論文2002-39SC-3-4

GF(2^m)상에서 셀룰러 오토마타를 이용한 곱셈/제곱 동시 연산기 설계

(Design of New Architecture for Simultaneously Computing Multiplication and Squaring over GF(2^m) based on Cellular Automata)

具 敎 敏 *, 夏 璟 珠 **, 金 炫 成 ***, 柳 基 永 ****

(Kyo Min Ku, Kyeoung Ju Ha, Hyun Sung Kim, and Kee Young Yoo)

요 약

본 논문에서는 셀룰러 오토마타를 이용하여, GF(2^m)상에서 모듈러 곱셈과 제곱의 연산을 m 클럭 사이클 만에 동시에 처리할 수 있는 연산기를 설계하였다. 이는 Diffie-Hellman key exchange, ElGamal과 같은 대부분의 공개키 암호화 시스템에서의 기본 연산인 유한 필드 상의 모듈러 지수승 연산기 설계에 효율적으로 이용될 수 있다. 또한 셀룰러 오토마타는 간단하고도 규칙적이며, 모듈화 하기 쉽고 계층화 하기 쉬운 구조이므로 VLSI 구현에도 효율적으로 활용될 수 있다.

Abstract

In this paper, a new architecture that can simultaneously process modular multiplication and squaring on GF(2^m) in m clock cycles by using the cellular automata is presented. This can be used efficiently for the design of the modular exponentiation on the finite field which is the basic computation in most public key crypto systems such as Diffie-Hellman key exchange, ElGamal, etc. Also, the cellular automata architecture is simple, regular, modular, cascadable and therefore, can be utilized efficiently for the implementation of VLSI.

Keyword : modular multiplication, modular squaring, crytosystem, cellular automata

* 正會員, 大邱敎育大學敎
(Daegu National Univ. of Education)
** 正會員, 慶山大學校 情報科學部
(Dept. of Information Processing, Kyungsan Univ.)
*** 正會員, 慶一大學校 컴퓨터工學
(Dept. of Computer Engineering, KyungIl Univ.)
**** 正會員, 慶北大學校 컴퓨터工學
(Dept. of Computer Engineering, Kyungpook Nationl Univ.)
接受日字:2001年11月15日, 수정완료일:2002年4月8日

Ⅰ. 서 론

최근 인터넷의 급속한 확산으로 인해 다양한 형태의 정보 및 정보 서비스를 언제라도 손쉽게 얻을 수 있게 되었다. 그러나 그 편리함과 유익성에 비례하여 위험하고 파괴적인 역기능이 뒤따르고 있는 것도 사실이다. 이에 따라 정보보호에 대한 필요성이 대두되어 여러 가지 보안기술이 개발되고 있는 실정이며, 정보보호의 핵심 기술이라 할 수 있는 암호화 시스템의 구현의 중요성이 점점 더 크게 부각되고 있다. 지난 30여년간 암

호화 시스템 등 여러 분야에서 유한 필드에 대한 연구가 이루어 졌으며[1], Diffie-Hellman key exchange, ElGamal과 같은 대부분의 공개키 암호화 시스템에서는 유한 필드 상의 모듈러 지수승(modular exponetiation) 연산을 기본으로 하고 있다[2-3]. 이러한 모듈러 지수승 연산기에서는 이의 구현을 위해 모듈러 곱셈(modular multiplication) 연산기를 기본 구조로서 사용하고 있다. 또한 타원 곡선 암호화 시스템에서는 정수배의 곱셈 연산을 기본으로 하고 있다[4]. 곱셈기를 구현하기 위한 알고리즘으로는 LSB 우선 곱셈 알고리즘[5]과 MSB 우선 곱셈 알고리즘[6] 및 몽고메리(montgomery) 알고리즘[7] 등이 있다.

본 논문에서는 공개키 암호 시스템에서의 기본 연산인 GF(2^m)상에서 지수승을 위한 VLSI 구현 시, 간단하고도 규칙적이며, 모듈화 하기 쉽고 계층화 하기 쉬운 구조를 개발하는데 중점을 두고 있다. 셀룰러 오토마타(Cellular Automata : CA)는 이와 같은 성질을 잘 만족하여 암호화 시스템에서 대칭키의 암호화 복호화등 많은 응용 분야에 사용되고 있다[7-8].

지금까지 모듈러 곱셈 연산을 위해 개발된 연구 결과들은 다음과 같다. 먼저 1차원 시스톨릭 구조 상에서는 LSB 우선 알고리즘의 경우 m 셀을 사용하여 $3m$ 클럭 사이클에 모듈러 곱셈 연산을 하고 있으며[5], MSB 우선 알고리즘의 경우 m 셀을 사용하여 $3m$ 클럭 사이클에 모듈러 곱셈 연산을 하고 있다[6]. LFSR 구조 상에서는 [10]의 경우 m 셀을 사용하여 $2m$ 클럭 사이클에 모듈러 곱셈 연산을 하고 있으며, [11]의 경우 m 셀을 사용하여 m 클럭 사이클에 모듈러 곱셈 연산을, m 셀을 사용하여 m 클럭 사이클에 모듈러 제곱 연산을 하고 있다. [5, 6, 10]에서 제시한 구조는 단순한 모듈러 곱셈기로서, 지수승 계산을 위하여 모듈러 곱셈과 제곱 연산을 동시에 수행하기 위해서는 제시된 구조를 두 번 반복하여야 한다. [11]의 경우에도 모듈러 곱셈과 제곱 연산을 동시에 수행하기 위해서는 제시된 곱셈과 제곱 구조를 같이 사용하여야 한다.

[12]에서는 CA 구조상에서 m 셀 $2m$ AND 게이트, 2^m XOR 게이트, 4개 레지스터를 사용하여 m 클럭 사이클에 곱셈을 할 수 있는 알고리즘을 제시하고 있다. 그러나 [12]에서 제시한 구조 또한 단순한 모듈러 곱셈기로서, 지수승 계산을 위하여 모듈러 곱셈과 제곱 연산을 동시에 수행하기 위해서는 제시된 구조를 두 번 반복하여야 한다.

본 논문에서는 3-이웃 CA를 이용하여, GF(2^m)상에서 효과적인 지수승 계산을 위하여 모듈러 곱셈과 제곱의 계산을 동시에 처리할 수 있는 구조를 제시한다. 제시된 구조에서는 m 셀, $3m$ AND 게이트, $3m$-1 XOR 게이트 그리고 $5m$개의 레지스터를 사용하여 m 클럭 사이클만에 곱셈과 제곱 연산을 동시에 수행할 수 있다. 이는 LSB 우선 곱셈의 성질 중 모듈러 곱셈과 제곱 연산에서 공통적으로 행해질 수 있는 연산의 부분을 찾아내고, 그 나머지를 병렬로 처리 함으로써, [11]에서 제시된 곱셈과 제곱 구조를 함께 사용하는 것과 [12]에서 제시된 구조를 두 번 반복 사용하는 것 보다는 공간적인 면에서 훨씬 효율적인 계산을 할 수 있으며, [5, 6, 10]에서 제시된 구조를 두 번 반복하여 사용 할 때 보다 시간적인 면에서나 공간적인 면에서 훨씬 더 효율적인 계산을 할 수 있다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 CA에 대해 살펴보고, Ⅲ장에서는 GF(2^m) 상에서 지수승을 위한 구조를 살펴 본다. Ⅳ장에서는 CA를 이용한 곱셈/제곱 동시 수행 구조를 제시하고, Ⅴ장에서는 성능 분석 및 이에 대한 시뮬레이션 결과를 분석한다. 마지막으로 Ⅵ장에서 결론을 맺는다.

Ⅱ. CA

CA는 규칙적으로 상호 연결된 많은 셀들로 구성되어져 있는 유한 상태 머신(finite state machine)이다[8-9]. 각 셀들의 다음 상태는 각 셀들과 연결된 이웃의 현재 상태 값에 따라 달라지게 된다. 이와 같이 CA는 상태의 변화에 관여하는 이웃의 수와 이들 이웃을 이용하여 상태를 갱신하기 위해 사용되는 함수인 법칙으로 구성된다. 이 때 이웃이란 자기 자신을 포함하여 셀의 상태 갱신에 직접적으로 영향을 미칠 수 있는 셀을 의미한다. 다음은 2-상태, 3-이웃 1-차원 CA의 두 가지 법칙에 대한 예이다.

이웃의 상태: 111 110 101 100 011 010 001 000
다음 상태 : 0 1 0 1 1 0 1 0 (법칙 90)
다음 상태 : 1 0 0 1 0 1 1 0 (법칙 150)

여기서 이웃의 상태는 시간 t에 3개의 이웃이 가질 수 있는 가능한 8개의 상태이며, 이를 나타내는 3개의 비트 중 가운데 비트가 자신의 상태를 나타내고, 이의

왼쪽, 오른쪽 비트는 각각 왼쪽, 오른쪽 이웃의 상태를 나타낸다. 법칙 90과 법칙 150은 시간 t+1에 i 번째 셀이 가지는 상태를 나타내고 있다. 여기서 90과 150의 의미는 다음 상태 8비트를 십진수로 나타낸 수이다. 법칙 90을 살펴보면, 자신의 왼쪽 이웃과 오른쪽 이웃의 상태 값을 XOR 하여 그 결과값으로 자신의 상태를 갱신하고 있으며, 법칙 150은 자신의 왼쪽, 오른쪽 이웃 그리고 자신의 상태 값을 XOR 한 결과값을 자신의 다음 상태로 갱신하고 있음을 알 수 있다. 따라서 $q_i(t)$를 시간 t에서 i번째 셀의 상태 값이라 했을 때, 법칙 90과 150은 다음과 같은 수식으로 표현될 수 있다.

법칙 90 : $q_i(t+1) = q_{i-1}(t) \oplus q_{i+1}(t)$

법칙 150 : $q_i(t+1) = q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)$

여기서 $\oplus$은 XOR 연산을 나타내며, q_{i-1}은 q_i의 왼쪽 이웃을 q_{i-1}은 q_i의 오른쪽 이웃을 나타낸다.

CA에 적용된 법칙이 XOR 연산만으로 구성 될 경우 이를 선형(linear) CA라 하며, 그 외의 경우를 비선형(non-linear) CA 라 한다. 그리고 CA의 구조에서 경계조건(boundary condition)을 고려해야하는데, 가장 왼쪽 셀의 왼쪽 이웃과 가장 오른쪽 셀의 오른쪽 이웃을 모두 '0'으로 간주하는 NBCA(Null Boundary CA), 가장 왼쪽 셀과 가장 오른쪽 셀이 서로 이웃한 것으로 간주하는 PBCA(Periodic Boundary CA) 등이 있다.

그림 1은 법칙 <90, 150, 90, 150>을 갖는 2-상태, 3-이웃, 1-차원, 4-셀 NBCA의 예를 나타내고 있다.

n개의 셀을 가지는 CA의 현재 상태는 $n-$벡터 $x=(x_0, x_1, \cdots, x_{n-1})$로 나타낼 수 있다. 여기서 x_i는 셀 i의 값이며, x_i는 GF(2)의 원소이다. 선형 CA에서 다음의 상태는 특성 행렬과 현재 상태의 벡터를 곱함으로써 결정될 수 있는데, 이 때 특성 행렬은 CA의 전체적인 법칙을 나타낸다. x^t를 시간 t에서의 CA의 상태라 하고(하나의 열벡터로 간주), 특성 행렬을 T라하면, 시간 t+1에서의 CA의 상태는 다음과 같이 표현된다.

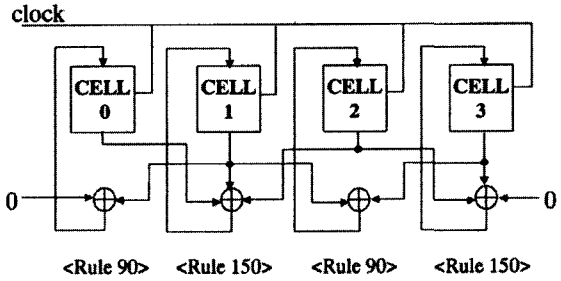


그림 1. 법칙 <90, 150, 90, 150>을 갖는 2-상태, 3-이웃, 1-차원, 4-셀 NBCA

Fig. 1. 2-state 3-neighbor 1-dimensional 4-cell NBCA having rules <90, 150, 90, 150>.

$$x^{t+1} = Tx^t$$

여기서의 산술 연산은 GF(2) 상에서 일어난다.

법칙 90에 대한 PBCA인 경우 특성 행렬은 다음과 같다.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

위의 예에서 i번째 행 j번째 열에 있는 행렬의 요소 '1'은 i번째 셀이 j번째 셀과 이웃에 대한 의존도가 있음을 나타낸다.

Ⅲ. GF(2^m) 상에서 지수승 연산을 위한 알고리즘

본 장에서는 GF(2^m) 상에서 $C(x)^E \bmod P(x)$를 구하는 일반적인 알고리즘에 대해 설명한다[13].

$B(x)$와 $C(x)$를 GF(2^m) 상의 한 원소라 하자. 그리고, GF(2^m) 상에서는 연산 후 연산 결과를 필드의 원소로 만들기 위해서 차수 m의 기약 다항식(irreducible polynomial) $P(x)$가 필요하며, 이 기약 다항식을 이용한 모듈러 연산이 필요하다.

그러면 세 개의 다항식 $B(x)$, $C(x)$, $P(x)$는 다음과 같다.

$$B(x) = b_{m-1}x^{m-1} + \cdots + b_1x^1 + b_0,$$
$$b_i \in GF(2),\ 0 \le i \le m-1.$$
$$C(x) = c_{m-1}x^{m-1} + \cdots + c_1x^1 + c_0,$$
$$c_i \in GF(2),\ 0 \le i \le m-1.$$

$$P(x)=x^m+p_{m-1}x^{m-1}+\cdots+p_1x^1+p_0,$$
$$p_i\in GF(2),\ 0\le i\le m-1.$$

먼저 $C(x)^E \bmod P(x)$의 계산은 지수 $E(=[e_{m-1}, e_{m-2},\cdots,e_1,e_0])$의 처리 방식에 따라 LSB 우선 방법과 MSB 우선 방법으로 나뉘어 지는데, 연산의 방법은 다음과 같다.

- LSB 우선 지수승 연산 : e_0 부터 e_{m-1} 순으로 연산
  $C(x)^E=C(x)^{e_0}(C(x)^2)^{e_1}(C(x)^4)^{e_2}\ldots(C(x)^{2^{m-1}})^{e_{m-1}}$
- MSB 우선 지수승 연산 : em-1 부터 e0순으로 연산
  $C(x)^E=(..((C(x)^{e_{m-1}})^2C(x)^{e_{m-2}})^2\ldots C(x)^{e_1})^2C(x)^{e_0}$

본 장에서는 LSB 우선 곱셈 방법에 대해 일반적인 알고리즘[13]에 대해 살펴본다.

알고리즘 1 : LSB 우선 곱셈에 의한 지수승 알고리즘[13]
입력 : $C(x),E,P(x)$
출력 : $B(x)=C(x)^E \bmod P(x)$
단계 1 : $B(x)=1, A(x)=C(x)$
단계 2 : for i=0 to m-1
단계 3 : if e_i ==1 $B(x)=A(x)B(x) \bmod P(x)$
단계 4 : $A(x)=A(x)A(x) \bmod P(x)$

알고리즘 1을 구현하기 위한 일반적인 방법은 두 개의 곱셈기를 이용하여 지수기를 설계하거나, 하나의 곱셈기와 하나의 제곱기를 이용하여 구현하는 방법을 이용한다. 그러나 본 논문의 다음 장에서는 두 가지 연산(모듈러 곱셈과 제곱 연산)의 공통 연산 부분을 찾아내어 이를 한번만 수행하고 나머지 연산을 병렬로 처리함으로써, 이를 효율적인 지수승 연산기 설계에 활용할 수 있음을 보여준다.

Ⅵ. 곱셈/제곱 동시 수행 연산기

본 장에서는 GF(2^m) 상에서 $C(x)^E \bmod P(x)$를 구함에 있어 필수적인 요소인 $A(x)B(x)$와 $A(x)A(x)$ 구조(알고리즘 1의 단계 3과 단계 4)의 곱셈을 CA를 이용하여 빠른 시간에 계산 할 수 있는 구조를 제시하고자 한다.

Ⅲ장에서 설명한 알고리즘 1에 의하면 LSB 우선 지수승의 계산을 위해서는 단계 3에 사용되는 모듈러 곱셈인 $M(x)=A(x)B(x)$와 단계 4에 사용되는 제곱인 $S(x)=A(x)A(x)$의 계산이 필요하다.

이 두 가지의 연산은 다음과 같이 표현될 수 있다.

$$\begin{aligned}M(x)&=A(x)B(x) \bmod P(x)\\&=b_oA(x)+b_1[A(x)x \bmod P(x)]+\\&\quad\cdots+b_{m-1}[A(x)x^{m-1} \bmod P(x)] \quad (1)\\S(x)&=A(x)A(x) \bmod P(x)\\&=a_oA(x)+a_1[A(x)x \bmod P(x)]+\\&\quad\cdots+a_{m-1}[A(x)x^{m-1} \bmod P(x)] \quad (2)\end{aligned}$$

위의 두 가지 방정식에서 [ ] 부분은 공통적으로 수행되는 부분이다. 따라서 곱셈과 제곱 연산을 동시에 수행함에 있어 공통적인 부분의 연산을 한번만 수행하여 이의 결과를 이용한다면 훨씬 더 효율적인 연산을 수행할 수 있을 것이다. (1)식은 다음과 같은 순환(recurrence) 형태로 다시 나타낼 수 있다.

$$\begin{aligned}&M^{(i)}(x)=M^{(i-1)}(x)+b_{i-1}A^{(i-1)}(x),\\&A^{(i)}(x)=A^{(i-1)}(x)x \bmod P(x)\ (1\le i\le m) \quad (3)\end{aligned}$$

여기서 $A^{(0)}(x)=A(x)$, $M^{(0)}(x)=0$, $M^{(i)}(x)=b_0A(x)+b_1[A(x)x \bmod P(x)]+b_2[A(x)x^2 \bmod P(x)+\cdots+b_{i-1}[A(x)x^{i-1} \bmod P(x)]$ 이고, $i=m$인 경우, $M^{(m)}(x)=M(x)=A(x)B(x) \bmod P(x)$이다.

(3) 식에서 두 개의 식은 병렬로 수행될 수 있다.

(2)번 식에서의 제곱 연산 또한 (3)번식과 비슷하게 LSB 우선 순환 형태로 다음과 같이 변환 할 수 있다.

$$\begin{aligned}&S^{(i)}(x)=S^{(i-1)}(x)+a_{i-1}A^{(i-1)}(x),\\&A^{(i)}(x)=A^{(i-1)}(x)x \bmod P(x)\ (1\le i\le m) \quad (4)\end{aligned}$$

여기서 $A^{(0)}(x)=A(x)$, $S^{(0)}(x)=0$, $S^{(i)}(x)=a_0A(x)+a_1[A(x)x \bmod P(x)]+a_2[A(x)x^2 \bmod P(x)+\cdots+a_{i-1}[A(x)x^{i-1} \bmod P(x)]$이고, $i=m$인 경우, $S^{(m)}(x)=S(x)=A(x)A(x) \bmod P(x)$이다.

(4) 식에서도 두 개의 식은 병렬로 수행될 수 있다. 따라서 본 논문에서는 식 (3)과 (4)에서 공통되는 부분인 $A^{(i)}(x)=A^{(i-1)}(x)x \bmod P(x)$ $(1\le i\le m)$를 중복 연산하지 않고, 한번만 연산하여 이의 결과를 이용하여 식(3)과 (4)의 나머지 부분을 각각 병렬로 구함으로써,

수행 시간은 모듈러 곱셈 연산과 동일하면서 모듈러 곱셈과 동시에 제곱의 연산까지 수행하여 $GF(2^m)$상에서 효율적인 지수승 계산을 할 수 있는 구조를 제시한다.

먼저 $A(x)$와 $B(x)$를 $GF(2^m)$ 상의 한 원소라 하면, 이들을 두 개의 이진 $m-$튜플(tuple)로 다음과 같이 나타낼 수 있다.

$$A(x)=(a_{m-1}...a_2a_1a_0) \quad (5)$$

$$B(x)=(b_{m-1}...b_2b_1b_0) \quad (6)$$

다음은 모듈러 곱셈과 제곱 연산에서 공통적으로 수행되는 $A^{(i)}(x)=A^{(i-1)}(x)x \bmod P(x)$ $(1\le i\le m)$를 수행하기 위한 기본적인 연산이다.

연산 1 : $A(x)$의 $m-$튜플을 cyclic left shift 한다. 그 결과는 다음과 같다.

$$(a_{m-2},\ldots,a_1,a_0,a_{m-1}) \quad (7)$$

연산 2 : 만약 $a_{m-1}=1$이면 (7)의 결과와 $(p_{m-1},\ldots,p_2,p_1,p_0+a_{m-1})$를 비트별로 모듈러 2 더하기 연산을 행한다. 즉 다음과 같은 연산을 행한다.

$$(a_{m-2},\ldots,a_1,a_0,a_{m-1})+a_{m-1}(p_{m-1},\cdots,p_2,p_1,p_0+a_{m-1})$$

연산 1을 수행하기 위하여, m개의 셀을 가지는 1차원 PBCA를 이용한다. CA의 m개의 셀에 $A(x)$를 입력하고, 특성 행렬은 각 셀들이 자기자신의 오른쪽 이웃의 값을 자신의 값으로 받고, 한 비트 순환 왼쪽 쉬프트를 위해 가장 왼쪽 셀과 가장 오른쪽 셀이 이웃한 PBCA의 특성을 가지므로, 다음과 같은 $m\times m$ 행렬 T를 사용한다.

이러한 특성 행렬을 가진 CA는 다음과 같이 법칙 170을 가지게 되며, 그 구조는 그림 2와 같다.

이웃의 상태: 111 110 101 100 011 010 001 000
다음 상태 : 1 0 1 0 1 0 1 0 (법칙 170)

이제부터 그림 2의 구조를 가지는 CA를 특성행렬 T를 가지는 PBCA라 하자.

연산 1의 결과로 얻어진 튜플을 z라 하자.
그 다음 연산 2를 수행하기 위해, $A(x)$의 MSB가 1이면, 즉 a_{m-1}이 1이면, $m-$튜플 z와 $(p_{m-1},\ldots,p_2,p_1,p_0)$를 비트별로 XOR 연산을 행한다.

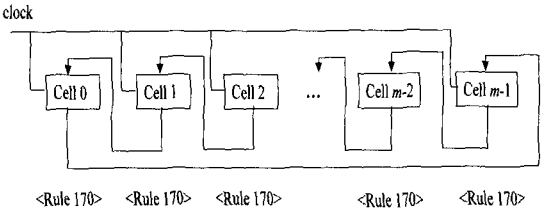


그림 2. 특성 행렬 T를 가지는 PBCA 구조
Fig. 2. PBCA structure having the characteristic matrix T.

이에 대한 구조도가 그림 3에 나타나 있다.

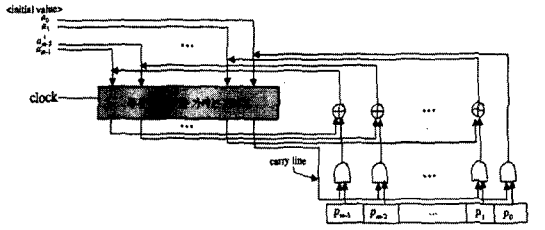


그림 3. 곱셈과 제곱의 공통 연산을 위한 구조도
Fig. 3. Structure of common operation of multiplication and squaring.

지금까지는 식(3)과 (4)의 공통부분을 동시에 수행하는 방법을 살펴보았다. 이제 이를 이용하여 모듈러 곱셈과 제곱 연산을 동시에 수행하는 방법에 대해 살펴본다.

$M^{(i)}(x)=M^{(i-1)}(x)+b_{i-1}A^{(i-1)}(x)$ $(1\le i\le m)$을 수행하기 위하여, 먼저 $b_{i-1}Az^{(i-1)}(x)$ 연산을 살펴본다. $b_{i-1}A^{(i-1)}(x)$ $(1\le i\le m)$연산을 위하여 그림 3의 CA에서 나온 결과 m 비트를, m개의 AND 게이트의 한 입력으로 하고, 나머지 하나의 입력은 b_{i-1}로한다. 그 다음 그 결과와 $M^{(i-1)}(x)$를 XOR하여 그 결과를 다시 $M^{(i-1)}(x)$에 저장한다. 초기의 각 $M^{(i-1)}(x)$ $(1\le i\le m)$ 레지스터의 값은 0으로 초기화 한다. 이를 위한 구조는 그림 4와 같다. 그림 4에서는 i번째 클럭에서의 연산을 나타내고 있다.

이제 제곱 연산을 수행하여야 하는데, 자세히 살펴보면 제곱연산은 모듈러 곱셈 연산에서 $B(x)$를 $A(x)$로 대치하면 된다. 따라서 위의 그림 4의 구조도에서 b_i 대신 a_i를 사용하면 된다.

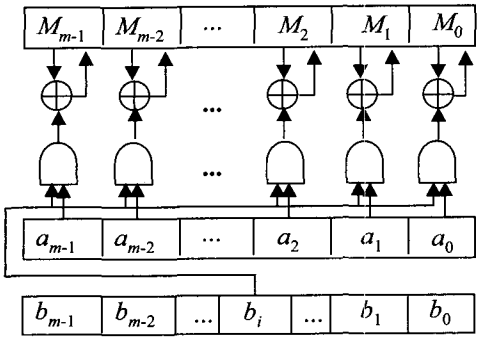


그림 4. 모듈러 곱셈을 위한 구조도
Fig. 4. Structure of modular multiplication.

이제 그림 3, 그림 4에서 제시된 구조도를 결합하면 지수승 연산의 기본이 되는 모듈러 곱셈 연산과 제곱 연산을 CA를 사용하여 동시에 수행 할 수 있는 그림 5와 같은 구조도가 생성된다. 그림 5는 i번째 클럭에서의 연산을 나타내고 있으며, 첫번째 연산에 앞서 특성행렬 T를 가지는 PBCA와 A, B, S, M, P $(0 \le i \le m-1)$레지스터에 초기값이 설정된다. 각각의 초기값은 다음과 같다.

- 특성행렬 T를 가지는 PBCA 초기값 :
  $A(x) = a_{m-1}...a_2a_1a_0$
- A 레지스터 초기값 : $A(x) = a_{m-1}...a_2a_1a_0$
- B 레지스터 초기값 : $B(x) = b_{m-1}...b_2b_1b_0$
- P 레지스터 초기값 : $P(x) = p_{m-1}...p_2p_1p_0$
- S, M 레지스터 초기값 : 0

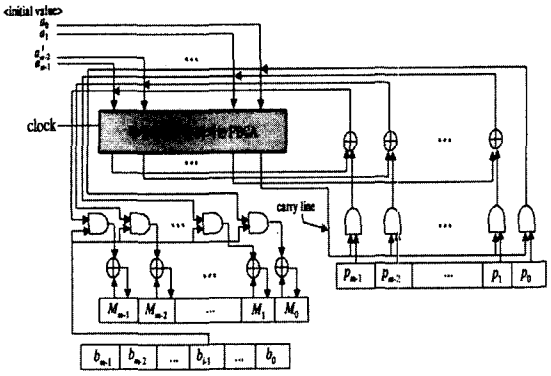


그림 5. CA를 이용하여 모듈러 곱셈과 제곱 연산을 동시에 수행하는 구조도
Fig. 5. Structure of simultaneously performing modular multiplication and squaring by using CA.

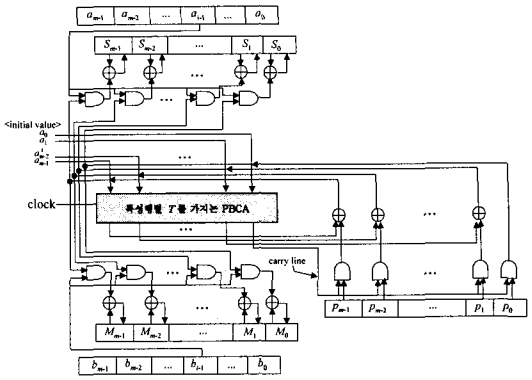


그림 6. CA를 이용하여 모듈러 곱셈과 제곱 연산을 동시에 수행하는 구조도
Fig. 6. Structure of simultaneously performing modular multiplication and squaring by using CA.

그림 5에서 제시된 구조를 사용 시 m개의 셀과 $3m$ AND 게이트 그리고 $3m$-1 XOR 게이트, $5m$개의 레지스터를 사용하여 m 클럭 사이클만에 곱셈과 제곱 연산을 동시에 수행할 수 있다.

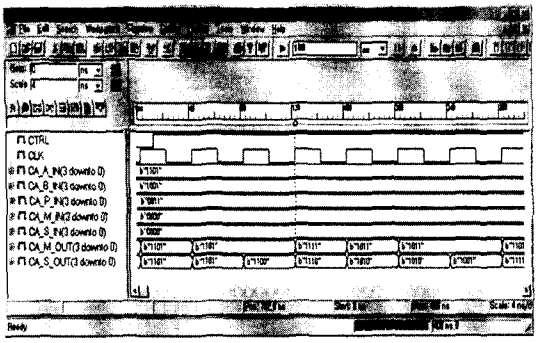
V. 성능 분석 및 시뮬레이션 결과

본 논문의 결과를 지금까지의 연구결과와 비교해 보면 <표 1>과 같다.

표 1. 여러 구조 상에서의 연구 결과 비교
Table 1. Comparison of performance on many structures.

구조	시스톨릭 구조		LFSR 구조				CA 구조	
	[5]	[6]	[10]	[11]			[12]	본 논문
수행 연산	곱셈 연산	곱셈 연산	곱셈 연산	곱셈 연산	제곱 연산	곱셈, 제곱 연산 동시 수행 시	곱셈 연산	곱셈 연산과 제곱 연산 동시 처리
수행 방법	LSB우선	MSB우선		MSB우선	MSB우선	MSB우선	LSB우선	LSB우선
셀의 개수	$m(2m)$	$m(2m)$	$m(2m)$	m	m	$2m$	$m(2m)$	m
AND게이트 수	$3m(6m)$	$3m(6m)$	$2m(4m)$	$2m$	$2m$	$4m$	$2m(4m)$	$3m$
XOR게이트 수	$2m(4m)$	$2m(4m)$	$2m$-$2(4m$-$4)$	$2m$	$2m$	$4m$	$2m(4m)$	$3m$-1
래치 수	$11m(22m)$	$9m(18m)$	0	0	0	0	0	0
MUX 수	$m(2m)$	1(2)	0	0	0	0	0	0
제어신호 수	2(4)	1(2)	1(2)	0	0	0	0	0
래지스터 수	0	0	$2m(4m)$	$4m$	$3m$	$7m$	$4m(8m)$	$5m$
수행 시간 (클럭 사이클)	$3m(3m)$	$3m(3m)$	$2m(2m)$	m	m	m	$m(m)$	m

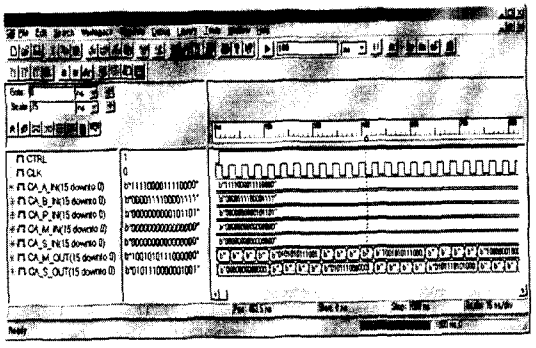
* ( )는 곱셈 연산만을 지원하는 각 구조에서 곱셈과 제곱 연산을 동시 수행 시 소요되는 비용



(a) 4비트 곱셈/제곱 동시 연산 시뮬레이션 결과

입 력 값					출 력 값				
A(x)	1	1	0	1	M(x) _out	1	1	1	1
B(x)	1	0	0	1	S(x) _out	1	1	1	0
P(x)	0	0	1	1					
M(x)	0	0	0	0					
s(x)	0	0	0	0					

(b) 4비트에서의 테스트 결과



(c) 16비트 곱셈/제곱 동시 연산 시뮬레이션 결과

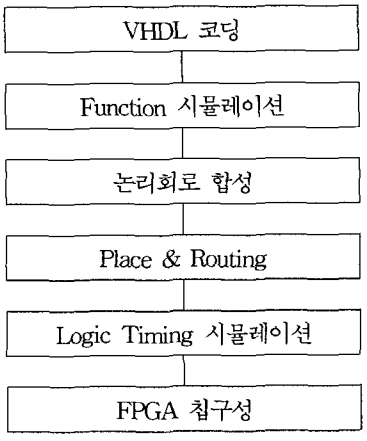
입력값																
$A(x)$	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
$B(x)$	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
$P(x)$	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1
$M(x)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$S(x)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
출력값																
$M(x)$ _out	1	0	0	1	0	1	0	1	1	1	0	0	0	0	0	0
$S(x)$ _out	0	1	0	1	1	1	0	0	0	0	0	0	1	0	0	1

(d) 16비트에서의 테스트 결과

그림 7. $GF(2^4)$와 $GF(2^{16})$에서의 시뮬레이션 결과
Fig. 7. Simulation results over GF(24) and GF(216).

[5, 6, 10]에서 제시된 구조를 사용하여 모듈러 곱셈 연산과 제곱 연산을 동시에 수행하기 위해서는 곱셈 구조를 두 번 반복하여야 하므로, 본 논문에서 제시된 구조가 시간, 공간적인 면에서 월등히 효율적이다. [12]의 경우는 $A^{(i)}(x)=A^{(i-1)}(x)x \bmod P(x)\ (1\le i\le m)$의 연산에 사용되는 구조를 제시하고 이 때 사용된 게이트, 레지스터의 수를 분석하고 있다. 따라서 곱셈을 위하여 나머지 연산 ($M^{(i)}(x)=M^{(i-1)}(x)+b_{i-1}A^{(i-1)}(x)\ (1\le i\le m)$)을 수행하기 위해서는 추가의 게이트와 레지스터가 필요하게 되어 전체 m 셀, $2m$개의 AND 게이트, $2m$개의 XOR 게이트, $4m$개 레지스터가 필요하다. 결과적으로 [12]의 경우 역시 모듈러 곱셈과 제곱 연산을 동시에 처리하기 위해서는 제시된 구조를 두 번 반복하여야 하므로, 본 논문에서 제시한 구조와 비교해 볼 때 시간적인 면에서는 동일하나, 공간적인 면에서 본 논문에서 제시한 구조가 훨씬 효율적이다. [11]의 경우에도 모듈러 곱셈과 제곱 연산을 동시에 처리하기 위해서는 제시된 곱셈과 제곱 구조를 같이 사용하여야하므로, 본 논문에서 제시한 구조와 비교해 볼 때 시간적인 면에서는 동일하나, 공간적인 면에서 본 논문에서 제시한 구조가 훨씬 효율적임을 알 수 있다.

본 논문에서 제시한 구조가 제대로 동작됨을 보이기 위하여, VHDL로 모델링한 곱셈/제곱기로부터 Synopsys사의 합성 툴(FPGA-Express Version : 2000.11-FE 3.5.1)을 사용하여 다음과 같은 절차로 논리 합성을 수행하였다.



그 결과 그림 6과 같은 결과를 얻음으로써 본 논문에서 제시한 구조가 m클럭 사이클만에 올바르게 동작됨을 확인할 수 있었다.

VI. 결 론

Diffie-Hellman key exchange, ElGamal과 같은 대부분의 공개키 암호화 시스템에서는 유한 필드 상의 모듈러 지수승 연산을 기본으로 하고 있다[2-3]. 이러한 모듈러 지수 연산기에서는 이의 구현을 위해 모듈러 곱셈 연산기를 기본 구조로서 사용하고 있다.

본 논문에서는 이러한 유한 필드상의 모듈러 지수 연산의 기본 연산인 모듈러 곱셈과 제곱 연산을 동시에 처리할 수 있는 구조를 제시하였다. 제시된 구조에서는 LSB 우선 곱셈의 성질 중 모듈러 곱셈과 제곱 연산에서 공통적으로 행해질 수 있는 연산의 부분을 찾아내고, 그 나머지를 병렬로 처리함으로써, m개의 셀과 $3m$개 AND 게이트와 $3m$-1개 XOR 게이트 그리고 $5m$개 레지스터를 사용하여 m 클럭 사이클만에 곱셈과 제곱 연산을 동시에 수행할 수 있었다.

또한 본 논문에서 사용된 CA구조는 간단하고도 규칙적이며, 모듈화 하기 쉽고 계층화 하기 쉬운 구조이므로 VLSI 구현이 용이하다. 그리고 본 논문에서 제시된 모듈러 곱셈과 제곱 연산기를 기반으로 하여 공개키 암호화 시스템에서 기반 구조로 사용되는 모듈러 지수기, 나눗셈기, 곱셈의 역원기 등을 효율적으로 구현할 수 있다.

참 고 문 헌

[1] R.J. McEliece, Finite Fields for Computer Scientists and Engineerings, New York : Kluwer Academic, 1987.

[2] W. Diffie and M.E. Hellman, "New directions in cryptography," IEEE Trans. on information theory, vol. 22, pp. 644-654, November 1976.

[3] T. ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Trans. on information theory, vol. 31(4). pp. 469-472, July 1985.

[4] A.J. Menezes, Elliptic Curve Public Key Cryptosystems, Kluwer Academic Publishers, 1993.

[5] C.-S. YEH, IRVING S. REED, T.K. TRUONG, "Systolic Multipliers for Finite Fields GF(2^m)," IEEE Trans. on computers, vol. C-33, no. 4, pp. 357-360, April 1984.

[6] C.L. Wang, J.L. Lin, "Systolic Array Implementation of Multipliers for Finite Fields GF(2m)," IEEE Trans. on circuits and systems, vol. 38, no. 7, pp. 796-800, July 1991.

[7] P.L. Montgomery, "Modular multiplication without trial division," Mathematics of Computation, 44(170) : 519-521, April 1985.

[8] M. Delorme, J. Mazoyer, Cellular Automata, KLUWER ACADEMIC PUBLISHERS, 1999.

[9] STEPHEN WOLFRAM, Cellular Automata and Complexity, Addison-Wesly Publishing Company, 1994.

[10] ELWYN R. BERLEKAMP, "Bit-Serial Reed-Solomon Encoders," IEEE Trans. on information theory, vol. IT-28, no. 6, pp. 869-874, November 1982.

[11] C.Parr, "Fast Arithmetic for Public-Key Algorithms in Galois Fields with Composite Exponents," IEEE Trans. on computers, vol. 48, no. 10, pp. 1025-1034, October 1999.

[12] P.P. Choudhury, R. Barua, "Cellular Automata Based VLSI Architecture for Computing Multiplication And Inverse In GF(2m)," IEEE Proceeding of the 7th International Conference on VLSI Design, pp. 279-282, January 1994.

[13] Knuth, THE ART OF COMPUTER PROGRAMMING, vol. 2/Seminumerical Algorithms, ADDISON-WESLEY, 1969.

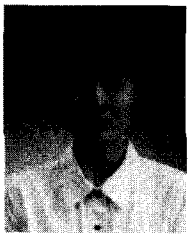
저 자 소 개

具 敎 敏(正會員)

1993년 : 경북대학교 컴퓨터공학과 학사. 1995년 : 경북대학교 대학원 컴퓨터공학과 석사. 2000년 : 경북대학교 대학원 컴퓨터공학과 박사 과정수료. 1995년 3월~1999년 1월 : 한국과학기술정보연구원 연구원. 1999년 3월~현재 : 대구교육대학교 전임강사. <관심분야 : 정보보안, 암호화시스템, 병렬처리>

金 炫 成(正會員)

1996년 : 경일대학교 컴퓨터공학과 졸업(공학사). 1998년 : 경북대학교 컴퓨터공학과 졸업(공학석사). 2002년 : 경북대학교 컴퓨터공학과 졸업(공학박사). 2002년~현재 : 경일대학교 컴퓨터공학과 전임강사. <관심분야 : 정보보안, 암호화 프로세서 설계, PKI, 병렬처리>



夏 璟 珠(正會員)

1987~1991 : 경북대학교 공과대학 컴퓨터공학과 졸업(공학사). 1991~1993 : 경북대학교 공과대학 대학원 컴퓨터공학과 졸업(공학석사). 1993~1996 : 경북대학교 공과대학 대학원 컴퓨터공학과 졸업(공학박사). 1996~1999 : 한국전자통신연구원 선임연구원. 1999~현재 : 경산대학교 정보과학부 조교수. <관심분야 : 정보보안, 암호화시스템설계, 병렬알고리즘>

柳 基 永(正會員) 第38券 SP編 第4號 參照