스마트카드 보안용 타원곡선 암호를 위한 GF(2^{163}) 스칼라 곱셈기

정상혁* · 신경욱*

A GF(2^{163}) Scalar Multiplier for Elliptic Curve Cryptography for Smartcard Security

Sang-Hyeok Jeong* · Kyung-Wook Shin*

이 논문은 2008년도 금오공과대학교 연구비를 지원받았음.

요 약

스마트카드 보안용 타원곡선 암호를 위한 스칼라 곱셈기를 설계하였다. 스마트카드 표준에 기술된 163-비트의 키 길이를 지원하며, 유한체 (finite field) 상에서 스칼라 곱셈의 연산량을 줄이기 위해 complementary recoding 방식을 적용한 Non-Adjacent Format (NAF) 변환 알고리듬을 적용하여 설계되었다. 설계된 스칼라 곱셈기 코어는 0.35-㎛ CMOS 셀 라이브러리로 합성하여 32,768 게이트로 구현되었으며, 150-MHz@3.3-V로 동작한다. 설계된 스칼라 승산기는 스마트카드용 타원곡선 암호 알고리듬의 전용 하드웨어 구현을 위한 IP로 사용될 수 있다.

ABSTRACT

This paper describes a scalar multiplier for Elliptic curve cryptography for smart card security. The scaler multiplier has 163-bits key size which supports the specifications of smart card standard. To reduce the computational complexity of scala multiplication on finite field, the non-adjacent format (NAF) conversion algorithm which is based on complementary recoding is adopted. The scalar multiplier core synthesized with a 0.35-㎛ CMOS cell library has 32,768 gates and can operate up to 150-MHz@3.3-V. It can be used in hardware design of Elliptic curve cryptography processor for smartcard security.

키워드

Elliptic curve cryptography (ECC), 공개키 암호화, 스칼라 곱셈, Complementary recoding

* 금오공과대학교 전자공학부

접수일자 : 2009. 05. 06
심사완료일자 : 2009. 06. 04

Ⅰ. 서 론

인터넷의 대중화와 네트워크 기술의 발달로 유무선 네트워크를 통한 정보교류와 전자금융 및 전자상거래가 활발하게 이루어지고 있으며, 이에 따라 모바일 환경에서 정보보호를 위한 보안기술의 중요성이 증대되고 있다. 특히, 사용자 인증과 서명, 무결성 검증 등을 위한 공개키 암호 (public key cryptography) 알고리듬의 중요성이 더욱 커지고 있다. 대표적인 공개키 암호 알고리듬으로 RSA (Rivest-Shal-Adleman)와 타원곡선 암호 (Elliptic Curve Cryptography; ECC) 알고리듬이 널리 사용되고 있다. 타원곡선 암호 알고리듬은 키 길이의 증가에 따라 안전도가 지수적으로 증가하며, 163-비트의 키 길이를 갖는 ECC는 1024-비트의 키 길이를 갖는 RSA 알고리듬과 유사한 안전도를 가지는 것으로 평가되고 있다[1]. 따라서 ECC는 메모리 용량과 처리속도가 제한된 모바일 환경에 적합한 차세대 암호 시스템의 표준으로 자리 잡아가고 있다.

최근 모바일 네트워킹 환경의 모델로서 스마트카드 (smartcard) 기술에 대한 관심이 고조되고 있다. 스마트카드는 연산을 위한 CPU가 장착되어 자체적인 정보처리가 가능하다는 장점을 갖는다. 그러나 모바일 기기의 특성상 작은 용량의 메모리와 저가의 마이크로프로세서가 사용되므로, ECC와 같은 공개키 알고리듬의 연산에는 적합지 않은 것으로 인식되고 있다. 특히 타원곡선 암호 알고리듬은 상대적으로 작은 키 길이에도 불구하고 알고리듬 특성상 계산의 복잡도가 높아 소프트웨어로 구현하는 경우 처리시간이 길어지는 문제점을 갖는다.

본 논문에서는 타원곡선 암호 프로세서의 핵심 블록인 스칼라 승산기를 구현하였으며, 스칼라 승산기의 연산량 감소를 통한 효율적인 구현을 위해 complementary recoding 방식을 적용하여 설계하였다.

Ⅱ. 타원곡선 암호 알고리듬

타원곡선 암호 알고리듬은 타원곡선 이산로그 문제 (Elliptic Curve Discrete Logarithmic Problem; ECDLP)에 근간을 두고 있다. ECDLP는 타원곡선 상의 임의의 한 점 P에 정수 k를 곱한 값이 $Q=kP$일 때, 점 Q와 P를 알고 있어도 정수 k를 계산하기 어렵다는 사실을 의미한다[2,3]. 이러한 타원곡선 암호 알고리듬의 핵심적인 연산과정은 $Q=kP$를 구하는 스칼라 곱셈연산이다. 스칼라 곱셈은 소수체 (prime field)와 유한체 (finite field) $GF(2^m)$ 상에서 모두 정의가 가능하다. 그러나 캐리전달이 필요한 소수체 대신 캐리전달이 없고 하드웨어 구현이 용이한 유한체 상에서의 연산이 일반적으로 사용된다.

실수에서 정의된 타원곡선은 식(1)과 같은 방정식을 만족시키는 점(x,y)의 집합이다.

$$y^2 = x^3 + ax + b \tag{1}$$

단, x, y, a, b는 실수

만약 여기서 계수들이 $4a^3+27b^2 \neq 0$을 만족시킨다면 이 방정식은 실제로 타원곡선 상에 존재하지 않는 무한원점 (point at infinity) O와 함께 덧셈에 대해 닫혀 있는 군을 형성한다. 따라서 타원곡선에서 행해지는 기본적인 연산은 덧셈이 된다. 이러한 타원곡선에서의 덧셈연산에 대한 역원은 식(2)와 같이 정의된다.

$$P=(x,y)\text{일 때, } -P=(x,-y) \tag{2}$$

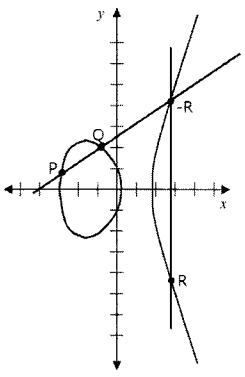

그림 1. 점 덧셈 연산의 기하학적 정의
Fig. 1 Geometric definition of point addition

타원곡선은 x축에 대해 대칭이기 때문에 덧셈에 대한 역원은 항상 타원곡선 상에 존재한다. 그림 1은 타원

곡선에서의 덧셈연산을 기하학적으로 보여주고 있다. 타원곡선의 서로 다른 두 점 P, Q를 더하기 위해 두 점을 지나는 연장선을 긋는다. 두 점이 서로 덧셈에 대한 역원이 아니라면 연장선은 타원곡선의 다른 한 점에서 반드시 교차하게 되어있으며 그 점을 $-R$이라고 하면, 구하고자 하는 $P+Q$의 값은 교차점 $-R$의 x축에 대한 대칭점 R로 정의된다. 이를 점 덧셈(point addition) 연산이라고 한다.

만약 두 점이 서로 덧셈에 대한 역원이면 두 점을 이은 연장선은 y축에 평행하고 타원곡선의 다른 한 점과 만나지 않는다. 이때 두 점의 합은 식 (3)과 같이 무한 원점으로 정의된다.

$$P+(-P)=O \tag{3}$$

식(3)을 정리하면 $P+O=P$이 성립된다. 즉, 타원곡선 위의 모든 점은 덧셈에 대한 항등원을 가지게 된다. 그림 2는 타원곡선 상의 동일한 한 점을 두 번 더하는 경우이다. 이 때 타원곡선 위의 점 P에서 접선이 그어지고 이 점이 x축에 접해있지 않다면, 접선은 반드시 타원곡선의 다른 한 점에서 교차하게 된다. 교차점 $-R$의 x축에 대한 대칭점은 $R=2P$로 정의되며 이 연산을 점 두 배 (point doubling) 연산이라고 한다.

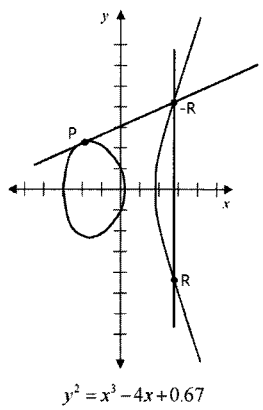

그림 2. 점 두배 연산의 기하학적 정의
Fig. 2 Geometric definition of point doubling

유한체에서 타원곡선 군은 유한한 점들의 집합으로 구성되므로 타원곡선 연산을 기하학적으로 표현하는 것은 불가능하며, 따라서 유한체에서 타원곡선 연산은 대수적인 연산에 의해 정의된다. 점 $P(x_1,y_1)$와 $Q(x_2,y_2)$를 지나는 직선의 방정식을 $y=\alpha x+\beta$라고 정의하면, α와 β는 식(4)와 같다.

$$\alpha=\frac{y_2-y_1}{x_2-x_1} \tag{4-a}$$

$$\beta=y_1-\alpha x_1 \tag{4-b}$$

만약 $(\alpha x+\beta)^2=x^3+ax+b$가 성립한다면, 점 P와 Q를 지나는 직선 위의 점 $(x,\ \alpha x+\beta)$는 타원 곡선 상의 점이 된다. 따라서 3차 방정식 $x^3-(\alpha x+\beta)^2+ax+b=0$의 각각의 근에 대하여 하나의 교차점이 존재하게 된다. $(x_1,\ \alpha x_1+\beta)$와 $(x_2,\ \alpha x_2+\beta)$는 타원곡선 위의 점 P, Q이므로, x_1, x_2가 근임을 알 수 있다. 나머지 근은 $x_3=\alpha_2-x_1-x_2$와 $y_3=\alpha x_3+\beta=y_1+\alpha(x_3-x_1)$이 된다. 따라서 유한체 상에서 점 덧셈의 결과 $R(x_3,y_3)=P+Q$는 식(5), 식(6)과 같이 되며, 여기서 λ는 식(7)과 같이 정의된다.

$$x_3=\lambda^2+\lambda+x_1+x_2+a \tag{5}$$

$$y_3=\lambda(x_1+x_3)+x_3+y_1 \tag{6}$$

$$\lambda=\frac{y_2-y_1}{x_2-x_1} \tag{7}$$

한편 $P=Q$이면, 유한체 상에서 점 두배 연산은 $R(x_3,y_3)=2P$가 되어 점 P에서 미분 값이 되므로, 음함수 미분을 하면 x_3과 y_3은 각각 식(8), 식(9)와 같이 되고, 여기서 λ는 식(10)과 같이 정의된다.

$$x_3=\lambda^2+\lambda+a \tag{8}$$

$$y_3=x_1^2+(\lambda+1)x_3 \tag{9}$$

$$\lambda=x_1+\frac{y_1}{x_1} \tag{10}$$

다음 절에서 설명되는 스칼라 곱셈은 점 덧셈과 점 두 배 연산들의 반복으로 이루어지며, 점 연산들은 유한체에서 덧셈, 곱셈 그리고 나눗셈 연산을 통해 이루어진다.

Ⅲ. 타원곡선 스칼라 곱셈

3.1 타원곡선 스칼라 곱셈

타원곡선 암호 시스템에서 암호화 연산과 복호화 연산은 스칼라 곱셈으로 이루어진다. 타원곡선 스칼라 곱셈을 구현하는 가장 기본적인 방법은 double-and-add 방법이며, 이는 RSA 암호 알고리듬에서 지수연산의 구현을 위해 사용되는 square-and-multiply 방법과 유사하다.

스칼라 k가 m-비트로 표현된다면, 일반적으로 타원곡선 스칼라 곱셈은 알고리듬 1의 과정으로 계산된다. b_i가 '1'인 경우에는 타원곡선 상의 점 P를 이전에 계산된 Q에 더하여 갱신한 후 점 두배 연산이 수행되고, b_i가 '0'일 경우에는 점 덧셈 연산을 수행하지 않고 점 두배 연산이 수행된다. 따라서 알고리듬 1에서 볼 수 있듯이, 타원곡선 스칼라 곱셈은 점 두배 연산과 점 덧셈 연산의 반복과정으로 계산된다. 한편, 스칼라 곱셈을 구성하는 점 덧셈과 점 두배 연산은 유한체 상의 덧셈, 곱셈, 나눗셈 연산으로 구현된다.

$k = \sum_{i=0}^{m-1} b_i 2^i \quad (b_i \in 0,1)$
$P = P(x_1, y_1)$
$Q = P$
for i from m-1 downto 0 begin
  if $(b_i = 1)$ *then* $Q = add(P, Q)$
  $Q = double(Q)$
end

알고리듬 1. 타원곡선 스칼라 곱셈
Algorithm 1. Elliptic curve scalar multiplication

3.2 NAF를 이용한 타원곡선 스칼라 곱셈

기존의 타원곡선 스칼라 곱셈 알고리듬은 최소 $(m-1)$회의 점 두배 연산과 k의 hamming weight 만큼의 점 덧셈 연산이 필요하다. 점 덧셈 연산을 줄이기 위한 방법으로 non-adjacent- format (NAF) 변환이 제안되었다[4]. 스칼라 k가 NAF 형태로 변환되면 k의 hamming weight가 감소되어, 스칼라 곱셈의 연산량이 줄어들게 된다[5,6,7]. NAF 변환을 이용한 타원곡선 스칼라 곱셈 과정은 알고리듬 2와 같다.

$NAF(k) = \sum_{i=0}^{m-1} k_i 2^i$
$Q = 0$
for i from m-1 downto 0 begin
  if $k_i = 1$ *then* $Q = Q + P$
  if $k_i = -1$ *then* $Q = Q - P$
  $Q = 2Q$
end

알고리듬 2. NAF 변환을 이용한 타원곡선 스칼라 곱셈
Algorithm 2. Elliptic curve scalar multiplication using NAF transformation

3.3 NAF 변환 알고리듬

NAF 변환을 이용한 스칼라 곱셈을 위해서는 우선 이진형태로 주어진 k 값을 redundant 형태의 NAF로 변환해야 한다. NAF 형태로 변환할 경우 k의 hamming weight가 감소되어 연산량 이 줄어들게 되지만, 이 경우 -1을 계산하기 위해 점 뺄셈 연산이 추가된다. 점 뺄셈 연산은 점 덧셈 연산과 점 역원 연산으로 이루어지며, 점 역원 연산은 XOR로 이루어진다. NAF 변환을 위한 여러 가지 알고리듬들이 제안되고 있으며, 기본적인 NAF 변환은 알고리듬 3과 같다.

$k = \sum_{i=0}^{m-1} b_i 2^i \quad (b_i \in 0,1)$
$c = k;\ l = 0$
while$(c > 0)$
  if $(c$ *is odd*$)$
    $s_l = 2 - (c \bmod 4)$
    $c = c - s_l$
  else
    $s_l = 0$
  end
  $c = c/2;\ l = l + 1$
end
return s

알고리듬 3. 기본적인 NAF 변환
Algorithm 3. Basic NAF transformation

예를 들어, 정수 $(2927)_{10}$을 이진형태로 표현하면 $(101101101111)_2$가 되어 hamming weight는 9가 된다. 한편 알고리듬 3에 의한 $(2927)_{10}$의 NAF 변환결과는

$(01100\bar{1}00\bar{1}000\bar{1})$ (단, $\bar{1}$은 -1을 의미함)가 되며, 따라서 hamming weight가 5로 감소하여 스칼라 곱셈의 연산량이 감소된다. 그러나 이와 같은 NAF 변환방법은 뺄셈과 모듈러 연산 등 복잡한 계산을 필요로 하므로 하드웨어 구현에 적합하지 않아 사용되지 않는다.

한편, 2004년 Okeya는 mutual opposite form (MOF)[8]라는 변환방법을 발표하였으며, MOF 변환은 알고리듬 4와 같다. MOF는 기본적인 NAF 변환 알고리듬보다 간단하여 하드웨어 구현에 유리한 장점을 갖는다.

$k = \sum_{i=0}^{m-1} b_i 2^i \ (b_i \in 0,1)$
$c = k$
$s_n = c_{n-1}$
for $i = n-1$ *downto* 1
  $s_i = c_{i-1} - c_i$
$s_0 = -c_0$
return s

알고리듬 4. MOF 변환
Algorithm 4. MOF transformation

Complementary recoding(CR) 방법[5,6]은 2003년 Chang에 의해 제안된 알고리듬으로서 이진형태의 k가 주어질 때 이것의 보수를 이용하여 NAF 형태로 변환하며, 알고리듬 5와 같다. 예를 들어, $(2927)_{10} = (101101101111)_2$을 알고리듬 5에 의해 NAF로 변환하면 식(11)과 같이 되며, NAF로 변환된 s의 hamming weight가 9에서 5로 감소하였다.

$$s = (1000000000000)_2 - (010010010000)_2 - 1_2 \quad (11)$$
$$= (10\bar{1}00\bar{1}00\bar{1}000\bar{1})$$

$k = \sum_{i=0}^{m-1} b_i 2^i \ (b_i \in 0,1)$
$c = \bar{k}$
$s = 2^m - c - 1$
return s

알고리듬 5. Complementary recoding
Algorithm 5. Complementary recoding

CR 알고리듬은 기존의 NAF 변환 방법들과 비교하여 간단한 하드웨어로 구현될 수 있으며, 본 논문에서는 CR 변환 방법을 이용하여 타원곡선 스칼라 승산기를 구현하였다.

Ⅳ. 하드웨어 구현

기존의 스마트카드 보안에 사용되던 공개키 암호화 시스템은 대부분 1024-비트의 키 길이를 갖는 RSA 시스템이다. 본 논문에서는 동일한 수준의 안전도를 제공하는 163-비트 키 길이의 타원곡선 암호 시스템을 설계한다. 사용된 타원곡선은 SECG의 SEC2 표준을 따르며, 기약다항식은 식 (12)와 같다[1].

$$x^{163} + x^8 + x^2 + x + 1 = 0 \quad (12)$$

설계된 타원곡선 스칼라 곱셈기는 CR 변환을 이용하여 유한체 $GF(2^{163})$ 상에서 $Q = kP$를 계산한다. 내부 구조는 그림 3과 같으며, 입·출력 인터페이스 블록, CR 변환기, 163-비트 유한체 ALU, 그리고 중간 결과값을 저장하기 위한 레지스터와 제어회로 등으로 구성된다. 입·출력 인터페이스 블록은 타원곡선 방정식의 계수 a와 f를 저장하는 레지스터와 데이터의 입·출력을 제어하는 회로로 구성된다. 내부 레지스터 ACC_Reg는 스칼라 곱셈을 위한 데이터인 타원곡선 상의 초기 좌표값 (P_x, P_y), 그리고 스칼라 k값과 함께 중간 결과값을 저장한다. 유한체 ALU 블록은 유한체 $GF(2^{163})$ 상에서의 덧셈, 곱셈, 나눗셈 연산을 수행하는 회로들로 구성된다.

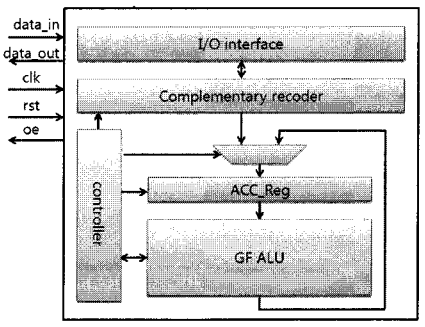

그림 3. 설계된 타원곡선 스칼라 곱셈기의 구조
Fig. 3. Architecture of the designed elliptic curve scalar multiplier

초기에 타원곡선 방정식의 계수 a 와 f, 타원곡선 상의 초기 좌표값 P, 유한체 연산을 위한 기약다항식의 계수, 그리고 스칼라 k 가 data_in 포트를 통해 순차적으로 입력된다. 스칼라 k는 CR 변환기에 의해 NAF로 변환되며, 변환된 $NAF(k)$의 값에 따라 점 덧셈, 점 두배 연산 그리고 점 역원 연산이 반복적으로 수행되어 스칼라 곱셈이 이루어진다. 한편, 점 덧셈, 점 두배 그리고 점 역원 연산들은 유한체 $GF(2^{163})$ 상에서의 덧셈, 곱셈 그리고 나눗셈 연산들로 구성된다.

4.1 유한체 연산기

유한체 연산기 GF_ALU는 유한체 $GF(2^{163})$ 상에서 덧셈, 곱셈, 나눗셈 연산을 수행하며, 그림 4와 같은 구조로 설계되었다. 유한체 덧셈 연산은 XOR 게이트로 구현하였으며, 유한체 곱셈과 나눗셈은 2차원 배열 구조의 단일 행으로 구현하였다. 유한체 곱셈은 MSB 우선 구조[9]를 사용하였으며, 유한체 나눗셈은 확장된 유클리드 알고리듬[10]을 사용하였다.

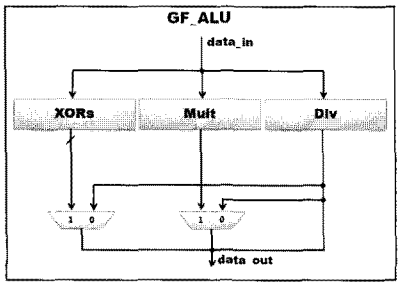

그림 4. GF_ALU 블록
Fig. 4. GF_ALU block

유한체 $GF(2^m)$ 상의 곱셈기는 그림 5-(a)와 같이 $m \times m$ 크기의 2차원 배열로 구현될 수 있으며, 나눗셈기는 그림 5-(b)와 같이 $(m+1) \times (2m-1)$ 크기의 2차원 배열로 구현될 수 있다. 일반적으로, 유한체 $GF(2^m)$ 상의 곱셈과 나눗셈 연산을 위한 완전병렬 2차원 배열은 m^2 에 비례하는 크기를 가지게 되며, 이를 직접 구현하기 위해서는 매우 큰 하드웨어를 필요로 하게 된다. 본 논문에서는 그림 5의 2차원 배열에서 단일 행 블록(점선으로 표시된 부분)을 사용하여 반복적인 연산으로 유한체 곱셈과 나눗셈이 계산되는 부분병렬 구조로 구현하였다.

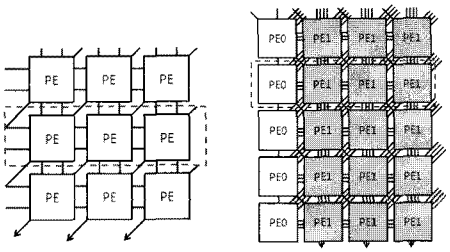

그림 5. 유한체 연산을 위한 2차원 어레이 구조
(a) 유한체 곱셈 (b) 유한체 나눗셈
Fig. 5. 2-D array for finite-field arithmetic
(a) finite-field multiplication (b) finite-field division

그림 4에서 유한체 곱셈을 연산하는 Mult 블록은 163열×1행의 처리요소(PE) 배열로 구성되며, 163번의 반복연산으로 유한체 곱셈이 계산된다. 배열을 구성하는 PE은 그림 6과 같이 설계되었다. 유한체 나눗셈을 연산하는 Div 블록은 1개의 처리요소 PE0과 (163열 × 1행)의 처리요소 PE1의 배열로 구성되며, 325 ($=163 \times 2-1$)번의 반복연산으로 유한체 나눗셈이 계산된다. 배열을 구성하는 처리요소 PE0와 PE1은 각각 그림 7-(a), 7-(b)와 같이 설계되었다[11].

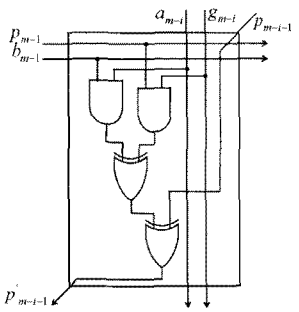

그림 6. 유한체 곱셈기의 처리요소
Fig. 6. PE for finite-field multiplication

$GF(2^{163})$ 상의 유한체 곱셈연산은 163 사이클의 반복연산과 데이터 입력과정을 포함해서 총 164 클록이 소요된다. 유한체 나눗셈연산은 유한체 곱셈에 비해 중간결과로 출력되는 변수가 매우 많기 때문에 실질적으로 피드백 루프를 이루는 레지스터의 길이는 유한체 나눗셈을 기준으로 결정되어야 한다. 이때, 레지스터는 $GF(2^{163})$ 상에서 $7m+1=1,142$ 비트가 필요하다. 또

한, $GF(2^{163})$ 상의 유한체 나눗셈연산은 325 사이클의 반복연산과 데이터 입력과정을 포함해서 총 326 클록이 소요된다.

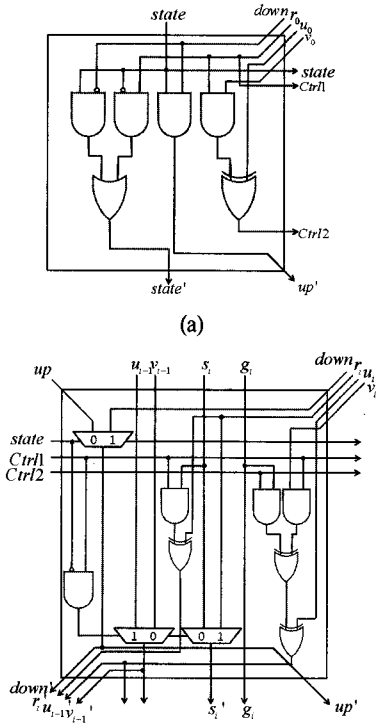

그림 7. 유한체 나눗셈기의 처리요소
(a) PE0 (b) PE1
Fig. 7. PE0, PE1 for finite-field division
(a) PE0 (b) PE1

4.2 Complementary recoder

Complementary recoder는 스칼라 k의 ham- ming weight를 줄여 스칼라 곱셈의 연산을 간소화시키는 블록이다. 알고리듬 5의 구현을 위해 그림 8과 같이 $c=\overline{k}$를 만들기 위한 163개의 인버터와 1을 가산하는 Add_one 회로로 구성된다.

$\overline{k}+1$을 계산하는 Add_one 블록은 피가수가 1로 고정되므로 간략화된 리플캐리(ripple- carry) 가산기로 구현될 수 있으며, MSB를 계산하는 Type-0 셀, LSB를 계산하는 Type-2 셀, 그리고 나머지 비트들을 계산하는 Type-1 셀을 그림 9와 같이 구현하였다. 한편, Type-0 셀은 MSB에서 발생되는 캐리출력을 고려하기 위해 Type-1셀과 다르게 NAND 게이트가 사용된다.

NAF로 변환된 스칼라 k는 $k_i \in \{-1,0,1\}$인 signed-digit 값을 가지므로, 2-비트의 코드로 표현되어야 한다. CR 알고리듬은 최상위 digit를 제외한 나머지 digit들은 항상 0 또는 -1이 되는 특성을 가지므로, 본 논문에서는 k_i를 $-1=10_2$, $0=00_2$, $1=01_2$로 정의하였다.

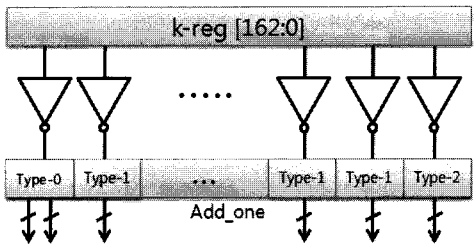

그림 8. Complementary recoder 블록
Fig. 8. Complementary recoder block

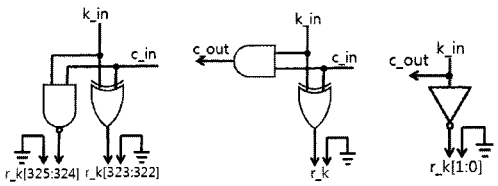

그림 9. Type-0, 1, 2 셀의 회로
(a) Type-0 (b) Type-1 (c) Type-2
Fig. 9. Type-0, 1, 2 cell circuits
(a) Type-0 (b) Type-1 (c) Type-2

V. 설계 검증 및 성능 평가

설계된 타원곡선 암호용 스칼라 곱셈기는 Verilog HDL을 이용하여 RTL 수준에서 모델링되었으며, ECC 관련 표준 문서에 제시된 테스트 벡터를 이용하여 기능검증을 수행하였다. 시뮬레이션에 사용된 벡터 및 시뮬레이션 결과는 그림 10과 같으며, a와 g는 타원곡선을 정의하는 방정식의 계수이고, k는 스칼라 값, P_x, P_y는 타원곡선 상의 초기 좌표값을 나타내며, Q_x, Q_y는 스칼라 곱셈의 결과 $Q=kP$의 좌표값을 나

타낸다. 표준 문서에 제시된 결과와 동일한 시뮬레이션 결과가 출력되어 설계된 회로가 정상 동작함을 확인하였다.

설계된 타원곡선 스칼라 곱셈기 코어를 0.35-μm CMOS 셀 라이브러리로 합성한 결과 32,768개의 게이트로 합성되었다. 그림 3의 스칼라 곱셈기 구조에서 가장 긴 연산시간을 필요로 하는 부분은 complementary recoder이며, 162 비트의 캐리전파가 발생되어 약 45.91-ns의 지연을 갖는다. 암호화 연산에 필요한 타원곡선 파라미터들이 입력되는 초기화 과정 동안 스칼라 k에 대한 CR 변환이 이루어지도록 함으로써 complementary recoder가 전체 스칼라 곱셈기의 동작속도에 영향을 미치지 않도록 하였다. 타이밍 분석결과, complementary recoder를 제외한 나머지 회로는 150MHz의 속도로 동작 가능하여 스칼라 곱셈에 약 1.31-ms가 소요될 것으로 평가되었다.

a	:2	5c4beac8	074b8c2d	9df63af9	1263eb82	29b3c967
g	:8	00000000	00000000	00000000	00000000	00000107
k	:6	a05cd513	a05cd513	a05cd513	a05cd513	00000001
P_x	:1	3029482d	3635dacb	35723dca	6a84bef1	358ac365
P_y	:2	479bc536	a5927c38	4795bde0	7325fd91	598cfe32
Q_x	:4	efd142fa	dea0286f	7325fd91	5d0f4617	3c976490
Q_y	:3	14166c4	e448b445	e007da5a	2b4ead5f	88666808

그림 10. 설계된 스칼라 곱셈기의 기능검증 결과
Fig. 10. Functional simulation results of the designed scalar multiplier

표 1은 설계된 스칼라 곱셈기의 성능을 참고문헌의 사례와 비교한 것이다. 본 논문의 스칼라 곱셈기는 문헌[12]의 경우와 비교하여, 게이트 수가 약 64% 많이 소요되지만, 스칼라 곱셈에 소요되는 클록 수와 20MHz 환산 수행시간이 약 65% 감소되어 고속 연산이 가능하다. 한편, 문헌 [14]의 경우와 비교하면, 20MHz 환산 수행시간은 약 2배 소요되나, 게이트 수는 약 45% 감소되었다.

표 1. 스칼라 승산기의 성능 비교
Table 1. Comparison of scalar multipliers

	gates	cycles	20MHz 환산 수행시간 [ms]
$GF(2^{167})$[12]	20k (1.0)	526,718 (1.0)	26.335 (1.0)
$GF(2^{163})$[13]	24k (1.2)	258,000 (0.5)	12.9 (0.5)
$GF(2^{193})$[14]	59k (2.95)	83,268 (0.16)	4.163 (0.16)
본 논문의 설계	32,768 (1.64)	183,400 (0.35)	9.17 (0.35)

VI. 결 론

본 연구에서는 차세대 공개키 암호 시스템에 사용되는 타원곡선 암호 시스템을 위한 스칼라 곱셈기를 설계하였다. 스칼라 곱셈을 위해 유한체 연산기를 이용해 점 덧셈 연산기와 점 두배 연산기를 구현하였으며, 스칼라 k를 NAF로 변환하여 연산량을 감소시켰다. NAF 변환을 위해 CR 알고리듬을 이용하였으며, $GF(2^{163})$에서 최적화된 회로를 구현하였다. 설계된 타원곡선 스칼라 곱셈기는 32,768개의 게이트로 구현되었으며, 150-MHz @3.3-V로 동작하여 스칼라 곱셈에 약 1.31-ms가 소요될 것으로 평가되었다.

참고문헌

[1] Certicom research, *The Elliptic Curve Cryptosystem*, Certicom, April 1997.

[2] C.D. Walter, "Systolic modular multipli- cation", *IEEE Trans. on Computers*, vol. 42, no. 3, pp. 376-378, Mar., 1993

[3] A. Menezes, *Elliptic curve pubic key cryptosystem*, Kluwer Academic Publishers, 1993.

[4] R. Schroeppel, H. Orman, S. O'Malley and O. Spatscheck, "Fast key exchange with elliptic curve systems", *Advances in Cryp- tology CRYPTO 95*, pp. 43-56, 1995.

[5] P. Balasubramaniam and E. Karthikeyan, "Elliptic curve scalar multiplication algorithm using complementary recoding", Elsevier, 2007.

[6] C.C Chang, Y.T. Kuo, and C.H. Lin, "Fast algorithm for common-multiplicand multipli- cation and exponentiation by performing complements", *Proc. of the 17th Int. Conf. on Advanced Information Networking and Application (AINA'03)*, pp. 807-811, 2003.

[7] A.D. Booth, "A signed binary multiplication technique", *Journal of Applied Mathematics* 4, pp. 236-240, 1951.

[8] K. Okeya, "Signed binary representations revisited", *Proceedings of CRYPTO'04*, pp. 123-139, 2004

[9] S.K. Jain, L. Song and K.K. Parhi, "Efficient Semi-Systolic Architectures for Finite Field Arithmetic", *IEEE Trans. VLSI Syst.*, vol. 6, no. 1, pp. 101-113, Mar. 1998.

[10] J.H. Guo and C.L. Wang, "Systolic Array Implementation of Euclid's Algorithm for Inversion and Division in $GF(2^m)$", *IEEE Trans. Computers.*, vol. 47, no. 10, pp. 1161- 1167, Oct. 1998.

[11] 김창훈, 권순학, 홍춘표, 유기영, "타원곡선 암호프로세서의 재구성형 하드웨어 구현을 위한 $GF(2^m)$ 상의 새로운 연산기", *정보과학회 논문지*, 제31권 제8호, pp. 453-464, 2004.

[12] G. Orlando, C. Paar, "A Super-Serial Galois Fields Multiplier for FPGAs and its Application to Public-Key Algorithms", *Pro. of 7th Annual IEEE Symp. on Field-Pro- grammable Custom Computing Machines*, pp. 232-239, 1999.

[13] 최용제, 김호원, 김무섭, 박영수, "IC 카드를 위한 polynomial 기반의 타원곡선 암호시스템 연산기 설계", *대한전자공학회 하계종합학술대회 논문집*, 제24권 제1호, pp. 305-308, 2001

[14] 문상국, "소프트웨어/하드웨어 최적화된 타원곡선 유한체 연산 알고리즘의 개발과 이를 이용한 고성능 정보보호 SoC 설계", *한국해양정보통신학회 논문지*, 제13권 제2호, pp. 293- 298, 2009.

감사의 말

반도체설계교육센터(IDEC)의 CAD Tool 지원에 감사드립니다.

저자소개

정상혁 (Sang-Hyeok Jeong)

2008년 8월 금오공과대학교 전자공학과 졸업
2009년 9월~현재 : 금오공과대학교 석사과정

※관심분야 : 통신 및 신호처리용 SoC 설계, 정보보호 SoC 설계, 반도체 IP 설계

신경욱(Kyung-Wook Shin)

1984년 2월 한국항공대학교 전자공학과(공학사)
1986년 2월 연세대학교 대학원 전자공학과(공학석사)
1990년 8월 연세대학교대학원 전자공학과(공학박사)
1990년 9월~1991년 6월 한국전자통신연구소
1991년 7월~현재 금오공과대학교 전자공학부(교수)
1995년 8월~1996년 7월 University of Illinois at Urbana-Champaign(방문교수)
2003년 1월~2004년 1월 University of California at San Diego(방문교수)

※관심분야 : 통신 및 신호처리용 SoC 설계, 정보보호 SoC 설계, 반도체 IP 설계