JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 28 No. 2, pp. 69-75, February 2023
https://doi.org/10.9708/jksci.2023.28.02.069

Efficient Semi-systolic Montgomery multiplier over GF(2^m)

Keewon Kim*

*Professor, Dept. of Computer Engineering, Mokpo National Maritime University, Mokpo, Korea

[Abstract]

Finite field arithmetic operations play an important role in a variety of applications, including modern cryptography and error correction codes. In this paper, we propose an efficient multiplication algorithm over finite fields using the Montgomery multiplication algorithm. Existing multipliers can be implemented using AND and XOR gates, but in order to reduce time and space complexity, we propose an algorithm using NAND and NOR gates. Also, based on the proposed algorithm, an efficient semi-systolic finite field multiplier with low space and low latency is proposed. The proposed multiplier has a lower area-time complexity than the existing multipliers. Compared to existing structures, the proposed multiplier over finite fields reduces space-time complexity by about 71%, 66%, and 33% compared to the multipliers of Chiou et al., Huang et al., and Kim-Jeon. As a result, our multiplier is proper for VLSI and can be successfully implemented as an essential module for various applications.

▸ **Key words**: Finite fields, Montgomery multiplication, Semi-systolic array, Cryptography

[요 약]

유한체 산술 연산은 현대 암호학(cryptography)과 오류 정정 부호(error correction codes) 등 다양한 응용에서 중요한 역할을 한다. 본 논문에서는 유한체상에서 몽고메리 곱셈 알고리즘을 사용한 효율적인 유한체 곱셈 알고리즘을 제안한다. 기존의 곱셈기들에서는 AND와 XOR 게이트를 사용하여 구현되었는데, 시간 및 공간 복잡도를 줄이기 위해서 NAND와 NOR 게이트를 사용하는 알고리즘을 제안하였다. 게다가 제안한 알고리즘을 기초로 적은 공간과 낮은 지연시간을 갖는 효율적인 세미-시스톨릭(semi-systolic) 유한체 곱셈기를 제안한다. 제안한 곱셈기는 기존의 곱셈기에 비해 낮은 공간-시간 복잡도(area-time complexity)를 가진다. 기존의 구조들과 비교하면, 제안한 유한체 곱셈기는 공간-시간 복잡도면에서 Chiou 등, Huang 등 및 Kim-Jeon의 곱셈기에 비해 약 71%, 66%, 33%가 감소되었다. 따라서 제안한 곱셈기는 VLSI 구현에 적합하며, 다양한 응용의 기본 구성 요소로 쉽게 적용될 수 있다.

▸ **주제어**: 유한체, 몽고메리 곱셈, 세미-시스톨렉 어레이, 암호학

• First Author: Keewon Kim, Corresponding Author: Keewon Kim
*Keewon Kim (kwkim@mmu.ac.kr), Dept. of Computer Engineering, Mokpo National Maritime University
• Received: 2023. 01. 10, Revised: 2023. 01. 30, Accepted: 2023. 01. 31.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

유한체 산술 연산은 현대 암호학과 오류 정정 부호(error correction codes) 등 다양한 응용에서 중요한 역할을 한다 [1]-[4]. 특히 유한체 상의 연산 중에서 곱셈은 가장 중요한 연산으로 나눗셈, 곱셈의 연원, 지수 등 시간 소모가 많은 연산들은 반복적인 곱셈을 수행하여 계산될 수 있다. 따라서 낮은 공간-시간 복잡도를 가지는 효율적인 고속의 곱셈기의 설계가 필요하다.

유한체 $GF(2^m)$상의 곱셈 관련 연산의 고속 구현을 위해 다양한 시스톨릭(systolic) 구조들이 제안되었다 [5]-[13]. Wang 등 [5]은 $GF(2^m)$상의 시스톨릭 곱셈기를 제안하였으며 Jain 등 [6]은 세미-시스톨릭(semi-systolic) 구조를 이용하여 Wang 등 [5]의 곱셈기보다 성능이 향상된 곱셈기를 제안하였다. 이후 세미-시스톨릭 곱셈기에 오류 검출 기능을 가진 구조들이 제안되었다 [7]-[9]. Chiou 등 [7]과 Lee 등 [8]은 유한체 $GF(2^m)$상에서 오류 검출 기능을 가진 다항식 기저의 세미-시스톨릭 곱셈기를 제안하였다. Lee 등 [8]은 몽고메리 곱셈(Montgomery multiplication) 알고리즘을 이용하였다. 몽고메리 곱셈 알고리즘은 빠른 모듈러 정수 곱셈을 위해 개발되었다 [14]. 이러한 몽고메리 곱셈은 Koc와 Acar에 의해 유한체 $GF(2^m)$로 확장하여 적용되었다 [15]. Huang 등 [9]은 기존의 곱셈기들보다 성능이 향상된 다항식 기저의 세미-시스톨릭 곱셈기를 제안하였다. 이들의 구조는 오류 검출과 교정 기능이 포함되어 있다. Kim과 Jeon [10]은 Huang 등 [9]의 곱셈기보다 낮은 지연시간을 가지는 성능이 향상된 곱셈기를 제안하였다. 기존에 제안된 곱세기들은 공간 및 시간 면에서 개선이 필요하며, 효율적인 세미-시스톨릭 곱셈기에 관한 연구가 필요하다.

본 논문은 몽고메리 곱셈 알고리즘을 이용하여 다항식 기저의 효율적인 세미-시스톨릭 곱셈기를 제안한다. 제안한 곱셈기는 기존에 Kim과 Jeon [10]의 곱셈기와 비교하면 공간-시간 복잡도면에서 약 33% 감소되었다.

본 논문의 구성은 다음과 같다. 제 2장에서는 $GF(2^m)$상의 몽고메리 곱셈 알고리즘을 소개한다. 제 3장에서는 효율적인 몽고메리 곱셈을 이용한 유한체 곱셈 알고리즘을 제안하였다. 그리고 이 알고리즘을 이용한 세미-시스톨릭 곱셈기를 제안한다. 제 4장에서는 제안한 곱셈기의 공간 및 시간 복잡도를 분석하고 기존의 곱셈기들과 비교한다. 마지막으로 제 5장에서 결론을 맺는다.

II. Montgomery multiplication on GF(2^m)

이 장에서는 $GF(2^m)$상에서의 몽고메리 곱셈에 대해 간략하게 소개한다. 먼저 $GF(2^m)$은 2^m개의 서로 다른 원소를 포함하는 유한체다. 이러한 유한체 $GF(2^m)$의 원소는 $m-1$ 차수 이하의 다항식으로 표현할 수 있으며, 다항식의 각 계수는 $GF(2)$ 유한체의 원소인 0 또는 1의 값을 가진다. 예를 들면, $GF(2^m)$의 어떤 원소 A는 $A=\sum_{i=0}^{m-1} a_i x^i$와 같은 다항식으로 나타낼 수 있다. 여기서 $a_i \in \{0,1\}$, $0 \le i \le m-1$이다. 모든 유한체는 적어도 하나 이상의 기약 다항식과 연관되어 있다. $GF(2^m)$에 대한 일반적인 기약다항식은 $G(x)=\sum_{i=0}^{m} g_i x^i$로 나타낼 수 있다. 여기서 $g_i \in GF(2)$, $0 \le i \le m-1$이다.

정수 상의 모듈러 곱셈을 효율적으로 수행하기 위해 몽고메리 알고리즘이 Montgomery에 의해 제안되었고[14], 몽고메리 알고리즘은 Koc과 Acar에 의해 유한체의 곱셈으로 확장되었다 [15]. $GF(2^m)$상에서의 몽고메리 곱셈을 살펴보면 다음과 같다. α와 β는 $GF(2^m)$상의 두 원소이고 $\delta = \alpha \cdot \beta \bmod G$라고 하자. 여기서 G는 $G(x)$를 나타낸다. 몽고메리 잉여(Montgomery residue) A와 B는 다음과 같이 정의된다.

$$A = \alpha \cdot R \bmod G = \sum_{i=0}^{m-1} a_i x^i \quad (1)$$

$$B = \beta \cdot R \bmod G = \sum_{i=0}^{m-1} b_i x^i \quad (2)$$

여기서 몽고메리 인자(Montgomery factor) R과 기약 다항식 G는 서로소이며, $\gcd(R, G) = 1$을 만족한다. $GF(2^m)$상의 몽고메리 곱셈은 $S = A \cdot B \cdot R^{-1} \bmod G$로 나타낼 수 있다. 여기서 R^{-1}을 R의 모듈러 G에 대한 곱셈의 역원(multiplicative inverse)이라고 하며, $R \cdot R^{-1} + G \cdot G' = 1$을 만족하는 $GF(2^m)$상의 원소 G'이 존재한다.

몽고메리 잉여를 사용하면 몽고메리 곱셈은 $S = (\alpha \cdot R) \cdot (\beta \cdot R) \cdot R^{-1} \bmod G = \gamma \cdot R \bmod G$로 표현된다. 여기서 S는 γ의 몽고메리 잉여이다.

III. Proposed Montgomery Multiplier

1. Proposed Montgomery Multiplication Algorithm

본 논문에서는 곱셈기 구조의 병렬성을 높이기 위해서 몽고메리 인자 $R = x^{\lfloor m/2 \rfloor}$ 를 사용한다. $GF(2^m)$상의 몽고메리 곱셈은 다음과 같이 나타낼 수 있다.

$$\begin{aligned} S &= A \cdot B \cdot R^{-1} \bmod G \\ &= A \cdot B \cdot x^{-\lfloor m/2 \rfloor} \bmod G \end{aligned} \tag{3}$$

x는 $G(x)$의 근(root)이며, 즉, 모든 기약다항식에 대해 $G(x) = 0$과 $g_m = g_0 = 1$ 이다. 따라서, x^m과 x^{-1}은 다음과 같다.

$$x^m \bmod G = \sum_{i=0}^{m-1} g_i x^i \tag{4}$$

$$x^{-1} \bmod G = x^{m-1} + \sum_{j=1}^{m-1} g_j x^{j-1} \tag{5}$$

$S = A \cdot B \cdot x^{-\lfloor m/2 \rfloor} \bmod G$에서 $k = \lfloor m/2 \rfloor$ 라고 두면 S는 다음과 같이 표현된다.

$$\begin{aligned} S &= A \cdot B \cdot x^{-\lfloor m/2 \rfloor} \bmod G \\ &= \left(\sum_{j=0}^{m-1} b_j A x^j \right) x^{-k} \bmod G \\ &= \left(\sum_{j=0}^{k-1} b_j A x^j + \sum_{j=k}^{m-1} b_j A x^j \right) x^{-k} \bmod G \\ &= \sum_{j=0}^{k-1} b_j A x^{j-k} \bmod G + \sum_{j=k}^{m-1} b_j A x^{j-k} \bmod G \end{aligned} \tag{6}$$

식 (6)의 마지막의 오른쪽 항은 두 부분으로 나눌 수 있다. 하나는 x의 음의 거듭제곱을 기반으로 하고 다른 하나는 x의 양의 거듭제곱을 기반으로 한다. S는 $S = C + D$의 형태로 나타낼 수 있으며, C와 D는 다음과 같다.

$$C = \sum_{j=0}^{k-1} b_j A x^{-k+j} \bmod G \tag{7}$$

$$D = \sum_{j=k}^{m-1} b_j A x^{-k+j} \bmod G \tag{8}$$

$N^{(i)}$와 $P^{(i)}$를 $N^{(i)} = A x^{-i} \bmod G$와 $P^{(i)} = A x^i \bmod G$로 정의하면 다음 식과 같이 표현된다.

$$N^{(i)} = \sum_{j=0}^{m-1} n_j^{(i)} x^j \tag{9}$$

$$P^{(i)} = \sum_{j=0}^{m-1} p_j^{(i)} x^j \tag{10}$$

식 (4)와 (5)를 이용하여 $N^{(i)}$와 $P^{(i)}$를 다시 표현하면 아래와 같다. 여기서 $N^{(0)} = P^{(0)} = A$이다.

$$\begin{aligned} N^{(i)} &= N^{(i-1)} x^{-1} \bmod G \\ &= \sum_{j=0}^{m-1} (n_{j+1}^{(i-1)} + n_0^{(i-1)} g_{j+1}) x^j \end{aligned} \tag{11}$$

$$\begin{aligned} P^{(i)} &= P^{(i-1)} x \bmod G \\ &= \sum_{j=0}^{m-1} (p_{j-1}^{(i-1)} + p_{m-1}^{(i-1)} g_j) x^j \end{aligned} \tag{12}$$

식 (9)와 (10)에서 $N^{(i)}$와 $P^{(i)}$의 계수는 다음과 같이 표현할 수 있다.

$$n_j^{(i)} = n_{j+1}^{(i-1)} + n_0^{(i-1)} g_{j+1} \tag{13}$$

$$p_j^{(i)} = p_{j-1}^{(i-1)} + p_{m-1}^{(i-1)} g_j \tag{14}$$

여기서 $n_m^{(i-1)} = p_{-1}^{(i-1)} = 0$, $g_m = g_0 = 1$이고 $0 \le j \le m-1$이다.

또한, $N^{(i)}$와 $P^{(i)}$를 사용하여, 식 (7)과 (8)의 C와 D는 다음과 같이 표현할 수 있다. 여기서 동일한 구조를 도출하기 위해 $zN^{(0)}$항을 C에 추가하였으며 $z = 0$이다. 그리고 $l = \lceil m/2 \rceil$ 이라고 정의한다.

$$\begin{aligned} C &= \sum_{j=0}^{k-1} b_j A x^{-k+j} \bmod G \\ &= \sum_{j=0}^{k-1} b_j N^{(k-j)} + z N^{(0)} \end{aligned} \tag{15}$$

$$\begin{aligned} D &= \sum_{j=k}^{m-1} b_j A x^{-k+j} \bmod G \\ &= \sum_{j=0}^{l-1} b_{k+j} P^{(j)} \end{aligned} \tag{16}$$

위의 식에서 다음과 같이 C와 D의 재귀식을 도출할 수 있다. 여기서 $C^{(0)} = D^{(0)} = 0$이다.

$$C^{(i)} = \begin{cases} C^{(i-1)} + zN^{(i-1)}, & i = 1 \\ C^{(i-1)} + b_{k-i+1}N^{(i-1)}, & 2 \le i \le k+1 \end{cases} \quad (17)$$

$$D^{(i)} = D^{(i-1)} + b_{k+i-1}P^{(i-1)}, \ 1 \le i \le l \quad (18)$$

식 (17)과 (18)로부터 $C^{(i)}$와 $D^{(i)}$의 계수를 구하여 표현하면 다음과 같다.

$$c_j^{(i)} = \begin{cases} c_j^{(i-1)} + zn_j^{(i-1)}, & i = 1 \\ c_j^{(i-1)} + b_{k-i+1}n_j^{(i-1)}, & 2 \le i \le k+1 \end{cases} \quad (19)$$

$$d_j^{(i)} = d_j^{(i-1)} + b_{k+i-1}p_j^{(i-1)}, \ 1 \le i \le l \quad (20)$$

여기서 $c_j^{(0)} = d_j^{(0)} = 0 (0 \le j \le m-1)$이고 $z = 0$이다.

식 (13), (14), (19), (20)에서의 연산을 하드웨어로 구현할 때, $GF(2)$에서의 덧셈은 XOR 연산으로 구현되고 곱셈은 AND 연산으로 구현된다. 이를 기반으로 식 (13), (14), (19)와 (20)을 다시 표현하면 다음 식과 같다. 여기서, $\oplus$는 논리 XOR 연산을 나타내고 $\wedge$는 논리 AND 연산을 나타낸다.

$$n_j^{(i)} = n_{j+1}^{(i-1)} \oplus (n_0^{(i-1)} \wedge g_{j+1}) \quad (21)$$

$$p_j^{(i)} = p_{j-1}^{(i-1)} \oplus (p_{m-1}^{(i-1)} \wedge g_j) \quad (22)$$

$$c_j^{(i)} = \begin{cases} c_j^{(i-1)} \oplus (z \wedge n_j^{(i-1)}) \\ c_j^{(i-1)} \oplus (b_{k-i+1} \wedge n_j^{(i-1)}) \end{cases} \quad (23)$$

$$d_j^{(i)} = d_j^{(i-1)} \oplus (b_{k+i-1} \wedge p_j^{(i-1)}) \quad (24)$$

적은 공간과 지연시간의 하드웨어 구조를 도출하기 위해, 식 (21)을 다음과 같이 유도할 수 있다. 여기서 $\vee$, $\odot$, $\overline{\wedge}$는 각각 논리적 OR, XNOR, NAND 연산을 나타낸다.

$$\begin{aligned} n_j^{(i)} &= n_{j+1}^{(i-1)} \oplus (n_0^{(i-1)} \wedge g_{j+1}) \\ &= (n_{j+1}^{(i-1)} \wedge \overline{(n_0^{(i-1)} \wedge g_{j+1})}) \\ &\quad \vee (\overline{n_{j+1}^{(i-1)}} \wedge (n_0^{(i-1)} \wedge g_{j+1})) \\ &= (n_{j+1}^{(i-1)} \wedge \overline{(n_0^{(i-1)} \wedge g_{j+1})}) \\ &\quad \vee (\overline{n_{j+1}^{(i-1)}} \wedge \overline{(n_0^{(i-1)} \wedge g_{j+1})}) \\ &= n_{j+1}^{(i-1)} \odot (n_0^{(i-1)} \overline{\wedge} \, g_{j+1}) \end{aligned} \quad (25)$$

식 (25)와 같은 방법으로 식 (22), (23), (24)를 다음과 같이 유도할 수 있다.

$$\begin{aligned} p_j^{(i)} &= p_{j-1}^{(i-1)} \oplus (p_{m-1}^{(i-1)} \wedge g_j) \\ &= p_{j-1}^{(i-1)} \odot (p_{m-1}^{(i-1)} \overline{\wedge} g_j) \end{aligned} \quad (26)$$

$$\begin{aligned} c_j^{(i)} &= c_j^{(i-1)} \oplus (b_{k-i+1} \wedge n_j^{(i-1)}) \\ &= c_j^{(i-1)} \odot (b_{k-i+1} \overline{\wedge} n_j^{(i-1)}) \end{aligned} \quad (27)$$

$$\begin{aligned} d_j^{(i)} &= d_j^{(i-1)} \oplus (b_{k+i-1} \wedge p_j^{(i-1))}) \\ &= d_j^{(i-1)} \odot (b_{k+i-1} \overline{\wedge} p_j^{(i-1))}) \end{aligned} \quad (28)$$

2. Proposed Montgomery Multiplier

앞 절에서 제안한 곱셈 알고리즘을 이용하여 적은 면적과 지연시간의 몽고메리 곱셈기를 제안한다. 식 (17)과 (18)의 $C^{(i)}$와 $D^{(i)}$는 동시에 병렬로 계산이 가능하다. 이는 서로 간에 데이터 의존성이 존재하지 않기 때문이다.

본 논문의 그림에서 ■는 1-비트 지연 소자를 의미한다. Fig. 1과 2는 각각 $GF(2^4)$와 $GF(2^5)$상에서 제안하는 몽고메리 곱셈기의 하드웨어 구조들이다. $GF(2^m)$에서 만약 m이 짝수이면 곱셈기는 $0.5m^2 - m$개 $\mathrm{U}_j^{(i)}$셀, m개 $\hat{\mathrm{U}}_j^{(i)}$셀, m개의 V_j셀로 구성된다. 만약 m이 홀수이면 곱셈기는 $0.5m^2 - 0.5m$개 $\mathrm{U}_j^{(i)}$셀, m개의 $\hat{\mathrm{V}}_j$셀로 구성된다.

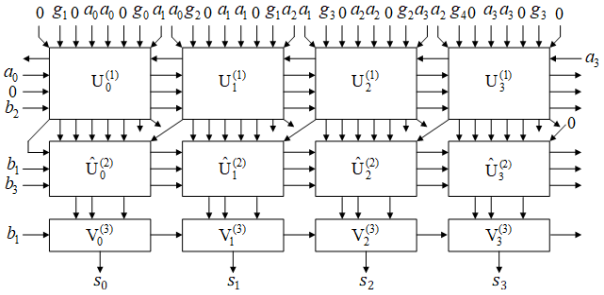


Fig. 1. Proposed Montgomery Multiplier over GF(2^4)

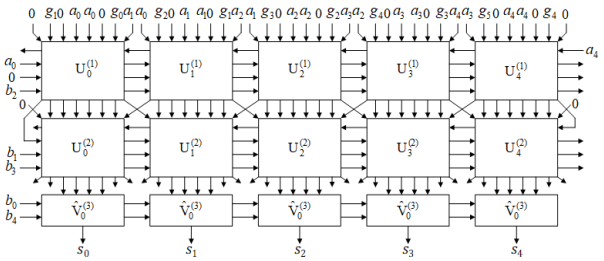
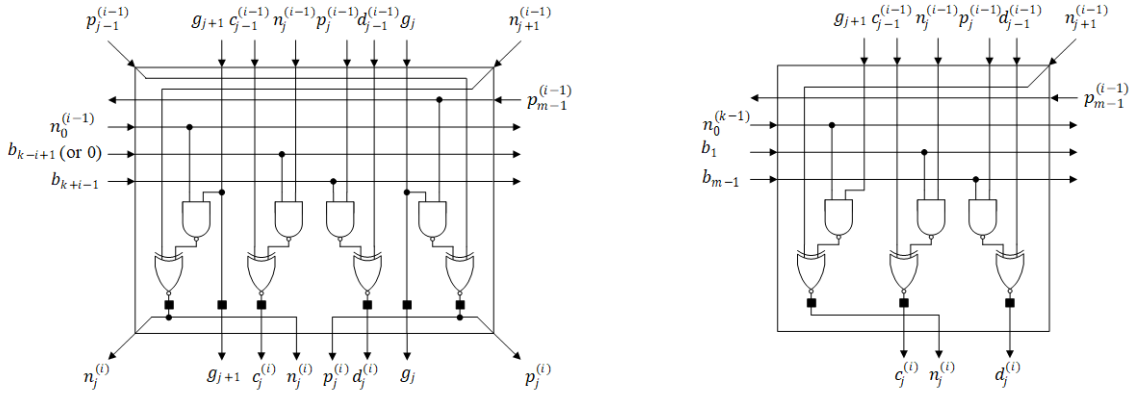
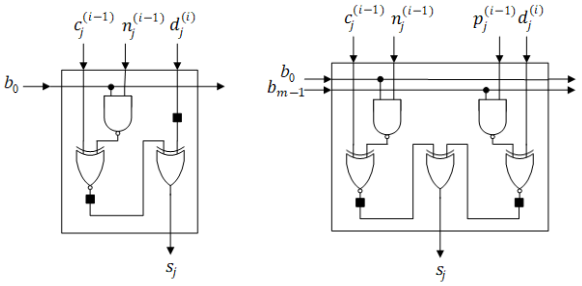


Fig. 2. Proposed Montgomery Multiplier over GF(2^5)



(a) $U_j^{(i)}$ cell (b) $\hat{U}_j^{(i)}$ cell

Fig. 3. $U_j^{(i)}$ and $\hat{U}_j^{(i)}$ cells



(a) $V_j^{(i)}$ cell (b) $\hat{V}_j^{(i)}$ cell

Fig. 4. $V_j^{(i)}$ and $\hat{V}_j^{(i)}$ cells

Fig. 1과 2에서의 $U_j^{(i)}$셀, $\hat{U}_j^{(i)}$셀, V_j셀과 $\hat{V}_j$셀의 자세한 구조는 Fig. 3과 4에서 제시되었다. $U_j^{(i)}$ 셀들은 식 $C^{(i)}$, $D^{(i)}$, $N^{(i)}$와 $P^{(i)}$를 동시에 계산하고, $\hat{U}_j^{(i)}$ 셀들은 $C^{(i)}$, $D^{(i)}$와 $N^{(i)}$를 동시에 계산하고, V_j 셀들은 $C^{(i)}$를 계산한 후에 S를 계산하고, $\hat{V}_j$는 $C^{(i)}$와 $D^{(i)}$를 동시에 계산한 후에 S를 계산한다.

IV. Complexity Analysis

이 장에서는 기존의 곱셈기들과 제안한 곱셈기의 성능을 분석하고 비교한다. 제안한 곱셈기의 논리 게이트의 공간 및 시간 복잡도를 분석하기 위해서, STMicroelectronics [16]의 CMOS065LP CMOS VLSI 기술 기반 표준 셀 라이브러리(standard cell library)를 채택한다. 2-입력 NAND 게이트, 2-입력 AND 게이트, 2-입력 XOR 게이트, 2-입력 XNOR 게이트, 및 D 플립플롭(flip-flop)은 각각 4, 6, 12, 12, 및 30개의 트랜지스터로 구현된다. 2-입력 NAND 게이트, 2-입력 AND 게이트, 2-

Table 1. Comparison of architectures for multiplication over GF(2^m)

Multiplier	Chiou et.al [7]	Huang et.al [9]	Kim-Jeon [10]		Proposed Multiplier	
			even m	odd m	even m	odd m
Area Complexity						
AND_2	$2m^2+2m$	$2m^2$	$2m^2+2m$	$2m^2+2m$	0	0
XOR_2	0	$2m^2$	$2m^2+3m$	$2m^2+3m$	m	m
XOR_3	m^2+m	0	0	0	0	0
$NAND_2$	0	0	0	0	$2m^2$	$2m^2$
$XNOR_2$	0	0	0	0	$2m^2$	$2m^2$
Latch	$3.5m^2+3.5m$	$3m^2$	$3m^2+4m$	$3m^2+3m$	$3m^2-m$	$3m^2-m$
Total Transistor	$117m^2+117m$	$126m^2$	$126m^2+168m$	$126m^2+138m$	$90m^2-30m$	$90m^2-30m$
Time Complexity						
Cell delay	0.19	0.15	0.15	0.15	0.14	0.14
Latency	$m+1$	m	$0.5m+1$	$0.5m+0.5$	$0.5m+1$	$0.5m+0.5$
Total delay	$0.19m+0.19$	$0.15m$	$0.075m+0.15$	$0.075m+0.075$	$0.07m+0.14$	$0.07m+0.07$
AT Complexity						
AT product	$22.23m^3+44.46m^2+22.23m$	$18.9m^3$	$9.45m^3+31.5m^2+25.2m$	$9.45m^3+19.8m^2+10.35m$	$6.3m^3+10.5m^2-4.2m$	$6.3m^3+4.2m^2-2.1m$

입력 XOR 게이트, 2-입력 XNOR 게이트, 및 D 플립플롭(flip-flop)의 전파 지연 시간은 각각 0.02, 0.03, 0.04, 0.04, 및 0.08 ns 이다. 즉, $T_{NAND2}=0.02$, $T_{AND2}=0.03$, $T_{XOR2}=0.04$, $T_{XOR3}=0.08$, $T_{XNOR2}=0.04$, $T_{DFF}=0.08$, 여기서 T_{GATEn}은 n 입력 게이트의 전파지연 시간이다.

Table 1은 기존의 곱셈기들과 제안한 구조를 비교한 것이다. 제안한 세미-시스톨릭 어레이의 트랜지스터 카운트는 $90m^2-30m$이다. Chiou 등 [7], Huang 등 [9], Kim-Jeon [10] 구조의 트랜지스터 카운트는 각각 $117m^2+117m$, $126m^2$, $126m^2+168m$이며, 제안한 세미-시스톨릭 어레이는 이에 비해 약 23%, 28%, 28% 감소되었다.

Chiou 등 [7], Huang 등 [9], Kim-Jeon [10], 제안한 세미-시스톨릭 어레이의 셀 처리 시간은 각각 $T_{AND2}+T_{XOR3}+T_{DFF}$, $T_{AND2}+T_{XOR2}+T_{DFF}$, $T_{AND2}+T_{XOR2}+T_{DFF}$, $T_{NAND2}+T_{XNOR2}+T_{DFF}$ 이다. Chiou 등 [7]과 Huang 등 [9]의 지연 시간은 각각 $m+1$, m이고, Kim-Jeon [10]과 제안한 세미-시스톨릭 어레이의 지연 시간은 $0.5m+1$클록 사이클이다. 시스톨릭 어레이의 처리 시간 전체를 비교하기 위해서, 지연시간과 셀 처리 시간을 같이 고려하면, 제안한 세미-시스톨릭 어레이는 Chiou 등 [7], Huang 등 [9], Kim-Jeon [10]에 비해 약 63%, 53%, 6.6% 감소하였다.

제안한 세미-시스톨릭 어레이와 Chiou 등 [7], Huang 등 [9], Kim-Jeon [10]의 세미-시스톨릭 어레이의 전체 처리시간과 트랜지스터 카운터의 곱인 공간-시간 복잡도(AT complexity)를 비교하면, 각각 약 71%, 66%, 33% 감소하였다.

V. Conclusions

본 논문은 $GF(2^m)$상의 몽고메리 곱셈을 활용하여 하드웨어에 효율적인 유한체 곱셈 알고리즘을 제안하였다. 기존의 알고리즘에서는 AND와 XOR 게이트를 사용하여 구현될 수 있는데, 시간 및 공간 복잡도를 줄이기 위해서 NAND와 NOR 게이트를 사용하는 알고리즘으로 개선하였다. 제안한 알고리즘을 기반으로 세미-시스톨릭 곱셈기를 설계하였다. 그리고 기존의 곱셈기들과 제안한 곱셈기의 공간 복잡도 및 지연 시간을 비교하고 분석하였다. 공간 복잡도는 약 23% 감소하였고, 시간복잡도는 약 3.3% 감소하여, 공간-시간 복잡도는 약 33% 감소하였다. 따라서 오류 검출과 정정 기능을 가지는 곱셈기로 구현할 경우 기존의 곱셈기들보다 높은 성능을 가질 수 있다. 또한 지수, 역원 및 나눗셈 연산과 같이 암호학과 오류 정정 코드에서의 중요한 연산의 구현에 기본적인 알고리즘과 모듈로 이용될 수 있다.

ACKNOWLEDGEMENT

This research was supported by Mokpo National Maritime University Research Grant in 2022.

REFERENCES

[1] R.E. Blahut, "*Theory and Practice of Error Control Codes*" Addison-Wesley, Reading, 1983.

[2] R. Lidl and H. Niederreiter, "*Introduction to Finite Fields and Their Applications*" New York, Cambridge University Press, 1994.

[3] A. J. Menezes, P.C. van Oorschot, and S.A. Vanstone, "*Handbook of Applied Cryptography*" Boca Raton, FL, CRC Press, 1996.

[4] B. Schneier, "*Applied Cryptography second edition*" John Wiley & Sons Inc., 1996.

[5] C. L. Wang, J. L. Lin, "Systolic Array Implementation of Multipliers for Finite Fields," IEEE Trans. Circuits Syst., Vol. 38, No. 7, pp. 796-800, Jul. 1991. DOI: 10.1109/31.135751

[6] S.K. Jain, L. Song, and K.K. Parehi, "Efficient Semisystolic Architectures for Finite-field Arithmetic," IEEE Trans. VLSI Syst., Vol. 6, No. 1, pp. 101-113, Mar. 1998. DOI:10.1109/92.661252

[7] C. W. Chiou, C. Y. Lee, A. W. Deng, and J. M. Lin, "Concurrent Error Detection in Montgomery Multiplication over GF(2^m)," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E89-A, No. 2, pp. 566-574, Feb. 2006. DOI:10.1093/ietfec/e89-a.2.566

[8] C.Y. Lee, C.W. Chiou, and J.M. Lin, "Concurrent Error Detection in a Polynomial Basis Multiplier over GF(2^m)," Journal of Electronic Testing: Theory and Applications, Vol. 22, pp. 143-150, Apr. 2006. DOI:10.1007/s10836-006-7446-9

[9] W. T. Huang, C. H. Chang, C. W. Chiou, and F. H. Chou, "Concurrent Error Detection and Correction in a Polynomial Basis Multiplier over GF(2^m)," IET Inf. Secur., Vol. 4, No. 3, pp. 111-124, Sep. 2010. DOI:10.1049/iet-ifs.2009.0160

[10] K.W. Kim and J.C. Jeon, "Polynomial Basis Multiplier Using Cellular Systolic Architecture," IETE Journal of Research, Vol. 60, No. 2, pp. 194-199, Jun. 2014. DOI:10.1080/03772063.2014.

914699

[11] K. Kim, “Efficient Semi-systolic AB^2 Multiplier over Finite Fields,” Journal of the Korea Society of Computer and Information, Vol. 25, No. 1, pp. 37-43, Jan. 2020. DOI:10.9708/jksci.2020.25.01.037

[12] K. Kim, “An Efficient Multiplexer-based AB^2 Multiplier Using Redundant Basis over Finite Fields,” Journal of the Korea Society of Computer and Information, Vol. 25, No. 1, pp. 13-19, Jan. 2020. DOI:10.9708/jksci.2020.25.01.013

[13] K. Kim, “Low-area Bit-parallel Systolic Array for Multiplication and Square over Finite Fields,” Journal of the Korea Society of Computer and Information, Vol. 25, No. 2, pp. 41-48, Feb. 2020. DOI:10.9708/jksci.2020.25.02.041

[14] P. Montgomery, “Modular Multiplication without Trial Division,” Math. Comput., Vol. 44, No. 170, pp. 519-521, Apr. 1985. DOI:10.1090/S0025-5718-1985-0777282-X

[15] C. Koc and T. Acar, “Montgomery multiplication in GF(2^k),” Des. Codes Cryptogr., Vol. 14, No. 1, pp. 57-69, Apr. 1998. DOI:10.1023/A:1008208521515

[16] STMICROELECTRONICS. 65 nm STMicroelectronics CMOS Technology, Standard Cell Library for 65 Nanometer CMOS065LP VLSI Digital Design Platform. [Online] Available at: <https://www.st.com/content/st.com/en.html>.

Authors

Keewon Kim received his M.S. and Ph.D. degrees in Computer Engineering from Kyungpook National University, Korea, in 2001 and 2006, respectively. He is currently an assistant professor in the department of Computer Engineering, Mokpo National Maritime University. He is interested in information security, security protocol, VLSI, and big data analysis.