論文2002-39SD-11-5

On-the-fly 키 스케줄러를 갖는 AES-128/192/256 Rijndael 암호 프로세서

(AES-128/192/256 Rijndael Cryptoprocessor with On-the-fly Key Scheduler)

安 河 基 *, 辛 卿 旭 **

(Ha-Kee Ahn and Kyung-Wook Shin)

요 약

차세대 블록 암호 표준인 AES (Advanced Encryption Standard) Rijndael(라인달) 암호 프로세서를 설계하였다. 라운드 변환블록 내부에 서브 파이프라인 단계를 삽입하여 현재 라운드의 후반부 연산과 다음 라운드의 전반부 연산이 동시에 처리되도록 하였으며, 이를 통하여 암·복호 처리율이 향상되도록 하였다. 라운드 처리부의 주요 블록들이 암호화와 복호화 과정에서 하드웨어 자원을 공유할 수 있도록 설계함으로써, 면적과 전력소모가 최소화되도록 하였다. 128-b/192-b/256-b의 마스터 키 길이에 대해 라운드 변환의 전반부 4 클록 주기에 on-the-fly 방식으로 라운드 키를 생성할 수 있는 효율적인 키 스케줄링 회로를 고안하였다. Verilog HDL로 모델링된 암호 프로세서는 Xilinx FPGA로 구현하여 정상 동작함을 확인하였다. 0.35-㎛ CMOS 셀 라이브러리로 합성한 결과, 약 25,000 개의 게이트로 구현되었으며, 2.5-V 전원전압에서 220-㎒ 클록으로 동작하여 약 520-Mbits/sec의 성능을 갖는 것으로 예측되었다.

Abstract

This paper describes a design of cryptographic processor that implements the AES (Advanced Encryption Standard) block cipher algorithm "Rijndael". To achieve high throughput rate, a sub-pipeline stage is inserted into a round transformation block, resulting that two consecutive round functions are simultaneously operated. For area-efficient and low-power implementation, the round transformation block is designed to share the hardware resources for encryption and decryption. An efficient on-the-fly key scheduler is devised to supports the three master-key lengths of 128-b/192-b/256-b, and it generates round keys in the first sub-pipeline stage of each round processing. The Verilog-HDL model of the cryptoprocessor was verified using Xilinx FPGA board and test system. The core synthesized using 0.35-㎛ CMOS cell library consists of about 25,000 gates. Simulation results show that it has a throughput of about 520-Mbits/sec with 220-㎒ clock frequency at 2.5-V supply.

* 正會員, (주)한기아

(SOC Research Center, Hangia Co., Ltd.)

** 正會員, 金烏工科大學校 電子工學部

(School of Electronic Engineering, Kumoh National University of Technology)

※ 본 연구는 한국과학재단 목적기초연구(2001-1-30200-006-1) 지원에 의한 연구결과의 일부이며, 반도체설계교육센터(IDEC)의 CAD Tool 지원을 받아 이루어졌음.

接受日字:2002年3月4日, 수정완료일:2002年10月23日

Ⅰ. 서 론

암호화는 컴퓨터에 저장되어 있거나 네트워크를 통해 전달되는 정보를 제삼자가 가로채어 그 내용을 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 핵심

기술이다.[1,2]

암호 시스템은 암호화 키 (key)와 복호화 키가 동일한가에 따라 대칭형 (비밀키 방식이라고도 함) 암호 시스템과 비대칭형 (공개키 방식이라고도 함) 암호 시스템으로 구분된다. 대칭형 암호 시스템은 1977년 미국에서 표준으로 정한 DES (Data Encryption Standard)[3]가 널리 사용되고 있으나, 컴퓨터의 계산능력 향상으로 인한 안전성 저하 문제가 대두되고 있다. 2000년 10월 미국 상무부 기술표준국은 DES 보다 보안 기능이 뛰어난 차세대 암호 표준 (Advanced Encryption Standard ; AES)으로 J. Daemen과 V. Rijmen에 의해 제안된 "Rijndael" (레인달) 암호 알고리듬을 최종 AES로 선정하였다.[1,6]

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 않는 단점을 갖는다. 반면에, 하드웨어 구현은 외부의 침입자에 의한 암호 알고리듬과 암호 키의 노출 및 조작이 불가능하여 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 있으며, AES Rijndael 알고리듬 전용 ASIC 설계[7~10]와 FPGA 구현[11,12]에 관한 연구 결과들이 발표되고 있다. 전용 하드웨어 구현은 다음과 같은 세 가지 형태로 구분될 수 있다. 첫째, 범용 마이크로 컨트롤러와 암호 프로세서 코어를 결합하여 구현하는 방식은 기존의 소프트웨어 환경을 사용할 수 있어 시스템 개발 시간을 단축할 수 있다는 장점을 갖는 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 대량 생산 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.[5]

본 논문에서는 AES Rijndael 암호 알고리듬의 전용 ASIC 및 FPGA 구현을 위한 효율적인 라운드 처리부 구조와 128-b/192-b/256-b의 마스터 키 길이를 지원하는 on-the-fly 키 스케줄링 회로구조를 제안하고, 이를 Verilog-HDL로 모델링하여 설계한 후, FPGA로 구현하여 동작을 검증하였다.

Ⅱ. Rijndael 블록 암호 알고리듬[4,6]

Rijndael 암호 알고리듬은 non-Feistel 구조를 바탕으로 하고 있으며, 역 변환이 가능한 3개의 독립된 변환으로 구성된다. 블록 길이는 128-b이고, 마스터 키 길이 N_k는 128-b/192-b/256-b 중에서 선택할 수 있으며, 라운드 수 N_r는 키 길이 N_k에 따라 10/12/14로 구성된다. Rijndael 알고리듬의 암호화 과정은 <그림 1>과 같으며, 초기 라운드 키 가산, (N_r-1) 번의 반복 라운드 변환과 최종 라운드 변환의 순서로 처리된다. 최종 라운드 변환을 제외한 (N_r-1) 번의 반복 라운드는 4행$\times N_b$열 (단, $N_b=4$)로 구성되는 State에 대해 ByteSub, ShiftRow, MixColumn 및 KeyAdd 등의 변환으로 구성된다. Rijndael의 복호화는 암호화의 역순으로 이루어지며, 라운드 연산의 역 변환 (InvByteSub, InvShiftRow, InvMixColumn)이 사용되고, 라운드 키도 암호화 연산과 역순으로 사용된다.

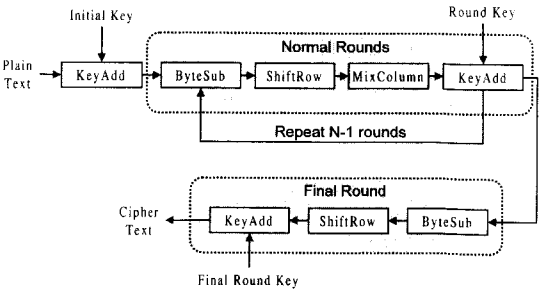


그림 1. AES Rijndael 알고리듬의 암호화 과정
Fig. 1. Encryption of AES Rijndael algorithm.

ByteSub 변환은 State를 구성하는 각각의 바이트에 대해 서로 독립적인 비선형 치환을 수행한다. 여기에 사용되는 치환 테이블을 S-box라고 하며, 이는 역 변환이 가능한 두 단계의 변환과정 즉, 유한체 (finite field) $GF(2^8)$에서 곱셈의 역원 (multiplicative

inverse)을 취하는 $x \rightarrow x^{-1}$ 매핑과 유한체 $GF(2)$에서의 affine 변환으로 구성된다. ShiftRow 변환은 State를 구성하는 4개의 행들을 행의 위치에 따라 0~3까지의 오프셋을 갖고 바이트 단위로 순환이동 (cyclic shifting) 시킨다. MixColumn 변환은 State의 행을 $GF(2^8)$의 다항식으로 생각하여 다항식 곱셈을 연산한다. 마지막으로, KeyAdd 블록은 State의 모든 바이트에 라운드 키를 가산하며, 이는 비트 단위의 XOR 연산으로 처리된다. 암·복호화 라운드에서 사용되는 라운드 키는 외부에서 입력된 마스터 키와 키 생성 알고리듬에 의해 생성된다.

Ⅲ. 회로 설계

1. 아키텍처 개요

Rijndael 알고리듬의 효율적인 하드웨어 구현을 위해 다음과 같은 사항들을 고려하였다.

첫째, Rijndael 알고리듬의 라운드 수는 마스터키의 길이 (128-b/192-b/256-b)에 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 N_r번의 라운드 연산이 반복 처리되도록 하였다.

둘째, 2장에서 언급된 바와 같이, Rijndael 알고리듬은 암호화를 위해 ByteSub, ShiftRow, MixColumn 등의 변환이 사용되고, 복호화 과정에서는 암호화의 역변환인 InvByteSub, InvShiftRow, InvMixColumn 등의 변환이 사용된다. 따라서, 기본적으로 암호화와 복호화에 독립적인 변환 블록들이 사용되므로 전체적인 하드웨어 복잡도가 커지게 된다. 본 논문에서는 라운드 처리부의 주요 블록들이 암호화와 복호화 과정에서 하드웨어 자원을 공유할 수 있도록 회로구조를 고안함으로써, 면적과 전력소모가 최소화되도록 하였다.

셋째, Rijndael 알고리듬의 라운드 연산에서 ShiftRow 변환은 State의 행 단위로 처리되고, MixColumn 변환은 열 단위로 처리된다. 본 논문에서는 라운드 연산을 전반부 처리 (ByteSub 및 ShiftRow 변환)와 후반부 처리 (MixColumn 변환, 라운드 키 가산) 두 부분으로 나누고, 이들 사이에 서브 파이프라인을 삽입함으로써 암·복호화 연산의 처리속도가 향상되도록 하였다.

넷째, 라운드 연산의 State가 4행×4-바이트의 2차원 배열로 구성되므로 16 바이트의 State를 병렬 처리하는 경우, ByteSub 변환을 위해 32개의 S-box와 MixColumn 변환을 위해 32개의 유한체 곱셈기가 필요하여 하드웨어가 매우 복잡해진다. 본 논문에서는 State를 4 바이트 단위로 병렬 처리함으로써 단일 라운드 연산이 4개 클록에 처리되도록 하였다.

다섯째, 128-b/192-b/256-b의 마스터 키 길이를 지원하기 위한 on-the-fly 키 스케줄러는 하드웨어가 복잡하고 지연경로가 길어 전체 암호 프로세서의 성능을 제한하는 요인이 된다. 본 논문에서는 라운드 변환의 전반부 4 클록 주기에 라운드 키를 on-the-fly 방식으로 생성하는 효율적인 키 스케줄링 회로를 고안하였다. 설계된 Rijndael 암호 프로세서의 전체 구조는 <그림 2>와 같으며, N_r번의 라운드 변환을 처리하는 라운드 처리부, 라운드 키 생성 블록, 그리고 제어블록 등으로 구성된다. 외부와의 인터페이스는 32-b씩 이루어지며, 라운드 처리부의 입력단에 32-b 레지스터 4개를 쉬프트 레지스터로 구성하여 4 클록동안 128-b의 평문이 입력되며, 키 생성블록의 입력단은 32-b 레지스터 8개를 쉬프트 레지스터로 구성하여 암호키 길이 지정신호에 따라 4/6/8 클록동안 128-b/192-b/256-b의 암호 키가 입력되도록 하였다.

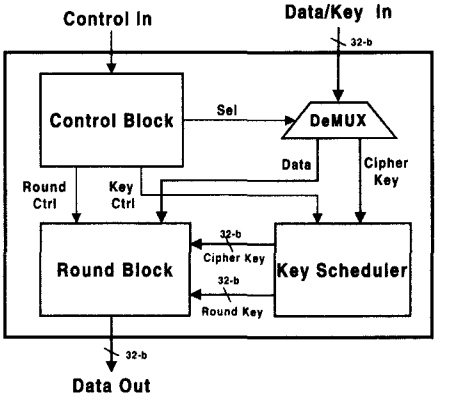


그림 2. Rijndael 암호 프로세서의 구조
Fig. 2. Architecture of Rijndael cryptoprocessor.

2. 라운드 변환 블록

1절에서 언급된 사항들을 적용하여 설계된 라운드 변환 블록의 내부 구조는 <그림 3>과 같으며, 암호화

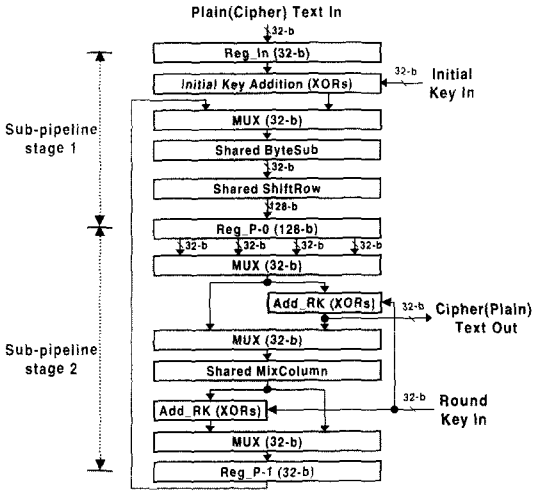


그림 3. 라운드 변환 블록
Fig. 3. Round transformation block.

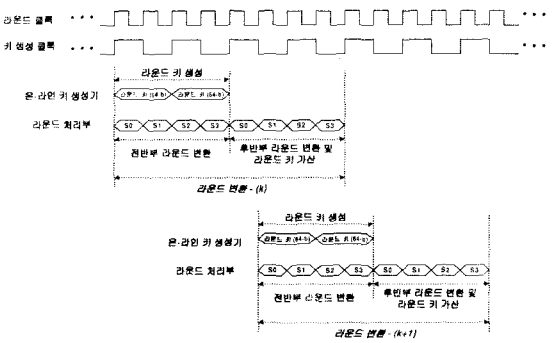
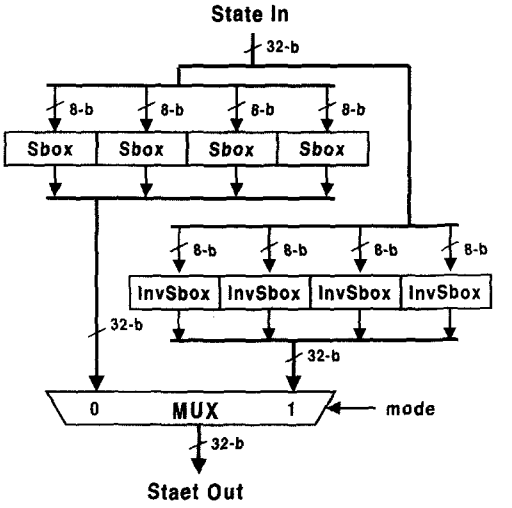


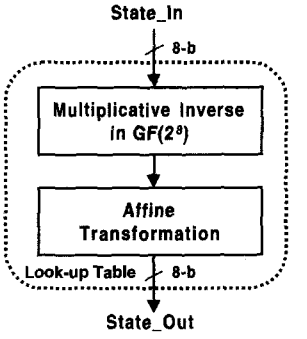
그림 4. 라운드 변환 블록의 파이프라인 동작
Fig. 4. Pipelined operation of round transformation block.

연산과 복호화 연산의 하드웨어 공유를 극대화하기 위하여 ByteSub 변환과 InvByteSub 변환을 하나의 공유 바이트서브 (Shared ByteSub) 블록으로, ShiftRow 변환화 InvShiftRow 변환을 하나의 공유 쉬프트로우 (Shared ShiftRow) 블록으로, 그리고 MixColumn 변환과 InvMixColumn 변환을 하나의 공유 믹스컬럼 (Shared MixCollum) 블록으로 구현하는 방식을 특징으로 한다. 라운드 변환을 전반부 처리 (ByteSub 변환, ShiftRow 변환)와 후반부 처리 (MixColumn 변환, 라운드 키 가산)로 나누어 서브 파이프라인을 삽입하였다. 데이터 패스는 32-b로 구성되어 4행×4-바이트의 State를 처리하는 서브 파이프라인 단은 4개의 클록으로 구현된다. <그림 4>는 라운드 처리부의 동작 타이밍도를 보인 것이며, 라운드 키는 해당 라운드의 전반부 처리가 진행되는 동안 on-the-fly 방식으로 생성되어 해당 라운드의 후반부 처리기간에 가산된다.

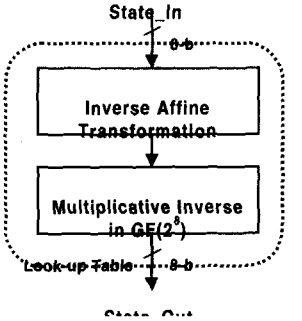
(1) S-box 공유를 이용한 Shared ByteSub 변환 블록



(a) ByteSub/InvByteSub 블록



(b) Sbox

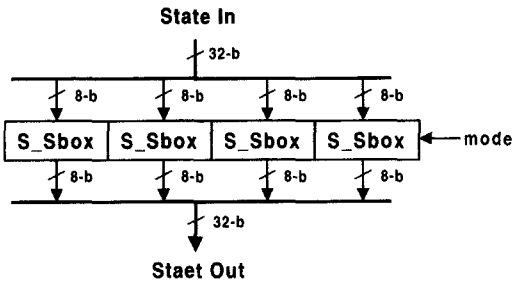


(c) InvSbox

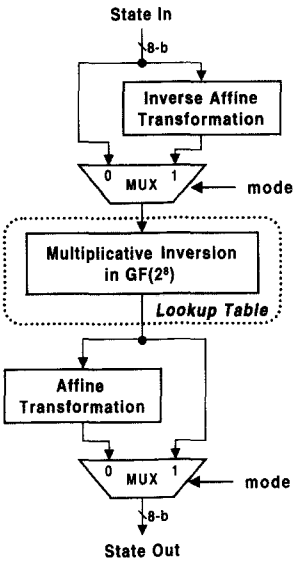
그림 5. ByteSub/InvByteSub 블록의 일반적인 구현
Fig. 5. Conventional implementation of ByteSub/InvByteSub block.

S-box는 라운드 블록의 하드웨어 복잡도에 큰 영향을 미치는 요소이므로, Rijndael 알고리듬의 하드웨어 구현에 있어서 우선적으로 고려해야 한다.

ByteSub/InvByteSub 블록을 구현하는 일반적인 방법은 <그림 5(a)>와 같이 암호화 연산을 위한 Sbox와 복호화 연산을 위한 InvSbox를 독립적으로 사용하는 것이며, 이때 Sbox와 InvSbox는 각각 <그림 5(b), (c)>와 같이 유한체 $GF(2^8)$에서 곱셈의 역원(multiplica- tive inverse) 계산과 affine 또는 inverse affine 변환을 하나의 치환 테이블 (256 바이트의 ROM에 해당 함)로 구현한다. 이와 같은 방법은 암호화 연산과 복호화 연산에 각각 4개씩 총 8개의 S-box가 필요하므로 하드웨어의 공유가 불가능하다.



(a) 본 논문의 ByteSub/InvByteSub 블록 구현



(b) Shared Sbox (S_Sbox)

그림 6. 공유 S-box (S_Sbox)를 이용한 ByteSub/InvByteSub 블록의 구현

Fig. 6. ByteSub/InvByteSub block based on Shared Sbox (S_Sbox).

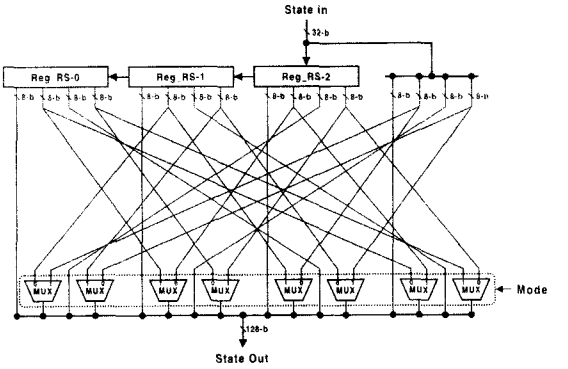


그림 7. 공유 ShiftRow 블록

Fig. 7. Shared ShiftRow block.

본 논문에서는 $GF(2^8)$에서 곱셈의 역원 연산을 lookup 테이블로 구현함으로써 암호화와 복호화 과정에서 S-box를 공유하여 사용하도록 하였다. <그림 6>은 본 논문에서 제안된 공유 Sbox (Shared Sbox)의 구조이며, $GF(2^8)$에서 곱셈의 역원을 계산하는 lookup 테이블과 affine 변환 블록으로 구성된다. 암호화 과정은 역원 계산 후에 affine 변환이 수행되며, 복호화 과정은 역 affine 변환이 먼저 수행된 후 역원이 계산된다. 본 논문에서 제안된 공유 Sbox 구조를 사용하여 ByteSub/InvByteSub 블록을 합성한 결과, 8개의 S-Box (암호화 과정에 4개, 복호화 과정에 4개)를 사용하는 일반적인 방법에서는 5,515 게이트로 구현되며, 4개의 S-Box만을 사용하는 본 논문의 방법은 3,190 게이트로 구현되어 ByteSub/InvByteSub 블록에서 약 42%의 게이트 감소가 얻어진다.

(2) 공유 ShiftRow 블록

ShiftRow/InvShiftRow 변환은 ByteSub 블록의 출력에 대한 행 단위의 순환이동으로 수행되므로, 순환기능을 갖는 쉬프트 레지스터로 구현이 가능하다. 그러나 ByteSub 변환은 열 단위로 처리되므로, State를 구성하는 16 바이트에 대한 ByteSub 변환이 모두 완료된 후에 ShiftRow 변환이 시작될 수 있다. 본 논문에서는 32-b 레지스터 3개와 출력 MUX를 이용하여 <그림 7>과 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.

(3) 공유 MixColumn 블록

MixColumn 변환은 유한체상의 행렬 곱셈으로 연산되며, 식(1)과 같이 정의되는 유한체상의 행렬 곱셈으로 연산된다.

$$B(x) = C(x) \otimes A(x) \bmod (x^4+1) \quad (1)$$

식(1)에서 기호 $\otimes$는 유한체 상의 다항식 곱셈을 나타내며, $C(x) = '02'x^3 + '01'x^2 + '01'x + '03'$이고, '1', '2', '3'은 16진수로 표현된 상수 값을 나타낸다. 한편, 식(1)은 식(2)와 같이 다시 표현할 수 있으며, 행렬계수 $C(x)$는 상수이므로, 유한체 연산의 특성을 이용하면 식(2)를 식(3)과 같이 전개할 수 있다.

$$\begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{pmatrix} \quad (2)$$

$$B_0 = '02'(A_0 \oplus A_1) \oplus (A_1 \oplus A_2 \oplus A_3) \quad (3\text{-a})$$

$$B_1 = '02'(A_1 \oplus A_2) \oplus (A_0 \oplus A_2 \oplus A_3) \quad (3\text{-b})$$

$$B_2 = '02'(A_2 \oplus A_3) \oplus (A_0 \oplus A_1 \oplus A_3) \quad (3\text{-c})$$

$$B_3 = '02'(A_3 \oplus A_0) \oplus (A_0 \oplus A_1 \oplus A_2) \quad (3\text{-d})$$

한편, InvMixColumn 변환은 식(4)와 같이 정의되며, 이를 식(5)와 같이 전개함으로써 식(3)이 나타내는 MixColumn 연산이 포함되도록 변형할 수 있다. 따라서, 본 논문에서는 암호화 연산과 복호화 연산에서 효율적으로 하드웨어를 공유할 수 있도록 공유 MixColumn (Shared MixColumn) 변환 블록을 <그림 8>과 같이 설계하였다. <그림 8>에서 MixColumn 회로는 <그림 9>와 같이 구현하였으며, xtime 연산은 $GF(2^8)$ 상의 16진수 '02'에 대한 곱셈을 의미하며, 해당 다항식에 x를 곱한 후 기약 다항식으로 modulo reduction을 취하여 구해진다.

$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{pmatrix} \quad (4)$$

$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{pmatrix} = \begin{pmatrix} 05 & 00 & 04 & 00 \\ 00 & 05 & 00 & 04 \\ 04 & 00 & 05 & 00 \\ 00 & 04 & 00 & 05 \end{pmatrix} \cdot \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{pmatrix} \quad (5)$$

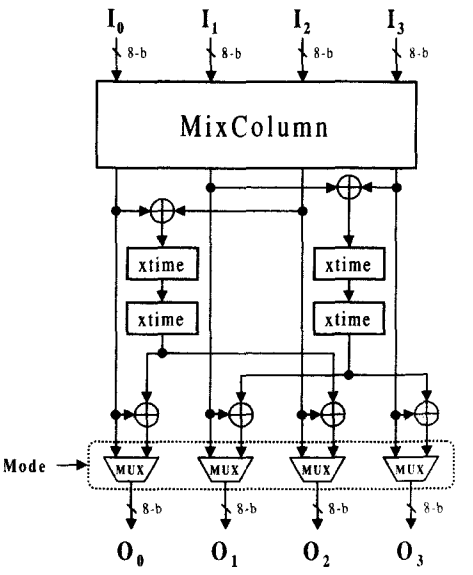


그림 8. 공유 MixColumn 블록
Fig. 8. Shared MixColumn block.

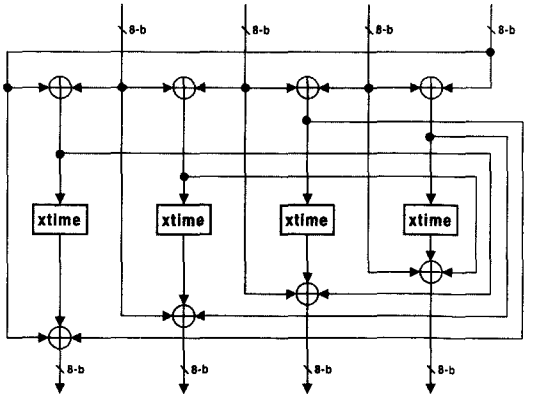


그림 9. MixColumn 회로
Fig. 9. MixColumn Circuit.

3. on-the-fly 방식의 키 스케줄러

Rijndael 암호 알고리듬에서는 외부에서 입력되는 암호 키 (cipher key)를 받아 라운드 변환에 사용되는 암호키를 생성해야 하며, 사용자에 의해 키 길이 128-b/192-b/256-b 중의 하나를 선택할 수 있다. 소프트웨어에 의한 라운드 키 생성은 키의 노출 위험과 함께 속도가 느리다는 단점을 가지므로, 하드웨어에 의한 라운드 키 생성이 요구된다. 오프-라인 생성방식은 필요한 모든 라운드 키를 한번에 생성한 후, 별도의 메모리나 레지스터에 저장하여 매 라운드마다 선택적으로 사용하는 방법이다. 한번 생성된 라운드 키는 새로운 암호

키가 입력되기 전까지 암호화뿐만 아니라 복호화에도 그대로 사용될 수 있으므로 동작속도나 전력소모 면에서 장점을 가지나, 생성된 라운드 키를 저장하기 위한 메모리나 레지스터의 부가 하드웨어를 필요로 하는 단점이 있다. 3가지 키 길이에 대한 on-the-fly 방식의 라운드 키 생성을 위해서는 키 생성부의 하드웨어가 매우 복잡해지며, 전체 암호 프로세서의 동작속도를 제한하는 요인이 되므로, 고성능 암호 프로세서의 구현을 위해서는 효율적인 라운드 키 생성 방식 및 회로구조의 고안이 필수적이다.

라운드 키 생성은 32-b/64-b/128-b 단위의 키 확장 연산을 통해 생성할 수 있다. 32-b 단위로 생성하는 경우, 하드웨어는 가장 적게 소요되나 on-the-fly 방식의 동작을 위한 라운드 키 생성에 4 클록이 필요하여 전체 성능을 제한하는 요인이 된다. 한편, 128-b 단위로 생성하는 경우는 전체 성능에 미치는 영향은 없으나 하드웨어가 많이 소요되는 단점을 갖는다. 본 논문에서는 64-b 단위의 키 생성방식을 선택함으로써 64-b씩 차이나는 3 가지의 키 길이 (128-b/192-b/256-b)를 지원하는 효율적인 키 스케줄러 구조를 제안하였으며, 이를 통해 전체적인 성능과 면적의 최적화를 이루었다.

본 논문에서 설계된 on-the-fly 방식의 키 스케줄러는 <그림 10>과 같으며, 외부에서 입력되는 키 길이 지정신호(ks), 암호/복호 동작모드신호(mode), 그리고 지정된 비트 길이의 마스터키를 받아 매 라운드 변환에 사용되는 128-b의 라운드 키를 생성한다. 라운드 키 생성기의 동작 타이밍은 <그림 4>와 같으며, 키 생성기 클록의 1 주기 당 64-b씩, 2클록 주기 (즉, 라운드 변환의 전반부 4클록 주기) 동안 128-b의 라운드 키를 on-the-fly 방식으로 생성하여 라운드 처리부로 전달하는 동작을 나타낸다. 초기 라운드 키는 모드신호에 의하여 암호키 저장 레지스터의 상위 128-b와 복호키 저장 레지스터의 하위 128-b 중에서 선택된다. 첫 번째 확장 사이클에는 모드신호와 init 신호에 의해 암호키 또는 복호키가 버스 라인으로 입력된다. 입력된 암호키/복호키는 모드에 따라 왼쪽 혹은 오른쪽으로 64-b 씩 이동하여 중간결과 레지스터(K0~K7)에 저장된다. 첫 번째 확장 사이클을 제외한 나머지 확장 사이클동안 중간결과 레지스터의 출력이 피드백 되어 키 버스에 입력되면 나머지 동작은 첫 번째 확장 사이클과 동일

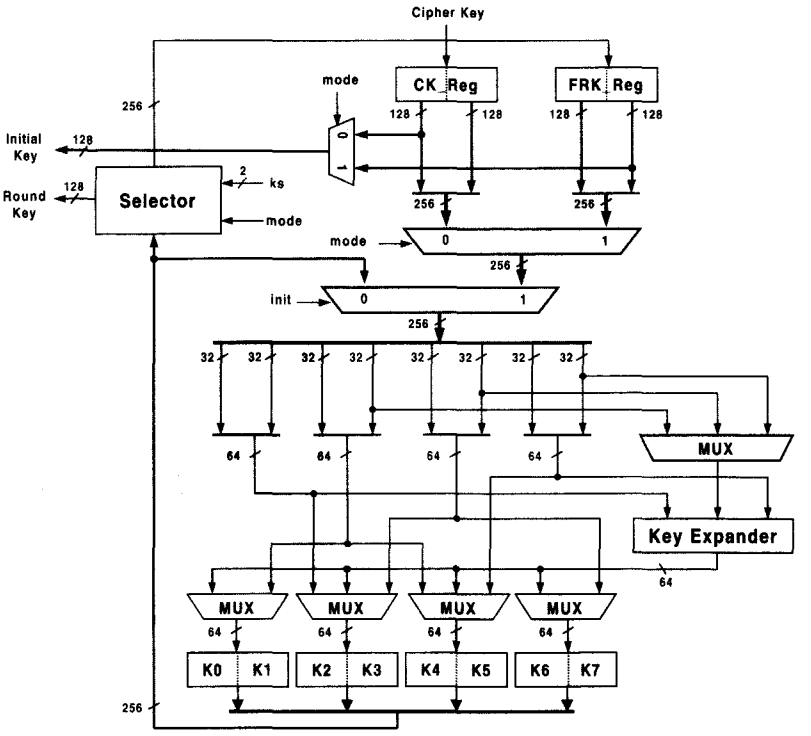


그림 10. 키 스케줄러
Fig. 10. Key scheduler.

한 과정으로 처리된다.

한편, 복호화 과정에서 사용되는 라운드 키는 외부에서 입력되는 마스터키로부터 직접 생성될 수 없으며, 먼저 암호화 동작의 마지막 라운드 키를 계산하여 이를 복호 키 레지스터 (FRK_Reg)에 저장하는 사전처리 과정이 필요하다. 이와 같은 사전처리 과정에 의해 복호 키가 결정되면, 암호화 과정과 동일하게 매 라운드 변환의 전반부 4클록 주기동안 on-the-fly 방식으로 복호화 라운드 키가 생성된다.

라운드 키 선택기(Selector)는 암호/복호 동작모드에 따라 중간결과 레지스터(K0~K7)에 저장된 4개의 64-b 값 중에서 128-b의 라운드 키 값을 선택하며, 또한 키 길이 지정신호(ks)에 의해 복호키 레지스터(FRK_Reg)에 저장될 적절한 값을 선택하는 기능을 수행한다. <그림 11>은 키 확장기 (Key Expander)의 내부 구성도이며, 키 길이 지정신호(ks)와 암호/복호 동작모드(mode)에 따라 중간결과 레지스터(K0~K7)의 값을 받아 64-b의 확장키 값을 생성한다. 내부에 4개의 Sbox로 구성되는 바이트서브(ByteSub) 블록과, 현재 라운드 값, 키 길이, 암호/복호 동작모드에 따라 32-b의 상수를 생성하는 라운드 상수 발생기(Rot&Rcon)와, 그리고 4개의 32-b XOR 회로 등의 조합회로로 구현되도록 하였다.

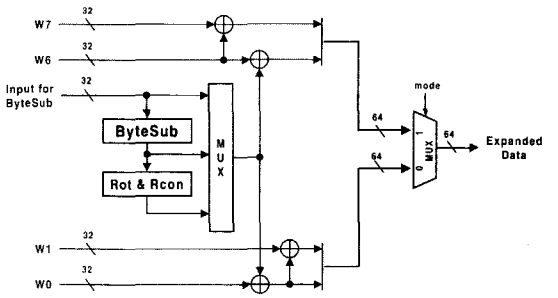


그림 11. 키 확장기
Fig. 11. Key expander.

Ⅳ. 회로설계 및 검증

설계된 Rijndael 암호 코어는 Verilog-HDL로 모델링되었으며, AES 표준 공고안[6]에 명시된 테스트 벡터를 이용하여 검증하였다. <그림 12>는 설계된 암호코어를 ModelSim 시뮬레이터로 검증한 결과이다. 128-b의 평문 "00112233445566778899AABBCCDDEEFF"와 256-b의 암호키 "000102030405060708090A0B0C0D0E0F1011121314151617181911A1B1C1D1E1F"를 입력벡터로 사용하였다. <그림 12>에서 암호화 연산의 결과로 암호문 "8EA2B7CA516745BFEAFC49904B496089"이 출력되었고, 이를 다시 복호화한 결과는 입력으로 사용된 평문과 동일한 값이 출력됨을 확인함으로써 모든 논리 기능이 정상적으로 동작함을 확인하였다.

시뮬레이션이 완료된 Verilog-HDL 모델은 최종적으로 FPGA 구현을 통해 검증하였으며, 검증 시스템은 <그림 13(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 13(b)>는 검증시스템의 실행 화면이며, 이미지 데이터를 암호화한 후 이를 다시 복호화하면 원래의 이미지와 동일한 내용이 출력됨을 확인할 수 있다. 이와 같은 FPGA 구현 검증을 통해 설계된 AES 암호 코어가 정상적으로 동작함을 확인하였다.

검증이 완료된 HDL 모델은 0.35-㎛ CMOS 셀 라이브러리와 Synopsys CAD 툴을 이용하여 회로합성을 하였다. 합성 결과, 라운드 처리부는 10,100 게이트, on-the-fly 키 스케줄러는 14,400 게이트, 그리고 제어부 및 기타로직은 300 게이트로 구현되었으며, 전체 Rijndael 암호 코어는 약 25,000게이트로 구현되었다. 설계된 암호 프로세서를 Avanti사의 Apollo tool에서 P&R을 수행한 레이아웃 도면은 <그림 14>와 같으며, 1160㎛×1148㎛의 면적으로 구현되었다.

시뮬레이션 결과, 설계된 회로의 최악 지연은 4.5-ns로서 2.5-V 전원전압에서 220-㎒로 동작 가능할 것으로 평가되었으며, 따라서 약 520-Mbits/sec의 성능을

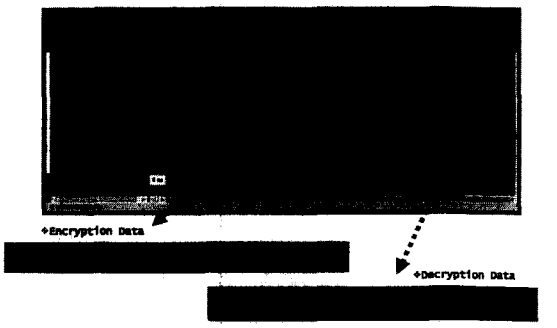
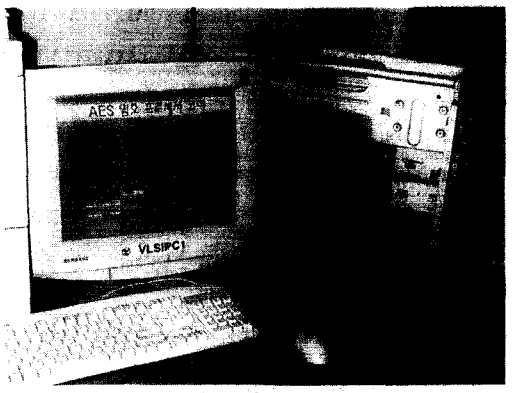
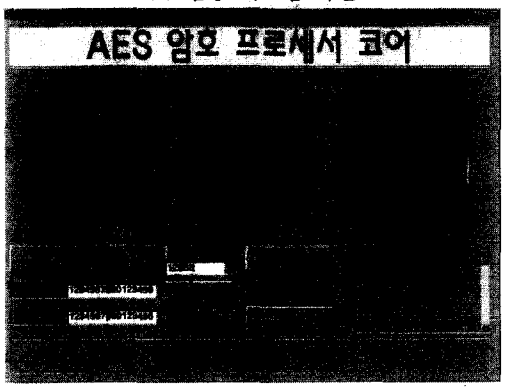


그림 12. 설계된 Rijndael 암호 코어의 시뮬레이션 결과
Fig. 12. Simulation results of the designed Rijndael cryptoprocessor.



(a) 검증 시스템 사진



(b) 이미지 데이터의 암호화 및 복호화 실행 화면

그림 13. 설계된 Rijndael 암호 프로세서의 FPGA 구현 및 검증

Fig. 13. FPGA implementation and verification of the designed Rijndael cryptoprocessor.

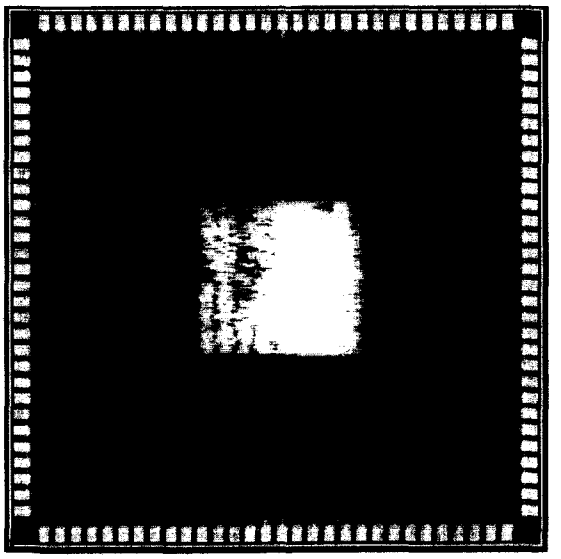


그림 14. 설계된 Rijndael 암호 프로세서의 레이아웃

Fig. 14. Layout of the designed Rijndael cryptoprocessor.

갖는 것으로 평가되었다. 설계된 Rijndael 암호 코어의 전기적 특성은 <표 1>과 같다.

<표 2>는 문헌에 발표된 Rijndael용 ASIC 구현사례의 성능을 본 논문의 설계결과와 비교한 것이다. 문헌 [7]-[9]의 설계사례는 16개의 S-Box를 사용하여 128-b씩 처리하는 회로구조를 가지며, 본 논문의 설계는 4개의 S-Box를 사용하여 32-b씩 처리하는 회로구조를 갖는다. 따라서, 본 논문에서 설계된 Rijndael 암호 프로세서 코어는 적은 칩 면적과 저전력 소모를 필요로 하는 분야에 적합할 것으로 판단된다.

표 1. 설계된 Rijndael 암호 코어의 특성

Table 1. Summary of the designed Rijndael cryptoprocessor.

암호 알고리듬		AES-128/192/256
회로 구조		단일라운드 반복 구조
게이트 수 / 최악 지연	라운드 블록	10,100 / 4.5-ns
	키 스케줄러	14,400 / 3.5-ns
	제어부	300 / 2.9-ns
	전체 코어	25,000 / 4.5-ns
동작 주파수		220-㎒ @2.5-V
칩 면적		1160㎛×1148㎛
평균 클록 수		5 클록/라운드
암·복호율		520-Mbits/sec
라운드 키 생성		on-the-fly 방식
라운드 키 setup latency		· 암호화 : 0 cycle · 복호화 : 54 cycles
데이터 입·출력		32-b

표 2. Rijndael 알고리듬용 ASIC 구현의 비교

Table 2. Comparison of ASIC implementations of Rijndael algorithm.

Reference	[7]	[8]	[9]	Our design
# of S-Box	16	16	16	4
Data block	128-b	128-b	128-b	32-b
Gate Count	1,000,000	173,000	44,300	25,000
Performance	320-Mbits /sec	910-Mbits /sec	1.28-Gbits /sec	520-Mbits/sec
Technology	0.5-㎛ Library	0.18-㎛ Library	0.5-㎛ Library	0.35-㎛ Library

V. 결 론

본 논문에서는 차세대 블록암호 표준 (AES)으로 선정된 Rijndael 암호 알고리듬용 프로세서 코어를 설계하였다. 블록 길이 128-b와 3가지 키 길이(128-b/192-b/256-b)를 지원하는 AES-128/192/256 알고리듬을 구현하였으며, 단일 라운드 변환회로를 사용하여 라운드 변환을 반복 수행하는 방식을 채택하였다. 라운드 변환을 전반부와 후반부로 나누어 서브 파이프라인을 삽입함으로써 암·복호율이 향상되도록 하였다. 라운드 처리부를 구성하는 주요 블록들이 암호화 연산과정과 복호화 연산과정에서 하드웨어 자원을 공유할 수 있도록 공유 바이트서브(Shared ByteSub)블록, 공유 쉬프트로우(Shared ShiftRow)블록, 공유 믹스컬럼(Shared MixColumn)블록의 회로구조를 제안하였으며, 이를 통해 게이트 수 감소와 저전력 특성을 갖도록 하였다. 또한, 3가지 키 길이에 대한 라운드 키를 라운드 변환의 전반부 4클록 주기에 on-the-fly 방식으로 생성하는 효율적인 회로구조 제안하였다.

설계된 Rijndael 코어는 0.35-μm CMOS 셀 라이브러리로 합성한 결과 약 25,000 개의 게이트로 구현되었으며, 2.5-V 전원전압에서 약 520-Mbits/sec의 암·복호율 성능을 갖는다. 설계된 Rijndael 암호 코어는 반도체 지적재산권인 소프트 IP (Intellectual Property)로 가공하였으며, 네트웍, 전자상거래, smart card 등을 위한 고속/고집적/저전력 보안모듈 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. Dept. of Commerce, Jan., 1977.

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

[5] 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 VLSI Systems, Vol. 9, No. 4, Aug., 2001.

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

[7] M. Bean, C. Ficke, T. Rozylowicz, and B. Weeks, "Hardware performance simulations of round 2 Advanced Encryption Standard Algorithms", http:csrc.nist.gob/encryption/aes/round2/NSA- AESfinalreport.pdf.

[8] 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.

[9] 전신우, 정용진, 권오준, "Rijndael 암호 알고리즘을 구현한 암호 프로세서의 설계", 정보보호학회 논문지, Vol. 11, No. 6, pp. 77~87, 2001. 12.

[10] 최병윤, "AES Rijndael 알고리즘용 암호 프로세서의 설계", 한국통신학회 논문지, Vol. 26, No. 10B, pp. 1491~1500, 2001. 10.

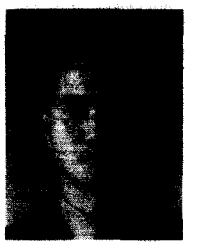
[11] M. McLoone and J.V. McCanny, "High performance single-chip FPGA Rijndael algorithm implementations", Workshop on Cryptographic Hardware and Embedded Systems 2001 (CHES 2001), pp. 65~76, May, 2001.

[12] V. Fischer and M. Drutarovsky, "Two methods of Rijndael implementation in reconfigurable hardware", Workshop on Cryptographic Hardware and Embedded Systems 2001 (CHES 2001), pp. 77~92, May, 2001.

저 자 소 개

辛 卿 旭(正會員)

1984년 2월 한국항공대학교 전자공학과 졸업 (공학사). 1986년 2월 연세대학교 대학원 전자공학과 졸업 (공학석사). 1990년 8월 연세대학교 대학원 전자공학과 졸업 (공학박사). 1990년 9월 - 1991년 6월 한국전자통신연구소 반도체연구단 선임연구원. 1991년 7월 - 현재 금오공과대학교 전자공학부 교수. 1995년 8월 - 1996년 7월 Univ. of Illinois at Urbana-Champaign 방문연구. <주관심분야 : 암호프로세서 설계, 통신 및 신호처리용 집적회로 설계, 저전압/저전력 집적회로 설계, 적외선 센서용 Readout 회로 설계>



安 河 基(學生會員)

2000년 2월 금오공과대학교 전자공학과 졸업 (공학사). 2002년 2월 금오공과대학교 대학원 전자공학과 졸업(공학석사). 2002년 3월 - 현재 (주)한기아 SOC 연구센터 연구원. <주관심분야 : 무선통신 및 신호처리용 집적회로 설계, 암호프로세서 설계, 정보보호>