논문 99-24-3A-8 한국통신학회논문지 '99-3 Vol.24 No.3A

고속 모듈라 멱승 연산 프로세서의 FPGA 구현

정회원 이 성 순*, 한 승 조**

The FPGA Implementation of A High Speed Modular Exponentiation Processor

Seong-Soon Lee*, Seung-Jo Han** *Regular Members*

요 약

RSA 암호 시스템에서 512비트 이상의 큰 정수 소수의 모듈라 멱승 연산이 필요하기 때문에 효율적인 암호화 및 복호화를 위해서는 모듈라 멱승 연산의 고속 처리가 필수적이다. 따라서 본 논문에서는 carry save 덧셈과 몫을 추정하여 중간 곱의 크기를 제한하는 interleaved 모듈라 곱셈 기법을 이용하여 모듈라 멱승 연산을 빠르게 수행하는 고속 모듈라 멱승 연산 프로세서를 논리 자동 합성 기법을 바탕으로 하는 탑다운 설계 방식으로 VHDL을 이용하여 모델링하고 SYNOPSYS 툴을 이용하여 합성 및 검증한 후 XILINX XC4052XLPG411-3 FPGA에 구현하여 성능을 평가 및 분석한다.

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 512bits. In this paper, we design a high speed modular exponentiation processor 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. Finally, we implement it with the XILINX XC4052XLPG411-3 FPGA, and present the test performance.

I. 서 론

컴퓨터 네트워크를 통한 암호화 및 디지털 서명 등의 서비스를 제공하기 위해서는 암호화와 복호화에 하나의 키를 사용하는 전통적인 비밀 키 암호 방식으로는 키분배, 인증 등의 문제점 때문에 그 이용이 적합하지 못하였다. 따라서 1975년 Diffie와 Hellman에 의해 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 복호화 하여 평문을 받아 볼 수 있게 된다. 공개키 암호 방식의 개념이 제안된 이후 부분합 문제에 기반을 둔 Merkle-Hellman의 Knapsack 암호 시스템이 제안되었으나 대부분의 Knapsack 암호 시스템이 안전하지 않다고 판정되었으며, 그 뒤를 이어 1978년 컴퓨터 시스템을 이용하여 구현될 수 있는 RSA 공개키 암호 시스템이 Rivest, Shamir와 Adleman에 의해 제안되었다([1]-[3]).

RSA 암호 시스템은 암호화와 전자서명 모두를 제공할 수 있고, 512비트 이상의 큰 정수의 소인수 분해가 어렵고 많은 계산을 요구한다는 사실에 안

* 조선대학교 전자공학과 ** 조선대학교 전자·정보통신학부

논문번호 : 98558-1231, 접수일자 : 1998년 12월 31일

※ 본 논문은 정보통신부 대학기초연구지원사업에 의해서 수행된 연구결과임

전도의 근간을 두고 있으며, 암호화 및 복호화 과정에서 모듈라 멱승 연산이 주된 연산으로서 모듈라 멱승 연산은 계층적으로 모듈라 곱셈과 모듈라 리덕션 연산으로 세분되고 전체 연산에서 모듈라 곱셈이 대부분을 차지한다. 또한 RSA 암호 시스템은 매우 큰 정수 소수의 멱승 및 모듈라 연산을 필요로 하고 모듈라 곱셈에서 많은 수의 메시지 블록이 연속적으로 계산되어야 하므로 S/W 구현시에 암호화 및 복호화 때 처리 속도의 지연이 가장 큰 문제가 되므로 모듈라 멱승 연산의 고속화를 위한 하드웨어 구현이 반드시 필요하다.

따라서 본 논문에서는 암호화를 고속으로 처리하기 위한 관점에서 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 프로세서를 VHDL을 이용하여 모델링하고 EDA Tool을 사용하여 시뮬레이션 및 합성을 하여 FPGA에 구현하여 성능을 평가 및 분석한다.

본 논문의 구성은 다음과 같다. 2장에서 interleaved 모듈라 곱셈을 이용한 고속 모듈라 멱승 연산 프로세서의 하드웨어를 설계하고, 3장에서 VHDL을 이용하여 설계된 프로세서의 분석과 FPGA에 구현된 결과를 분석하였으며, 제4장에서 결론을 맺고자 한다.

Ⅱ. 하드웨어 설계

RSA 암호 시스템에서 공개키 (E,N)과 비밀키 (D,N)을 이용하여 암호화 및 복호화하는 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]).

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 연산을 고속으로 수행하기 위하여 본 논문에서 설계한 프로세서의 최상위 레벨 블록도는 그림 1과 같다. 먼저 암호화 또는 복호화에 사용될 키를 입력받아 저장하고 이 키를 사용하여 입력되는 데이터를 암·복호화한다. 내부에서 데이터는 512비트 단위로 처리되며 외부와의 입출력은 8비트 단위로 이루어지는 범용성 프로세서를 설계하였다.

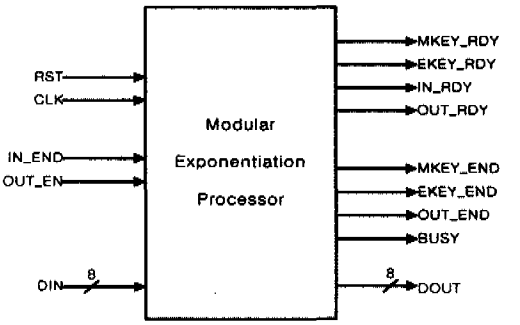


그림 1. 최상위 레벨 블록도
Fig. 1 Top level block schematic

프로세서의 구조는 그림 2와 같이 다음 4개의 블록으로 설계된다.

- 전체 블록을 제어하기 위한 Control 블록
- 외부와 데이터의 입출력을 위한 Data Interface 블록
- 암호화 및 복호화에 사용되는 키와 입·출력 텍스트를 저장하기 위한 Key/Text Shift Register 블록
- 데이터를 암호화 및 복호화 하기 위해 모듈라 멱승 연산을 수행하는 Modular Exponentiation 블록으로 구성된다.

프로세서 설계의 전체 동작은 다음과 같다. 먼저 외부의 제어 신호에 따라 Control 블록이 Data Interface 블록에 입력 데이터를 처리하도록 신호를 보내며, Data Interface 블록은 입력 포트를 통하여 키와 8비트의 입력 데이터를 Key/Text Shift Register의 키 레지스와 입력 레지스터에 각각 저장한다. Modular Exponentiation 블록은 입력된 데이터를 연산하여 Key/Text Shift Register의 출력 레지스터에 저장하고 연산이 완료되었다는 신호를 Control 블록에 보낸다. Control 블록은 연산이 완료되었다는 신호를 Modular Exponentiation 블록으로부터 받으면 외부에 출력이 가능하다는 신호를 보내고 외부의 제어 신호에 따라 Data Interface 블록

에 암호화 또는 복호화된 데이터를 출력하도록 신호를 보내면 Data Interface 블록은 출력 레지스터의 데이터를 출력 포트를 통하여 외부로 출력한다.

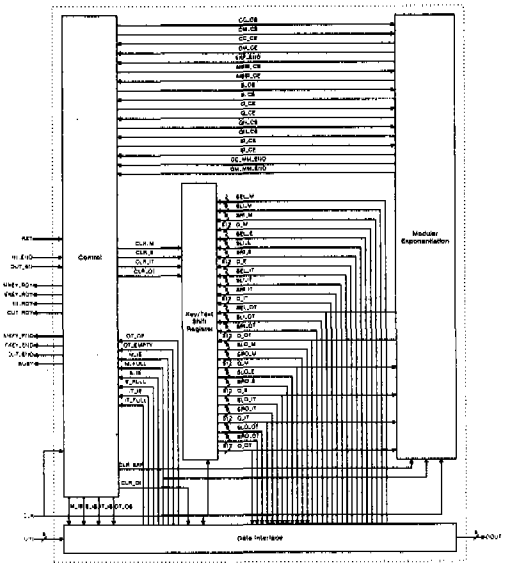


그림 2. 프로세서의 설계
Fig. 2 Design of the processor

2.1 Control 블록의 설계

Control 블록은 각 블록의 동작을 제어하기 위해 제어 신호를 생성한다. Control 블록을 FSM으로 설계하기 위한 상태 천이도는 그림 3과 같다.

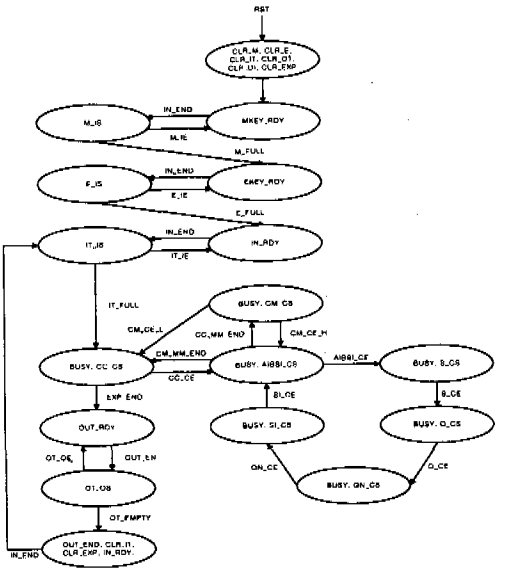


그림 3. Control 블록의 상태 천이도
Fig. 3 State transition diagram of the Control Unit

RST 신호가 활성화되면 각 블록이 클리어되고 다음 클럭에서 모듈러스 키 값을 입력 받는다는 것을 MKEY_RDY 신호를 보내어 외부에 알린다. 외부에서 입력 포트에 데이터가 입력되고 IN_END 신호가 활성화되면 Data Interface 블록에 M_IS 신호를 보내어 모듈러스 레지스터에 입력 데이터가 저장되도록 한다. 모듈러스 레지스터에서 M_IE 신호를 활성화하여 저장이 완료되었다는 신호가 오면 다시 MKEY_RDY 신호를 외부에 보내어 모듈러스 키 값을 입력 받는다는 것을 알린다. 이와 같이 모듈러스 키 값을 M_FULL 신호가 활성화될 때까지 모듈러스 레지스터에 저장한다. M_FULL 신호가 활성화되면 모듈라 감소 연산의 몫을 추정하는데 미리 계산되어야 하는 값을 처리하기 위해 R_CS를 Data Interface 블록에 보내고 지수 키를 입력 받는다는 것을 EKEY_RDY 신호를 활성화하여 외부에 알린다. 모듈러스 키 값을 입력받는 것과 마찬가지로 E_FULL 신호가 활성화될 때까지 지수 키 값을 입력 받아 지수 레지스터에 저장한다. 지수 키 값이 입력되면 IN_RDY 신호를 보내어 암호화 또는 복호화할 데이터를 입력받는다는 것을 외부에 알린다. 입력 텍스트 레지스터에 암·복호화 메시지의 저장이 완료되어 IT_FULL 신호가 활성화되면 BUSY 신호를 외부에 보내어 모듈라 멱승 연산을 수행하고 있다는 것을 알리고 Modular Exponentiation 블록에 CC_CS 신호를 보내어 모듈라 멱승 연산을 시작한다. 모듈라 멱승 연산은 멱승 연산과 모듈라 곱셈 및 감소 연산으로 구성되는데 CC_CE 신호가 활성화되면 모듈라 곱셈을 위한 두 값이 결정된다. 모듈라 곱셈은 중간 곱의 크기를 제한하는 interleaved 모듈라 곱셈 방식을 사용하는데 5단계로 수행된다. 각 단계의 상태를 알리기 위해 ABSI_CS, AIBSI_CE, S_CS, S_CE, Q_CS, Q_CE, QN_CS, QN_CE, SI_CS, SI_CE 신호가 사용된다. 이 5단계가 수행되어 모듈라 곱셈이 끝나면 CC_MM_END 신호가 활성화된다. CC_MM_END 신호가 활성화되면 CM_CS 신호를 Modular Exponentiation 블록에 보내어 다음 모듈라 곱셈을 위한 두 값을 결정한다. CM_CE_H가 활성화되면 모듈라 곱셈이 수행되고, CM_CE_L 신호나 CM_MM_END 신호가 활성화되면 다음 멱승 연산을 수행한다. 이러한 과정을 반복하여 모든 연산이 완료되어 EXP_END 신호가 활성화되면 OUT_RDY 신호를 외부에 보내어 연산이 완료되어 결과를 출력할 수 있다는 것을 알린다. OUT_EN 신호가 활성화 되면 Data Interface

블록에 OT_OS 신호를 보내어 출력 텍스트 레지스터의 값을 출력 포트를 통하여 외부로 8비트씩 출력한다. 출력 텍스트 레지스터의 모든 값이 출력되어 OT_EMPTY 신호가 활성화되면 다음 메시지를 처리하기 위해 IN_RDY 신호를 보내고 입·출력 텍스트 레지스터를 클리어 시킨다.

2.2 Data Interface 블록의 설계

Data Interface 블록은 Control 블록에서 생성된 제어 신호 M_IS와 E_IS가 활성화되면 암·복호화에 사용될 모듈러스 키와 지수 키를 8비트씩 입력 포트 DIN으로부터 입력받아 각각 Modulus Shift Register와 Exponent Shift Register에 저장하고 M_IE 신호와 E_IE 신호를 Control 블록에 보내어 저장되었음을 알린다. 이와 같이 8비트씩 모듈러스 키 값을 입력하여 전체 키 값의 입력이 완료되면 M_FULL 신호를 Control 블록에 보내어 키의 입력이 끝났음을 알리며, Modular Exponentiation 블록에 보내어 몫을 추정할 때 미리 계산되어야 하는 값을 연산하도록 알린다. 또한 전체 지수 키 값의 입력이 완료되면 E_FULL 신호를 Control 블록에 보낸다.

이와 동일하게 IT_IS 신호가 활성화되면 입력 텍스트를 8비트씩 Input Text Shift Register에 저장하고 IT_IE 신호를 Control 블록에 보내어 입력이 완료되었음을 알리며, 전체 입력 텍스트의 입력이 완료되면 IT_FULL 신호를 Contol 블록에 보낸다. 또한 OT_OS 신호가 활성화되면 Output Text Shift Register에 저장되어 있는 연산 결과를 8비트씩 출력 포트 DOUT으로 출력하며, 전체 출력 텍스트의 출력이 완료되면 OT_OE 신호를 Control 블록에 보낸다.

Data Interface 블록의 구조는 그림 4와 같다.

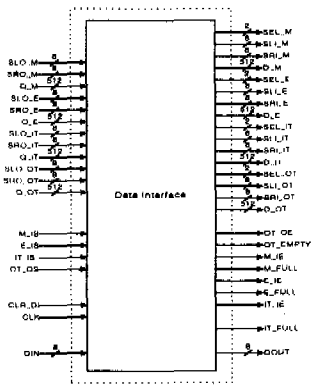


그림 4. Data Interface 블록의 구조
Fig. 4 Structure of the Data Interface Unit

2.3 Key/Text Shift Register 블록의 설계

512비트 RSA 연산($C=M^E \bmod N$, $M\equiv C^D \bmod N$)을 수행하는데 필요한 키 값과 입·출력 텍스트를 저장하기 위해 4개의 512비트 시프트 레지스터를 사용하였다. Modulus Shift Register에 공개키 N이 저장되고, Exponent Shift Register에 지수 E가 저장된다. Input Text Shift Register와 Output Text Shift Register에 암호화할 경우에는 각각 평문 M과 암호문 C가 저장되며, 복호화할 경우에는 각각 암호문 C와 평문 M이 저장된다. 사용된 4개의 시프트 레지스터는 선택신호 SEL_M, SEL_E, SEL_IT, SEL_OT에 따라 좌·우측 시프트, 홀딩, 로딩 기능을 수행할 수 있도록 설계하였다. SLI_M, SLI_E, SLI_IT, SLI_OT는 좌측 시프트일 경우의 8비트 시프트 입력 데이터 값이며 SRI_M, SRI_E, SRI_IT, SRI_OT는 우측 시프트일 경우의 8비트 시프트 입력 데이터 값이 된다. 또한 SLO_M, SLO_E, SLO_IT, SLO_OT는 좌측 시프트일 경우의 8비트 시프트 출력 데이터 값이며 SRO_M, SRO_E, SRO_IT, SRO_OT는 우측 시프트일 경우의 8비트 시프트 출력 데이터 값이 된다. D_M, D_E, D_IT, D_OT와 Q_M, Q_E, Q_IT, Q_OT는 각각 레지스터의 512비트 입·출력 신호다.

Key/Text Shfit Register 블록의 구조는 그림 5와 같다.

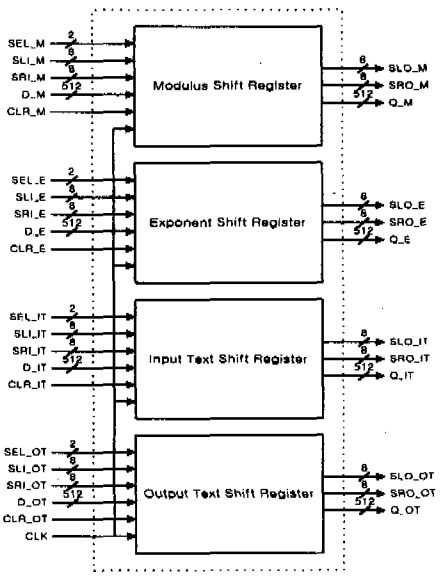


그림 5. Key/Text Shift Register 블록의 구조
Fig. 5 Structure of the Key/Text Shift Register Unit

2.4 Modular Exponentiation 블록의 설계

모듈라 멱승 연산은 모듈라 곱셈의 반복으로서, 전체 연산 시간을 단축시키기 위해서는 모듈라 곱셈의 수행 시간을 단축시키거나, 모듈라 곱셈의 반복 회수를 줄이는 것이 필요하다([4]-[9]). 모듈라 곱셈의 반복 회수를 감소시키기 위하여 본 논문에서는 이진 알고리듬(binary algorithm)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 된다. 이때 지수 E의 이진 표현과 모듈라 멱승 연산은 식 (2.1)과 같이 나타낼 수 있다.

$$E = \sum_{i=0}^{n-1} E_i \cdot 2^i$$
$$C \equiv M^E \bmod N \equiv \prod_{i=0}^{n-1} M^{E_i \cdot 2^i} \bmod N \tag{2.1}$$

여기서 이진 알고리듬은 지수의 이진 표현을 왼쪽에서 오른쪽으로 검색하면서, 즉 MSB에서 LSB 쪽으로 검색하면서 곱셈과 제곱을 수행하게 되며, 지수 E의 이진 표현에서 $E_i=1$이 되는 수가 총 곱셈 회수를 결정한다. 즉, 제곱은 지수 E의 비트 수만큼 수행되고 나머지 곱셈 과정은 $E_i=1$인 경우에만 수행된다.

이진 알고리듬에서 모듈라 멱승 계산이 수행되면 대단히 큰 수가 얻어지므로 이 큰 숫자를 저장하는 데 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 곱셈 계산 도중에 생성되는 중간 곱에 대하여 모듈라 감소 연산을 수행하는 interleaved 모듈라 곱셈 방식을 사용한다.

2^t 진법의 다정도 정수 A는 식 (2.2)와 같이 표현될 수 있다.

$$A = \sum_{i=0}^{n-1} 2^{ti} A_i \tag{2.2}$$

따라서 모듈라 곱셈은 식 (2.3)과 같이 표현될 수 있다.

$$\begin{aligned} AB \bmod N &\equiv \left(\sum_{i=0}^{n-1} 2^{ti} A_i B\right) \bmod N \\ &\equiv ((\ldots(A_{n-1}B2^t + A_{n-2}B)2^t + \ldots \\ &\quad + A_1 B)2^t + A_0 B) \bmod N \\ &\equiv ((\ldots((A_{n-1}B \bmod N)2^t + \\ &\quad A_{n-2}B \bmod N)2^t + \ldots \\ &\quad + A_1 B \bmod N)2^t + A_0 B) \bmod N \end{aligned} \tag{2.3}$$

위 식을 순환 형식으로 다시 쓰면 식 (2.4)와 같다.

$$\begin{aligned} S_{-1} &\equiv 0 \\ S_i &\equiv (S_{i-1}2^t + A_{n-1-i}B) \bmod N \end{aligned} \tag{2.4}$$

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 식 (2.5)의 개선된 Barrett 알고리듬을 이용한다.

$$\hat{q} = \lfloor \frac{S}{N} \rfloor = \lfloor \frac{\lfloor \frac{S}{2^{n+\beta}} \rfloor \lfloor \frac{2^{n+\alpha}}{N} \rfloor}{2^{\alpha-\beta}} \rfloor \quad (n = |M|,\ \alpha = t+3,\ \beta = -2) \tag{2.5}$$

여기서 $\lfloor \frac{2^{n+\alpha}}{N} \rfloor$는 고정된 모듈러스 N에 대해 상수가 되고 미리 계산될 수 있다.

이진 알고리듬과 interleaved 모듈라 곱셈 방식을 이용한 Modular Exponentiation 블록의 구조 설계도는 그림 6와 같다.

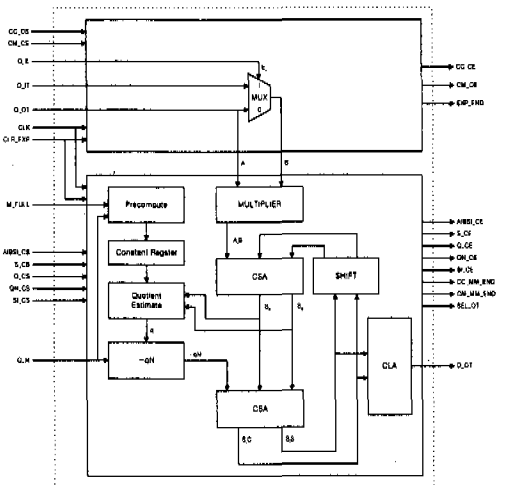


그림 6. Modular Exponentiation 블록의 구조
Fig. 6 Structure of the Modular Exponentiation Unit

Modular Exponentiation 블록은 모듈러스 키의 입력이 끝난 후에 M_FULL 신호가 활성화되면 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 Control 블록에 보낸다. 모듈라 곱셈 블록은 AIBSI_CS 신호가 활성화되면 A의 1자리와 B의 곱셈을 계산하고 AIBSI_CE 신호를 Control 블록에 보낸다. S_CS 신호가 활성화되면 A와 B의 곱셈 결과에 이전 부분 모듈라 곱셈 결과의 두배를 더하고 S_CE 신호를 Control 블록에 보낸다. Q_CS 신호가 활성화되면 상수 레지스터에 저장된 값을 사용하여 몫을 추정하고 Q_CE 신호를 Control 블록에 보낸다. QN_CS 신호가 활성화되면 추정된 몫과 모듈러스의 2의 보수의 곱셈을 수행하고 QN_CE 신호를 Control 블록에 보낸다. SI_CS 신호가 활성화되면 중간 곱에 대해 모듈라 감소를 수행하고 SI_CE 신호를 Control 블록에 보낸다. 이와 같이 중간 곱에 대해 모듈라 곱셈이 종료될 때까지 모듈라 감소를 수행한다. Control 블록은 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. Modular Exponentiation 블록은 A, B의 값을 결정하고 CM_CE 신호를 Control 블록에 보내어 모듈라 곱셈을 수행하도록 한다. 이와 같은 과정을 전체 모듈라 멱승 연산이 종료될 때까지 계속해서 수행하여 최종 결과를 출력 텍스트 레지스터에 저장하고 Control 블록에 EXP_END 신호를 보낸다.

Ⅲ. 설계 분석 및 결과

FPGA를 이용한 구현은 미리 설계된 칩에 프로그래밍을 함으로써 원하는 기능의 회로를 구현하는데 목적이 있으며, IC 설계 제작에 비하여 소요되는 비용과 시간을 절감할 수 있을 뿐만 아니라, 원하는 회로에 대한 신속한 실험 제작(prototyping)이 가능하다는 이득이 있다. XILINX XC4052XL PG411-3 FPGA는 조합 논리 회로와 메모리를 구현할 수 있는 CLB(Configurable Logic Block), 입출력을 위한 IOB(Input/Output Block), 그리고 CLB들과 IOB들을 서로 연결하여 회로를 구현하기 위한 배선 구조로 이루어져 있다.

본 논문에서 사용한 탑다운 설계 방식의 설계 흐름은 그림 2와 같다.

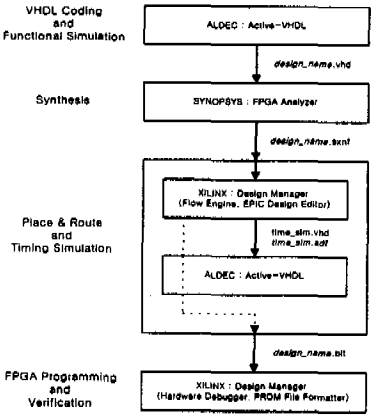


그림 7. 설계 흐름
Fig. 7 Design flow

설계 초기 단계에서 ALDEC사의 Active-VHDL을 사용하여 VHDL 모델을 작성하고 컴파일하였다. 컴파일을 거쳐 생성된 파일을 이용하여 기능 시뮬레이션을 수행하였고, 설계에 부합되는 동작이 검증된 후 SYNOPSYS사의 FPGA Analyzer를 사용하여 합성하였다. 이렇게 합성된 설계를 XILINX사의 배선 및 배치 툴인 Design Manager을 사용하여 배선 및 배선을 수행하였고, 생성된 SDF 파일과 VHDL 네트리스트 파일을 Active-VHDL의 VHDL 시뮬레이터에 입력하고 XILINX VITAL 라이브러리를 이용하여 타이밍 시뮬레이션을 수행하였다. 타이밍 시뮬레이션 후 생성된 비트스트림 파일을 이용하여 XILINX사의 XC4052XLPG411-3 FPGA를 프로그래밍하고 테스트하였다.

설계된 프로세서의 타이밍도는 그림 8과 같고 전체 블록 스키메틱은 그림 9와 같다. 또한 합성 결과는 그림 10과 같다.

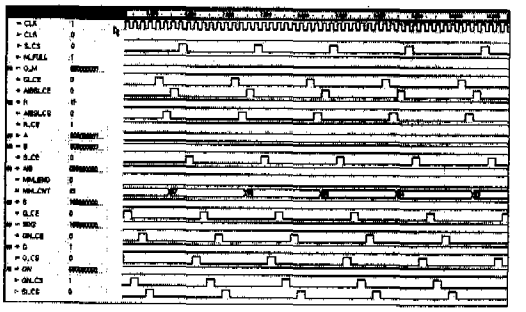


그림 8. 설계된 프로세서의 타이밍도
Fig. 8 Timing chart of the designed processor

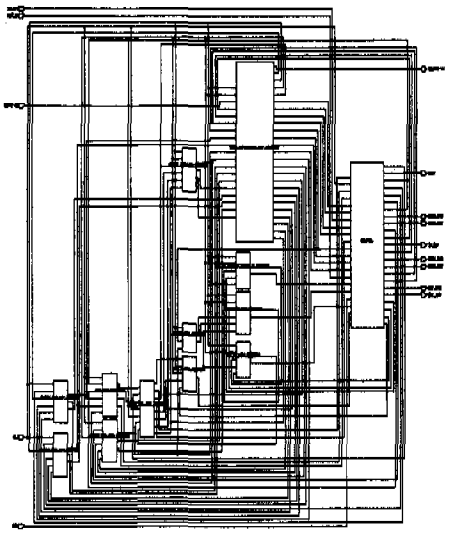


그림 9. 설계된 프로세서의 블록 스키메틱
Fig. 9 Block schematic of the designed processor

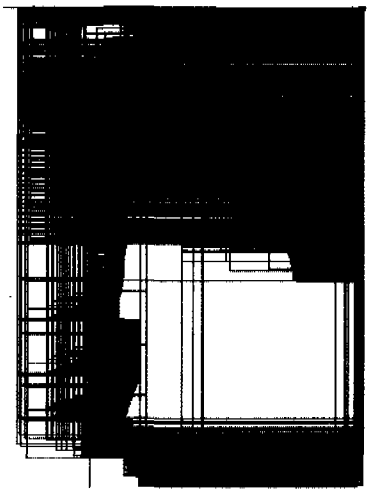


그림 10. 합성 결과
Fig. 10 Synthesized result

XC4052XLPG411-3를 이용한 디바이스 사용도는 표 1과 같이 CLB 16500, IOB 28, FG Function Generator 28572, H Function Generator 6924를 사용하였다.

표 1. XC4052XLPG411-3을 이용한 디바이스 사용도
Table 1 Device utilization using XC4052XLPG411-3

CLBs	16500
Bonded IOBs	28
FG Func. Generators	28572
H Func. Generators	6924

설계된 프로세서의 동작 주파수는 40MHz이고 1회의 RSA 연산을 수행하는데 소요되는 전체 클럭 사이클은 0.55M이므로 512비트 당 처리 속도는 37.2Kbits/s가 된다.

Ⅳ. 결 론

암호화와 전자서명 모두를 제공할 수 있고, 512비트 이상의 큰 정수의 소인수 분해가 어렵고 많은 계산을 요구한다는 사실에 안전도의 근간을 두고 있으며, 암호화 및 복호화 과정에서 모듈라 멱승 연산이 주된 연산인 RSA 암호 시스템은 매우 큰 정수 소수의 멱승 및 모듈라 연산을 필요로 하고 모듈라 곱셈에서 많은 수의 메시지 블록이 연속적으로 계산되어야 하므로 암호화 및 복호화 때 처리 속도의 지연이 가장 큰 문제가 되므로 모듈라 멱승 연산의 고속화를 위한 하드웨어 구현이 필요하다. 따라서 본 논문에서는 몫 추정 모듈라 곱셈 기법과 carry save 덧셈을 이용하여 중간 값의 크기를 제한하는 interleaved 모듈라 멱승 연산을 수행하는 고속 모듈라 멱승 연산 프로세서를 ALDEC사의 Active- VHDL을 사용하여 VHDL 모델을 작성한 후에 컴파일하고 기능 시뮬레이션을 수행하였으며, SYNOPSYS사의 FPGA Analyzer를 사용하여 합성하고 XILINX사의 Design Manager을 사용하여 배선 및 배선을 수행하여 XC4052XLPG411-3 FPGA에 구현하였다. 구현된 프로세서의 동작 주파수는 40MHz로 1회의 RSA 연산을 수행하는데 0.55M의 클럭 사이클이 소요되었고 512비트 당 처리 속도는 37.2Kbits/s였다.

참 고 문 헌

[1] William Stallings, "Network and Internetwork Security Principles and practice," IEEE PRESS, 1995.

[2] Bruce Schneier, "Applied Cryptography", pp 466-474, John Wiely & Sons, Inc.

[3] Arto Salomaa, "Public-Key Cryptography", pp 125-157, Splinger-verlag.

[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] P. L. Montgomery, "Modular multiplication w

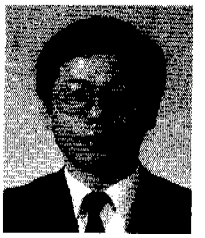
ithout trial division," Mathematics of Computation, vol. 42, no. 3, pp. 376-378, Mar, 1993.
[6] Ernest F. Brickell. A Survey of Hardware Implementation of RSA. In CRYTO '89, 1989.
[7] Holger. Orup, "Simplifying quotient determination in high-radix modular multiplication," in Proceedings of the 12th Symposium on Computer Arithmetic, pp. 193-9, July 1995.
[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. 433-6, 1994.
[9] S.Y.Kung, VLSI array processors. Prentice-Hall, 1988.
[10] Z. Navadi, "Using VHDL for model and design of processing unit," Proceeding of the 1992 ASIC Conf. and Ex., pp.315-326. 1992.
[11] James R. Armstrong & F. Gail, Structured Logic Design with VHDL, Prentice-Hall.
[12] David R. Coelho, The VHDL Handbook, Kluwer Academic Publishers, Norwell, Massachusetts, 1989.
[13] Roger Lipsett, Carl Schaefer, Cary Ussery, VHDL : Hardware Description and Design, Kluwer Academic Publishers, Norwell, Massachusetts, 1989.

이 성 순(Seoung-Soon Lee) 정회원

1995년 : 조선대학교 전자공학과 학사
1997년 : 조선대학교 전자공학과 석사
1997년 2월~현재 : 조선대학교 전자공학과 박사과정

<주관심분야> 통신보안시스템설계, 네트워크보안, 음성합성, ASIC설계

한 승 조(Seung-Jo Han) 정회원



1980년 : 조선대학교 전자공학과 학사
1982년 : 조선대학교 대학원 전자계산학과 박사
1994년 : 충북대학교 대학원 전자계산학과 박사
1997년 : 조선대학교 전자·정보통신공학부 교수
1986년 6월~1987년 3월 : Univ. of New Oleans 객원 교수
1995년 2월~1996년 1월 : Univ. of Tenas 객원교수

<주관심분야> 통신보안시스템설계, 네트워크보안, 음성합성, ASIC설계