J Korea Industr Inf Syst Res Volume 18 Number 2
http://dx.doi.org/10.9723/jksiis.2013.18.2.001

정보보호 시스템을 위한 FPGA 기반 하드웨어 가속기 설계†

(Design of FPGA Hardware Accelerator for Information Security System)

차 정 우*, 김 창 훈*
(Jeong Woo Cha and Chang Hoon Kim)

요 약 정보보호 시스템은 소프트웨어, 하드웨어, FPGA(Field Programmable Array) 디바이스를 이용하여 구현되었다. S/W의 구현은 다양한 정보보호 알고리즘에 대해 높은 유연성을 제공하나 속도, 전력, 안전성 측면에서 매우 취약하며, ASIC 구현은 속도, 전력 측면에서는 매우 우수하지만 구현의 특성상 다양한 보안 플랫폼을 지원할 수 없다. 이러한 문제점들의 상충관계를 개선하기 위해 최근 FPGA 디바이스 상에서의 구현이 많이 이루어 졌다. 본 논문에서는 다양한 환경에서의 정보보호 서비스를 제공하기위한 정보보호 시스템을 위한 FPGA 기반 하드웨어 가속기를 설계한다. 개발한 정보보호 시스템은 비밀키 암호알고리즘(AES : Advanced Encryption Standard), 암호학적 해쉬(SHA-256 : Secure Hash Algorithm-256), 공개키 암호알고리즘(ECC : Elliptic Curve Cryptography)을 수행할 수 있으며, Integrated Interface에 의해 제어된다. 또한 기존의 시스템에 비해 다양한 정보보호 알고리즘을 지원하여 활용도를 높였으며, 파라미터에 따라 상충관계를 개선 할 수 있기 때문에 저 비용 응용뿐만 아니라 고속의 통신장비에도 적용이 가능하다.

핵심주제어 : 정보보호, FPGA, AES, SHA-256, ECC

Abstract Information Security System is implemented in software, hardware and FPGA device. Implementation of S/W provides high flexibility about various information security algorithm, but it has very vulnerable aspect of speed, power, safety, and performing ASIC is really excellent aspect of speed and power but don't support various security platform because of feature's realization. To improve conflict of these problems, implementation of recent FPGA device is really performed. The goal of this thesis is to design and develop a FPGA hardware accelerator for information security system. It performs as AES, SHA-256 and ECC and is controlled by the Integrated Interface. Furthermore, since the proposed Security Information System can satisfy various requirements and some constraints, it can be applied to numerous information security applications from low-cost applications and high-speed communication systems.

Key Words : Security Information, FPGA, AES, SHA-256, ECC

† 본 연구는 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임 (2010-0006324)
* 대구대학교 대학원 컴퓨터정보공학과
주저자 : 차정우
교신저자 : 김창훈

1. 서 론

최근 정보보호는 매우 중요한 문제로 인식되어, 현재 국·내외에서 다양한 연구를 진행 중에 있으며, 연구 분야는 크게 보안 정책, 보안 프로토콜, 인증 기법, 키 관리 기법, 암호 및 해쉬 알고리즘 설계, 암호 알고리즘의 효율적인 소프트웨어 및 하드웨어 구현으로 구분 할 수 있다[1].

정보보호를 위한 핵심적인 부분은 암호 시스템이다. 즉, 노드들의 인증(Authentication), 데이터의 기밀성(Confidentiality), 데이터 무결성(Integrity), 비밀 키 교환 등 다양한 보안 서비스를 위해선 비밀키[2,3,4,5] 및 공개키[6,7,8,9] 암호 알고리즘, 해쉬함수, 난수 생성기의 구현이 요구된다.

그러나 기존의 정보보호 시스템은 하나의 정보보호 알고리즘을 설계하여 특정한 응용에만 활용이 가능하도록 되어있다. 현재의 정보보호의 목적이 정보보호 시스템에 의해 처리되는 정보의 무결성, 가용성, 기밀성을 확보함으로써 안전성과 신뢰를 확보할 수 있다. 제안한 시스템에서는 데이터의 기밀성을 제공하기 위한 블록암호 모듈, 메시지 무결성과 메시지 인증을 위한 SHA-256 모듈과 키교환을 위한 ECC모듈이다.

본 논문에서는 정보보호 시스템을 위한 FPGA 기반 하드웨어 가속기를 개발한다. 개발한 정보보호 시스템은 32bit 기반 저면적 AES(Advanced Encryption Standard), 32 bit 기반 저면적 SHA(Secure Hash Algorithm)-256, ECC(Elliptic Curve Cryptographic)이다. 메인 CPU와의 통신 및 다양한 정보보호 알고리즘을 제어하고 통신을 하기위한 Integrated Interface 및 메모리를 설계한다.

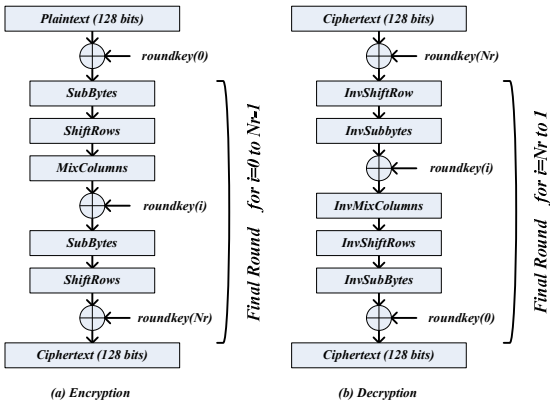
기존의 논문[12,13,14,15,16,17,18]은 각각의 정보보호 알고리즘을 사용하여 구현이 되어서 특정 응용에만 활용이 가능하지만 본 논문에서는 설계한 하드웨어 가속기는 비밀키, 해쉬, 공개키 암호 알고리즘을 모두 구현함으로써 다양한 정보보호 서비스 응용에 활용이 가능하다.

본 논문의 구성은 다음과 같다. 2절에서는 정보보호 알고리즘을 사용한 저면적 정보보호 모듈을 설계하고 3절에서 각각의 정보보호 모듈을 제어하고 통신하기 위한 Integrated Interface를 설계한다. 4절에서는 본 논문에서 제안한 정보보호 시스템의 FPGA 구현결과 및 성능을 분석한 후 5절에서 결론을 맺는다.

2. 정보보호 모듈 설계

2.1 32bit 기반 저면적 AES 설계

AES 알고리즘은 송신자와 수신자 모두 암호화와 복호화하기 위해 동일한 키를 사용하는 대칭 블록 암호이다[3]. 아래 그림 1은 AES의 암호화와 복호화 순서를 나타낸다.



<그림 1> AES 암호화 및 복호화

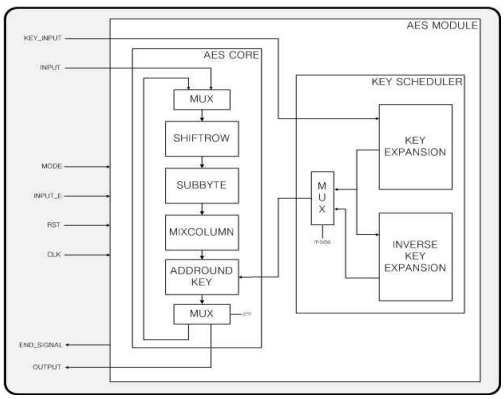
AES는 128, 196 또는 256bit 세 개의 키 길이 선택을 허용하지만 블록 길이는 128bit로 제안되어 있다. 본 논문에서는 128bit 키 사용을 다루도록 한다.

AES의 전반적인 구조는 다음과 같다. ① Rijndael 알고리즘은 Feistel 구조를 사용하지 않고 각 라운드에서 순열과 치환을 수행하는 동안 전체 데이터 블록이 병렬처리 된다. ② 입력으로 주어지는 128비트의 키는 44개의 32비트 워드, w[*i*]로 확장되고 4개의 서로 다른 워드(128비트)는 각 라운드에서 라운드 키로 사용된다. ③ 1회의 순열과 3회의 치환으로 구성된 4단계(Substitute bytes, Shift row, Mix columns, Add round key)를 사용한다. ④ 암호화와 복호화 모두 라운드 키 추가 단계로 시작되고 4단계 모두를 포함하는 9회의 라운드가 이어지고, 3단계(열 혼합 제외)만을 포함하는 10번째 라운드가 이어진다. ⑤ 실제적으로 라운드 키 단계는 그 자체로는 강력하지 않으며 다른 3개의 단계와 연대하여 혼란, 확산 및 비선형성을 제공하지만, 키를 사용하지 않기 때문에 보안성을 제공하지는 못한다. AES 암호는 블록의 변형된 *XOR* 연산(라운드 키 추가), 블록의 혼합(바이트 치환, 행

이동, 열 혼합), *XOR* 연산 등의 순서로 진행되어 효율적이고 안전성이 높다. ⑥ 복호화의 경우 바이트 치환, 행 이동, 열 혼합 단계들은 역함수를 이용하며 라운드 키 추가 단계는 $A \oplus B \oplus B = A$ 를 이용한다. ⑦ 대부분의 블록 암호 알고리즘에서 복호화 알고리즘은 확장키의 역순을 사용하여 이루어지지만 복호화 알고리즘과 암호화 알고리즘은 동일하지 않으며 둘 다 마지막 라운드는 3단계만으로 구성된다는 것이 AES의 특수한 구조이다[5,14].

기존의 AES 모듈들을 보면 128bit의 Data를 한 번에 처리 하도록 되어있어서 메모리 낭비 및 하드웨어 면적이 넓은 문제점을 가지고 있지만 본 논문에서는 32bit씩 4번 수행하도록 되어있어서 메모리 및 하드웨어의 낭비를 감소시킨다. 그러나 수행 횟수가 늘어나기 때문에 처리속도는 기존의 방식에 비해 증가한다[14,15].

아래 그림 2는 저면적 AES의 전체 구조를 나타낸다.



<그림 2> 저면적 AES 전체 구조

AES의 SubByte에 대한 S-box table 정보는 내부 ROM에 저장하도록 설계되어있다. 내부 ROM에 S-box table 정보를 저장함으로써 S-box의 값을 구하기 위한 연산을 수행하지 않아도 되는 이점이 있으며 연산을 하지 않아도 되기 때문에 향상된 높은 속도를 보여준다.

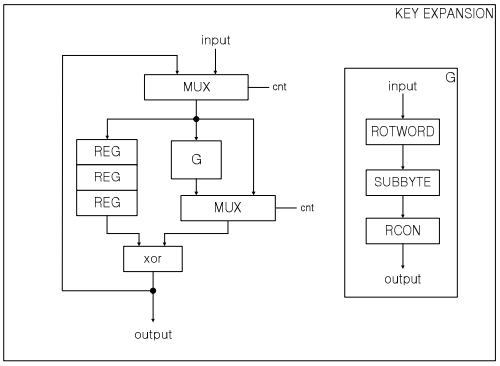
ShiftRow 연산에서는 128 bit의 데이터를 시프트 연산으로 처리한다. 그러나 본 논문의 AES 모듈은 32bit씩 연산이 이루어지기 때문에 ShiftRow연산을 하기 위해서는 128 bit 데이터를 저장해야 한다. 그래서 Shift-Register를 이용하여 한 클럭마다 32bit씩 데이터를 출력하도록 모듈을 구성한다.

본 논문에서의 Mixcolumn은 내부적으로 암호화에 사용되는 Forward Mixcolumn과 복호화에 사용되는 Inverse Mixcolumn으로 구성된다. 32bit 단위의 데이터 처리를 하기위해 각 모듈은 32bit 데이터를 입력으로 취하고 32bit Mixcolumn 값을 출력한다. 입력된 32bit 데이터는 8bit 씩 4개의 원소로 분리된 원소들은 4개의 32bit 레지스터에 8bit 단위로 쉬프트 되어 저장되고 이후 비트단위 쉬프트 연산과 *XOR* 연산을 통해 새로운 32bit 워드로 출력된다.

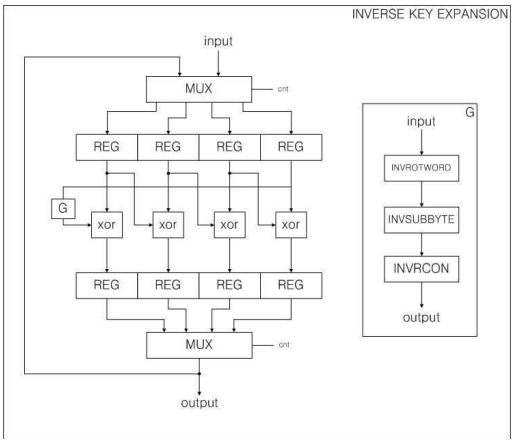
$GF(2^8)$에서의 덧셈은 비트단위 *XOR* 연산으로 간단하게 구해지며 곱셈은 좌측 쉬프트 연산으로 수행한다. 비트단위 쉬프트 연산에서 최상위 비트가 1이면 x^4, x^3, x, 1에 해당하는 비트에 최상위 비트를 더함으로써 곱셈 연산을 간단하게 구현하고 병렬적으로 32비트 데이터를 처리함으로써 처리 속도를 향상시킨다. 본 논문에서의 Key Scheduler 모듈은 Key Expansion과 Inverse Key Expansion으로 구성된다. 32bit 기반 AES Core에 적합하도록 Key Scheduler 모듈도 32bit 단위로 키 값을 입력받고 32bit 키 값을 출력한다.

기존의 Key Scheduler 방식은 크게 두 가지로 나누어 볼 수 있다. 하나는 각 라운드 별로 사용되는 key 값을 AES 수행과 동시에 수행함으로써 key 생성을 위한 추가적인 수행시간을 줄여보고자 하는 on-the fly 방식이고 나머지는 key를 사전에 미리 계산하여 저장해놓은 뒤에 AES 수행 시에 이 값을 가져가는 방식이다. 미리 key를 계산하는 방식은 계산한 key값을 메모리에 저장하기 때문에 메모리 측면에서 비효율적이다. on-the-fly 방식은 encryption 연산 시에는 차례대로 key를 생성하면 되기 때문에 추가적인 수행시간을 줄일 수 있다. 그러나 기존의 저면적 AES의 경우 decryption 연산 시에는 key가 가장 마지막 연산 값부터 사용되기 때문에 key생성을 먼저 해 놓아야한다는 단점이 있지만 본 논문에서 제안한 Inverse Key Expansion의 경우 encryption 연산 시 수행되는 on-the-fly 방식과 동일하게 수행이 된다.

Key Scheduler 회로의 구조를 살펴보면 그림 3, 그림 4와 같다.



<그림 3> Key Expansion

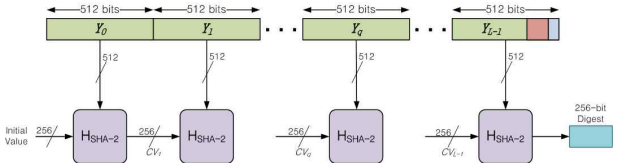


<그림 4> Inverse Key Expansion

2.2 32bit 기반 저면적 SHA-256 설계

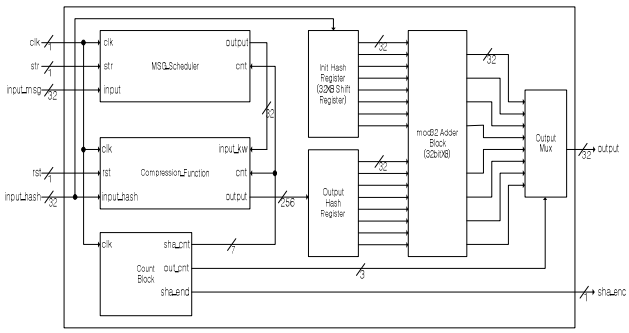
SHA(Secure Hash Algorithm)는 미 연방정부의 디지털 서명 표준인 DSA(Digital Signature Algorithm)를 위해서 NIST(National Institute of Standards and Technology)에서 개발되었다. MD5와 유사한 구조로 설계되었으나 보다 안전한 것으로 인정되고 있다. 256비트 길이의 출력을 내는 SHA-256은 대부분의 인터넷 응용이나 국제/업계 표준들에서 기본 해쉬 함수로 사용되고 있다[16].

SHA-256은 2^{64}비트 미만의 최대 길이를 갖는 메시지를 입력으로 취하고, 256비트 메시지 다이제스트를 출력으로 생성하는 암호학적 해쉬 알고리즘으로서 전체적인 처리는 아래의 그림 5와 같다[17].



<그림 5> SHA-256 전체 구조

아래 그림 6는 본 논문에서 설계한 SHA-256의 전체적인 구조를 나타낸다.



<그림 6> SHA-256 하드웨어 구성도

본 논문에서 설계한 32bit 기반 저면적 SHA-256은 앞에서 설명한 32bit 기반 저면적 AES와 동일하게 32bit씩 데이터가 처리되도록 연산한다.

설계된 SHA-256은 크게 2개의 블록 Message Scheduler와 Compression Function 블록과 이들을 포함하는 IO Interface 블록으로 구성된다.

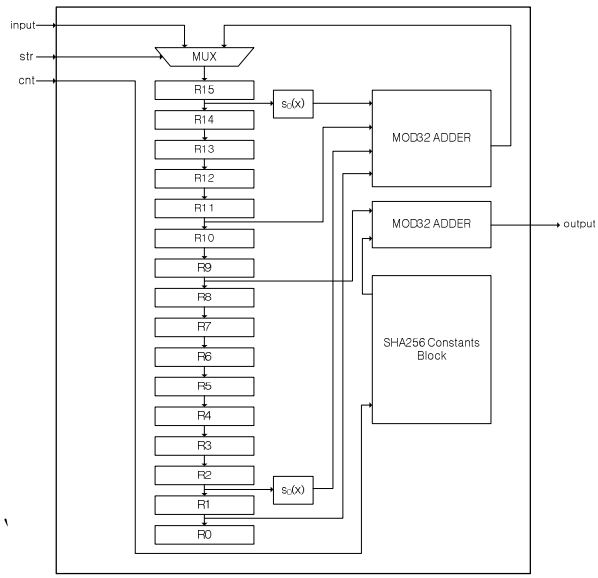
IO Interface는 SHA-256 연산에 필요한 512 비트의 메시지 블록, 한 블록의 메시지 처리를 위해 입력되는 초기 해쉬 값, 시작을 알리는 str 신호와 리셋을 알리는 rst 신호를 입력으로 받아 Message Scheduler 블록과 Compression Function 블록에 전달하는 역할을 한다. 또한, 연산이 완료된 256 비트의 해쉬 값을 매 clock 마다 32비트 씩 8번 출력하는 역할도 한다.

SHA-256의 처리 과정에서의 데이터는 32비트를 기본단위로 연산이 이루어지며 하나의 Count Block을 사용하여 전체적인 데이터 및 연산의 동기를 맞추도록 설계한다.

모든 연산을 마친 결과는 최초 입력으로 사용되었던 초기 해쉬 값에 더해져 최종 결과로 출력된다. SHA-256에서 사용되는 덧셈은 mod 32에서 이루어지는 덧셈이므로 초기 값과 연산결과로 얻은 해쉬 값을

mod32 Adder 블록에 의해 각 32 비트씩 덧셈을 하여 Output Mux에 연결된다.

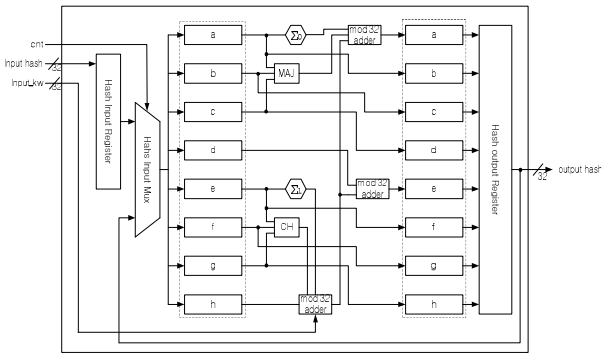
Message Scheduler는 512 비트 메시지 블록을 입력으로 64개의 32비트 워드를 출력한다. 0에서 15 clock 동안에 주어지는 입력은 블록의 쉬프트 레지스터에 연결되어 순차적으로 값이 채워지게 되며 이후 채



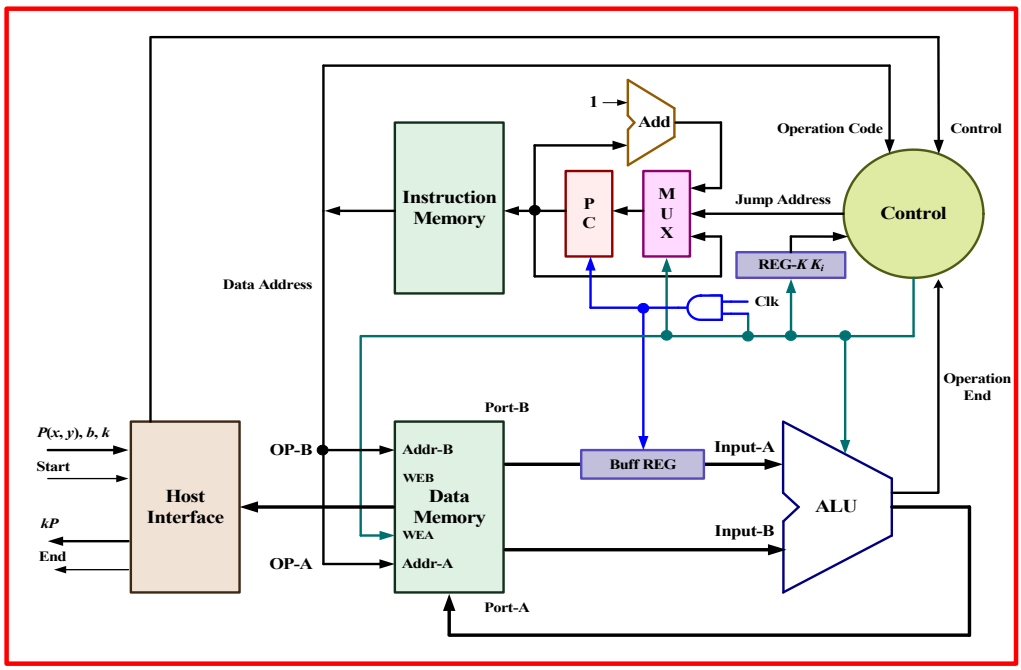
<그림 7> Message Scheduler 내부 구성

워진 값들은 σ0, σ1, 논리 함수를 통과하여 mod 32 Adder에 의해 더해져서 확장된 워드를 생성하게 되고, 이렇게 생성된 워드들은 다시 쉬프트 레지스터의 입력으로 주어지게 된다. 다음 그림 7은 Message Scheduler의 내부 구성을 보여준다.

Message Scheduler의 출력은 Compression Function의 입력으로 연결된다. 쉬프트 레지스터의 R0 레지스터에서 출력을 내보내지 않고 R9 레지스터에서 출력을 내보내는 것은 초기화 해쉬 값의 입력이 끝나는 타이밍과 맞추어 Compression Function의 연산을 수행하기 위함이며 이러한 설계는 R8 레지스터부터



<그림 8> SHA-256 Compression Function 구성도



<그림 9> $GF(2^{163})$상의 타원곡선 암호 프로세서 구조

R0 레지스터까지 데이터가 이동해서 출력되는 9 클록을 줄여주기 위함이다.

SHA-256의 핵심은 압축 함수이다. 그림 8은 본 논문에서 설계한 SHA-256의 압축 함수의 구성도를 나타낸다. IO Interface에서 입력되는 값 중 Input_hash 값은 Compression Function에서 해쉬 연산의 초기 값으로 입력된다. 32비트 워드의 해쉬 값을 8번, 총 256비트를 입력으로 취한다. 입력이 끝나는 시점에서 Message Scheduler 블록에서 확장된 워드인 w와 SHA-256 상수인 k값이 더해져서 Compression Function의 입력으로 들어오게 된다. IO Interface에서 입력되는 카운트 값을 통해 초기 해쉬 값 256비트가 입력되는 동안에는 연산을 메인 루프에 값을 넣지 않게 되며 초기 해쉬 값의 입력이 끝나면 매인 루프에 값을 전달하는 구조로 설계한다. 이 후 연산된 해쉬 값은 다시 매인루프로 전달되어지며 총 64번의 반복을 통해 최종 해쉬 값을 얻어낸다. 출력된 결과는 IO Interface에서 초기화 해쉬와 더해져 최종 결과를 출력한다.

2.3 Elliptic Curve Cryptosystem(ECC) 설계 [9,10]

타원곡선 암호 시스템은 RSA나 ElGamal과 같은 다른 암호 시스템에 비해 현저히 작은 키를 사용하면서(약 1/6 정도) 동일한 안전도를 가진다. ECC는 소프트웨어로 쉽게 구현이 가능하며, 이 경우 높은 유연성을 제공하지만 낮은 속도로 실시간 응용에는 적합하지 않다. 즉, 작은 키를 사용한다는 것은 계산 시간, 전력 소모, 저장 공간의 감소를 의미한다.

그림 9는 본 논문에서 설계한 타원곡선 암호프로세서의 전체적인 구조를 나타낸다. 본 연구에서 개발된 타원곡선 암호 프로세서는 크게 5개의 블록 Host Interface, Data Memory, Instruction Memory, Control, ALU로 구성된다. Host Interface는 타원곡선의 베이스 포인터, 곡선 파라메타 b, 비밀키 k 그리고 연산의 시작을 알리는 start 신호를 Host 프로세서로부터 입력 받아 타원곡선 암호프로세서로 전달한다. 타원곡선 프로세서의 데이터 전송 및 연산은 모두 163-비트로 이루어지며, Instruction Memory로부터 명령어 및 데이터 메모리의 주소를 전송받아 컨트롤 신호와 함께 연산을 수행한다.

Data Memory는 Dual Port 메모리로 구성하였으며, Port-A는 상승 에지에 Port-B는 하강 에지에 각각 동작한다. 그 외 모든 연산기 및 레지스터는 상승 에지에 동작한다. 따라서 데이터의 동기를 맞추기 위해 ALU의 Input-A와 Data Memory의 Port-B 사이에 Buffer 레지스터를 두었다. 따라서 모든 연산은 4사이클(명령어 패치 + 데이터 패치 + 데이터 로더 + 연산 수행 및 저장) 만에 수행된다.

<표 1> 데이터 메모리의 구성

변수명	주소
x	0000
y	0001
b	0010
x_0	0011
y_0	0100
X_1	0101
X_2	0110
Z_1	0111
Z_2	1000
Z_3	1001

곱셈 연산의 경우[$m/d+1$] 사이클이 소요되며, 나눗셈 연산의 경우 *Kim* 등[10]이 제안한 확장 바이너리 *GCD* 알고리즘을 이용하여 2^m 사이클이 소요된다.

타원곡선 정수 곱셈 알고리즘을 수행하기 위해 총 10개의 데이터 메모리가 필요하며, 표 1에 그 기능을 요약하였다.

표 2의 *López-Dahab* 타원곡선 정수 곱셈 알고리즘과 PB의 특성을 이용하면 표 3과 같이 총 46개의 연산 시퀀스를 얻을 수 있다. 표 2에는 표 3로부터 얻어진 3가지의 명령어를 요약하였다.

<표 2> *López-Dahab* 알고리즘에 기반한 명령어

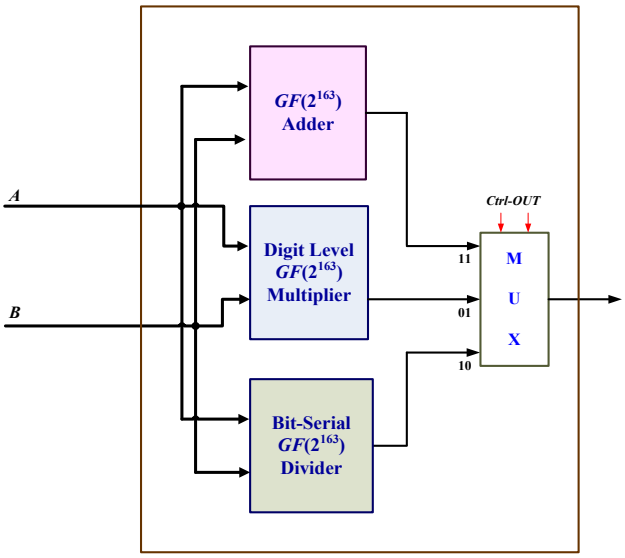
명령어	기능	명령어 인코딩	클럭 사이클
Mult	$C \leftarrow A \times B$	01	$m/d + 1$
Div	$C \leftarrow A / B$	10	$2m$
Add	$C \leftarrow A + B$	11	1

López-Dahab 알고리즘은 대부분의 연산이 곱셈으로 구성되어있기 때문에 빠른 연산 결과를 위해서 디지트-시리얼 곱셈기와 적은 하드웨어 복잡성을 위해서 비트-시리얼 나눗셈기를 사용하였다.

<표 3> *López-Dahab* 정수 곱셈 알고리즘에 기반 한 연산 시퀀스

순번	연산	비고	
0	$X_2 \leftarrow x \cdot x$	Initialize	
1	$X_2 \leftarrow X_2 \cdot X_2$		
2	$X_2 \leftarrow X_2 + b$		
3	$Z_2 \leftarrow x \cdot x$		
4	$x_0 \leftarrow X_1 \cdot Z_2$	Main Loop	
5	$y_0 \leftarrow X_2 \cdot Z_1$		
6	$Z_3 \leftarrow x_0 + y_0$		
7	$x_0 \leftarrow x_0 \cdot y_0$		
8	$X_1 \leftarrow Z_3 \cdot Z_3$	if k_i=1	Main Loop
9	$X_1 \leftarrow X_1 \cdot x$		
10	$X_1 \leftarrow X_1 + x_0$		
11	$Z_1 \leftarrow Z_3 \cdot Z_3$		
12	$x_0 \leftarrow Z_2 \cdot Z_2$		
13	$x_0 \leftarrow x_0 \cdot x_0$		
14	$x_0 \leftarrow x_0 \cdot b$		
15	$y_0 \leftarrow X_2 \cdot X_2$		
16	$y_0 \leftarrow y_0 \cdot y_0$		
17	$Z_2 \leftarrow X_2 \cdot Z_2$		
18	$Z_2 \leftarrow Z_2 \cdot Z_2$		
19	$X_2 \leftarrow y_0 + x_0$		
20	$X_2 \leftarrow Z_3 \cdot Z_3$	if k_i=0	
21	$X_2 \leftarrow X_2 \cdot x$		
22	$X_2 \leftarrow X_2 + x_0$		
23	$Z_2 \leftarrow Z_3 \cdot Z_3$		
24	$x_0 \leftarrow Z_1 \cdot Z_1$		
25	$x_0 \leftarrow x_0 \cdot x_0$		
26	$x_0 \leftarrow x_0 \cdot b$		
27	$y_0 \leftarrow X_1 \cdot X_1$		
28	$y_0 \leftarrow y_0 \cdot y_0$		
29	$Z_1 \leftarrow X_1 \cdot Z_1$		
30	$Z_1 \leftarrow Z_1 \cdot Z_1$		
31	$X_1 \leftarrow y_0 + x_0$		
32	$x_0 \leftarrow 1/Z_1$	Coordinate Conversion	
33	$x_0 \leftarrow X_1 \cdot x_0$		
34	$X_1 \leftarrow x + x_0$		
35	$Z_1 \leftarrow 1/Z_2$		
36	$Z_1 \leftarrow X_2 \cdot Z_1$		
37	$Z_1 \leftarrow x + Z_1$		
38	$Z_1 \leftarrow Z_1 \cdot X_1$		
39	$Z_2 \leftarrow x \cdot x$		
40	$Z_2 \leftarrow Z_2 + y$		
41	$Z_1 \leftarrow Z_1 + Z_2$		
42	$Z_1 \leftarrow Z_1 \cdot X_1$		
43	$y_0 \leftarrow 1/x$		
44	$y_0 \leftarrow Z_1 \cdot y_0$		
45	$y_0 \leftarrow y_0 + y$		

본 논문에서는 $GF(2^m)$상의 곱셈기 및 나눗셈기 그리고 명령어로부터 그림 10과 같은 ALU를 설계하였다.



<그림 10> $GF(2^{163})$상의 PB를 이용한 ALU

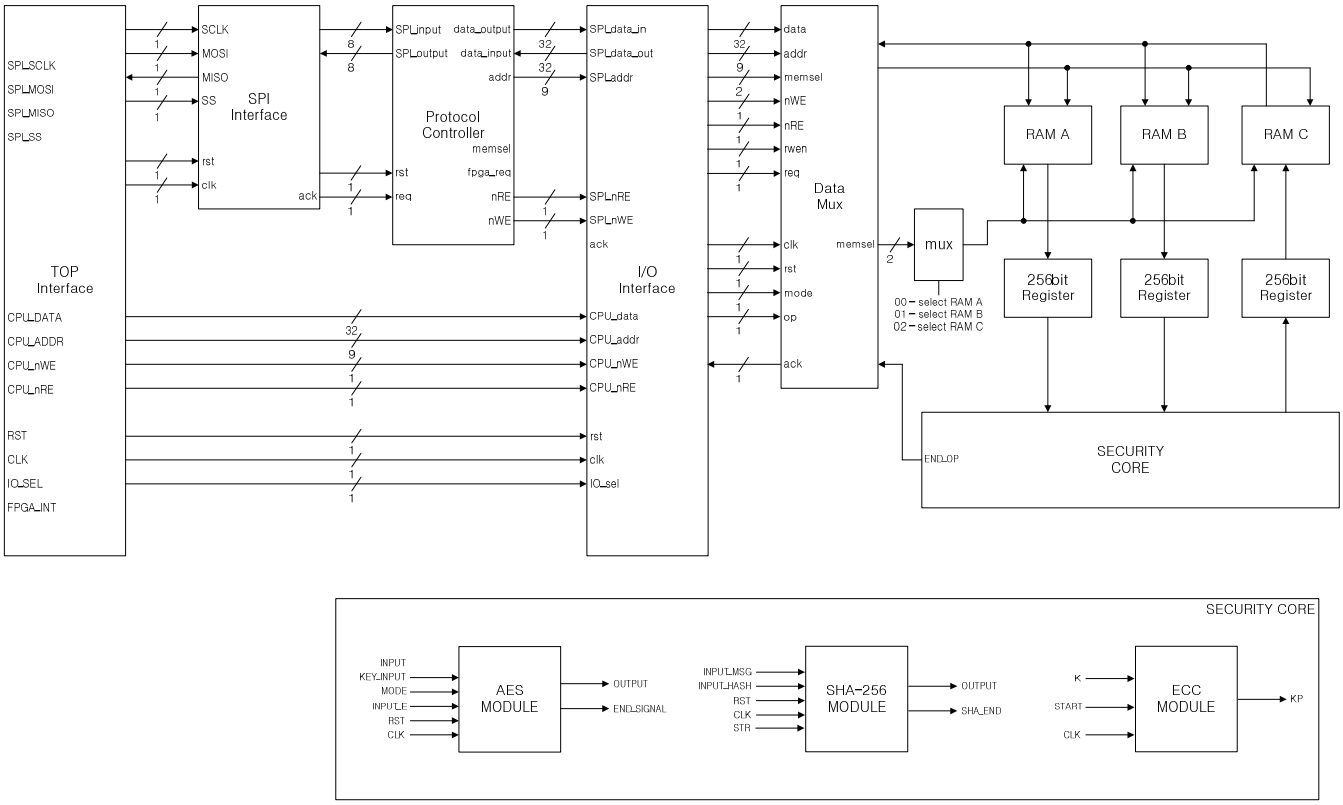
그림 10은 *NIST*와 *IEEE 1363*의 필드 크기 중 163을 선택하였다. 그림 10에 나타나듯이 곱셈 및 나눗셈 연산기에 덧셈기와 출력을 제어하기 위한 멀티플렉서를 추가하였다. 명령어에 따른 출력 제어 함수의 기능을 표 4에 요약하였다.

<표 4> 명령어에 따른 Ctrl-Out 신호

비트-시퀀스	명령어
11	*Add*
01	*Mult*
10	*Div*

3. 통합 인터페이스의 전체 구조

본 논문에서 구현된 보안 모듈은 하나의 공통된 인터페이스를 공유함으로써 통합된 형태로 구현된다. Security Core는 구현된 보안 모듈인 AES와 SHA-256, ECC 모듈이 포함되며 각각의 모듈은 FPGA_IO 모듈을 통해 제어신호와 입력 값을 전달받게 된다. FPGA_IO 모듈은 보드상의 CPU인 PXA272로부터 데이터와 제어신호를 전달받게 되며 전달된 데이터는 각각의 보안 모듈에 따라 RAM_A와 RAM_B에 입력된다. 또한 전달된 제어신호는 Security Core에 입력되어 주어진 데이터가 어떠한 모듈을 사용할지 결정



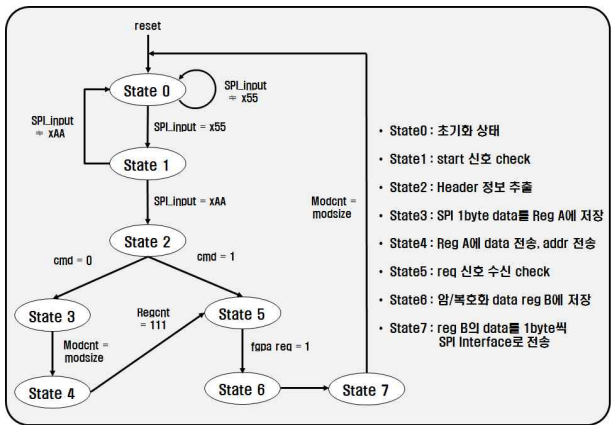
<그림 11> 통합 인터페이스의 전체 구조

하게 된다. 보안 모듈의 수행을 거친 데이터는 RAM_B에 적재되고 이와 동시에 출력되는 end 신호에 의해 다시 CPU로 출력한다. 그림 11은 설계된 통합 인터페이스의 전체 구조를 나타낸다. Security Module과 내부 RAM 및 내부 ROM의 제어를 맡고 있고 내부 레지스터 맵의 정보에 따라 데이터를 각각의 모듈에 전송한다.

3.1 저성능 MCU를 위한 SPI 통신 모듈

센서 네트워크와 같은 저성능의 MCU를 사용하는 경우에는 데이터 핀과 어드레스 핀에 할당할 수 있는 핀의 수가 부족하다. 그래서 본 논문에서 개발한 SoC 칩은 그 부분을 보완하기 위하여 SPI 통신을 이용한 통신방식을 지원한다[11].

아래 그림 12는 통합 인터페이스의 내부 SPI 통신 모듈의 State Diagram이다. 총 7단계로 이루어져있으며 프로토콜 방식으로 데이터를 송수신한다.

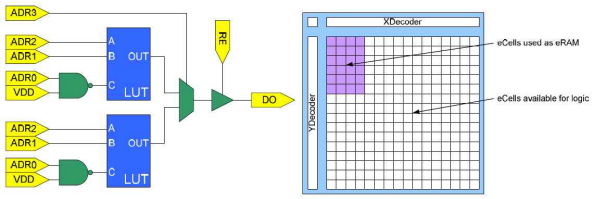


<그림 12> SPI 통신 State Diagram

3.2 내부 RAM

보안 모듈에서의 암호화 데이터 및 암호 키 값을 저장하기 위해서 설계되었다. 아래 그림 13은 eASIC社의 eRAM 구조를 나타내고 있다. eRAM은 Nextreme

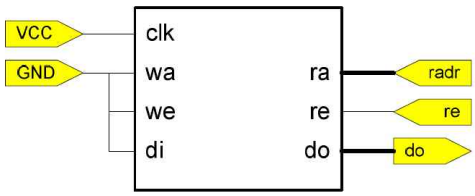
구조의 ASIC 장치에서 사용하는 전용 메모리이다. 각각의 암호 모듈이 RAM의 주소를 참조하여 데이터를 송 · 수신 받는다.



<그림 13> eASIC社의 eRAM 구조

3.3 내부 ROM

각각의 보안 모듈에서의 미리 정의된 데이터에 대해서는 ROM에 저장하도록 설계가 되어있다. 예를 들어 AES의 S-Box, ECC 연산에 필요한 명령어 Set, SHA-256 연산에 필요한 상수 값은 암호 모듈 연산에 필수 데이터이고 연산을 하더라도 변하지 않는 값이므로 ROM에 저장하여 읽기만 수행하도록 설계하였다. 아래 그림 14는 eASIC社의 eROM 구조를 나타내고 있다.



<그림 14> eASIC社의 eROM 구조

4. 성능 평가

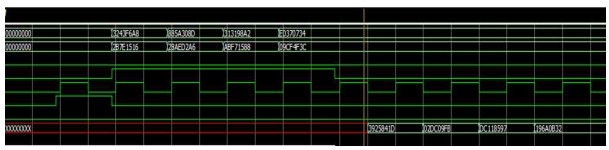
본 장에서는 설계한 SoC의 성능을 평가하기위해 정보보호 시스템을 위한 구현 환경 을 설명하고, 그 결과를 바탕으로 본 논문에서 제안한 시스템의 성능을 분석한다.

4.1 성능 평가 환경

본 논문에 제안된 정보보호 시스템의 FPGA 구현 및 기능 검증을 위해 VHDL로 회로를 기술하였고, Xilinx사의 회로합성 툴(XST : Xilinx synthesis technology)을 사용하여 회로를 합성, Net-list 파일을 추출한 후, Mento Graphics 사의 ModelSim SE 6.0을 이용하여 시뮬레이션 하였다. 또한 Xilinx사의 ISE 10.0i를 이용하여 Place & Route 과정을 거친 후, 타이밍 및 칩 사용율에 대해 분석하였다. FPGA 칩은 Xilinx사의 Virtex4 시리즈인 XC4VLX60을 대상 디바이스로 선택하였다.

4.2 제안된 정보보호 시스템의 결과 분석

본 절에서는 본 논문에서 제안한 정보보호 시스템의 구현 결과를 분석한다. 제안된 정보보호 시스템은 기존의 한가지의 보안 표준만을 지원하는 정보보호 시스템에 비해 여러 가지 보안 표준을 선택적으로 사용할 수 있으며 사용자의 설정에 따라 여러 가지의 보안 표준을 결합하여 사용할 수 있도록 설계되어있다.



<그림 15> AES 테스트 결과

<표 5> AES 구현 결과

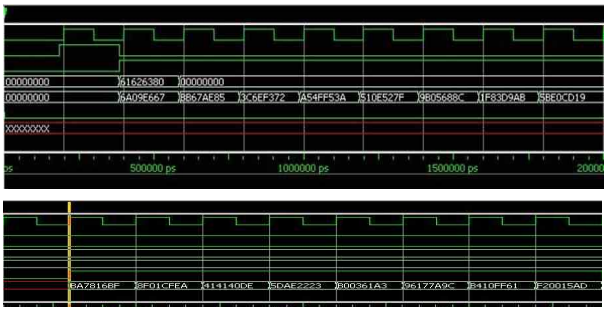
SLICE 개수	LUT	FF	최대 주파수
1898	3657	399	87MHz

아래 표 6은 AES의 SLICE 개수를 비교한 것이다. 본 논문에서는 저면적 AES를 설계함으로써 다른 AES설계 논문보다 SLICE 개수가 현저히 낮은 것을 볼 수 있다.

<표 6> AES의 비교 분석

구분	SLICE 개수	Throughput (Mbps)
Gaj 등[12]	2900	331
Dandalis 등[13]	5673	353
Elbirt 등[14]	9004	1940
Standaert 등[15]	2257	1563
본 논문	1898	121

SHA-256 모듈을 테스트하기 위해 표준 문서의 테스트 값인 "abc" 문자열을 사용하여 테스트하였다. 본 논문에서 설계된 SHA-256 모듈은 메시지와 초기 해쉬값을 입력으로 주어야하기 때문에 입력으로 해쉬값이 같이 주어진다.



<그림 16> SHA-256 테스트 결과

<표 7> SHA-256 구현 결과

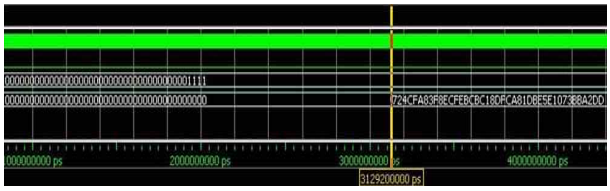
SLICE 개수	LUT	FF	최대 주파수
1994	3359	1320	43MHz

아래 표 8은 SHA-256의 SLICE 개수를 비교한 것이다. 본 논문에서는 저면적 SHA-256를 설계함으로써 동일한 출력값의 SHA-256설계 논문보다 SLICE 개수가 현저히 낮은 것을 볼 수 있다.

<표 8> SHA-256의 비교 분석

구분	SLICE 개수	Throughput (Mbps)
McEvoy 등[16]	2898	908
N. Sklavos 등[17]	1261	693
본 논문	1994	856

ECC 모듈의 입력 값으로는 키 값인 k와 두 좌표인 x, y 값을 넣어주었으며 그 결과는 그림 17과 같이 출력되었다.



<그림 17> ECC 테스트 결과

<표 9> ECC 구현 결과

SLICE 개수	LUT	FF	최대 주파수
5195	9640	2120	106MHz

아래 표 10은 다른 논문에서 제안된 프로세서를 비교분석한 결과이다. 아래 표에서 나타나듯이 훨씬 낮은 하드웨어 복잡도와 지연시간을 보인다.

<표 10> ECC의 비교 분석

구분	LUT	FF	지연시간 (μs)
Gura 등[18]	19,508	6,442	143
본 논문	9640	2120	52

5. 결 론

본 논문에서의 정보보호 시스템을 위한 FPGA 기반 하드웨어 가속기는 최신 정보보호 시스템의 다양한 요구사항인 속도, 면적, 전력, 유연성, 안전성을 만족시키기 위해서 저면적 32bit AES, SHA-256, ECC를 설계 및 정보보호 모듈에 최적화된 FPGA 통합 인터페이스를 개발하였다. 비밀키, 암호학적 해쉬, 공개키 암호알고리즘 탑재가 가능한 임베디드용 정보보호 시스템을 개발하기 위해서 비밀키 알고리즘(AES), 암호학적 해쉬함수(SHA), 공개키 암호시스템(ECC)의 모든 부분에 대해 자체 개발을 완료하고 연산레벨에서 시스템 개발까지 기존의 시스템과 다르게 개발하였다.

본 논문에서는 임베디드 시스템에 적용이 가능하도록 저면적으로 설계를 함으로써 다양한 정보보호 알고리즘을 추가하여도 시스템의 면적이 크게 늘어나지 않는 장점을 가진다. 기존의 시스템에서 단일의 정보보호 알고리즘만 지원하는 방식이 아닌 모듈형식의 다양한 정보보호 알고리즘을 지원하여 그 활용도를 높이고 RFID/USN 장비에도 적용이 가능하도록 SPI 통신으로 데이터 송수신이 가능하게 설계하여 저비용의 응용 뿐만 아니라 고속 통신 장비 시스템에도 적용이 가능하기 때문에 그 활용분야는 매우 높을 것으로 예상한다.

참 고 문 헌

[1] 김신효, 강유성, 정병호, 정교일, “u-센서 네트워크 보안 기술 동향,” 전자통신동향 분석, Vol. 20, No. 1 pp. 93-99, 2005. 2월.

[2] 주학수, 주홍돈, 김승주, "고속 암호연산 프로세서 개발 현황", 정보보호학회지, 제 12권, 제 3호, pp. 48 - 56, 2002. 6.

[3] NIST, Data Encryption Standard(DES), FIPS 46, 1977.

[4] A.J. Menezes, P.C. vanOorschot, and S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997.

[5] R.L. Rivest, M. Robshaw, R. Sidney, and Y. Yin, “The RC6 Block Cipher,” First Advanced Encryption Standard (AES) Conference, 1998.

[6] R.L. Rivest, The MD5 Message Digest Algorithm, RFC 1321.

[7] J.R. Goodman, Energy Scalable Reconfigurable Cryptographic Hardware for Portable Applications, PhD thesis, MIT, 2000.

[8] 김창훈, 김태호, 홍춘표, "GF(2m)상의 고속 타원곡선 암호 프로세서", 한국정보과학회 논문지 A - 시스템 및 이론 , Vol. 34, No. 3, pp. 113-123, April 2007

[9] C.H. Kim, S. Kwon, and C.P. Hong, "FPGA implementation of high performance elliptic curve cryptographic processor over GF(2^{163})", Journal of Systems Architecture, Vol. 54, pp. 893-900, August 27, 2008.

[10] 김창훈, 이남곤, 권순학, 홍춘표, “유한체 GF(2^m)의 응용을 위한 새로운 나눗셈 회로,” 한국정보처리학회 논문지 A, Vol.12-A, No.3, June. 2005.

[11] 송영석, 박성모, 김영민, “저가형 시스템을 위한 소프트웨어 SPI 통신 프로토콜구현”, 한국멀티미디어학회 춘계학술발표대회, Vol. 12, No. 1, pp. 260-262, 2009.

[12] K. Gaj and P. Chodowiec, “Comparison of the Hardware Performance of the AES Candidates using Reconfigurable Hardware,” The Third Advanced Encryption Standard (AES3) Candidate Conference, New York, USA, April 13-14, 2000.

[13] A. Dandalis et al., “A Comparative Study of Performance of AES Candidates Using FPGA’s,” The Third Advanced Encryption Standard (AES3) Candidate Conference, New York, USA, April 13-14, 2000.

[14] A.J. Elbirt et al., “An FPGA Implementation and Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists,” The Third Advanced Encryption Standard (AES3) Candidate Conference, New York, USA, April 13-14, 2000.

[15] F. X. Standaert et al., “A Methodology to Implement Block Ciphers in Reconfigurable Hardware and its Application to Fast and Compact AES Rijndael,” The Field Programmable Logic Array Conference, Monterey, California , pp.216-224, 2003.

[16] McEvoy R.P, Crowe F.M, Murphy C.C, Marnane W.P, “Optimisation of the SHA-2 family of hash functions on FPGAs”, Emerging VLSI Technologies and Architectures, IEEE Computer Society Annual Symposium, Vol. 00, pp. 2-3, 2006.

[17] K. K. Ting, S. C. L. Yuen, K.-H. Lee, and P. H. W. Leong, “An FPGA based SHA-256 processor”, Springer in FPL, Vol. 2438, pp. 577-585, 2002.

[18] N. Gura, S.C. Shantz, H.E. Sumit Gupta, V. Gupta, D. Finchelstein, E. Goupy, and D. Stebila, “An End-toEnd Systems Approach to Elliptic Curve Cryptography,” CHES '02, LNCS 2523, pp. 349-365, 2002.

차 정 우 (Jeong Woo Cha)

- 2008년 2월 : 대구대학교 멀티미디어공학과 (공학사)
- 2008년 3월 ~ 2010년 2월 : 대구대학교 컴퓨터정보공학과 (공학석사)
- 2010년 3월 ~ 현재 : 대구대학교 컴퓨터정보공학과 박사과정
- 관심분야 : 정보보호 시스템, Embedded System, VLSI 설계

김 창 훈 (Chang Hoon Kim)

- 종신회원
- 2001년 2월 : 대구대학교 컴퓨터정보공학부 (공학사)
- 2003년 2월 : 대구대학교 컴퓨터정보공학과 (공학석사)
- 2006년 8월 : 대구대학교 컴퓨터정보공학과 (공학박사)
- 2006년 9월 : 대구대학교 정보통신공학부, BK21 연구교수
- 2007년 8월 ~ 현재 : 대구대학교 컴퓨터·IT공학부, 조교수
- 관심분야 : 암호시스템, Embedded System, RFID/USN 보안

논 문 접 수 일 : 2012년 11월 29일
1차수정완료일 : 2013년 02월 10일
2차수정완료일 : 2013년 03월 15일
게 재 확 정 일 : 2013년 03월 15일