JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 21, No. 6 : 1083~1091 Jun. 2017

224-비트 소수체 타원곡선을 지원하는 공개키 암호 프로세서의 저면적 구현

박병관 · 신경욱*

A small-area implementation of public-key cryptographic processor for 224-bit elliptic curves over prime field

Byung-Gwan Park · Kyung-Wook Shin*

School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, Kyungbuk 39177, Korea

요 약

NIST 표준에 정의된 소수체(prime field) $GF(p)$ 상의 224-비트 타원곡선을 지원하는 타원곡선 암호 프로세서를 설계하였다. 타원곡선 암호의 핵심 연산인 스칼라 점 곱셈을 수정형 Montgomery ladder 알고리듬을 이용하여 구현하였다. 점 덧셈과 점 두배 연산은 투영(projective) 좌표계를 이용하여 연산량이 많은 나눗셈 연산을 제거하였으며, 소수체 상의 덧셈, 뺄셈, 곱셈, 제곱 연산만으로 구현하였다. 스칼라 점 곱셈의 최종 결과값은 다시 아핀(affine) 좌표계로 변환되어 출력하며, 이때 사용되는 역원 연산은 Fermat's little theorem을 이용하여 구현하였다. 설계된 ECC 프로세서를 Virtex5 FPGA로 구현하여 정상 동작함을 확인하였다. 0.18㎛ 공정의 CMOS 셀 라이브러리로 합성한 결과 10 MHz의 동작 주파수에서 2.7-Kbit RAM과 27,739 GE로 구현되었고, 최대 71 MHz의 동작 주파수를 갖는다. 스칼라 점 곱셈에 1,326,985 클록 사이클이 소요되며, 최대 동작 주파수에서 18.7 msec의 시간이 소요된다.

ABSTRACT

This paper describes a design of cryptographic processor supporting 224-bit elliptic curves over prime field defined by NIST. Scalar point multiplication that is a core arithmetic function in elliptic curve cryptography(ECC) was implemented by adopting the modified Montgomery ladder algorithm. In order to eliminate division operations that have high computational complexity, projective coordinate was used to implement point addition and point doubling operations, which uses addition, subtraction, multiplication and squaring operations over $GF(p)$. The final result of the scalar point multiplication is converted to affine coordinate and the inverse operation is implemented using Fermat's little theorem. The ECC processor was verified by FPGA implementation using Virtex5 device. The ECC processor synthesized using a 0.18 um CMOS cell library occupies 2.7-Kbit RAM and 27,739 gate equivalents (GEs), and the estimated maximum clock frequency is 71 MHz. One scalar point multiplication takes 1,326,985 clock cycles resulting in the computation time of 18.7 msec at the maximum clock frequency.

키워드 : 타원곡선 암호, 투영 좌표계, Jacobian 좌표계, 페르마의 소정리, ECDH 키 교환 프로토콜

Key word : ECC, projective coordinate, Jacobian's coordinate, Fermat's little theorem, ECDH key exchange protocol

Received 25 May 2017, **Revised** 27 May 2017, **Accepted** 31 May 2017

* **Corresponding Author** Kyung-Wook Shin(E-mail:kwshin@kumoh.ac.kr, Tel:+82-54-478-7427)
School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, Kyungbuk 39177, Korea

Open Access https://doi.org/10.6109/jkiice.2017.21.6.1083 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

사물인터넷(Internet of Things; IoT)을 통하여 모든 사물들이 인터넷과 연결되고 지능화되고 있으며, 사용자 중심의 지능형 서비스가 확산되고 있다. IoT 디바이스가 폭발적으로 늘어나고 있는 추세이며, NOKIA는 2025년까지 사물인터넷 접속기기의 수가 약 300억까지 증가할 것으로 추정한다[1]. 이러한 상황에서 IoT 디바이스의 보안위협은 기존의 보안위협과 달리 사용자의 신체나 생명, 재산 등에 직접적인 피해를 발생시킬 수 있다. 따라서, 사물인터넷 산업을 더욱 안전하게 활성화하기 위해서는 사용자 인증(authentication), 데이터의 기밀성(confidentiality), 그리고 무결성(integrity) 검증 등을 위한 다양한 보안기술을 강화할 필요가 있다.

정보보안을 위해서 AES(Advanced Encryption Standard)[2]와 같은 대칭 키 암호시스템(symmetric key cryptography)과 서로 보완적인 역할을 하며, 사용자 인증, 키 교환 등에 사용되는 비대칭 키 암호시스템(asymmetric key cryptography)의 중요성이 더욱 증가하고 있다. 대표적인 비대칭 키 암호 알고리듬으로 RSA(Rivest, Shamir, and Adleman)[3]와 타원곡선 암호(Elliptic Curve Cryptography; ECC)[4]가 있다. RSA는 큰 수의 인수분해가 어렵다는 점에 안전성을 두고 있으며, 타원곡선 암호는 타원곡선 군(group)의 이산대수 문제에 안전성을 두고 있다.

타원곡선 암호는 비트 당 안전도가 다른 공개키 암호보다 효율적이라는 것이 알려지면서 ANSI(American National Standards Institute), WAP(Wireless Application Protocol) 등에서 공개키 암호 표준으로 채택되었으며, 한국정보통신기술협회는 타원곡선 암호를 이용한 인증서 기반 전자서명 알고리듬(EC-KCDSA)을 정보통신단체표준으로 제정하였다[5]. 특히, 1024-비트의 키 길이를 지원하는 RSA와 유사한 안전성을 타원곡선 암호는 160-비트의 짧은 키 길이로 제공한다. 즉, ECC는 적은 양의 메모리 사용과 크기가 제한된 IoT 디바이스, IC 카드와 같은 소형 정보보호 어플리케이션에 적합한 공개키 암호시스템으로 제안되고 있다.

미국 표준기술연구소(NIST)에서 2011년을 기준으로 224-비트 이상의 키 길이를 지원하는 타원곡선 암호시스템을 권고하고 있다. 본 논문에서는 FIPS 186-2에 정의된 P-224 타원곡선을 지원하는 ECC-P224 프로세서를 설계하고, FPGA 구현을 통해 하드웨어 동작을 확인하였다. Ⅱ장에서는 타원곡선 암호 알고리듬에 대해 설명하고, Ⅲ장에서는 ECC-P224 프로세서 설계에 대해 설명한다. 설계된 회로의 기능검증 및 FPGA 구현 결과를 Ⅳ장에 기술하고, Ⅴ장에서 결론을 맺는다.

Table. 1 Elliptic curves recommended by NIST

FIPS 186-2		
	Prime Field	Binary Field
Elliptic Curve	P-192	B-163
		K-163
	P-224	B-233
		K-233
	P-256	B-283
		K-283
	P-384	B-409
		K-409
	P-521	B-571
		K-571

Ⅱ. 타원곡선 암호 알고리듬

타원곡선은 유한체(finite field) 연산이 구현되는 체에 따라 소수체(prime field)와 이진체(binary field)로 구분되며, NIST FIPS 186-2에 정의되어 있는 타원곡선은 표 1과 같다[4]. 소수체에서 5가지의 타원곡선이 정의되어 있으며, 이진체에서 10가지의 타원곡선이 정의되어 있다. 소수체의 경우, 식 (1)의 타원곡선 정의 식에서 계수 a와 b가 $4a^3+27b^2 \neq 0$을 만족한다면, 이 방정식은 실제 타원곡선 상에 존재하지 않는 무한원점 O와 함께 덧셈에 대해 닫혀 있는 군(group)을 형성하게 된다. 소수체 상의 타원곡선은 여인자(cofactor)를 1로 갖는다. 반면에 이진체 상의 타원곡선은 타원곡선을 정의하는 계수 a, b에 따라서 2 또는 4로 비교적 높은 여인자 값을 갖는다. 여인자는 타원곡선의 위수(order)를 기본점(base point)의 위수로 나눈 값으로, 암호학적 응용에서 작은 값일수록 안전하다. 본 설계에서는 높은 안전성을 제공하는 소수체 상의 타원곡선을 지원하는 ECC 프로세서를 설계하였다.

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

ECC가 근간을 두고 있는 타원곡선 이산로그 문제 (Elliptic Curve Discrete Logarithmic Problem; ECDLP)는 타원곡선 상의 임의의 한 점 P에서 양의 정수 k를 곱한 결과값이 $Q = k*P$일 때, 점 P와 Q를 알고 있어도 역연산을 통하여 k를 알아내기가 어렵다는 것을 의미한다. 위 연산을 타원곡선 상의 스칼라 곱셈이라고 하며, 정수 k는 사용자의 비밀키이고 결과값 점 Q는 공개키로 사용된다.

스칼라 곱셈은 점 덧셈 연산과 점 두배 연산으로 계산될 수 있다. 일반적으로 사용되는 affine 좌표상에서 타원곡선 상의 임의의 두 점 $A(x_0, y_0)$와 $B(x_1, y_1)$가 있을 때, 점 연산의 결과값 $C(x_2, y_2)$는 표 2와 같이 유한체 상의 덧셈, 뺄셈, 곱셈, 나눗셈, 제곱 연산으로 계산된다. 특히, 유한체 상의 나눗셈 연산을 구현하기 위해서는 추가적인 레지스터와 제어신호 등이 필요하여 하드웨어 복잡도가 높아진다.

본 논문에서는 소요 사이클이 크고, 하드웨어 복잡도가 높은 나눗셈 연산을 최소화하기 위하여 투영 좌표계의 일종인 Jacobian 좌표를 이용하여 점 연산이 수행되도록 설계하였다. 소수체 상의 타원곡선 방정식을 Jacobian 좌표계를 사용하여 나타내면 식 (2)와 같다. 이 때, $(x, y) = (X/Z^2, Y/Z^3), Z \neq 0$이다.

$$Y^2 = X^3 + aXZ^4 + bZ^6 \qquad (2)$$

Jacobian 좌표계에서 임의의 두 점 $A(X_0, Y_0, Z_0)$와 $B(X_1, Y_1, Z_1)$가 있을 때, 점 연산의 결과값 $C(X_2, Y_2, Z_2)$는 표 3과 같이 계산된다[6]. 설계된 ECC-P224 프로세서의 스칼라 곱셈 결과값은 타 프로세서와의 호환성을 위하여 다시 affine 좌표로 변환하여 출력되도록 설계하였다.

대표적인 스칼라 곱셈 알고리듬은 double-and-add

Table. 2 Point addition and point doubling operations over prime field using affine coordinate

Point addition (C = A + B)	Point doubling (C = 2A)
$\lambda = (y_1 - y_0)/(x_1 - x_0)$ $x_2 = \lambda^2 - x_0 - x_1$ $y_2 = \lambda(x_0 - x_2) - y_0$	$\lambda = (3x_0^2 + a)/(2y_0)$ $x_2 = \lambda^2 - 2x_0$ $y_2 = \lambda(x_0 - x_2) - y_0$

Table. 3 Point addition and point doubling operations over prime field using Jacobian's coordinate

Point addition (C = A + B)	Point doubling (C = 2A)
$U_0 = X_0 Z_1^2,\ U_1 = X_1 Z_0^2$	$M = 3X_0^2 + aZ_0^4$
$S_0 = Y_0 Z_1^3,\ S_1 = Y_1 Z_0^3$	$T = -2S + M^2$
$H = U_1 - U_0,\ R = S_1 - S_0$	$S = 4X_0 Y_0^2$
$X_2 = -H^3 - 2U_0 H^2 + R^2$	$X_2 = T$
$Y_2 = -S_0 H^3 + R(U_0 H^2 - X_2)$	$Y_2 = -8Y_0^4 + M(S - T)$
$Z_2 = Z_0 Z_1 H$	$Z_2 = 2Y_0 Z_0$

[7], Non-Adjacent Form(NAF) [8], Montgomery ladder [9] 알고리듬이 있다. Double-and-add는 비밀키 k의 hamming weight에 따라서 매 루프마다 반복되는 점 덧셈 연산에 점 두배 연산이 추가로 계산된다. NAF 알고리듬의 경우 비밀키 k의 hamming weight를 감소시켜 double-and-add 알고리듬보다 연산량을 줄일 수 있다. 하지만 위의 두 알고리듬은 비밀키의 hamming weight에 따라 연산량이 다르므로, 공격자는 단순 전력분석과 같은 부채널 공격을 통해 비밀키에 대한 정보를 얻을 수 있다. Montgomery ladder 알고리듬의 경우, 비밀키의 hamming weight에 관련없이 동일한 횟수의 점 덧셈 연산과 점 두배 연산을 반복하므로 연산량이 많은 단점은 있으나, 부채널 공격에 보다 안전하다.

소수체 $GF(p)$ 상의 곱셈기 연산은 다양한 알고리듬을 이용하여 구현가능 하다. Interleaved modular 알고리듬[10]의 경우, 제어가 간단하지만 소수체 상의 뺄셈 연산을 필요로 한다. Montgomery modular 알고리듬[10]의 경우 뺄셈 연산은 필요하지 않지만, 피연산자들을 montgomery 도메인으로 변환하기 위해 추가적인 연산이 필요하다. 소수체 $GF(p)$ 상의 덧셈/뺄셈 연산은 소수 p를 모듈러로 하는 연산이며, 캐리 전파에 의한 최대 지연이 하드웨어의 동작 주파수에 영향을 미치게 된다. 캐리 전파 지연을 줄이기 위해서 CSelA(Carry Select Adder), CSavA(Carry Save Adder), CLAA(Carry Look Ahead Adder) 등을 이용하여 소수체 상의 덧셈/뺄셈기를 구현할 수 있다.

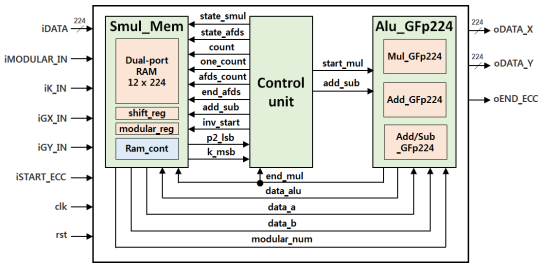
Jacobian 좌표를 사용할 경우 점 연산에서는 소수체 상의 역원 연산이 필요하지 않지만 affine 좌표로 역변

환하는 과정에서 한번의 역원 연산이 필요하다. 소수체 상의 역원 연산 알고리듬은 Binary 알고리듬[11]과 Fermat's little theorem[12]이 있다. Binary 알고리듬의 경우, 역원 연산에 필요한 클록 사이클은 적지만 매 루프마다 소수체 상의 덧셈/뺄셈 연산이 필요하여 제어가 복잡하고, Fermat's little theorem을 이용한 역원기 보다 레지스터가 추가로 필요하다. Fermat's little theorem을 이용한 역원기의 경우 반복적인 유한체 곱셈 연산으로 역원을 구한다.

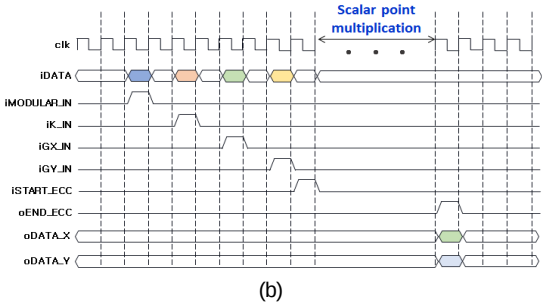
Ⅲ. 타원곡선 암호 프로세서 설계

3.1. 전체 구조 및 스칼라 곱셈 구현

FIPS 186-2에 정의된 소수체 상의 224-비트 타원곡선 P-224를 지원하는 ECC 프로세서를 설계하였다. 전체 구조는 그림 1-(a)와 같으며, 스칼라 곱셈을 위한 정수 k, 모듈러 연산을 위한 소수 p, 그리고 스칼라 곱셈의 중간 결과값을 저장하는 Smul_Mem 블록, 소수체 상의 곱셈, 덧셈/뺄셈 연산을 수행하는 Alu_GFp224 블록, 그리고 FSM(Finite State Machine)을 이용하여 스칼라 곱셈 연산을 제어하는 제어블록으로 구성된다.

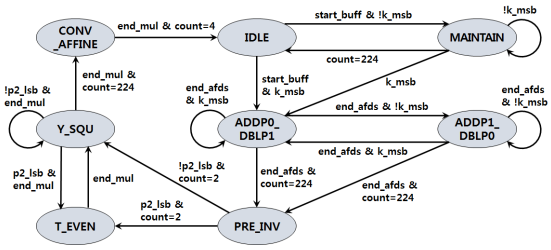


(a)

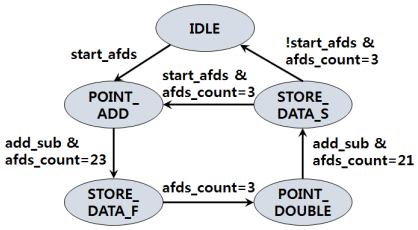


(b)

Fig. 1 Architecture and timing diagram of the ECC-P224 Processor (a) architecture (b) timing diagram



(a)



(b)

Fig. 2 Finite-state machines for ECC-P224 Processor (a) scalar point multiplication (b) point addition followed by point doubling

ECC-P224 프로세서는 그림 1-(b)와 같은 동작 타이밍으로 스칼라 곱셈 연산을 수행한다. 하나의 데이터 입력포트를 통해 224-비트의 소수 p, 정수 k, 그리고 affine 좌표계의 시작점 x 좌표, y 좌표를 입력받는다. 그리고 iSTART_ECC 신호와 함께 affine 좌표계의 시작점을 Jacobian 좌표로 변환하고 스칼라 곱셈을 시작한다. 스칼라 곱셈의 결과값은 다시 affine 좌표로 역변환되어 oEND_ECC 신호와 함께 2개의 224-비트 출력포트를 통해 출력된다.

Smul_Mem 블록은 시작점의 좌표, 소수 p, 정수 k, 그리고 스칼라 곱셈의 중간 결과값을 저장하기 위해 12 $\times$224-비트 dual-port RAM으로 구성된다. 또한, 224-비트 레지스터 2개와 메모리 제어블록으로 구성된다. shift_reg는 스칼라 곱셈 연산을 수행할 때 정수 k를 저장하며, end_afds 신호와 함께 1 비트씩 왼쪽으로 쉬프트한다. 그리고 소수체 상의 역원 연산을 수행할 때는 $p-2$의 값을 저장하며, inv_start 신호와 함께 1 비트씩 오른쪽으로 쉬프트한다. modular_reg는 스칼라 곱셈이 시작될 때 메모리로부터 소수 p를 저장하여, Alu_GFp224 블록에서 모듈러 연산을 수행하기 위해 필요한 소수 p에 대한 접근성을 용이하게 하였다.

제어블록은 그림 2와 같이 2개의 FSM으로 구현되었

```
Input : An integer k > 0 and a point P
Output : Q = kP

k = (k_{l-1} ... k_1 k_0)_2
P_0 = O, P_1 = P
for i = l − 1 to 0 do
        if k_i = 1 then
                P_0 ← P_0 + P_1
                P_1 ← 2P_1
        else
                P_1 ← P_1 + P_0
                P_0 ← 2P_0
        end
end for
return Q ← P_0
```

Fig. 3 Pseudo code of modified Montgomery ladder algorithm

으며, 그림 2-(a)는 수정형 Montgomery ladder 알고리듬을 적용한 스칼라 곱셈을 제어하는 부분, Fermat's little theorem을 이용하여 결과값 Z좌표의 역원을 구하는 부분, 그리고 Jacobian 좌표계의 결과값을 affine 좌표로 역변환하는 부분으로 구분된다. 본 논문에서는 스칼라 곱셈을 구현하기 위해 단순 전력분석과 같은 부채널 공격에 안전한 MML(Modified Montgomery Ladder) 알고리듬을 사용하였으며, 슈도 코드는 그림 3과 같다. MML 알고리듬은 정수 k의 최상위 비트에서 최하위 비트까지 값을 확인하며 매 루프마다 한번의 점 덧셈 연산과 점 두배 연산을 반복한다. Z 좌표의 역원을 구하기 위해서 Fermat's little theorem을 이용하였으며, 슈도 코드는 그림 4와 같다. 그림 4의 슈도 코드에서 최종 결과값은 $X = A^{-1} (\mathrm{mod} p) = A^{p-2} (\mathrm{mod} p)$이다. 마지막 상태인 CONV_AFFINE에서 Z^{-2}, Z^{-3}을 계산하여, 최종 결과값 $(x, y) = (X/Z^2, Y/Z^3)$을 출력한다.

```
Input : A, p two number
                of degree ≤ m − 1
Output : X = A^{-1} mod p

X = 1, Y = A, T = p − 2
for i = 0 to m − 1 do
        if T[i] = 1 then
                X ← X * Y (mod p)
        end if
        Y ← Y * Y (mod p)
end for
return X
```

Fig. 4 Pseudo code of Fermat's Little Theorem

그림 2-(b)는 그림 2-(a)의 ADDP0_DBLP1 상태 또는 ADDP1_DBLP0 상태에서 동작하는 FSM으로 점 덧셈 연산과 점 두배 연산을 순차적으로 제어한다. Jacobian 좌표계에서 점 덧셈 연산과 점 두배 연산을 수행할 때 메모리 사용을 최소화 하고 중간 결과값을 줄이기 위해 그림 5와 같이 점 연산을 수행한다[13]. NIST에서 정의하는 소수체 상의 타원곡선 방정식의 계수 a는 항상 -3의 값을 가지므로, 그림 5-(b)의 알고리듬을 사용함으로써 점 두배 연산에서 필요한 소수체 상의 곱셈 연산 횟수를 감소시킬 수 있다. 또한, 점 덧셈 또는 점 두배 연산에서 중간 결과값 $R_x (1 \le x \le 7)$를 저장하기 위한 메모리를 적절하게 공유하여 10×224-비트 메모리만을 사용하도록 설계하였다.

3.2. Alu_GFp224 블록 구현

Alu_GFp224 블록은 소수체 상의 곱셈기, 덧셈기, 그리고 덧셈/뺄셈기로 이루어져 있다. 실제 점 덧셈 연산과 점 두배 연산에서 제곱 연산을 필요로 하지만 하드웨어 자원 소모를 절감하기 위해 제곱 연산기를 곱셈 연산기로 대체하였다.

Mul_GFp224 곱셈기는 Interleaved modular 알고리듬을 수정한 그림 6과 같은 알고리듬을 사용하여 구현하였으며, 하드웨어 자원 소모가 큰 곱셈 연산을 사용

```
Input : X_1, Y_1, Z_1, X_2, Y_2, Z_2
Output : X_3, Y_3, Z_3
R_2 = X_1, R_3 = Y_1, R_4 = Z_1
R_5 = X_2, R_6 = Y_2, R_7 = Z_2

R_1 ← R_7^2
R_2 ← R_2 * R_1
R_3 ← R_3 * R_7
R_3 ← R_3 * R_1
R_1 ← R_4^2
R_5 ← R_5 * R_1
R_6 ← R_6 * R_4
R_6 ← R_6 * R_1
R_5 ← R_5 − R_2
R_7 ← R_4 * R_7
R_7 ← R_5 * R_7
R_6 ← R_6 − R_3
R_1 ← R_5^2
R_4 ← R_6^2
R_2 ← R_2 * R_1
R_5 ← R_5 * R_1
R_4 ← R_4 − R_5
R_4 ← R_4 − R_2
R_4 ← R_4 − R_2
R_2 ← R_2 − R_4
R_6 ← R_6 * R_2
R_1 ← R_3 * R_5
R_1 ← R_6 − R_1

X_3 = R_4, Y_3 = R_1, Z_3 = R_7
```

(a)

```
Input : X_1, Y_1, Z_1
Output : X_2, Y_2, Z_2
R_2 = X_1, R_3 = Y_1, R_4 = Z_1

R_5 ← R_3^2
R_5 ← R_5 + R_5
R_6 ← R_2 * R_5
R_6 ← R_6 + R_6
R_5 ← R_5 * R_5
R_5 ← R_5 + R_5
R_3 ← R_3 * R_4
R_3 ← R_3 + R_3
R_4 ← R_4 * R_4
R_2 ← R_2 + R_4
R_4 ← R_4 + R_4
R_4 ← R_2 − R_4
R_2 ← R_2 * R_4
R_4 ← R_2 + R_2
R_2 ← R_2 + R_4
R_4 ← R_2 * R_2
R_4 ← R_4 − R_6
R_4 ← R_4 − R_6
R_6 ← R_6 − R_4
R_2 ← R_2 * R_6
R_2 ← R_2 − R_5

X_2 = R_4, Y_2 = R_2, Z_2 = R_3
```

(b)

Fig. 5 Pseudo code for point operation over $GF(p)$ using Jacobian's coordinate (a) point addition (b) point doubling

하지 않고, 소수체 상의 덧셈/뺄셈 연산을 이용하여 곱셈 연산이 수행되도록 설계하였다. Interleaved modular 알고리듬의 경우 하드웨어 복잡도가 낮고 제어가 간단하지만, 두 번의 덧셈 연산이 모두 끝나고 모듈러 연산을 수행하기 때문에 중간 결과값이 피연산자보다 2-비트 더 커질 수 있다. 그림 6의 알고리듬을 사용하는 경우, 덧셈 연산과 동시에 모듈러 연산을 수행함으로써 중간 결과값의 비트가 피연산자와 동일하다. 두 데이터 A, B가 입력될 때 A의 최하위 비트 값에 따라서 C에 B 또는 0이 저장된다. 이후 m-1번의 루프 연산을 반복하게 되며, 덧셈 연산 후에 중간 결과값이 모듈러 p보다 클 경우 뺄셈 연산을 한다. 그림 6의 알고리듬과 이진체 상의 shift-and-add 곱셈 알고리듬[14]을 비교했을 때, 유한체 상의 덧셈/뺄셈 연산과 모듈러 연산을 제외한 레지스터와 제어가 매우 유사하다.

그림 7은 Alu_GFp224 블록의 하드웨어 구조를 나타낸다. 224-비트의 iDATA_A와 iDATA_B 포트를 통해 곱셈 또는 덧셈/뺄셈의 피연산자가 입력되고, iDATA_P 포트를 통해 모듈러 연산을 위한 소수 p가 입력된다. 곱셈 연산을 할 경우 두 데이터는 B_reg 레지스터와 Shift_reg 레지스터에 저장되며, iDATA_A의 최하위 비트 값에 따라 C_reg 레지스터에 0 또는 iDATA_B가 저장된다. 그리고 Add_GFp224 덧셈기를 통해 그림 6의 알고리듬에서 $B \leftarrow B+B$ 연산을 수행하고, Add/Sub_GFp224 덧셈기를 통해 $C \leftarrow C+B$ 연산을 수행한다.

```
Input : A, B two number
                of degree ≤ m − 1

Output : C = A * B mod p

if a_0 = 1 then
        C ← B
else
        C ← 0
end if
for i = 1 to m − 1 do
        B ← B + B
        if B ≥ p then
                B ← B − p
        end if
        if a_i = 1 then
                C ← C + B
                if C ≥ p then
                        C ← C − p
                end if
        end if
end for
return C
```

Fig. 6 Pseudo code of shift-and-add algorithm over $GF(p)$

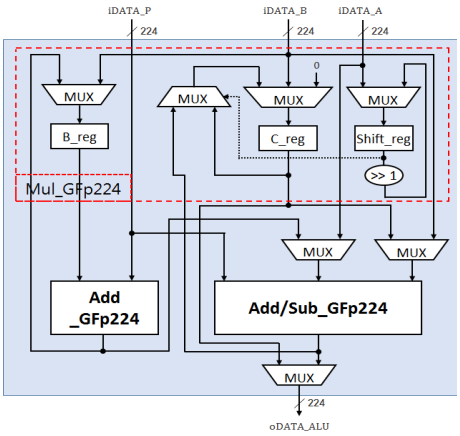


Fig. 7 Architecture of the Alu_GFp224

위와 같이 2번의 덧셈 연산을 Add_GFp224와 Add/Sub_GFp224를 이용하여 한 클록 사이클만에 처리함으로써 소수체 상의 곱셈 연산에 소요되는 클록 사이클을 줄일 수 있다. 설계된 곱셈기는 데이터를 입력받아 224 클록 사이클 후에 end_mul 신호와 함께 C_reg 레지스터로부터 연산 결과값을 출력한다.

Add_GFp224는 $B \leftarrow B+B$와 같이 소수체 상의 두 배 연산만을 수행함으로써, CSelA와 쉬프트 연산 등으로 비교적 간단하게 하드웨어를 구현하였다. 반면, Add/Sub_GFp224은 Alu_GFp224가 곱셈 연산을 수행할 경우에는 $C \leftarrow C+B$ 연산을 수행하고, Alu_GFp224가 소수체 상의 덧셈 또는 뺄셈 연산을 수행할 때는 포트 iDATA_A와 iDATA_B로부터 덧셈/뺄셈의 피연산자를 입력받아 한 클록만에 연산을 수행하도록 설계하였다. 그래서 224-비트 CSelA 2개, 224-비트 CSavA 1개, 쉬프트 연산자 등 Add_GFp224와 비교하면 약간 더 복잡한 하드웨어 구조를 가진다. 하지만 본 논문에서는 그림 6의 슈도 코드에서 $if\, B \geq p\ then$ 또는 $if\, C \geq p\ then$ 연산에 필요한 비교기를 사용하지 않고, 단순 MUXs만을 추가하여 비교기를 대체함으로써 하드웨어 자원 소모를 절감하였다.

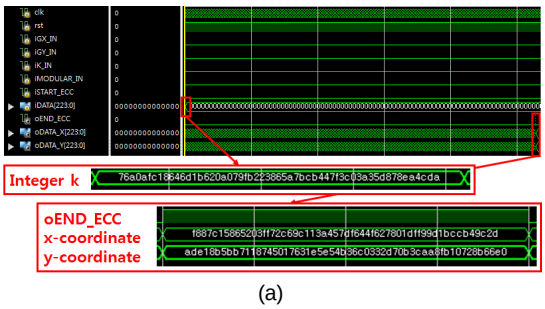
Ⅳ. 기능검증 및 FPGA 구현

설계된 ECC-P224 프로세서는 RTL 시뮬레이션에 의한 기능검증과 FPGA 구현에 의한 하드웨어 동작을 확

인하였다. 그림 8은 Xilinx ISim을 이용한 ECC-P224 프로세서의 시뮬레이션 결과값과 문헌 [5]의 참조 구현 값을 보여준다. NIST FIPS 186-2 [4]에 정의되어 있는 Curve P-224 타원곡선 파라미터를 사용하였으며, 224-비트 정수 k "76a0afc1 8646d1b6 20a079fb 223865a7 bcb447f3 c03a35d8 78ea4cda"를 생성점 $G(x, y)$에 스칼라 곱셈하였다. 그림 8-(a)에서 스칼라 곱셈 연산이 완료된 두 좌표값이 oEND_ECC 신호와 함께 동시에 출력되는 것을 확인할 수 있다. 스칼라 곱셈이 완료된 x 좌표 "f887c158 65203ff7 2c69c113 a457df64 4f627801 dff99d1b ccb49c2d"와 y좌표 "ade18b5b b7118745 017631e5 e54b36c0 332d70b3 caa8fb10 728b66e0"는 그림 8-(b)의 참조 구현 값과 정확히 일치함을 확인할 수 있다.

기능검증이 완료된 ECC-P224 프로세서는 FPGA 구현을 통해 하드웨어 동작을 검증하였다. FPGA 검증 시스템은 그림 9와 같이 FPGA 보드, UART 인터페이스, 구동 소프트웨어로 구성된다. Virtex5 XC5VSX95T FPGA 디바이스가 사용되었으며, PC와 FPGA 사이의 데이터 송수신의 RS232C를 통해 이루어진다.

그림 10은 ECDH(Elliptic Curve Diffie-Hellman) 키 교환 프로토콜을 사용한 FPGA 검증 결과를 보이고 있다. Alice와 Bob은 동일한 생성점의 두 좌표 G_x와 G_y를 공유하고, 1 이상 생성점의 위수 미만에 속하는 랜덤한 정수 k_a와 k_b를 생성한다. k_a는 Alice의 비밀키로 사



(a)

```
k = 76A0AFC1 8646D1B6 20A079FB 223865A7 BCB447F3
    C03A35D8 78EA4CDA
단계 2 kG=(x1, y1)의 계산 :
x1 = F887C158 65203FF7 2C69C113 A457DF64 4F627801
     DFF99D1B CCB49C2D
y1 = ADE18B5B B7118745 017631E5 E54B36C0 332D70B3
     CAA8FB10 728B66E0
```

(b)

Fig. 8 Functional simulation results of ECC-P224 processor (a) simulation results (b) reference data

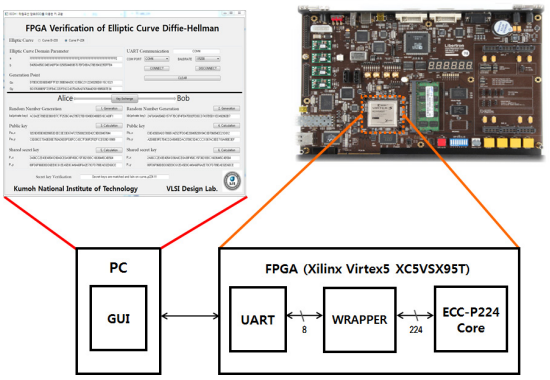


Fig. 9 FPGA verification setup

용되고, k_b는 Bob의 비밀키로 사용된다. 버튼 3을 클릭하여 데이터 전송을 실행시키면, 생성된 정수 k_a , 그리고 G_x, G_y가 UART 통신을 통해 FPGA에 구현된 ECC-P224 프로세서로 전송된다. ECC-P224 프로세서는 키 값과 생성점의 좌표값을 이용하여 스칼라 곱셈 $P_a = k_a * G$ 연산을 하고, 결과값은 Alice의 공개키(public key)에 표시된다. 버튼 4를 클릭하면, ECC-P224 프로세서는 $P_b = k_b * G$ 연산을 수행하고, 그 결과가 Bob의 공개키에 표시된다.

버튼 5와 6을 순차적으로 클릭하면, Alice와 Bob의 공개키 교환이 이루어진다. Alice는 ECC-P224 프로세서를 이용하여 본인의 비밀키 k_a와 Bob의 공개키 P_b를 곱하여 $P_{alice} = k_a * P_b$를 계산한다. Bob은 비밀키 k_b와 Alice의 공개키 P_a를 곱하여 $P_{bob} = k_b * P_a$를 계산한다. 구동 소프트웨어는 최종 결과값 P_{alice}와 P_{bob}의 동일성을 판단하며, 두 결과값이 타원곡선 P-224에 존재하는지 검증한다. 이러한 일련의 과정을 통하여 설계된 ECC-P224 프로세서가 올바르게 동작함을 확인할 수 있다.

설계된 224-비트 타원곡선 암호 프로세서는 타원곡선 상의 스칼라 곱셈 연산에 1,326,985 클록 사이클이 소요되며, 0.18 ㎛ CMOS 표준 셀 라이브러리로 합성한 결과 10 MHz의 동작 주파수에서 2.7-Kbit RAM과 27,739 GE로 구현되었다. 최대 동작 주파수 71 MHz에서 2.7-Kbit RAM과 33,505 GE로 구현되었으며, 스칼라 곱셈 연산에 18.7 msec가 소요된다.

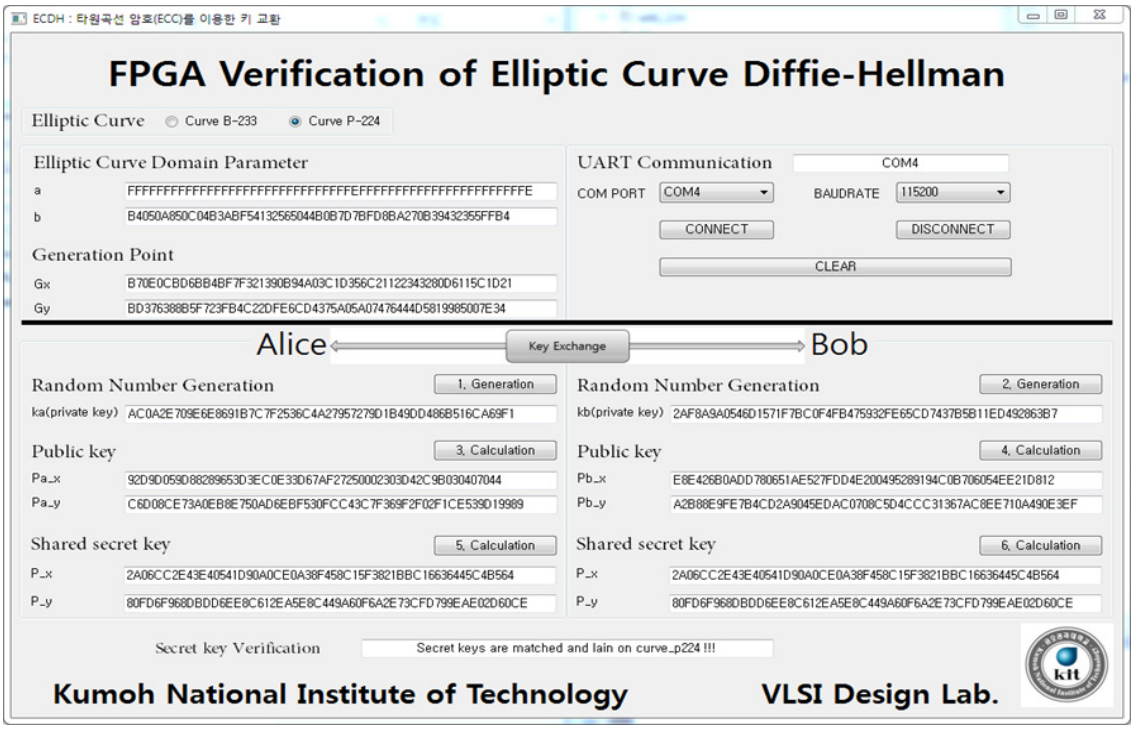


Fig. 10 FPGA verification results of ECC-P224 processor using ECDH key exchange protocol

Ⅴ. 결 론

NIST FIPS 186-2 표준으로 정의되어 있는 타원곡선 P-224를 지원하는 타원곡선 암호 프로세서를 설계하고, FPGA 구현을 통한 하드웨어 동작을 검증하였다. Modified Montgomery ladder 알고리듬을 이용하여 단순 전력분석에 안전하도록 설계하였다. 소수체 상의 곱셈 연산은 쉬프트 연산과 소수체 상의 덧셈기만을 사용하여 구현하였으며, 덧셈기는 캐리보존 및 캐리 선택 가산기를 사용하여 캐리 전파에 의한 지연을 최소화하였다. ECC-P224 프로세서는 0.18 ㎛ CMOS 공정에서 10 MHz의 동작 주파수로 합성한 결과 2.7-Kbit RAM과 27,739 GE로 구현되었다.

최대 71 MHz의 클록 주파수로 동작 가능하며, 스칼라 곱셈 연산에 18.7 msec가 소요된다. ECC-P224 프로세서는 공개키 암호의 복잡한 연산을 적은 하드웨어로 처리하므로, 제한된 하드웨어 자원을 갖는 분야에 활용이 가능하다.

ACKNOWLEDGMENTS

- This work was supported by Industrial Core Technology Development Program (10049009, Development of Main IPs for IoT and Image-Based Security Low-Power SoC) funded by the Ministry of Trade, Industry & Energy.
- Authors are thankful to IDEC for EDA software support.

REFERENCES

[1] NOKIA, “LTE-M-Optimizing LTE for the Internet of Things,” White Paper, 2015.

[2] NIST Std. FIPS-197, *Advanced Encryption Standard*, National Institute of Standard and Technology (NIST), November, 2001.

[3] R. Rivest, A. Shamir and L. Adleman, “A method for

obtaining Digital Signatures and Public-Key Crypto-systems," *Communications of Association for Computing Machinery (ACM)*, vol. 21, no. 2, pp. 120-126, Feb. 1978.

[4] NIST Std. FIPS PUB 186-2, *Digital Signature Standard (DSS)*, National Institute of Standard and Technology (NIST), Jan. 2000.

[5] TTA Std. TTAK.KO-12.0015/R1, *Digital Signature Mechanism with Appendix (Part 3) Korean Certificate-based Digital Signature Algorithm using Elliptic Curves*, Telecommunications Technology Association (TTA), Dec. 2012.

[6] T. Akishita and T. Takagi, "Zero-value point attacks on elliptic curve cryptosystem," *International Conference on Information Security, Springer Berlin Heidelberg*, pp. 218-233, 2003.

[7] D. Amiet, A. Curiger, and P. Zbinden, "Flexible FPGA-Based Architectures for Curve Point Multiplication over GF(p)," *IEEE Euromicro Conference on Digital System Design*, pp. 107-114, 2016.

[8] H. Alrimeih and D. Rakhmatov, "Fast and flexible hardware support for ECC over multiple standard prime fields," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2661-2674, Dec. 2014.

[9] J. Vliegen et al, "A compact FPGA-based architecture for elliptic curve cryptography over prime fields," *IEEE International Conference on Application-specific Systems Architectures and Processors (ASAP)*, pp. 313-316, 2010.

[10] J. Guajardo et al, "Efficient hardware implementation of finite fields with applications to cryptography," in *Acta Applicandae Mathematicae*, vol. 93, pp. 75-118, 2006.

[11] M.S. Hossain and Y. Kong, "High-Performance FPGA Implementation of Modular Inversion over F_256 for Elliptic Curve Cryptography," *2015 IEEE International Conference on Data Science and Data Intensive Systems (DSDIS)*, pp. 169-174, 2015.

[12] J. Bosmans et al, "A tiny coprocessor for elliptic curve cryptography over the 256-bit NIST prime field," *IEEE 2016 29th International Conference on VLSI Design, 2016 15th International Conference on Embedded Systems*, pp. 523-528, 2016.

[13] T. Izu, B. Moller, and T. Takagi, "Improved elliptic curve multiplication methods resistant against side channel attacks," *International Conference on Cryptology in India*, Springer Berlin Heidelberg, pp. 296-313, 2002.

[14] M. Amara and A. Siad, "Hardware implementation of Elliptic Curve Point Multiplication over GF(2^m) for ECC protocols," *International Journal for Information Security Research (IJISR)*, vol. 2, no. 1, pp. 106-112, March. 2012.

박병관(Byung-Gwan Park)

2016년 2월 금오공과대학교 전자공학부(공학사)
※관심분야 : 통신 및 신호처리용 반도체 IP 설계, 정보보호용 반도체 IP 설계

신경욱(Kyung-Wook Shin)

1984년 2월 한국항공대학교 전자공학과(공학사)
1986년 2월 연세대학교대학원 전자공학과(공학석사)
1990년 8월 연세대학교대학원(공학박사)
1990년 9월~1991년 6월 한국전자통신연구소 반도체연구단(선임연구원)
1991년 7월~현재 금오공과대학교 전자공학부(교수)
1995년 8월~1996년 7월 University of Illinois at Urbana-Champaign(방문교수)
2003년 1월~2004년 1월 University of California at San Diego(방문교수)
2013년 2월~2014년 2월 Georgia Institute of Technology(방문교수)
※관심분야 : 통신 및 신호처리용 SoC 설계, 정보보호 SoC 설계, 반도체 IP 설계