JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 21, No. 7 : 1267~1275 Jul. 2017

233-비트 이진체 타원곡선을 지원하는 암호 프로세서의 저면적 구현

박병관 · 신경욱*

A small-area implementation of cryptographic processor for 233-bit elliptic curves over binary 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 이진체(binary field) 상의 233-비트 타원곡선을 지원하는 타원곡선 암호(elliptic curve cryptography; ECC) 프로세서를 설계하였다. 타원곡선 암호 시스템의 핵심 연산인 스칼라 점 곱셈을 수정형 Montgomery ladder 알고리듬을 이용하여 구현함으로써 단순 전력분석에 강인하도록 하였다. 점 덧셈과 점 두배 연산은 아핀(affine) 좌표계를 기반으로 유한체 $GF(2^{233})$ 상의 곱셈, 제곱, 나눗셈으로 구현하였으며, shift-and-add 방식의 곱셈기와 확장 유클리드 알고리듬을 이용한 나눗셈기를 적용함으로써 저면적으로 구현하였다. 설계된 ECC 프로세서를 Virtex5 FPGA로 구현하여 정상 동작함을 확인하였다. 0.18㎛ 공정의 CMOS 셀 라이브러리로 합성한 결과 49,271 GE로 구현되었고, 최대 345 MHz의 동작 주파수를 갖는다. 스칼라 점 곱셈에 490,699 클록 사이클이 소요되며, 최대 동작 주파수에서 1.4 msec의 시간이 소요된다.

ABSTRACT

This paper describes a design of cryptographic processor supporting 233-bit elliptic curves over binary field defined by NIST. Scalar point multiplication that is core arithmetic in elliptic curve cryptography(ECC) was implemented by adopting modified Montgomery ladder algorithm, making it robust against simple power analysis attack. Point addition and point doubling operations on elliptic curve were implemented by finite field multiplication, squaring, and division operations over $GF(2^{233})$, which is based on affine coordinates. Finite field multiplier and divider were implemented by applying shift-and-add algorithm and extended Euclidean algorithm, respectively, resulting in reduced gate counts. The ECC processor was verified by FPGA implementation using Virtex5 device. The ECC processor synthesized using a 0.18 um CMOS cell library occupies 49,271 gate equivalents (GEs), and the estimated maximum clock frequency is 345 MHz. One scalar point multiplication takes 490,699 clock cycles, and the computation time is 1.4 msec at the maximum clock frequency.

키워드 : 타원곡선 암호, 공개키 암호, 수정형 몽고메리 래더 알고리듬, ECDH 키 교환 프로토콜

Key word : ECC, public key cryptography, modified Montgomery ladder algorithm, ECDH key exchange protocol

Received 15 February 2017, **Revised** 21 February 2017, **Accepted** 09 March 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.2017.21.7.1267 print ISSN: 2234-4772 online ISSN: 2288-4165

ⓒThis is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(http://creativecommons.org/li-censes/ by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Copyright Ⓒ The Korea Institute of Information and Communication Engineering.

Ⅰ. 서 론

네트워크 기술이 발달하면서 PC 기반 정보교류 뿐만 아니라 스마트홈, 헬스케어 등 다양한 사물인터넷(Internet of Things; IoT)을 통한 정보교류와 전자금융 및 전자상거래가 폭발적으로 늘어나고 있는 추세이다. 특히, IoT 서비스는 가전, 교통, 의료, 환경, 건설, 에너지 분야로 분류되어, 분야별로 다양한 보안 강도를 가진 정보보안을 필요로 한다. 정보보안이란 데이터의 기밀성(confidentiality), 무결성(integrity), 그리고 인증(authentication) 등 다양한 측면에서 정보를 보호하는 것으로, 정보보안을 위한 암호 기술은 크게 대칭키 암호시스템(symmetric key cryptography)과 공개키 암호시스템(public key cryptography)으로 나누어진다.

대칭키 암호시스템이란 두 사용자가 동일한 비밀키를 공유하고 있는 상태에서 데이터를 암호/복호화하는 시스템으로, 키의 안전한 보관과 분배가 암호 시스템의 안전성과 직결된다. 대표적인 대칭키 암호 알고리듬은 미국의 연방 표준 알고리듬 AES(Advanced Encryption Standard) [1]와 국내 표준 알고리듬 ARIA(Academy, Research Institute, Agency) [2] 등이 있다. 대칭키 암호 알고리듬은 구현하기가 비교적 쉽고, 대용량의 데이터를 암호화 또는 복호화 할 수 있다. 하지만 정보를 주고받는 두 사용자는 사전에 안전하게 키를 공유하고 있어야 한다는 단점을 가지고 있다.

최근 정보보안 응용분야가 확대됨에 따라 대칭키 암호와 상호 보완적인 역할을 하며 송수신자 사이의 상호인증, 키 교환, 무결성 검증 등을 위한 공개키 암호시스템의 중요성이 더욱 증가하고 있다. 대표적인 공개키 암호 알고리듬은 RSA(Rivest, Shamir, and Adleman) [3]와 타원곡선 암호(Elliptic Curve Cryptography; ECC) [4] 알고리듬 등이 있다. RSA는 큰 수의 인수분해가 어렵다는 점에 안전성을 두고 있으며, ECC는 타원곡선군의 이산대수 문제에 안정성을 두고 있다.

타원곡선 암호는 비트 당 안전도가 다른 공개키 암호보다 효율적이라는 것이 알려지면서 ANSI(American National Standards Institute), WAP(Wireless Application Protocol) 등에서 표준 암호로 채택되었으며, 한국정보통신기술협회는 ECC를 이용한 인증서 기반 전자서명 알고리듬(EC-KCDSA)을 정보통신단체표준으로 제정하였다[5]. 특히, 160-비트의 키 길이를 지원하는 ECC는 1024-비트의 키 길이를 지원하는 RSA와 동일한 안정성을 제공하며[6], 메모리나 프로그램 코드의 크기가 제한된 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 B-233 타원곡선을 지원하는 ECC 프로세서를 설계하고, FPGA 구현을 통해 하드웨어 동작을 확인하였다. Ⅱ장에서는 타원곡선 암호 알고리듬에 대해 설명하고, Ⅲ장에서는 ECC 프로세서 설계에 대해 설명한다. 설계된 회로의 기능검증 및 FPGA 구현 결과를 Ⅳ장에 기술하고, Ⅴ장에서 결론을 맺는다.

Ⅱ. 타원곡선 암호 알고리듬

NIST FIPS 186-2에 정의되어 있는 타원곡선은 표 1과 같다 [4]. 타원곡선은 유한체(finite field) 연산이 구현되는 체에 따라 소수체(prime field)와 이진체(binary field)로 구분된다. 소수체에서 5가지의 타원곡선이 정의되어 있으며, 이진체에서 10가지의 타원곡선이 정의되어 있다. 이진체의 경우, 식 (1)과 같이 타원곡선을 정의하는 식에서 계수 a와 b의 값에 따라 Koblitz 곡선과 Pseudo-random 곡선으로 구분된다. Koblitz 곡선은 Frobenius 사상을 가지고 있어 일반적인 타원곡선에 비해 빠른 구현이 가능하나, 여인자(cofactor)가 2 또는 4로 비교적 높은 값을 갖는다. 여인자는 타원곡선의 위수(order)를 기본점(base point)의 위수로 나눈 값으로, 암호학적 응용에서 타원곡선의 안정성을 위해 가능한 작도록 하는 것이 바람직하다. Pseudo-random 곡선은 계수값과 상관없이 여인자를 2로 가져 Koblitz 곡선에 비해 비교적 높은 안정성을 보장한다.

$$y^2 + xy = x^3 + ax^2 + b \quad (1)$$
$$where \quad a, b \in GF(2^m) \ and \ \ b \neq 0$$

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

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

역연산을 통하여 k를 알아내기가 어렵다는 것을 의미한다. 위 연산을 타원곡선 상의 스칼라 곱셈이라고 하며, 정수 k는 사용자의 비밀키이고 결과값 Q 점은 공개키로 사용된다.

스칼라 곱셈은 점 덧셈 연산과 점 두배 연산으로 계산될 수 있다. 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, 곱셈, 나눗셈, 제곱 연산으로 계산된다. 즉, 스칼라 곱셈과 유한체 연산기의 성능이 ECC 프로세서의 고속화와 최적화에 크게 영향을 미치는 요인이 된다.

대표적인 스칼라 곱셈 알고리듬은 binary method [7], Non-Adjacent Form(NAF) [8], Montgomery ladder [9] 알고리듬이 있다. Binary method은 가장 기본적인 알고리듬으로 매 루프마다 점 두배 연산을 하고, 주어진 키의 hamming weight만큼 점 덧셈 연산을 수행한다. 이때 NAF 알고리듬을 사용하는 경우 키의 hamming weight를 감소시켜 binary method보다 연산량을 줄일 수 있다. 하지만 NAF 변환을 위한 추가적인 하드웨어가 소요되며, 위의 두 알고리듬은 키의 hamming weight에 따라 연산량이 다르므로, 공격자는 단순 전력분석과 같은 부채널 공격을 통해 키에 대한 정보를 얻을 수 있다. Montgomery ladder 알고리듬은 키 값에 상관없이 동일한 횟수의 점 덧셈 연산과 점 두배 연산을 수행하므로 부채널 공격에 보다 안전하다는 장점이 있다.

Table. 2 Point addition and point doubling operations over binary field $GF(2^m)$

Point addition (C = A + B)	Point doubling (C = 2A)
$\lambda = (y_1 + y_0)/(x_1 + x_0)$	$\lambda = x_0 + y_0/x_0$
$x_2 = \lambda^2 + \lambda + x_0 + x_1 + a$	$x_2 = \lambda^2 + \lambda + a$
$y_2 = \lambda(x_0 + x_2) + x_2 + y_0$	$y_2 = x_0^2 + (\lambda + 1)x_2$

$GF(2^m)$ 상의 덧셈 연산은 비트 단위의 XOR 게이트로 구현가능 하나, 곱셈 연산과 나눗셈 연산은 매우 다양한 알고리듬이 존재한다. Montgomery 곱셈기[10]는 부분곱의 갯수를 감소시켜 곱셈 연산에 소요되는 클록 사이클을 감소시킬 수 있다. 반면에 shift-and-add 곱셈기[7]는 키 길이만큼 클록 사이클이 소요되지만, 단순 쉬프트 연산과 XOR 연산만 사용되고, 알고리듬이 간단하여 연산을 제어하기 위한 하드웨어가 단순화된다.

유한체 상의 나눗셈 연산은 곱셈의 역원을 구하여 곱하는 것으로 연산이 가능하다. Fermat's little theorem [10]을 사용하는 경우 $a^{-1} (\mathrm{mod} p) = a^{p-2} (\mathrm{mod} p)$와 같은 특성을 이용하여 제수의 곱셈에 대한 역원을 구하고, 다시 곱셈기를 이용하여 피제수와 곱하게 된다. 확장 유클리드(Extended Euclidean) 알고리듬[11]을 사용하는 경우에는 초기값에 제수와 피제수를 대입하여 곱셈 연산을 따로 하지 않고 바로 나눗셈 결과값을 얻을 수 있다. 그러나 연산 과정에서 중간 결과값을 저장하기 위한 레지스터가 추가로 필요하다.

Ⅲ. 타원곡선 암호 프로세서 설계

3.1. 전체 구조 및 스칼라 곱셈 구현

FIPS 186-2의 B-233 타원곡선을 지원하는 ECC 프로세서를 설계하였다. 전체 구조는 그림 1-(a)와 같으며, 스칼라 곱셈의 중간 결과값 저장과 덧셈 연산을 수행하는 Reg_Add 블록, 유한체 상의 곱셈, 제곱, 나눗셈 연산을 수행하는 Alu_GF233 블록, 그리고 FSM(Finite State Machine)을 이용하여 스칼라 곱셈 연산을 제어하는 제어블록으로 구성된다.

ECC 프로세서는 그림 1-(b)와 같은 동작 타이밍도로 스칼라 곱셈 연산을 수행한다. 하나의 데이터 입력포트를 통해 233-비트의 정수 k와 시작점의 x 좌표, y 좌표를 입력받는다. 그리고 iSTART_ECC 신호와 함께 스칼라 곱셈을 시작하고, 스칼라 곱셈의 결과값 두 좌표는 2

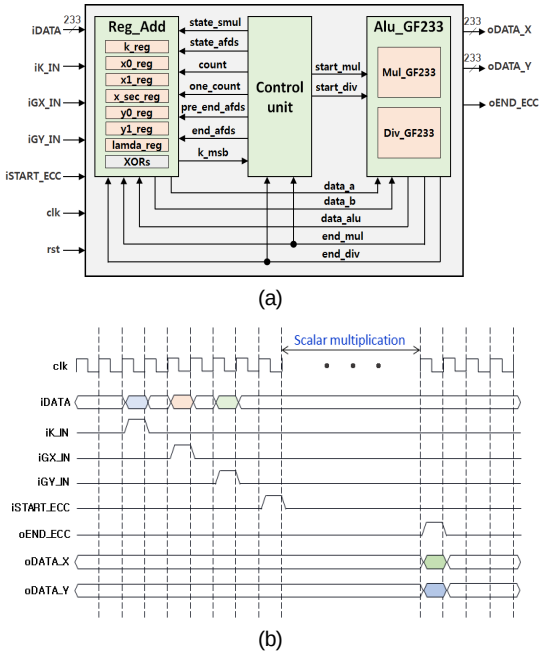


Fig. 1 Architecture and timing diagram of the ECC Processor (a) architecture (b) timing diagram

개의 233-비트 출력포트를 통해 동시에 출력된다.

Reg_Add 블록은 시작점의 두 좌표, 정수 k, 그리고 스칼라 곱셈의 중간 결과값을 저장하는 레지스터를 포함하여 총 7개의 233-비트 레지스터로 구성된다. 또한, 유한체 상의 덧셈 연산을 위한 XOR 게이트를 포함한다. 본 논문에서는 스칼라 곱셈을 구현하기 위해 단순 전력분석과 같은 부채널 공격에 안전한 modified Montgomery ladder(MML) 알고리듬을 사용하였으며, 정수 k의 최상위 비트가 1의 값을 갖는 경우만을 고려

$$
\begin{aligned}
&Input: An\ integer\ k>0\ and\ a\ point\ P\\
&Output: Q=kP\\
&\\
&k=(k_{l-1}\dots k_1k_0)_2\\
&P_0=O, P_1=P\\
&for\ i=l-1\ to\ 0\ do\\
&\quad if\ k_i=1\ then\\
&\qquad P_0=P_0+P_1\\
&\qquad P_1=2P_1\\
&\quad else\\
&\qquad P_1=P_1+P_0\\
&\qquad P_0=2P_0\\
&\quad end\\
&end\ for\\
&return\ Q=P_0
\end{aligned}
$$

Fig. 2 Pseudo code of modified Montgomery ladder algorithm

한 기존 Montgomery ladder 알고리듬의 제약조건을 해결하였다. MML 알고리듬의 슈도 코드는 그림 2와 같으며, 정수 k의 최상위 비트(MSB)에서 최하위 비트(LSB)까지 값을 확인하며 연산을 반복한다. k_i의 값이 1일 때는 점 덧셈 연산의 결과값 P_0의 x 좌표는 레지스터 x0_reg, y 좌표는 레지스터 y0_reg에 저장하고, 점 덧셈 연산이 끝난 후 점 두배 연산을 하여 결과값 P_1의 x 좌표는 레지스터 x1_reg, y 좌표는 레지스터 y1_reg에 저장한다. 반대로 k_i의 값이 0일 때는 점 덧셈 연산의 결과값 P_1의 좌표를 레지스터 x1_reg와 y1_reg에 저장하고, 점 두배 연산의 결과값 P_0의 좌표를 레지스터 x0_reg, y0_reg에 저장한다.

MML 알고리듬을 적용한 스칼라 곱셈 연산을 위해서 그림 3과 같이 두 개의 FSM으로 구성되는 제어블록을 설계하였다. 그림 3-(a)에서 ADDP0_DBLP1는 k_i의 값이 1 일 때 상태이며, ADDP1_DBLP0은 k_i의 값이 0일 때 상태이다. 그리고 MAINTAIN 상태는 정수 k의 값을 MSB부터 확인하며 처음으로 1의 값이 나올 때 까지 어떠한 연산도 하지 않는 상태이다. 두 상태 ADDP0_DBLP1 또는 ADDP1_DBLP0으로 천이되면 start_afds 신호를 출력하여 3-(b)의 FSM이 동작하도록 한다. 그림 3-(b)의 FSM은 점 덧셈과 점 두배 연산을 제어한다. 상태 LAMDA0에서는 점 덧셈 연산의 λ 값을 구하고, SQU_LAMDA0에서는 Alu_GF233 블록에서 λ^2 값을 구하고, Reg_Add 블록에서 XOR 연산을 이용하여 점 덧셈 연산의 x_2 값을 계산한다. FINAL_AF 상태에서는 Alu_GF233 블록에서 $\lambda(x_0+x_2)$ 값을 구하고, Reg_Add 블록에서 덧셈 연산을 하여 y_2 값을 계산한다. LAMDA1 상태부터는 점 두배 연산을 시작하며, Alu_GF233 블록에서 y_0/x_0 값을 구하고, Reg_Add 블록에서 덧셈 연산을 하여 점 두배 연산의 λ 값을 계산한다. SQU_LAMDA1의 경우에는 λ^2 값을 구하여 x_2 값을 계산한다. CHANGE_LAMDA1 상태에서는 λ 값이 저장되어 있던 레지스터에 $(\lambda+1)x_2$ 값을 구하여 저장한다. 그리고 FINAL_DS 상태에서 y_2 값을 구하게 되며, 점 덧셈 연산과 점 두배 연산이 끝나게 된다. 이후 start_afds 신호에 따라서 IDLE 상태로 천이하여 다음 신호가 입력될 때까지 대기하거나, LAMDA0 상태로 천이하여 다시 연산을 반복하게 된다.

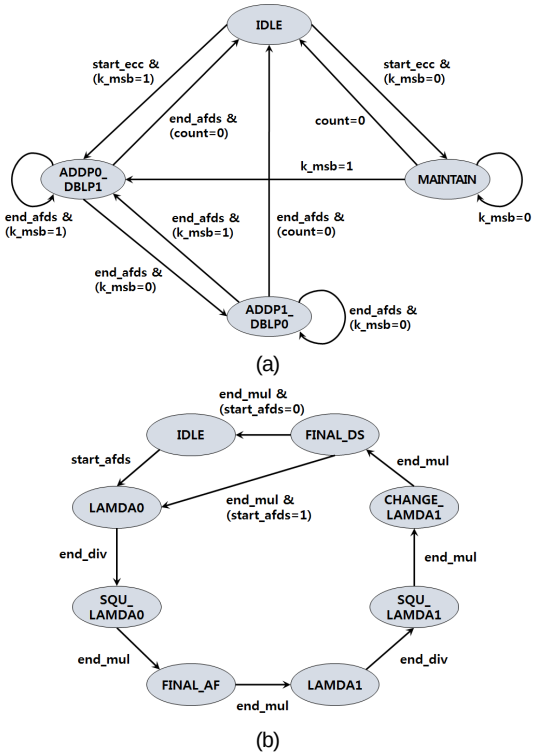


Fig. 3 Finite-state machines for scalar multiplication based on modified Montgomery ladder algorithm (a) MML (b) point addition followed by point doubling

3.2. Alu_GF233 블록 구현

Alu_GF233 블록은 $GF(2^{233})$ 상의 곱셈 연산과 나눗셈 연산을 수행한다. 실제 점 덧셈 연산과 점 두배 연산에서 제곱 연산을 필요로 하지만, 하드웨어 자원을 절감하기 위해 제곱 연산기를 곱셈 연산기로 대체하였다. $GF(2^{233})$ 상의 유한체 연산은 기약 다항식 $f(x)$를 모듈러로 하는 다항식 연산을 하며, B-233 타원곡선 파라미터에 정의되어 있는 $f(x)=x^{233}+x^{74}+1$을 기약 다항식을 사용하였다. 본 설계에서는 하드웨어 자원 소모가 큰 곱셈기와 나눗셈 연산자를 사용하지 않고, 단순 쉬프트 연산과 XOR 연산만을 사용하여 곱셈기와 나눗셈기를 저면적으로 구현하였다.

3.2.1. Mul_GF233 곱셈기 구현

Mul_GF233 곱셈기는 shift-and-add 알고리듬을 사용하여 구현하였으며, 키 길이만큼의 클록 사이클을 소요하지만, 단순 쉬프트 연산과 XOR 연산만을 사용하고 비교적 제어가 간단하여 저면적 하드웨어 구현에 적합하다. shift-and-add 알고리듬의 슈도 코드는 그림 4와 같으며, 두 데이터 $A(x), B(x)$가 입력될 때 $A(x)$의 최하위 비트 값에 따라서 $C(x)$에 $B(x)$ 또는 0이 저장된다. 이후 m-1 클록 사이클 동안 $B(x)$ 값을 왼쪽으로 1 비트씩 쉬프트하며, $B(x)$ 값의 최상위 비트가 1일 때는 쉬프트된 데이터에 기약 다항식을 XOR 연산하여 모듈러 곱셈 연산이 수행된다.

그림 5는 shift-and-add 알고리듬을 사용한 곱셈기의 하드웨어 구조를 나타낸다. 곱셈기는 233-비트 레지스터 3개와 XOR 게이트 등으로 구성된다. 두 입력 데이터는 레지스터 Shift_reg와 B_reg에 각각 저장되며, iDATA_A의 최하위 비트 값에 따라 C_reg에는 iDATA_B 또는 0이 저장된다. 그리고 Shift_reg에 저장된 데이터는 클록 사이클마다 1비트씩 오른쪽으로 순환 이동을 하여 Shift_reg[1]의 값이 1 일 때 $C=C+B$ 연산을 한다. AND 게이트는 B_reg에 저장된 데이터의 최상위 비트가 1 일 때 기약 다항식과 모듈러 연산을 수행한다. 설계된 곱셈기는 데이터를 입력받아 233 클록 사이클 후에 end_mul 신호와 함께 C_reg 레지스터로부터 연산 결과값을 출력하도록 설계하였다.

3.2.2. Div_GF233 나눗셈기 구현

유한체 상에서 나눗셈은 곱셈의 역원을 구하여 곱하는 것으로 계산된다. 본 설계에서는 확장 유클리드 알고리듬을 이용하여 역원 연산과 곱셈 연산을 동시에 처리하도록 하였으며, 2*m 클록 사이클만에 나눗셈 연산이 완료되도록 하였다. 확장 유클리드 알고리듬의 슈도

```
Input : A(x), B(x) two binary polynomial
                        of degree ≤ m − 1
Output : C(x) = A(x) * B(x) mod F(x)

if a_0 = 1 then
        C = B
else
        C = 0
end if
for i = 1 to m − 1 do
        B = B · x mod F(x)
        if a_i = 1 then
                C = C + B
        end if
end for
return C
```

Fig. 4 Pseudo code of shift-and-add multiplication algorithm

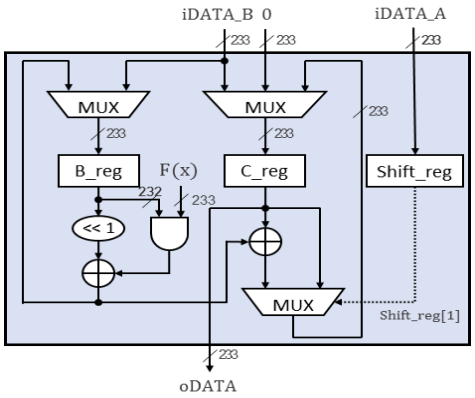


Fig. 5 Finite field multiplier

코드는 그림 6과 같으며, 두 데이터 $A(x), B(x)$가 입력되면 제수 $B(x)$는 R에 저장되고, 피제수 $A(x)$는 U에 저장된다. 이후 2*m 클록 사이클 동안 연산이 반복되고, 결과값 $C(x) = A(x)/B(x) mod F(x)$는 V로부터 출력된다. 이때 R과 S는 234-비트 데이터를 저장하고, U, V, T는 233-비트 데이터를 저장한다. 확장 유클리드 알고리듬의 연산 과정을 간단하게 정리하면 표 3과 같다. 나눗셈기의 연산은 제어신호 state, r_m, count의 값에 의해서 구분되며, r_m은 R에 저장된 데이터를 왼쪽으로 1비트 쉬프트 한 후의 최상위 비트 값을 의미

```
Input : A(x), B(x) two binary polynomial of degree ≤ m − 1
Output : C(x) = A(x)/B(x) mod F(x)

R = B(x); S = F(x); U = A(x); V = T = 0; state = 0; count = 0;
for i = 1 to 2m do
        R = x · R; T = x · T mod F(x)
        if state = 0 then
                count = count + 1
                if r_m = 1 then (r_m : the coefficient of x^m of R)
                        R ↔ S
                        R = R + S
                        T = U
                        state = 1
                end if
        else
                count = count − 1
                if r_m = 1 then
                        R = R + S
                        T = T + U
                end if
                if count = 0 then
                V = V + T
                U ↔ V
                state = 0
                end if
        end
end for
return V = C(x) = A(x)/B(x) mod F(x)
```

Fig. 6 Pseudo code of extended Euclidean algorithm

Table. 3 Operations of extended Euclidean algorithm

state	r_m	count	R	S	U	V	T
0	0	0	xR	S	U	V	xT
0	1	0	xR+S	xR	U	V	U
0	0	≥ 1	xR	S	U	V	xT
0	1	≥ 1	xR+S	xR	U	V	U
1	0	1	xR	S	V+xT	U	xT
1	0	> 1	xR	S	U	V	xT
1	1	1	xR+S	S	V+ xT+U	U	xT+U
1	1	> 1	xR+S	S	U	V	xT+U

한다. 나눗셈 연산이 시작되면 R은 제어신호에 따라서 xR 또는 $xR+S$ 값을 저장하며, xR은 왼쪽으로 1비트 단순 쉬프트를 의미하고, 더하기 연산은 단순 XOR 연산을 의미한다. S는 저장된 데이터를 유지하거나, xR 값을 저장한다. U, V도 제어신호에 따라서 표와 같이 3가지, 2가지의 경우로 데이터를 저장하게 된다. T는 3가지 경우로 데이터를 저장하는데, xT는 왼쪽으로 1비트 쉬프트하고 기약 다항식을 이용하여 모듈러 연산한 결과 값이다.

확장 유클리드 알고리듬과 표 3을 이용하여 최적화된 나눗셈기를 그림 7과 같은 구조로 설계하였다. 나눗셈기는 233-비트 레지스터 3개, 234-비트 레지스터 2개, XOR 게이트 등으로 구성된다. 233-비트 입력포트 iDATA_A로 나눗셈 연산의 피제수 값이 입력되며 레지스터 U_reg에 저장된다. 나눗셈 연산의 제수 값은 233-비트 입력포트 iDATA_B로 입력되며 레지스터 R_reg에 저장된다. T_reg와 R_reg에 저장된 데이터는 매 클록 사이클 마다 왼쪽으로 1비트씩 쉬프트하게 되

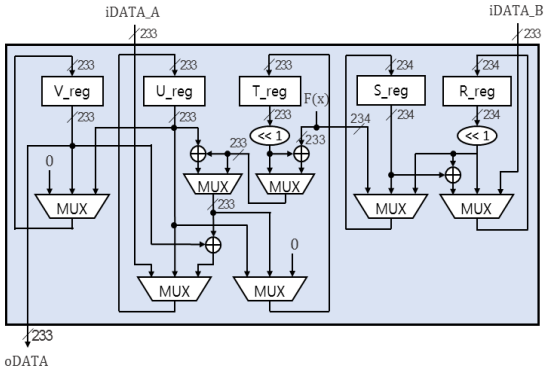
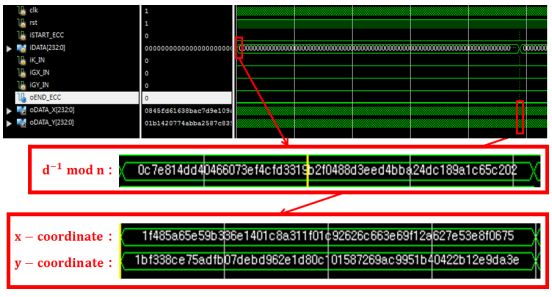


Fig. 7 Finite field divider

며, T_reg에 저장된 데이터의 경우 쉬프트 후에 기약 다항식을 이용하여 모듈러 연산을 하게 된다. 설계된 나눗셈기는 표 3과 같이 3개의 제어신호를 사용하여 전체적인 연산을 제어하게 되며, 데이터를 입력받아 466 클록 사이클 후에 end_div 신호와 함께 V_reg 레지스터로부터 연산 결과값을 출력하도록 설계하였다.

Ⅳ. 기능검증 및 FPGA 구현

설계된 ECC 프로세서는 RTL 시뮬레이션에 의한 기능검증과 FPGA 구현에 의한 하드웨어 동작을 확인하였다. 그림 8은 Xilinx ISim을 이용한 ECC 프로세서의 시뮬레이션 결과값과 문헌 [5]의 참조 구현 값을 보여준다. NIST FIPS 186-2 [4]에 정의되어 있는 Curve B-233 타원곡선 파라미터를 사용하였으며, 233-비트 정수 k “0c7 e814dd40 466073ef 4cfd3319 b2f0488d 3eed4bba 24dc189a 1c65c202”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 x좌표 “1f4 85a65e59 b336e140 1c8a311f 01c92626 c663e69f 12a627e5 3e8f0675”와 y좌표 “1bf 338ce75a dfb07deb



(a)

$d^{-1} \bmod n$ = 00C7 E814DD40 466073EF 4CFD3319 B2F0488D 3EED4BBA 24DC189A 1C65C202

$Q = d^{-1}G = (x, y)$

x = 01F4 85A65E59 B336E140 1C8A311F 01C92626 C663E69F 12A627E5 3E8F0675

y = 01BF 338CE75A DFB07DEB D962E1D8 0C101587 269AC995 1B40422B 12E9DA3E

(b)

Fig. 8 Functional simulation results of ECC processor (a) Scalar multiplication results from ECC processor (b) Reference data

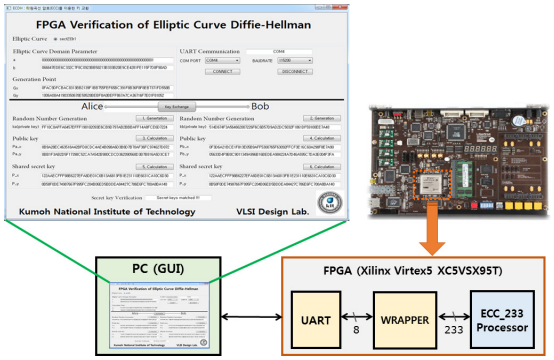


Fig. 9 FPGA verification setup

d962e1d8 0c101587 269ac995 1b40422b 12e9da3e”는 그림 8-(b)의 참조 구현 값과 정확히 일치함을 확인할 수 있다.

기능검증이 완료된 ECC 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 사용되고, k_b는 Bob의 비밀키로 사용된다. 버튼3을 클릭하여 데이터 전송을 실행시키면, 생성된 정수 k_a , 그리고 G_x, G_y가 UART 통신을 통해 FPGA에 구현된 ECC 프로세서로 전송된다. ECC 프로세서는 키 값과 생성점의 좌표값을 이용하여 스칼라 곱셈 $P_a = k_a * G$ 연산을 하고, 결과값은 Alice의 공개키(public key)에 표시된다. 버튼4를 클릭하면, ECC 프로세서는 $P_b = k_b * G$ 연산을 수행하고, 그 결과가 Bob의 공개키에 표시된다. 버튼 5와 6을 순차적으로 클릭하면, Alice와 Bob은 서로의 공개키를 교환한다. 그리고 Alice는 ECC 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. 이와 같이 ECDH 키교환 과정에 의해, Alice와 Bob은 동일한

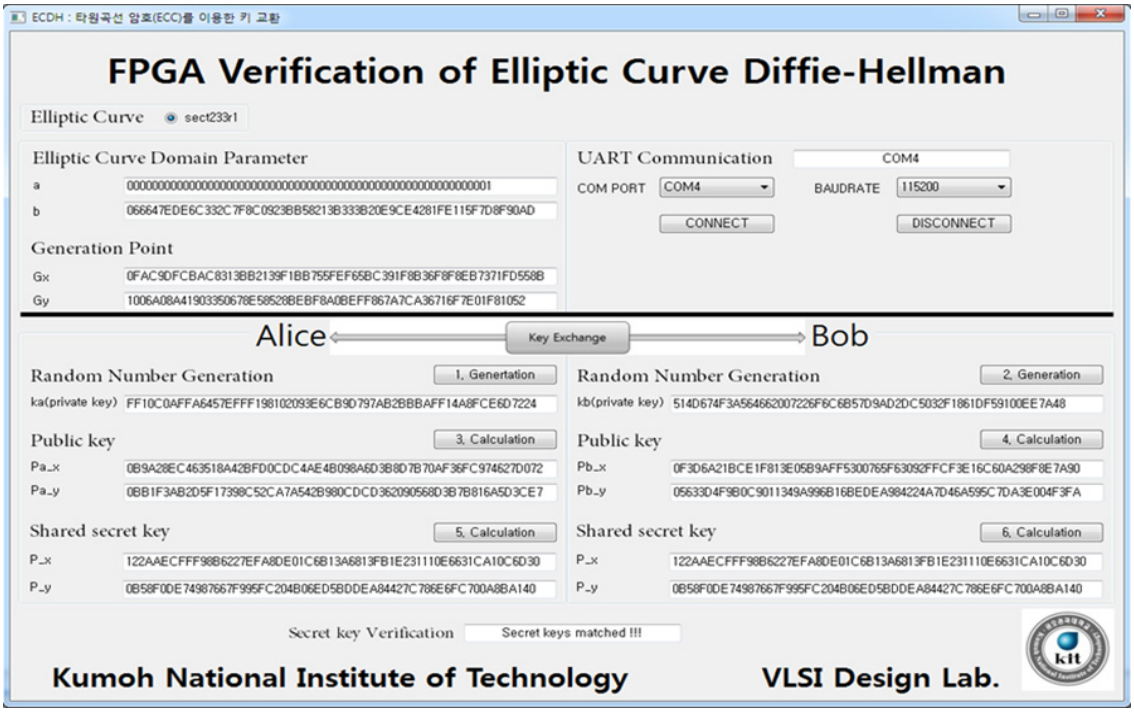


Fig. 10 FPGA verification results of ECC processor using ECDH key exchange protocol

비밀키($P_{alice} = P_{bob}$)를 갖게 되며, 따라서 설계된 ECC 프로세서가 올바르게 동작함을 확인할 수 있다.

설계된 233비트 타원곡선 암호 프로세서는 타원곡선 상의 스칼라 곱셈 연산에 490,699 클록 사이클이 소요되며, 0.18 ㎛ CMOS 표준 셀 라이브러리로 합성한 결과 100 MHz의 동작 주파수에서 49,271 GE로 구현되었다. 최대 동작 주파수 345 MHz에서 스칼라 곱셈 연산에 1.4 msec가 소요되는 것으로 평가되었다.

Ⅴ. 결 론

NIST FIPS 186-2 표준으로 정의되어 있는 타원곡선 B-233을 지원하는 타원곡선 암호 프로세서를 설계하고, ISim을 이용한 기능검증과 FPGA 구현을 통한 하드웨어 동작을 검증하였다. modified Montgomery ladder 알고리듬을 이용하여 단순 전력분석에 안전하도록 설계하였으며, 유한체 상의 곱셈과 나눗셈 연산은 단순 쉬프트 연산과 XOR 게이트만을 사용하여 하드웨어 자원을 최소화하였다. 설계된 ECC 프로세서는 0.18 ㎛ CMOS 공정에서 100 MHz의 동작 주파수로 합성한 결과 49,271 GE로 구현되었다. 최대 345 MHz의 클록 주파수로 동작 가능하며, 스칼라 곱셈 연산에 1.4 msec가 소요된다. 설계된 ECC 프로세서는 공개키 암호의 복잡한 연산을 빠른 속도로 처리하여, 코드의 크기나 메모리가 제한된 응용분야에 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] NIST Std. FIPS-197, *Advanced Encryption Standard*, National Institute of Standard and Technology (NIST), Nov., 2001.

[2] KS X 1213, *128 bit Block Encryption Algorithm ARIA*, Korean Agency for Technology and Standards (KATS), 2004.

[3] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining Digital Signatures and Public-Key Crypto-systems," *Communications of the 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] H. A. Selma and H. M'hamed, "Elliptic curve cryptographic processor design using FPGAs," *Proceedings of the IEEE 2015 3rd International Conference on Control, Engineering & Information Technology (CEIT)*, Univ. of Tlemcen Tlemcen, Algeria, pp. 1-6, May 2015.

[7] 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, Mar. 2012.

[8] V. R. Venkatasubramani, G. R. Kumar, and K. Vignesh, "Fast computation of scalar multiplication over binary edwards curve processor against side channel attack," *Proceedings of the IEEE 2014 International Conference on Electronics and Communication Systems (ICECS)*, Karpagam College, India, pp. 1-7, Feb. 2014.

[9] C. Rebeiro, S. S. Roy, and D. Mukhopadhyay. "Pushing the limits of high-speed GF (2^m) elliptic curve scalar multiplication on FPGAs," *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, Springer Berlin Heidelberg, pp. 494-511, Jan. 2012.

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

[11] J. Park, J. T. Hwang, and Y. C. Kim, "FPGA and ASIC implementation of ECC processor for security on medical embedded system," *Proceedings of the IEEE Third International Conference on Information Technology and Applications (ICITA 2005)*, Sydney, Australia, vol. 2, pp. 547-551, July 2005.

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