ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 19-01-09
58

j.inst.Korean.electr.electron.eng.Vol.23,No.1,58~67,March 2019
http://dx.doi.org/10.7471/ikeee.2019.23.1.58

$GF(2^m)$ 상의 NIST 타원곡선을 지원하는 ECC 프로세서의 경량 하드웨어 구현

A Lightweight Hardware Implementation of ECC Processor Supporting NIST Elliptic Curves over $GF(2^m)$

이 상 현*, 신 경 욱*★

Sang-Hyun Lee*, Kyung-Wook Shin*★

Abstract

A design of an elliptic curve cryptography (ECC) processor that supports both pseudo-random curves and Koblitz curves over $GF(2^m)$ defined by the NIST standard is described in this paper. A finite field arithmetic circuit based on a word-based Montgomery multiplier was designed to support five key lengths using a datapath of fixed size, as well as to achieve a lightweight hardware implementation. In addition, Lopez-Dahab's coordinate system was adopted to remove the finite field division operation. The ECC processor was implemented in the FPGA verification platform and the hardware operation was verified by Elliptic Curve Diffie-Hellman (ECDH) key exchange protocol operation. The ECC processor that was synthesized with a 180-nm CMOS cell library occupied 10,674 gate equivalents (GEs) and a dual-port RAM of 9 kbits, and the maximum clock frequency was estimated at 154 MHz. The scalar multiplication operation over the 223-bit pseudo-random elliptic curve takes 1,112,221 clock cycles and has a throughput of 32.3 kbps.

요 약

NIST 표준으로 정의된 $GF(2^m)$ 상의 슈도 랜덤 곡선과 Koblitz 곡선을 지원하는 타원곡선 암호(ECC) 프로세서 설계에 대해 기술한다. 고정된 크기의 데이터 패스를 사용하여 5가지 키 길이를 지원함과 아울러 경량 하드웨어 구현을 위해 워드 기반 몽고메리 곱셈기를 기반으로 유한체 연산회로를 설계하였다. 또한, Lopez-Dahab 좌표계를 사용함으로써 유한체 나눗셈을 제거하였다. 설계된 ECC 프로세서를 FPGA 검증 플랫폼에 구현하고, ECDH(Elliptic Curve Diffie-Hellman) 키 교환 프로토콜 동작을 통해 하드웨어 동작을 검증하였다. 180-nm CMOS 표준 셀 라이브러리로 합성한 결과 10,674 등가 게이트와 9 kbit의 dual-port RAM으로 구현되었으며, 최대 동작 주파수는 154 MHz로 평가되었다. 223-비트 슈도 랜덤 타원곡선 상의 스칼라 곱셈 연산에 1,112,221 클록 사이클이 소요되며, 32.3 kbps의 처리량을 갖는다.

Key words : ECC, Public-key cryptography, Binary field, Lopez-Dahab, Montgomery Multiplication

* School of Electronic Engineering, Kumoh National Institute of Technology
★ Corresponding author
E-mail : kwshin@kumoh.ac.kr, Tel : +82-54-478-7427
※ Acknowledgment
・This research was supported by Kumoh National Institute of Technology (2018-104-072)
・Authors are thankful to IDEC for supporting EDA software.
Manuscript received Feb. 28, 2019; revised Mar. 18, 2019; accepted Mar. 22, 2019

Ⅰ. 서론

주변의 사물이 인터넷과 연결되어 유기적으로 정보를 주고받는 사물인터넷(Internet of Things; IoT) 기술이 급속히 발전함에 따라 유·무선 네트워크를 통해 유통되는 정보를 안전하게 보호하기 위한 정보보안의 중요성이 크게 대두되고 있다. 오늘날 널리 사용되고 있는 정보보안 체제로는 암호화, 전자서명, 인증 등이 있으며, 대칭키(symmetric- key) 암호, 공개키(public-key) 암호, 해시 함수 등이 사용된다[1]. 전자서명, 인증, 키교환 등의 보안 프로토콜에 필수적인 공개키 암호방식은 RSA (Rivest, Shamir, and Adleman) [2], 타원곡선 암호 (Elliptic Curve Cryptography; ECC) [3] 등이 널리 사용되고 있다.

1985년 Koblitz [4]와 Miller [5]에 의해 제안된 타원곡선 암호는 타원곡선 군(group)의 이산대수 문제(elliptic curve discrete logarithmic problem)에 안전성의 기반을 두고 있으며, 대표적인 공개키 암호 방식인 RSA 보다 짧은 키 길이를 사용하면서도 유사한 안전성을 제공한다는 장점이 있어 응용분야가 급속히 확대되고 있다. ECC는 RSA에 비해 연산량과 요구되는 메모리가 작아 제한된 하드웨어 자원을 갖는 IoT, IC 카드, 스마트 단말기 등의 보안에 적합하므로, ECC의 경량 하드웨어 구현에 관한 다양한 연구가 이루어지고 있다. ECC는 타원곡선이 정의되는 유한체에 따라 소수체(prime field) $GF(p)$ 상의 ECC와 이진체(binary field) $GF(2^m)$ 상의 ECC로 구분되며, 적용되는 체에 따라 타원곡선 방정식도 달라진다. ECC의 하드웨어 구현을 위해서는 타원곡선 종류, 타원곡선이 정의되는 유한체(finite field), 타원곡선 상의 점을 표현하는 좌표계, 필드 크기, 스칼라 곱셈 알고리듬, 유한체 연산 알고리듬 등 다양한 요소들이 고려되어야 하며, 이들 요소에 따라 ECC 프로세서의 성능, 하드웨어 복잡도 등이 영향을 받는다[6, 7].

최근, ECC의 하드웨어 구현에 관한 많은 연구결과들이 발표되고 있으며 [8-14], ECC 프로세서의 연산 성능, 하드웨어 복잡도, 지원되는 유한체와 타원곡선의 종류 등 다양한 측면에서 각기 장·단점을 갖는 것으로 평가된다. 문헌 [11]은 다양한 타원곡선의 지원이 가능한 ECC 프로세서 설계 사례이며, 하드웨어 복잡도가 커서 하드웨어 리소스가 제한적인 응용분야에 적합하지 않은 것으로 평가된다. 문헌 [12, 13]은 저면적 및 고속 동작이 가능한 ECC 프로세서 사례이다. 문헌 [12, 13]은 아핀 좌표계를 사용하여 빠른 연산이 가능하지만 스칼라 곱셈에서 역원 연산을 위한 부가적인 하드웨어 자원이 필요하다는 단점을 갖는다.

본 논문의 ECC 프로세서는 Lopez-Dahab 좌표계와 페르마의 소정리(Fermat's little theorem)를 사용하여 역원 연산을 위한 하드웨어 자원을 절약하였으며, 워드 기반 몽고메리 곱셈기 기반의 고정된 하드웨어로 다양한 키 길이의 타원곡선을 지원할 수 있도록 설계하여 경량 하드웨어로 구현하였다. 2장에서는 타원곡선 암호 알고리듬에 대하여 간략히 설명하고, 3장에서는 ECC 프로세서의 하드웨어 구현에 대해 설명한다. 설계된 ECC 프로세서의 FPGA 검증 결과와 성능평가는 4장에 기술하였으며, 5장에서 결론을 맺는다.

Ⅱ. 타원곡선 암호 알고리듬

비대칭키 암호방식인 타원곡선 암호는 타원곡선 상의 한 점 P에 양의 정수 k를 곱해서 타원곡선 상의 다른 한 점 Q를 얻는 스칼라 곱셈 $Q = kP$를 기반으로 한다. ECC에서 k는 비밀키로 사용되고, P와 Q는 공개키로 사용된다. $GF(2^m)$ 상의 타원곡선은 식 (1)과 같이 정의되며, $b \neq 0$이다.

$$y^2 + xy = x^3 + ax^2 + b \quad (1)$$

ECC의 스칼라 곱셈 $Q = kP$은 타원곡선의 상의 점 가산(point addition)과 점 두배(point doubling) 연산의 반복으로 구현되며, 유한체와 좌표계에 따라 점 연산 수식이 달라진다. 타원곡선이 정의되는 유한체는 이진체 $GF(2^m)$와 소수체 $GF(p)$로 구분되며, NIST 표준에서는 $GF(p)$ 상의 슈도 랜덤(pseudo-random) 타원곡선 5가지(P-192, P-224, P-256, P-384, P-521)와 $GF(2^m)$ 상의 슈도 랜덤 타워곡선 5가지 (B-163, B-233, B-283, B-409, B-571) 그리고 Kobliz 타워곡선 5가지(K-163, K-233, K-283, K-409, K-571)를 정의하고 있다. 슈도 랜덤 타원곡선은 여인자(cofactor)가 2로 안정성이 높으며, Koblitz 타원곡선은 타원곡선의 계수에 따라 여인자가 2 또는 4로 상대적으로 안정성이 낮지만 준동형 사상(homo-morphism)을 가지고 있어 빠른

Table. 1. Point addition and point doubling operations of elliptic curves over $GF(2^m)$.
표 1. $GF(2^m)$ 상의 타원곡선 점 덧셈과 점 두 배 연산

	affine coordinate	Lopez-Dahab coordinate
point addition	$\lambda = (y_1 \oplus y_0)/(x_1 \oplus x_0)$ $x_2 = \lambda^2 \oplus \lambda \oplus x_0 \oplus x_1 \oplus a$ $y_2 = \lambda(x_0 \oplus x_2) \oplus x_2 \oplus y_0$	$A_0 = Y_1 Z_0^2,\ A_1 = Y_0 Z_1^2,\ B_0 = X_1 Z_0,\ B_1 = X_0 Z_1$ $C = A_0 \oplus A_1,\ D = B_0 \oplus B_1,\ E = Z_0 Z_1,\ F = DE$ $G = D^2(F \oplus aE^2),\ H = CF,\ I = D^2 B_0 E \oplus X_2,\ J = D^2 A_0 \oplus X_2$ $Z_2 = F^2,\ X_2 = C^2 \oplus H \oplus G,\ Y_2 = HI \oplus Z_2 J$
point doubling	$\lambda = x_0 \oplus x_0/y_0$ $x_2 = \lambda^2 \oplus \lambda \oplus a$ $y_2 = x_0^2 \oplus (\lambda \oplus 1)x_2$	$Z_2 = Z^{2_0} X^{2_0}$ $X_2 = X_0^4 Z_2 \oplus bZ_0^4$ $Y_2 = bZ_0^4 Z_2 \oplus X_2(aZ_2 \oplus Y_0^2 \oplus bZ^{4_0})$

연산 속도를 갖는다. 여인자는 타원곡선의 위수(order)를 점의 위수로 나눈 값으로 1에 가까울수록 안정성이 높다.

타원곡선 상의 점을 표현하기 위해 사용되는 좌표계는 일반 좌표계인 아핀(affine) 좌표계와 투영(projection) 좌표계인 Lopez-Dahab, Jacobian 좌표계 등이 있다. 표 1은 아핀 좌표계와 Lopez-Dahab 좌표계를 적용한 점 연산 수식의 비교를 보이고 있다. 아핀 좌표계는 상대적으로 연산 수식이 간단하지만 나눗셈 연산이 필요하며, 투영 좌표계는 연산 수식이 다소 복잡하지만 나눗셈 연산이 없어 두 좌표계가 서로 장점과 단점을 갖는다. 본 논문의 ECC 프로세서는 NIST에서 정의한 $GF(2^m)$ 상의 10가지 타원곡선을 지원하며, 경량 하드웨어 구현에 적합하도록 Lopez-Dahab 좌표계를 적용하여 설계하였다.

III. ECC 프로세서의 경량 하드웨어 설계

1. ECC 프로세서 구조

NIST 표준으로 정의된 $GF(2^m)$ 상의 타원곡선 10가지를 지원하도록 설계된 ECC 프로세서의 전체 구조는 그림 1과 같다. 스칼라 곱셈 연산에 사용되는 데이터와 중간 결과 값을 저장하는 Data_MEM 블록, 유한체 연산에 사용될 32-비트의 데이터와 개인키를 저장하는 REGs 블록, 유한체 곱셈과 덧셈 연산을 수행하는 GFb_ALU 블록, 스칼라 곱셈 연산을 제어하는 제어블록으로 구성된다.

Data_MEM 블록은 288×32-비트의 듀얼 포트 메모리로 구성되며, 타원곡선 계수, 개인키, 기약다항식, 타원곡선의 생성점, 몽고메리 도메인 변환 데이터, 그리고 스칼라 곱셈 연산의 중간결과 값을 저장한다. REGs 블록은 5×32-비트의 레지스터로 구성되며, 32-비트 단위의 유한체 곱셈의 승수와 유한체 곱셈 중간결과 값을 WMM_Data_ REGs에 저장한다. 그리고 32-비트의 개인키 워드를 레지스터 Key_reg에 저장하여 MSB 값에 따라 스칼라 곱셈 알고리듬의 연산에 사용된다. 레지스터 Key_reg는 한 사이클의 점 연산이 끝날 때 마다 1-비트씩 왼쪽으로 시프트되며 32번 시프트 할 때마다 다음 워드의 개인키를 메모리부터 입력받아 저장한다.

GFb_ALU 블록은 워드 기반 몽고메리 곱셈 알고리듬 [15]에 의해 유한체 곱셈과 덧셈을 수행하며, 워드 기반 유한체 곱셈기와 유한체 가산을 수행하는 XOR 게이트로 구성된다. 투영좌표계에서 일반 좌표계로 변환할 때 사용되는 유한체 나눗셈은 제수의 곱의 역원을 구하여 피제수와 곱하는 연산으로 구현되었으며, 곱의 역원은 페르마의 소정리를 이용하여 구현되었다. 제어 블록은 FSM(finite state machine)으로 구현되며, Data_MEM, REGs, GFb_ALU 블록의 내부 동작을 제어하고, 또한 타원곡선의 키 길이와 연산모드에 따른 소요 클록 사이클 수를 반영하여 ECC 프로세서의 전체 동작을 제어한다.

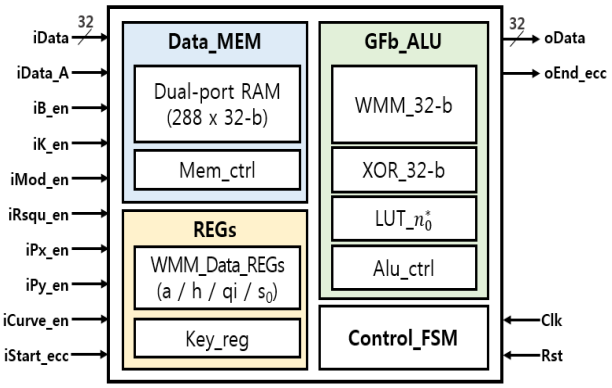


Fig. 1. Architecture of ECC processor in $GF(2^m)$.
그림 1. $GF(2^m)$ 상의 ECC 프로세서 구조

$Input : A = \{a_{m-1}, \cdots, a_1, a_0\}_{2^w}, 0 < A < order$
$B = \{b_{m-1}, \cdots, b_1, b_0\}_{2^w}, 0 < B < order$
$N = \{n_{m-1}, \cdots, n_1, n_0\}_{2^w}$
$Pre\_computed : n_0^* = n_0^{-1} \bmod 2^{32}$
$Output : S = A \times B \times R^{-1} \bmod N$

01 : $for\ i = 0\ to\ m-1\ do$
02 : $\quad C1_{-1} \leftarrow 0, C2_{-1} \leftarrow 0$
03 : $\quad for\ j = 0\ to\ m-1\ do$
04 : $\quad\quad \{C1_j, P1\} \leftarrow a_i \times b_j$
05 : $\quad\quad H \leftarrow P1 \wedge S_j \wedge C1_{j-1}$
06 : $\quad\quad if\ j = 0\ then$
07 : $\quad\quad\quad q_i \leftarrow H \times n_0^* \bmod 2^{32}$
08 : $\quad\quad end\ if$
09 : $\quad\quad \{C2_j, P2\} \leftarrow q_i \times n_j$
10 : $\quad\quad S_{j-1} \leftarrow P2 \wedge H \wedge C2_{j-1}$
11 : $\quad end\ for$
12 : $\quad S_{m-1} \leftarrow C1_{m-1} \wedge C2_{m-1}$
13 : $end\ for$
14 : $return\ S$

Fig. 2. Pseudo-code for word-based Montgomery multiplication algorithm.
그림 2. 워드 기반 몽고메리 곱셈 알고리듬의 슈도코드

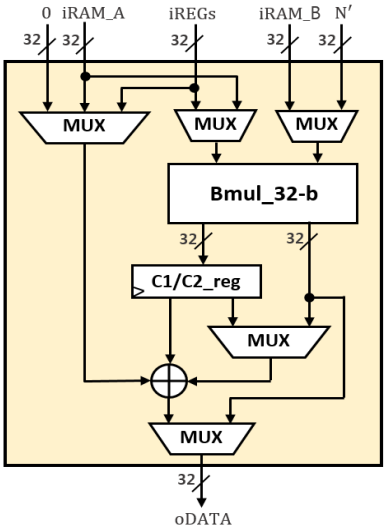


Fig. 3. Word-based Montgomery multiplier.
그림 3. 워드 기반 몽고메리 곱셈기

본 논문의 ECC 프로세서는 Lopez-Dahab 좌표계와 페르마의 소정리를 적용하여 역원 연산을 위한 하드웨어 자원을 절약하였으며, 32-비트 워드 기반 몽고메리 곱셈기 기반의 고정된 크기의 하드웨어로 다양한 키 길이의 타원곡선을 지원하도록 경량 하드웨어로 설계되었다.

2. 워드 기반 유한체 곱셈기

$GF(2^m)$ 상의 유한체 곱셈기는 그림 2의 워드 기반 몽고메리 곱셈 알고리듬의 슈도코드를 기반으로 설계되었다. 그림 2의 슈도코드에서 w는 워드 크기이며, m은 워드 개수를 나타낸다. 본 논문에서는 데이터 패스를 32-비트로 설계하여 워드 크기는 $w = 32$이고, 워드 개수 m은 타원곡선의 길이를 워드 크기로 나누고 자리올림을 한 값으로서 B-163의 경우 $m = 6$이다. 워드 기반 몽고메리 곱셈기는 유한체 곱셈의 승수와 피승수를 32-비트의 워드들로 분할하고, 각 워드에 대한 반복 연산을 통해 임의의 길이의 유한체 곱셈을 연산하며, 합동 특성을 이용하여 모듈러 축약이 이루어진다. 이와 같이, 워드 단위의 유한체 연산 하드웨어를 기반으로 연산 사이클 수를 가변시킴으로써 다양한 키 길이의 타원곡선 연산을 지원할 수 있도록 설계하였다.

그림 3은 그림 2의 슈도코드를 기반으로 설계된 워드 기반 몽고메리 곱셈기의 내부 구성도이며, 32-비트 이진체 곱셈기, XOR 가산기, 곱셈 결과의 상위 워드를 저장하는 32-비트 레지스터 두 개, 그리고 멀티플렉서 등으로 구성된다. 그림 2의 슈도코드에서 j-루프 내부의 이진체 곱셈은 $j = 0$일 때 단계-4, 단계-7, 단계-9에서 3회 연속 수행되며, $j > 0$인 경우에는 단계-4와 단계-9에서 2회 연속 수행되어야 한다. 세 개의 이진체 곱셈기를 사용하면 병렬 처리를 극대화할 수 있으나, 본 논문에서는 저면적 하드웨어 구현을 위해 하나의 이진체 곱셈기를 사용하였으며, 이진체 곱셈에 한 클록 사이클이 소요되도록 설계하였다.

그림 2의 슈도코드에서 j-루프는 피승수 B와 승수 A의 한 워드에 대한 연산을 나타내며, 승수 A의 워드 개수 m만큼 i-루프를 반복하여 연산 결과를 얻는다. 모듈러 합동을 만들기 위해 사용되는 데이터 n_0^*은 타원곡선에 따라 고정된 값을 가지며, 표 2와 같이 LUT로 구현하였다. 그림 2의 슈도코드 단계-7에서 q_i는 최하위 워드가 0이 되도록 모듈러 합동을 만들기 위해 사용되며, 데이터 q_i의 생성을 위해 $j = 0$일 때 3 클록 사이클이 소요되며, 나머지는 2 클록 사이클이 소요된다. 따라서 j-루프 동안 $3 + 2(m-1)$ 클록 사이클이 소요되며, i-루프가 끝나기 전에 S_{m-1} 값의 생성을 위해 1 클록 사이클이 추가로 소요되어 한 번의 i-루프에 $2(m+1)$ 클록 사이클이 소요된다. 따라서 본 논문에서 설계된 이진체 상의 유한체 곱셈기는 최종 곱셈 결과를 얻기 위해 총 $2m(m+1)$ 클록 사이클이 소요된다.

Table. 2. Data $n^{*}{}_{0}$ according to elliptic curves.

표 2. 타원곡선에 따른 데이터 $n^{*}{}_{0}$

Curve	$n_0^* = n_0^{-1} \bmod 2^{32}$
B-163	2115400329
B-233	1
B-283	4286239905
B-409	1
B-571	2193074037

3. 제어 FSM

제어 FSM 블록은 GFb_ALU 블록, Data_Mem 그리고 REGs 블록의 하드웨어 자원을 이용하여 스칼라 곱셈 연산이 수행되도록 ECC 프로세서의 전체적인 동작을 제어하며, 그림 4의 상태 천이도를 갖는 유한상태머신으로 구현되었다. 설계된 ECC 프로세서는 데이터 및 파라미터 입력, 매핑, 점 연산, 좌표계 변환, 리매핑, 데이터 출력의 과정을 통해 스칼라 곱셈 연산을 수행한다.

데이터 입력 단계에서는 스칼라 곱셈 연산에 필요한 파라미터를 입력 받는다. 32-비트 입력포트 iData를 통해 타원곡선의 길이, 타원곡선 계수, 개인키, 기약다항식, 타원곡선의 생성점, 몽고메리 도메인 변환 데이터가 입력되며, NIST에 정의된 타원곡선의 계수 a는 0 또는 1이므로 1-비트 입력포트 iData_A를 통해 입력된다.

상태 MAPPING은 Lopez-Dahab 3차원 좌표계 변환과 몽고메리 도메인 변환을 처리하는 단계이며, 3차원 좌표계의 점 (X, Y, Z)에서 초기 값 Z는 1이다. 몽고메리 도메인 변환은 피연산자에 R^2을 곱하여 $WMM(A, R^2, N) = A \times R$의 형태로 변환하는 과정이며, 워드기반 몽고메리 곱셈기가 사용된다. 여기서 R은 $2^m \bmod P$이고, m은 타원곡선이 정의되는 이진체의 크기를 나타낸다.

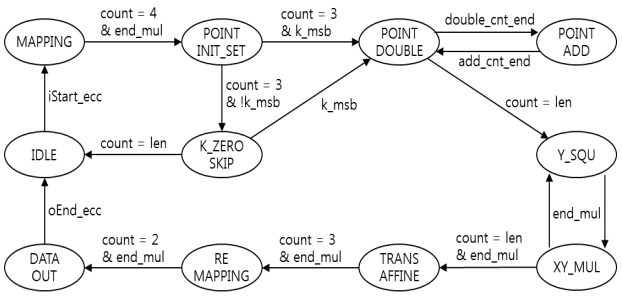


Fig. 4. State transition diagram of control FSM.

그림 4. 제어 FSM의 상태 천이도

변환이 완료된 데이터는 몽고메리 래더 알고리듬[16]을 적용하여 점 덧셈과 점 두 배 연산의 반복으로 스칼라 곱셈이 연산된다. 상태 POINT_ INIT_SET에서는 매핑된 타원곡선의 생성점을 점 연산의 초기값으로 설정하며, 상태 K_ZERO_SKIP에서는 개인키가 저장된 레지스터 Key_reg를 왼쪽으로 시프트 시키면서 MSB를 스캔하여 처음으로 1이 나올 때 까지 대기한다. 이 후 MSB 값에 따라 몽고메리 래더 알고리듬을 수행한다. 상태 POINT_ADD와 상태 POINT_DOUBLE에서는 각각 점 덧셈과 점 두 배 연산이 수행된다.

Lopez-Dahab 좌표계를 이용한 점 가산과 점 두 배 연산은 그림 5의 과정으로 계산된다. Koblitz 타원곡선의 경우에 타원곡선의 계수 $a = 0$이므로, 점 덧셈 연산은 단계-15와 단계-16이 생략되고, 점 두 배 연산은 단계-10이 생략되어 연산 시간이 단축된다.

스칼라 곱셈이 완료된 결과는 3차원 좌표계인

Input : $X_0,\ Y_0,\ Z_0,\ X_1,\ Y_1,\ Z_1$
Output : $X_2,\ Y_2,\ Z_2$
$R_1 \leftarrow X_0, R_2 \leftarrow Y_0, R_3 \leftarrow Z_0$
$R_4 \leftarrow X_1, R_5 \leftarrow Y_1, R_6 \leftarrow Z_1$

01: $T_0 \leftarrow R_1 * R_6$
02: $T_1 \leftarrow R_4 * R_3$
03: $T_0 \leftarrow T_0 \wedge T_1$
04: $T_2 \leftarrow R_3 * R_6$
05: $T_3 \leftarrow R_3 * R_3$
06: $T_4 \leftarrow R_5 * T_3$
07: $T_3 \leftarrow R_6 * R_6$
08: $T_3 \leftarrow T_3 * R_2$
09: $T_3 \leftarrow T_3 \wedge T_4$
10: $R_1 \leftarrow T_0 * T_2$
11: $T_0 \leftarrow T_0 * T_0$
12: $R_3 \leftarrow R_1 * R_1$
13: $R_2 \leftarrow T_3 * R_1$
14: $T_3 \leftarrow T_3 * T_3$
if $a = 1$ *then*
15: $T_5 \leftarrow T_2 * T_2$
16: $R_1 \leftarrow R_1 \wedge T_5$
end
17: $R_1 \leftarrow R_1 * T_1$
18: $R_1 \leftarrow R_1 \wedge R_2$
19: $R_1 \leftarrow R_1 \wedge T_3$
20: $T_1 \leftarrow T_1 * T_0$
21: $T_1 \leftarrow T_1 * T_2$
22: $T_1 \leftarrow T_1 \wedge R_1$
23: $T_0 \leftarrow T_0 * T_4$
24: $T_0 \leftarrow T_0 \wedge R_1$
25: $T_0 \leftarrow T_0 * R_3$
26: $R_2 \leftarrow R_2 * T_1$
27: $R_2 \leftarrow R_2 \wedge T_0$

$X_2 \leftarrow R_1, Y_2 \leftarrow R_2, Z_2 \leftarrow R_3$

(a)

Input : $X_1,\ Y_1,\ Z_1, b$
Output : $X_2,\ Y_2,\ Z_2$
$R_0 \leftarrow b$
$R_1 \leftarrow X_1, R_2 \leftarrow Y_1, R_3 \leftarrow Z_1$

01: $R_3 \leftarrow R_3 * R_3$
02: $T_0 \leftarrow R_3 * R_3$
03: $T_0 \leftarrow T_0 * R_0$
04: $R_1 \leftarrow R_1 * R_1$
05: $R_3 \leftarrow R_3 * R_1$
06: $R_1 \leftarrow R_1 * R_1$
07: $R_1 \leftarrow R_1 \wedge T_0$
08: $R_2 \leftarrow R_2 * R_2$
09: $R_2 \leftarrow R_2 \wedge T_0$
if $a = 1$ *then*
10: $R_2 \leftarrow R_2 \wedge R_3$
end
11: $R_2 \leftarrow R_2 * R_1$
12: $T_0 \leftarrow T_0 * R_3$
13: $R_2 \leftarrow R_2 \wedge T_0$

$X_2 \leftarrow R_1, Y_2 \leftarrow R_2, Z_2 \leftarrow R_3$

(b)

Fig. 5. Pseudo code for point operations using Lopez- Dahab's coordinate, (a) point addition, (b) point doubling.

그림 5. Lopez-Dahab 좌표계를 사용하는 점 연산 슈도코드, (a) 점 덧셈, (b) 점 두 배

Lopez-Dahab 좌표계로 출력된다. 스칼라 곱셈 결과 (XR, YR, ZR)를 2차원 좌표계인 아핀 좌표계 $(x', y') = (XR/Z, YR/Z^2)$로 역변환 하는 연산은 상태 XY_MUL, Y_SQU, TRANS_AFFINE를 통해 수행된다. 좌표계 변환을 위해서는 유한체 나눗셈 연산이 필요하며, 상태 XY_MUL와 상태 Y_SQU에서 페르마 소정리를 이용하여 곱의 역원을 구하고, 상태 TRANS_AFFINE에서 연산된 곱의 역원을 사용해 아핀 좌표계로 변환한다.

상태 RE_MAPPING에서는 몽고메리 도메인의 결과를 일반 도메인으로 역변환 하는 연산이 수행된다. 상태 MAPPING과 동일하게 워드 기반 몽고메리 곱셈기가 사용되며, 몽고메리 도메인의 AR에 상수 1을 곱하여 $WMM(AR, 1, N) = A$의 형태로 변환된다. 마지막으로, 모든 연산이 완료된 최종 결과는 상태 DATA_OUT를 통해 최하위 워드부터 외부로 출력된다.

Ⅳ. 기능검증 및 성능평가

설계된 $GF(2^m)$ ECC 프로세서는 ModelSim으로 RTL 기능검증을 하였으며, FPGA 구현을 통해 하드웨어 동작을 검증하였다. NIST FIPS 186-2에 정의되어 있는 타원곡선 계수, 타원곡선의 생성점 등의 파라미터를 검증에 사용하였다.

1. RTL 기능검증

그림 6-(a)는 키 길이 233-비트 슈도 랜덤 타원곡선 상의 스칼라 곱셈 연산에 대한 시뮬레이션 결과이며, 개인키 “00C7 E814DD40 466073EF 4CFD3319 B2F0488D 3EED4BBA 24DC189A 1C65C202”를 최하위 워드부터 입력하여 스칼라 곱셈 연산 결과로 x-좌표 값 “01F4 85A65E59 B336E140 1C8A311F 01C92626 C663E69F 12A627E5 3E8F0675”, y-좌표 값 “01BF 338CE75A DFB07DEB D962E1D8 0C101587 269AC995 1B40422B 12E9DA3E”가 최하위 워드부터 출력된다. 그림 6-(b)는 키 길이 233-비트 Koblitz 타원곡선의 스칼라 곱셈 연산에 대한 시뮬레이션 결과이며, 개인키 “007C 916E6A3B A7D23900 ED41E74D FD50F55E E2C2DF89 134E9CEA F471C6CA”를 최하위 워드부터 입력하여 스칼라 곱셈 연산 결과로 x-좌표 값 “01E9 1DEFBD41 AE655105 E046E03E C13E3860 0E9A2C9A 920B8E75 53721605”, y-좌표 값 “0112 9C2706D1 9D134891 C7BAD84A 5600C2AF F86068C4 7497F5BD 498D0B76”가 최하위 워드부터 출력된다. 그림 6의 시뮬레이션 결과는 문헌 [17]의 참조 구현 값과 정확히 일치하여 설계된 ECC 프로세서가 올바로 동작함을 확인하였다.

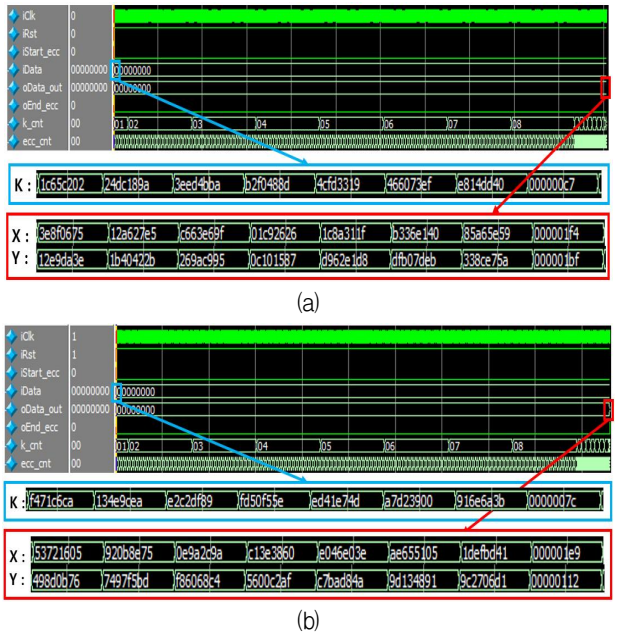


Fig. 6. RTL simulation results for scalar multiplication of ECC processor, (a) 233-bit pseduo-random curve, (b) 233-bit Koblitz curve.

그림 6. ECC 프로세서의 스칼라 곱셈 연산에 대한 RTL 시뮬레이션 결과 (a) 233-비트 슈도 랜덤 커브, (b) 233-비트 Koblitz 커브

2. FPGA 구현을 통한 하드웨어 동작 검증

RTL 기능검증이 완료된 ECC 프로세서는 FPGA 구현을 통해 하드웨어 동작을 검증하였다. FPGA 검증 플랫폼은 그림 7과 같으며, FPGA 보드, uart 인터페이스, Python 기반 구동 및 GUI 소프트웨어로 구성되며, FPGA 소자는 Xilinx Virtex5 XC5VSX-95T 디바이스가 사용되었다. FPGA에 구현된 ECC 프로세서로 테스트 벡터를 로딩한 후, ECC 연산을 수행한 결과가 PC로 전송되어 GUI 화면에 표시된다.

그림 8은 ECDH(Elliptic Curve Diffie-Hellman) 키 교환 프로토콜을 구현한 FPGA 검증화면이다. 그림 8-(a)는 키 길이 571-비트의 슈도 랜덤 타원곡선에 대한 ECDH 프로토콜의 동작결과이며, 그림 8-(b)는 키 길이 571-비트의 Koblitz 타원곡선에 대한 ECDH 프로토콜의 동작결과이다. FPGA 검증 시스템의 동작 과정은 다음과 같다.

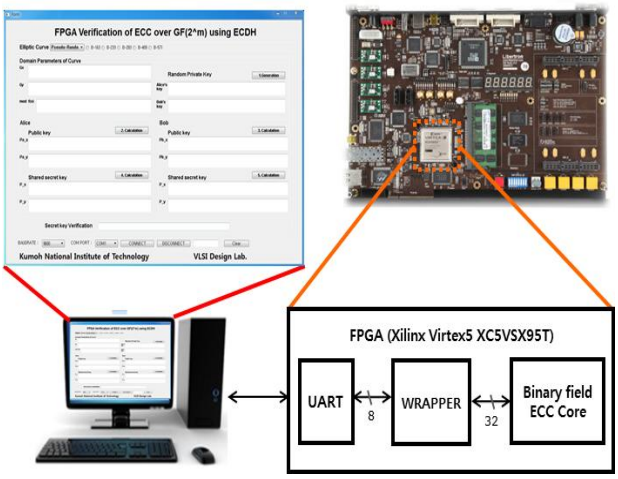
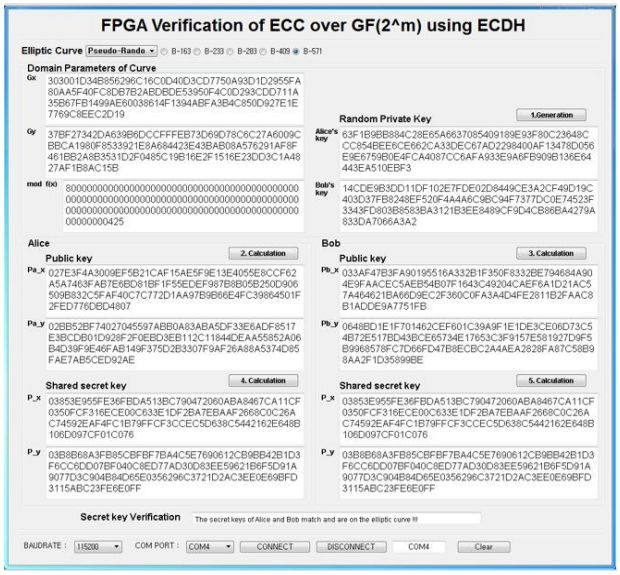


Fig. 7. FPGA verification platform for ECC processor.
그림 7. ECC 프로세서의 FPGA 검증 플랫폼

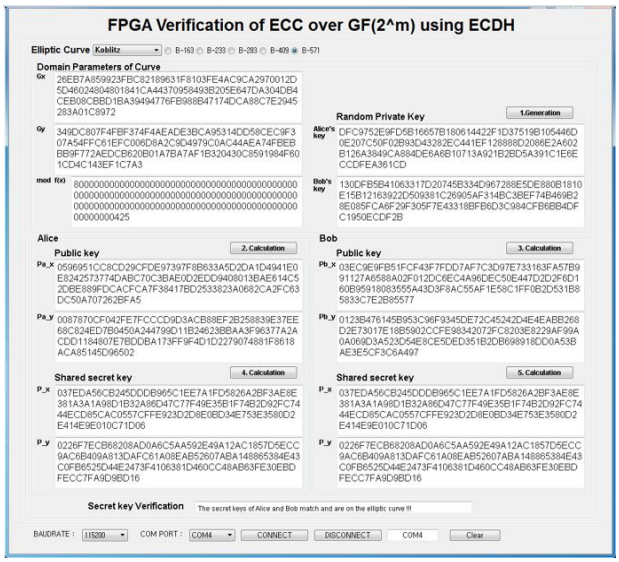
GUI 상단의 타원곡선 종류를 선택하면 타원곡선의 생성점과 모듈러 값이 화면에 표시된다. 1.Generation 버튼을 클릭하면 Alice와 Bob의 개인키가 생성되며, 2.Calculation 버튼을 클릭하면 Alice의 개인키, 생성점, 모듈러 값 등의 파라미터가 FPGA로 전송되고, FPGA에 구현된 ECC 프로세서에서 스칼라 곱셈이 연산된 후, 그 결과가 PC로 전송되어 Alice의 공개키가 화면에 표시된다. 3.Calculation 버튼을 클릭하면 Bob의 개인키, 생성점, 모듈러 값 등의 파라미터가 FPGA로 전송되고, ECC 프로세서에서 스칼라 곱셈이 연산된 후, 그 결과가 PC로 전송되어 Bob의 공개키가 화면에 표시된다. 4.Calculation 버튼을 클릭하면 Alice의 개인키와 생성된 공개키 등의 파라미터가 FPGA로 전송되고, ECC 프로세서에서 스칼라 곱셈이 연산된 후, 그 결과가 PC로 전송되어 공유된 비밀키가 화면에 표시된다. 마찬가지로, 5.Calculation 버튼을 클릭하면 Bob의 개인키와 생성된 공개키 등의 파라미터가 FPGA로 전송되고, ECC 프로세서에서 스칼라 곱셈이 연산된 후, 그 결과가 PC로 전송되어 공유된 비밀키가 화면에 표시된다. 이상의 과정으로 Alice와 Bob 사이에 공유된 비밀키와 소프트웨어로 계산된 결과의 일치 여부가 화면에 표시된다. 이와 같은 ECDH 키 교환 프로토콜을 통해 설계된 ECC 프로세서의 하드웨어 동작을 검증하였다.

3. 성능평가

Verilog HDL로 설계된 이진체 ECC 프로세서를 180-nm CMOS 표준 셀 라이브러리로 합성한 결과



(a)



(b)

Fig. 8. Screenshots of FPGA verification results of the ECC processor, (a) ECDH using 571-bit pseudo-random curve, (b) ECDH using 571-bit Koblitz curve.
그림 8. ECC 프로세서의 FPGA 검증결과 화면, (a) 571-비트 슈도 랜덤 커브를 이용한 ECDH, (b) 571-비트 Koblitz 커브를 이용한 ECDH

10,674 GEs(gate equivalences)와 9 kbit의 메모리로 구현되었으며, 최대 동작 주파수는 154 MHz로 평가되었다. Virtex5 XC5VSX-95T FPGA 디바이스로 합성한 결과, 1,075 슬라이스와 한 개의 BRAM으로 구현되었으며, 최대 동작 주파수는 96 MHz로 평가되었다. 표 3은 타원곡선 종류와 키 길이에 따른 스칼라 곱셈 연산에 소요되는 클록 사이클 수를 보인 것이며, Koblitz 타원곡선의 연산 속도가 약 4% 빠른 것으로 나타났다.

Table. 3. Clock cycles required for scalar multiplication.
표 3. ECC 스칼라 곱셈의 소요 클록 사이클 수

Koblitz curve	# of cycles	Pseudo-random curve	# of cycles
K-163	467,297	B-163	467,297
K-233	1,064,995	B-233	1,112,221
K-283	1,597,372	B-283	1,666,205
K-409	4,564,326	B-409	4,760,722
K-571	11,818,743	B-571	12,259,754

표 4는 본 논문에서 설계된 이진체 ECC 프로세서와 문헌에 발표된 사례들을 비교한 것이다. 본 논문과 동일하게 NIST 표준의 10가지 $GF(2^m)$ 상의 타원곡선을 지원하는 문헌 [11]의 사례는 본 논문의 ECC 프로세서에 비해 약 7.4배 많은 슬라이스가 사용되고, 출력율은 약 41% 낮다. 두 가지의 슈도 랜덤 타원곡선 B-233, B-283을 지원하는 문헌 [12]의 사례는 아핀 좌표계를 사용하여 데이터 처리율이 85.1 kbps로 본 논문의 ECC 프로세서에 비해 약 4.2배 높으나, 약 2.8배 많은 슬라이스를 필요로 한다. 문헌 [13]의 사례는 하나의 슈도 랜덤 타원곡선 B-233을 지원하며, 본 논문의 설계에 비해 최대 동작 주파수와 출력율은 높으나, 약 4.6배 많은 게이트 수를 필요로 한다. 본 논문에서 설계된 ECC 프로세서는 워드 기반 몽고메리 곱셈기를 사용하여 고정된 하드웨어로 다양한 키 길이의 타원곡선들을 지원하며, 경량 하드웨어로 구현되었다는 장점을 갖는다.

V. 결론

NIST 표준에 정의된 $GF(2^m)$ 상의 10가지 타원곡선을 지원하는 ECC 프로세서를 Lopez-Dahab 좌표계와 워드 기반 몽고메리 곱셈기를 적용하여 경량 하드웨어로 설계하였으며, FPGA 구현을 통해 하드웨어 동작을 검증하였다. 본 논문의 ECC 프로세서를 180-nm CMOS 표준 셀 라이브러리로 합성한 결과 10,674 GEs와 9 kbit의 RAM으로 구현되었으며, 154 MHz의 클록 주파수로 동작하여 약 32 kbps의 성능을 갖는 것으로 평가되었다. 본 논문의 이진체 ECC 프로세서는 10가지 타원곡선을 지원하므로 유용성 측면에서 우수하며, 경량 하드웨어로 구현되어 제한된 자원을 갖는 IoT, 무선 센서 네트워크 (WSN) 등의 공개키 암호 시스템 구현에 적합한 것으로 평가된다.

References

[1] A. Firestone, “Information Security Overview,” *Security Industry Association*, pp. 1-25, 2018.

Table 4. Comparison of ECC processors.
표 4. ECC 프로세서의 비교

		This paper	[11]	[12]	[13]
Field size [bits]		163, 233, 283, 409, 571	163, 233, 283, 409, 571	233, 283	233
Elliptic curve		Koblitz, pseudo-random	Koblitz, pseudo-random	pseudo-random	pseudo-random
Coordinate system		Lopez-Dahab	Lopez-Dahab	Affine	Affine
Finite field multiplication		32-bit word-based Montgomery multiplier	Karatsuba-Ofman multiplication algorithm	Interleaved modular reduction algorithm	233-b shift-and-add multiplier
Technology	ASIC	180 nm	-	-	180 nm
	FPGA	Virtex-5	Virtex-5	Kintex-7	-
Hardware complexity	ASIC	10,674 GEs, 9 kbit RAM	-	-	49,271 GEs
	FPGA	1,075 slices, 1 BRAM	7,978 slices	3,016 slices	-
# of clock cycles for scalar multiplication*		1,112,221	-	679,776	490,699
Throughput* [kbps]	ASIC	32.3	-	-	166.4
	FPGA	20.1	11.8	85.1	-
Max. frequency [MHz]	ASIC	154	-	-	345
	FPGA	96	154	255	-

* B-233 pseudo-random curve

[2] 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, 1978.
DOI: 10.1145/359340.359342
[3] NIST Std. FIPS PUB 186-2, Digital Signature Standard (DSS), National Institute of Standard and Technology (NIST), 2000.
[4] N. Koblitz, “Elliptic Curve Cryptosystems,” *Mathematics of Computation*, vol. 48, no. 177, pp. 203-309, 1987.
DOI: 10.1090/S0025-5718-1987-0866109-5
[5] V. S. Miller, “Uses of Elliptic Curves in Cryptography,” *Advances in cryptography-CRYPTO '85*, LNCS 218, Springer-Verlag, pp. 417-426, 1986. DOI: 10.1007/3-540-39799-X_31
[6] H. Marzouqi, M. Al-Qutayri and K. Salah, “Review of Elliptic Curve Cryptography processor designs,” *Microprocessors and Microsystems*, vol. 39, pp. 97-112, 2015.
DOI: 10.1016/j.micpro.2015.02.003
[7] B. G. Park and K. W. Shin, “A Lightweight ECC Processor Supporting Elliptic Curves over NIST Prime Fields,” *Journal of The Institute of Electronics and Information Engineers,* vol. 55, no. 9, pp. 35-43, 2018.
DOI: 10.5573/ieie.2018.55.9.35
[8] P. M. Matutino, J. Araújo, L. Sousa and R. Chaves, “Pipelined FPGA coprocessor for elliptic curve cryptography based on residue number system,” *2017 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation(SAMOS)*, Pythagorion, pp. 261-268, 2017. DOI: 10.1109/SAMOS.2017.8344638
[9] Z. He and X. Chen, “Design and implementation of high-speed configurable ECC co-processor,” *2017 IEEE 12th International Conference on ASIC (ASICON)*, Guiyang, pp. 734-737, 2017.
DOI: 10.1109/ASICON.2017.8252580
[10] K. M. John and S. Sabi, “A novel high performance ECC processor architecture with two staged multiplier,” *2017 IEEE International Conference on Electrical, Instrumentation and Communication Engineering (ICEICE)*, Karur, pp. 1-5, 2017. DOI: 10.1109/ICEICE.2017.8191885
[11] K. C. Cinnati Loi, Sen An and Seok-Bum Ko, “FPGA Implementation of Low Latency Scalable Elliptic Curve Cryptosystem Processor in $GF(2^m)$,” *Proceedings of 2014 IEEE International Symposium on Circuits and Systems (ISCAS'14)*, Melbourne, pp. 822-825, 2014.
DOI: 10.1109/ISCAS.2014.6865262
[12] M. S. Hossain, E. Saeedi, and Y. Kong. “High-Speed, Area-Efficient, FPGA-Based Elliptic Curve Cryptographic Processor over NIST Binary Fields,” *Data Science and Data Intensive Systems (DSDIS), 2015 IEEE International Conference on. IEEE*, 2015. pp. 175-181.
DOI: 10.1109/DSDIS.2015.44
[13] B. G. Park, and K. W. Shin, “A small-area implementation of cryptographic processor for 233-bit elliptic curves over binary field,” *Journal of the Korea Institute of Information and Communication Engineering,* vol. 21, no. 7, pp. 1267-1275, 2017.
[14] L. Li and S. Li, “High-Performance Pipelined Architecture of Point Multiplication on Koblitz Curves,” in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 11, pp. 1723-1727, 2018. DOI: 10.1109/TCSII.2017.2785382
[15] C. K. koc, T. Acar, and B. S. Kaliski, “Analyzing and comparing Montgomery multiplication algorithms,” *IEEE Micro*, vol. 16, no. 3, pp. 26-33, 1996. DOI: 10.1109/40.502403
[16] P. L. Montgomery, “Speeding the Pollard and elliptic curve methods of factorization,” *Mathematics of computation*, vol. 48, no. 177 pp. 243-264, 1987.
DOI: 10.1090/S0025-5718-1987-0866113-7
[17] 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), 2012.

BIOGRAPHY

Sang-Hyun Lee (Member)

2017 : BS degree in Electronic Engineering, Kumoh National Institute of Technology.
2017~ : Graduate student, Kumoh National Institute of Technology

Kyung-Wook Shin (Member)

1984 : BS degree in Electronic Engineering, Korea Aerospace University
1986 : MS degree in Electronic Engineering, Yonsei University
1990 : Ph.D. degree in Electronic Engineering, Yonsei University
1990~1991 : Senior Researcher, Semiconductor Research Center, Electronics and Telecommunications Research Institute (ETRI)
1991~ : Professor in School of Electronic Engineering, Kumoh National Institute of Technology
1995~1996 : University of Illinois at Urbana- Champaign (Visiting Professor)
2003~2004 : University of California at San Diego (Visiting Professor)
2013~2014 : Georgia Institute of Technology (Visiting Professor)