RSA 지수 연산기 설계

허영준°, 박혜경, 유기영
대구미래대학 컴퓨터게임제작과
한국전자통신연구원 교환전송기술연구소
경북대학교 컴퓨터공학과
yjheo@tfc.ac.kr, phk@etri.re.kr, yook@bh.kyungpook.ac.kr

Design of RSA Exponentiation Processor

Young-Jun Heo°, Hae-Kyeong Park, and Kee-Young Yoo
Department of Computer Game Engineering, Taegu Future College
Electronics and Telecommunications Research Institute
Department of Computer Engineering, Kyungpook National University

요 약

본 논문에서는 몽고메리 알고리즘과 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.

1. 서론

RSA 암호시스템[1]에서 송신자는 모든 사용자들에게 공개되는 수신자의 공개키 e를 이용하여 암호화된 메시지 $C=M^e$ $(mod\ N)$을 보낸다. 수신자는 자신의 비밀키 d를 이용해서 원래 메시지 $M=C^d$ $(mod\ N)$을 복원한다. 암호시스템의 안전성을 확보하기 위해 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 $(mod\ N)$ 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[4,5,6], 모듈러 곱셈 수행 회수를 줄이는 방법에 대한 연구가 이루어졌다[2,3,4,7,9].

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. Ivey는 512비트 모듈러스를 사용하는 RSA 암호시스템을 구현하였고[11], Orup은 모듈러스 길이가 561비트이고 25인 곱셈기를 사용하는 암호 시스템을 발표하였다[12].

본 논문에서는 RSA 암호시스템에 사용할 수 있는 단일 칩 모듈러 지수 연산 프로세서를 설계하고, 지수 연산기의 성능을 알아보기 위해 VHDL을 이용하여 모델링하고 ALTERA사의 MAX+PLUS II[10]를 사용하여 시뮬레이션한다.

2. 지수 연산기 설계

2.1. 지수 연산 알고리즘

지수연산 알고리즘에는 $C=M$ $(mod\ N)$으로부터 시작하여 $C=CM$ $(mod\ N)$의 모듈러 곱셈 연산을 e-1번의 곱셈을 하는 원시 알고리즘과 지수 e를 한비트씩 스캔하는 "이진 제곱 곱셈" 알고리즘[4]과 스캔되는 비트 수가 log_2m이면 m-ary 방식이 있다. 암호화 키 e의 이진 표현에 계수 "-1"을 추가하여 결과적으로 "0"이 아닌 비트 수를 줄여 곱셈 연산 회수를 줄이는 수정된 부호화 디지트(modified signed digit)알고리즘, 암호화 키 e를 표현함에 있어서 열 치환 표현 기법을 사용하는 K-SR 알고리즘과 e의 이진표현에서 고정된 1개의 크기 내에서 홀수로 표현 가능한 비트 패턴을 찾아서 대치하는 $SS(l)$ 알고리즘이 있다[8].

이 중에서 이진 제곱 곱셈 알고리즘은 각 반복문이 단지 모듈러 곱셈과 간단한 이진 결정만을 취하기 때문에 특히 하드웨어 상에서 수행하기에 효율적이다. 최상위 비트에서 최하위 비트 순서로 처리하는 Left-to-Right(LR)과 그 역순으로 처리하는 Right-to-Left(RL) 알고리즘이 있다.

본 논문에서는 기억장소와 수행 시간을 고려하여 하드웨어 구현이 가장 용이한 LR 이진 제곱 곱셈 알고리즘과 Montgomery 알고리즘을 사용하여 지수 연산기를 설계한다. Montgomery LR 이진 제곱 곱셈 알고리즘은 다음과 같다.

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

여기서 R은 r^n을 의미한다. LR 이진제곱 곱셈 알고리즘에 Montgomery 모듈러 곱셈 알고리즘을 사용하기 위해서는 정규 모듈러 수 체계를 Montgomery 모듈러 수 체계로 변환하여야 한다. 위 알고리즘에서 $\overline{M}$, $\overline{C}$ 는 M, C의 Montgomery 수 표현을 의미한다. 1, 2단계에서 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, 데이터($\overline{C}$)의 제곱을 계산하기 위해 Montgomery

모듈러 곱셈 $MMM(\overline{C}, \overline{C})$을 수행하고 지수 e_i의 값이 "1"이면 데이터($\overline{M}$)와 데이터($\overline{C}$)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.

2.2 지수 연산기

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 1과 같다.

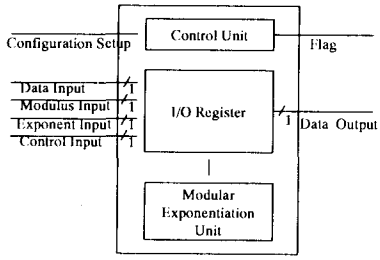

그림 1. 모듈러 지수 연산기

2.2.1 제어장치(Control Unit)

제어장치는 사용자에 의해 선택된 구성에 대해 프로세서의 기능성을 구성하는 역할을 한다. 프로세서와의 통신은 사용자가 정의한 인터페이스 신호에 의해 제어되고 상태 플래그(state flag)는 프로세서에 의해 생성된다. 제어장치는 각 장치의 동작을 제어하기 위해 제어 신호를 생성한다. 제어 장치를 유한 상태도(Finite State Machine)로 설계하기 위한 상태 천이도는 그림 2와 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)값들을 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.

2.2.2 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.

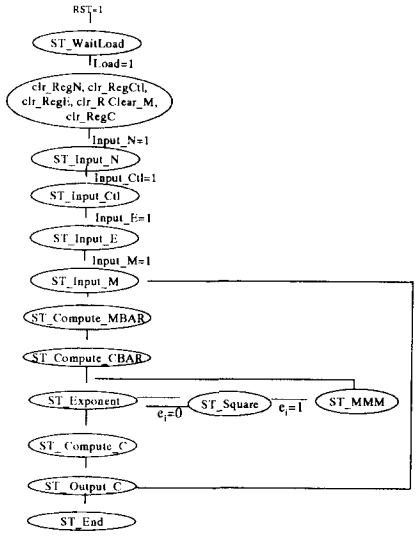

그림 2. 제어장치의 상태 천이도

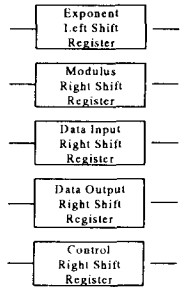

그림 3. 입출력 시프트 레지스터

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 3과 같다. 모듈러스 시프트 레지스터, 입력 시프트 레지스터, 출력 시프트 레지스터, 제어신호 시프트 레지스터는 Right cyclic 시프트 레지스터인 반면 Exponent 시프트 레지스터는 Left cyclic 시프트 레지스터이다.

2.2.3 모듈러 지수 연산 장치

모듈러 지수 연산장치의 블록구조는 그림 4와 같다. 지수 연산 장치는 지수를 저장하고 있는 레지스터 RegE의 최상의 비트부터 최하위 비트 순으로 검사한다. 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)로 보내어 제곱 연산, *CC mod N*, 을 수행하고 결과값을 RegC 레지스터에 저장한다. 만약 지수 e_i의 값이 "1"이면 *MC mod N*을 계산하기 위해 RegCtl 레지스터, RegN 레지스터, 입력 데이터를 저장하고 있는 RegM 레지스터와 RegC 레지스터의 값들을 곱셈 장치에 전달하여 곱셈 연산이 수행되도록 한다.

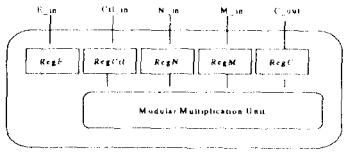

그림 4. 지수 연산장치 블록도

2.3 VHDL 실험

모듈러 지수 연산기의 성능을 알아보기 위해 각 모듈에 대해 VHDL 코드를 작성하고 작성된 코드의 동작을 검증하기 위해 MAX+PLUS II 도구를 이용하여 시뮬레이션한 결과는 그림 5와 같다. Clock는 클락 주기, Ctl_in, N_in, E_in, M_in은 지수 연산기로 입력되는 입력 값을 나타내고, C_out은 계산 결과를 나타낸다.

입력 값의 자리수가 4, 8, 16, 32, 64비트, 지수 E_in값 중에서 1의 개수가 $n/2$일 때 ALTERA사의 FLEX10K 장치를 사용하여 실험한 지수 연산 프로세서의 지연 시간과 사용 게이트 수는 표 1과 같다. 만약 메시지 블록의 길이가 n=512이고 지수 *E_in*값 중에서 1의 개수가 256일 때 설계된 암호시스템의 지연 시간은 59.5㎳이다.

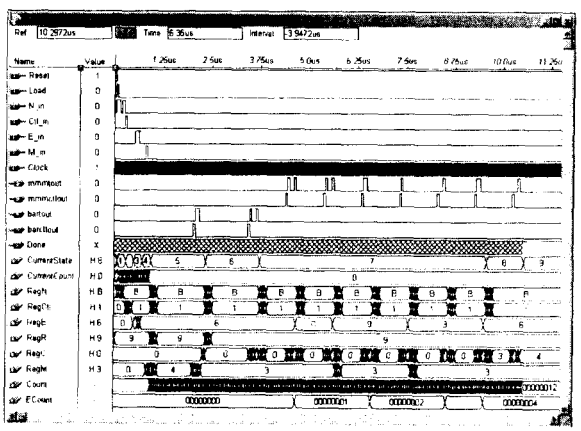

그림 5. 실험 결과

표 1. 자리수별 지연시간과 게이트 수(r=2)

자리수 (n)	지연시간 (μs)	사용 게이트수 (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

3. 결론

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 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. 설계된 RSA 모듈러 지수 연산기는 시뮬레이션 결과를 통해 암·복호화가 정확히 수행됨을 확인하였고, 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.

4. 참고문헌

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

[5] P.L. Montgomery, "Modular Multiplication Without Trial Division," *Mathematics of Computation*. vol. 44, pp. 519-521, 1985.

[6] A. Selby and C. Mitchell, "Algorithms for software implementations of RSA," *IEE Proceedings*, vol. 136 Pt. E, no. 3, pp.166-170, 1989.

[7] 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.

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

[9] 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.

[10] ALTERA, *"MAX+PLUS II Getting Started,"* ALTERA Corp. 11, 1995.

[11] 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.

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