Low complexity bit-parallel $GF(2^m)$ multiplier for all-one polynomials

Yin Li[1], Gong-liang Chen[2], and Xiao-ning Xie

[1] Xinyang local taxation bureau, Henan, China. Email:yunfeiyangli@gmail.com,
[2] School of Information Security Engineering, Shanghai Jiaotong University, Shanghai, China. Email:chengl@sjtu.edu.cn

Abstract. This paper presents a new bit-parallel multiplier for the finite field $GF(2^m)$ generated with an irreducible all-one polynomial. Redundant representation is used to reduce the time delay of the proposed multiplier, while a three-term Karatsuba-like formula is combined with this representation to decrease the space complexity. As a result, the proposed multiplier requires about 10 percent fewer AND/XOR gates than the most efficient bit-parallel multipliers using an all-one polynomial, while it has almost the same time delay as the previously proposed ones.

1 Introduction

The arithmetic operations in $GF(2^m)$ are frequently desired in many areas of computer algebra and public-key cryptosystems [1]. Thus it is necessary to design efficient algorithms for finite field operations such as addition, multiplication, exponentiation, and multiplicative inversion. In particular, efficient design of the multiplier is required by hardware implementation of multiplication, since other time consuming operations such as exponentiation and inversion can be carried out by iterative multiplications. The multiplication in a polynomial basis consists of multiplying two polynomials and then reducing the result modulo an irreducible polynomial. The choice of corresponding irreducible polynomial is important to perform the reduction efficiently. The most used irreducible polynomials are the all-one polynomials (AOP) [2, 3] and the sparse polynomials such as trinomials [4, 5] and pentanomials [5, 6].

In hardware implementation, the efficiency of the architecture is always evaluated by space and time complexity. The former is expressed as the number of logic gates (XOR and AND) and the latter is expressed as the sum of gates delay of the critical path. Based on the special form of AOP, efficient bit-parallel multipliers are proposed using polynomial basis (PB)[2], normal basis (NB) [3, 7, 8], weakly dual basis [9] and non-conventional basis [10]. These multipliers are known as the most efficient ones among the previous bit-parallel multipliers and require m^2 AND gates, $m^2 - 1$ XOR gates and $T_A + (1 + \lceil \log_2(m-1) \rceil)T_X$ delays. Some other bit-parallel multipliers using an AOP have been proposed in [11–14].

In a recent paper, Chang et al. [15] has proposed a low complexity bit-parallel multiplier for AOP using redundant representation. A redundant representation [16] was derived from the minimal cyclotomic ring. In this case, the field $GF(2^m)$ is represented as a subring of a residue class ring $\mathbb{F}_2[x]/(x^n+1)$ with $n > m$. Advantages of using redundant representation include free squaring operation and elimination of the modular reduction step in the field multiplication. Based on this representation, Chang et al. can apply Karatsuba approach to reduce the space complexity without increasing gates delay. Consequently, with nearly the same time complexity, the proposed multiplier only requires approximately 3/4 gates compared with previous multipliers.

In this paper, we present an improvement of Chang et al. approach. Unlike previous multipliers, we apply a three-term Karatsuba-like formula [17, 18] to the polynomial multiplication which can save even more logic gates. On the other hand, it still maintains a relatively low time complexity with the use of redundant representation. In consequence, we can obtain a bit-parallel multiplier with approximately 8/9 gates of Chang et al. multiplier, while the time complexity is $T_A + (3 + \lceil \log_2 \frac{m}{3} \rceil)T_X$. For a large number of AOPs, this delay is equal to $T_A + (1 + \lceil \log_2(m-1) \rceil)T_X$ which matches the best results.

The rest of this paper is organized as follows: In Section 2, we briefly review the redundant representation for $GF(2^m)$ defined by the AOP. Then we propose a new bit-parallel multiplier architecture using the redundant representation in Section 3. Section 4 presents a comparison between the proposed multiplier and some others. Finally, some conclusions are drawn.

2 Redundant representation

The notion of redundant representation was firstly used by Itoh and Tsujii [16] for design of efficient multiplier. Later, such representation had been applied in many multipliers [19–21]. The idea was to use the minimal cyclotomic ring $\mathbb{F}_2[x]/(x^n+1)$ where the current field can be embedded and the field arithmetic operations are performed in such ring. If the field $GF(2^m)$ is generated with an irreducible AOP, we have $n = m+1$ which has the minimal redundant bits.

The irreducible AOP in the form $f(x) = \sum_{i=0}^{m} x^i$ over $\mathbb{F}_2$ exist if and only if $m+1$ is prime and 2 is primitive modulo $m+1$ [1]. The definition of redundant representation with respect to AOP is presented as follows:

Definition 1. *[15] Let x be a root of the irreducible AOP $f(x)$ of degree m. Then the set $\{1, x, x^2, \cdots, x^{m-1}\}$ induce a polynomial basis of $GF(2^m)$. Expand such set as $\{1, x, x^2, \cdots, x^m\}$. Any element $\alpha \in GF(2^m)$ is represented as*

$$\alpha = \sum_{i=0}^{m} \alpha_i x^i, \quad \alpha_i \in \mathbb{F}_2.$$

The representation is called as a redundant representation of $GF(2^m)$.

The modular reduction using redundant representation is very simple. Since $x^{m+1} = 1$, for any $A = \sum_{i=0}^{m} a_i x^i \in GF(2^m)$, we have

$$A \cdot x^i \bmod x^{m+1} + 1 = a_{m-i}x^m + \cdots + a_0 x^i + a_m x^{i-1} + \cdots + a_{m-i+1}.$$

Therefore, $A \cdot x^i$ can be computed by an i-bit left cyclic shift of A. In hardware implementation, this can be carried out by a simple rewiring and does not cost any logic gates.

3 Multiplier based on 3-term Karatsuba-like formula

The classic Karatsuba method [24] has been used to improved the efficiency of bit-parallel multiplier for $GF(2^m)$ generated by an AOP [15] and a trinomial [22, 23]. This method starts with a way to multiply two 2-term polynomials using three scalar multiplications, thus it can reduce the space complexity by approximately a factor of 3/4. However, there is also a way to multiply two 3-term polynomials using six scalar multiplications [17, 18]. In $\mathbb{F}_2[x]$, we can check that

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

In the following, we will use this approach, referred as 3-term Karatsuba-like formula, to computed the polynomial multiplication and to reduce the space complexity even further. Moreover, based on the property of redundant representation presented in Section 2, the proposed multiplier can also obtain a low time complexity.

Consider the field $GF(2^m)$ generated with an irreducible AOP $f(x)$. Let x be a root of $f(x)$, then we represent the field element $U \in GF(2^m)$ as $U = \sum_{i=0}^{m} u_i x^i,\ u_i \in \mathbb{F}_2$. Since $m+1$ is a prime, $m+1 \bmod 3 = 1$ or $m+1 \bmod 3 = 2$, then we have $3|m$ or $3|m-1$. In order to apply the 3-term Karatsuba-like formula, we split the polynomial into two blocks with one contains m bits (or $m-1$ bits) and another contains the left bits, then use this formula to m-bit (or $m-1$-bit) polynomial multiplications. We deal with these two cases separately.

(i) *when* $3|m$: Assume that $A = A'x + a'$, $B = B'x + b'$ where $A' = \sum_{i=0}^{m-1} a_i x^i, B' = \sum_{i=0}^{m-1} b_i x^i, a_i, b_i, a', b' \in \mathbb{F}_2$. The multiplication AB can be computed as:

$$\begin{aligned}AB &= (A'x + a')(B'x + b')\\ &= A'B'x^2 + (A'b' + B'a')x + a'b'.\end{aligned} \tag{1}$$

Let $k = m/3$ and we partition $A' = A_3x^{2k} + A_2x^k + A_1,\ B' = B_3x^{2k} + B_2x^k + B_1$ where $A_i = \sum_{j=0}^{k-1} a_{j+(i-1)k}x^j,\ B_i = \sum_{j=0}^{k-1} b_{j+(i-1)k}x^j$, for $i = 1, 2, 3$. Then we multiply A', B' with the 3-term Karatsuba formula and do some transformation

as follows:

$$\begin{aligned}A'B' &= (A_3x^{2k} + A_2x^k + A_1) \cdot (B_3x^{2k} + B_2x^k + B_1)\\ &= A_3B_3(x^{4k} + x^{3k} + x^{2k}) + A_2B_2(x^{3k} + x^{2k} + x^k)\\ &\quad + A_1B_1(x^{2k} + x^k + 1) + C_3D_3x^{3k} + C_2D_2x^{2k} + C_1D_1x^k\\ &= (A_3B_3x^{2k} + A_2B_2x^k + A_1B_1)(x^{2k} + x^k + 1)\\ &\quad + (C_3D_3x^{2k} + C_2D_2x^k + C_1D_1)x^k.\end{aligned}$$

where $C_3 = A_3 + A_2, C_2 = A_3 + A_1, C_1 = A_2 + A_1$ and $D_3 = B_3 + B_2, D_2 = B_3 + B_1, D_1 = B_2 + B_1$. Thus (1) can be rewritten as:

$$\begin{aligned}AB &= (A_3B_3x^{2k} + A_2B_2x^k + A_1B_1)(x^{2k+2} + x^{k+2} + x^2)\\ &\quad + (C_3D_3x^{2k} + C_2D_2x^k + C_1D_1)x^{k+2}\\ &\quad + (A'b' + B'a')x + a'b'.\end{aligned} \tag{2}$$

Let $S_1 = (A_3B_3x^{2k} + A_2B_2x^k + A_1B_1)(x^{2k+2} + x^{k+2} + x^2)$ and S_2 denote the left parts of (2). Then the modular multiplication $AB \bmod x^{m+1} + 1 = S_1 + S_2 \bmod x^{m+1} + 1$. We compute $S_1,\ S_2$ modulo $x^{m+1} + 1$ separately.

Firstly, we consider the computation of $(A_3B_3x^{2k} + A_2B_2x^k + A_1B_1) \bmod x^{m+1} + 1$. Let $A_1B_1 = (\sum_{i=0}^{k-1} a_ix^i) \cdot (\sum_{i=0}^{k-1} b_ix^i) = \sum_{i=0}^{2k-2} r_ix^i$. These r_is can be calculated as following equation:

$$r_i = \begin{cases} \sum_{j=0}^{i} a_jb_{i-j}, & 0 \le i \le k-1,\\ \sum_{j=i-k+1}^{k-1} a_jb_{i-j}, & k \le i \le 2k-2.\end{cases} \tag{3}$$

Similarly, we get the coefficients of $A_2B_2 = \sum_{i=0}^{2k-2} s_ix^i$ and $A_3B_3 = \sum_{i=0}^{2k-2} t_ix^i$ as:

$$s_i = \begin{cases} \sum_{j=0}^{i} a_{j+k}b_{i-j+k}, & 0 \le i \le k-1,\\ \sum_{j=i-k+1}^{k-1} a_{j+k}b_{i-j+k}, & k \le i \le 2k-2.\end{cases} \tag{4}$$

and

$$t_i = \begin{cases} \sum_{j=0}^{i} a_{j+2k}b_{i-j+2k}, & 0 \le i \le k-1,\\ \sum_{j=i-k+1}^{k-1} a_{j+2k}b_{i-j+2k}, & k \le i \le 2k-2.\end{cases} \tag{5}$$

Note that $k = m/3$ and $m + 1 = 3k + 1$, the expression $(A_3B_3x^{2k} + A_2B_2x^k + A_1B_1) \bmod x^{m+1} + 1 = \sum_{i=0}^{3k} z_ix^i$ can be obtained as follows:

$$z_i = \begin{cases} r_i + t_{i+k+1}, & 0 \le i \le k-3,\\ r_i, & k-2 \le i \le k-1,\\ s_{i-k} + r_i, & k \le i \le 2k-2,\\ s_{i-k}, & i = 2k-1,\\ s_{i-k} + t_{i-2k}, & 2k \le i \le 3k-2,\\ t_{i-2k}, & 3k-1 \le i \le 3k.\end{cases} \tag{6}$$

According to (3), (4), (5) and (6), we can find that, for $0 \le i \le k-2$ and $i = 3k$, z_i consists of sums of $k-1$ elements, for $k-1 \le i \le 3k-1$, z_i consists of sums of k elements. Therefore, circuit implementation of $\sum_{i=0}^{3k} z_ix^i$ requires

Table 1. Space and time complexities of $S_1 \bmod x^{m+1}+1$, if $3|m$

Operation	# AND	#XOR	Time delay
$A_3B_3x^{2k}+A_2B_2x^k+A_1B_1$	$3k^2$	$3k^2$	$T_A+\lceil \log_2 k \rceil T_X$
$S_1 \bmod x^{m+1}+1$	-	$5k+1$	$2T_X$

$k(k-1)+(2k+1)k = 3k^2$ XOR. Note that the computation of A_3B_3, A_2B_2, A_1B_1 also need $3k^2$ AND gates. Consequently, the space complexity for this operation is equal to $3k^2$ XOR and $3k^2$ AND, and time delay is $T_A + \lceil \log_2 k \rceil T_X$.

Moreover, based on previous description, the operation $(A_3B_3x^{2k}+A_2B_2x^k+A_1B_1)\cdot x^n$, for $n = 2, k+2, 2k+2$ can be computed by a n-bit left cyclic shift of $\sum_{i=0}^{3k} z_i x^i$, we can obtain the result by a simple rewiring without using any gates. Therefore,

$$\begin{aligned} S_1 \bmod x^{m+1}+1 = \textstyle\sum_{i=0}^{m} \Big(& z_{i+2 \bmod m+1} \\ & + z_{i+k+2 \bmod m+1} + z_{i+2k+2 \bmod m+1} \Big) x^i. \end{aligned} \tag{7}$$

It can be seen that the partial sums can be reused in the addition of (7) and $k+1$ XOR gates can be saved. The space and time complexities of $S_1 \bmod x^{m+1}+1$ are summarized in Table 1.

Then we consider the computation of

$$\begin{aligned} & S_2 \bmod x^{m+1}+1 \\ & \quad = (C_3D_3x^{2k}+C_2D_2x^k+C_1D_1)x^{k+2} \\ & \qquad +(A'b'+B'a')x+a'b' \bmod x^{m+1}+1. \end{aligned} \tag{8}$$

Since $(C_3D_3x^{2k}+C_2D_2x^k+C_1D_1)x^{k+2}$ modulo $x^{m+1}+1$ can be carried out by a $k+2$ -bit left cyclic shift of $C_3D_3x^{2k}+C_2D_2x^k+C_1D_1 \bmod x^{m+1}+1$, we only need to computed this expression. Note that $C_i, D_i, i = 1,2,3$ consist of the same bits as A_i, B_i, we just follow the same line as the computation in (6) to obtain the result. It also costs $3k^2$ XOR plus $3k^2$ AND, with time delay is $T_A + \lceil \log_2 k \rceil T_X$.

Table 2. Space and time complexities of $S_2 \bmod x^{m+1}+1$, if $3|m$

Operation	# AND	#XOR	Time delay
C_1, C_2, C_3	-	$3k$	T_X
D_1, D_2, D_3	-	$3k$	
$C_3D_3x^{2k}+C_2D_2x^k+C_1D_1$	$3k^2$	$3k^2$	$T_A+\lceil \log_2 k \rceil T_X$
$(A'b'+B'a')x+a'b'$	$2m+1$	m	T_A+T_X
$S_2 \bmod x^{m+1}+1$	-	$m+1$	T_X

Furthermore, note that $\deg A' = \deg B' = m-1$, the computation of $(A'b' + B'a')x + a'b'$ do not need any reduction. Thus such expression totally requires $2m+1$ AND gates plus m XOR gates and the time delay is $T_A + T_X$. The space and time complexities of the two expressions are presented in Table 2.

As seen in Table 1 and 2, the two operations have the same time delay thus they can be computed in parallel. Finally, we add together (7) and (8). This addition requires $m+1$ XOR gates and T_X gate delay. Thus the total space and time complexities of the proposed architecture can be calculated from Table 1 and 2 plus extra gates for the final addition:

$$\begin{aligned} \#\text{ AND} &= \tfrac{2m^2}{3} + 2m + 1, \\ \#\text{ XOR} &= \tfrac{2m^2}{3} + \tfrac{20m}{3} + 3, \\ \text{Time delay} &= T_A + (3 + \lceil \log_2 k \rceil T_X). \end{aligned}$$

Since $k = m/3$, we have $\lceil \log_2 k \rceil \leq \lceil \log_2 m \rceil - 1$. But this is interesting only if

$$\lceil \log_2 k \rceil < \lceil \log_2 m \rceil - 1. \tag{9}$$

This happens frequently when $m = 2^n + c$ where c is less than 2^{n-1}. In this case, the time delay of our architecture becomes $T_A + (1 + \lceil \log_2 m \rceil T_X)$. Since m is even, $\lceil \log_2 m \rceil = \lceil \log_2 (m-1) \rceil$, the time complexity can be rewritten as $T_A + (1 + \lceil \log_2 (m-1) \rceil T_X)$.

(ii) *when* $3|m-1$: In this case, assume that $A = A'x^2 + a'_1 x + a'_0$, $B = B'x^2 + b'_1 x + b'_0$ where $A' = \sum_{i=0}^{m-2} a_i x^i, B' = \sum_{i=0}^{m-2} b_i x^i$, for all the coefficients in $\mathbb{F}_2$. Let $k = (m-1)/3$ and we partition $A' = A_3 x^{2k} + A_2 x^k + A_1, B' = B_3 x^{2k} + B_2 x^k + B_1$ with each part consists of k elements. Similar with previous case, the multiplication AB is computed as:

$$\begin{aligned} AB = {} & (A_3 B_3 x^{2k} + A_2 B_2 x^k + A_1 B_1)(x^{2k+4} + x^{k+4} + x^4) \\ & + (C_3 D_3 x^{2k} + C_2 D_2 x^k + C_1 D_1) x^{k+4} + A'(b'_1 x^3 + b'_0 x^2) \\ & + B'(a'_1 x^3 + a'_0 x^2) + (a'_1 x + a'_0)(b'_1 x + b'_0). \end{aligned}$$

where $C_3 = A_3 + A_2, C_2 = A_3 + A_1, C_1 = A_2 + A_1$ and $D_3 = B_3 + B_2, D_2 = B_3 + B_1, D_1 = B_2 + B_1$. Then we divide above expression into two blocks and compute each block modulo $x^{m+1} + 1$, separately.

$$S_1 = (A_3 B_3 x^{2k} + A_2 B_2 x^k + A_1 B_1)(x^{2k+4} + x^{k+4} + x^4)$$

and

$$\begin{aligned} S_2 = {} & (C_3 D_3 x^{2k} + C_2 D_2 x^k + C_1 D_1) x^{k+4} + A'(b'_1 x^3 + b'_0 x^2) \\ & + B'(a'_1 x^3 + a'_0 x^2) + (a'_1 x + a'_0)(b'_1 x + b'_0). \end{aligned}$$

The computation processes are nearly the same as those presented in the case of $3|m$. We can easily compute $S_1,\ S_2$ modulo $x^{m+1} + 1$ using the similar strategy. The space and time complexities with respect to the two expressions above are summarized in Table 3 and Table 4.

Table 3. Space and time complexities of $S_1 \bmod x^{m+1}+1$, if $3|m-1$

Operation	# AND	#XOR	Time delay
$A_3B_3x^{2k}+A_2B_2x^k+A_1B_1$	$3k^2$	$3k^2$	$T_A+\lceil\log_2 k\rceil T_X$
$S_1 \bmod x^{m+1}+1$	-	$5k+3$	$2T_X$

Table 4. Space and time complexities of $S_2 \bmod x^{m+1}+1$, if $3|m-1$

Operation	# AND	#XOR	Time delay
C_1, C_2, C_3	-	$3k$	T_X
D_1, D_2, D_3	-	$3k$	
$C_3D_3x^{2k}+C_2D_2x^k+C_1D_1$	$3k^2$	$3k^2$	$T_A+\lceil\log_2 k\rceil T_X$
$A'(b'_1x^3+b'_0x^2)+B'(a'_1x^3$ $+a'_0x^2)+(a'_1x+a'_0)(b'_1x+b'_0)$	$4m-2$	$3m-2$	T_A+3T_X
$S_2 \bmod x^{m+1}+1$	-	$m+1$	T_X

Note that, in practical application $\lceil\log_2 k\rceil$ is usually larger than 3, thus operations of the third and forth row in Table 4 can be carried out in parallel with time delay $T_A+\lceil\log_2 k\rceil T_X$. In the end, another $m+1$ XOR gates and T_X delay should be added to compute the final result. Thus the total space complexity and time complexity of the proposed architecture can be calculated from Tables 3, 4 and extra gates on adding $S_1 \bmod x^{m+1}+1$ and $S_2 \bmod x^{m+1}+1$ together:

$$\begin{gathered}\#\text{ AND} = \tfrac{2m^2}{3}+\tfrac{8m}{3}-\tfrac{4}{3},\\ \#\text{ XOR} = \tfrac{2m^2}{3}+\tfrac{22m}{3},\\ \text{Time delay} = T_A+(3+\lceil\log_2(\tfrac{m-1}{3})\rceil T_X).\end{gathered}$$

Furthermore, if $m-1=2^n+c$ with $c\le 2^{n-1}$, the time delay of our architecture can be rewritten as $T_A+(1+\lceil\log_2(m-1)\rceil T_X)$.

4 Comparison

Table 5 gives a comparison of different implementations of bit-parallel multipliers in the class of fields defined by an irreducible AOP. From Table 5, we can see that our multiplier requires about 33% fewer circuit gates than the previous five architectures and about 10% fewer circuit gates than Chang et al. multiplier. On the other hand, the time complexity of the proposed multiplier is $T_A+(3+\lceil\log_2(\frac{m}{3})\rceil)T_X$. In fact, we have searched all irreducible AOP with degree m less than 8000 and it is found that for more than 50% AOPs satisfy the inequality (9). In this case, the time delay of our architecture is equal to $T_A+(1+\lceil\log_2(m-1)\rceil)T_X$ which matches the best results.

In addition, there are many general bit-parallel multipliers for the field $GF(2^m)$ generated with other irreducible polynomials. Those multipliers can be divided

Table 5. Comparison of bit-parallel Multipliers for $GF(2^m)$ generated with AOP

Multiplier	# AND	#XOR	Time delay
Hasan [3]	m^2	m^2-1	$T_A+(1+\lceil\log_2(m-1)\rceil)T_X$
Koç [2]	m^2	m^2-1	$T_A+(2+\lceil\log_2(m-1)\rceil)T_X$
Wu [9]	m^2	m^2-1	$T_A+(1+\lceil\log_2(m-1)\rceil)T_X$
Kim [10]	m^2	m^2-1	$T_A+(1+\lceil\log_2(m-1)\rceil)T_X$
Reyhani-Masoleh [8]	m^2	m^2-1	$T_A+(1+\lceil\log_2(m-1)\rceil)T_X$
Chang [15]	$\frac{3m^2}{4}+2m+1$	$\frac{3m^2}{4}+3m+1$	$T_A+(1+\lceil\log_2(m+1)\rceil)T_X$
$3\|m$	$\frac{2m^2}{3}+2m+1$	$\frac{2m^2}{3}+\frac{20m}{3}+3$	$T_A+(3+\lceil\log_2(\frac{m}{3})\rceil)T_X$
$3\|m-1$	$\frac{2m^2}{3}+\frac{8m}{3}-\frac{4}{3}$	$\frac{2m^2}{3}+\frac{22m}{3}$	$T_A+(3+\lceil\log_2(\frac{m-1}{3})\rceil)T_X$

into two categories according to space complexity, namely, quadratic multipliers and subquadratic multipliers. The best multiplier for the first type contains at least m^2 AND and m^2-1 XOR gates with time delay of $T_A+(1+\lceil\log_2 m\rceil)T_X$ [25] (for good field it is equal to $T_A+\lceil\log_2 m\rceil T_X$). The multiplier of the second type contains about $O(m^{\log_2 3})$ circuit gates and usually requires at least $T_A+(1+2\lceil\log_2 m\rceil)T_X$ gates delay [26, 27]. It is obvious that our architecture still has certain advantage compared with other general multipliers. The time complexity of our multiplier is comparable to the fastest bit-parallel multiplier and much faster than the subquadratic multipliers, but the space complexity is smaller than the fastest ones [5]. Therefore, our architecture offers a proper tradeoff between space and time complexity.

In [28], Ciet et al. resuscitate elliptic curve cryptography (ECC) over the field which is generated with an irreducible AOP. They have proposed a sequence of such finite fields which are secure for ECC. Among the four examples suggested for ECC in [28], there are two fields, namely, $GF(2^{178})$ and $GF(2^{1186})$, which satisfy the inequality (9). In these cases, our architecture outperforms Chang et al. approach by saving 10% logic gates with the same time delay. The details are presented in Table 6.

Table 6. Complexity for practical field defined by irreducible AOP

Field	our method			Chang et al.		
	#AND	#XOR	Time	#AND	#XOR	Time
$GF(2^{178})$	21596	22428	T_A+9T_X	24120	24298	T_A+9T_X
$GF(2^{1186})$	940892	946428	T_A+12T_X	1057320	1058506	T_A+12T_X

5 Conclusion

In this paper, a new bit-parallel multiplier architecture for all-one polynomial is proposed. In the proposed architecture, redundant representation and a 3-

term Karatsuba-like formula are combined which can reduce the time complexity and space complexity, respectively. This multiplier can be used in area-critical occasion because it has low space complexity but maintains a relatively low time delay.

Moreover, besides the 3-term Karatsuba-like formula, other Karatsuba-like formulas could also be used for the optimization. We are currently working on applying the five and seven term Karatsuba-like formulae to the AOP multiplier.

References

1. A.J. Menezes, I.F. Blake, X. Gao, R.C. Mullin, S.A. Vanstone, and T. Yaghoobian, Applications of Finite Fields, Kluwer Academic, Norwell, Massachusetts, USA, 1993.
2. Ç. K. Koç, B. Sunar, Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields, IEEE Trans. Comput. 47 (3) (1998) 353–356.
3. M. A. Hasan, M. Z. Wang, V. K. Bhargava, A modified massey-omura parallel multiplier for a class of finite fields, IEEE Trans. Comput. 42 (10) (1993) 1278–1280.
4. B. Sunar, Ç. K. Koç, Mastrovito multiplier for all trinomials, IEEE Trans. Comput. 48 (5) (1999) 522–527.
5. H. Fan, M. A. Hasan, Fast bit parallel shifted polynomial basis multiplier in $GF(2^n)$, IEEE Trans. Circuits and Systems I, Foundamental Theory and Applications, 53 (12) (2006) 2606–2615.
6. F. Rodríguez-Henríquez, Çetin Kaya Koç, Parallel multipliers based on special irreducible pentanomials, IEEE Trans. Comput. 52 (12) (2003) 1535–1542.
7. A. Reyhani-Masoleh, M. A. Hasan, A new construction of massey-omura parallel multiplier over $GF(2^m)$, IEEE Trans. Comput. 51 (5) (2002) 511–520.
8. A. Reyhani-Masoleh, M. A. Hasan, Efficient multiplication beyond optimal normal bases, IEEE Trans. Comput. 52 (4) (2003) 428–439.
9. H. Wu, M. A. Hasan, Low complexity bit-parallel multipliers for a class of finite fields, IEEE Trans. Comput. 47 (8) (1998) 883–887.
10. C. H. Kim, S. Oh, J. Lim, A new hardware architecture for operations in $GF(2^m)$, IEEE Trans. Comput. 51 (1) (2002) 90–92.
11. M. Leone, A new low complexity parallel multiplier for a class of finite fields, in: CHES '01: Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems, Springer-Verlag, London, UK, 2001, pp. 160–170.
12. H.-S. Kim, S.-W. Lee, LFSR multipliers over $GF(2^m)$ defined by all-one polynomial, Integr. VLSI J. 40 (4) (2007) 473–478.
13. P. K. Meher, Y. Ha, C.-Y. Lee, An optimized design for serial-parallel finite field multiplication over $GF(2^m)$ based on all-one polynomials, in: ASP-DAC '09: Proceedings of the 2009 Asia and South Pacific Design Automation Conference, IEEE Press, Piscataway, NJ, USA, 2009, pp. 210–215.
14. H.-S. Kim, S.-W. Lee, Area and time efficient AB^2 multipliers based on cellular automata, Comput. Stand. Interfaces 31 (1) (2009) 137–143.
15. K.-Y. Chang, D. Hong, H.-S. Cho, Low complexity bit-parallel multiplier for $GF(2^m)$ defined by all-one polynomials using redundant representation, IEEE Trans. Comput. 54 (12) (2005) 1628–1630.

16. T. Itoh, S. Tsujii, Structure of parallel multipliers for a class of fields $GF(2^m)$, Inf. Comput. 83 (1) (1989) 21–40.
17. P. L. Montgomery, Five, six, and seven-term karatsuba-like formulae, IEEE Trans. Comput. 54 (3) (2005) 362–369.
18. 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.pdf`
19. H. Wu, M. A. Hasan, I. F. Blake, S. Gao, Finite field multiplier using redundant representation, IEEE Trans. Comput. 51 (11) (2002) 1306–1316.
20. A. H. Namin, H. Wu, M. Ahmadi, A new finite-field multiplier using redundant representation, IEEE Trans. Comput. 57 (5) (2008) 716–720.
21. A. H. Namin, H. Wu, M. Ahmadi, A high-speed word level finite field multiplier in F_{2^m} using redundant representation, IEEE Trans. Very Large Scale Integr. Syst. 17 (10) (2009) 1546–1550.
22. M. Elia, M. Leone, C. Visentin, Low complexity bit-parallel multipliers for $GF(2^m)$ with generator polynomial $x^m + x^k + 1$, Electronic Letters 35 (7) (1999) 551–552.
23. H. Shen, Y. Jin, Low complexity bit parallel multiplier for $GF(2^m)$ generated by equally-spaced trinomials, Inf. Process. Lett. 107 (6) (2008) 211–215.
24. D. E. Knuth, Art of Computer Programming, Volume 2: Seminumerical Algorithms, 3rd Edition. Addison-Wesley Professional, 1997.
25. H. Fan, J. Sun, M. Gu, K.-Y. Lam, Overlap-free Karatsuba-Ofman polynomial multiplication algorithms, IET Inf. Secur. 4 (1) (2010) 8–14.
26. H. Fan, M. A. Hasan, A new approach to subquadratic space complexity parallel multipliers for extended binary fields, IEEE Trans. Comput. 56 (2) (2007) 224–233.
27. H. Fan, M. A. Hasan, Subquadratic computational complexity schemes for extended binary field multiplication using optimal normal bases, IEEE Trans. Comput. 56 (10) (2007) 1435–1437.
28. M. Ciet, J.-J. Quisquater, F. Sica, A Secure Family of Composite Finite Fields Suitable for Fast Implementation of Elliptic Curve Cryptography, Proc. Indocrypt 2001, (2001) 108–116.