논문 02-27-9C-5 한국통신학회논문지 '02-9 Vol.27 No.9C

가변길이 고속 RSA 암호시스템의 설계 및 하드웨어 구현

준회원 박 진 영*, 서 영 호* 정회원 김 동 욱*

Design and Hardware Implementation of High-Speed Variable-Length RSA Cryptosystem

Jin-Young Park*, Young-Ho Seo* *Associate members*

Dong-Wook Kim* *Regular Member*

요 약

본 논문에서는 RSA 암호 알고리즘의 연산속도 문제에 초점을 맞추어 동작속도를 향상시키고 가변길이 암호화가 가능하도록 하는 새로운 구조의 1024-비트 RSA 암호시스템을 제안하고 이를 하드웨어로 구현하였다. 제안한 암호시스템은 크게 모듈러 지수승 연산 부분과 모듈러 곱셈 연산 부분으로 구성되었다. 모듈러 지수승 연산은 제곱 연산과 단순 곱셈 연산을 병렬적으로 처리할 수 있는 RL-이진 방법을 개선하여 적용하였다. 그리고 모듈러 곱셈 연산은 가변길이 연산과 부분 곱의 수를 감소하기 위해서 Montgomery 알고리즘에 4 단계 CSA 구조와 기수-4 Booth 알고리즘을 적용하였다.

제안한 RSA 암호시스템은 하이닉스 0.35um Phantom Cell Library를 사용하여 하드웨어로 구현하였고 최대 1024-비트까지 가변길이 연산이 가능하였다. 또한 소프트웨어로 RSA 암호시스템을 구현하여 하드웨어 시스템의 검증에 사용하였다. 구현된 하드웨어 RSA 암호시스템은 약 190K의 게이트 수를 나타내었으며, 동작 클록 주기는 150MHz이었다. 모듈러스 수의 가변길이를 고려했을 때, 데이터 출력률은 기존 방법의 약 1.5배에 해당한다. 따라서 본 논문에서 제안한 가변길이 고속 RSA 암호시스템은 고속 처리를 요구하는 각종 정보보호 시스템에서의 사용 가능성을 보여주었다.

ABSTRACT

In this paper, with targeting on the drawback of RSA of operation speed, a new 1024-bit RSA cryptosystem has been proposed and implemented in hardware to increase the operational speed and perform the variable-length encryption. The proposed cryptosystem mainly consists of the modular exponentiation part and the modular multiplication part. For the modular exponentiation, the RL-binary method, which performs squaring and modular multiplying in parallel, was improved, and then applied. And 4-stage CSA structure and radix-4 booth algorithm were applied to enhance the variable-length operation and reduce the number of partial product in modular multiplication arithmetic.

The proposed RSA cryptosystem which can calculate at most 1024 bits at a time was mapped into the integrated circuit using the Hynix Phantom Cell Library for Hynix 0.35㎛ 2-Poly 4-Metal CMOS process. Also, the result of software implementation,which had been programmed prior to the hardware research, has been used to verify the operation of the hardware system. The size of the result from the hardware implementation

* 광운대학교 전자재료공학과 Digital Design & Test 연구실

논문번호 : 020281-0624, 접수일자 : 2002년 6월 24일

※ 이 논문은 2002년도 광운대학교 교내 학술연구비 지원에 의해 연구되었음.

was about 190k gate count and the operational clock frequency was 150MHz. By considering a variable-length of modulus number, the baud rate of the proposed scheme is one and half times faster than the previous works. Therefore, the proposed high speed variable-length RSA cryptosystem should be able to be used in various information security system which requires high speed operation.

Ⅰ. 서 론

네트워크와 인터넷의 발달로 인해 정보의 교환이 급속도로 활성화되어 다양한 회선망을 통한 송/수신 데이터의 보안문제가 중요한 사안이 되고 있다. 이러한 정보 보호기능 문제의 해결책으로 암호 알고리즘의 개발 및 그것의 효율적인 구현에 대한 연구가 활발히 이루어지고 있다[1].

암호 알고리즘은 암호화/복호화 과정에서 동일한 키를 사용하는 대칭키 또는 비밀키 암호 알고리즘과 암호화 과정에서는 공개키를 사용하고 복호화 과정에서는 비밀키를 사용하는 비대칭키 또는 공개키 암호 알고리즘으로 구분한다. 전자는 정보의 기밀성만을 제공하지만 후자는 기밀성, 키 교환 및 인증 기능을 제공할 수 있으므로 개방형 네트워크에서 적합한 암호솔루션으로 인식되고 있다[2]. 공개키 암호방식의 개념은 1976년 Diffie와 Hellman에 의해 제안된[3] 이후 Rivest, Shamir 그리고 Adleman에 의해 1977년에 RSA 알고리즘[4]이 개발되어 1978년에 처음 공포되었다. RSA 알고리즘은 간단한 수학적 배경, 체계적인 시스템, 그리고 높은 안전도로 전 세계적으로 상용화되고 있으나 암호화/복호화가 수학적 연산으로 수행되기 때문에 순열과 치환에 의한 대칭키 암호 알고리즘과 비교하여 연산 속도가 느린 단점을 가지고 있다[5]. 따라서 수행속도를 향상하기 위한 RSA 암호시스템의 하드웨어 구현에 관한 연구가 많이 진행되어 왔다.

RSA 암호시스템의 하드웨어 구현의 관점에서 수학적 연산을 최소화하기 위해 1985년 P. L. Montgomery는 RSA 알고리즘의 모듈러 곱셈을 효율적으로 수행할 수 있는 알고리즘을 개발하으며[6], 그 이후 많은 개선된 Montgomery 알고리즘들이 제안되었다[7]-[11]. 그 중 1999년 Jyh-Huei Guo는 [7]에서 제안되었던 Montgomery 알고리즘에서 곱셈 부분과 모듈러 연산 부분을 분리하는 개념을 이용하여 시스톨릭 어레이(systolic array)구조의 RSA 암호시스템을 구현하여 시스템의 수행속도를 향상시켰다[9]. 또한 2000년 Jin-Hua Hong는 Montgomery 알고리즘에 기수-4 Booth 알고리즘을 적용하여 더욱 효율적인 모듈러 곱셈 알고리즘을 제안하고 하드웨어로 구현하였다[10]. 이상의 알고리즘들은 셀 구조에 있어 CPA(Carry Propagation Adder)를 사용하는 시스톨릭 어레이 구조로 구현되어 고정길이 연산을 수행하므로 속도에 한계가 있다. 반면 CSA (Carry Save Adder)에 기초한 구조로 구현하는 것은 수행속도보다 하드웨어 면적을 최소화하기 위해 적용되어왔다[11].

본 논문에서는 수행속도를 더욱 향상하기 위해 암호시스템에서 요구되는 모듈러 곱셈 알고리즘과 모듈러 지수승 알고리즘을 제안한다. 이 모듈러 곱셈 알고리즘은 Montgomery 알고리즘에 CSA 구조와 기수-4 Booth 알고리즘을 적용하고 모듈러 지수승 알고리즘은 RL-이진 방법[22]을 개선하여 적용한다. 또한 이러한 구조적 특성을 이용하여 암호화 수행에 있어서 고정길이가 아닌 가변길이의 연산 특성을 나타내도록 설계한다. 제안한 RSA 암호시스템은 최대 1024-비트 가변길이 고속 RSA 암호 시스템을 구현하여 기존의 방법과 동작속도 및 하드웨어 자원 사용 등에 대해 비교한다.

Ⅱ. RSA 알고리즘

RSA 알고리즘은 소인수분해의 어려움에 안전도의 근간을 두고 있고, n-비트의 길이를 갖는 메시지 블록, 공개키(public key), 비밀키(private key) 그리고 모듈러스(modulus) 수를 사용하여 암호화/복호화 연산뿐만 아니라 키 교환과 전자서명 모두를 제공한다. 이 알고리즘은 안전도를 위해 일반적으로 1024-비트를 사용하고 스마트 카드에서는 2048-비트까지 사용한다. 암호화/복호화 과정은 각각 식(1), (2)와 같이 모듈러 지수승 연산에 의해서 이루어진다.

$$C = Me \bmod n \quad (1)$$

$$C = Cd \bmod n \quad (2)$$

여기서 암호문(C)은 공개키 (n, e)를 사용하여 계산되고 메시지 블록 또는 평문(M)은 비밀키 (d)를 사용하여 계산된다. 이 알고리즘에서 대부분의 연산은 모듈러 지수승 연산에 의해서 이루어지고 매우

큰 정수를 사용하므로 암호시스템에서 상당한 수학적 계산량을 필요로 한다. 따라서 본 장에서는 RSA 알고리즘을 효율적으로 구현하기 위해 기존에 제안되었던 수학적 연산 방식을 설명한다.

II-1. 모듈러 지수승 연산

RSA 알고리즘의 직접적인 연산방법은 ME mod N을 계산하는 경우 M 에 대한 곱셈을 (E-1)번 수행하고 다시 그 결과에 대해 mod N 연산을 수행하는 것이다. 실제의 1024-비트 암호화 과정에서는 많은 곱셈 연산이 요구되고 곱셈의 중간 결과는 매우 큰 정수가 발생된다. 이러한 문제점들을 개선하기 위해 RL(Right to Left)-이진 방식[12]이 제안되었으며 식 (3)과 같은 모듈러 연산의 특성을 이용하여 모듈러 곱셈의 중간 결과를 축소한다.

$$(M \times M) \bmod N = [(M \bmod N) \times (M \bmod N)] \quad (3)$$

따라서 이 연산을 이용하면 식 (3)의 좌변 계산 중에 발생되는 매우 큰 중간결과 (M×M)을 배제할 수 있다. 그리고 모듈러 지수승 연산을 이진법으로 표현하면 식 (4)와 같고, 이것은 i번의 모듈러 거듭제곱 연산과 지수 비트 값에 따라 모듈러 제곱 연산의 결과에 단순 모듈러 곱셈을 반복함으로써 모듈러 지수승 연산 결과를 얻어 모듈러 곱셈의 반복 횟수를 감소시킬 수 있다.

$$M^E \bmod N = M^{\sum_{e_i} 2^i} \bmod N = [\prod_{e_i=0} M^{(2^i)}] \bmod N = \prod_{e_i=0} [M^{(2^i)} \bmod N] \quad (4)$$

식(3)과 식(4)를 이용하여 M^E mod N을 효율적으로 계산할 수 있는 RL-이진 방식의 의사코드는 다음과 같다.

```
RL Binary Method(M, E, N) {
-- Output : C = M^E mod N
            S = M ;     C = 1 ;
    for (i=0 ; i<k ; i++) {
      if (e_i=1)  C = C × S mod N ;
      else        C = C ;
      S = S × S mod N ;
     }
     return C ;
}
```

이 알고리즘은 모듈러 제곱 연산과 모듈러 단순 곱셈을 병렬적으로 처리하므로 모듈러 지수승의 연산 속도를 향상할 수 있다. 따라서 본 논문은 RL-이진 방식을 개선하여 적용한다

II-2. 모듈러 곱셈 연산

모듈러 곱셈의 직접적인 방법은 두 수를 곱한 후 그 결과에 대해 나눗셈 연산을 하는 것이다. 그러나 이 연산 또한 곱셈 결과가 매우 큰 값을 갖게 되고 나눗셈 연산으로 인한 연산시간의 증가와 함께 하드웨어 구현의 복잡도가 매우 증가한다. 이러한 문제를 해결하기 위해 이진 연산을 기초로 하여 덧셈 연산만으로 모듈러 곱셈 연산을 수행하는 Montgomery 알고리즘[6]이 개발되었다. 이 알고리즘은 두 수의 모듈러 곱셈을 위해 홀수로 가정한 k-비트 모듈러스 N(nk-1, ..., n1, n0), k-비트 정수인 피승수 Y(yk-1,yk-2, ... y1, y0)와 승수 X(xk-1, xk-2, ... x1, x0)를 입력으로 사용한다. 아래의 의사코드는 $Y \cdot X \pmod N$를 계산하기 위한 Montgomery 모듈러 곱셈 알고리즘을 나타내고 있다.

```
Montgomery Algorithm(Y, X, N) {
--Output Z = Y · X · R_1( mod N),
                0 ≤ Z < N,
                R = 2k, Z = 0 ;
    for(i=0 ; i<k ; i++) {
       qi = (Z +X_iY) mod 2 ;
       Z = (Z + X_iY + qiN) / 2 ;
    }
    if (Z > N)    Z = Z - N ; return Z ;
}
```

이 알고리즘은 이진 곱셈을 위해 부분 곱의 누적 덧셈 연산을 수행하고 매 중간 결과에서 qi의 결과에 따라 0, N 또는 -N을 더하여 최하위 비트를 "0"으로 유지함으로써 모듈러 연산을 수행한다. 이 반복 연산 동안 나누기 2 연산을 k번 반복함으로써 모듈러 곱셈의 결과는 0≤ Z < 2N의 범위에 있게 된다. 따라서 경우에 따라 한번의 부가적인 뺄셈 연산이 요구된다. 이러한 연산 방식의 결과는 $Z = Y \cdot X \cdot 2^{-k} \pmod N$의 값을 갖게 되므로 후처리 과정으로 Z에 2^k을 곱함으로써 $Y \cdot X \pmod N$의 결과를 얻을 수 있다. 하지만 이 후처리 과정은 전체 RSA 알고리즘에서 모듈러 지수 연산의 초기 값을 2^{2k}로 설정함으로써 제거 될 수 있다. 그리고 덧셈 연산 이외에 mod 2 와 나누기 2의 연산이 필요하지만 하드웨어 구현에서 각각 두 수의 최하위 비

트에 대한 XOR(Exclusive OR)와 쉬프트(shift)연산으로 간단하게 처리할 수 있다. 따라서 Montgomery 알고리즘은 효율적인 모듈러 곱셈 연산과 하드웨어 구현의 용이성 때문에 RSA 암호시스템에서 가장 많이 사용되고 본 논문에서도 이를 개선하여 사용한다.

Ⅲ. 제안한 RSA 암호시스템의 알고리즘

본 장에서는 가변길이 고속 RSA 암호시스템을 구현하기 위해 필요한 알고리즘 중 본 논문에서 제안하는 알고리즘을 기술한다. 제안한 알고리즘에 기초한 암호화 또는 복호화 진행과정은 그림 1과 같이 모듈러 지수승 연산과 모듈러 곱셈 연산으로 이루어진다.

Ⅲ-1. 고속 모듈러 곱셈 알고리즘

제안한 모듈러 곱셈 알고리즘은 기본적으로 2장에서 설명한 Montgomery 알고리즘과 기수-4 Montgomery 알고리즘[10]을 개선하여 적용한다. 이 수정된 기수-4 모듈러 곱셈(Modified Radix-4 Modular Multiplication, MRMM) 알고리즘은 CSA[13] 구조와 기수-4 Booth 알고리즘[13]을 사용하며 그 내용은 아래의 의사코드와 같다. 이 알고리즘에서 Ys,c는 모듈러 곱셈의 피승수, Xs,c는 승수, 그리고 N은 모듈러스 수를 각각 나타낸다.

```
Modified Radix-4 Modular Multiplication
(Ys,c, Xs,c, N) {
-- bi  : Booth encode value
```

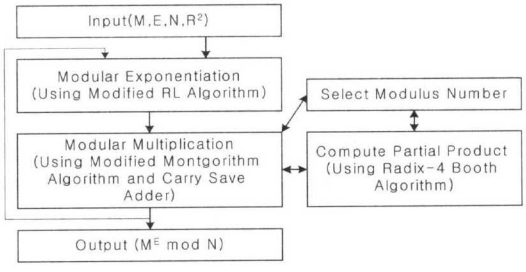


그림 1. 제안한 RSA 암호시스템의 동작순서

```
-- Output Zs,c = Ys,c · Xs,c · R^-1 ( mod N ) =
                 Ys,c · Xs,c · 4^-m ( mod N )
   m = ⌈ (n − l + 3)/2 ⌉ ;
   Zs,c = 0 ;
   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=(nk-1, ..., n1, n0)은 k-비트 홀수이다. 아래 첨자 "s,c"는 CSA 연산에서 발생되는 합과 캐리의 두 값 모두를 갖는 변수이고 "s"는 합(sum) 그리고 "c"는 캐리(carry) 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 횟수이다. 기수-4 Booth 알고리즘은 표 1에서 보이고 있고 이 알고리즘에 의해서 부분 곱의 수는 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 [(n-l+3)/2] 로 감소시켜 전체 암호시스템의 동작속도를 줄이고 가변길이 연산을 할 수 있다. bi는 승수 Xs,c의 i번째 비트 그룹에 대한 Booth 코드로 부분 곱을 계산한다. 그리고 qi는 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.

표 1. 기수-4 Booth 알고리즘

X_{i+1}	X_i	X_{i-1}	Booth Code	b(recode)	연산
0	0	0	00	0(000)	0
0	1	1	01	1(000)	+Y
1	0	0	1'0	-2(110)	-2Y
1	1	1	01'	-1(101)	-Y
0	0	0	01	1(001)	+Y
0	1	1	10	2(010)	+2Y
1	0	0	01'	-1(101)	-Y
1	1	1	00	0(000)	0

Ⅲ-2. 고속 모듈러 지수승 알고리즘

RSA 알고리즘은 실질적으로 모듈러 지수승 연산을 통해 암호화 또는 복호화 과정을 수행한다. 본 논문에서는 아래에 기술한 모듈러 지수 연산 알고리즘을 제안한다. 입력은 평문(M), 공개키(E) 그리고 모듈러스 수(N)이고 최종 출력은 암호문(P)이다. 이 알고리즘은 모듈러 제곱 연산과 모듈러 단순 곱셈을 병렬적으로 처리할 수 있는 RL-이진방식을 기초로 한다. 그리고 이전 절에서 제시한 수정된 기수-4 모듈러 곱셈 함수(MRMM)를 반복 수행함으로써 모듈러 지수승 연산이 수행된다.

```
Modified Modular Exponentiation Algorithm
(M, E , C, N) {
-- Constant : C = R^2(mod N)
                = 4^{2⌈(n-l+3)/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 (e_i=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(Ck-1, ... C1, C0)는 모듈러 곱셈의 결과로 R-1이 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 [(n-l+3)/2 ×(n-l)]이다.

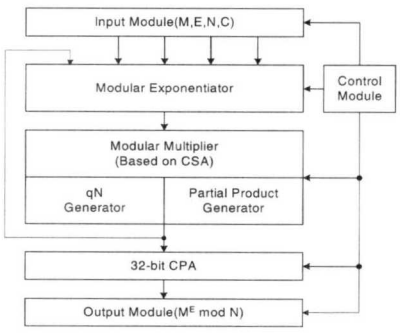


그림 2. 제안한 RSA 암호시스템의 구조

제안한 알고리즘은 $(n-l)$번의 반복 연산 후 최종 결과를 출력하기 위해 합과 캐리를 더한다. 그리고 기수-4 Booth 알고리즘이 적용되므로 P는 음수 값이 생성될 수 있다. 그 경우 다시 N을 더하여 암호문을 출력한다.

Ⅳ. 제안한 RSA 암호시스템의 구조

제안한 1024-비트 RSA 암호시스템의 구조는 그림 2와 같이 여섯 개의 기능블록으로 구성된다. 입력 모듈은 32-비트 단위의 값을 연속적으로 입력받아 레지스터에 저장한다. 32-비트 덧셈 연산 모듈은 CSA 연산에 의해 발생한 합과 캐리 부분을 더하기 위해 CPA를 사용한다. 출력 모듈은 CPA 모듈의 값을 입력받아 32-비트 단위로 최종 출력한다. 제어 모듈은 나머지 모듈들을 제어한다. 이러한 모듈들은 매우 간단한 구조로 설계될 수 있다. 따라서 본 장에서는 암호시스템의 핵심 블록으로 사용되는 모듈러 지수승 연산을 위한 모듈러 지수승 연산기 블록(Modular Exponentiator)과 모듈러 곱셈을 위한 모듈러 곱셈기 블록(Modular Multiplier)에 대한 하드웨어 구조를 기술한다. 여기서 모듈러 곱셈기 블록은 CSA 구조이고, 클럭 주기를 하나의 전가산기 크기로 최소화하기 위해 모듈러스 수 선택(qN Generation) 블록과 기수-4 Booth 알고리즘을 이용한 부분 곱 생성(Partial Product Generation) 블록은 분리된다.

Ⅳ-1. 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 추가하여 총 4개의 CSA구조가 필요하다. 이러한 4 단계 CSA구조를 적용한 모듈러 곱셈기의 하드웨어 구조는 그림 3과 같다. 이 구조는 각 단계마다 레지스터를 삽입하여 파이프라인(pipeline)화함으로써 4개의 모듈러 곱셈을 연속적으로 수행할 수 있다. 이때 요구되는 4개 모듈러 곱셈을 위해서 제안한 RSA 암호시스템은 2개의 메시지 블록을

동시에 입력받는 것으로 가정하였다.

Ⅳ-1-1. 4 단계 CSA 구조

모듈러 곱셈기의 4단계 CSA 구조는 그림 4와 같은데, 여기서는 지면상 4-비트 연산을 위한 구조를 보이고 있으며 n-비트 연산을 위해서는 각 셀을 n개로 확장하면 된다. 이 구조는 n-비트 연산을 위해 3n개의 전가산기((3,2) counter)와 n개의 반가산기((2,2) counter)로 구성된다. 각 단계에서 최상위 두 개의 셀은 모듈러 곱셈에서 중간 출력 결과의 부호 비트를 확장하기 위해 사용된다. 단계1 CSA의 입력은 초기값 또는 모듈러 곱셈의 중간 출력으로 캐리와 합 그리고 부분 곱 Ys이다. 단계2 CSA에서는 이전의 출력과 부분 곱 Yc가 입력되고 단계3 CSA는 이전의 출력 합과 캐리를 다시 한번 더한다. 두 번째 셀은 전가산기로 2ulp(unit in the last position)를 추가 입력으로 받는다. 2ulp는 부분 곱 생성 블록에서 음수가 발생될 경우 2의 보수를 만들기 위해 1을 더하는 의미를 갖는다. 제안한 알고리즘에서 피승수는 합과 캐리 두 부분으로 분리되어 처리되므로 2를 더해야 한다. 따라서 두 번째 셀에서 2ulp가 입력된다. 단계4 CSA는 이전의 출력과 모듈러스 수 선택 블록에 의해 생성된 qN이 입력된다. 따라서 이 과정이 m번 반복되면 모듈러 곱셈의 결과가 출력된다.

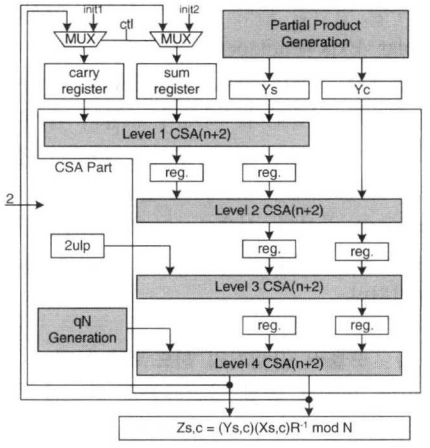


그림 3. 모듈러 곱셈기의 하드웨어구조

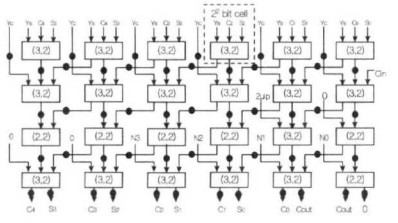


그림 4. 4 단계 CSA의 구조(4-bit)

Ⅳ-1-2. 부분곱 생성 모듈

모듈러 곱셈기의 부분 곱 생성 모듈은 그림 5에 나타내었다. 제안한 암호시스템은 Booth 코드 값을 생성하기 위해 X의 합과 캐리를 2-비트씩 더한다. 이 코드 값은 부분 곱의 값을 효율적으로 선택하기 위해 다시 recode 모듈을 통해 표 1의 bi로 변환된다. 이렇게 생성된 recode[2..0] 값이 Y-select 모듈에 입력되어 0, Y, -Y, 2Y, 또는 -2Y값이 선택된다

제안한 모듈러 곱셈기는 4개의 모듈러 곱셈을 연속적으로 수행하므로 부분 곱은 4클록마다 발생되어야 하고, 매 클록은 전가산기의 지연시간을 가지므로 부분 곱 생성 모듈은 이 조건을 만족하도록 설계되었다. 즉, 2-비트 가산기, recode 모듈, 그리고 Y-select 모듈은 4개의 전가산기 지연시간을 가지므로 4 클록마다 부분 곱의 값을 출력할 수 있다.

Ⅳ-1-3. 모듈러스 수 생성 모듈

모듈러 곱셈기의 모듈러스 수 생성 모듈(qN Generator)은 그림 6과 그림 7로 구성된다. 모듈러 곱셈 알고리즘에서 모듈러스 수를 선택하기 위한 방법은 i번째 연산의 출력 Zs,c와 부분 곱 Ys, c와 부분 곱 Ys,c 그리고 모듈러스 수 N[1]을 더하고 mod 4를 적용하여 qi를 계산한다. 여기서 연산의 지연시간을 최소화하기 위해 덧셈은 그림 6과 같이 CSA를 이용하고, 이 결과는 그림 7와 같은 구조에 의해 모듈러스 수를 선택한다. 이 과정은 3개의 전가산기 지연시간으로 수행될 수 있으므로, 3클록마다 단계4 CSA에 qiN을 입력할 수 있다.

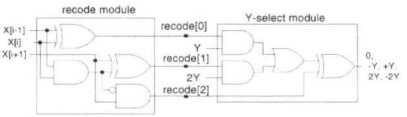


그림 5. 부분 곱 생성 블록

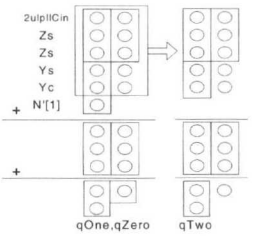


그림 6. qi를 위한 덧셈

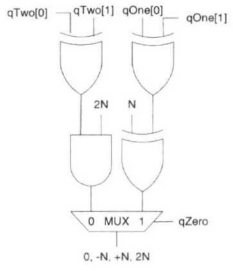


그림 7. 모듈러스 수 선택 모듈

Ⅳ-2. 모듈러 지수승 연산기의 하드웨어 설계

제안한 모듈러 지수승 알고리즘은 RL-알고리즘에 기초하여 모듈러 곱셈을 반복한다. 따라서 모듈러 지수승 연산기의 하드웨어 구조는 그림 8과 같이 두 개의 레지스터와 멀티플렉서 그리고 하나의 제안한 모듈러 곱셈기를 이용해 쉽게 하드웨어로 구현될 수 있다. 그러므로 두 개의 n-비트 평문의 암호화에서 모듈러스 N의 유효비트가 $(n-l)$이라면 $\lceil (n-l+3)/2 \rceil \cdot 4(n-l)$ 클록 후에 2개의 암호문을 연속적으로 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.

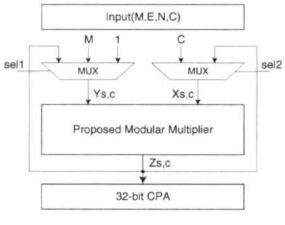


그림 8. 모듈러 지수승 연산기의 하드웨어 구조

Ⅴ. 구현 및 실험결과

3장과 4장에서 설명한 본 논문의 가변길이 고속 RSA 암호시스템은 소프트웨어 및 하드웨어로 구현하여 그 동작과 성능을 실험하였다. 소프트웨어의 구현은 제안한 알고리즘의 검증과 구현된 암호시스템의 동작 검증에 사용할 목적으로 이루어졌으며, 하드웨어는 본 논문에서 제안한 암호시스템의 성능을 측정하고 그 결과를 비교하기 위해 구현하였다.

Ⅴ-1. RSA 암호시스템의 구현

제안한 RSA 암호시스템은 그림 9와 같은 진행 과정을 통해 소프트웨어와 하드웨어로 구현되었다. 소프트웨어 구현은 마이크로소프트사의 C 언어를 사용하였다. 구현된 소프트웨어 암호시스템은 테스트 벡터를 생성하여 하드웨어 구현 과정에서의 시뮬레이션 결과를 검증하기 위해 사용하였다.

하드웨어 구현은 제안한 알고리즘을 기능 단위모듈로 분리하여 게이트 수준으로 설계하였다. 사용된 하드웨어 기술 언어는 VHDL(Very high speed integrated circuit Hardware Description Language)을 사용하였고, 기술된 알고리즘은 Max PLUS+II에서 기능적 시뮬레이션을 수행하고 테스트 벡터와 비교하여 타당성을 검증하였다. 게이트 수준 합성을 위해 Synopsys사의 Design Compiler를 사용하였고 P&R(Place and Route)은 Avant!사의 Apollo를 사용하였다. 그리고 최종 시뮬레이션은 Cadance사의 NC-Verilog를 사용하였고 이 결과 또한 테스트 벡터와 비교하였다. 하드웨어 구현 과정 동안 하이닉스 0.35㎛ 2-poly 4-metal CMOS 공정을 위한 하이닉스 Phantom Cell library를 사용하였다.

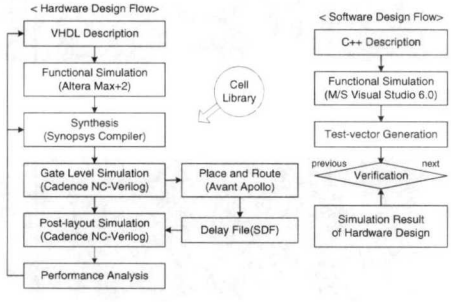


그림 9. 설계 흐름도

Ⅴ-2. 합성 및 시뮬레이션 결과

본 논문에서 제안한 1024-비트 RSA 암호시스템의 합성 결과는 약 190k의 게이트수에 해당하는 크기를 보였다. 하드웨어 사용 재원은 CSA 구조에서 (n+2)-비트 전가산기 3개, 반 가산기 1개 그리고 각 단계마다 레지스터를 사용하여 하드웨어 면적에 있어서는 레지스터 부분만 추가되어 기존에 제안되었던 것과 비슷하다. 그러나 모듈러 곱셈 연산에서 승수는 비트-직렬로, 피승수는 비트-병렬로 입력되어 연산되고 모듈러스 수에 따른 가변길이 연산을 수행하므로 하드웨어 처리속도는 매우 효율적이다. 구현 과정에서 가변길이 연산을 위한 입력 핀을 두어

고정된 1024-비트 연산이 아니라 입력 값에 따른 n-비트 RSA 암호시스템으로 동작할 수 있도록 하였다. 시뮬레이션은 편의상 32-비트 RSA 암호시스템의 결과를 보이며, 그림 10은 입력 부분의 파형을 확대해서 보이고 있다. 입력 신호들을 할당한 후 start 신호에 의해 암호화 또는 복호화가 진행된다. 신호 din은 가변길이 연산을 위해서 모듈러 곱셈 반복횟수와 모듈러 지수승의 반복 횟수를 입력받는다. 그리고 신호 m_n과 r_e는 평문, 모듈러스 수, 키 값 그리고 초기값을 연속적으로 입력받는다. 제안한 암호시스템은 두 개의 평문을 동시에 암호화하는 것으로 가정하였으므로, 각각의 입력 신호는 연속적으로 두 개의 값을 입력받는다. 그림 11은 출력 부분의 파형을 확대해서 보이고 있다. 최종 결과는 약 $2\times(32-0)^2$클록 후에 2개의 32비트 암호문이 연속적으로 sum과 carry 신호를 통해 출력되는 것을 보이고 있다. 여기서 l은 모듈러스 N과 관계없이 0으로 하여 시뮬레이션하였다.

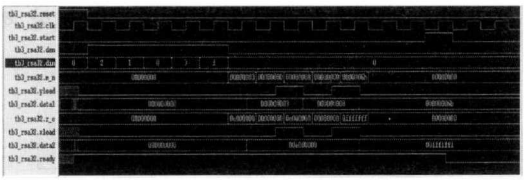


그림 10. 입력 파형

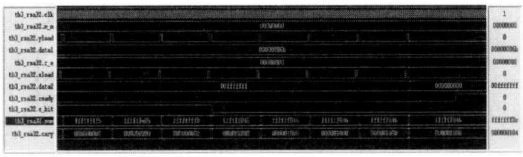


그림 11. 출력 파형

V-3. 기존의 RSA 암호시스템과의 비교

표 2는 기존에 제안되었던 암호시스템과 제안한 RSA 암호시스템을 비교하고 있다. 기존의 암호시스템 구조는 가산기의 적용 종류에 따라 CPA에 기초한 구조와 CSA에 기반한 구조로 분류되는데, 가산기의 특성상 전자는 하드웨어의 면적보다는 수행속도에 중점을 두고 후자는 수행속도보다 면적을 최소화하는데 중점을 두고 구현되어왔다. 표 2에 나타난 바와 같이 본 논문에서 제안한 암호시스템은 CSA에 기초로 하였지만 CPA에 기초한 구조와 비교하여 클럭 주기와 게이트의 수는 유사하다. 하지만 모듈러 곱셈기 반복 횟수는 모듈러스 N의 유효

표 2. 다른 RSA 암호시스템과 비교

Architecture	CPA based			CSA based	
Author	Yang[8]	Guo[9]	Hong[10]	kwon[11]	Our
Year	1988	1999	2000	2001	2002
Gate count	74k	132k	111k	112k	190k
Number of bit	512	512	512	1024	1024
Technology	0.6um	0.6um	-	0.5um	0.35um
Clock(Hz)	125M	143M	150M	50M	150M
Iteration of modular multiplier	$2n^2$	$2n^2$	n^2	n(n+34)	(n+l)
Baud rate (bit/s)	118k	278k	300k	-	min (150k)

비트의 수에 따라 충분히 감소될 수 있다. 즉, 유효 비트 $(n-l)$에서 l이 증가한다면 모듈러 곱셈의 반복의 수는 현저히 감소하므로 데이터 출력률은 상당히 증가한다. 다른 시스템들은 512-비트에서의 출력률을 보이고 있는데, 제안한 시스템에서도 512-비트를 고려한다면 유효 비트 $(512-0)$인 최악의 경우에서 출력률은 약 300kbps(bit per second)로 기존의 암호시스템과 유사하다. 하지만 유효 비트의 수가 400-비트이면 출력률은 380kbps로 상대적으로 상당히 증가한다. 그러므로 기존의 암호시스템보다 동작속도가 평균적으로 약 1.5배 향상되었다.

표 5-2의 데이터 출력률 150kbps는 1024-비트 암호시스템에서 유효 비트 $(1024-0)$인 최악의 경우의 결과이다.

V. 결 론

본 논문에서는 RSA 암호시스템의 동작속도를 증가시키고 가변길이 연산이 가능하도록 하는 알고리즘을 제안하고 이를 하드웨어로 구현하였다. 동작속도를 향상시키기 위해서 모듈러 곱셈기에 기수-4 Booth 알고리즘을 적용하여 부분 곱의 수를 약 n/2로 감소시켰고, 그 과정에서 요구되는 덧셈 연산 부분에 4단계 CSA 구조를 사용하여 캐리 전파를 제거함과 동시에 파이프라인화하여 데이터 출력률을 향상시켰다. 또한 모듈러 곱셈기의 출력인 합과 캐리의 두 값 모두를 모듈러 곱셈기의 입력으로 사용하여 CPA를 제거함으로써 기존의 시스톨릭 어레이 구조에 기초한 RSA 암호시스템과 동일한 모듈러 곱셈 반복횟수를 가지도록 하였다. 그리고 CSA 구조의 특성을 이용해 모듈러스 수에 따른 가변길이 암호화 또는 복호화 연산을 수행할 수 있도록 하여 기존의 방법들보다 데이터 출력률을 향상시켰다.

제안한 1024-비트 가변길이 고속 RSA 암호시스템은 하이닉스 0.35㎛ 2-poly 4-metal CMOS 공정

을 위한 하이닉스 Phantom Cell library를 사용하여 하드웨어로 구현하였다. 암호시스템에서 사용된 게이트 수는 약 190k이었으며, 이 때 동작 주파수는 150MHz이었다. 이 속도는 기존에 CPA를 사용한 방법들의 필적하는 속도이며, 모듈러스 N의 유효 비트수가 감소할 경우에는 기존의 RSA 암호시스템보다 평균적으로 약 1.5배 정도 향상된 데이터 출력률을 보였다. 그리고 CSA를 사용하는 기존방법과 비교하여 약 3배 이상의 동작속도 증가를 얻었다.

따라서 본 논문에서 제안한 RSA 암호시스템은 많은 가입자를 대상으로 하는 대형 서버(server)의 암호화/복호화, 전자 서명 그리고 키 교환 등의 각종 공개키 응용 시스템에서 더욱 유용할 것으로 판단된다. 또한 구현한 RSA 암호시스템은 IP(Intellectual Property) 코어(core)로 면적보다는 동작속도에 중점을 두는 각종 정보보호 시스템의 기능모듈 또는 시스템으로 사용될 수 있을 것으로 기대된다.

참 고 문 헌

[1] 이만영, 김지홍, 류재철, 송유진, 염흥열, 이임영, "전자상거래 보안 기술", *생능 출판사*, 8, 1999.

[2] William Stallings, "Cryptography and Network Security: Principles and Practice", *Prentice-Hall, Inc.*, 2/e , pp.163-203, 1999.

[3] W. Diffie, and M. E. Hellman, "New directions in cryptography", *IEEE Trans. Inform.* Theory, vol. 22, pp.644-654, Nov. 1976.

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

[5] 한국정보보호센터, "RSA 공개키 암호시스템의 현황", 1998.

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

[7] P. S. Chen, S. A. Hwang, and C. W. Wu, "A systolic RSA public key cryptosystem", *in Proc. ISCAS*, 1996, vol. 4, pp.408-411.

[8] C. C. Yang, T. S. Chang, and C. W. Jen, "A new RSA cryptosystem hardware design based on Montgomery's algorithm," *IEEE Trans. Circuits and Systems*, vol. CAS-45, pp.908-913, July 1998.

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

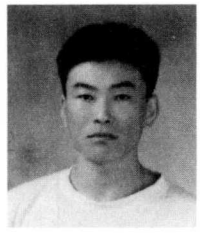
[10] J. H. Hong, C. W. Wu, "Radix-4 modular multiplication and exponentiation algorithms for the RSA public-key cryptosystem," *Design Automation Conference, 2000. Proceedings of the ASP-DAC 2000. Asia and South Pacific*, pp.565 -570, 2000.

[11] 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 age(s): 650 -653.

[12] T. Cormen, C. Leiserson, R. Rivest, "Introduction to Algorithms", *Cambridge, MA: MIT Press*, 1990.

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

박 진 영(Jin-Young Park) 준회원



2000년 8월 : 광운대학교 전자재료공학과 졸업(공학사)
2002년 8월 : 광운대학교 대학원졸업(공학석사)
2002년 9월~현재 : (주)팬택 연구원

<주관심 분야> 암호학, 워터마킹, 영상처리

서 영 호(Young-Ho Seo) 준회원

1999년 2월 : 광운대학교 전자재료공학과 졸업 (공학사).
2001년 2월 : 광운대학교 대학원졸업(공학석사).
2000년 3월~2001년 12월 : 인티스닷컴(주) 연구원.
2001년 3월~현재 : 광운대학교 전자재료공학과 박사

과정.

<주관심 분야> Image Processing/Compression, 워터마킹, 암호학, FPGA/ASIC 설계

e-mail : axl@explore.gwu.ac.kr

김 동 욱(Dong-Wook Kim) 정회원

1983년 2월 : 한양대학교 전자공학과 졸업(공학사).

1985년 2월 : 한양대학교 대학원 졸업(공학석사).

1991년 9월 : Georgia 공과대학 전기공학과 졸업 (공학박사).

1992년 3월~현재 : 광운대학교 전자재료공학과 정교수. 광운대학교 신기술 연구소 연구원.

1997년 12월~현재 : 광운대학교 IDEC 운영위원.

2000년 3월~현재 : 인티스닷컴(주) 연구원.

<주관심 분야> 디지털 VLSI Testability, VLSI CAD, DSP 설계, Wireless Communication

e-mail : dwkim@disy.gwu.ac.kr