효율적인 SEED 암호알고리즘 구현을 위한 최적화 회로구조

An Optimum Architecture for Implementing SEED Cipher Algorithm with Efficiency

신 광 철*
Shin Kwang Cheul

이 행 우**
Lee Haeng Woo

요 약

본 논문에서는 128-bit 블록암호인 SEED 알고리즘을 하드웨어로 구현하는데 있어서 면적을 줄이고 연산속도를 증가시키는 회로구조에 대하여 논하였고 설계결과를 기술하였다. 연산속도를 증가시키기 위해 pipelined systolic array 구조를 사용하였으며, 입출력회로에 어떤 버퍼도 사용하지 않는 간단한 구조이다. 이 회로는 10 MHz 클럭을 사용하여 최대 320 Mbps의 암호화 속도를 달성할 수 있다. 회로설계는 VHDL 코딩방식으로 수행하였으며, 50,000 gates 급의 FPGA에 구현하였다.

Abstract

This paper describes the architecture for reducing its size and increasing the computation rate in implementing the SEED algorithm of a 128-bit block cipher, and the result of the circuit design. In order to increase the computation rate, it is used the architecture of the pipelined systolic array. This architecture is a simple thing without involving any buffer at the input and output part. By this circuit, it can be recorded 320 Mbps encryption rate at 10 MHz clock. We have designed the circuit with the VHDL coding, implemented with a FPGA of 50,000 gates.

☞ Keyword : SEED, hardware circuit design, pipelined systolic array

1. 서 론

정보보호기술은 전자상거래 등 통신의 수요가 증가할수록 그 중요성이 더욱 증대되고 있으나, 대부분의 시스템이 아직 소프트웨어 방식으로 구현되고 있어서 암호화 속도 및 해킹에 대한 문제가 대두되고 있다. 이에 대한 해결책으로 DES 알고리즘을 하드웨어로 구현하여 사용하고 있으나, 최근 고속 프로세서의 개발로 알고리즘 자체의 안전성에 심각한 위협이 되고 있다. 따라서 미국에서는 DES의 대안으로 새로운 대칭형 암호 표준인 AES (Advanced Encryption Standard)를 선정하여 보급하고 있다. 그동안 암호 알고리즘에 대한 대부분의 연구는 미국, 유럽, 일본 등 몇몇 선진국에 의해 주도되어 왔으나, 국내에서도 한국 정보보호진흥원을 중심으로 관련 전문가들이 연구하여 독자적인 128-bit 블록암호 알고리즘인 SEED를 개발하여 표준으로 정하였다.[1]

SEED 암호의 응용분야는 기존 통신망 보호 시스템, PC 보호 시스템 등 각종 전산 시스템과 전자상거래, 온라인 금융거래, 무역 자동화 등 각종 정보통신 서비스분야의 암호기술로 고속의 정보보호 시스템분야를 지원할 수 있다.

SEED 암호 알고리즘은 안전성을 강화하기 위하여 128-bit 형태를 사용하였으며, DES에 비해 복잡도가 증가한 Feistel 구조를 갖고 있다. 따라서 SEED 알고리즘은 DES 알고리즘에 비해 안전성은 증가하였으나, 하드웨어의 복잡성으로 인해 회로

* 정 회 원 : 성결대학교 e-비즈니스 IT학부 교수
skcskc12@sungkyul.edu(제1저자)
** 정 회 원 : 남서울대학교 정보통신공학과 교수
hwlee@nsu.ac.kr
[2005/08/23 투고 - 2005/09/12 심사 - 2005/11/03 심사완료]

면적이 증가하고 암호화 속도가 떨어지는 단점이 있다.[2] 본 논문에서는 이와 같은 회로 면적의 증가와 속도저하 문제를 해결하기 위한 회로구조를 제안한다. SEED 암호의 연산과정은 동일한 연산을 16 round 반복하여 수행하기 때문에 직렬 또는 병렬방식으로 설계할 수 있다. 응용분야에 따라 암호화 속도를 증가시키는 것이 주요 문제이면 병렬방식으로, 또는 속도보다 회로 면적의 축소가 관건이면 직렬방식으로 구현한다. 그런데 이 둘 중 하나만 중요하다기보다 대부분의 경우 속도와 면적이 모두 주요 이슈가 되기 때문에 이러한 응용을 위해서 직렬방식의 면적과 병렬방식의 속도를 둘 다 고려한 복합방식을 필요로 한다. 따라서 순수 병렬방식에 비해 면적을 1/4 축소하면서 속도는 직렬방식의 4배에 이르는 2가지 복합방식을 소개하고 회로 및 동작 메커니즘을 비교한다. 또한 pipelined systolic array 구조를 사용하여 선택된 복합방식의 회로구현방법에 대하여 논한다.[3-5]

본 논문의 구성은 2장에서 SEED 암호 알고리즘을 간단히 소개하고, 3장에서 이 알고리즘의 구현을 위한 최적화 회로구조를 제안하며, 4장에서는 회로설계 결과에 대하여 논하고, 끝으로 5장에서 결론을 유도하였다.

Ⅱ. SEED 암호 알고리즘

암호 알고리즘은 암,복호화에 사용되는 키의 특성에 따라 암,복호화 키가 동일한 대칭키 암호알고리즘과 암,복호화 키가 서로 다른 공개키 암호 알고리즘으로 분류할 수 있으며, 대칭키 암호 알고리즘은 데이터 처리형식에 따라 스트림 암호 알고리즘과 블록 암호 알고리즘으로 나누어진다. .SEED는 대칭키 암호 알고리즘으로 블록 단위로 메시지를 처리하는 블록 암호 알고리즘이다.

대칭키 블록 암호알고리즘은 비밀성을 제공하는 암호시스템에서 널리 사용되는 주요 알고리즘이다. n-bit 블록 암호알고리즘은 고정된 n-bit 평문을 같은 길이의 n-bit 암호문으로 변환시킨다. 이러한 변형 과정에 암,복호키가 작용하여 암호화와 복호화를 수행한다.

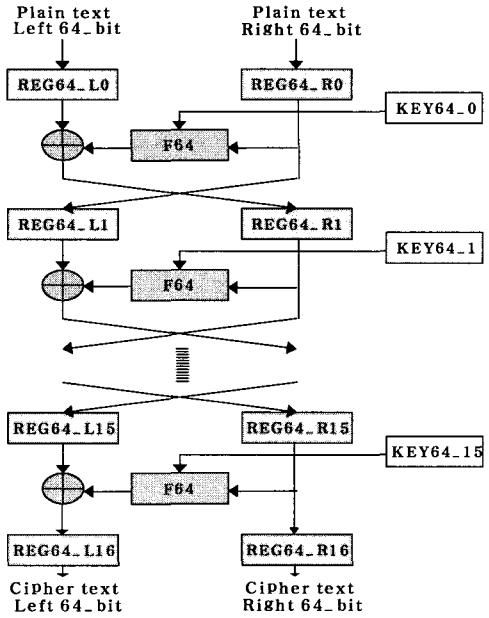
블록 암호알고리즘은 대부분 Feistel 구조로 설계된다. Feistel 구조란 각각 t-bit인 L_0, R_0 블록으로 이루어진 2t-bit 평문 블록(L_0, R_0)이 r-round를 거쳐 암호문 (L_r, R_r)으로 변환되는 반복 연산구조를 말한다. 반복 연산구조란 평문 블록이 여러 라운드를 거쳐 암호화되는 과정을 말한다. 라운드 함수 $i(1 \le i \le r)$는 암호키 K로부터 생성된 각 서브키 K_i를 주요 입력으로 하여 $L_i = R_{i-1}$, $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$를 통해 $(L_{i-1}, R_{i-1}) \xrightarrow{K_i} (L_i, R_i)$로 변환시켜 준다. 또한 전체 알고리즘의 라운드 수는 요구되는 비도와 수행 효율성의 상호관계에 의해 절충(trade-off)되어 정해진다. 보통 Feistel 구조는 3-round 이상이며, 짝수 라운드로 구성된다. 이러한 Feistel 구조는 첫째, 라운드 함수에 관계없이 역변환이 가능하며(즉 암·복호화 과정이 같다) 둘째, 두 번의 수행으로 블록간의 완전한 확산이 이루어진다. 셋째, 알고리즘의 수행속도가 빠르며 넷째, H/W 및 S/W로 구현이 용이하고 마지막으로 아직 구조적인 문제점이 발견되지 않았다는 장점을 갖고 있다.

2.1 알고리즘 전체 구성도

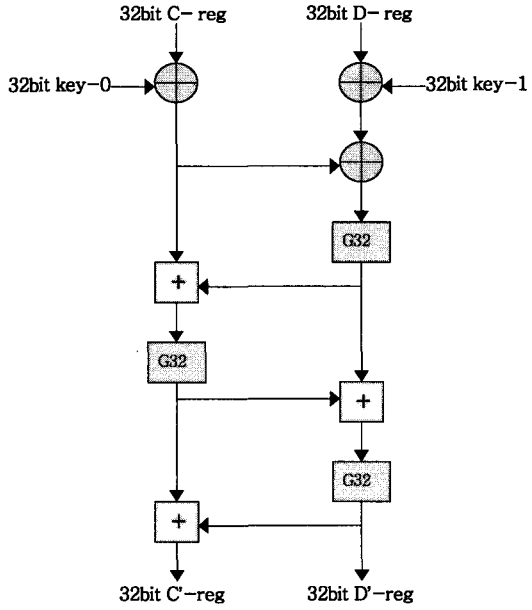
본 알고리즘의 전체 구조는 Feistel 구조로 이루어져 있으며, 128-bit의 평문 블록단위당 128-bit 키로부터 생성된 16개의 64-bit 라운드 키를 입력으로 사용하여 총 16-round를 거쳐 128-bit 암호문 블록을 출력한다. 그림 1은 SEED 알고리즘의 연산 구조를 도식화한 것이다.

2.2 F 함수

Feistel 구조를 갖는 블록 암호알고리즘은 F 함수의 연산과정에 따라 특징지어진다(그림 2). SEED의 F 함수는 수정된 64-bit Feistel 형태로 구성된다. F 함수는 각 32-bit 블록 2개(C, D)를 입력 받아, 32-bit 블록 2개(C', D')를 출력한다. 즉, 암호화 과정에서 64-bit right 레지스터 블록(C, D)와



〈그림 1〉 SEED 연산 구조



〈그림 2〉 F 함수 연산과정

64-bit 라운드 키 $K_i = (K_{i,0}; K_{i,1})$를 F 함수의 입력으로 받아 연산한 후 64-bit 블록(C', D')을 출력한다.

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

$$D' = G[G[G\{(C \oplus K_{i0}) \oplus (D \oplus K_{i1})\} \boxplus (C \oplus K_{i0})] \boxplus G\{(C \oplus K_{i0}) \oplus (D \oplus K_{i1})\}] \quad (2)$$

2.3 G 함수

F 함수의 연산은 내장된 G 함수[그림 3]에 의하여 특징지어진다. 32-bit G 함수는 8-bit 4개 블록으로 분리되어 각각 S-box 변환 등의 연산처리 후 8-bit 4개 블록을 출력한다. G 함수의 연산과정은 다음과 같은 식으로 표현된다.

여기서 $m_0 = 0xfc, m_1 = 0xf3, m_2 = 0xcf, m_3 = 0x3f$이다.

2.4 S Box

SEED 알고리즘의 S-box[그림 3]에서는 8-bit 입력, 즉 0~255 숫자를 받아 8-bit 데이터를 출력한다. 이 Box는 2개의 다른 형태가 존재하며 S_1 box와 S_2box로 구성되어 있다.

2.5 라운드 키 생성과정

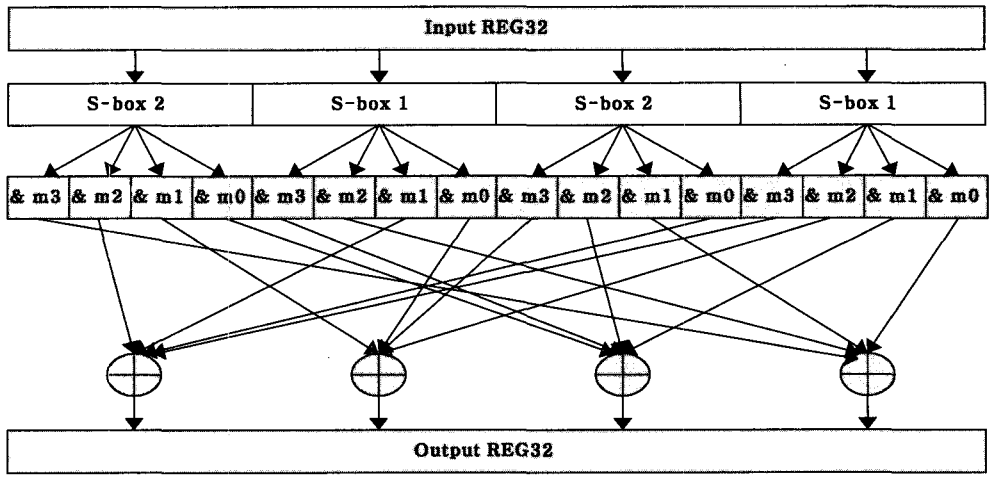
SEED의 라운드 키 생성과정은 128-bit 암호키를 64-bit 씩 좌우로 나누어 이들을 교대로 8-bit 씩 좌/우로 회전 이동한 후 결과의 4-word들에 대한 간단한 산술연산과 G 함수를 적용하여 라운드

$$Z_3 = (S_1(X_0)\ m_3) \oplus (S_2(X_1)\ m_0) \oplus (S_1(X_2)\ m_1) \oplus (S_2(X_3)\ m_2) \quad (3)$$

$$Z_2 = (S_1(X_0)\ m_2) \oplus (S_2(X_1)\ m_3) \oplus (S_1(X_2)\ m_0) \oplus (S_2(X_3)\ m_1) \quad (4)$$

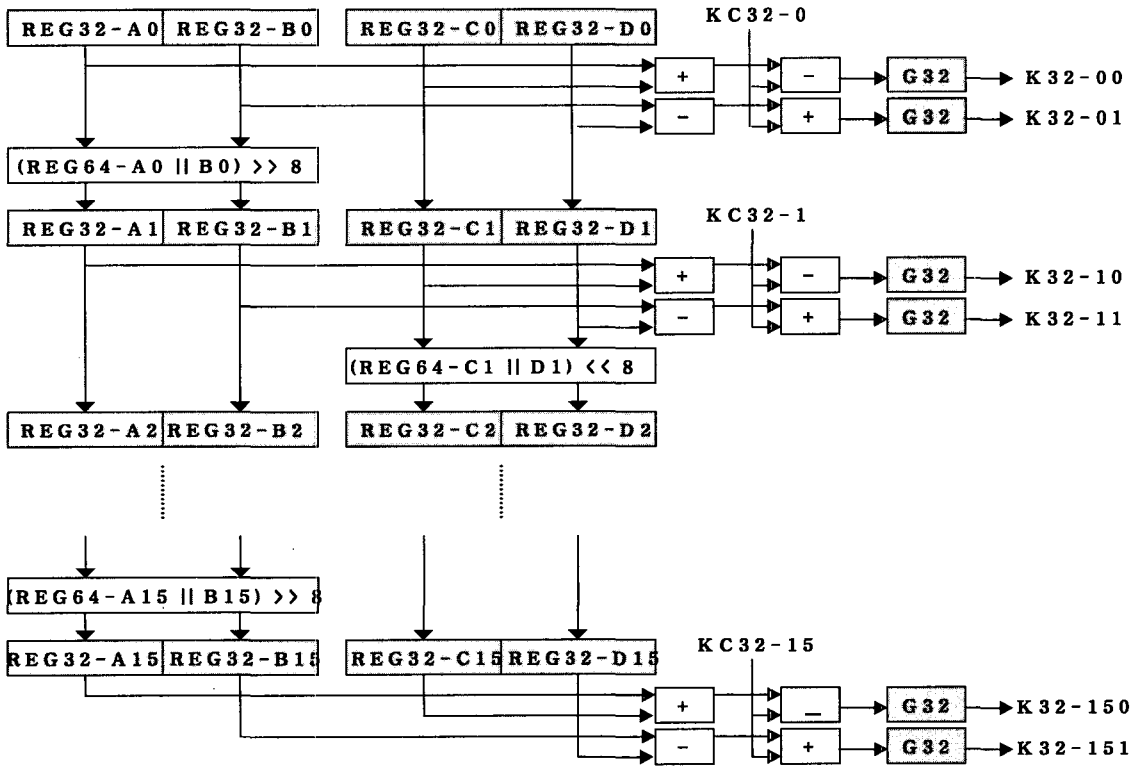
$$Z_1 = (S_1(X_0)\ m_1) \oplus (S_2(X_1)\ m_2) \oplus (S_1(X_2)\ m_3) \oplus (S_2(X_3)\ m_0) \quad (5)$$

$$Z_0 = (S_1(X_0)\ m_0) \oplus (S_2(X_1)\ m_1) \oplus (S_1(X_2)\ m_2) \oplus (S_2(X_3)\ m_3) \quad (6)$$

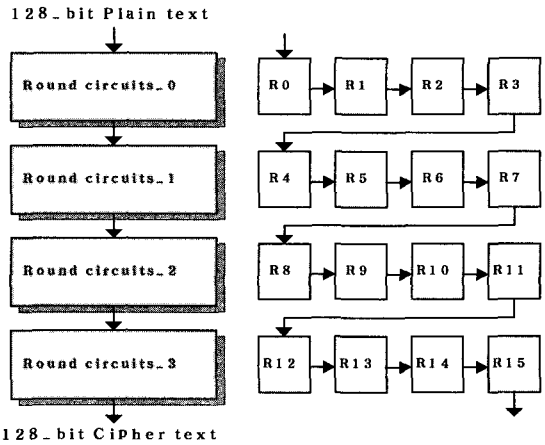


〈그림 3〉 G 함수 연산과정

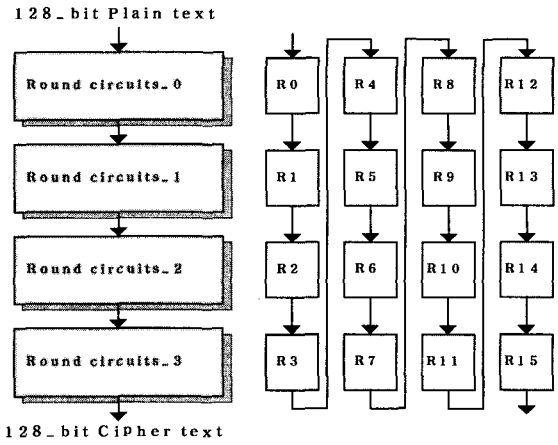
키를 생성한다(그림 4). 라운드 키 생성과정은 기본적으로 하드웨어나 제한된 자원을 갖는 스마트카드와 같은 응용에서의 효율성을 위하여 암호화나 복호화시 암호키로부터 필요한 라운드 키를 간단히 계산할 수 있도록 설계하였다. 각 라운드 키의 생성과정은 다음 식으로 표현된다.



〈그림 4〉 라운드 키 생성과정



〈그림 5〉 회로 동작과정 A



〈그림 6〉 회로 동작과정 B

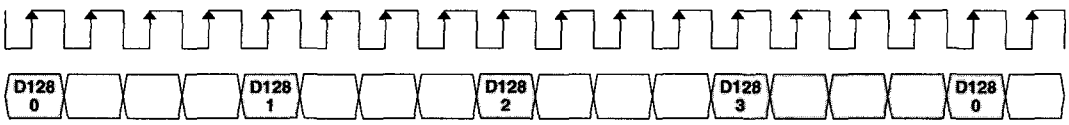
$$K_{i0} = G[(A_i \boxplus C_i) \boxminus KC_i] \quad (7)$$

$$K_{i1} = G[(B_i \boxminus D_i) \boxplus KC_i] \quad (8)$$

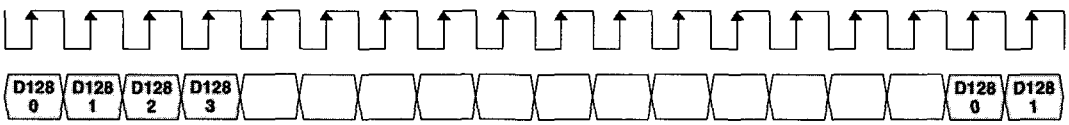
Ⅲ. 최적화 회로구조

회로구조는 응용분야에 따라 달라질 수 있다. 즉 연산속도가 높아야 하는지 아니면 회로 크기가 작아야 하는지에 따라 회로구조가 결정되어야 한다. SEED 알고리즘은 동일한 연산과정을 16회 반복해서 수행하기 때문에 응용분야의 요구에 따라서 연산속도가 높은 병렬방식이나 회로 면적이 작은 직렬방식으로 구현할 수 있다. 그러나 대부분의 응용분야에서는 회로 크기가 작으면서도 연산속도가 높은 하드웨어를 요구하고 있다. 따라서 본 논문에서는 이를 효과적으로 수용할 수 있는 2가지 복합방식의 회로구조를 논하고 상대적으로 우수한 회로구조를 선정하여 설계하고자 한다. 또한 연산속도와 회로의 크기 등에서 유리한 pipelined systolic array 방식을 이용하여 설계한다.

복합방식의 회로구조로서 동시에 4-round를 연산할 수 있는 병렬회로를 설계하고, 이 회로를 통하여 직렬방식으로 4회 반복해서 수행하도록 한다. 이와 같은 형태의 구현으로 회로의 크기는 병렬방식에 비해 1/4로 축소되고, 연산속도는 직렬방식에 비해 4배 향상되는 효과를 얻는다. 복합방식의 회로를 동작시키는 방법은 그림 5, 그림 6과 같이 2가지로 나누어볼 수 있으며 동작과정의 입력파형은 그림 7과 같다.

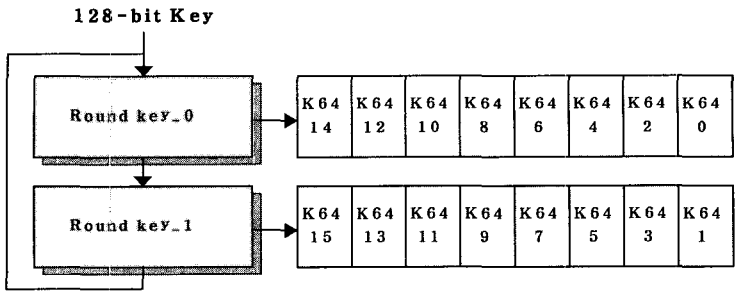


(a) 동작과정 A



(b) 동작과정 B

〈그림 7〉 동작과정의 입력파형



〈그림 8〉 라운드 키 생성과정

회로 동작 A는 128-bit 평문의 입력속도보다 4배 빠른 클럭을 이용하여 각 라운드 회로에서 4-round씩 반복 연산을 수행함으로서 16-round 연산을 완료한다. 즉 라운드 회로-0에서 0~3-round 연산을 연속적으로 수행하고, 이어서 라운드 회로-1에서 4~7-round 연산을 연속적으로 수행하는 방식으로 16-round 연산을 처리한다. 이와 같이 동작하기 위해서는 각 라운드 회로의 데이터 입력부분에 다중화기가 위치해 있어야 한다.

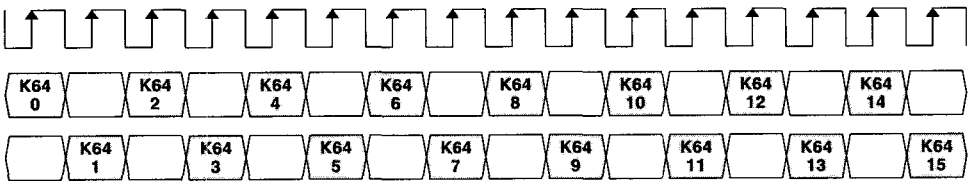
회로 동작 B도 128-bit 평문의 입력속도보다 4배 빠른 클럭을 이용하며 각 라운드 회로에서 1-round씩 연산을 4회 반복 수행함으로서 16-round 연산을 완료한다. 즉 라운드 회로-0에서 라운드 회로-3까지 각각 0~3-round 연산을 연속적으로 수행하고, 다시 라운드 회로-0에서 라운드 회로-3까지 4~7-round 연산을 연속적으로 수행하는 방식으로 16-round 연산을 처리한다. 이와 같이 동작한다면 라운드 회로-0의 데이터 입력부분에만 다중화기가 위치하면 된다. 그러나 128-bit 평문이 회로 동작과 동기 되어 입력되기 위해서는 4개의 입력 데이터를 저장할 버퍼가 필요하고 출력 측에도 동일한 버퍼가 요구된다.

128-bit 입력키를 사용하여 16-round 키를 생성한다. 라운드 키는 데이터 암호화 처리가 시작되기 전 최초 시스템이 작동하는 순간 미리 생성해 둔다. 따라서 연산속도는 중요하지 않으므로 직렬방식을 사용하여 1회에 2 라운드 키씩 동일한 과정을 8회 반복해서 16 라운드 키를 생성하도록 한다. 그리고 만들어진 라운드 키는 16 버퍼에 저장한다. 라운드 키 생성회로의 동작방법은 그림 8과 같으며 라운드 키 생성과정의 동작파형은 그림 9와 같다.

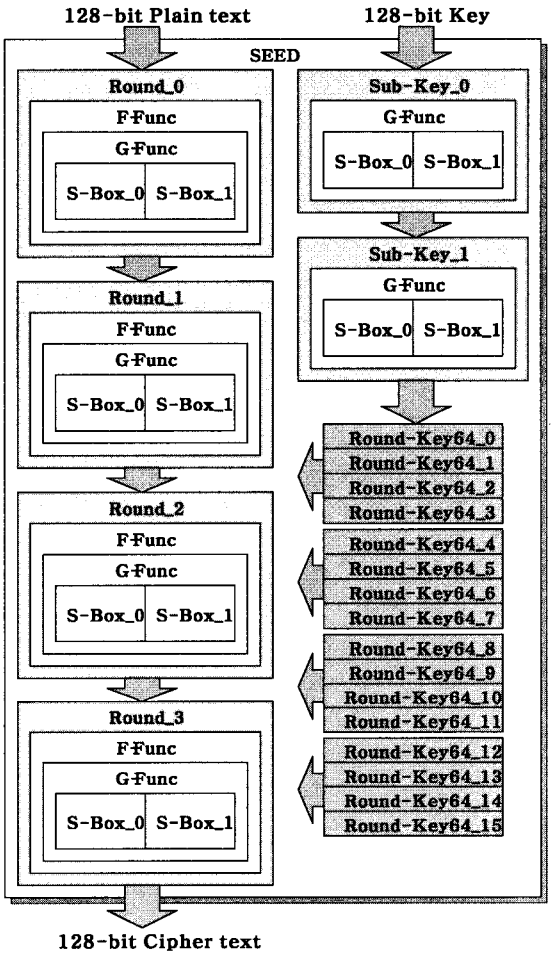
회로동작은 128-bit 입력키를 사용하여 round-0에 공급될 64-bit 서브키를 생성하고, 64-bit 좌측 레지스터의 데이터를 8-bit shift-right한 후 round-1에 공급될 64-bit 서브키를 생성한다. 다시 64-bit 우측 레지스터의 데이터를 8-bit shift-left한 후 위의 연산을 8회 반복 수행함으로서 최종적으로 16-round에 공급될 서브키를 생성한다. 이와 같이 동작하기 위해서는 round-key_0 블록의 입력부분에 다중화기가 있어야 한다.

Ⅳ. 회로설계 결과

SEED 알고리즘의 구현은 입출력 버퍼 없이 간단하게 설계할 수 있는 회로 동작과정 A방식을 이용하여 이루어졌다. Max+plus II 상에서 VHDL



〈그림 9〉 라운드 키 생성과정의 동작파형

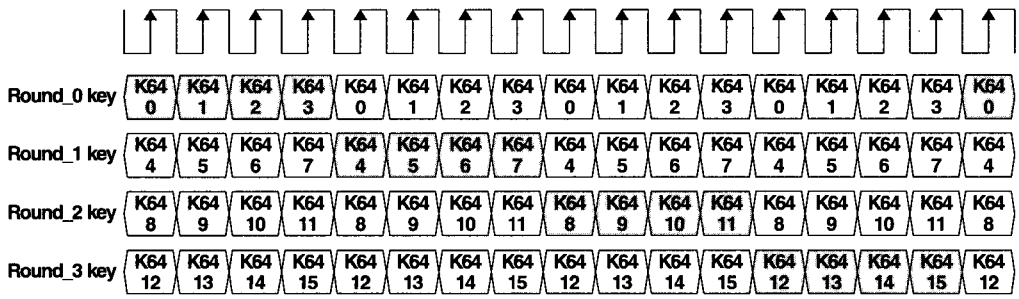


〈그림 10〉 SEED 회로설계 블록도

코딩을 통하여 설계하였다. 회로는 총 7개 파일로 이루어져 있으며, S-box는 ROM을 사용하여 구현하였다. 그리고 주 클럭으로 10 MHz를 사용하여 암호화 속도는 320 Mbps를 달성한다. 데이터에 대한 암호화 연산이 개시되기 전에 미리 16개 라운드 키를 생성하여 버퍼에 저장해 놓는다. 그림 10은 설계한 하드웨어 블록도로서 내부 기능블록들의 구성과 블록 간 연결을 통해 데이터 흐름을 파악할 수 있다.

이 회로는 그림 5의 동작과정 A방식으로 동작하므로 4개 라운드 블록에 공급되는 라운드 키의 순서는 다음 그림 11과 같이 진행된다. Round_0 블록에는 64-bit round-key_0~3이 순서대로 반복적으로 공급되고, 동시에 round_1 블록에는 round-key_4~7, round_2 블록에는 round-key_8~11, 그리고 round_3 블록에는 round-key_12~15 등이 반복적으로 공급된다. Round_0 블록에서는 라운드 키를 사용하여 연속적으로 4회 반복해서 동일 연산이 이루어지고, 그 결과가 다음 클럭에 round_1 블록으로 출력되어 다시 유사한 연산이 이루어지며, 이러한 과정이 round_3 블록까지 반복적으로 수행된다. 이때 라운드 키 생성부에서는 매 클럭마다 4개의 라운드 키가 각 라운드 블록에 공급된다. 암호화 회로는 Pipelined systolic array 방식에 의하여 동작되기 때문에 128-bit 평문의 암호화가 완료되기 위해서는 16개의 주 클럭이 요구된다. 따라서 전체 암호화에 소요되는 처리시간은 1.6 μs가 된다.

회로설계 후 테스트를 위해 모의실험을 수행하였다. 실험을 위한 입력 데이터는 표 1의 데이터들을 사용하였다. 이 데이터들은 암호키가 없고 평문만 제공되는 경우와 평문이 없고 암호키만 제공되는 경우의 테스트 값이다. 암호키와 평문을 회로에 입력시켜 각 라운드 키와 암호문이 이상 없이 출력되는 것을 확인하였다.



〈그림 11〉 라운드 키 공급순서 파형

〈표 1〉 회로 테스트 값

암호키	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
평문	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F		
암호문	5E BA C6 E0 05 4E 16 68 19 AF F1 CC 6D 34 6C DB		
라운드 키			
K1_0 = 0x7c8f8c7e	K1_1 = 0xc737a22c	K9_0 = 0x76cc05d5	K9_1 = 0xe97a7394
K2_0 = 0xff276cdb	K2_1 = 0xa7ca684a	K10_0 = 0x50ac6f92	K10_1 = 0x1b2666e5
K3_0 = 0x2f9d01a1	K3_1 = 0x70049e41	K11_0 = 0x65b7904a	K11_1 = 0x8ec3a7b3
K4_0 = 0xae59b3c4	K4_1 = 0x4245e90c	K12_0 = 0x2f7e2e22	K12_1 = 0xa2b121b9
K5_0 = 0xa1d6400f	K5_1 = 0xdbc1394e	K13_0 = 0x4d0bfde4	K13_1 = 0x4e888d9b
K6_0 = 0x85963508	K6_1 = 0x0c5f1fcb	K14_0 = 0x631c8ddc	K14_1 = 0x4378a6c4
K7_0 = 0xb684bda7	K7_1 = 0x61a4aeae	K15_0 = 0x216af65f	K15_1 = 0x7878c031
K8_0 = 0xd17e0741	K8_1 = 0xfee90aa1	K16_0 = 0x71891150	K16_1 = 0x98b255b0

암호키	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F		
평문	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
암호문	C1 1F 22 F2 01 40 50 50 84 48 35 97 E4 37 0F 43		
라운드 키			
K1_0 = 0xc119f584	K1_1 = 0x5ae033a0	K9_0 = 0x2bd30235	K9_1 = 0x51679ce6
K2_0 = 0x62947390	K2_1 = 0xa600ad14	K10_0 = 0xfa8d6b76	K10_1 = 0xa9f37e02
K3_0 = 0xf6f6544e	K3_1 = 0x596c4b49	K11_0 = 0x8b99cc60	K11_1 = 0x0f6092d4
K4_0 = 0xc1a3de02	K4_1 = 0xce483c49	K12_0 = 0xbdaefcfa	K12_1 = 0x489c2242
K5_0 = 0x5e742e6d	K5_1 = 0x7e25163d	K13_0 = 0xf6357c14	K13_1 = 0xcfccb126
K6_0 = 0x8299d2b4	K6_1 = 0x790a46ce	K14_0 = 0xa0aa6d85	K14_1 = 0xf8c10774
K7_0 = 0xea67d836	K7_1 = 0x55f354f2	K15_0 = 0x47f4fec5	K15_1 = 0x353ae1ba
K8_0 = 0xc47329fb	K8_1 = 0xf50db634	K16_0 = 0xfeccea48	K16_1 = 0xa4ef9f9b

V. 결 론

국내에서 개발된 128-bit 블록암호인 SEED 알고리즘은 기존 통신망 및 PC 보호시스템 등 각종 전산시스템과 전자상거래, 온라인 금융거래, 무역자동화 등 각종 정보통신 서비스분야의 암호기술로 고속의 정보보호 시스템분야에 활용될 수 있다. 응용분야에 따라 크기가 최소화되는 것을 필요로 하는 시스템이 있고, 암호화 속도가 최대화되는 것을 필요로 하는 시스템이 있다.

본 논문에서는 이 암호 알고리즘을 하드웨어로 구현하는데 있어서 면적을 줄이고 연산속도를 증가시키는 회로구조에 대하여 논하였고 설계결과를 기술하였다. 연산속도를 증가시키기 위해 Pipelined systolic array 구조를 사용하였으며, 입출력회로에 어떤 버퍼도 사용하지 않는 간단한 구조이다. 이 회로는 10 MHz 클럭을 사용하여 최대 320 Mbps의 암호화 속도를 달성할 수 있다.

참 고 문 헌

[1] 한국정보보호센터, 128비트 블록 암호알고리즘(SEED) 개발 및 분석 보고서, Dec. 1998.

[2] 전신우, 정용진, "128비트 SEED 암호 알고리즘의 고속처리를 위한 하드웨어 구현," 한국통신정보보호학회 논문지, Vol. 11, No. 1, pp. 13-23, Feb. 2001.

[3] 송홍복, 조경연, "SEED 형식 암호에서 S 박스와 G 함수 구성에 관한 연구," 한국통신학회 논문지, Vol. 27, No. 4A, Apr. 2002.

[4] Jenes-Peter Kaps, High Speed FPGA Architecture for the Data Encryption Standard, Master Thesis, May 1998.

[5] H. Feistel, "Block Cipher Cryptographic System," U.S. Patent, #3,798,359,19 Mar 1974.

[6] Bruce Schneuer, Applied Cryptography, Wiley, 1996

[7] William Stalings, Network and Internetwork Security, Prentice Hall International Edition, 1995

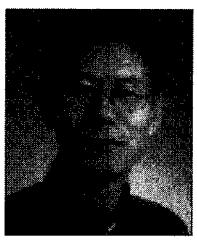
[8] Mano, M. Morris, Computer System Architecture, P.260, Prentice Hall(Sd), 1996

[9] Young-Ho Seo, Jong-Hyeon Kim, Yong-Jin Jung, and Dong-Wook Kim, "Area Efficient Implementation of 128-bit Block Cipher, SEED", ITC-CSCC 2000, Kwang-Woon Univ, Korea, 2000.

[10] 이선근, "DES의 데이터 처리속도 향상을 위한 변형된 Feistel 구조에 관한 연구", 전자공학회논문지, 제37권, 제12호, pp. 91-97, 2000.

[11] 정진욱, 최병윤, "SEED와 TDES 암호 알고리즘을 구현하는 암호 프로세서의 VLSI 설계", 대한전자공학회 하계 종합 학술대회 논문집, 제23권, 제1호, 동의대학교, 2000. 6 pp.166-172

◐ 저 자 소 개 ◑



신 광 철 (shin kwang cheul)
1985년 서울산업대학교 전자계산학과 졸업(학사)
1990년 국방대학원 전자계산학과 졸업(석사)
2003년 성균관대학교 대학원 정보공학과 졸업(박사)
2004~현재 성결대학교 E-비즈니스 IT학부 교수
관심분야 : 전자상거래보안, 라우터보안, 모바일 컨텐츠보안
E-mail : skcskc12@sungkyul.edu

이 행 우 (lee haeng woo)
1985년 광운대학교 전자공학과 졸업(학사)
1987년 서강대학교 대학원 전자공학과 졸업(석사)
2001년 전북대학교 대학원 전자공학과 졸업(박사)
2001~현재 남서울대학교 정보통신공학과 교수
관심분야 : ASIC 설계, 암호 알고리즘
E-mail : hwlee@nsu.ac.kr