通信情報保護學會論文誌
第 10 券 第 1 號, 2000. 3

Interleaved 모듈라 곱셈 기반의 고속 RSA 암호 칩의 설계*

조 현 숙**, 한 승 조***, 이 상 호****

The design of a high speed RSA crypto chip based on interleaved modular multiplication

Hyun-sook Cho**, Seung-jo Han***, Sang-ho Lee****

요 약

공개키 암호 시스템 중에서 가장 널리 사용되는 RSA 암호 시스템은 키의 분배와 관리가 용이하고, 디지털 서명이 가능한 장점이 있으나, 암호화와 복호화 과정에서 512 비트 이상의 큰 수에 대한 멱승과 모듈라 감소 연산이 요구되기 때문에 처리 속도의 지연이 큰 문제가 되므로 모듈라 멱승 연산의 고속 처리가 필수적이다. 따라서 본 논문에서는 몫을 추정하여 중간 곱의 크기를 제한하는 interleaved 모듈라 곱셈 기법을 이용하여 모듈라 멱승 연산을 수행하는 고속 RSA 암호 칩을 VHDL을 이용하여 모델링하고 Faraday FG7000A 라이브러리를 이용하여 합성하고 타이밍 검증하여 단일 칩 IC로 구현하였다. 구현된 암호 칩은 75,000 게이트 수준으로 합성되었으며, 동작 주파수는 50MHz이고 1회의 RSA 연산을 수행하는데 소요되는 전체 클럭 사이클은 0.25M이며 512비트 당 처리 속도는 102.4Kbits/s였다.

ABSTRACT

In RSA crytosystem fast computation of modular exponentiation is essential for the efficient encryption and decryption since it requires the modular exponentiation of large integer prime numbers more than 512 bits. In this paper, we design a high speed RSA crypto chip which computes fast modular exponentiation with the carry save addition and the interleaved modular multiplication scheme which limits partial products by quotient estimation. It is modeled using VHDL by top-down design process based on automatic synthesis methodology. Synthesis and verification is then performed by using SYNOPSYS tools. It took 0.25M clock cycles to finish a 512-bit RSA encryption(decryption) and achieved a baud rate of 102.4Kbits/s at 50MHz.

keyword : RSA, Public-key, interleaved, Barrett, modular exponentiation, hardware, VHDL

Ⅰ. 서 론

키 관리와 사용자 인증을 위하여 제안된 공개 키 암호 방식은 컴퓨터 네트워크 가입자의 암호화 키를 서로 공개하여 한 가입자가 다른 가입자에게 정보를 안전하게 전송하고자 할 때, 공개된 다른 가입자의 암호화 키로 정보를 암호화해서 보내면 다른 가입자는 그 암호문을 받은 후 자신의 비밀 키인 복호화 키로

* 이 논문은 정보통신부 대학기초연구지원사업에 의해서 수행된 연구결과임
** 한국전자통신연구원
*** 조선대학교 전자정보통신공학부
**** 충북대학교 컴퓨터과학과

복호화 하여 평문을 받아 볼 수 있게 된다. Diffie와 Hellman에 의해 공개키 암호 방식이 제안된 이후 Knapsack 알고리듬과 RSA 알고리듬과 같은 많은 공개키 암호 알고리듬이 제안되었다.[1]-[3] 여러 알고리듬 중에서 1978년 Rivest, Shamir와 Adleman에 의해 제안된 RSA 알고리듬이 가장 잘 알려져 있고 널리 사용되는 공개키 암호 시스템이다.[2]

RSA 암호 시스템은 소인수 분해의 어려움을 이용하는 방식으로서 키의 분배와 관리가 용이하고, 디지털 서명이 가능한 장점이 있다.[2] 또한 주된 연산은 모듈라 멱승 연산으로서 모듈라 멱승 연산은 계층적으로 모듈라 곱셈과 모듈라 리덕션 연산으로 세분되고 전체 연산에서 모듈라 곱셈이 대부분을 차지한다.[4] 그러나 RSA 암호 시스템은 암호화와 복호화 과정에서 512 비트 이상의 큰 수에 대한 멱승과 모듈라 감소 연산이 요구되기 때문에 고속 처리하기가 어렵다. 따라서 암호화 및 복호화 때 처리 속도의 지연이 가장 큰 문제가 되므로 모듈라 멱승 연산의 고속화를 위한 하드웨어 구현이 요구된다.

1993년 Walter[5]는 이전의 모듈라 승산에서 모듈러스의 배수가 부분 결과의 상위 자리수에 의하여 결정되므로 빨셈할 모듈러스의 배수가 전달되는 방향과 캐리가 전달되는 방향의 충돌과 캐리의 누적으로 인한 지연을 해결하기 위해 몽고메리[9] 모듈라 승산을 위한 이차원 시스톨릭 어레이를 제안하였고, 1994년 Orup[7]은 병렬 처리와 기수 2^5의 곱셈기를 이용하여 100Kbits/s의 RSA 칩을 설계하였으며, 1995년 Chen[6]은 Walter가 제안한 알고리듬과 연산 시간이 동일하나 커다란 나머지 문제와 추가의 감산이 없는 변형된 몽고메리 알고리듬을 제안하였다.

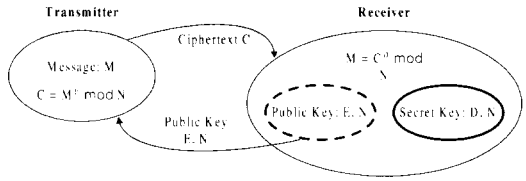
본 논문에서는 스마트 카드와 같은 제한된 환경에 사용하기 위해 필요한 메모리의 용량을 최소화하는 Barrett[11] 알고리듬을 변형하여 사용한다. 또한 암호화 및 복호화를 고속으로 처리하기 위하여 몫을 추정하여 모듈라 감소를 수행하고 중간 곱의 크기를 제한하는 interleaved 모듈라 곱셈 방식을 이용하여 RSA의 주된 계산 방식인 정수의 모듈라 및 멱승 연산을 고속 처리하는 암호 칩을 VHDL을 이용하여 모델링하고 시뮬레이션 및 합성을 하여 단일 칩 IC로 구현하여 성능을 평가 및 분석한다.

본 논문의 구성은 다음과 같다. 2장에서 모듈라 멱승 및 곱셈 알고리즘에 대해 논의하며, 3장에서는 고속 RSA 암호 칩의 하드웨어를 설계한다. 4장에서는 하드웨어 설계 방법론과 Faraday FG7000A 라이브러리를 이용하여 구현된 결과를 분석하며, 제5장에서 결론을 맺고자 한다.

II. 모듈라 멱승 및 곱셈

1. RSA 암호 시스템

RSA 암호 시스템의 구성은 그림 1과 같다.



〔그림 1〕 RSA 암호 시스템
〔Fig. 1〕 RSA cryptosystem

여기서 공개키 (E, N)과 비밀키 (D, N)의 생성과정은 다음과 같다.

① 두 개의 큰 소수 p와 q를 생성한다.
② $N=p\cdot q$와 euler totient 함수 값 $\phi(N)=(p-1)\cdot(q-1)$을 계산한다.
③ $GCD(E, \phi(N))=1$인 정수 E $(1<E<\phi(N))$를 선택한다.
④ 확장된 유클리드 알고리즘을 이용하여 $E\cdot D\equiv 1 \bmod \phi(N)$인 유일한 정수 D $(1<D<\phi(N))$를 계산한다.

또한 RSA 암호 시스템의 암호화 과정과 복호화 과정은 다음과 같다.

- 암호화 : $C\equiv M^E \bmod N$
- 복호화 : $M\equiv C^D \bmod N$

공개키 (E, N)을 이용하여 메시지 M을 암호화하는 과정은 먼저 메시지 M을 $[0; N-1]$ 사이의 정수로 표현하고 암호문 $C\equiv M^E \bmod N$를 계산하며, 암호문 C를 복호화 하는 과정은 비밀키 D를 이용하여 $M\equiv C^D \bmod N$을 계산한다.[1]-[3]

2. 모듈라 멱승

RSA 암호 시스템을 구현하기 위한 가장 핵심적인

부분은 모듈라 멱승 연산으로, 이를 효율적으로 구현하는 것이 RSA 암호 시스템의 속도 향상에 있어 중요한 문제이다. 모듈라 멱승 연산은 모듈라 곱셈의 반복으로서, 전체 연산 시간을 단축시키기 위해서는 모듈라 곱셈의 수행 시간을 단축시키거나, 모듈라 곱셈의 반복 회수를 줄이는 것이 필요하다.[4]-[8]

모듈라 곱셈의 반복 회수를 감소시키기 위하여 본 논문에서는 이진 알고리듬(binary algorithm)[7]을 사용한다. 이진 알고리듬은 지수 E의 이진 표현에 근거하여 반복하여 제곱과 곱셈을 수행한다. $(E_{n-1}, E_{n-2}, E_{n-3}, \cdots\cdots, E_1, E_0)$를 지수 E의 이진 표현이라고 하면 지수 E는 n비트 길이의 수이고 E_{n-1}는 MSB(Most Significant Bit)가 되고 E_0은 LSB(Least Significant Bit)가 되며, 식 (1)은 지수 E의 이진 표현과 모듈라 멱승 연산을 나타낸다.

$$E=\sum_{i=0}^{n-1} E_i \cdot 2^i$$

$$C \equiv M^E \bmod N \equiv \prod_{i=0}^{n-1} M^{E_i \cdot 2^i} \bmod N \qquad (1)$$

이진 알고리듬은 지수의 이진 표현을 왼쪽에서 오른쪽으로 검색하면서, 즉 MSB에서 LSB 쪽으로 검색하면서, 곱셈과 제곱을 수행하게 된다.

이진 알고리듬은 그림 2와 같이 나타낼 수 있다. 여기서 M은 모듈라 멱승 연산에 있어 메시지를 뜻하고 E는 지수, N은 법(modulus)을 가리킨다. 그리고 계산된 값은 C로 반환된다.

모듈라 멱승 연산을 함에 있어 이진 알고리즘을 사용할 경우에는 지수 E의 이진 표현에서 $E_i=1$이 되는 수가 총 곱셈 회수를 결정한다. 즉, 제곱은 지수 E의 비트 수만큼 수행되고 나머지 곱셈 과정은

```
1  C=1
2  for i=n-1 downto 0
3       C=C·C (mod N)
4       if E_i=1 then
5                C=C·M (mod N)
6       endif
7  endfor
8  return C
```

〔그림 2〕 이진 알고리즘
〔Fig. 2〕 Binary algorithm

$E_i=1$인 경우에만 수행된다. 모듈라 곱셈과 제곱을 계산하는 시간이 동일하다고 가정하고 한번의 모듈라 곱셈 연산에 소요되는 시간을 T_n이라고 한다면, $M^E \bmod N$을 계산하기 위한 최악의 경우 시간 복잡도는 $2nT_n$이 되며 평균적인 경우 시간 복잡도는 $\frac{3}{2}nT_n$이 된다.

3. 모듈라 곱셈

모듈라 멱승 계산이 수행되면 대단히 큰 수가 얻어지므로 이 큰 숫자를 저장하는데 소요되는 기억용량과 모듈라 감소 연산에 소요되는 계산 시간을 줄이기 위한 효율적인 모듈라 곱셈 방법이 요구된다. 본 논문에서는 모듈라 곱셈을 빠르게 수행하기 위하여 모듈라 곱셈을 다정도 곱셈과 모듈라 감소 연산으로 나누고 멱승 계산 도중에 생성되는 중간 곱에 대하여 모듈라 감소 연산을 수행한다.

다정도 정수 A의 크기가 $|A|=n \cdot b$이라면 식 (2)와 같이 b 진법으로 표현될 수 있다.

$$A=A_{n-1}b^{n-1}+A_{n-2}b^{n-2}+\ldots+A_1 b+A_0$$
$$=\sum_{i=0}^{n-1} b^i A_i \qquad (2)$$

여기서 $b=2^t$ 이면, 식 (2)는

$$A=\sum_{i=0}^{n-1} 2^{ti} A_i \qquad (3)$$

이 되며, 모듈라 곱셈은 식 (4)과 같이 표현될 수 있다.

$$AB \bmod N \equiv \left(\sum_{i=0}^{n-1} 2^{ti} A_i B\right) \bmod N$$
$$\equiv((\ldots(A_{n-1}B2^t+A_{n-2}B)2^t+\ldots$$
$$+A_1B)2^t+A_0B) \bmod N$$
$$\equiv((\ldots((A_{n-1}B \bmod N)2^t+A_{n-2}B \bmod N)2^t+\ldots$$
$$+A_1B \bmod N)2^t+A_0B) \bmod N \qquad (4)$$

위 식을 순환 형식으로 다시 쓰면 식 (5)와 같다.

$$S_{-1} \equiv 0$$
$$S_i \equiv (S_{i-1}2^t+A_{n-1-i}B) \bmod N \qquad (5)$$

```
1  S_{-1}=0
2  for i=0 to n-1
3        S=S_{i-1}2^t+A_{n-1-i}B
4        μ=2^{2n} div N
5        q̂=((S div 2^{t(n-1)})μ) div 2^{t(n+1)}
6        q̂=((S div 2^{t(n-1)})μ) div 2^{t(n+1)}
7        if S_i<0 then
8              S_i=S_i+2^{t(n+1)}
9        endif
10 endfor
11 if S_{n-1}>N then
12       S=S_{n-1}-N
13 else
14       S=S_{n-1}
15 endif
```

〔그림 3〕 모듈라 곱셈
〔Fig. 3〕 Modular multiplication

식 (5)에서 변수 S_{n-1}는 n 번의 계산 수행 후의 결과가 된다.

여기서 모듈라 감소

$$(S_{i-1}2^t+A_{n-1-i}B)-\lfloor\frac{A\cdot B}{N}\rfloor\cdot N$$
$$=(S_{i-1}2^t+A_{n-1-i}B)-q\cdot N$$

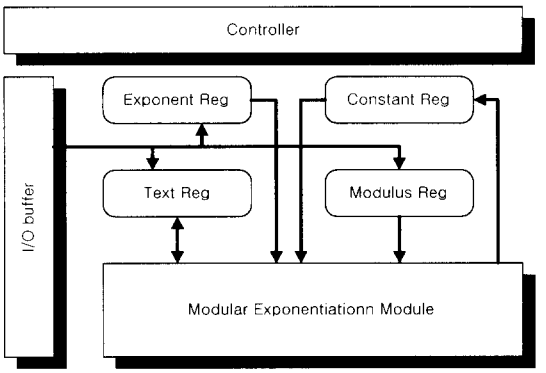
를 수행할 때 나눗셈 몫 q의 정확한 값을 계산하는 대신에 q의 값을 추정하는데, q의 값을 추정하기 위해 본 논문에서는 스마트 카드와 같은 제한된 환경의 디바이스에 사용될 수 있는 알고리듬, 즉 계산에 필요한 메모리의 용량을 최소화하기 위해 식 (6)과 같이 Barrett[11] 알고리듬을 변형하여 이용한다.

$$\hat{q}=\lfloor\frac{S}{N}\rfloor=\lfloor\frac{\lfloor\frac{S}{2^{n+\beta}}\rfloor\lfloor\frac{2^{n+\alpha}}{N}\rfloor}{2^{\alpha-\beta}}\rfloor$$
$$(n=|M,\ \alpha=t+3,\ \beta=-2) \qquad (6)$$

여기서 $\lfloor\frac{2^{n+\alpha}}{N}\rfloor$는 고정된 모듈러스 N에 대해 상수가 되고 미리 계산될 수 있다. 따라서 모듈라 곱셈은 그림 3과 같이 나타낼 수 있다.

Ⅲ. 하드웨어 설계

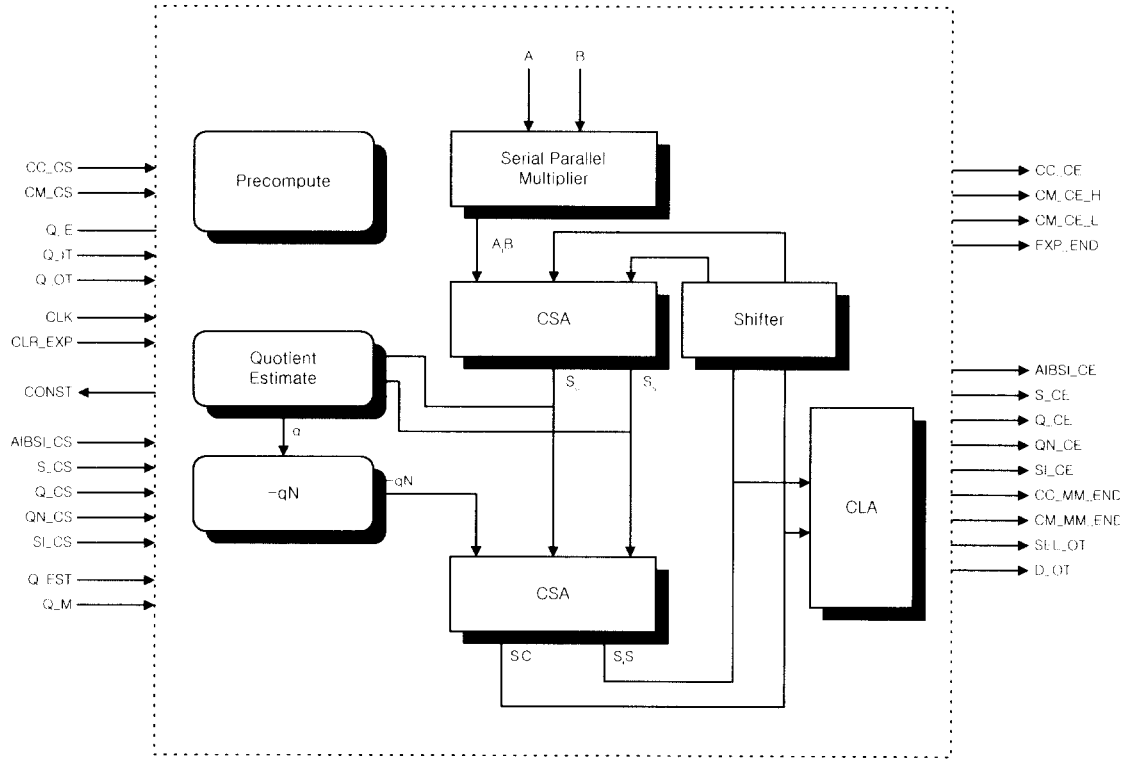
모듈라 멱승 연산을 고속으로 수행하기 위하여 본 논문에서 설계한 암호 칩의 구조는 그림 4와 같으며 전체 시스템을 제어하기 위한 Controller, 외부와 데이터의 입·출력을 위한 I/O buffer, 암호화 및 복호화에 사용되는 키를 저장하기 위한 Exponent Reg, Modulus Reg, 메시지를 저장하기 위한 Text Reg, 전처리에 사용되는 미리 계산된 상수를 저장하기 위한 Constant Reg, 메시지를 암호화 및 복호화 하기 위해 모듈라 멱승 연산을 수행하는 Modular Exponentiation Module로 구성된다.[12]-[13]



〔그림 4〕 암호 칩의 구조
〔Fig. 4〕 Architecture of the crypto chip

내부에서 데이터는 512비트 단위로 처리되고 외부와의 입·출력은 8비트 단위로 이루어지며, 512 비트 RSA 연산(M^E mod N)을 수행하는데 필요한 오퍼랜드를 저장하기 위해 4개의 512 비트 시프트 레지스터를 사용한다. 모듈러스 레지스터는 공개키을 N 저장하며, 상수 레지스터는 알고리듬에서 전처리에 사용되는 미리 계산된 상수 ($\lfloor\frac{2^{n+\alpha}}{N}\rfloor$)를 저장하고, 지수 레지스터는 지수 E를 저장하고 텍스트 레지스터는 메시지 M을 저장한다.

암호 칩의 전체 동작은 다음과 같다. 초기 단계에서 RSA 오퍼랜드가 8 비트 입력 버퍼를 통해 시프트 레지스터에 로딩되고 메시지 M이 텍스트 레지스터에 로딩되는 동안 지수 레지스터는 최상위 비트가 0이 아닐 때까지 시프트하고 지수의 비트 수를 카운트한다. Modular Exponentiation 모듈은 입력된 메시지를 연산하고 연산이 완료되었다는 신호를 Controller에 보낸다. Controller는 연산이 완료되었다는 신호를 Modular Exponentiation 모듈로부터 받으면 외부에 출력이 가능하다는 신호를 보내고 외부의 제어 신호에 따라 출력 버퍼를 통해 암호화 또는 복호화된 데이터를 출력한다.



〔그림 5〕 Modular Exponentiation 블록의 구조
〔Fig. 5〕 Structure of the Modular Exponentiation Unit

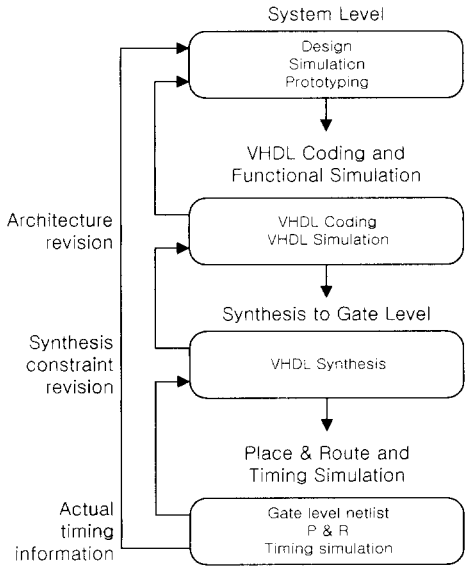
Controller는 각 블록의 동작을 제어하기 위해 제어 신호를 생성한다. Controller는 리셋 신호가 활성화되면 각 블록을 클리어하고 다음 클럭에서 모듈러스 키 값을 입력 받는다는 것을 외부에 알린다. 외부에서 데이터가 입력 버퍼를 통해 입력되면 모듈러스 레지스터에 입력 데이터가 저장되도록 한다. 모듈라 감소 연산의 몫을 추정하는데 미리 계산되어야 하는 값을 처리하기 위해 지수 키를 입력 받는다는 것을 외부에 알린다. 모듈러스 키 값을 입력받는 것과 마찬가지로 지수 키 값을 입력 받아 지수 레지스터에 저장한다. 지수 키 값이 입력되면 암호화 또는 복호화할 데이터를 입력받는다는 것을 외부에 알린다. 입력 텍스트 레지스터에 암·복호화 메시지의 저장이 완료되면 외부에 모듈라 멱승 연산을 수행하고 있다는 것을 알리고 모듈라 멱승 모듈에 신호를 보내어 모듈라 멱승 연산을 시작한다. 모듈라 멱승 연산은 멱승 연산과 모듈라 곱셈 및 감소 연산으로 구성되는데 모듈라 곱셈을 위한 두 값이 결정된 후 모든 연산이 완료되면 신호를 외부에 보내어 연산이 완료되어 결과를 출력할 수 있다는 것을 알리고 외부의 제어 신호에 따라 출력 버퍼를 통해 외부로 8비트씩 출력한다.

512비트 RSA 연산을 수행하는데 필요한 오퍼랜드를 저장하기 위해 사용된 4개의 512비트 시프트 레지스터는 선택 신호에 따라 좌·우측 시프트, 홀딩, 로딩 기능을 수행할 수 있도록 설계하였다.

이진 알고리듬과 몫을 추정하여 중간 곱을 제한하는 interleaved 모듈라 곱셈 방식을 이용한 모듈라 멱승 모듈의 구조는 그림 5와 같다.

모듈라 멱승 블록은 모듈러스 키의 입력이 끝난 후에 몫을 추정할 때 사용되는 미리 계산될 수 있는 값을 구하여 상수 레지스터에 저장하고 지수 키, 입력 텍스트의 저장이 완료된 후에 CC_SS 신호가 활성화되면 이진 알고리듬에 의해서 모듈라 제곱 연산을 수행할 두 값을 결정하고 CC_CE 신호를 Controller에 보낸다. 모듈라 곱셈 블록은 AIBSI_CS 신호가 활성화되면 A의 1자리와 B의 곱셈을 계산하고 AIBSI_CE 신호를 Controller에 보낸다. S_CS 신호가 활성화되면 A와 B의 곱셈 결과에 이전 부분 모듈라 곱셈 결과의 두배를 더하고 S_CE 신호를 Cont-

roller에 보낸다. Q_CS 신호가 활성화되면 상수 레지스터에 저장된 값을 사용하여 몫을 추정하고 Q_CE 신호를 Controller에 보낸다. QN_CS 신호가 활성화되면 추정된 몫과 모듈러스의 2의 보수의 곱셈을 수행하고 QN_CE 신호를 Controller에 보낸다. SI_CS 신호가 활성화되면 중간 곱에 대해 모듈라 감소를 수행하고 SI_CE 신호를 Controller에 보낸다. 이와 같이 중간 곱에 대해 모듈라 곱셈이 종료될 때까지 모듈라 감소를 수행한다. Controller는 모듈라 곱셈이 종료되어 CC_MM_END 신호가 활성화되면 CM_CS 신호를 다시 보내어 Q_E의 이진 값에 따라 모듈라 곱셈을 수행할 A와 B를 결정하도록 알린다. 모듈라 멱승 블록은 A, B의 값을 결정하고 CM_CE 신호를 Controller에 보내어 모듈라 곱셈을 수행하도록 한다. 이와 같은 과정을 전체 모듈라 멱승 연산이 종료될 때까지 계속해서 수행한다.

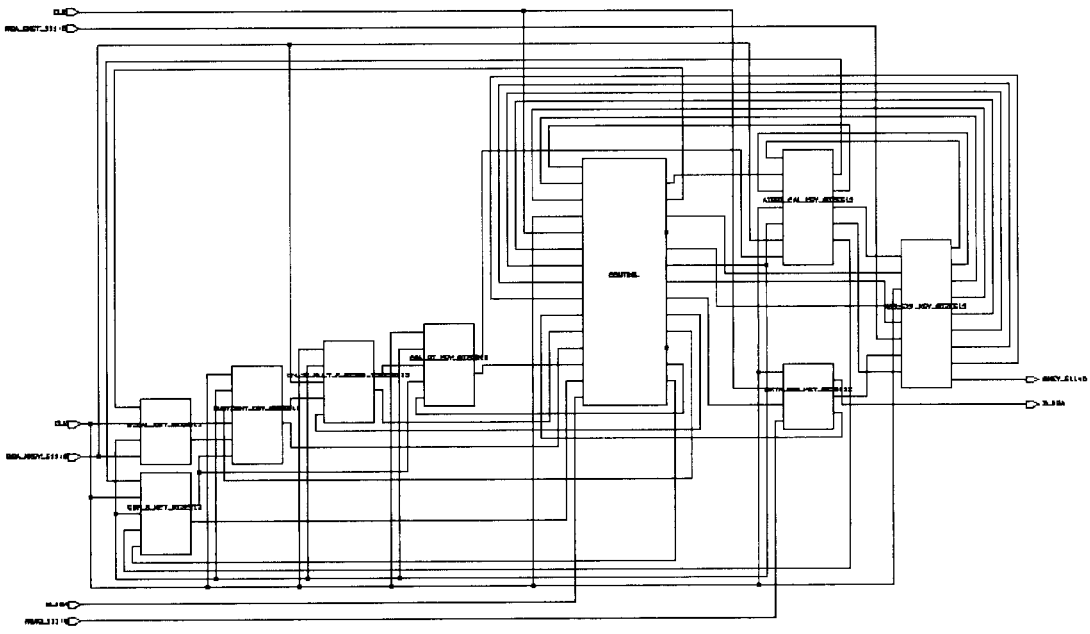


[그림 6] 설계 흐름
[Fig. 6] Design flow

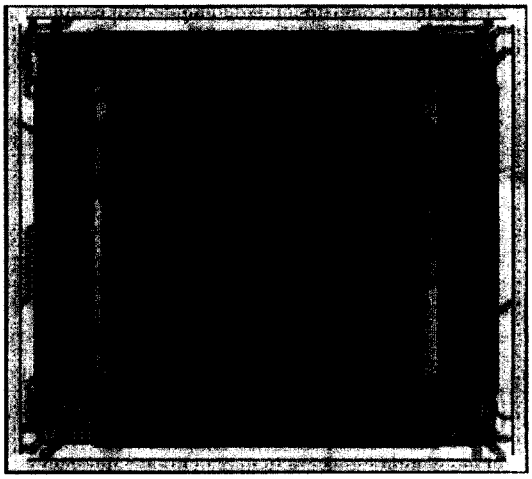
Ⅳ. 설계 분석 및 결과

본 논문에서 사용한 탑다운 방식의 설계 흐름은 그림 6과 같다.

초기 시스템 단계에서 시스템 규격의 만족 여부를 검증하고, 이를 VHDL을 사용하여 구성한 후 Synopsys와 Cadence를 이용하여 함수 레벨의 동작 검증을 하였다. 이후 Faraday FG7000A 라이브러리를 이용하여 Synopsys에서 합성을 하여 게이트 레벨의 타이밍 검증을 수행하였다.[15]



[그림 7] 설계된 암호 칩의 블록 스키메틱
[Fig. 7] Block schematic of the designed crypto chip



〔그림 8〕 RSA 암호 칩의 레이아웃
〔Fig. 8〕 Layout of the RSA crypto chip

설계된 암호 칩의 전체 블록 스키메틱은 그림 7와 같고 레이 아웃은 그림 8과 같다.

구현된 암호 칩은 75,000 게이트 수준으로 합성되었으며, 동작 주파수는 50MHz이고 1회의 RSA 연산을 수행하는데 소요되는 전체 클럭 사이클은 0.26M이며 512비트 당 처리 속도는 102.4Kbits/s였다. 구현된 암호 칩의 칩 면적 등 주요 정보는 표 1과 같으며 다른 RSA 칩과의 비교 결과는 표 2와 같다.

〔표 1〕 주요 칩 정보
〔Table 1〕 Some features of our RSA chip

Technology	CMOS 0.45㎛
Package	80 CQFP
Gate counts (2 input NAND)	75370
Chip size	7998.7㎛×6995.7㎛
Baud rate (512-bit)	102.4Kbits/s, 50Mhz
I/O	8-bit parallel
Control	on-chip
Average number of clocks (512-bit)	0.25M

〔표 2〕 RSA 칩의 비교
〔Table 2〕 A comparison of RSA chips

	Year	Gate counts	Bits per chip	# of clocks 512 bits	Technology	Clock (Hz)	Baud rate (bits/s)
Orup[7]	1994	75K	512	0.125M	1㎛	25M	100K
NTT[8]	1994	105K	1024	1M	0.5㎛ gate array	40M	20K
Chen[8]	1995	77K	512	1.05M	0.8㎛	50M	24.3K
Ours	2000	75K	512	0.26M	0.45㎛	50M	102.4K

V. 결 론

소인수 분해의 어려움을 이용하는 RSA 암호 시스템은 키의 분배와 관리가 용이하고, 디지털 서명이 가능한 장점이 있으나, 암호화와 복호화 과정에서 512 비트 이상의 큰 수에 대한 멱승과 모듈라 감소 연산이 요구되기 때문에 처리 속도의 지연이 큰 문제가 되므로 모듈라 멱승 연산의 고속화를 위해 몫을 추정하여 모듈라 감소를 수행하고 중간 곱의 크기를 제한하는 interleaved 모듈라 곱셈 방식과 carry save 덧셈을 이용하여 RSA의 주된 계산 방식인 정수의 모듈라 및 멱승 연산을 고속 처리하는 암호 칩을 VHDL을 이용하여 모델링하고 Faraday FG7000A 라이브러리를 이용하여 합성하고 타이밍 검증하여 단일 칩 IC로 구현하였다.

구현된 암호 칩은 75,000 게이트 수준으로 합성되었으며, 동작 주파수는 50MHz이고 1회의 RSA 연산을 수행하는데 소요되는 전체 클럭 사이클은 0.26M이며 512비트 당 처리 속도는 102.4Kbits/s였다.

참 고 문 헌

〔1〕 Bruce Schneier, "Applied Cryptography", pp. 466-474, John Wiely & Sons, Inc.

〔2〕 Arto Salomaa, "Public-Key Cryptography", pp. 125-157, Splinger-verlag.

〔3〕 William Stallings, "Network and Internetwork Security Principles and practice," IEEE PRESS, 1995.

〔4〕 N. Takagi and S. Yajima, "Modular multiplication hardware algorithms with a redundant representation and their application to RSA cryptosystem," IEEE transaction on Computers, Vol. 41, No. 7, pp. 887-891, July 1992.

〔5〕 Walter, "Systolic modular multiplication," IEEE Transactions on Computers, vol. 42, pp. 376-378, March 1993.

〔6〕 Chen, C.-W. Wu, "VLSI implementation

for a systolic RSA public key crypto -system," Thesis work of Po-Song Chen, National Tsing Hua University.

[7] Holger Orup, "A 100Kbits/s Single Chip Modular Exponentiation Processor," in HOT Chips VI, Symposium Record, pp. 53-59.

[8] Ishii, S., Ohyama, K., Yamanaka, K., "A single-chip RSA processor implemented in a 0.5㎛ rule gate array," in Proceedings of 7th Annual IEEE International ASIC Conference and Exhibit, pp. 433-436, 1994.

[9] P. L. Montgomery, "Modular multiplication without trial division," Mathematics of Computation, Vol. 42, No. 3, pp. 376-378, March, 1993.

[10] Holger, Orup, "Simplifying quotient determination in high-radix modular multiplication," In Proceedings of the 12th Symposium on Computer Arithmetic, pp. 193-9, July 1995.

[11] P. Barrett, "Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard signal processor," In Advances in Cryptology - CRYTO '86, Santa Barbara, California, vol. 263 of LNCS, pp. 311-323, 1987.

[12] F. Anceau, VLSI-processor architecture and design. In VLSI architecture, pp. 138-148, Prentice Hall International, 1983.

[13] S.Y.Kung, VLSI array processors, Prentice-Hall, 1988.

[14] Z. Navadi, "Using VHDL for model and design of processing unit," Proceeding of the 1992 ASIC Conf. and Ex., pp. 315-326, 1992.

〈著者紹介〉

조 현 숙 (Hyun-sook Cho)
전남대학교 수학과 졸업(학사)
충북대학교 대학원 전자계산학과 졸업(석사)
1982년~현재 한국전자통신연구원 책임연구원 정보보호기술연구본부장
〈관심분야〉Network Security, Conditional Access

한 승 조 (Seung-jo Han)
1980년 조선대학교 전자공학과(학사)
1982년 조선대학교 대학원 전자공학과(석사)
1994년 충북대학교 대학원 전자계산학과(박사)
1986년 6월~1987년 3월 Univ. of New Oleans 객원 교수
1995년 2월~1996년 1월 Univ. of Texas 객원 교수
〈관심분야〉 통신보안시스템설계, S/W불법복제방지시스템, ASIC 설계

이 상 호 (Sang-ho Lee)
1976년 숭실대학교 전자계산학과 졸업(학사)
1981년 숭실대학교 대학원 전자계산학과 졸업(석사)
1989년 숭실대학교 대학원 전자계산학과 졸업(박사)
1981년~현재 충북대학교 컴퓨터과학과 교수
1992년~1993년 캐나다 UBC초청교수
1994년~1998년 충북대학교 전자계산소 소장
〈관심분야〉 프로토콜 공학, 통신보안, 시뮬레이션