논문 2008-45SD-8-11

합성체 기반의 S-Box와 하드웨어 공유를 이용한 저면적/고성능 AES 프로세서 설계

(A design of compact and high-performance AES processor using composite field based S-Box and hardware sharing)

양 현 창*, 신 경 욱**

(Hyun-Chang Yang and Kyung-Wook Shin)

요 약

다양한 하드웨어 공유 및 최적화 방법을 적용하여 저면적/고성능 AES(Advanced Encryption Standard) 암호/복호 프로세서를 설계하였다. 라운드 변환블록 내부에 암호연산과 복호연산 회로의 공유 및 재사용과 함께 라운드 변환블록과 키 스케줄러의 S-Box 공유 등을 통해 회로 복잡도가 최소화되도록 하였으며, 이를 통해 S-Box의 면적을 약 25% 감소시켰다. 또한, AES 프로세서에서 가장 큰 면적을 차지하는 S-Box를 합성체 $GF(((2^2)^2)^2)$ 연산을 적용하여 구현함으로써 $GF(2^8)$ 또는 $GF((2^4)^2)$ 기반의 설계에 비해 S-Box의 면적이 더욱 감소되도록 하였다. 64-비트 데이터패스의 라운드 변환블록과 라운드 키 생성기의 동작을 최적화시켜 라운드 연산이 3 클록주기에 처리되도록 하였으며, 128비트 데이터 블록의 암호화가 31 클록주기에 처리되도록 하였다. 설계된 AES 암호/복호 프로세서는 약 15,870 게이트로 구현되었으며, 100 MHz 클록으로 동작하여 412.9 Mbps의 성능이 예상된다.

Abstract

A compact and high-performance AES(Advanced Encryption Standard) encryption/decryption processor is designed by applying various hardware sharing and optimization techniques. In order to achieve minimized hardware complexity, sharing the S-Boxes for round transformation with the key scheduler, as well as merging and reusing datapaths for encryption and decryption are utilized, thus the area of S-Boxes is reduced by 25%. Also, the S-Boxes which require the largest hardware in AES processor is designed by applying composite field arithmetic on $GF(((2^2)^2)^2)$, thus it further reduces the area of S-Boxes when compared to the design based on $GF(2^8)$ or $GF((2^4)^2)$. By optimizing the operation of the 64-bit round transformation and round key scheduling, the round transformation is processed in 3 clock cycles and an encryption of 128-bit data block is performed in 31 clock cycles. The designed AES processor has about 15,870 gates, and the estimated throughput is 412.9 Mbps at 100 MHz clock frequency.

Keywords: Advanced Encryption Standard(AES), Security, Cryptography, Composite-field arithmetic

I. 서 론

최근 유·무선 통신 시스템의 급속한 확대에 따라 통신망을 통한 정보의 유통이 급격하게 증가하고 있으며, 이에 따라 망을 통해 유통되는 정보가 제 삼자에게 유출되거나 위·변조 되지 못하도록 하는 정보보안의 중요성이 날로 높아지고 있다. 특히, 무선 환경에서는 기지국 영역 내에 있는 모든 단말기들이 다른 사람의 정보를 수신할 수 있으므로, 허가된 수신자 이외에 다른 사람이 정보를 보지 못하게 하는 데이터 기밀성과 사용자 인증 등 정보보안 기술이 필수적으로 요구된다. 정

* 학생회원, ** 정회원, 금오공과대학교 전자공학부
(School of Electronics Engineering, Kumoh National Institute of Technology)

※ 본 논문은 2007년도 금오공과대학교 학술연구비의 지원으로 수행된 연구결과임.
접수일자: 2008년4월2일, 수정완료일: 2008년7월17일

보보안을 위한 가장 기본적인 방법은 유통되는 정보를 제 삼자가 알아보지 못하게 하는 암호화 기술이다.

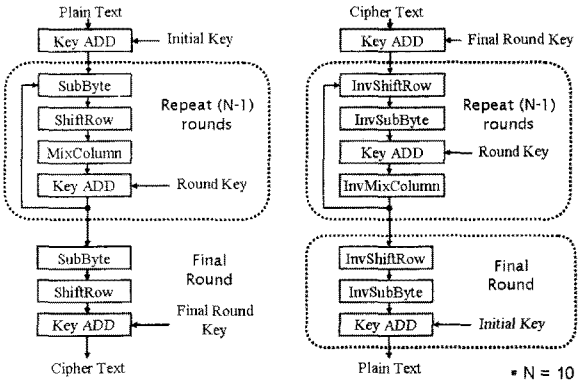
2000년 이전까지는 DES(Data Encryption Standard)가 국제표준 블록암호 알고리듬으로 널리 사용되어 왔다. 그러나 컴퓨터의 계산능력 향상으로 인해 DES의 안전성이 더 이상 보장받을 수 없게 되자 미국 NIST (National Institute of Standards and Technology)는 AES(Advanced Encryption Standard)를 새로운 블록암호 표준으로 채택하였다[1~2]. AES는 강력한 보안성능과 다양한 운영모드를 제공하므로 무선 랜, 와이브로, Zigbee, 무선 USB 등 다양한 유·무선 통신 시스템의 보안 알고리듬으로 폭넓게 사용되고 있다.

암호 알고리듬은 소프트웨어 또는 하드웨어로 구현이 가능하나, 고속 데이터 처리가 요구되는 통신망에서 실시간 처리를 위해서는 암호 알고리듬의 하드웨어 구현이 필수적이다. 본 논문에서는 무선 환경에서 실시간으로 암호/복호를 수행하는 고속/저면적의 AES 코어를 설계하였다. AES의 하드웨어 구현 시 가장 큰 면적을 필요로 하는 SubByte/InvSubByte 연산을 합성체 연산 기반의 효율적인 방법으로 구현하였으며, 키 생성기와 SubByte 연산에 이용되는 S-Box를 공유하여 기존의 방식 보다 면적을 크게 감소시켰다.

Ⅱ. AES 알고리듬

대칭키 블록암호 알고리듬인 AES[1~2]는 블록길이가 128비트이고, 키 길이는 128/192/256비트 중에서 선택할 수 있으며, 키 길이에 따라 10/12/14번의 반복 라운드 변환을 갖는다. AES의 암호/복호 연산은 그림 1과 같으며, 초기 라운드 키 가산 후, 9번의 반복 라운드 변환과 최종 라운드 변환의 과정으로 처리된다. 라운드 변환은 4바이트×4바이트로 구성되는 데이터(state)에 대해 SubByte, ShiftRow, MixColumn 및 KeyAdd 등의 변환연산으로 구성된다. 복호화는 암호화의 역순으로 처리되며, 역변환 함수인 InvSubByte, InvShiftRow, InvMixColumn이 사용되고, 라운드 키도 암호화 과정의 역순으로 사용된다.

SubByte/InvSubByte 변환은 바이트 단위의 비선형 치환으로 구현되며, 비선형 치환은 Affine/InvAffine 변환과 유한체(finite field) $GF(2^8)$ 상의 곱의 역원(multiplicative inverse) 연산으로 계산된다. ShiftRow/InvShiftRow 변환은 state의 첫째 행을 제외한 나머지



(a) Encryption (b) Decryption

그림 1. AES 암호/복호 알고리듬
Fig. 1. AES encryption/decryption algorithm.

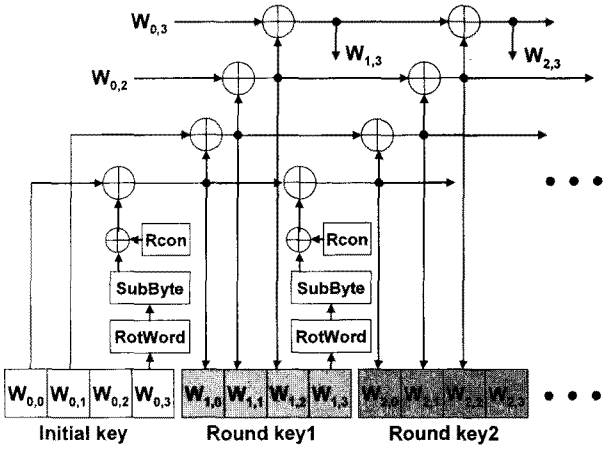


그림 2. AES의 라운드 키 확장
Fig. 2. Round key expansion of AES.

행들을 서로 다른 offset으로 바이트 단위로 순환이동 시키는 연산이다. MixColumn/InvMixColumn 변환은 $GF(2^8)$ 상의 행렬연산으로 구현되며, KeyAdd는 state에 바이트 단위의 라운드 키를 가산하는 과정이며 비트 단위 XOR 연산으로 처리된다. 라운드 변환에 사용되는 라운드 키는 외부에서 입력되는 초기키로부터 키 확장 알고리듬에 의해 생성된다.

AES의 라운드 키 확장은 그림 2와 같이 구현되며, 초기키로부터 RotWord, SubByte, Rcon 가산 등의 연산을 거쳐 다음 라운드의 키가 생성된다. RotWord는 1바이트 왼쪽 순환이동 연산이며, SubByte는 암호화 라운드 변환의 SubByte 연산과 동일한 바이트 단위 치환 연산이다. Rcon 가산은 라운드별로 고정된 상수 값과의 XOR 연산이다.

그림 2에서 보는 바와 같이, AES의 $(n+1)$-번째 라운드 키는 n-번째 라운드 키로부터 생성된다. n번째

라운드 키의 네 번째 워드($W_{n,3}$)에 대해 RotWord, SubByte (S-Box), Rcon 가산 등의 연산이 수행된 후, $W_{n,0}$와 XOR 연산을 통해 $(n+1)$-번째 라운드 키의 첫 번째 워드($W_{n+1,0}$)가 생성된다. $W_{n,1}$, $W_{n,2}$, $W_{n,3}$과 $W_{n+1,0}$, $W_{n+1,1}$, $W_{n+1,2}$를 각각 XOR 연산하여 라운드 키의 나머지 워드($W_{n+1,1}$, $W_{n+1,2}$, $W_{n+1,3}$)가 생성된다.

Ⅲ. 회로 설계

설계된 AES 암호/복호 프로세서는 10번의 라운드 변환을 처리하는 라운드 변환 블록, 라운드 키 생성기, 그리고 제어블록 등으로 구성되며, 그림 3의 구조를 갖는다. 외부와의 인터페이스는 64비트씩 이루어지며, 라운드 변환 블록의 데이터 패스를 64비트로 구현하여 라운드 변환이 3 클록주기에 처리되도록 하였다. 그림 4는 암호화 과정의 동작 타이밍 도이다. ldkey_en 신호에 의해 128비트의 암호키가 2 클록주기 동안 입력되고, ldtext_en 신호에 의해 128비트의 평문(암호문)이 2 클록주기 동안 입력된다. 그 후, 10번의 라운드 변환에 30 클록주기가 소요된 후, 암호문(평문)이 2 클록주기 동안 출력된다.

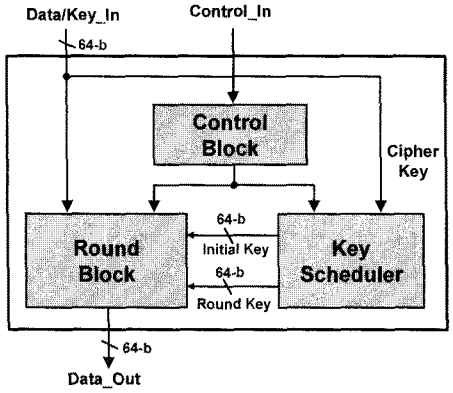


그림 3. AES 암호/복호 프로세서의 구조
Fig. 3. Architecture of AES encryption/decryption processor.

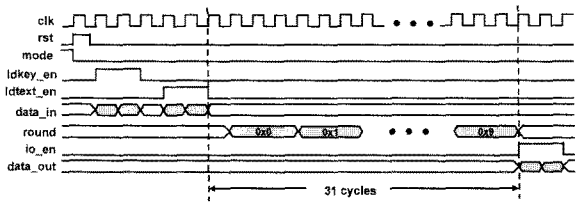


그림 4. AES 프로세서의 동작 타이밍 도 (암호화)
Fig. 4. Timing diagram of the AES processor (encryption).

1. 하드웨어 공유 구조의 라운드 변환 블록

그림 1에서 보는 바와 같이, AES의 라운드 변환은 암호과정과 복호과정에서 서로 역 변환 관계를 갖는 변환함수들이 사용되며, 변환함수들의 연산순서도 반대이다. 그러나 다음과 같은 AES의 알고리듬 특성을 이용하면 암호/복호 공유구조의 라운드 변환회로를 설계할 수 있다. 첫째, 바이트 단위의 순환이동인 ShiftRow/InvShiftRow 연산과 바이트 단위의 치환인 SubByte/InvSubByte 연산은 순서를 바꾸어도 동일한 결과가 얻어진다[2]. 둘째, ShiftRow 연산과 InvShiftRow 연산은 순환이동 방향만 반대이므로 MUX 회로를 사용하여 Shared ShiftRow 구조로 구현할 수 있다. 셋째, MixColumn 연산과 InvMixColumn 연산의 수식을 분석해 보면, InvMixColumn이 MixColumn 연산을 포함하는 Shared MixColumn 구조로 구현될 수 있다. 넷째, SubByte와 InvSubByte 연산은 암호연산과 복호연산의 하드웨어 공유를 최대화하기 위하여 Shared SubByte 구조로 구현될 수 있으며, 이에 대해서는 다음 절에서 상세히 논의한다. 이와 같은 분석을 토대로, 본 논문에서는 그림 5와 같이 암호/복호 연산을 선택적으로 처리할 수 있는 하드웨어 공유 및 재사용 구조의 라운드 변환 블록을 설계하였다.

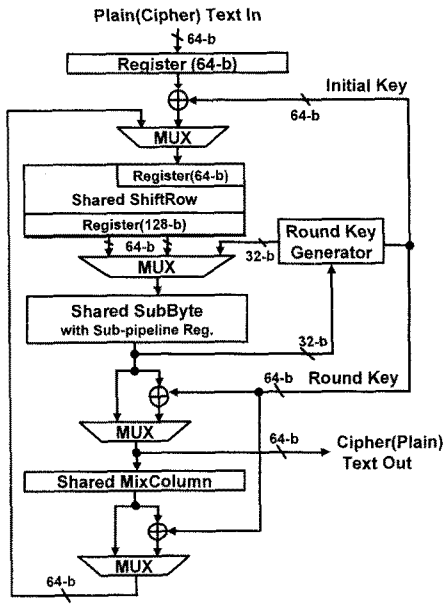


그림 5. 하드웨어 공유를 이용한 라운드 변환 블록
Fig. 5. Round transformation block using hardware sharing.

2. 효율적인 S-Box 구현

바이트 단위의 비선형 치환 연산인 SubByte/InvSubByte 연산은 AES 프로세서에서 가장 큰 하드웨어 면적을 차지하는 부분이며, 따라서 설계 최적화를 위한 다양한 고려가 필요하다.

가장 기본적인 방법은 SubByte/InvSubByte 연산을 256 바이트의 LUT (Look-Up Table)로 구현하는 것이다(방법-①). 암호연산을 위한 S-Box와 복호연산을 위한 InvS-Box가 각각 독립된 LUT로 구현되어야 하고, 64-비트 데이터 패스의 경우 라운드 블록에 8개의 LUT가 필요하므로 총 2 KBytes의 큰 LUT가 필요하다. 한편, SubByte와 InvSubByte 연산을 각각 LUT+Affine과 InvAffine+LUT로 계산하는 방법[3]도 제안되었다(방법-②). 여기서 LUT는 유한체 $GF(2^8)$상의 곱의 역원 값을 저장하므로 암호화와 복호화에 공통으로 이용될 수 있으며, 따라서 총 8개의 LUT와 Affine/InvAffine 변환기로 구성된다. 방법-①에 비해 LUT의 크기가 반으로 감소하나, 여전히 큰 하드웨어를 필요로 한다. 방법-②의 LUT 크기를 더욱 줄이기 위해 $GF(2^8)$을 $GF((2^4)^2)$으로 변환하는 합성체 연산[4~5]방법이 제안되었다(방법-③). 이 방법은 $GF(2^4)$ 상의 곱의 역원을 LUT로 저장하므로 4바이트의 LUT 8개와 단순한 조합 논리회로로 S-box/InvS-box가 구현된다.

본 논문에서는 체 변환을 이용하여 $GF(2^8)$을 $GF(((2^2)^2)^2)$으로 변환[6~7]한 후, $GF((2^2)^2)$ 상의 유한체 곱셈을 통해 회로를 최적화하는 방법을 적용하여 설계하였다. AES 알고리듬에 사용되는 유한체 (finite field) $GF(2^8)$는 식(1)과 같이 $GF(2)$와 기약다항식 (irreducible polynomial) $P_0(x)$, $P_1(x)$, $P_2(x)$를 이용하여 합성체 (composite field) $GF(((2^2)^2)^2)$로 변환될 수 있다. 식(1)에서 $\phi \in GF(2^2)$와 $\lambda \in GF((2^2)^2)$는 $P_1(x)$, $P_2(x)$가 각각 $GF(2^2)$과 $GF((2^2)^2)$ 상에서 irreducible 조건을 만족한다[7]. 따라서 ϕ, λ는 식(2)에 주어진 16가지 조합 중 하나가 되며, 본 논문에서는 $\phi = \{10\}_2$, $\lambda = \{1100\}_2$을 선택하였다.

$$\begin{cases} GF(2) \Rightarrow GF(2^2), & P_0(x) = x^2 + x + 1 \\ GF(2^2) \Rightarrow GF((2^2)^2), & P_1(x) = x^2 + x + \phi \\ GF((2^2)^2) \Rightarrow GF(((2^2)^2)^2), & P_2(x) = x^2 + x + \lambda \end{cases} \quad (1)$$

$$\begin{array}{ll} \phi = \{10\}_2 & \lambda = \{1000\}_2,\ \lambda = \{1100\}_2 \\ \phi = \{11\}_2 & \lambda = \{1001\}_2,\ \lambda = \{1101\}_2 \\ & \lambda = \{1010\}_2,\ \lambda = \{1110\}_2 \\ & \lambda = \{1011\}_2,\ \lambda = \{1111\}_2 \end{array} \quad (2)$$

$GF((2^4)^2)$상의 원소를 $(P_H x + P_L)$로 표현하면 (단, $P_H, P_L \in GF(2^4)$) x가 $P_2(x)$의 근일 때, 확장 유클리디안 알고리듬에 의해 $GF((2^4)^2)$상의 곱의 역원은 식(3)과 같이 표현된다.

$$\begin{aligned} [P_H x + P_L]^{-1} = P_H [P_H^2 \lambda + (P_H + P_L) P_L]^{-1} x \\ + (P_H + P_L) [P_H^2 \lambda + (P_H + P_L) P_L]^{-1} \end{aligned} \quad (3)$$

식(3)에 의하면, $GF(2^8)$ 상의 곱의 역원은 합성체 $GF(((2^2)^2)^2)$를 이용하여 그림 6과 같이 구현된다. 그림 6에서 보는 바와 같이, $GF(((2^2)^2)^2)$ 상의 곱의 역원은 $GF((2^2)^2)$ 상의 유한체 곱셈기 3개와 $GF(2^4)$ 상의 곱의 역원회로(x^{-1}), 제곱(x^2) 및 XOR 연산으로 구현된다. $GF((2^2)^2)$ 상의 유한체 곱셈은 $GF(2^2)$ 상의 유한체 곱셈기 3개와 XOR 게이트로 구현되며, 또한 $GF(2^2)$ 상의 곱셈기는 3개의 AND 게이트와 4개의 XOR로 구현된다. 한편, $GF(2^4)$ 상의 곱의 역원인 x^{-1} 연산은 수식을 직접 하드웨어로 구현하는 방법과 LUT로 구현하는 방법이 있으나, LUT로 구현하는 것이 유리한 것으로 평가되었으며, 본 논문에서는 표 1과 같이 8바이트의 LUT로 구현하였다.

그림 7은 합성체 $GF(((2^2)^2)^2)$ 연산을 기반으로 설계된 shared S-Box 회로이다. 암호연산의 경우, 데이터는 체 변환 행렬 δ에 의해 $GF(2^8)$에서 $GF(((2^2)^2)^2)$으로 변환된 후 곱의 역원이 계산되고, 체 역변환 행렬 δ^{-1}에 의해 다시 $GF(2^8)$로 변환되어 Affine 연산에 의

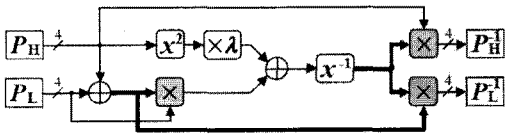


그림 6. $GF(((2^2)^2)^2)$를 이용한 곱의 역원 회로
Fig. 6. Multiplicative inverse circuit based on $GF(((2^2)^2)^2)$.

표 1. $GF(2^4)$상의 곱의 역원
Table 1. Multiplicative inverse on $GF(2^4)$.

Input	Inversion	Input	Inversion
0000	0000	1000	1010
0001	0001	1001	0110
0010	0011	1010	1000
0011	0010	1011	1001
0100	1111	1100	0101
0101	1100	1101	1110
0110	1001	1110	1101
0111	1011	1111	0100

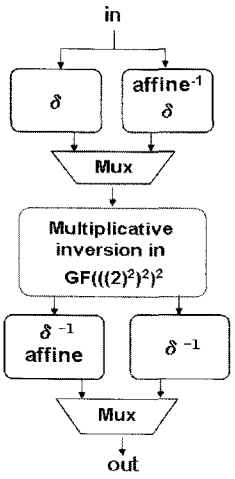


그림 7. $GF(((2^2)^2)^2)$ 기반의 shared S-Box 구현
Fig. 7. Shared S-Box based on $GF(((2^2)^2)^2)$.

해 SubByte 연산이 계산된다. 복호연산의 경우, InvAffine 행렬연산 후, $GF(2^8)$에서 $GF(((2^2)^2)^2)$으로 체 변환되어 곱의 역원이 계산되고, 체 역변환 행렬 δ^{-1}에 의해 다시 $GF(2^8)$로 변환되어 InvSubByte 연산이 완료된다. $GF(2^8)$과 $GF(((2^2)^2)^2)$ 사이의 체 변환 행렬은 식(4)와 같으며, XOR 게이트를 이용한 단순 조합논리회로로 구현된다. 체 변환 (δ와 δ^{-1})과 곱의 역원 LUT는 암호/복호 연산에 공통으로 사용되고, Affine/InvAffine 변환은 암호/복호에 선택적으로 적용된다. 본 논문에서는 critical path 지연의 최소화를 위해 δ^{-1}+Affine과 InvAffine+δ을 결합하여 구현하였다.

$$\delta=\begin{pmatrix}11000010\\01001010\\01111001\\01100011\\01110101\\00110101\\01111011\\00000101\end{pmatrix}\quad \delta^{-1}=\begin{pmatrix}10101110\\00001100\\01111001\\01111100\\01101110\\01000110\\00100010\\01000111\end{pmatrix} \tag{4}$$

3. SubByte 블록과 키 생성기의 S-Box 공유

AES 암호 알고리듬에서 라운드 키는 오프라인 방식 또는 온라인 방식으로 생성될 수 있다. 오프라인 방식의 라운드 키 생성은 모든 라운드 키를 미리 생성하여 별도의 메모리에 저장한 후, 매 라운드마다 선택하여 사용하는 것이며, 부가 하드웨어를 필요로 하는 단점이 있다. 본 논문에서는 라운드 연산에서 필요한 라운드 키를 즉석으로 생성하여 사용하는 온라인 방식을 채택하였다. 그림 2의 라운드 키 생성 알고리듬에 의하면, 라운드 키 생성기는 4×256 바이트의 S-Box가 필요하다. 라운드 변환 블록의 데이터 패스를 64비트로 구현하는 경우, 라운드 키 생성기를 포함하여 총 12×256 바이트의 S-Box가 필요하다(그림 8-(a)).

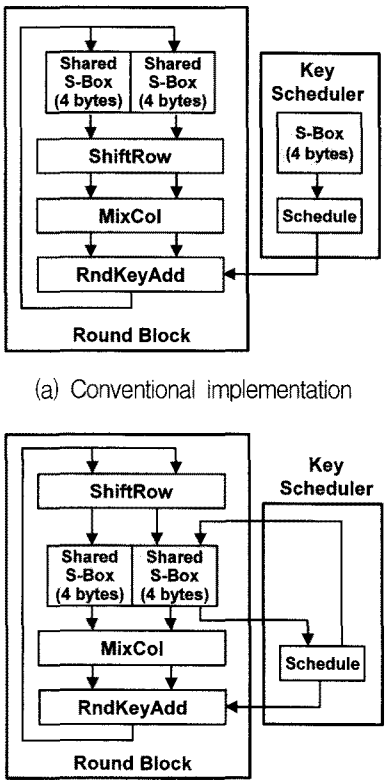


그림 8. S-Box 공유를 통한 하드웨어 최적화
Fig. 8. Hardware optimization based on S-Box sharing.

표 2. S-Box 구현방법의 게이트 수 비교*㈜
Table 2. Comparison of gate count for S-Box designs.

	그림 8-(a) 방법	본 논문의 방법
Shared S-Box (라운드변환 블록)	4,248 게이트	4,248 게이트
S-Box (키 생성기)	1,428 게이트	0
합 계	5,676 (1.0)	4,248 (0.75)

㈜ 64-비트 데이터 패스의 경우

본 논문에서는 그림 8-(b)와 같이 라운드 키 생성기에 별도의 S-Box를 사용하지 않고, 라운드 변환블록의 S-Box를 공유하도록 설계함으로써 4×256 바이트의 S-Box를 제거하였다. 라운드 변환블록에는 SubByte와 InvSubByte를 선택적으로 수행하는 그림 7의 Shared S-Box가 사용되며, 키 생성기에는 SubByte만 수행하는 S-Box가 사용된다. 표 2는 합성된 S-Box의 게이트

수를 보인 것이며, 본 논문의 방법은 그림 8-(a)의 일반적인 방법에 비해 25% 적은 게이트로 구현되었다.

Ⅳ. 설계검증 및 성능평가

그림 9는 설계된 AES 프로세서의 RTL 수준 시뮬레이션 결과를 보인 것이다. 128-비트의 평문 "00112233 44556677 8899aabb ccddeeff"를 128-비트의 암호 키 "00010203 04050607 08090a0b 0c0d0e0f"로 암호화한 결과, 암호문 "69c4e0d8 6a7b 0430 d8cdb780 70b4c55a"가 출력되었다. 암호문을 동일한 암호키로 복호한 결과, 암호과정에 입력된 평문과 동일한 값이 출력되어 설계된 AES 프로세서의 논리기능이 정상적으로 동작함을 확인하였다.

시뮬레이션을 통해 기능검증이 완료된 AES 프로세서는 최종적으로 FPGA 구현을 통해 동작을 확인하였다. Xilinx Spartan-3 FPGA 디바이스와 UART 시리얼 포트로 구성된 보드를 검증에 사용하였으며, Visual C++로 테스트 프로그램을 작성하여 PC와 UART 시리얼 통신을 통해 설계된 AES 프로세서의 동작을 검증하였다. 그림 10은 검증 시스템의 실행화면이다. 실행화면의 좌측에 표시된 평문과 128-비트의 암호키를 입력하여 암호화를 실행한 결과, 실행화면의 중앙부에 표시된 암호문이 출력되었다. 출력된 암호문을 동일한 암호키로 복호화 한 결과, 실행화면의 우측에 표시된 복호문이 출력되었으며, 이는 암호화 과정에서 입력된 평문과 동일함을 확인할 수 있다. 이와 같은 FPGA 구현 검증을 통해 설계된 AES 프로세서가 정상적으로 동작함을 확인하였다.

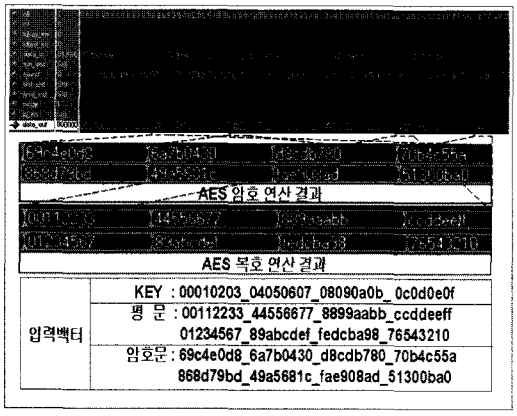


그림 9. 설계된 AES 프로세서의 기능검증 결과
Fig. 9. Simulation result of the designed AES processor.

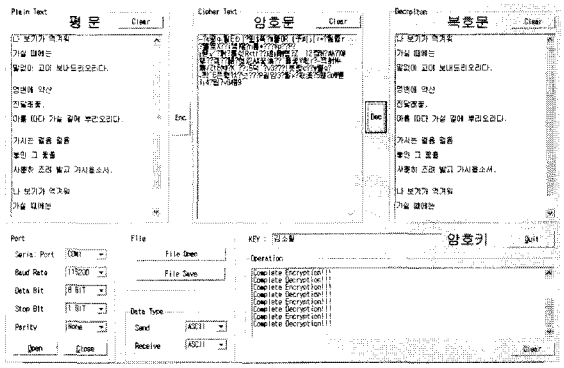


그림 10. AES 프로세서의 FPGA 구현 검증 결과
Fig. 10. FPGA verification result of AES processor.

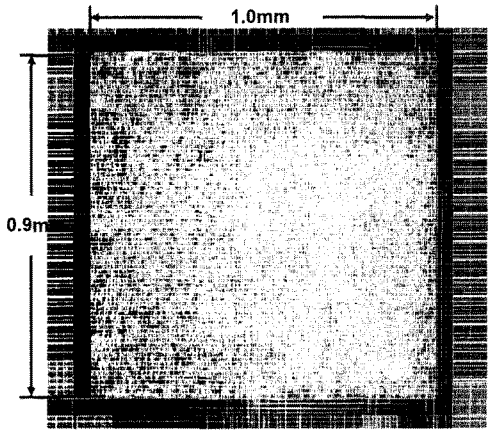


그림 11. 설계된 AES 프로세서의 레이아웃 도면
Fig. 11. Layout of the designed AES processor.

검증이 완료된 HDL 모델을 0.35-㎛ 셀 라이브러리를 이용하여 합성한 결과 15,870 게이트로 구현되었다. 타이밍 분석 결과, 설계된 AES 프로세서의 최대 지연시간은 9.04-ns로 나타나 100 MHz로 동작 가능하며, 412.9 Mbps의 성능을 갖는 것으로 평가되었다. 합성 후 P&R을 수행하였으며, 그림 11은 AES 프로세서의 레이아웃 도면을 보인 것이다. 코어부분의 면적은 약 1.0×0.9-㎟이고, 코어 이용률(utilization)은 약 86%로 나타났다. 설계된 AES 프로세서의 특성은 표 3과 같다.

표 4는 본 논문의 설계결과를 문헌에 발표된 사례와 비교한 것이다. 참고 문헌 [8]과 참고 문헌 [9]의 설계는 40-48개의 S-Box를 사용하여 128비트씩 처리하는 방식이며, 본 논문과 참고 문헌 [6]의 설계는 8개의 S-Box를 사용하여 64비트씩 처리하는 방식이다. 본 논문에서 설계된 AES 프로세서는 참고 문헌 [8]과 참고 문헌 [9]에 비해 매우 작은 게이트 수로 구현되었으며, 참고 문헌 [6]과 비슷한 게이트 수를 갖는다. 참고 문헌 [6]은 0.11-um 공정으로 구현되어 875 Mbps의 성능을

표 3. 설계된 AES 프로세서의 특성
Table 3. Features of the designed AES processor.

구분	성능
게이트 수	15,870
동작 주파수	100 MHz
평균 클럭 수	3 클록/라운드
동작 성능	412.9 Mbps
데이터 패스	64-비트
공정	0.35-㎛
코어 사이즈	1.0x0.9 ㎟
셀 사용율	86 %

표 4. AES 알고리듬 구현사례 비교
Table 4. Comparison of AES implementation cases.

문헌	[6]	[8]	[9]	본 논문
No. of S-Box	8	48	40	8
Cycles/round	3	1	1	3
Max. Freq. [MHz]	218.82	100	NA	100
성능 [Mbps]	875.28	1,820	509.7	412.9
게이트 수	14,777	173,000	33,850	15,870
공정	0.11-㎛	0.18-㎛	0.35-㎛	0.35-㎛

가지나, 본 논문의 설계도 유사한 공정으로 구현하는 경우에 비슷한 성능을 가질 것으로 예상된다.

V. 결 론

본 논문에서는 AES 암호/복호 프로세서의 효율적인 설계에 대하여 기술하였다. 저면적 구현을 위해 다양한 하드웨어 공유 및 최적화 방법을 적용하여 설계하였다. 합성체 $GF(((2^2)^2)^2)$ 기반의 연산방식을 적용하여 S-Box를 구현하였으며, 또한 라운드 변환 블록과 키 생성기의 S-Box 공유와 암호/복호 회로의 공유 및 재사용 등을 통해 면적의 최적화를 이루었다. 64-비트 데이터 패스를 갖도록 설계된 AES 프로세서는 15,870 게이트로 구현되었으며, 0.35-㎛ 셀 라이브러리 공정에서 100 MHz로 동작하여 412.9 Mbps의 성능을 갖는 것으로 평가되었다. 0.13-㎛ 급의 공정으로 구현한다면 800 Mbps 이상의 성능이 예상되며, 따라서 무선 랜, 와이브로, 무선 USB 등 유·무선 통신 시스템의 보안 하드웨어에서부터 고성능 보안 서버의 구현에 사용될 수 있을 것이다.

감사의 글

반도체설계교육센터(IDEC)의 CAD Tool 지원에 감사드립니다.

참 고 문 헌

[1] J. Daemen and V. Rijmen, "AES Proposal : Rijndael Block Cipher", *NIST Document ver.2*, http://www.nist.gov/aes, Mar., 1999.
[2] FIPS Publication 197, "Advanced Encryption Standard (AES)," U.S. Doc/NIST
[3] 안하기, 신경욱, "AES Rijndael 블록 암호 알고리듬의 효율적인 하드웨어 구현", *한국 정보보호학회 논문지*, 제12권 2호, pp. 53-64, 2002.
[4] 황석기, 김종환, 신경욱, "IEEE 802.11i 무선 랜 보안을 위한 AES 기반 CCMP 코어 설계", *한국통신학회 논문지*, 제31권 제6A호, pp. 640-647, 2004.
[5] V. Rijndael, "Efficient implementation of the Rijndael S-Box", http://www.esat.kuleuven.ac.be/~rijnmen/rijndael/sbox.pdf.
[6] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A compact Rijndael hardware architecture with S-box optimization", *Proc. ASIACRYPT 2001*, pp. 239-254, Dec. 2001.
[7] Xinmiao Zhang, Keshab K. Parhi, "High-Speed VLSI Architectures for the AES Algorithm", *IEEE Trans. Systems.*, vol. 12, no. 9, Sep., 2004.
[8] H. Kuo and I. Verbauwhede, "Architectural optimization for a 1.82 Gbits/sec VLSI implementation of the AES Rijndael Algorithm", *Workshop on Cryptographic Hardware and Embedded Systems 2001 (CHES 2001)*, pp.53-67, May, 2001.
[9] T. Ichikawa, T. Tokita, and M. Matsui, "On Hardware Implementation of 128-bit Block Ciphers (III)", *2001 Symp. on Cryptography and Information Security (SCIS 2001)*, pp. 669-674, Jan., 2001.

저 자 소 개

양 현 창(학생회원)

2006년 금오공과대학교 전자공학부 졸업 (공학사)

2008년 8월 금오공과대학교 대학원 전자공학과 졸업 (공학석사)

<주관심분야 : 암호 알고리듬, 시스템 및 네트워크 보안, 영상처리>

신 경 욱(정회원)

1984년 한국항공대학교 전자공학과 공학사

1986년 연세대학교 전자공학과 공학석사

1990년 연세대학교 전자공학과 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)

<주관심분야 : 통신 및 신호처리용 SoC 설계, 정보보호 SoC 설계, 반도체 IP 설계>