논문 2006-43SD-3-6

타원곡선 암호를 위한 시스톨릭 Radix-4 유한체 곱셈기 설계

(Design of a systolic radix-4 finite-field multiplier for the elliptic curve cryptography)

박 태 근*, 김 주 영**

(Taegeun Park and Juyoung Kim)

요 약

타원곡선 암호 시스템에서 유한체 연산은 핵심적인 부분을 차지하고 있지만 곱셈의 경우 연산 과정이 복잡하여 이를 위한 효율적인 알고리즘 및 하드웨어 설계가 필요하다. 본 논문에서는 매우 큰 소수 m을 가지는 $GF(2^m)$상에서 효율적인 면적과 연산시간을 갖는 Radix-4 시스톨릭 곱셈기를 제안한다. 제안된 유한체 곱셈기는 표준기저 방식을 사용하였으며 수학적 정리를 통해 보다 효율적인 알고리즘을 제안하고 이를 VLSI 설계에 적합하도록 시스톨릭 구조를 이용하여 설계하였다. 제안된 구조는 기존의 병렬 곱셈기 및 직렬 곱셈기, 시스톨릭 곱셈기와 비교해서 효율적인 면적과 연산 시간을 갖는다. 본 연구에서는 $GF(2^{193})$에서 동작하는 유한체 곱셈기를 설계하였으며, 하이닉스 0.35㎛ 표준 셀 라이브러리를 사용하여 합성한 결과 최대 동작 주파수는 400㎒이다.

Abstract

The finite-field multiplication can be applied to the elliptic curve cryptosystems. However, an efficient algorithm and the hardware design are required since the finite-field multiplication takes much time to compute. In this paper, we propose a radix-4 systolic multiplier on $GF(2^m)$ with comparative area and performance. The algorithm of the proposed standard-basis multiplier is mathematically developed to map on low-cost systolic cells, so that the proposed systolic architecture is suitable for VLSI design. Compared to the bit-parallel, bit-serial and systolic multipliers, the proposed multiplier has relatively effective high performance and low cost. We design and synthesis $GF(2^{193})$ finite-field multiplier using Hynix 0.35㎛ standard cell library and the maximum clock frequency is 400㎒.

Keywords: finite field multiplier, systolic, radix-4, VLSI

I. 서 론

유한체(finite-field)는 보통 Galois Field(GF)라고 하는데, 유한체를 이용한 연산은 타원 곡선 암호 시스템에서 핵심적인 역할을 수행한다[1]. 유한체 연산은 이진 연산과 다르며 많은 응용분야에서 실시간 처리가 요구된다. 하지만 연산 과정이 복잡하고 면적과 전력 소비의 제한이 따르는 경우가 많기 때문에 유한체 연산을 위한 효율적인 알고리즘 및 이에 대한 하드웨어 설계가 필요하다.

유한체 $GF(2)$의 확장체인 $GF(2^m)$은 이진법을 사용하므로 VLSI 설계 시 효율적인 구조가 가능하며 고속으로 동작하는 시스템을 설계하는데 매우 효과적이다. $GF(2^m)$에서의 덧셈은 단순히 XOR 연산만 수행하면 되므로 매우 간단하지만 곱셈은 복잡하기 때문에 많은 처리 시간과 자원을 필요로 한다. 따라서 $GF(2^m)$ 상의 유한체 곱셈기를 구현하기 위한 다양한 알고리즘이 연구되고 있다.

타원곡선 암호 시스템에서 사용하는 기저 방식에는 크게 두 가지가 있는데 첫 번째는 정규기저(normal

* 정회원, ** 학생회원, 가톨릭대학교 정보통신전자공학부
(School of Information, Communications and Electronics Engineering, The Catholic University of Korea)

접수일자: 2005년6월21일, 수정완료일: 2006년3월8일

basis) 방식을 사용하는 Massey-Omura 곱셈기이며, 두 번째는 표준기저(standard basis)방식을 사용하는 곱셈기이다. 정규기저 곱셈기는 역원과 제곱, 지수승을 연산하는데 매우 효율적인 장점이 있지만 기저의 변환이 필요하다는 단점이 있다. 기저 변환의 복잡도는 체를 생성하는 원시 기약다항식의 선택에 의존한다. 반면 표준기저 방식은 기저의 변환이 필요하지 않으므로 구조적으로 단순하고 규칙적이다. 또한 모듈방식이기 때문에 낮은 하드웨어 복잡도를 가지며 VLSI 설계에 적합하고 어떤 입출력 시스템에도 쉽게 적용할 수 있어서 매우 다양한 응용에서 사용될 수 있으며 높은 차수로의 확장이 정규기저 곱셈기 방식에 비해 쉽다[2]. 유한체 연산을 위한 기약 다항식의 선택은 계산의 복잡도 감소를 위하여 3항 다항식이나 5항식을 권장하고 있다[1].

구조적 측면에서 현재 연구되는 유한체 곱셈기는 직렬 곱셈기, 병렬 곱셈기 및 하이브리드(hybrid) 곱셈기로 크게 나누어 볼 수 있다. Mastrovito에 의하여 제안되어 유한체 곱셈기의 가장 기본적인 구조로 자리잡아온 직렬 유한체 곱셈기는 m 비트의 곱셈을 수행하기 위해서 m 개의 연산 모듈이 필요하다[3]. 매우 단순한 프로세서를 사용하며 모듈성이 높아서 VLSI 구현에 적합한 장점이 있지만 m이 커질수록 레이턴시(latency)가 증가한다는 단점이 있다. 이와는 반대로 m 배의 자원을 더 투자하여 m 배의 속도를 얻어낸 결과가 병렬 유한체 곱셈기이다. 이러한 병렬 곱셈기는 성능은 우수하지만 매우 큰 m을 갖는 타원곡선 암호 시스템의 경우 하드웨어로 구현하는 것이 사실상 불가능하다[4]. 직렬 곱셈기와 병렬 곱셈기의 장점을 혼합하여 구현한 구조로써 기존의 Mastrovito의 직렬 곱셈기 안에 기본적인 연산기로 병렬 곱셈기를 탑재한 방식으로 1999년 Paar에 의하여 제안된 하이브리드 곱셈기가 있다[5]. 직렬 곱셈기와 병렬 곱셈기의 중간 이하의 하드웨어 자원만으로도 수행 속도가 직렬 곱셈기에 비해 상당히 빠른 곱셈기이지만 유한체의 차수가 합성수여야 한다는 제한이 있다.

최근 연구를 살펴보면 VLSI 설계 알고리즘 가운데 시스톨릭 구조를 이용하여 설계하는 경우가 많다[6,7]. 시스톨릭 구조는 동일한 연산을 수행하는 PE(Processing Element)들을 상호 연결함으로써 구현되는데 N차원의 구조가 N-1차원으로 감소되는 효과가 있으며 VLSI 설계에 있어서 중요한 특징인 모듈성과 규칙성을 가지고 있어서 하드웨어 복잡도를 낮추는데 효과적이다. 유한체 곱셈은 덧셈 연산을 반복적으로 수행하며 규칙적으로 입출력이 발생하므로 시스톨릭 구조를 적용하면 효율적으로 설계할 수 있다.

본 연구에서는 타원곡선 암호 시스템에서 동작하는 유한체 곱셈기를 구현하기 위하여 큰 소수 m을 갖는 $GF(2^m)$에서 사용하는 유한체 곱셈기에 대한 VLSI 구조를 제안하였다. 효율적인 면적과 연산시간을 얻기 위하여 m 비트의 승수를 수학적 전개를 통하여 2 비트씩 묶어서 동시에 연산함으로써 기존의 시스톨릭 구조보다 적은 레이턴시와 2 배의 성능을 나타낸다.

제안된 곱셈기는 VLSI 구조에 적합하도록 표준기저 방식과 시스톨릭 구조를 이용하였으며 3항 다항식을 이용하여 구현하였으나 $f_{m-1}=0$인 기약다항식은 차수에 상관없이 모두 사용가능하다. 본 연구에서 제안된 구조는 HDL로 모델링되어 검증되었고 Synopsys사의 Design Compiler를 사용하여 합성하였다.

본 논문의 구성은 다음과 같다. 먼저 Ⅱ장에서는 유한체 곱셈의 이론적 배경을 소개하였고, Ⅲ장에서는 제안된 알고리즘과 그에 대한 하드웨어 구조를 설명하였다. 그리고 Ⅳ장에서는 성능분석을 하였고 마지막으로 Ⅴ장은 본 논문의 결론이다.

Ⅱ. 배 경 이 론

1. $GF(2^m)$유한체 곱셈

차수 m을 가지는 $GF(2)$의 확장체를 $GF(2^m)$이라고 한다. $F(x)=x^m+f_{m-1}x^{m-1}+f_1x+f_0$이 $f_i \in GF(2)$인 차수 m을 가지는 기약다항식일 때 확장체 $GF(2^m)$은 $(GF(2)[x])/F(x)$과 동일구조를 갖는다. 표준기저방식에서 유한체 곱셈은 확장체 내의 두 원소 $A(x)=a_{m-1}x^{m-1}+\cdots+a_1x+a_0$와 $B(x)=b_{m-1}x^{m-1}+\cdots+b_1x+b_0$의 곱 $P(x)=p_{m-1}x^{m-1}+\cdots+p_0=A(X)B(x) mod F(x)$ 와 같이 표현할 수 있으며 $P(x)$도 $GF(2^m)$내의 원소이다. 유한체 곱셈은 아래의 식 (1)을 m 번 반복 수행함으로써 이루어진다.

$$P'^{(i)}=xP'^{(i-1)} mod F(x)+b_{m-i}A(x) \quad (1)$$

(for $i=1,2,\cdots,m; P'^{(0)}=0; P(x)=P'^{(m)}$)

식 (1)에서 $xP'^{(i-1)}$은 $F(x)$에 의해서 모듈러 연산이 될 수 있는 차수가 m인 다항식이다.

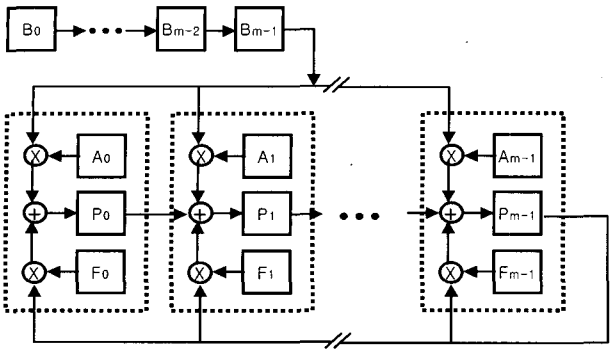


그림 1. $GF(2^m)$ Radix-2 직렬 곱셈기의 구조
Fig. 1. Structure of Radix-2 serial multiplier on $GF(2^m)$.

2. Radix-2 직렬 곱셈기 구조

식 (1)에서 $xP^{(i-1)}$은 $F(x)$에 의해서 모듈러 연산이 될 수 있는 차수가 m인 다항식이다. 위의 식은 Radix-2 직렬 곱셈기로서 그림 1과 같이 하드웨어 구조로 구현될 수 있다[3].

각각의 PE(Processing element)는 3개의 레지스터 A_i, P_i, F_i와 $GF(2)$에서 곱셈기로 동작하는 두 개의 AND 게이트와 덧셈기로 동작하는 한 개의 XOR 게이트 구성되어 있다. 각각의 PE는 외부로부터 B와 P_{m-1}의 입력을 받고 이전 프로세서의 P_{i-1}레지스터로부터 입력을 받아 곱셈을 수행한다. 이러한 직렬 곱셈기는 $GF(2^m)$ 위에서 m개의 프로세서를 사용하며 m 사이클 동안 유한체 곱셈을 수행하게 된다. 이처럼 직렬 곱셈기내의 각각의 프로세서는 단순하고 하드웨어 자원을 많이 소모하지 않는다는 장점이 있지만 m이 커질수록 계산 결과가 늦게 나온다는 단점을 가지고 있다.

3. 시스톨릭 어레이 구조

시스톨릭 어레이는 규칙적으로 입출력을 처리하고 동일한 연산을 수행하는 PE들의 상호 연결로써 이루어진다. 또한 N차원의 구조를 한쪽 방향으로 투영함으로써 새로운 N-1차원의 구조를 얻게 되는데 이때 PE 사이에 적절한 지연 요소를 삽입하므로 레이턴시는 증가하지만 파이프라인으로 인하여 하드웨어의 성능은 증가된다. 따라서 시스톨릭 구조를 적용함으로써 VLSI 설계에 적합한 규칙성과 모듈성을 얻을 수 있다. 그림 1은 [7]에서 제안한 구조로써 시스톨릭 구조를 이용하여 1차원의 Radix-2 시스톨릭 어레이 구조로 구현한 것이다. 시스톨릭 구조를 사용하지 않은 직렬 곱셈기와 비교해서 m개의 프로세서와 m 사이클마다 결과가 출력되는 것은

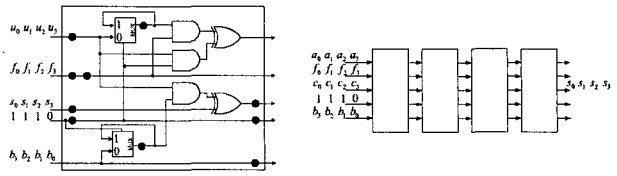


그림 2. [7]에서 제안한 Radix-2 시스톨릭 어레이구조
Fig. 2. Radix-2 systolic array structure in [7].

동일하지만 레이턴시가 $3m$으로 증가하였다. 하지만 VLSI 설계 관점에서 볼 때 임계경로가 짧아지고 효율적인 설계가 가능하다는 장점이 있다.

Ⅲ. 제안된 Radix-4 시스톨릭 알고리즘 및 구조

1. $GF(2^m)$ 유한체 곱셈기의 Radix-4 시스톨릭 알고리즘

본 논문에서는 타원곡선 암호 시스템에서 효율적으로 동작하는 Radix-4 시스톨릭 유한체 곱셈기를 구현하기 위하여 매우 큰 소수 m을 가정하였으며 $f_{m-1}=0$인 모든 기약 다항식에 대해서 적용이 가능하다. 이를 위하여 기존의 Radix-2 유한체 곱셈기의 PE를 단순히 2개씩 묶는 것이 아니라 Radix-4를 구현하기 위한 좀 더 효율적인 알고리즘을 제안한다. 우선 $B(x)$의 값 들을 2비트 씩 묶어서 수식을 정리하면 식 (2)와 같다.

$$\begin{aligned} P(x) &= A(x)B(x) \bmod F(x) \\ &= A(x)\{b_{m-1}x^{m-1}+\cdots+b_1x+b_0\} \bmod F(x) \\ &= A(x)\{b_{m-1}x^{m-1}+b_{m-2}x^{m-2}\} \bmod F(x)+\cdots+ \\ &\quad A(x)\{b_2x^2+b_1x\} \bmod F(x)+\cdots+A(x)b_0 \bmod F(x) \end{aligned} \quad (2)$$

식 (2)의 각각의 항을 $K_i(i=1,3,\cdots,m)$으로 다시 정리하면 아래와 같다.

$$\begin{aligned} K_1 &= A(x)\{b_{m-1}x^{m-1}+b_{m-2}x^{m-2}\} \bmod F(x) \\ &= [A(x)\{b_{m-1}x+b_{m-2}\} \bmod F(x)]x^{m-2} \bmod F(x) \\ &= P_1x^{m-2} \bmod F(x) \\ &\vdots \\ K_i &= K_{i-2}+A(x)\{b_{m-i}x^{m-i}+b_{m-i-1}x^{m-i-1}\} \bmod F(x) \\ &= [P_{i-2}x^2+A(x)\{b_{m-i}x+b_{m-i-1}\} \bmod F(x)]x^{m-i-1} \\ &\quad \bmod F(x) = P_ix^{m-i-1} \bmod F(x) \\ &\vdots \\ K_m &= K_{m-2}+A(x)b_0 \bmod F(x) \\ &= (P_{m-2}x+A(x)b_0) \bmod F(x) = P_m \end{aligned}$$

위의 식에서 정리한 결과를 보면 알 수 있듯이 결국

K_m은 $P(x)=A(x)B(x)modF(x)$와 같으며 이를 정리하면 알고리즘 1과 같다.

알고리즘 1

READ $A,B,F,M,P=0;i=M-1$
DO WHILE $i>2$
  IF $b_ib_{i-1}=00$ THEN $A\leftarrow 0$
  ELSE IF $b_ib_{i-1}=01$ THEN $A\leftarrow A$
  ELSE IF $b_ib_{i-1}=10$ THEN $A\leftarrow Ax+F$
  ELSE IF $b_ib_{i-1}=11$ THEN $A\leftarrow (Ax+F)+A$
  END IF
  $P\leftarrow P+A$
  IF $i>2$ THEN
    $P\leftarrow (Px+F)x+F$; $i\leftarrow i-2$
  END IF
END DO
$P\leftarrow (Px+F)+Ab_0$
P is the Product

위의 알고리즘을 구현하기 위해서는 $Ax+F$, $(Ax+F)+A$ 와 Px^2을 처리하는 부분이 필요하다. $A(x)$에 x를 곱하면 $A(x)x=a_{m-1}x^m+a_{m-2}x^{m-1}+\cdots+a_1x^2+a_0x$ 이며 이 때, x^m은 $F(x)$를 이용하여 아래와 같이 구할 수 있다.

$$x^m=f_{m-1}x^{m-1}+f_{m-2}x^{m-2}+\cdots+f_1x+f_0 \tag{3}$$

위의 식들을 이용하여 아래와 같이 정리할 수 있다.

$$A(x)x+F=a_{m-1}(f_{m-1}x^{m-1}+\cdots+f_0)+a_{m-2}x^{m-1}+\cdots+a_1x^2+a_0x$$
$$=(a_{m-1}f_{m-1}+a_{m-2})x^{m-1}+\cdots+(a_{m-1}f_0) \tag{4}$$

위와 유사한 방법으로

$$P(x)=p_{m-1}x^{m-1}+p_{m-2}x^{m-2}+\cdots+p_1x+p_0 \tag{5}$$

와 같이 되며 따라서 식 (5)에 x^2을 곱한 결과

$$P(x)x^2=p_{m-1}x^{m+1}+p_{m-2}x^m+\cdots+p_1x^3+p_0x^2 \tag{6}$$

와 같이 정리할 수 있다. 식 (3)을 이용하여 x^{m+1}을 구하면

$$x^{m+1}=f_{m-1}x^m+f_{m-2}x^{m-1}+\cdots+f_1x^2+f_0x \tag{7}$$

와 같으며 식 (3)과 (7)을 식 (6)에 대입함으로써 다음의 결과를 얻을 수 있다.

$$P(x)x^2=p_{m-1}(f_{m-1}x^m+f_{m-2}x^{m-1}+\cdots+f_0x)+$$
$$p_{m-2}(f_{m-1}x^{m-1}+f_{m-2}x^{m-1}+\cdots+f_0)+$$
$$p_{m-3}x^{m-1}+\cdots+p_1x^3+p_0x^2$$

타원곡선 암호시스템에서 사용하는 3항 다항식이나 5항 다항식의 공통된 특성은 계수 f_{m-1}이 항상 0을 갖는다는 것이다[1]. 따라서 위의 식을 아래의 식 (8)과 같이 정리할 수 있다.

$$P(x)x^2=(p_{m-1}f_{m-2}+p_{m-3})x^{m-1}+$$
$$(p_{m-1}f_{m-3}+p_{m-2}f_{m-2}+p_{m-4})x^{m-2}+$$
$$\cdots+(p_{m-1}f_2+p_{m-2}f_3+p_1)x^3+$$
$$(p_{m-1}f_1+p_{m-2}f_2+p_0)x^2+$$
$$(p_{m-1}f_0+p_{m-2}f_1)x+p_{m-2}f_0 \tag{8}$$

알고리즘 2

READ $A, B, F, m, P=0, i=j=m-1$
$a_{-1}=f_{-1}=f_{-2}=p_{-1}=p_{-2}=p_{-3}=p_{m-1}=p_{m-2}=0$
DO WHILE $i\geq 2$
 DO WHILE $j\geq 0$
  IF $b_ib_{i-1}=00$ THEN $a_ja_{j-1}\leftarrow 00$
  ELSE IF $b_ib_{i-1}=01$ THEN $a_ja_{j-1}\leftarrow a_ja_{j-1}$
  ELSE IF $b_ib_{i-1}=10$ THEN $a_ja_{j-1}\leftarrow a_{j-1}a_{j-2}$
  ELSE IF $b_ib_{i-1}=11$ THEN $a_ja_{j-1}\leftarrow (a_{j-1}\oplus a_j)(a_{j-2}\oplus a_{j-1})$
  END IF
  $p_jp_{j-1}\leftarrow ((p_{m-1}f_j\oplus(a_{m-1}b_j\oplus p_{m-2})f_j\oplus p_{j-2})\oplus a_j)$
  $((p_{m-1}f_{j-2}\oplus(a_{m-1}b_j\oplus p_{m-2})f_{j-1}\oplus p_{j-3})\oplus a_{j-1})$
  $j\leftarrow j-2$
 END DO
 $i\leftarrow i-2; j=m-1$
END DO
DO WHILE $j\geq 0$
 $p_jp_{j-1}\leftarrow (p_{m-1}f_j\oplus p_{j-1}\oplus b_0a_j)(p_{m-1}f_{j-1}\oplus p_{j-2}\oplus b_0a_{j-1})$
 $j\leftarrow j-2$
END DO

위의 식 (4)와 (8)을 보면 a_{m-1}과 b_i의 논리곱과 p_{m-2}가 1인 경우에 $F(x)$가 중복되어 더해지므로 이를 피하기 위하여 a_{m-1}과 b_i의 논리곱과 p_{m-2}가 상호 배타적으로 1인 경우에만 $F(x)$를 더하도록 한다. 위의 과정을 통해 얻은 수식을 이용하여 $A(x)$와 $P(x)$의 값을 MSB부터 2비트 씩 묶어서 처리하면 알고리즘 2를 얻을 수 있다.

2. $GF(2^m)$ 유한체 곱셈기의 Radix-4 시스톨릭 어레이 구조

제안한 알고리즘에 대한 DG(dependency graph)는

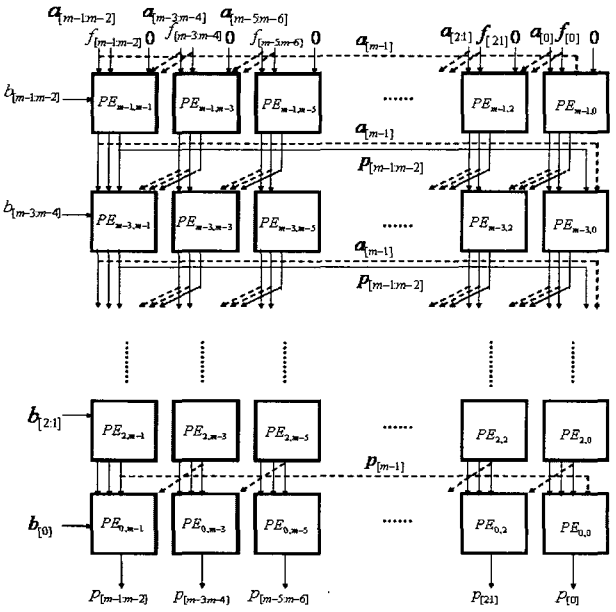


그림 3. $GF(2^m)$ Radix-4 유한체 곱셈기의 2차원 DG
Fig. 3. Two Dimensional DG of Radix-4 finite-field multiplier on $GF(2^m)$.

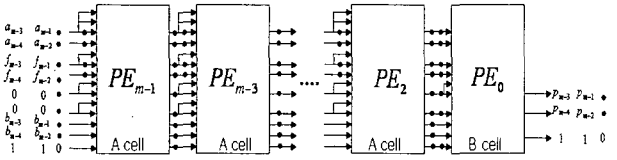
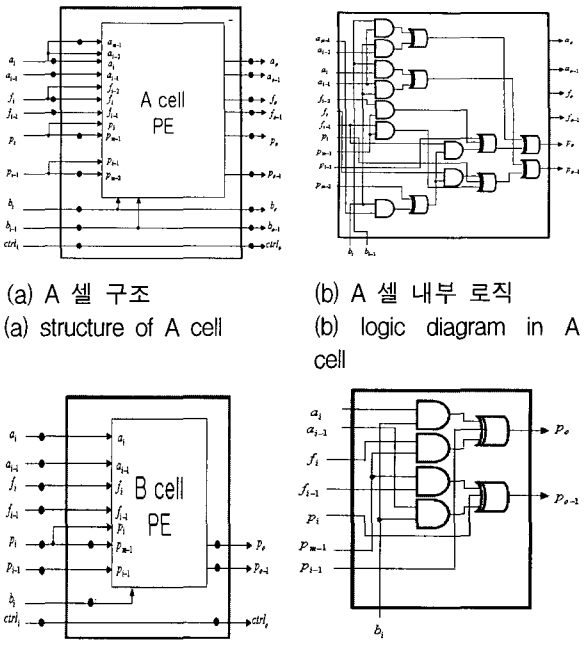


그림 4. $GF(2^m)$ Radix-4 유한체 곱셈기의 시스톨릭 어레이 구조
Fig. 4. Systolic array architecture of Radix-4 finite-field multiplier on $GF(2^m)$.

그림 3과 같으며 모두 $((m+1)/2)^2$개의 PE로 구성되어 있다. 그림에서 실선은 2 비트씩 입력되는 것을 의미하며 점선은 1비트씩 입력되는 것을 의미한다. 그래프를 바탕으로 projection vector $\vec{d}=(1,0)$과 scheduling vector $\vec{s}=(1,-2)$를 이용하여 시스톨릭 매핑하면 행방향의 양방향 데이터 흐름이 제거되어 그림 4와 같은 시스톨릭 구조를 얻을 수 있다. '●' 은 한 사이클의 지연을 갖는 소자를 의미하며 시스톨릭 구조에서 셀마다 적절한 입력과 출력을 맞추기 위해서 삽입된다. 이 때 2차원 배열 상의 PE와 1차원 배열 상의 PE의 내부 구조는 동일하다.

그림 4의 PE(m-1)부터 PE(2)까지의 A 셀 내부 회로는 그림 5(a)와 같고 A 셀 내부의 PE의 내부 회로는 그림 5(b)와 같다. A 셀은 ctrl 신호가 0 일 때 a_{m-1}, p_{m-1}, p_{m-2}, b_i 및 b_{i-1} 의 값을 저장하며 ctrl 신호가 1일 때 각각의 셀에서 곱셈 연산을 수행 한다.

시스톨릭 구조에 의하여 입력 단과 출력 단에 모두



(a) A 셀 구조
(a) structure of A cell
(b) A 셀 내부 로직
(b) logic diagram in A cell
(c) B 셀 구조
(c) structure of B cell
(d) B 셀 내부 로직
(d) logic diagram in B cell

그림 5. PE의 회로도
Fig. 5. The logic diagram of PEs.

19개의 플립플롭이 사용되며 A 셀에서 모두 2 사이클이 소요된다. 그림 5(b)의 출력단의 실선은 입력단의 x_i 값이 바로 출력단의 x_o로 연결되었음을 의미한다.

m이 소수이기 때문에 마지막의 B 셀은 b_0 한 비트에 대한 유한체 곱셈 연산만을 수행하게 된다. 때문에 그림 5(c)(d)와 같이 회로가 매우 단순하게 구현된다. 즉, $Ax+F$와 $(Ax+F)+A$의 연산이 없으므로 a_{m-1}을 저장할 필요가 없으며 곱셈 결과는 $(Px+F)+Ab_0$ 연산만 수행하면 되므로 역시 P_{m-2}를 저장할 필요가 없다. 따라서 ctrl 신호가 0일 때 P_{m-1}과 b_i를 저장하기 위한 플립플롭만 셀 내부에 존재하면 된다. 입력과 출력 단에 모두 12 개의 플립플롭이 사용되며 B 셀에서 모두 2 사이클이 소요된다.

Ⅳ. 결과 및 성능분석

제안한 알고리즘을 이용하여 곱셈기의 동작을 검증하기 위해서 $GF(2^{193})$에서 동작하는 유한체 곱셈기를 설계하였으며 기약다항식은 $F(x)=x^{193}+x^{15}+1$을 사용하였다. 193 비트의 타원곡선 암호 기반의 암호키는 향후 약 20년간 안전한 것으로 알려져 있기 때문에 구현 대상으로 선택하였다[8].

표 1. 기존의 유한체 곱셈기와 제안된 곱셈기간의 성능비교
Table 1. Performance and area of the various finite-field multipliers.

	Ref.[7] Lee	Ref.[9] Chiou	Ref.[10] Kwon	proposed
Throughput	1	$1/m$	$1/m$	$2/(m+3)$
Latency(*cycles*)	$2m-1$	m	$3m$	$m+1$
HW complexity	m^2 AND2 m^2+m-1 XOR2 $3m^2+2m-2$ FFs	m AND2 $m+1$ XOR2 $3m$ FFs	$3m$ AND2 $2m$ XOR2 $2m$ MUX2 10 FFs	$(9m-1)/2$ AND2 $(9m-1)/2$ XOR2 $(19m+5)/2$ FFs
Critical path delay	$T_{2A}+T_{2X}+T_{FF}$	$m(T_{2A}+T_{2X})+T_{FF}$	$T_{2A}+T_{2X}+T_{FF}$	$2T_{2A}+4T_{2X}+T_{FF}$

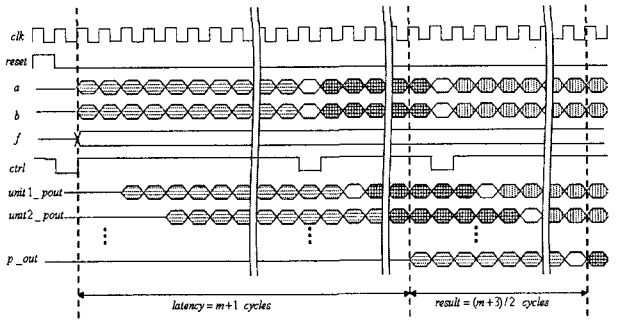


그림 6. 제안한 Radix-4 시스톨릭 어레이 구조의 $GF(2^{193})$ 유한체 곱셈기의 타이밍 다이어그램
Fig. 6. Timing diagram for the proposed Radix-4 systolic array multiplier on $GF(2^{193})$.

그림 6은 전체적인 유한체 곱셈의 타이밍 다이어그램을 보여준다. reset 신호가 1이 되면 곱셈기 내의 모든 플립플롭이 리셋 되고 reset이 0이 되면 제어신호 ctrl 이 한 사이클 동안 0이 되며, 그 다음 사이클부터 A 셀에 승수 $A(x)$, $B(x)$와 $F(x)$의 원소가 2 비트씩 입력되어 유한체 곱셈을 수행한다. 시스톨릭 구조를 갖기 때문에 두 사이클 이후에 A 셀의 곱셈 결과가 출력되어 다음 셀에 입력된다. 이와 같은 과정은 마지막 B 셀까지 동일하게 동작되어 결국 첫 번째 입력이 곱셈기에 입력되어 첫 번째 곱셈 결과가 출력되기까지의 레이턴시는 $m+1$ 사이클이 되며 Radix-4를 사용하므로 $(m+3)/2$ 마다 곱셈 결과를 얻을 수 있다.

제안된 구조는 HDL로 모델링 되었으며 Synopsys Design Compiler를 이용하여 게이트 수준의 합성 결과를 얻었다. 하이닉스 0.35㎛ 표준 셀 라이브러리를 사용하여 합성한 결과 제안한 Radix-4 시스톨릭 어레이 구조의 유한체 곱셈기의 최대 동작 주파수는 400㎒이며 약 21K 게이트가 사용되었다.

본 논문은 타원곡선 암호에서 사용하는 3항 기약다항식과 5항 기약식의 특징인 $f_{m-1}=0$을 이용하여 구조를 개선한 알고리즘을 제안하였으므로 타원 곡선 암호 시스템에 적용 가능한 기약 다항식을 사용하는 다양한 구조의 기존 유한체 곱셈기와 성능을 표 1과 같이 비교 및 평가한다. 표의 하드웨어 복잡도에서 AND2과 XOR2는 각각 2-input AND 게이트와 2-input XOR 게이트를 의미하며 FFs는 1 비트 플립 플롭을 의미한다. 시간 복잡도에서 T_{2A}와 T_{2X}, T_{FF}는 각각 2-input AND 게이트와 2-input XOR 게이트, 1 비트 플립플롭의 지연시간을 의미하며 편의상 3-input XOR 게이트는 2개의 2-input XOR 게이트로 계산하였다.

[7]에서 제안한 병렬 시스톨릭 곱셈기(bit-parallel systolic multiplier)는 m 비트 크기의 입력 값이 곱셈기에 동시에 입력되어 한 사이클 뒤에 곱셈 결과를 얻는 구조이며 $O(m^2)$의 하드웨어를 사용한다. 기존의 다른 병렬 곱셈기보다 시간복잡도는 작지만 공간 복잡도가 증가하였다.

[9]에서 제안한 선형 어레이(linear array) 곱셈기 구조는 초기 m 사이클 후에 처음 곱셈 결과가 출력되며 이후 m 사이클에 한번 결과를 출력되는 구조로써 가장 적은 하드웨어 비용을 갖지만 매우 긴 시간 복잡도를 가지므로 타원곡선 암호 시스템과 같이 매우 큰 m의 유한체 곱셈기를 수행하는 경우 동작 주파수가 상당히 느려질 것으로 예상된다. 또한 기약 다항식에 의해 하드웨어 구조가 고정되므로 다양한 기약 다항식에 유동적으로 적용하는 것이 불가능하다.

[10]에서 제안한 구조는 시스톨릭 어레이(systolic array)구조로써 $3m$의 레이턴시를 갖는다. 또한 한 클록에 한 비트씩 입력되므로 m 사이클마다 결과가 출력된다. [10]의 구조에서 2-input AND 게이트와 2-input XOR 게이트 및 플립플롭의 수는 본 논문에서 제안한 구조보다 적지만 $2m$개의 2-input MUX를 사용하므로 하드웨어 비용면에서는 거의 유사하다. 하지만 성능은 본 논문에서 제안한 구조가 비록 시간 복잡도 면에서는 다소 증가하였지만 처리량은 2배 빠르며 레이턴시는 3배 감소하였다.

V. 결 론

본 논문은 타원곡선 암호시스템에서 핵심적인 연산을 수행하는 $GF(2^m)$ 유한체 곱셈기를 효율적인 면적을 요구하면서도 빠른 곱셈 결과를 얻기 위하여 승수와 피승수를 2 비트씩 묶어서 처리하는 Radix-4 알고리즘 및 효율적인 시스톨릭 어레이 구조를 제안하였다. 이를 위하여 수학적 정리를 바탕으로 한 Radix-4 알고리즘을 제안하였다. 제안된 Radix-4 시스톨릭 곱셈기는 효율적인 알고리즘을 통해서 셀의 내부 구조를 최적화하였으며 $f_{m-1}=0$인 모든 기약다항식에 대해서 차수에 상관없이 적용이 가능하다. 따라서 기존의 병렬 유한체 곱셈기보다 성능은 $2/m$ 정도 감소하였지만 공간 복잡도는 $O(m^2)$에서 $O(m)$으로 감소하였다. Radix-4를 사용하므로 [10]에서 제안한 구조보다 2배의 처리량을 가지면서도 레이턴시는 $1/3$로 감소한 반면 하드웨어 비용은 거의 유사하다. 또한 규칙적인 입출력 구조를 사용하고 시스톨릭 구조를 사용함으로써 하드웨어 구현에 보다 적합하며 기약 다항식의 차수가 증가에 따른 확장이 용이하다. 본 논문에서 제안한 Radix-4 시스톨릭 어레이 구조의 유한체 곱셈기는 면적 대비 성능면에서 효율적이므로 적은 면적에서 고속으로 동작하는 타원곡선 암호시스템의 유한체 곱셈기에 적용이 가능할 것으로 본다.

감사의 글

저자들은 본 연구를 위하여 설계 환경을 제공하여 준 IDEC(IC Design Education Center)에 감사드린다.

참 고 문 헌

[1] IEEE P1363, Standard Specifications for Public key Cryptography, 2000.
[2] I. S. Hsu, T. K. Truong, L. J. Deutsch, and I. S. Reed, "A comparison of VLSI Architecture of Finite Field Multipliers Using Dual, Normal, or Standard Bases," IEEE Trans. on Computers, vol.37, No. 6, pp.735-739, 1988.
[3] E. Mastrovito, "VLSI Architectures for Computation in Galois Fields," Ph. D thesis, Dept. of Electrical Eng., Linkoping Univ., Sweden, 1991.
[4] L. Adleman, and J. DeMarrais, "A Subexponential Algorithm for Discrete Logarithms over All Finite Fields," Advances in Cryptography-CRYPTO 93, D. Stinson, ed., pp.147-158, 1993.
[5] G. Orlando, and C. Paar, "A Super-serial Galois Fields Multiplier for FPGAs and its Application to Public-Key Algorithms," Proceedings of 7th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, pp.232-239, 1999.
[6] Chin-Liang Wang and Jung-Lung Lin, "Systolic Array Implementation of Multipliers for Finite Fields $GF(2^m)$," IEEE Trans. on Circuits and Systems, vol. 38, No. 7, pp.796-800, 1991.
[7] Chiou-Yng Lee, "Low complexity bit-parallel systolic multiplier over $GF(2^m)$ using irreducible trinomials," IEE Proc.-Comput. Digit. Tech., Vol. 150 No. 1, pp.39-42, 2003.
[8] 이찬호, 이정호, "ECC 연산을 위한 가변 연산 구조를 갖는 정규기저 곱셈기와 역원기," 대한전자공학회논문지 제 40권 SD편 제 12호, 2003.
[9] C. W. Chiou, L. C. Lin, F. H. Chou and S. F. Shu, "Low-complexity finite field multiplier using irreducible trinomials," IEE Electronics Letters Vol. 39, No. 24, pp.1709-1711, 2003.
[10] Sonnhak Kwon, Chang Hoon Kim, and Chun Pyo Hong, "Compact linear systolic arrays for multiplication using a trinomial basis in $GF(2^m)$ for high speed cryptographic processors," Computational Science and Its Applications - ICCSA 2005 LNCS 3480, pp.508-518, 2005.

저 자 소 개

박 태 근(정회원)
1985년 연세대학교
전자공학과 졸업.
1988년 Syracuse Univ.
Computer 공학 석사
1993년 Syracuse Univ.
Computer 공학 박사
1991년~1993년 Coherent Research Inc.
VLSI 설계 엔지니어
1994년~1998년 현대전자 System IC 연구소
책임연구원
1998년~현재 가톨릭대학교
정보통신전자공학부 부교수
<주관심분야 : VLSI 설계, CAD, 병렬처리 등임>

김 주 영(학생회원)
2005년 가톨릭대학교
정보통신공학과 졸업.
2005년~현재 가톨릭대학교
컴퓨터공학과 석사과정.
<주관심분야 : VLSI 설계, 영상처리 및 암호시스템 등>