고속 RSA 모듈러 곱셈을 위한 시스톨릭 어레이의 설계†

강 민섭, 남 승용
안양대학교 정보통신·컴퓨터공학부
e-mail : mskang@aycc.anyang.ac.kr

Design of Systolic Array for Fast RSA Modular Multiplication

Min-Sup Kang, and Sung-Yong Nam
Dept. of Computer Science & Electrical Engineering, Anyang University

요 약

본 논문은 RSA 암호시스템에서 고속 모듈러 곱셈을 위한 최적화된 시스톨릭 어레이의 설계를 제안한다. 제안된 방법에서는 미리 계산된 가산결과를 사용하여 개선된 몽고메리 모듈러 곱셈 알고리듬을 제안하고, 고속 모듈러 곱셈을 위한 새로운 구조의 시스톨릭 어레이를 설계한다. 미리 계산된 가산결과를 얻기 위해 CLA(Carry Look-ahead Adder) 를 사용하였으며, 이 가산기는 덧셈연산에 있어서 캐리전달 지연이 제거되므로 연산 속도를 향상 시킬 수 있다.

제안된 시스톨릭 구조는 VHDL(VHSIC Hardware Description Language)을 사용하여 동작적 수준을 기술하였고, Ultra 10 Workstation 상에서 Synopsys™ 툴을 사용하여 합성 및 시뮬레이션을 수행하였다. 또한, FPGA 구현을 위하여 Altera MaxplusII 를 사용하여 타이밍 시뮬레이션을 수행하였고, 실험을 통하여 제안한 방법을 효율성을 확인하였다.

1. 서 론

공개키 암호화 시스템을 기본으로 한 데이터 암호화는 수 년 동안 데이터 통신과 전자상거래 등에서 광범위 하게 사용되어 왔다[1-2]. 이러한 시스템 중에서 RSA 암호시스템은 디지털 서명과 암호화된 메시지를 보낼 때 가장 많이 사용하는 암호화 시스템이다. RSA 암호시스템의 핵심 연산은 매우 큰 수를 사용한 모듈러 곱셈을 사용하여 모듈러 멱승 연산을 수행한다. 그러나, 이 연산은 큰 수의 모듈러 멱승 연산을 반복적으로 요구하기 때문에 많은 연산 시간을 필요로 한다.

다양한 버전의 몽고메리 알고리듬이 빠른 모듈러 멱승연산을 위해 제안되었고, 다양한 고성능 암호시스템이 하드웨어로 구현되었다[3-12]. 1985 년에 효율적인 모듈러 곱셈을 위한 몽고메리 알고리듬이 제안된 후, 모듈러 곱셈의 연산 속도를 향상 시키기 위한 개선된 몽고메리 알고리듬이 제안되었다[8-12].그러나 대부분의 알고리듬들은 over-large residue 문제 가 존재하거나, 많은 하드웨어 자원을 필요로 한다. C.D.Walter[4]는 몽고메리 모듈러 곱셈 알고리즘의하드웨어 구현을 위한 시스트릭 어레이 구조를 제안하였다.

본 논문에서 RSA 암호시스템에서 고속 모듈러 곱셈을 위한 최적화된 시스톨릭 어레이의 설계를 제안한다. 제안된 방법은 미리 계산된 가산결과를 사용하여 개선된 몽고메리 모듈러 곱셈 알고리듬을 제안하고, 고속 모듈러 곱셈을 위한 새로운 구조의 시스톨릭 어레이를 설계한다. 미리 계산된 가산결과를 얻기 위해 CLA(Carry Look-ahead Adder)를 사용하였으며, 이 가산기는 덧셈연산에 있어서 캐리전달 지연이 제거되므로 연산 속도를 향상 시킬 수 있다.

제안된 시스톨릭 구조는 VHDL(VHSIC Hardware Description Language)을 사용하여 동작적 수준을

†본 연구는 연구년 기간중에 연구되었음.

기술하였고, Ultra 10 Workstation 상에서 Synopsys™ 툴을 사용하여 합성 및 시뮬레이션을 수행하였다[13]. 또한, FPGA 구현을 위하여 Altera MaxplusII 를 사용하여 타이밍 시뮬레이션을 수행하였고, 실험을 통하여 제안한 방법을 효율성을 확인하였다.

2. 몽고메리 알고리듬

몽고메리 알고리듬은 AB mod N 의 모듈러 곱셈을 계산 하기 위하여 효과적으로 사용되어져 왔다. 여기서 A, B, N 은 k-bit 의 2 진수 이다. 이 알고리듬은 수체계의 변환을 이용하여 모듈러 연산이 쉽게 되는 방법을 사용하고 있다. C. D. Walter 는 이 알고리듬을 하드웨어 구현에 적합한 시스톨릭 어레이 구조를 제안하였다.

<그림 1>은 몽고메리 곱셈 알고리듬을 하드웨어 구현에 적합한 C-like 코드를 나타낸다.

```
MM(A, B, N)
{
1:  P0 = 0;
2:  for (i = 0; i < n; i++) {
2a:    Q[i] = (Pi + A[i] ·B) mod 2;
2b:    Pi+1 = ( Pi + A[i] ·B + Q[i]·N) / 2;
    }
3:  if (Pn ≥ N) return Pn -N;
    else return Pn;
}
```

<그림 1> 몽고메리 알고리듬

MM()의 최종 결과는 P_n 또는 $P_n - N$ 인 $P_n = A \cdot B \cdot 2^{-1} \pmod N$ 이며, 여기서 n 은 N 의 비트수를 나타낸다. 따라서 P_n 은 $2^n P = A \cdot B + Q \cdot N$ 이므로 $P \equiv 2^{-n} A \cdot B \pmod N$ 을 만족한다. 단, $0 \le P_i < B + N < 2N$ (i = 0, ... , n-1) 임. 정확한 결과값을 얻기 위하여 이 알고리즘은 Step 3 에서의 뺄셈 연산을 해주어야 하는 후처리 과정이 필요하다. 몽고메리 알고리듬을 기본으로한 시스톨릭 어레이에 사용되는 PE(Processing Unit)는 두개의 n-bit 전 가산기와 모듈러 감소로 구성되어져 있다. 그러므로 계산의 복잡성은 P_n 을 계산하기 위한 두 개의 n-bit 가산 연산에 주어진다.

<그림 2>는 모듈러 멱승 연산을 위한 알고리듬을 나타내며, 이를 위해서 몽고메리 모듈러 곱셈 알고리듬 MM()을 사용한다.

```
MonExp(M, N, E)
   M = M·R mod N;
   X = R mod N;
   for (i = k-1; i >= 0; k--) {
       X = MM(X, X);
       if (Ei == 1) X = MM(M, X);
   }
   X = MM(X, 1);
   return X;
```

<그림 2> 모듈러 멱승 알고리듬

<그림 2>의 전처리 과정에서는 M = M·R mod N 을 계산하며, 이때 R=2 이면, $M = \sum_{i=0}^{n-1} M[i] \cdot 2^i$ 을 만족시킨다. 이 알고리듬의 최종 결과는 0 ≤ A, B < N 인 조건을 항상 만족시킬 수 없으므로 결과 M 은 MM()의 입력으로 직접 사용할 수 없다. 또한, MM()에서 사용되는 입력 값 M, X 를 얻기 위하여 추가적인 시간과 하드웨어 비용이 요구된다.

3. 개선된 모듈러 곱셈 알고리듬

종래에 제안된 모듈러 곱셈 알고리듬은 신호의 흐름을 제어하기 위해 3 비트의 제어 신호를 사용하였다[13]. 본 논문에서는 최적화된 고 성능의 암호시스템을 설계하기 위하여 2 비트의 신호선을 이용한다. <그림 3>은 개선된 모듈러 곱셈 알고리듬을 나타내며, 이 알고리듬은 문헌 [13]에서 제안된 방법을 기본으로 하고 있다.

```
Mod_Mul(A, B, N)
{
(1)  S[0] = 0;

(2)  BPN = Σ_{i=0}^{n-1} xi · 2^i = B + N;
(3)  for(i = 0; i < n+1; i++) {
(3a)    qi = (S[i] + Ai·B) mod 2;
        Sel2[0] = Ai, Sel2[1] = qi;
(3b)    case (Sel2[])
          "00": S[i+1] = S[i] ;
          "01": S[i+1] = (S[i] + N)/2;
          "10": S[i+1] = (S[i] + B)/2;
          "11": S[i+1] = (S[i] + BPN)/2;
    }
(4)  return S[n+1];
}
```

<그림 3> 개선된 모듈러 곱셈 알고리듬

이 알고리듬은 step 2 와 3 의 두개의 단계로 나누어진다. Step 2 에서는 추가되는 가산 연산이 이루어지고 Step 3b 에서는 모듈러 감소 연산이 수행 된다. BPN 이 미리 연산이 이루어진 후 Sel2 시그널이 초기화가 되고 Sel2 시그널에 의한 선택적인 가산연산의 단 한번의 수행으로 S[i+1]의 값을 얻을 수 있다. 기존의 몽고메리 알고리듬에서는 세개의 오퍼랜드를 통한 두번의 곱셈과 두번의 가산을 요구하였다.

MM()은 두개의 오퍼랜드에 대한 비교와 감산 연산의 후처리 과정이 더 요구된다.

루프를 n+1 번 수행함으로써 마지막 $A_{n+1} = 0$ 을 알 수 있음으로 최종결과 S[n+1]이 S[n+1] < 2N 을 만족함을 알 수 있다. 따라서 MM()에서의 후처리 과정에 대한 오버해드는 간단히 해결 할 수 있다. 따라서 Mod_Mul()은 $S[n+1] = A \cdot B \cdot 2^{-(n+1)} \pmod N$ 을 계산 한다.

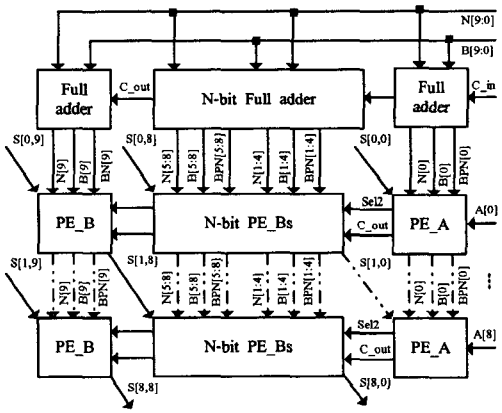
MM()에서는 $\overline{M} = M \cdot R \bmod N$ 의 연산에 MM()을 사용 할 수 없다. 이 문제를 해결하기 위하여 $R_2 = 2^{n+1}$ 와 $R_4 = (R_2)^2$ 를 정의 하고 $\overline{R_4} = R_4$

mod N 을 정의 하여 Mod_Mul()에 대하여 0 ≤ A and B < N 을 만족하게 만든다.

4. 새로운 시스톨릭 어레이 구조

이 장에서는 우리의 모듈러 멱승 알고리듬을 기본으로 한 최적화된 시스톨릭 어레이 설계에 대하여 설명한다. 제안된 시스톨릭 모듈러 곱셈기는 n-bit CLA 가산기와 n-bit Processing Elements(PEs)로 구성되어져 있다. n-bit RCA 와 ripple 가산기는 Mod_Mul()의 전처리 구간인 BPN 의 계산을 위해 사용된다.

PE 는 두 가지 종류의 셀로 구성되어진다. PE_A 라는 컨트롤 셀과 PE_B 라는 기본 셀로 구성된다. PE_B 는 n-bit CLA 를 사용하며 n-bit 의 연속적인 결과를 얻을 수 있다. 그림 4 는 새로운 시스톨릭 어레이의 구조에 적용된데이터 흐름도를 보여준다.

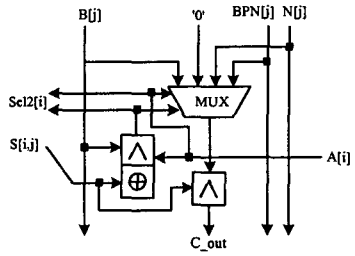


<그림 4> 제안된 시스톨릭 어레이 구조

N[9:0]은 입력되는 N 의 비트수를 나타낸다. II 장에서 설명한 Mod_Mul()은 두개의 step 으로 구성되어진다. BPN 을 구하는 동작과 몽고메리 모듈러 연산으로 이루어 진다. BPN 은 모듈러 연산을 수행하는 부분으로 할당이 되어지고 모듈러 연산의 PE 의 부분 연산 값은 다음 PE_B 또는 다음 열의 PE_A 로 할당이 되어지고 S[n+1]의 값을 얻을 때 까지 반복 되어진다. 제안된 시스톨릭 어레이는 n+1 bit 의 열과 행을 가지지만 미리 연산되는 값들을 사용함으로써 기존의 몽고메리 알고리듬보다 빠르다.

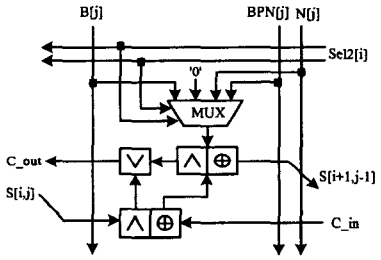
BPN 값을 얻기 위한 n-bit Full Adder 는 Ripple-carry adder 이다.

qi 는 각각의 PE_A 에서 한번 계산된다. 이것은 Ai·B[0]의 논리 AND 와 S[i]의 LSB 와 Exclusive_Oring 을 통해서 얻을 수 있다. 그림 5 는 제안된 PE_A 이다. 4*1 MUX, 1 개의 HA 와 AND gate 로 구성되어져 있다.



<그림 5> Sel2 생성을 위한 제안된 PE_A

Sel2 시그널은 0, B, N 또는 BPN 중 한 개의 값을 선택하여 C_out 를 생성시키며, 또한 PE_B 를 제어할 신호로써 오른쪽에서 왼쪽으로 지속 적으로 전달한다. 그림 6 은 모듈러 곱셈연산을 향상시키기 위한 새로 제안된 PE_B 를 나타낸다. 제안된 PE_B 는 1 개의 FA 와 4*1 MUX 로 구성되어있다.



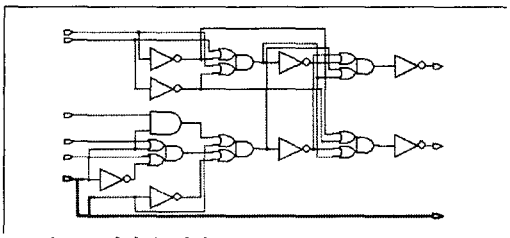
<그림 6> 모듈러 감소를 위한 제안된 PE_B 구조

미리 계산된 BPN 은 직접 MUX 에 입력이 되고 S[i+1,j+1]을 계산하기 위해 사용되어진다. 그리고 C_out 은 다음의 PE_B 의 입력 값으로 쓰여진다

5. 구현 및 실험 결과

VHDL(VHSIC Hardware Description Language)을 사용하여 설계하였으며, IDEC* 에서 지원된 Ultra 10 Workstation 상에서 Synopsys™ software 를 사용하여 합성 및 시뮬레이션을 수행하였다.

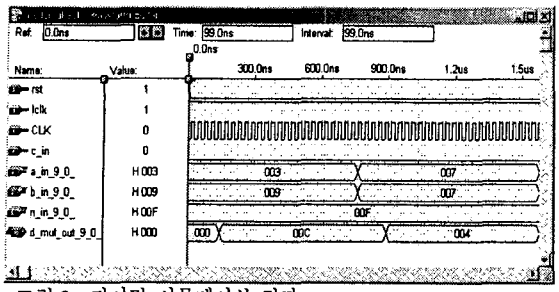
합성을 위하여 Altera 10K library 를 사용하였다. 그리고 셀의 수는 약 10 이다. 그림 7 은 합성을 통하여 얻어진 제안된 알고리듬의 기본 셀을 나타내며, 셀 수는 약 6 이다.



<그림 7> 제안된 기본 셀(PE_B)

그림 8 은 Altera MaxplusII softwar 상에서 8-bit 모듈러

곱셈의 타이밍 시뮬레이션 결과이다.



<그림 8> 타이밍 시뮬레이션 결과

Maxplus II software 사용하여 몽고메리 방식과 제안된 방식과의 타이밍 시뮬레이션 결과를 비교하였다. <표 1>은 16-bit, 32-bit, 64-bit, 128-bit 에대한 모듈러 곱셈의 하드웨어 비용과 Critical path 의 비교 결과를 나타낸다.

<표 1> 실험 결과

Items \ Approaches			walter	Our1	Our2
No. of cells	Systolic Array	16-bit	162.3	146	137.5
		32-bit	314.3	278	261.5
		64-bit	618.3	542	509.5
		128-bit	1226.3	1070	1135
Critical Path (ns)	PE_A	1-bit	12.1	12.6	12.6
	PE_B	1-bit	15.6	14	–
	PE_B4	4-bit	–	–	23.6
	Systolic Array	16-bit	102.3	71.8	78.1
		32-bit	191.9	128.6	135.7
		64-bit	361.5	269	287.2
		128-bit	737.5	482.1	537.3

Our1 은 1-bit X_PE, 1-bit Y_PE, 1-bit RCA 를 사용하였으며, 여기서 1-bit RCA 를 기본으로 n-bit RCA 를 사용하여 BPN 값은 얻기 위해 사용 하였다. Our2 는 1-bit X_PE, 4-bit Y_PE, 4-bit CLA 를 사용하였으며, 4-bit CLA 를 기본으로 n-bit CLA 를 사용하여 B+N 값을 얻기 위해 사용 하였다. <표 1>의 실험 결과를 통하여 본 논문에서 제안한 시스톨릭 어레이 구조가 하드웨어 비용 및 속도면에서 Walter 의 방법보다 성능이 우수함을 보여주고 있다.

현재, 제안된 알고리듬을 기본으로하여 1024 비트 고속 암호 프로세서의 설계 중에 있다.

6. 결 론

본 논문에서는 RSA 암호시스템에서 고속 모듈러 곱셈을 위한 최적화된 시스톨릭 어레이의 설계에 관하여 기술하였다. 제안된 방법에서는 미리 계산된 가산 결과를 사용하여 개선된 몽고메리 모듈러 곱셈 알고리듬을 제안하였고, 고속 모듈러 곱셈을 위한 새로운 구조의 시스톨릭 어레이를 설계하였다.

제안된 시스톨릭 구조는 VHDL 을 사용하여 동작적 수준을 기술하였고, Ultra 10 Workstation 상에서 Synopsys™ 툴을 사용하여 합성 및 시뮬레이션을 수행하였다. 또한, FPGA 구현을 위하여 Altera MaxplusII 를 사용하여 타이밍 시뮬레이션을 수행하였고, 실험을 통하여 제안한 방법을 효율성을 확인하였다.

<참고문헌>

[1]R. L. Rivest, A. Shrmir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," Communications of the ACM, Vol.21, No.2, pp.120- 126, Feb. 1978.

[2] T. ElGmal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," IEEE Trans. On Information Theory, Vol. IT-31, No. 4, pp. 469-472, 1985.

[3] S. E. Eldridge and C. D. Walter, "Hardware implementation of Montgomery's modular multiplication algorithm," IEEE Trans. On Computers, Vol. 42, No. 6, pp. 693-699, July 1993.

[4] C. D. Walter, "Systolic modular multiplication," IEEE Trans. On Computers, Vol. 42, pp. 376-378, 1993.

[5] Ko, C. K. and Hung, C. Y., "Bit-level systolic arrays for modular multiplication," Journal of VLSI Signal Processing, Vol. 3, pp. 215-223, 1991.

[6] C. K. Koc, "RSA Hardware Implementation," RSA Lab. Aug. 1995.

[7] P. L. Montgomery, "Modular multiplication without trial division," Math. Of Comput., Vol. 44, No. 170, pp. 519-521, April 1985.

[8] Takenaka, M., Toni. N., Hansebe, T., and Akiyama. R., "A study of efficient RSA encryption algorithm suitable for network key management(in Japanese)," Tech. Report IEICE, IN-9315, pp. 11-16,1993.

[9] K. Iwamura, T. Matsumoto, and H. Imai, "Systolic arrays for multiplication algorithm for systolic arrays," Proc. Euro CRYPT'92, pp. 477-481, 1992.

[10] J. B. Shin, et. Al., "Optimization of Montgomery modular multiplication algorithm for systolic arrays," ELECTRONICS LETTERS, Vol. 34, No. 19, pp. 1830-1831, 1998.

[11] C. Y. Su, S. A. Hwang, P. S. Chen, and C. W. Wu, "An Improved Montgomery's algorithm for high-speed RSA public-key cryptosystem," IEEE Trans. On VLSI System, Vol. 7, No. 2, June 1999.

[12] C. C. Yang, T. S. Chand, and C. E. Jen, "A new RSA cryptosystem hardware design based on Montgomery's algorithm," IEEE Trans. On Circuit and System-II, Vol. 45, No. 7, July 1998.

[13] M. S. Kang and D. W. Kim, "Systolic array based on fast modular multiplication algorithm for RSA cryptosystem," Proc. Of IEEE TENCON99, pp. 305-308, Sept. 1999.

---

* 본 연구는 IDEC(반도체설계교육센터)의 일부 지원에 의해서 수행되었음.