通信情報保護學會論文誌
第 11 券 第 1 號, 2001. 2

128비트 SEED 암호 알고리즘의 고속처리를 위한 하드웨어 구현*

전 신 우**, 정 용 진***

High Performance Hardware Implementation of the 128-bit SEED Cryptography Algorithm

Shin-woo Jeon**, Yong-jin Jeong***

요 약

본 논문에서는 우리 나라 128 비트 블록 암호 알고리즘 표준인 SEED를 하드웨어로 구현하였다. 먼저 하드웨어 구현 측면에서 SEED를 같은 비밀키 블록 암호 알고리즘으로 AES 최종 후보 알고리즘인 MARS, RC6, RIJNDAEL, SERPENT, TWOFISH와 비교 분석하였다. 동일한 조건하에서 분석한 결과, SEED는 MARS, RC6, TWOFISH보다는 암호화 속도가 빨랐지만, 가장 빠른 RIJNDAEL보다는 약 5배정도 느렸다.

이에 속도 측면에서 우수한 성능을 가질 수 있는 고속 SEED 구조를 제안한다. SEED는 동일한 연산을 16번 반복 수행하므로 1라운드를 J1 함수 블록, J2 함수 블록, key mixing 블록을 포함한 J3 함수 블록의 3단계로 나누고, 이를 파이프라인 시켜 더 빠른 처리 속도를 가지도록 하였다. G 함수는 구현의 효율성을 위해 4개의 확장된 4바이트 SS-box들의 xor로 처리하였다. 이를 Verilog HDL을 사용하여 ALTERA FPGA로 검증하였으며, 0.5um 삼성 스탠다드 셀 라이브러리를 사용할 경우 파이프라인이 가능한 ECB 모드의 암호화와 ECB, CBC, CFB 모드의 복호화 시에는 384비트의 평문을 암복호화하는데 총 50클럭이 소요되어 97.1MHz의 클럭에서 745.6Mbps의 성능을 나타내었다. 파이프라인이 불가능한 CBC, OFB, CFB 모드의 암호화와 OFB 모드의 복호화 시에는 동일 환경에서 258.9Mbps의 성능을 보였다.

ABSTRACT

This paper implemented into hardware SEED which is the KOREA standard 128-bit block cipher. First, at the respect of hardware implementation, we compared and analyzed SEED with AES finalist algorithms - MARS, RC6, RIJNDAEL, SERPENT, TWOFISH, which are secret key block encryption algorithms. The encryption of SEED is faster than MARS, RC6, TWOFISH, but is as five times slow as RIJNDAEL which is the fastest.

We propose a SEED hardware architecture which improves the encryption speed. We devided one round into three parts, J1 function block, J2 function block, J3 function block including key mixing block, because SEED repeatedly executes the same operation 16 times, then we pipelined one round into three parts, J1 function block, J2 function block, J3 function block including key mixing block, because SEED repeatedly executes the same operation 16 times, then we pipelined it to make it more faster. G-function is implemented more easily by xoring four extended 4 byte SS-boxes. We tested it using ALTERA FPGA with Verilog HDL. If the design is synthesized with 0.5 um Samsung standard cell library, encryption of ECB and decryption of ECB, CBC, CFB, which can be pipelined would take 50 clock cycles to encrypt 384-bit plaintext, and hence we have 745.6 Mbps assuming 97.1 MHz clock frequency. Encryption of CBC, OFB, CFB and decryption of OFB, which cannot be pipelined have 258.9 Mbps under same condition.

keyword : *cryptography, secret-key block cipher, pipelined, SEED, AES*

---

* 이 논문은 2000년도 광운대학교 교내 학술 연구비 지원에 의해 연구되었습니다.
** 광운대학교 전자통신공학과 실시간 구조 연구실 (swjun@explore.gwu.ac.kr)
*** 광운대학교 전자공학부 조교수 (yjjeong@daisy.gwu.ac.kr)

Ⅰ. 서 론

컴퓨터와 통신 기술의 발달로 통신망을 통한 정보의 전송이 급속히 늘어남에 따라 이들 정보의 보안 문제가 대두하게 되었다. 이런 안전한 정보의 전송 및 보관을 위해서는 정보의 암호화가 필요하다. 암호는 데이터를 암호 처리하여 제3자가 데이터를 취득하여도 내용을 알 수 없으며, 오직 키를 가진 사람만이 복호화하여 데이터를 이용할 수 있도록 하는 기술이다.

암호화 방법에는 키의 공개 여부에 따라 공개키 암호와 비밀키 암호로 나뉜다. 공개키 암호는 암호화할 때의 키와 복호화할 때의 키가 다른 시스템이며, 대표적인 예로 RSA와 ECC가 있다. 비밀키 암호는 암호화할 때의 키와 복호화할 때의 키가 같은 시스템으로, 블록 암호 알고리즘과 스트림 암호 알고리즘으로 나뉘며, 1977년 IBM이 개발해 미국정부에 의해 국가 암호 표준으로 채택된 DES(Data Encryption Standard)가 현재 널리 사용되고 있다. 그러나 최근 컴퓨터 병렬 처리 기술의 발달로 인해 DES가 해독되는 등 보안 및 관리에 문제점이 지적되고 있다. 이로 인해 새 세기에 대비한 새로운 표준 암호 알고리즘의 필요성이 제기되었고, 이에 미국 NIST(National Institute of Standards and Technology)에서는 차세대 미국 암호 표준을 개발하기 위한 AES(Advanced Encryption Standard) 개발 과제가 수행되었다.

최종 라운드에 MARS, RC6, SERPENT, RIJNDAEL, TWOFISH 5개의 후보가 올랐고, 2000년 10월에 그 중 하나인 RIJNDAEL이 최종 AES 알고리즘으로 채택되었다. 그리고 1년 정도의 시험 과정을 거친 후, 미국 표준으로 공식 발표할 예정이다. 그리고 유럽을 중심으로 AES 최종 과정에 기여할 뿐만 아니라 그와 독립적으로 세계 표준을 만들기 위한 움직임이 일어나 암호 원천 기술 공모를 발표하였고, 우리 나라에서도 여기에 우리 나라 암호 표준중의 하나로 128비트 블록 암호 알고리즘인 SEED를 제출하려고 하는 과정에 있다. SEED는 국내 정보보호센터 주관으로 만들어진 128비트 블록 암호 알고리즘으로, 국내 암호 표준으로 제정되어 현재 민간 분야에서 폭넓게 활용되고 있다.

데이터 보호를 위한 암호 알고리즘은 보통 구현이 용이하고 개발비용이 저렴한 이유 등으로 주로 소프트웨어로 구현된다. 그러나 소프트웨어로의 구현은 컴퓨터나 네트워크의 발전으로 증가하는 처리 속도를 따라가지 못하며 보안에서도 약점을 보인다. 따라서 암호 처리 속도를 증가시키고 보안을 강화하기 위해 암호 알고리즘의 하드웨어로의 구현이 필요하다. 본 논문에서는 먼저 SEED를 AES 최종 후보 알고리즘들과 하드웨어 구현 측면에서 비교하고, 속도 측면에서 우수한 성능을 가질 수 있는 고속의 SEED 하드웨어 구조를 제안한다.

본 논문의 구조는 2장에서 이전에 SEED를 하드웨어로 구현한 논문들의 구조와 성능을 분석하고, 3장에서는 블록 암호를 위한 운영 모드에 대해서 알아본다. 4장에서는 SEED 암호 알고리즘에 대해서 알아보고, 5장에서는 SEED 암호 알고리즘을 AES 최종 후보 알고리즘들과 하드웨어 구현 측면에서 비교한다. 6장에서는 최고 3배까지의 속도 향상을 기할 수 있도록 하는 SEED의 하드웨어 구조를 제안하고, 이 구조에서의 동작과정에 대해 설명한다. 7장에서는 제안한 하드웨어 구조의 크기와 성능을 분석하고, 다른 논문과 성능을 비교한 후, 마지막으로 결론을 이끌어 내었다.

Ⅱ. Previous Work

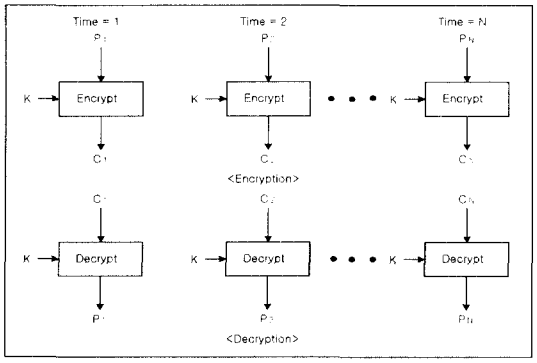
SEED를 하드웨어로 구현한 기존의 방법들을 살펴보면, 현재 일부 연구소와 회사들이 자체 설계를 가지고 있으나 내부 구조를 공개하지 않으므로 여기에서는 공개되어 있는 몇 가지 구현사례에 대해 알아본다. [8]에서는 DES, Triple DES와 SEED 암호 알고리즘을 모두 수행하는 암호 프로세서를 설계하였고, 4 가지 동작 모드(ECB, CBC, CFB, OFB)를 모두 지원한다. SEED의 경우 1라운드 구조의 하드웨어를 배치하고 16라운드 동안 반복적으로 사용하는 구조를 사용한다. F 함수가 복잡하여 1클럭으로 1라운드를 구현하면 많은 하드웨어가 소요되고 클럭 주파수를 감소시키므로 SEED 라운드 동작을 3개의 클럭으로 구현하고 라운드 키의 계산은 이전 라운드에 파이프라인 방식으로 미리 계산하는 방법을 사용한다. 3개의 클럭으로 SEED의 1라운드를 구현함으로써 하드웨어 공유를 극대화시켜 각 클럭에 1개의 G 함수만 필요하게 되어 하드웨어 면적을 1라운드/1 클럭 방식에 비해 2/3 정도 감소시켰다. 이를 0.25um CMOS 공정으로 합성하여 최장 지연 경로는 약 9.38ns이었고, 100Mhz의 동작 주파수에서 250Mbps까지 가능하다고 서술되어있다.[8]

또한 [9]에서는 필요한 자원을 최소로 줄여 SEED의 모든 알고리즘이 구현되도록 하였다. F 함수에서의 3번의 G 함수 사용을 1 개의 G 함수로 처리하기 위해 결과를 귀환시켜 처리함으로써 F 함수에서 필요한 G 함수의 수를 1개로 줄였고, G 함수 동작에 필요한 S1-box와 S2-box를 각각 1개씩만 구현하여 이를 순차적으로 사용함으로써 G 함수 구현에 필요한 S-box의 수를 반으로 줄였다. 내부 메모리를 사용하여 FPGA내에 모든 SEED 암호 알고리즘을 구현하였고, 최대 28Mhz의 클럭에서 동작이 가능하며 최대 14.9Mbps로 암호화를 수행한다.[9]

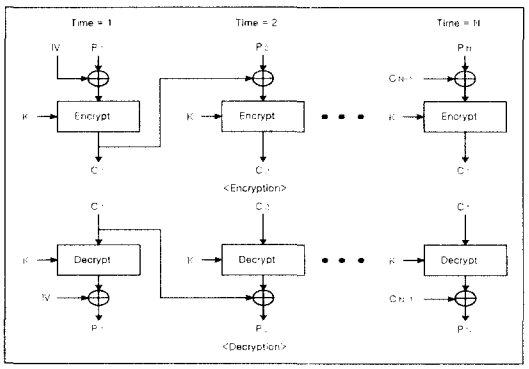
Ⅲ. 블록 암호의 운영 모드

임의의 길이 메시지를 암호화시키기 위하여 블록 암호를 이용할 때 블록 암호를 위한 운용 모드를 이용하는데, 운용모드를 유용하게 이용하기 위해서는 사용된 암호자체 만큼 안전하고 효율적으로 처리되어야 한다. 또한 이러한 운용 모드는 기본적인 암호의 고유 속성이외의 속성을 가질 수도 있다. 어떤 블록 크기의 블록 암호에도 적용할 수 있는 일반화된 네 가지의 운용 모드가 국제 표준인 ISO/IEC 10116으로 표준화되어 있으며, 이러한 모드에는 ECB(Electronic Code Block), CBC(Cipher Block Chaining), CFB (Cipher Feedback), 그리고 OFB(Output Feedback) 모드가 있다.

ECB 모드는 [그림 1]과 같이 가장 단순한 모드로 평문 블록별로 암호 시스템에 의해 암호화한다. 일반적으로 쉽게 생각할 수 있는 모드로 동일한 평문과 키에 의한 암호문은 항상 똑같다는 것이 단점이나 데이터의 암호 처리 속도의 향상을 위해 파이프라인구조의 사용이 가능하다.



[그림 1] ECB 모드의 암복호화 구조



[그림 2] CBC 모드의 암복호화 구조

CBC 모드는 [그림 2]와 같은 구조로, 먼저 평문을 초기 값과 xor 연산을 한다. 이렇게 해서 첫번째 평문을 암호화하여 전송하고 다음 평문부터는 이전 암호문을 저장하고 있다가 이것과 xor한 값을 암호화하여 전송한다. 이런 CBC 모드의 암호화는 이전의 암호문에 영향을 받으므로 동일한 평문과 키에 의한 암호문이라 할지라도 암호문이 경우에 따라 다르게 나타나서 암호의 해독을 더욱더 어렵게 할 수 있다. 그리고 이 모드는 통신로 상의 비트 손실이나 삽입이 일어났는지를 알 수 있다. 예를 들어, i 번째 암호문이 제대로 수신되지 않았다면 i번째 평문을 복호화해내지 못함은 물론, i+1 번째 평문을 얻어내지도 못한다. 이런 이유로 이 CBC 모드는 정보의 인증에 사용된다. 평문을 이전 암호문과 xor 연산을 한 후 암호화하므로, 암호화 시에 파이프라인 구조의 사용이 불가능하고 복호화 시에만 파이프라인 구조를 사용할 수 있다.

OFB 모드는 위성 통신의 암호화에 쓰이며, 암호 시스템에서 나온 데이터와 평문을 xor하여 암호문을 만드는 것으로 오류 전파 현상이 발생하지 않는다. 그러나 송수신 사이에 동기를 인위적으로 조절해야 하고 전송 중 비트의 삽입이나 손실이 발생하면 동기를 조절하여야 한다. 데이터의 종속성으로 보아 파이프라인 구조가 불가능하다.

CFB 모드는 이전의 암호문이 암호 시스템의 입력으로 사용되어 의사 난수 출력을 생성하고, 이것을 다시 평문과 xor하여 암호문을 생성한다. 암호 시스템의 입력을 이전의 암호문에서 받음으로 송수신 사이에 한 비트의 오류가 발생하면 전체 비트까지 오류가 전파된다. 반면에 별도의 동기 조절 기법이 필요하지 않고 자동적으로 동기가 조절된다. 이 모드는 CBC 모드와 같이 정보의 인증에 사용되고, 복

호화 시에만 파이프라인 구조를 사용할 수 있다.

위의 4가지 모드를 지원하도록 암호 알고리즘을 하드웨어로 구현 시 데이터의 처리 속도 향상을 위해 파이프라인 구조를 사용할 수 있다. 그러나 데이터의 종속성으로 보아 CBC, OFB, CFB 모드의 암호화와 CFB 모드의 복호화 시에는 파이프라인 구조를 사용할 수 없다.

Ⅳ. SEED 암호 알고리즘

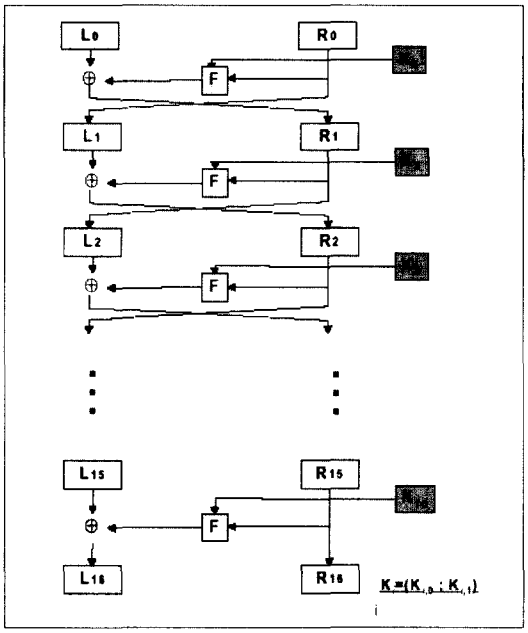
SEED 암호 알고리즘은 Feistel 구조로 이루어져 있으며, 128비트의 평문 블록단위당 128비트 키로부터 생성된 16개의 64비트 라운드 키를 입력으로 사용하여 총 16라운드를 거쳐 128비트 암호문 블록을 출력한다. Feistel 구조란 각각 t비트인 L_0, R_0 블록으로 이루어진 2t비트 평문 블록(L_0, R_0)이 r 라운드(r≥1)를 거쳐 암호문(Lr, Rr)으로 변환되는 반복 구조를 말한다. 이러한 Feistel 구조는 라운드 함수에 관계없이 역 변환이 가능하며, 두 번의 수행으로 블록간의 완전한 diffusion이 이루어지며, 알고리즘의 수행속도가 빠르고, H/W 및 S/W로 구현이 용이하고, 아직 구조상의 문제점이 발견되고 있지 않다는 장점을 지니고 있다.[1]

4.1 SEED 전체 구조

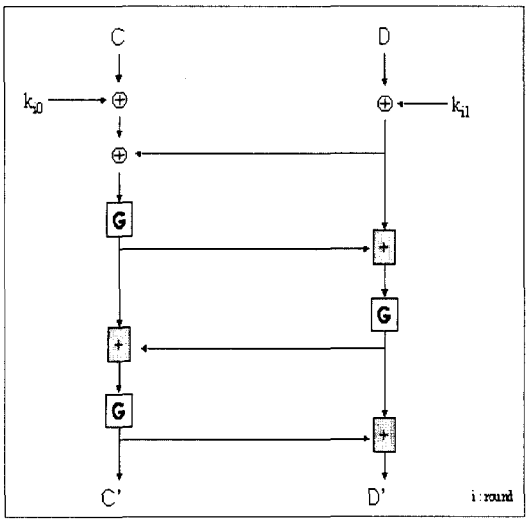
128비트 입력 평문블록을 2개의 64비트 블록(L_0(64), R_0(64))으로 나누어, 16개의 64비트 라운드 키를 이용하여 16라운드를 수행한 후, 최종 128비트 암호문 블록(L_{16}(64), R_{16}(64))을 출력한다. [그림 3]은 SEED 알고리즘의 전체 구조를 나타낸 것이다.

4.2 F 함수

Feistel 구조를 갖는 블록 암호알고리즘은 F 함수의 특성에 따라 구분될 수 있다. SEED의 F 함수는 수정된 64비트 Feistel 형태로 구성된다. F 함수는 각 32비트 블록 2개(C, D)를 입력으로 받아, 32비트 블록 2개(C′, D′)를 출력한다. 즉, 암호화 과정에서 64비트 블록(C, D)와 64비트 라운드 키 Ki=(Ki,0;Ki,1)를 F 함수의 입력으로 처리하여 64비트 블록(C′, D′)을 출력한다. F 함수의 구조는 [그림 4]와 같이 key mixing부분과 32비트 모듈러 덧셈기를 포함한 3개의 G 함수 부분으로 구성되어



[그림 3] SEED 전체 구조



[그림 4] F 함수 구조

있으며, 수식으로 나타내면 식 (1)과 같이 기술되어진다. (i=라운드 수)

$$C' = G\{G\{G\{(C \oplus K_{i0}) \oplus (D \oplus K_{i1})\} \boxplus (C \oplus K_{i0})\} \boxplus G\{G\{(C \oplus K_{i0}) \oplus (D \oplus K_{i1})\} \boxplus G\{G\{(C \oplus K_{i0}) \oplus (D \oplus K_{i1})\} \boxplus (C \oplus K_{i0})\}$$

$$D' = G\{G\{G\{(C \oplus K_{i0}) \oplus (D \oplus K_{i1})\} \boxplus (C \oplus K_{i0})\} \boxplus G\{(C \oplus K_{i0}) \oplus (D \oplus K_{i1})\}\} \quad (1)$$

4.3 G 함수

전체 G 함수는 식 (2)와 같이 기술되며, [그림 5]는 G 함수의 구조를 나타낸 것이다.

$Y_3=S_2(X_3),\ Y_2=S_1(X_2),\ Y_1=S_2(X_1),\ Y_0=S_1(X_0)$

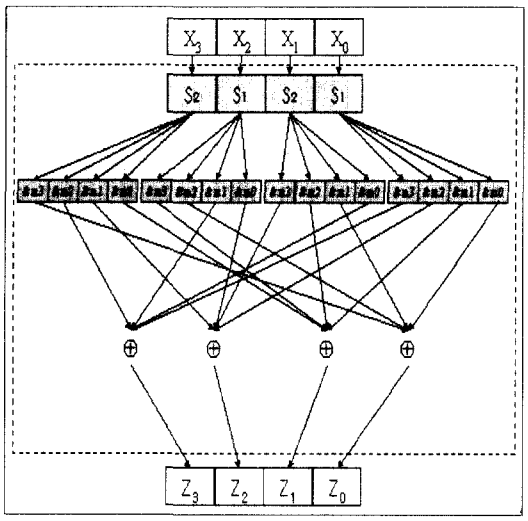
$$\begin{aligned}Z_3&=(Y_0\&m_3)\oplus(Y_1\&m_0)\oplus(Y_2\&m_1)\oplus(Y_3\&m_2)\\ Z_2&=(Y_0\&m_2)\oplus(Y_1\&m_3)\oplus(Y_2\&m_0)\oplus(Y_3\&m_1)\\ Z_1&=(Y_0\&m_1)\oplus(Y_1\&m_2)\oplus(Y_2\&m_3)\oplus(Y_3\&m_0)\\ Z_0&=(Y_0\&m_0)\oplus(Y_1\&m_1)\oplus Y_2\&m_2)\oplus(Y_3\&m_3)\end{aligned} \quad (2)$$

where m0=0xfc, m1=0xf3, m2=0xcf, m3=0x3f
(& : bit-wise AND)

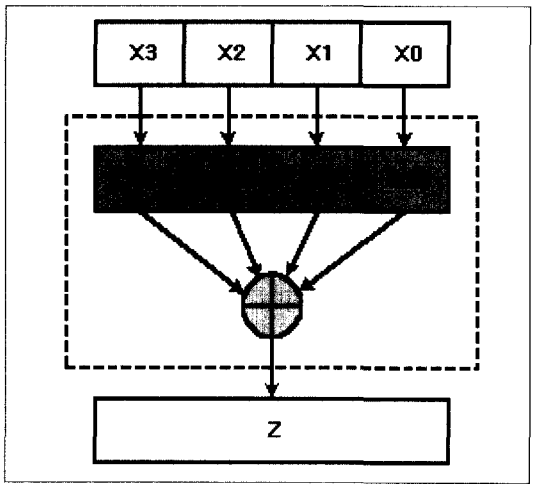
G 함수는 구현의 효율성을 위해 4개의 확장된 4 바이트 SS-box들(4K bytes)의 xor로 구현할 수 있다. 이를 위해 식 (3)과 같은 4개의 SS-box들을 저장해야 한다.

$$\begin{aligned}SS3&=S2(X3)\&m2\ ||\ S2(X3)\&m1\ ||\\ &\quad S2(X3)\&m0\ ||\ S2(X3)\&m3\\ SS2&=S1(X2)\&m1\ ||\ S1(X2)\&m0\ ||\\ &\quad S1(X2)\&m3\ ||\ S1(X2)\&m2\\ SS1&=S2(X1)\&m0\ ||\ S2(X1)\&m3\ ||\\ &\quad S2(X1)\&m2\ ||\ S2(X1)\&m1\\ SS0&=S1(X0)\&m3\ ||\ S1(X0)\&m2\ ||\\ &\quad S1(X0)\&m1\ ||\ S1(X0)\&m0\end{aligned} \quad (3)$$

(여기서, ||는 concatenation)



[그림 5] G 함수 구조 - S-box 사용시



[그림 6] G 함수 구조 - SS-box 이용시

이 확장 SS-box들을 이용하면 G 함수의 결과는 식 (4)와 같이 기술되며, 구조는 [그림 6]와 같다.

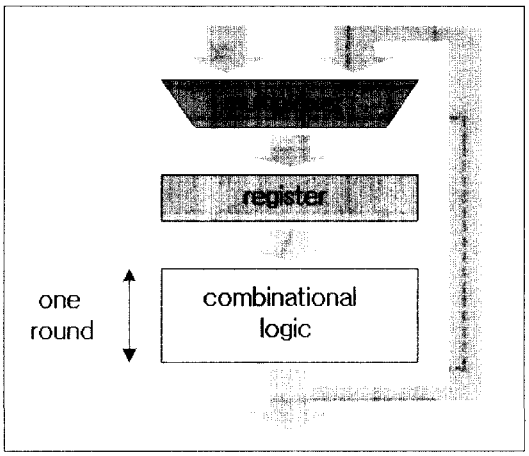
$$Z=SS3(X3)\oplus SS2(X2)\oplus SS1(X1)\oplus SS0(X0) \quad (4)$$

V. SEED와 AES 최종 후보 알고리즘들의 비교 분석

5.1 기본 가정

본 논문에서는 SEED의 하드웨어 구현 측면에서의 성능을 알아보기 위해 같은 비밀키 블록 암호 알고리즘으로 AES 최종 후보 알고리즘들이었던 MARS, RC6, RIJNDAEL, SERPENT, TWOFISH와 비교하였다. SEED와 AES 최종 후보 알고리즘들의 동등한 비교를 위해 다음과 같은 가정 하에 분석한다.

(1) 128 비트의 평문과 128비트의 키를 사용한다.
(2) 키 생성 부는 구현하지 않는다. 그 대신에, 외부의 키 생성 부에서 읽어올 라운드 키를 저장할 메모리를 포함시킨다. 이는 키 생성과정이 매번 일어나는 것이 아니므로, 필요할 때마다 키 생성부를 외부에서 소프트웨어로 구현하고 로딩할 수 있다는 가정 하에서이다.
(3) [그림 7]과 같이 1 라운드만 구현하고, 여기에 레지스터와 멀티플랙서를 각각 1개씩 추가한다. 먼저 멀티플랙서를 통해 평문을 입력받아 레지스터에 저장한다. 다음 클럭에서 1라운드만을



〔그림 7〕 SEED와 AES 최종 후보 알고리즘들의 비교를 위한 기본 구조

구현한 라운드 수행부를 수행하고, 이를 멀티플랙서를 통해 귀환시켜 레지스터에 저장한다. 이 과정을 필요한 라운드 수만큼 반복하여 데이터를 암호화한다.

(4) 0.5um 삼성 스탠다드 셀 라이브러리를 근거로 성능을 예측하고, 라우팅에 의해 생기는 지연은 고려하지 않는다.

5.2 최장 지연 패스

위의 4가지 가정 하에 SEED와 AES 최종 후보 알고리즘들을 하드웨어로 구현 시 클럭의 주기를 결정하는 최장 지연 패스는 〔표 1〕과 같다.

최장 지연 패스에 의해 SEED와 AES 최종 후보 알고리즘들은 2 가지 부류로 구분할 수 있다. 첫 번째 부류로, SEED와 MARS, RIJNDAEL, SERPENT는 S-box와 여러개의 XOR에 의해 최장 지연 패스가 결정되었다. 여기에 SEED와 RIJNDAEL의 최장 지연 패스에는 덧셈기도 포함된다. 두 번째 부류로 RC6와 TWOFISH는 최장 지연 패스에 S-box가 포함되지 않은 것으로, RC6는 배럴 쉬프트와 XOR, TWOFISH는 덧셈기와 XOR에 의해 최장지연 패스가 이루어져 있다.

5.3 성능 비교

ASIC칩으로 구현할 경우를 위해 0.5um 삼성 스탠다드 셀 라이브러리를 근거로 성능을 예측하였다. 클럭의 주기는 〔표 1〕의 최장 지연 패스를 수행하는데 걸리는 시간으로, init MUX는 로딩되어 들어오는 새로운 평문 블록과 라운드를 수행하고 귀환되어 들어오는 데이터 중 하나를 선택하는 MUX, XORn은 n비트의 XOR, MUX2는 2개의 입력을 가지는 MUX, ADDn은 n 비트의 덧셈기, Barrel Shift32는 32비트의 Barrel Shift, S-box n×m은 n비트의 입력과 m비트의 출력을 가지는 S-box이다. 클럭의 주기는 S-box를 구현 시 사용한 메모리의 access time, 덧셈기와 배럴 쉬프트와 XOR 지연 등의 합으로 결정되었다. SEED와 AES 최종 후보 알고리즘들의 클럭 주기와 전체 수행 라운드 수, 클럭 수, 성능은 〔표 2〕와 같다. 성능은 클럭 주기와 클럭 수를 곱해서 구해지는 총 수행 시간을 1 개의 평문 블록 크기인 128비트로 나누어 구한다.

〔표 1〕 SEED와 AES 최종 후보 알고리즘들의 최장 지연 패스

Cipher	Components in the critical path (path flow/list of operations)
SEED	register -> G-function -> addition -> xor -> init MUX -> key-mixing
	1 Flip-Flop, 5 XOR2, 1 MUX2, 1 ADD32, 1 SS-box 8*32
MARS	register -> S-box -> xor -> xor
	1 Flip-Flop, 2 XOR2, 1 S-box 9*32
RC6	register -> xor -> data-dependent rotation
	1 Flip-Flop, 1 XOR2, 1 Barrel -Shift32
RIJNDAEL	S-box -> ShiftRow -> MixColumn -> xor -> init MUX
	1 XOR2, 1 XOR3, 1 MUX2, 1 S-box 8*8, 2 ADD8
SERPENT	register -> key-mixing -> S-box -> linear transformation -> init MUX
	1 Flip-Flop, 1 XOR2, 2 XOR3, 1 MUX2, 1 S-box 4*4
TWOFISH	register -> key-addition -> xor -> init MUX
	1 Flip-Flop, 1 XOR2, 1 MUX2, 1 ADD32

5.3.1 MARS

MARS는 forward mixing부분의 8라운드, cryptographic core부분의 16라운드, backwards mixing 부분의 8 라운드로 이루어져있다.[2] MARS의 암호화 과정은 총 32라운드로 구성되며, 1개의 평문을

〔표 2〕 SEED와 AES 최종 후보 알고리즘들의 성능 비교

Cipher	Clock period (ns)	# of rounds (# of clocks)	Speed (Mbps)
SEED	10.3	16 (48)	258.9
MARS	5	32 (114)	224.6
RC6	5	20 (122)	209.8
RIJNDAEL	9.7	10 (10)	1319.6
SERPENT	6.3	32 (32)	634.9
TWOFISH	5	16 (128)	200.0

암호화하는데 5ns의 클럭 주기로 114클럭이 걸리며, 224.6Mbps의 성능을 가진다.

5.3.2 RC6

RC6는 다른 알고리즘들과 달리 S-box를 사용하지 않으며 20 개의 라운드로 구성되어있다.[3] 5ns의 클럭 주기로 122클럭을 수행하여 암호화 과정을 마치며, 209.8Mbps의 성능을 가진다.

5.3.3 RIJNDAEL

RIJNDAEL은 ByteSub, ShiftRow, MixColumn, AddRoundKey transformation부분으로 1라운드가 이루어진다. 1개의 평문을 암호화하는데 이를 10번 반복 수행해야하고, 9.7ns의 클럭 주기로 10클럭이 걸린다.[4] 성능은 1319.6Mbps로, SEED와 AES 최종 후보 알고리즘들 중 가장 좋다.

5.3.4 SERPENT

SERPENT는 SP-network 구조로, key mixing 부분과 S-box부분과 linear transformation 부분으로 구성된 1라운드를 32번 반복 수행하여 평문을 암호화한다.[5] 1개의 평문을 암호화하는데 6.3ns의 클럭 주기로 32클럭이 걸리며, 634.9Mbps의 성능을 가진다.

5.3.5 TWOFISH

TWOFISH는 Feistel-like 구조로, S-box와 MDS (Maximum Distance Separable) Matrices를 사용한 G 함수와 PHT(Pseudo Hadamard Transformation)와 key addition으로 구성된 라운드 함수를 16번 반복 수행하여 암호화 과정을 수행한다.[6] 5ns의 클럭 주기를 사용하여 암호화 과정을 처리하는데 128클럭이 소요되어 200.0Mbps의 성능을 가진다.

5.3.6 SEED

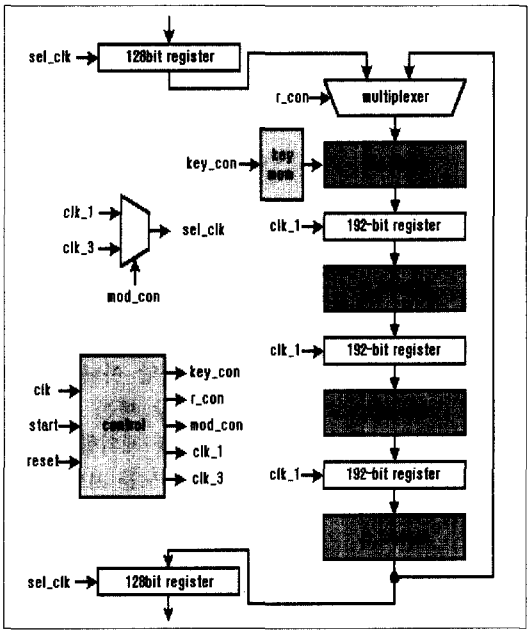
SEED는 10.3ns의 클럭 주기로 16라운드를 48클럭에 수행하여 258.9Mbps의 성능을 가진다.

분석 결과, RIJNDAEL이 1319.6Mbps로 가장 빠른 처리 속도를 가진다. 이는 두 번째로 빠른 SERPENT보다 2배 이상 빠를 뿐만 아니라 가장 느린 TWOFISH보다는 6.5배 이상 빠른 처리 속도다. RIJNDAEL과 SERPENT를 제외한 나머지 알고리즘들은 200Mbps대의 처리속도를 보이며, 그 중에서는 SEED가 가장 빠른 처리 속도를 보였다.

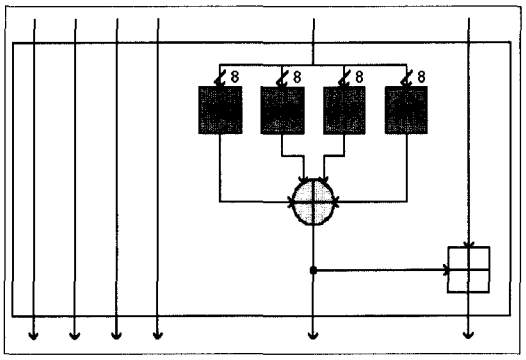
Ⅵ. SEED의 하드웨어 설계

본 장에서는 속도 측면에서 우수한 성능을 가질 수 있는 고속 SEED 구조를 제안한다. 고속의 암호화 과정을 처리할 수 있도록 제안한 SEED의 하드웨어 구조는 〔그림 8〕과 같다. 이 구조는 크게 암호화 처리 블록과 컨트롤 블록의 두 부분으로 나뉘어진다. 암호화 처리 블록은 1라운드를 key mixing 블록, J1 함수 블록, J2 함수 블록, J3 함수 블록의 네 부분으로 나누어 구현하였고, 이를 라운드 수만큼 반복 수행함으로써 데이터를 암호화한다. key mixing 블록은 64비트 라운드 키와 평문의 xor 연산으로 이루어져 있다. J1, J2, J3 함수 블록은 SS-box를 이용하여 구현한 G 함수 블록과 32비트 모듈로 덧셈기로 이루어져 있고, J3 함수 블록은 xor 연산이 추가된다. 처리 속도를 높이기 위해 G 함수 블록에 사용되는 4개의 SS-box를 메모리에 각각 따로 저장하여 병렬로 처리하였다. 〔그림 9〕는 이중 하나인 J2 함수 블록을 나타낸 것이다. 컨트롤 블록은 암호화 처리 블록의 동작을 제어하기 위한 카운터와 간단한 state machine으로 이루어져 있다.

본 논문에서 제안한 SEED engine은 블록 암호의 운영 모드 4가지인 ECB, CBC, OFB, CFB 모드를 모두 지원하도록 설계하였다. 모드에 따라 파이프라인 구조를 사용하는 경우와 파이프라인 구조를 사용하지 않는 경우의 두 가지로 나뉘어진다. 3절에서와 같이 ECB 모드의 암호화, ECB, CBC, CFB 모드의 복호화 시에는 파이프라인 구조를 사

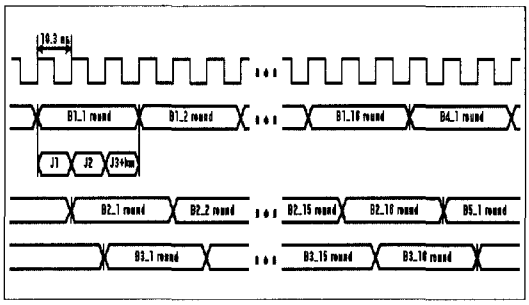


〔그림 8〕 제안한 SEED engine 전체 구조

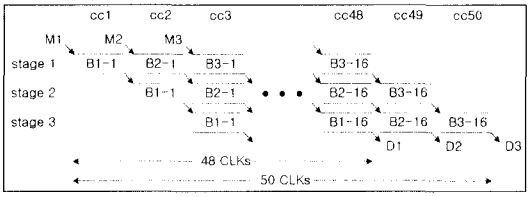


〔그림 9〕 J2 함수 구조

용할 수 있고 나머지 경우에는 파이프라인 구조를 사용할 수 없다. 파이프라인 구조를 사용하는 경우에는 고속의 암호 처리를 위해 암호화 블록을 J1 함수 블록, J2 함수 블록, key mixing 블록을 포함한 J3 함수 블록의 3 개의 단계로 나누고, 이를 파이프라인을 시켜 3클럭에 1라운드가 수행되도록 하였다. 1 라운드에 SS-box를 사용하는 G 함수가 3번 수행되는데, 파이프라인 구조를 사용하면 G 함수 블록 1 개로 공유하여 사용할 수 없으므로 메모리의 사용이 3 배로 늘어나게 된다. 데이터의 종속성으로 인해 파이프라인 구조를 사용할 수 없는 경우에는 파이프라인 구조를 사용할 때 보다 성능이 1/3 정도로 감소한다.

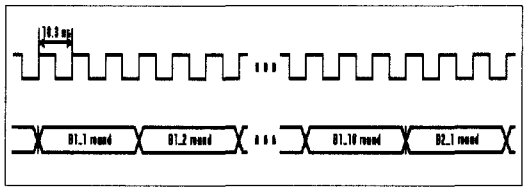


〔그림 10〕 파이프라인 구조를 사용한 ECB 모드의 암복호화와 CBC 모드의 복호화 시 타이밍도



〔그림 11〕 파이프라인 구조를 사용 시 블록별 수행도

먼저 파이프라인 구조를 사용하는 경우, 암호화 시 전체 타이밍도는 〔그림 10〕과 같다. 먼저 로드된 첫 번째 128 비트 평문은 멀티플랙서를 통과하고 key mixing 블록을 수행한 후 레지스터1에 저장된다. 다음 클럭에서 첫 번째 평문은 J1 함수 블록을 수행한 후 레지스터2에 저장되고, 두 번째 평문은 전 클럭에서의 첫 번째 평문처럼 멀티플랙서를 통과하고 key mixing 블록을 수행한 후 레지스터1에 저장된다. 그리고 다음 클럭에서는 첫 번째 평문은 J2 함수 블록을 수행한 후 레지스터3에 저장되고, 두 번째 평문은 J1 함수 블록을 수행한 후 레지스터2에 저장되고, 마지막 세 번째 평문은 다른 평문들과 마찬가지로 멀티플랙서를 통과하고 key mixing 블록을 수행한 후 레지스터1에 저장된다. 다음 클럭부터 멀티플랙서는 로드되어 입력되는 평문이 아닌 J3 함수 블록을 수행하고 귀환되어 오는 데이터를 통과시키며, 3개의 평문은 각각 다음 함수 블록들을 수행한다. 이렇게 3개의 평문은 1라운드만을 구현한 암호화 블록을 각각 16번씩 반복 수행하여 암호문을 생성한다. 그러므로 파이프라인 구조를 사용하면 3개의 평문이 동시에 암호화 과정을 수행하게 된다. 3개의 평문이 암호화 과정을 거쳐 암호문이 생성된 후 다음 3 개의 평문을 위의 과정처럼 멀티플랙서를 통해 받아서 암호화 과정을 수행하게 된다. 이렇게 파이프라인 구조를 사용하여 암호화할 때 각 블록별로 수행되는 과정은 〔그림 11〕과 같다.



〔그림 12〕 파이프라인 구조를 사용하지 않는 CBC 모드의 암호화 시 타이밍도

파이프라인 구조를 사용하지 않는 경우의 암호화 시에는 〔그림 12〕와 같이 하나의 평문을 로드해서 이를 16번 반복 수행하여 암호문을 생성한 후, 다음 평문을 암호화하게 된다.

Ⅷ. 결과 및 고찰

본 논문에서 제안한 SEED engine은 Verilog HDL을 사용해서 ALTERA FPGA로 검증하였으며, 검증을 위해 사용한 값들은 한국정보보호센터에서 제공한 테스트 벡터를 이용하였다.

본 논문에서 제안한 SEED engine의 총 하드웨어 리소스와 성능을 0.5um 삼성 스탠다드 셀 라이브러리를 근거로 산출하였다. 하드웨어 리소스를 구할 때 게이트 카운트는 NAND를 1로 계산하였다. 아울러 〔8〕을 동일한 조건하에서 분석하여 본 논문의 제안 구조와 성능을 비교하였다.

하드웨어 리소스는 〔표 3〕과 같으며, J 함수들은 8×32 SS-box 4개와 32비트 모듈로 덧셈기 1개, 4-input xor 연산 32개가 사용되고 key mixing 블록에서는 xor 연산만이 사용된다. 여기에 입출력 값의 저장과 파이프라인 구조를 위한 레지스터, 로드되어 들어오는 데이터와 귀환되어 들어오는 데이터 중 하나를 선택하는 멀티플랙서가 추가되어, 총 게이트 카운트를 계산하면 약 9006.9개의 게이트와 96k 비트의 메모리가 사용된다.

〔표 3〕 제안한 SEED engine의 하드웨어 리소스

	1 round		Total
Hardware Resource	Key mixing	XOR2 96개	XOR2 160개 XOR4 96개 2-1MUX 128개 Register128 6개 ADD32 3개 8*32 SS-box 12개 ➡ 9006.9 gate 96k bits capacity memory
	J1 함수	XOR4 32개 ADD32 1개 8*32 SS-box 4개	
	J2 함수	XOR4 32개 ADD32 1개 8*32 SS-box 4개	
	J3 함수	XOR2 64개 XOR4 32개 ADD32 1개 8*32 SS-box 4개	

〔표 4〕 제안한 SEED engine의 성능

	ECB mode CBC mode (decryption)		CBC mode (encryption)	
	1 round	Total	1 round	Total
# of clock	1clock*3 =3clock	3clock*16 +2clock (pipeline) =50clock	1clock*3 =3clock	3clock*16 =48clock
Execution time	3clock *10.3ns =30.9ns	50clock *10.3ns =0.515us (384-bit)	3clock *10.3ns =30.9ns	48clock *10.3ns =0.4944us (128-bit)
Speed	745.6Mbps		258.9Mbps	
*External memory	451.8Mbps		156.9Mbps	

제안한 SEED engine의 클럭 수와 수행 시간, 성능은 〔표 4〕에서와 같다. 클럭의 주기는 xor 연산 3개, 32 비트 모듈로 덧셈기 1개, mux 1개, 메모리 1개로 구성된 최장 지연 패스에 의해 결정된다. 최장 지연 패스의 수행 시간은 3.3ns의 응답 속도를 가지는 synchronous contact ROM을 사용하여 10.3ns의 지연을 가지게 된다. 성능은 파이프라인 구조를 사용했을 때와 사용하지 않았을 때의 두 가지로 나누어진다. 파이프라인 구조를 사용하는 ECB 모드의 암호화와 ECB, CBC, CFB 모드의 복호화 시에는 1라운드가 3클럭에 수행되고 전체는 16라운드로 구성되므로 48클럭이 걸리고, 3단계의 파이프라인 구조이므로 2클럭이 추가로 소요된다. 그러므로 384비트의 평문을 암호화하는데 총 50클럭이 소요되어 97.1Mhz의 클럭에서 745.6Mbps의 성능을 가진다. 내부 메모리 대신 외부 메모리를 사용하면 가장 빠른 메모리의 응답 속도가 10ns이므로 58.8Mhz의 클럭에서 451.8Mbps의 성능을 가진다. 파이프라인 구조를 사용하지 않는 CBC, OFB, CFB 모드의 암호화와 OFB 모드의 복호화 시에는 128비트의 평문이 암호화하는데 48클럭이 소요되므로 내부 메모리를 사용하면 258.9Mbps, 외부 메모리를 사용하면 156.9Mbps의 성능을 가진다.

본 논문에서 제안한 SEED engine의 객관적인 성능 평가를 위해 최근까지 발표된 논문들 중 가장 좋은 성능을 나타내는 〔8〕과 비교하였다. 〔8〕과 본

〔표 5〕 제안한 SEED engine과 〔8〕의 성능 비교

	〔8〕	Here	
		CBC mode (encryption)	ECB mode CBC mode (decryption)
# of clock	3clock*16 =48clock	3clock*16 =48clock	3clock*16 + 2clock =50clock
execution time	48clock *10ns =0.480us (128bit)	48clock *10ns =0.480us (128bit)	50clock *10ns =0.500us (384bit)
speed	266.7Mbps	266.7Mbps	768.0Mbps

논문의 성능 비교는 〔표 5〕에 보였다. 서로 공정이 다르지만 최장 지연 패스가 거의 비슷하므로, 공정상의 변수를 제거하기 위해 100MHz 클럭을 기준으로 적용하였다. 〔8〕에서는 1라운드를 수행하는데 3클럭이 걸리며, 하나의 평문을 암호화하는데 총 48클럭이 소요되어 100Mhz의 클럭 주파수에서 266.7Mbps의 성능을 가지게 된다. 본 논문과 같은 조건으로 하기 위해 라운드 키의 온라인 및 사전 계산 기법으로 인해 생기는 3클럭은 제외하였다. 이는 본 논문의 파이프라인 구조를 사용하지 않는 경우와 거의 같은 구조와 성능을 가진다. 그러나 ECB 모드의 암호화와 ECB, CBC, CFB 모드의 복호화 시 본 논문에서는 파이프라인 구조를 사용하여 〔8〕보다 약 3배 향상된 768.0 Mbps의 성능을 가진다.[8]

Ⅸ. 결 론

본 논문에서는 SEED와 AES 최종 후보 알고리즘들을 하드웨어 구현 측면에서 비교하고, 속도 측면에서 우수한 성능을 가질 수 있는 고속의 SEED 하드웨어 구조를 제안하였다. 제안한 구조는 효율적인 구현을 위해 SS-box를 이용하여 G 함수 블록을 설계하였고, 모드에 따라 파이프라인 구조를 사용하는 경우와 사용하지 않는 경우의 두 가지 구조를 제공한다. 파이프라인 구조를 사용하는 경우에는 1라운드를 3단계로 나누어 파이프라인을 시켜 세 개의 평문을 동시에 암호화할 수 있도록 하여 파이프라인 구조를 사용하지 않았을 때보다 3배의 성능 향상을 이루었다. 그리고 데이터의 종속성으로 인해 파이프라인 구조를 사용하지 않는 경우에는 평문 한 개씩 암호화된다. 그리고 본 논문에서 제안한 SEED engine은 ECB 모드만 암호화와 복호화 모두 파이프라인 구조가 가능하므로 암복호가 동일한 속도를 요구하는 분야에서는 파이프라인 구조에 의한 성능 향상이 ECB 모드에서만 의미가 있다.

〔표 4〕에서 보인 SEED engine의 성능은 0.5um 기술인 〔7〕을 근거로 하였지만, 이에 0.25나 0.18um 기술을 적용하면 속도나 크기에 있어서 한층 향상된 성능을 가질 수 있을 것이다. 제안한 SEED engine은 VPN(Virtual Private Network)이나 방화벽 등의 고속 처리를 요구하는 여러 분야에 활용될 수 있을 것이다. 그리고 본 논문에서 제안한 파이프라인 구조가 암호화 모드 중 일부만 적용 가능하므로, 파이프라인 구조가 적용되기 어려운 모드에 대한 고속화 방식에 대한 연구가 필요하다.

참 고 문 헌

〔1〕 "128비트 블록 암호알고리즘 표준", 한국정보통신기술협회, http://www.kisa.or.kr/technology/sub1/99-06-29-SEED표준제안서.hwp, 1999. 4.

〔2〕 Carolynn Burwick, Don Coppersmith, Edward D'Avignon, Rosario Gennaro, Shai Halevi, Charanjit Jutla, Stephen M. Matyas Jr., Luke O'Connor, Mohammad Peyravian, David Safford, Nevenko Zunic, "MARS - a candidate cipher for AES", http://csr c.nist.gov/encryption/aes/round2/AESAlgs/MARS/mars.pdf, Aug. 1999.

〔3〕 Ronald L. Rivest, M.J.B. Robshaw, R. Sidney, Y. L. Yin, "The RC6 Block Cipher", http://csrc.nist.gov/encryption/aes/round2/AESAlgs/RC6/cipher.pdf, 1999.

〔4〕 Joan Daemen, Vincent Rijmen, "AES Proposal:Rijndael", http://csrc.nist.gov/encryption/aes/round2/AESAlgs/Rijndael/Rijndael.pdf, 1999.

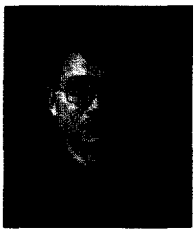
〔5〕 Ross Anderson, Eli Biham, Lars Knudsen, "Serpent: A Proposal for the Advanced Encryption Standard", http://csrc.nist.gov/encryption/aes/round2/AESAlgs/Serpent/Serpent.pdf, 1999.

〔6〕 Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, "Twofish:

A 128-Bit Block Cipher", http://csrc.nist.gov/encryption/aes/round2/AESAlgs/Twofish/Twofish.pdf, 1999.
[7] "Samsung STD85/STDM85 0.5um High Density CMOS Standard Cell Library", Samsung Elect., 1997.
[8] 최병윤, 서정욱, "SEED 알고리즘용 암호 보조 프로세서의 설계", 한국통신학회논문지, 2000. 9.
[9] Young-Ho Seo, Jong-Hyeon Kim, Yong-Jin Jung, Dong-Wook Kim, "Area Efficient Implementation of 128-bit Block Cipher, SEED", ITC-CSCC 2000, 2000.

〈著者紹介〉

전 신 우 (Shin-woo Jeon)
2000년 2월 : 광운대학교 전자공학부 졸업
2000년 3월~현재 : 광운대학교 전자공학과 석사과정
〈관심분야〉 통신용 칩 설계, 무선 통신, 정보보호



정 용 진 (Yong-jin Jeong) 정회원
1983년 2월 : 서울대학교 제어계측공학과 졸업
1983년 3월~1989년 8월 : 전자통신연구원 (ETRI)
1991년 5월 : 미국 UMASS 전자전산공학과 석사
1995년 2월 : 미국 UMASS 전자전산공학과 박사
1995년 4월~1999년 2월 : 삼성전자 반도체 수석 연구원
1999년 3월~현재 : 광운대학교 전자공학부 조교수
〈관심분야〉 컴퓨터 연산 알고리즘, ASIC 설계, 무선 통신, 정보보호