通信情報保護學會論文誌
第 10 券 第 4 號, 2000. 12

RSA 암호시스템을 위한 모듈러 지수 연산 프로세서 설계

허 영 준*, 박 혜 경*, 이 건 직**, 이 원 호**, 유 기 영**

Design of Modular Exponentiation Processor for RSA Cryptography

Young-Jun Heo*, Hae-Kyeong Park*, Keon-Jik Lee**, Won-Ho Lee**, Kee-Young Yoo**

요 약

본 논문에서는 몽고메리 알고리즘을 사용하여 모듈러 곱셈을 빠르게 수행하는 선형 시스톨릭 어레이를 설계하고, 이 곱셈기와 LR 이진 제곱 곱셈 알고리즘을 사용하여 n 비트 메시지 블록에 대해 모듈러 지수 연산을 수행하는 지수 연산 프로세서를 설계한다. 이 프로세서는 제어장치, 입출력 시프트 레지스터, 지수 연산 장치 등 3개의 영역으로 나누어진다. 설계된 지수 연산 프로세서의 동작을 검증하기 위해 VHDL를 사용하여 모델링하고 MAX+PLUS II를 사용하여 시뮬레이션 한다. 메시지 블록의 길이 n=512일 때 설계된 지수 연산 프로세서의 지연 시간은 59.5ms이다. 설계된 모듈러 지수 연산 프로세서는 RSA 칩(chip)에 이용될 수 있을 것이다.

ABSTRACT

In this paper, we design modular multiplication systolic array and exponentiation processor having n bits message block. This processor uses Montgomery algorithm and LR binary square and multiply algorithm. This processor consists of 3 divisions, which are control unit that controls computation sequence, 5 shift registers that save input and output values, and modular exponentiation unit. To verify the designed exponention processor, we model and simulate it using VHDL and MAX+PLUS II. Consider a message block length of n=512, the time needed for encrypting or decrypting such a block is 59.5ms. This modular exponentiation unit is used to RSA cryptosystem.

keyword : *RSA Cryptosystem, Modular Processor, Systolic Array, Montgomery Algorithm*

Ⅰ. 서 론

정보화 사회에서 음성, 화상, 데이터 등 다양한 종류의 정보를 교환하고 저장하는 대량 정보 통신 시스템에서 이 시스템의 신뢰성과 안전성이 필수 불가결한 요건이며, 특히 시스템 내부 또는 각 시스템 상호간 통신에서의 각종 정보에 대한 보안(security) 문제를 해결하기 위해 암호화 알고리즘과 그것의 효율적인 구현에 대한 연구가 이루어지고 있다. 여러 암호화 기법 중에서 1978년 공개된 RSA 공개키 암호시스템이 현재 가장 널리 사용되고 있다[1].

RSA 암호시스템에서 송신자는 모든 사용자들에게 공개되는 수신자의 공개키 e를 이용하여 암호화된 메시지 $C=M^e \pmod N$을 보낸다. 수신자는 자신의 비밀키 d를 이용해서 원래 메시지 $M=C^d \pmod N$을 복원한다. 암호시스템의 안전성을 확보하기 위해

* 한국전자통신연구원(yjheo@etri.re.kr)
** 경북대학교 컴퓨터공학과

M, e, N은 512비트 이상의 아주 큰 수를 사용하기 때문에 모듈러 지수 연산 속도가 RSA 암호시스템 성능에 큰 영향을 미친다. 모듈러 지수 연산은 단순히 $AB \pmod N$ 형태의 곱셈을 연속적으로 수행함으로 결과를 얻을 수 있는데, 빠른 암호시스템을 구현하기 위해 모듈러 곱셈 연산의 단위 수행시간을 단축시키는 것으로 빠른 곱셈 알고리즘을 제안하거나 하드웨어로 곱셈기를 설계하였고[4,7~9], 모듈러 곱셈 수행 회수를 줄이는 방법에 대한 연구가 이루어졌다[2~7,12~14].

RSA 암호시스템은 암호화 및 복호화 때 처리 속도의 지연이 가장 큰 문제가 되므로 모듈러 지수 연산의 고속화를 위한 하드웨어 구현이 필수적이다. 첫 번째로 개발된 모듈러 지수 연산 프로세서는 분산된 여러 요소들을 하나의 보드위에 모아서 모듈러 지수 연산을 수행하였다. 1980년 Rivest, Shamir와 Adleman은 512비트 모듈러 연산기를 1개의 칩(chip)으로 구현하였다. 1981년 Sandia National 연구소에서 336비트 칩을 개발하였는데 이는 두 개의 칩을 합성함으로써 336비트 키를 사용하였다. Ivey는 512비트 모듈러스를 사용하는 RSA 암호시스템을 구현하였고[19], Orup은 모듈러스 길이가 561비트이고 2^5인 곱셈기를 사용하는 암호 시스템을 발표하였다[3].

본 논문에서는 RSA 암호시스템에 사용할 수 있는 단일 칩 모듈러 지수 연산 프로세서를 설계한다. 우선 Montgomery 알고리즘을 이용하여 모듈러 곱셈 시스톨릭 어레이를 설계하고 제어 장치와 입출력 레지스터 등 주변장치들을 추가하여 지수 연산을 수행하는 프로세서를 설계한다. 설계된 지수 연산기의 하드웨어 구현 가능성과 성능을 알아보기 위해 VHDL을 이용하여 모델링하고 ALTERA사의 MAX+PLUS II[21]를 사용하여 시뮬레이션한다.

본 논문의 구성은 다음과 같다. 먼저 2장에서 모듈러 곱셈을 수행하는 선형 시스톨릭 어레이에 대해 기술하고, 3장에서는 LR 이진 제곱 지수 연산 알고리즘을 이용하여 모듈러 지수 연산 프로세서를 설계하고, VHDL을 이용하여 시뮬레이션 한다. 마지막으로 4장에서는 결론을 내린다.

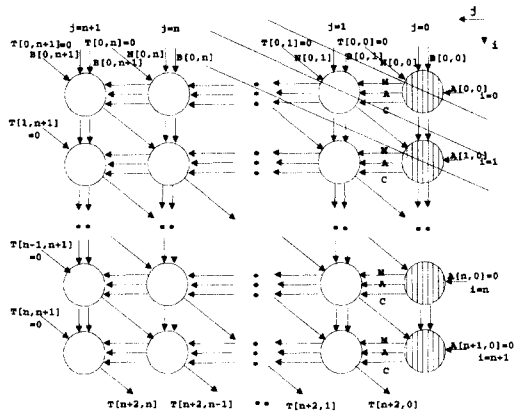
II. 모듈러 곱셈 시스톨릭 어레이

모듈러 곱셈 알고리즘에는 $2n$ 자리수 C를 n 자리수 N에 대해 모듈러 감소시키는 고전적인(Classic) 알고리즘[7], 한번에 몫을 추정하여 모듈러 감소 연산을 하는 Barrett 알고리즘[4], 큰 수를 r^n으로 나누었을 때 몫과 나머지를 구하는 것은 각각 그 수의 n자리 이상의 수를 취하는 것과 n자리 미만의 수를 취하게 하여 빠른 모듈러 감소 연산을 수행하게 하는 Montgomery 알고리즘[8], N이 주어지면 적당한 N의 배수 512개를 Table[512]에 저장하고 이것을 이용하여 모듈러 감소를 하는 Selby 알고리즘[9] 등이 있는데, 이중에서 Montgomery 알고리즘이 가장 빠르다고 알려져 있다[10,11].

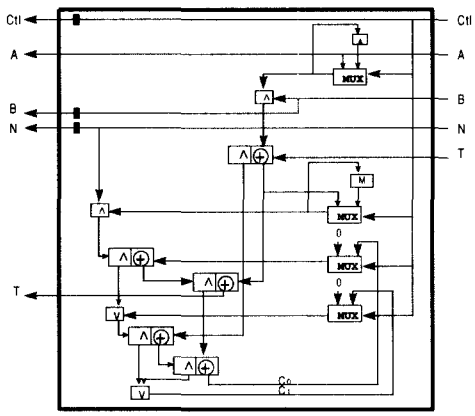
Montgomery 알고리즘을 이용하여 모듈러 곱셈을 수행하는 시스톨릭 어레이 설계에 대해 많은 연구가 있었다[2,21~23]. Sauerbrey[2]는 곱셈 연산을 수행하는 두개의 시스톨릭 어레이로 Montgomery 모듈러 곱셈기를 만들었고, Iwamura[22]등은 Montgomery 모듈러 곱셈을 수행하는 선형 시스톨릭 어레이를 설계하였다. Walter[23]는 평면 시스톨릭 어레이를 제안하였다. 본 논문에서는 Montgomery 알고리즘을 이용하여 모듈러 곱셈을 수행하는 선형 시스톨릭 어레이를 설계한다.

Montgomery 모듈러 곱셈 알고리즘을 분석하여 빠른 모듈러 연산을 수행할 수 있도록 알고리즘을 개선하고 정규 순환방정식으로 표현하면 아래와 같다.

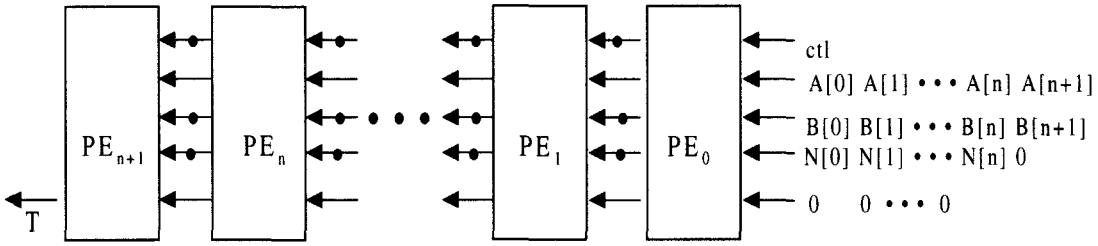
for $i=0$ *to* $n+1$ *do*
 for $j=0$ *to* $n+1$ *do* {
  if $(j=0)$ {
   $M[i,j+1]=(T[i,j]+A[i,j]B[i,0]) \bmod r$
   $C_0[i,j+1]=(T[i,j]+A[i,j]B[i,j]$
    $+M[i,j+1]N[i,j]+C[i,j]) \ div\ r \bmod r$
   $C_1[i,j+1]=(T[i,j]+A[i,j]B[i,j]$
    $+M[i,j+1]N[i,j]+C[i,j]) \ div\ r \ div\ r$
  }
  else {
   $M[i,j+1]=M[i,j]$
   $C_0[i,j+1]=(T[i,j]+A[i,j]B[i,j]$
    $+M[i,j]N[i,j]+C_0[i,j]$
    $+C_1[i,j]r) \ div\ r \bmod r$
   $C_1[i,j+1]=(T[i,j]+A[i,j]B[i,j]$
    $+M[i,j]N[i,j]+C_0[i,j]$
    $+C_1[i,j]r) \ div\ r \ div\ r$
   $T[i+1,j-1]=(T[i,j]+A[i,j]B[i,j]$
    $+M[i,j]N[i,j]+C_0[i,j]) \bmod r$
  }
  $A[i,j+1]=A[i,j] \quad B[i+1,j]=B[i,j]$
  $N[i+1,j]=N[i,j]$
}



〔그림 1〕 종속 그래프



〔그림 3〕 처리기 구조



〔그림 2〕 모듈러 곱셈 시스톨릭 어레이

〔표 1〕 모듈러 곱셈 시스톨릭 어레이의 성능 비교

	처리기수	수행시간	입출력 포트수	사용 시스톨릭 어레이수
Sauerbrey	$n/2+1$	$2n$	-	2
Iwamura	$n+1$	$3n+4$	$O(n)$	1
Walter	$(n+1)(m+2)$	$2n+m+2$	$O(n)$	1
Proposed Array	$n+2$	$3n+4$	$O(1)$	1

알고리즘에 내재되어 있는 병렬성을 찾기 위해 정규순환방정식을 종속 그래프로 나타내면 〔그림 1〕과 같다. 종속 그래프로부터 자료 흐름 벡터들과 초기값의 위치를 구한 다음, 대수적으로 시간-공간 변환을 통해 모듈러 곱셈을 수행하는 시스톨릭 어레이를 설계된 시스톨릭 어레이는 〔그림 2〕와 같고 모든 처리기의 구조는 〔그림 3〕과 같다.

모듈러스 N의 길이가 n 비트일 때 본 논문에서 제안된 시스톨릭 어레이와 기존의 다른 시스톨릭 어레이의 성능을 비교하면 〔표 1〕과 같다.

Ⅲ. 모듈러 지수연산 프로세서 설계

3.1 모듈러 지수연산 알고리즘

아주 큰수의 모듈러 지수연산은 RSA 암호 시스템의 기본적인 연산이며 일반 사용자들이 수용할 수 있을 정도의 빠른 속도로 수행되어져야 한다. 안정성을 확보하기 위해 C, M, N은 아주 큰 수이어야 하며, 일반적으로 512비트나 1024비트가 사용되고 있다. 이러한 모듈러 지수승은 소프트웨어나 하드웨어 상에서 모듈러 곱셈의 반복적인 연산으로 수행된다. 현재까지 RSA 암호시스템의 수행속도를 증가시키기 위해 많은 알고리즘이 제시되었다. 본 장에서는 기존의 지수 연산 알고리즘에 대해 알아본다.

원시 알고리즘은 $C=M \pmod N$으로부터 시작하여 $C=CM \pmod N$의 모듈러 곱셈 연산을 e-1번의 곱셈을 한다. 원시적인 방법을 개선하여 곱셈 수행 회수를 줄이기 위해 제안된 것이 "이진 제곱 곱셈(binary square and multiply)" 알고리즘이다[7]. 이진 제곱 곱셈 알고리즘은 지수 e의 이진 표

현에 근거하여 반복하여 제곱과 곱셈을 수행한다. $(e_{n-1}e_{n-2}e_{n-3},\cdots,e_1e_0)$를 지수 e의 이진 표현이라 하면 지수 e는 n비트 길이의 수이고 e_{n-1}는 최상위 비트(most significant bit)가 되고 e_0은 최하위 비트(least significant bit)가 되며, 지수 e의 이진 표현과 모듈러 지수 연산은 아래와 같다.

$$e=\sum_{i=0}^{n-1} e_i 2^i$$

$$C=M^e(\mathrm{mod}\,N)=\Pi_{i=0}^{n-1}M^{e_i 2^i}(\mathrm{mod}\ N)$$

이 알고리즘은 각 반복문이 단지 모듈러 곱셈과 간단한 이진 결정(binary decision)만을 취하기 때문에 특히 하드웨어 상에서 수행하기에 효율적이다. 이진 제곱 곱셈 알고리즘은 지수 e의 비트가 처리되는 순서에 따라 두 가지 방법이 있다. 최상위 비트에서 최하위 비트 순서로 처리하는 Left-to-Right(LR)과 최하위 비트에서 최상위 비트 순서로 처리하는 Right-to-Left(RL) 알고리즘이 있다. LR 이진제곱 알고리즘은 아래와 같다.

ModExp(M,e,N) : $C=M^e \bmod N$
$C=1$
for $i=n-1$ downto 0
 $C=CC \pmod N$
 if $e_i=1$ then $C=CM \pmod N$
return C

이 알고리즘은 지수 e를 이진수로 표현했을 때 "1"의 개수에 의해 모듈러 곱셈의 수행 회수가 결정된다. 평균적으로 e가 가지는 "1"의 개수가 $n/2$일 때 n번의 모듈러 제곱과 $n/2$번의 모듈러 곱셈이 수행되므로 전체 모듈러 곱셈의 연산 수행 회수는 $3n/2$번이다. 최악의 경우 $2n$번, 최상의 경우는 n번의 모듈러 곱셈 연산을 수행한다. 연산에 필요한 기억장치는 메시지 값 M과 계산 결과 C값을 저장하기 위한 기억공간이 필요하다.

이진 곱셈 제곱 알고리즘은 e의 비트를 1비트씩 스캔하였으나, 다른 방법으로 스캔되는 비트 수에 따라, 2비트씩 스캔하면 quaternary 방식, 3비트씩 스캔하면 octal 방식으로 분류된다. 일반적으로 스캔되는 비트 수가 log_2m이면 m-ary 방식이라 한다. 이 방식은 곱셈의 회수를 줄일 수 있어나 M^2값부터 M^{m-1}까지 값의 선계산(precomputation)이 필요하고 계산된 값들을 저장할 기억공간이 필요한 단점이 있어 하드웨어 구현이 용이하지 못하다.

또 다른 방법으로 제시된 알고리즘은 암호화 키 e의 이진 표현에 계수 "-1"을 추가하여 결과적으로 "0"이 아닌 비트 수를 줄여 곱셈 연산 회수를 줄이는 방법으로 이진 잉여 표현법(binary redundant representation)을 사용하는 수정된 부호화 디지트(Modified signed digit)알고리즘이 있다[13]. 이 알고리즘은 M^{-1}의 선계산을 하여 메모리에 저장하여야 하고 지수 e값의 표현에서 "0"이 아닌 수를 줄이는 선계산 과정을 수행하여야 하는 단점이 있다.

K-SR 알고리즘은 암호화 키 e를 표현함에 있어서 열 치환 표현 기법을 사용한다. 즉, 치환할 크기 K가 정해진 후에 $2\le i\le K$를 만족하는 i에 대해서 i개의 연속적인(consecutive) 비트 '1'을 2^i-1로 변환시키고, 2^K-1이하의 M의 지수(M^3,M^7,M^{15}...)를 선계산하여 메모리에 저장하고 필요시 사용하며 단지 연속적인 K개의 비트 '1'만을 K-SR 표현으로 변환시킬 수 있다[13].

최근에 제안된 $SS(l)$ 알고리즘은 e의 이진표현에서 고정된 1개의 크기 내에서 홀수(odd number)로 표현 가능한 비트 패턴을 찾아서 대치하는 것이다[18]. 이렇게 함으로써 모듈러 지수 연산시 선계산되는 연산의 수를 반으로 줄이고 또 선계산 결과를 저장하는 메모리 공간을 줄일 수 있다.

본 논문에서는 위에서 고찰한 알고리즘 중 수행에 필요한 기억장소와 수행 시간을 고려하여 하드웨어 구현이 가장 용이한 LR 이진 제곱 곱셈 알고리즘을 사용하여 모듈러 지수 연산 프로세서를 설계한다. Montgomery 곱셈 알고리즘(MMM)을 사용한 LR 이진 제곱 곱셈 알고리즘은 다음과 같다.

Montgomery ModExp(M,e,N) : $C=M^e \bmod N$
$\overline{M}=MMM(M,R^2)$
$\overline{C}=MMM(1,R^2)$
for $i=n-1$ *downto* 0
 $\overline{C}=MMM(\overline{C},\ \overline{C})$
 if $e_i=1$ then $\overline{C}=MMM(\overline{C},\ \overline{M})$
$C=MMM(\overline{C},\ 1)$
return C

여기서 R은 r^n을 의미한다. LR 이진제곱 곱셈 알고리즘에 Montgomery 모듈러 곱셈 알고리즘을 사용하

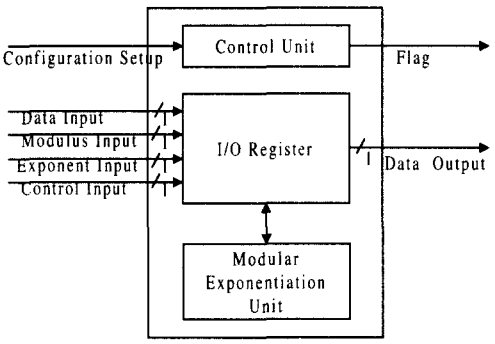
기 위해서는 정규 모듈러 수 체계를 Montgomery 모듈러 수 체계로 변환하여야 한다. 위 알고리즘에서 $\overline{M}$, $\overline{C}$는 M, C의 Montgomery 수 표현을 의미한다. 단계 1, 2에서는 정규 수 체계인 메시지 M과 결과 값을 저장하기 위한 C를 Montgomery 모듈러 수 체계 표현으로 변경하는 계산이 선행된다. 단계 3에서 모듈러 지수 연산이 수행될 때, 중간결과($\overline{C}$)의 제곱을 계산하기 위해 Montgomery 모듈러 곱셈 $MMM(\overline{C}, \overline{C})$을 수행하고 지수 ei의 값이 "1"이면 메시지($\overline{M}$)와 중간결과($\overline{C}$)의 Montgomery 모듈러 곱셈 $MMM(\overline{M}, \overline{C})$을 한번 더 수행한다. 단계 4에서 Montgomery 수 체계의 계산 결과를 정규 모듈러 수 체계로 변환하기 위해 '1'과 지수 연산 결과 값 $\overline{C}$를 사용하여 Montgomery 모듈러 곱셈 $MMM(\overline{C}, 1)$을 한번 더 수행한다. 여기서 정규 수 체계 표현을 Montgomery 수 체계 표현으로 변경하고 다시 이것을 정규 수 체계로 변경하는 부가적인 연산이 필요하나 모듈러스 N에 대해 $AA \bmod N$ 형태의 제곱 연산과 $AB \bmod N$ 형태의 곱셈 연산을 반복적으로 수행하는 시간에 비해 이 연산들의 계산 시간은 매우 미비하므로 부가적인 계산 시간을 무시할 수 있다.

3.2 모듈러 지수연산 프로세서 설계

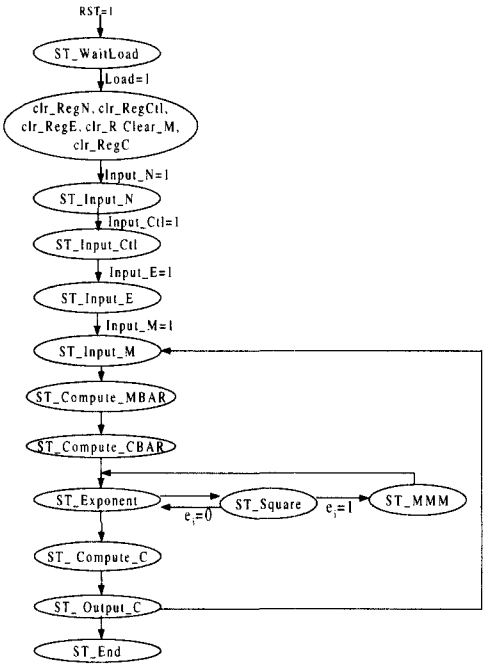
본 절에서는 2장에서 기술한 모듈러 곱셈기와 LR 곱셈 제곱 알고리즘을 이용하여 RSA 암호시스템에 사용되는 모듈러 지수 연산 프로세서를 설계한다. 모듈러 지수연산을 고속으로 수행하기 위해 설계한 지수 연산 프로세서는 사용할 키를 입력받아 레지스터에 저장하고 이것을 이용하여 입력되는 메시지를 암호화한다. 지수 연산 프로세서 내부에서 모든 데이터는 1비트 단위로 처리되며 외부와의 입출력도 1비트 단위로 이루어지는 범용성 처리기이다. 모듈러 지수 연산 프로세서는 [그림 4]와 같이 제어장치, 입출력 레지스터, 지수 연산 장치 등 3개 영역으로 구성된다.

3.2.1 제어장치(Control Unit)

제어장치는 사용자에 의해 선택된 구성에 대해 프로세서의 기능성을 구성하는 역할을 한다. 게다가 제어 장치는 지수 연산 프로세서의 계산 순서를 제어하고 인터페이스 프로토콜을 구현한다. 프로세서와의 통신은 사용자가 정의한 인터페이스 신호에 의해 제어되고 상태 플래그(state flag)는 프로세서에



[그림 4] 모듈러 지수 연산 프로세서



[그림 5] 제어장치의 상태 천이도

의해 생성된다. 제어장치는 각 장치의 동작을 제어하기 위해 제어 신호를 생성한다. 제어 장치를 유한 상태도(Finite State Machine)로 설계하기 위한 상태 천이도는 [그림 5]와 같다.

제어장치는 3개의 단계로 구성된다. 먼저 모듈러 지수 연산 프로세서의 환경 설정을 위한 인터페이스 프로토콜을 구현한다. 연산 프로세서에 입력되는 값들의 자리수를 지정하고, 초기화 과정으로 연산 프로세서의 내부 레지스터들을 초기화한다. 그리고 지수(e), 모듈러스(N), 메시지(M), 제어신호(Ctl)값 들을 지수 연산기 내부 레지스터에 저장한다. 마지막으로 지수 연산 장치에서 일어나는 계산과정으로 정규

수 체계를 Montgomery 수 체계로 변환하여 지수 연산을 수행하고 계산 결과를 다시 정규 수 체계 표현으로 변환한다. 지수의 각 비트를 검사하여 제곱과 곱셈 연산이 올바르게 수행되도록 한다.

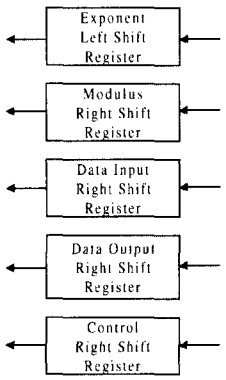
3.2.2 I/O 레지스터

입출력(Input/Output) 레지스터는 지수 키 e와 데이터 입출력과 저장을 위해 데이터를 결합하고 계산 중간 결과를 저장하고 계산에 필요한 값들을 지수 연산 프로세서에 전달한다. I/O 레지스터는 시프트(shift) 레지스터로 구성된다. 암·복호화에 사용되는 키 모듈러스(N), 지수(e), 제어 신호(Ctl)와 입력 데이터(M)를 입력하여 각 레지스터에 저장하고 곱셈과 제곱의 중간 계산 결과를 저장한다.

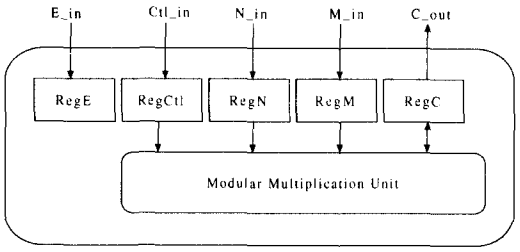
RSA 지수 연산을 수행하는데 필요한 피연산자(operand)를 저장하기 위해 5개의 시프트 레지스터를 사용하였다. 모듈러스 시프트 레지스터(RegN)에 공개키 N이 저장되고, Exponent 시프트 레지스터(RegE)에 지수 e가 저장된다. 입력 데이터 M와 출력 데이터 C는 각각 RegM과 RegC 레지스터에 저장되고, 모듈러 곱셈 시스톨릭 어레이의 동작을 제어하기 위해 제어 신호(Ctl)가 Control 시프트 레지스터(RegCtl)에 저장된다. 지수 연산 프로세서에 사용되는 시프트 레지스터의 블록도는 [그림 6]과 같다. 모듈러스 시프트 레지스터, 입력 시프트 레지스터, 출력 시프트 레지스터, 제어 신호(Control) 시프트 레지스터는 Right cyclic 시프트 레지스터인 반면 Exponent 시프트 레지스터는 Left cyclic 시프트 레지스터이다.

3.2.3 모듈러 지수 연산 장치

모듈러 지수 연산 장치는 LR 이진 제곱 알고리즘을 사용하여 설계하였다. 모듈러 지수 연산장치의



[그림 6] 입출력 시프트 레지스터



[그림 7] 지수 연산장치 블록도

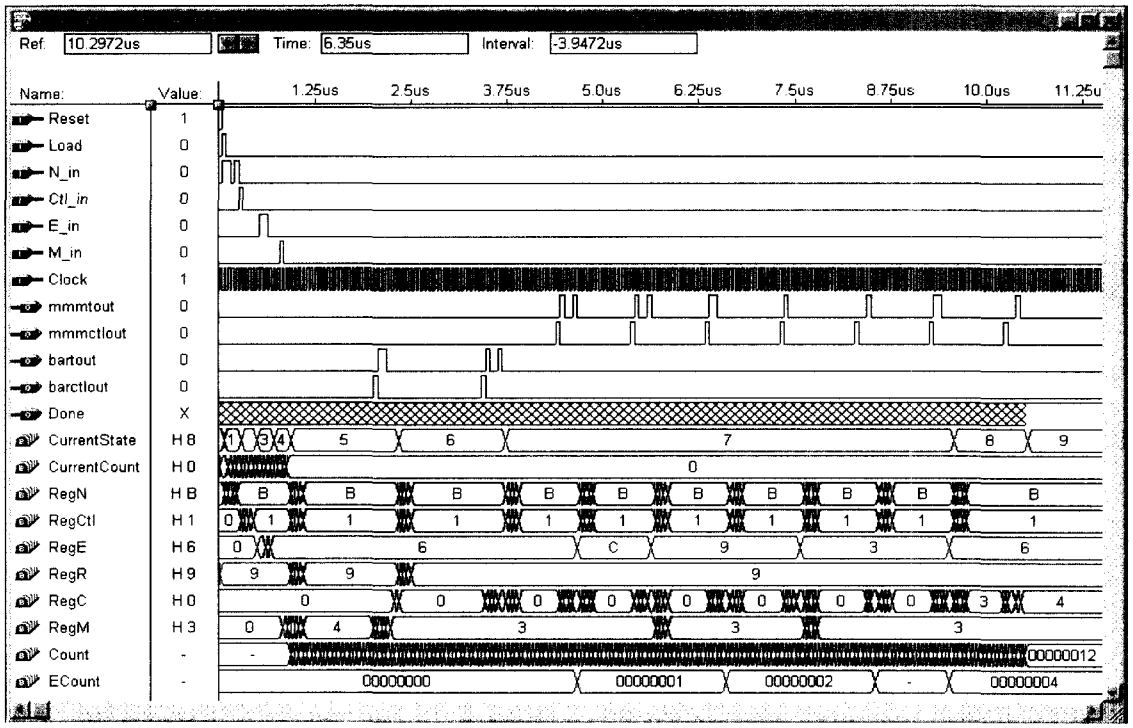
블록구조는 [그림 7]과 같다.

지수 연산 장치는 지수를 저장하고 있는 레지스터 RegE의 최상위 비트(MSB)부터 최하위 비트(LSB) 순으로 검사한다. 지수 값 e를 저장하는 레지스터 RegE의 i번째 값 e_i에 대해 제어신호를 저장하고 있는 RegCtl 레지스터, 지수 연산의 중간 계산 결과 값을 저장하고 있는 RegC 레지스터, 모듈러스를 저장하고 있는 RegN 레지스터는 저장하고 있는 값을 1비트씩 모듈러 곱셈 장치(Modular Multiplicatiom Unit)으로 보내어 제곱 연산, $CC \bmod N$, 을 계산하여 결과값을 RegC 레지스터에 저장한다. 만약 지수 e_i의 값이 "1"이면 $MC \bmod N$을 계산하기 위해 RegCtl 레지스터, RegN 레지스터, 입력 데이터를 저장하고 있는 RegM 레지스터와 RegC 레지스터의 값들을 곱셈 장치에 전달하여 곱셈 연산이 수행되도록 한다. 즉, e_i 값에 대해 제곱 연산을 수행하고 e_i 값에 따라 곱셈 연산도 하기도 한다. 레지스터 RegM 값은 e_i값의 비트값에 따라 e_i값이 "1"이면 레지스터 값을 모듈러 곱셈 장치로 출력한다. e_i 비트에 대한 연산이 끝난 후 레지스터 RegE의 값을 1비트 왼쪽 시프트 하여 e_{i-1} 번째 비트를 검사하여 위의 연산 과정을 반복한다.

3.3 VHDL 실험

모듈러 지수 연산기의 하드웨어 성능을 알아보기 위해 각 모듈에 대해서 하드웨어 기술 언어인 VHDL 코드를 작성하고 작성된 코드의 동작을 검증하기 위해 ALTERA사의 MAX+PLUS II 도구를 이용하여 시뮬레이션한 결과는 [그림 8]과 같다. Clock는 클락 주기, Ctl_in, N_in, E_in, M_in은 지수 연산기로 입력되는 입력 값을 나타내고, C_out은 계산 결과를 나타낸다.

M_in="1111", N_in="1011", E_in="0101"일 때 지수 연산의 결과 C_out는 "1001"임을 알 수 있다.



〔그림 8〕 실험 결과

지수 연산기의 Clock 주기는 50ns임을 알 수 있고, 결과 값의 출력이 시작되는 지연시간은 10.2972μs임을 알 수 있다. 입력 데이터 N_in, Ctl_in, E_in, M_in의 입력시간은 949.5ns이고 한번의 모듈러 곱셈 수행시간이 972ns로 9번의 곱셈을 수행하므로 한번의 모듈러 지수 연산에 소요되는 지연 시간은 10.2μs임을 알 수 있다.

입력 값의 자리수가 4, 8, 16, 32, 64비트, 지수 E_in값 중에서 1의 개수가 $n/2$일 때 MAX+PLUS II와 FLEX10K 장치를 사용하여 실험한 지수 연산 프로세서의 지연 시간과 사용 게이트 수는 〔표 2〕와 같다. 만약 메시지 블록의 길이가 n=512이고 지수 *E_in*값 중에서 1의 개수가 256일 때 설계된 암호시스템의 지연 시간은 59.5ms로 예상된다. 수행시간이 500ms보다 작은 암호시스템은 스마트 카드에 이용될 수 있는데, 본 논문에서 설계된 지수 연산 프로세서는 스마트 카드에 이용될 수 있다.

〔표 2〕 자리수별 지연시간과 게이트 수(r=2)

자리수 (n)	지연시간 (μs)	사용 게이트 수(개)	사용 장치
4	10.2	13,400	EPF10K20TC144-3
8	24.1	20,700	EPF10K30TC208-3
16	76.1	35,200	EPF10K40RC208-3
32	266.5	68,600	EPF10K70RC240-2
64	992.9	130,200	EPF10K100GC503-3

V. 결 론

본 논문에서는 RSA 암호시스템에 사용되는 모듈러 곱셈 시스톨릭 어레이와 지수 연산 프로세서를 설계하였다. 설계된 지수 연산 프로세서의 구조는 계산 순서를 제어하는 제어장치, 시프트 레지스터로 구현되는 입출력(I/O) 레지스터, 지수연산을 수행하는 지수 연산 장치의 3개 영역으로 구성된다.

설계된 지수 연산 프로세서의 VHDL 코드를 사용하여 모델링하고 회로의 동작을 검증하기 위해 MAX+PLUS II를 이용하여 시뮬레이션하고 성능을 확인하였다. 입력 값의 자리수가 4비트 일 때 VHDL 언어로 시뮬레이션하고 ALTERA사의 EPM9400CLC84-15 장치(device)를 사용하여 합성을 한 결과 지수 연산 프로세서의 지연시간은 10.2972us이다. 만약 메시지 블록의 길이가 n=512이고 지수 *E_in*값 중에서 "1"의 개수가 256일 때 지수 연산 프로세서의 지연 시간

은 59.5ms이다.

설계된 RSA 모듈러 곱셈 시스톨릭 어레이와 지수 연산기는 시뮬레이션 결과를 통해 암·복호화가 정확히 수행됨을 확인하였고, 하드웨어 구현이 가능하여 하드웨어를 이용한 빠른 모듈러 연산을 수행할 수 있다. 향후 연구과제로는 새로운 곱셈 알고리즘 및 지수 알고리즘의 개발과 다른 지수 알고리즘을 사용하여 지수 연산 프로세서를 설계하여 비교 분석하여 가장 효율적인 지수 연산 프로세서를 설계한다.

참 고 문 헌

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," *Commu. ACM*, Vol. 21, pp. 120 ~ 126, 1978.

[2] J. Sauerbrey, "A Modular Exponentiation unit based on Systolic Arrays," *Abstract AUSCRYPT '92*, pp. 1219~1224, 1992.

[3] H. Orup, "Exponentiation, Modular Multiplication and VLSI Implementation of High-Speed RSA Cryptography," *Ph.D. thesis, University of Aarhus*, 1995.

[4] Paul Barrett, "Implementing the Rivest Shamir and Adleman Public Key encryption algorithm on a standard digital signature Processor," *Advances in Cryptology-Crypto '86*, LNCS 263, pp. 311~323, 1986.

[5] Shand, M. and Vuillemin, J. "Fast implementation of RSA cryptography," *Procedings of the 11th IEEE Symposium on Computer Arithmetic*, IEEE Computer Society Press, Los Alamitos, CA, pp. 252~259, 1993

[6] C. N. Zhang, "An improved binary algorithm for RSA," *Computer Mathematics Applications*, Vol. 25, No. 6, pp. 15~25, 1993.

[7] D. E. Knuth, "*The Art of Programming*," Vol. 2: Seminumerical Algorithms, 2nd Ed., Addison-Wesley, 1981.

[8] P.L. Montgomery, "Modular Multiplication Without Trial Division," *Mathematics of Computation*, Vol. 44, pp. 519~521, 1985.

[9] A. Selby and C. Mitchell, "Algorithms for software implementations of RSA," *IEE Proceedings*, Vol. 136 Pt. E, No. 3, pp. 166~170, 1989.

[10] Antoon Bosselaers, Rene Govaerts, and Joos Vandewalle, "Comparison of three modular reduction functions," *Advances in Cryptology-CRYPTO '93*, pp. 175~186, August, 1993.

[11] 황효선, 임채훈, 이필중, "PC에서 모듈러 곱셈 연산의 구현과 비교 분석," 통신정보보호학회지, Vol 4, No. 3, pp. 34~59, 1994.

[12] N. Takagi and S.Yajima, "Modular Multiplication Hardware Algorithms with Redunt Representation and Their Application to RSA Cryptosystem," *IEEE Transactions on computers*, Vol. 41, No. 7, pp. 887~891, 1992.

[13] A.J. Menezes, P.V. Oorschot, S. Vanstone, "*Handbook of Applied Cryptography*," CRC Press, 1997.

[14] K.Y. Lam and L.C.K. Hui, "Efficient of SS(l) square and multiply exponentiation algorithms," *Electronics letters*, Vol. 30, No. 25, pp. 2115~2116, 1994.

[15] IEEE, "*IEEE MICRO*," IEEE Computer Society, Vol. 14, No. 3, 1996.

[16] L. Harn, "Public-key cryptosystem design based on factoring and discrete logorithms," *IEE Proceedings Computer Digit Technology*, Vol. 141, No. 3, pp. 193~195, 1994.

[17] C.K. Koc, "RSA Hardware Implementation," *Technical Report*, RSA Laboratories, 1995.

[18] Gieseke, B., "A 600MHz superscalar RISC microprocessor with out-of-order execution," *ISSCC Digest of Technical Papers*, 1997.

[19] Ivey, P. A., Walker, S. N., J. M. and Davidson, S., "An ultra-high speed public key encryption processor," *Proceedings of the IEEE 1992 Custom Integrated Circuits Conference*, pp. 19.6.1~19.6.4, 1992.

[20] E. F. Brickell, A survey of hardware implementations of RSA, *Advances in Cryp-*

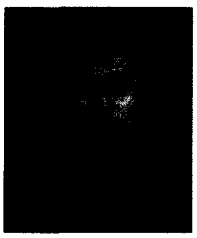
tology - CRYPTO '89, pp. 368~370, 1989.
[21] ALTERA, "*MAX+PLUS II Getting Started*," ALTERA Corp. 11, 1995.
[22] K. Iwamura, T. Matsumoto, and H. Imai, "Systolic Arrays for Modular Exponentiation using Montgomery Method," *Proc. Euro CRYPT '92*, pp. 477~481, 1992.
[23] C. D. Walter, "Systolic Modular Multiplication," *IEEE Trans. on Computers*, Vol. 42, pp. 376~378, 1993.

〈著者紹介〉

허 영 준 (Young-Jun Heo) 정회원
1993년2월 : 대구대학교전자계산학과 졸업
1996년 2월 : 경북대학교 컴퓨터공학과 석사
2000년 2월 : 경북대학교 컴퓨터공학과 박사
1998년 9월~2000년 8월 : 대구미래대학 컴퓨터게임제작과 전임강사
2000년 9월 - 현재 : 한국전자통신연구원 선임연구원
〈관심분야〉 처리기 설계, 암호, 네트워크 보안

박 혜 경 (Hae-Kyoeng Park) 비회원
1990년 2월 : 창원대학교 전자계산학과 졸업
1992년 2월 : 경북대학교 컴퓨터공학과 석사
1997년 3월 : 경북대학교 컴퓨터공학과 박사
1997년 3월~현재 : 한국 전자통신연구원 선임연구원
〈관심분야〉 병렬알고리즘, Routing Protocol, MPLS,

이 건 직 (Keon-Jik Lee) 비회원
1996년 2월 : 대구대학교 전자계산학과 졸업
1998년 2월 : 경북대학교 컴퓨터공학과 석사
1998년 3월~현재 : 경북대학교 컴퓨터공학과 박사과정
〈관심분야〉 배열 처리기 설계, 보안/암호



이 원 호 (Won-Ho Lee) 비회원
1998년 2월 : 대구대학교 전자계산학과 졸업
2000년 2월 : 경북대학교 컴퓨터공학과 석사
2000년 3월~현재 : 경북대학교 컴퓨터공학과 박사과정
〈관심분야〉 배열처리기 설계, 보안/암호

유 기 영 (Kee-Young Yoo) 종신회원
1976년 2월 : 경북대학교 수학교육학과 졸업
1978년 2월 : 한국과학기술원 전산학과 석사
1992년 3월 : R.P.I 박사
1978년 3월~현재 : 경북대학교 컴퓨터공학과 교수
〈관심분야〉 정보보호/보안, 병렬처리, 병렬 컴파일러, DSP array processor