디지털산업정보학회 논문지 제16권 제2호-2020년 6월
http://dx.doi.org/10.17662/ksdim.2020.16.2.001

Resource and Delay Efficient Polynomial Multiplier over Finite Fields $GF(2^m)$

Lee Keonjik*

유한체상의 자원과 시간에 효율적인 다항식 곱셈기

이 건 직

〈Abstract〉

Many cryptographic and error control coding algorithms rely on finite field GF(2m) arithmetic. Hardware implementation of these algorithms needs an efficient realization of finite field arithmetic operations. Finite field multiplication is complicated among the basic operations, and it is employed in field exponentiation and division operations. Various algorithms and architectures are proposed in the literature for hardware implementation of finite field multiplication to achieve a reduction in area and delay. In this paper, a low area and delay efficient semi-systolic multiplier over finite fields GF(2m) using the modified Montgomery modular multiplication (MMM) is presented. The least significant bit (LSB)-first multiplication and two-level parallel computing scheme are considered to improve the cell delay, latency, and area-time (AT) complexity. The proposed method has the features of regularity, modularity, and unidirectional data flow and offers a considerable improvement in AT complexity compared with related multipliers. The proposed multiplier can be used as a kernel circuit for exponentiation/division and multiplication.

Key Words : Modular Multiplication, Finite Field Arithmetic, Systolic Array

Ⅰ. 서론

Finite fields $GF(2^m)$ has many applications in areas such as coding theory and cryptography [1, 2]. However, the finite field size is relatively large for use with elliptic curve cryptography (ECC) and the efficient field arithmetic implementations affect the performance of ECC. Among them, multiplication is an essential arithmetic. Addition is relatively inexpensive, whereas multiplication is more costly in terms of delay and circuit complexity. Other operations such as division and exponentiation can be performed by repetitive multiplications. It is thus desirable to design $GF(2^m)$ multiplier with low AT complexity, where AT

* 대구대학교 자유전공학부 조교수 (단독저자)

complexity measures depend on two factors, chip area (A) and computation time (T). Hardware complexity is associated with a tradeoff between area and time, instead of one of the two measures. The area-time product complexity is the balanced parameter for overall performance comparison of various architectures rather than area and time individually.

$GF(2^m)$ can be viewed as an m-dimensional vector over $GF(2)$ whose elements are 0 and 1. The elements of $GF(2^m)$ are represented using a set of m specific independent field elements called basis of $GF(2^m)$. Several bases such as polynomial basis, dual basis, normal basis, and redundant basis are available. Among them, arithmetic in polynomial basis is more regular, simple, and scalable in hardware implementation. The performance of $GF(2^m)$ multiplier depends on the irreducible polynomial selected. Irreducible polynomials are classified into generic polynomial, equally spaced polynomial, trinomial, and pentonomial. $GF(2^m)$ multipliers based on trinomial and pentanomial are more efficient, while multipliers based on generic polynomial are applicable for a wider range of applications. Depending on the style of implementation various $GF(2^m)$ multiplier can be developed. Based on the input type, the structures can be bit-level or digit-level. Digit-level architectures process a group of bits called a digit at a time. These architectures are complicated and facilitate the area-time trade-o ff. Parallel architectures generate output in one clock cycle at the cost of excessive hardware. Among them, systolic array architectures have advantages such as modularity, regularity, concurrency and local interconnections. These structures are more suitable for VLSI implementation and provide high throughput while their area and latency are usually very large.

Several systolic multipliers for $GF(2^m)$ have been reported [3-11]. Lee et al. [3] and Chiou et al. [4] proposed a semi-systolic multiplier with concurrent error detection. Huang et al. [5] proposed a semi-systolic multiplier to reduce the time and space complexities. Kim and Kim [6] proposed an area-efficient semi-systolic multiplier. Choi and Lee [7] proposed the AT efficient parallel and serial systolic unified multiplier/ squarer, which involves little hardware overhead, based on the effective LSB-first multiplication and LSB-first exponentiation. Chiou et al. [8] proposed a semi- systolic multiplier to reduce the time complexity. Lee and Kim [9] proposed a semi-systolic multiplier using redundant basis. Mathe and Boppana [10] proposed a sequential polynomial basis multiplier for generic irreducible polynomials with a latency of m clock cycles. This architecture is designed to take one operand in parallel and another operand serially during computation. Ibrahim [11] proposed efficient parallel and serial systolic structures for combined multiplication and squaring over $GF(2^m)$. The proposed structures have the advantage of computing both modular multiplication and squaring simultaneously for fast execution of modular exponentiation. However, their high circuit complexities and long delays are crucial limitations in cryptographic applications. Thus, further research on efficient multiplication with low space and time complexities is required.

In this paper, we propose a low AT complexity semi-systolic MMM multiplier over $GF(2^m)$.

Applying the LSB-first scheme and two-level parallel computation to MMM can achieve obvious improvement in AT complexity compared to the related semi-systolic multipliers. The proposed multiplier can be used as a kernel circuit for exponentiation/division and multiplication.

Ⅱ. Montgomery multiplication in $GF(2^m)$

Let $G(z) = \sum_{j=1}^{m+1} g_{j-1} z^{j-1}$ be the irreducible polynomial generating the finite field $GF(2^m)$, where $g_m = g_0 = 1$ and $g_j \in \{0, 1\}$ for $j = 1$ to $m - 1$. Assuming x is a root of $G(z)$, i.e., $G(x) = 0$, each element in $GF(2^m)$ is represented as a polynomial of degree less than m over $GF(2)$, e.g., $A \in GF(2^m)$ is represented $A = a_{m-1}x^{m-1} + \ldots + a_1 x + a_0$, where $a_j \in \{0, 1\}$ for $j = 0$ to $m - 1$. For simplicity, $G(x)$ is abbreviated as G. The addition is trivial because it can be performed with bit-wise XORing two operands; however, the multiplication is more complicated because the intermediate partial product needs further reduction by $x^m = \sum_{j=1}^{m} g_{j-1} x^{j-1}$. The MMM algorithm was originally proposed for efficient integer modular multiplication [12]. Later, it was shown that MMM is also applicable to $GF(2^m)$ [13]. The idea is to transform input operands to MMM residues, and then compute the multiplication with these residues. Finally, the result is converted back to the normal representation. Let a and β be two elements of $GF(2^m)$ to be multiplied. Instead of computing $T' = a\beta \bmod G$, using an ingenious representation of the residue class modulo G, MMM of A ($= aR \bmod G = \sum_{j=1}^{m} a_{j-1} x^{j-1}$) and B ($= \beta R \bmod G = \sum_{j=1}^{m} b_{j-1} x^{j-1}$) is given by $T = ABR^{-1} \bmod G = \sum_{j=1}^{m} t_{j-1} x^{j-1}$, where A (resp., B) is the MMM residue of a (resp., β) and R is a special fixed element of $GF(2^m)$ satisfying $\gcd(R, G) = 1$. The correct result T' is then obtained by executing MMM using arguments T and 1, i.e., $T' = TR^{-1} \bmod G = a\beta \bmod G$. Accordingly, MMM is only beneficial if we compute several consecutive multiplications in the exponentiation, owing to the pre- and post-processing requirements. In this paper, an efficient LSB-first semi-systolic multiplication scheme is considered and we assume m is odd because in practical applications it is more significant than an even m. Efficient implementation of ECC needs that m is odd, or more strongly m is prime. For example, all the five binary fields $GF(2^m)$, m = 163, 233, 283, 409, 571 suggested by National Institute of Standards and Technology (NIST) for ECC have the property that m = odd. In that case we define $R = x^{(m-1)/2}$ so that $T = ABR^{-1} \bmod G$ can be formulated as the sum of two independently computed polynomials C and D, where $C = \sum_{j=1}^{m} c_{j-1} x^{j-1}$ and $D = \sum_{j=1}^{m} d_{j-1} x^{j-1}$, as follows:

$$T = \left(\sum_{j=1}^{m} b_{j-1} x^{j-1}\right) A x^{-(m-1)/2} \bmod G$$
$$= \sum_{j=(m+1)/2}^{m} b_{j-1} A x^{(2j-m-1)/2} \bmod G \quad (1)$$
$$+ \sum_{j=(m-1)/2}^{1} b_{j-1} A x^{(2j-m-1)/2} \bmod G \quad (2)$$

We first consider (1) for $i \in [1, (m+1)/2]$. Let $A^{(i)} = A^{(i-1)} x \bmod G$, where $A^{(i)} = \sum_{j=1}^{m} a_{j-1}^{(i)} x^{j-1}$ denotes the intermediate result at the ith iteration and $A^{(0)} = A$. Then, $A^{(i)}$ can be expressed in expanded form as

$$\begin{aligned} A^{(i)} &= \left(\sum_{j=1}^{m} a_{j-1}^{(i-1)} x^{j-1}\right) x \bmod G \\ &= \sum_{j=1}^{m} a_{j-1}^{(i-1)} x^{j} \bmod G \\ &= a_{m-1}^{(i-1)} \sum_{j=1}^{m} g_{j-1} x^{j-1} + \sum_{j=1}^{m-1} a_{j-1}^{(i-1)} x^{j} \end{aligned} \quad (3)$$

It is clear that (3) is obtained by shifting the coefficients of $A^{(i-1)}$ to the left (i.e., the MSB direction) by one and reducing the term $a_{m-1}^{(i-1)} x^m$ by G. Therefore, we have the coefficients of $A^{(i)}$ as follows:

$$a_j^{(i)} = a_{j-1}^{(i-1)} + a_{m-1}^{(i-1)} g_j, \; m \geq j \geq 1, \quad (4)$$

where $a_j^{(0)} = a_j$ for $m-1 \geq j \geq 0$ and $a_0^{(i)}$ is set to $a_{m-1}^{(i-1)}$ for $1 \leq i \leq (m-1)/2$. Now, (1) can be represented as

$$C^{(i)} = C^{(i-1)} + b_{(m+2i-3)/2} A^{(i-1)}, \quad (5)$$

where $C^{(i)} = \sum_{j=1}^{m} c_{j-1}^{(i)} x^{j-1}$ denotes the ith intermediate result. Then, $C^{(i)}$ can be rewritten as

$$\begin{aligned} C^{(i)} &= \sum_{j=1}^{m} c_{j-1}^{(i)} x^{j-1} \\ &= \sum_{j=1}^{i} b_{(m+2j-3)/2} A x^{j-1} \\ &= \sum_{j=1}^{m} c_{j-1}^{(i-1)} x^{j-1} + b_{(m+2i-3)/2} A x^{i-1} \\ &= \sum_{j=1}^{m} c_{j-1}^{(i-1)} x^{j-1} \\ &\quad + b_{(m+2i-3)/2} \sum_{j=1}^{m} a_{j-1}^{(i-1)} x^{j-1} \end{aligned} \quad (6)$$

where $C^{(0)} = 0$ and (6) is easily obtained by multiplication and accumulation operations [7]. From (6), we have the coefficients of $C^{(i)}$ as follows:

$$c_{j-1}^{(i)} = c_{j-1}^{(i-1)} + b_{(m+2i-3)/2} a_{j-1}^{(i-1)}, \; m \geq j \geq 1, \quad (7)$$

where $c_j^{(0)} = 0$ for $m-1 \geq j \geq 0$, $a_j^{(0)} = a_j$ for $m-1 \geq j \geq 0$, and $a_0^{(i)} = a_{m-1}^{(i-1)}$ for $1 \leq i \leq (m-1)/2$. After $(m+1)/2$ iterations, the result $C^{(m+1)/2}$ is obtained. Note that (4) and (7) are also independent and can thus be computed in parallel, where the common term $a_{j-1}^{(i-1)}$ is used in both equations.

Next, we consider (2) for $i \in [1, (m+1)/2]$ and $j \in [1, m]$. In this case, let $A^{(i)} = A^{(i-1)} x^{-1} \bmod G$, where $A^{(i)} = \sum_{j=1}^{m} a_{j-1}^{(i)} x^{j-1}$ and $A^{(0)} = A$. For any irreducible polynomial, $g_0 = g_m = 1$, and we know that x is a root of G. Using this fact, multiplying both sides of $G = \sum_{j=1}^{m+1} g_{j-1} x^{j-1} = 0$ by x^{-1} and rearranging the terms, we can obtain $x^{-1} = \sum_{j=1}^{m} g_j x^{j-1}$. Using x^{-1}, $A^{(i)}$ can be derived as

$$\begin{aligned} A^{(i)} &= \left(\sum_{j=1}^{m} a_{j-1}^{(i-1)} x^{j-1}\right) x^{-1} \bmod G \\ &= \sum_{j=1}^{m} a_{j-1}^{(i-1)} x^{j-2} \bmod G \\ &= a_0^{(i-1)} \sum_{j=1}^{m} g_j x^{j-1} + \sum_{j=2}^{m} a_{j-1}^{(i-1)} x^{j-2} \end{aligned} \quad (8)$$

Equation (8) is computed by shifting $A^{(i-1)}$ to the LSB direction by one and reducing $a_0^{(i-1)} x^{-1}$ by G. From (8), the coefficients of $A^{(i)}$ are obtained as

$$a_{m-j-1}^{(i)} = a_{m-j}^{(i-1)} + a_0^{(i-1)} g_{m-j}, \; m \geq j \geq 1, \quad (9)$$

where $a_j^{(0)} = a_j$ for $m-1 \geq j \geq 0$ and $a_{m-1}^{(i)}$ is set to $a_0^{(i-1)}$ for $1 \leq i \leq (m-1)/2$. Now, (2) can be expressed as $D^{(i)} = D^{(i-1)} + b_{(m-2i+1)/2} A^{(i-1)}$. $D^{(i)}$ can be written in expanded form

$$\begin{aligned} D^{(i)} &= \sum_{j=1}^{m} d_{j-1}^{(i)} x^{j-1} \\ &= \sum_{j=1}^{i} b_{(m-2j+1)/2} A x^{j-1} \\ &= \sum_{j=1}^{m} d_{j-1}^{(i-1)} x^{j-1} + b_{(m-2i+1)/2} A x^{i-1} \\ &= \sum_{j=1}^{m} d_{j-1}^{(i-1)} x^{j-1} \\ &\quad + b_{(m-2i+1)/2} \sum_{j=1}^{m} a_{j-1}^{(i-1)} x^{j-1} \end{aligned} \quad (10)$$

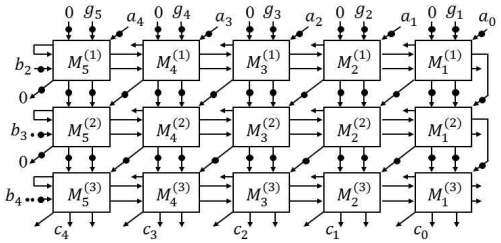
where $D^{(0)} = 0$. We obtain the coefficients of $D^{(i)}$ as

$$d_{m-j}^{(i)} = d_{m-j}^{(i-1)} + b_{(m-2i+1)/2} a_{m-j}^{(i-1)},\ m \geq j \geq 1, \qquad (11)$$

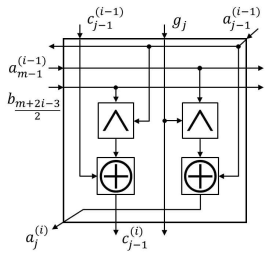
where $d_j^{(0)} = 0$ for $m-1 \geq j \geq 0$, $b_{(m-1)/2} = 0$, $a_j^{(0)} = a_j$ for $m-1 \geq j \geq 0$, and $a_{m-1}^{(i)} = a_0^{(i-1)}$ for $1 \leq i \leq (m-1)/2$. Note that both (9) and (11) can be computed concurrently and involve a common term $a_{m-j}^{(i-1)}$. Finally, to obtain the MMM product T as output, $C^{(m+1)/2}$ and $D^{(m+1)/2}$ need to be added by m 2-input XOR gates (XOR_2).

Based on (4) and (7), the semi-systolic array for C is shown in <Figure 1>. It consists of $m \times (m+1)/2$ M-cells of <Figure 2> and involves unidirectional data flow, where $m = 5$. A system with unidirectional data flow has advantages over a system with contraflow in terms of chip cascadability, fault tolerant design, and possible integration [14]. One 1-bit latch (denoted by "•") is placed at each vertical link and diagonal link. The basic cell at position (i, j) performs the logic operations described in (4) and (7). The initial positions and index points of each input are as follows ($1 \leq i \leq (m+1)/2$). First, b_{i+1} enters index $[i, m]$ from the left side and flows in the direction $[0, -1]$. Next, c_j ($0 \leq j \leq m-1$) enters index $[1, j+1]$ from the top and its computation result flows in the direction [1, 0], where $c_j = 0$.

Then, g_j ($1 \leq j \leq m$) enters index $[1, j]$ from the top and flows in the direction [1, 0]. Next, a_j ($0 \leq j \leq m-1$) enters $[1, j+1]$ from the top and its result flows in the direction [1, 1]. Note that $a_{m-1}^{(i-1)}$ generated from $M_{m-1}^{(i-1)}$ cell flows in the direction $[0, -1]$. The output $C^{(m+1)/2}$ emerges in parallel from the bottom row of the array, where the output order is normally maintained.



<Figure 1> Array architecture for C



<Figure 2> Circuit of $M_j^{(i)}$cell

Each basic cell includes two 2-input AND gates (AND_2) and two XOR_2, as shown in <Figure 2>, where ⊕ and ∧ denote XOR_2 and AND_2, respectively. $M_j^{(i)}$ cell receives $a_{j-1}^{(i-1)}$ as its input from $M_{j-1}^{(i-1)}$; $c_{j-1}^{(i-1)}$ and g_j from $M_j^{(i-1)}$; and $a_{m-1}^{(i-1)}$ and $b_{(m+2i-3)/2}$ from $M_{j+1}^{(i)}$. In <Figure 1>, the left side input b_{i+2} is delayed by one clock cycle relative to b_{i+1} for $1 \leq i \leq (m-1)/2$. Note that the structure and function of the semi-systolic array for D based on (9) and (11) are the same as those depicted in <Figure 1> and <Figure 2>; thus, they are not considered in detail.

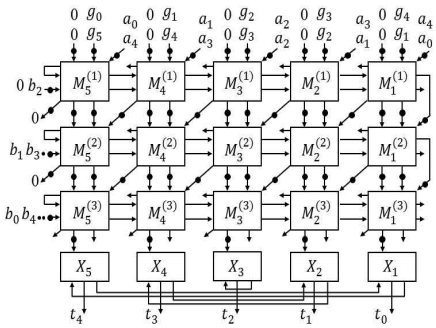
As stated earlier, because C and D have no data dependency on each other and have identical and

independent computation structures, we can implement them in a pipelined fashion using a single array. As a result, the proposed unified semi-systolic multiplier is shown in <Figure 3>, where all the inputs for D are staggered by one clock cycle relative to all the inputs for C. <Figure 3(a)> consists of two kinds of cells; one generates temporary results C and D using $m \times (m+1) / 2$ M-cells (<Figure 3(b)>), and the other adds them to yield the correct answer T using m X-cells (<Figure 3(c)>), where each X cell includes one XOR_2 and one 1-bit latch. Each t_j $(m-1 \geq j \geq 0)$ is obtained through m parallel additions of X cells, where one delay element is required at each X cell to synchronize the arrival of data at the X cell. Note that $A^{(m+1)/2}$ computed in (4) and (9) is of no use. The latency of the proposed multiplier is $(m+7) / 2$ clock cycles, and the propagation delay is the total delay of one AND_2 and one XOR_2.

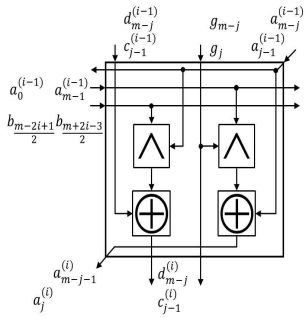
Ⅲ. Analysis and comparison

In this section, the area and time complexities of the proposed systolic multiplier are compared with that of similar multipliers available in the literature. In <Table 1>, the analytical expressions for area and time complexities of the proposed multiplier together with that of available multipliers are presented. The analytical comparisons presented in <Table 1> can be better understood by evaluating them for a specific value of field order along with a specific technology-based area and time complexity estimations of gates. The analytical expressions presented in <Table 1> are evaluated for m = 571 using the area and time complexity estimations of logic gates built from Samsung electronics company, where m = 571 is one of the field sizes recommended by NIST for ECC applications.

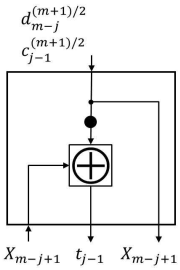
We obtained the area of the logic gates and their delays from the Samsung STD 150 0.13μm 1.2V CMOS Standard Cell Library databook. The following summarizes the time and area requirements of the cells used in our analysis, where T denotes the time (ns), A denotes the area (transistor count), and Gate$n$ denotes the n-input logic cell gate, respectively [7]: T_{AND2}: 0.094, A_{AND2}: 6.68, T_{XOR2}: 0.167, A_{XOR2}: 12.00, T_{LATCH}: 0.157, and



<Figure 3> (a) Proposed multiplier



(b) $M_j^{(i)}$ cell



(c) X_j cell

A_{LATCH}: 16.00.

<Table 1> presents the analytical comparison of latency, data flow, throughput, area complexity, time complexity, AT complexity, and improvement of the proposed multiplier with the multipliers considered for comparison. The area required for the proposed multiplier is computed in terms of number of AND_2 gates, XOR_2 gates, XOR_3 gates, and latches. We also present the area complexity for the other related systolic-type multipliers in a similar way to compare with that of the proposed multiplier. The expressions for area complexities of the proposed multiplier are derived using <Figure 3>. <Fig. 3(a)> contains $m(m+1)/2$ M-cells and m X-cells. Each M-cell contains two AND_2, two XOR_2, and three 1-bit latches. Each X-cell includes one XOR_2 and two 1-bit latches. Hence, the proposed multiplier architecture requires $m^2 + m$ AND_2 gates, $m^2 + 2m$ XOR_2 gates, and $1.5m^2 + 3.5m$ 1-bit latches. Using A_{AND2}, A_{XOR2}, and A_{LATCH}, the transistor-count of the proposed multiplier is $42.7m^2 + 86.7m$

<Table 1> presents the comparison of estimated transistor -count, total delay, and AT of the proposed multiplier. It is observed that the proposed multiplier requires the least number of transistors. It is clear from <Table 1> (% reduction in #transistors row) that the proposed multiplier achieves area efficiency of 50%, 50%, 53%, and 16% when compared with multipliers [5, 6, 8, 9], respectively.

The cell delay (i.e., critical path delay) and latency of the proposed multiplier are 0.26 ns and $(m+7)/2$ clock cycles, respectively and the total delay is (cell delay) × (latency). <Table 1> (% reduction in total delay row) shows that the proposed multiplier achieves time efficiency of 49%,

<Table 1> Complexity comparison of semi-systolic multipliers

Multipliers	Huang et al.[5]	Kim-Kim [6]	Chiou et al.[8]		Lee-Kim [9]	Proposed	
# cells	m^2	S:$m(m-1)/2$, V:m	U:$m(m-2)$, V:$2m$		$(m^2+3m)/2+1$	M:$m(m+1)/2$, X:m	
Latency	$m+1$	$(m+3)/2$	$m+1$		$m/2+3$	$(m+7)/2$	
Data flow	bidirectional	bidirectional	unidirectional		unidirectional	unidirectional	
Throughput	1	1	1		1	1	
Area complexity		S	U	V		M	X
#AND_2	$2m^2$	$2(m^2-m)$	m^2-2m	$2m$	m^2+3m+2	m^2+m	0
#XOR_2	$2m^2$	m^2-m	$3(m^2-2m)$	$8m$	m^2+4m+3	m^2+m	m
#XOR_3	0	$(m^2-m)/2$	0	0	0	0	0
#Latch	$3m^2+m+1$	$3m^2-2m-1$	$3(m^2-2m)$	$6m$	$2m^2+8m+6$	$1.5(m^2+m)$	$2m$
# transistors	$85.4m^2+16(m+1)$	$85.4m^2+56m$	$90.7m^2+24m$		$50.7m^2+196.1m+145.4$	$42.7m^2+86.7m$	
Time complexity							
Cell delay	0.26	0.43	0.75		0.26	0.26	
Total delay	$0.26m+0.26$	$0.22m+0.66$	$0.75m+0.75$		$0.13m+0.78$	$0.13m+0.91$	
AT complexity	$22.2m^3+26.4m^2+8.3m+4.2$	$18.8m^3+68.7m^2+37m$	$68.0m^3+86.0m^2+18m$		$6.6m^3+65.0m^2+171.8m+113.4$	$5.6m^3+50.1m^2+78.9m$	
Improvement							
Area	50%	50%	53%		16%	-	
Time	49%	40%	82%		-0.17%	-	
AT	75%	70%	91%		15%	-	

40%, 82%, and −0.17% when compared with multipliers [5, 6, 8, 9], respectively. It is observed that the multiplier [9] require less total delay (-0.17%) and more hardware (16%) compared to the proposed multiplier. However, the proposed multiplier achieves best AT product compared to the multipliers considered for comparison. It is evident from <Table 1> (% reduction in AT complexity row) that the proposed multiplier achieves AT efficiency of 75%, 70%, 91%, and 15% when compared with multipliers [5, 6, 8, 9], respectively. Hence, it is clear from the estimation values presented in <Table 1> that the proposed multiplier is area and AT product efficient.

The proposed multiplier uses a two-dimensional systolic array structure and requires $O(m^2)$ space complexity and $O(m)$ time complexity. The proposed multiplier has the high-throughput, with one multiplication per each clock cycle if many data independent multiplications are applied in order and concurrently executed in the systolic multiplier. The high-throughput multiplier is urgently needed because the elliptic curve digital signature algorithm (ECDSA) requires many multiplications and inversions, where the inversion can be computed by iterative multiplications. A comparison of results shows that our systolic multiplier is fully pipelined to multiply operands with an extremely high-throughput rate. The proposed high-throughput low AT-complexity systolc multiplier is thus suitable for computing digital signatures of ECDSA, which requires mass multiplications.

Ⅳ. Conclusion

In this paper, we presented efficient implementation of a polynomial-based systolic multiplier to perform multiplication of two arbitrary $GF(2^m)$ field elements. The proposed multiplier exploits the characteristics of the MMM, LSB-first scheme, and two-level parallel computation for proper mapping into a low complexity systolic structure. Detailed complexity analysis and comparison have also been presented to show the higher performance of the proposed multiplier over the existing ones. Besides, the proposed multiplier has the features of regularity, simplicity, modularity, concurrency, and unidirectional data flow, and can be used as a kernel circuit for exponentiation/division and multiplication. Future research focuses on the application of the proposed systolic multiplier to obtain efficient cryptosystem in various resource- constrained environment.

참고문헌

[1] Diffie, W. and Hellman, M. E., “New directions in cryptography,” IEEE Transaction Information Theory, Vol.22, No.6, 1976, pp.644–654.

[2] Kobliz, N., “Elliptic curve cryptography,” Mathematics of Computation, Vol.48, No.177, 1987, pp.203–209.

[3] Lee, C. Y., Chiou, C. W. and Lin, J. M., “Concurrent error detection in a polynomial basis multiplier over $GF(2^m)$,” Journal of Electronic Testing, Vol.22, No.2, 2006, pp.143–150.

[4] Chiou, C. W., Lee, C. Y., Deng, A. W. and Lin, J. M., "Concurrent error detection in Montgomery multiplication over $GF(2^m)$," IEICE Transactions on Fundamentals of Electronics, Vol.E89-A, No.2, 2006, pp.566-574.

[5] Huang, W. T., Chang, C. H., Chiou, C. W. and Chou, F. H., "Concurrent error detection and correction in a polynomial basis multiplier over $GF(2^m)$," IET Information Security, Vol.4, No.3, 2010, pp.111-124.

[6] Kim, K. W. and Kim, S. H., "A low latency semi-systolic multiplier over $GF(2^m)$," IEICE Electronics Express, Vol.10, No.13, 2013, p.20130354.

[7] Choi, S. H. and Lee, K. J., "Efficient systolic modular multiplier/squarer for fast exponentiation over $GF(2^m)$," IEICE Electronics Express, Vol.12, No.11, 2015, p.20150222.

[8] Chiou, C. W., Lee, C. M., Sun, Y. S., Lee, C. Y. and Lin, J. M., "High-throughput Dickson basis multiplier with a trinomial for lightweight cryptosystems," IET Computers & Digital Techniques, Vol.12, No.5, 2018, pp.187-191.

[9] Lee, H. H. and Kim, K. W., "Efficient semi-systolic finite field multiplier using redundant basis," International Scholarly and Scientific Research & Innovation, Vol.10, No.10, 2016, pp.1563-1567.

[10] Mathe, S. E. and Boppana, L., "Design and implementation of a sequential polynomial basis multiplier over $GF(2^m)$," KSII Transactions on Internet and Information Systems, Vol.11, No.4, 2017, pp.2680-2700.

[11] Ibrahim, A., "Efficient parallel and serial systolic structures for multiplication and squaring over $GF(2^m)$," Canadian Journal of Electrical and Computer Engineering, Vol.42, No.2, 2019, pp.114-120.

[12] Montgomery, P. L., "Modular multiplication without trial division," Mathematics of Computation, Vol.44, No.170, 1985, pp.519-521.

[13] Koc, C. K and Acar, T., "Montgomery multiplication in $GF(2^k)$," Designs Codes and Cryptography, Vol.14, No.1, 1998, pp.57-69.

[14] Lee, W. H., Lee, K. J. and Yoo, K. Y., "New digit-serial systolic arrays for power-sum and division operation in $GF(2^m)$," Lecture Notes in Computer Science, Vol.3045, 2004, pp.638-647.

■ 저자소개 ■

이 건 직
Lee, Keon Jik

2020년 현재 대구대학교 자유전공학부 조교수
2001년 8월 경북대학교 컴퓨터공학과(공학박사)

관심분야 : Computer Arithmetic Algorithm, Parallel Processing, Information Security
E-mail : othiin@naver.com

논문접수일 : 2020년 3월 2일
수 정 일 : 2020년 4월 8일
게재확정일 : 2020년 4월 10일