디지털산업정보학회 논문지 제18권 제1호-2022년 3월
http://dx.doi.org/10.17662/ksdim.2022.18.1.001

Low Complexity Systolic Montgomery Multiplication over Finite Fields $GF(2^m)$

Lee Keonjik*

유한체상의 낮은 복잡도를 갖는 시스톨릭 몽고메리 곱셈

이 건 직

〈Abstract〉

Galois field arithmetic is important in error correcting codes and public-key cryptography schemes. Hardware realization of these schemes requires an efficient implementation of Galois field arithmetic operations. Multiplication is the main finite field operation and designing efficient multiplier can clearly affect the performance of compute-intensive applications. Diverse algorithms and hardware architectures are presented in the literature for hardware realization of Galois field multiplication to acquire a reduction in time and area. This paper presents a low complexity semi-systolic multiplier to facilitate parallel processing by partitioning Montgomery modular multiplication (MMM) into two independent and identical units and two-level systolic computation scheme. Analytical results indicate that the proposed multiplier achieves lower area-time (AT) complexity compared to related multipliers. Moreover, the proposed method has regularity, concurrency, and modularity, and thus is well suited for VLSI implementation. It can be applied as a core circuit for multiplication and division/exponentiation.

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

Ⅰ. Introduction

Many important applications, such as digital signal processing, cryptography, and coding theory, are based on finite field $GF(2^m)$ arithmetic. In particular, elliptic curve cryptography (ECC) which needs smaller key sizes than RSA public key cryptography by offering same security level require finite field operations [1, 2]. In the finite field, the performance of a cryptosystem is primarily determined by an efficient implementation of the arithmetic operations, e.g., addition, multiplication, division and exponentiation. Addition is relatively inexpensive, whereas division and exponentiation can be carried out using repeated

* 대구대학교 자유전공학부 교수

multiply and square algorithm. Therefore, efficient multiplication architectures over $GF(2^m)$ should be designed to implement elliptic curve cryptosystem.

The efficiency of finite field multiplications depends on the selected basis representations for elements in $GF(2^m)$. There are several basis representations such as polynomial basis (PB), normal basis, dual basis, and redundant basis. Each basis has its own distinct advantages. Among these basis, polynomial basis arithmetic is more simple, regular, and scalable in hardware realization. Furthermore, the efficiency of finite field multiplier is dependent on the irreducible polynomial chosen. Irreducible polynomials are categorized into generic polynomial, equally spaced polynomial (ESP), trinomial, and pentanomial. A 1-ESP is known as an all one polynomial (AOP). Trinomial and pentanomial-based multipliers are more effective, while generic polynomial-based multipliers are suitable for a broader range of applications. According to the implementation method, varied multipliers can be devised. Bit serial multipliers require a small area, but are slow and take m clock cycles to carry out the multiplication of two elements. Conversely, bit parallel multipliers get the result in one clock cycle at the expense of immoderate hardware. A systolic array architecture has the characteristic features of regularity, modularity, local homogeneous interconnection and concurrency. Due to the nature of pipelining, it is possible to use a high clock frequency under high resource utilization.

In this paper, we propose a semi-systolic modular multiplier array over $GF(2^m)$ to reduce the time and area overhead by partitioning MMM into two independent and identical parts. These two can be processed efficiently in a pipelined manner on a semi-systolic array. The proposed systolic array architecture is based on the general irreducible polynomial and can be adopted to any more efficient types of special polynomials, i.e., trinomial and pentanomial. We develop a parallel-in-parallel-out systolic array by deriving the dependence graph (DG) of the proposed MMM algorithm and applying the cut-set systolization to the DG array. This structure is very regular and has local interconnections, making it very suitable for VLSI implementations. The proposed scheme can be used as a kernel component for both inversion/exponentiation and multiplication.

II. Montgomery multiplication for finite field

Several systolic modular multiplication algorithms and architectures have been proposed [3-12]. Lee et al. [3] and Chiou et al. [4] presented a semi-systolic array multiplier with an error detection. Huang et al. [5] proposed an efficient semi-systolic array multiplier to decrease the time and area costs. Choi and Lee [6] proposed new bit-parallel and bit-serial systolic multipliers simplifying quotient determination necessary for an exact division when performing MMM in a prime field $GF(p)$. The critical operation of the integer MMM is a three-operand addition within an iteration loop, where addition of long integers can cause a significant delay due to carry propagation, which limits the clock frequency. One way of solving this problem is to employ a systolic array. Note that

$GF(p)$ arithmetic is different from the binary field $GF(2^m)$ arithmetic because no carry computation exists in $GF(2^m)$. Choi and Lee [7] developed highly area-time efficient serial and parallel systolic array for unified multiplication and squaring with little hardware overhead. It has the feature that it computes both multiplication and squaring concurrently for fast modular exponentiation. The performance of this architecture is based on the new LSB-first multiplication and LSB-first exponentiation. Chiou et al. [8] presented a semi-systolic array multiplier to lessen the time complexity. Recently, Lee [9] proposed a new resource and delay efficient semi-systolic MMM multiplier with two-level systolic computation and LSB-first Montgomery multiplication. The multiplier presented by Mathe and Boppana [10] has both serial input and parallel input. Ibrahim [11] presented the systolic array structure for multiplication and squaring. Recently, Pillutla and Boppana [12] proposed a polynomial basis $GF(2^m)$ systolic multiplier applicable for a narrow class of trinomials which includes both the recommended trinomials for m = 233 and 409 fields. Although several multipliers have been developed with a polynomial basis of $GF(2^m)$, their high hardware complexities and long delay times are important limitations in security applications. Thus, further research on efficient multiplication architectures with low area and time complexities is needed.

The development of fast algorithms and structures of modular multiplications has received considerable interest. The Montgomery modular multiplication algorithm without a division operation was first proposed by P. L. Montgomery to improve the performance of modular integer multiplications [6, 13]. It was proved that MMM is appropriate to $GF(2^m)$ [14]. The basic idea is to convert input values to Montgomery residues, and compute the modular multiplication using these residues. Finally, the output is transformed back to the original representation.

Let $G = \sum_{j=0}^{m} g_j x^j$ be the irreducible polynomial generating the finite field $GF(2^m)$, where $g_m = g_0 = 1$ and $g_j \in GF(2)$ for $1 \le j \le m - 1$. It is conventional to represent the elements of $GF(2^m)$ as a power of the primitive element z where z is the root of G. The set $\{1, z, ..., z^{m-1}\}$ is referred to as the polynomial basis. Each element in $GF(2^m)$ is a unique linear combination consisting of polynomials of degree less than m in $GF(2)$. The addition is bitwise exclusive-OR (XOR); but, the multiplication is a little complicated since the intermediate result needs further modular reduction by $x^m = \sum_{j=0}^{m-1} g_j x^j$.

Let x and y be two elements of $GF(2^m)$ to be multiplied. Instead of computing $S = xy \bmod G$, using a novel notation of the residue class, Montgomery multiplication of A (= $xR \bmod G = \sum_{j=0}^{m-1} a_j x^j$) and B (= $yR \bmod G = \sum_{j=0}^{m-1} b_j x^j$) is computed by $T = ABR^{-1} \bmod G = \sum_{j=0}^{m-1} t_j x^j$, where A (resp., B) is the Montgomery residue of x (resp., y) and R is a special element satisfying $\gcd(R, G) = 1$. The final result S is then taken by computing MMM using inputs T and 1, i.e., $S = TR^{-1} \bmod G = xy \bmod G$. Therefore, MMM is favourable in several applications involving successive multiplications, such as inversion, exponentiation, and elliptic curve point multiplication due to the pre- and post-transformation needs.

Note that in most practical applications, m is an odd number. For elliptic curve cryptography (ECC),

the National Institute of Standard and Technology (NIST) recommends five binary fields: $GF(2^{163})$, $GF(2^{233})$, $GF(2^{283})$, $GF(2^{409})$, and $GF(2^{571})$. Of these five fields, m = 233 and 409 are irreducible trinomials and the others are pentanomials. In this paper, we only consider odd values of m and we construct a low complexity semi-systolic Montgomery multiplier. Using $R = x^{(m-1)/2}$, the Montgomery multiplication $T = ABR^{-1} \bmod G$ can be formulated as

$$\begin{aligned} T &= A(b_0 + b_1x + \cdots + b_{(m-1)/2}x^{(m-1)/2} + \cdots \\ &\quad + b_{m-1}x^{m-1})x^{-(m-1)/2} \bmod G \\ &= A(b_0x^{-(m-1)/2} + b_1x^{-(m-3)/2} + \cdots + b_{(m-3)/2}x^{-1}) \\ &\quad + A(b_{(m-1)/2}x^0 + b_{(m+1)/2}x^1 + \cdots + b_{m-1}x^{(m-1)/2}) \bmod G \end{aligned} \quad (1)$$

To derive the recurrence relations for the proposed semi-systolic MMM multiplier, (1) is represented as the sum of two polynomials C and D as follows:

$$\begin{aligned} C &= Ab_{m-1}x^{(m-1)/2} + Ab_{m-2}x^{(m-3)/2} + \cdots \\ &\quad + Ab_{(m+1)/2}x^1 + Ab_{(m-1)/2} \bmod G \\ &= (\cdots((Ab_{m-1})x \bmod G + Ab_{m-2})x \bmod G + \cdots \\ &\quad + Ab_{(m+1)/2})x \bmod G + Ab_{(m-1)/2} \end{aligned} \quad (2)$$

$$\begin{aligned} D &= Ab_{(m-3)/2}x^{-1} + Ab_{(m-5)/2}x^{-2} + \cdots \\ &\quad + Ab_1x^{-(m-3)/2} + Ab_0x^{-(m-1)/2} \bmod G \\ &= (\cdots((Ab_0)x^{-1} \bmod G + Ab_1)x^{-1} \bmod G + \cdots \\ &\quad + Ab_{(m-3)/2})x^{-1} \bmod G \end{aligned} \quad (3)$$

Let C_i and D_i be the results of i-th recursion of (2) and (3), respectively, which can be computed recursively from the results of $(i-1)$-th pair of recursions. The recursive equation of (2) at step i for $1 \le i \le (m+1)/2$ can be obtained as

$$C_i = C_{i-1}x \bmod G + Ab_{m-i}, \quad (4)$$

where $C_0 = 0$.

Similar to (4), we can express (3) recursively as

$$D_i = D_{i-1}x^{-1} \bmod G + Ab_{i-1}, \quad (5)$$

where $D_0 = b_{(m-1)/2} = 0$.

Note that the value $b_{(m-1)/2} = 0$ is required in the computation of the last $D_{(m+1)/2}$. According to (4) and (5), it is clear that there is no data dependency in C_i and D_i, so they can be computed concurrently. With the bit-level representation, substituting the expansion of x^m on (4), the reduced form of C_i for $1 \le i \le (m+1)/2$ can be obtained by

$$\begin{aligned} C_i &= c_{i-1,m-2}x^{m-1} + \ldots + c_{i-1,1}x^2 + c_{i-1,0}x \\ &\quad + c_{i-1,m-1}(g_{m-1}x^{m-1} + \ldots + g_1x + g_0) \\ &\quad + b_{m-i}(a_{m-1}x^{m-1} + \ldots + a_1x + a_0) \\ &= c_{i,m-1}x^{m-1} + \ldots + c_{i,1}x + c_{i,0} \end{aligned} \quad (6)$$

Finally, C at step i can be represented in a recursive manner as

$$c_{i,m-1-j} = c_{i-1,m-2-j} + c_{i-1,m-1}g_{m-1-j} + b_{m-i}a_{m-1-j}, \quad (7)$$

where $c_{0,j} = c_{i-1,-1} = 0$ and $m - 1 \ge j \ge 0$.

For any irreducible polynomial, $g_0 = 1$ and $g_m = 1$, and we notice that x is a root of G. Thus, multiplying each side of G by x^{-1}, and reorganizing the terms, we obtain $x^{-1} = \sum_{j=1}^{m} g_j x^{j-1}$. Similar to (6), substituting the expansion of x^{-1} on (5), D_i can be rewritten as follows:

$$\begin{aligned} D_i &= d_{i-1,m-1}x^{m-2} + \ldots + d_{i-1,1} \\ &\quad + d_{i-1,0}(g_m x^{m-1} + \ldots + g_2 x + g_1) \\ &\quad + b_{i-1}(a_{m-1}x^{m-1} + \ldots + a_1 x + a_0) \\ &= d_{i,m-1}x^{m-1} + \ldots + d_{i,1}x + d_{i,0}. \end{aligned} \quad (8)$$

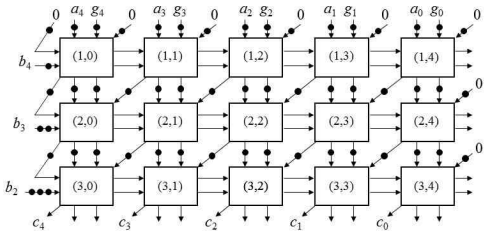
The recursive equation of (8) at step i for $1 \le i \le (m+1)/2$ can be represented as

$$d_{i,j} = d_{i-1,j+1} + d_{i-1,0}g_{j+1} + b_{i-1}a_j, \quad (9)$$

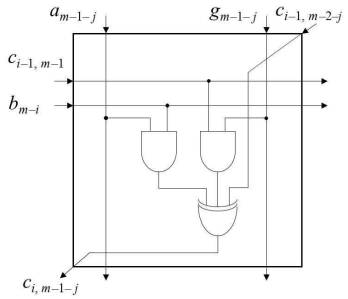
where $d_{0,j} = d_{i-1,m} = b_{(m-1)/2} = 0$ and $0 \le j \le m-1$. Finally, $C^{(m+1)/2}$ and $D^{(m+1)/2}$ should be summed up by using m 2-input XOR gates (XOR_2) to obtain T.

For simplicity of discussion, the binary field $GF(2^5)$ is used to illustrate the semi-systolic architecture. A most significant bit (MSB)-first semi-systolic array for the computation of C based on (7) is shown in <Figure 1>, where $m \times (m+1)/2$ basic cells of <Figure 2> are used and "●" denotes a 1-bit latch. In <Figure 2>, the cell at position (i, j) computes (7). In Fig. 1(a), the cell at position (i, j) receives $c_{i-1,m-2-j}$ from the neighbor cell at the position $(i-1, j+1)$ of the previous row and computes $c_{i,m-1-j}$.

The index points and initial locations of all inputs are as following. First, b_{m-i}, $1 \le i \le (m+1)/2$, enters index $(i, 0)$ from the left direction and flows into the direction of [0, 1]. The values a_{m-1-j} and g_{m-1-j}, $m-1 \ge j \ge 0$, then enter index [1, j] from the top and flow in the direction of [1, 0], respectively. Next, $c_{0,m-2-j}$, $0 \le j \le m-1$, enters [1, j] index from the top, and then is computed with the partial products generated by the previous row to give new partial products that are passed on to the next row, and then flows in the direction of [1, −1].



<Figure 1> Array architecture for C



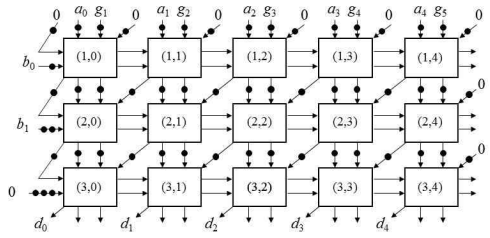
<Figure 2> Circuit of (i, j) cell

The values a_{m-1-j} and g_{m-1-j}, $m-1 \ge j \ge 0$, then enter index [1, j] from the top and flow in the direction of [1, 0], respectively. Next, $c_{0,m-2-j}$, $0 \le j \le m-1$, enters [1, j] index from the top, and is computed with the partial products generated by the previous row to give new partial products that are passed on to the next row, and then flows in the direction of [1, −1]. Then, $c_{i-1,m-1}$, $1 \le i \le (m+1)/2$, the MSB of C_{i-1}, enters index [i, 0] from the left direction and flows into the direction of [0, 1]. The result, $C_{(m+1)/2}$, is obtained from the bottom row of the array after $(m+1)/2$ iterations. In <Figure 2>, the basic cell consists of two 2-input AND gates (AND_2) and one 3-input XOR gate (XOR_3). The (i, j) cell receives $c_{i-1,m-2-j}$ as its input from the $(i-1, j+1)$-th cell; a_{m-1-j} and g_{m-1-j}, from the $(i-1, j)$-th cell; and b_{m-i}, from the $(i, j-1)$-th cell. The critical path delay of this structure is the total delay of one

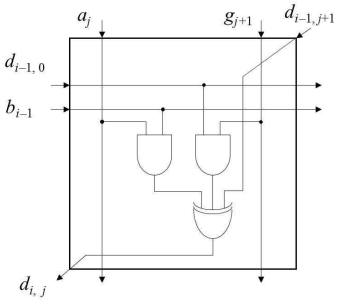
AND_2 and one XOR_3. In <Figure 1>, the left side input b_{m-i+1} is delayed by one clock cycle relative to b_{m-i} for $2 \le i \le (m+1)/2$.

Similar to <Figure 1>, the least significant bit (LSB)-first semi-systolic array can be adopted efficiently to compute D by rearranging coefficients of B, A, and G based on (9), as depicted in <Figure 3>, where $m \times (m+1)/2$ basic cells of <Figure 4> are used. Note that $(m+1)/2$-th row is added for the final x^{-1} operation.

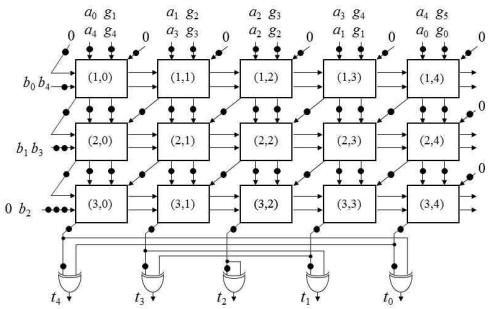
The proposed multiplier can be implemented by using the above two arrays and m XOR_2. It is noted that the structures and data flows of <Figure 1> are the same as those of <Figure 3>. Since (7) and (9) have an identical and independent computation structure, we can process two equations in pipelined fashion using one systolic array. Therefore, by unifying and retiming <Figure 1> and <Figure 3>, the new semi-systolic multiplier architecture in <Figure 5> and <Figure 6> can be derived. In <Figure 5>, each cell (i, j) includes two AND_2 and one XOR_3, and computes C of (7) and D of (9) in sequence. After $(m+1)/2$ iterations, the final result T is the summation of $C_{(m+1)/2}$ and $D_{(m+1)/2}$, where $C_{(m+1)/2}$ is delayed by one clock cycle relative to $D_{(m+1)/2}$. The latency of the presented multiplier is $(m+1)/2 + 3$ clock cycles; the critical propagation delay time is the total delay of one AND_2 and one XOR_3. Note that for the further parallel computation and critical propagation delay reduction, a 3-input XOR gate in <Figure 6> can be constructed using two 2-input XOR gates. The architecture of <Figure 6> can be reorganized similarly to the cell structure of the systolic array in [9]. As a result, the critical path delay can be reduced to the total delay of one AND_2 and one XOR_2 with a little area overhead.



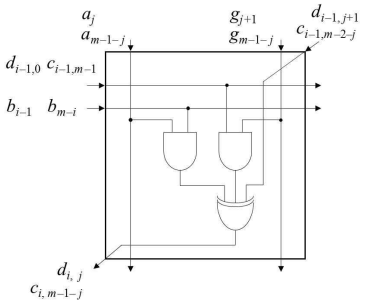
<Figure 3> Array architecture for D



<Figure 4> Circuit of (i, j) cell



<Figure 5> Proposed multiplier for $T = C + D$



<Figure 6> Circuit of (i, j) cell

Ⅲ. Analysis and comparison

The analytical expressions shown in <Table 1> are analyzed for m = 571 using the time and area complexity approximations of logic gates built from the Samsung electronics, where m = 571 is one of the finite field sizes preferred by NIST for elliptic curve applications.

We obtained the area of the gates and latch along with their worst-case intrinsic delays pertaining to unit drive-strength from the Samsung STD 150 0.13μm 1.2V CMOS Standard Cell Library databook. Using these data, we estimate the time and area complexities of the proposed and related works. The following is the area and time requirements of the cells used in <Table 1>, where T represents the time (ns), A represents the area (transistor count), and Gate$n$ represents the n-input logic gate, respectively [7]: T_{AND2}: 0.094, A_{AND2}: 6.68, T_{XOR2}: 0.167, A_{XOR2}: 12.00, T_{LATCH}: 0.157, and A_{LATCH}: 16.00.

<Table 1> shows the analytical comparison of data flow, latency, throughput, time complexity, area complexity, AT complexity, and improvement of the presented multiplier with the multipliers considered for comparison. The area requirement for the presented multiplier is analyzed in terms of number of XOR_2, AND_2, latches, and transistors.

It is noticed that the presented multiplier needs a comparable number of transistors. It is evident from <Table 1> (% reduction in #transistors row) that the presented multiplier gets area efficiency of 38%, 42%, and -18% when compared with multipliers [5, 8, 9], respectively. The critical path delay (i.e., cell delay) and latency of the presented multiplier are 0.26 ns and $(m+7)/2$ clock cycles,

<Table 1> Complexity comparison of semi-systolic multipliers

Multipliers	Huang et al.[5]	Chiou et al.[8]		Lee [9]		Proposed
# cells	m^2	U:$m(m-2)$, V:$2m$		M:$m(m+1)/2$, X:m		$m(m+1)/2$
Latency	$m+1$	$m+1$		$(m+7)/2$		$(m+7)/2$
Data flow	bidirectional	unidirectional		unidirectional		unidirectional
Throughput	1	1		1		1
Area complexity		U	V	M	X	
#AND_2	$2m^2$	m^2-2m	$2m$	m^2+m	0	m^2+m
#XOR_2	$2m^2$	$3(m^2-2m)$	$8m$	m^2+m	m	$m^2+(7m+1)/2$
#Latch	$3m^2+m+1$	$3(m^2-2m)$	$6m$	$1.6m^2+2m$	$2m$	$2.1m^2+6.5m$
#transistors	$85.4m^2+16(m+1)$	$90.7m^2+24m$		$44.28m^2+94.68m$		$52.28m^2+152.7m+6$
Time complexity						
Cell delay	0.26	0.75		0.26		0.26
Total delay	$0.26m+0.26$	$0.75m+0.75$		$0.13m+0.91$		$0.13m+0.91$
AT complexity	$22.2m^3+26.4m^2+8.3m+4.2$	$68.0m^3+86.0m^2+18m$		$5.8m^3+52.6m^2+86.2m$		$6.8m^3+67.4m^2+139.7m$
Improvement						
Area	38%	42%		-18%		-
Time	49%	82%		0%		-
AT	69%	90%		-17%		-

respectively and the total delay time is (latency) × (cell delay). <Table 1> (% reduction in total delay row) indicates that the presented architecture gets time efficiency of 49%, 82%, and 0% when compared with multipliers [5, 8, 9], respectively. It is noticed that the multiplier [9] needs the same total delay (0%) and less hardware (−18%) compared to the proposed multiplier. Although the proposed multiplier needs a little high-chip area, it is realizable and affordable by VLSI implementation.

The presented systolic multiplier takes the high throughput with one multiplication result per clock cycle if data independent multiplications are applied in order and computed in parallel in the proposed multiplier. The high-throughput systolic multiplier is desperately required because the elliptic curve digital signature algorithm (ECDSA) needs a large number of field multiplications and inversions, where the inversion operation can be done by repetitive multiplications. A comparison of results notices that the proposed systolic multiplier is pipelined to multiply operands with high throughput rate. Thus, the proposed high-throughput low AT-complexity systolic multiplier is suitable for executing digital signatures of ECDSA, which needs many multiplications.

Ⅳ. Conclusion

This study proposes a new low-complexity semi-systolic multiplier for efficient MMM, which is the crucial operation in finite field arithmetic. Note that the proposed multiplier consists of the MSB-first PB array (Figure 1) and the LSB-first MMM array (Figure 3). Two equally divided arrays can be independently performed, and it is well suited to implement both arrays in the same hardware. In this way, we can decrease the hardware and time requirements. The effectiveness of our study is demonstrated by the reduced AT complexity as compared to related works. Due to its low-complexity, the proposed array can be particularly useful for implementing applications in resource constrained environments. The regularity, simplicity, modularity, and concurrency of our proposed architecture allow for easy extension and thus are well suitable for VLSI implementation.

References

[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] Choi, S. H. and Lee, K. J., "New systolic modular multiplication architecture for efficient Montgomery multiplication," IEICE Electronics Express, Vol.12, No.2, 2015, p.20141051.

[7] Choi, S. H. and Lee, K. J., "Efficient systolic modular multiplier/squarer for fast exponent-iation 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, K. J., "Resource and delay efficient polynomial multiplier over finite field $GF(2^m)$," Korea Society of Digital Industry and Information Management, Vol.16, No.2, 2020, pp.1-9.

[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] Pillutla, S. R. and Boppana, L., "Low-latency area-efficient systolic bit-parallel $GF(2^m)$ multiplier for a narrow class of trinomials," Microelectronics Journal, Vol.117, 2021, p.105275.

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

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

■ 저자소개 ■

이 건 직
Lee, Keon Jik

2022년 현재 대구대학교 자유전공학부 교수
2001년 8월 경북대학교 컴퓨터공학과(공학박사)

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

논문접수일 : 2021년 12월 20일
수 정 일 : 2022년 1월 5일
게재확정일 : 2022년 1월 11일