개선된 F함수를 이용한 SEED 암호 프로세서의 FPGA 구현

장태민*, 전병찬, 전진오, 유수봉, 강민섭
안양대학교 컴퓨터공학과
e-mail: mskang@anyang.ac.kr

FPGA Implementation of SEED Cipher Processor Using Modified F Function

Tae-Min Chang*, Byung-chan Jun, Jeen-Oh Jun,
Su-Bong Ryu, Min-Sup Kang,
Department of Computer Engineering, Anyang University

요 약

본 논문에서는 개선된 F함수를 이용 하여 국내 표준 128비트 블록 암호화 알고리듬인 SEED 암호 프로세서의 FPGA 구현에 관하여 기술한다. 제안한 SEED 암호 프로세서는 Verilog-HDL를 사용하여 구조적 모델링을 하였으며, Xilinx사의 ISE 9.1i 툴을 이용하여 논리 합성을 수행하였다. 설계 검증은 Modelsim 6.2c 툴을 이용하여 타이밍 시뮬레이션을 수행하였으며, FPGA Prototype 시스템을 사용하여 설계된 하드웨어 동작을 검증하였다.

1. 서론

컴퓨터와 인터넷의 발전과 확대는 사용자 급속한 증가로 인한 정보 보호의 중요성을 증대시키고 있다. 정보 보호는 기밀성, 무결성, 인증 등과 같은 정보 보호 서비스들을 제공하며, 정보 보호 서비스를 제공하는 가장 일반적인 방법은 암호 시스템을 사용하는 것이다[1-6]. 암호 시스템은 대칭키와 비대칭키 암호 시스템으로 구분된다. 대칭키 암호 시스템은 암・복호화 키가 같은 암호 시스템이고, 비대칭키 암호 시스템은 암・복호화 키가 다른 암호 시스템이다. 대칭키 알고리듬은 암호화 단위에 의해 스트림 암호 알고리듬과 블록 암호 알고리듬으로 구분된다. 스트림 암호알고리듬은 비트 또는 바이트 단위로 암호화하고, 블록 암호 알고리듬은 블록 단위로 암호화한다[2].

증가하는 정보 보안의 요구에 대한 소프트웨어적인 암・복호화 처리는 데이터의 연산 및 비교 처리하여 암・복호화를 수행하는데 많은 계산이 요구된다. 소프트웨어의 특성상 처리 속도의 한계가 존재하기 때문에 이를 효율적으로 처리 할 수 있는 하드웨어의 사용이 필수적인 것이다. 하드웨어의 처리는 소프트웨어의 부담 감소와 처리속도의 향상을 얻을 수 있다. 또한 주기적인 대칭키의 변경은 보안의 효율성과 데이터 보안성 증가를 얻을 수 있다[4-6]. 설계 기법에 따라 특히 크게 차이 나는 하드웨어 구현은 암호 시스템과 같은 알고리듬 내에 동일한 동작을 하는 블록이 존재하는 경우에 그 블록을 설계하고 운영하는 기법에 따라 크게 성능 차이를 보인다.

기존 SEED 암호 알고리듬이 F함수 블록과 키 생성 블록 등으로 함께 하나의 칩에 집적되기 위해서는 적정한 처리성능을 유지하면서도 면적요구량이 최소화되는 구현이 되어야 한다. 그러나 소용량의 FPGA 구현을 위해서는 표준안 사양의 구조대로 구현할 경우에 면적요구량이 지나치게 커지게 되는 문제점이 있다[4-7].

본 논문에서는 개선된 F함수를 이용 하여 국내 표준 128비트 블록 암호화 알고리듬인 SEED 암호 프

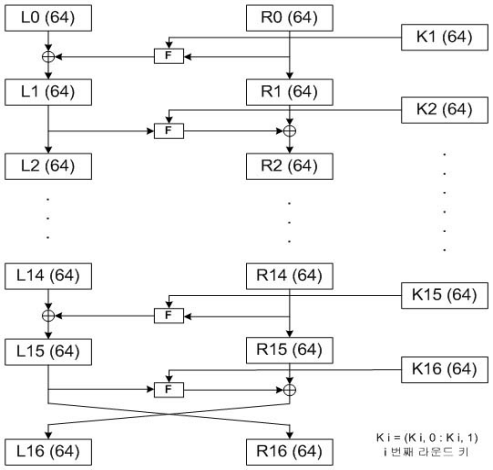
* 본 연구는 2006년도 경기중소기업청 산학연공동기술개발 컨소시엄 및 IDEC 지원으로 수행되었음.

로세서의 FPGA 구현에 관하여 기술한다. 개선된 F 함수는 G함수와 S박스의 사용 개수를 줄임으로서 면적의 감소와 처리율 향상을 얻을 수 있다. 개선된 F함수를 기본으로 한 SEED 암호 프로세서의 FPGA 구현을 위하여 Xilinx ISE 9.1i 툴을 이용하였고, 타이밍 검증은 Modelsim 6.2c 툴을 이용하였다. 또한 FPGA Prototype 시스템을 사용하여 설계된 하드웨어 동작을 검증하였다.

2. SEED 알고리듬

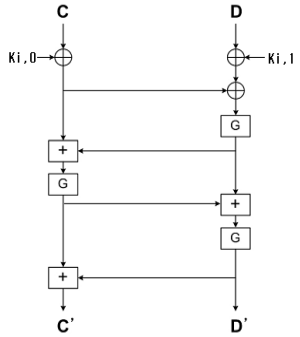
SEED는 대부분의 대칭키 알고리듬이 사용하는 Feistel 구조로 이루어져 있으며 전체적인 구조는 (그림 1)과 같다. 128비트 크기의 평문과 암호키를 입력 받은 후 평문을 각각 64비트 블록으로 나누어 암호키에 의해 생성된 각 라운드 키와 함께 F함수를 통해 암호화가 수행된다. 모두 16번의 라운드를 수행한 후 최종 128비트 암호문을 출력한다[2].

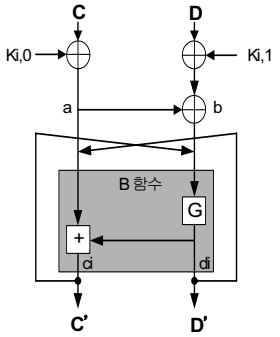
(그림 1)은 SEED 알고리듬의 전체구조를 나타낸다.



(그림 1) SEED 알고리듬의 전체구조

(그림 2)는 기존의 SEED 알고리듬에 대한 F함수의 구조를 나타내며, (그림 3)은 개선된 F함수의 구조를 나타낸다.





(그림 2) 기존의 F함수 구조, (그림 3) 개선된 F함수의 구조

(그림 2)에서 알 수 있듯이 32비트 단위의 2개의 블록(C, D)을 입력으로 받아, 32비트 단위의 2개의 블록(C′, D′)을 각각 출력한다[2].

(그림 3)은 기존 F함수에 사용된 3개의 G함수를 한 개의 단일 블록(B함수)의 구조를 갖는다.

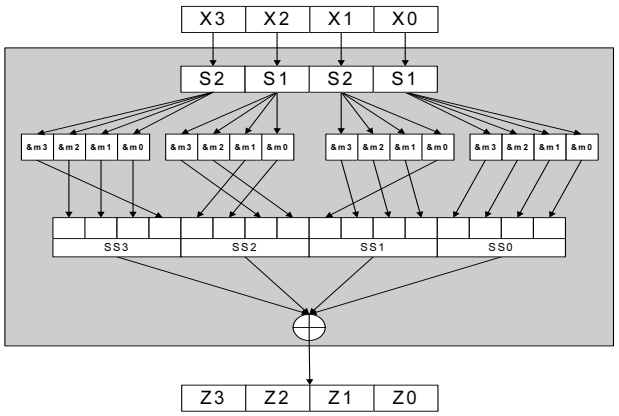
최초 a, b 입력은 각각 a = C⊕Ki0, b = (C⊕Ki0) ⊕(D⊕Ki1) 연산을 거친 후에 B함수 연산을 통하여 Ci, Di 값을 출력한다. 그 후 동일한 알고리듬의 B함수를 2회 교차 입력을 실행하여 최종 C′, D′ 값을 얻을 수 있다.

개선된 F함수의 출력을 위한 식은 다음과 같다.

- C0 = B(a,b), D0 = B(b)
- Ci = B(Di-1, Ci-1)
- Di = B(Ci-1)

여기서 i = 라운드, B는 B함수를 나타낸다.

(그림 4)는 기존의 G함수의 구조를 나타낸다[2].



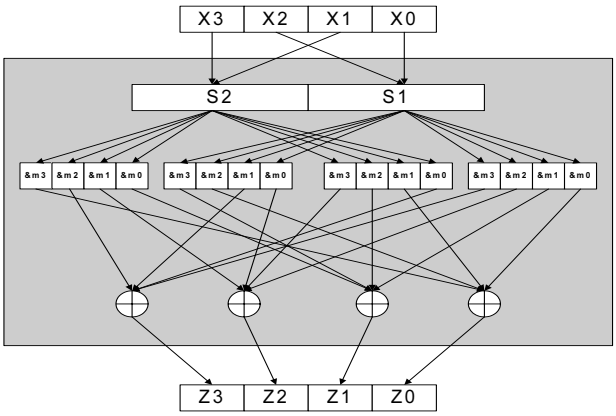
(그림 4) G함수의 구조

G함수는 F함수 및 라운드 키 생성시에 사용되는 주요 함수이다. 위의 G함수는 4개의 확장된 4바이트 SS박스들(4Kbytes)의 XOR로 구현할 수 있다. 4개의 SS박스를 출력을 위한 부울식은 다음과 같다[2].

$SS0 = S_1(x_0)\&m_3 \,||\, S_1(x_0)\&m_2 \,||\, S_1(x_0)\&m_1 \,||\, S_1(x_0)\&m_0$
$SS1 = S_1(x_0)\&m_0 \,||\, S_1(x_0)\&m_3 \,||\, S_1(x_0)\&m_2 \,||\, S_1(x_0)\&m_1$
$SS2 = S_1(x_0)\&m_1 \,||\, S_1(x_0)\&m_0 \,||\, S_1(x_0)\&m_3 \,||\, S_1(x_0)\&m_2$
$SS3 = S_1(x_0)\&m_2 \,||\, S_1(x_0)\&m_1 \,||\, S_1(x_0)\&m_0 \,||\, S_1(x_0)\&m_3$

단, m_0 = 0xfc, m_1 = 0xf3, m_2 = 0xcf, m_3 = 0x3f 이고, || 는 concatenation이다. 그리고 &는 bit-wise AND 연산을 의미한다.

(그림 5)는 개선된 G함수 구조를 나타낸다.



(그림 5) 개선된 G함수 구조

개선된 G함수의 연산은 X3와 X2 비트가 각각의 S박스를 연산하여 레지스터에 저장되고, X1과 X0 비트가 동일한 S박스 연산을 통하여 레지스터에 저장된다. 저장된 레지스터를 마스킹 후 XOR 하여 최종 Z값을 출력한다.

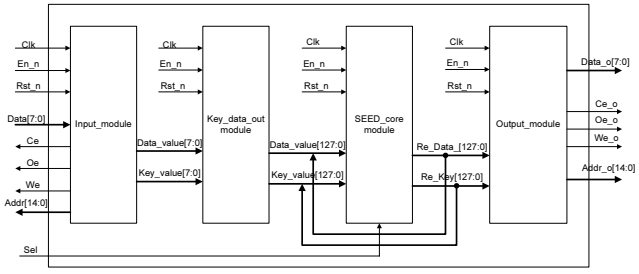
개선된 G함수는 다음과 같이 유도 할 수 있다.

$$Y_3 = S_2(X_3),\ Y_2 = S_1(X_2),\ Y_1 = S_2(X_1),\ Y_0 = S_1(X_0)$$
$$Z_3 = (Y_0\ m_3)\oplus(Y_1\ m_0)\oplus(Y_2\ m_1)\oplus(Y_3\ m_2)$$
$$Z_2 = (Y_0\ m_2)\oplus(Y_1\ m_3)\oplus(Y_2\ m_0)\oplus(Y_3\ m_1)$$
$$Z_1 = (Y_0\ m_1)\oplus(Y_1\ m_2)\oplus(Y_2\ m_3)\oplus(Y_3\ m_0)$$
$$Z_0 = (Y_0\ m_0)\oplus(Y_1\ m_1)\oplus(Y_2\ m_2)\oplus(Y_3\ m_3)$$

기존 G함수에 비해 개선된 G함수 구조는 S1, S2박스의 개수를 줄임으로서 처리속도가 빠를 뿐만 아니라 하드웨어 오버헤드를 줄일 수 있다.

3. 개선된 SEED 암호 프로세서 설계

(그림 6)은 개선된 F함수를 이용한 SEED 암호프로세서의 Top 블록을 나타낸다.

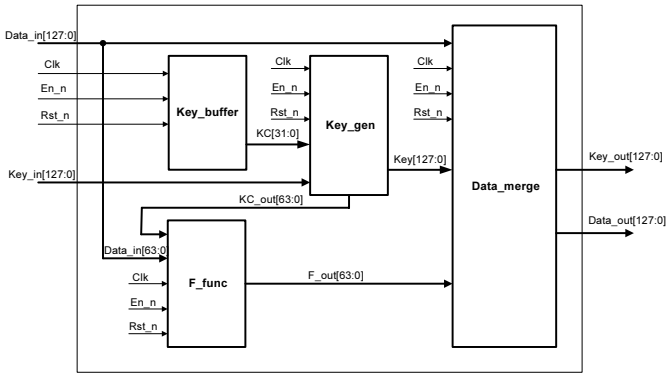


(그림 6) SEED 암호 프로세서의 Top 블록

(그림 6)에서 알 수 있듯이 SEED의 Top 블록은 4개의 모듈, 즉 Input_module, Key_data_out_module, SEED_core_module, 그리고 Output_module로 구성된다. Input_module은 외부 RAM으로부터 8비트 데이터 입력을 받아 8비트의 데이터와 키 값을 구분하여 출력한다. Key_data_out_module은 Input_module로부터 8비트 데이터와 키 값을 입력받아 128비트의 데이터와 키 값으로 변환하여 출력한다.

SEED_core_module은 라운드 키 생성부와 라운드 계산부로 나누며, Sel 신호에 의하여 암호화/복호화 동작을 결정한다. Output_module은 외부 RAM에 데이터를 저장하기 위하여 128비트 데이터와 키 값을 입력받아 8비트 데이터로 변환하여 출력한다.

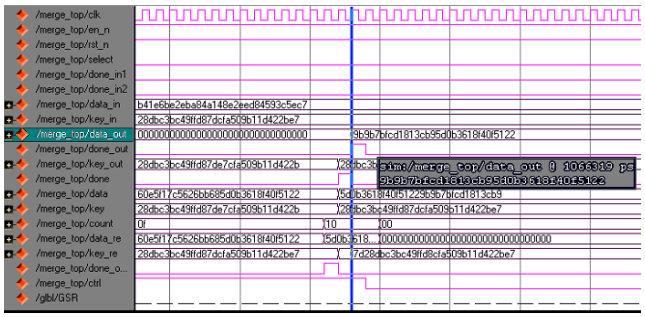
(그림 7)에서 SEED_core_module의 구조는 Key_buffer, Key_gen, F_func, 그리고 Data_merge 블록으로 구성된다.



(그림 7) SEED_core_module의 구조

(그림 7)에서 Key_buffer 블록은 각 라운드에 사용되는 32비트 KC 값을 생성하여 출력한다. Key_gen 블록은 라운드 키 상수 값을 입력 받아 라운드 키를 출력한다. F_func 블록은 64비트의 데이터와 64비트의 라운드 키를 이용하여 16라운드를 수행한 후, Data_merge 블록을 통해 128비트 데이터와 128비트 키를 출력하는 구조를 나타낸다.

(그림 8)은 암호화의 과정을 검증하기 위한 타이밍 시뮬레이션 결과를 나타낸다.

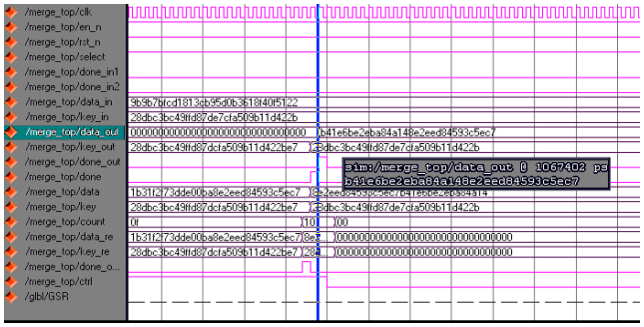


(그림 8) 암호화 과정 시뮬레이션

사용된 입력값은 평문 "b41e6be2eba84a148e2eed84593c5ec7" 이며, 키값은 "28dbc3bc49ffd87dcfa509b11d422be7" 이다[10].

시뮬레이션 결과로부터 암호화된 "9b9b7bfcd1813cb95d0b3618f40f5122" 결과값을 얻을 수 있었다.

(그림 9)는 SEED의 복호화 과정의 시뮬레이션을 나타낸다.



(그림 9) 복호화 과정 시뮬레이션

입력값으로 암호화된 "9b9b7bfcd1813cb95d0b3618f40f5122" 와 키 값 "28dbc3bc49ffd87dcfa509b11d422be7"을 사용하여 복호화 된 결과 값 "b41e6be2eba84a148e2eed84593c5ec7" 출력되는 것을 확인하였다.

4. 시스템 구현 및 성능평가

본 논문에서 제안된 암호 프로세서의 각 모듈은 Verilog-HDL을 이용하여 설계하였다. 또한, Xilinx ISE 9.1i 툴을 이용하여 합성을, 설계 검증을 위한 타이밍 시뮬레이션은 Modelsim을 이용하였고, FPGA는 Xilinx사의 Spartan-II(XC2S200-5PQ208)를 타겟으로 시스템을 구현하였다.

본 논문에서 제안한 개선된 F함수와 기존 F함수의 성능을 비교 분석한 결과는 다음과 같다.

기존 F함수를 합성한 결과 23,194개의 Gates를 차지한 반면, 개선된 F함수는 8,890개의 Gates로 약3배의 면적이 감소하였고 동작 속도가 개선되었다.

본 논문에서 개선된 F함수를 기본으로 한 SEED 암호프로세서의 구현 결과 전체 블록 합성결과는 총 1,755개의 Slices 사용하였다. 최대 동작 클럭 속도는 45.4MHz이고, Throughput은 360Mbps(128bits * 45MHz / 16) 였다.

<표 1> SEED 암호 프로세서 성능 비교

Item / Method	Gates	Clock Freq.(MHz)	Throughput (Mbps)	Process
Pipelined[4]	144,909	38	304	Xilinx FPGA
Iterative[4]	52,839	33	264	Xilinx FPGA
Iterative[5]	35,397	10.202	82	Xilinx FPGA
Proposed	30,325	45	360	Xilinx FPGA

<표 1>의 SEED 암호 프로세서 성능 비교에서 알 수 있듯이 Throughput은 기존 Iterative [4,5] 방식보다 대폭 개선되었음을 알 수 있다. 또한 Gates 수의 감소로 면적이 대폭 줄었음을 보여준다.

5. 결론

본 논문에서는 개선된 F함수를 기본으로 한 SEED 암호 프로세서의 구현에 관하여 기술하였다. Pipelined 방식은 병렬로 처리하여 높은 성능을 처리하는 반면 많은 면적을 필요하지만, Iterative 방식으로 개선된 F함수를 기본으로 한 SEED 암호 프로세서는 적은 면적으로 성능 저하 문제를 극복하였다.

제안된 암호 프로세서는 Verilog-HDL을 사용하여 구조적 모델링을 행하였으며, Xilinx사의 ISE 9.1i 툴과 Modelsim 툴을 이용하여 시뮬레이션 및 합성을 수행하였다.

향후 F함수 부분에서의 G함수와 덧셈기의 개수를 줄이고 효율적인 병렬 처리를 통하여 시스템의 성능을 상당히 높일 수 있을 것으로 기대된다.

참고문헌

[1] 인터넷보안기술포럼 "Implementation Technology for secure VPN in IP Layers", 2001.

[2] 한국정보보호센터, "128비트 블록 암호알고리듬(SEED) 개발 및 분석보고서", KISA, 2003.

[3] an Daemen, Vincent Rijmen, "AES Proposal Rijndal", http://csrc.nist.gov/encryption/aes.

[4] 이광호, 남승용, 강민섭, "개선된 LUT 방식을 이용한 SEED 알고리듬의 FPGA 구현", 대한전자공학회, SOC설계발표논문집, 2004.

[5] 김영미, 이정엽, 전은아, 정석원, "SEED Triple-DES 전용 암호칩의 설계 및 구현", 한국사이버테러정보전학회 춘계학술발표대회, 2004.

[6] 전신우, 정용진, "128비트 SEED 암호 알고리즘의 고속처리를 위한 하드웨어 구현", 한국통신정보보호학회논문지, 2001.

[7] 엄성용, 이규원, 박선화, "SEED 블록 암호 알고리즘의 파이프라인 하드웨어 설계", 한국정보과학회 논문지, 2001.

[8] 채봉수, 김기용, 조용범, "Pipeline 구조의 SEED 암호화 프로세서 구현 및 설계", 대한전자공학회, 하계종합학술대회 논문집, 2002.

[9] 참조구현값, 한국정보통신기술협회.