가변길이 고속 RSA 암호시스템의 VLSI 구현

박 진 영, 서 영 호, 김 동 욱
광운대학교 전자재료공학과
전화: 02-940-5167/E-mal: codesign@explore.gwu.ac.kr

VLSI Implementation of High Speed Variable-Length RSA Cryptosystem

Jin Young Park, Young Ho Seo, Dong Wook Kim
Dept. of Electronic Material , Kwangwoon University
Tel: 02-940-5167/E-mail : codesign@explore.gwu.ac.kr

Abstract

In this paper, a new structure of 1024-bit high-speed RSA cryptosystem has been proposed and implemented in hardware to increase the operation speed and enhance the variable-length operation in the plain text. The proposed algorithm applied a radix-4 Booth algorithm and CSA(Carry Save Adder) to the Montgomery algorithm for modular multiplication.

As the results from implementation, the clock period was approached to one delay of a full adder and the operation speed was 150MHz. The total amount of hardware was about 195k gates. The cryptosystem operates as the effective length of the inputted modulus number, which makes variable-length encryption rather than the fixed-length one. Therefore, a high-speed variable-length RSA cryptosystem could be implemented.

I. 서론

네트워크와 인터넷의 발달로 다양한 회선망을 통한 정보의 교환이 급속도로 활성화되어 송/수신 데이터의 보안 문제가 중요한 사안이 되고 있다. 이러한 정보 보호 기능의 문제를 해결하기 위해 암호 알고리즘의 개발 및 그것의 효율적인 구현에 대한 연구가 활발히 이루어지고 있다.

정보의 기밀성, 키 교환 및 인증기능을 모두 제공하는 공개키 암호방식의 개념이 1976년 Diffie 등에 의해 제안되었고 그 이후 Rivest 등에 의해 1977년 RSA 공개키 암호 알고리즘[1]이 개발되었다. 이 RSA 알고리즘은 간단한 수학적 배경, 체계적인 시스템, 그리고 높은 안전도로 전 세계적으로 상용화되고 있으나 암호화 및 복호화가 수학적 연산을 기초로 수행되기 때문에 순열과 치환에 의한 대칭키 암호 알고리즘과 비교하여 연산 속도가 느린 단점을 가지고 있다. 따라서 수행속도를 향상하기 위한 RSA 암호시스템의 하드웨어 구현에 관한 연구가 많이 진행되어왔다.

본 논문에서는 RSA 암호시스템의 수행속도를 향상시키기 위한 알고리즘들을 제안한다. 이때 모듈러 곱셈 알고리즘은 Montgomery 알고리즘[2]에 CSA(Carry Save Adder)[3] 구조와 기수-4 Booth 알고리즘[3]을 적용하고 모듈러 지수승 알고리즘은 RL-이진 방법[3]을 개선한다. 암호화 수행에 있어서 이러한 구조의 특성은 고정길이가 아닌 가변길이 연산을 가능하게 하여 수행속도를 향상할 수 있다. 제안한 RSA 암호시스템은 하드웨어로 구현하여 기존의 방법과 동작속도 및 하드웨어 재원 사용 등에 대해 비교한다.

II. 제안한 RSA 암호시스템

본 논문에서 제시하는 RSA 암호시스템의 구조는 그림 1과 같이 여섯 개의 기능블록으로 구성된다. 1024-비트 메시지 입력을 위한 32-비트 입력 블록(Input Module), 모듈러 지수승 연산을 위한 모듈러 지수승 연산기 블록(Modular Exponentiator), 모듈러 곱셈을 위한 모듈러 곱셈기 블록(Modular Multiplier), CSA 연산에 의해 발생한 합(sum)과 캐리(carry) 부분을 최종 출력하기 위한 32-비트 덧셈 연산 블록(32-bit CPA, Carry Propagation Adder), 암호문의 출력 블록(Output Module) 그리고 나머지 블록들을 제어하기 위한 제어 블록(Control Module)으로 구성된다.

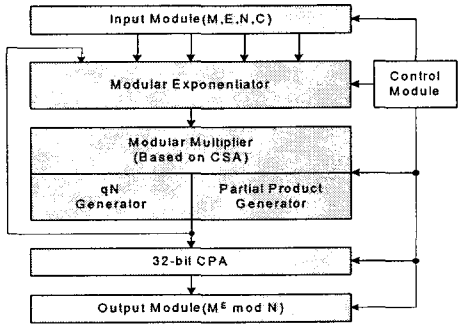

그림 1. 제안한 RSA 암호시스템의 구조

Ⅱ-1. 고속 모듈러 곱셈 알고리즘

제안한 모듈러 곱셈 알고리즘은 Montgomery 알고리즘[1]과 기수-4 Montgomery 알고리즘[4]을 개선하여 적용한다. 이 수정된 기수-4 모듈러 곱셈(Modified Radix-4 Modular Multiplication, MRMM) 알고리즘은 CSA(Carry Save Adder)[3] 구조와 기수-4 Booth 알고리즘[3]을 사용하며 내용은 아래의 의사코드와 같다. 알고리즘에서 Ys,c는 모듈러 곱셈의 피승수, Xs,c는 승수, 그리고 N은 모듈러스 수를 각각 나타낸다.

```
MRMM(Ys,c, Xs,c, N) {
-- Output Zs,c = Ys,c · Xs,c · R^-1 ( mod N )
-- bi : Booth recode value,  R=4^m
   Zs,c = 0 ;     m = ⌈ (n−l+3)/2 ⌉ ;
   for(i=0 ; i<m ; i++) {
      bi = booth(Xs, Xc, i) ;
      qi = [ {Zs,c + bi(Ys+Yc)} × -N1,0 ] mod 4 ;
      Zs,c = Zs + Zc + biYs ;
      Zs,c = Zs + Zc + biYc ;
      Zs,c = (Zs + Zc + qiN) / 4 ;
   }
   return Zs,c ;
}
```

여기서 N=(n_{k-1}, ..., n_1, n_0)은 k-비트 홀수이다. 아래첨자 "s,c"는 CSA 연산에서 발생되는 합과 캐리의 두 값 모두를 갖는 변수이고 "s"는 합 그리고 "c"는 캐리 값만을 갖는 변수를 의미한다. 모듈러 곱셈 함수의 입력이 -N≤(Ys+Yc), (Xs+Xc)<N의 범위에 있다고 가정하면 m번의 반복 연산 후에 -N≤(Zs+Zc)<N 범위의 최종 결과를 출력한다. m은 Montgomery 알고리즘에 기수-4 Booth 알고리즘이 적용되었을 때 모듈러 곱셈 연산의 반복 횟수이며, 이로써 부분 곱의 수가 약 n/2으로 감소한다. 여기서 l은 주어진 N의 최상위 비트로부터 최하위 비트까지 '1'의 값을 검색했을 때 최초로 '1'이 되는 비트의 위치를 의미한다. 즉, $(n-l)$은 주어진 n-비트 수의 유효비트 수를 나타낸다. 따라서 n-비트 RSA 알고리즘은 모듈러 곱셈의 반복 횟수를 약 $\lceil (n-l+3)/2 \rceil$로 감소시켜 전체 암호시스템 연산 수행속도를 줄이고 가변길이 연산을 할 수 있다. bi는 승수 Xs,c의 Booth 코드(code)로 부분 곱을 계산한다. 그리고 q_i는 m번의 반복 연산 동안 모듈러스 수를 결정한다. 제안한 알고리즘의 입/출력은 합과 캐리를 사용함으로써 n-비트 CPA의 사용을 피해 가산기 지연시간을 제거한다. 위의 알고리즘 기술은 CSA 연산이 가능하도록 세 단계로 분리하였다.

Ⅱ-2. 고속 모듈러 지수승 알고리즘

본 논문의 모듈러 지수승 알고리즘은 모듈러 제곱연산과 모듈러 단순 곱셈을 병렬적으로 처리할 수 있는 RL-이진방식[3]을 기초로 하고, 이전 절에서 제시한 MRMM을 반복 수행함으로써 모듈러 지수 연산이 수행된다. 그 내용은 아래와 같고 입력은 평문(M), 키(E) 그리고 모듈러스 수(N)이고 최종 출력은 암호문(P)이다.

```
Modified Modular Exponentiation(M, E , C, N) {
-- Constant : C = R^2 (mod N)
-- Output : P = M^E mod N
    Ss = M ;     Sc = 0 ;
    Fs = 1 ;     Fc = 0 ;
    Ss,c = MRMM(Ss,c, C, N) ;
    for (i=0 ; i<k ; i++) {
      Ss,c = MRMM(Ss,c, Ss,c, N) ;
      if (ei=1)  Fs,c = MRMM(Fs,c, Ss,c, N) ;
      else       Fs,c = Fs,c ;
    }
   P = Fs + Fc ;
   if (P<0) P = P + N ;
}
```

이 알고리즘에서 상수 C(C_{k-1}, ... C_1, C_0)는 모듈러 곱셈의 결과로 R^{-1}이 누적되는 것을 막기 위한 초기값이다. 이 값은 가변길이 연산이 가능하도록 모듈러 곱셈의 승수로 사용한다. 모듈러 지수 연산의 반복 횟수는 키 값(E)의 유효 비트에 의존한다. 키 값의 선택 조건은 N보다 작기 때문에 E의 유효 비트는 $(n-l)$보다 작으므로 모듈러 지수승 연산 과정에서 모듈러 곱셈의 반복횟수는 $\lceil (n-l+3)/2 \rceil \times (n-l)$이다.

Ⅱ-3. 모듈러 곱셈기의 하드웨어 설계

제안한 모듈러 곱셈 알고리즘을 하드웨어로 구현하기 위해서는 3개의 n-비트 CSA구조가 필요하다. 그리고 부분 곱 생성 결과가 음수일 경우, 캐리 전파 없이 2의 보수를 만들기 위해서 하나의 n-비트 CSA구조를 추가한다. 따라서 총 4개의 CSA구조가 필요하다. 이러한 4 단계 CSA구조를 적용한 모듈러 곱셈기(Modular Multiplier)의 하드웨어 구조는 그림 2와 같다. 이 구조는 각 단계마다 레지스터를 삽입하여 파이프라인(pipe-line)화함으로써 4개의 모듈러 곱셈을 연속적으로 수행할 수 있다. 이때 요구되는 4개 모듈러 곱셈을 위해서 제안한 RSA 암호시스템은 2개의 메시지 블록을 동시에 입력받는 것으로 가정하였다. 모듈러 곱셈기의 4단계 CSA 구조는 3n개의 전가산기와 n개의 반가산기로 구성된다. 각 단계에서 최상위 두 비트는 부호 비트를 확장하기 위해 사용된다. 3단계에서는 반가산기를 사용하고 두 번째 비트는 2의 보수 연산을 위해 전가산기가 사용된다.

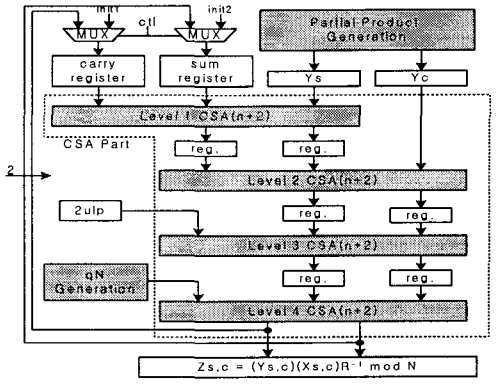

그림 2. 모듈러 곱셈기의 하드웨어 구조

그림 2의 모듈러스 수 생성 블록과 부분 곱 생성 블록은 CSA 부분과 분리하여 매 클록마다 독립적으로 결과를 발생하도록 하였다. 물론 하나의 모듈러 곱셈만 고려했을 경우 네 클록마다 결과를 출력하면 된다. 부분 곱 생성 모듈(Partial Product Generator)은 네 클럭마다 결과를 출력하기 위해 그림 3과 같이 구성된다. 피승수 Xs,c의 합과 캐리는 2-비트씩 더해서 Booth 코드를 생성하고 이 값은 부분 곱을 효율적으로 선택하기 위해 다시 recode 모듈과 Y-select 모듈에 입력되어 부분 곱을 생성한다. 모듈러스 수 생성 모듈(qN Generator)은 세 클럭마다 결과를 출력하기 위해 그림 4의 qi를 생성하기 위한 모듈과 그림 5의 모듈러스 선택 모듈로 분리되어 계산된다. 따라서 모듈러 곱셈기의 클록 주기는 전가산기의 지연시간에 근접한다.

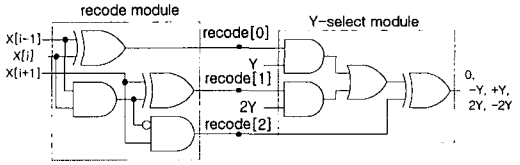

그림 3. 부분 곱 생성 블록

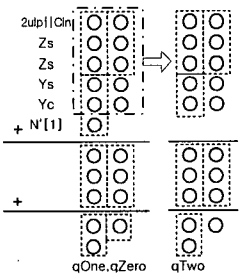

그림 4. qi를 위한 덧셈

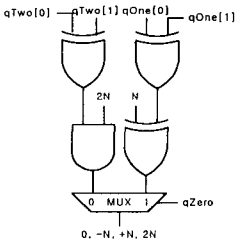

그림 5. 모듈러스 수 선택 모듈

Ⅱ-4. 모듈러 지수승 연산기의 하드웨어 설계

모듈러 지수승 연산기는 레지스터와 두 개의 멀티플렉서 그리고 하나의 제안한 모듈러 곱셈기를 이용해 쉽게 하드웨어로 구현될 수 있다. 결과적으로 두 개의 n-비트 평문의 암호화에서 N의 유효비트가 $(n-l)$이라면 $\lceil (n-l+3)/2 \rceil \cdot 4(n-l)$ 클록 후에 2개의 암호문을 출력한다. 따라서 하나의 암호문은 약 $(n-l)^2$클록 후에 생성되고, 클록 주기는 전가산기의 지연시간에 근접한다. 그러므로 기존의 방법들보다 모듈러 곱셈의 반복횟수가 감소되어 데이터 출력률이 향상된다. 또한 제안한 RSA 암호시스템은 1024-비트로 구현되었지만 더 작은 비트의 암호화/복호화도 가능하고, 수행속도 역시 모듈러스 수의 유효비트 수에 따라 유동적으로 수행될 수 있다.

Ⅲ. 구현 및 결과

본 논문에서 제안한 1024-비트 RSA 암호시스템은 하이닉스 0.35㎛ 2-poly 4-metal CMOS 공정을 위한 하이닉스 Phantom Cell library를 사용하여 구현하였다. Synopsys에서 합성한 결과의 하드웨어 양은 약 195k gate 수이었다. 시뮬레션은 NC-Verilog에서 수행되었고 동작속도는 150MHz이었다. 구현 과정에서 가변길이 연산을 위한 입력 핀을 두어 고정길이 연산이 아니라 입력에 따른 가변길이 RSA 암호시스템으로 동작하도록 하였다.

표 1은 기존에 제안되었던 RSA 암호시스템과 제안한 암호시스템을 비교하고 있다. 제안한 암호시스템은 CSA에 기초로 하였지만 CPA에 기초한 구조와 비교하여 클럭과 게이트의 수는 유사하다. 그러나 모듈러스 N의 유효 비트 $(n-l)$에서 l 이 증가한다면 모듈러 곱셈의 반복횟수는 현저히 감소하므로 데이터 출력률은 상당히 증가한다. 다른 암호시스템들은 512-비트에서의 출력률을 보이고 있는데, 제안한 암호시스템에서도 512-비트를 고려한다면 유효 비트 $(512-0)$인 최악의 경우에서 출력률은 약 300k bps(bit per second)로 기존의 가장 빠른 암호시스템과 유사하다. 그러나 유효 비트의 수가 400-비트이면 출력률은 480k bps로 상대적으로 증가하므로 기존의 암호시스템보다 수행속도가 향상되었다.

표 1. 다른 RSA 암호시스템과 비교

Architecture	CPA based		CSA based	
Author	Guo[5]	Hong[4]	kwon[6]	Our
Year	1999	2000	2001	2002
Gate count	132k	111k	112k	195k
Number of bit	512	512	1024	1024
Technology	0.6um	-	0.5um	0.35um
Clock(Hz)	143M	150M	50M	150M
# of modular multiplier	$2n^2$	n^2	$n(n+34)$	$(n-l)^2$
Baud rate(bit/s)	278k	300k	-	min (150k)

Ⅳ. 결론

본 논문은 공개키 암호 알고리즘으로 널리 사용되고 있는 RSA 암호시스템을 가변길이로 연산할 수 있는 RSA 암호시스템을 제안하고, 이를 하드웨어로 구현하였다. 연산속도를 향상하기 위해 모듈러 곱셈기에 기수-4 Booth 알고리즘을 적용하였고, 그 과정에서 요구되는 덧셈 연산 부분에 4단계 CSA 구조를 사용하여 캐리 전파를 제거함과 동시에 파이프라인화하여 데이터 출력률을 향상시켰으며, 클럭 주기를 전가산기의 지연시간에 근접하도록 하였다. 그리고 CSA 구조의 특성으로 인해 모듈러스 수에 따른 가변길이 연산을 수행할 수 있어 데이터 출력률을 향상하였고 가변길이 암호시스템으로 동작할 수 있었다.

제안한 1024-비트 RSA 암호시스템은 하이닉스 0.35㎛ Phantom Cell library를 사용하여 하드웨어로 구현하였고 게이트 수는 195k이었으며 이 때 동작 주파수는 150MHz이었다. 모듈러스 N의 유효 비트수가 감소할 경우에는 기존의 RSA 암호시스템보다 평균적으로 약 1.5배 정도 향상된 데이터 출력률을 보였다. 그리고 CSA를 사용하는 방법과 비교하여 최소 3배 이상의 수행속도 증가를 보였다. 따라서, 본 논문에서 제안한 RSA 암호시스템은 IP(Intellectual Property) 형태로 구현되어 고속 연산을 요구하는 각종 정보보호 시스템의 기능모듈 또는 시스템으로 사용될 수 있을 것으로 사료된다.

참고문헌

[1] R. L. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, vol. 21, pp.120-126, Feb, 1978.

[2] P. L. Montgomery, "Modular multiplication without trial division," Math. Computation, vol. 44, pp.519-521, 1985.

[3] Behrooz Parhami, "Computer arithmetic algorithms and hardware design," Oxford university press, 2000

[4] J. H. Hong, C. W. Wu, "Radix-4 modular multiplication and exponentiation algorithms for the RSA public-key cryptosystem," Design Automation Conference, 2000. Proceedings of the ASP-DAC 2000. Asia and South Pacific, pp.565 -570, 2000

[5] J. H. Guo, C. L. Wang, H. C. Hu, "Design and implementation of an RSApublic-key cryptosys tem," Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on , vol 1, pp.504 -507, 1999

[6] T. W. Kwon, C. S. You, W. S. Heo, Y. K. Kang, J. R. Choi, "Two implementation methods of a 1024-bit RSA cryptoprocessor based on modified Montgomery algorithm", *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, vol 4, 2001.