$GF(2^m)$상의 고속 타원곡선 암호 프로세서
(High Performance Elliptic Curve Cryptographic Processor for $GF(2^m)$)

김 창 훈 † 김 태 호 †† 홍 춘 표 †††
(Chang Hoon Kim) (Tae Ho Kim) (Chun Pyo Hong)

요 약 본 논문에서는 $GF(2^m)$상의 고속 타원곡선 암호 프로세서를 제안한다. 제안한 암호 프로세서는 타원곡선 정수 곱셈을 위해 López-Dahab Montgomery 알고리즘을 채택하고, $GF(2^m)$상의 산술 연산을 위해 가우시안 정규 기저(Gaussian Normal Basis: GNB)를 이용한다. 본 논문에서 구현한 타원곡선 암호 프로세서는 m=163을 선택하였으며 NIST(National Institute of Standard and Technology)에서 권고하는 5개의 $GF(2^m)$ 필드 크기 중에서 가장 작은 값으로 GNB 타입 4가 존재한다. 제안한 타원곡선 암호 프로세서는 Host Interface, Data Memory, Instruction Memory, Control로 구성되어 있으며 Xilinx XCV2000E FPGA칩을 이용하여 구현한다. FPGA 구현결과 제안된 타원곡선 암호 프로세서는 기존의 연구결과에 비해 속도에서 약 2.6배의 성능 향상을 보이며 훨씬 낮은 하드웨어 복잡도를 가진다.

키워드 : 타원곡선 암호시스템, 암호 프로세서, 유한체, 가우시안 정규 기저, VLSI

Abstract This paper presents a high-performance elliptic curve cryptographic processor over $GF(2^m)$. The proposed design adopts López-Dahab Montgomery algorithm for elliptic curve point multiplication and uses Gaussian normal basis for $GF(2^m)$ field arithmetic operations. We select m=163 which is the smallest value among five recommended $GF(2^m)$ field sizes by NIST and it is Gaussian normal basis of type 4. The proposed elliptic curve cryptographic processor consists of host interface, data memory, instruction memory, and control. We implement the proposed design using Xilinx XCV2000E FPGA device. Based on the FPGA implementation results, we can see that our design is 2.6 times faster and requires significantly less hardware resources compared with the previously proposed best hardware implementation.

Key words : Elliptic Curve Cryptosystem, Cryptographic Processor, Finite Field, Gaussian Normal Basis, VLSI

1. 서 론

1980년대 중반 Victor Miller[1]와 Neal Kobliz[2]에 의해 제안된 타원곡선 암호 시스템(Elliptic Curve Cryptosystem: ECC)은 최근 학계나 산업계로부터 많은 관심을 모으고 있다. ECC는 RSA[3]나 ElGamal[4]과 같은 다른 암호 시스템에 비해 현저히 작은 키를 사용하면서(약 1/6 정도) 동일한 안전도를 가진다[5-7].

· 본 연구는 2005학년도 대구대학교 학술연구비에 의하여 수행 되었음
† 정 회 원 : 대구대학교 정보통신공학과 교수
chkim@dsp.daegu.ac.kr
†† 학생회원 : 대구대학교 정보통신공학과
thkim@dsp.daegu.ac.kr
††† 정 회 원 : 대구대학교 정보통신공학과 교수
cphong@daegu.ac.kr
논문접수 : 2006년 8월 3일
심사완료 : 2006년 12월 8일

즉, 작은 키를 사용한다는 것은 계산 시간, 전력 소모, 저장 공간의 감소를 의미한다. 이러한 장점 때문에 현재까지 ECC의 하드웨어 구현에 관한 많은 연구결과가 발표되었다[8-10,14,19-27]. 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.

ECC에서 가장 중요한 연산은 kP이다. 여기서 k는 큰 정수이고 P는 타원곡선상의 한 포인트이다. kP 연산은 반복된 포인트 배 연산 및 덧셈 연산으로 계산될 수 있다. 만약 $GF(2^m)$상에서 ECC를 구현한다면 kP 연산은 포인트 곱셈 알고리즘, 좌표계 시스템, 기저 표기법에 의해 그 성능이 좌우된다. kP 연산을 위해 가장 일반적으로 사용되는 타원곡선 정수 곱셈 알고리즘으로

Double-and-Add[7], Non-Adjacent Form(NAF) Double-and-Add/Subtract[7], López-Dahab Montgomery[10]가 있다. Double-and-Add 알고리즘은 최소 m-1번의 포인트 배 연산과 k를 이진수로 표현했을 때 Hamming Weight만큼의 포인트 덧셈 연산이 필요하며, NAF를 이용한 방식은 kP 연산을 위해 k의 1의 개수를 줄여서 연산을 수행하는 방법으로 k의 각 비트를 1, 0, -1로 나타낸다. 이를 이용하면 포인트 덧셈 연산을 평균 $m/3$으로 줄일 수 있지만 k를 NAF 표기법으로 미리 바꾸어야 하는 단점이 있다. López-Dahab Montgomery 알고리즘은 평균적으로 Double-and-Add 방식보다 2.2배, NAF Double-and-Add/Subtract 방식보다 1.4배 빠르다[7]. 또한 이 알고리즘은 기본 연산이 매우 규칙적일 뿐만 아니라 분기조건이 없기 때문에 타이밍, 전력, 전자기장 공격에 높은 면역을 가진다[7,10].

$GF(2^m)$의 대표적인 원소 표기법으로 다항식 기저(Polynomial Basis: PB)와 정규 기저(Normal Basis: NB)가 있다. $GF(2^m)$상의 산술 연산에 있어 덧셈과 뺄셈을 제외한 나머지 연산들은 기저의 선택에 따라 그 성능이 좌우된다. 각 기저표기법은 장점과 단점을 가지는데, NB를 이용할 경우 A^{2^s}연산을 간단한 s-비트 순환 쉬프트 연산으로 구현할 수 있는 반면 곱셈 연산이 매우 복잡하며 하드웨어 구조가 규칙적이지 못하다. PB를 이용한 산술 연산기(Arithmetic Unit: AU)의 하드웨어 구현은 서로 다른 m에 대해 높은 규칙성 및 확장성을 가진다. 이러한 이유로 PB를 이용한 $GF(2^m)$상의 AU에 관한 많은 연구결과가 발표되었다[11,12].

GNB는 NB의 특별한 경우로서 PB와 함께 IEEE 1363[11], NIST[12] 등의 다양한 표준으로 채택되었으며, 8로 나누어떨어지지 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. Kwon 등[15]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 López-Dahab Montgomery 알고리즘을 채택하고, $GF(2^m)$상의 산술 연산을 위해 GNB를 이용한다. 본 논문에서 구현한 타원곡선 암호 프로세서는 m=163을 선택하였으며 NIST에서 권고하는 5개의 $GF(2^m)$ 필드 크기($m\in\{163, 233, 283, 409, 571\}$) 중에서 가장 작은 값으로 GNB 타입 4가 존재한다. 제안한 타원곡선 암호 프로세서를 Xilinx XCV2000E FPGA칩을 이용하여 구현하고 기존에 제안된 타원곡선 암호 프로세서와 성능을 비교 분석한 결과, 가장 빠른 속도를 보이고 제안된 프로세서 다음으로 속도가 빠른 Gura 등[14]이 제안한 설계보다 약 2.6배의 성능 향상을 보이며, 훨씬 낮은 하드웨어 복잡도를 보인다.

본 논문의 구성은 다음과 같다. 2절에서 López-Dahab kP 알고리즘과 GNB를 이용한 곱셈 알고리즘을 알아본 후, 3절에서 GNB를 이용한 $GF(2^m)$상의 워드-레벨 곱셈기를 유도한다. 4절에서는 워드-레벨 곱셈기에 기반한 새로운 AU를 유도하고, 타원곡선 암호 프로세서에 필요한 연산 시퀀스 및 명령어를 설계한 후 최종적으로 타원곡선 암호 프로세서를 완성한다. 5절에서는 본 논문에서 제안한 타원곡선 암호 프로세서의 FPGA 구현 결과와 기존에 제안된 프로세서의 성능을 비교 분석한 후 6절에서 결론을 맺는다.

2. 수학적 배경

본 장에서는 $GF(2^m)$상의 ECC 프로세서의 구현에 핵심이 되는 López- Dahab의 포인터 곱셈 알고리즘과 GNB상의 곱셈 알고리즘에 대해 살펴본다.

2.1 $GF(2^m)$상의 López-Dahab kP 알고리즘

E를 다음과 같이 정의된 $GF(2^m)$상의 타원곡선이라 하자.

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

López-Dahab kP 알고리즘은 사영좌표계를 사용하며 포인트($X: Y: Z$)는 식 (1)의 affine 좌표계 포인트에 (X/Z, Y/Z^2)로 대응된다. 여기서 $Z\neq 0$이다. 즉, 식 (1)에 $x=X/Z$, $y=Y/Z^2$을 대입하고 양변에 Z^4을 곱하여 식 (2)를 얻을 수 있다.

$$E: Y^2+XYZ=X^3Z+aX^2Z^2+bZ^4 \quad (2)$$

P, P_1, P_2가 E상의 포인트이고, P_2는 P_1과 P의 합으로 표현된다고 하자. 또한 P_i의 x-좌표 값은 X_i/Z_i로 표현할 수 있다고 하면, 여기서 $i\in\{1,2\}$이다. P_1+P_2 또는 $2P_i$의 x-좌표 계산은 y-좌표의 세 포인트(X, Y, Z) 중에서 어떠한 포인트도 요구하지 않는다. 따라서 알고리즘의 메인 연산에서 y-좌표를 생략할 수 있다. 메인 연산이 끝난 후, 식 (3)을 이용하여 y-좌표의 결과 값을 얻을 수 있다[10].

$$y_0=\frac{1}{x}\cdot(x+\frac{X_1}{Z_1})\left\{(x+\frac{X_1}{Z_1})(x+\frac{X_2}{Z_2})+x^2+y\right\}+y \quad (3)$$

[알고리즘 1] López-Dahab 타원곡선 정수 곱셈 알고리즘[13]

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

식 (1), (2), (3)으로부터 [알고리즘 1]을 이용하여 kP 연산을 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.

2.2 GNB 타입 k를 이용한 $GF(2^m)$상의 곱셈 알고리즘

2.2.1 $GF(2^m)$의 GNB 타입 k

m, k를 소수 $p \neq 2$에 대해, $p=mk+1$인 양의 정수라 하고, $K=<\tau>$는 $GF(p)^\times$상에서 위수(order) k인 유일한 부분군이라 하자. β가 $GF(2^{mk})$의 p번째 원시근이라 하면, 아래 원소

$$\alpha=\sum_{j=0}^{k-1}\beta^{r^j} \tag{4}$$

을 $GF(2)$상의 타입 (m, k) 가우스 주기(gauss period)라 한다. Ord_p2를 mod p에 대한 2의 위수라 하고, $\gcd(mk/ord_p2,\ m)=1$이라고 가정하면, α는 $GF(2^m)$상의 정규 원소이다. 즉, $0 \le i \le m-1$에 대해, $\alpha_i=\alpha^{2^i}$라 놓으면, $\{\alpha_0,\alpha_1,\alpha_2,\cdots,\alpha_{m-1}\}$은 $GF(2)$상의 $GF(2^m)$에 대한 기저이고, 이를 $GF(2^m)$의 GNB 타입 k라 부른다. $K=<\tau>$가 순환군 $GF(p)^\times$상의 위수 k인 부분군이기 때문에, 잉여군(quotient group) $GF(p)^\times/K$는 위수 m인 순환군이고, 군의 생성원은 $2K$이다. 따라서 $GF(p)^\times$의 coset decomposition을 식 (5)와 같이 disjoint union으로 나타낼 수 있다.

$$GF(p)^\times=K_0 \cup K_1 \cup K_2 \cup \cdots \cup K_{m-1} \tag{5}$$

여기서 $K_i=2^iK\ (0 \le i \le m-1)$이고 $GF(p)^\times$의 모든 원소는 임의의 $0 \le s \le k-1$과 $0 \le t \le m-1$에 대해 τ^s2^t로 유일하게 표현되고, $0 \le i \le m-1$에 대해 식 (6)을 얻을 수 있다.

$$\alpha\alpha_i=\sum_{s=0}^{k-1}\beta^{r^s}\sum_{t=0}^{k-1}\beta^{r^t2^i}=\sum_{s=0}^{k-1}\sum_{t=0}^{k-1}\beta^{r^s(1+r^{t-s}2^i)}=\sum_{s=0}^{k-1}\sum_{t=0}^{k-1}\beta^{r^s(1+r^t2^i)} \tag{6}$$

예제로 $GF(2^7)$이 GNB 타입 4라 하자. 여기서 $m=7$, $k=4$이다. 즉, $p=29=mk+1$이다. 이 경우 $GF(29)^\times$상에서 위수 4인 유일한 순환군은 $K=\{1,2^7,2^{14},2^{21}\}=\{1,12,28,17\}$이다. 여기서 β를 $GF(2^{28})$상에서 29번째 원시근이라 하고, $\tau=12$로 두면, $GF(2^7)$상의 정규 원소 α는 $\alpha=\beta+\beta^{12}+\beta^{17}+\beta^{28}$로 표현되고, $\{\alpha_0,\alpha_1,\cdots,\alpha_6\}$는 NB이다. $0 \le i \le 6$에 대해, $\alpha\alpha_i$의 계산은 표 1로부터 얻을 수 있다. 각 블록 K와 K'는 엔트리 (s,t), $0 \le s \le 3$, $0 \le t \le 6$에 대해 각각 τ^s2^t와 $1+\tau^s2^t$의 값을 가진다.

표 1 $GF(2^7)$상의 GNB 타입 4를 이용한 K_i와 K_i'의 계산

K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_0'	K_1'	K_2'	K_3'	K_4'	K_5'	K_6'
1	2	4	8	16	3	6	2	3	5	9	16	4	7
12	24	19	9	18	7	14	13	25	20	10	19	8	15
28	27	25	21	13	26	23	0	28	26	22	14	27	24
17	5	10	20	11	22	15	18	6	11	21	12	23	16

표 1로부터 $\alpha\alpha_0=\alpha_1$이고, 나머지 부분은 식 (7)과 같다.

$$\begin{aligned}
&\alpha\alpha_1=\alpha_0+\alpha_2+\alpha_5+\alpha_6, && \alpha\alpha_2=\alpha_1+\alpha_3+\alpha_4+\alpha_5,\\
&\alpha\alpha_3=\alpha_2+\alpha_5, && \alpha\alpha_4=\alpha_2+\alpha_5,\\
&\alpha\alpha_5=\alpha_0+\alpha_2+\alpha_5+\alpha_6, && \alpha\alpha_6=\alpha_1+\alpha_3+\alpha_4+\alpha
\end{aligned} \tag{7}$$

예를 들어, 블록 K_2'에 대한 $\alpha\alpha_2$는 다음과 같이 나타낼 수 있다. 블록 K_2'의 원소는 5, 20, 26, 11이며 K_i 블록들의 $5 \in K_1$, $20 \in K_3$, $26 \in K_5$, $11 \in K_4$에서 찾을 수 있다. 따라서 $\alpha\alpha_2=\alpha_1+\alpha_3+\alpha_4+\alpha_5$이다. 식 (7)로부터 곱셈 행렬 (λ_{ij})은 식 (8)과 같다.

$$(\lambda_{ij})=\begin{pmatrix}
0&1&0&0&0&0&0\\
1&0&1&0&0&1&1\\
0&1&0&1&1&1&0\\
0&0&1&0&0&1&0\\
0&0&0&1&0&0&1\\
0&1&1&1&0&0&1\\
0&1&0&0&1&1&1
\end{pmatrix} \tag{8}$$

2.2.2 GNB를 이용한 $GF(2^m)$상의 비트-레벨 곱셈 알고리즘

$A=\sum_{i=0}^{m-1}a_i\alpha_i$, $B=\sum_{j=0}^{m-1}b_j\alpha_j$를 $GF(2^m)$상의 두 원소라 하면, 두 원소의 곱 $C=AB=\sum_{s=0}^{m-1}c_s\alpha_s$는 식 (9)와 같다.

$$C=\sum_{i=0}^{m-1}a_i\alpha_i\sum_{j=0}^{m-1}b_j\alpha_j=\sum_{i,j}a_ib_j\alpha_i\alpha_j \quad (9)$$
$$=\sum_{i,j}a_ib_j\sum_{s=0}^{m-1}\lambda_{ij}^{(s)}=\sum_{s=0}^{m-1}(\sum_{i,j}a_ib_j\lambda_{ij}^{(s)})\alpha_s$$

또한 Kwon[15]의 보조 정리 1로부터 $C=\sum_{s=0}^{m-1}c_s\alpha_s$의 계수 c_s는

$$c_s=\sum_{i,j}a_{i+s}b_{j+s}\lambda_{ij}^{(0)}=\sum_{i,j}a_{i+s}b_{j+s}\lambda_{ij} \quad (10)$$
$$=\sum_{j=0}^{m-1}(\sum_{i=0}^{m-1}a_{i+s}\lambda_{ij})b_{j+s}$$

와 같다. $m\times m$행렬 $X=(x_{st})$에 대해 $GF(2)$상의 원소 x_{st}, $0\le s,t\le m-1$을 식 (11)과 같이 정의하면,

$$x_{st}=(\sum_{i=0}^{m-1}a_{i+s}\lambda_{it})b_{t+s} \quad (11)$$

X의 t번째 열벡터 X_t는 식 (12)와 같다.

$$X_t=(x_{0t},x_{1t},\cdots,x_{m-1,t})^T \quad (12)$$

여기서 $(x_{0t},x_{1t},\cdots,x_{m-1,t})^T$는 행벡터 $(x_{0t},x_{1t},\cdots,x_{m-1,t})$의 전치 행렬이다. 또한 $\sum_{t=0}^{m-1}x_{st}=c_s$이기 때문에 모든 열벡터 X_t, t=0,1,$\cdots$,m-1의 합은 정확히 식 (13)과 같다.

$$(c_0,c_1,\cdots,c_{m-1})^T \quad (13)$$

여기서 $m-1=2\nu$라 하고 $Y=(y_{st})$를 X의 열벡터 치환에 의한 $m\times m$행렬이라 정의하면, ν가 홀수일 때 Y는 식 (14)와 같이 정의되고,

$$Y=(X_\nu,\cdots,X_3,X_1,X_{m-1},X_{m-3},\cdots,X_{m-\nu},X_{\nu-1},\cdots,X_2,X_0,X_{m-2},\cdots,X_{m-\nu+1}) \quad (14)$$

ν가 짝수일 때, Y는 식 (15)와 같이 정의된다.

$$Y=(X_\nu,\cdots,X_2,X_0,X_{m-2},\cdots,X_{m-\nu},X_{\nu-1},\cdots,X_3,X_1,X_{m-1},X_{m-3},\cdots,X_{m-\nu+1}) \quad (15)$$

여기서, $Y_t=(y_{0t},y_{1t},\cdots,y_{m-1,t})^T$인 Y의 모든 열벡터 Y_t, t=0,1,$\cdots$,m-1의 합은 $(c_0,c_1,\cdots,c_{m-1})^T$인 X의 모든 열벡터 X_t, t=0,1,$\cdots$,m-1의 합과 같다. 따라서 패러럴 입출력 곱셈기를 설계하기 위해 Y의 열벡터 합을 계산하는 대신 Y의 순환 쉬프트된 대각 벡터의 합을 계산한다. 즉, 행렬 Y의 표현에서 벡터 X_t와 X_{m-t}사이에 정확히 t-1개의 열이 존재한다. 또한 X_t의 s번째 원소와 X_{m-t}의 $s+t$번째 원소는 그들의 가수(summand)에서 동일한 a_{is}를 가진다. 즉, 식 (11)로부터 식 (16)을 얻을 수 있다.

$$x_{s+t,m-t}=(\sum_{i=0}^{m-1}a_{i+s+t}\lambda_{i,m-t})b_s \quad (16)$$
$$=(\sum_{i=0}^{m-1}a_{i+s}\lambda_{i-t,m-t})b_s=(\sum_{i=0}^{m-1}a_{i+s}\lambda)b_s$$

x_{st}와 $x_{s+t,m-t}$는 같은 항 $\sum_{i=0}^{m-1}a_{i+s}\lambda_{it}$를 가진다. 이는 AB의 계산에 있어 사용되는 XOR 게이트의 개수를 줄인다. 지금까지 설명한 내용을 바탕으로 [알고리즘 2]를 얻을 수 있다.

[알고리즘 2] GNB를 이용한 $GF(2^m)$상의 비트-레벨 곱셈 알고리즘[15]

Input : $A,B\in GF(2^m)$
Output : D, $D_i=c_i$ for all $0\le i\le m-1$,
where $AB=\sum_{i=0}^{m-1}c_i\alpha_i$
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 $0\le t\le m-1$
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

3. GNB를 이용한 $GF(2^m)$상의 워드-레벨 곱셈기

본 장에서는 2장의 GNB상의 비트-레벨의 곱셈 알고리즘으로부터, 워드-레벨의 곱셈 알고리즘을 유도하고 $GF(2^7)$상의 GNB 타입 4에 대한 VLSI 구조를 그 예로 제시한다.

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

[알고리즘 2]로부터 비트-시리얼 또는 비트-패러럴 곱셈기를 만들 수 있지만 속도가 느리기 때문에 큰 m(최소 163)을 요구하는 ECC에 적합하지 않다. 따라서 많은 타원곡선 암호 프로세서들은 비트-레벨 곱셈기의 단점을 극복하기 위해 워드-레벨 곱셈기를 채택하였다[16,17]. 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 2]로부터 [알고리즘 3]과 같은 GNB를 이용한 $GF(2^m)$상의 워드-레벨 곱셈 알고리즘을 얻을 수 있다.

[알고리즘 3] $GF(2^m)$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 $0 \le t \le L-2$
2. for $0 \le s \le m-1$
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 $0 \le s \le m-1$
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

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

[알고리즘 3]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($y_{s,s+tw} + y_{s,s+tw+1} + \cdots + y_{s,s+tw+(w-1)-\gamma} + D_{s+tw-\gamma}$)이 동시에 계산된다. 즉, 고정된 s에 대한 마지막 출력값 D_s는 다음 식과 같이 연속적으로 계산된다.

$$D_s = \overbrace{y_{s,s}+y_{s,s+1}+\cdots+y_{s,s+(\omega-1)}}^{D_{s+\omega}} + \overbrace{y_{s,s+\omega}+y_{s,s+\omega+1}+\cdots+y_{s,s+2\omega-1}}^{D_{s+2\omega}} + y_{s,s+2\omega}+\cdots+y_{s,s+m-1} = \sum_{i=0}^{m-1} y_{s,s+i} = c_s \quad (17)$$

NIST에 의해 권고된 $GF(2^m)$상의 필드 크기가 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 3]의 $L-1$번째 단계에서 계산되며 규칙적인 하드웨어 구조 설계에 방해가 된다. 즉, 하드웨어 구조는 [알고리즘 3]의 단계 3, 8을 수행하기 위해 각각 다르게 설계되어야 한다. 본 논문에서는 이 문제를 해결하기 위해 L길이의 컨트롤 신호(111,$\cdots$,10)를 사용한다. 보다 자세한 설명은 3.2절에 기술한다.

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

예제로 $w=2$인 $GF(2^7)$상의 워드-레벨 곱셈기를 설계한다. 식 (8)의 곱셈 행렬과 식 (10), (11), (14), (15)으로부터 곱셈 결과 $C=AB=\sum_{i=0}^{6} c_i\alpha_i$는 식 (18)과 같이 얻을 수 있다.

$$\begin{aligned}
c_0 &= (a_2+a_5)b_3 + a_{0256}b_1 + a_{1456}b_6 + (a_2+a_6)b_4 + a_{1345}b_2 + a_1b_0 + a_{1236}b_5\\
c_1 &= (a_3+a_6)b_4 + \underline{a_{1360}b_2 + a_{2560}b_0} + (a_3+a_0)b_5 + a_{2456}b_3 + a_2b_1 + a_{2340}b_6\\
c_2 &= (a_4+a_0)b_5 + a_{2401}b_3 + \underline{a_{3601}b_1} + (a_4+a_1)b_6 + a_{3560}b_4 + a_3b_2 + a_{3451}b_0\\
c_3 &= (a_5+a_1)b_6 + a_{3512}b_4 + a_{4012}b_2 + \underline{(a_5+a_2)b_0 + a_{4601}b_5} + a_4b_3 + a_{4562}b_1\\
c_4 &= (a_6+a_2)b_0 + a_{4623}b_5 + a_{5123}b_3 + (a_6+a_3)b_1 + \underline{a_{5012}b_6 + a_5b_4} + a_{5603}b_2\\
c_5 &= (a_0+a_3)b_1 + a_{5034}b_6 + a_{6234}b_4 + (a_0+a_4)b_2 + a_{6123}b_0 + a_6b_6 + a_{6014}b_3\\
c_6 &= \underline{(a_1+a_4)b_2} + a_{6145}b_0 + a_{0345}b_5 + (a_1+a_5)b_3 + a_{0234}b_1 + a_0b_6 + \underline{a_{0125}b_4}
\end{aligned} \quad (18)$$

식 (18)의 밑줄은 $w=2$, a_{ijkl}는 $a_i+a_j+a_k+a_l$을 의미한다. 그림 1은 $w=2$에 대하여 $GF(2^7)$상에서 GNB 타입 4를 이용한 $C=AB$에 대응되는 쉬프트 레지스터 회로이고, 그림 2는 그림 1의 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 1의 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 1의 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.

4. $GF(2^m)$상의 ECC 프로세서 설계

4.1 GNB를 이용한 곱셈 및 역원 AU 설계

NB를 이용할 경우, 임의의 원소의 역원을 구하기 위해 Fermat 이론이 사용된다. A를 $GF(2^m)$상의 원소라 하면, $GF(2^m)^\times$의 위수는 2^m-1이므로 A^{2^m-1}이다. 즉, $A^{-1}=A^{2^m-2}=A^{2(2m-1-1)}$ 이다. 여기서

$$2^s-1 = \begin{cases} (2^{\frac{s}{2}}-1)(2^{\frac{s}{2}}+1), & \text{if } s=\text{even} \\ 2(2^{\frac{s-1}{2}}-1)(2^{\frac{s-1}{2}}+1), & \text{if } s=\text{odd} \end{cases} \quad (19)$$

을 이용하여 [알고리즘 4]를 유도할 수 있다.

[알고리즘 4] Itoh-Tsujii의 역원 알고리즘[18]
Input : $A \in GF(2^m)$

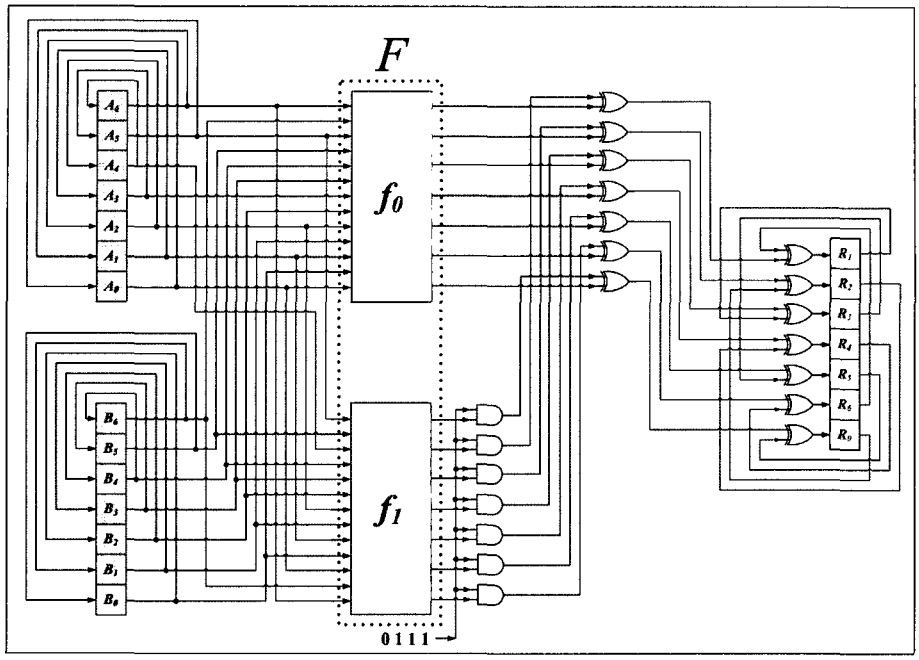

그림 1 $m=7$, $w=2$에 대한 $GF(2^m)$상의 GNB 타입 4 곱셈기

Output : $B=A^{2^m-2}=A^{-1}$

1. $B \leftarrow 1$, $A \leftarrow A^2$, $s \leftarrow m-1$
2. while $s \geq 1$ do
3. if s=even, then

$$A \leftarrow A^{2^{\frac{s}{2}}+1},\ s \leftarrow \frac{s}{2}$$

4. else, then

$$B \leftarrow BA,\ A \leftarrow A^{2(2^{\frac{s-1}{2}}+1)},\ s \leftarrow \frac{s-1}{2}$$

5. end while

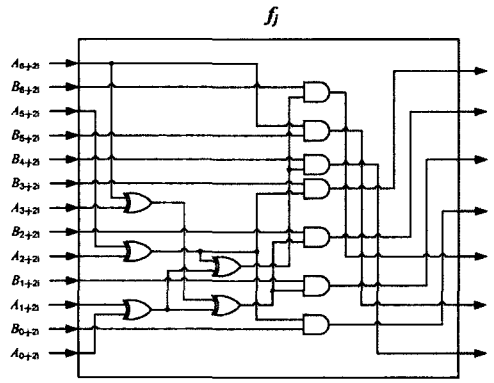

그림 2 그림 1의 f_j

[알고리즘 4]에서 $A \in GF(2^m)$의 역원을 구하기 위하여 $\lfloor \log_2(m-1) \rfloor + H(m-1) - 1$개의 곱셈 연산이 필요하다. 여기서 H는 주어진 정수의 2진수 표현의 Hamming Weight이다. 만약 $m=163$이면, $m-1=162=(10100010)_2$ 이므로 필요한 곱셈의 개수는 7+3-1=9이고 식 (20)의 지수 연산 순서로 A의 역원을 구할 수 있다.

$$A^{-1}=A^{2(2^{81}+1)(2(2^{40}+1)(2^{20}+1)(2^{10}+1)(2^5+1)(2(2^2+1)(2+1)+1)+1)} \quad (20)$$

식 (20)과 그림 1의 워드-레벨 곱셈기로부터 그림 3과 같은 $GF(2^{163})$상의 곱셈 및 역원을

위한 VLSI 구조를 유도할 수 있다. 그림 3의 AU는 곱셈과 역원을 수행하기위해 163-비트의 3-to-1 멀티플렉서 및 11-to-1 멀티플렉서 그리고 7-비트의 제어신호(Ctrl)를 추가한다.

4.2 명령어 설계

[알고리즘 1]과 GNB의 특성을 이용하면 표 2와 같이 총 31개의 연산 시퀀스를 얻을 수 있다. GNB상에서 2^s

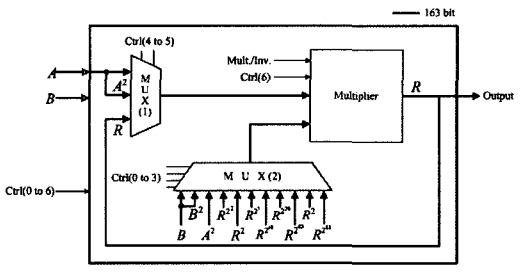

그림 3 $GF(2^{163})$상의 곱셈 및 역원 VLSI 구조

표 2 López-Dahab 타원곡선 정수 곱셈 알고리즘에 기반 한 연산 시퀀스

순번	연산	비고		순번	연산	비고
0	$X_2 \leftarrow (x)^4 + b$	Initialize		18	$x_0 \leftarrow 1/Z_1$	Coordinate Conversion
1	$Z_2 \leftarrow (x)^2$			19	$x_0 \leftarrow X_1 x_0$	
2	$x_0 \leftarrow X_1 Z_2$	Main Loop		20	$X_1 \leftarrow x + x_0$	
3	$y_0 \leftarrow X_2 Z_1$			21	$Z_1 \leftarrow 1/Z_2$	
4	$Z_3 \leftarrow x_0 + y_0$			22	$Z_1 \leftarrow X_2 Z_1$	
5	$x_0 \leftarrow x_0 y_0$			23	$Z_1 \leftarrow x + Z_1$	
6	$X_1 \leftarrow (Z_3)^2 x$	if k_i=1		24	$Z_1 \leftarrow Z_1 X_1$	
7	$X_1 \leftarrow X_1 + x_0$			25	$Z_2 \leftarrow x^2 + y$	
8	$Z_1 \leftarrow (Z_3)^2$			26	$Z_1 \leftarrow Z_1 + Z_2$	
9	$x_0 \leftarrow (Z_3)^2 b$			27	$Z_1 \leftarrow Z_1 X_1$	
10	$X_2 \leftarrow (X_2)^2 + x_0$			28	$y_0 \leftarrow 1/x$	
11	$Z_2 \leftarrow (X_2 Z_2)^2$			29	$y_0 \leftarrow Z_1 y_0$	
12	$X_2 \leftarrow (Z_3)^2 x$	if k_i=0		30	$y_0 \leftarrow y + y_0$	
13	$X_2 \leftarrow X_2 + x_0$					
14	$Z_2 \leftarrow (Z_3)^2$					
15	$x_0 \leftarrow (Z_1)^4 b$					
16	$X_1 \leftarrow (X_1)^4 + x_0$					
17	$Z_1 \leftarrow (X_1 Z_1)^2$					

표 3 López-Dahab 알고리즘에 기반한 명령어

명령어	기능	클럭 사이클
Sqr	$C \leftarrow A^2$	1
Add-1	$C \leftarrow A + B$	1
Add-2	$C \leftarrow A^2 + B$	1
Add-3	$C \leftarrow A^4 + B$	1
Mult.-1	$C \leftarrow A \times B$	$\lceil 163/w \rceil + 1$
Mult.-2	$C \leftarrow A^2 \times B$	$\lceil 163/w \rceil + 1$
Mult.-3	$C \leftarrow A^4 \times B$	$\lceil 163/w \rceil + 1$
Mult.-4	$C \leftarrow (A \times B)^2$	$\lceil 163/w \rceil + 1$
Inv.	$C \leftarrow A^{-1}$	$\lceil 163/w \rceil \times 9 + 9$

는 s-비트만큼의 간단한 순환 쉬프트 연산이기 때문에 [알고리즘 1]에서 $GF(2^m)$상의 임의의 원소 A^2 혹은 A^4 연산을 한 사이클에 처리할 수 있다. 이는 각 루프에 있어 상당한 사이클의 감소를 의미한다. 표 3에는 표 2로부터 얻어진 9가지의 명령어를 요약하였다.

4.3 $GF(2^{163})$상의 타원곡선 암호프로세서

그림 3의 AU와 표 3의 명령어로부터 그림 4와 같은 새로운 AU를 얻을 수 있다. 그림 4에서는 m=163이고 GNB 타입 4이다. 그림 4에 나타나듯이 그림 3의 워드-

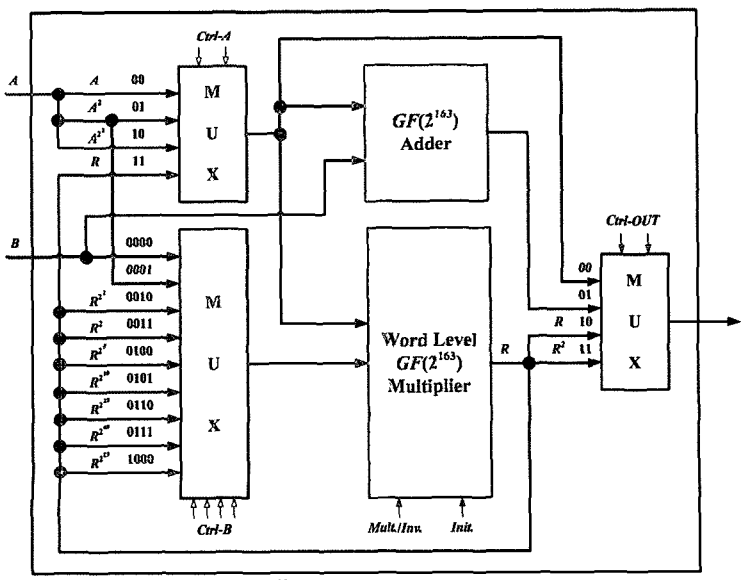

그림 4 $GF(2^{163})$상의 GNB를 이용한 새로운 AU

레벨 곱셈기에 덧셈 및 제곱연산 부분과 입출력을 제어하기 위해 멀티플렉서를 추가한다. 명령어에 따른 입출력 제어 함수의 기능을 표 4, 표 5, 표 6에 각각 요약하였다.

그림 5는 본 논문에서 제안한 $GF(2^{163})$상의 타원곡선 암호프로세서의 전체적인 구조를 나타낸다. 그림 5에 나타나듯이 타원곡선 암호 프로세서는 크게 5개의 블록 Host Interface, Data Memory, Instruction Memory, Control, AU로 구성된다. Host Interface는 타원곡선의 베이스 포인트 P, 곡선 파라메타 b, 비밀키 k, 연산의 시작을 알리는 *start* 신호를 Host 프로세서로부터 입력받아 타원곡선 암호 프로세서로 전달한다. 그림 5의 타원곡선 암호 프로세서의 데이터 전송 및 연산은 모두 163-비트로 이루어지며, Instruction Memory로부터 명령어 및 Data Memory의 주소를 전송받아 컨트롤 신호와 함께 연산을 수행한다. Data Memory는 Dual Port 메모리로 구성되며, Port-A는 상승 에지에 Port-B는 하강 에지에 각각 동작한다. 그 외 모든 AU 및 레지스터는 상승 에지에 동작한다. 따라서 데이터의 동기를 맞추기 위해 AU의 Input-A와 Data Memory의 Port-B 사이에 Buffer 레지스터를 추가한다. 곱셈과 역원 연산을 제외하면 모든 연산은 3사이클(명령어 패치+데이터 로더+연산 수행 및 저장)에 수행된다. 곱셈 연산의 경

표 4 명령어에 따른 Ctrl-A 신호

비트-시퀀스	명 령 어
00	Add-1, Mult.-1, Mult.-4
01	Load, Add-2, Mult.-2
10	Add-3, Mult-3
11	Inv.

표 5 명령어에 따른 Ctrl-Out 신호

비트-시퀀스	명 령 어
00	Sqr
01	Add-1, Add-2, Add-3
10	Mult.-1, Mult.-2, Mult.-3
11	Mult.-4, Inv.

표 6 명령어에 따른 Ctrl-B 신호

비트-시퀀스	명령어	
0000	Mult.-1, Mult.-2, Mult.-3, Mult.-4	
0001	A^2	Inv.
0010	R^{2^2}	
0011	R^2	
0100	R^{2^5}	
0101	$R^{2^{10}}$	
0110	$R^{2^{20}}$	
0111	$R^{2^{40}}$	
1000	$R^{2^{81}}$	

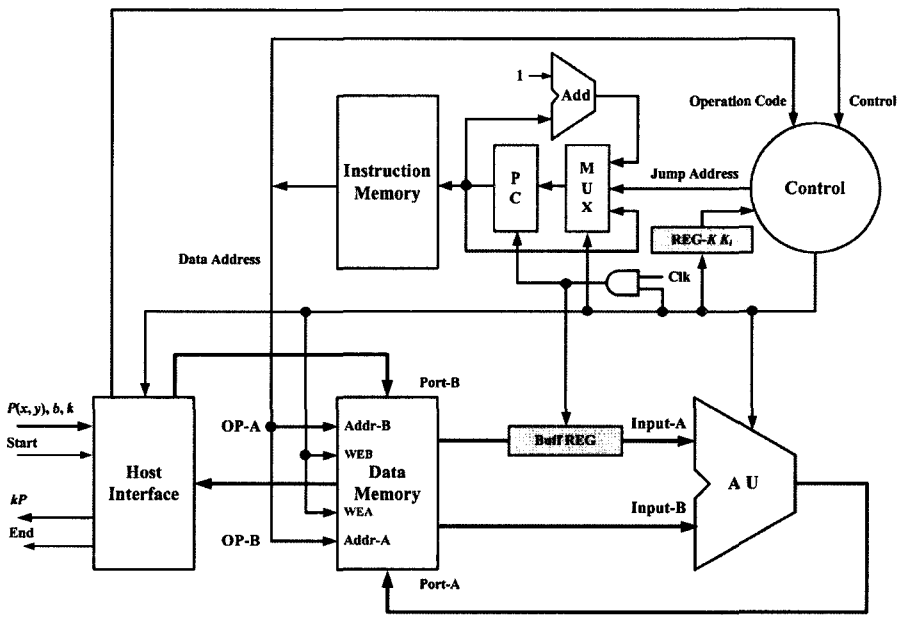

그림 5 $GF(2^{163})$상의 타원곡선 암호 프로세서 구조

우 레지스터에 데이터를 로드 하는데 1사이클이 소요되기 때문에 총 $\lceil 163/w \rceil + 4$사이클이 소요된다. 역원 연산의 경우 매번 곱셈마다 1사이클의 데이터로드 연산을 가지기 때문에 총 $(\lceil 163/w \rceil \times 9 + 9 + 2)$시간이 소요된다. 하지만 전체적인 연산에 있어 데이터가 로드될 때 다음에 실행할 명령어를 패치한다. 즉, 패러럴하게 처리되기 때문에 모든 명령어는 1사이클씩 줄어든다. $GF(2^{163})$상에서 한 번의 정수 곱셈에 소요되는 총 사이클 수는 곱셈기의 워드 크기 w에 따라 다르게 나타난다. 표 7에 명령어 수와 소요되는 클럭 사이클 수를, 표 8에 w의 크기에 따른 클럭 사이클 수를 요약하였다.

5. FPGA 구현 및 성능분석

본 논문에서 제안한 타원곡선 암호 프로세서의 기능을 검증하고 성능을 분석하기 위해 Xilinx사의 Vertex-II XCV2000E FPGA칩을 이용하여 구현하였다. 이를 위해 VHDL로 회로를 기술하였으며, Synopsys사의 FPGA Compiler II를 이용하여 회로합성을 하였다. 합성된 회로는 Xilinx사의 Foundation 소프트웨어를 이용하여 Place-and-Route를 수행하고 타이밍 분석을 하였다. 또한 Mentor Graphics사의 Model Sim을 이용하여 그 기능을 검증하고 리버트론사의 SoC(System-on-Chip) 테스트 보드를 이용하여 FPGA에 타원곡선 암호 프로세서를 구현 하였다. 테스트 보드는 ARM7TDMI 마이크로프로세서와 Xilinx XCV2000E FPGA칩을 탑재하고 있으며, 최대 50MHz의 클럭 주파수를 지원한다. 표 9에 w의 크기에 따른 하드웨어 및 클럭 주파수에 대하여 요약하였다. 또한 표 10에 기존에 제안된 타원곡선 암호 프로세서와 성능을 비교하였다. 각 프로세서의 구

표 9 그림 5 타원곡선 암호 프로세서의 FPGA 구현 결과

워드크기 (w)	LUT 개수	FF 개수	클럭 주파수(MHz)
14	5,295	667	74.92
19	7,726	667	74.21
33	11,096	667	73.89
41	13,123	667	73.61
55	16,783	667	72.72

표 7 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 8 워드크기 w에 따른 클럭 사이클 수

워드 크기(w)	14	19	33	41	55
클럭 사이클 수	14,698	10,066	6,042	5,042	4,039

표 10 타원곡선 암호 프로세서 성능비교

	필드	하드웨어 플랫폼	클럭 주파수 (MHz)	속도(ms)	비 고	
그림 5	$GF(2^{163})$ GNB	Xilinx XCV2000E	72.72	0.055	그림 4의 AU	
[14]	$GF(2^{163})$ PB	Xilinx XCV2000E	66.5	0.143	Karatsuba Multiplier	
[19]	$GF(2^{160})$ PB	0.13-μm CMOS ASIC	510.2	0.19	64-bit multiplier	
[10]	$GF(2^{155})$ ONB	ASIC	40	3.9	Massey-Omura multiplier	
[20]	$GF(2^{155})$ ONB	Xilinx XC4020XL	15	15.9	Massey-Omura multiplier	
[21]	$GF(2^{155})$ ONB	Xilinx XCV300-4	36	6.8	Massey-Omura multiplier	
[22]	$GF(2^{155})$ ONB	Xilinx XC4085XLA	37	1.29	Massey-Omura multiplier	
[23]	$GF(2^{53})$ ONB	Xilinx XC4044XL		2.4	Massey-Omura multiplier	
[8]	$GF(((2^2)^4)^{21})$ PB	Xilinx XC4062	16	4.5	Special composite field multiplier	
[24]	$GF(2^{167})$ PB	Xilinx XCV400E	76.7	0.21	167-bit×16-bit multiplier and 167-bit×167-bit squarer For $P(x)=x^{167}+x^6+1$	
[25]	$GF(2^{191})$ PB	Xilinx XC4000XL		17.71	Standard form	$P(x) = x^{191}+x^9+1$
				11.82	Hessian form	
[26]	$GF(2^{163})$ PB	0.25-μm CMOS ASIC	66	1.1	Random curve	288-bit × 8-bit multiplier
				0.65	Koblitz curve	
		ALTERA EPF10K259AGC599-2	3	80.3	Random curve	82-bit × 4-bit multiplier
				45.6	Koblitz curve	
[27]	$GF(2^{160})$ PB	ASIC	50	7	1024-bit adder	

현에 이용된 플랫폼, 기저, m이 다르기 때문에 정확한 성능비교는 어렵다. 그러나 표 10에 나타나듯이 본 논문에서 제안된 타원곡선 암호 프로세서가가 속도에서 가장 좋은 성능을 보인다.

그림 5는 Xilinx XCV2000E FPGA칩을 이용하여 구현되었으며 72.72MHz 클럭 주파수에서 0.055ms의 연산속도를 보인다. [19]의 프로세서는 0.13-μm CMOS ASIC을 이용하며 510MHz 클럭 주파수에서 연산속도가 0.19ms이다. [8,10,21-24]의 프로세서는 단순하고 작은 곱셈기를 위하여 최적 정규 기저(Optimal Normal Basis: ONB) 또는 합성 필드(composite field)를 사용하였다. 그러나 이 프로세서들은 IEEE, NIST 등에서 권고하는 것과 다른 필드 파라메타를 사용하기 때문에 응용 범위가 매우 좁다. [27]의 프로세서는 $P(x)=x^{167}+x^6+1$을 사용하는 필드상에서 167×16-비트 곱셈기 및 167×167-비트 역원기를 이용하였다. [24]의 프로세서는 임의의 기약 다항식에서 동작할 수 있다. 그러나 불균형한 288×8-비트 모듈러 곱셈기가 사용되었기 때문에 유연성이 떨어진다. [25]의 프로세서는 덧셈기 기반의 하드웨어 구조이며 정수상의 모듈러 연산 및 임의의 다항식에서 타원곡선 정수 곱셈 연산을 지원한다. 그러나 1024-비트의 덧셈기는 타원곡선 정수 곱셈 연산을 위한 길이로는 너무 길다. 표 11에는 본 논문에서 제안된 프로세서 다음으로 속도가 빠른 Gura 등이 제안한 설계와 조금 더 구체적으로 비교 분석하였다. 표 11에 나타나듯이 속도에서 약 2.6배의 성능 향상을 보이며, 훨씬 낮은 하드웨어 복잡도를 보인다. 이렇게 우수한 성능을 보이는 이유는 크게 세 가지로 요약할 수 있다. 1) 기존의 프로세서가 다항식 기저상의 워드레벨 곱셈기 혹은 ONB상의 비트-시리얼 곱셈기를 사용한 반면, 본 연구에서는 GNB를 사용한 새로운 워드레벨 곱셈기를 채택하였다. 2) 기존의 프로세서들이 역원 연산을 위해 바이너리 Double-and-Add (m-1번의 곱셈연산 필요) 알고리즘을 구현한 반면, 본 연구팀은 Itoh 역원 알고리즘을 직접적으로 구현하였다. 3) GNB 상에서는 2^s연산은 s-비트만큼 순환 쉬프트연산이기 때문에 2장의 López-Dahab kP연산을 고속으로 수행 할 수 있을 뿐만 아니라 간단한 신호 인덱스 변환을 통하여, 다른 연산 모듈과 하드웨어 공유가 가능하였다.

6. 결 론

본 논문에서는 $GF(2^m)$상의 고속 타원곡선 암호 프로세서를 제안하였다. 제안된 타원곡선 암호 프로세서를 위해 하드웨어 면적 및 속도에 있어 상충관계를 개선할 수 있는 $GF(2^m)$상의 워드-레벨 곱셈기를 유도하였다. 또한 워드-레벨 곱셈기와 Itoh-Tsujii 역원 알고리즘을 기반으로 곱셈 및 역원 AU를 유도하고, López-Dahab 타원곡선 정수 곱셈 알고리즘을 이용하여 연산 시퀀스 및 명령어를 설계하였다. 이를 바탕으로 Host Interface, Data Memory, Instruction Memory, Control을 추가하여 데이터 패스를 완성하였다. 제안된 타원곡선 암호 프로세서를 Xilinx XCV2000E FPGA칩을 이용하여 구현 하였고 Gura 등이 제안한 타원곡선 암호 프로세서와 성능을 비교 분석한 결과, 속도에서 약 2.6배의 성능 향상을 보였고 훨씬 낮은 하드웨어 복잡도를 가졌다. 더욱이 본 논문에서 제안된 타원곡선 암호 프로세서는 GNB를 이용한 새로운 워드-레벨 곱셈기를 채택하였기 때문에 워드의 크기 설정에 따라, 스마트카드와 같은 저면적 응용에서부터 어플리케이션 서버와 같은 고속의 데이터 처리를 요구하는 응용 분야까지 다양한 통신매체에 적용 될 수 있다.

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] RSA Laboratories, *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*, RFC 3447, 2003.

[4] T. ElGamal, "A Publick Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *in Advances in Cryptology-Proc. of CRYPTO'84*, pp. 10-18, 1985.

[5] I.F. Blake, G. Seroussi, and N.P. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, 1999.

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

[7] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Eliptic Curve Cryptography*, Springer 2004.

표 11 Gura 등이 제안한 설계와의 성능비교

	m, w	LUT 개수	FF 개수	블록 메모리	지연시간 (μs)	클럭 주파수 (MHz)	칩계열 (Xilinx)
Gura 등[14]	163, 64	19,508	6,442	9K-bits	143	66.5	XCV2000E
그림 5	163, 55	16,783	667	2.2K-bits	55	72.72	XCV2000E
성능 개선배율	n/a	1.16	9.66	4.1	2.6	1.1	n/a

[8] L. Gao, S. Shrivastava, and G. Sobelman, "Elliptic Curve Scalar Multiplier Design Using FPGAs," *Proc. Cryptographic Hardware and Embedded Systems (CHES '99)*, pp. 257-268, Aug. 1999.

[9] G. Orlando and C. Parr, "A High Performance Reconfigurable Elliptic Curve Processor for *GF* (2^m)," *CHES 2000*, LNCS 1965, 2000.

[10] G. Agnew, R. Mullin, I. Onyszchuk, and S. Vanstone, "An Implementation of Elliptic Curve Cryptosystems over F_2^{155}," *IEEE J. Selected Areas Comm.*, vol. 11, pp. 804-813, Jun. 1993.

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

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

[13] J. Lopez and R. Dahab, "Improved Algorithms for Elliptic Curve Arithmetic in $GF(2^m)$," *SAC '98*, LNCS Springer Verlag, 1998.

[14] N. Gura, S.C. Shantz, H.E. Sumit Gupta, V. Gupta, D. Finchelstein, E. Goupy, and D. Stebila, "An End-toEnd Systems Approach to Elliptic Curve Cryptography," *CHES '02*, LNCS 2523, pp. 349-365, 2002.

[15] 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*, LNCS 3156, pp. 76-91, 2004.

[16] J. L. Massey and J.K. Omura, "Computational method and apparatus for finite field arithmetic," *US Patent No. 4587627*, 1986.

[17] A. Reyhani-Masoleh and M.A. Hasan, "Efficient Digit-Serial Normal Basis Multipliers over *GF* (2^m)," *ACM Trans. Embedded Computing Systems (TECS)*, special issue on embedded systems and security, vol. 3, no. 3, pp. 575-592, Aug. 2004.

[18] 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.

[19] 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.

[20] S. Sutikno, A. Surya, and R. Effendi, "An Implementation of ElGamal Elliptic Curves Cryptosystems," *Proc. 1998 IEEE Asia-Pacific Conf. Circuits and Systems (APCCAS '98)*, pp. 483-486, Nov. 1998.

[21] S. Sutikno, R. Effendi, and A. Surya, "Design and Implementation of Arithmetic Processor F_2^{155} for Elliptic Curve Cryptosystems," *Proc. 1998 IEEE Asia-Pacific Conf. Circuits and Systems (APCCAS '98)*, pp. 647-650, Nov. 1998.

[22] K.H. Leung, K.W. Ma, W.K. Wong, and P.H.W. Leong, "FPGA Implementation of a Microcoded Elliptic Curve Cryptographic Processor," *Proc. 2000 IEEE Symp. Field Programmable Custom Computing Machines (FCCM '99)*, pp. 68-76, Apr. 2000.

[23] M. Ernst, S. Klupsch, O. Hauck, and S.A. Huss, "Rapid Prototyping for Hardware Accelerated Elliptic Curve Public-Key Cryptosystems," *Proc. 12th Int'l Workshop Rapid System Prototyping (RSP 2001)*, pp. 24-29, Jun. 2001.

[24] M.C. Rosner, "Elliptic Curve Cryptosystems on Reconfigurable Hardware," *master's thesis, Worcester Polytechnic Inst.*, May 1998, http://www.ece.wpi.edu/research/cry-pt/publications/documents/ms_mrosner.pdf.

[25] N.P. Smart, "The Hessian Form of an Elliptic Curve," *Proc. Cryptographic Hardware and Embedded Systems(CHES 2001)*, pp. 118-125, May 2001.

[26] S. Okada, N. Torii, K. Itoh, and M. Takenaka, "Implementation of Elliptic Curve Cryptographic Coprocessor over $GF(2^m)$ on an FPGA," *Proc. Cryptographic Hardware and Embedded Systems (CHES 2000)*, pp. 25-40, Aug. 2000.

[27] J. Goodman and A. Chandrakasan, "An Energy Efficient Reconfigurable Public-Key Cryptography Processor Architecture," *Proc. Cryptographic Hardware and Embedded Systems (CHES 2000)*, pp. 175-190, Aug. 2000.

김 창 훈

2001년 2월 대구대학교 컴퓨터정보공학부, 학사. 2003년 2월 대구대학교 컴퓨터정보공학과, 석사. 2006년 8월 대구대학교 컴퓨터정보공학과, 박사. 2006년 9월~현재 대구대학교 정보통신공학과 BK21, 연구교수. 관심분야는 암호 시스템, Embedded System, RFID/USN 보안

김 태 호

2006년 2월 대구대학교 컴퓨터IT공학부, 학사. 2006년 3월~현재 대구대학교 정보통신공학과, 석사과정. 관심분야는 암호 시스템, Embedded System, RFID/USN 보안

홍 춘 표

1978년 2월 경북대학교 전자공학과, 학사. 1986년 12월 Georgia Institute of Technology ECE, 석사. 1991년 12월 Georgia Institute of Technology ECE, 박사. 1994년 9월~현재 대구대학교 정보통신공학부, 교수. 관심분야는 DSP 하드웨어 및 소프트웨어, 컴퓨터 구조, VLSI 신호처리, Embedded System