DOI: 10.3745/KIPSTC.2008.15-C.2.133

타원곡선 암호 시스템의 고속 구현을 위한 VLSI 구조

김 창 훈†

요 약

본 논문에서는 $GF(2^{163})$상의 고속 타원곡선 암호 프로세서를 제안한다. 제안한 암호 프로세서는 타원곡선 정수 곱셈을 위해 수정된 Lóez-Dahab Montgomery 알고리즘을 채택하고, $GF(2^{163})$상의 산술 연산을 위해 가우시안 정규 기저(Gaussian Normal Basis: GNB)를 이용한다. 높은 처리율을 위해 López-Dahab 방식에 기반한 규칙적인 주소화 방식의 병렬 타원곡선 좌표 덧셈 및 배 연산 알고리즘을 유도하고 $GF(2^{163})$상의 연산을 수행하는 두 개의 워드-레벨 산술 연산기(Arithmetic Unit: AU)를 설계한다. 제안된 타원곡선 암호 프로세서는 Xilinx사의 XC4VLX80 FPGA 디바이스에 구현되었으며, 24,263개의 슬라이스를 사용하고 최대 동작주파수는 143MHz이다. 제안된 구조를 Shu 등의 하드웨어 구현과 비교했을 때 하드웨어 복잡도는 약 2배 증가 하였지만 4.8배의 속도 향상을 보인다. 따라서 제안된 타원곡선 암호 프로세서는 네트워크 프로세서와 웹 서버등과 같은 높은 처리율을 요구하는 타원곡선 암호시스템에 적합하다.

키워드 : 타원곡선 암호시스템, 가우시안 정규기저, 유한체, VLSI

VLSI Architecture for High Speed Implementation of Elliptic Curve Cryptographic Systems

Chang Hoon Kim†

ABSTRACT

In this paper, we propose a high performance elliptic curve cryptographic processor over $GF(2^{163})$. The proposed architecture is based on a modified López-Dahab elliptic curve point multiplication algorithm and uses Gaussian normal basis for $GF(2^{163})$ field arithmetic. To achieve a high throughput rates, we design two new word-level arithmetic units over $GF(2^{163})$ and derive a parallelized elliptic curve point doubling and point addition algorithm with uniform addressing based on the López-Dahab method. We implement our design using Xilinx XC4VLX80 FPGA device which uses 24,263 slices and has a maximum frequency of 143MHz. Our design is roughly 4.8 times faster with 2 times increased hardware complexity compared with the previous hardware implementation proposed by Shu. et. al. Therefore, the proposed elliptic curve cryptographic processor is well suited to elliptic curve cryptosystems requiring high throughput rates such as network processors and web servers.

Key Words : Elliptic Curve Cryptosystem, Gaussian Normal Basis, Finite Field, VLSI

1. 서 론

1980년대 중반 Victor Miller[1]와 Neal Kobliz[2]에 의해 제안된 타원곡선 암호 시스템(ECC: Elliptic Curve Cryptographic system)은 RSA나 ElGamal과 같은 다른 암호 시스템에 비해 현저히 작은 키를 사용하면서(약 1/6 정도) 동일한 안전도를 가진다. 따라서 ECC는 최근 학계나 산업계로부터 많은 관심을 모으고 있다. 작은 키를 사용한다는 것은 계산 시간, 전력 소모, 저장 공간의 감소를 의미하기 때문에 현재까지 ECC의 소프트웨어[3,4] 및 하드웨어[5-13,21-23] 구현에 관한 많은 연구결과가 발표되었다. 또한 최근 IEEE 1363[14] 및 NIST(National Institute of Standards and Technology)[15]는 공개키 암호 시스템을 위해 ECC에 기반한 타원곡선 전자서명 알고리즘(Elliptic Curve Digital Signature Algorithm: ECDSA)을 표준으로 채택하였다.

ECC의 하드웨어 구현을 위해 사용되는 유한체는 $GF(p)$, $GF(2^m)$이 있으며 여기서 p는 소수이다. 특히 $GF(2^m)$은 $GF(2)$의 m차원 확장 벡터로서 산술 연산에 있어 캐리가 발생하지 않기 때문에 하드웨어 구현에 적합하다.

$GF(2^m)$상의 타원곡선 암호 프로세서는 기저의 선택에 따라 구조 및 성능이 좌우된다. 대표적인 원소 표기법으로 다항식 기저(Polynomial Basis: PB)와 정규 기저(Normal Basis: NB)가 있다. 각 기저표기법은 장점과 단점을 가지는데, NB

※ 본 연구는 정보통신부 및 정보통신연구진흥원의 지원을 받아 수행되었음. (08-기반-13, IT특화연구소 : "유비쿼터스 신기술 연구센터" 설립 및 운영)
† 정 회 원 : 대구대학교 컴퓨터·IT 공학부 전임강사
논문접수 : 2007년 9월 3일, 심사완료 : 2008년 3월 4일

를 이용할 경우 A^{2^s}연산을 간단한 s-비트 순환 쉬프트 연산으로 구현할 수 있는 반면 곱셈 연산이 복잡하다. PB를 이용한 AU의 하드웨어 구현은 서로 다른 m에 대해 높은 규칙성 및 확장성을 가진다. PB의 이러한 장점 때문에 NB에 비해 더 많은 연구결과가 발표되었다[5-9].

GNB는 NB의 특별한 경우로서 PB와 함께 IEEE 1363[14], NIST[15] 등의 다양한 표준으로 채택되었으며, 8로 나누어 떨어지지 않는 모든 양의 정수 m에 대하여 존재한다[11]. GNB는 정수 k에 의해 결정되며, GNB 타입 k라 부른다. 여기서 GNB 곱셈기의 속도 및 하드웨어 복잡도는 k에 의해 결정된다. 다시 말해서 주어진 m을 위한 GNB가 하나이상 존재할 때, 최소 k에 의해 효율적인 $GF(2^m)$상의 곱셈기를 구현할 수 있다. Kwon 등[17]은 $GF(2^m)$상에서 GNB를 이용한 효율적인 비트-레벨 곱셈 알고리즘을 유도하였으며, GNB 타입 2, 4의 경우에 대해 VLSI 구조를 제안하였다.

본 논문에서는 $GF(2^m)$상의 고속 타원곡선 암호 프로세서를 제안한다. 제안한 암호 프로세서는 타원곡선 정수 곱셈을 위해 수정된 López-Dahab Montgomery 알고리즘을 채택하고, $GF(2^m)$상의 산술 연산을 위해 GNB를 이용한다. 제안된 구조의 세 가지 주요 특징은 다음과 같다. 1) 워드-레벨 곱셈기에 기반한 고속의 AU를 사용, 2) 규칙적인 주소화 방식의 병렬화된 좌표 덧셈 및 배 연산기 설계, 3) GNB 표기법의 장점을 최대한 활용. 위의 세 가지 기법을 적용한 제안된 구조는 기존에 제안된 하드웨어 구현과 비교할 때 상당한 속도 향상을 보인다. 뿐만 아니라 제안된 구조는 모듈성과 단순한 제어 구조를 가지기 때문에 VLSI 구현에 매우 적합하다.

2. GNB를 이용한 $GF(2^m)$상의 워드-레벨 곱셈기

본 절에서는 GNB를 이용한 $GF(2^m)$상의 워드-레벨 곱셈기를 설계한다. 워드-레벨 곱셈기는 Kwon 등[17]이 제안한 비트-레벨 곱셈 알고리즘에 기반하며 선택된 워드의 크기에 따라 속도 및 면적 복잡도가 정해진다.

2.1 $GF(2^m)$상의 워드레벨 곱셈 알고리즘

[알고리즘 1] : $GF(2^m)$상의 비트-레벨 곱셈 알고리즘

Input : $A,\ B \in GF(2^m)$
Output : $D=(D_0, D_1, \cdots, D_{m-1}),\ D_s = c_s$ for all $0 \le s \le m-1$
1. for $0 \le t \le m-1$
2. for $0 \le s \le m-1$
3. $D_{s+t+1} \leftarrow y_{s,s+t} + D_{s+t}$
4. end for
5. end for
6. return D

[알고리즘 1]은 Kwon 등[17]이 제안한 $GF(2^m)$상의 비트-레벨 곱셈 알고리즘을 나타낸다. [알고리즘 1]로부터 비트-시리얼 또는 비트-패러럴 곱셈기를 쉽게 유도할 수 있지만 비트-시리얼은 속도가 너무 느리고 비트-패러럴은 큰 m(최소 163)을 요구하는 ECC와 같은 응용에는 적합하지 않다. 따라서 많은 타원곡선 암호 프로세서들은 비트-시리얼 및 비트-패러럴 곱셈기의 단점을 극복하기 위해 워드-레벨 곱셈기를 채택하였다. 워드-레벨 구조는 데이터 크기가 m-비트이고 워드의 크기가 w-비트이면 워드의 개수는 $L=\lceil m/w \rceil$이 되고 L클럭 사이클마다 결과를 출력한다. [알고리즘 1]로부터 [알고리즘 2]와 같은 GNB를 이용한 $GF(2^m)$상의 워드-레벨 곱셈 알고리즘을 얻을 수 있다.

[알고리즘 2]에서 $0 \le t \le L-2$일 때, 모든 $0 \le s \le m-1$에 대한 $w+1$개의 항$(y_{s,s+tw},\ y_{s,s+tw+1},\ \cdots,\ y_{s,s+tw+(w-1)},\ D_{s+tw-\gamma})$은 동시에 계산된다. 하지만 $t=L-1$일 때는 γ개의 블록이 $t=0$일 때의 원소와 중복되므로 모든 $0 \le s \le m-1$에 대한 $w-\gamma+1$개의 항$(y_{s,s+tw+1},\cdots,y_{s,s+tw+(w-1)-\gamma},D_{s+tw-\gamma})$이 동시에 계산된다. 즉, 고정된 s에 대한 마지막 출력값 D_s는 다음 식과 같이 연속적으로 계산된다.

$$D_s = \overbrace{\underbrace{y_{s,s}+y_{s,s+1}+\cdots+y_{s,s+(\omega-1)}}_{D_{s+\omega}}+y_{s,s+\omega}+y_{s,s+\omega+1}+\cdots+y_{s,s+2\omega-1}}^{D_{s+w}} + y_{s,s+2\omega}+\cdots+y_{s,s+m-1} = \sum_{i=0}^{m-1} y_{s,s+i} = c_s \quad (1)$$

NIST에서 권고하는 $GF(2^m)$상의 필드 크기($m \in 163, 233, 283, 409, 571$)가 소수이기 때문에 항상 $\gamma \neq 0$이며 연산의 마지막 반복에서 중복되는 계산을 피하기 위해 $m \cdot \gamma$ 항이 구분되어 수행되어야한다. 여기서 $m \cdot \gamma$ 항은 [알고리즘 2]의 $(L-1)$번째 단계에서 계산되기 때문에 규칙적인 하드웨어 구조 설계에 방해가 된다. 즉, 하드웨어 구조는 [알고리즘 2]의 단계 3, 8을 수행하기 위해 각각 다르게 설계되어야 한다. 본 논문에서는 이 문제를 해결하기 위해 L길이의 컨트롤 신호(111…10)를 사용한다. 보다 자세한 설명은 2.2절에 기술한다.

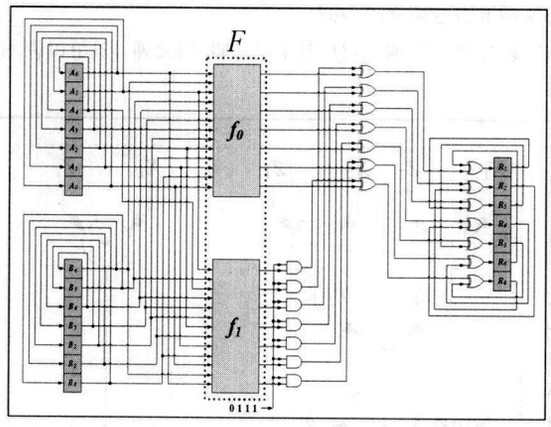
[알고리즘 2] $GF(2^m)$상의 GNB를 이용한 워드-레벨 곱셈 알고리즘

Input : $A,\ B \in GF(2^m)$
Output : $D=(D_0, D_1, \cdots, D_{m-1})$,
$D_s = c_s$ for all $0 \le s \le m-1$, where $AB = \sum_{s=0}^{m-1} c_s \alpha_s$.
Initial : $A \leftarrow (a_0, a_1, \cdots, a_{m-1}),\ B \leftarrow (b_0, b_1, \cdots, b_{m-1})$,
$D \leftarrow (D_0, D_1, \cdots, D_{m-1}) \leftarrow (0, 0, \cdots, 0)$.
1. for $t=0$ to $L-2$
2. for $s=0$ to $m-1$
3. $D_{s+(t+1)w-\gamma} \leftarrow y_{s,s+tw} + y_{s,s+tw+1} + \cdots + y_{s,s+tw+(w-1)} + D_{s+tw-\gamma}$
4. end for
5. end for
6. $t=L-1$
7. for $s=0$ to $m-1$
8. $D_{s+(t+1)w-\gamma} \leftarrow y_{s,s+tw} + y_{s,s+tw+1} + \cdots + y_{s,s+tw+(w-1)-\gamma} + D_{s+tw-\gamma}$
9. end for
10. return D

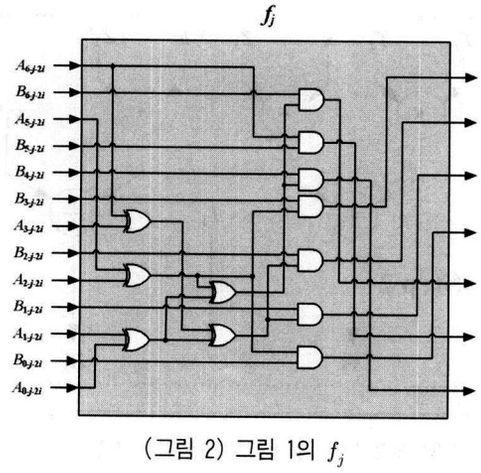
$\gamma = L \cdot \omega - m,\ L = \lceil \frac{m}{\omega} \rceil$

2.2 GNB 타입 4를 이용한 $GF(2^m)$상의 워드-레벨 곱셈기

예제로 $w=2$인 $GF(2^7)$상의 워드-레벨 곱셈기를 설계한다. [알고리즘 1]로 부터 곱셈 결과 $C=AB=\sum_{i=0}^{6} c_i\alpha_i$는 식 (2)와 같이 얻을 수 있다. 식 (2)의 밑줄은 $w=2$, a_{ijkl}는 $a_i+a_j+a_k+a_l$을 의미한다. (그림 1)은 $w=2$에 대하여 $GF(2^7)$상에서 GNB 타입 4를 이용한 $C=AB$에 대응되는 쉬프트 레지스터 회로이고, (그림 2)는 (그림 1)의 f_j 블록 구조를 나타낸다. 밑줄 친 두 개의 원소를 XOR 연산하여 레지스터에 저장하며 밑줄 친 원소의 첫 번째 항은 a_i의 공통된 항을 가지는 주대각 원소를 계산하는 f_0 블록이고, 두 번째 항은 주대각 원소의 벡터 a_i, b_i를 한 번 순환 쉬프트하여 계산하는 f_1 블록이다. 워드-레벨 구조에서는 한 클럭 사이클마다 $w=2$개의 원소를 연산하므로 A, B, R 레지스터는 w 크기만큼 순환 쉬프트하며 $L=4$ 클럭 사이클 후에 모든 연산이 끝나고 결과가 출력된다. m이 홀수이기 때문에 두 개의 원소씩 처리하면 마지막 클럭 사이클의 연산에 중복되는 원소가 나타나며 이를 회피해야 한다. 워드의 크기 w에 따라 중복되는 원소의 개수 $w-\gamma$도 다르게 나타나며 중복되는 원소의 개수만큼 마지막 클럭 사이클에서 연산을 회피해야 한다. (그림 1)의 곱셈기는 (1,1,1,0)의 제어 신호를 사용하여 마지막 제어 신호에서 중복되는 값을 회피한다. 또한 m/w가 정수가 아니기 때문에 곱셈 결과 레지스터 R_i는 정확한 c_i를 가지지 않는다. 따라서 정확한 출력을 위해 R_i는 γ만큼의 순환 쉬프트가 필요하다. (그림 1)의 워드-레벨 곱셈기는 처리 지연 시간이 $2T_A+(\lceil \log_2 k \rceil + \lceil \log_2 w+1 \rceil)T_X$이고 $(wm+\gamma m)$개의 AND 게이트, $(wm+w\frac{m-1}{2}(k-1))$개의 XOR 게이트, $3m$개의 Flip-Flop(FF)이 필요하다. 여기서 T_A는 2-입력 AND 게이트 지연시간이고 T_X는 2-입력 XOR 게이트의 지연시간이다.



(그림 1) GNB 타입 4를 이용한 $GF(2^7)$상의 워드-레벨 곱셈기



(그림 2) 그림 1의 f_j

$$
\begin{aligned}
c_0 &= \underline{(a_2+a_5)b_3+a_{0256}b_1}+a_{1456}b_6+(a_2+a_6)b_4+a_{1345}b_2+a_1b_0+a_{1236}b_5\\
c_1 &= (a_3+a_6)b_4+\underline{a_{1360}b_2+a_{2560}b_0}+(a_3+a_0)b_5+a_{2456}b_3+a_2b_1+a_{2340}b_6\\
c_2 &= (a_4+a_0)b_5+a_{2401}b_3+\underline{a_{3601}b_1+(a_4+a_1)b_6}+a_{3560}b_4+a_3b_2+a_{3451}b_0\\
c_3 &= (a_5+a_1)b_6+a_{3512}b_4+a_{4012}b_2+\underline{(a_5+a_2)b_0+a_{4601}b_5}+a_4b_3+a_{4562}b_1\\
c_4 &= (a_6+a_2)b_0+a_{4623}b_5+a_{5123}b_3+\underline{(a_6+a_3)b_1+a_{5012}b_6+a_5b_4}+a_{5603}b_2\\
c_5 &= (a_0+a_3)b_1+a_{5034}b_6+a_{6234}b_4+(a_0+a_4)b_2+a_{6123}b_0+\underline{a_6b_5+a_{6014}b_3}\\
c_6 &= \underline{(a_1+a_4)b_2}+a_{6145}b_0+a_{0345}b_5+(a_1+a_5)b_3+a_{0234}b_1+a_0b_6+\underline{a_{0125}b_4}
\end{aligned}
\tag{2}
$$

3. 타원곡선 암호 프로세서를 위한 AU

본 절에서는 타원곡선 암호 프로세서의 코어인 GNB상의 새로운 AU를 설계한다. 새로운 AU는 López-Dahab 알고리즘에 기반하며 여러 개의 곱셈기를 배치함으로써 López-Dahab 상수배 알고리즘을 고속으로 수행할 수 있다.

3.1 $GF(2^m)$상의 López-Dahab 상수배 알고리즘

E를 다음과 같이 정의된 $GF(2^m)$상의 타원곡선이라 하자.

$$E: y^2+xy=x^3+ax^2+b,\ a,b\in GF(2^m),\ b\neq 0 \tag{3}$$

$P=(x,y)$는 $E(GF(2^m))$상의 큰 소수 위수를 가지는 타원곡선 E상의 좌표이고, $k\geq 0$는 정수라 하자. k를 이진수 표현으로 나타내면 식 (4)와 같다.

$$
\begin{aligned}
&k=(k_{s-1}k_{s-2}\cdots k_1k_0)_2=k_{s-1}2^{s-1}+k_{s-2}2^{s-2}+\cdots+k_12+k_0,\\
&k_i=0,1,\ k_{s-1}=1
\end{aligned}
\tag{4}
$$

식 (3), (4)로부터 [알고리즘 3]의 López-Dahab 알고리즘 [18]을 이용하여 상수배 연산 kP를 수행할 수 있다. 곱셈 연산은 순차적으로 진행되며 반복 연산의 각 단계는 이전 단계의 결과 값을 이용한다. López-Dahab 타원곡선 정수 곱셈 알고리즘의 장점은 크게 두 가지가 있다. 1) 사영 좌표계상의 역원 연산을 수행하지 않는다. 2) 기본 연산이 매우 규칙적일 뿐만 아니라 분기조건이 없기 때문에 다른 알고리즘에 비해 타이밍, 전력, 전자기장 공격에 높은 면역을 가진다.

[알고리즘 3] López-Dahab 타원곡선 정수 곱셈 알고리즘

Input : $P=(x,y)\in E(GF(2^m))$, an integer $k\geq 0$
Output : $kP=(x_0,y_0)$
1. if $k=0$ or $x=0$, then stop and output $kP=O$ or P
2. $k\leftarrow(k_{l-1},\cdots,k_1,k_0)_2$
3. $(X_1,Z_1)\leftarrow(x,1),(X_2,Z_2)\leftarrow(x^4+b,x^2)$
4. for $i=s-2$ down to 0 do
5. $\quad Z_3\leftarrow(X_1Z_2+X_2Z_1)^2$

[알고리즘 3] López-Dahab 타원곡선 정수 곱셈 알고리즘(계속)

6. if $k_i = 1$ then
7. $X_1 \leftarrow xZ_3 + (X_1Z_2)(X_2Z_1)$, $Z_1 \leftarrow Z_3$,
. $X_3 \leftarrow X_2^4 + bZ_2^4$, $Z_2 \leftarrow X_2^2 Z_2^2$, $X_2 \leftarrow X_3$
8. else
9. $X_2 \leftarrow xZ_3 + (X_1Z_2)(X_2Z_1)$, $Z_2 \leftarrow Z_3$,
. $X_3 \leftarrow X_1^4 + bZ_1^4$, $Z_1 \leftarrow X_1^2 Z_1^2$, $X_1 \leftarrow X_3$
10. end if
11. end for
12. $x_0 \leftarrow \frac{X_1}{Z_1}$,
. $y_0 \leftarrow \frac{1}{x} \cdot (x + \frac{X_1}{Z_1})\left\{(x + \frac{X_1}{Z_1})(x + \frac{X_2}{Z_2}) + x^2 + y\right\} + y$
13. return $kP = (x_0, y_0)$

3.2 좌표 덧셈 및 배 연산을 위한 AU

[알고리즘 3]의 메인루프를 살펴보면, K_i에 상관없이 좌표 덧셈 및 배 연산이 동일하기 때문에 적당한 수정을 통해 동일한 연산 구조를 구성할 수 있다. 이러한 특성에 기반하여 Ansari 등[20]은 [알고리즘 4]와 같은 좌표 덧셈 및 배 연산을 위한 수정된 알고리즘을 제안하였다.

[알고리즘 4] 규칙적인 주소화 방식의 좌표 덧셈 및 배 연산

if $k_{l-2} = 1$ then
  Swap(X_1, X_2), Swap(Z_1, Z_2)
end if
for $i = l-2$ down to 0 do
  $Z_3 \leftarrow (X_1Z_2 + X_2Z_1)^2$,
  $X_2 \leftarrow xZ_3 + (X_1Z_2)(X_2Z_1)$, $Z_2 \leftarrow Z_3$,
  $X_1 \leftarrow X_1^4 + bZ_1^4$, $Z_1 \leftarrow X_1^2 Z_1^2$
  if $(l \neq 0$ and $k_l \neq k_{l-1})$ or $(l = 0$ and $k_l = 1)$ then
    Swap(X_1, X_2), Swap(Z_1, Z_2)
  end if
end for

[알고리즘 4]로부터 만약 하나의 곱셈기를 사용하면 여섯 번의 곱셈이 요구된다. 그러나 [알고리즘 4]를 살펴보면 계산되어 질 수 있는 세 개의 곱셈을 찾을 수 있다. 곱셈 $X_1Z_2(T_1)$, $X_2Z_1(T_2)$, X_1Z_1는 첫 번째 단계에, 곱셈 T_1T_2, xZ_3, bZ_1^4는 두 번째 단계에서 계산되어 질 수 있다. 이와 같은 특성으로부터 [알고리즘 5]와 같은 병렬화된 좌표 덧셈 및 배 연산 알고리즘을 얻을 수 있다.

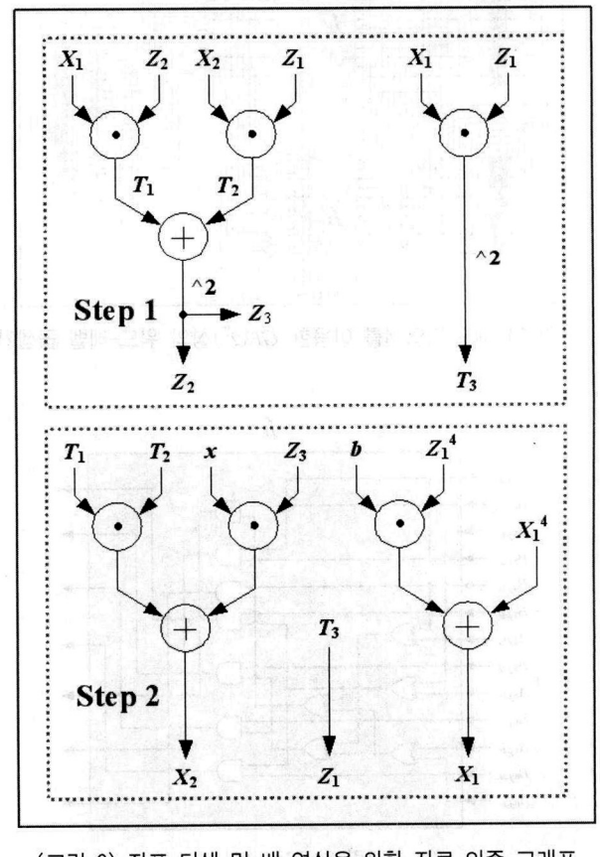
[알고리즘 5] 규칙적인 주소화 방식의 병렬화된 좌표 덧셈 및 배 연산

if $k_{l-2} = 1$ then
  Swap(X_1, X_2), Swap(Z_1, Z_2)
end if
for $i = l-2$ down to 0 do
  1. $T_1 \leftarrow (X_1Z_2)$, $T_2 \leftarrow (X_2Z_1)$, $Z_3 \leftarrow (T_1 + T_2)^2$,
    $T_3 \leftarrow (X_1Z_1)^2$, $Z_2 \leftarrow Z_3$
  2. $X_2 \leftarrow T_1T_2 + xZ_3$, $X_1 \leftarrow bZ_1^4 + X_1^4$, $Z_1 \leftarrow T_3$
  if $(l \neq 0$ and $k_l \neq k_{l-1})$ or $(l = 0$ and $k_i = 1)$ then
    Swap(X_1, X_2), Swap(Z_1, Z_2)
  end if
end for

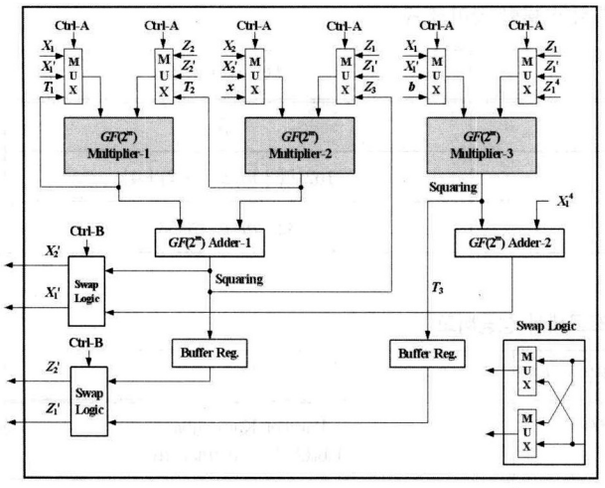
[알고리즘 5]에 해당하는 좌표 덧셈 및 배 연산을 위한 자료 의존 그래프는 (그림 3)과 같다. (그림 3)의 자료 의존 그래프를 기반으로 (그림 4)와 같은 새로운 AU를 얻을 수 있다. (그림 4)에 나타나듯이 AU는 (그림 3)의 자료 의존 그래프의 단계 1과 단계 2가 합쳐진 구조이다. (그림 4)에서 입력과 출력값은 2-비트 Ctrl-A와 1-비트 Ctrl-B 신호에 의해 제어되어지며 이는 매우 간단한 제어 구조를 유도한다. (그림 4)의 새로운 AU는 세 개의 곱셈기, 두 개의 덧셈기, 여섯 개의 3-to-1 멀티플렉서, 두 개의 버퍼 레지스터, 두 개의 스왑 논리 블록으로 구성된다. 참고로 $GF(2^m)$상에서 GNB를 사용할 경우 원소 A의 A^{2^s} 연산은 s-비트 만큼 순환 쉬프트 연산이기 때문에 이러한 연산을 위한 어떠한 논리 회로도 요구되지 않는다. (그림 4)에서 X_i, Z_i, X_1^4, Z_1^4와 b, x는 Resister File로부터 전달되어진다. 그리고 X_i'와 Z_i'는 임시 계산 결과를 의미하며 다음 반복을 위한 입력값으로 직접 사용되기 때문에 Register File에서 데이터 패치를 위한 클럭 사이클 수를 줄일 수 있다. 여기서 $i \in 1, 2$이다. 두 개의 버퍼 레지스터는 입력 타이밍을 수정하기위해 사용된다. 즉, 임시 결과값 Z_1, Z_2는 첫 번째 단계에 계산되어지지만 이 결과값들은 임시 결과값 X_1, X_2와 함께 다음 반복의 입력으로 동일한 시간에 입력된다. (그림 4)에서 López-Dahab 좌표 곱셈의 메인 루프에 소요되는 클럭 사이클은 $(l-1) \cdot 2$과 같다.

3.3 좌표변환을 위한 AU

좌표 덧셈 및 배 연산 알고리즘과 다르게 [알고리즘 3]에



(그림 3) 좌표 덧셈 및 배 연산을 위한 자료 의존 그래프



(그림 4) 좌표 덧셈 및 배 연산을 위한 AU

서의 좌표 변환 루틴은 역원 연산을 포함한다. 따라서 우선 $GF(2^m)$상의 역원 연산을 유도한다.

A를 $GF(2^m)$ 상의 원소라 하면, $GF(2^m)^{\times}$의 위수는 2^m-1이므로 $A^{2^m-1}=1$. 즉

$$A^{-1}=A^{2^m-2}=A^{2(2^{m-1}-1)} \tag{5}$$

이다. 식 (5)에 기반하여 Itoh-Tsujii[19]는 효율적인 역원 알고리즘을 제안하였다. Itoh-Tsujii 알고리즘은 다음과 같은 식에 기반한다.

$$2^s-1=\begin{cases}(2^{\frac{s}{2}}-1)(2^{\frac{s}{2}}+1), & \text{if } s=\text{even}\\ 2(2^{\frac{s-1}{2}}-1)(2^{\frac{s-1}{2}}+1), & \text{if } s=\text{odd}\end{cases} \tag{6}$$

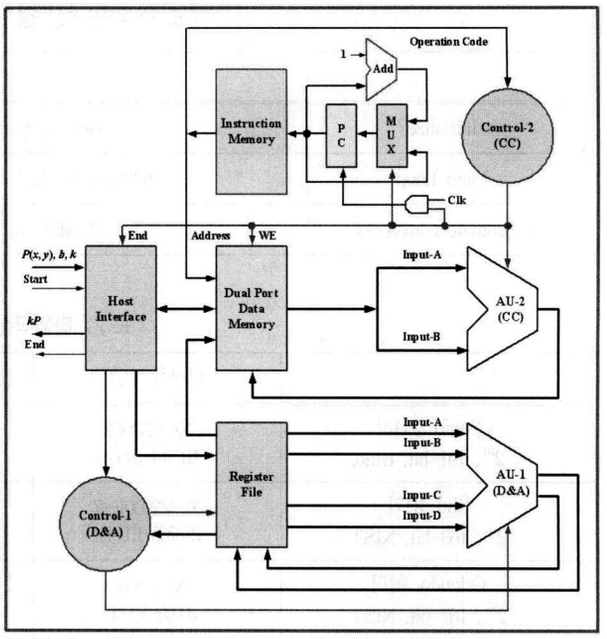
식 (6)을 이용하면 $A\in GF(2^m)$의 역원을 구하기 위해서 $\lfloor \log2(m-1)\rfloor+H(m-1)-1$개의 곱셈 연산이 필요하다. 여기서 H는 주어진 정수의 2진수 표현의 Hamming Weight이다. IEEE 1363[14] 표준 필드 크기 중 163에 대해서 예를 들면 다음과 같다. $A\in GF(2^{163})$이라 하자. 그러면 $A^{-1}=A^{2^{163}-2}$이다. 여기서

$$A^{-1}=A^{2(2^{81}+1)[2(2^{40}+1)(2^{20}+1)(2^{10}+1)(2^5+1)2(2^2+1)(2+1)+1+1]} \tag{7}$$

이다. 또한 $m=163$일 때, $m-1=162=(10100010)_2$이므로 필요한 곱셈의 개수는 $7+3-1=9$이다. 식 (7)를 바탕으로 그림 5와 같은 $GF(2^{163})$상의 좌표변환을 위한 새로운 VLSI 구조를 얻을 수 있다. (그림 5)에 나타나듯이 163-비트 3-to-1 멀티플렉서, 163-비트 9-to-1 멀티플렉서, 163-비트 2-to-1 멀티플렉서, 한 개의 $GF(2^{163})$상의 덧셈기, 8-비트 컨트롤 신호를 추가한다.

4. $GF(2^{163})$상의 타원곡선 암호 프로세서

본 절에서는 [알고리즘 5], 좌표 덧셈 및 배 연산을 위한 AU(AU-1), 좌표 변환을 위한 AU(AU-2)에 기반하여 완전한 $GF(2^{163})$상의 타원곡선 암호 프로세서를 설계한다.



(그림 5) $GF(2^{163})$상의 타원곡선 암호 프로세서 구조

4.1 $GF(2^{163})$상의 타원곡선 좌표 곱셈을 위한 데이터패스

본 논문에서 제안된 $GF(2^{163})$상의 새로운 타원곡선 암호 프로세서는 (그림 6)과 같다. 타원곡선 암호 프로세서는 크게 여덟 개의 블록 Host Interface, Data Memory, Register File, Instruction Memory, Control-1, Control-2, AU-1, AU-2로 구성된다. Host Interface는 Host kP를 위한 Start 신호와 함께 모든 파라미터를 Host 마이크로프로세서로부터 입력받아 타원곡선 암호 프로세서로 전달한다. Host 마이크로프로세서는 Host Interface로부터 kP 결과와 End 신호를 수신한다. Data Memory는 8×163-비트 Dual Port 메모리로 구성되고 Instruction Memory는 11-비트 워드의 13개 마이크로코드를 포함한다. 좌표 덧셈 및 배 연산의 고성능 구현을 위해 7×163-비트 Resister File을 추가한다. 레지스터 파일은 Host Interface로부터 데이터를 수신하고 데이터 메모리로 임시 계산 결과(X_1, X_2, Z_1, Z_2)를 전송한다. AU-1은 좌표 덧셈 및 배 연산을 위해 사용되어지며 Control-1 신호에 의해 제어되어진다. AU-2는 알고리즘 1과 같은 좌표변환을 수행한다. Control-2 신호는 명령어 메모리로부터 연산코드를 받아오며 AU-2, 데이터 메모리, Host Interface를 위한 컨트롤 신호를 생성한다. <표 1>은 $GF(2^{163})$상의 타원곡선 좌표 곱셈을 수행하기 위해 요구되는 클럭 사이클 수를 요약하였다. 여기서 $w=55$이고, $L=3$으로 가정하였다.

4.2 FPGA 구현 및 성능 분석

본 논문에서 제안한 타원곡선 암호 프로세서의 기능을 검증하고 성능을 분석하기 위해 Xilinx사의 XC4VLX80 FPGA 칩을 이용하여 구현하였다. 이를 위해 VHDL로 회로를 기술하였으며, Synopsys사의 FPGA Compiler II를 이용하여 회로합성을 하였다. 합성된 회로는 Xilinx사의 Foundation 소프

〈표 1〉 타원곡선 정수 곱셈을 위한 클럭 사이클 수

	명령어 수	클럭 사이클 수
Initialize	Add: 1, Sqr: 1	5
Main Loop	162(Mult: 6, Add: 3, Sqr: 1)	$162\{6(\lceil m/w \rceil + 2) + 4\}$
Coordinate conversion	Inv: 3, Mult: 5, Add: 5	$32\lceil m/w \rceil + 53$

〈표 2〉 타원곡선 암호 프로세서 성능비교

	디바이스/크기	클럭 주파수 / 속도	비고
Saqib 등[13] 2^m, 191-bit, trino.	XCV3200E 18,314 Slices	9.99 56 μs	Parallel Karatsuba 24 bRAMs, No final inv.
Shu 등[7] 2^m, 163-bit, NIST	XCV2000E-7 25,763 LUTs	68.9 48 μs	6 MSD Multipliers $D=32, 8$
Orlando 등[6] 2^m, 167-bit, NIST	XCV400E 3,002 LUTs	76.7 210 μs	167-bit × 16-bit Multi- plier, 10 bRAMs
Jarvinen 등[12] 2^m, 163-bit, NIST	XC2V8000-5 18,079 Slices	90.2 106 μs	Generator
Grabbe 등[11] 2^m, 233-bit, NIST	XC2V6000 19,440 LUTs	100 130 μs	Hybrid KOA Generator
Daneshbeh 등[9] 2^m, < 256-bit, Any	0.18 ^{1}m CMOS 35,000 Gates	700 734 μs	Div./Mult. Affine Coordinate.
Eberle 등[10] 2^m, < 256-bit, NIST 2^m, < 256-bit, Any	XCV2000E-7 20,068 LUTs	66.4 144 μs 302 μs	MSD $D=64$ Result for 163-bit Result for 163-bit
Satoh 등[5] 2^m, 160-bit, Any	0.13 μs CMOS 117,500 Gates	510.2 190 μs	MMM, $D=64$
Gura 등[8] 2^m, 163-bit	XCV2000E-7	66.5 143 μs	LSD Multiplier, $D=64$
Benaissa 등[21] 2^m, 160-bit	XCV2000E-7 19,508 LUTs	150 660 μs	D×D Multiplier, $D=64$
McIvor 등[22] GF(p), 256-bit	XC2VP125 15,755 Slices	45.68 3,860 μs	256(18×18) Multipliers
Chen 등[23] GF(p), 256-bit	0.13 CMOS 122,000 Gates	556 1,010 μs	Divider and Multiplier (Systolic Array)
This Work 2^m, 163-bit	XC4VLX80 24,363 Slices	143 10 μs	3 GNB Multipliers, $D=55$

트웨어를 이용하여 Place-and- Route를 수행하고 타이밍 분석을 하였다. 또한 Mentor Graphics사의 ModelSim을 이용하여 그 기능을 검증하고 리버트론사의 SoC(System-on-Chip) 테스트 보드를 이용하여 FPGA에 타원곡선 암호 프로세서를 구현 하였다. 테스트 보드는 인텔 PXA272 마이크로프로세서와 Xilinx XC4VLX80 FPGA칩을 탑재하고 있다.

<표 2>에 기존에 제안된 구조와 시간 및 하드웨어 면적 측면에서 성능을 비교하였다. <표 2>에 나타나듯이 제안된 각각의 구조는 서로 다른 하드웨어 플랫폼, 유한체, 체의 크기, 곱셈기를 사용할 뿐만 아니라 서로 다른 연구의 목적을 가지기 때문에 정확한 비교는 쉽지 않다. 예를 들면, [13, 7, 23]은 고속의 구현에 초점을 맞추었고 [5]에서 제안된 구조는 유한체와 필드 크기에 대해 유연성을 제공하는 것이 목적이다. 그러나 표 3에 나타나듯이 제안된 구조는 ASIC 구현을 포함해서 가장 빠른 설계이다. 보다 구체적으로 비교하면 본 논문에서 제안된 타원곡선 암호 프로세서는 기본에 제안된 가장 빠른 구조인 Shu등이 제안한 설계에 비해 약 2배의 하드웨어 자원을 사용하지만 4.8배 빠르다. 따라서 제

안된 구조는 네트워크 프로세서와 웹 서버등과 같은 높은 처리율을 요구하는 응용에 보다 적합하다 할 수 있다. 참고로 Xilinx XC4VLX80 디바이스의 슬라이스 하나는 2개의 LUT를 가진다.

5. 결 론

본 논문에서는 아래와 같은 절차에 따라 고성능 타원곡선 암호 프로세서를 제안하였다.

- $GF(2^m)$상에서 GNB를 이용한 효율적인 워드-레벨 곱셈기 설계
- 규칙적인 주소화 방식의 병렬화된 좌표 덧셈 및 배 연산 알고리즘 설계
- 고성능 좌표 덧셈 및 배연산기 설계
- Itoh, Tsujii 역원 알고리즘에 기반한 고성능 좌표변환 연산기 설계
- 단일 타원곡선 좌표 곱셈을 위한 10μs 시간 지연을 가지는 완성된 데이터패스 설계

제안된 타원곡선 암호 프로세서는 Xilinx XC4VLX80 FPGA 디바이스에 구현되었으며, 24,263 슬라이스를 사용하고 최대 143MHz에 동작 한다. 제안된 설계는 Shu. 등[7]이 제안한 하드웨어 구조와 비교했을 때 약 2배의 하드웨어 복잡도 증가를 보이지만 4.8배의 속도 향상을 보인다. 따라서 제안된 타원곡선 암호 프로세서는 네트워크 프로세서와 웹 서버등과 같은 높은 처리율을 요구하는 ECC에 적합하다. 뿐만 아니라 제안된 구조는 모듈성과 단순한 컨트롤 구조를 가지기 때문에 VLSI 구현에 매우 적합하다.

참 고 문 헌

[1] V.S. Miller, "Use of Elliptic Curves in Cryptography," *in Advances in Cryptology-Proc. of CRYPTO'85*, pp.417-426, 1986.

[2] N. Koblitz, "Elliptic Curve Cryptosystems," *Mathematics of Computation*, vol.48, pp.203-209, 1987.

[3] M. Rosing, *Implementing Elliptic Curve Cryptography*, Manning, 1999.

[4] D. Hankerson, J. Hernandez, and A. Menezes, "Software Implementation of Elliptic Curve Cryptography Over Binary Fields," *Proc. of CHES 2000, Lecture Notes in Computer Science*, Vol.1965, pp.1-24, 2000.

[5] A. Satoh and K. Takano, "A Scalable Dual-Field Elliptic Curve Cryptographic Processor," *IEEE Trans. on Computers*, Vol.52, No.4, pp.449-460, Apr. 2003.

[6] G. Orlando and C. Parr, "A High Performance Reconfigurable Elliptic Curve Processor for $GF(2^m)$," *CHES 2000, Lecture Notes in Computer Science*, Vol.1965, 2000.

[7] C. Shu, K. Gaj, and T. El-Ghazawi, "Low Latency Elliptic Curve Cryptography Accelerators for NIST Curves over Binary Fields," *FPT 2005* 1965, pp.309-310, 2005.

[8] N. Gura, S.C. Shantz, H. Eberle, S. Gupta, V. Gupta, D. Finchelstein, E. Goupy, and D. Stebila, "An End-to-End Systems Approach to Elliptic Curve Cryptography," *CHES 2002, Lecture Notes in Computer Science*, Vol.2523, pp. 349-365, 2002.

[9] A. K. Daneshbeh, M. A. Hasan, "Area efficient high speed elliptic curve cryptoprocessor for random curves," *IEEE Symposium on Information Technology: Coding and Computing (ITCC)*, Vol.2, pp.588-592, 2004.

[10] H. Eberle, N. Gura, S. Chang-Shantz, and Vipul Gupta, "A cryptographic processor for arbitrary elliptic curves over $GF(2^m)$," *Application-Specific Systems, Architectures, and Processors (ASAP)*, pp.444-454, 2003.

[11] C. Grabbe, M. Bednara, J. von zur Gathen, J. Shokrollahi, J. Teich, "A high performance vliw processor for finite field arithmetic," *Reconfigurable Architectures Workshop (RAW)*, 2003.

[12] K. Järvinen, M. Tommiska, J. Skyttä, "A scalable architecture for elliptic curve point multiplication," *IEEE Field-Programmable Technology (FPT)*, pp.303-306, 2004.

[13] N.A. Saqib, F. Rodríguez-Henríquez, A. Díaz-Pierez, "A parallel architecture for fast computation of elliptic curve scalar multiplication over $GF(2^m)$," *Parallel & Distributed Processing Symposium (IPDPS)*, 2004.

[14] IEEE 1363, *Standard Specifications for Publickey Cryptography*, 2000.

[15] NIST, Recommended elliptic curves for federal government use, May 1999.
http://csrc.nist.gov/encryption.

[16] A.J. Menezes, I.F. Blake, X. Gau, R.C. Mullin, S.A. Vanstone, and T. Yaghoobian, *Applications of Finite Fields*, Kluwer Academic Publisher, 1993.

[17] S. Kwon, K. Gaj, C. H. Kim, and C. P. Hong, "Efficient Linear Array for Multiplication in $GF(2^m)$ Using a Normal Basis for Elliptic Curve Cryptography," *CHES 2004 Lecture Notes in Computer Science*, Vol.3156, pp.76-91, 2004.

[18] J. López and R. Dahab, "Fast Multiplication on Elliptic Curves over $GF(2^m)$ without Precomputation," *CHES 1999, Lecture Notes in Computer Science*, Vol.1717, pp.316-327, 1999.

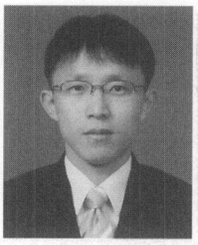
[19] T. Itoh and S. Tsuji, "A fast algorithm for computing multiplicative inverses $GF(2^m)$ in using normal bases," *Information and Computing*, Vol.78, No.3, pp.171-177, 1988.

[20] B. Ansari, M. Anwar Hasan, "High Performance Architecture of Elliptic Curve Scalar Multiplication," Tech. Report CACR 2006-01, 2006.

[21] M. Benaissa and W.M. Lim, "Design of Flexible $GF(2^m)$ Elliptic Curve Cryptography Processors," *IEEE Trans. VLSI Syst.*, Vol.14, No.6, pp.659-662, June 2006.

[22] C.J. McIvor, M. McLoone, and J.V. McCanny, "Hardware Elliptic Curve Cryptography Processor over *GF(p)*," *IEEE Trans. Circuits Syst. I: Reg. Papers*, Vol.53, No.9, pp.1946-1957, Sept. 2006.

[23] G. Chen, G. Bai, and H. Chen, "A High-Performance Elliptic Curve Cryptographic Processor for General Curves Over *GF(p)* Based on a Systolic Arithmetic Unit," *IEEE Trans. Circuits Syst. II: Express Briefs*, Vol.54, No.5, pp.412-416, May 2007.



김 창 훈

e-mail : kimch@daegu.ac.kr

2001년 대구대학교 컴퓨터정보공학부 (학사)

2003년 대구대학교 컴퓨터정보공학과 (공학석사)

2006년 대구대학교 컴퓨터정보공학과 (공학박사)

2006년~2007년 대구대학교 정보통신공학부 BK21 연구교수

2007년~현 재 대구대학교 컴퓨터·IT 공학부 전임강사

관심분야 : 암호 시스템, 임베디드 시스템, RFID/USN 보안