J Korea Industr Inf Syst Res Volume 18 Number 2
http://dx.doi.org/10.9723/jksiis.2013.18.2.041

유한 체상의 몽고메리 알고리즘 및 하드웨어 구조 설계†

(Design of Montgomery Algorithm and Hardware Architecture over Finite Fields)

김 기 원*, 전 준 철**
(Kee-Won Kim and Jun-Cheol Jeon)

요 약 유한체상의 곱셈기는 오류 제어 코드, 암호시스템 및 디지털 신호처리와 같은 여러 분야의 기본적인 구성 요소이다. 최근 다양한 유한체상의 곱셈기가 세미-시스톨릭 구조를 기반으로 제안되었다. 또한, 몽고메리 알고리즘은 효율적인 곱셈 연산 알고리즘으로 잘 알려져 있다. 본 논문은 유한체 상에서 다항식 표현을 사용하여 효율적인 몽고메리 곱셈 알고리즘을 유도하고 이를 기반으로 세미-시스톨릭 몽고메리 곱셈기를 제안한다. 제안한 곱셈기는 병렬 구조에 적합한 몽고메리 인자를 선택하였으며 전체 계산 구조를 두 부분으로 나누어 동시에 계산할 수 있다. 제안한 곱셈기는 기존의 곱셈기에 비해 시간 복잡도를 30%~50% 정도 줄임으로써 전체 시간 복잡도의 30% 정도를 줄였다.

핵심주제어 : 유한 체, 세미-시스톨릭 구조, 몽고메리 알고리즘

Abstract Finite field multipliers are the basic building blocks in many applications such as error-control coding, cryptography and digital signal processing. Recently, many semi-systolic architectures have been proposed for multiplications over finite fields. Also, Montgomery multiplication algorithm is well known as an efficient arithmetic algorithm. In this paper, we induce an efficient multiplication algorithm and propose an efficient semi-systolic Montgomery multiplier based on polynomial basis. We select an ideal Montgomery factor which is suitable for parallel computation, so our architecture is divided into two parts which can be computed simultaneously. In analysis, our architecture reduces 30%~50% of time complexity compared to typical architectures.

Key Words : finite fields, semi-systolic architecture, Montgomery algorithm

1. 서 론

유한체는 현대 암호 시스템과 오류 제어 코드 분야에서 중요한 역할을 한다[1]-[4]. 특히 이진체 $GF(2^m)$는 본질적으로 VLSI(Very Large Scale Integration) 구현에 적당하여 관심을 끌고 있다. 유한체상의 곱셈은 가장 중요한 연산이며, 지수, 나눗셈, 및 역원 등은 반복적인 곱셈에 의해 계산될 수 있다. 따라서 낮은 공간-시간 복잡도를 가지는 고속의 곱셈기의 설계가 필요하다.

† 본 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(No. 2011-0014977).
* 단국대학교 소프트웨어학과 교수
** 금오공과대학교 컴퓨터공학과 교수(jcjeon@kumoh.ac.kr)

유한체 $GF(2^m)$상의 곱셈을 고속 구현을 위해 다양한 비트-병렬 시스톨릭(systolic) 구조들이 제안되었다[5]-[10]. 세미-시스톨릭 구조에서는 브로드캐스트 신호를 허용하지만 일반적인 시스톨릭 구조에서는 허용하지 않는다. 이로 인해 세미-시스톨릭 구조에서 래치 개수 및 계산 지연 시간이 일반적인 시스톨릭 구조보다 적은 장점이 있다. Wang 등[5]은 $GF(2^m)$상의 일반적인 시스톨릭 곱셈 구조를 제안하였으며 Jain 등[6]은 세미-시스톨릭 구조를 이용하여 Wang 등[5]의 곱셈기보다 향상된 곱셈 구조를 제안하였다. 이 후에 오류 검출 기능을 가진 세미-시스톨릭 곱셈기들이 제안되었다[7]-[10]. Chiou 등[7]과 Lee 등[8]은 $GF(2^m)$ 상에서 오류 검출이 가능한 다항식 기저의 세미-시스톨릭 곱셈기를 제안하였고, Lee 등[8]은 몽고메리 곱셈을 이용하였다. 몽고메리 곱셈[11]은 효율적인 정수 모듈러 곱셈을 위해 제안되었으며, Koc과 Acar [12]에 의해 유한체 $GF(2^m)$ 상의 곱셈으로 확장되었다. Bayat-Sarmadi와 Hasan [9]은 다양한 기저에서의 오류 검출이 가능한 세미-시스톨릭 곱셈기들을 제안하였다. 최근에 Huang 등 [10]은 기존의 곱셈기들보다 성능이 향상된 다항식 기저의 세미-시스톨릭 곱셈기를 제안하였다. 기존에 제안된 구조들은 여전히 높은 회로 복잡도와 긴 지연 시간으로 인해 효율적인 세미-시스톨릭 곱셈기에 관한 연구가 필요하다.

본 논문은 몽고메리 곱셈 알고리즘을 이용하여 다항식 기저의 세미-시스톨릭 구조를 제안한다. 제안하는 곱셈기는 새로운 몽고메리 인자를 선택하고 효율적인 알고리즘을 유도한다. 결과적으로 기존의 곱셈기와 비교하여 공간 및 시간 복잡도에서 우수한 성능을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 유한체와 $GF(2^m)$상에서 몽고메리 곱셈 알고리즘을 고찰한다. 3장에서는 개선된 몽고메리 곱셈을 제안하고 이를 이용한 세미-시스톨릭 곱셈기를 설계한다. 4장에서는 제안한 곱셈기의 공간 및 시간 복잡도를 분석하고 기존의 곱셈기와 비교한다. 마지막으로 5장에서 결론을 맺는다.

2. GF(2^m)상의 몽고메리 곱셈

유한체 $GF(2^m)$는 2^m 개의 다른 원소들을 포함하는 유한체이며 0과 1을 포함하는 $GF(2)$에서 확장된 것이다. 확장된 이진체 $GF(2^m)$는 $GF(2)$상의 다음의 m차의 기약 다항식과 관련이 있다.

$$G(\omega) = g_m\omega^m + g_{m-1}\omega^{m-1} + \cdots + g_1\omega + g_0 \quad (1)$$

여기서 $g_i \in GF(2)$. $G(\omega)$의 근을 x라고 가정하자. 즉, $G(x)=0$.

정수 상에서 효율적인 모듈러 곱셈을 위해 몽고메리 곱셈 알고리즘이 제안되었고, Koc과 Acar에 의해 유한체 $GF(2^m)$의 곱셈으로 확장되었다[11-12]. $GF(2^m)$상에서 몽고메리 곱셈을 살펴보면 다음과 같다. α와 β는 $GF(2^m)$상의 두 원소이고 $\delta = \alpha\beta \bmod G$ 라고 하자. 몽고메리 잉여(Montgomery residue) A와 B는 다음과 같이 정의된다.

$$A = \alpha r \bmod G = \sum_{j=0}^{m-1} a_j x^j \quad (2)$$

$$B = \beta r \bmod G = \sum_{j=0}^{m-1} b_j x^j \quad (3)$$

여기서 r은 $\gcd(r,G)=1$을 만족하는 몽고메리 인자(Montgomery factor)이다. r^{-1}을 r의 역수(multiplicative inverse)라고 하면, $rr^{-1}+GG'=1$을 만족하는 $GF(2^m)$상의 원소 G'이 존재함을 알 수 있다. 그러면, $GF(2^m)$상의 몽고메리 곱셈은 $P=ABr^{-1} \bmod G$이다.

몽고메리 인자의 선택에 따라 몽고메리 곱셈 알고리즘은 간단하고 효율적인 하드웨어로 구현될 수 있다. 문헌 [12]에서 r을 x^m으로 선택하였다. 식 (2)와 (3)의 몽고메리 잉여 A와 B를 사용하면 몽고메리 곱셈 은 $P=(a\,r)(\beta\,r)r^{-1} \bmod G = \delta\,r \bmod G$ 이다. 여기서 P는 $a\beta$의 몽고메리 잉여이다.

3. 제안하는 몽고메리 곱셈기

3.1 제안하는 몽고메리 곱셈 알고리즘

본 논문에서는 문헌 [13]에서 제시된 몽고메리 인자 $r = x^{\lfloor m/2 \rfloor}$을 사용하여 보다 효율적인 알고리즘과 개선된 곱셈기를 설계한다. 그러면 $GF(2^m)$상의 몽고메리 곱셈은 다음과 같다.

$$P = ABx^{-\lfloor m/2 \rfloor} \bmod G \tag{4}$$

식 (1)의 G로부터 x^m과 x^{-1}은 다음과 같다.

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

$$x^{-1} \bmod G = x^{m-1} + \sum_{j=1}^{m-1} g_j x^{j-1} \tag{6}$$

식 (3)을 이용하면 식 (4)는 다음과 같다.

$$\begin{aligned} P &= \left(\sum_{j=0}^{m-1} b_j A x^j \right) x^{-\lfloor m/2 \rfloor} \bmod G \\ &= \left(\sum_{j=0}^{\lfloor m/2 \rfloor - 1} b_j A x^j + \sum_{j=\lfloor m/2 \rfloor}^{m-1} b_j A x^j \right) x^{-\lfloor m/2 \rfloor} \bmod G \\ &= \sum_{j=0}^{\lfloor m/2 \rfloor - 1} b_j A x^{j-\lfloor m/2 \rfloor} \bmod G + \sum_{j=\lfloor m/2 \rfloor}^{m-1} b_j A x^{j-\lfloor m/2 \rfloor} \bmod G \end{aligned} \tag{7}$$

위 식에서 오른쪽의 마지막항은 아래식과 같이 두 부분으로 나눌 수 있다.

$$P = C + D \tag{8}$$

여기서

$$C = \sum_{j=0}^{\lfloor m/2 \rfloor - 1} b_j A x^{j-\lfloor m/2 \rfloor} \bmod G \tag{9}$$

$$D = \sum_{j=\lfloor m/2 \rfloor}^{m-1} b_j A x^{j-\lfloor m/2 \rfloor} \bmod G \tag{10}$$

식 (9)과 (10)에서 Ax^{-j}와 Ax^j의 계산이 필요하다. $\overline{A}^{(j)} = Ax^{-j} \bmod G$와 $A^{(j)} = Ax^j \bmod G$라고 정의하고 $\overline{A}^{(j)}$와 $A^{(j)}$는 다음과 같이 표현된다.

$$\overline{A}^{(i)} = \sum_{j=0}^{m-1} \overline{a}_j^{(i)} x^j, \tag{11}$$

$$A^{(i)} = \sum_{j=0}^{m-1} a_j^{(i)} x^j, \tag{12}$$

여기서, $\overline{A}^{(0)} = A^{(0)} = A$ 이다.

$\overline{A}^{(i)}$의 순환식을 얻기 위해 식 (6)과 식(11)을 이용하면 다음과 같이 전개된다. 여기서 $\overline{A}^{(0)} = A$이고 $1 \le i \le \lfloor m/2 \rfloor$ 이다.

$$\begin{aligned} \overline{A}^{(i)} &= \overline{A}^{(i-1)} x^{-1} \bmod G \\ &= \sum_{j=0}^{m-1} \overline{a}_j^{(i-1)} x^{j-1} \bmod G \\ &= \sum_{j=1}^{m-1} \overline{a}_j^{(i-1)} x^{j-1} + \overline{a}_0^{(i-1)} x^{-1} \bmod G \\ &= \sum_{j=0}^{m-1} (\overline{a}_{j+1}^{(i-1)} + \overline{a}_0^{(i-1)} g_{j+1}) x^j \end{aligned} \tag{13}$$

여기서 $g_m = 1$이고 $\overline{a}_m^{(i-1)} = 0$이다.

$A^{(i)}$의 순환식을 얻기 위해 식 (5)과 식 (12)를 이용하면 다음과 같이 전개되며 $A^{(0)} = A$이고 $1 \le i \le \lceil m/2 \rceil - 1$ 이다.

$$\begin{aligned} A^{(i)} &= A^{(i-1)} x \bmod G \\ &= \sum_{j=0}^{m-1} a_j^{(i-1)} x^{j+1} \bmod G \\ &= \sum_{j=0}^{m-2} a_j^{(i-1)} x^{j+1} + a_{m-1}^{(i-1)} x^m \bmod G \\ &= \sum_{j=0}^{m-1} (a_{j-1}^{(i-1)} + a_{m-1}^{(i-1)} g_j) x^j \end{aligned} \tag{14}$$

여기서 $g_0 = 1$이고 $a_{-1}^{(i-1)} = 0$이다.

식 (9)는 A(i)를 이용하면 다음과 같이 표현된다.

$$\begin{aligned} C &= z\overline{A}^{(0)} + b_{\lfloor m/2 \rfloor - 1} \overline{A}^{(1)} + b_{\lfloor m/2 \rfloor - 2} \overline{A}^{(2)} + \cdots + b_1 \overline{A}^{(\lfloor m/2 \rfloor - 1)} + b_0 \overline{A}^{(\lfloor m/2 \rfloor)} \\ &= z\overline{A}^{(0)} + \sum_{i=1}^{\lfloor m/2 \rfloor} b_{\lfloor m/2 \rfloor - i} \overline{A}^{(i)} \end{aligned} \tag{15}$$

여기서 식의 순환구조를 맞추기 위해 $z\overline{A}^{(0)}$항을 추가하였으며 $z = 0$이다.

위의 식 (17)로부터 C의 계산을 순환식으로 표현하면 다음과 같다. 단, $C^{(0)}$=0 이고 i번째 중간 결과는 $C^{(i)} = \sum_{j=0}^{m-1} c_j^{(i)} x^j$이다.

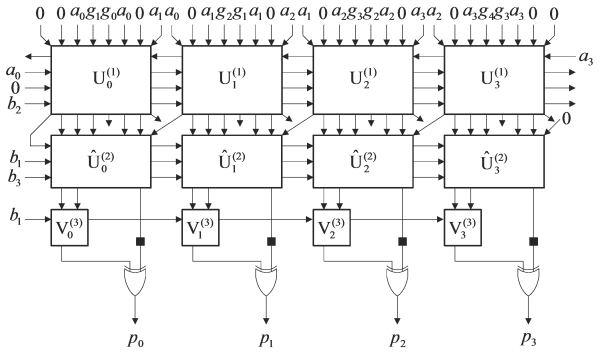
$$C^{(i)} = \begin{cases} C^{(i-1)} + z\overline{A}^{(i-1)} \\ \quad , \text{for } i = 1 \\ C^{(i-1)} + b_{\lfloor m/2 \rfloor + 1 - i}\overline{A}^{(i-1)} \\ \quad , \text{for } 2 \le i \le \lfloor m/2 \rfloor + 1 \end{cases} \tag{16}$$

식 (10)의 D는 $A^{(i)}$를 이용하여 표현하면 다음과 같다.

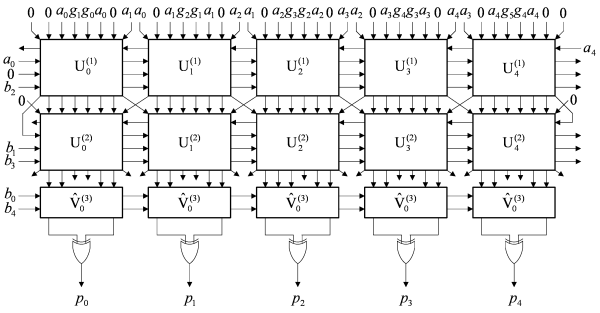
$$\begin{aligned} D = & \; b_{\lfloor m/2 \rfloor} A^{(0)} + b_{\lfloor m/2 \rfloor + 1} A^{(1)} \\ & + \cdots + b_{m-2} A^{(\lceil m/2 \rceil - 2)} \\ & + b_{m-1} A^{(\lceil m/2 \rceil - 1)} \\ = & \sum_{i=0}^{\lceil m/2 \rceil - 1} b_{\lfloor m/2 \rfloor + i} A^{(i)} \end{aligned} \tag{17}$$

식 (17)로부터 D의 계산을 순환식으로 표현하면 다음과 같다. 단, $D^{(0)}$=0 이고 i번째 중간 결과는 $D^{(i)} = \sum_{j=0}^{m-1} d_j^{(i)} x^j$이다.

$$\begin{aligned} D^{(i)} = & \; D^{(i-1)} + b_{\lfloor m/2 \rfloor - 1 + i} A^{(i-1)} \\ & , \text{for } 1 \le i \le \lceil m/2 \rceil \end{aligned} \tag{18}$$



<그림 1> $GF(2^4)$상에서 제안하는 몽고메리 곱셈기

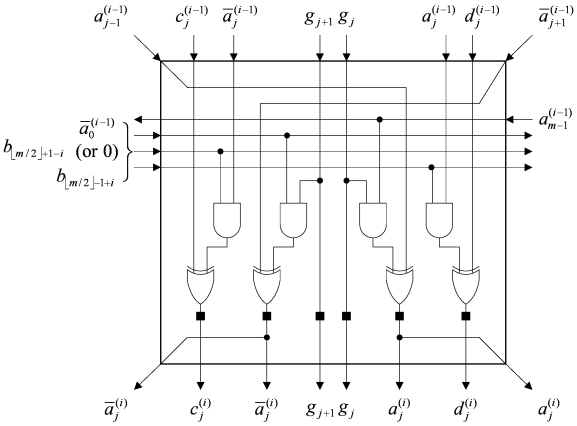


<그림 2> $GF(2^5)$상에서 제안하는 몽고메리 곱셈기

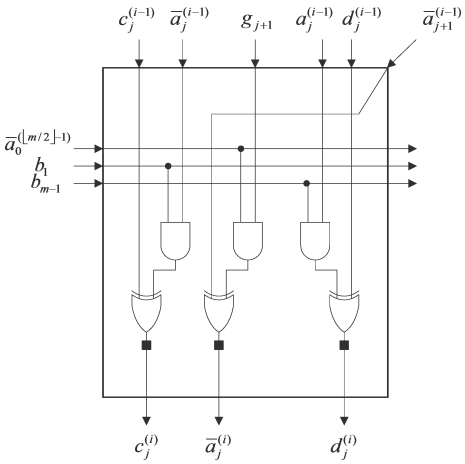
3.2 제안하는 몽고메리 곱셈기

앞 절에서 제안한 몽고메리 곱셈 알고리즘을 이용하여 낮은 지연시간의 몽고메리 곱셈기를 제안한다.

식 (16)과 (18)에서 $C^{(i)}$와 $D^{(i)}$는 데이터 의존성이 존재하지 않으므로 동시에 계산이 가능하다. 즉 몽고메리 인자로 $r=x^{\lfloor m/2 \rfloor}$을 선택함으로서 식 (8)에서 C와 D를 병렬로 동시에 계산할 수 있다. 따라서 식 (16)과 식 (18)의 계산은 동시에 수행될 수 있다.



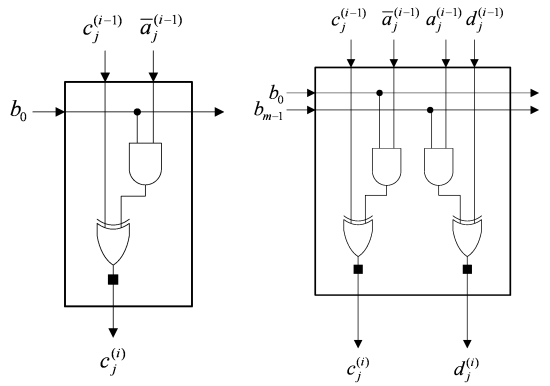
(a) $U_j^{(i)}$ 셀



(b) $\hat{U}_j^{(i)}$ 셀

<그림 3> $U_j^{(i)}$과 $\hat{U}_j^{(i)}$셀의 자세한 회로

본 논문의 그림에서 ■는 1-비트 지연 소자(1-bit latch)를 의미한다. 그림 1과 2는 각각 $GF(2^4)$와 $GF(2^5)$상에서 제안하는 몽고메리 곱셈기의 하드웨어 구조들이고, 곱셈기들의 아랫부분의 XOR 게이트들은 식 (8)에서와 같이 출력되는 결과 C와 D를 합하는 연



(a) V_j 셀 (b) $\hat{V}_j$ 셀

<그림 4> V_j와 $\hat{V}_j$셀의 자세한 회로

산을 수행한다. 그림 1과 2에서의 $U_j^{(i)}$셀, $\hat{U}_j^{(i)}$셀, V_j셀과 $\hat{V}_j$셀은 각각 그림 3과 4에서 자세한 구조를 제시한다.

짝수 m일 때, C와 D의 값은 각각 $\lfloor m/2 \rfloor$과 $\lfloor m/2 \rfloor+1$ 클럭 사이클 후에 얻을 수 있고, 홀수 m이면, $\lfloor m/2 \rfloor+1$ 클럭 사이클 후에 얻을 수 있다. 따라서 제안한 곱셈기는 m값에 따라 다른 구조를 가진다. 예를 들면, m=4일 때 C와 D값은 각각 2와 3클럭 사이클 후에 얻을 수 있고, m=5일 때 3클럭 사이클 후에 얻을 수 있다.

만약 m이 짝수이면 곱셈기는 $0.5m^2-m$개 $U_j^{(i)}$셀, m개 $\hat{U}_j^{(i)}$셀, m개의 V_j셀, m개의 XOR 게이트 및 m개의 1-비트 지연소자로 구성된다. 만약 m이 홀수이면 곱셈기는 $0.5m(m-1)$개 $U_j^{(i)}$셀, m개의 $\hat{V}_j$셀, m개의 XOR 게이트로 구성된다. $U_j^{(i)}$ 셀들은 식 $C^{(i)}$, $D^{(i)}$, $\overline{A}^{(i)}$ 및 $A^{(i)}$를 동시에 계산하고, $\hat{U}_j^{(i)}$ 셀들 $C^{(i)}$, $D^{(i)}$ 및 $\overline{A}^{(i)}$를 동시에 계산하고, V_j 셀들은 $C^{(i)}$를 계산하고, $\hat{V}_j$는 $C^{(i)}$ 및 $D^{(i)}$를 동시에 계산한다.

4. 성능 비교

이 장에서는 제안한 곱셈기와 기존의 곱셈기들의 성능을 분석하고 비교한다. CMOS VLSI 기술에서 n-입력 AND, n-입력 OR, n-입력 XOR와 1-비트 래치는 각각 $2n$+2, $2n$+2, $2n$+2, 그리고 8개의 트랜지스터로 구성된다[14]. 또한 시간 복잡도 비교의 편의를 위해 기존 문헌과 동일하게 2-입력 AND, 2-입력 XOR, 3-입력 XOR, 1-비트 래치의 전파 지연시간을 각각 7, 12, 24, 13으로 가정한다. 그리고 3-입력 XOR 게이트는 두 개의 2-입력 XOR 게이트로 구성된다고 가정한다.

표 1에서 기존의 곱셈기보다 제안한 곱셈기가 효율적인 것을 볼 수 있다. 제안한 곱셈기의 트랜지스터 개수는 $48m^2-2m$이며, 전체 지연시간은 짝수 m일 때, $16m$+32이고, 홀수 m일 때, $16m$+16이다. 기존의 Huang [10]의 곱셈기와 비교하면 유사한 공간 복잡도를 유지하면서 전체 지연시간(latency)을 50%정도 감소하였다. 문헌 [13]에서 제안된 곱셈기에 비해서는 공간복잡도는 10% 미만으로 조금 높아진 반면, 전체 지연시간을 30%가량 줄임으로써 보다 적은 시간 복잡도를 가진다.

<표 1> $GF(2^m)$상에서 곱셈기의 성능 비교

곱셈기	Huang 등[10]	Kim 등[13]		제안한 곱셈기	
		짝수 m	홀수 m	짝수 m	홀수 m
#AND	$2m^2$	$2m^2+m$	$2m^2+2m$	$2m^2$	$2m^2$
#XOR_2	$2m^2$	m	$2m$	$2m^2+m$	$2m^2+m$
#XOR_3	0	m^2+m	m^2	0	0
#1-bit latch	$3m^2$	$3m^2+2m$	$3m^2-2m$	$3m^2-m$	$3m^2-m$
Cell delay	32	44	44	32	32
Latency	m	$0.5m$+1	$0.5m$+0.5	$0.5m$+1	$0.5m$+0.5
#Transistor	$48m^2$	$44m^2+36m$	$44m^2+10m$	$48m^2-2m$	$48m^2-2m$
Total delay	$32m$	$22m$+44	$22m$+22	$16m$+32	$16m$+16

5. 결 론

본 논문은 $GF(2^m)$상의 다항식 기저에서 몽고메리 곱셈을 위한 새로운 알고리즘을 제안하였다. 제안한 알고리즘을 기반으로 병렬 연산이 가능한 세미-시스톨릭 곱셈기를 설계하였다. 제안한 구조는 기존의 곱셈기와 비교하여 시간 복잡도를 30%~50% 정도 감소하였다. 따라서 오류 검출 및 정정 기능을 가지는 구조로 확장할 경우 기존에 제안된 곱셈기보다 높은 성능을 가진다. 또한, 시간 복잡도가 중요한 오류 정정 코드 및 암호학에서의 중요한 연산인 지수, 역원 및 나눗셈 연산의 기본적인 알고리즘 및 구조로 효율적으로 이용될 수 있다.

참 고 문 헌

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

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

[3] 이진호, 김현성, "공개키 암호 시스템을 위한 LFSR 곱셈기 설계", 한국산업정보학회 논문지, 제9권, 제1호, pp.43-48, 2004.

[4] 조경연, 송홍복, "암호와 복호가 동일한 변형 AES", 한국산업정보학회 논문지, 제15권, 제2호, pp.1-9, 2010.

[5] C.L. Wang and J.L. Lin, "Systolic array implementation of multipliers for finite fields GF(2^m)", IEEE Transactions on Circuits and Systems, vol. 38, no. 7, pp.796-800, 1991.

[6] S.K. Jain, L. Song, and K.K. Parehi, "Efficient semisystolic architectures for finite-field Arithmetic", IEEE Transaction on VLSI Systems, vol. 6, no. 1, pp.101-113, 1998.

[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, vol. E89 - A, no. 2, pp.566-574, 2006.

[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, 2006.

[9] S. Bayat-Sarmadi and M.A. Hasan, "Concurrent error detection in finite field arithmetic operations using pipelined and systolic architectures", IEEE Transaction on Computers, vol. 58, no. 11, pp.1553-1567, 2009.

[10] 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 Information Security, vol. 4, Issue 3, pp.111-124, 2010.

[11] P. L. Montgomery, "Modular multiplication without trial division", Mathematics of Computation, vol. 44, pp.519-521, 1985.

[12] C. K. Koc and T. Acar, "Montgomery multiplication in GF(2^k)", Designs Codes and Cryptography, vol. 14, pp.57-69, 1998.

[13] 김기원, 전준철, "GF(2^m)상의 셀룰라 시스톨릭 어레이 기반 몽고메리 곱셈 구조", 한국정보기술학회논문지, 제10권, 제9호, pp.1-6, 2012.

[14] N. Weste, K. Eshraghian, Principles of CMOS VLSI design: a system perspective, Addison-Wesley, Reading, MA, 1985.

김 기 원 (Kee-Won Kim)

- 정회원
- 경성대학교 전산통계학과 이학사
- 경북대학교 컴퓨터공학과 공학석사
- 경북대학교 컴퓨터공학과 공학박사
- 단국대학교 소프트웨어학과 교수
- 관심분야 : 정보보안, 보안프로토콜, 암호H/W

전 준 철 (Jun-Cheol Jeon)

- 정회원
- 금오공과대학교 컴퓨터공학과 공학사
- 경북대학교 컴퓨터공학과 공학석사
- 경북대학교 컴퓨터공학과 공학박사
- 금오공과대학교 컴퓨터공학과 교수
- 관심분야 : 정보보안, RFID, 암호H/W

논 문 접 수 일 : 2012년 11월 08일
1차수정완료일 : 2013년 03월 19일
게 재 확 정 일 : 2013년 03월 19일