논문 2005-42SD-1-8

크기 가변 유한체 연산기를 이용한 타원곡선 암호 프로세서

(Elliptic Curve Cryptography Coprocessors Using Variable Length Finite Field Arithmetic Unit)

이 동 호*

(Dong-Ho Lee)

요 약

고속 스칼라곱 연산은 타원곡선 암호 응용을 위해서 매우 중요하다. 보안 상황에 따라 유한체의 크기를 변경하려면 타원곡선 암호 보조프로세서가 크기 가변 유한체 연산 장치를 제공하여야 한다. 크기 가변 유한체 연산기의 효율적인 연산 구조를 연구하기 위하여 전형적인 두 종류의 스칼라곱 연산 알고리즘을 FPGA로 구현하였다. Affine 좌표계 알고리즘은 나눗셈 연산기를 필요로 하며, projective 좌표계 알고리즘은 곱셈 연산기만 사용하나 중간 결과 저장을 위한 메모리가 더 많이 소요된다. 크기 가변 나눗셈 연산기는 각 비트마다 궤환 신호선을 추가하여야 하는 문제점이 있다. 본 논문에서는 이로 인한 클록 속도 저하를 방지하는 간단한 방법을 제안하였다. Projective 좌표계 구현에서는 곱셈 연산으로 널리 사용되는 디지트 serial 곱셈 구조를 사용하였다. 디지트 serial 곱셈기의 크기 가변 구현은 나눗셈의 경우보다 간단하다. 최대 256 비트 크기의 연산이 가능한 크기 가변 유한체 연산기를 이용한 암호 프로세서로 실험한 결과, affine 좌표계 알고리즘으로 스칼라곱 연산을 수행한 시간이 6.0 msec, projective 좌표계 알고리즘의 경우는 1.15 msec로 나타났다. 제안한 타원곡선 암호 프로세서를 구현함으로써, 하드웨어 구현의 경우에도 나눗셈 연산을 사용하지 않는 projective 좌표계 알고리즘이 속도 면에서 우수함을 보였다. 또한, 메모리의 논리회로에 대한 상대적인 면적 효율성이 두 알고리즘의 하드웨어 구현 면적 요구에 큰 영향을 미친다.

Abstract

Fast scalar multiplication of points on elliptic curve is important for elliptic curve cryptography applications. In order to vary field sizes depending on security situations, the cryptography coprocessors should support variable length finite field arithmetic units. To determine the effective variable length finite field arithmetic architecture, two well-known curve scalar multiplication algorithms were implemented on FPGA. The affine coordinates algorithm must use a hardware division unit, but the projective coordinates algorithm only uses a fast multiplication unit. The former algorithm needs the division hardware. The latter only requires a multiplication hardware, but it need more space to store intermediate results. To make the division unit versatile, we need to add a feedback signal line at every bit position. We proposed a method to mitigate this problem. For multiplication in projective coordinates implementation, we use a widely used digit serial multiplication hardware, which is simpler to be made versatile. We experimented with our implemented ECC coprocessors using variable length finite field arithmetic unit which has the maximum field size 256. On the clock speed 40 MHz, the scalar multiplication time is 6.0 msec for affine implementation while it is 1.15 msec for projective implementation. As a result of the study, we found that the projective coordinates algorithm which does not use the division hardware was faster than the affine coordinate algorithm. In addition, the memory implementation effectiveness relative to logic implementation will have a large influence on the implementation space requirements of the two algorithms.

Keywords : Elliptic Curve Cryptography, Finite Field, Multiplication, Division, Computer Architecture

I. 서 론

1985년 Kobliz와 Miller에 의하여 타원곡선 암호(Elliptic Curve Cryptography)가 제안된 이후 많은 연구자들이 타원곡선 암호의 효율적인 구현에 대하여 연구하였다. 타원곡선 암호 알고리즘은 어떤 유한체 상에서 타원곡선을 정의하고 그 곡선 상의 점들로 정의된 군 연산에 기반을 두고 있다. 널리 사용되는 유한체에는 $GF(p)$와 $GF(2^m)$이 있다. 참고문헌 [1,2]에서 Henkerson과 Brown 등은 타원곡선 암호 알고리즘의

* 정회원, 경북대학교 전자전기컴퓨터학부
(School of Electrical Engineering and Computer Science, Kyungpook National University)
접수일자: 2004년5월3일, 수정완료일: 2004년12월24일

소프트웨어 구현에 대하여 다방면으로 연구하였다. 순수한 하드웨어 구현을 위해서는 $GF(2^m)$이 효과적이다. 하드웨어 구현의 효율성을 위하여 많은 타원곡선 알고리즘들이 정해진 하나의 유한체 상에서 구현되었다[3,4]. 그러나, 참고문헌 [5]에서는 scalable 프로세서가 연구되었다. Scalable 프로세서의 경우 사용 가능한 유한체 크기가 연산기의 크기에 따라 정해지지 않고 프로세서의 메모리 크기에 의하여 결정된다. 특히, 이제까지 연구된 scalable 프로세서들은 $GF(p)$와 $GF(2^m)$ 유한체를 모두 지원한다[5]. Scable 특성은 $GF(p)$ 연산에서는 매우 중요하다. 그 이유는 $GF(p)$는 정수 연산을 이용하여 구현되고 정수 연산에서 캐리 전송이 연산기의 임계 경로 지연(critical path delay)에 매우 큰 영향을 미치기 때문이다. Scalable 연산 구조를 각 비트 당 소요 자원에 상대적으로 적은 $GF(2^m)$ 곱 연산기 설계에 적용할 경우 부분곱 축적에 사용되어야 하는 Mux 비용을 고려하면 장점이 거의 없다. 참고문헌 [6]에는 가변 길이 유한체 연산기인 versatile 유한체 연산기가 연구되었다. [6]에서는 dual basis 연산과 dot product 하드웨어를 사용하였다. 곱셈과 나눗셈 연산에 소요되는 클록 수는 각각 m과 $2m$이다. 그러나 Dot product 하드웨어는 $O(log_2 m)$의 임계 경로 지연을 가진다. 참고문헌 [7]에서는 서버 시스템을 위한 versatile 유한체 연산기가 연구되었다. [7]에서는 $m \times m$ 곱셈을 위하여 $w_1 \times w_2$ $(w_1 \leq m, w_2 \leq m)$ 하드웨어 곱셈기를 사용하였다. 이 곱셈기의 임계 경로 길이는 w_2에 비례하였으며 곱셈에 소요되는 클록 수는 $(m/w_1)*(m/w_2)$에 비례하였다.

본 논문에서는 참고문헌 [10,12]에서 각각 제안된 나눗셈 연산 구조와 곱셈 연산 구조를 이용하여 두 가지 종류의 가변 길이 유한체 연산기를 사용하는 타원곡선 암호 알고리즘들을 구현하고 성능과 소요 자원을 비교한다. 두 알고리즘은 각각 affine 좌표계와 projective 좌표계를 사용한다. Affine 좌표계 알고리즘은 나눗셈 연산기를 필요로 하며 projective 좌표계 알고리즘은 곱셈 연산기만 사용하나 중간 결과 저장을 위한 메모리가 더 많이 요구된다. 가변 길이 유한체 연산의 경우 효과적인 제곱 연산 구조가 존재하지 않으므로 두 알고리즘 모두 제곱 연산으로 곱셈기를 사용하였다. 연구 결과, 하드웨어 구현의 경우에도 소프트웨어 구현이 경우와 마찬가지로 나눗셈기를 사용하지 않는 projective 좌표계 알고리즘이 속도 면에서 우수함을 알 수 있었다. 또한 반도체 공정과 설계 소프트웨어에서 메모리를 매우 효율적으로 구현하는 경우 projective 좌표계 알고리즘 구현이 더 효율적이다. 마지막으로, 제안된 가변 길이 유한체 연산기를 이용한 타원곡선 암호 알고리즘 프로세서들은 이제까지 연구 문헌에서 알려진 것들보다 구조적으로 간단하여 클록 속도 면에서 우수할 것으로 기대된다.

Ⅱ. 본 론

1. 타원곡선 군 연산 구현

ECC 알고리즘을 구현하기 위해서는 먼저 타원곡선의 계수와 사용하는 유한체의 크기와 표현 방법을 결정하여야 한다. 암호 시스템 설계를 위한 타원곡선 계수 결정에는 두 가지 방법이 있다. 첫 번째 방법은 NIST에서 제공하는 암호 이론을 적용하여 잘 선택된 타원곡선 계수를 사용하는 것이다. NIST는 유한체 크기와 표현 방법에 따라 최적의 타원곡선 계수를 추천하였다. 일반적으로 유한체의 크기는 암호학적 강도를 나타낸다. 두 번째 방법은 난수 함수를 이용하여 계수를 생성하고 효과적인 암호학적 성능 검증 알고리즘을 사용하는 것이다. 어느 방법을 사용하더라도 보안 서버나 다양한 응용 분야에 사용되는 클라이언트의 경우 여러 종류의 타원곡선 계수와 유한체 크기 및 표현 방식을 지원하는 암호 보조프로세서를 사용하는 것이 필요하다.

$GF(2^m)$에 기반을 둔 타원곡선 암호 알고리즘은

$$y^2 + xy = x^3 + ax^2 + b \tag{1}$$

으로 주어지는 Weierstrass 형식의 타원곡선을 사용한다. 이 경우 타원곡선은 계수 a, b에 의하여 결정된다. 곡

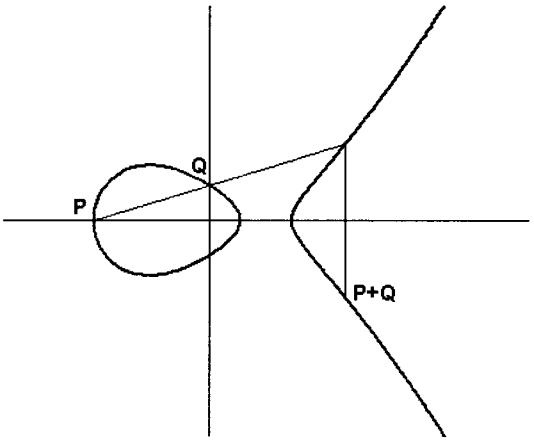

그림 1. 타원곡선 상의 두 점 가산
Fig. 1. Adding two points on ECC.

$$\lambda = \frac{(y_0 + y_1)}{(x_0 + x_1)}$$

$$x_2 = \lambda^2 + \lambda + a + x_0 + x_1$$

$$y_2 = (x_0 + x_2)\lambda + x_2 + y_1$$

(a) $P(x_0,y_0) + Q(x_1,y_1)$

$$\lambda = x_1 + \frac{y_1}{x_1}$$

$$x_2 = \lambda^2 + \lambda + a$$

$$y_2 = (x_1 + x_2)\lambda + x_2 + y_1$$

(b) $2P(x_1,y_1)$

그림 2. 타원곡선 점 덧셈과 2배 연산
Fig. 2. Addition and doubling in elliptic curve points.

Algorithm 1 (Double – and – add)
input P
$Q \leftarrow P$
for i *from* $l-2$ *to* 0 *do*
  $Q \leftarrow 2Q$
  if $n_i = 1$ *then* $Q \leftarrow Q + P$
output Q

그림 3. Double-and-Add 스칼라곱 알고리즘
Fig. 3. Double-and-Add scalar multiplication algorithm.

선 상의 점은 한 쌍의 유한체 값들로 표현된다. 타원곡선 암호에 사용되는 군(group)은 유한체 타원곡선 상의 점들과 항등원으로 작용하는 infinity 점으로 구성된다. 군 합산(addition)은 <그림 1>에 보인 바와 같이 기하학적으로 정의된다.

두 점 P와 Q가 같은 경우 $P+Q$는 $2P$가 되며 이는 위 그림에서 Q가 P에 접근할 때 $P+Q$의 극한치로 나타낼 수 있다. 위의 기하학적인 정의와 타원곡선 식을 이용하면 <그림 2>와 같이 군 합산(group addition) 점과 2배(doubling) 점을 유한체 좌표값을 이용한 유한체 연산식으로 나타낼 수 있다.

타원곡선 상의 어떤 점 P를 n번 더하는 연산을 nP라 표시하며 이것을 계산하는 연산 과정을 스칼라곱 알고리즘이라 부른다. 스칼라곱은 간단한 <그림 3>의 Double-and-Add 알고리즘으로 구현할 수 있다.

연산 속도를 증가시키거나 전력 소모를 줄이기 위하여 많은 경우에 하드웨어 보조프로세서를 이용하여 스칼라곱을 계산한다. 다음에는 두 가지 유한체 연산 회로들을 이용한 타원곡선 암호 프로세서 구현 방법들을 설명한다.

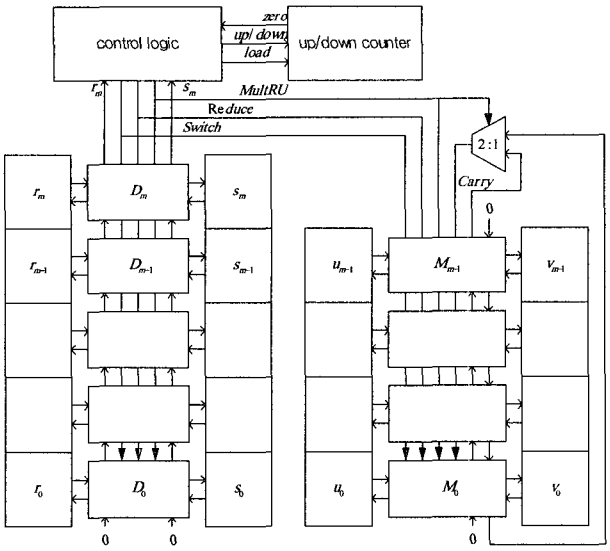

그림 4. Galois 유한체 역원 계산을 위한 하드웨어 구조
Fig. 4. Hardware architecture for Galois field inversion.

2. Affine 좌표계 ECC 프로세서 구현

가. 유한체 나눗셈 연산기

유한체에서 나눗셈은 곱셈의 역원을 곱하는 것이다. 하드웨어 구현에서는 참고문헌 [10]의 하드웨어 곱셈의 역원을 구하는 알고리즘을 이용하였다. 나눗셈은 곱셈 역원 계산 알고리즘에 곱셈의 항등원 1 대신에 dividend을 입력한다. 이 알고리즘은 Euclid의 GCD 알고리즘을 $GF(2^m)$ 유한체 곱셈 역원 계산에 적합하도록 변경한 것으로 나눗셈에는 $2m$ 클록이 소요된다. 아래 <그림 4>에는 하드웨어 구조가 주어져 있다[10].

나. 크기 가변 나눗셈 연산기

본 논문에서는 참고문헌 [8]의 연구 결과를 확장하여 가변 길이 유한체 연산기를 사용하는 타원곡선 암호 프로세서를 설계하였다. 유한체 크기 m이 하드웨어 레지스터 크기 L보다 작은 경우 유한체 값은 MSB (most significant bit) 위치에서부터 시작하여 저장된다. 가변 길이 유한체 연산기의 경우 유한체의 크기 m이 구현된 레지스터 크기 L보다 작은 임의의 수이므로 <그림 4>의 궤환(feedback) 신호를 구현하는데 어려움이 있다. 실험 결과 궤환 신호를 모든 비트 위치에 연결하는 것은 임계 경로 지연(critical path delay)을 지나치게 크게 만들어 최대 동작 주파수를 현저히 감소시킨다.

본 논문에서는 <그림 5>에 보인 바와 같이 소수의

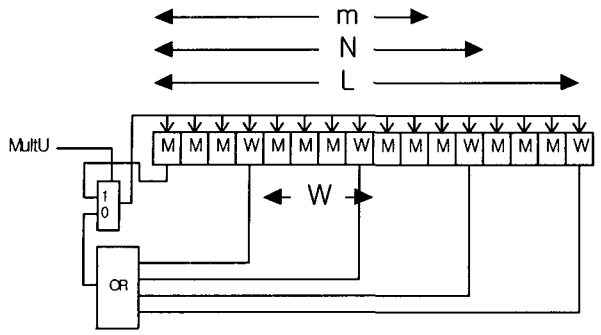

그림 5. 가변 길이 나눗셈 연산기 구조
Fig. 5. Variable length division arithmetic architecture.

궤환 신호를 균일하게 배분하여 일부의 비트 위치에만 연결하였다. 궤환 신호를 각 W 비트 위치마다 연결하면 임의의 유한체 크기 m이 주어질 때 <그림 5>에서의 N은

$$N=\lceil m/W\rceil * W, |N-m| \le W \quad (2)$$

로 정해진다. 즉 N의 크기를 $m+W$보다 작거나 같게 유지할 수 있다.

<그림 4>의 확장 Euclid 알고리즘을 <그림 5>의 가변 나눗셈 연산기 구조 상에서 수행하면 궤환 신호가 LSB (least significant bit)에 직접 연결되어 있지 않기 때문에 나눗셈 결과는 결과 레지스터에서 궤환점까지를 포함하는 영역에 표현되어 있다. 나눗셈 결과는 MSB 방향으로 정렬되어 있어야 하므로 간단한 후처리가 필요하다. 이 후처리는 결과 레지스터를 우로 $N-m$번 쉬프트한 다음 좌로 $N-m$번 쉬프트하면 된다. 이 과정의 수학적 정확성은 [7]에 자세히 설명되어 있다. 따라서 나눗셈을 완료하는데 걸리는 클록 수는 $2N$이 된다. 이와 같이 변경된 데이터패스 상에서 유한체 곱셈 알고리즘을 구현하면 곱셈은 N 클록에 완료할 수 있다.

다. 스칼라곱 구현

<그림 4>의 데이터패스를 앞의 나항에서 보인 바와 같이 크기 가변 유한체 연산을 가능하게 변경한 후 glue logic을 추가하여 곱셈과 덧셈을 가능하게 한다. 참고문헌 [8]에서 <그림 4>의 나눗셈 연산 구조를 이용하여 m 클록에 곱셈을 완료하는 유한체 LSBF (LSB-first) 곱셈 알고리즘을 추가적인 하드웨어 없이 구현할 수 있음을 보였다. 유한체 가산기 또한 <그림 4>의 연산 구조 상에서 쉽게 구현할 수 있다. 암호 프로세서 설계를 위하여 마이크로프로그램과 유한 상태기를 이용하여 <그림 2>의 타원곡선 군 계산 방법과 <그림 3>의 스칼라곱 계산 방법을 유한체 연산 데이터패스 상에 구현하였다.

INPUT : An integer k > 0 and a point P.
OUTPUT : Q = kP.

1. *set* $k \leftarrow (k_{l-1}k_{l-2}\cdots k_1k_0)_2$.
2. *set* $P_1 \leftarrow P$, $P_2 \leftarrow 2P$.
3. *for* i *from* l-2 *downto* 0 *do*
   if $k_i = 1$ *then*
   set $P_1 \leftarrow P_1 + P_2$, $P_2 \leftarrow 2P_2$.
   else
   set $P_2 \leftarrow P_2 + P_1$, $P_1 \leftarrow 2P_1$.
4. Return ($Q = P_1$)

그림 6. Montgomery 스칼라곱 알고리즘
Fig. 6. Montgomery scalar multiplication algorithm.

3. Projective 좌표계 ECC 프로세서 구현

타원곡선 암호 알고리즘의 소프트웨어 구현에서는 projective 좌표계를 사용하여 스칼라곱을 계산하는 것이 효율적임이 알려져 있다[1,2]. Projective 좌표계를 사용하면 계산량이 많은 나눗셈 연산을 최대한 사용하지 않으므로 속도가 향상된다. 소프트웨어 구현에서 특별히 나눗셈과 곱셈의 연산 시간 비가 큰 경우에는 projective 좌표계를 사용하면 계산 속도 면에서 매우 유리하다. 본절 다항에서 제시된 하드웨어 구현에서의 나눗셈과 곱셈의 계산 속도 비는 2로 소프트웨어 구현에 비하여 매우 작다. 그러나 곱셈 연산은 나눗셈 연산보다 많은 병렬성을 가지고 있다. 곱셈의 병렬성을 이용하면 하드웨어 구현에서도 곱셈의 속도를 매우 향상시킬 수 있다.

가. Montgomery 스칼라곱 알고리즘

본 절에서는 하드웨어에서 가장 효과적으로 스칼라곱을 구현할 수 있는 projective 좌표계 알고리즘인 Montgomery 스칼라곱 알고리즘을 소개한다[9].

<그림 6>의 Montgomery 알고리즘은 <그림 3>의 Double-and-Add 알고리즘의 간단한 변형이다. <그림 6>의 알고리즘에서 invariant는 $P_2 = P_1 + P$이다. P_1은 <그림 3>에서 변수 Q에 해당하며 P_2를 계산하는 두 치환문(assignment)은 이 invariant를 유지하기 위한 목적으로만 사용된다. 단계 4에서 최종 결과로 P_1의 값

을 되돌려 준다. Lopez와 Dahab은 위의 알고리즘을 이용하여 projective 좌표계에서 X 좌표와 Z 좌표만 사용하는 알고리즘을 제안하였다[9]. 그들의 알고리즘에는 for loop 내부에는 나눗셈이 전혀 사용되지 않는다. 본 논문에서는 이 알고리즘을 이용하여 타원곡선 스칼라곱 연산을 구현하였다. 최후의 affine 변형에서 한 번의 나눗셈을 해야 한다. 소프트웨어 경우와 달리 하드웨어 구현에서는 나눗셈 연산을 사용하지 않으므로 Fermat의 little 정리를 이용하여 곱셈기를 사용하여 나눗셈을 구현한다. 이 방법은 매우 많은 클록 수가 요구된다.

곱셈기를 이용하여 나눗셈을 구현하는 데에는 <그림 7>의 간단한 알고리즘을 이용한다. 이 방법을 이용한 나눗셈 방법은 $2m^2$ 클록 사이클이 소요된다. 소프트웨어 구현의 경우 더 빠른 알고리즘이 있다. 위 알고리즘 정확성은 유한체 상의 Fermat의 정리의 결과인 다음 등식에 의존한다.

$$x^{-1} = x^{2^1} x^{2^2} x^{2^3} \ldots x^{2^{m-1}} \quad (3)$$

유한체 제곱은 소프트웨어 구현이나 유한체의 크기와 표현이 고정되어 있을 경우 매우 효율적으로 구현할 수 있다[3]. 그러나, 가변 길이 유한체 연산의 경우 효율적인 제곱 구현 방법이 알려져 있지 않다. Projective 좌표계 구현에서도 affine 좌표계 구현에서와 같이 곱셈 연산 하드웨어를 이용하여 제곱을 수행한다.

INPUT : $x \in GF(2^m)$
OUTPUT : $y = x^{-1}$

1. set $y \leftarrow 1$
2. for i from 0 to $m-2$ do
   set $y \leftarrow x \times y$
   set $y \leftarrow y^2$
3. return y

그림 7. $GF(2^m)$ 상의 역원 계산 알고리즘
Fig. 7. $GF(2^m)$ inverse computation algorithm.

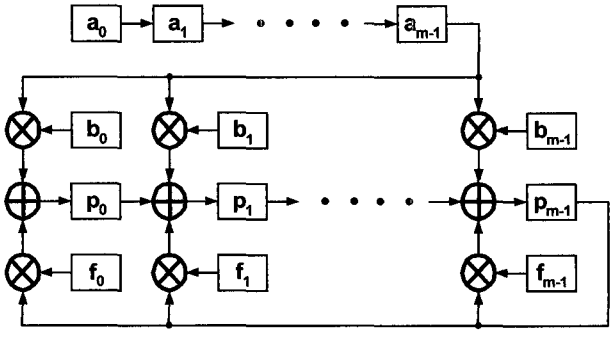

그림 8. MSBF 곱셈 알고리즘
Fig. 8. MSB-first multiplication datapath.

나. 디지트 serial 곱셈 연산기

비트 serial $GF(2^m)$ 유한체 곱셈기의 구현에는 LSBF 알고리즘과 MSBF (MSB-first) 알고리즘이 있다. 두 알고리즘 모두 가변 길이 유한체 연산에 사용될 수 있으나 본 논문에서는 MSBF 알고리즘을 사용하였다. MSBF 알고리즘은 <그림 8>의 데이터패스로 구현할 수 있다.

비트 serial 연산기의 데이터패스는 매우 간단하므로 확장하여 디지트 serial 연산기를 구성하면 디지트의 크기에 비례하는 연산 속도 증가를 얻을 수 있다.

디지트 serial 연산 알고리즘의 하드웨어 구조는 참고문헌 [12]에 자세하게 설명되어 있다. <그림 9>에는 유한체 크기가 8이며 디지트 크기가 4인 경우 디지트 serial 곱셈 방법을 나타내고 있다. $p[7:0]$는 4 비트 MSB 방향으로 쉬프트되어 있으며 피승수 $b[7:0]$는 $a[i]$의 weight에 따라 0, 1, 2, 3 비트씩 MSB 방향으로 쉬프트되어 있다. 만약 $a[i]$가 0이면 피승수는 b 대신 0을 사용한다. 유한체 크기가 8이므로 7보다 weight가 크면 유한체를 정의하는 m차 다항식에서 최고차항을 제외한 식에 해당하는 $f[7:0]$을 더한다. 본 논문에서는 유한체 정의 다항식 f의 상위 $D-1$ 비트가 모두 0이라 가정한다. 이는 [12]에서 효율적인 디지트 serial 곱셈 연산기를 얻을 수 있는 조건임이 증명되어 있다. <그림 9>에서 이 조건은 $f_7 = f_6 = f_5 = 0$이다. 8 비트 디지트 serial 곱셈은 D=4인 경우 2 클록에 완료된다.

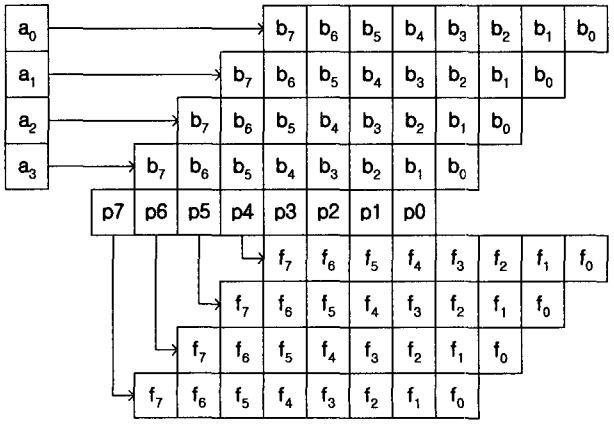

그림 9. 디지트 serial 곱셈 (D=4, m=8)
Fig. 9. Digit serial multiplication (D=4, m=8).

다. 크기 가변 유한체 곱셈기 구현

<그림 8>에서 a 레지스터는 승수, b 레지스터는 피승수, 그리고 f는 유한체를 정의하는 원시 다항식을 저장한다. MSBF 곱셈기를 이용하여 가변 길이 유한체 연산을 수행하기 위해서는 유한체 크기가 하드웨어 레지스터 크기보다 작을 경우 유한체 값을 MSB부터 연속적으로 입력해야 한다. 중간값 저장 메모리에는 모든 유한체 값들이 MSB 쪽으로 정렬되어 있다.

D>0이며 유한체 크기 m이 디지트 크기 D의 정수배가 아닌 경우에도 b, p, f 레지스터에 load 되는 피승수, 부분합, 원시 다항식 등이 모두 MSB 쪽으로 정렬되어 있기 때문에 별다른 전처리 혹은 후처리가 필요하지 않다. 이는 <그림 9>에서 보인 바와 같이 궤환선들이 MSB들로부터 나오기 때문이다. 그러나 a 레지스터의 경우 곱하기 시작 전에 m mod D 만큼 LSB 쪽으로 쉬프트하여야 한다. 이것을 조합회로로 구현하기 위해서는 barrel 쉬프터를 사용해야 하며 디지트 크기 D가 클 경우 멀티 클록에 구현하여 자원 사용을 줄일 필요가 있다.

디지트 serial 곱셈 데이터패스 상에서 덧셈을 수행하기 위해서는 입력값들을 b 레지스터와 f 레지스터에 입력하고 a 레지스터와 p 레지스터는 덧셈을 활성화할 수 있도록 최상위 비트가 1이 되게 설정한다. 구현된 암호 프로세서에서는 스칼라곱 연산은 디지트 serial 데이터패스 상에서 Montgomery 군 알고리즘을 이용하여 구현되었다.

4. affine 좌표계 구현과 projective 좌표계 구현의 비교

Affine 타원곡선 프로세서의 경우 <그림 2>와 <그림 3>의 알고리즘을 직접적으로 구현한다. 사용되는 중간 결과 메모리는 *x0*, *y0*, *a*, *λ*, *x1*, 그리고 *y1*이다. 이들은 <그림 4>에 나타난 유한체 변수값들을 저장하는 데 사용된다. <그림 4>의 나눗셈 연산 구조에 존재하는 4개의 레지스터 중 V 레지스터는 연산에서 accumulator 스타일 중앙 처리 장치에서 accumulator의 역할을 한다. 유한체 나눗셈 연산을 수행할 때에는 4개의 레지스터가 모두 사용되며 곱셈 연산에서는 R 레지스터는 승수 기억 장소로 사용하며 U 레지스터는 피승수를 저장하며 V 레지스터는 부분합을 저장한다. Projective 타원곡선 알고리즘은 <표 1>에 나타난 16개의 중간값 저장 기억 장소를 필요로 한다.

실험에서 사용된 FPGA는 충분한 메모리를 가지고 있으므로 projective 구현의 추가적인 메모리 요구는 문제가 되지 않았다. ASIC이나 다른 종류의 FPGA에 구현하려 할 때에는 affine 구현이 유리한 경우도 있다. 중간 결과 메모리 이외에 타원곡선 암호 알고리즘의 구현에서 칩의 면적을 가장 많이 차지하는 부분은 연산 회로 부분이다. $GF(2^m)$ 연산 장치가 소모하는 면적은 최대 가능한 유한체 크기에 비례한다.

표 1. Projective 좌표 구현에서 사용되는 중간값 변수
Table 1. Intermediate value variables for projective coordinates implementation.

변수명	해설
x0	입력 타원곡선 점의 x 좌표
y0	입력 타원곡선 점의 y 좌표
para	Weierstrass 타원곡선의 상수 b
x1	출력 타원곡선 점의 x 좌표
y1	출력 타원곡선 점의 y 좌표
gxh	유한체 정의 원시 다항식
lsb	LSB만 비트 *1*인 유한체 값
msb	MSB만 비트 *1*인 유한체 값
zero	모든 비트가 *0*인 유한체 값
temp1	중간값 저장
temp2	중간값 저장
temp3	중간값 저장
tx1	중간값 저장
tz1	중간값 저장
tx2	중간값 저장
tz2	중간값 저장

표 2. 구성 셀들의 구성 게이트 수
Table 2. Gate counts of comprising cells.

	AND	XOR	MUX (cell)	MUX (reg)	INV
M cell	*2*	*2*	*4*	*5*	*0*
D cell	*1*	*1*	*4*	*4*	*0*
W cell	*4*	*2*	*4*	*5*	*1*
E cell	*2*	*1*	*5*	*4*	*1*
C cell	*2*D*	*2*D*	*0*	*8*	*0*

<표 2>는 암호 프로세서 연산 회로 부분에 사용된 셀들의 포함 게이트 수를 보여준다. 이는 구현된 원시 프로그램에서 얻은 값으로 표준 셀을 이용한 ASIC 구현 시에 게이트 수들과 일치한다. M 셀은 <그림 4>와 <그림 5>에 나타나 있는 기본 셀을 의미하며 W 셀은 <그림 5>에 나타난 가변 길이 유한체 연산을 위하여 확장한 셀이다. E 셀은 <그림 4>의 D 셀을 확장한 것이다. 연산 회로에 포함된 레지스터들은 군(group) 수준의 알고리즘을 구현하기 위한 데이터 멀티플렉서 기능을 포함하여야 한다. <표 2>에서 MUX(reg)는 레지스터에 부착된 멀티플렉서를 의미한다. <표 2>에서 C 셀 행은 projective 좌표계 구현에서 사용된 구성 게이

트 수로 AND 게이트와 XOR 게이트 수는 디지트 크기 D에 비례한다. Affine 좌표계 구현의 경우 연산 부분은 구현하기 위한 전체 게이트 수는

$$3x+(3a+8m)*\frac{W-1}{W}+(6a+9m+2i)*\frac{1}{W}+9M \quad (4)$$

이다. Projective 좌표계 구현의 경우에는 전체 구현 게이트 수는 $(2aD+2xD+8M)$이다. 여기서 a, x, $m(M)$, i는 각각 하나의 AND, XOR, multiplexor, inverter의 구현에 필요한 게이트 수이다. 이들 세 가지 게이트의 면적 기여도가 같다고 가정하면 D가 4에 이르면 projective 좌표계 구현 연산 회로 게이트 수가 affine 좌표계 구현의 연산 회로 게이트 수에 근접하게 된다. 실제로는 affine 좌표계 연산 회로의 신호 연결 구조가 더 복잡하므로 두 연산기가 같은 면적을 가지는 D는 4보다 클 것으로 기대된다. Affine 알고리즘과 D=4인 projective 알고리즘의 구현 면적 효율성을 비교하면 전자는 더 작은 메모리를 사용하며 후자는 더 작은 연산 회로를 가지고 있다. 따라서 CAD 기술에서 메모리 구현의 논리 구현에 대한 상대적 효율성에 따라 두 구현 방법의 공간 효율성이 결정된다.

물리적인 배치 및 라우팅를 고려하지 않고 최대 동작 주파수를 구하는 것은 매우 어렵다. <그림 5>에서 보면 나눗셈 연산 회로의 임계 경로 지연은 궤환 OR 게이트에 의해 결정되어 $O(log_2\ L/W)$이다. 여기서 W는 구현에 따라 결정된다. 3절 나항에 언급한 바와 같이 m이 W의 정수배에서 멀어질수록 W가 커지면 유한체 연산 클록 수가 다소 증가한다. 한 클록에 동작하는 비교 연산자를 구현하면 임계 경로 지연은 $O(log_2 L)$이 된다. 비교 연산자는 자주 사용되지 않기 때문에 멀티 클록 구현으로 임계 경로 길이를 줄일 수 있다. 디지트 serial 연산 회로의 임계 경로 지연은 $O(log_2 D)$이다[12]. 그러나, 이 연산기를 사용하는 projective 좌표계 암호 연산기의 최대 동작 속도는 D의 크기에 따라 매우 완만하게 증가함이 실험을 통하여 확인되었다.

표 3. 스칼라곱 연산 클록 수 비교
Table 3. Comparison of required clock cycles.

	affine	projective
point doubling	$4m$	$7m/D$
point addition	$4m$	$5m/D$
coord. conv.	0	$2m^2/D$
kP	$6m^2$	$14m^2/D$

<표 3>에는 affine 좌표계 스칼라곱 연산에 소요되는 클록 수와 projective 좌표계 곱 연산에 소요되는 클록 수가 주어져 있다. 참고문헌 [4]에서 나타난 클록 수보다 많은 이유는 두 구현의 경우 모두 곱 연산 회로를 이용하여 제곱 연산을 하기 때문이다. 참고문헌 [4]에서는 FPGA 프로그램밍을 이용하여 유한체 크기를 변경하므로 유한체 크기 변경 시간이 수백 msec 단위이다. D가 3보다 크면 projective 좌표계를 이용한 스칼라곱 구현이 연산 클록 수가 더 작음을 알 수 있다.

III. 실 험 결 과

두 가지 가변 길이 유한체 연산을 이용한 타원곡선 암호 프로세서 구조를 비교하기 위하여 ALTERA EXCALIBUR EPXA10F1020C2를 이용한 SoC 설계 검증 키트 상에서 실험을 수행하였다. ALTERA EXCALIBUR EPXA10F1020C2는 ARM922T 프로세서와 100만 게이트 FPGA가 AMBA 버스를 통하여 연결되어 있다. 먼저 Verilog HDL를 이용하여 알고리즘을 프로그램하고 ModelSim 논리회로 시뮬레이터를 이용하여 검증하였다. 시뮬레이션에서는 QUARTUS-II FPGA 설계 환경에서 제공하는 bus function 모델을 사용하여 AMBA 버스로 키와 평문을 공급하고 암호문을 얻어 소프트웨어 구현과 비교하였다. 시뮬레이션 상에서 설계가 정확하게 동작하면 QUARTUS-II의 논리회로 합성기와 설계 및 배치 도구를 이용하여 합성 결과를 얻고 키트에 다운로드하여 정확한 동작을 확인하였다. ALTERA EXCALIBUR에서 ARM922T와 FPGA는 32 비트 버스 구조로 연결되어 있으나 암호 보조프로세서는 호스트 프로세서와 통신하기 위하여 하위 8 비트만 사용한다. 호스트 프로세서에는 [15]에서 제공하는 타원곡선 암호 소프트웨어가 수행된다. 실험에서는 간단한 Diffe-Hellman 키 공유 프로토콜을 수행한다. 암호 프로세서는 AMBA 버스에 슬레이브로 연결되어 있어서 필요한 계수들과 데이터의 입출력과 스칼라곱 시작은 호스트 마이크로프로세서 상의 C 프로그램에 의하여 수행된다. Configuration 함수에 의하여 m, 원시 다항식, 타원곡선 계수가 출력된다. 유한체 값은 8 비트 버스를 통하여 다중 입출력에 의하여 전송된다. 호스트는 스칼라곱 연산 계산을 위하여 두 점의 X, Y 좌표들을 전송하고 계산이 완료되기를 기다려 출력 점 좌표들

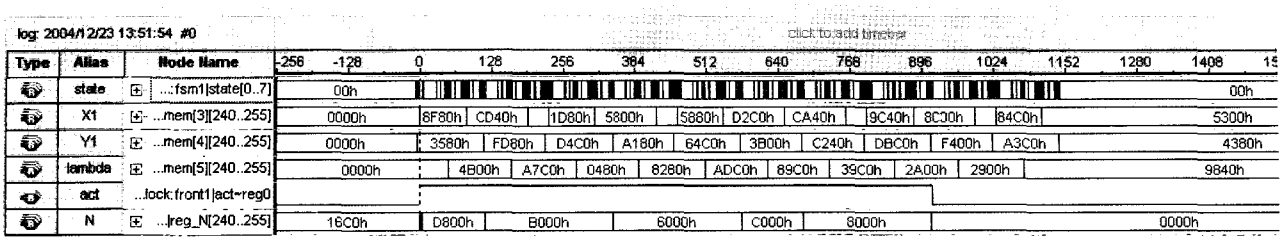

그림 10. Affine 좌표계 FPGA 구현의 결과 파형
Fig. 10. Affine coordinates FPGA implementation result waveforms.

표 4. Affine 좌표계 구현 합성 결과 (W=8)
Table 4. Synthesis results of affine coordinates implementation (W=8).

function blk	# logic cells	# registers	# mem. bits
arithmetic	6270	1624	0
register file	31	0	2048
control	272	10	0
front	2151	441	0
host_int	51	18	0
amba slave	99	24	0
Total	8874	2238	2048

표 5. Affine 좌표계 구현 합성의 최대 동작 주파수
Table 5. Maximum clock frequency of affine implemen-tations.

W	Max. clk. freq.	# of LUT's
1	31.1 MHz	11624
2	29.9 MHz	10840
4	37.7 MHz	9020
8	39.4 MHz	8874
16	44.3 MHz	9108
32	40.1 MHz	9118

표 6. Projective 좌표계 구현 합성 결과 (D=4)
Table 6. Area results of projective coordinates implementation (D=4).

	# of LUT's	# registers	# mem. bits
arithmetic	4350	1031	0
register file	278	260	4096
control	362	43	0
front	1629	585	0
host_int	38	14	0
amba slave	79	15	0
Total	6736	1948	4096

표 7. Projective 좌표계 구현의 동작 주파수
Table 7. Maximum clock frequency for projective coordinates implementation.

D	clock freq.	# of LUT's
1	48.1 MHz	4714
2	49.0 MHz	5391
4	46.8 MHz	6657
8	48.96 MHz	8169
16	42.23 MHz	11950

을 읽어 온다. 계산 시간은 암호 프로세서에서 계산 동안 사용된 시스템 클록을 카운트하여 얻었다.

<그림 10>에는 최대 256 비트 유한체를 지원하는 affine 좌표계 FPGA 구현에서 원시 다항식 $x^{10}+x^3+1$을 사용하는 $a6$=0x9d인 타원곡선 상의 점 (0x23e, 0xd6)를 0x16c0배 곱하여 (0x14c, 0x10e)를 구하는 결과 파형이 나타나 있다. 그림에서 N을 제외한 모든 유한체 값은 MSB 쪽으로 정렬되어 있다.

Affine 좌표계 구현은 <그림 5>에서 보인 바와 같이 워드 크기 W로 계수화되어 있다. W가 작으면 W 셀과 E 셀의 수가 증가한다. 극단적으로 W=1이면 모든 비트 위치가 W 셀과 E 셀로 구성되어 있다. 전체 레지스터 크기는 $W*K$=256으로 고정되어 있으며 K는 W 셀의 수를 나타낸다. <표 4>에는 W=8인 경우 affine 좌표계 구현 합성 결과가 나타나 있다.

Front 블록과 host_int 블록은 함께 호스트 프로세서 인터페이스를 구현한다. Amba slave 블록은 AMBA 버스 Slave 설계로 QUARTUS-II의 설계 예제로부터 얻을 수 있다. <표 4>를 보면 arithmetic 블록, front 블록, 그리고 register file 블록이 대부분의 면적을 소모함을 알 수 있다. <표 4>의 마지막 열의 2048 비트 메모리 중 실제는 1536 비트만 사용되었다.

<표 5>에는 워드 크기 W에 대한 암호 프로세서 최대 동작 주파수와 소요 LUT의 수를 나타낸다. 표에서 알 수 있는 바와 같이 대체로 W가 증가함에 따라 동작 속도가 증가하며 LUT 수는 감소함을 알 수 있다.

<표 6>은 projective 좌표계 구현의 합성 결과이다. <표 4>의 결과와 비교하면 LUT의 수는 25% 작으나 RAM 메모리 비트 수는 2배이다. ASIC 설계의 경우 소요될 면적은 사용하는 반도체 기술과 CAD 소프트웨어에 따라 다르게 나타날 것이다. 특히 메모리 구현이 효율적이면 affine 좌표계 알고리즘이 우수할 것이다.

<표 7>에는 디지트 크기 D가 1에서부터 16까지 변화할 때 스칼라곱 프로세서의 동작 속도와 LUT 수가

표 8. 스칼라곱 연산에 필요한 클록 수
Table 8. Total number of clocks required for scalar multiplications.

m	affine w=1	affine w=8	proj. D=1	proj. D=4	proj. D=16
32	6928	6928	13368	7265	4967
65	28414	30990	62363	22859	12938
75	35229	37249	81727	28359	15017
111	76365	76985	174292	55436	25364
128	104329	104329	194394	71051	31115
129	105074	110282	234273	73281	33033
130	105819	110283	237777	73847	33287
131	106564	110284	239748	73932	33324
134	113467	115019	250465	77365	34090
155	149874	154354	333587	100117	41682
163	173888	178848	368023	109505	45935
193	231536	239432	511011	149715	59391
233	331528	340992	738058	211007	77731

나타나 있다. 사용된 LUT 수는 D가 증가함에 따라 꾸준히 증가하나 동작 속도는 D가 8이 될 때까지 변화가 거의 없다. 구현 보고서를 자세히 조사해 보면 D가 16이 이르기 전에는 연산 회로 부분이 전체 회로의 임계 경로가 아님을 알 수 있다. II장 4절의 이론적인 경우와는 달리 디지트 크기가 8인 경우에도 전체 사용된 LUT의 수는 affine 좌표계 구현의 경우보다 작다.

<표 8>은 유한체 크기에 따른 스칼라곱 연산 완료에 소요되는 클록 수를 나타낸다. 첫 번째 열은 유한체 크기를 나타낸다. 두 번째 열과 세 번째 열은 affine 좌표계 알고리즘이 스칼라곱을 완료하는 데에 걸리는 시간이다. W=1 열은 궤환 신호를 모든 비트 위치에 연결한 경우이며 W=8 열은 매 8번째 비트 위치에만 궤환 신호를 연결한 경우이다. 사용된 원시 다항식은 마지막 열의 경우에는 NIST에서 권고하는 233 비트 유한체를 위한 다항식이며 그 외에는 모두 참고문헌 [15]에서 얻었다. 사용된 스칼라 값들은 난수 알고리즘에서 얻은 것이다. 특히 이들 스칼라 값들의 Hamming weight는 모두 유한체 크기의 약 절반이다. 유한체 크기가 큰 경우 스칼라곱 연산 완료에 필요한 클록은 D의 크기에 반비례함을 알 수 있다. 클록 속도를 40 MHz라 하면 유한체 크기가 m=193인 경우 스칼라곱 연산 시간은 affine 좌표계 구현의 경우 $239432/(40*10^6)$=6.0 msec가 되며 projective 좌표계의 경우 D의 값에 따라 12.7 msec로부터 1.5 msec에 이른다.

회로의 크기와 속도는 공정 기술과 CAD S/W에 많은 영향을 받는다. 따라서 본 논문의 FPGA 구현 실험 결과와 참고문헌 [6,7]의 결과를 직접 비교하기는 어렵다. 따라서

표 9. 다른 가변 길이 프로세서와 비교
Table 9. Comparing with other versatile processors.

	Dual basis [6]	$w1 \times w2$ [7]	affine	projective
technology	CMOS 0.5um	FLEX10KE	APEX20KE	APEX20KE
arithmetic structure	64 bit serial	82x4 bit partial	256 bit serial	256 bit digit serial
Clock Speed	50 MHz	3 MHz	40 MHz	40 MHz
Scalar mult. (163 bit)	N.A.	80 msec	6 msec	1.15 msec

<표 9>에서 이들 결과와 본 논문의 결과를 간단히 요약하고 먼저 이들 논문에서 사용한 암호 프로세서를 연산 구조 면에서 비교하고 <표 9>의 결과에 대하여 설명한다. 참고문헌 [6,7]에는 회로의 크기에 관한 자료가 나타나 있지 않다. 참고문헌 [6]의 Dual basis 구현은 연산 구조의 임계 경로 지연이 유한체 크기에 지수 함수적으로 증가한다. 본 논문의 affine 구현의 경우도 임계 경로 지연이 <그림 5>의 OR 게이트 입력 수에 지수 함수적으로 증가할 수 있다. 다만 W를 증가시켜 임계 경로 지연 증가를 제한할 수 있다. [7]의 암호 연산기의 장점은 아주 작은 유한체 곱셈기를 사용할 수 있다는 점이다. 본 논문의 projective 구현의 경우 비트 serial 곱셈기보다 작은 곱셈기를 사용할 수 없으나 회로의 구조가 간단하여 동일한 성능의 경우 [7]의 암호 연산기보다 크기 면에서 유리하다. <표 9>에서 나타나 있는 바와 같이 본 논문에서 제안된 프로세서들이 최대 가능 비트 수가 4배 크며 FPGA로 구현되었음에도 참고문헌 [6]에서 보고된 ASIC 구현 프로세서보다 클록 속도 면에서 빠르나 두 논문에서 사용한 공정 기술의 차이도 상당한 영향을 미쳤을 것으로 보인다. 세 번째 난에는 참고문헌 [7]에서 제시한 직사각형 유한체 곱셈기를 이용한 ECC 프로세서의 FPGA 구현 결과를 요약한 것으로 구현 구조를 볼 때 클록 속도는 최신 공정을 사용할 경우 개선이 가능하다.

마지막으로 참고문헌 [17]에서 제안된 프로세서는 각 비트 당 적어도 2개의 AND 게이트, 6개의 2 입력 Mux, 2개의 XOR 게이트, 3개의 4 입력 Mux, 그리고 1개의 5 입력 Mux를 사용한다. 이를 식 (4)과 같이 나타내면 $(2x+2a+6m+13M)$이 되어 단위 비트 당 연산자 수는 본 논문의 Affine 구현의 경우와 유사하게 된다. 또한 나눗셈 연산에 $4m$ 클록 사이클을 사용하므로 스칼라곱 계산에 필요한 클록 사이클이 증가한다.

Ⅳ. 결 론

본 논문에서는 타원곡선 암호 알고리즘을 위한 두 가지 가변 길이 유한체 연산기 회로에 대하여 연구하였다. 하나의 알고리즘에서는 확장된 Euclid 알고리즘을 사용했으며 다른 하나에서는 디지트 serial 곱셈 기법을 이용하였다. 가변 길이 유한체 연산기는 회로 구현 시 정해진 크기보다 작은 유한체 연산을 모두 지원한다. 제안된 각각의 유한체 연산기를 이용하여 타원곡선 암호 알고리즘에서 널리 사용되는 스칼라곱 연산 보조프로세서를 구현하였다. 나눗셈 연산기를 가변 길이화하기 위해서는 각 비트마다 궤환 신호선을 추가하여야하는 문제점이 있다. 본 논문에서는 이로 인한 클록 속도 저하를 방지하는 간단한 방법을 제안하였다. 디지트 serial 곱셈 연산에 기반을 둔 projective 좌표계 구현은 디지트 크기를 변경하여 속도와 크기를 자유롭게 조절할 수 있다. 디지트 serial 곱셈기를 가변 길이화는 나눗셈보다 간단하나 디지트 크기에 비례하는 배럴 쉬프터가 필요하다. 다만 필요한 중간 결과값 저장 메모리는 나눗셈기를 이용한 affine 좌표계 구현보다 2배 이상 사용한다. 따라서 어떠한 반도체 설계 기술에 따라서는 affine 좌표계 구현이 더욱 효율적일 수도 있다.

실험에서는 최대 256 비트 크기의 연산이 가능한 크기 가변 유한체 연산기를 사용한 암호 프로세서를 ALTERA EXCALIBUR 상에서 구현하고 자원 요구량과 성능을 비교하였다. Affine 좌표계 구현의 경우 2048 비트의 메모리를 사용하였으며 스칼라곱 수행 시간은 6 msec이었다. Projective 좌표계 구현의 경우 4096 비트의 메모리를 사용하였으며 디지트 크기가 1에서 16까지 변화할 때 스칼라곱 시간은 16 msec에서 1.15 msec로 감소한다. 이때 사용 LUT 수는 약 3배가량 증가하였다.

제안된 스칼라곱 연산 프로세서들은 [6,7]에서 제안된 가변 길이 유한체 연산기를 이용한 타원곡선 암호 프로세서 스칼라곱 연산기들과 비교하였을 때 구조적으로 간단하여 동일한 성능의 경우 자원 요구량이 적을 것으로 기대된다.

참 고 문 헌

[1] D. Hankerson, J. L. Hernandez, and A. Menezes, "Software implementation of elliptic curve cryptography over binary fields," *Cryptographic Hardware and Embedded Systems(CHES 2000), LNCS* 1965, Springer, pp. 2-24, Worcester, MA, USA, August 2000.

[2] M. Brown, D. Hankerson, J. López, and A. Menezes, "Software implementation of the NIST elliptic curves over prime fields," *CT-RSA 2001, LNCS* 2020, Springer, pp. 250-265, 2001.

[3] G. B. Agnew, R. C. Mullin, and S. A. Vanstone, "An implementation of elliptic curve cryptosys -tems over $F_{2^{155}}$," *IEEE Journal on Selected Areas in Communications*, Vol. 11, no. 5, pp. 804-813, June 1993.

[4] G. Orlando and C. Paar, "A high-performance reconfigurable elliptic curve processor for GF(2^m)," *Cryptographic Hardware and Embedded Systems(CHES 2000), LNCS* 1965, Springer, pp. 41-56, Worcester, MA, USA, August 2000.

[5] E. Savas, A. F. Tenca, and Cetin K. Koc, "A scalable and unified multiplier architecture for finite fields GF(p) and GF(2^m)," *Cryptographic Hardware and Embedded Systems(CHES 2000), LNCS* 1965, Springer, pp. 277-292, Worcester, MA, USA, August 2000.

[6] M. A. Hasan and A. G. Wassal, "VLSI algorithms, architectures, and implementation of a versatile GF(2^m) processor," *IEEE Transactions on Computers*, Vol. 49, no. 10, pp. 1064-1073, October 2000.

[7] S. Okada, N. Torii, K. Itoh, and M. Takenaka, "Implementation of elliptic curve cryptographic coprocessor over GF(2^m) on an FPGA," *Cryptographic Hardware and Embedded Systems(CHES 2000), LNCS* 1965, Springer, pp. 25-40, Redwood Shores, CA, USA, August 2000.

[8] J. H. Kim and D. H. Lee, "A compact finite field processor over GF(2^m) for elliptic curve cryptography," *IEEE International Symposium on Circuits and Systems 2002(ISCAS 2002)*, Vol. 2, pp. II-340-II-343, May 2002.

[9] J. López and R. Dahab, "Fast multiplication on elliptic curves over GF(2^m) without precomputa -tion," *Cryptographic Hardware and Embedded Systems(CHES '99), LNCS* 1717, Springer, pp. 316-327, Worcester, MA, USA, August 1999.

[10] H. Brunner, A. Curiger, and M. Hofstetter, "On computing multiplicative inverses in GF(2^m)," *IEEE Transactions on Computers*, Vol. 42, no. 8, pp. 1010-1015, August 1993.

[11] M. Ernst, M, Jung, F. Madlener, S. Huss, and R. Bluemel, "A reconfigurable system on chip implementation for elliptic curve cryptography over GF(2^m)," *Cryptographic Hardware and*

Embedded Systems(CHES 2002), LNCS 2523, Springer, pp. 382-399, Worcester, MA, USA, August 2002.

[12] L. Song and K. K. Parhi, "Low-energy digit -serial/parallel finite field multipliers," *Journal of VLSI Signal Processing Systems*, Vol. 2, no. 22, pp. 1-17, August 1997.

[13] H. Wu, "Low complexity bit parallel finite field arithmetic using polynomial basis," *Cryptogra -phic Hardware and Embedded Systems(CHES '99), LNCS* 1717, Springer, pp. 280-291, Worcester, MA, U.S.A, August 1999.

[14] K. Okeya and K. Sakurai, "Fast multi-scalar multiplication methods on elliptic curves with precomputation strategy using montgomery trick," *Cryptographic Hardware and Embedded Systems(CHES 2002), LNCS* 2523, Springer, pp. 566-581, Worcester, MA, USA, August 2002.

[15] M. Rosing, *"Implementing Elliptic Curve Cryptography,"* Manning Publications Co., 1999

[16] Quartus-II S/W On Line Manual, Altera Corp, http://www.altera.com/product/software/pld/q2/qts -index.html

[17] J. Goodman and A. P. Chandrakasan, "An energy efficient reconfigurable public key cryptographic processor," *IEEE Journal of Solid State Circuits*, Vol. 36, no. 11, pp. 1808-1820, September 2001.

저 자 소 개

이 동 호(정회원)

1979년 서울대학교 전자공학과 학사 졸업.
1981년 KAIST 전산학과 석사 졸업.
1992년 (미) Iowa대 컴퓨터과학과 박사 졸업.
1981년~1992년 ETRI 선임연구원
1992년~1993년 (미) Motorolar senior CAD engineer
1993 ~ 현재 경북대학교 전자전기컴퓨터학부 부교수

<주관심분야 : 컴퓨터 구조, VLSI 설계, 디스플레이 하드웨어>