情報保護學會論文誌
第12券 第2號, 2002. 4

# AES Rijndael 블록 암호 알고리듬의 효율적인 하드웨어 구현*

안 하 기**, 신 경 욱***

# An Efficient Hardware Implementation of AES Rijndael Block Cipher Algorithm

Ha-Kee Ahn**, Kyung-Wook Shin***

## 요 약

본 논문은 차세대 블록 암호 표준으로 선정된 Rijndael(레인달) 암호 프로세서의 설계에 대해 기술한다. 단일 라운드 블록을 사용하여 라운드 변환을 반복 처리하는 구조를 채택하여 하드웨어 복잡도를 최소화하였다. 또한, 라운드 블록 내부에 서브 파이프라인 단계를 삽입하여 현재 라운드의 후반부와 다음 라운드의 전반부 연산이 동시에 처리되도록 하였으며, 이를 통하여 암·복호 처리율이 향상되도록 하였다. 라운드 블록은 32비트의 데이터 패스로 설계되었으며, 따라서 서브 파이프라인 단계는 4클록으로 수행된다. 라운드 블록의 복잡도에 가장 큰 영향을 미치는 S-box는 일반적인 룩-업 테이블 구현방식 대신에 유한체 $GF(2^8)$상의 곱셈의 역원 계산 회로를 이용하여 구현하였으며, 이를 통해 암호화와 복호화에서 S-box를 공유할 수 있도록 하였다. 라운드 키는 키 스케줄러에 의해 on-the-fly 방식으로 생성된다. Verilog-HDL로 모델링된 암호 프로세서는 0.25-㎛ CMOS 셀 라이브러리로 합성한 결과, 약 23,000개의 게이트로 구현되었으며, 8-ns의 최대지연을 가져 2.5-V 전원전압에서 120-㎒ 클록으로 안전하게 동작할 것으로 예측되었다. 설계된 암호 프로세서는 Xilinx FPGA로 구현하여 정상 동작함을 확인하였다.

## ABSTRACT

This paper describes a design of cryptographic processor that implements the AES (Advanced Encryption Standard) block cipher algorithm, "Rijndael". An iterative looping architecture using a single round block is adopted to minimize the hardware required. To achieve high throughput rate, a sub-pipeline stage is added by dividing the round function into two blocks, resulting that the second half of current round function and the first half of next round function are being simultaneously operated. The round block is implemented using 32-bit data path, so each sub-pipeline stage is executed for four clock cycles. The S-box, which is the dominant element of the round block in terms of required hardware resources, is designed using arithmetic circuit computing multiplicative inverse in $GF(2^8)$ rather than look-up table method, so that encryption and decryption can share the S-boxes. The round keys are generated by on-the-fly key scheduler. The crypto-processor designed in Verilog-HDL and synthesized using 0.25-㎛ CMOS cell library consists of about 23,000 gates. Simulation results show that the critical path delay is about 8-ns and it can operate up to 120-㎒ clock frequency at 2.5-V supply. The designed core was verified using Xilinx FPGA board and test system.

**Keyword :** *AES(Advanced Encryption Standard), Rijndael Algorithm, Block Cipher Algorithm, Cryptographic Processor*

* 본 연구는 한국과학재단 목적기초연구(2001-1-30200-006-1) 지원과 IDEC의 CAD Tool 지원으로 수행되었음.
** 금오공과대학교 전자공학부 VLSI 설계연구실(hkan@knut.kumoh.ac.kr)
*** 금오공과대학교 전자공학부 부교수(kwshin@knut.kumoh.ac.kr)

## Ⅰ. 서 론

암호화는 컴퓨터 네트워크를 통해 전달되는 정보 및 데이터를 제삼자가 가로채어 그 내용을 외부로 노출시키거나 의도적으로 내용을 조작·변경하는 등의 보안공격으로부터 정보를 보호하기 위한 수단으로 사용되며, 컴퓨터와 인터넷을 중심으로 한 정보화 사회가 도래함에 따라 그 중요성이 점점 증대되고 있는 핵심기술이다. 암호화 기술은 군사용과 국가 기관의 정보 보호를 목적으로 극히 제한적인 분야에만 사용되어 왔으나, 최근 인터넷을 기반으로 한 민간분야로 응용분야가 급속히 확산되고 있다. 암호화 기술은 인터넷 뱅킹 및 전자상거래, 증권거래 시스템의 안전성 확보, 차세대 이동전화, smart card, 전자주민증 등 휴대형 정보 단말기의 보안, 컴퓨터 파일 시스템의 기밀성 확보 등 매우 폭넓게 응용되고 있다[1,2].

DES를 중심으로 한 대칭키 암호 방식의 경우, 컴퓨터의 계산 능력 향상으로 인한 안전성 저하 문제가 대두되고 있다. 미국 상무부 기술표준국 (NIST)은 DES 보다 보안 기능이 훨씬 뛰어난 차세대 암호 표준 (Advanced Encryption Standard: AES)을 결정하기 위해 1997년 1월부터 준비작업을 시작했으며, 약 3년 동안의 평가과정을 통해 2000년 10월 미니에폴리스 연구그룹의 Joan Daemen과 Vincent Rijmen에 의해 제안된 "Rijndael" (레인달) 암호 알고리듬을 최종 AES로 선정하였다[6]. Rijndael로 암호화된 메시지의 해독을 위해서는 현재의 컴퓨터로 148조년이 걸리는 것으로 평가되는 등 모든 알려진 보안 공격에 대해 안전성이 뛰어나며, 다양한 형태의 플랫폼에서 처리 속도와 코드의 간결성, 그리고 설계의 단순성 등을 특징으로 한다.

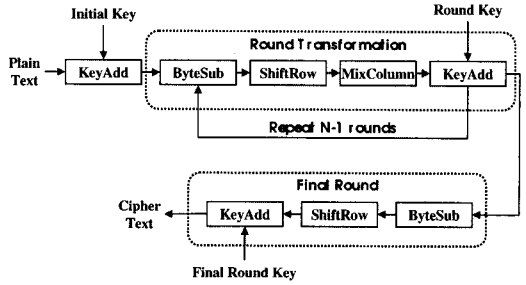
정보보안 시스템은 크게 나누어 소프트웨어 구현과 하드웨어 구현으로 구분된다. 소프트웨어 구현은 암호 알고리듬의 업그레이드가 용이하고 시스템간의 이식성과 유연성이 좋다는 장점을 가지나, 동작 속도가 느리고 해킹에 의한 암호 키의 노출 가능성이 있어 물리적인 안전성이 완벽히 보장되지 않는다는 단점을 갖는다. 반면에, 하드웨어 구현은 외부의 침입자에 의한 암호 알고리듬과 암호 키의 노출 및 조작이 불가능하므로 물리적인 안정성이 보장된다는 장점을 갖는다.

최근, 휴대형 정보 단말기를 통한 고속 정보 서비스의 확대와 함께 물리적인 안전성이 강조되면서 전용 하드웨어를 이용한 보안 시스템의 구현이 궁극적인 방안으로 인식되고 있다. 이들 휴대형 정보기기를 위해서는 고속/고집적/저전력 특성을 갖는 암호화 전용 프로세서의 개발이 필요하며, 다음과 같은 세 가지 형태로 구현된다. 첫째, 범용 마이크로 컨트롤러와 암호 프로세서 코어를 결합하여 구현하는 방식은 기존의 소프트웨어 환경을 사용할 수 있어 시스템 개발 시간을 단축할 수 있다는 장점이 있다. 반면에, 2개의 프로세서로 구성되므로 하드웨어 부담이 크다는 단점이 있다. 두 번째 방식은 암호화 전용 ASIC (Application Specific Integrated Circuit)을 개발하는 것으로, 고속/저전력 실현이 가능하고 대량생산 시 저렴한 가격으로 구현이 가능하다는 장점이 있으나, 개발 기간이 길며 알고리듬 및 파라메터의 변경이 불가능하여 유연성이 떨어진다. 셋째 방식은 FPGA(Field-Programmable Gate Array)와 같은 프로그램 가능 디바이스를 이용하여 구현하는 것으로, 암호 알고리듬의 업그레이드 및 파라메터 변경 등이 가능하면서도 물리적인 안전성이 보장된다는 장점을 갖는다[7,8].

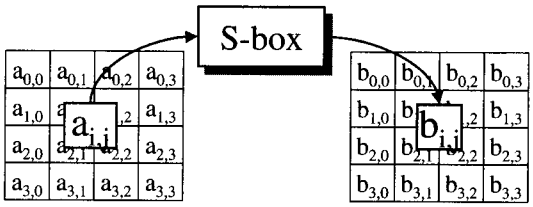
본 논문에서는 AES Rijndael 암호 알고리듬의 효율적인 구현을 위한 하드웨어 구조를 제안하고, 이를 Verilog-HDL로 모델링하여 설계하였다. Ⅱ장에서는 Rijndael 암호 알고리듬에 대해 간략히 기술하고, Ⅲ장에서는 아키텍쳐 및 내부 기능블록의 설계와 검증 결과를 기술하였다. Ⅳ장에서는 설계된 Rijndael 암호 코어의 검증결과를 기술하고, Ⅴ장에서 결론을 제시하였다.

## Ⅱ. Rijndael 블록 암호 알고리듬[6]

대부분의 대칭키 암호 시스템은 Feistel 구조[9]의 라운드 변환을 기반으로 하는 반면에, Rijndael 암호 알고리듬은 non-Feistel 구조를 바탕으로 하고 있으며, 역 변환이 가능한 3개의 독립된 라운드 변환으로 구성된다. 블록 길이는 128비트이고, 키 길이는 128비트/192비트/256비트 중에서 선택할 수 있으며, 라운드 수 (Nr)는 키 길이 (Nk)에 따라 10/12/14로 구성된다. Rijndael 알고리듬의 암호화 연산 과정은 [그림 1]과 같으며, 초기 라운드 키 가산, (Nr-1)번의 반복 라운드 및 최종 라운드의 순서로 처리된다. 최종 라운드를 제외한(Nr-1)번의 라운드는 ByteSub, ShiftRow, MixColumn 및 KeyAdd 등의 변환으로 구성되며, 이들은 4행×Nb열(단, Nb=4)로 구성




〔그림 1〕 AES Rijndael 알고리듬의 라운드 변환



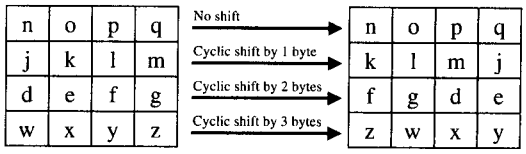

〔그림 2〕 ByteSub 변환

되는 State(128비트 데이터를 4×4바이트의 2차원 배열로 만든 것을 State라고 한다.)에 대해 다음과 같은 연산을 수행한다.

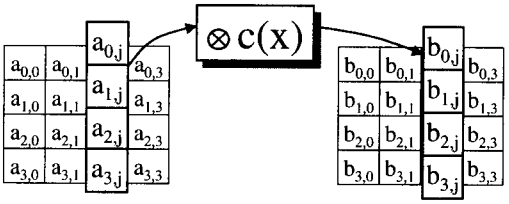
첫째, ByteSub 변환은 〔그림 2〕에서 보는 바와 같이, State를 구성하는 각각의 바이트에 대해 서로 독립적인 비선형 치환을 수행한다. 여기에 사용되는 치환 테이블을 S-box라고 하며, 이는 역변환이 가능한 두 단계의 변환 과정 즉, 유한체(finite field) $GF(2^8)$에서 곱셈의 역원(multiplicative inverse)을 취하는 $x \rightarrow x^{-1}$ 매핑과 $GF(2)$에서의 affine 변환으로 구성된다.

둘째, ShiftRow 변환은 〔그림 3〕과 같으며, State의 값을 변경시키지 않으면서 State를 구성하는 바이트들의 위치를 교환한다. 첫째 행을 제외한 나머지 행들을 각각 서로 다른 offset으로 바이트 단위의 순환 이동을 통해 위치를 교환한다. 즉, Row 0를 제외한 Row 1, 2, 3은 각각 1, 2, 3 바이트만큼 왼쪽으로 순환이동 시킨다.

셋째, MixColumn 변환은 State의 행을 유한체 $GF(2^8)$의 다항식으로 생각하여 〔그림 4〕에 나타낸 것과 같은 $b(x) = c(x) \otimes a(x)$의 다항식 곱셈을 연산한다. 마지막으로, KeyAdd 블록은 State의 모든 바이트에 라운드 키를 가산하며, 이는 비트 단위의 EXOR 연산으로 처리된다. 암·복호화 라운드에서 사용되는 라운드 키는 외부에서 입력된 암호키와 Rijndael 키 생성 알고리듬에 의해 생성된다.




〔그림 3〕 ShiftRow 변환




〔그림 4〕 MixColumn 변환

한편, Rijndael의 복호화 연산은 암호화의 역순으로 이루어지며, 암호화에 사용된 연산의 역 변환(InvByteSub, InvShiftRow, InvMixColumn)이 사용되고, 라운드 키는 암호화 연산의 역순으로 사용된다.

## Ⅲ. 회로 설계

본 장에서는 Ⅱ장에서 설명된 AES Rijndael 블록 암호 알고리듬의 하드웨어 구현에 대해 기술한다. 설계된 Rijndael 암호 프로세서 코어는 블록길이와 암호 키 길이가 모두 128비트인 AES-128 알고리듬을 구현하였으며, 동작모드 신호에 의해 암호화와 복호화 처리가 제어된다.

### 3.1 아키텍쳐 개요

Rijndael 알고리듬의 효율적인 하드웨어 구현을 위해 다음과 같은 설계 원칙을 고려하였다.

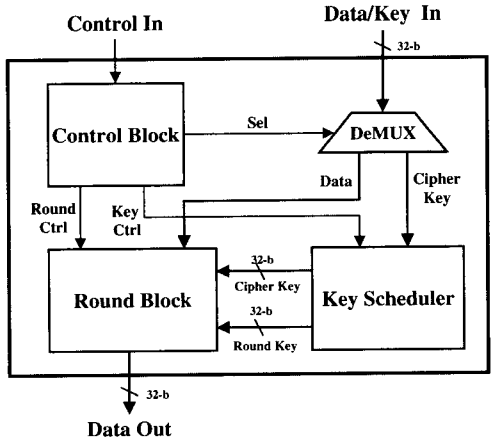
첫째, AES 표준안에 의하면, Rijndael 알고리듬의 라운드 수는 암호 키의 길이에 따른 가변 라운드 구조를 갖는다. 따라서, 라운드 연산에 개별적인 회로를 사용하면 암호 키 길이의 변경에 대한 융통성이 떨어지고, 또한 전체적인 하드웨어 복잡도가 너무 크게된다. 본 논문에서는 단일 라운드 연산회로를 사용하여 Nr번의 라운드 연산을 반복 수행하는 방식을 채택하였다.

둘째, 하드웨어 복잡도와 연산처리 시간 사이의 trade-off 관계에 대한 분석을 통해 라운드 연산 블록의 회로 구조를 결정하였다. 라운드 연산의 State가 4행×4-바이트의 2차원 배열로 구성되므로 16

바이트(즉, 128비트)의 State를 병렬 처리하는 경우, ByteSub 변환을 위해 32개의 S-box(암호화에 16개, 복호화에 16개)와 MixColumn 변환을 위해 32개의 유한체 곱셈기(암호화에 16개, 복호화에 16개)가 필요하여 하드웨어가 매우 복잡해진다. 본 논문에서는 State를 4 바이트 단위로 병렬 처리함으로써 완전 병렬처리 방식에 비해 하드웨어 복잡도를 1/4로 줄였으며, 라운드 연산이 4 클록에 처리되도록 하였다.

셋째, Ⅱ장에서 언급된 바와 같이, Rijndael 알고리듬의 라운드 연산에서 ShiftRow 변환은 State의 행 단위로 처리되고, MixColumn 변환은 열 단위로 처리된다. 따라서, 본 논문에서는 라운드 연산을 전반부 처리(ByteSub 변환, ShiftRow 변환)와 후반부 처리(MixColumn 변환, 라운드 키 가산) 두 부분으로 나누고, 이들 사이에 파이프라인을 삽입함으로써 암·복호화 연산 처리의 속도가 향상되도록 하였다.

넷째, 하드웨어 구현의 측면에서 Rijndael 알고리듬을 분석하면, S-box는 하드웨어 복잡도에 가장 큰 영향을 미치는 요소이며, 나머지 ShiftRow 변환, MixColumn 및 라운드 키 가산 등의 연산은 상대적으로 하드웨어 복잡도가 작다. 본 논문에서는 이와 같은 분석을 토대로 암·복호화 연산의 하드웨어 공유를 극대화하기 위한 방안으로서, S-box의 일반적인 구현 방법인 룩업 테이블(lookup table) 대신에 유한체 연산회로를 이용하였다. 이를 통해, 4개의 S-box만을 사용하여 암·복호화의 라운드 연산이 구현되도록 하였다.



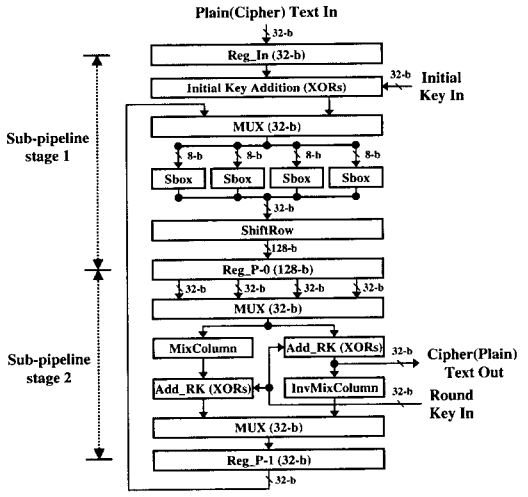

〔그림 5〕 AES 암호 프로세서의 구조

설계된 AES 암호 프로세서의 내부 구조는 〔그림 5〕와 같으며, Ⅱ장에서 설명된 Nr번의 라운드 연산을 처리하는 라운드 블록(Round Block), 라운드 키 생성블록(Key Scheduler), 그리고 이들 두 블록에서 필요한 각종 제어신호를 생성하는 제어블록 등으로 구성된다. 외부와의 인터페이스는 32비트씩 이루어지며, 입·출력 핀 수를 최소화하기 위하여 암호 키와 평문 데이터의 입력은 동일 포트를 사용하도록 하였으며, 이를 위해 DeMUX 블록을 추가하였다. 키 생성블록의 입력단은 32비트 레지스터 4개를 쉬프트 레지스터로 구성하여 4 클록동안 128비트의 암호 키가 입력되도록 하였다.

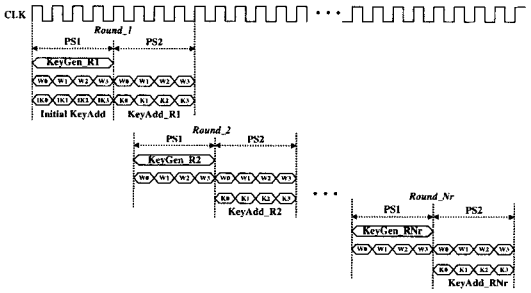
### 3.2 라운드 변환 블록

3.1절에서 설명된 아키텍쳐 설계 원칙을 적용하여 설계된 라운드 연산 블록의 내부 구조는 〔그림 6〕과 같다. 라운드 연산을 전반부 처리(ByteSub 변환, ShiftRow 변환)와 후반부 처리(MixColumn 변환, 라운드 키 가산)로 나누어 파이프라인을 삽입하였다. 데이터 패스는 32비트로 구성되어 단일 클록에 4 바이트씩 처리되도록 하였으며, 따라서 각 파이프라인 스테이지는 4 클록으로 동작된다.

초기 라운드 키 가산 블록은 32개의 XOR로 구성된다. ByteSub 변환은 암호화 연산과 복호화 연산에 하드웨어 공유 기법을 적용하여 4개의 S-box로 구현하였다. 이를 위해 기존의 룩업 테이블 구현 방법 대신에 유한체상의 곱셈의 역원 계산 회로를 이용




〔그림 6〕 라운드 변환 블록




〔그림 7〕 라운드 변환 블록의 서브 파이프라인 동작

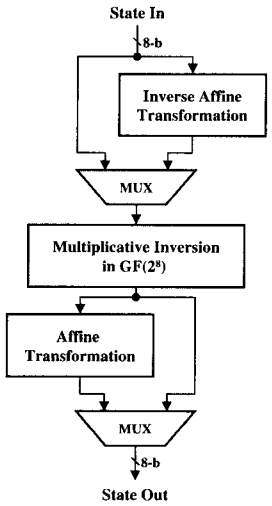
하여 S-box를 설계하였다(3.3절 참조). S-box에서 처리된 State는 ShiftRow 블록을 거쳐 파이프라인 레지스터로 인가된다. MixColumn 변환은 알고리듬 특성상 하드웨어 공유가 불가능하므로 암호화 연산 블록(MixColumn)과 복호화 연산 블록(InvMixColumn)을 병렬로 배치하고 그 출력에 MUX를 사용하여 구현하였다. 라운드 키 가산은 XOR로 구현되며, 그 결과는 다시 ByteSub 블록으로 귀환되어 다음 라운드 연산이 처리된다.

〔그림 7〕은 라운드 연산블록의 동작 타이밍도를 보인 것이다. 3.1절에서 언급된 바와 같이, 라운드 연산은 전반부(ByteSub 변환, ShiftRow 변환)와 후반부(MixColumn 변환, 라운드 키 가산)의 두 부분으로 나누어 파이프라인 처리된다. 그런데, 라운드 연산 전반부의 ShiftRow 변환은 State의 행 단위로 수행되고 후반부의 MixColumn 변환은 열 단위로 처리되므로 이들 사이에 직접 파이프라인을 삽입하는 것은 불가능하다. 본 논문에서는 i-번째 라운드의 후반부 연산과 (i+1)-번째 라운드의 전반부 연산 사이에 파이프라인 처리가 이루어지도록 하였다.

설계된 라운드 연산블록은 32비트 데이터 패스를 가지므로(그림 6 참조), 4행×4-바이트의 State를 처리하기 위해 각 파이프라인 스테이지는 4개의 클록으로 구현된다. 한편, 라운드 키는 해당 라운드의 전반부 처리가 진행되는 동안 on-the-fly 방식으로 생성되며, 후반부 처리기간에 가산된다.

### 3.3 $GF(2^8)$의 곱셈의 역원회로를 이용한 S-box구현

S-box는 라운드 블록의 하드웨어 복잡도에 가장 큰 영향을 미치는 요소이므로, Rijndael 알고리듬의 하드웨어 구현에 있어서 가장 우선적으로 고려해야 하는 요소로 인식되고 있다. 〔그림 7〕과 같이 단




〔그림 8〕 제안된 S-box 구조

일 클록에 4 바이트씩 처리하는 하드웨어 구조 및 동작 방식의 경우, 암호화 연산과 복호화 연산에 각각 4개씩 총 8개의 S-box가 필요하며, 키 생성 블록에서도 4개의 S-box가 필요하다.

S-box의 일반적인 구현방법은 S-box와 affine 변환을 하나로 묶어 치환 테이블(256바이트의 ROM에 해당)로 구현하는 것이며, 이 경우 설계가 단순하고 속도가 빠르다는 장점을 갖는다. 그러나, Rijndael의 하드웨어 구현에서는 암호화/복호화, 그리고 라운드 키 생성을 위해 각각 서로 다른 S-box가 필요하므로 하드웨어의 공유가 불가능하다는 단점을 갖는다.

본 논문에서는 치환 테이블 대신에 $GF(2^8)$에서 곱셈의 역원을 연산하는 회로를 사용함으로써 암호화와 복호화 과정에서 S-box를 공유하여 사용하도록 하였다. 〔그림 8〕은 본 논문에서 제안된 S-box의 구조이며, $GF(2^8)$에서 곱셈의 역원을 계산하는 회로와 affine 변환 블록으로 구성된다. 암호화 과정은 역원 계산 후에 affine 변환이 수행되며, 복호화는 역 affine 변환이 먼저 수행된 후 역원이 계산된다.

$GF(2^8)$에서 곱셈의 역원 연산은 Euclid 알고리듬을 이용하는 방법과 Fermat 정리를 이용하는 방법으로 구분할 수 있다. Euclid 알고리듬을 이용한 시스토릭 어레이(systolic array) 구조는 연산 속도가 빠르고 체(field)의 위수에 관계없이 규칙적인 구조를 갖는 장점이 있으나, 면적과 전력소모가 크다는 단점을 갖는다[10,11]. 한편, Fermat 정리는 역원 연산을 유한체상의 멱승(power) 연산과 곱셈 연산으로 분해하여 연산하는 방법이며, 유한체의 원

소 표시 방법에 따라 회로 구조가 달라진다. 정규기저(normal basis) 표현의 경우, 멱승 회로는 순환 이동(cyclic shifting)으로 구현할 수 있어 회로가 단순하지만 곱셈 회로가 복잡하며[12,13], 다항식 기저(polynomial basis) 표현의 경우에는 정규기저에 비해 멱승회로는 복잡하지만 곱셈회로가 단순하고 기저변환이 필요 없다는 장점을 갖는다[14]. 그 외에 이중기저(dual basis)를 이용하는 구현 방법[15]과 power-sum 연산을 이용하는 방법[16] 등이 제안되고 있다. 본 논문에서는 Fermat 정리를 이용한 역원 회로를 설계하였다.

Rijndael 알고리듬은 다항식 기저 표현을 사용하며, 기약 다항식(irreducible polynomial)은 다음과 같다.

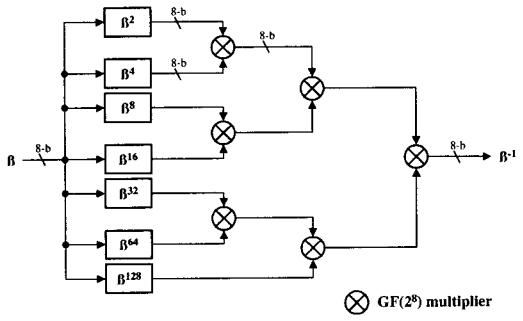
$$G(\alpha)=1+\alpha+\alpha^3+\alpha^4+\alpha^8 \quad (1)$$

Fermat 정리에 의하면, 유한체 $GF(2^m)$상의 임의의 원소 $\beta$에 대하여 $\beta^{2^m}=\beta$의 관계가 성립하며, 유한체 연산의 특성을 이용하면 $\beta$의 역원은 식(2)과 같이 표현된다.

$$\beta^{-1}=\beta^2\cdot\beta^4\cdot\beta^8\cdot\cdots\beta^{2^{m-1}} \quad (2)$$

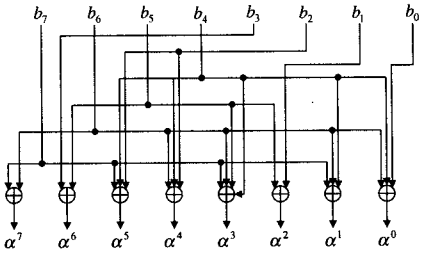
본 논문에서는 식(2)를 이용하여 $GF(2^8)$상의 역원 연산 회로를 [그림 9]와 같이 설계하였으며, $GF(2^8)$상의 멱승 회로와 곱셈 회로로 구성된다.

한편, 유한체 $GF(2^8)$상의 멱승 연산은 식(3)와 같이 정의되며, 식(1)에 정의된 Rijndael 알고리듬의 원시 다항식으로 modular reduction을 취하여 정리하면 유한체 상의 원소 $\beta$에 대한 멱승 $\beta^2$, $\beta^4$, $\beta^8$, $\cdots$ $\beta^{128}$를 계산할 수 있다. [그림 10]은 $\beta^2$을 계산하는 멱승 회로를 보인 것이다.

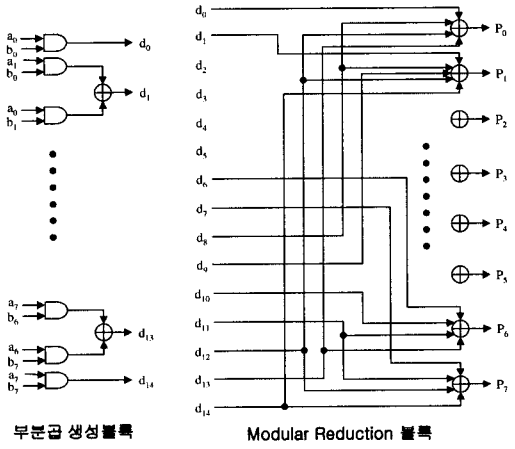



[그림 9] $GF(2^8)$상의 곱셈의 역원

$$\beta^2=(b_0\oplus b_4\oplus b_6)+(b_4\oplus b_6\oplus b_7)\alpha+(b_1\oplus b_5)\alpha^2+(b_4\oplus b_5\oplus b_6\oplus b_7)\alpha^3+(b_2\oplus b_4\oplus b_7)\alpha^4+(b_5\oplus b_6)\alpha^5+(b_3\oplus b_5)\alpha^6+(b_6\oplus b_7)\alpha^7$$




[그림 10] $GF(2^8)$상의 멱승 ($\beta^2$) 회로




[그림 11] $GF(2^8)$ 곱셈기

$$\beta^{2^i}=\sum_{k=0}^{m-1}b_k\cdot\alpha^{k(2^i)} \quad (3)$$

단, $i=1,2,3,\cdots\ m-1$

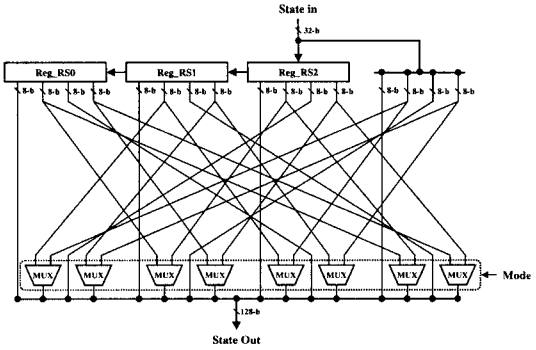
유한체 $GF(2^8)$상의 두 원소 $A(x)$와 $B(x)$의 곱셈은 부분곱 $D(x)=A(x)B(x)$을 구한 후, modular reduction $P(x)=D(x)$ mod $G(x)$을 취하여 계산되며, 다항식 $D(x)$와 $P(x)$의 계수들은 각각 식(4) 및 식(5)와 같이 정의된다. 이를 이용하여 구현된 $GF(2^8)$상의 곱셈기 회로는 [그림 11]과 같다.

$$d_k=\sum_{j=0}^{m-1}a_{k-j}\cdot b_j,\ (\ 0\le k\le 2m-2) \quad (4)$$

$$P_k=d_k+\sum_{i=0}^{2m-3}d_{m-1}\cdot g_{i,k},\ (0\le k\le m-1) \quad (5)$$

### 3.4 ShiftRow 및 MixColumn 블록

ShiftRow 변환은 ByteSub 블록의 출력에 대




〔그림 12〕 ShiftRow 변환기

해 행 단위의 순환이동으로 수행되므로, 순환기능을 갖는 쉬프트 레지스터로 구현이 가능하다. 그러나, ByteSub 변환은 열 단위로 처리되므로, State를 구성하는 16바이트에 대한 ByteSub 연산이 모두 완료된 후에 ShiftRow 연산이 시작될 수 있다. 본 논문에서는 32비트(4바이트) 레지스터 3개와 출력 MUX를 이용하여 〔그림 12〕와 같이 설계하였으며, 출력 쪽의 MUX는 ShiftRow 변환(암호화 연산)과 InvShiftRow 변환(복호화 연산)의 하드웨어 공유를 위해 사용되었다. 입력단의 3×4-바이트 레지스터는 ByteSub 블록에서 열 단위로 입력되는 State를 4 바이트씩 쉬프트 하여 저장하며, 실제 순환이동 동작은 쉬프팅 대신에 바이트 단위의 배선에 의해 이루어지도록 하였다. ShiftRow 변환기의 출력은 파이프라인 레지스터를 거쳐 MixColumn 변환기로 인가된다.

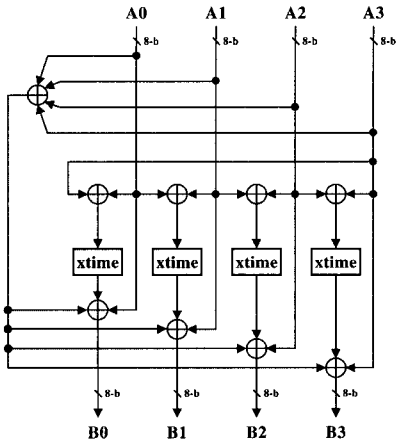
MixColumn 변환은 식(6)과 같이 정의되는 유한체상의 행렬 곱셈으로 연산된다.

$$B(x) = C(x) \otimes A(x) \bmod (x^4 + 1) \tag{6}$$

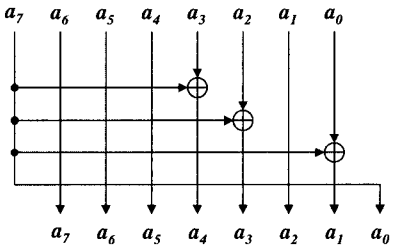
식(6)에서 기호 ⊗는 유한체 상의 다항식 곱셈을 나타내며, $C(x) = '02'x^3 + '01'x^2 + '01'x + '03'$이고, '01', '02', '03'은 16진수 상수 값을 나타낸다.

한편, 식(6)은 식(7)와 같이 다시 표현할 수 있으며, 이는 유한체 $GF(2^8)$상의 곱셈과 XOR 연산에 의해 구현될 수 있다.

$$\begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{pmatrix} \tag{7}$$




〔그림 13〕 MixColumn 변환기




〔그림 14〕 Xtime 회로

행렬계수 $C(x)$가 상수이므로, 유한체 연산의 특성을 이용하면 식(7)을 식(8)과 같이 전개할 수 있고, 따라서 본 논문에서는 유한체 곱셈기를 사용하는 대신에 〔그림 13〕과 같이 Xtime 연산회로와 XOR를 사용하여 구현하였다.

$$B_0 = '02'(A_0 \oplus A_1) \oplus (A_0 \oplus A_1 \oplus A_2 \oplus A_3) \oplus A_0 \quad (8-a)$$
$$B_1 = '02'(A_1 \oplus A_2) \oplus (A_0 \oplus A_1 \oplus A_2 \oplus A_3) \oplus A_1 \quad (8-b)$$
$$B_2 = '02'(A_2 \oplus A_3) \oplus (A_0 \oplus A_1 \oplus A_2 \oplus A_3) \oplus A_2 \quad (8-c)$$
$$B_3 = '02'(A_3 \oplus A_0) \oplus (A_0 \oplus A_1 \oplus A_2 \oplus A_3) \oplus A_3 \quad (8-d)$$

여기서, Xtime 연산은 $GF(2^8)$상의 16진수 '02'에 대한 곱셈을 의미하며, 다항식 표기법인 경우 다항식에 $x$를 곱한 후 기약 다항식으로 modulo reduction을 취하여 구해지며, 〔그림 14〕와 같이 구현하였다.

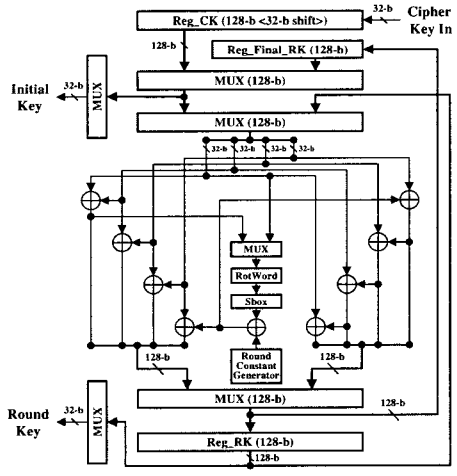
### 3.5 라운드 키 생성블록

Rijndael 암호 알고리듬에서는 외부에서 입력되

는 암호 키(cipher key)를 받아 라운드 연산에 사용되는 암호키를 생성하며, 오프라인 또는 on-the-fly 방식으로 생성될 수 있다. 오프라인 생성방식은 필요한 모든 라운드 키를 미리 생성하여 별도의 메모리나 레지스터에 저장한 후, 매 라운드마다 선택적으로 사용하는 방법이다. 한번 생성된 라운드 키는 새로운 암호키가 입력되기 전까지 암호화뿐만 아니라 복호화에도 그대로 사용될 수 있으므로 동작속도나 전력 소모 면에서 장점을 가지며, 라운드 키 생성을 위한 하드웨어는 작지만 생성된 라운드 키를 저장하기 위한 메모리나 레지스터의 부가 하드웨어를 필요로 하는 단점이 있다.

본 논문에서는 매 라운드마다 on-the-fly 방식으로 라운드 키를 생성하여 사용하는 방법을 채택하였다. 이 방법은 암호화 라운드 키 생성을 위한 하드웨어와 복호화 라운드 키 생성을 위한 부가의 하드웨어가 필요하지만 라운드 키를 저장하는 별도의 메모리나 레지스터가 필요 없다. 그러나 복호화에서는 암호화 과정의 역순으로 라운드 키가 사용되므로 이를 생성하기 위해 추가의 시간이 필요하다. 본 논문에서는 라운드 블록의 전반부 연산 4 클록 동안에 라운드 키를 생성하여 후반부 연산 4 클록에서 사용되도록 하였으며([그림 7] 참조), 암호화 최종 라운드 키를 레지스터에 저장하여 복호화 키 생성에 소요되는 시간을 줄였다.

설계된 라운드 키 생성블록의 내부 구조는 [그림 15]와 같다. 외부에서 32비트씩 입력되는 암호 키는 입력 레지스터에 32비트씩 쉬프트 되어 저장된 후, 키 스케줄러로 인가된다. 키 스케줄러는 S-box, 라운드 상수 생성기, RotWord, XOR 게이트 및 MUX 등으로 구성된다. 암호화와 복호화 과정의 라운드 키 생성을 위해 각각 4×32비트개의 XOR 게이트를 사용하였으며, 32비트 단위의 처리를 위해 4개의 S-box가 사용되었다.

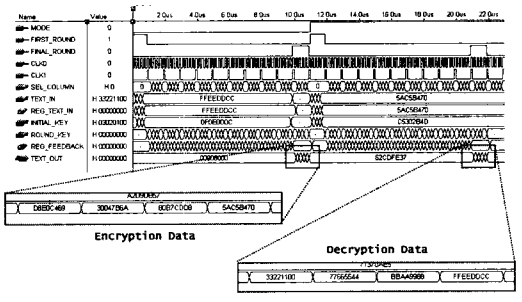



[그림 15] 라운드 키 생성블록

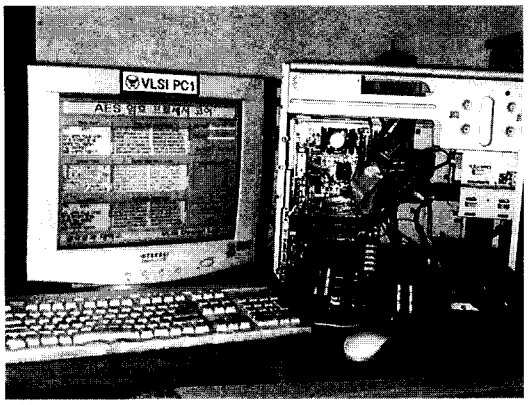
## Ⅳ. 설계 검증

설계된 Rijndael 암호 코어는 Verilog-HDL로 모델링하였으며, AES 표준안[17]에 명시된 테스트 벡터를 이용하여 검증하였다. 128비트의 평문 "FF EEDDCCBBAA99887766554433221100"와 128비트의 암호키 "F0E0D0C0B0A0908070605040 30201000"을 입력벡터로 사용한 시뮬레이션 결과는 [그림 16]과 같으며, 암호화 결과값 "5AC5B47080 B7CDD830047B6AD8E0C469"이 출력되었고, 이를 다시 복호화한 결과는 입력으로 사용된 평문과 동일한 값이 출력되었으며, 따라서 모든 논리기능이 정상적으로 동작함을 확인하였다. 시뮬레이션이 완료된 Verilog-HDL 모델은 최종적으로 FPGA 구현을 통해 검증하였으며, 검증 시스템은 [그림 17(a)]와 같이 FPGA 보드, PC 및 ISA 인터페이스 보드, 구동 소프트웨어 등으로 구성된다. FPGA 디바이스는 Xilinx XCV1000E를 사용하였으며, Visual C++ 언어로 테스트 프로그램을 작성하였다. [그림 17(b)]는 검증시스템의 실행 화면이며, 평문과 암호키를 입력하면 암호문이 출력되고 이를 다시 복호화하면 원래의 평문과 동일한 내용이 출력됨을 확인할 수 있다. 이와 같은 FPGA 구현 검증을 통해 설계된 AES 암호 코어가 정상적으로 동작함을 확인하였다.

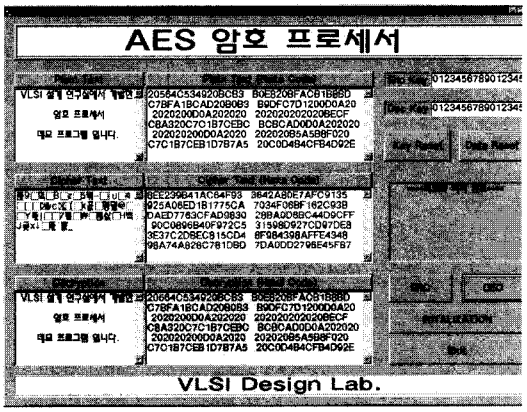
검증이 완료된 HDL 모델은 0.25-㎛ CMOS 셀 라이브러리와 Synopsys CAD 툴을 이용하여 회로




[그림 16] 설계된 Rijndael 암호 코어의 시뮬레이션 결과




(a) Verification System




(b) Verification result

〔그림 17〕 AES Rijndael 코어의 검증 시스템 및 실행 화면

〔표 1〕 설계된 Rijndael 암호 코어의 특징

| 암호 알고리듬 | AES-128 |
|---|---|
| 게이트 수 | 23,000 |
| 동작 주파수 | 120-㎒ @2.5-V |
| 평균 클록 수 | 5 클록/라운드 |
| 암·복호율 | 300-Mbps |
| 라운드 키 생성 | on-the-fly 방식 |
| 라운드 키 setup layency | · 암호화 : 0 cycle<br>· 복호화 : 20 cycles |

〔표 2〕 Rijndael 구현 비교

| 문헌 | 〔18〕 | 〔19〕 | 〔20〕 | 본 논문 |
|---|---|---|---|---|
| S-box | 16 | 16 | 16 | 4 |
| round block | 128-b | 128-b | 128-b | 32-b |
| 성능 〔Mbps〕 | 320 | 910 | 1,280 | 300 |
| 게이트 수 | 1,000,000 | 173,000 | 44,300 | 23,000 |
| 공정 | 0.5-㎛ | 0.18-㎛ | 0.5-㎛ | 0.25-㎛ |

합성을 하였다. S-box는 1,900 게이트, 라운드 블록은 10,720 게이트로 구현되었으며, 전체 Rijndael 암호 코어는 총 23,000게이트로 구현되었다. 설계된 회로의 시뮬레이션 결과에 의하면, 최대지연은 8-㎱이며, 최대 2.5-V 전원전압에서 120-㎒로 동작 가능할 것으로 평가되었다. 설계된 Rijndael 암호 코어의 특징은 〔표 1〕과 같다. 〔표 2〕는 본 논문의 설계 결과를 문헌에 발표된 구현 사례와 비교한 것이다. 문헌 〔18〕-〔20〕의 설계사례는 16개의 S-box를 사용하여 128비트씩 처리하는 회로구조를 가지며, 본 논문의 설계는 4개의 S-box를 사용하여 32비트씩 처리하는 회로구조를 갖는다. 따라서, 본 논문에서 설계된 Rijndael 암호 프로세서 코어는 적은 칩 면적과 저전력 소모를 필요로 하는 분야에 적합할 것으로 판단된다.

## V. 결 론

본 논문에서는 미 상무부 기술표준국에서 차세대 블록 암호 표준(AES)으로 선정된 Rijndael 암호 알고리듬용 프로세서 코어를 설계하였다. 블록 길이와 키 길이가 모두 128비트인 AES-128 알고리듬을 구현하였으며, 단일 라운드 연산회로를 사용하여 라운드 연산을 반복 수행하는 방식을 채택하였다. 또한, 하드웨어 복잡도를 줄이기 위해 단일 라운드 연산이 4 클록으로 처리되도록 하였으며, 라운드 연산 중간에 파이프라인을 삽입함으로써 연산속도가 향상되도록 하였다. 라운드 키는 on-the-fly 방식으로 생성되며, 라운드 블록의 전반부 연산 4 클록 동안에 라운드 키를 생성하여 후반부 연산 4 클록에서 사용되도록 하였다. S-box는 치환 테이블 대신에 $GF(2^8)$에서 곱셈의 역원 연산회로를 사용함으로써 암호화와 복호화 과정에서 S-box가 공유되도록 설계하였다. 설계된 AES Rijndael 암호 코어는 Xilinx FPGA 디바이스로 구현하여 정상 동작함을 확인하였다. 0.25-㎛ CMOS 셀 라이브러리로 합성한 결과, 약 22,310개의 게이트로 구성되며, 2.5-V의 전원전압에서 120-㎒로 동작 가능할 것으로 평가되어 약 300-Mbps의 암·복호율의 성능을 갖는다.

본 논문에서는 AES-128 알고리듬만을 구현하였으나, AES 표준안에 정의된 확장모드인 AES-192(키 길이 192비트)과 AES-256(키 길이 256비트)알고리듬은 라운드 키 생성블록을 수정하여 쉽게 구현될 수 있다. 설계된 AES 암호 코어는 반도체 지적재산권인 소프트 IP(Intellectual Property)로 가공하여 네트웍, 전자상거래, smart card, 전자주민증 등 휴대형 정보 단말기의 보안 모듈 설계에 사용될 수 있을 것으로 판단된다.

## 참 고 문 헌


[1] 박창섭, 암호이론과 보안, 대영사, 1999.

[2] W. Stalling, *Cryptography and Network Security*, Prentice Hall, 1999.

[3] National Bureau of Standards, NBS FIPS PUB 46, "Data Encryption Standard", *National Bureau of Standards*, U.S. Department of Commerce, Jan., 1977.

[4] R.L. Rivest, A. Shamir, and L.M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, Vol. 21, No. 2, pp. 120~126, Feb., 1978.

[5] I.F. Blake, G. Seroussi, and N.P. Smart, *Elliptic Curves in Cryptography*, London Mathematical Society Lecture Note Series, 265, Cambridge University Press, 1999.

[6] J. Daemen and V. Rijmen, "AES Proposal : Rijndael Block Cipher", NIST Document ver.2, Mar., 1999, *http://www.nist.gov/aes*.

[7] H. Kuo and I. Verbauwhede, "An embedded Cryptographic processor for the Rijndael AES algorithm", *Annual Research Review* 2000, UCLA, 2000.

[8] A.J. Elbirt, W. Yip, B. Chetwynd, and C. Parr, "An FPGA-based Performance evaluation of the AES block cipher candidate algorithm finalists", *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, Vol. 9, No. 4, Aug., 2001.

[9] B. Schneier, *Applied Cryptography : Protocols, Algorithms and Source Code in C*, John Wiley & Sons, Inc., 1996.

[10] J.H. Guo and C.L. Wang, "Systolic array implementation of Euclid's algorithm for inversion and division in GF($2^m$)", *Proc. of the IEEE Int. Sym. on Circuits and Systems*, Vol. 2, pp. 481~484, May. 1996.

[11] C.T. Huang and C.W. Wu, "Highspeed easily testable Galois-field inverter", *IEEE Trans. on Circuits & Systems II-Analog And Digital Signal Processing*, Vol. 47, No. 9, pp. 909~918, Sep. 2000.

[12] L. Gao and G.E. Sobelman, "Improved VLSI designs for multiplication and inversion in GF($2^m$) over normal bases", *13th Annual IEEE International ASIC/SOC Conference*, pp. 97~101, Sep. 2000.

[13] G.L. Feng, "A VLSI architecture for fast inversion in GF(2m)", *IEEE Trans. Comput., Vol. 38, No. 10*, pp. 1389~1386, Oct. 1989.

[14] A.V. Dinh, R.J. Palmer, R.J. Bolton, and R. Mason, "A low latency architecture for computing multiplicative inverses and divisions in GF($2^m$)", *IEEE Pro. of the 2000 Canadian Conf. on Electrical and Computer Engineering*, Vol. 1, pp. 43~47, May. 2000.

[15] S.T.J. Fenn, M. Benaissa, and D. Taylor, "GF($2^m$) multiplication and division over the dual basis", *IEEE Trans. on Computers*, Vol. 45, No. 3, pp. 319~327, Mar. 1998.

[16] S.W. Wei, "VLSI architecture for computing exponentiations, multiplicative inverses, and divisions", *IEEE Trans. on Circuits and Systems II-Analog & Digital Signal Processing*, Vol. 44, No. 10, pp. 847~855, Oct. 1997.

[17] NIST, "Announcing the Advanced Encryption Standard(AES)", *FIPS PUB ZZZ*, 2001, http://www.nist.gov/aes.

[18] M. Bean, C. Ficke, T. Rozylowicz, and B. Weeks, "Hardware performance simula-

tions of round 2 Advanced Encryption Standard Algorithms", *http:csrc.nist.gob/encryption/aes/round2/NSA-AESfinalreport.pdf.*

[19] H. Kuo and I. Verbauwhede, "Architectural optimization for a 1.82Gbits/sec VLSI implementation of the AES Rijndael Algorithm", *Workshop on Cryptographic Hardware and Embedded Systems 2001*(CHES 2001), pp. 51~64, May, 2001.

[20] 전신우, 정용진, 권오준, "Rijndael 암호 알고리즘을 구현한 암호 프로세서의 설계" 정보보호학회 논문지, Vol. 11, No. 6, pp. 78~87, 2001, 12.

## 〈著者紹介〉

**안 하 기 (Ha-Kee Ahn) 학생회원**
2000년 2월 : 금오공과대학교 전자공학과 졸업
2002년 2월 : 금오공과대학교 대학원 전자공학과 공학석사
2002년 3월~현재 : (주)한기아 연구원
〈관심분야〉 통신/신호처리용 집적회로 설계, 암호 프로세서 설계, 정보보호

**신 경 욱 (Kyung-Wook Shin) 정회원**
1984년 2월 : 한국항공대학교 전자공학과 졸업
1986년 2월 : 연세대학교 대학원 전자공학과 공학석사
1990년 8월 : 연세대학교 대학원 전자공학과 공학박사
1990년 9월~1991년 6월 : 한국전자통신연구소 반도체연구단
1995년 8월~1996년 7월 : University of Illinois at UC 방문교수
1991년 7월~현재 : 금오공과대학교 전자공학부 부교수
〈관심분야〉 통신/신호처리용 집적회로 설계, 암호 프로세서 설계, 정보보호, ROIC 설계