JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 26, No. 4: 548~555, Apr. 2022

NIST P-521 타원곡선을 지원하는 고성능 ECC 프로세서

양현준[1] · 신경욱[2*]

A High-Performance ECC Processor Supporting NIST P-521 Elliptic Curve

Hyeon-Jun Yang[1] · Kyung-Wook Shin[2*]

[1]Graduate Student, Department of Electronic Engineering, Kumoh National Institute of Technology, Gumi, 39177 Korea
[2*]Professor, School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, 39177 Korea

요 약

본 논문은 타원곡선 디지털 서명 알고리듬 (Elliptic Curve Digital Signature Algorithm; ECDSA)의 핵심 연산으로 사용되는 타원곡선 암호 (Elliptic Curve Cryptography; ECC)의 하드웨어 구현에 대해 기술한다. 설계된 ECC 프로세서는 NIST P-521 곡선 상의 8가지 연산 모드 (점 연산 4가지, 모듈러 연산 4가지)를 지원한다. 점 스칼라 곱셈 (PSM)에 필요한 연산량을 최소화하기 위해 5가지 PSM 알고리듬과 4가지 좌표계에 따른 연산 복잡도 분석을 토대로 radix-4 Booth 인코딩과 수정된 자코비안 좌표계를 적용하여 설계하였다. 모듈러 곱셈은 수정형 3-Way Toom-Cook 정수 곱셈과 수정형 고속 축약 알고리듬을 적용하여 구현되었다. 설계된 ECC 프로세서는 xczu7ev FPGA 디바이스에 구현하여 하드웨어 동작을 검증하였다. 101,921개의 LUT와 18,357개의 플립플롭 그리고 101개의 DSP 블록이 사용되었고, 최대 동작주파수 45 MHz에서 초당 약 370번의 PSM 연산이 가능한 것으로 평가되었다.

ABSTRACT

This paper describes the hardware implementation of elliptic curve cryptography (ECC) used as a core operation in elliptic curve digital signature algorithm (ECDSA). The ECC processor supports eight operation modes (four point operations, four modular operations) on the NIST P-521 curve. In order to minimize computation complexity required for point scalar multiplication (PSM), the radix-4 Booth encoding scheme and modified Jacobian coordinate system were adopted, which was based on the complexity analysis for five PSM algorithms and four different coordinate systems. Modular multiplication was implemented using a modified 3-Way Toom-Cook multiplication and a modified fast reduction algorithm. The ECC processor was implemented on xczu7ev FPGA device to verify hardware operation. Hardware resources of 101,921 LUTs, 18,357 flip-flops and 101 DSP blocks were used, and it was evaluated that about 370 PSM operations per second were achieved at a maximum operation clock frequency of 45 MHz.

키워드 : 타원곡선 암호, 점 스칼라 곱셈, Booth 인코딩, 자코비안 좌표계, 타원곡선 디지털서명

Keywords : Elliptic curve cryptography, point scalar multiplication, Booth encoding, Jacobian coordinate system, ECDSA

Received 10 March 2022, **Revised** 16 March 2022, **Accepted** 23 March 2022

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

Open Access http://doi.org/10.6109/jkiice.2022.26.4.548 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/licenses/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

블록체인 (block chain)의 트랜잭션 검증과 자동차의 자율주행을 위한 V2X (vehicle-to-X) 통신 보안에는 타원곡선 디지털 서명 알고리듬 (elliptic curve digital signature algorithm; ECDSA) 프로토콜이 사용된다 [1, 2]. ECDSA의 서명 생성과 검증 과정은 해시 함수, 타원곡선 점 연산, 모듈러 연산, 난수 생성이 사용된다. V2X 통신은 주변 차량과 인프라 등으로부터 수신되는 많은 양의 메시지를 검증해야 하므로, 수신된 데이터를 빠르게 검증하는 것이 중요하며, V2X 통신에서는 초 당 1,000회 정도의 ECDSA 서명 검증이 요구되는 것으로 알려지고 있다 [3]. ECC의 핵심 연산인 점 스칼라 곱셈 (point scalar multiplication; PSM)은 ECDSA 서명 검증 과정에서 가장 많은 시간을 소요하는 연산이므로 PSM의 고속 연산이 중요하다.

PSM을 구현하는 다양한 알고리듬이 사용되고 있으며, 선택된 알고리듬에 따라 연산량, 부채널 공격 내성이 영향을 받는다. PSM을 구성하는 점 덧셈 (point addition; PA)과 점 두배 (point doubling; PD) 연산은 모듈러 연산을 기반으로 하며, 사용된 좌표계에 따라 모듈러 곱셈과 모듈러 나눗셈 (또는 모듈러 역원)의 연산 횟수가 달라진다. PA와 PD 연산에는 모듈러 곱셈이 사용되므로, 모듈러 곱셈은 PSM의 핵심 연산이다 [4]. 모듈러 곱셈은 정수 곱셈과 축약 연산으로 구성되며, 사용된 알고리듬에 따라 연산기의 회로 복잡도, 소요 클럭 사이클이 결정되므로 설계 사양에 맞는 적절한 알고리듬을 선택하는 것이 중요하다.

ECC 프로세서는 PSM 알고리듬, 좌표계, 모듈러 곱셈 알고리듬 등에 의해 성능이 결정된다. 본 논문의 고성능 ECC 프로세서는 radix-4 Booth 인코딩 (R4BE)을 적용한 PSM 알고리듬, 수정된 자코비안 좌표계, 수정형 3-Way Toom-Cook 알고리듬과 수정형 고속 축약 알고리듬 기반의 모듈러 곱셈 알고리듬을 사용하여 설계되었으며, P-521 곡선 상의 8가지 동작 모드를 지원한다. 2장에서는 타원곡선 암호, PSM 알고리듬, 수정된 자코비안 좌표계에 대해 설명하며, 3장에서는 수정된 자코비안 좌표계 상의 PA와 PD 연산 및 모듈러 연산 알고리듬에 대해 설명한다. 4장은 고성능 ECC 프로세서 설계에 대해 기술하고, 5장에서 검증 및 성능 평가를 설명한 후, 6장에서 결론을 맺는다.

Ⅱ. 타원곡선 암호

2.1. 타원곡선 암호

ECC는 타원곡선 상의 점 연산을 이용한 공개키 암호 방식이다. 공개키 암호는 개인키와 공개키로 구성된 암호이며, ECC 외에 RSA가 존재한다. ECC는 RSA보다 작은 키 크기를 사용해도 동등한 수준의 보안을 제공하는 장점이 있다. ECC의 PSM은 식 (1)로 표현되며, 점 P를 k번 더해 점 Q가 얻어진다. 이때 k는 개인키, Q는 공개키로 사용된다. PSM은 타원곡선 상의 점 이동을 의미하며, PA와 PD 연산의 반복으로 수행된다. 타원곡선 상의 PA, PD 연산은 유한체 (finite filed) 곱셈, 덧셈, 나눗셈, 역원 등의 연산으로 구현된다. 유한체 $GF(p^m)$은 유한한 개수의 원소를 갖는 체를 의미한다. 유한체 연산의 결과는 유한체 내에만 존재하며, p가 소수이고 $m=1$인 경우를 소수체 (prime field), p가 2이고 m은 양의 정수인 경우를 이진체 (binary field)라고 한다. 본 논문의 설계사양인 P-521 타원곡선은 소수체 상의 곡선이며, 식 (2)와 같이 정의된다. a, b는 타원곡선 파라미터이며, 정의된 소수체에 따라 달라진다.

$$Q = k \cdot P \tag{1}$$

$$y^2 = x^3 + ax + b,\ (4a^3 + 27b^2 \neq 0) \tag{2}$$

2.2. 점 스칼라 곱셈 알고리듬

식 (1)의 PSM을 계산하기 위한 다양한 방법들이 제안되었으며, Right-to-Left (R-L), Left-to-Right (L-R), binary Non-Adjacent Format (NAF), 몽고메리 래더 (Montgomery ladder; ML), R4BE 등이 사용된다 [5,6]. 표 1은 P-521 타원곡선에 대해 PSM 알고리듬에 따른 PA와 PD 연산 횟수를 보인 것이며, 본 논문에서는 PA의 연산 횟수가 가장 적은 R4BE를 PSM 알고리듬으로 사용하였다. Radix-4 Booth 인코딩 [7]을 이용하는 R4BE 알고리듬은 다른 PSM 알고리듬과 비교하여 PD 연산 횟수는 비슷하지만 PA 연산이 약 50% 감소하는 장점을 갖는다. 그림 1은 R4BE를 적용한 PSM의 슈도코드이

Table. 1 Comparison of the number of point operations according to PSM algorithms

	R4BE	ML	MML, R-L, L-R
PA	261	520	521
PD	522	521	521

$Input: k, P(x, y), (0 \le k < 2^L), n = \left\lceil \frac{1}{2} L \right\rceil$
$b_i \in \{0, \pm 1, \pm 2\}, Booth\ Encoding\ of\ k$
$output: Q = k \cdot P$

01: $(b_{n-1}, b_{n-2}, \cdots, b_2, b_1, b_0) \leftarrow \boldsymbol{BE}(k)$;
02: $Q_0 \leftarrow 0$; $Q_1 \leftarrow 0$; $Q_2 \leftarrow P$;
03: **for** $i = 0$ **to** $n - 1$ **do**
04: $Q_1 \leftarrow 2Q_2$;
05: **if** $b_i = 0$ **then**
06: $Q_0 \leftarrow Q_0 + 0$;
07: **else if** $b_i = 1$ **then**
08: $Q_0 \leftarrow Q_0 + Q_2$;
09: **else if** $b_i = -1$ **then**
10: $Q_0 \leftarrow Q_0 - Q_2$;
11: **else if** $b_i = 2$ **then**
12: $Q_0 \leftarrow Q_0 + Q_1$;
13: **else if** $b_i = -2$ **then**
14: $Q_0 \leftarrow Q_0 - Q_1$;
15: **end if**
16: $Q_2 \leftarrow 2Q_1$;
17: **end for**
18: **Return** $Q = Q_0$

Fig. 1 Pseudo-code of PSM using R4BE algorithm

다. 단계-1은 키 값 k를 Booth 인코딩하여 signed digit b_i를 생성한다. 단계-2에서 초기값을 할당한 뒤 n-회의 반복 연산을 통해 PSM 결과가 얻어진다. 유한체 크기 L이 짝수인 경우 $n = (L+2)/2$이고, 홀수인 경우에는 $n = (L+1)/2$이다. 예를 들어, 체 크기가 521-비트인 경우 $n = (521+1)/2 = 261$이 된다. 1회 루프에는 두 번의 PD 연산과 한 번의 PA 연산이 사용된다. 단계-4의 PD 연산은 $b_i = \pm 2$에 해당하는 단계-11 ~ 단계-14의 PA 또는 PS (point subtraction) 연산에 사용될 값을 생성한다. PS 연산은 점의 x-축 대칭과 PA 연산으로 구현된다. 단계-16의 PD 연산은 다음 반복루프에서 $b_i = \pm 1$에 해당하는 단계-7 ~ 단계-10의 PA 연산에 사용될 값을 생성한다. $b_i = 0$인 경우의 단계-5 ~ 단계-6은 PA 연산이 필요 없지만, 키 값의 해밍 웨이트에 따른 부채널 공격 (side channel attack)을 방지하기 위해 PA 연산이 수행되도록 하였다. PSM은 총 n회의 PA 연산과 $2n$회의 PD 연산으로 계산된다.

2.3. 좌표계

PSM을 계산하기 위한 PA와 PD 연산은 적용되는 좌표계에 따라 유한체 연산의 복잡도가 달라진다. 일상에서 흔히 사용되는 2차원 (x,y) 아핀 (affine) 좌표계를 적용한 점 연산은 모듈러 나눗셈 (또는 모듈러 역원) 연산이 포함되므로 하드웨어 구현 시 소요 사이클 측면에서 매우 비효율적이다. 점 연산을 투영 좌표계로 변환해서 연산하면, 모듈러 역원 연산이 제거되므로 하드웨어 구현에 유리하여 많이 사용된다. 여러 가지 형태의 투영 좌표계가 존재하며, 대표적으로 표준 투영 (standard projective), 자코비안 (Jacobian), 처드노프스키 자코비안 (Chudnovsky Jacobian), 수정된 자코비안(modified Jacobian) 좌표계 등이 있다 [8].

Table. 2 Comparison of the number of modular multiplications according to PSM algorithms and coordinates

Coordinate	R4BE	ML	MML, R-L, L-R
Standard	9,918	13,532	13,546
Jacobian	9,396	13,530	13,546
Chudnovsky Jacobian	9,396	13,011	13,025
Modified Jacobian	9,135	14,048	14,067

PA와 PD 연산은 좌표계에 따라 계산 수식이 달라지며, 표 2는 PSM 알고리듬과 좌표계에 따라 PA와 PD 연산에 필요한 모듈러 곱셈 횟수 (즉, 연산복잡도)의 비교를 보이고 있다. PSM 알고리듬 측면에서는 R4BE의 연산복잡도가 약 30% 정도 작으며, 좌표계 측면에서는 수정된 자코비안 좌표계의 연산복잡도가 최소이다. 이는 수정된 자코비안 좌표계의 PA에 사용되는 모듈러 곱셈의 연산 횟수가 다른 좌표계들 보다 많지만, PD 연산의 모듈러 곱셈 횟수가 적기 때문이다. 한편, R4BE를 제외한 나머지 4개의 PSM 알고리듬에서는 수정된 자코비안 좌표계의 모듈러 곱셈 연산 횟수가 가장 많다. 본 논문에서는 이와 같은 연산복잡도 분석을 토대로 수정된 자코비안 좌표계를 적용하여 설계하였다.

수정된 자코비안 좌표계는 (X, Y, Z, aZ^4)의 4차원으로 표현된 좌표계이며, 3차원으로 표현된 자코비안 좌표계와 달리 aZ^4 항이 추가된 것이 특징이다. aZ^4의 경우 PD 연산을 기존의 자코비안 좌표계보다 빠르게 수행할 수 있도록 한다. 아핀 좌표계에서 수정된 좌표계로의 변환은 $(x, y) \rightarrow (X, Y, 1, a)$로 추가적인 연산 없이 처리된다. 연산을 완료한 후, 다시 아핀 좌표계로의 역변환은 $(x, y) = (X/Z^2, Y/Z^3)$의 연산을 통해 이루어지며, 역변환 과정에서 $1/Z^2$과 $1/Z^3$ 연산을 위해 모듈러 역원 (modular inversion) 연산이 필요하다.

Ⅲ. 점 연산 및 유한체 연산

3.1. 점 연산

본 논문의 ECC 프로세서는 PSM, PD, PA, 점 뺄셈 (PS) 연산을 지원한다. PSM은 PA와 PD 연산의 반복으로 구현되며, PD, PA는 좌표계에 따라 연산 수식이 달라진다. 본 논문에서 사용한 수정된 자코비안 좌표계의 경우 PA와 PD에 모듈러 곱셈이 각각 19, 8회 사용된다. 그림 2는 수정된 자코비안 좌표계의 PA 슈도코드이다. 곡선 상의 점 P_1, P_2에 대한 PA 연산 수식을 단계별로 정리하였으며, 단계를 나누는 기준은 모듈러 곱셈이다. PA은 두 개의 점 P_1, P_2와 곡선 계수 a를 입력으로 받으며 출력은 곡선 상의 점 R을 얻는다. 점 뺄셈은 빼고자 하는 점의 Y좌표를 음수로 만들어 준 뒤 PA 연산으로 구현된다. 20개의 단계로 구성되며, $P_1 \neq \pm P_2$의 조건을 만족해야한다. 그림 3은 PD 연산의 슈도코드이다. 단계-1~9로 구성되며, PA와 동일한 기준으로 분류되었다. 점 P_1을 입력으로 받아 PD 연산을 수행하며 곡선 상의 점 R을 결과로 출력한다. 모든 단계(단계-1~단계-9)에서 모듈러 곱셈과 모듈러 덧셈이 동시에 수행된다. 모듈러 덧셈의 경우 모듈러 곱셈보다 소요되는 시간이 짧으므로, 하나의 모듈러 곱셈이 수행되는 동안 2회의 모듈러 덧셈도 연산 가능하다.

$Input: P_1(X_1, Y_1, Z_1, aZ_1^4),\ P_2(X_2, Y_2, Z_2, aZ_2^4), a$
$output: R = P_1 + P_2 = (X_3, Y_3, Z_3, aZ_3^4)$

$01: A_2 \leftarrow Z_2 \times Z_2;$
$02: A_3 \leftarrow Z_1 \times Z_1;$
$03: A_1 \leftarrow A_2 \times X_1;$
$04: A_0 \leftarrow A_3 \times X_2;$
$05: A_2 \leftarrow A_2 \times Z_2; \quad A_4 \leftarrow A_0 - A_1;$
$06: A_3 \leftarrow A_3 \times Z_1;$
$07: A_2 \leftarrow A_2 \times Y_1;$
$08: A_3 \leftarrow A_3 \times Y_2;$
$09: A_0 \leftarrow A_4 \times A_4; \quad A_5 \leftarrow A_3 - A_2;$
$10: A_1 \leftarrow A_1 \times A_0;$
$11: A_3 \leftarrow A_4 \times A_0;$
$12: A_3 \leftarrow A_3 \times A_2; \quad A_0 \leftarrow A_3 + A_1;$
$13: A_2 \leftarrow A_5 \times A_5; \quad A_0 \leftarrow A_1 + A_0;$
$14: A_2 \leftarrow A_4 \times Z_2; \quad A_0 \leftarrow A_2 - A_0;$
$15: A_2 \leftarrow A_2 \times Z_1; \quad A_1 \leftarrow A_1 - A_0;$
$16: A_1 \leftarrow A_5 \times A_1;$
$17: A_4 \leftarrow A_2 \times A_2; \quad A_1 \leftarrow A_1 - A_3;$
$18: A_3 \leftarrow A_4 \times A_4;$
$19: A_3 \leftarrow a \times A_3;$
$20: \textbf{\textit{Return}}\ R = (A_0, A_1, A_2, A_3)$

Fig. 2 Pseudo-code of PA using modified Jacobian coordinate

$Input: P_1(X_1, Y_1, Z_1, aZ_1^4)$
$output: R = 2P_1 = (X_3, Y_3, Z_3, aZ_3^4)$

$01: D_4 \leftarrow X_1 \times X_1; \quad D_5 \leftarrow Y_1 + Y_1;$
$02: D_3 \leftarrow D_5 \times Y_1; \quad D_4 \leftarrow 2D_4 + D_4;$
$03: D_1 \leftarrow D_3 \times X_1; \quad D_4 \leftarrow D_4 + (aZ_1^4);$
$04: D_0 \leftarrow D_4 \times D_4; \quad D_1 \leftarrow D_0 + D_1;$
$05: D_3 \leftarrow D_3 \times D_3; \quad D_0 \leftarrow D_0 - 2D_1;$
$06: D_2 \leftarrow D_5 \times Z_1; \quad D_3 \leftarrow D_3 + D_3;$
$\quad D_1 \leftarrow D_1 - D_0;$
$07: D_1 \leftarrow D_4 \times D_1; \quad D_5 \leftarrow D_3 + D_3;$
$08: D_3 \leftarrow D_5 \times (aZ_1^4); \quad D_1 \leftarrow D_1 - D_3;$
$\quad D_4 \leftarrow -D_1$
$09: \textbf{\textit{Return}}\ R = (D_0, D_1, D_2, D_3)$

Fig. 3 Pseudo-code of PD using modified Jacobian coordinate

3.2. 모듈러 연산

본 논문의 ECC 프로세서는 모듈러 곱셈 (modular multiplication; MM), 모듈러 역원 (modular inversion; MI), 모듈러 덧셈 (modular addition; MA), 모듈러 뺄셈 (modular subtraction; MS)의 4가지 모듈러 연산을 지원한다. 모듈러 곱셈은 정수 곱셈 후 모듈러 축약을 연산하는 2단계 방법을 적용하여 구현하였으며, 정수 곱셈은 3-Way Toom-Cook (3WTC)정수 곱셈을 수정한 m3WTC 알고리듬을 고안하여 적용하였고, 축약 연산에는 고속 축약 알고리듬을 521-비트 곱셈에 적합하도록 수정한 mFRed 알고리듬을 고안하여 적용하였다 [9]. 3WTC 기반 모듈러 곱셈은 결과값에 3이 곱해져 출력되므로, 최종 모듈러 곱셈 결과를 얻기 위해서는 $1/3$을 곱하는 추가 연산과정이 필요하다. 그러나 PSM의 PA, PD 연산에서는 모듈러 곱셈이 반복적으로 연산되므로, 매 모듈러 곱셈 결과에 $1/3$을 곱하는 연산을 반복하는 것은 비효율적이다. 본 논문에서는 이를 개선하기 위해 Toom-Cook (TC) 도메인 개념을 적용하였다. TC 도메인의 장점은 피승수와 승수 모두 TC 도메인 상의 데이터인 경우, 결과값도 TC 도메인에 존재한다는 것이다. 따라서 TC 도메인을 적용할 경우 $1/3$을 곱하는 연산 과정이 필요치 않으며, 점 연산의 처음과 마지막에서 각각 TC 도메인 매핑과 리매핑 연산이 포함된다.

좌표계 변환에 필요한 모듈러 역원 연산은 수정된 double plus-minus 이진 모듈러 역원 알고리듬이 적용되

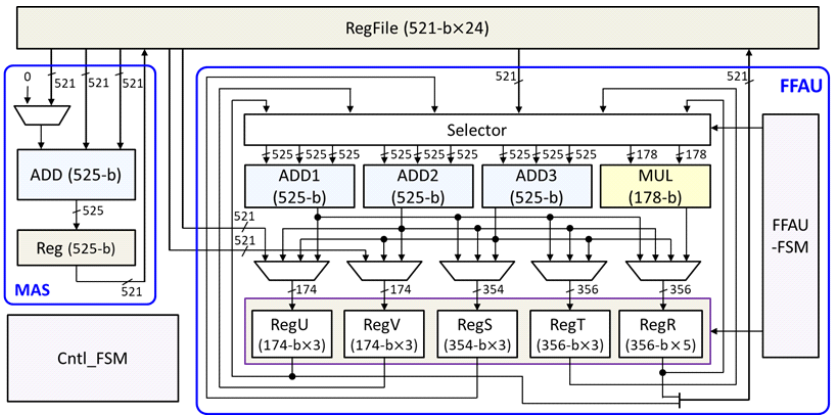

Fig. 4 Architecture of ECC processor

었다. 점 연산에 필요한 MA와 MS는 일반적인 덧셈, 뺄셈 연산으로 간단하게 구현되며, MI, MA, MS은 모두 가산기만으로 구현된다.

Ⅳ. ECC 프로세서 설계

4.1. ECC 프로세서 구조

ECC 프로세서는 P-521 곡선 상의 4가지 점 연산(PSM, PA, PD PS)과 4가지 모듈러 연산(MM, MI, MA, MS) 모드를 지원하도록 설계되었다. 그림 4는 ECC 프로세서의 내부 구조이며, 유한체 연산을 담당하는 FFAU 블록, 모듈러 가산기 MAS 블록 그리고 제어블록 Cntl_FSM으로 구성된다. FFAU 블록은 178-비트 곱셈기 1개와 525-비트 가산기 3개, 그리고 연산 중간결과 값을 저장하는 5개의 레지스터 블록과 데이터 선택기 등으로 구성되며, 모듈러 연산을 통해 점 연산을 수행한다. MAS 블록은 525-비트 가산기와 레지스터로 구성되며, 점 연산 모드에서 FFAU 블록의 연산성능 향상을 위한 보조적인 연산기능을 수행한다. 하나의 클록 사이클에 여러 개의 피연산자가 사용되는 FFAU 블록의 동작 특성을 고려하여 연산 중간결과 값의 저장을 위해 RAM 대신 레지스터를 사용했으며, 이를 통해 ECC 프로세서의 전체적인 연산성능을 극대화하였다.

4.2. ECC 프로세서의 연산처리 흐름

ECC 프로세서는 그림 5와 같이 6개의 처리단계로 나누어 동작하며, 연산모드에 따라 적용되는 처리단계가 달라진다. 4가지 점 연산의 경우 6개의 처리단계가 모두 사용되며, 모듈러 곱셈은 5개 그 외의 모듈러 연산(역원, 덧셈, 뺄셈)은 3개의 처리단계만 사용된다. 각 처리단계에서 수행되는 동작은 다음과 같다. 단계-1은 입력 데이터를 저장하며, 아핀 좌표계에서 수정된 자코비안 좌표계로의 좌표계 변환을 수행한다. 연산에 필요한 입력 데이터가 저장되며, 점 연산과 모듈러 곱셈의 경우 단계-2로 천이되며, 그 외의 연산모드는 단계-3으로 천이된다. 단계-2는 TC 도메인 매핑 단계이다. 점 연산의 경우 곡

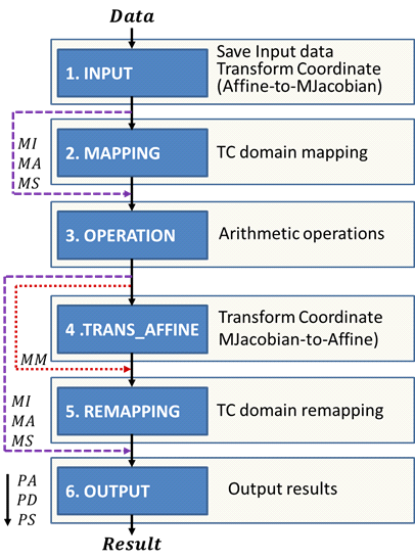

Fig. 5 Processing steps of ECC processor depending on operation modes

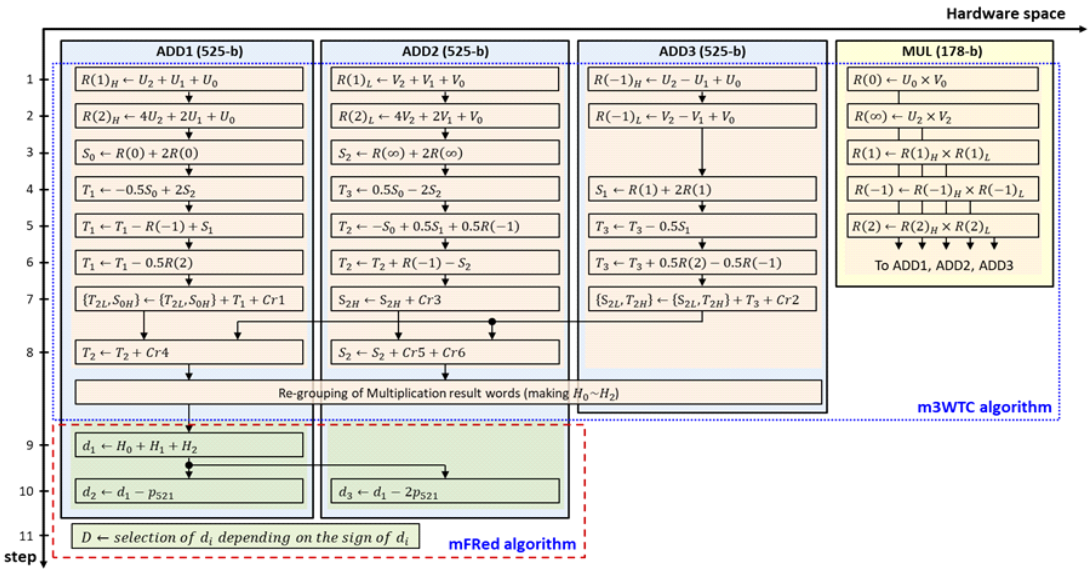

Fig. 6 Space-time mapping of sub-operations for MM onto FFAU

선 상의 점, 모듈러 곱셈인 경우 피승수와 승수를 TC 도메인으로 매핑한다. 매핑 동작에는 모듈러 곱셈이 사용된다. 매핑이 종료되면 단계-3으로 천이된다. 단계-3은 실제 연산이 수행되는 단계이며, 각 연산 모드별로 해당 알고리듬의 연산이 수행된다. 연산이 종료되면 점 연산의 경우 단계-4로 천이하며, 모듈러 곱셈은 단계-5, 그 외는 단계-6으로 천이한다. 단계-4는 좌표계 변환 단계이다. 수정된 자코비안 좌표계 상의 점을 아핀 좌표계로 변환하며 모듈러 역원 연산이 적용된다. 단계-5는 TC 도메인 상의 결과값을 일반 도메인으로 리매핑하는 단계이며, 이를 위해 모듈러 곱셈 연산이 적용된다. 단계-6은 연산 결과값을 oData_en 신호와 함께 출력한다. 점 연산의 경우 곡선 상의 좌표로 출력되므로 2개의 데이터가 출력된다.

그림 6은 단계-3에서 연산되는 모듈러 곱셈(MM)의 하위연산들이 FFAU 내부의 가산기 (ADD1, ADD2, ADD3)와 곱셈기 (MUL)에 할당되어 연산되는 과정을 space-time 매핑 관계로 보인 것이다. 3.2절에서 언급된 바와 같이, 모듈러 곱셈은 m3WTC 알고리듬과 mFRed 알고리듬에 의해 연산되며, 본 논문에서는 모듈러 곱셈에 11 클록 사이클이 소요된다.

Ⅴ. 검증 및 성능 평가

설계된 ECC 프로세서는 xczu7ev FPGA 디바이스의 PL 영역에 구현하여 하드웨어 동작을 검증하였다. 검증 플랫폼은 PC와 FPGA이며, PC는 연산에 필요한 데이터를 FPGA에 전송한다. FPGA는 결과값을 PC로 보내주며, PC는 수신된 결과값을 GUI 화면에 보여준다. PC와 FPGA의 데이터 통신은 UART 통신을 사용하였다. 그림 7은 PSM 연산의 검증 결과가 표시된 GUI 화면 캡쳐이며, 파이썬 소프트웨어 모델의 결과값과 FPGA의 결과값이 일치하는 것을 확인할 수 있다.

설계된 ECC 프로세서는 xczu7ev FPGA 디바이스에 구현되어 LUT 101,921개, 플립플롭 18,357개, DPS 블록 101개의 하드웨어 자원이 사용되었으며, 최대 동작 주파수는 45 MHz로 평가되었다 [10]. PSM 연산에 평균 121,419 클럭 사이클이 소요된다. 모듈러 역원 연산의 경우 피연산자의 값에 따라 소요 클럭 사이클 수가 달라지므로 평균 소요 사이클 수를 적용하였으며, 좌표계 변환에도 모듈러 역원 연산이 사용되므로 점 연산에도 평균 소요 사이클 수를 적용하여 평가했다.

표 3은 문헌에 발표된 ECC 프로세서와의 비교를 보이고 있다. 문헌 [11]의 사례는 본 논문의 ECC 프로세서에 비해 LUT가 약 10% 적고 DSP가 사용되지 않았지만,

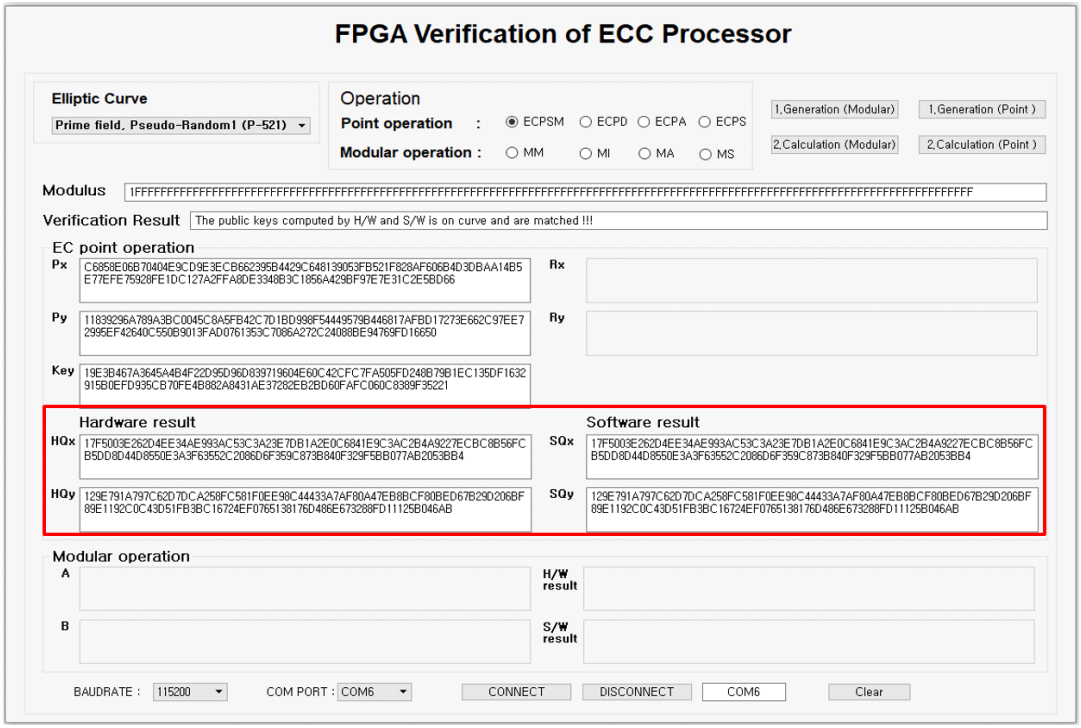

Fig. 7 Screenshot of FPGA verification result for PSM operation

연산성능이 약 20% 낮다. 문헌 [12]의 사례와 비교하면, 본 논문의 설계는 LUT를 약 3배 더 많이 사용하지만 DSP는 65% 적게 사용되었으며, 연산성능이 약 1.4배 우수한 것으로 나타났다. 문헌 [13]의 사례는 경량 하드웨어 구현의 장점은 있으나, 연산성능이 본 논문의 ECC 프로세서가 약 5배 더 높다. 결론적으로 본 논문의 ECC 프로세서는 고성능 ECC 연산이 필요한 응용분야에 적합한 것으로 평가된다.

Table. 3 Comparison of ECC processors

Design		[11]	[12]	[13]	This paper
FPGA device		Virtex-5	Virtex-6	Virtex-7	Xczu7ev
EC		P-521			
Area	LUT FF	90.8k NA	32.9k NA	3k 4.6k	102k 18.4k
	DSP	0	289	8 BRAMs	101
Frequency (MHz)		172	100	184	45
Latency (msec/PSM)		3.32	3.91	13.3	2.7
Throughput (PSM/sec)		301	255	75	370

Ⅵ. 결 론

설계된 ECC 프로세서는 NIST P-521 타원곡선 상의 8가지 연산모드를 지원하며, 점 스칼라 곱셈을 위해 R4BE 알고리듬, 고속 모듈러 곱셈을 위해 m3WTC와 mFRed 알고리듬, 수정된 double plus-minus 이진 모듈러 역원 알고리듬 그리고 수정된 자코비안 좌표계를 적용함으로써 전체 설계 수준에서 최적화를 이루었다. Xczu7ev FPGA 디바이스에 구현하여 성능을 평가한 결과 최대 동작 주파수는 45 MHz이며, PSM 연산에 121,419 클럭 사이클이 소요되어 초당 약 370번의 PSM을 연산할 수 있는 것으로 평가되었다. 본 논문에서는 하드웨어 동작 검증을 위해 FPGA 디바이스에 구현하였으나, 나노 급의 첨단 반도체 공정으로 제작하여 150 MHz 클록 주파수로 동작시키면, 초당 1,000번 이상의 PSM을 연산할 수 있어 자율주행 자동차의 통신 보안과 블록체인을 위한 보안 SoC에 탑재되어 ECDSA, ECDH, ECIEC 등 ECC 기반 보안 프로토콜의 하드웨어 구현에 핵심 IP로 사용될 수 있다.

ACKNOWLEDGEMENT

- This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2020R1I1A3A04038083)
- This paper was supported by Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (P0017011, HRD Program for Industrial Innovation)
- The authors are thankful to IDEC for EDA tool support.

REFERENCES

[1] H. Xiong, C. Jin, M. Alazab, K. -H. Yeh, H. Wang, T. R. R. Gadekallu, W. Wang, C. Su, "On the design of blockchain-based ECDSA with fault-tolerant batch verication protocol for blockchain-enabled IoMT," *IEEE Journal of Biomedical and Health Informatics*, p. 99, Sep. 2021.

[2] C. Hicks and F. D. Garcia, "A Vehicular DAA Scheme for Unlinkable ECDSA Pseudonyms in V2X," in *2020 IEEE European Symposium on Security and Privacy*, Genoa, pp. 460-473, 2020.

[3] M. Knežević, V. Nikov, and P. Rombouts, "Low-latency ECDSA signature verification - a road toward safer traffic," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 11, pp. 3257-3267, Nov. 2016.

[4] M. R. Hossain and M. S. Hossain, "Efficient FPGA Implementation of Modular Arithmetic for Elliptic Curve Cryptography," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, Cox'sBazar, Bangladesh, pp. 1-6, 2019. DOI: 10.1109/ECACE.2019.8679419.

[5] B. Rashidi, "A survey on hardware implementations of elliptic curve cryptosystems," *arXiv preprint arXiv:1710.08336, 2017*. [Online]. Available: https://arxiv.org/abs/1710.08336.

[6] S. H. Lee, "A Lightweight ECC Processor Supporting Dual Field Elliptic Curves of GF(p) and GF(2^m)," M. S. thesis, Graduate School of Kumoh National Institute of Technology, Jun. 2019.

[7] S. Moon, "Elliptic Curve Scalar Point Multiplication Using Radix-4 Modified Booth's Algorithm," in *Journal of the Korea Institute of Information and Communication Engineering*, vol. 8, no.6, pp. 80-83, Oct. 2004.

[8] H. Cohen, A. Miyaji, and T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates," in *International Conference on the Theory and Application of Cryptology and Information Security*, Berlin, Heidelberg, vol. 1514, pp. 51-65, Oct. 1998.

[9] H. J. Yang and K. W. Shin, "A 521 bits high-performance modular multiplier using 3-way Toom-Cook multiplication and fast reduction algorithm," *Journal of the Korea Institute of Information and Communication Engineering*, vol. 25, no. 12, pp. 1882-1889, Dec. 2021.

[10] H. J. Yang, "A Security SoC embedded with High-Performance ECC Processor," M. S. thesis, Graduate School of Kumoh National Institute of Technology, Feb. 2022.

[11] Y. A. Shah, K. Javeed, S. Azmat, and X. Wang, "A high-speed RSD-based flexible ECC processor for arbitrary curves over general prime field," *International Journal of Circuit Theory and Applications*, vol. 46, no. 10, pp. 1858-1878, Jun. 2018.

[12] 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.

[13] A. Salman, A. Ferozpuri, E. Homsirikamol, P. Yalla, J. -P. Kaps, and K. Gaj, "A scalable ECC processor implementation for high-speed and lightweight with side-channel countermeasures," in *2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, Cancun, pp. 1-8, Dec. 2017.

양현준(Hyeon-Jun Yang)

2020 : BS degree in Electronic Engineering, Kumoh National Institute of Technology.
2022 : Master degree in Electronic Engineering, Kumoh National Institute of Technology.

신경욱(Kyung-Wook Shin)

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, 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)