RSA 암호화 프로세서에 최적화한 32비트 곱셈기 설계

문 상 국*

Design of an Optimized 32-bit Multiplier 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 논문에서는 기본 워드를 128비트로 하고 곱셈 결과의 누적기로는 256비트의 레지스터를 사용하는 타겟 RSA 프로세서에서, 128 비트 곱셈을 효율적으로 수행하기 위하여 실험을 통하여 최적화한 32비트 * 32비트 곱셈기를 설계하고 검증하였다. 본 논문에서 설계한 곱셈기는 128비트 곱셈에 필요한 누적곱셈을 효율적으로 구현하는 데 필수적인 연산모듈이 된다. 구현된 곱셈기는 자동으로 합성하였고, 기준이 되는 RSA 프로세서의 동작 주파수에서 정상적으로 동작하였다.

ABSTRACT

RSA cryptoprocessors 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, a fast 32bit modular multiplier which is required to execute 128-bit MAC (multiplication and accumulation) operation is proposed. The proposed architecture prototype of the multiplier unit was automatically synthesized, and successfully operated at the frequency in the target RSA processor.

키워드

RSA, hybrid-multiplier, microprocessor

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 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

* 목원대학교 정보전자영상공학부

접수일자 2008. 07. 24

동시에 많은 소비자들은 보안 허점이 많은 인터넷을 안전하게 만들도록 요구하고 있다. 이에 따라 PC에 하드웨어 형태의 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 제공한다[5][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]. 본 논문에서는 고성능 누적곱셈연산의 기본요소가 되며 이 연산을 가장 효율적으로 처리할 수 있는 곱셈기의 구조에 대해 연구하여 제시하였다. 논문의 II장에서는 RSA 알고리즘에 대해 소개하고, III장에서는 곱셈기의 핵심 부분인 덧셈기에 대해 설명하고 IV장에서는 제안한 곱셈기를 논한다. V장에서 실험결과에 대해 토의하고 VI장에서 결론을 맺는다.

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.

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

3. RSA 공개키 암호화와 복호화

B가 A에게 보낼 메시지 m을 암호화하고, A는 복호화 한다.

a. 암호화 (encryption) : B는 다음 과정을 수행한다.
- A의 신뢰할 수 있는 공개키 (n,e)를 가져온다.
- $[0, n-1]$ 의 간격으로 메시지를 정수 m으로 나타낸다.

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

-$c = m^e \bmod n$ 을 계산한다.
- 암호문 c를 A에게 보낸다.

b. 복호화 (decryption) : A는 다음 과정을 수행한다.
- 암호문 c에서 평문 m을 얻기 위해 비밀키 (private key) d를 사용한다.
-$m = c^d \bmod n$을 계산한다.

4. RSA 복호화 과정

$ed \equiv 1 (\bmod \phi)$ 이기 때문에, $ed = 1 + k\phi$를 만족하는 정수 k가 존재한다.

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

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

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

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

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

Ⅲ. 덧셈기 구조

가산기는 직렬구조 (serial), 트리구조 (tree), 하이브리드구조 (hybrid) 등 크게 3가지로 분류할 수 있다.

직렬 구조 가산기는 최악조건 지연시간이 O(n)일 경우에 면적이 O(n)이 되는 특징을 가진다. 트리 구조 가산기는 최악조건 지연시간이 크게 감소되는 고성능 가산기 구조이다. 이 구조의 가산기는 O(logn)의 지연 시간을 가지는 장점을 가지고 있지만, O(nlogn)의 큰 면적을 갖는 단점을 가지고 있다. 하이브리드 구조의 가산기는 직렬구조와 트리구조의 중간성능을 갖는다. 하이브리드 구조는 n비트 블록으로 나누어서 병렬적으로 연산한다. 하이브리드 구조의 가산기는 면적이 O(n)이지만 지연시간은 O($\sqrt{n}$)이 된다. 면적은 직렬 가산기와 동일하지만, 지연시간이 더 빠른 장점을 가진다 [10].

덧셈기는 7가지 구조를 실제로 구현해 보고 회로 자동합성기술을 사용하여 합성한 다음 성능을 비교하였다. 구현한 가산기의 방식은 리플캐리가산기 (RCA), 캐리예측가산기 (CLA), 캐리예측가산기 기반의 리플캐리가산기 (RCAon CLA), 리플캐리가산기 기반의 캐리예측가산기 (CLAonRCA), 캐리예측가산기 기반의 캐리선택가산기 (CSAonCLA), 리플캐리가산기 기반의 캐리선택가산기 (CSAonRCA) 등 7가지 방식이다. 7가지 다른 구조로 설계하고 시뮬레이션해본 결과, 타겟으로 삼은 RSA 프로세서에 가장 적합한 덧셈기는 16비트 캐리예측가산기를 기반으로 하는 리플캐리가산기로 선택되었다. 그림 1은 선택된 하이브리드 구조의 기본 구조가 되는 16비트 덧셈기의 블록도이다.

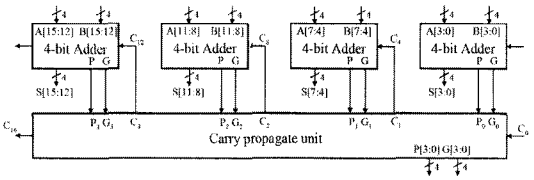

그림 1. 16비트 하이브리드 덧셈기 블록도
Fig. 1. 16-bit hybrid adder block diagram

Ⅳ. 제안하는 곱셈기의 구조

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.

2 제안하는 곱셈기 구조

Radix-4 수정 부스 알고리즘 (Booth's algorithm) 은 곱수 (multiplier)를 두 자리씩 분리하고 앞의 한 자리와 겹치도록 세비트씩 묶은 후에 이 안에서 부스 알고리즘을

적용시킨다. 분리된 세비트에 따라 피곱수에 +2X, +1X, 0X, -1X, -2X 등의 동작을 수행한다. 이것에 대해 더 진화한 알고리즘은 radix-8을 사용하는 것인데 곱수의 네비트씩 참조하여 피곱수에 +4X, +3X +2X, +1X, 0X, -1X, -2X, -3X, -4X 등의 동작을 수행하는 것이다. RSA에 최적화된 곱셈기의 구조를 찾기 위하여 각각에 대하여 구현하고 비교하였다. 수정 부스 알고리즘은 radix에 따라 다음 그림 2와 같이 곱수를 각각 분리하여야 한다.

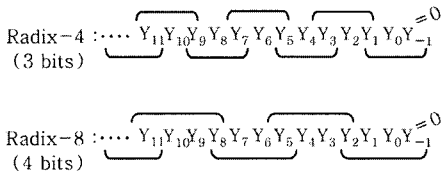

그림 2. 수정된 부스 알고리즘의 곱수 분리
Fig. 2. Bit grouping in the Modified Booth's algorithm

곱셈을 수행하기 위해서는 부분곱을 모두 더해야 하는데, 각각의 부분곱 열에서 발생하는 캐리의 전파(propagation)를 최소화하기 위해 월레스 트리를 사용하였다. 월레스 트리는 덧셈기의 덧셈 수행 시간과 캐리 전파 시간이 중첩되어 최적의 효과를 발생시키도록 덧셈기 모듈을 배열하여 구현하는 것으로, 부분곱을 합하는 과정에서 지연시간을 최대로 줄일 수 있도록 설계하였다.

제안하는 곱셈기가 탑재될 타겟 RSA 프로세서가 ARM 코어와 같이 동작하므로, 제안하는 곱셈기는 50Mhz 이상에서 동작이 가능하고, 휴대기기에서의 인증 가능한 암호화 프로세서 안에 장착되어야 하므로 면적은 작을수록 유리하다. 또한, 최대 2048비트의 연산을 수행해야 하므로 이러한 조건을 만족시키면서 많은 비트의 연산을 처리할 수 있는 곱셈기가 필요하다. 따라서 아래와 같이 6가지의 서로다른 구조의 곱셈기를 설계하였다. 다음 장에서 구현한 덧셈기를 내장시킨 여러 구조의 곱셈기의 성능을 비교한다.

1) 32*32 (radix-4 + 3:2 CSA)
2) 32*32 (radix-8 + 4:2 CSA)
3) 32*32 (verilog * operator)
4) 64*32 (radix-4 + 3:2 CSA)
5) 64*32 (verilog * operator)
6) 64*64 (verilog * operator)

V. 구현 및 토론

덧셈기는 자동 합성 도구를 사용하여 합성하였다. 모든 덧셈기는 Verilog HDL을 사용하여 탑-다운 설계방식으로 설계하였고, 검증은 Cadence의 Verilog로 합성된 설계물의 결과와 + 연산자를 이용한 결과가 같은지를 비교함으로써 검증하였다. 사용된 라이브러리는 삼성전자 0.18um CMOS 표준 셀 라이브러리를 사용하였고, 2.3V, 100℃ 최악조건에서 시뮬레이션을 수행하였다. 면적은 nand2 게이트 한 개를 기준으로 한다. 합성 옵션은 group/ungroup으로 분리하여 비교하였다.

시뮬레이션 결과를 분석하면, 캐리예측가산기를 사용하며 ungroup 옵션을 사용하는 것이 유리하다는 것을 보여준다. RSA 프로세서를 제어하는 ARM 프로세서가 40Mhz에서 동작하며 ARM 프로세서 1개의 동작에 내재된 알고리즘에 의하면 덧셈기는 4번 실행이 되어야 하므로, 덧셈기는 최소 160Mhz 이상의 동작주파수를 가져야 한다. 또한 가장 작은 면적을 가져야 하므로 본 연구에서는 16비트 캐리예측가산기를 기반으로 하는 256 비트의 RCA를 선택하였다. 이렇게 선택된 가산기는 250Mhz의 속도로 동작하며, nand2 게이트 기준으로 약 4328의 면적을 가진다. 표 2에서 다양한 방식으로 합성한 서로다른 구조의 덧셈기의 사양을 비교한다.

표 2. 덧셈기 성능 비교
Table 2. Adder performance comparison

덧셈기 구조	그룹 크기	64 bits		128 bits		256 bits	
		시간	면적	시간	면적	시간	면적
RCA	.	2.16	1711	2.69	3268	3.74	6438
CLA	.	2.37	1342	2.57	2515	3.37	5192
(+) 자동합성	.	2.18	1382	2.37	2902	2.82	5400
RCAonCLA	4	2.29	1596	2.73	3029	3.49	5531
	8	2.71	1290	4.26	2140	6.62	4404
	16	2.42	1420	3.32	2574	4.99	4331
CLAonRCA	4	2.12	1838	2.79	3329	2.92	5804
	8	2.15	1604	2.95	3149	3.54	6020
	16	2.67	1504	2.84	3268	3.32	6648
CSAonCLA	4	2.24	1256	2.74	2351	3.09	4779
	8	2.30	1374	2.74	2448	3.17	5207
	16	2.90	1568	3.18	2956	3.27	5935
CSAonRCA	4	4.05	1635	9.85	3063	22.52	5607
	8	2.84	1832	4.15	3343	8.72	6486
	16	3.09	1879	3.75	3694	4.96	4534

곱셈기는 RSA 알고리즘에 적용되도록 128 폭의 두 오퍼랜드의 곱이 가능하면 되는데, IV장에서 보인 6가

지 서로 다른 구조를 합성하여 덧셈기와 마찬가지 방식으로 비교하였다.

RSA 암호프로세서에 탑재될 곱셈기는 덧셈기와 연동되어 동작한다. RSA 암호프로세서의 최대 연산비트가 2048비트이므로 덧셈기는 큰 편이 유리하지만, 곱셈기는 덧셈기에 비해 크기가 비트수의 제곱에 비례하므로 곱셈기 선택은 반드시 분석을 통하여야 한다. 본 논문에서 타겟으로 하는 RSA 프로세서가 스마트카드와 같은 소형 휴대기기이므로 어느정도 동작속도를 보장하면서 가능하면 최소 크기를 선택하여야 한다.

입력 비트가 최대 2048비트이므로 곱셈기의 크기가 클수록 곱셈연산이나 MAC (Multiply -And-Add) 연산 사이클 수는 줄게 된다. 표 3에는 서로 다른 6가지 방식으로 구현한 곱셈기의 성능을 비교하였다. 크기를 줄이면서 성능을 끌어내기위한 구조는 64*32비트 구조인데, 실제로 속도로 얻는 이득보다는 면적에 대한 손실이 크게 분석되어 본 연구에서는 면적과 동시에 연산시간도 확보하기 위하여 radix-4 방식의 4:2 CSA를 내장한 32*32 곱셈기를 선택하였다. 제안한 구조의 곱셈기는 127Mhz에서 동작하며 nand2 게이트 기준으로 약 10963 게이트 면적을 차지한다.

표 3. 곱셈기 구조 비교
Table 3. Multiplier structure comparison

multiplier	area (nand2)	time (ns)
32 * 32 multiplier (Radix 4, (3, 2) CSA)	11250	6.55
32 * 32 multiplier (Radix 8, (4, 2) CSA)	10963	7.87
32 * 32 multiplier ("*" operator)	7716	20.65
64 * 32 multiplier (Radix 4, (3, 2) CSA)	21279	6.93
64 * 32 multiplier ("*" operator)	14921	21.10
64 * 64 multiplier ("*" operator)	27914	40.78

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 요구되며, 이를 구성하는 곱셈기에 대한 최적화가 필수요건이 된다. 본 논문에서 제안한 최적화된 곱셈기의 구조는 몽고메리 알고리즘을 처리하는 데 있어 가장 빈번히 사용되는 연산모듈이기때문에 속도와 면적을 동시에 고려하였다. 제안된 곱셈기의 구조는 소면적 스마트카드에 내장되는 RSA 암호화 프로세서의 성능을 최적화하여 복잡한 계산을 요하는 인증 연산 등 고비도의 정보보호 어플리케이션을 처리하는 데 효율적으로 적용될 수 있다.

VII. 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

[10] 허석원, "스마트카드 구현에 적합한 최적화된 RSA 암호화프로세서 설계", 연세대학교 석사학위 졸업논문, 2003.

저자소개

문상국 (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