RSA 암호화 프로세서에 적용 가능한 효율적인 누적곱셈 연산기 설계

문 상 국*

Design of an Efficient MAC Unit for RSA Cryptoprocessors

Sangook Moon*

요 약

1024비트 이상의 고비도 RSA 프로세서에서는 몽고메리 알고리즘을 효율적으로 처리하기 위하여 전체 키 스트림을 정해진 블록 단위로 처리한다. 본 논문에서 기본으로 하는 RSA 프로세서는 기본 워드를 128비트로 하고 곱셈 결과의 누적기로는 256비트의 레지스터를 사용한다. 128 비트 곱셈을 효율적으로 수행하기 위하여 32비트 * 32비트 곱셈기를 사용하며 각 연산 결과는 128비트 크기의 8개 레지스터에 필요에 따라 저장되어 몽고메리 알고리즘을 수행하는데 사용된다. 본 논문에서는 128비트 곱셈에 필요한 누적곱셈 (MAC; multiply-and-aCcumultaion)을 효율적으로 계산하기 위하여 모든 연산 단계를 미리 분석하여 불필요한 연산단계를 수행하지 않고 곱셈 횟수를 줄여 효율적인 누적곱셈 연산기를 구현하였다. 구현된 누적곱셈 연산기는 자동으로 합성하였고, 본 논문 작성에서 기준이 되는 RSA 프로세서의 동작 주파수인 20MHz에서 정상적으로 동작하였다.

ABSTRACT

RSA crypto-processors equipped with more than 1024 bits of key space handle the entire key stream in units of blocks. The RSA processor which will be the target design in this paper defines the length of the basic word as 128 bits, and uses an 256-bits register as the accumulator. For efficient execution of 128-bit multiplication, 32b*32b multiplier was designed and adopted and the results are stored in 8 separate 128-bit registers according to the status flag. In this paper, an efficient method to execute 128-bit MAC (multiplication and accumulation) operation is proposed. The suggested method pre-analyze the all possible cases so that the MAC unit can remove unnecessary calculations to speed up the execution. The proposed architecture prototype of the MAC unit was automatically synthesized, and successfully operated at 20MHz, which will be the operation frequency in the target RSA processor.

키워드

MAC, RSA, multiplication, microprocessor

* 목원대학교 정보전자영상공학부

접수일자 : 2007. 8. 10

Ⅰ. 서 론

네트워크 기술의 발달로 현대사회는 유무선 네트워크와 같은 거대한 공통매체로 정보를 공유하게 되었다. 필요에 따라 이러한 정보들은 암호화되어 보호되어야 하기에, 개인의 정보를 인증해 줄 수 있는 IC카드와 같은 정보 보호 기술이 반드시 필요하게 되었다. 이러한 정보 보호기술은 암호학적인 알고리즘에 의해 복잡한 연산을 거쳐 이루어진다. 데이터를 암호화하는 기술은 암호에 사용되는 많은 수학연산을 처리하기 위해 높은 컴퓨팅 파워를 요구한다[1][2]. 이러한 이유로 인해 지난수년 동안 웹사이트에서의 신용카드 구매와 같은 경우를 제외하고는 대부분의 비즈니스에서 사용되지 않았으나, 최근에는 자신들이 가지고 있는 서버에 간단하게 add-on 보드 혹은 appliance 고속암호연산 프로세서 제품들을 장착하여 암호화의 수행을 빠르게 연산 가능하도록 할 수 있게 되었다[3][4]. 회사의 경영자들은 업무가 네트워크 의존적이 되어감에 따라 더 나은 보안을 요구할 것이며 동시에 많은 소비자들은 보안 허점이 많은 인터넷을 안전하게 만들도록 요구하고 있다. 이에 따라 IDC는 2005년에는 모든 인터넷 트래픽이 암호화되는 수준으로 발전할 것이라 예측하고 있다. 물론, 하드웨어의 미래는 아직도 불투명하지만, PC에 암호연산 프로세서가 기본적으로 탑재되는 날도 멀지 않을 것으로 예측하고 있다.

RSA란 암호화와 인증을 할 수 있는 공개키 암호 시스템이다. 이것은 1977년 Ron Rivest와 Adi Shamir, Leonard Adleman에 의해서 개발되었다. RSA는 대단히 큰 정수의 인수분해가 곤란함을 기반으로 보안성과 전자서명을 제공한다5][6][7].

RSA의 연산은 법 곱셈 (modular multiplication)과 법 거듭제곱 연산 (modular exponentiation)이 주를 이루게 되는데 이러한 법 연산을 고속으로 처리할 수 있는 방법 중의 하나인 몽고메리 알고리즘을 효율적으로 처리하기 위해서는 고성능의 누적곱셈 연산기 (MAC; multiply and accumulator) 가 요구된다[8][9] 문에서 연구한 누적곱셈 연산기는 몽고메리 알고리즘을 처리하는 데 있어 제한된 레지스터의 용량을 가늠하여 누적곱셈의 가능한 모든 연산 경우를 미리 분석하고, 불필요한 연산을 제거함으로써 처리의 효율성을 높이도록 하였다.

Ⅱ. RSA 암호 알고리즘

RSA란 암호화와 인증을 할 수 있는 공개키 암호 시스템이다. 이것은 1977년 Ron Rivest와 Adi Shamir, Leonard Adleman에 의해서 개발되었다. RSA는 대단히 큰 정수의 인수분해가 곤란함을 기반으로 보안성과 전자서명을 제공한다. 이것은 다음과 같은 동작 원리를 가진다.

2.1. RSA의 키 생성 알고리즘

- 각각의 주체 (서로 암호문을 주고받을 대상)는 RSA 공개키 (public key)와 그에 상응하는 비밀키 (private key)를 생성해야 한다.

- 각각의 주체 A는 다음과 같은 과정을 수행해야 한다.

. 개략적으로 비슷한 크기의 큰 임의의 소수 p, q를 생성한다.

. $n = pq$ 와 $\Phi = (p-1)(q-1)$를 계산한다.

. 임의의 정수 e 선택

($\gcd(e, \Phi) = 1$을 만족하는 $1 < e < \Phi$)

. 확장 유클리드 알고리즘을 사용하여 특별한 정수 d를 계산한다.

($ed \equiv 1 (\mathrm{mod}\, \Phi)$을 만족하는 $1 < d < \Phi$)

. 공개키 (n, e), 비밀키 (d)

2.2. 확장 유클리드 알고리즘 (extended Euclidean algorithm)

- INPUT : $a \geq b$ 인 두개의 음이 아닌 정수
- OUTPUT : $d = \gcd(a, b)$와 $ax + by = d$ 를 만족하는 정수 x, y

표 1. 확장 유클리드 알고리즘 계산 흐름
Table 1. Extended Euclid algorithm calculation flow

1		if $b=0$ then set $d \leftarrow a,\ x \leftarrow 1,\ y \leftarrow 0$, and return $(d,\ x,\ y)$.
2		Set $x_2 \leftarrow 1,\ x_1 \leftarrow 1,\ y_2 \leftarrow 0,\ y_1 \leftarrow 1$.
3		while $b > 0$ do the following:
	a	$q \leftarrow [a/b],\ r \leftarrow (a-qb),\ x \leftarrow (x_2 - qx_1),\ y \leftarrow (y_2 - qy_1)$.
	b	$a \leftarrow b,\ x \leftarrow x_2,\ x_1 \leftarrow x,\ y_2 \leftarrow y_1,\ y_1 \leftarrow y$.
4		Set $d \leftarrow a,\ x \leftarrow x_2,\ y \leftarrow y_2$, and return $(d,\ x,\ y)$.

- RSA 키 생성 알고리즘에서 생성된 정수 e와 d를 암호화 지수 (exponent) 또는 복호화 지수라 부르고, n을 모듈러스 (modulus)라 한다.

2.3. RSA 공개키 암호화와 복호화

B가 A에게 보낼 메시지 m을 암호화하고, A는 복호화 한다.

* 암호화 (encryption) : B는 다음 과정을 수행한다.

- A의 신뢰할 수 있는 공개키 (n, e)를 가져온다.
- $[0,\ n-1]$ 의 간격으로 메시지를 정수 m으로 나타낸다.
- $c = m^e \bmod n$ 을 계산한다.
- 암호문 c를 A에게 보낸다.

* 복호화 (decryption) : A는 다음 과정을 수행한다.

- 암호문 c에서 평문 m을 얻기 위해 비밀키 (private key) d를 사용한다.
- $m = c^d \bmod n$을 계산한다.

2.4. RSA 복호화 과정

$ed \equiv 1 \pmod{\phi}$ 이기 때문에, $ed = 1 + k\phi$ 를 만족하는 정수 k가 존재한다.

페르마의 정리 (Fermat's theorem)에 의해 만약 $\gcd(m, p) = 1$ 이면 $m^{p-1} \equiv 1 \pmod{p}$ 이다.

이식의 양변에 $k(q-1)$ 제곱을 하고, m을 곱하면 식은 다음 식 (1)과 같다.

$$m^{1+k(p-1)(q-1)} \equiv m \pmod{p} \qquad (1)$$

만약 $\gcd(m, p) = p$이면 $\bmod p$ 의 0 양변이 합동이기 때문에 위의 합동은 유효하다. 그러므로 모든 경우에 대해 $m^{ed} \equiv m \pmod{p}$이다.

같은 방식으로, $m^{ed} \equiv m \pmod{q}$일 때, p, q가 다른 소수이기 때문에 $m^{ed} \equiv m \pmod{n}$ 이고 $c^d \equiv (m^e)^d \equiv m \pmod{n}$ 이다.

III. RSA 암호화 프로세서의 구조

2.1 RSA 프로세서 상위 블록

RSA암호화 블록은 MAC 블록과 이를 제어해 주는 제어 블록, 대용량 레지스터를 포함한 버퍼 블록, 주소 선택기로 이루어져 있다. MAC 블록은 128비트 단위의 덧셈과 곱셈을 수행할 수 있는 승산기와 MAC 동작을 위한 256비트 덧셈기, 이들을 제어하는 제어블록으로 이루어진 MAC은 이 모듈의 핵심블록으로써, 128 블록 단위 몽고메리 알고리즘에 필요한 여러 가지 연산을 수행해 주게 된다. RSA 암호화 블록의 전체적인 모습은 그림 1과 같이 나타낼 수 있다.

3.2 32b*32b 곱셈기

곱셈기는 128 폭의 두 오퍼랜드의 곱이 가능하면 되는데, 성능 대비 면적을 최소화하기 위하여 32*32 곱셈기를 택하였고, 실제로 부호 비트를 고려하면 33*33 곱

셈기라고 볼 수 있다. 이 곱셈기에서는 부분합을 구하는 회로로서 radix-4 modified Booth's algorithm을 사용하여 부분합을 구하는 방식을 택하였다. 3:2 CSA를 사용하여 월레스 트리를 구성하였고, 월레스 트리에서는 sign-generation 방법을 사용하여 계산을 간소화 하는 방법을 택하였다. 32*32 곱셈기의 구조를 다음 그림 2에 보인다.

곱셈기의 입력으로는 곱수 (multiplier)와 피곱수 (multiplicand)가 있고, 곱수는 부스 인코더에 의해 -2X, -1X, 0X, +1X, +2X의 형태로 인코딩되어 부분곱 생성기의 제어 입력으로 사용되고, 각 부분곱들은 월레스 트리의 입력으로 사용되어 한 클럭 사이클에 모든 곱셈을 처리하여 캐리 벡터와 합 벡터를 생성하고, 최종적으로 덧셈기를 거쳐 곱셈 결과 값을 산출한다.

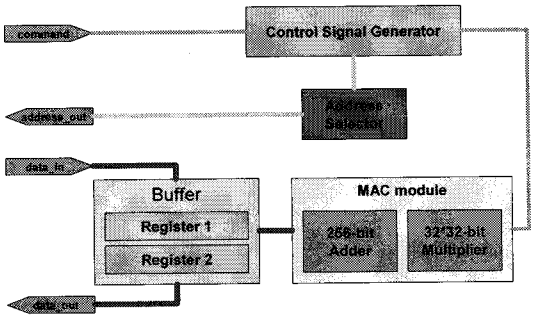

그림 1. RSA 암호처리기 상위 블록도
Fig. 1. Top module block diagram of an RSA cryptoprocessor

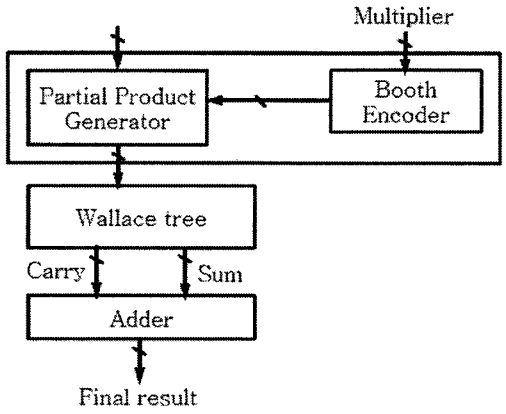

그림 2. 32b*32b 곱셈기의 구조
Fig. 2. 32b*32b multiplier block architecture

Ⅳ. MAC 유닛 설계

MAC 유닛은 128비트의 워드 단위를 기본으로 하여 128비트 단위의 곱셈, 덧셈 및 256비트 단위의 곱셈 누적 연산 (multiply and accumulate)을 지원하도록 설계되었다. 기본적으로 128비트 크기의 레지스터 8개와 1비트 크기의 캐리 3개를 내장하며, 이들을 이용하여 필요한 연산을 수행한 후 저장하도록 하고 있다. 앞서 나온 256비트 크기의 가산기와 32*32 규모의 곱셈기를 내장하고 있으며, 이를 연산에 따른 입력 값이나 레지스터 값을 이용하여 연산을 수행하도록 하고 있다. 그림 3에서는 32*32 규모의 곱셈기를 이용하여 128비트 단위의 곱셈을 수행하기 위한 16단계의 흐름을 보이며, 그림 4에서는 MAC의 블록 다이어그램을 보인다.

몽고메리 곱셈을 위한 MAC에서 지원하는 연산은 그림 4에 보이는 8개의 일반적인 128 레지스터와 상태 비트를 이용하여 몽고메리 알고리즘을 구현하기 위한 다양한 연산을 지원한다. 표 2는 몽고메리 연산용 MAC 유닛에서 지원하는 연산과 그에 해당하는 RSA 프로세서의 명령어를 보인다.

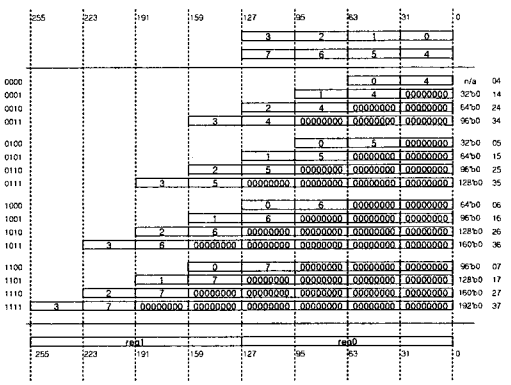

그림 3. 32b*32b 곱셈기를 이용한 128비트 단위의 곱셈 처리 방법
Fig. 3. Multiplication implementation in unit of 128 bits using 32b*32b booth's multiplier

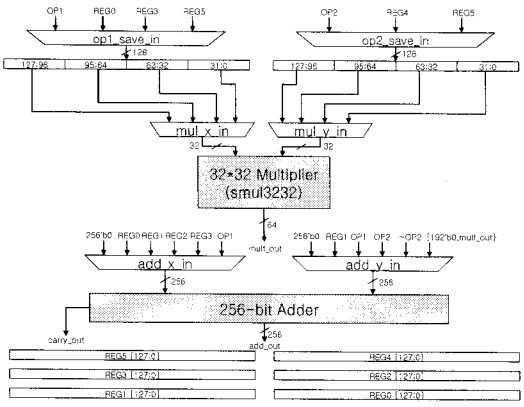

그림 4. 몽고메리 연산용 MAC 유닛의 블록 다이어그램
Fig. 4. A MAC unit block diagram for Montgomery multiplication

V. 구 현

본 연구에서는 128 비트를 한 블록으로 하는 블록 단위의 몽고메리 곱셈을 수행하는 RSA 프로세서에 적용 가능한 효율적인 누적곱셈연산기를 제안하였다. 성능을 좌우하는 가장 중요한 연산 유닛으로 면적 대비 효율적인 32*32 곱셈기를 선택하였다. 검증을 위한 레지스터 파일을 모델링 하였고, C 프로그램에서 연산을 수행한 블록 몽고메리 연산의 결과와 레지스터 파일에 저장된 결과를 비교하여 검증을 수행하였다. 검증하기 위하여 선택한 FPGA 디바이스는 Altera Stratix II를 선택하였고, 실제 FPGA 칩과 동일한 검증을 위해 Altera Quartus 프로그램으로 합성한 네트리스트와 Altera에서 제공하는 Stratix II와 실제 물리적 조건을 같게 모델링한 타이밍 모델을 사용하여 ModelSim 상에서 Verilog HDL로 게이트 레벨 시뮬레이션을 수행하여, 50MHz에서 정상 동작하는 것을 확인하였다. 그림 5에 제안한 누적곱셈연산기를 사용하여 구현한 블록 몽고메리 연산의 결과가 t 레지스터에 차례로 저장되는 시점에서의 게이트 레벨 시뮬레이션 파형을 보인다.

표 2. MAC에서 지원하는 연산
Table 2. Supported operations in the MAC unit

smart_cmd	operation
00_0001	(REG0)=(REG1) + carry
00_0100	(REG0)= (126'b0,carry2,carry1) + carry
00_0101	(carry1,REG0)=(RIN0) + (REG1)
00_1000	(REG1,REG0)=(RIN0) + (REG1)
00_1001	(carry,REG2)=(RIN0) + (REG1)
00_1010	(REG1,REG0)=(REG1,REG0) + (RIN0)
00_1100	(carry1,REG1,REG0)=(carry1,REG1,REG0) + (RIN0)
00_1101	(carry2,carry,REG2)=(REG1) + (carry1,126'b0,carry2,1'b0) + carry
00_1110	(carry2,carry,REG2)=(carry,REG2) + (carry1,126'b0,carry2,1'b0)
01_0000	(REG3)=(REG0)*(REG4)
01_0001	(REG1,REG0)=(RIN0)*(RIN1)
01_0010	(REG1,REG0)=(REG5)*(REG5)
01_0011	(REG1,REG0)=((REG1) + (REG5)*(REG5))
01_1001	(REG1,REG0)=((REG0) + (REG3)*(RIN1))
01_1010	(REG1,REG0)=((REG1) + (RIN0)*(RIN1))
01_1011	(REG1,REG0)=((REG0) + (RIN0)*(RIN1))
01_1100	(carry1,REG1,REG0)=((REG1) + (RIN1)*(REG5)*2)
10_0000	(REG0,carry)=(carry,RIN0)
10_0001	(carry,REG0)=(RIN0) + (RIN1) + carry
10_0010	(carry,REG0)=(RIN0) + (~(RIN1)) + carry
10_1000	(REG0) = (RIN0)
10_1001	(REG1) = (RIN1)
10_1010	(REG2) = (RIN0)
10_1100	(REG4) = (RIN0)
10_1101	(REG5) = (RIN0)
11_0000	all carry zero setting
11_0001	all carry one setting
11_0010	carry = (RIN0)[0]
11_0011	carry = (RIN0)[0] ^ carry
11_1000	(REG0) set output
11_1001	(REG1) set output
11_1010	(REG2) set output

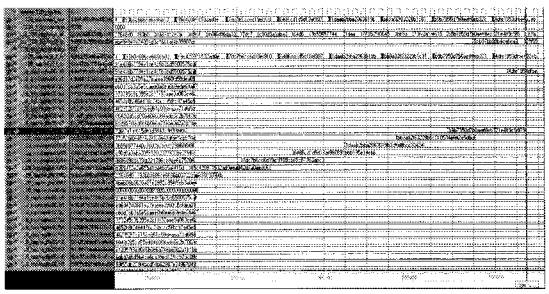

그림 5. 블록 몽고메리 연산 수행을 보이는 게이트 레벨 시뮬레이션 파형
Fig. 5. Gate simulation wave result in block Montgomery multiplication execution

VI. 결론 및 토의

RSA 연산은 법 곱셈 (modular multiplication)과 법 거듭제곱 연산 (modular exponentiation)이 주를 이루게 되는데 이러한 법 연산을 고속으로 처리할 수 있는 방법 중의 하나인 몽고메리 알고리즘을 효율적으로 처리하기 위해서는 고성능의 누적 곱셈 연산기 (MAC; multiply and accumulator)가 요구된다. 본 논문에서 연구한 누적 곱셈 연산기는 몽고메리 알고리즘을 처리하는 데 있어 제한된 레지스터의 용량을 가늠하여 누적곱셈의 가능한 모든 연산 경우를 미리 분석하고, 불필요한 연산을 제거함으로써 처리의 효율성을 높이도록 하였다. 또한 이 구조는 소면적 스마트카드에 내장되는 RSA 암호화 프로세서에서 사용할 수 있다.

참고문헌

[1] 김철, *암호학의 이해*, 영풍문고, 1996.

[2] 서광석, 김용태, 임종인, 김창한, *수론과 암호학*, 경문사, 1998.

[3] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography*, CRC press, 1997.

[4] R. L. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," *Communications of the ACM*, Vol. 21, pp. 120-126, Feb. 1978.

[5] P. L. Montgomery, "Modular multiplication without trial division," *Math. Comput.*, 44, pp. 519-521, 1985,.

[6] William Stallings, Cryptography and network security principles and practice, 3rd Edition, (c) 003 by Pearson Education, In

[7] J. L. Hennessy and D. A. Patterson, "Computer Architecture : A Quantiative Approach, 3rd edition", Morgan Kaufmann Publishers, CA, 2003.

[8] Douglas R. Stinson, Cryptography Theory and Practice 2nd Edition (c) 2002 by Chapman & Hall/CRC

[9] T. Izu and B. Möller, "Improved Parallel Elliptic Curve Multiplication Method Resistant against Side Channel Attacks", LNCS 2551, pp.296-313, 2002

저자소개

문 상 국(Sangook Moon)

1995 연세대학교 전자공학 학사
1997 연세대학교 전자공학 석사
2002 연세대학교 전자공학 박사
2002~2004 하이닉스반도체 선임연구원
2004~현재 목원대학교 전자정보보호공학부 조교수
※관심분야 : 정보보호 VLSI 설계, Data encryption, 유비쿼터스 컴퓨팅 보안