JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 22, No. 1 : 100~108 Jan. 2018

CIOS 몽고메리 모듈러 곱셈 알고리즘 기반 Scalable RSA 공개키 암호 프로세서

조욱래 · 신경욱*

Scalable RSA public-key cryptography processor based on CIOS Montgomery modular multiplication Algorithm

Wook-Lae Cho · Kyung-Wook Shin*

School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, Kyungbuk 39177, Korea

요 약

512/1,024/2,048/3,072 비트의 4가지 키 길이를 지원하는 scalable RSA 공개키 암호 프로세서를 설계하였다. RSA 암호의 핵심 연산블록인 모듈러 곱셈기를 CIOS (Coarsely Integrated Operand Scanning) 몽고메리 모듈러 곱셈 알고리듬을 이용하여 32 비트 데이터 패스로 설계하였으며, 모듈러 지수승 연산은 Left-to-Right (L-R) 이진 멱승 알고리듬을 적용하여 구현하였다. 설계된 RSA 암호 프로세서를 Virtex-5 FPGA로 구현하여 하드웨어 동작을 검증하였으며, 512/1,024/2,048/3,072 비트의 키 길이에 대해 각각 456,051/3,496,347/26,011,947/88,112,770 클록 사이클이 소요된다. 0.18 ㎛ CMOS 표준셀 라이브러리를 사용하여 100 MHz 동작 주파수로 합성한 결과, 10,672 GE와 6×3,072 비트의 메모리로 구현되었다. 설계된 RSA 공개키 암호 프로세서는 최대 동작 주파수는 147 MHz로 예측되었으며, 키 길이에 따라 RSA 복호 연산에 3.1/23.8/177/599.4 ms 가 소요되는 것으로 평가되었다.

ABSTRACT

This paper describes a design of scalable RSA public-key cryptography processor supporting four key lengths of 512/1,024/ 2,048/3,072 bits. The modular multiplier that is a core arithmetic block for RSA crypto-system was designed with 32-bit datapath, which is based on the CIOS (Coarsely Integrated Operand Scanning) Montgomery modular multiplication algorithm. The modular exponentiation was implemented by using L-R binary exponentiation algorithm. The scalable RSA crypto- processor was verified by FPGA implementation using Virtex-5 device, and it takes 456,051/3,496347/26,011,947/88,112,770 clock cycles for RSA computation for the key lengths of 512/1,024/2,048/3,072 bits. The RSA crypto-processor synthesized with a 0.18㎛ CMOS cell library occupies 10,672 gate equivalent (GE) and a memory bank of 6×3,072 bits. The estimated maximum clock frequency is 147 MHz, and the RSA decryption takes 3.1/23.8/177/599.4 msec for key lengths of 512/1,024/2,048/3,072 bits.

키워드 : RSA, 공개키 암호, 몽고메리 모듈러 곱셈 알고리듬, 모듈러 곱셈기, CIOS

Key word : RSA, public-key cryptography, Montgomery modular multiplication algorithm, modular multiplier, CIOS

Received 07 September 2017, **Revised** 12 September 2017, **Accepted** 05 January 2018

* **Corresponding Author** Kyung-Wook Shin(E-mail:kwshin@kumoh.ac.kr, Tel:+82-54-478-7427)
School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, Kyungbuk 39177, Korea

Open Access http://doi.org/10.6109/jkiice.2018.22.1.100 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

최근, 사물인터넷 (Internet of Things: IoT) 기술이 빠르게 발전하면서 스마트 에너지, 스마트 교통, 스마트 홈, 스마트 빌딩, 스마트 헬스/의료 등 다양한 분야로 확산되고 있다. IoT 기반 융합 서비스가 활성화됨에 따라 인터넷에 연결되는 장치의 수가 기하급수적으로 증가하게 되고, 보안 공격 대상과 위협 요소도 급격히 증가하고 있다. IoT 디바이스에 저장되고, 전송되는 정보가 유출되거나 변조되는 경우에는 경제적 피해를 넘어 국가와 사회적으로 큰 혼란이 유발될 수 있으며, 피해 정도가 현재와 비교할 수 없을 정도로 엄청날 것으로 예상된다. 따라서 IoT 디바이스나 네트워크에 적절한 수준의 보안 적용이 필수적이며, 정보보안 기술은 IoT 장치나 시스템의 핵심 기술요소로 인식되고 있다[1].

IoT는 다양한 센서와 장치들이 연결되는 이종 복합 시스템 형태를 가지므로, IoT 장치와 서비스의 설계, 개발단계에서부터 정보보안 체계를 고려해야 한다. IoT 정보보안은 기존의 인터넷 보안 시스템과 유사하게 대칭키 암호, 공개키 암호, 해시 암호 등으로 구성되며, 정보의 무결성 (integrity) 및 기밀성 (confidentiality), 기기 간 인증 (authentication), 키 분배 (key distribution) 기술 등이 핵심 요소이다. 공개키 암호는 암호화 키와 복호화 키가 서로 다른 비대칭키 암호이며, 기기 간의 인증 및 키 분배를 목적으로 사용되어 IoT 정보보안 시스템에서 핵심 요소로 사용된다.

대표적인 공개키 암호 시스템으로는 RSA (Rivest, Shamir, Adleman)[2]와 타원곡선 암호 (Elliptic Curve Cryptography: ECC)[3] 등이 있다. RSA는 큰 정수의 인수분해가 어렵다는 점에 안정성을 두며, ECC는 타원곡선 군의 이산대수 문제에 안정성을 둔다. ECC는 RSA 보다 짧은 키 길이로 동일한 안정성을 가져 차세대 공개키 암호로 주목받고 있으며, RSA는 전자서명, 인증 등의 정보보안에 광범위하게 사용되고 있다.

RSA의 암호화와 복호화는 모듈러 멱승 (modular exponentiation) 연산으로 구성되며, 모듈러 멱승 연산은 반복적인 모듈러 곱셈에 의해 계산된다. 1,024 비트 이상의 큰 키 길이에 대한 반복적인 모듈러 곱셈을 직접 하드웨어로 구현하면 회로 복잡도가 매우 커지게 된다. 따라서 모바일 및 IoT 보안을 위한 경량화된 RSA 프로세서를 위해서는 모듈러 곱셈과 모듈러 멱승 알고리듬을 워드 기반 하드웨어로 구현하는 방법을 고려해야 한다. 최근에는 모바일과 IoT 환경에 적합한 경량화된 RSA 프로세서 구현에 관한 연구결과들이 발표되고 있다[4-7].

본 논문에서는 512/1,024/ 2,048/3,072의 4가지 키 길이를 지원하는 scalable RSA 프로세서의 저면적 설계에 관해 기술한다. Ⅱ장에서는 RSA 공개키 암호 알고리듬과 RSA의 핵심 연산 블록인 몽고메리 모듈러 곱셈 알고리듬에 대해 설명한다. Ⅲ장에서 scalable RSA 암호 프로세서 설계에 대해 설명하고, 설계된 프로세서의 기능 검증과 FPGA 검증 결과를 Ⅳ장에서 설명한다.

Ⅱ. RSA 공개키 암호 및 몽고메리 모듈러 곱셈 알고리듬

2.1. RSA 공개키 암호 알고리듬

Rivest, Shamir, Adleman에 의해 제안된 RSA [2]는 공개키 암호 시스템 중 하나이며, 두 개의 큰 소수 (prime number)를 곱한 결과를 인수분해하기 어렵다는 사실에 암호학적 안정성의 기반을 두고 있으며, 국제표준화기구(ISO)를 비롯하여 ITU, ANSI, IEEE 등 여러 국제기구에서 공개키 암호 표준으로 채택되고 있다. RSA 알고리듬의 암호화·복호화에 사용되는 공개키 N과 e, 개인키 d의 생성과정은 다음과 같다.

① 서로 다른 두 소수 p, q를 선택한다.
② $N = p \cdot q$를 구한다.
③ $\Phi(N) = (p-1) \cdot (q-1)$를 구한다.
④ $1 < e < \Phi(N)$인 정수 e를 찾는다. (e와 $\Phi(N)$은 서로소)
⑤ $d \cdot e = 1 \bmod \Phi(N)$을 만족하는 정수 d를 찾는다.

RSA 암호화, 복호화 연산은 각각 식 (1), 식 (2)와 같이 표현되며, 암호화에 사용되는 (e, N)은 공개키이고, 복호화에 사용되는 (d, N)는 개인키이다. 통상, 모듈러 N과 메시지 M은 1,024 비트 이상의 크기를 갖는다.

$$C = M^e \bmod N \quad (1)$$

$$M = C^d \bmod N \quad (2)$$

2.2. 몽고메리 모듈러 곱셈 알고리듬

모듈러 곱셈은 RSA 공개키 암호의 핵심 연산이며, 이를 위한 다양한 모듈러 곱셈 알고리듬들이 연구되었다. 가장 널리 사용되고 있는 알고리듬이 몽고메리 모듈러 곱셈 알고리듬[8] 이다. 몽고메리 모듈러 곱셈 알고리듬은 모듈러 곱셈을 단순 덧셈과 부분 곱셈 및 시프트 연산만으로 구현할 수 있다는 장점이 있다. 두 정수 A와 B에 대한 몽고메리 모듈러 곱셈 알고리듬은 식 (3)과 같이 표현된다. $\gcd(R,N)=1\,(R=2^k)$이고, k는 키 길이를 나타내며, $2^{k-1}<N<2^k$를 만족한다.

$$Z = A \bullet B \bullet R^{-1} \bmod N \tag{3}$$

기본 몽고메리 곱셈 알고리듬을 하드웨어로 구현할 시 성능저하가 예상되며, 이러한 성능 저하를 개선하기 위해 다양한 몽고메리 곱셈기의 구현 사례들이 발표되었다. modified Booth 기반 몽고메리 곱셈[9], modified 몽고메리 곱셈[10,11], high-radix 기반 몽고메리 곱셈[12], Koc [13]가 발표한 5가지 몽고메리 곱셈 알고리듬 등이 있으며, FIPS (FInely Integrated Product Scanning) [4], CIOS (Coarsely Integrated Operand Scanning) [14], 등의 설계 사례가 발표되었다. modified Booth 기반 몽고메리 곱셈은 기본 몽고메리 곱셈 알고리듬에 비해 연산량이 감소되나, k 비트 레지스터가 필요한 점은 동일하기 때문에 많은 하드웨어 자원을 소모하는 단점이 있다. Koc가 발표한 5가지 곱셈 알고리듬 (CIOS, SOS, FIOS, FIPS, CIHS)은 메모리를 이용할 수 있어 적은 하드웨어로 구현이 가능하다는 장점이 있다.

그림 1은 CIOS 몽고메리 모듈러 곱셈 알고리듬의 슈도코드[14]이며, 각각의 k 비트 정수(A, B, N)를 w 비트의 워드 s개로 분할하여 연산을 진행한다. 하나의 외부 반복 루프와 두 개의 내부 반복 루프로 구성된다. 외부 반복 루프는 승수를 워드 단위로 스캔하여 내부 반복 루프에서 사용될 부분곱 (partial product)을 생성한다. 내부의 첫 번째 반복루프는 부분 곱을 가산하며, 두 번째 반복 루프는 리덕션 (reduction) 연산을 수행한다. 식 (3)과 그림 1의 CIOS 몽고메리 모듈러 곱셈은 곱셈 결과에 R^{-1}을 포함하므로, 올바른 연산을 도출하기 위해서는 매 몽고메리 곱셈마다 R을 곱하여 R^{-1}을 제거해야한다. 몽고메리 곱셈마다 R을 곱하는 것은 비효율적이므로, 전처리 과정인 매핑(mapping)과 후처리 과정

Input : $A = \{a_{s-1}, \cdots, a_1, a_0\}_{2^w}$
$B = \{b_{s-1}, \cdots, b_1, b_0\}_{2^w}$
$N = \{n_{s-1}, \cdots, n_1, n_0\}_{2^w}$
$R = 2^k$
$W = 2^w$
$n' = -n_0^{-1} \bmod W$

Output : $Z = \{z_{s-1}, \cdots, z_1, z_0\}_{2^w}$
$= ABR^{-1} \bmod N$

```
1 :   Z = 0; u = 0; C = 0;
2 :   for i = 0 to s − 1 do
3 :     for j = 0 to s − 1 do
4 :       (C, z_j) ← z_j + a_j * b_i + C;
5 :     end for
6 :       (u, z_s) ← z_s + C;
7 :       m ← z_0 * n' mod W;
8 :       C ← z_0 + m * n_0;
9 :     for h = 1 to s − 1 do
10:       (C, z_{h−1}) ← z_h + m * n_h + C;
11:   end for
12:       (C, z_{s−1}) ← z_s + C;
13:       z_s ← u + C; C ← 0;
14:   end for
15:   if Z > N then Z ← Z − N;
```

Fig. 1 Word-based CIOS Montgomery multiplication algorithm

인 역매핑(re-mapping)을 적용하면, 몽고메리 곱셈을 효율적으로 구현할 수 있다[7]. A와 B를 각각 $AR \bmod N$과 $BR \bmod N$으로 매핑하고, 매핑된 두 수를 몽고메리 곱셈하면 $ABR \bmod N$이 얻어진다. 매핑된 두 수에 몽고메리 곱셈을 반복 적용해도 곱셈 결과에 R만 남는 특징이 있다. 모듈러 곱셈과 모듈러 멱승 연산을 모두 수행한 후, 최종적으로 곱셈 결과에 포함된 R을 제거하기 위해 역매핑을 한다. $ABR \bmod N$을 역매핑하면 모듈러 곱셈 결과 $AB \bmod N$이 얻어진다.

Ⅲ. Scalable RSA 프로세서 설계

3.1. 전체 구조

그림 2-(a)는 512/1024/2048/3072 비트의 4가지 키 길이를 지원하도록 설계된 scalable RSA 프로세서의 구조이며, 문헌 [7]과 유사한 구조를 갖는다. RSA 프로세서는 CMM (CIOS Montgomery Multiplier) 블록, 제어 블록, 그리고 6×3,072 비트 메모리 뱅크로 구성된다. 메

모리 KeyM은 3,072 비트의 키 (공개키 또는 개인키) 값을 저장하며, 메모리 Msg는 평문·암호문 메시지 값을 저장한다. CMM 블록은 32 비트 단위로 몽고메리 모듈러 곱셈을 수행하며, 중간 결과 값은 MMz 메모리에 저장된다. 메모리 RSAt는 RSA 암호·복호화 중간결과를 저장하고, 메모리 RsqrN은 매핑인자 $R^2 mod N$을 저장하며, 메모리 modN은 모듈러 상수 N을 저장한다. 2 비트의 key_sel 신호에 의해 RSA 동작모드가 결정되며, key_sel 신호가 '0'일 때 RSA- 512 모드, '1'일 때 RSA-1,024 모드, '2'일 때 RSA- 2,048 모드, '3'일 때 RSA-3,072 모드로 동작한다.

Scalable RSA의 RSA-3,072 모드에 대한 동작 타이밍 도는 그림 2-(b)와 같다. 매핑인자 $R^2 \bmod N$, 모듈러 상수 N, 공개키 및 개인키 그리고 평문·암호문이 각각의 해당 메모리에 32 비트 단위로 96 클록 주기에 걸쳐 순차적으로 입력된다. 평문·암호문의 입력이 완료되면 mm_start 신호에 의해 전처리 매핑 2회, 모듈러 멱승 연산, 역매핑 과정이 순차적으로 수행되어 RSA 암호 또는 복호 연산이 완료된다. 암호·복호 연산이 종료되면, 암호·복호문이 32 비트 단위로 96 클록 주기 동안 out_en 신호와 함께 출력된다.

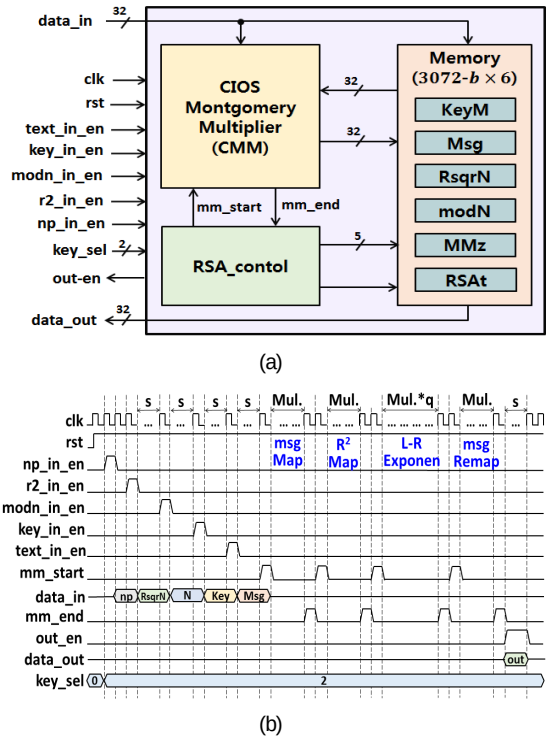


Fig. 2 (a) Overall architecture and (b) timing diagram of the scalable RSA processor

3.2. L-R 이진 모듈러 멱승 연산

모듈러 멱승 연산은 모듈러 곱셈을 지수만큼 반복 연산으로 구현될 수 있다. RSA 암호화·복호화 연산의 지수에 해당하는 키는 1,024, 2,048 또는 3,072 비트로 매우 큰 정수이므로, 모듈러 곱셈을 지수만큼 반복한다면 소요 사이클 수가 매우 커져 성능이 매우 낮아지게 된다. 멱승을 계산하는 대표적인 알고리듬은 이진 방법 (binary method), m-ary method, 슬라이딩 윈도우 방법 (sliding window method) 등이 있으며, 추가적인 레지스터가 필요하지 않은 이진 방법이 많이 사용된다. 이진 방법은 최상위 비트(MSB)부터 지수를 스캔하는 L-R (left-to-right) 방식과 최하위 비트 (LSB)부터 스캔하는 R-L (right-to-left) 방식으로 구분되며, R-L 방식은 L-R 방식에 비해 연산량이 절반인 장점이 있지만, 하드웨어 자원이 두 배로 사용되는 단점이 있다[7]. 본 논문에서는 하드웨어 자원의 최소화를 위해 L-R 방식을 적용하여 모듈러 멱승 연산을 구현하였다.

L-R 이진 모듈러 멱승 연산을 제어하기 위한 제어 블록은 그림 3과 같은 상태 천이도를 갖는 유한상태머신으로 설계되었으며, 문헌 [7]과 유사한 구조를 갖는다. Cons_in 상태에서 모듈러 상수 N, 개인키 또는 공개키 값, 매핑인자 $R^2 \bmod N$ 등을 입력받고, Msg_in 상태에서 평문/암호문을 입력받은 후 Start 상태를 거쳐 Msg_mapping, R2_mapping 상태로 천이된다. 일반적으로, RSA 암호화에 사용되는 공개키 e는 개인키에 비해 작은 비트의 정수가 사용되므로, 공개키 e의 비

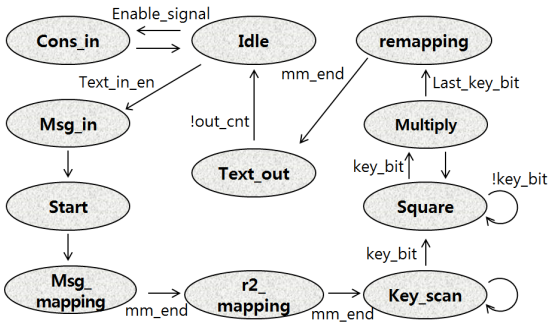


Fig. 3 State transition diagram for computing exponentiation

트 수를 유동적으로 확인할 수 있는 Key_scan 상태를 거친다. Key_scan 상태에서는 메모리 KeyM에 저장된 키 값의 MSB부터 스캔하여 '1'이 처음 나오는 비트 위치를 찾는다. Key_scan을 위해 약 2,000 사이클 정도 소요되지만, 수십 ~ 수백만 사이클이 소요되는 RSA 연산에서 차지하는 비율이 매우 낮으므로, RSA 연산 성능에 미치는 영향이 매우 작다. Key_scan 상태에서 키 길이가 확인되면, Square 상태로 천이되어 L-R 이진 멱승 알고리듬의 연산이 시작된다. L-R 이진 멱승 연산은 메모리 KeyM에 저장된 공개키 또는 개인키의 비트 값에 따라 Square 상태에서 제곱 연산과 Multiply 상태에서 곱셈 연산이 반복 수행된다. 멱승 연산이 완료되면 Remapping 상태로 천이되어 역매핑을 수행하며, Text_out 상태에서 최종 암호문 또는 평문을 출력한다.

3.3. CIOS 몽고메리 모듈러 곱셈기

입력 A, B, N을 받아 모듈러 곱셈을 계산하여 $Z = ABR^{-1} \bmod N$을 출력하는 CIOS 몽고메리 모듈러 곱셈기(CMM)의 내부 구조는 그림 4와 같으며, 32 비트 레지스터 5개와 XOR, MUX 및 Mul_add 블록으로 구성된다. Mul_add 블록은 그림 1의 슈도코드가 나타내는 연산을 수행하며, Mul_add의 출력 64 비트 중, 상위 32 비트는 C 레지스터에 저장되고, 하위 32 비트는 Zs 또는 m 레지스터에 저장된다.

CMM 블록은 그림 5의 상태 천이도에 의해 그림 1의 각 단계에 해당하는 모듈러 곱셈을 수행한다. mm_start

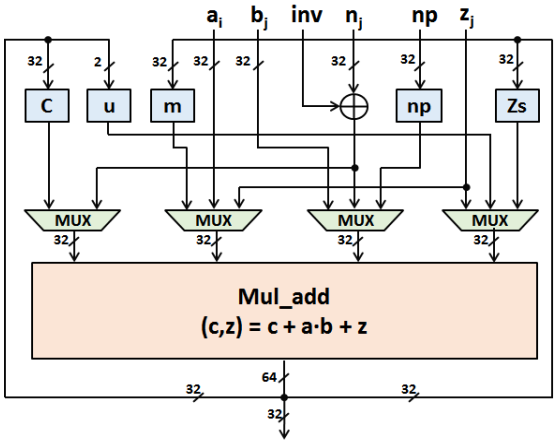


Fig. 4 CIOS Montgomery multiplier(CMM)

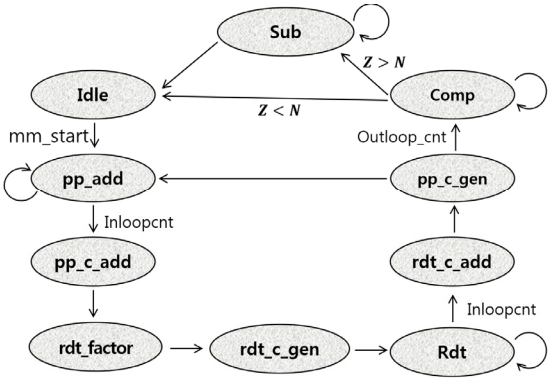
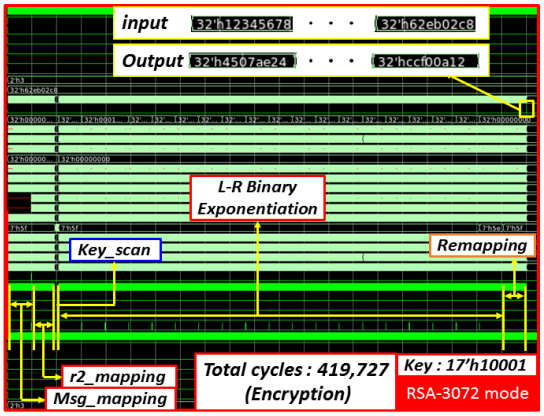


Fig. 5 State transition diagram for CMM

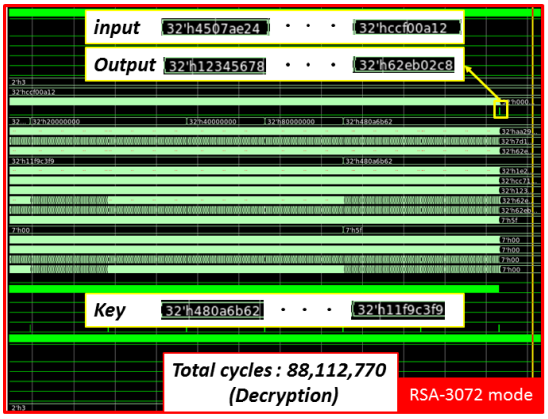
신호에 의해서 곱셈 연산이 시작되며, pp_add 상태에서 승수의 한 워드와 피승수의 곱이 연산되고, pp_c_add 상태는 pp_add 상태에서 마지막에 발생한 캐리 값 C를 이전에 저장된 캐리 값과 더하여 저장한다. rdt_factor에서 리덕션에 사용될 계수 m을 생성하며, rdt_c_gen에서는 리덕션 초기 캐리 값을 생성한다. rdt 상태에서 리덕션 연산을 수행한 후, 리덕션에서 발생한 캐리 값의 저장을 위해 rdt_c_add 상태로 천이되고, pp_add를 연산하기 위한 캐리를 생성하는 pp_c_gen 상태로 천이된다. pp_c_gen 상태에서 Outloop_cnt 값이 s가 되면 곱셈기의 출력 값과 모듈러 N의 값을 비교하기 위해 Comp 상태로 천이한다. 모듈러 곱셈기의 출력 값인 Z와 모듈러 N을 상위 32 비트부터 비교하여 $Z < N$이면 모듈러 곱셈연산을 종료하고 Idle로 천이하며, $Z > N$이면 Sub상태로 천이하여 하위 32 비트부터 Z-N 뺄셈이 시작된다. 만약 Z=N이면, Comp 상태를 유지하고 그 다음 32 비트를 비교한다. 비교 또는 뺄셈 연산을 마치고 Idle 상태로 천이될 때, mm_end 신호가 생성되어 곱셈 연산이 종료됨을 알리며, 다음 mm_start 신호가 입력될 때 까지 Idle 상태를 유지한다.

Ⅳ. 기능검증 및 FPGA 구현

Verilog HDL로 설계된 scalable RSA 프로세서를 RTL 시뮬레이션을 통한 기능 검증과 FPGA 구현으로 하드웨어 동작을 검증하였다. 그림 6은 scalable RSA 프로세서의 키 길이 3,072 비트에 대한 암호화・복호화



(a)



(b)

Fig. 6 RTL simulation results of scalable RSA processor (a) encryption (b) decryption for key length of 3,072-bit

동작의 기능 검증 결과 중 일부를 보인 것이다. 키와 데이터가 입력된 후, Msg_mapping과 r2_mapping 과정이 수행되고, Key_scan을 통해 유효 키 길이를 확인한 후, L-R 이진 멱승 연산이 시작된다. 멱승 연산이 완료되면 역매핑을 거쳐 결과 값이 32 비트 단위로 96 클록 주기에 걸쳐 출력된다. 그림 6-(a)는 암호화 동작에 대한 시뮬레이션 결과이며, 키 값은 "17'h10001"가 사용되었다. 3,072 비트 평문 "3072'h12345678…62eb02c8"이 암호화된 결과로 암호문 "3072'h4507ae24…ccf00a12"가 출력되었다. 그림 6-(b)는 복호화 시뮬레이션 결과이며, 키 값은 모듈러 $\Phi(N)$ 상에서 암호키("17'h10001")의 역원인 "3072'h480a6b62…11f9c3f9"가 사용되었다. 3,072 비트의 암호문 "3072'h4507ae24…ccf00a12"를

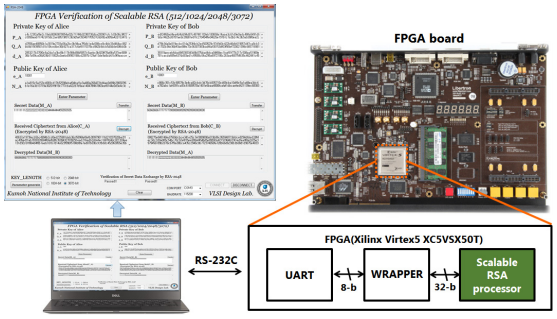


Fig. 7 FPGA verification setup for scalable RSA processor

복호한 결과로 원래의 평문 "3072'h12345678 … 62eb02c8"이 출력되었으며, 따라서 설계된 RSA 프로세서의 기능이 올바로 동작함을 확인하였다. 키 길이 3,072 비트의 RSA 암호화 동작에 419,727 클록 사이클, 복호화 동작에 88,112,770 클록 사이클이 소요된다.

RTL 시뮬레이션을 통해 검증된 scalable RSA 프로세서를 Virtex5 XC5VSX50T FPGA 디바이스에 구현하여 하드웨어 동작을 검증하였다. FPGA 디바이스, UART 인터페이스, C# 기반 구동 소프트웨어로 구성된 FPGA 검증 시스템은 그림 7과 같다.

그림 8은 FPGA에 구현된 scalable RSA 프로세서를 이용하여 3,072 비트 키 길이의 RSA 키 전송 프로토콜의 동작을 구현한 결과이다. Alice는 비밀키로 사용될 랜덤 정수 M_A를 생성하고, Bob의 공개키 (e_B, N_B)를 이용하여 M_A를 암호화한 후 Bob에게 전송한다. Bob은 전송받은 데이터를 Bob의 개인키 d_B를 이용하여 복호화하며, 그 결과로 Alice가 생성한 랜덤 정수 M_A와 동일한 값이 얻어졌다. 따라서 설계된 scalable RSA 프로세서가 올바르게 동작함을 확인할 수 있다.

Scalable RSA 프로세서는 4가지 키 길이를 지원을 하며, 512/1,024/2,048/3,072 비트 키 길이에 대해 RSA 연산에 각각 456,051/3,496,347/26,011,947/88,112,770 클록 사이클이 소요된다. 설계된 scalable RSA 프로세서를 0.18 ㎛ CMOS 표준 셀로 합성한 결과, 100 MHz의 동작 주파수에서 10,672 GE와 6×3,072-b (총 18 kbits)의 메모리로 구현되었다. 최대 동작 주파수 147 MHz에서 키 길이에 대한 복호화 연산에 3.1/23.8/177/599.4 ms가 소요되는 것으로 평가되었다.

표 1은 몽고메리 모듈러 곱셈기의 구현 사례들을 최대 동작주파수, 면적, 데이터 처리율 등의 측면에서 비

Table. 1 Comparison of Montgomery modular multipliers

	[15]	[16]	[17]	Ours [2,048 mode]
key size [bit]	2,048	2,048	2,048	512~3,072
FPGA Technology	Vertex5	Vertex2	Vertex7	Vertex5
Slice LUTs	18,468	10,698	18,636	480
Slice Registers	16,411	N/A	14,551	174
Max. frequency [MHz]	417.2	101	314.5	85.6
Throughput [Mbps]	417.23	95.06	618.1	20.75
APT [Area/Throughput]	83.6	112.6	53.69	31.52

Table. 2 Comparison of RSA cryptography processors

	[4]	[5]	[6]	[7]	Ours
key size [bit]	1,024	2,048	1,024	2,048	512~3,072
Max. frequency [MHz]	30	50	125.03	165	147
Area [GE]	30,000	38,000	107,000	12,540	10,672
Memory [bits]	N/A	N/A	N/A	12 k	18 k
Throughput [kbps]	9.23	18.4	121.27	13.2	11.6 [2,048 bit]
APT	3.25	2.06	0.88	0.95	0.92

교한 결과이다. 문헌 [15]-[17]의 몽고메리 곱셈기들은 단일 키 길이 2,048 비트만 지원하며, 본 논문의 몽고메리 곱셈기는 4가지 키 길이를 지원한다. 표 1에서 APT (area per throughput) 파라미터는 처리율 대비 면적(하드웨어 복잡도)을 나타내며, 작을수록 우수한 설계를 의미한다. 문헌 [15]-[17]의 사례와 비교하여, 본 논문에서 설계한 몽고메리 모듈러 곱셈기는 APT 값이 31.52로 가장 작아 우수하다.

표 2는 RSA 프로세서들의 성능 비교를 보이고 있다. 문헌 [4], [6]의 RSA 프로세서는 1,024 비트의 키 길이를 지원하고, 문헌 [5], [7]의 사례는 2,048 비트의 키 길이를 지원한다. 문헌 [4], [5]의 사례는 APT가 3.25, 2.06으로 커서 APT 성능이 나쁘며, 문헌 [6]의 사례는 APT가 0.88로 우수하지만, 본 논문의 설계에 비해 약 10배의 게이트를 사용하므로, 경량화 구현이 필요한 분야에는 적합하지 않다. 본 논문의 scalable RSA 프로세서는 4가지 키 길이를 지원하면서도 APT가 0.92로 작아 저면적 구현이 필요한 응용분야에 적합하다.

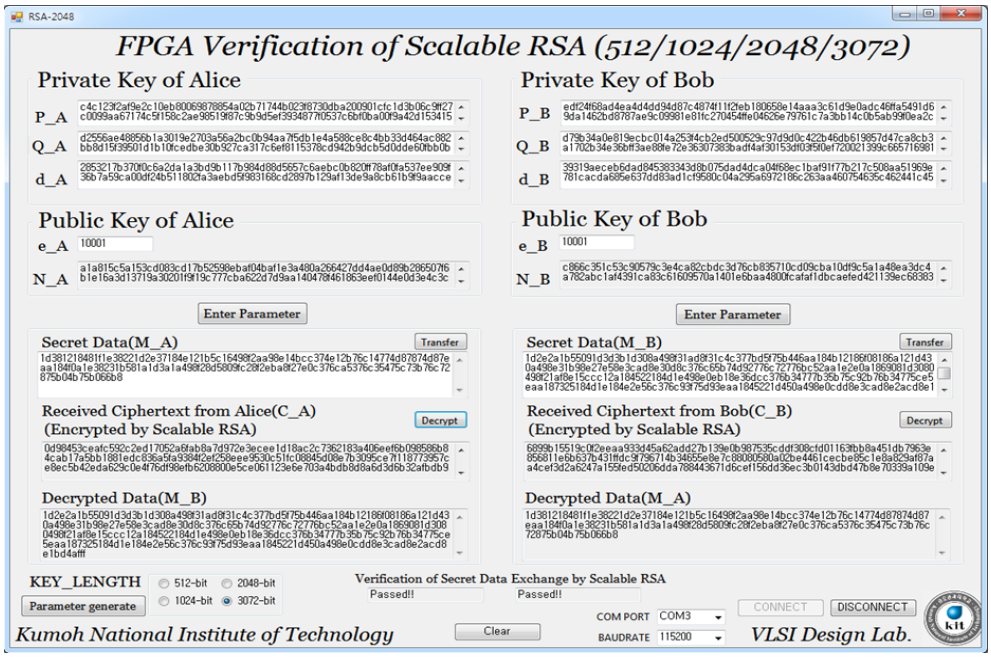


Fig. 8 FPGA verification results of scalable RSA processor for key transmission protocol

Ⅴ. 결 론

인증, 키교환 등의 정보보안을 위해 폭넓게 사용되고 있는 RSA 공개키 암호 알고리듬을 저면적 하드웨어로 구현하고, FPGA 구현을 통해 하드웨어 동작을 검증했다. 설계된 scalable RSA 공개키 암호 프로세서는 512/1,024/2,048/3,072 비트의 4가지 키 길이를 지원하며, RSA 암호의 핵심 연산인 모듈러 곱셈기를 32비트 워드 기반의 CIOS 몽고메리 모듈러 곱셈 알고리듬을 적용하여 저면적으로 설계하였다. 설계된 scalable RSA 프로세서를 100 MHz의 동작 주파수로 합성한 결과, 10,672 GE와 18 kbits의 메모리가 사용되었으며, 최대 동작 주파수는 147 MHz로 예측되었다. 본 논문에서 설계된 scalable RSA 암호 프로세서는 4가지 키 길이 지원과 우수한 APT 성능을 가지므로, 하드웨어 자원이 제한된 모바일 및 IoT 디바이스의 공개키 기반 보안 하드웨어 구현에 활용이 가능하다. 향후, 전력 모니터링 분석, 소요시간 분석 등의 부채널 공격에 강인한 RSA 하드웨어 설계에 관한 추가적인 연구가 필요하다.

ACKNOWLEDGEMENTS

- This paper was supported by Kumoh National Institute of Technology
- Authors are thankful to IDEC for supporting EDA softwares

REFERENCES

[1] Korea Internet & Security Agency (KISA). IoT Common Security Principle v1.0 [Internet]. Available: http://www.kisa.or.kr/public/laws/laws3_View.jsp?mode=view&p_No=259&b_No=259&d_No=67&ST=T&SV=/.

[2] R. Rivest, A. Shamir, 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.

[3] NIST Std. FIPS PUB 186-2: *Digital Signature Standard (DSS)*, NIST, Jan. 2000.

[4] J. Shao, L. Wu, and X. Zhang, “Design and implimentation of RSA for dual interface bank IC card,” *2013 IEEE 10th International Conference on ASIC (ASICON)*, Shenzhen, pp. 1-4, 2013.

[5] M. S. Kim, Y. S. Kim, and H. S. Cho, “Design of Cryptographic Hardware Architecture for Mobile Computing,” *Journal of Information Processing Systems*, vol. 5, no. 4, pp. 187-196, Dec. 2009.

[6] X. Zheng, Z. Liu, and B. Peng, “Design and Implementation of Ultra low power RSA coprocessor,” *in proceeding of the 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM’08)*, Dalian, pp. 1-5, 2008.

[7] W. L. Cho, and K. W. Shin, “2,048 bits RSA public-key cryptography processor based on 32-bit Montgomery modular multiplier,” *Journal of the Korea Institute of Information and Communication Engineering (KIICE)*, vol. 21, no. 8, pp. 1471-1479, Aug. 2017.

[8] P. L. Montgomery, “Modular multiplication without trial division,” *Mathematics of computation*, vol. 44, no. 170, pp. 519-521, Apr. 1985.

[9] A. Kauther, S. Sami, and A. Ahmed, “Enhancement of hardware modular multiplier radix-4 algorithm for fast RSA cryptosystem,” *International Conference on Computing, Electrical and Electronic Engineering (ICCEEE)*, pp. 692-696, Khartoum, 2013.

[10] A. Nadjia, and A. Mohamed, “High throughput parallel montgomery modular exponentiation on FPGA,” *in Proceeding of the 9th International Symposium on Design and Test*, Algiers, pp. 225-230, 2014.

[11] B. Hanindhito, N. Ahmadi, H. Hogantara, A. I. Arrahmah, and T. Adiono, “FPGA implementation of modified serial montgomery modular multiplication for 2048-bit RSA cryptosystems,” *2015 International Seminar on Intelligent Technology and its Applications (ISITIA)*, Surabaya, pp. 113-118, 2015.

[12] A. Rezai, and P. keshavarzi, “High-throughput modular multiplication and exponentiation algorithms using multibit-scan-multibit-shift technique,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 9, pp. 1710-1719, Sep. 2015.

[13] C. K. koc, T. Acar, and B. S. Kaliski, “Analyzing and comparing Montgomery multiplication algorithms,” *IEEE Micro*, vol. 16, no. 3, pp. 26-33, Jun. 1996.

[14] S. Tamura, C. Yamada, and S. Ichikawa, “Implementation and Evaluation of modular multiplication based on Coarsely Integrated Operand Scanning,” *IEEE 2012 Third*

International Conference on Networking and Computing (ICNC), Hangzhou, pp. 334-335, 2012.

[15] R. Verma, M. Duttam, and R. Vig, "FPGA Implementation of Modified Montgomery for RSA Cryptosystem," *International Journal of Computer Science and Telecommunications*, Vol. 4, no. 1, pp. 42-46, Jan. 2013.

[16] M. Huang, K. Gai, and T. EI-Ghazawi, "New Hardware Architectures for Montgomery Modular Multiplication Algorithm," *IEEE Transactions on computers*, vol. 60, no. 7, pp. 923-936, Jul. 2011.

[17] S. Erdem, T. Yanık, and A. Çelebi, "A General Digit-Serial Architecture for Montgomery Modular Multiplication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 5, pp. 1658-1668, May 2017.

조욱래(Wook-Lae Cho)

2016년 2월 금오공과대학교 전자공학부(공학사)
2016년 3월~현재 금오공과대학교 대학원 전자공학과 석사과정 재학 중
2018년 1월~현재 픽셀플러스(주) 연구원
※관심분야 : 통신 및 신호처리용 반도체 IP 설계, 정보보호용 반도체 IP 설계

신경욱(Kyung-Wook Shin)

1984년 2월 한국항공대학교 전자공학과(공학사)
1986년 2월 연세대학교대학원 전자공학과(공학석사)
1990년 8월 연세대학교대학원(공학박사)
1990년 9월~1991년 6월 한국전자통신연구소 반도체연구단(선임연구원)
1991년 7월~현재 금오공과대학교 전자공학부(교수)
1995년 8월~1996년 7월 University of Illinois at Urbana-Champaign(방문교수)
2003년 1월~2004년 1월 University of California at San Diego(방문교수)
2013년 2월~2014년 2월 Georgia Institute of Technology(방문교수)
※관심분야 : 통신 및 신호처리용 SoC 설계, 정보보호 SoC 설계, 반도체 IP 설계