \cdots & l_{\Delta+3}^{(1)} & l_{\Delta+2}^{(1)} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & l_{m-1}^{(1)} & l_{m-2}^{(1)} & n_{(k-1)} - 3 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & l_{m-1}^{(1)} & n_{(k-1)} - 2 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & n_{(k-1)} - 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & m-2 \end{bmatrix} \quad (9)$$

As seen above, the new matrix $\mathbf{L}^{(1)}$ which will be reduced in the next step is also triangular. This means the new children will also be in the same form, except that they will contain more zero terms at the beginning. Thus, it is clear that if the same procedure is recursively applied, the submatrix $\mathbf{U}^{(0)}$ will never change, and the forms of the matrices \mathbf{Xi} and \mathbf{L} will remain the same after every step, i.e., \mathbf{Xi} matrices will be trapezoidal and \mathbf{L} will be triangular. The entries which are zero and outside the indicated geometric regions after the first iteration will always remain zero. Only the values inside these regions will be changed during the rest of the reduction. The number of nonzero rows of \mathbf{L} after step j , i.e., the number of nonzero rows in $\mathbf{L}^{(j)}$ is given as

$$r_j = (m-1) - j(m - n_{(k-1)}) = (m-1) - j\Delta, \quad (10)$$

since there are $(m-1)$ nonzero rows initially, i.e., $r_0 = (m-1)$, and the number is reduced by $\Delta = (m - n_{(k-1)})$ after each step. Thus, it will take

$$N[m, \Delta] = \left\lceil \frac{m-1}{\Delta} \right\rceil \quad (11)$$

steps to reduce the whole matrix $\mathbf{L}^{(0)}$. This number is also equal to the number of nonzero terms in the row \mathbf{L}_0 at the end of step j , i.e., the number of nonzero terms in the row $\mathbf{L}_0^{(j)}$ for $j = 1, 2, \dots, N[m, \Delta] - 1$. Note that the range of j does not include $j = N[m, \Delta]$, as the number of nonzero rows becomes zero after step $N[m, \Delta]$, but the number r_j will be negative for $j = N[m, \Delta]$.

First the matrix \mathbf{M} is divided into the upper and lower submatrices $\mathbf{U}^{(0)}$ and $\mathbf{L}^{(0)}$. As shown in Figure 1, the matrix $\mathbf{L}^{(0)}$ is reduced into k matrices using the irreducible polynomial, while $\mathbf{U}^{(0)}$ is kept unchanged. The upper and lower parts of the new matrices are separated as illustrated in Figure 2. The upper parts form the

$\mathbf{Xi}^{(1)}$ matrices and the lower parts are accumulated into the matrix $\mathbf{L}^{(1)}$, as shown in Figure 3. The merged lower parts are to be further reduced in the next step. This procedure is repeated until the matrix \mathbf{L} becomes the zero matrix, i.e., all rows are reduced. The total reduction process is illustrated in Figure 4. The sum of the matrices in the last row, i.e., $\mathbf{U}^{(0)}$ and $\mathbf{Xi}^{(f)}$ for $i = 0, 1, \dots, (k-1)$, yields the Mastrovito matrix \mathbf{Z} .

A close inspection shows that all submatrices formed by the nonzero rows of the matrices produced after the first iteration are Toeplitz matrices. When the matrix $\mathbf{L}^{(1)}$ is reduced further, the children will be added to the nonzero rows of the matrices $\mathbf{Xi}^{(1)}$ and $\mathbf{L}^{(2)}$, which are Toeplitz submatrices. Since the sum of two Toeplitz matrices is also a Toeplitz matrix [1], the submatrices formed by the nonzero rows will all be Toeplitz submatrices. Furthermore, these matrices are special Toeplitz matrices, computing only the first nonzero rows of $\mathbf{Xi}^{(f)}$ is sufficient to reconstruct them. Similarly, the matrix \mathbf{M} and the submatrices $\mathbf{U}^{(0)}$ and $\mathbf{L}^{(0)}$ can be constructed using only the row \mathbf{M}_{m-1} . Furthermore, since all first nonzero rows of the matrices $\mathbf{Xi}^{(f)}$ are identical, we only need to compute one of them. Thus, it suffices to work on $\mathbf{X}\mathbf{0}_0^{(f)}$ whose value is computed as follows:

$$\sum_{j=0}^{N[m,\Delta]-1} \mathbf{L}_0^{(j)} = \mathbf{L}_0^{(0)} + \mathbf{L}_0^{(1)} + \dots + \mathbf{L}_0^{(N[m,\Delta]-1)}. \quad (12)$$

This will be used with $\mathbf{U}^{(0)}$ to construct the final matrix \mathbf{Z} . First, the rows \mathbf{Z}_{n_i} are constructed by adding the corresponding rows of the matrix $\mathbf{U}^{(0)}$ to $\mathbf{Xi}_{n_i}^{(f)}$ for $i = 0, 1, \dots, k-1$. Then, they are extended to larger vectors \mathbf{Yi} by concatenating the necessary parts of $\mathbf{U}^{(0)}$ to their beginning so that the shifts of \mathbf{Yi} produce the rows below them up to the row n_{i+1} , or up to the row $(m-1)$ if $i = (k-1)$. This will simplify the construction of the matrix \mathbf{Z} . To further simplify the representations, the first nonzero rows of \mathbf{Xi} , which are all identical, represented by the vector \mathbf{V} , will be used. Instead of referring to $\mathbf{U}^{(0)}$ and $\mathbf{L}^{(0)}$, we will use the original multiplication matrix \mathbf{M} or its entries a_i . The summary of the proposed method is given below:

1. First, we compute \mathbf{V} given as

$$\mathbf{V} = \sum_{j=0}^{N[m,\Delta]-1} \mathbf{L}_0^{(j)} = [0 \ v_{m-1} \ v_{m-2} \ \dots \ v_3 \ v_2 \ v_1] \quad (13)$$

using the recursive definition of

$$\mathbf{L}_0^{(j)} = \sum_{\substack{(k-1) \geq i \geq 0 \\ r_{(j-1)} > (m-n_i)}} (\mathbf{L}_0^{(j-1)} \rightarrow (m-n_i)) \quad (14)$$

for $1 \leq j \leq N[m,\Delta]-1$ to reduce everything to the sum of shifts of the row $\mathbf{L}_0^{(0)}$ or equivalently to the sum of rows in \mathbf{M} . The above summation means that the row $\mathbf{L}_0^{(j-1)}$ is shifted $(m-n_{(k-1)})$ times, $(m-n_{k-2})$ times, etc., until all entries become zero. Then, these are all added to form $\mathbf{L}_0^{(j)}$. Here we note that \mathbf{V} is not computed until it is completely reduced to the sum of rows of \mathbf{M} , since there might be cancellations. This fact will be taken into account during the complexity analysis.

2. Then, we compute \mathbf{Z}_{n_i} for $i = 0, 1, \dots, (k-1)$ using the following recursive relations:

$$\mathbf{Z}_0 = [a_0] \parallel [\mathbf{V}]_{1 \times (m-1)} = [a_0 \ v_{m-1} \ v_{m-2} \ \dots \ v_3 \ v_2 \ v_1], \quad (15)$$

$$\mathbf{Z}_{n_i} = ([\mathbf{M}_{m+n_{(i-1)}}]_{1 \times \Delta_i} \parallel [\mathbf{Z}_{n_{(i-1)}} \rightarrow \Delta_i]_{1 \times (m-\Delta_i)}) + \mathbf{V}, \quad (16)$$

where $\Delta_i = (n_i - n_{(i-1)})$ for $i = 1, 2, \dots, (k-1)$. Thus, if \mathbf{V} and $\mathbf{Z}_{n_{(i-1)}}$ are given as

$$\begin{aligned} \mathbf{V} &= [0 \ v_{m-1} \ v_{m-2} \ \dots \ v_3 \ v_2 \ v_1], \\ \mathbf{Z}_{n_{(i-1)}} &= [a_{n_{(i-1)}} \ w_{m-1} \ w_{m-2} \ \dots \ w_3 \ w_2 \ w_1], \end{aligned}$$

then \mathbf{Z}_{n_i} is obtained as

$$\mathbf{Z}_{n_i} = [a_{n_i} \ (a_{n_{i-1}} + v_{m-1}) \ \dots \ (a_{n_{(i-1)}} + v_{m-\Delta_i}) \ (w_{m-1} + v_{m-1-\Delta_i}) \ \dots \ (w_{\Delta_i+1} + v_1)].$$

by adding the vector \mathbf{V} to the vector formed by concatenation, which requires $(m - 1)$ XOR gates since the vector \mathbf{V} has only $(m - 1)$ nonzero terms. Thus, we need $(k - 1)(m - 1)$ XOR gates to compute all \mathbf{Z}_{n_i} for $i = 1, 2, \dots, (k - 1)$. Since the time needed to compute a single row \mathbf{Z}_{n_i} is T_X , the total delay to compute all rows is $(k - 1)T_X$. The vectors $\mathbf{Y0}$ and $\mathbf{Y1}$ are then found using Equation (17) by rewiring. The construction of \mathbf{Z} in Equation (18) is also performed using rewiring since it consists of shifts.

To find the final result $c(x)$ via the product $\mathbf{c} = \mathbf{Z}\mathbf{b}$, we also need m^2 AND and $m(m - 1)$ XOR gates. Each coefficient of the final result $c(x)$ can be computed independently via the product $\mathbf{c}_i = \mathbf{Z}_i\mathbf{b}$. All multiplications can be done in one level, and the m terms can be added using the binary tree method in $\lceil \log_2 m \rceil T_X$ time. Therefore, the entire computation for the general case requires m^2 AND gates and

$$(m - 1)(m + k - 1) + \sum_{j \in \mathcal{S}^*} (2m - 1 - j) \quad (19)$$

XOR gates. The total delay of the circuit is given as

$$T_A + (\lceil \log_2 |\mathcal{S}| \rceil + (k - 1) + \lceil \log_2 m \rceil) T_X . \quad (20)$$

3.2 An Example

As an example, we select the irreducible polynomial as $p(x) = x^7 + x^5 + x^3 + x + 1$, where $m = 7$, $k = 4$, $n_3 = 5$, and $\Delta = 7 - 5 = 2$. First we show the steps of the standard reduction technique to obtain the final \mathbf{Z} matrix. The standard reduction reduces the last $\Delta = 2$ rows of matrix \mathbf{M} at each step, and finds the final \mathbf{Z} matrix in $N[7, \Delta] = \lceil (7 - 1)/\Delta \rceil = \lceil 6/2 \rceil = 3$ steps. The steps are summarized below:

$$\mathbf{M} = \begin{bmatrix} a_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_1 & a_0 & 0 & 0 & 0 & 0 & 0 \\ a_2 & a_1 & a_0 & 0 & 0 & 0 & 0 \\ a_3 & a_2 & a_1 & a_0 & 0 & 0 & 0 \\ a_4 & a_3 & a_2 & a_1 & a_0 & 0 & 0 \\ a_5 & a_4 & a_3 & a_2 & a_1 & a_0 & 0 \\ a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ 0 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 \\ 0 & 0 & a_6 & a_5 & a_4 & a_3 & a_2 \\ 0 & 0 & 0 & a_6 & a_5 & a_4 & a_3 \\ 0 & 0 & 0 & 0 & a_6 & a_5 & a_4 \\ 0 & 0 & 0 & 0 & 0 & a_6 & a_5 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_6 \end{bmatrix} \rightarrow \begin{bmatrix} a_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_1 & a_0 & 0 & 0 & 0 & 0 & 0 \\ a_2 & a_1 & a_0 & 0 & 0 & 0 & 0 \\ a_3 & a_2 & a_1 & a_0 & 0 & 0 & 0 \\ a_4 & a_3 & a_2 & a_1 & a_0 & a_6 & a_5 \\ a_5 & a_4 & a_3 & a_2 & a_1 & (a_0 + a_6) & (a_5 + a_6) \\ a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & (a_0 + a_6) \\ 0 & a_6 & a_5 & a_4 & a_3 & (a_2 + a_6) & (a_1 + a_5) \\ 0 & 0 & a_6 & a_5 & a_4 & a_3 & (a_2 + a_6) \\ 0 & 0 & 0 & a_6 & a_5 & (a_4 + a_6) & (a_3 + a_5) \\ 0 & 0 & 0 & 0 & a_6 & a_5 & (a_4 + a_6) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} a_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_1 & a_0 & 0 & 0 & 0 & 0 & 0 \\ a_2 & a_1 & a_0 & a_6 & a_5 & (a_4 + a_6) & (a_3 + a_5) \\ a_3 & a_2 & a_1 & (a_0 + a_6) & (a_5 + a_6) & (a_4 + a_5 + a_6) & (a_3 + a_4 + a_5 + a_6) \\ a_4 & a_3 & a_2 & a_1 & (a_0 + a_6) & (a_5 + a_6) & (a_4 + a_5 + a_6) \\ a_5 & a_4 & a_3 & (a_2 + a_6) & (a_1 + a_5) & (a_0 + a_4) & (a_3 + a_6) \\ a_6 & a_5 & a_4 & a_3 & (a_2 + a_6) & (a_1 + a_5) & (a_0 + a_4) \\ 0 & a_6 & a_5 & (a_4 + a_6) & (a_3 + a_5) & (a_2 + a_4) & (a_1 + a_3) \\ 0 & 0 & a_6 & a_5 & (a_4 + a_6) & (a_3 + a_5) & (a_2 + a_4) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} a_0 & a_6 & a_5 & (a_4 + a_6) & (a_3 + a_5) & (a_2 + a_4) & (a_1 + a_3) \\ a_1 & (a_0 + a_6) & (a_5 + a_6) & (a_4 + a_5 + a_6) & (a_3 + a_4 + a_5 + a_6) & (a_2 + a_3 + a_4 + a_5) & (a_1 + a_2 + a_3 + a_4) \\ a_2 & a_1 & (a_0 + a_6) & (a_5 + a_6) & (a_4 + a_5 + a_6) & (a_3 + a_4 + a_5 + a_6) & (a_2 + a_3 + a_4 + a_5) \\ a_3 & (a_2 + a_6) & (a_1 + a_5) & (a_0 + a_4) & (a_3 + a_6) & (a_2 + a_5 + a_6) & (a_1 + a_4 + a_5 + a_6) \\ a_4 & a_3 & (a_2 + a_6) & (a_1 + a_5) & (a_0 + a_4) & (a_3 + a_6) & (a_2 + a_5 + a_6) \\ a_5 & (a_4 + a_6) & (a_3 + a_5) & (a_2 + a_4) & (a_1 + a_3) & (a_0 + a_2) & (a_1 + a_6) \\ a_6 & a_5 & (a_4 + a_6) & (a_3 + a_5) & (a_2 + a_4) & (a_1 + a_3) & (a_0 + a_2) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} .$$

The first 7 rows the above result forms the \mathbf{Z} matrix. On the other hand, the proposed method obtains the above result by performing the following steps:

1. First, we find the vector \mathbf{V} using Equation (13)

$$\mathbf{V} = \sum_{j=0}^{N[7,5]-1} \mathbf{L}_0^{(j)} = \mathbf{L}_0^{(0)} + \mathbf{L}_0^{(1)} + \mathbf{L}_0^{(2)} .$$

This can be computed using the recursive definition in Equation (14)

$$\mathbf{L}_0^{(j)} = \sum_{\substack{3 \geq i \geq 0 \\ r_{(j-1)} > (7 - n_i)}} (\mathbf{L}_0^{(j-1)} \rightarrow (7 - n_i)) .$$

Using this formula, we can reduce $\mathbf{L}_0^{(1)}$ and $\mathbf{L}_0^{(2)}$ to the following:

$$\begin{aligned} \mathbf{L}_0^{(1)} &= \sum_{\substack{3 \geq i \geq 0 \\ 6 > (7 - n_i)}} (\mathbf{L}_0^{(0)} \rightarrow (7 - n_i)) = (\mathbf{L}_0^{(0)} \rightarrow (7 - 5)) + (\mathbf{L}_0^{(0)} \rightarrow (7 - 3)) \\ &= (\mathbf{L}_0^{(0)} \rightarrow 2) + (\mathbf{L}_0^{(0)} \rightarrow 4) = \mathbf{M}_9 + \mathbf{M}_{11} . \end{aligned}$$

$$\begin{aligned} \mathbf{L}_0^{(2)} &= \sum_{\substack{3 \geq i \geq 0 \\ (7 - n_i) < 4}} (\mathbf{L}_0^{(1)} \rightarrow (7 - n_i)) = \mathbf{L}_0^{(1)} \rightarrow (7 - 5) \\ &= ((\mathbf{L}_0^{(0)} \rightarrow 2) + (\mathbf{L}_0^{(0)} \rightarrow 4)) \rightarrow (7 - 5) = (\mathbf{L}_0^{(0)} \rightarrow 4) + (\mathbf{L}_0^{(0)} \rightarrow 6) \\ &= (\mathbf{L}_0^{(0)} \rightarrow 4) = \mathbf{M}_{11} . \end{aligned}$$

2. Thus, the final form of the vector \mathbf{V} will be

$$\mathbf{V} = \mathbf{L}_0^{(0)} + \mathbf{L}_0^{(1)} + \mathbf{L}_0^{(2)} = \mathbf{M}_7 + (\mathbf{M}_9 + \mathbf{M}_{11}) + \mathbf{M}_{11} = \mathbf{M}_7 + \mathbf{M}_9 ,$$

which is computed as

$$\begin{aligned} \mathbf{V} &= [0 \ a_6 \ a_5 \ a_4 \ a_3 \ a_2 \ a_1] + [0 \ 0 \ 0 \ a_6 \ a_5 \ a_4 \ a_3] \\ &= [0 \ a_6 \ a_5 \ (a_4 + a_6) \ (a_3 + a_5) \ (a_2 + a_4) \ (a_1 + a_3)] . \end{aligned}$$

As can be seen from above, we have $\mathcal{S} = \{7, 9\}$ and $\mathcal{S}^* = \{9\}$.

3. Next we compute \mathbf{Z}_0 , \mathbf{Z}_1 , \mathbf{Z}_3 and \mathbf{Z}_5 using the recursive relations in Equations (15) and (16) as

$$\begin{aligned}\mathbf{Z}_0 &= [a_0] \parallel [\mathbf{V}]_{1 \times (7-1)}, \\ \mathbf{Z}_1 &= ([\mathbf{M}_7]_{1 \times 1} \parallel [\mathbf{Z}_0 \rightarrow 1]_{1 \times (7-1)}) + \mathbf{V}, \\ \mathbf{Z}_3 &= ([\mathbf{M}_{7+1}]_{1 \times 2} \parallel [\mathbf{Z}_1 \rightarrow 2]_{1 \times (7-2)}) + \mathbf{V}, \\ \mathbf{Z}_5 &= ([\mathbf{M}_{7+3}]_{1 \times 2} \parallel [\mathbf{Z}_3 \rightarrow 2]_{1 \times (7-2)}) + \mathbf{V},\end{aligned}$$

which are obtained as follows:

$$\begin{aligned}\mathbf{Z}_0 &= \begin{bmatrix} a_0 & a_6 & a_5 & (a_4 + a_6) & (a_3 + a_5) & (a_2 + a_4) & (a_1 + a_3) \end{bmatrix} \\ \mathbf{Z}_1 &= \begin{bmatrix} a_1 & a_0 & a_6 & a_5 & (a_4 + a_6) & (a_3 + a_5) & (a_2 + a_4) \end{bmatrix} \\ &+ \begin{bmatrix} 0 & a_6 & a_5 & (a_4 + a_6) & (a_3 + a_5) & (a_2 + a_4) & (a_1 + a_3) \end{bmatrix} \\ &= \begin{bmatrix} a_1 & (a_0 + a_6) & (a_5 + a_6) & (a_4 + a_5 + a_6) & (a_3 + a_4 + a_5 + a_6) & (a_2 + a_3 + a_4 + a_5) & (a_1 + a_2 + a_3 + a_4) \end{bmatrix} \\ \mathbf{Z}_3 &= \begin{bmatrix} a_3 & a_2 & a_1 & (a_0 + a_6) & (a_5 + a_6) & (a_4 + a_5 + a_6) & (a_3 + a_4 + a_5 + a_6) \end{bmatrix} \\ &+ \begin{bmatrix} 0 & a_6 & a_5 & (a_4 + a_6) & (a_3 + a_5) & (a_2 + a_4) & (a_1 + a_3) \end{bmatrix} \\ &= \begin{bmatrix} a_3 & (a_2 + a_6) & (a_1 + a_5) & (a_0 + a_4) & (a_3 + a_6) & (a_2 + a_5 + a_6) & (a_1 + a_4 + a_5 + a_6) \end{bmatrix} \\ \mathbf{Z}_5 &= \begin{bmatrix} a_5 & a_4 & a_3 & (a_2 + a_6) & (a_1 + a_5) & (a_0 + a_4) & (a_3 + a_6) \end{bmatrix} \\ &+ \begin{bmatrix} 0 & a_6 & a_5 & (a_4 + a_6) & (a_3 + a_5) & (a_2 + a_4) & (a_1 + a_3) \end{bmatrix} \\ &= \begin{bmatrix} a_5 & (a_4 + a_6) & (a_3 + a_5) & (a_2 + a_4) & (a_1 + a_3) & (a_0 + a_2) & (a_1 + a_6) \end{bmatrix}\end{aligned}$$

4. We then extend \mathbf{Z}_0 , \mathbf{Z}_1 , \mathbf{Z}_3 and \mathbf{Z}_5 using Equation (17) and obtain the following vectors:

$$\begin{aligned}\mathbf{Y}_0 &= [\mathbf{M}_7]_{1 \times (1-1)} \parallel \mathbf{Z}_0 = \mathbf{Z}_0 \\ &= [a_0 \ a_6 \ a_5 \ (a_4 + a_6) \ (a_3 + a_5) \ (a_2 + a_4) \ (a_1 + a_3)] . \\ \mathbf{Y}_1 &= [\mathbf{M}_8]_{1 \times (2-1)} \parallel \mathbf{Z}_1 \\ &= [a_2 \ a_1 \ (a_0 + a_6) \ (a_5 + a_6) \ (a_4 + a_5 + a_6) \\ &\quad (a_3 + a_4 + a_5 + a_6) \ (a_2 + a_3 + a_4 + a_5) \ (a_1 + a_2 + a_3 + a_4)] . \\ \mathbf{Y}_2 &= [\mathbf{M}_{10}]_{1 \times (2-1)} \parallel \mathbf{Z}_3 \\ &= [a_4 \ a_3 \ (a_2 + a_6) \ (a_1 + a_5) \\ &\quad (a_0 + a_4) \ (a_3 + a_6) \ (a_2 + a_5 + a_6) \ (a_1 + a_4 + a_5 + a_6)] . \\ \mathbf{Y}_3 &= [\mathbf{M}_{12}]_{1 \times (2-1)} \parallel \mathbf{Z}_5 \\ &= [a_6 \ a_5 \ (a_4 + a_6) \ (a_3 + a_5) \ (a_2 + a_4) \ (a_1 + a_3) \ (a_0 + a_2) \ (a_1 + a_6)] .\end{aligned}$$

5. Finally, the whole \mathbf{Z} matrix is constructed using Equation (18)

$$\begin{bmatrix} \mathbf{Y}_0 \\ \mathbf{Y}_1 \\ \mathbf{Y}_1 \rightarrow 1 \\ \mathbf{Y}_2 \\ \mathbf{Y}_2 \rightarrow 1 \\ \mathbf{Y}_3 \\ \mathbf{Y}_3 \rightarrow 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_6 & a_5 & (a_4 + a_6) & (a_3 + a_5) & (a_2 + a_4) & (a_1 + a_3) \\ a_1 & (a_0 + a_6) & (a_5 + a_6) & (a_4 + a_5 + a_6) & (a_3 + a_4 + a_5 + a_6) & (a_2 + a_3 + a_4 + a_5) & (a_1 + a_2 + a_3 + a_4) \\ a_2 & a_1 & (a_0 + a_6) & (a_5 + a_6) & (a_4 + a_5 + a_6) & (a_3 + a_4 + a_5 + a_6) & (a_2 + a_3 + a_4 + a_5) \\ a_3 & (a_2 + a_6) & (a_1 + a_5) & (a_0 + a_4) & (a_3 + a_6) & (a_2 + a_5 + a_6) & (a_1 + a_4 + a_5 + a_6) \\ a_4 & a_3 & (a_2 + a_6) & (a_1 + a_5) & (a_0 + a_4) & (a_3 + a_6) & (a_2 + a_5 + a_6) \\ a_5 & (a_4 + a_6) & (a_3 + a_5) & (a_2 + a_4) & (a_1 + a_3) & (a_0 + a_2) & (a_1 + a_6) \\ a_6 & a_5 & (a_4 + a_6) & (a_3 + a_5) & (a_2 + a_4) & (a_1 + a_3) & (a_0 + a_2) \end{bmatrix}$$

The above matrix found by the proposed method agrees with the 7×7 matrix obtained by the standard reduction method, as expected. There are 4 additions in the first computation of vector \mathbf{V} and $3 \cdot 6 = 18$ additions in the computation of rows \mathbf{Z}_i . The vectors \mathbf{Y}_i and the construction of the matrix \mathbf{Z} is performed using rewiring. Furthermore, we need additional 7^2 AND and $7 \cdot 6 = 42$ XOR gates to find the final result $c(x)$ via the product $\mathbf{c} = \mathbf{Z}\mathbf{b}$. Therefore, the entire computation for the general case requires 49 AND gates and $4 + 3 \cdot 6 + 42 = 64$ XOR gates. We can alternately use Equation (19) to compute the required XOR gates, where $\mathcal{S}^* = \{9\}$, $m = 7$, and $k = 4$. These values also give

$$(7-1)(7+4-1) + \sum_{j \in \{9\}} (14-1-j) = 60 + 4 = 64$$

XOR gates. Furthermore, we obtain the delay of the multiplication circuit using $\mathcal{S} = \{7, 9\}$, $m = 7$, and $k = 4$ in Equation (20) as

$$T_A + (\lceil \log_2 2 \rceil + (4 - 1) + \lceil \log_2 7 \rceil) T_X = T_A + 7T_X .$$

4 Binomials

When the reduction polynomial has a special form, the reduction process can be simplified, yielding simpler multipliers. Starting with the binomial case, we will study trinomials, AOPs, and ESPs in the following sections. When the reduction polynomial $p(x)$ is a binomial, it is always reducible, and thus, a Galois field cannot be constructed. However, reduction using a binomial is still needed and used. For example, the reduction process using the AOP

$$n(x) = x^{m-1} + x^{m-2} + x^{m-3} + \cdots + x^2 + x^1 + 1$$

is usually performed using the binomial $p(x) = x^m + 1$ since it is related to the AOP as $(x + 1)n(x) = x^m + 1$. Thus, we analyze the binomial case, since it is an intermediate step for the analysis of the AOP.

When we use the binomial $p(x) = x^m + 1$ to reduce the rows of $\mathbf{L}^{(0)}$, each row will have a single child which will be in \mathbf{U} , as explained in the general case. Furthermore the row number of the child in \mathbf{U} will be the same as the reduced row in $\mathbf{L}^{(0)}$. In this case, all reductions can be performed in a single step using $\mathbf{Z} = \mathbf{U}^{(0)} + \mathbf{L}^{(0)}$ as follows:

$$\begin{aligned} \mathbf{Z} &= \begin{bmatrix} a_0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ a_1 & a_0 & 0 & \cdots & 0 & 0 & 0 \\ a_2 & a_1 & a_0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{m-3} & a_{m-4} & a_{m-5} & \cdots & a_0 & 0 & 0 \\ a_{m-2} & a_{m-3} & a_{m-4} & \cdots & a_1 & a_0 & 0 \\ a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_2 & a_1 & a_0 \end{bmatrix} + \begin{bmatrix} 0 & a_{m-1} & a_{m-2} & \cdots & a_3 & a_2 & a_1 \\ 0 & 0 & a_{m-1} & \cdots & a_4 & a_3 & a_2 \\ 0 & 0 & 0 & \cdots & a_5 & a_4 & a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & a_{m-1} & a_{m-2} \\ 0 & 0 & 0 & \cdots & 0 & 0 & a_{m-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} a_0 & a_{m-1} & a_{m-2} & \cdots & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_{m-1} & \cdots & a_4 & a_3 & a_2 \\ a_2 & a_1 & a_0 & \cdots & a_5 & a_4 & a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{m-3} & a_{m-4} & a_{m-5} & \cdots & a_0 & a_{m-1} & a_{m-2} \\ a_{m-2} & a_{m-3} & a_{m-4} & \cdots & a_1 & a_0 & a_{m-1} \\ a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_2 & a_1 & a_0 \end{bmatrix} . \end{aligned}$$

As can be seen above, the computation of the last matrix requires no gates. In fact, all reductions can be performed by just working on a single vector. We define the intermediate vector \mathbf{Y} as

$$\begin{aligned} \mathbf{Y} &= M_{m-1} \parallel [M_{m-1} \rightarrow 1]_{1 \times (m-1)} \\ &= [a_{m-1} \ a_{m-2} \ \cdots \ a_2 \ a_1 \ a_0 \ a_{m-1} \ a_{m-2} \ \cdots \ a_2 \ a_1] . \end{aligned}$$

The whole \mathbf{Z} matrix can be constructed as follows:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Y} \\ \mathbf{Y} \rightarrow 1 \\ \mathbf{Y} \rightarrow 2 \\ \vdots \\ \mathbf{Y} \rightarrow (m-3) \\ \mathbf{Y} \rightarrow (m-2) \\ \mathbf{Y} \rightarrow (m-1) \end{bmatrix}_{m \times m} = \begin{bmatrix} a_0 & a_{m-1} & a_{m-2} & \cdots & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_{m-1} & \cdots & a_4 & a_3 & a_2 \\ a_2 & a_1 & a_0 & \cdots & a_5 & a_4 & a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{m-3} & a_{m-4} & a_{m-5} & \cdots & a_0 & a_{m-1} & a_{m-2} \\ a_{m-2} & a_{m-3} & a_{m-4} & \cdots & a_1 & a_0 & a_{m-1} \\ a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_2 & a_1 & a_0 \end{bmatrix} . \quad (21)$$

4.1 Complexity Analysis

Since the construction of \mathbf{Y} requires zero gates, the rows of the matrix \mathbf{Z} are obtained by only rewiring, and thus, the final \mathbf{Z} matrix costs nothing in terms of gates. The final product $c(x)$ is obtained using the matrix-vector product $\mathbf{c} = \mathbf{Z}\mathbf{b}$, which requires m^2 AND and $m(m-1)$ XOR gates. This computation is performed using the binary tree method, thus, the delay of the circuit will be $T_A + \lceil \log_2 m \rceil T_X$.

5 Trinomials

In this section, we study the reduction process using the trinomial $p(x) = x^m + x^n + 1$. We will study the EST $x^m + x^{\frac{m}{2}} + 1$ in §6 together with the ESPs. When the irreducible trinomial $p(x)$ is used to reduce the rows of $\mathbf{L}^{(0)}$, each row will have two children. The one corresponding to the constant term $x^{n_0} = 1$ is guaranteed to be in \mathbf{U} , but the other child might need to be further reduced depending on the position of the parent row. To simplify the notation, we will use two additional matrices: The reductions due to the constant term will always be added to the matrix $\mathbf{X0}$, and the reductions due to the x^n term will always be added to $\mathbf{X1}$. At the end of the first step, the following matrices are produced:

$$\begin{aligned}
\mathbf{Z}^{(1)} &= \mathbf{U}^{(0)} + \mathbf{X0}^{(1)} + \mathbf{X1}^{(1)} \\
&= \begin{bmatrix} a_0 & 0 & \cdots & 0 & 0 & 0 \\ a_1 & a_0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{m-3} & a_{m-4} & \cdots & a_0 & 0 & 0 \\ a_{m-2} & a_{m-3} & \cdots & a_1 & a_0 & 0 \\ a_{m-1} & a_{m-2} & \cdots & a_2 & a_1 & a_0 \end{bmatrix} + \begin{bmatrix} 0 & a_{m-1} & a_{m-2} & \cdots & a_2 & a_1 \\ 0 & 0 & a_{m-1} & \cdots & a_3 & a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{m-1} & a_{m-2} \\ 0 & 0 & 0 & \cdots & 0 & a_{m-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \\
&+ \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 \\ 0 & a_{m-1} & a_{m-2} & \cdots & a_n & \cdots & a_2 & a_1 \\ 0 & 0 & a_{m-1} & \cdots & a_{n+1} & \cdots & a_3 & a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{m-1} & \cdots & a_{\Delta+1} & a_{\Delta} \end{bmatrix} \begin{matrix} 0 \\ \vdots \\ (n-1) \\ n \\ (n+1) \\ \vdots \\ (m-1) \end{matrix} \\
\mathbf{L}^{(1)} &= \begin{bmatrix} 0 & \cdots & 0 & a_{m-1} & a_{m-2} & \cdots & a_{\Delta+2} & a_{\Delta+1} \\ 0 & \cdots & 0 & 0 & a_{m-1} & \cdots & a_{\Delta+3} & a_{\Delta+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & a_{m-1} & a_{m-2} \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & a_{m-1} \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \begin{matrix} 0 \\ 1 \\ \vdots \\ (n-3) \\ (n-2) \\ (n-1) \\ \vdots \\ (m-2) \end{matrix}
\end{aligned}$$

As can be seen above, the new matrix $\mathbf{L}^{(1)}$ is also triangular, which means that the new children will also be in the same form, except that they will contain more zero terms at the beginning. Thus, it is clear that if the same procedure is recursively applied, as in the general case, the submatrix $\mathbf{U}^{(0)}$ will never change, and the forms of the matrices \mathbf{Xi} and \mathbf{L} will remain the same after every step, i.e., \mathbf{Xi} matrices will be trapezoidal and \mathbf{L} matrix will be triangular. The entries which are zero and outside the indicated geometric regions after the first iteration will remain zero. Only the values inside these regions will be changed during the rest of the reduction. The number of nonzero rows of \mathbf{L} after step j , i.e., the number of nonzero rows in $\mathbf{L}^{(j)}$ is found by

$$r_j = (m-1) - j(m-n) = (m-1) - j\Delta,$$

since there are $(m-1)$ nonzero rows initially and the number is reduced by $\Delta = (m-n)$ after each step. Thus, it will take

$$N[m, \Delta] = \left\lceil \frac{m-1}{\Delta} \right\rceil$$

steps to reduce the whole matrix $\mathbf{L}^{(0)}$. This number is also equal to the number of nonzero terms in the row $\mathbf{L}_0^{(0)}$ at the end of step j , i.e., the number of nonzero terms in the row $\mathbf{L}_0^{(j)}$ for $j = 1, 2, \dots, N[m, \Delta] - 1$. Note that, similar to the general case, the range of j does not include $j = N[m, \Delta]$, since the number of nonzero rows becomes zero after step $N[m, \Delta]$, but the number r_j will be negative for $j = N[m, \Delta]$.

A close inspection shows that, as in the general case, all submatrices formed by the nonzero rows of the matrices produced after the first iteration are Toeplitz matrices. When the matrix $\mathbf{L}^{(1)}$ is reduced further, the children will be added to the nonzero rows of the matrices $\mathbf{X}\mathbf{0}^{(1)}$, $\mathbf{X}\mathbf{1}^{(1)}$, and $\mathbf{L}^{(2)}$, which are Toeplitz submatrices. Since the sum of Toeplitz matrices is again a Toeplitz matrix [1], the submatrices formed by the nonzero rows will also be Toeplitz submatrices. Also, computing only the first nonzero rows of $\mathbf{X}\mathbf{0}^{(f)}$ and $\mathbf{X}\mathbf{1}^{(f)}$ is sufficient to reconstruct them. Similarly the matrix \mathbf{M} and the submatrices $\mathbf{U}^{(0)}$ and $\mathbf{L}^{(0)}$ can be constructed using only the row \mathbf{M}_{m-1} . Since the first row of $\mathbf{X}\mathbf{0}^{(f)}$ and the n th row of $\mathbf{X}\mathbf{1}^{(f)}$ are identical, it is sufficient to compute only one of them. Thus, it suffices to work on only $\mathbf{X}\mathbf{0}_0^{(f)}$ which is computed using

$$\sum_{j=0}^{N[m, \Delta]-1} \mathbf{L}_0^{(j)} = \mathbf{L}_0^{(0)} + \mathbf{L}_0^{(1)} + \dots + \mathbf{L}_0^{(N[m, \Delta]-1)} .$$

Also the recursive relation in Equation (14) simplifies for trinomials to the following:

$$\mathbf{L}_0^{(j)} = \sum_{\substack{1 \geq i \geq 0 \\ r_{(j-1)} \geq (m-n_i)}} (\mathbf{L}_0^{(j-1)} \rightarrow (m-n_i)) = (\mathbf{L}_0^{(j-1)} \rightarrow (m-n)) = (\mathbf{L}_0^{(j-1)} \rightarrow \Delta) .$$

Therefore, $\mathbf{X}\mathbf{0}_0^{(f)}$ becomes

$$\sum_{j=0}^{N[m, \Delta]-1} (\mathbf{L}_0^{(0)} \rightarrow j\Delta) = \mathbf{L}_0^{(0)} + (\mathbf{L}_0^{(0)} \rightarrow \Delta) + \dots + (\mathbf{L}_0^{(0)} \rightarrow (N[m, \Delta] - 1)\Delta) .$$

This will be used with $\mathbf{U}^{(0)}$ to construct the final matrix \mathbf{Z} . First, the rows \mathbf{Z}_0 and \mathbf{Z}_n are formed by adding the corresponding rows of the matrix $\mathbf{U}^{(0)}$ to $\mathbf{X}\mathbf{0}_0^{(f)}$ and $\mathbf{X}\mathbf{1}_n^{(f)}$. Then they are extended to larger vectors $\mathbf{Y}\mathbf{0}$ and $\mathbf{Y}\mathbf{1}$ by concatenating the proper parts of $\mathbf{U}^{(0)}$ to their beginning, so that the shifted versions of $\mathbf{Y}\mathbf{0}$ and $\mathbf{Y}\mathbf{1}$ produce the rows below them. This will simplify the construction of the matrix \mathbf{Z} . Furthermore, the vectors $\mathbf{X}\mathbf{0}_0^{(f)}$ and $\mathbf{X}\mathbf{1}_n^{(f)}$, which are identical, will be represented by the vector \mathbf{V} . Instead of $\mathbf{U}^{(0)}$ and $\mathbf{L}^{(0)}$, we will use the original multiplication matrix \mathbf{M} or its entries a_i . The vector \mathbf{V} is defined as

$$\begin{aligned} \mathbf{V} &= \sum_{j=0}^{N[m, \Delta]-1} (\mathbf{L}_0^{(0)} \rightarrow j\Delta) = \sum_{j=0}^{N[m, \Delta]-1} \mathbf{M}_{m+j\Delta} \\ &= [0 \quad v_{m-1} \quad v_{m-2} \quad \dots \quad v_3 \quad v_2 \quad v_1] . \end{aligned}$$

Before proceeding any further, we consider the following equality

$$\begin{aligned} \mathbf{V} &= \sum_{j=0}^{N[m, \Delta]-1} (\mathbf{L}_0^{(0)} \rightarrow j\Delta) \\ &= \mathbf{L}_0^{(0)} + \sum_{j=1}^{N[m, \Delta]-1} (\mathbf{L}_0^{(0)} \rightarrow j\Delta) \end{aligned}$$

$$\begin{aligned}
&= \mathbf{L}_0^{(0)} + \left[\sum_{j=1}^{N[m,\Delta]-1} (\mathbf{L}_0^{(0)} \rightarrow j\Delta) \right] + (\mathbf{L}_0^{(0)} \rightarrow N[m,\Delta]\Delta) \\
&= \mathbf{L}_0^{(0)} + \sum_{j=1}^{N[m,\Delta]} (\mathbf{L}_0^{(0)} \rightarrow j\Delta) \\
&= \mathbf{L}_0^{(0)} + \left[\sum_{j=0}^{N[m,\Delta]-1} (\mathbf{L}_0^{(0)} \rightarrow j\Delta) \right] \rightarrow \Delta \\
&= \mathbf{L}_0^{(0)} + (\mathbf{V} \rightarrow \Delta) .
\end{aligned}$$

Note that the row $(\mathbf{L}_0^{(0)} \rightarrow N[m,\Delta]\Delta)$ added to the sum above is a zero vector since $\mathbf{L}_0^{(0)}$ has a zero as the first entry, and is shifted at least $(m-1)$ positions to the right. This equality suggests the following recursive formula for the computation of the entries of the vector \mathbf{V} .

$$v_i = \begin{cases} 0 & i = m, \\ a_i & (m-1) \geq i > (n-1), \\ a_i + v_{i+\Delta} & (n-1) \geq i \geq 1. \end{cases} \quad (22)$$

The formulae for \mathbf{Z}_0 and \mathbf{Z}_n can be found using the general recursive formula in Equations (15) and (16)

$$\begin{aligned}
\mathbf{Z}_0 &= [a_0] \parallel [\mathbf{V}]_{1 \times (m-1)} = [a_0 \ v_{m-1} \ v_{m-2} \ \cdots \ v_3 \ v_2 \ v_1] . \\
\mathbf{Z}_n &= (\mathbf{M}_n + (\mathbf{V} \rightarrow n)) + \mathbf{V} \\
&= [a_n \ (a_{n-1} + v_{m-1}) \ \cdots \ (a_0 + v_\Delta) \ (v_{m-1} + v_{\Delta-1}) \ \cdots \ (v_{n+1} + v_1)] .
\end{aligned}$$

However, there is another simplification here if the entries of \mathbf{Z}_n and \mathbf{Z}_0 are compared. The first n entries in \mathbf{Z}_n are identical to the last n entries of \mathbf{Z}_0 , and thus need not be recomputed again. This can be proved using the recursive formula of the vector \mathbf{V} in Equation (22) as follows:

$$\begin{aligned}
(\mathbf{Z}_0 \leftarrow \Delta) &= (\mathbf{M}_0 \leftarrow \Delta) + (\mathbf{V} \leftarrow \Delta) \\
&= \mathbf{V} \leftarrow \Delta \\
&= (\mathbf{L}_0^{(0)} + (\mathbf{V} \rightarrow \Delta)) \leftarrow \Delta \\
&= (\mathbf{L}_0^{(0)} \leftarrow \Delta) + ((\mathbf{V} \rightarrow \Delta) \leftarrow \Delta) \\
&= \mathbf{M}_n + ((\mathbf{V} \rightarrow \Delta) \leftarrow \Delta) .
\end{aligned}$$

During the second shift, the first n entries of the second term above are preserved, i.e., they are the same as in vector \mathbf{V} . Thus, the difference between the first n entries of the vectors $(\mathbf{Z}_0 \leftarrow \Delta)$ and \mathbf{Z}_n is only the vector $(\mathbf{V} \rightarrow n)$. Since the first $(n+1)$ entries of the vector $(\mathbf{V} \rightarrow n)$ are all zero, the vectors $(\mathbf{Z}_0 \leftarrow \Delta)$ and \mathbf{Z}_n agree in the first n positions. Therefore, during the computation of \mathbf{Z}_n , the first n positions can be produced by rewiring the last n entries of \mathbf{Z}_0 or \mathbf{V} , as they are the same, denoted as $[\mathbf{V}]_{1 \times n}$. Also note that the last Δ entries of the sum $(\mathbf{M}_n + (\mathbf{V} \rightarrow n))$ is equal to

$$[(\mathbf{Z}_0 \rightarrow n)]_{1 \times \Delta} = [a_0 \ v_{m-1} \ \cdots \ v_{n+1}] = [a_0 \ a_{m-1} \ \cdots \ a_{n+1}] .$$

Here Equation (22) is used to substitute $v_i = a_i$ for $i = (n+1), \dots, (m-1)$.

The steps of the proposed method are given below:

1. First, we compute the vector \mathbf{V} using the recursive definition in Equation (22).
2. Then, we compute \mathbf{Z}_0 and \mathbf{Z}_n using

$$\mathbf{Z}_0 = [a_0] \parallel [\mathbf{V}]_{1 \times (m-1)} = [a_0 \ v_{m-1} \ v_{m-2} \ \cdots \ v_3 \ v_2 \ v_1] . \quad (23)$$

$$\begin{aligned}
\mathbf{Z}_n &= [\mathbf{V}]_{1 \times n} \parallel ([\mathbf{V}]_{1 \times \Delta} + [(\mathbf{Z}_0 \rightarrow n)]_{1 \times \Delta}) \\
&= [v_n \ v_{n-1} \ \cdots \ v_1] \parallel ([v_\Delta \ v_{\Delta-1} \ \cdots \ v_1] + [a_0 \ a_{m-1} \ \cdots \ a_{n+1}]) \\
&= [a_n \ v_{n-1} \ \cdots \ v_1 \ (a_0 + v_\Delta) \ (a_{m-1} + v_{\Delta-1}) \ \cdots \ (a_{n+1} + v_1)] . \quad (24)
\end{aligned}$$

3. Next, we find $\mathbf{Y0}$ and $\mathbf{Y1}$ by extending $\mathbf{Z0}$ and \mathbf{Zn} .

$$\begin{aligned} \mathbf{Y0} &= [\mathbf{M}_m]_{1 \times (n-1)} \parallel \mathbf{Z0} \\ &= [a_{n-1} \ a_{n-2} \ \cdots \ a_1 \ a_0 \ v_{m-1} \ v_{m-2} \ \cdots \ v_2 \ v_1] . \end{aligned} \quad (25)$$

$$\begin{aligned} \mathbf{Y1} &= [\mathbf{M}_{m+n}]_{1 \times (\Delta-1)} \parallel \mathbf{Zn} \\ &= [a_{m-1} \ a_{m-2} \ \cdots \ a_{n+1} \ a_n \ v_{n-1} \ \cdots \ v_1 \ (a_0 + v_\Delta) \ (a_{m-1} + v_{\Delta-1}) \ \cdots \ (a_{n+1} + v_1)] . \end{aligned} \quad (26)$$

4. Finally, we construct the whole \mathbf{Z} matrix as

$$\begin{bmatrix} \mathbf{Y0} \\ \mathbf{Y0} \rightarrow 1 \\ \vdots \\ \mathbf{Y0} \rightarrow (n-1) \\ \mathbf{Y1} \\ \mathbf{Y1} \rightarrow 1 \\ \vdots \\ \mathbf{Y1} \rightarrow (\Delta-2) \\ \mathbf{Y1} \rightarrow (\Delta-1) \end{bmatrix}_{m \times m} = \begin{bmatrix} a_0 & v_{m-1} & \cdots & v_{\Delta+1} & v_\Delta & \cdots & v_1 \\ a_1 & a_0 & \cdots & v_{\Delta+2} & v_{\Delta+1} & \cdots & v_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \cdots & a_0 & v_{m-1} & \cdots & v_n \\ a_n & v_{n-1} & \cdots & v_1 & (a_0 + v_\Delta) & \cdots & (a_{n+1} + v_1) \\ a_{n+1} & a_n & \cdots & v_2 & v_1 & \cdots & (a_{n+2} + v_2) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m-2} & a_{m-3} & \cdots & v_{\Delta-1} & v_{\Delta-2} & \cdots & (a_{m-1} + v_{\Delta-1}) \\ a_{m-1} & a_{m-2} & \cdots & v_\Delta & v_{\Delta-1} & \cdots & (a_0 + v_\Delta) \end{bmatrix} . \quad (27)$$

Note that the entries of type $v_{\Delta-i}$ for $i \geq 0$ in the last rows are present only if their indices are less than n . Otherwise they will be taken as $a_{\Delta-i}$.

5.1 Complexity Analysis

The computation of the vector \mathbf{V} using the recursive relation in Equation (22) requires only one XOR gate for each of the last $(n-1)$ entries, which gives a total of $(n-1)$ XOR gates. The total delay is $(N[m, \Delta] - 1)T_X$ since the entries can be computed in blocks of Δ . The first block does not require any computation and, furthermore, the computations within each block can be performed in parallel.

The construction of $\mathbf{Z0}$ requires only rewiring. During the computation of \mathbf{Zn} , the first n positions are produced by rewiring, and only Δ XOR gates are needed to compute the last Δ entries since each entry requires a single addition. This computation takes T_X time, since all entries can be added in parallel.

The vectors $\mathbf{Y0}$ and $\mathbf{Y1}$ are then found using only rewiring. The construction of the matrix \mathbf{Z} is also accomplished using only rewiring, since it consists of shifted versions of the vectors $\mathbf{Y0}$ and $\mathbf{Y1}$. Finally, we compute the product $c(x)$ using $\mathbf{c} = \mathbf{Zb}$, which requires m^2 AND and $m(m-1)$ XOR gates. The delay for this last part is found as $T_A + \lceil \log_2 m \rceil T_X$ since the binary tree method is used to compute the sums.

Thus, the total computation for the trinomial case requires m^2 AND gates and $(n-1) + \Delta + m(m-1) = m^2 - 1$ XOR gates. The total delay of the circuit is

$$T_A + (N[m, \Delta] + \lceil \log_2 m \rceil) T_X = T_A + \left(\left\lceil \frac{m-1}{\Delta} \right\rceil + \lceil \log_2 m \rceil \right) T_X . \quad (28)$$

5.2 An Example

We illustrate the construction of the Mastrovito multiplication matrix for the field $GF(2^7)$ using the irreducible trinomial $p(x) = x^7 + x^4 + 1$. Here we have $m = 7$ and $n = 4$. The standard reduction method reduces the last $\Delta = 7 - 4 = 3$ rows of matrix \mathbf{M} at each step, and finds the final \mathbf{Z} matrix in $N[7, 4] = \lceil (7-1)(7-4) \rceil = 2$ steps. The operations performed by the standard reduction method to obtain the final matrix \mathbf{Z} are summarized below.

$$\mathbf{M} = \begin{bmatrix} a_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_1 & a_0 & 0 & 0 & 0 & 0 & 0 \\ a_2 & a_1 & a_0 & 0 & 0 & 0 & 0 \\ a_3 & a_2 & a_1 & a_0 & 0 & 0 & 0 \\ a_4 & a_3 & a_2 & a_1 & a_0 & 0 & 0 \\ a_5 & a_4 & a_3 & a_2 & a_1 & a_0 & 0 \\ a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ 0 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 \\ 0 & 0 & a_6 & a_5 & a_4 & a_3 & a_2 \\ 0 & 0 & 0 & a_6 & a_5 & a_4 & a_3 \\ 0 & 0 & 0 & 0 & a_6 & a_5 & a_4 \\ 0 & 0 & 0 & 0 & 0 & a_6 & a_5 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_6 \end{bmatrix} \rightarrow \begin{bmatrix} a_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_1 & a_0 & 0 & 0 & 0 & 0 & 0 \\ a_2 & a_1 & a_0 & 0 & 0 & 0 & 0 \\ a_3 & a_2 & a_1 & a_0 & a_6 & a_5 & a_4 \\ a_4 & a_3 & a_2 & a_1 & a_0 & a_6 & a_5 \\ a_5 & a_4 & a_3 & a_2 & a_1 & a_0 & a_6 \\ a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ 0 & a_6 & a_5 & a_4 & (a_3 + a_6) & (a_2 + a_5) & (a_1 + a_4) \\ 0 & 0 & a_6 & a_5 & a_4 & (a_3 + a_6) & (a_2 + a_5) \\ 0 & 0 & 0 & a_6 & a_5 & a_4 & (a_3 + a_6) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
\rightarrow \begin{bmatrix} a_0 & a_6 & a_5 & a_4 & (a_3 + a_6) & (a_2 + a_5) & (a_1 + a_4) \\ a_1 & a_0 & a_6 & a_5 & a_4 & (a_3 + a_6) & (a_2 + a_5) \\ a_2 & a_1 & a_0 & a_6 & a_5 & a_4 & (a_3 + a_6) \\ a_3 & a_2 & a_1 & a_0 & a_6 & a_5 & a_4 \\ a_4 & (a_3 + a_6) & (a_2 + a_5) & (a_1 + a_4) & (a_0 + a_3 + a_6) & (a_2 + a_5 + a_6) & (a_1 + a_4 + a_5) \\ a_5 & a_4 & (a_3 + a_6) & (a_2 + a_5) & (a_1 + a_4) & (a_0 + a_3 + a_6) & (a_2 + a_5 + a_6) \\ a_6 & a_5 & a_4 & (a_3 + a_6) & (a_2 + a_5) & (a_1 + a_4) & (a_0 + a_3 + a_6) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The first 7 rows of the matrix \mathbf{M} above is the 7×7 matrix \mathbf{Z} . On the other hand, our proposed method performs the following steps to obtain \mathbf{Z} .

1. First, we compute \mathbf{V} using the recursive relations in Equation (22)

$$\begin{aligned}
\mathbf{V} &= [0] \parallel [a_6 \ a_5 \ a_4] \parallel [(a_3 + v_{3+3}) \ (a_2 + v_{2+3}) \ (a_1 + v_{1+3})] \\
&= [0] \parallel [a_6 \ a_5 \ a_4] \parallel [(a_3 + a_6) \ (a_2 + a_5) \ (a_1 + a_4)] \\
&= [0 \ a_6 \ a_5 \ a_4 \ (a_3 + a_6) \ (a_2 + a_5) \ (a_1 + a_4)].
\end{aligned}$$

2. Then, we compute \mathbf{Z}_0 and \mathbf{Z}_4 using Equations (23) and (24)

$$\begin{aligned}
\mathbf{Z}_0 &= [a_0] \parallel [\mathbf{V}]_{1 \times (7-1)} \\
&= [a_0 \ a_6 \ a_5 \ a_4 \ (a_3 + a_6) \ (a_2 + a_5) \ (a_1 + a_4)]. \\
\mathbf{Z}_4 &= [\mathbf{V}]_{1 \times 4} \parallel ([\mathbf{V}]_{1 \times 3} + [(\mathbf{Z}_0 \rightarrow 4)]_{1 \times 3}) \\
&= [a_4 \ (a_3 + a_6) \ (a_2 + a_5) \ (a_1 + a_4)] \parallel [(a_3 + a_6) \ (a_2 + a_5) \ (a_1 + a_4)] + [a_0 \ a_6 \ a_5] \\
&= [a_4 \ (a_3 + a_6) \ (a_2 + a_5) \ (a_1 + a_4) \ (a_0 + a_3 + a_6) \ (a_2 + a_5 + a_6) \ (a_1 + a_4 + a_5)].
\end{aligned}$$

3. Next, we find the vectors \mathbf{Y}_0 and \mathbf{Y}_1 using Equations (25) and (26)

$$\begin{aligned}
\mathbf{Y}_0 &= [\mathbf{M}_7]_{1 \times (4-1)} \parallel \mathbf{Z}_0 \\
&= [a_3 \ a_2 \ a_1 \ a_0 \ a_6 \ a_5 \ a_4 \ (a_3 + a_6) \ (a_2 + a_5) \ (a_1 + a_4)]. \\
\mathbf{Y}_1 &= [\mathbf{M}_{11}]_{1 \times (3-1)} \parallel \mathbf{Z}_4 \\
&= [a_6 \ a_5 \ a_4 \ (a_3 + a_6) \ (a_2 + a_5) \ (a_1 + a_4) \ (a_0 + a_3 + a_6) \ (a_2 + a_5 + a_6) \ (a_1 + a_4 + a_5)].
\end{aligned}$$

4. Finally, we construct the whole \mathbf{Z} matrix using Equation (27)

$$\begin{bmatrix} \mathbf{Y0} \\ \mathbf{Y0} \rightarrow 1 \\ \mathbf{Y0} \rightarrow 2 \\ \mathbf{Y0} \rightarrow 3 \\ \mathbf{Y1} \\ \mathbf{Y1} \rightarrow 1 \\ \mathbf{Y1} \rightarrow 2 \end{bmatrix}_{7 \times 7} = \begin{bmatrix} a_0 & a_6 & a_5 & a_4 & (a_3 + a_6) & (a_2 + a_5) & (a_1 + a_4) \\ a_1 & a_0 & a_6 & a_5 & a_4 & (a_3 + a_6) & (a_2 + a_5) \\ a_2 & a_1 & a_0 & a_6 & a_5 & a_4 & (a_3 + a_6) \\ a_3 & a_2 & a_1 & a_0 & a_6 & a_5 & a_4 \\ a_4 & (a_3 + a_6) & (a_2 + a_5) & (a_1 + a_4) & (a_0 + a_3 + a_6) & (a_2 + a_5 + a_6) & (a_1 + a_4 + a_5) \\ a_5 & a_4 & (a_3 + a_6) & (a_2 + a_5) & (a_1 + a_4) & (a_0 + a_3 + a_6) & (a_2 + a_5 + a_6) \\ a_6 & a_5 & a_4 & (a_3 + a_6) & (a_2 + a_5) & (a_1 + a_4) & (a_0 + a_3 + a_6) \end{bmatrix}.$$

The final matrix found by the proposed method agrees with the standard method presented before, and also with the one in [14].

There are 3 additions in the computation of the vector \mathbf{V} . The vector \mathbf{Z}_4 is obtained by rewiring the first 4 entries and using 3 additions for the last 3 entries. The construction of the vectors $\mathbf{Y0}$ and $\mathbf{Y1}$ and the matrix \mathbf{Z} is performed using rewiring. In order to find the final result $c(x)$ via the product $\mathbf{c} = \mathbf{Z}\mathbf{b}$, we need 7^2 AND and $7 \cdot 6 = 42$ XOR gates. Therefore, the entire computation requires 49 AND gates and $3 + 3 + 42 = 48$ XOR gates. The total delay of the circuit can be found using Equation (28) with $m = 7$ and $\Delta = 3$

$$T_A + \left(\left\lceil \frac{7-1}{3} \right\rceil + \lceil \log_2 7 \rceil \right) T_X = T_A + 5T_X.$$

6 Equally-Spaced-Polynomials

Let the irreducible polynomial $p(x)$ be an ESP defined as

$$p(x) = x^{k\Delta} + x^{(k-1)\Delta} + \dots + x^\Delta + 1,$$

where $m = k\Delta$ and $k \geq 2$. The ESP reduces to the EST $x^m + x^{\frac{m}{2}} + 1$ for $k = 2$, and it reduces to the AOP $x^m + x^{m-1} + \dots + x + 1$ for $\Delta = 1$. The analysis given here covers both special cases.

When the row \mathbf{M}_j for $j \geq (k+1)\Delta$ of the initial matrix \mathbf{M} is reduced, it produces children on the rows with indices $(j - \Delta), (j - 2\Delta), \dots, (j - k\Delta)$. Similarly, when the child on the row $(j - \Delta)$ is further reduced, the new children will be on the rows $(j - 2\Delta), (j - 3\Delta), \dots, (j - (k+1)\Delta)$. Since all of these are identical, the new ones will cancel all of the old ones. The only new child left will be the one on the row $(j - (k+1)\Delta)$. Hence, we can say that the only offspring of the row \mathbf{M}_j for $j \geq (k+1)\Delta$ will be added to the row $(j - (k+1)\Delta)$. Because, what effectively is done above is the reduction of the row using the $(x^\Delta + 1)$ multiple of the original irreducible polynomial $p(x)$, which is given as

$$(x^\Delta + 1)p(x) = x^{(k+1)\Delta} + 1.$$

All rows in $\mathbf{L}^{(0)}$ except the first Δ of them can be reduced using the above polynomial. The first Δ rows have to be reduced using $p(x)$ itself. In terms of the matrix \mathbf{M} , this means that the rows

$$\mathbf{M}_{(k+1)\Delta}, \mathbf{M}_{(k+1)\Delta+1}, \dots, \mathbf{M}_{2m-2}$$

will be reduced using the multiple $(x^\Delta + 1)p(x)$, and the rows

$$\mathbf{M}_{k\Delta}, \mathbf{M}_{k\Delta+1}, \dots, \mathbf{M}_{(k+1)\Delta-1}$$

will be reduced using the original polynomial $p(x)$. The first part of the reduction does not use any addition since all reduced entries will be added to the upper triangular section of the matrix $\mathbf{U}^{(0)}$ which consists of zeros. The second part of the reduction can be managed using two distinct approaches, which we name Method I and Method II, respectively.

6.1 Method I

The proposed method computes only $\mathbf{Z}_{n_i} = \mathbf{Z}_{i\Delta}$ for $i = 0, 1, \dots, (k-1)$, however, the vector \mathbf{V} is not used this time. When this procedure is used, the row $\mathbf{Z}_{i\Delta}$ becomes

$$\begin{aligned} \mathbf{Z}_{i\Delta} &= (\mathbf{M}_{i\Delta} + \mathbf{M}_{i\Delta+(k+1)\Delta}) + \mathbf{M}_{k\Delta} \\ &= [a_{i\Delta} \ a_{i\Delta-1} \ \cdots \ a_0 \ 0 \ \cdots \ 0 \ a_{k\Delta-1} \ \cdots \ a_{(i+1)\Delta+1}] + \\ &\quad [0 \ a_{k\Delta-1} \ \cdots \ a_{(k-i)\Delta} \ a_{(k-i)\Delta-1} \ \cdots \ a_{(k-i-1)\Delta} \ a_{(k-i-1)\Delta-1} \ \cdots \ a_1] \\ &= [a_{i\Delta} \ (a_{i\Delta-1} + a_{k\Delta-1}) \ \cdots \ (a_0 + a_{(k-i)\Delta}) \ a_{(k-i)\Delta-1} \ \cdots \\ &\quad \cdots \ a_{(k-i-1)\Delta} \ (a_{k\Delta-1} + a_{(k-i-1)\Delta-1}) \ \cdots \ (a_{(i+1)\Delta+1} + a_1)]. \end{aligned}$$

We combine the first two vectors; no gates are needed to add them. The combined vector has Δ consecutive zeros, except for the row $\mathbf{Z}_{(k-1)\Delta}$ which has only $\Delta - 1$. As a result of this and the leading zero in $\mathbf{M}_{k\Delta}$, computation of $\mathbf{Z}_{i\Delta}$, for $i = 0, \dots, (k-2)$ requires only $((k-1)\Delta - 1)$ additions, and computation of $\mathbf{Z}_{(k-1)\Delta}$ requires $(k-1)\Delta$ additions.

However there is a further simplification in the computation due to appearance of the same additions. Using the definition of $\mathbf{Z}_{i\Delta}$, we find the row $\mathbf{Z}_{(k-i-1)\Delta}$ as

$$\begin{aligned} \mathbf{Z}_{(k-i-1)\Delta} &= [a_{(k-i-1)\Delta} \ (a_{(k-i-1)\Delta-1} + a_{k\Delta-1}) \cdots \ (a_0 + a_{(i+1)\Delta}) \ a_{(i+1)\Delta-1} \ \cdots \\ &\quad \cdots \ a_{i\Delta} \ (a_{k\Delta-1} + a_{i\Delta-1}) \ \cdots \ (a_{(k-i)\Delta+1} + a_1)]. \end{aligned}$$

As seen above, the first $(k-i-1)\Delta$ entries of the row $\mathbf{Z}_{(k-i-1)\Delta}$ are identical to the last $(k-i-1)\Delta$ entries of the row $\mathbf{Z}_{i\Delta}$. Furthermore, the last $i\Delta$ entries of the row $\mathbf{Z}_{(k-i-1)\Delta}$ are identical to the first $i\Delta$ entries of the row $\mathbf{Z}_{i\Delta}$. Thus, in order to obtain the row $\mathbf{Z}_{(k-i-1)\Delta}$, we need to perform a single addition: $(a_0 + a_{(i+1)\Delta})$. The rest of the terms can be constructed using certain entries of the row $\mathbf{Z}_{i\Delta}$ and the original matrix \mathbf{M} by rewiring.

When k is odd, the row $\mathbf{Z}_{\frac{(k-1)}{2}\Delta}$ is found as

$$\begin{aligned} \mathbf{Z}_{\frac{(k-1)}{2}\Delta} &= [a_{\frac{(k-1)}{2}\Delta} \ (a_{\frac{(k-1)}{2}\Delta-1} + a_{k\Delta-1}) \ \cdots \ (a_0 + a_{\frac{(k+1)}{2}\Delta}) \ a_{\frac{(k+1)}{2}\Delta-1} \ \cdots \\ &\quad \cdots \ a_{\frac{(k-1)}{2}\Delta} \ (a_{k\Delta-1} + a_{\frac{(k-1)}{2}\Delta-1}) \ \cdots \ (a_{\frac{(k+1)}{2}\Delta+1} + a_1)]. \end{aligned}$$

The first $\frac{(k-1)}{2}\Delta$ and the last $\frac{(k-1)}{2}\Delta$ entries of the above vector are identical. Thus, it is sufficient to compute the first $\frac{(k-1)}{2}\Delta$ entries and the term $(a_0 + a_{\frac{(k+1)}{2}\Delta})$. The proposed method computes the rows of the final matrix \mathbf{Z} directly using the following set of recursive relations:

- For $i = 0, 1, \dots, \lfloor \frac{k-2}{2} \rfloor$, we have

$$\mathbf{Z}_{i\Delta} = (\mathbf{M}_{i\Delta} + \mathbf{M}_{(k+i+1)\Delta}) + \mathbf{M}_{k\Delta}. \quad (29)$$

- For $i = \lceil \frac{k+1}{2} \rceil, \dots, (k-1)\Delta$, we have $\mathbf{Z}_{i\Delta} =$

$$[\mathbf{Z}_{(k-i-1)\Delta}]_{1 \times i\Delta} \parallel [(a_0 + a_{(k-i)\Delta}) \ a_{(k-i)\Delta-1} \cdots \ a_{(k-i-1)\Delta}] \parallel [\mathbf{Z}_{(k-i-1)\Delta} \rightarrow (i+1)\Delta]_{1 \times (k-i-1)\Delta}. \quad (30)$$

- For $i = \frac{k-1}{2}$, where k is odd, we have $\mathbf{Z}_{i\Delta} =$

$$[a_{i\Delta} \ (a_{i\Delta-1} + a_{k\Delta-1}) \cdots \ (a_0 + a_{(i+1)\Delta}) \ a_{(i+1)\Delta-1} \cdots \ a_{i\Delta} \ (a_{k\Delta-1} + a_{i\Delta-1}) \cdots \ (a_{(i+1)\Delta+1} + a_1)]. \quad (31)$$

When k is odd, the last $(i\Delta - 1)$ entries in the $i = \frac{k-1}{2}$ case are not computed; they are obtained from the first part of the same vector by rewiring. After computing $\mathbf{Z}_{i\Delta}$, we find $\mathbf{Y}_i\Delta$ for $i = 0, 1, \dots, (k-1)$ by extending $\mathbf{Z}_{i\Delta}$ as follows:

$$\begin{aligned} \mathbf{Y}_i\Delta &= [\mathbf{M}_{m+i\Delta}]_{1 \times (\Delta-1)} \parallel \mathbf{Z}_{i\Delta} \\ &= [a_{(i+1)\Delta-1} \ \cdots \ a_{i\Delta} \ (a_{i\Delta-1} + a_{k\Delta-1}) \ \cdots \ (a_0 + a_{(k-i)\Delta}) \ a_{(k-i)\Delta-1} \ \cdots \\ &\quad \cdots \ a_{(k-i-1)\Delta} \ (a_{k\Delta-1} + a_{(k-i-1)\Delta-1}) \ \cdots \ (a_{(i+1)\Delta+1} + a_1)]. \end{aligned} \quad (32)$$

Finally, the whole \mathbf{Z} matrix is constructed as follows:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Y0} & 0 \\ \mathbf{Y0} \rightarrow 1 & 1 \\ \vdots & \vdots \\ \mathbf{Y0} \rightarrow (\Delta - 1) & (\Delta - 1) \\ \mathbf{Y}\Delta & \Delta \\ \mathbf{Y}\Delta \rightarrow 1 & (\Delta + 1) \\ \vdots & \vdots \\ \mathbf{Y}(i-1)\Delta \rightarrow (\Delta - 1) & (i\Delta - 1) \\ \mathbf{Y}i\Delta & i\Delta \\ \mathbf{Y}i\Delta \rightarrow 1 & (i\Delta + 1) \\ \vdots & \vdots \\ \mathbf{Y}(k-2)\Delta \rightarrow (\Delta - 1) & ((k-1)\Delta - 1) \\ \mathbf{Y}(k-1)\Delta & (k-1)\Delta \\ \mathbf{Y}(k-1)\Delta \rightarrow 1 & ((k-1)\Delta + 1) \\ \vdots & \vdots \\ \mathbf{Y}(k-1)\Delta \rightarrow (\Delta - 1) & (m-1) \end{bmatrix}_{m \times m} \quad (33)$$

6.2 Complexity Analysis of Method I

When k is even, the rows $\mathbf{Z}_{i\Delta}$ with indices $i = 0, 1, \dots, \frac{k-2}{2}$ can be computed using $(k-1)\Delta - 1$ additions each. Also, the rows for $i = \frac{k}{2}, \dots, (k-1)$ can be computed using one addition each, which gives a total of

$$\left(\frac{k-2}{2} + 1\right)((k-1)\Delta - 1) + (k-1 - \frac{k}{2} + 1) = \frac{k}{2}(k-1)\Delta$$

additions. When k is odd, the first $\frac{(k-1)}{2}\Delta$ and the last $\frac{(k-1)}{2}\Delta$ entries in the row $\mathbf{Z}_{\frac{(k-1)}{2}\Delta}$ are identical, thus, it is sufficient to perform the first $\frac{(k-1)}{2}\Delta$ additions. The rows $\mathbf{Z}_{i\Delta}$ with $i = 0, 1, \dots, \frac{(k-1)}{2} - 1$ can be computed using $(k-1)\Delta - 1$ additions each. Also, the rows with indices $i = \frac{(k+1)}{2}, \dots, (k-1)$ can be computed using one addition each as in the previous case, which gives a total of

$$\frac{(k-1)}{2}((k-1)\Delta - 1) + \frac{(k-1)}{2}\Delta + [(k-1) - (\frac{(k+1)}{2} + 1) + 1] = \frac{(k-1)}{2}k\Delta$$

additions. Therefore, in both cases, we need $\frac{k(k-1)\Delta}{2} = \frac{m(k-1)}{2}$ XOR gates to compute the rows $\mathbf{Z}_{i\Delta}$ matrix. The vectors $\mathbf{Y}i\Delta$ and the final matrix \mathbf{Z} can be found using only rewiring. Also, we need $m^2 = k^2\Delta^2$ AND and $m(m-1)$ XOR gates to find the final result $c(x)$ via the product $\mathbf{c} = \mathbf{Z}\mathbf{b}$. Hence, the entire computation requires m^2 AND gates and

$$\frac{m(k-1)}{2} + m(m-1) = m^2 + m\frac{(k-3)}{2} \quad (34)$$

XOR gates. The total delay of the circuit that computes the matrix \mathbf{Z} is only T_X since all entries requires a single addition, and can be computed in parallel. The final product $\mathbf{c} = \mathbf{Z}\mathbf{b}$ is computed using the binary tree method, which gives the delay of the circuit as

$$T_A + (1 + \lceil \log_2 m \rceil) T_X .$$

6.3 Method II

There is also another approach which depends on the observation that the last vector used in the computation of $\mathbf{Z}_{i\Delta}$ is always the same row, i.e., $\mathbf{M}_{k\Delta} = \mathbf{M}_m$. Then, the contribution of this vector can be computed

separately, and can be added to the contribution of the other vectors for each row. The result $c_{i\Delta} = \mathbf{Z}_{i\Delta} \mathbf{b}$ can be partitioned as follows:

$$c_{i\Delta} = \mathbf{Z}_{i\Delta} \mathbf{b} = \mathbf{Z}'_{i\Delta} \mathbf{b} + \mathbf{Z}''_{i\Delta} \mathbf{b} ,$$

where the vector $\mathbf{Z}'\mathbf{b}$ contains the contribution of the first two vectors in the definition of $\mathbf{Z}_{i\Delta}$ in Equations (29), (30), and (31). Also, $\mathbf{Z}''\mathbf{b}$ contains the contribution of the last vector. However, since the results are not vectors but numbers, shifting cannot be applied to construct the rows of \mathbf{c} with indices which are not multiples of Δ . Thus, we have to construct the whole matrices \mathbf{Z}' and \mathbf{Z}'' . However, there will be simplifications since there are only Δ different rows in the matrix \mathbf{Z}'' . In terms of the row \mathbf{c}_i , this corresponds to following partition:

$$\mathbf{c}_i = \mathbf{Z}_i \mathbf{b} = (\mathbf{M}_i + \mathbf{M}_{i+(k+1)\Delta}) \mathbf{b} + \mathbf{M}_{k\Delta+j} \mathbf{b} , \quad (35)$$

where $i \equiv j \pmod{\Delta}$ and $0 \leq j < \Delta$. The result \mathbf{c} is then directly found by using these entries.

6.4 Complexity Analysis of Method II

As shown before, the sum $(\mathbf{M}_i + \mathbf{M}_{i+(k+1)\Delta})$ requires no gates, and has Δ consecutive zeros, except for the rows $\mathbf{Z}_{(k-1)\Delta+l}$ with $0 \leq l < \Delta$, where it has only $(\Delta - 1 - l)$ zeros. Thus, the first product requires $(k-1)\Delta$ AND and $(k-1)\Delta - 1$ XOR gates for $i = 0, 1, \dots, (k-1)\Delta - 1$, and $((k-1)\Delta + 1 + l)$ AND and $((k-1)\Delta + l)$ XOR gates for $i = (k-1)\Delta + l$ with $0 \leq l < \Delta$.

The row $\mathbf{M}_{k\Delta+j}$ has $(k\Delta - j - 1)$ nonzero terms, thus, the second product requires $(k\Delta - j - 1)$ AND gates and $(k\Delta - j - 2)$ XOR gates for $0 \leq j < \Delta$. An additional XOR gate is needed for the final sum of each row. Thus, the second approach requires

$$(k-1)\Delta((k-1)\Delta) + \left[\sum_{l=0}^{\Delta-1} (k-1)\Delta + 1 + l \right] + \left[\sum_{j=0}^{\Delta-1} k\Delta - j - 1 \right] = k^2 \Delta^2 = m^2$$

AND gates and

$$(k-1)\Delta((k-1)\Delta - 1) + \left[\sum_{l=0}^{\Delta-1} (k-1)\Delta + l \right] + \left[\sum_{j=0}^{\Delta-1} k\Delta - j - 2 \right] + k\Delta = m^2 - \Delta$$

XOR gates. All vector products can be carried out in parallel. The bottleneck for the delay is the last row in the first product, i.e., when $l = (\Delta - 1)$, where two full vectors of length m are multiplied. When the binary tree method is used to compute the additions, the total delay becomes

$$T_A + (1 + \lceil \log_2 m \rceil) T_X .$$

6.5 Comparing Methods I and II

The delay and the number of AND gates required by the two methods are the same for all possible values of k and Δ . When $k = 2$, i.e., in the case of ESTs, Method I requires $m^2 + m \frac{(k-3)}{2} = m^2 - \frac{m}{2}$ XOR gates, which is equal to the number of gates required by Method II since $2\Delta = m$ implies $\Delta = \frac{m}{2}$, and thus, we have $m^2 - \Delta = m^2 - \frac{m}{2}$. Therefore, both methods give the same XOR complexity for the ESTs.

For $k \geq 3$, Method I requires $m^2 + \frac{m(k-3)}{2} \geq m^2$ XOR gates, while Method II requires $m^2 - \Delta < m^2$ XOR gates since $\Delta = \frac{m}{k} > 0$. In other words, Method II requires fewer XOR gates when $k \geq 3$.

6.6 An Example

In this section, we illustrate the reduction process for the irreducible EST $p(x) = x^6 + x^3 + 1$. Here, we have $\Delta = 3$ and $m = k\Delta = 6$. Since $k = 2$, Methods I and II solutions have the same efficiency. We illustrate both methods and compare the reduction process below.

The multiple $(x^3 + 1)p(x)$ is found as $(x^3 + 1)p(x) = x^9 + 1$. In the proposed methods, the rows \mathbf{M}_9 and \mathbf{M}_{10} will be reduced using this multiple, and the rows \mathbf{M}_6 , \mathbf{M}_7 and \mathbf{M}_8 will be reduced using the original polynomial. First, we illustrate the standard reduction technique to obtain the matrix \mathbf{Z} .

$$\mathbf{M} = \begin{bmatrix} a_0 & 0 & 0 & 0 & 0 & 0 \\ a_1 & a_0 & 0 & 0 & 0 & 0 \\ a_2 & a_1 & a_0 & 0 & 0 & 0 \\ a_3 & a_2 & a_1 & a_0 & 0 & 0 \\ a_4 & a_3 & a_2 & a_1 & a_0 & 0 \\ a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ 0 & a_5 & a_4 & a_3 & a_2 & a_1 \\ 0 & 0 & a_5 & a_4 & a_3 & a_2 \\ 0 & 0 & 0 & a_5 & a_4 & a_3 \\ 0 & 0 & 0 & 0 & a_5 & a_4 \\ 0 & 0 & 0 & 0 & 0 & a_5 \end{bmatrix} \rightarrow \begin{bmatrix} a_0 & 0 & 0 & 0 & a_5 & a_4 \\ a_1 & a_0 & 0 & 0 & 0 & a_5 \\ a_2 & a_1 & a_0 & 0 & 0 & 0 \\ a_3 & a_2 & a_1 & a_0 & 0 & 0 \\ a_4 & a_3 & a_2 & a_1 & a_0 & 0 \\ a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ 0 & a_5 & a_4 & a_3 & a_2 & a_1 \\ 0 & 0 & a_5 & a_4 & a_3 & a_2 \\ 0 & 0 & 0 & a_5 & a_4 & a_3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} a_0 & a_5 & a_4 & a_3 & (a_2 + a_5) & (a_1 + a_4) \\ a_1 & a_0 & a_5 & a_4 & a_3 & (a_2 + a_5) \\ a_2 & a_1 & a_0 & a_5 & a_4 & a_3 \\ a_3 & (a_2 + a_5) & (a_1 + a_4) & (a_0 + a_3) & a_2 & a_1 \\ a_4 & a_3 & (a_2 + a_5) & (a_1 + a_4) & (a_0 + a_3) & a_2 \\ a_5 & a_4 & a_3 & (a_2 + a_5) & (a_1 + a_4) & (a_0 + a_3) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The first 6 rows of the above matrix form the matrix \mathbf{Z} . Now, we illustrate both methods. Method II first computes \mathbf{Z}_0 and \mathbf{Z}_3 using Equations (29), (30), and (31) as

$$\begin{aligned} \mathbf{Z}_0 &= (\mathbf{M}_0 + \mathbf{M}_9) + \mathbf{M}_6 \\ &= ([a_0 \ 0 \ 0 \ 0 \ 0 \ 0] + [0 \ 0 \ 0 \ 0 \ a_5 \ a_4]) + [0 \ a_5 \ a_4 \ a_3 \ a_2 \ a_1] \\ &= [a_0 \ a_5 \ a_4 \ a_3 \ (a_2 + a_5) \ (a_1 + a_4)]. \\ \mathbf{Z}_3 &= [a_3 \ (a_2 + a_5) \ (a_1 + a_4) \ (a_0 + a_3) \ a_2 \ a_1]. \end{aligned}$$

We then find $\mathbf{Yi}\Delta$ using Equation (32) as

$$\begin{aligned} \mathbf{Y0} &= [\mathbf{M}_6]_{1 \times 2} \parallel \mathbf{Z}_0 = [a_2 \ a_1 \ a_0 \ a_5 \ a_4 \ a_3 \ (a_2 + a_5) \ (a_1 + a_4)], \\ \mathbf{Y1} &= [\mathbf{M}_9]_{1 \times 2} \parallel \mathbf{Z}_3 = [a_5 \ a_4 \ a_3 \ (a_2 + a_5) \ (a_1 + a_4) \ (a_0 + a_3) \ a_2 \ a_1]. \end{aligned}$$

The whole \mathbf{Z} matrix can be constructed using Equation (33) as follows:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Y0} \\ \mathbf{Y0} \rightarrow 1 \\ \mathbf{Y0} \rightarrow 2 \\ \mathbf{Y1} \\ \mathbf{Y1} \rightarrow 1 \\ \mathbf{Y1} \rightarrow 2 \end{bmatrix}_{7 \times 7} = \begin{bmatrix} a_0 & a_5 & a_4 & a_3 & (a_2 + a_5) & (a_1 + a_4) \\ a_1 & a_0 & a_5 & a_4 & a_3 & (a_2 + a_5) \\ a_2 & a_1 & a_0 & a_5 & a_4 & a_3 \\ a_3 & (a_2 + a_5) & (a_1 + a_4) & (a_0 + a_3) & a_2 & a_1 \\ a_4 & a_3 & (a_2 + a_5) & (a_1 + a_4) & (a_0 + a_3) & a_2 \\ a_5 & a_4 & a_3 & (a_2 + a_5) & (a_1 + a_4) & (a_0 + a_3) \end{bmatrix}.$$

We need 3 XOR gates to construct \mathbf{Z}_0 and \mathbf{Z}_3 . Furthermore, we need $m^2 = 36$ AND and $m(m - 1) = 30$ XOR gates to find the final result $c(x)$ via the product $\mathbf{c} = \mathbf{Zb}$. Hence, the computation requires 36 AND gates and $3 + 30 = 33$ XOR gates. The total delay is $(T_A + 4T_X)$.

When applying Method II, we first compute the column vector $\mathbf{Z}'\mathbf{b}$ as:

$$\mathbf{Z}'\mathbf{b} = \begin{bmatrix} a_0 & 0 & 0 & 0 & a_5 & a_4 \\ a_1 & a_0 & 0 & 0 & 0 & a_5 \\ a_2 & a_1 & a_0 & 0 & 0 & 0 \\ a_3 & a_2 & a_1 & a_0 & 0 & 0 \\ a_4 & a_3 & a_2 & a_1 & a_0 & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} a_0b_0 + a_5b_4 + a_4b_5 \\ a_1b_0 + a_0b_1 + a_5b_5 \\ a_2b_0 + a_1b_1 + a_0b_2 \\ a_3b_0 + a_2b_1 + a_1b_2 + a_0b_3 \\ a_4b_0 + a_3b_1 + a_2b_2 + a_1b_3 + a_0b_4 \\ a_5b_0 + a_4b_1 + a_3b_2 + a_2b_3 + a_1b_4 + a_0b_5 \end{bmatrix},$$

which requires 24 AND gates and 18 XOR gates. Then, we compute the column vector $\mathbf{Z}''\mathbf{b}$ as

$$\mathbf{Z}''\mathbf{b} = \begin{bmatrix} 0 & a_5 & a_4 & a_3 & a_2 & a_1 \\ 0 & 0 & a_5 & a_4 & a_3 & a_2 \\ 0 & 0 & 0 & a_5 & a_4 & a_3 \\ 0 & a_5 & a_4 & a_3 & a_2 & a_1 \\ 0 & 0 & a_5 & a_4 & a_3 & a_2 \\ 0 & 0 & 0 & a_5 & a_4 & a_3 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} a_5b_1 + a_4b_2 + a_3b_3 + a_2b_4 + a_1b_5 \\ a_5b_2 + a_4b_3 + a_3b_4 + a_2b_5 \\ a_5b_3 + a_4b_4 + a_3b_5 \\ a_5b_1 + a_4b_2 + a_3b_3 + a_2b_4 + a_1b_5 \\ a_5b_2 + a_4b_3 + a_3b_4 + a_2b_5 \\ a_5b_3 + a_4b_4 + a_3b_5 \end{bmatrix}.$$

There are only $\Delta = 3$ different rows in the vector $\mathbf{Z}''\mathbf{b}$, thus, its computation requires $5 + 4 + 3 = 12$ AND gates and $4 + 3 + 2 = 9$ XOR gates. Addition of this vector to the column vector $\mathbf{Z}'\mathbf{b}$ requires additional 6 XOR gates. Therefore, a total of $24 + 12 = 36$ AND and $18 + 9 + 6 = 33$ XOR gates are needed to compute the final result $c(x)$ directly using Method II. The total delay is $(T_A + 4T_X)$. Since $k = 2$, both methods require exactly the same number of gates and they have the same delay.

7 All-One-Polynomials

Let the irreducible polynomial be the AOP

$$p(x) = x^m + x^{(m-1)} + \dots + x + 1.$$

All rows, except the first one, in the lower submatrix can be reduced using the multiple

$$(x + 1)p(x) = x^{(m+1)} + 1.$$

This produces the partial result

$$\mathbf{M} = \begin{bmatrix} a_0 & 0 & a_{m-1} & a_{m-2} & \dots & a_3 & a_2 & a_1 \\ a_1 & a_0 & 0 & a_{m-1} & \dots & a_4 & a_3 & a_2 \\ a_2 & a_1 & a_0 & 0 & \dots & a_5 & a_4 & a_3 \\ a_3 & a_2 & a_1 & a_0 & \dots & a_6 & a_5 & a_4 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{m-3} & a_{m-4} & a_{m-5} & a_{m-6} & \dots & a_0 & 0 & a_{m-1} \\ a_{m-2} & a_{m-3} & a_{m-4} & a_{m-5} & \dots & a_1 & a_0 & 0 \\ a_{m-1} & a_{m-2} & a_{m-3} & a_{m-4} & \dots & a_2 & a_1 & a_0 \\ 0 & a_{m-1} & a_{m-2} & a_{m-3} & \dots & a_3 & a_2 & a_1 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{bmatrix}.$$

The first row in the lower submatrix needs to be reduced using the original irreducible polynomial $p(x)$. Either Method I or II can be used here.

7.1 Complexity Analysis

The number of AND gates required by both methods is $m^2 = k^2$. The delay is found as $T_A + (1 + \lceil \log_2 m \rceil) T_X$. Method II requires $m^2 - 1$ XOR gates since $\Delta = 1$. The number of XOR gates required by Method I depends on the choice of $p(x)$, however, Method I requires at least m^2 XOR gates, except when $p(x) = x^2 + x + 1$, in which case it requires $m^2 - \frac{m}{2} = 3$ XOR gates. Thus, Method II is the best method to utilize here.

7.2 An Example

Let the irreducible polynomial be the AOP

$$p(x) = x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1 .$$

Here we have $m = k = 10$ and $\Delta = 1$. The multiple $(x + 1)p(x)$ is equal to $(x + 1)p(x) = x^{11} + 1$. Method II computes the column vectors

$$\mathbf{Z}'\mathbf{b} = \begin{bmatrix} a_0 & 0 & a_9 & a_8 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 \\ a_1 & a_0 & 0 & a_9 & a_8 & a_7 & a_6 & a_5 & a_4 & a_3 \\ a_2 & a_1 & a_0 & 0 & a_9 & a_8 & a_7 & a_6 & a_5 & a_4 \\ a_3 & a_2 & a_1 & a_0 & 0 & a_9 & a_8 & a_7 & a_6 & a_5 \\ a_4 & a_3 & a_2 & a_1 & a_0 & 0 & a_9 & a_8 & a_7 & a_6 \\ a_5 & a_4 & a_3 & a_2 & a_1 & a_0 & 0 & a_9 & a_8 & a_7 \\ a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 & 0 & a_9 & a_8 \\ a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 & 0 & a_9 \\ a_8 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 & 0 \\ a_9 & a_8 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \end{bmatrix}$$

and

$$\begin{aligned} \mathbf{Z}''\mathbf{b} &= \begin{bmatrix} 0 & a_9 & a_8 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \end{bmatrix} \\ &= a_9b_1 + a_8b_2 + a_7b_3 + a_6b_4 + a_5b_5 + a_4b_6 + a_3b_7 + a_2b_8 + a_1b_9 . \end{aligned}$$

The computation of $\mathbf{Z}'\mathbf{b}$ requires $9 \cdot 9 + 10 = 91$ AND gates and $8 \cdot 9 + 9 = 81$ XOR gates. Similarly, the computation of $\mathbf{Z}''\mathbf{b}$ requires 9 AND gates and 8 XOR gates. The final result is obtained by adding the result of the product $\mathbf{Z}''\mathbf{b}$ to every row of the vector $\mathbf{Z}'\mathbf{b}$, which uses additional 10 XOR gates. Thus, the total number of gates required to find the result \mathbf{c} is found as 100 AND gates and 99 XOR gates.

The most time-consuming operation is the multiplication $\mathbf{Z}''\mathbf{b}$, which takes $(T_A + \lceil \log_2 10 \rceil T_X)$ units of delay, assuming that the binary tree method is used. The addition of $\mathbf{Z}''\mathbf{b}$ to this result takes another T_X time. Thus, the total delay of the circuit becomes $(T_A + 5T_X)$.

8 Conclusions

In this paper, we present a new architecture for the Mastrovito multiplication and rigorous analysis of the complexity for a general irreducible polynomial. Eleven years ago in the *6th International Conference on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*, E. D. Mastrovito presented his original idea for the bit-parallel multiplication algorithm in the finite field of 2^m elements. The Mastrovito multiplier for the special irreducible trinomial $x^m + x + 1$ is shown to require m^2 AND gates and $(m^2 - 1)$ XOR gates in the original work and elsewhere [7, 8, 12, 13]. However, the number of XOR gates required by the Mastrovito multiplier is further reduced when the irreducible polynomial generating field is an equally-spaced-trinomial of the form $x^m + x^{\frac{m}{2}} + 1$, in which case the XOR complexity is obtained as $(m^2 - \frac{m}{2})$, as shown in [14].

In this paper, we have several results:

- We give a rigorous analysis of the Mastrovito multiplier for a general irreducible polynomial, and show that it requires m^2 AND gates and $(m - 1)(m + k - 1) + \sum_{j \in \mathcal{S}^*} (2m - 1 - j)$ XOR gates, where \mathcal{S}^* is defined in §3.1
- We extend this analysis to certain types of polynomials. For example, we show that the Mastrovito multiplier reduction technique using the binomial $x^m + 1$ requires m^2 AND gates and $(m^2 - m)$ XOR gates. Since $x^m + 1$ is always reducible for $m \geq 2$, this case does not constitute a legitimate finite field multiplication, however, it serves as an intermediate step in the reduction process using other types of irreducible polynomials, for example, the AOPs
- We apply the proposed method to the irreducible trinomials $x^m + x^n + 1$, where $m - 1 \geq n \geq 2$ and $n \neq \frac{m}{2}$, and prove that it requires m^2 AND and $(m^2 - 1)$ XOR gates. This case provides an alternative method to that of [14] which also gives the same complexity values, therefore, strengthening the belief that the Mastrovito multiplier using these trinomials indeed requires a minimum of $(m^2 - 1)$ XOR gates.
- We apply the proposed method to the irreducible equally-spaced-polynomials, which are polynomials of the form $x^{k\Delta} + x^{(k-1)\Delta} + \dots + x^\Delta + 1$, where $m = k\Delta$. We show that the Mastrovito multiplier in this case requires m^2 AND and $(m^2 - \Delta)$ XOR gates. This is the most significant result of this paper. A special case of the equally-spaced-polynomials is the equally-spaced-trinomial $x^m + x^{\frac{m}{2}} + 1$, in which case, the Mastrovito multiplier requires $(m^2 - \frac{m}{2})$ XOR gates. This result was also obtained in [14] using an entirely different construction technique. Furthermore, this result is the lowest XOR complexity known to date for a Mastrovito multiplier using any irreducible reduction polynomial. We believe that this is an absolute lower bound for the Mastrovito multiplier.

Conjecture: The Mastrovito multiplier for the field $GF(2^m)$ requires at least m^2 AND gates and $(m^2 - \frac{m}{2})$ XOR gates for any irreducible reduction polynomial.

- We apply the proposed technique to the all-one-polynomials, and show that it requires m^2 AND and $(m^2 - 1)$ XOR gates. This result also matches the best known XOR complexity to date, as given in [5].

The XOR complexity results for the Mastrovito multiplier in $GF(2^m)$ are summarized in Table 1.

Table 1: The XOR complexity results for the Mastrovito multiplier.

Polynomial	XOR Complexity	Reference
Trinomial	$m^2 - 1$	[7] [8] [12] [13] [14] & This paper §5
EST	$m^2 - \frac{m}{2}$	[14] & This paper §6
AOP	$m^2 - 1$	[5] & This paper §7
General	$(m - 1)(m + k - 1) + \sum_{j \in \mathcal{S}^*} (2m - 1 - j)$	This paper §3
ESP	$m^2 - \Delta$	This paper §6

Finally, we note here that a shortened version of this paper (covering only the sections 1, 2, 3, and 8) was presented in the *13th Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*, in Honolulu, Hawaii, on November 15-19, 1999, and appeared in the conference proceedings [2].

References

- [1] G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.
- [2] A. Halbutoğulları and Ç. K. Koç. Mastrovito multiplier for general irreducible polynomials. In M. Fossorier, H. Imai, S. Lin, and A. Poli, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, Lecture Notes in Computer Science, No. 1719, pages 498–507. Springer, Berlin, Germany, 1999.
- [3] 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 Transactions on Computers*, 41(8):962–971, August 1992.
- [4] T. Itoh and S. Tsujii. Structure of parallel multipliers for a class of finite fields $GF(2^m)$. *Information and Computation*, 83:21–40, 1989.
- [5] Ç. K. Koç and B. Sunar. Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields. *IEEE Transactions on Computers*, 47(3):353–356, March 1998.
- [6] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and Their Applications*. Cambridge University Press, New York, NY, 1994.
- [7] E. D. Mastrovito. VLSI architectures for multiplication over finite field $GF(2^m)$. In T. Mora, editor, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, Lecture Notes in Computer Science, No. 357, pages 297–309. Springer, Berlin, Germany, 1988.
- [8] E. D. Mastrovito. *VLSI Architectures for Computation in Galois Fields*. PhD thesis, Linköping University, Department of Electrical Engineering, Linköping, Sweden, 1991.
- [9] A. J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, Boston, MA, 1993.
- [10] A. J. Menezes, I. F. Blake, X. Gao, R. C. Mullen, S. A. Vanstone, and T. Yaghoobian. *Applications of Finite Fields*. Kluwer Academic Publishers, Boston, MA, 1993.
- [11] J. Omura and J. Massey. Computational method and apparatus for finite field arithmetic. U.S. Patent Number 4,587,627, May 1986.
- [12] C. Paar. *Efficient VLSI Architectures for Bit Parallel Computation in Galois Fields*. PhD thesis, Universität GH Essen, VDI Verlag, 1994.
- [13] C. Paar. A new architecture for a paralel finite field multiplier with low complexity based on composite fields. *IEEE Transactions on Computers*, 45(7):856–861, July 1996.
- [14] B. Sunar and Ç. K. Koç. Mastrovito multiplier for all trinomials. *IEEE Transactions on Computers*, 48(5):522–527, May 1999.

Figure 1: The reduction of the lower submatrix.

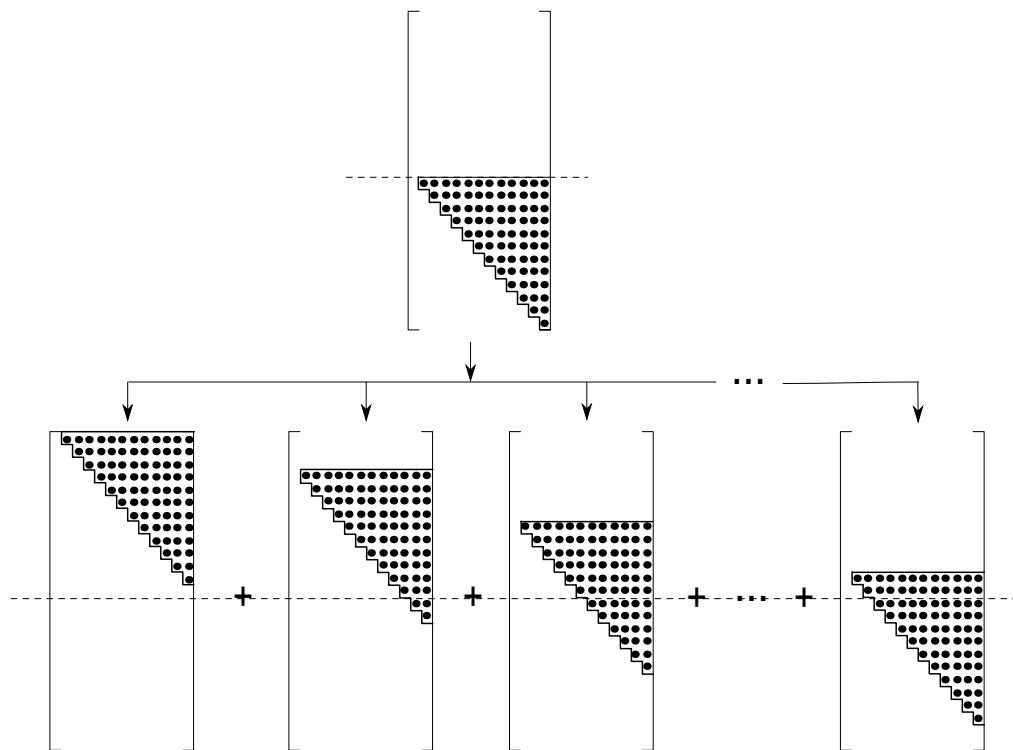


Figure 2: The separation of the upper and lower parts.

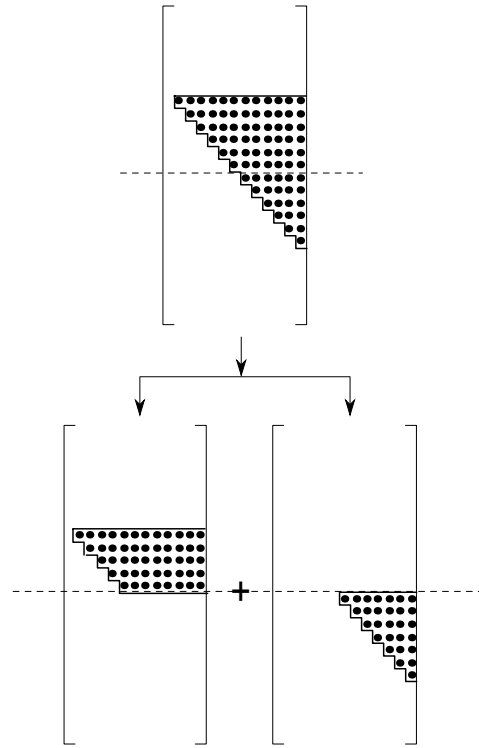


Figure 3: The accumulation of the lower parts.

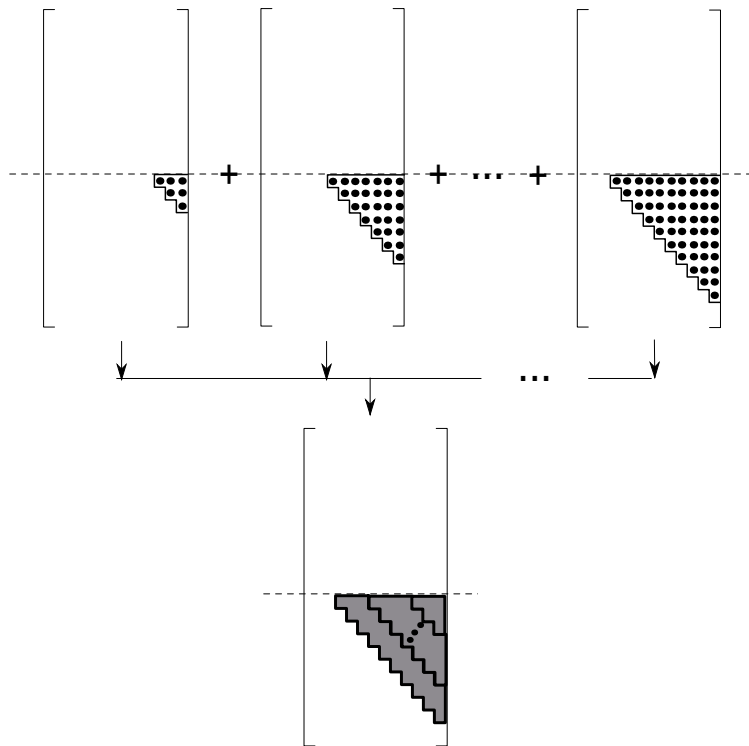


Figure 4: The total reduction process.

