Parallel Multipliers Based on Special Irreducible Pentanomials

Francisco Rodríguez-Henríquez, *Member*, *IEEE*, and Çetin Kaya Koç, *Senior Member*, *IEEE*

Abstract—The state-of-the-art Galois field $GF(2^m)$ multipliers offer advantageous space and time complexities when the field is generated by some special irreducible polynomial. To date, the best complexity results have been obtained when the irreducible polynomial is either a trinomial or an equally spaced polynomial (ESP). Unfortunately, there exist only a few irreducible ESPs in the range of interest for most of the applications, e.g., error-correcting codes, computer algebra, and elliptic curve cryptography. Furthermore, it is not always possible to find an irreducible trinomial of degree m in this range. For those cases where neither an irreducible trinomial nor an irreducible ESP exists, the use of irreducible pentanomials has been suggested. Irreducible pentanomials are abundant, and there are several eligible candidates for a given m. In this paper, we promote the use of two special types of irreducible pentanomials. We propose new Mastrovito and dual basis multiplier architectures based on these special irreducible pentanomials and give rigorous analyses of their space and time complexity.

Index Terms—Finite fields arithmetic, parallel multipliers, pentanomials, multipliers for $GF(2^m)$.

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 elliptic curve cryptosystems [9], [10]. For these implementations, the measure of efficiency is the space complexity, i.e., the number of XOR and AND gates, and the time complexity, i.e., the total gate delay of the circuit. The representation of the field elements plays a crucial role in the efficiency of the architectures for the arithmetic operations. Several architectures have been reported for multiplication in $GF(2^m)$. For example, efficient bit-parallel multipliers for both polynomial and normal basis representation have been proposed [4], [12], [6], including the Mastrovito multiplier [15].

Another technique, which was first suggested in [1], is known as the dual basis multiplier [11], [2], [17], [18]. Conventional dual basis multipliers have the property that one of the input operands is given in the polynomial basis while the other input is in the dual basis. The product is then obtained in the dual basis [1]. In this paper, we use a new approach for dual basis multipliers that was suggested in [13]. In contrast to the conventional approach, the technique proposed in [13] assumes that both operands are given in the polynomial basis. This assumption yields less time and space complexity for certain types of irreducible polynomials.

- *F. Rodríguez-Henríquez is with CINVESTAV-IPN, Depto. de Ingeniería Eléctrica, Sección de computación, Av. Instituto Politécnico Nacional No. 2508, Col San Pedro Zacatenco, México, D.F. 07300. E-mail: francisco@delta.cs.cinvestav.mx.*
- *Ç.K. Koç is with the Department of Electrical and Computer Engineering, Oregon State University, Corvallis, OR 97331. E-mail: koc@ece.orst.edu.*

Manuscript received 7 Jan. 2000; revised 14 Nov. 2001; accepted 26 July 2002. For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 111185.

In all of these state-of-the-art techniques for finite field $GF(2^m)$ multipliers, less space and time complexity have been reported when the irreducible polynomial used to construct the field is either an equally spaced polynomial defined as:

$$p(x) = x^m + x^{(k-1)d} + \cdots + x^{2d} + x^d + 1, \tag{1}$$

where $m = kd$, or a trinomial [7], [8], [17], [15], [3]. Unfortunately, irreducible equally spaced polynomials (ESP) are very rare. There are only 81 m values less than 1,024 such that an irreducible ESP of degree m exists [17].

On the other hand, an irreducible trinomial does not exist for every value of m. In fact, there are 468 m values less than 1,024 such that an irreducible trinomial of degree m does not exist [14]. Since, in finite fields of characteristic 2, an irreducible polynomial must have an odd number of nonzero coefficients, the next option is to use irreducible pentanomials. It has been suggested [5] that an irreducible pentanomial can be used whenever there does not exist an irreducible trinomial of degree m. This is a good, practical suggestion since there exists either an irreducible trinomial or pentanomial of degree $m \in [2, 10,000]$ as was established by enumeration in [14]. In fact, there is no known value of m for which either an irreducible trinomial or pentanomial does not exist [14]. Therefore, the design of multipliers using irreducible pentanomials is of practical importance, particularly for cryptographic applications, and efforts to obtain efficient implementations are well justified. This work is a step in this direction.

In this paper, we study the time and space complexity for multipliers in $GF(2^m)$ generated by using certain special classes of irreducible pentanomials. We consider the following types of irreducible pentanomials, which we name arbitrarily as type 1 and type 2 pentanomials:

$$\begin{aligned} &\text{Type 1}: x^m + x^{n+1} + x^n + x + 1, \\ &\qquad \text{where } 2 \le n \le \lfloor m/2 \rfloor - 1. \\ &\text{Type 2}: x^m + x^{n+2} + x^{n+1} + x^n + 1, \\ &\qquad \text{where } 1 \le n \le \lfloor m/2 \rfloor - 1. \end{aligned} \tag{2}$$

There are many values of m for which an irreducible pentanomial of these types exists: There are 416 m values less than 515 such that an irreducible type 1 pentanomial of degree m exists. Furthermore, there are 304 m values less than 526 such that an irreducible type 2 pentanomial exists. Thus, pentanomials type I and type II are abundant and, as we will see, they offer advantageous design options for Mastrovito and dual basis multipliers, respectively.

In this paper, we present efficient architectures for two different types of multipliers: the Mastrovito and the dual basis multipliers. We give rigorous analyses of these multipliers in terms of their space and time complexity. In Section 2, we introduce efficient Mastrovito multipliers based on the aforementioned type I irreducible pentanomial, and give their complexity analyses. We then introduce efficient dual basis multipliers in Section 3, based on the methodology proposed in [13]. The analyses of the dual basis multiplier used with the special pentanomials type II are given in Section 4. Finally, we summarize the findings of this research and give a comparative analysis of similar multipliers in Section 5.

2 Mastrovito Multipliers and Their Analysis

The algorithms for multipliers in $GF(2^m)$ based on the polynomial basis usually consist of two steps: the polynomial multiplication and the modular reduction. Let $A(x)$, $B(x)$, and $C'(x)$ be elements of $GF(2^m)$ and $P(x)$ be the degree m irreducible polynomial defining the field $GF(2^m)$. In order to compute $C'(x) = A(x)B(x) \bmod P(x)$, we first obtain the product polynomial $C(x)$ which is of degree at most $2m-2$ using

$$C(x) = A(x)B(x) = \left(\sum_{i=0}^{m-1} a_i x^i\right)\left(\sum_{i=0}^{m-1} b_i x^i\right). \tag{3}$$

Then, the reduction operation is performed in order to obtain the $(m-1)$-degree polynomial $C'(x)$, which is defined as

$$C'(x) = C(x) \bmod P(x). \tag{4}$$

Once the irreducible polynomial $P(x)$ is selected and fixed, the reduction step can be accomplished using only XOR gates. The Mastrovito algorithm formulates these two steps into a single matrix-vector product and then reduces the product matrix using the irreducible polynomial defining the field.

We propose an architecture for computation of the final product $C'(x)$ in (4) by first computing the product to obtain the vector C which has $2m-1$ elements. By using a standard matrix-vector product, it can be shown that C can be computed with a total space and time complexity given as:

$$\begin{aligned} \text{AND Gates} &= m^2 \\ \text{XOR Gates} &= (m-1)^2 \\ \text{Total Delay} &= T_A + \lceil \log_2 m \rceil T_X. \end{aligned} \tag{5}$$

In order to obtain the final product after the reduction in (4), we need to use the irreducible polynomial defining the field. The complexity of this computation is determined by the properties of the irreducible polynomial. The complexity results for several types of irreducible polynomials have been obtained [7], [8], [12], [15], [3]. Below, we derive the space and time complexity for irreducible type 1 pentanomials using Mastrovito multipliers.

2.1 Type 1 Pentanomials

Let the field $GF(2^m)$ be constructed using the irreducible type 1 pentanomial defined in (2). In order to obtain the final product $C'(x)$, we compute the reduction array as defined in [15]. We use the property $P(\alpha) = 0$ and write

$$\begin{aligned} \alpha^m &= 1 + \alpha + \alpha^n + \alpha^{n+1} \\ \alpha^{m+1} &= \alpha + \alpha^2 + \alpha^{n+1} + \alpha^{n+2} \\ \alpha^{m+2} &= \alpha^2 + \alpha^3 + \alpha^{n+2} + \alpha^{n+3} \\ &\vdots \\ \alpha^{2m-n-2} &= \alpha^{m-n-2} + \alpha^{m-n-1} + \alpha^{m-2} + \alpha^{m-1} \\ \alpha^{2m-n-1} &= \alpha^{m-n-1} + \alpha^{m-n} + \alpha^{m-1} + 1 + \alpha + \alpha^n \\ &\quad + \alpha^{n+1} \\ \alpha^{2m-n} &= \alpha^{m-n} + \alpha^{m-n+1} + 1 + \alpha + \alpha^n + \alpha^{n+1} + \alpha \\ &\quad + \alpha^2 + \alpha^{n+1} + \alpha^{n+2} \\ &= \alpha^{m-n} + \alpha^{m-n+1} + 1 + \alpha^n + \alpha^2 + \alpha^{n+2} \\ \alpha^{2m-n+1} &= \alpha^{m-n+1} + \alpha^{m-n+2} + \alpha + \alpha^{n+1} + \alpha^3 \\ &\quad + \alpha^{n+3} \\ \alpha^{2m-n+2} &= \alpha^{m-n+2} + \alpha^{m-n+3} + \alpha^2 + \alpha^{n+2} + \alpha^4 + \alpha^{n+4} \\ &\vdots \\ \alpha^{2m-3} &= \alpha^{m-3} + \alpha^{m-2} + \alpha^{n-3} + \alpha^{2n-3} + \alpha^{n-1} + \alpha^{2n-1} \\ \alpha^{2m-2} &= \alpha^{m-2} + \alpha^{m-1} + \alpha^{n-2} + \alpha^{2n-2} + \alpha^n + \alpha^{2n}. \end{aligned}$$

The above equations can be summarized based on their number of operands as follows:

$$\alpha^{m+i} = \begin{cases} \alpha^i + \alpha^{i+1} + \alpha^{n+i} + \alpha^{n+i+1} & \text{for } i = 0, 1, \ldots, m-n-2 \\ \alpha^i + \alpha^{i+1} + \alpha^{n+i} + 1 + \alpha + \alpha^n + \alpha^{n+1} & \text{for } i = m-n-1 \\ \alpha^i + \alpha^{i+1} + \alpha^{i-(m-n)} + \alpha^{i-m+2n} + \alpha^{i-(m-n)+2} + \alpha^{i-m+2n+2} & \text{for } i = m-n, m-n+1, \ldots, m-2. \end{cases} \tag{6}$$

In order to obtain the coordinates of the product C' as given by (4), we follow the method in [15]. From the matrix representation shown above, we just need to add the nonzero elements of each one of the m columns. For instance, in order to obtain the first coordinate c'_0, we just need to add the nonzero coefficients of the first column to

TABLE 1
The Coordinates in (7) as Classified by the Number of Operands

Coordinates	Number of Equations	Number of Operands	XOR Gates
c'_0	1	4	3
$c'_1 \cdots c'_{n-2}$	$n-2$	5	4
c'_{n-1}	1	4	3
$c'_n \cdots c'_{2n-2}$	$n-1$	7	6
c'_{2n-1}	1	6	5
c'_{2n}	1	6	5
$c'_{2n+1} \cdots c'_{m-2}$	$m-2n-2$	5	4
c'_{m-1}	1	4	3

the first coordinate of the product polynomial c_0. We can see that the nonzero elements for the first column of the matrix are the coordinates c_m, c_{2m-n-1}, and c_{2m-n} added to the c_0 coordinate, giving the first coordinate as

$$c'_0 = c_0 + c_m + c_{2m-n-1} + c_{2m-n}.$$

The entire set of coordinates of C' are obtained as follows:

$$\begin{aligned}
c'_0 &= c_0 + c_m + c_{2m-n-1} + c_{2m-n} \\
c'_1 &= c_1 + c_m + c_{m+1} + c_{2m-n-1} + c_{2m-n+1} \\
c'_2 &= c_2 + c_{m+1} + c_{m+2} + c_{2m-n} + c_{2m-n+2} \\
&\vdots \\
c'_{n-2} &= c_{n-2} + c_{m+n-3} + c_{m+n-2} + c_{2m-4} + c_{2m-2} \\
c'_{n-1} &= c_{n-1} + c_{m+n-2} + c_{m+n-1} + c_{2m-3} \\
c'_n &= c_n + c_m + c_{m+n-1} + c_{m+n} + c_{2m-n-1} + c_{2m-n} + c_{2m-2} \\
c'_{n+1} &= c_{n+1} + c_m + c_{m+1} + c_{m+n} + c_{m+n+1} + c_{2m-n-1} \\
&\quad + c_{2m-n+1} \\
&\vdots \\
c'_{2n-2} &= c_{2n-2} + c_{m+n-3} + c_{m+n-2} + c_{m+2n-3} \\
&\quad + c_{m+2n-2} + c_{2m-4} + c_{2m-2} \\
c'_{2n-1} &= c_{2n-1} + c_{m+n-2} + c_{m+n-1} + c_{m+2n-2} + c_{m+2n-1} + c_{2m-3} \\
c'_{2n} &= c_{2n} + c_{m+n-1} + c_{m+n} + c_{m+2n-1} + c_{m+2n} + c_{2m-2} \\
c'_{2n+1} &= c_{2n+1} + c_{m+n} + c_{m+n+1} + c_{m+2n} + c_{m+2n+1} \\
c'_{2n+2} &= c_{2n+2} + c_{m+n+1} + c_{m+n+2} + c_{m+2n+1} + c_{m+2n+2} \\
&\vdots \\
c'_{m-2} &= c_{m-2} + c_{2m-n-3} + c_{2m-n-2} + c_{2m-3} + c_{2m-2} \\
c'_{m-1} &= c_{m-1} + c_{2m-n-2} + c_{2m-n-1} + c_{2m-2}.
\end{aligned} \tag{7}$$

In order to obtain the space and time complexities in the computation of (7), we can classify these equations according to their number of operands, as shown in Table 1.

Therefore, the total number of XOR gates needed to obtain all coordinates of the product C' is obtained as:

$$\begin{aligned}
&3 + 4(n-2) + 3 + 6(n-1) + 5 + 5 + 4(m-2n-2) + 3 \\
&= 4m + 2n - 3.
\end{aligned}$$

However, taking advantage of the inherent redundancy of the set of equations in (7), this number can be reduced further. For example, we need exactly three XOR gates to compute c'_{m-1}. Then, in the computation of c'_{m-2}, we notice a redundancy since two of the operands of this coordinate have been already added in the previous computation, allowing us to save a single XOR gate. Examining the equations in (7) more closely, we observe that the coordinates between c'_{n+1} and c'_{m-2} have the following structure:

$$\begin{aligned}
c'_{n+i} &= c_{m+i-1} + c_{m+i} + c_{m+n+i-1} + c_{m+n+i} + \cdots \\
c'_{n+i+1} &= c_{m+i} + c_{m+i+1} + c_{m+n+i} + c_{m+n+i+1} + \cdots \\
c'_{n+i+2} &= c_{m+i+1} + c_{m+i+2} + c_{m+n+i+1} + c_{m+n+i+2} + \cdots
\end{aligned}$$

for $i = 1, 2, \ldots, m-n-4$. Clearly, we can take advantage of the redundancy of this structure by using the term $c_{m+i} + c_{m+n+i}$ twice, in the computation of c'_{n+i} and c'_{n+i+1}. Also, we can use the term $c_{m+i+1} + c_{m+n+i+1}$ twice, in the computation of c'_{n+i+1} and c'_{n+i+2}. Applying this strategy to the whole range of coordinates from c'_{n+1} and c'_{m-2}, we can save a single XOR gate in the computation of each coordinate. This implies that we can save a total of $m-2-(n+1)+1 = m-n-2$ XOR gates. Also notice that the term $c_{2m-n-1} + c_{2m-n}$ appears in the equations for the coordinates c'_0 and c'_n and, furthermore, the term $c_{2m-n-1+i} + c_{2m-n+i}$ appears in the equations for the coordinates c'_i and c'_{n+i}, for $i = 0, 1, \cdots, n-1$. Hence, we can save n XOR gates for those $2n$ coordinate equations. These two strategies together yield a total saving of $m-n-2+n = m-2$ XOR gates. Thus, the complete set of the coordinates c'_i in (7) can be obtained using only

$$4m + 2n - 3 - (m-2) = 3m + 2n - 1$$

XOR gates. On the other hand the gate delay depends on the largest number of terms to be added, which is equal to 6 as seen in Table 1, giving the gate delay as

$$\lceil \log_2(6) \rceil T_X = 3T_X.$$

Therefore, we obtain the total complexity of the Mastrovito multipler based on the irreducible type 1 pentanomial as:

$$\begin{aligned}
&\text{AND Gates} = m^2 \\
&\text{XOR Gates} = (m-1)^2 + 3m + 2n - 1 = m^2 + m + 2n \\
&\text{Total Delay} = T_A + (3 + \lceil \log_2 m \rceil) T_X.
\end{aligned} \tag{8}$$

In the case of $n = 2$, i.e., when the irreducible polynomial is given as $x^m + x^3 + x^2 + x + 1$, a small improvement in the time and space complexities can be obtained.

As we derived, the XOR complexity of the reduction step was $3m + 2n - 1$. By taking $n = 2$, we obtain the XOR complexity as $3m + 3$. However, a small improvement can be obtained for this case. By examining (7), we observe that the coordinate c'_n with $n = 2$ is given by:

$$\begin{aligned} c'_{n=2} &= c_n + c_m + c_{m+n-1} + c_{m+n} + c_{2m-n-1} + c_{2m-n} + c_{2m-2} \\ &= c_n + c_m + c_{m+n-1} + c_{m+n} + c_{2m-n-1}, \end{aligned}$$

yielding an extra saving of two XOR gates. Also, notice that the term $c_m + c_{2m-3}$ is present in three equations for the coordinates c'_0, c'_1, and c'_2. By computing this term once and reusing it as needed, we obtain an extra saving of two XOR gates. In summary, $3m + 3 - 4 = 3m - 1$ XOR gates are sufficient in the reduction step. This gives the XOR complexity of the proposed multiplier for these irreducible special pentanomials as $(m-1)^2 + 3m - 1 = m^2 + m$. Therefore, the total complexity result for this case is

$$\begin{aligned} &\text{AND Gates} = m^2 \\ &\text{XOR Gates} = m^2 + m \\ &\text{Total Delay} = T_A + (3 + \lceil \log_2 m \rceil) T_X. \end{aligned} \tag{9}$$

3 Dual Basis Multiplication

In this section, we briefly describe the dual basis multiplication algorithm proposed in [13]. A set of m elements $\{\beta_0, \beta_1, \beta_2, \ldots, \beta_{m-1}\}$ forms a basis for $GF(2^m)$ if the β_is are linearly independent over the field $GF(2)$. Let $p(x)$ be a degree-m polynomial, irreducible over $GF(2)$. Also let α be a root of $p(x)$, i.e., $p(\alpha) = 0$. Then, the set $\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\}$ is a basis for $GF(2^m)$ and is called the polynomial (canonical) basis of the field [9]. An element $A \in GF(2^m)$ is expressed in this basis as $A = \sum_{i=0}^{m-1} a_i \alpha^i$. The trace of $\beta \in GF(2^m)$ relative to the subfield $GF(2)$ is defined by:

$$\mathrm{Tr}(\beta) = \sum_{i=0}^{m-1} \beta^{2^i}. \tag{10}$$

It is well-known [9] that the trace function is a linear mapping from the finite field $GF(2^m)$ onto the finite field $GF(2)$. Let $\{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{m-1}\}$ and $\{\beta_0, \beta_1, \beta_2, \ldots, \beta_{m-1}\}$ be any two bases for $GF(2^m)$ and also let $\gamma \in GF(2^m)$ with $\gamma \neq 0$. Then, these two bases are said to be *dual* with respect to γ if [2]:

$$\mathrm{Tr}(\gamma \alpha_i \beta_j) = \begin{cases} 1 & \text{if} \quad i = j, \\ 0 & \text{if} \quad i \neq j. \end{cases} \tag{11}$$

Let γ be a fixed nonzero element of the field $GF(2^m)$ and let $\{\beta_0, \beta_1, \beta_2, \ldots, \beta_{m-1}\}$ be a dual basis of $\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\}$, the polynomial basis previously defined. Then, any element A can be expressed either in the polynomial basis or in the dual basis as:

$$A = \sum_{i=0}^{m-1} a_i \alpha^i = \sum_{i=0}^{m-1} a_i^* \beta_i. \tag{12}$$

Using (11), we can obtain the jth coordinate of the element A in the dual basis as:

$$\mathrm{Tr}(\gamma \alpha^j A) = \mathrm{Tr}\left(\gamma \alpha^j \sum_{i=0}^{m-1} a_i^* \beta_i\right) = \sum_{i=0}^{m-1} a_i^* Tr(\gamma \alpha^j \beta_i) = a_j^*. \tag{13}$$

The algorithm proposed in [13] takes the input operands A and B in the polynomial basis, and computes the product C^* in the dual basis with respect to γ. This is in contrast to the standard definition of the dual basis multiplication, where one of the input operands needs to be represented in the dual basis. In the rest of this section, we give a brief description of that algorithm.

Let $A, B \in GF(2^m)$ be given in the polynomial basis as $A = \sum_{i=0}^{m-1} a_i \alpha^i$ and $B = \sum_{i=0}^{m-1} b_i \alpha^i$, where $a_i, b_i \in GF(2)$ are their coordinates, respectively. Given a fixed element $\gamma \in GF(2^m)$, we are interested in computing the product C^* in the dual basis with respect to γ, given as,

$$C^* = \sum_{k=0}^{m-1} c_k^* \beta_k. \tag{14}$$

Using e(13), the coefficient c_k^* is given by $c_k^* = \mathrm{Tr}(\gamma \alpha^k C) = \mathrm{Tr}(\gamma \alpha^k AB)$ for $k = 0, 1, \ldots, (m-1)$ as

$$\begin{aligned} c_k^* &= \mathrm{Tr}\left(\gamma \alpha^k \left(\sum_{i=0}^{m-1} a_i \alpha^i\right)\left(\sum_{j=0}^{m-1} b_j \alpha^j\right)\right) \\ &= \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \mathrm{Tr}(\gamma \alpha^{i+j+k}) b_j a_i. \end{aligned} \tag{15}$$

Thus, the coefficient c_k^* can be written as:

$$c_k^* = \sum_{i=0}^{m-1} t_{i+k} a_i, \tag{16}$$

where the trace coefficients t_{i+k} for $i, k = 0, 1, \ldots, (m-1)$ are defined by:

$$t_{i+k} = \sum_{j=0}^{m-1} \mathrm{Tr}\big(\gamma \alpha^{i+j+k}\big) b_j. \tag{17}$$

Therefore, the field product C^* can be expressed as a matrix-vector product:

$$C^* = \begin{bmatrix} c_0^* \\ c_1^* \\ c_2^* \\ \vdots \\ c_{m-2}^* \\ c_{m-1}^* \end{bmatrix} = \begin{bmatrix} t_0 & t_1 & t_2 & \cdots & t_{m-2} & t_{m-1} \\ t_1 & t_2 & t_3 & \cdots & t_{m-1} & t_m \\ t_2 & t_3 & t_4 & \cdots & t_m & t_{m+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ t_{m-2} & t_{m-1} & t_m & \cdots & t_{2m-4} & t_{2m-3} \\ t_{m-1} & t_m & t_{m+1} & \cdots & t_{2m-3} & t_{2m-2} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{m-2} \\ a_{m-1} \end{bmatrix}. \tag{18}$$

Each row of the multiplication matrix in (18) corresponds to a state of the shift register in Berlekamp's bit-serial multiplier of [1], holding the dual basis factor γ.

Provided that the trace coefficients t_k for $k = 0, 1, \ldots, (2m-2)$ are all available, the space and time complexities for computing the matrix-vector product in (18) are obtained as:

$$\begin{aligned} &\text{AND Gates} = m^2, \\ &\text{XOR Gates} = m^2 - m, \\ &\text{Total Delay} = T_A + \lceil \log_2 m \rceil T_X. \end{aligned} \tag{19}$$

On the other hand, from (17) we see that, in order to obtain all $(2m-1)$ trace coefficients required in (18), we need to compute a total of $(3m-2)$ different traces. This can be accomplished by using the following transformation matrix of dimension $(2m-1) \times m$, which we will call the *extended Gram matrix*:

$$\begin{bmatrix} t_0 \\ t_1 \\ t_2 \\ \vdots \\ t_{m-1} \\ \hline t_m \\ t_{m+1} \\ \vdots \\ t_{2m-2} \end{bmatrix} = \begin{bmatrix} \mathrm{Tr}(\gamma) & \mathrm{Tr}(\gamma\alpha) & \mathrm{Tr}(\gamma\alpha^2) & \cdots & \mathrm{Tr}(\gamma\alpha^{m-1}) \\ \mathrm{Tr}(\gamma\alpha) & \mathrm{Tr}(\gamma\alpha^2) & \mathrm{Tr}(\gamma\alpha^3) & \cdots & \mathrm{Tr}(\gamma\alpha^{m}) \\ \mathrm{Tr}(\gamma\alpha^2) & \mathrm{Tr}(\gamma\alpha^3) & \mathrm{Tr}(\gamma\alpha^4) & \cdots & \mathrm{Tr}(\gamma\alpha^{m+1}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathrm{Tr}(\gamma\alpha^{m-1}) & \mathrm{Tr}(\gamma\alpha^{m}) & \mathrm{Tr}(\gamma\alpha^{m+1}) & \cdots & \mathrm{Tr}(\gamma\alpha^{2m-2}) \\ \hline \mathrm{Tr}(\gamma\alpha^{m}) & \mathrm{Tr}(\gamma\alpha^{m+1}) & \mathrm{Tr}(\gamma\alpha^{m+2}) & \cdots & \mathrm{Tr}(\gamma\alpha^{2m-1}) \\ \mathrm{Tr}(\gamma\alpha^{m+1}) & \mathrm{Tr}(\gamma\alpha^{m+2}) & \mathrm{Tr}(\gamma\alpha^{m+3}) & \cdots & \mathrm{Tr}(\gamma\alpha^{2m}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathrm{Tr}(\gamma\alpha^{2m-2}) & \mathrm{Tr}(\gamma\alpha^{2m-1}) & \mathrm{Tr}(\gamma\alpha^{2m}) & \cdots & \mathrm{Tr}(\gamma\alpha^{3m-3}) \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{m-1} \end{bmatrix}. \tag{20}$$

The matrix-vector equation in (20) provides a method to compute the remaining trace coefficients required in (18) by using only the coordinates of the operand B in the polynomial basis. The space complexity for computing all trace coefficients defined in (20) depends only on the number of nonzero entries in the extended Gram matrix, which is a function of the irreducible polynomial $p(x)$ defining the field and the element $\gamma \in GF(2^m)$. Once the parameter γ is fixed, the elements of the extended Gram matrix are fixed zero and one values. Thus, the trace coefficients in (20) can be computed using only XOR gates, i.e., no AND gates are required. A good selection of γ is crucial in order to obtain an extended Gram matrix with as few ones as possible.

The total complexity of the proposed multiplier consists of two parts:

- The space complexity for computing all $(2m-1)$ trace coefficients which are defined in (17) or (20) and used in (18). The first m trace coefficients are simply equal to the coordinates of the operand B expressed in the dual basis with respect to the selected element $\gamma \in GF(2^m)$. The remaining $(m-1)$ coefficients are determined using the extended Gram matrix given by (20).

TABLE 2
The Trace Coefficients in (23) Classified by the Number of Operands

Trace Coefficients	Number of Equations	Number of Operands	XOR Gates
t_m	1	3	2
$t_{m+1} \cdots t_{2m-n-4}$	$m-n-4$	4	3
$t_{2m-n-3} \cdots t_{2m-n-2}$	2	5	4
$t_{2m-n-1} \cdots t_{2m-n}$	2	6	5
$t_{m-n+1} \cdots t_{2m-2}$	$n-2$	7	6

- The complexity of computing the matrix-vector product in (18), which was established in (19), assuming that the coordinates of the operand A expressed in the polynomial basis and all $(2m-1)$ trace coefficients are given.

4 Analysis of Dual Basis Multipliers for Irreducible Pentanomials

In this section, we analyze the complexity of the trace coefficient computation for the dual basis multiplier, as defined in (20), using certain types of irreducible pentanomials as generating polynomials of the field $GF(2^m)$.

4.1 Type 2 Pentanomials

Let the field $GF(2^m)$ be constructed using the irreducible pentanomial $P(x) = x^m + x^{n+2} + x^{n+1} + x^n + 1$, where $2 \le n \le \lceil m/2 \rceil - 1$. It has been shown [2], [11] that there exists a γ such that the dual basis of the polynomial basis $\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\}$ is given as:

$$\{1+\alpha^n, \alpha^{n-1}, \alpha^{n-2}, \ldots, 1, \alpha^{m-1}, \alpha^{m-2}, \ldots, \alpha^{n+2}, \alpha^{m-1}+\alpha^{n+1}\}. \tag{21}$$

Therefore, the trace coefficients t_k for $k = 0, 1, \ldots, (m-1)$ are obtained directly from the polynomial basis coordinates of the operand B using these relations:

$$\begin{aligned} t_0 &= b_0^* = b_0 + b_n \\ t_k &= b_k^* = b_{n-k} \text{ for } k = 1, 2, \ldots, n \\ t_k &= b_k^* = b_{m+n-k} \text{ for } k = n+1, n+2, \ldots, m-2 \\ t_{m-1} &= b_{m-1}^* = b_{m-1} + b_{n+1}. \end{aligned} \tag{22}$$

In order to obtain the remaining trace coefficients t_k for $k = m, m+1, \ldots, (2m-2)$, we use the property $P(\alpha) = 0$ and write

$$\begin{aligned} \alpha^m &= 1 + \alpha^n + \alpha^{n+1} + \alpha^{n+2} \\ \alpha^{m+1} &= \alpha + \alpha^{n+1} + \alpha^{n+2} + \alpha^{n+3} \\ \alpha^{m+2} &= \alpha^2 + \alpha^{n+2} + \alpha^{n+3} + \alpha^{n+4} \\ &\vdots \\ \alpha^{2m-n-3} &= \alpha^{m-n-3} + \alpha^{m-3} + \alpha^{m-2} + \alpha^{m-1} \\ \alpha^{2m-n-2} &= \alpha^{m-n-2} + \alpha^{m-2} + \alpha^{m-1} + \alpha^m \\ &\vdots \\ \alpha^{2m-2} &= \alpha^{m-2} + \alpha^{m+n-2} + \alpha^{m+n-1} + \alpha^{m+n}. \end{aligned}$$

Due to the linearity property of the trace function and using (22), we obtain:

$$\begin{aligned} t_m &= t_0 + t_n + t_{n+1} + t_{n+2} = b_n + b_{m-1} + b_{m-2} \\ t_{m+1} &= t_1 + t_{n+1} + t_{n+2} + t_{n+3} = b_{n-1} + b_{m-1} + b_{m-2} + b_{m-3} \\ t_{m+2} &= t_2 + t_{n+2} + t_{n+3} + t_{n+4} = b_{n-2} + b_{m-2} + b_{m-3} + b_{m-4} \\ &\vdots \\ t_{m+n} &= t_n + t_{2n} + t_{2n+1} + t_{2n+2} = b_0 + b_{m-n} + b_{m-n-1} \\ &\quad + b_{m-n-2} \\ t_{m+n+1} &= t_{n+1} + t_{2n+1} + t_{2n+2} + t_{2n+3} = b_{m-1} + b_{m-n-1} \\ &\quad + b_{m-n-2} + b_{m-n-3} \\ &\vdots \\ t_{2m-n-3} &= t_{m-n-3} + t_{m-3} + t_{m-2} + t_{m-1} = b_{2n+3} + b_{n+3} \\ &\quad + b_{n+2} + b_{n+1} + b_{m-1} \\ t_{2m-n-2} &= t_{m-n-2} + t_{m-2} + t_{m-1} + t_m = b_{2n+2} + b_{n+2} + b_{n+1} \\ &\quad + b_n + b_{m-2} \\ t_{2m-n-1} &= t_{m-n-1} + t_{m-1} + t_m + t_{m+1} = b_{2n+1} + b_{n+1} + b_{m-1} \\ &\quad + b_n + b_{n-1} + b_{m-3} \\ t_{2m-n} &= t_{m-n} + t_m + t_{m+1} + t_{m+2} = b_{2n} + b_n + b_{n-1} + b_{n-2} \\ &\quad + b_{m-2} + b_{m-4} \\ t_{2m-n+1} &= t_{m-n+1} + t_{m+1} + t_{m+2} + t_{m+3} = b_{2n-1} + b_{n-1} \\ &\quad + b_{m-1} + b_{n-2} + b_{n-3} + b_{m-3} + b_{m-5} \\ t_{2m-n+2} &= t_{m-n+2} + t_{m+2} + t_{m+3} + t_{m+4} = b_{2n-2} + b_{n-2} \\ &\quad + b_{m-2} + b_{n-3} + b_{n-4} + b_{m-4} + b_{m-6} \\ &\vdots \\ t_{2m-2} &= t_{m-2} + t_{m+n-2} + t_{m+n-1} + t_{m+n} \\ &= b_{n+2} + b_2 + b_{m-n+2} + b_1 + b_0 + b_{m-n} + b_{m-n-2}. \end{aligned} \tag{23}$$

Some intermediate steps to derive the final $m-1$ equations are not explicity shown above. These equations can be classified by their number of operands, as shown in Table 2, which shows the number of XOR gates needed to implement each one of the trace equations based on the number of operands.

The first m trace coefficients are obtained from the polynomial basis coordinates of the operand B using the transformation given by (22). This computation is performed using rewiring in all coefficients except the first and the last one. Hence, we only need two XOR gates to obtain this first block of traces. Therefore, the total number of XOR gates needed to obtain all the $2m-1$ trace coefficients from t_i for $i = 0, 1, \ldots, 2m-2$ is given as:

TABLE 3
The Summary of the Complexity Results

Irreducible Polynomial	XOR Gates	Gate Delays	References
$x^m + x + 1$	$m^2 - 1$	$T_A + (1 + \lceil \log_2 m \rceil)T_X$	[7] [8] [12]
$x^m + x^n + 1$	$m^2 - 1$	$T_A + (2 + \lceil \log_2 m \rceil)T_X$	[15]
$x^m + x^{\frac{m}{2}} + 1$	$m^2 - \frac{m}{2}$	$T_A + (1 + \lceil \log_2 m \rceil)T_X$	[15]
$x^m + x^{(k-1)d} + \cdots + x^d + 1$	$m^2 - d$	$T_A + (1 + \lceil \log_2 m \rceil)T_X$	[3]
$x^m + x^{m-1} + \cdots + x + 1$	$m^2 - 1$	$T_A + (1 + \lceil \log_2 m \rceil)T_X$	[3]
$x^m + x^{n+1} + x^n + x + 1$	$m^2 + m + 2n$	$T_A + (3 + \lceil \log_2 m \rceil)T_X$	Mastrovito
$x^m + x^3 + x^2 + x + 1$	$m^2 + m$	$T_A + (3 + \lceil \log_2 m \rceil)T_X$	Mastrovito
$x^m + x^3 + x^2 + x + 1$	$m^2 + m + 2$	$T_A + (3 + \lceil \log_2 m \rceil)T_X$	Dual Basis
$x^m + x^{n+2} + x^{n+1} + x^n + 1$	$m^2 + 2m - \lceil \frac{m-2}{2} \rceil + 3n - 4$	$T_A + (3 + \lceil \log_2 m \rceil)T_X$	Dual Basis
$x^m + x^{n_3} + x^{n_2} + x^{n_1} + 1$	$m^2 + 2m - 3$	$T_A + (3 + \lceil \log_2 m \rceil)T_X$	[16]

$$2 + 2 + 3(m - n - 4) + 4 \cdot 2 + 5 \cdot 2 + 6(n - 2)$$
$$= 3m + 3n - 2.$$

Taking advantage of the inherent redundancy of the $m - 1$ trace equations in (23), the above number can be further reduced. Any of the $m - 1$ trace coefficients from t_m to t_{2m-1} can be implemented by reusing one of the XOR gates used in computing a previous coefficient. When k is odd, the computation of t_{m+k} requires one less XOR gate since the XOR gate in the computation of t_{m+k-1} can be reused. For instance, we notice that the computation of t_{m+1} requires two XOR gates if we reuse one of the XOR gates required for computing t_m. We need three XOR gates to compute t_{m+2}. Then, the computation of t_{m+3} requires only two XOR gates. Continuing in this fashion, it is not hard to prove that this strategy can be extended. Therefore, the complete set of $2m - 1$ coefficients can be computed using only

$$3m - \lceil (m-2)/2 \rceil + 3n - 4$$

XOR gates. Furthermore, only $3T_X$ gate delays are sufficient to obtain the entire set of t_k terms for $k = 0, 1, \ldots, (2m - 2)$. Combining these results with (19), we obtain the complexity of the proposed multiplier for an irreducible type 2 pentanomial as:

$$\begin{aligned} &\text{AND Gates} = m^2 \\ &\text{XOR Gates} = m^2 + 2m - \lceil (m-2)/2 \rceil + 3n - 4 \\ &\text{Total Delay} = T_A + (3 + \lceil \log_2 m \rceil)T_X. \end{aligned} \tag{24}$$

For the special case of $n = 1$, i.e., when the irreducible pentanomial is given as $x^m + x^3 + x^2 + x + 1$, it can be shown that the space and time complexities of the dual basis multiplier reduce to

$$\begin{aligned} &\text{AND Gates} = m^2 \\ &\text{XOR Gates} = m^2 - m + 2(m+1) = m^2 + m + 2 \\ &\text{Total Delay} = T_A + (3 + \lceil \log_2 m \rceil)T_X. \end{aligned} \tag{25}$$

5 Summary of Results and Conclusions

The complexity results of the proposed Mastrovito and dual basis multipliers are given in Table 3. This table also contains the complexity results of previously proposed multipliers based on irreducible trinomials and equally spaced polynomials [7], [8], [12], [15], [3].

While the multipliers based on trinomials and ESPs offer more advantageous designs, we have no choice but to consider other irreducible polynomials whenever irreducible trinomials or EPSs do not exist. This paper promotes the use of special types of irreducible pentanomials, as defined in Section 1. We proposed new Mastrovito and dual basis multiplier architectures and obtained their complexity results, using these special pentanomials.

It has been shown in [16] that an irreducible polynomial with Hamming weight (the number of terms) equal to r would require $(m-1)^2 + (r-1)(m-1)$ XOR gates. We also give this complexity result as applied to pentanomials ($r = 5$) in Table 3. As can be seen from Table 3, the special multipliers described in this paper using the pentanomials type I and type II, require $m - 2n + 3$ and $\lceil \frac{m-2}{2} \rceil - 3n + 1$ fewer XOR gates than the multiplier in [16], respectively.

Acknowledgments

F. Rodríguez-Henríquez was with Oregon State University as a doctoral student while part of this work was done.

References

[1] E.R. Berlekamp, "Bit-Serial Reed-Solomon Encoders," *IEEE Trans. Information Theory*, vol. 28, no. 6, pp. 869-874, Nov. 1982.
[2] S.T.J. Fenn, M. Benaissa, and D. Taylor, "$GF(2^m)$ Multiplication and Division over the Dual Basis," *IEEE Trans. Computers*, vol. 45, no. 3, pp. 319-327, Mar. 1996.
[3] A. Halbutogulllari and Ç.K. Koç, "Mastrovito Multiplier for General Irreducible Polynomials," *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, M. Fossorier, H. Imai, S. Lin, and A. Poli, eds., pp. 498-507, Berlin: Springer-Verlag, 1999.
[4] M.A. Hasan, M.Z. Wang, and V.K. Bhargava, "A Modified Massey-Omura Parallel Multiplier for a Class of Finite Fields," *IEEE Trans. Computers*, vol. 42, no. 10, pp. 1278-1280, Nov. 1993.
[5] IEEE P1363, "Standard Specifications for Public-Key Cryptography," Draft Version 13, 12 Nov. 1999.
[6] Ç.K. Koç and B. Sunar, "Low-Complexity Bit-Parallel Canonical and Normal Basis Multipliers for a Class of Finite Fields," *IEEE Trans. Computers*, vol. 47, no. 3, pp. 353-356, Mar. 1998.
[7] E.D. Mastrovito, "VLSI Architectures for Multiplication over Finite Field GF(2^m)," *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, T. Mora, ed., pp. 297-309, Berlin: Springer-Verlag, 1988.
[8] E.D. Mastrovito, "VLSI Architectures for Computation in Galois Fields," PhD thesis, Dept. of Electrical Eng., Linköping Univ., Sweden, 1991.
[9] R.J. McEliece, *Finite Fields for Computer Scientists and Engineers*. Boston: Kluwer Academic, 1987.

[10] A.J. Menezes, I.F. Blake, X. Gao, R.C. Mullen, S.A. Vanstone, and T. Yaghoobian, *Applications of Finite Fields.* Bston: Kluwer Academic, 1993.
[11] M. Morii, M. Kasahara, and D.L. Whiting, "Efficient Bit-Serial Multiplication and the Discrete-Time Wiener-Hopf Equation over Finite Fields," *IEEE Trans. Information Theory,* vol. 35, no. 6, pp. 1177-1183, Nov. 1989.
[12] C. Paar, "Efficient VLSI Architectures for Bit Parallel Computation in Galois Fields," PhD thesis, Universität GH Essen, VDI Verlag, 1994.
[13] F. Rodríguez-Henríquez, "New Algorithms and Architectures for Arithmetic in $GF(2^m)$ Suitable for Elliptic Curve Cryptography," PhD thesis, Oregon State Univ., 2000.
[14] G. Seroussi, "Table of Low-Weight Binary Irreducible Polynomials," Hewlett-Packard, HPL-98-135, Aug. 1998.
[15] B. Sunar and Ç.K. Koç, "Mastrovito Multiplier for All Trinomials," *IEEE Trans. Computers,* vol. 48, no. 5, pp. 522-527, May 1999.
[16] H. Wu, "Low Complexity Bit-Parallel Finite Field Arithmetic Using Polynomial Basis," *Cryptographic Hardware and Embedded Systems,* Ç.K. Koç and C. Paar, eds., pp. 280-291, Berlin: Springer-Verlag, 1999.
[17] H. Wu and M.A. Hasan, "Low Complexity Bit-Parallel Multipliers for a Class of Finite Fields," *IEEE Trans. Computers,* vol. 47, no. 8, pp. 883-887, Aug. 1998.
[18] H. Wu, M.A. Hasan, and I.F. Blake, "New Low-Complexity Bit-Parallel Finite Field Multipliers Using Weakly Dual Bases," *IEEE Trans. Computers,* vol. 47, no. 11, pp. 1223-1233, Nov. 1998.

Francisco Rodríguez-Henríquez received the PhD (2000) degree in electrical and computer engineering from Oregon State University, the MS (1992) degree in electrical and computer engineering from the National Institute of Astrophysics, Optics and Electronics (INAOE), México and the BS (1989) degree in electrical engineering from the University of Puebla, México. He is a visiting professor in the Computer Science Section of the Department of Electrical Engineering at CINVESTAV-IPN, México City, which he joined in 2002. His research interests are in data and communication security, wireless LANs Security, cryptography, finite fields, error correcting codes, and mobile computing. He is a member of the IEEE and he is also an alumni member and research associate of the Information Security Laboratory at Oregon State University.

Çetin Kaya Koç received the PhD (1988) and MS (1985) degrees in electrical and computer engineering from the University of California at Santa Barbara and the MS (1982) and BS (1980, summa cum laude) degrees in electrical engineering from Istanbul Technical University. He is a professor in the Department of Electrical and Computer Engineering at Oregon State University (OSU), which he joined in 1992. He is the founder and director of the Information Security Laboratory at OSU. He received the OSU College of Engineering Research Award for Outstanding and Sustained Research Leadership in 2001. His research interests are in security, cryptography, computer arithmetic, finite fields, and mobile computing. He was the founder and program chair of the Workshop on Cryptographic Hardware and Embedded Systems (CHES). He was the guest editor of a special issue (April 2003) of the *IEEE Transactions on Computers* devoted to cryptographic hardware and embedded software development. He is also a member of the editorial board of the new journal *IEEE Transactions on Mobile Computing*. He is a senior member of the IEEE and a member of the professional societies, IEEE Computer Society, IEEE Information Theory Society, and International Association for Cryptologic Research (IACR).