IDEA 암호 알고리즘의 FPGA 구현

송 영 아, 장 경 선
한남대학교 컴퓨터공학과
전화 : (042)629-7983 / 팩스: (042)629-7658

A FPGA Implementation of IDEA Algorithm

Young-Ah Song, Kyoung-Son Jhang
Dept. of Computer Engineering, Hannam University
E-mail : songya@ce.hannam.ac.kr

Abstract

In this paper, we present a FPGA implementation of IDEA algorithm. Target technology is Altera FLEX 10K FPGA. The correctness of the implementation is verified by the timing simulation with max+plus II.

I. 서론

통신 시스템의 발달로 많은 양의 정보를 공유하게 되면서, 인가받지 못한 제 3자에 의한 정보의 유출이 심각한 문제로 떠오르고 있다. 이러한 문제의 해결책으로, 허용된 특정인 이외에는 알아볼 수 없는 형태로 정보를 전송하여 접근이 허가되지 않은 이들에 의한 정보의 불법적인 사용을 막고 있다.

따라서 이제까지 DES(Data Encryption Standard)를 비롯한 많은 암호화 알고리즘이 만들어졌다. 하지만 주로 기존의 CPU를 이용하고 있으며 실질적인 하드웨어 칩을 이용한 접근은 많지 않다. 최근에 인터넷을 이용한 전자상거래가 늘어나면서 암호화해야 하는 정보의 양도 증가하였고, 이런 대량의 데이터가 암·복호화되어 사용자에게 전송되는 경우에 CPU가 사용된다면 부하가 과중하게 된다. 이를 위해서 별도의 암·복호화 칩을 이용한다면 CPU의 부하를 상당히 줄일 수 있을 것이다[1].

현재 가장 보편적으로 실용화되어 사용되는 암호 알고리즘은 IBM의 Lucifer 알고리즘을 기반으로 개발한 DES이다. 64 비트의 데이터를 암호화하기 위해 64 비트의 키를 사용하는 DES는 미국 상무성 표준으로 채택되어 사용되어 왔으나 키 길이가 짧은 것이 문제점으로 지적되어 왔다[2]. 따라서 DES를 대체하기 위한 알고리즘이 제안되었고, 그 중의 하나가 IDEA이다.

IDEA(International Data Encryption Algorithm)는 스위스 연방 기술 기관의 Xuejia Lai와 James Massey가 개발한 새로운 블록 암호 알고리즘이며, 64 비트 블록에서 데이터를 암호화하기 위해 64 비트 키를 사용하는 DES와는 달리, 64 비트 블록에 대해 128비트 키를 사용한다. 블록 길이는 통계적 분석을 막을 수 있을 만큼 길어야 하고 키의 길이 역시 방대한 키 검색을 효율적으로 막을 만큼 충분히 길어야 하는데, 일반적으로 64 비트의 블록 크기와 128 비트 키 길이라면 충분히 강력하다고 여겨진다. 또한 각 평문 비트가 모든 암호문 비트에 영향을 미치는 것을 확산(diffusion)이라 하는데, IDEA는 round 부분의 곱셈/덧셈(MA) 구조를 통해 매우 효율적인 확산으로 높은 암호학적 강도를 제공한다[3].

본 논문에서는 IDEA 알고리즘을 Altera 10K를 target으로 VHDL(Very high speed integrated circuits Hardware Description Language)을 이용하여 구현하였다. 논문은 다음과 같이 구성되었다. 제 2 절에서는 구현된 IDEA의 개략적인 구조 및 내부 블록 구성과 기능에 관해 설명한다. 제 3 절에서는 타이밍 시뮬레이션 결과를 통해 설계된 회로의 소프트웨어적인 검증과 분석을, 마지막으로 제 4 절에서는 결론과 향후 연구 방향을 기술한다.

II. IDEA

IDEA 는 64 비트로 이루어진 블록을 한꺼번에 암호화하는 블록 암호화 방식 중의 하나로서, 전체 구조는 크게 <그림 1>과 같이 제어부(controller), 키 스케쥴러(key scheduler), 라운드(round), 이렇게 세 부분으로 나누어질 수 있다.

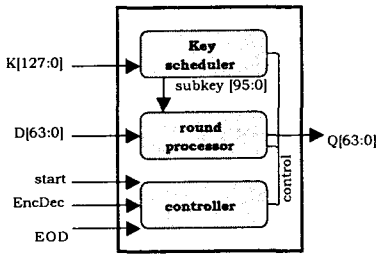

그림 1 IDEA의 기본 구조[4]

1. 키 스케쥴러

IDEA는 암호화와 복호화를 위한 서브키가 각각 다르다. 암호화를 위한 서브키는 128 비트의 키를 입력받아 52개의 16 비트 서브키가 만들어진다. <표 1>의 처음 8개의 서브키(Z_1-Z_8)는 입력받은 128 비트 키가 그대로 적용되며, 이를 25 비트 왼쪽 순환 쉬프트 시키면 다음 8개의 서브키를 얻을 수 있다. 이 절차를 52개의 서브키들이 추출될 때까지 반복한다. 복호화를 위한 서브키는 생성된 암호화 서브키를 이용하여 multiplication mod 2^{16}+1과 addition mod 2^{16}의 역원을 취하여 얻어진다.

표 1 IDEA의 키스케쥴링 방식[3]

	암호화	복호화
상태	선정	선정
반복1	$Z_1Z_2Z_3Z_4Z_5Z_6$	Z_{49}^{-1}-Z_{50}-$Z_{51}Z_{52}^{-1}Z_{47}Z_{48}$
반복2	$Z_7Z_8Z_9Z_{10}Z_{11}Z_{12}$	Z_{43}^{-1}-Z_{45}-$Z_{44}Z_{46}^{-1}Z_{41}Z_{42}$
반복3	$Z_{13}Z_{14}Z_{15}Z_{16}Z_{17}Z_{18}$	Z_{37}^{-1}-Z_{39}-$Z_{38}Z_{40}^{-1}Z_{35}Z_{36}$
반복4	$Z_{19}Z_{20}Z_{21}Z_{22}Z_{23}Z_{24}$	Z_{31}^{-1}-Z_{33}-$Z_{32}Z_{34}^{-1}Z_{29}Z_{30}$
반복5	$Z_{25}Z_{26}Z_{27}Z_{28}Z_{29}Z_{30}$	Z_{25}^{-1}-Z_{27}-$Z_{26}Z_{28}^{-1}Z_{23}Z_{24}$
반복6	$Z_{31}Z_{32}Z_{33}Z_{34}Z_{35}Z_{36}$	Z_{19}^{-1}-Z_{21}-$Z_{20}Z_{22}^{-1}Z_{17}Z_{18}$
반복7	$Z_{37}Z_{38}Z_{39}Z_{40}Z_{41}Z_{42}$	Z_{13}^{-1}-Z_{15}-$Z_{14}Z_{16}^{-1}Z_{11}Z_{12}$
반복8	$Z_{43}Z_{44}Z_{45}Z_{46}Z_{47}Z_{48}$	Z_7^{-1}-Z_9-$Z_8Z_{10}^{-1}Z_5Z_6$
변환	$Z_{49}Z_{50}Z_{51}Z_{52}$	Z_1^{-1}-Z_2-$Z_3Z_4^{-1}$

이 블록에서는 EncDec 비트를 두어 '0'이면 암호화 서브키를 생성하고, '1'이면 암호화 서브키를 생성한 후 이를 이용하여 복호화 서브키를 생성한다. 이렇게 하여 생성된 암호화 서브키와 복호화 서브키를 64 비트의 데이터가 들어올 때마다 사용하려면 저장해야 하는데, 여기서는 레지스터 대신 Altera 칩에 내장되어 있는 EAB(Embedded Array Block)를 사용하기 위해 dual-port RAM으로 설계하였다. start 신호가 들어오면 서브키를 만들기 시작하고, 생성된 서브키는 키 스케쥴러 블록 안에 있는 RAM 에 저장된다. 라운드 블록에서 한 번에 사용되는 서브키는 16 비트 6개이다. 따라서 RAM은 한 번에 16 비트의 입·출력이 가능한 dual-port RAM 6개를 사용하였다.

2. 라운드

<그림 2>는 한 라운드가 수행되는 연산 과정을 나타낸 것이다. IDEA의 라운드 구조는 한 라운드를 수행시킨 결과가 다음 라운드의 입력으로 사용되며, 이 라운드를 총 8회 반복한 후 마지막 변환 단계를 거쳐 암·복호화가 이루어진다.

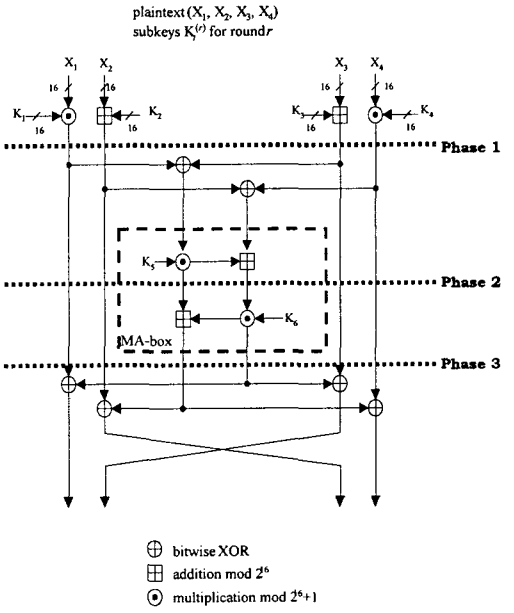

그림 2 IDEA의 한 라운드[5]

IDEA의 라운드 블록은 입력의 복잡한 변환을 위해 xor(exclusive-OR), multiplication mod 2^{16}+1, addition mod 2^{16}, 이렇게 세 가지 연산을 사용하는데, 이는 xor 함수에만 의존하는 DES보다 암호 해독을 더 어렵게 하기 위해서이다. 또한 효율적인 확산 효과를 제공하기 위해 <그림 2>의 점선 안에 보여지는 곱셈/덧셈(MA) 구조를 사용하는데, 이는 라운드의 각 결과가 서

브키의 모든 비트와 평문에서 유도된 입력의 모든 비트에 의존하는지를 결정한다[3].

VHDL로 설계시에는 칩의 면적을 줄이기 위해, 8 라운드 전체를 만드는 대신 1 라운드만을 설계하여 라운드의 출력을 다시 입력으로 사용하는 방식으로 하였다. 제어 블록에서 서브키가 다 만들어지면 Ciphering 신호를 보내는데, 이 신호가 '1'이 되면 라운드를 시작한다. 라운드는 64 비트 데이터와 6개의 16 비트 서브키를 입력으로 받아, 8번의 라운드 반복과 변환 과정을 거쳐 64 비트의 출력을 내보낸다.

라운드의 multiplication mod 2^{16}+1 연산은 Altera 칩의 LC(Logic Cell) 수를 많이 차지한다. 따라서 한정되어 있는 LC 수를 최대한 활용하기 위해, <그림 2>의 직선으로 그어진 점선처럼 라운드를 3 부분으로 나누어 라운드의 데이터가 들어가는 부분과 MA 부분의 multiplication mod 2^{16}+1 연산자를 공유한다.

3. 제어부

이 블록은 키 스케쥴러 블록과 라운드 블록에 제어 신호를 제공하는 역할을 수행한다. 먼저 암·복호화의 시작을 알리는 start 신호가 들어오면 EncDec 신호를 보고 '0'이면 암호화를, '1'이면 복호화를 시작한다. 서브키가 다 생성되면 라운드 블록에 Ciphering 신호를 보내는데, 라운드 블록은 이 신호가 '1'이 되면 키 스케쥴러 블록의 RAM에 저장되어 있는 서브키를 읽어와서 암·복호화를 시작한다. 데이터는 64 비트 단위로 받아들여 64 비트 암·복호화된 값을 생성하고, 이 절차는 외부에서 데이터의 끝을 알리는 EOD(End of Data) 신호가 '1'이 입력될 때까지 계속 반복된다. 서브키가 생성되었거나 입력된 64 비트의 데이터가 암·복호화 되면, 데이터를 입력받을 준비가 되었다는 Ready 신호를 보내 데이터를 입력받는다.

III. 실험 결과 및 분석

1. 시뮬레이션 결과

설계한 IDEA 알고리즘은 Altera FLEX 10K의 셀 라이브러리를 사용하여 합성하였으며, Altera MAX+plus II를 사용하여 타이밍을 고려한 시뮬레이션 값이 c 원시 코드와 같게 나오는 것을 확인하였다.

64 비트 데이터(0000 0001 0002 0003 : HEX)와 128 비트 키(0001 0002 0003 0004 0005 0006 0007 0008 : HEX)를 입력하였을 때, 64 비트 암호화된 출력값(11fb

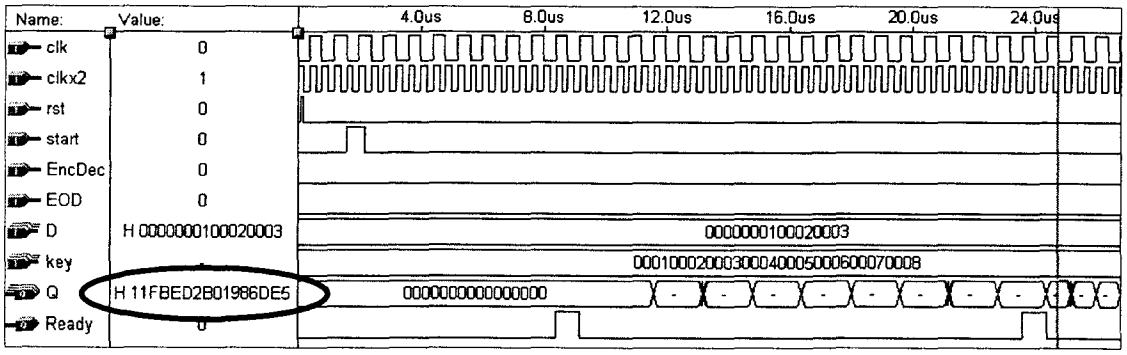

그림 3 암호화된 결과 값

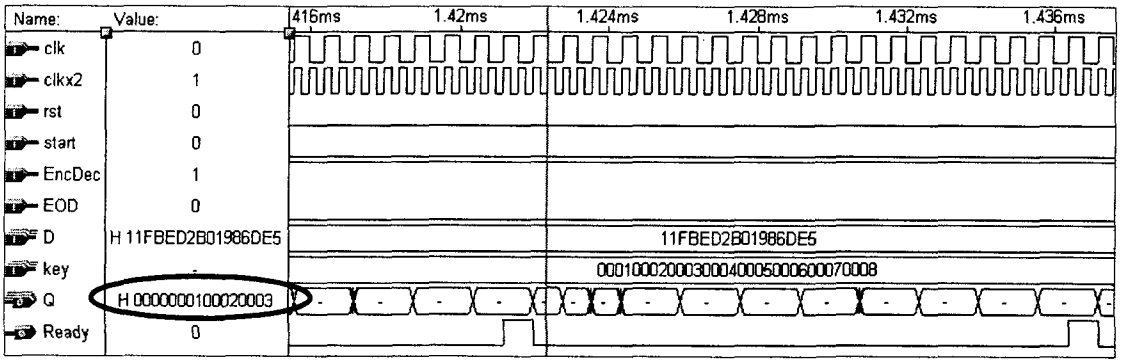

그림 4 복호화된 결과 값

ed2b 0198 6de5 : HEX)값을 얻을 수 있었으며, 시뮬레이션 결과는 <그림 3>과 같다. <그림 4>는 암호화된 출력값을 복호화하였을 때, 원래의 데이터값이 나오는 것을 볼 수 있다.

<그림 5>는 IDEA의 MAX+plus II에서 생성된 최상위 레벨 심볼이며, 합성 결과는 <표 2>와 같다.

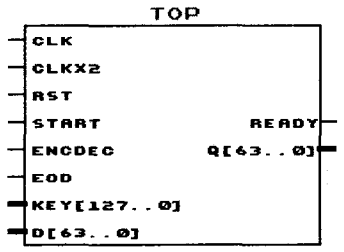

그림 5 생성된 최상위 레벨 심볼

표 2 합성된 IDEA 칩의 결과

Block Name	LC 수	EAB 수
key scheduler	1,835	12,288
round	2,170	-
controller	1,789	-

2. 타이밍 분석

IDEA의 라운드 블록에서 사용하는 연산자 multiplication mod 2^{16}+1, addition mod 2^{16}, xor는 각각 최대 204 ns, 61 ns, 21 ns의 지연 시간을 갖는다. 실질적인 속도를 내는 라운드 블록을 최적화하기 위해 <그림 2>와 같이 3 부분으로 나누어 구현하였다.

<표 3>은 Max+plus II의 타이밍 분석기(Timing Analyzer)를 사용한 worst case 분석 결과이다. 스케쥴링 방법의 변화에 따른 면적과 속도간의 trade-off를 위해, 1 라운드 블록을 1 클럭 주기에 수행되도록 설계한 경우(case 1), phase 1과 phase 2, 3으로 나누었을 경우(case 2), phase 1, 2, 3으로 나누어서 설계하였을 경우(case 3)를 비교해 보았다. case 1에서는 1초에 처리하는 데이터 양은 많았지만 multiplication mod 2^{16}+1 연산자와 addition mod 2^{16} 연산자를 공유하지 않고 사용하므로 LC 수에 있어서 최대가 된다. case 2에서는 라운드 블록이 phase 1의 결과값이 생성된 후에 phase 2와 3이 수행된다는 특성을 고려하여 phase 1을 1 클럭 주기에, phase 2와 3을 다음 클럭 주기에 수행되도록 하여 곱셈기와 덧셈기를 공유할 수 있도록 설계하였다. 따라서 LC 수는 감소하였지만, 클럭 주기가 여전히 크다는 것을 알 수 있다. 본 논문에서 설계한 case 3의 방법은 case 2의 방법에서 phase 2와 3을 다른 클럭에 수행하여 이전 곱셈과 덧셈에서 데이터가 나올 때까지 기다리는 시간을 줄였다. 따라서 1 라운드 수행 시간이 case 2보다 적으면서 나머지 방법들에 비해 적절한 LC 수를 가짐을 알 수 있다. <표 2>에 제시된 라운드에 대한 LC 수는 case 3에 의한 것이다.

표 3 라운드 스케쥴링 방법에 따른 성능 비교

	case 1	case 2	case 3
데이터 처리량	16M bit/sec	8M bit/sec	10M bit/sec
LC 수	3,572	2,122	2,170
클럭 주기	640.6 ns	470.6 ns	274.2 ns

V. 결론

본 논문은 IDEA 알고리즘을 Altera FLEX 10K를 target으로 구현하기 위하여 VHDL로 설계한 것이다. IDEA는 암호화와 복호화 키가 생성되는 구조가 다르다는 점을 제외하면 데이터 암·복호화 절차가 같다.

본 논문에서는 블록 암호화 알고리즘의 실질적인 속도를 내는 라운드 블록을 LC 수가 제한된 FPGA에 구현하기 위해 연산자 공유를 통해 면적을 줄이는 방향으로 설계하였다. 곱셈기의 성능을 개선한다면, 처리량의 향상이 기대된다.

참고문헌

[1] 류승석, 오재곤, 정연모, "GOST 알고리즘을 이용한 암호화칩 설계", CAD 및 VLSI 설계 연구회 학술발표회 논문집, pp. 49-54, 1997

[2] 이재철, 강민섭, "3중 DES 알고리듬의 FPGA 설계 및 구현", 한국정보처리학회 춘계 학술발표논문집 제6권 제1호, pp. 820-823, 1999

[3] 최용락, 소우영, 이재광, 이임영, 통신망 정보보호, 그린출판사, pp. 306-317, 1996

[4] http://www.mentorg.com/inventra/

[5] Alfred J. Menezes, Paul C.van Oorschot, and Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, pp. 263-266