Hindawi
VLSI Design
Volume 2018, Article ID 9269157, 7 pages
https://doi.org/10.1155/2018/9269157

Research Article

Efficient Nonrecursive Bit-Parallel Karatsuba Multiplier for a Special Class of Trinomials

Yin Li, Yu Zhang, and Xiaoli Guo

Department of Computer Science and Technology, Xinyang Normal University, Nanhu Road 237, Xinyang, Henan, China

Correspondence should be addressed to Yin Li; yunfeiyangli@gmail.com

Received 15 August 2017; Revised 1 December 2017; Accepted 10 December 2017; Published 11 January 2018

Academic Editor: Junqing Sun

Recently, we present a novel Mastrovito form of nonrecursive Karatsuba multiplier for all trinomials. Specifically, we found that related Mastrovito matrix is very simple for equally spaced trinomial (EST) combined with classic Karatsuba algorithm (KA), which leads to a highly efficient Karatsuba multiplier. In this paper, we consider a new special class of irreducible trinomial, namely, $x^m + x^{m/3} + 1$. Based on a three-term KA and shifted polynomial basis (SPB), a novel bit-parallel multiplier is derived with better space and time complexity. As a main contribution, the proposed multiplier costs about 2/3 circuit gates of the fastest multipliers, while its time delay matches our former result. To the best of our knowledge, this is the first time that the space complexity bound is reached without increasing the gate delay.

1. Introduction

Efficient hardware implementation of the finite field arithmetic, especially for $GF(2^m)$, is frequently desired in coding theory and public-key cryptosystems [1, 2]. Among these arithmetic operations in $GF(2^m)$, multiplication is of the most importance, as other complicated field operations such as exponentiation and inversion can be carried out by iterative multiplications. Thus, it is necessary to design efficient multiplier.

The field elements are usually represented by a certain basis such as polynomial basis (PB), normal basis (NB), and dual basis (DB). In PB representation, the multiplication consists of multiplying two polynomials and reducing the result modulo an irreducible polynomial. The choice of such an irreducible polynomial is critical to perform the reduction operation efficiently. Irreducible trinomial is one of the most common considerations [3, 4]. During recent years, many bit-parallel multipliers using PB representation are proposed for $GF(2^m)$ defined by irreducible trinomials, some of which can be found in [3, 5–8]. The efficiency of the architecture is always evaluated by space and time complexity. The former one is expressed in terms of the number of logic gates (XOR and AND) and the latter one is expressed in terms of the sum of XOR and AND gates delay of the critical path. Among these multipliers, the fastest bit-parallel multipliers nowadays are proposed by Fan and Hasan [9] and Hariri and Reyhani-Masoleh [10]. If $GF(2^m)$ is defined by $f(x) = x^m + x^k + 1,\ 1 < k \leq m/2$, the corresponding multiplier requires m^2 AND and $m^2 - 1$ XOR gates with time delay $T_A + \lceil \log_2(2m - k) \rceil T_X$ (for good fields, the time delay is $T_A + \lceil \log_2 m \rceil T_X$), where T_A and T_X are the circuit delay of one AND gate and one XOR gate, respectively. Except for these multipliers for general trinomials, there are also several proposals for special types of irreducible trinomials [11–13]. These multipliers usually utilize the special form of the trinomial to obtain efficient implementation.

The Karatsuba algorithm (KA) works recursively by breaking down one big multiplication into two or more submultiplications. It is a typical divide-and-conquer algorithm. Please note that the classic KA starts with a way to multiply two 2-term polynomials using three scalar multiplications. Some other variations are also investigated. More details can be found in [14–16]. The KA can be adopted to design subquadratic complexity multiplier [14, 17] or hybrid multiplier [18, 19]. Specially, there is another type of hybrid multiplier, namely, nonrecursive Karatsuba multiplier, which only applies KA once in the polynomial multiplication [8, 20]. These multipliers regularly require 3/4 circuits gates compared to the fastest bit-parallel multipliers, while its time delay increased by a small number of T_X. For example, Elia et al. [8] costs at least two more T_X.

Recently, we proposed a novel nonrecursive Karatsuba multiplier that is based on Mastrovito approach [21]. It is shown that our multiplier only requires one more T_X compared with the fastest multipliers [9, 10]. However, it costs a few more logic gates than Elia's result. Except for the nonrecursive Karatsuba multiplier for general trinomials, Shen and Jin [13] proposed a new Karatsuba multiplier that fully exploited equally spaced trinomial and the classic KA to simplify the modular reduction. Consequently, the space complexity of their scheme matches Elia's result. Meanwhile, the time complexity is $T_A + (1 + \lceil \log_2(m-1) \rceil)T_X$, which is roughly equal to the fastest results. Furthermore, we observe that the special case $m = 2k$ of our multiplier coincides with their scheme. (Here, the trinomial $x^m + x^k + 1$ $(m = 2k)$ is an equally spaced trinomial.)

In this paper, we explore another special case of our former scheme to obtain even more efficient nonrecursive Karatsuba multipliers. Our main idea is analogous to Shen and Jin [13], where a special type of trinomials and a KA variation are utilized to simplify the structure of corresponding Mastrovito matrix. More explicitly, we consider the irreducible trinomial $x^m + x^{m/3} + 1$ and a three-term Karatsuba algorithm. It is demonstrated that the corresponding Mastrovito matrix can be simplified further under this condition. The shifted polynomial basis (SPB) [4] is also utilized to reduce the critical path delay further. Consequently, we proposed a bit-parallel multiplier that costs approximately 2/3 circuit gates of the fastest bit-parallel multipliers. On the other hand, the time complexity is $T_A + \lceil \log_2(8m/3) \rceil T_X$, which almost matches the best known results.

The rest of this paper is organized as follows: In Section 2, we briefly review the Mastrovito approach based on SPB representation and some relevant notions. Then we introduce a three-term KA formula and investigate the structure of related Mastrovito matrix. A new bit-parallel multiplier architecture is then proposed in Section 3. Section 4 presents a comparison between the proposed multiplier and some others. Finally, some conclusions are drawn.

2. Preliminary

In this section, we briefly review some related notations and algorithms used throughout this paper. Consider the finite field $GF(2^m)$ generated with an irreducible trinomial $x^m + x^{m/3} + 1$. Let x be a root of $x^m + x^{m/3} + 1$ and the set $M = \{x^{m-1}, x^{m-2}, \ldots, x, 1\}$ constitute a polynomial basis (PB). Therefore, every element of $GF(2^m)$ can be represented as a polynomial over $\mathbb{F}_2$ of degree less than m. The shifted polynomial basis (SPB) is a variation of the polynomial basis, which is obtained by multiplying the set M by certain exponentiation of x.

Definition 1 (see [4]). Let v be an integer and the ordered set $M = \{x^{m-1}, \ldots, x, 1\}$ be a polynomial basis of $GF(2^m)$ over $\mathbb{F}_2$. The ordered set $x^{-v}M := \{x^{i-v} \mid 0 \le i \le m-1\}$ is called the shifted polynomial basis with respect to M.

Generally speaking, the optimal choice of v for irreducible trinomial is equal to the middle term degree or it minus one [4]. In this case, we have $v = m/3$ and use this denotation thereafter. It follows that the field element $A \in GF(2^m)$ can be expressed with respect to SPB as follows:

$$\begin{aligned} A &= x^{-m/3} \sum_{i=0}^{m-1} a_i x^i \\ &= a_{m-1} x^{2m/3-1} + \cdots + a_1 x^{-m/3+1} + a_0 x^{-m/3}. \end{aligned} \tag{1}$$

Given two elements of $GF(2^m)$ under SPB representation, that is, $A(x) = \sum_{i=0}^{m-1} a_i x^{i-m/3}$, $B(x) = \sum_{i=0}^{m-1} b_i x^{i-m/3}$, the field multiplication can be performed as

$$C(x)\, x^{-m/3} = A(x)\, x^{-m/3} \cdot B(x)\, x^{-m/3} \bmod f(x). \tag{2}$$

Obviously, the product $D = AB$ is thus equal to

$$D(x) = x^{-2m/3} \left(\sum_{i=0}^{m-1} a_i x^i \right) \left(\sum_{i=0}^{m-1} b_i x^i \right). \tag{3}$$

Analogous to ordinary polynomial multiplication, this product can be computed by a matrix-vector multiplication $\mathbf{d} = \mathbf{A} \cdot \mathbf{b}$, where $\mathbf{b}, \mathbf{d}$ express the coefficient vectors of $B(x)$ and $D(x)$, and the matrix $\mathbf{A}$ is given by

$$\mathbf{A} = \begin{array}{c|cccccc|} -\dfrac{2m}{3} & a_0 & 0 & 0 & \cdots & 0 & 0 \\ -\dfrac{2m}{3}+1 & a_1 & a_0 & 0 & \cdots & 0 & 0 \\ -\dfrac{2m}{3}+2 & a_2 & a_1 & a_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \dfrac{m}{3}-2 & a_{m-2} & a_{m-3} & a_{m-4} & \cdots & a_0 & 0 \\ \dfrac{m}{3}-1 & a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_1 & a_0 \\ \dfrac{m}{3} & 0 & a_{m-1} & a_{m-2} & \cdots & a_2 & a_1 \\ \dfrac{m}{3}+1 & 0 & 0 & a_{m-1} & \cdots & a_3 & a_2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \dfrac{4m}{3}-3 & 0 & 0 & 0 & \cdots & a_{m-1} & a_{m-2} \\ \dfrac{4m}{3}-2 & 0 & 0 & 0 & \cdots & 0 & a_{m-1} \end{array}. \tag{4}$$

The difference between the above matrix and the usual PB case [3] is simply the labels of the lines in left side, which indicate the exponent of indeterminate x for each line.

We then reduce the above matrix in view to obtain the field product expressed in SPB representation. The reduced matrix, denoted by $\mathbf{M}$, is called Mastrovito matrix. Thus, the SPB field multiplication is rewritten as

$$\mathbf{c} = \mathbf{M} \cdot \mathbf{b}, \tag{5}$$

where **c** denotes the coefficient vector of $C(x)$. The structure of **M** relies on **A** and the modular reduction rule. In this case, we should obey the following reduction rule:

$$x^i = x^{i-2m/3} + x^{i-m}, \quad \text{for } \frac{2m}{3} \le i \le \frac{4m}{3} - 2,$$
$$x^i = x^{m+i} + x^{m/3+i}, \quad \text{for } -\frac{2m}{3} \le i \le -\left(\frac{m}{3} + 1\right). \tag{6}$$

However, if we directly reduce the product matrix presented in (4) using the above formulae and perform matrix-vector multiplication, there is no difference between this computation and the general case. In the following section, we will construct a new Mastrovito matrix using a three-term Karatsuba algorithm and describe a highly efficient bit-parallel multiplier.

Moreover, one can check that the irreducible trinomial in the form of $x^m + x^{m/3} + 1$ exists when $m = 3 \times 7^i$ where i is a nonnegative integer [1]. Although the number of this type of irreducible trinomials is not that abundant, there still exist some trinomials in the range of interest for practical application.

In the end, we also introduce some notations pertaining to matrices and vectors, which are already proposed in [21, 23] and extensively used throughout this paper.

(i) $\mathbf{Z}(i,:)$ represents the ith row vector in matrix $\mathbf{Z}$;

(ii) $\mathbf{Z}(:,j)$ represents the jth column vector in matrix $\mathbf{Z}$;

(iii) $\mathbf{Z}(i,j)$ represents the entry with position (i,j) in matrix $\mathbf{Z}$.

3. Mastrovito Multiplier Using a Three-Term Karatsuba Algorithm

The Karatsuba algorithm [2] has been applied to improve the efficiency of bit-parallel multiplier for $GF(2^m)$ generated by an AOP [20] and a trinomial [8, 13, 21]. It starts with a way to multiply two two-term polynomials using three scalar multiplications which can reduce the space complexity of the multipliers by approximately a factor of 3/4. Besides the classic algorithm, there exist several generalizations with respect to the Karatsuba algorithm [14–16]. Here, we are only focus on a simple Karatsuba algorithm variation, three-term Karatsuba algorithm, which multiplies two three-term polynomials using six scalar multiplications. Given two three-term polynomials in $\mathbb{F}_2[x]$, one can check that

$$\begin{aligned}&\left(a_2x^2 + a_1x + a_0\right)\left(b_2x^2 + b_1x + b_0\right)\\ &\quad = a_2b_2\left(x^4 + x^3 + x^2\right) + a_1b_1\left(x^3 + x^2 + x\right)\\ &\qquad + a_0b_0\left(x^2 + x + 1\right) + \left(a_2 + a_1\right)\left(b_2 + b_1\right)x^3\\ &\qquad + \left(a_2 + a_0\right)\left(b_2 + b_0\right)x^2 + \left(a_1 + a_0\right)\left(b_1 + b_0\right)x.\end{aligned} \tag{7}$$

In general, the Mastrovito multiplication utilizing the KA will increase the time complexity. Our former result shows that a Mastrovito multiplier using classic KA costs one more T_X than the fastest ones. However, some literature sources [13] indicated that this result would be further improved for some special cases, for example, the EST $x^m + x^{m/2} + 1$. In the following, we will show that for the trinomial $x^m + x^{m/3} + 1$, applying the three-term Karatsuba-like formula will also simplify the reduction operation and lead to fast implementation.

Let $f(x) = x^m + x^{m/3} + 1$ be an irreducible trinomial and $A = x^{-m/3}\sum_{i=0}^{m-1} a_ix^i$, $B = x^{-m/3}\sum_{i=0}^{m-1} b_ix^i$ be two field elements in SPB representation. We partition A, B into three parts, with each part consisting of $m/3$ bits. In order to simplify related expressions, we denote $m/3$ as k. Then,

$$A = A_2x^k + A_1 + A_0x^{-k},$$
$$B = B_2x^k + B_1 + B_0x^{-k}, \tag{8}$$

where $A_i = \sum_{j=0}^{k-1} a_{j+ik}x^j$, $B_i = \sum_{j=0}^{k-1} b_{j+ik}x^j$, for $i = 0, 1, 2$. Then we multiply A and B using the three-term Karatsuba-like formula and do the following transformation:

$$\begin{aligned}AB &= \left(A_2x^k + A_1 + A_0x^{-k}\right)\cdot\left(B_2x^k + B_1 + B_0x^{-k}\right)\\ &= A_2B_2\left(x^{2k} + x^k + 1\right) + A_1B_1\left(x^k + 1 + x^{-k}\right)\\ &\quad + A_0B_0\left(1 + x^{-k} + x^{-2k}\right) + C_2D_2x^k + C_1D_1\\ &\quad + C_0D_0x^{-k}\\ &= \left(A_2B_2x^k + A_1B_1 + A_0B_0x^{-k}\right)\left(x^k + 1 + x^{-k}\right)\\ &\quad + \left(C_2D_2x^k + C_1D_1 + C_0D_0x^{-k}\right),\end{aligned} \tag{9}$$

where $C_2 = A_2 + A_1$, $C_1 = A_2 + A_0$, $C_0 = A_1 + A_0$, $D_2 = B_2 + B_1$, $D_1 = B_2 + B_0$, $D_0 = B_1 + B_0$. We divide (9) into two parts,

$$S_1 = \left(A_2B_2x^k + A_1B_1 + A_0B_0x^{-k}\right)\left(x^k + 1 + x^{-k}\right),$$
$$S_2 = \left(C_2D_2x^k + C_1D_1 + C_0D_0x^{-k}\right), \tag{10}$$

and compute each part modulo $f(x)$ independently.

3.1. *Computation of* $S_1 \bmod f(x)$.

We first consider the computation of S_1 in detail. Note that S_1 actually consists of three different parts: A_0B_0, A_1B_1, A_2B_2 (others can be obtained by shift of these parts). When S_1 is rewritten as a matrix-vector form, we have

$$S_1 = \mathbf{A}\cdot\mathbf{b}$$

$$= \begin{bmatrix} \mathbf{A}_{0,L}, & \mathbf{0}_{k\times k}, & \mathbf{0}_{k\times k} \\ \mathbf{A}_{0,L}+\mathbf{A}_{0,H}, & \mathbf{A}_{1,L}, & \mathbf{0}_{k\times k} \\ \mathbf{A}_{0,L}+\mathbf{A}_{0,H}, & \mathbf{A}_{1,L}+\mathbf{A}_{1,H}, & \mathbf{A}_{2,L} \\ \mathbf{A}_{0,H}, & \mathbf{A}_{1,L}+\mathbf{A}_{1,H}, & \mathbf{A}_{2,L}+\mathbf{A}_{2,H} \\ \mathbf{0}_{k\times k}, & \mathbf{A}_{1,H}, & \mathbf{A}_{2,L}+\mathbf{A}_{2,H} \\ \mathbf{0}_{k\times k}, & \mathbf{0}_{k\times k}, & \mathbf{A}_{2,H} \end{bmatrix} \quad (11)$$

$$\cdot \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}.$$

For simplicity, we do not write the labels of the product matrix here, which indicate the degree of x^i in S_1. Note that these degrees are in the range $[-2k, 2m-2k-2]$. In the above expression, $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ represent the coefficient vectors of B_0, B_1, B_2, respectively. $\mathbf{0}_{k\times k}$ is a $k \times k$ zero matrix, $\mathbf{A}_{i,L}$ ($i = 0, 1, 2$) are $k \times k$ lower-triangular Toeplitz matrices, and $\mathbf{A}_{i,H}$ ($i = 0, 1, 2$) are $k \times k$ upper-triangular Toeplitz matrices. Please note that the matrix on the right side actually contains $6k = 6 \cdot m/3 = 2m$ rows and the product matrix in fact contains $2m-1$ rows. However, the last row of the above matrix is $\mathbf{0}$, which does not affect the result. These submatrices have the following form:

$$\mathbf{A}_{i,L} = \begin{bmatrix} a_{ik+0} & 0 & \cdots & 0 \\ a_{ik+1} & a_{ik+0} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{ik+k-1} & a_{ik+k-2} & \cdots & a_{ik+0} \end{bmatrix},$$

$$\mathbf{A}_{i,L} = \begin{bmatrix} 0 & a_{ik+k-1} & \cdots & a_{ik+2} & a_{ik+1} \\ 0 & 0 & \cdots & a_{ik+3} & a_{ik+2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{ik+k-1} \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}, \quad (12)$$

for $i = 0, 1, 2$. It is easy to check that the products S_1 contain the terms of degrees out of the range $[-k, m-k-1]$; we have to perform the reduction operation for the product matrix in (21). According to Mastrovito scheme, the reduction can be regarded as the construction of product matrices from $\mathbf{A}$ using the reduction rule in (6). Denoted by $\mathbf{M}_A$, the Mastrovito matrix is related to S_1. Then, we investigate the construction details for this matrix $\mathbf{M}_A$. We have the following proposition.

Proposition 2. *The Mastrovito matrix* $\mathbf{M}_A$ *can be constructed as*

$$\mathbf{M}_A = \mathbf{M}_{A,1} + \mathbf{M}_{A,2}, \quad (13)$$

where

$$\mathbf{M}_{A,1} = \begin{bmatrix} \mathbf{A}_{0,L}+\mathbf{A}_{0,H}, & \mathbf{A}_{1,L}+\mathbf{A}_{1,H}, & \mathbf{A}_{2,L}+\mathbf{A}_{2,H} \\ \mathbf{A}_{0,L}+\mathbf{A}_{0,H}, & \mathbf{A}_{1,L}+\mathbf{A}_{1,H}, & \mathbf{A}_{2,L}+\mathbf{A}_{2,H} \\ \mathbf{A}_{0,L}+\mathbf{A}_{0,H}, & \mathbf{A}_{1,L}+\mathbf{A}_{1,H}, & \mathbf{A}_{2,L}+\mathbf{A}_{2,H} \end{bmatrix},$$

$$\mathbf{M}_{A,2} = \begin{bmatrix} \mathbf{A}_{0,L}, & \mathbf{0}_{k\times k}, & \mathbf{0}_{k\times k} \\ \mathbf{0}_{k\times k}, & \mathbf{A}_{1,H}, & \mathbf{A}_{2,L}+\mathbf{A}_{2,H} \\ \mathbf{0}_{k\times k}, & \mathbf{0}_{k\times k}, & \mathbf{A}_{2,H} \end{bmatrix}. \quad (14)$$

Proof. The proof is analogous with the proof of observation 3.1 in [21]. Note that the product matrix $\mathbf{A}$ contains $2m-1$ nonzero rows (the last row $\mathbf{A}(2m,:)$ is a zero vector), each of which corresponds to the polynomial degree from $-2k$ to $2m-2k-2$. It is easy to check that the first k rows and the last $m-k-1$ rows correspond to the degrees that are out of the range $[-k, m-k-1]$. Thus, we need to reduce these rows.

According to the reduction rule in (6), we have to reduce $\{-2k, -2k+1, \ldots, -k-1\}$ by adding them to the row $\{-k, \ldots, -1\}$ and $\{m-2k, \ldots, m-k-1\}$ and reduce the rows $\{m-k, \ldots, 2m-2k-2\}$ by adding them to the row $\{0, \ldots, m-k-2\}$ and $\{-k, \ldots, m-2k-2\}$. Obviously, the first k row here is $[\mathbf{A}_{0,L}, \mathbf{0}_{k\times k}, \mathbf{0}_{k\times k}]$ and the last $m-k-1$ rows constitute

$$\begin{bmatrix} \mathbf{0}_{k\times k}, & \mathbf{A}_{1,H}, & \mathbf{A}_{2,L}+\mathbf{A}_{2,H} \\ \mathbf{0}_{k\times k}, & \mathbf{0}_{k\times k}, & \mathbf{A}_{2,H} \end{bmatrix}. \quad (15)$$

We compare the line number and obtain the result immediately. □

Based on Proposition 2, we can compute S_1 as follows:

$$S_1 \bmod f(x) = \mathbf{M}_A \cdot \mathbf{b} = \mathbf{M}_{A,1} \cdot \mathbf{b} + \mathbf{M}_{A,2} \cdot \mathbf{b}. \quad (16)$$

By swapping and combining some overlapped entries, expression (16) now can be rewritten as

$$S_1 = \mathbf{M'}_{A,1} \cdot \mathbf{b} + \mathbf{M'}_{A,2} \cdot \mathbf{b}$$

$$= \begin{bmatrix} \mathbf{A}_{0,H}, & \mathbf{A}_{1,L}, & \mathbf{A}_{2,H} \\ \mathbf{A}_{0,H}, & \mathbf{A}_{1,L}, & \mathbf{0}_{k\times k} \\ \mathbf{A}_{0,H}, & \mathbf{A}_{1,L}, & \mathbf{0}_{k\times k} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} \quad (17)$$

$$+ \begin{bmatrix} \mathbf{0}_{k\times k}, & \mathbf{A}_{1,H}, & \mathbf{A}_{2,L} \\ \mathbf{A}_{0,L}, & \mathbf{0}_{k\times k}, & \mathbf{0}_{k\times k} \\ \mathbf{A}_{0,L}, & \mathbf{A}_{1,H}, & \mathbf{A}_{2,L} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}.$$

We just compute two submatrix-vector multiplications and add them up to obtain S_1. Some tricks can apply to save more logic gates. We mainly utilized the computation strategy presented in [7] and fully considered the overlapped parts of the two above matrices. The computation can be divided into two steps:

Table 1: Space and time complexities of $S_1 \bmod f(x)$.

Operation	#AND	#XOR	Time delay
Inner products in (18)	$3k^2$	-	T_A
Partial addition in (19)	-	$3k^2-4k+1$	$\lceil \log_2 k \rceil T_X$
$S_1 \bmod f(x)$	-	$4k-1$	$2T_X$

(i) Perform row-vector products:

$$\begin{gathered}\mathbf{A}_{0,L} * \mathbf{b}_0,\\ \mathbf{A}_{0,H} * \mathbf{b}_0,\\ \mathbf{A}_{1,L} * \mathbf{b}_1,\\ \mathbf{A}_{1,H} * \mathbf{b}_2,\\ \mathbf{A}_{2,L} * \mathbf{b}_2,\\ \mathbf{A}_{2,H} * \mathbf{b}_1,\end{gathered} \tag{18}$$

in parallel. The symbol "$*$" represents only row-vector product related to $\mathbf{A}_{i,L}$ (or $\mathbf{A}_{i,H}$) and $\mathbf{b}_i$, $i = 0, 1, 2$. For example, $\mathbf{A}_{0,H} * \mathbf{b}_0$ represents computing the inner product $[\mathbf{A}_{0,H}(i,1) \cdot b_0, \ldots, \mathbf{A}_{0,H}(i,k) \cdot b_{k-1}]$, for $i = 1, 2, \ldots, k$ in parallel.

(ii) Sum up all the $2m$ entries of each row using binary XOR tree. Specially, consider some products of each row are zero; we compute the following summations:

$$\begin{gathered}\mathbf{A}_{0,H} \cdot \mathbf{b}_0 + \mathbf{A}_{1,L} \cdot \mathbf{b}_1,\\ \mathbf{A}_{2,H} \cdot \mathbf{b}_3,\\ \mathbf{A}_{1,H} \cdot \mathbf{b}_1 + \mathbf{A}_{2,L} \cdot \mathbf{b}_2,\\ \mathbf{A}_{0,L} \cdot \mathbf{b}_0\end{gathered} \tag{19}$$

using binary XOR tree firstly and then add these results together.

Remarks 3. It is easy to see that the row-vector products (18) contain all the possible row-vector products in (17). In addition, $\mathbf{A}_{0,H}, \mathbf{A}_{1,L}, \mathbf{A}_{1,H}$, and $\mathbf{A}_{2,L}$ are all triangular matrices; one can easily check that each row of both $[\mathbf{A}_{0,H}, \mathbf{A}_{1,L}]$ and $[\mathbf{A}_{1,H}, \mathbf{A}_{2,L}]$ consists of at most k nonzero entries. After the computation of (18) and (19), certain number of XOR gates is required to obtain the final result. Table 1 summarizes the space and time complexity of S_1 for all the steps.

3.2. Computation of $S_2 \bmod f(x)$.

Then we consider the computation of $S_2 \bmod f(x)$ in detail. Since $S_2 = (C_2 D_2 x^k + C_1 D_1 + C_0 D_0 x^{-k})$ and C_i, $D_i (i = 0, 1, 2)$ consist of k bits, we can follow similar line as the computation of S_1 to obtain the result. More explicitly, we rewrite S_2 in matrix-vector form:

$$S_2 = \begin{bmatrix} \mathbf{C}_{0,L}, & \mathbf{0}_{k\times k} & \mathbf{0}_{k\times k} \\ \mathbf{C}_{0,H}, & \mathbf{C}_{1,L} & \mathbf{0}_{k\times k} \\ \mathbf{0}_{k\times k}, & \mathbf{C}_{1,H}, & \mathbf{C}_{2,L} \\ \mathbf{0}_{k\times k}, & \mathbf{0}_{k\times k}, & \mathbf{C}_{2,H} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{d}_0 \\ \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix}. \tag{20}$$

Table 2: Space and time complexities of $S_2 \bmod f(x)$.

Operation	#AND	#XOR	Time delay
C_0, C_1, C_2	-	$3k$	T_X
D_0, D_1, D_2	-	$3k$	
Inner products in (22)	$3k^2$	-	T_A
Partial addition in (23)	-	$3k^2-4k+1$	$\lceil \log_2 k \rceil T_X$
$S_2 \bmod f(x)$	-	$2k-2$	T_X

Here, $\mathbf{C}_{i,L}$ $(i = 0, 1, 2)$ are $k \times k$ lower-triangular Toeplitz matrices and $\mathbf{C}_{i,H}$ $(i = 0, 1, 2)$ are $k \times k$ upper-triangular Toeplitz matrices, which are constructed from the coefficients of C_0, C_1, C_2 and are similar to $\mathbf{A}_{i,L}$ and $\mathbf{A}_{i,H}$. Vectors $\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2$ represent the coefficient vectors of D_0, D_1, D_2.

The reduction of S_2 modulo $f(x)$ is relatively simpler: we only need to eliminate the last k rows by adding them to the lines labeled with $\{-k, \ldots, -2\}$ and $\{0, \ldots, k-1\}$. Thus, we have

$$S_2 \bmod f(x) = \begin{bmatrix} \mathbf{C}_{0,L}, & \mathbf{0}_{k\times k} & \mathbf{C}_{2,H} \\ \mathbf{C}_{0,H}, & \mathbf{C}_{1,L} & \mathbf{C}_{2,H} \\ \mathbf{0}_{k\times k}, & \mathbf{C}_{1,H}, & \mathbf{C}_{2,L} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{d}_0 \\ \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix}. \tag{21}$$

Analogous with the computation of $S_1 \bmod f(x)$, we first perform row-vector products:

$$\begin{gathered}\mathbf{C}_{0,L} * \mathbf{d}_0,\\ \mathbf{C}_{0,H} * \mathbf{d}_0,\\ \mathbf{C}_{1,L} * \mathbf{d}_1,\\ \mathbf{C}_{1,H} * \mathbf{d}_2,\\ \mathbf{C}_{2,L} * \mathbf{d}_2,\\ \mathbf{C}_{2,H} * \mathbf{d}_1,\end{gathered} \tag{22}$$

in parallel. Then, we compute the following summations:

$$\begin{gathered}\mathbf{C}_{0,L} \cdot \mathbf{d}_0,\\ \mathbf{C}_{0,H} \cdot \mathbf{d}_0 + \mathbf{C}_{1,L} \cdot \mathbf{d}_1,\\ \mathbf{C}_{1,H} \cdot \mathbf{d}_1 + \mathbf{C}_{2,L} \cdot \mathbf{d}_2,\\ \mathbf{C}_{2,H} \cdot \mathbf{d}_2,\end{gathered} \tag{23}$$

using binary XOR tree firstly, and then add related results together. Please note that each row of $[\mathbf{C}_{0,H} * \mathbf{d}_0, \mathbf{C}_{1,L} * \mathbf{d}_1]$ and $[\mathbf{C}_{1,H} * \mathbf{d}_1, \mathbf{C}_{2,L} * \mathbf{d}_2]$ consists of at most k nonzero entries. We can calculate $\mathbf{C}_{0,H} \cdot \mathbf{d}_0 + \mathbf{C}_{1,L} \cdot \mathbf{d}_1$ and $\mathbf{C}_{1,H} \cdot \mathbf{d}_1 + \mathbf{C}_{2,L} \cdot \mathbf{d}_2$ in $\lceil \log k \rceil T_X$. Finally, we have to add all these summations to obtain the result. It costs $2k-2$ more XOR gates with one T_X delay. Related space and time complexities for the computation of $S_2 \bmod f(x)$ are summarized in Table 2.

From Tables 1 and 2, it is clear that the computations of S_1, S_2 modulo $f(x)$ have the same time delay. So they can be implemented in parallel. Finally, another m XOR gates are needed to add the two results together, which also requires

Table 3: Comparison of bit-parallel multipliers for $GF(2^m)$ generated with $x^m + x^{m/3} + 1$.

Multiplier	#AND	#XOR	Time delay
Sunar and Koç [3]	m^2	$m^2 - 1$	$T_A + (2 + \lceil \log_2 m \rceil) T_X$
Wu [5]	m^2	$m^2 - 1$	$T_A + (2 + \lceil \log_2 m \rceil) T_X$
Wu [6]	m^2	$m^2 - 1$	$T_A + (2 + \lceil \log_2 m \rceil) T_X$
Fan and Dai [4]	m^2	$m^2 - 1$	$T_A + \left\lceil \log_2 \frac{5m}{3} \right\rceil T_X$
Elia et al. [8]	$\frac{3m^2}{4}$	$\frac{3m^2}{4} + \frac{13m}{3} - \frac{23}{4}$	$T_A + (3 + \lceil \log_2 m \rceil) T_X$
Negre [7]	m^2	$\frac{23m^2}{18} - \frac{3m}{2}$	$T_A + \left\lceil \log_2 \frac{5m}{3} \right\rceil T_X$
Fan [22] Type-A	$m^2 - \frac{m}{3}$	$m^2 - \frac{m}{3}$	$T_A + \left\lceil \log_2 \left(\max \left(2m - 1, \frac{4m}{3} + 2^v \right) \right) \right\rceil T_X$
Fan [22] Type-B	$m^2 - \frac{m}{3}$	$m^2 - \frac{2m}{3} + \frac{m}{3} \cdot W\left(\frac{m}{3}\right)$	$T_A + \lceil \log_2 (2m - 1) \rceil T_X$
Li et al. [21]	$\frac{3m^2 + 2m - 1}{4}$	$\frac{3m^2}{4} + \frac{m}{2} + O(m \log_2 m)$	$T_A + \left(1 + \left\lceil \log_2 \frac{5m}{3} \right\rceil \right) T_X$
This paper	$\frac{2m^2}{3}$	$\frac{2m^2}{3} + \frac{7m}{3} - 1$	$T_A + \left\lceil \log_2 \frac{8m}{3} \right\rceil T_X$

Note. $2^{v-1} < m/3 \leq 2^v$ and $W(*)$ is the hamming weight of the number.

one T_X delay. As a consequence, the total space and time complexity of proposed architecture are

$$\#\text{AND} = 6k^2 = \frac{2m^2}{3},$$

$$\#\text{XOR} = 6k^2 + 4k - 1 + m = \frac{2m^2}{3} + \frac{7m}{3} - 1, \quad (24)$$

$$\text{Time delay} = T_A + (3 + \lceil \log_2 k \rceil) T_X$$

$$= T_A + \left\lceil \log_2 \frac{8m}{3} \right\rceil T_X.$$

Furthermore, if $m = 2^n + c$ where c is smaller relatively to 2^{n-1}, we have $\lceil \log_2(8m/3) \rceil = 1 + \lceil \log_2 m \rceil$. In this case, the time delay of our architecture becomes $T_A + (1 + \lceil \log_2 m \rceil T_X)$, which is almost equal to the delay of the fastest bit-parallel multipliers [9].

4. Theoretic Comparison

Table 3 gives a comparison of different implementation methods of bit-parallel multipliers in the fields generated by trinomials $x^m + x^{m/3} + 1$. From Table 3, we can see that our multiplier requires about 2/3 circuit gates compared with the previous architectures without using divide-and-conquer algorithm. On the other hand, the time complexity of the proposed multiplier is $T_A + \lceil \log_2(8m/3) \rceil T_X$, which is very close to the fastest result. In fact, we have checked this type of trinomials with degree $m = 3 \cdot 7^i$, $i = 1, 2, \ldots, 1000$, and found that there are 585 such trinomials reaching the bound $T_A + (1 + \lceil \log_2 m \rceil T_X)$ (others require only one more T_X).

In Table 4, we give a small example of field $GF(2^{147})$ defined by $x^{147} + x^{49} + 1$. It shows that, compared with other approaches, our architecture may be the best choice if the space and time complexity are both considered. In addition, compared with the fastest Karatsuba multiplier for general trinomials [21], it is argued that the space and time complexities can be reduced even further if special KA and irreducible polynomial are combined together.

Table 4: Complexity for practical field $GF(2^{147})$.

Basis	#AND	#XOR	Time
PB [3, 5]	21609	21608	$T_A + 10T_X$
PB [8]	16280	16838	$T_A + 11T_X$
SPB [4]	21609	21608	$T_A + 9T_X$
SPB [9]	21609	21608	$T_A + 8T_X$
SPB [7]	21609	27391	$T_A + 8T_X$
PB-CRT Type A [22]	21560	21560	$T_A + 9T_X$
PB-CRT Type B [22]	21560	21658	$T_A + 9T_X$
SPB [21]	16280	17394	$T_A + 9T_X$
SPB (this paper)	14406	14748	$T_A + 9T_X$

5. Conclusion

In this paper, a new Mastrovito multiplier architecture for trinomial of the form $x^m + x^{m/3} + 1$ is proposed. We show that the space and time complexity of our former Mastrovito-Karatsuba multiplier can be further reduced for special form of trinomial combined with a KA variation. This multiplier can be used in some area-critical occasions because it has low space complexity but maintains a relatively low time delay. To find more polynomials which can use the proposed strategy will be the future work.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the Natural Science Foundation of China (nos. 61402393, 61601396) and Shanghai Key

Laboratory of Integrated Administration Technologies for Information Security (no. AGK201607).

References

[1] R. Lidl and H. Niederreiter, *Finite Fields*, Cambridge University Press, New York, NY, USA, 1996.

[2] D. E. Knuth, *Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Addison-Wesley Professional, 3rd edition, 1997.

[3] B. Sunar and Ç. K. Koç, "Mastrovito multiplier for all trinomials," *IEEE Transactions on Computers*, vol. 48, no. 5, pp. 522–527, 1999.

[4] H. Fan and Y. Dai, "Fast bit-parallel GF(2^n) multiplier for all trinomials," *IEEE Transactions on Computers*, vol. 54, no. 4, pp. 485–490, 2005.

[5] H. Wu, "Bit-parallel finite field multiplier and squarer using polynomial basis," *IEEE Transactions on Computers*, vol. 51, no. 7, pp. 750–758, 2002.

[6] H. Wu, "Montgomery multiplier and squarer for a class of finite fields," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 521–529, 2002.

[7] C. Negre, "Efficient parallel multiplier in shifted polynomial basis," *Journal of Systems Architecture*, vol. 53, no. 2-3, pp. 109–116, 2007.

[8] M. Elia, M. Leone, and C. Visentin, "Low complexity bit-parallel multipliers for $GF(2^m)$ with generator polynomial $x^m + x^k + 1$," *IEEE Electronics Letters*, vol. 35, no. 7, pp. 551-552, 1999.

[9] H. Fan and M. A. Hasan, "Fast bit parallel-shifted polynomial basis multipliers in $GF(2^n)$," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 12, pp. 2606–2615, 2006.

[10] A. Hariri and A. Reyhani-Masoleh, "Bit-serial and bit-parallel montgomery multiplication and squaring over $GF(2^m)$," *IEEE Transactions on Computers*, vol. 58, no. 10, pp. 1332–1345, 2009.

[11] Y. J. Choi, K.-Y. Chang, D. W. Hong, and H. S. Cho, "Hybrid multiplier for $GF(2^m)$ defined by some irreducible trinomials," *IEEE Electronics Letters*, vol. 40, no. 14, pp. 852-853, 2004.

[12] C.-Y. Lee, "Low-latency bit-parallel systolic multiplier for irreducible $x^m + x^n + 1$ with $GCD(m, n) = 1$," *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. E86A, no. 11, pp. 2844–2852, 2003.

[13] H. Shen and Y. Jin, "Low complexity bit parallel multiplier for $GF(2^m)$ generated by equally-spaced trinomials," *Information Processing Letters*, vol. 107, no. 6, pp. 211–215, 2008.

[14] A. Weimerskirch and C. Paar, *Generalizations of the Karatsuba Algorithm for Efficient Implementations*, 2003, http://www.crypto.ruhr-uni-bochum.de/imperia/md/content/texte/kaweb.pdfAvailable on.

[15] P. L. Montgomery, "Five, six, and seven-term Karatsuba-like formulae," *IEEE Transactions on Computers*, vol. 54, no. 3, pp. 362–369, 2005.

[16] H. Fan, M. Gu, J. Sun, and K.-Y. Lam, "Obtaining more Karatsuba-like formulae over the binary field," *IET Information Security*, vol. 6, no. 1, pp. 14C–19, 2012.

[17] H. Fan, J. Sun, M. Gu, and K.-Y. Lam, "Overlap-free Karatsuba-Ofman polynomial multiplication algorithms," *IET Information Security*, vol. 4, no. 1, pp. 8–14, 2010.

[18] F. Rodríguez-Henríquez and Ç. K. Koç, "On fully parallel Karatsuba multipler for $GF(2^m)$," in *proceedings of the International Conference on Computer Science and Technology (CST, '03)*, pp. 405–410, ATA Press, 2003.

[19] J. Von Zur Gathen and J. Shokrollahi, "Efficient FPGA-based Karatsuba multipliers for polynomial over $\mathbb{F}_2$," in *Proceedings of the 12th Workshop on Selected Areas in Cryptography (SAC '05)*, vol. 3897, pp. 359-359, Springer.

[20] K.-Y. Chang, D. Hong, and H.-S. Cho, "Low complexity bit-parallel multiplier for $GF(2^m)$ defined by all-one polynomials using redundant representation," *IEEE Transactions on Computers*, vol. 54, no. 12, pp. 1628–1630, 2005.

[21] Y. Li, X. Ma, Y. Zhang, and C. Qi, "Mastrovito form of non-recursive karatsuba multiplier for all trinomials," *IEEE Transactions on Computers*, vol. 66, no. 9, pp. 1573–1584, 2017.

[22] H. Fan, "A Chinese remainder theorem approach to bit-parallel $GF(2^n)$ polynomial basis multipliers for irreducible trinomials," *IEEE Transactions on Computers*, vol. 65, no. 2, pp. 343–C352, 2016.

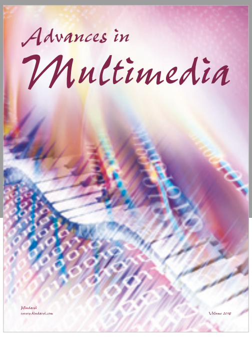
[23] T. Zhang and K. K. Parhi, "Systematic design of original and modified Mastrovito multipliers for general irreducible polynomials," *IEEE Transactions on Computers*, vol. 50, no. 7, pp. 734–749, 2001.

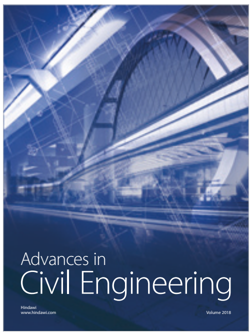
Journal of
Engineering

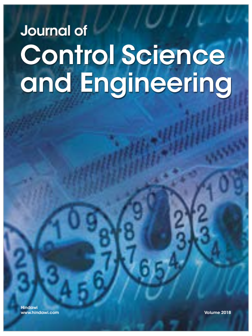
The Scientific
World Journal

International Journal of
Rotating
Machinery

Journal of
Sensors

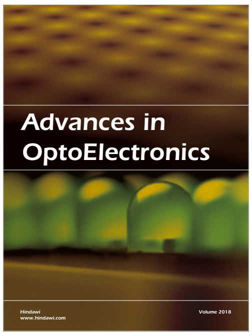

Advances in
Multimedia


Advances in
Civil Engineering


Journal of
Control Science
and Engineering

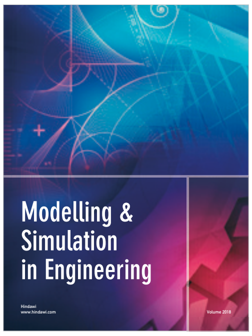
Journal of
Robotics

Journal of
Electrical and Computer
Engineering

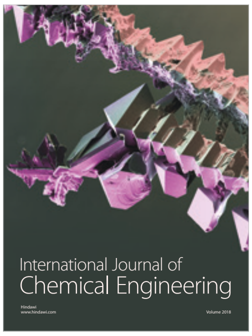

Advances in
OptoElectronics

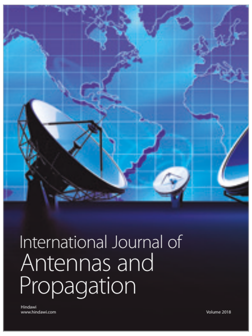
VLSI Design

International Journal of
Navigation and
Observation

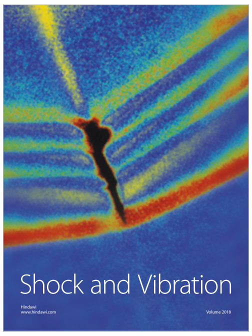

Modelling &
Simulation
in Engineering

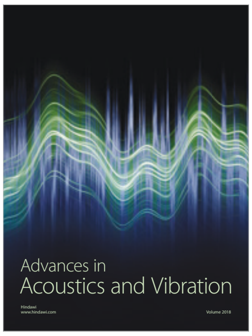
International Journal of
Aerospace
Engineering


International Journal of
Chemical Engineering


International Journal of
Antennas and
Propagation

Active and Passive
Electronic Components


Shock and Vibration


Advances in
Acoustics and Vibration