FPGA를 이용한 IDEA의 설계 및 구현

이상덕*, 이계호**, 한승조*
조선대학교 전자·정보통신공학부 정보통신시스템설계 Lab.*
한국통신 인력개발본부**

Design and Implementation of IDEA Using for FPGA

Sang Duck Lee*, Kye ho Lee**, Seung Jo Han*
*School of Electronic and Information communication Eng. Chosun Univ.
**Korea Communication

개 요

본 논문에서 DES를 대체하기 위해 몇 년에 걸쳐 제안된 관용 암호알고리즘의 하나인 IDEA(International Data Encryption Algorithm)의 구현을 제안하고자 한다. IDEA의 암호화 수행시간의 개선을 위하여 VHDL(VHSIC Hardware Description Language)을 이용하여 하드웨어로 설계하였고 설계된 알고리즘은 EDA tool인 Synopsys를 사용하여 Synthesis하였으며, Xilinx의 FPGA XC4052XL을 이용하여 One Chip화 시켰다. 입력 클럭으로 30Mhz를 사용하였을 때, data arrival time은 780.09ns였으며, 80.01 Mbps의 속도로 동작하였다. 본 논문은 설계 언어로서 VHDL을 사용하였고, FPGA Chip에 구현하여 동작 확인을 하였다.

Ⅰ. 서론

통신 기술과 컴퓨터 네트워크 기술의 발달은 대량의 정보를 신속 정확하게 처리하여 전송함으로써 정보의 공유 개념을 가져오게 되었다. 이로인해 개인이 컴퓨터를 이용한 정

보의 송수신이나 건물 내에서의 통신을 이용한 결제 및 다른 곳으로의 유용한 데이터의 전송이 손쉽게 되었다. 정보사회에서 정보의 신속한 전송 및 보관 또는 비인가자로부터의 정보의 보호가 중요한 문제로 대두되고 있다. 이러한 무형적인 정보가 저장중이거나 통신망을 통해 전송 될 때 불법 침입자로부터 데이터를 보호하고 도청이나 내용의 변조를 막기 위해서는 직접적으로 정보를 암호화하는 것이 절실히 요구되게 되었으며 통신선로 상에서 전송중이거나 보관중인 정보의 직접적인 보호문제도 크게 대두되고 있다.

암호화(encryption)란 키를 사용하여 데이터를 무의미하게 나타나도록 스크램블(scramble)하는 것을 말하며, 이 무의미한 데이터를 복호 과정을 통하여 원래의 데이터로 복원시킬 수 있게 보호하는 과정을 말한다. 암호방식은 대칭키(비밀키)방식과 비대칭키(공개키)방식으로 분류할 수 있으며, 대칭키 암호방식은 하나의 키를 암호화 및 복호화에 공유비밀키로 사용한다. 그리고 이 방식은 쌍방간에 고유비밀키를 분배할 수 있는 안전한 비밀키 분배 방법이 필요하다. 비대칭키 암호방식은 암호화 및 복호화에 쌍으로 이루어진 두 개의 키를 사용하여, 하나는 공개키로, 다른 하나는 비밀키로 운용한다.

Ⅱ. IDEA알고리즘의 구조

본 논문에서 제안한 알고리즘인 IDEA는 스위스의 연방 기술 연구소의 Xuejia Lai와 James L. Massey에 의해 개발되었으며, 1990년에 발표되었고, 1991년에 알고리즘상의 단점을 보완하여 발료하였다. DES처럼, IDEA는 64비트의 데이터 블록을 암호화하기 위해 하나의 비밀키를 사용하는 비밀키 암호화 알고리즘으로써 동일한 키 값이 원래의 64비트 클리어 테스트로 복구시키기 위해 해독하는데 다시 사용된다.

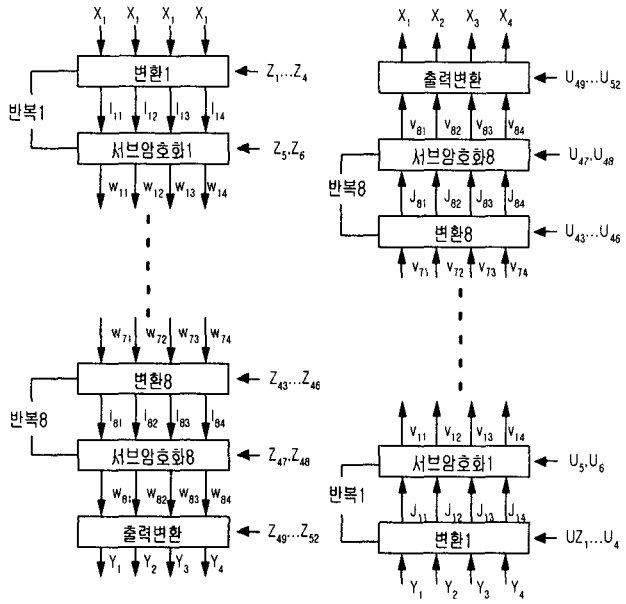


그림 1. IDEA의 암·복호화 과정

IDEA 알고리즘은 일련의 라운드로 작동하며, 그것은 완전한 암호 키로부터 각 라운드에 생성된 하나의 서브키를 포함한다. 또한 DES처럼, 이른바 맹글러(mangler) 기능이 각 라운드에서 비트들을 스크램블하기 위해 사용된다. 일반적으로 사용되는 치환은 사용되지 않으므로 하드웨어만큼 소프트웨어적으로도 쉽게 구현할 수 있다.

Ⅲ. IDEA 블록의 설계

IDEA 블록의 전반적인 설계과정은 다음과 같이 나타낼 수 있다. 각각의 블록을 VHDL을 사용하여 설계한 후, Function Simulation을 거쳐 기능적인 동작검증을 마치게 된다. Function Simulation 과정을 마친 각 블록들은 Synthesis를 거쳐 실제 회로로 만들어지게 되고, FPGA상에 Place & Route 시키게 된다. 이러한 과정을 거쳐 SDF File을 생성하고 Timing Simulation을 수행한다. 실제의 Delay를 고려한 Simulation을 수행하여 최종적인 동작을 확인하게 된다. Timing Simulation이 끝난 회로는 FPGA상에 Verification된다. FPGA는 사용자가 직접 회로를 구현하여 사용할 수 있는 집적회로로서 칩 상에 미리 제작된 배선과 논리 블록들을 EPROM에 프로그래밍함으로써 원하는 기능의 회로을 구현할 수 있다. FPGA를 이용하면 논리 회로를 설계할 때 칩의 제조 공정을 거치지 않고, FPGA를 사용하여 Emulation을 할 수 있기 때문에 칩의 설계 및 제작 과정에서 많은 시

간과 비용을 절약할 수 있다. FPGA는 프로그래밍 가능한 CLB(Configuration Logic Block) 블록과 CLB 사이의 연결 형태를 지정할 수 있는 배선 채널 및 외부와의 입출력을 담당하는 입출력 블록으로 구성되어 있고, 프로그래밍에서는 논리 블록들의 기능을 지정해 주고 배선 채널 및 입출력 블록들의 연결 상태를 지정해주게 된다. 이렇게 원칩으로 제작된 FPGA를 이용하여, 집적도나 수행속도 등을 검증하여 더욱 개선된 Modeling을 할 수 있다. 본 논문에서는 IDEA의 블록을 크게 Control Block, Key Block, Encryption Block의 세 Block으로 나누었다.

1. Control Block의 모델링

control blcok은 암호기 전체를 제어하는 controller와 현재 동작중인 Round를 계수하는 RoundCounter, 세 개의 cycle을 가지는 클럭을 발생하는 State Gen으로 구성되어있다. 외부에서 입력된 클럭은 세 개의 주기를 가지는 t1, t2, t3로 나눈어지며, 전체적인 동작제어는 enable과 rst를 통하여 제어한다. 설계에 쓰여진 알고리즘은 8Round 모두를 구현하는게 아니고, 단일 라운드만을 구성하여 출력을 입력으로 feedback시켜 8회에 거쳐 동작을 시키게 된다. 그러므로 암호화 시에는 수행중일 Round 수를 계수할 필요가 있고, RoundCounter는 암호화, 복호화, 그리고 키 생성시 수행중인 round를 지시하며, 동작이 완료되면 round_end 신호를 발생시킨다. round_end 신호가 '1'이 되면 모든 블록의 동작이 중지된다. Top-Level의 동작시 모든 블록들은 배타적인 동작을 수행할 필요가 있다. keyblk와 encryblk가 동시에 동작하게 되면, 블록에서 소요되는 시간에 의해 서로 동기가 맞지 않고 입력키의 오류를 일으켜 잘못된 연산을 수행할 우려가 있다. 클럭 발생부는 3개의 각기 다른 클럭을 발생시키고, Keybox나 제어부에 사용되어질 클럭은 가장 빠른 클럭을 사용하고, Encryption 부에

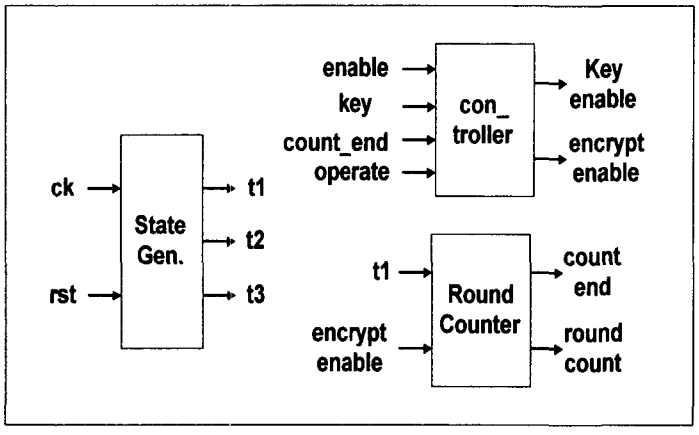


그림 2 Control Block의 블록도

는 다음 순위의 클럭이 사용된다. 실제 설계에 사용된 클럭발생기는 외부의 클럭을 받아들이어 3개의 cycle로 분주시키는 3분주 회로를 사용하였다. 외부의 Enable, operate 등의 제어신호를 입력으로 받아들이며, 모든 블록들에 제어신호를 보낸다. 출력되는 제어신호는 encryption과 key generation 수행 여부를 결정짓는다. 궁극적으로 controller는 암·복호화부와 키 발생부의 우선순위를 정해주는 것이 목적이다.

2. Key Block의 모델링

IDEA에 사용되어진 52개의 16bit 서브키들은 128bit의 암호키로부터 생성된다. 생성 방식은 처음 8개의 서브키들은 최상위 16bit 키들을 그대로 사용하고, 그 다음의 16bit 키도 차례로 나머지 비트와 대응되어 받아들여진다. 입력으로 받아들여진 128bi의 키를 사용하여 8개의 서브키들이 생성되고, 그 다음에 사용되어질 키들은 128bit의 키를 25bit 순환 shift 시킴으로써 생성된다. 단일 라운드에 사용되는 96개의 서브키 비트는, 처음과 8번째 반복을 제외하고 인접해있지 않고 한 라운드의 서브키와 다른 라운드의 서브키 사이의 간단한 이동관계 조차 없다. 이런 현상이 나타나는 이유는 각 Round에서 단지 6개의 서브키들이 사용되어 지는 반면에, 서브키들은 키의 각 로테이션에 의해 8개씩 추출되기 때문이다. encryption시 입력으로 사용되어질 6개의 서브키들은 각 round에서 사용될 키들을 Cnt값에 의해 미리 계산되어진 값들을 출력하게 하였으며, key는 미리 생성되어 encryption에 사용되어야 하므로, 가장 빠른 클럭인 t1을 사용하였으며, key_enable 신호에 의해 동작의 여부를 결정짓는다.

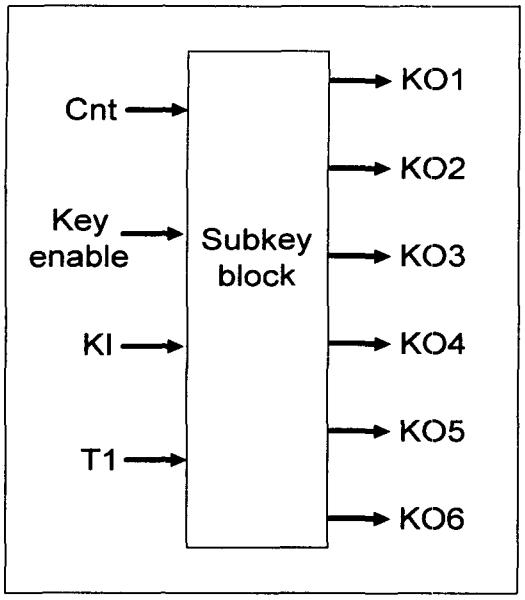


그림 3 Key Block의 블록도

```
PROCEDURE keygen(key_data, en_desig, subkey)
    BEIGN
        I : = key_data
        if en_desig = 1 then
                subkey1 := I ;
                divid (subkey1) ;
                subkey2 := I ;
                divid (sybkey2) ;


                subkey9 := I ;
                shift_left [divid (subkey9)]

                subkey52 := I ;
                divid (subkey52) ;
                else
                subkey := subkey ^ 1 ;
        end if ;
    END ;
```

3. Encryption Block의 모델링

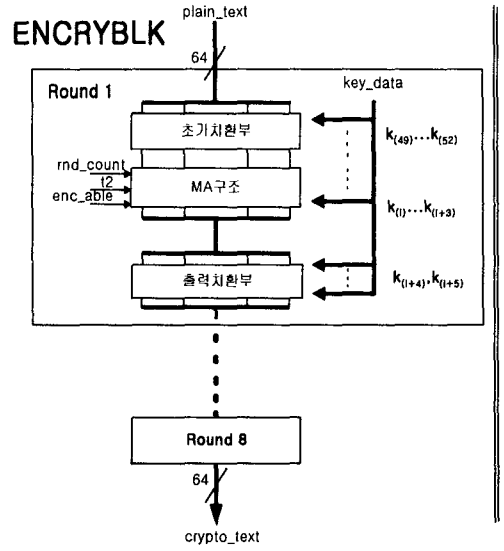
IDEA는 기본 모듈인 MA구조를 가지고 있는데, 이 MA구조의 세 가지의 연산을 사용하여 확산, 혼돈 효과를 가져온다. MA구조는 효율적인 확산을 제공하기 위하여 알고리즘상에서 8번 반복된다.

암호 처리부 - 64 비트의 평문을 입력으로 받아 다음의 3가지 연산을 수행한다.

-. 비트 대 비트의 XOR연산.

-. 각 비트의 덧셈연산.

-. 각 비트에 대한 modular연산.



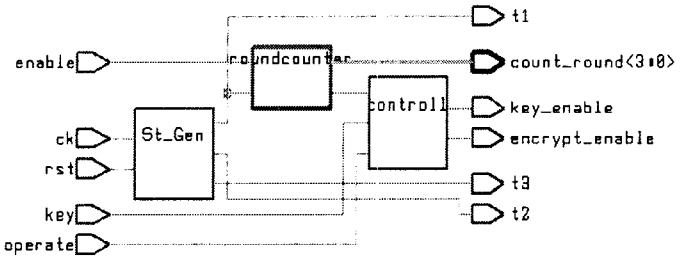

```
PROCEDEUR Encryption (Round, Plain_text)
  BEGIN
    Encryption1 := Round[ plain_text ] ;
    Encryption2 := Round[ Encryption2 ] ;
                 •
                 •
                 •
    Encryption8 := Round[ Encryption7 ] ;
  END;
```

그림 4. ENCRYBLK의 블럭도

Encryption Block은 Control Block으로부터 operate 신호를 받게 되면 동작을 수행하게 된다. cnt는 현재 수행중인 round를 계수하며, 8 round의 동작이 끝나면 permutation 과정을 거쳐 암호문을 출력한다. permutation은 암호화 과정과는 무관하나 복호 시에 동일 알고리즘을 사용하기 위하여 data의 자리바꿈을 해주는 기능을 한다. encryption 과정은 key 생성 후에 동작이 가능하므로 key보다 느린 클럭을 사용하였다.

Ⅳ. 구현고찰

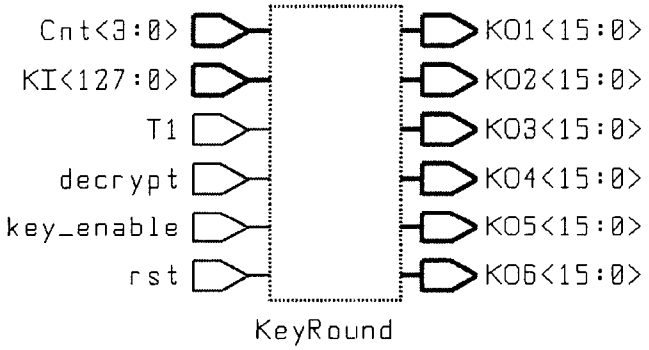
본 논문에서는 Xilinx FPGA 4052를 Target으로 삼았고, 암·복호화를 단일 Chip에 구현하기 위해 단일 라운드를 설계한 뒤 8회의 동일 알고리즘의 수행하게 하였다. 8라운드를 모두 설계하여 사용하는 것보다는 수행시간 면에서 성능이 저하되지만 칩 면적의 감소를 고려할 때, 단일 라운드를 구성한 뒤 8회 반복하여 사용하였다.



Area : 26

Timing : 21.94

그림 5. Control Block의 Schematic



Area : 1899

Timing : 238.50

그림 6. Key Block의 Schematic

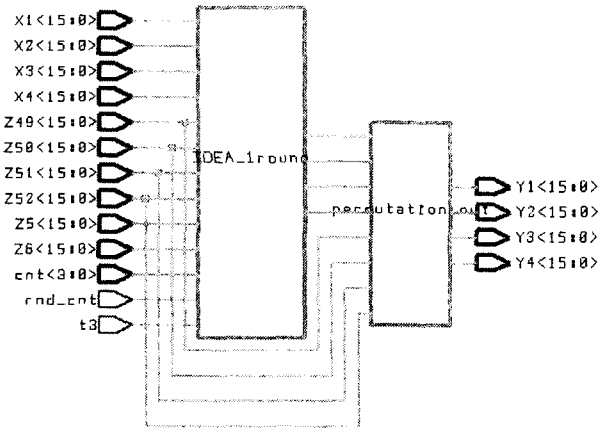


그림 7. Encryption Block의 Schematic

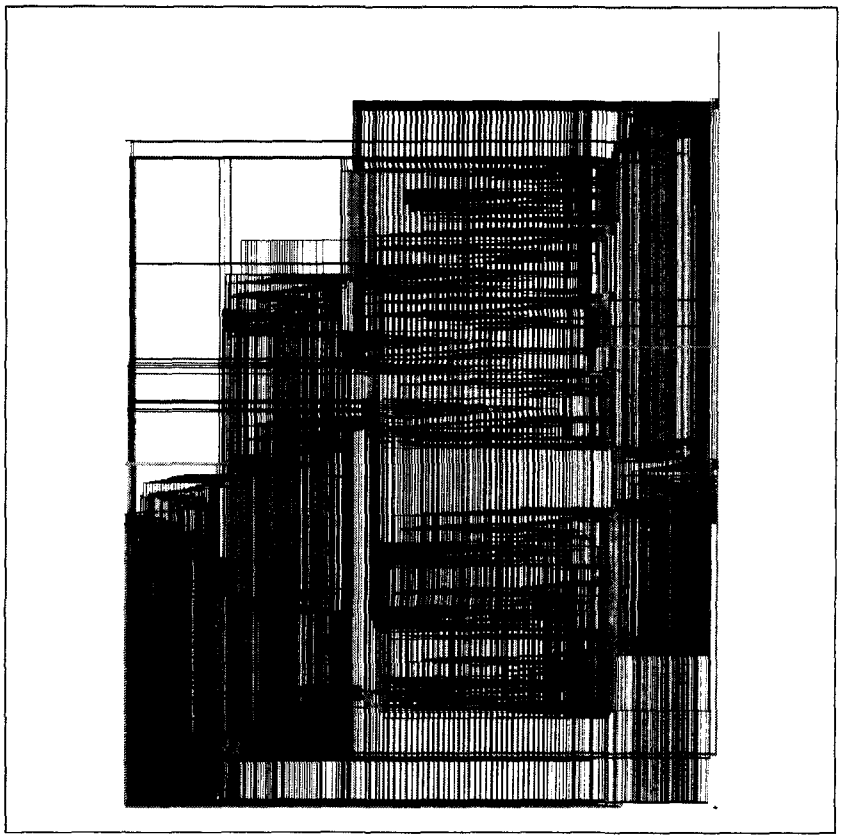


그림 8 IDEA의 전체 합성 결과

그림 7은 IDEA의 전체 블록의 합성결과이다. 합성시 tool은 Synopsys를 사용하였으며, Simulation은 V-system을 사용하여 검증하였다. 구현물은 FPGA-4052를 대상으로 하여 합성하였다. 표1에서와 같이 Encryption Block이 가장 많은 면적과 데이터의 도달시간을 차지하였다. IDEA는 암호화 과정에서 가산기, 비교기, 승산기의 수가 많이 사용되므로 이 부분을 병렬처리하여 암호화 부의 크기를 출이는 것이 필요하다. IDEA는 복호화 과정에서 입력으로 받아들여진 data를 통해 자동으로 key를 계산하여야 함으로 면적이나 delay가 많이 생기게 된다. 이 부분을 미리 계산된 data를 저장한 ROM으로 대치시키면 속도면 에서 매우 향상 될 것이라고 고려되어진다. 또한 서브키 생성과 encryption시 서로 다른 cycle의 두 클럭을 가지고 수행시켰는데, 동일한 클럭을 사용하여 약간의 delay를 두고 연산을 시키면 더욱 속도를 개선시킬 수 있으리라고 기대된다.

표 1. Chip의 area와 time 분석

	area	time
controlblock	26	21.94
keyblock	1899	11.65
encryptionblock	3831	238.50
total	5756	272.09

V. 결론

컴퓨터 기술과 통신 시스템의 발달은 대량의 정보를 신속 정확하게 처리하여 정보를 제공함으로써 정보의 효율성, 활용성, 편의성의 증대를 가져왔다. 정보보호가 중요시되고 있는 정보사회에 발맞추어 본 논문에서는 국제 표준 암호 알고리즘인 IDEA의 특성을 분석하고, VHDL을 이용하여 FPGA XC4052XL에 구현을 하였다. 고속의 데이터 암호화를 위해서 전체적인 시스템의 복잡도를 최소화하였으며, 각 라운드의 배타적인 동작을 수행하게 하였다. 설계된 IDEA는 입력 클럭으로 30Mhz를 사용

하여 64비트의 데이터를 암호화하는데 80.01Mbps의 속도로 동작하였다. 향후 과제로는 소프트웨어, 하드웨어 혼합 설계방식을 이용하여 전체적인 설계비용의 저하하는 것과 단일 클럭을 사용하여 암호기 전체를 병렬 처리하는 것이 남아있다.

<참 고 문 헌>

[1] Marc Farley, Tom Stearns, Jeffrey Hsu, *Date Integrity and Security*, McGraw-Hill, 1997.

[2] Charles P. Pfleeger, *Security in Computing*, Prentice Hall, 1989.

[3] Ralph C. Merkle, "Fast Software Functions," CRYPTO '90, 1990.

[4] Deborah Williams and Harvey J. Hindin, "Can software do encryption job?," Electronics, 1980.

[5] J. B. Kam, & S. E. Tavares, "Structured Design of Substitution Permutation Encryption Network," IEEE Transactions on Compiters, Vol.28, no.10, pp.747-753m 1989.

[6] Schneirer Bruce, Applied Cryptography, John Wiley & Sons, Inc., pp.219-296.

[7] H. H. Evertse, "Liner Structures in Block Ciphers," Advabeces in Cryptology-EUROCRYPT '87, proceedings. Berlin:Springer-Verlag, pp.249-266, 1987.

[8] G. I. Davida, D. J. Linton, C. P. Szelag &D. L. Well, "Data base security," IEEE Transactions on Software Engineering, Vol.SE-4, no.6, pp.531-533, 1978.

[9] William Stallings, "Network and Internetwork Security Principles and practice," IEEE PRESS, 1995.

[10] Z. Navadi, "Using VHDL for model and design of processing unit," Proceeding of the 1992 ASIC Conf. and Ex., pp.315-326. 1992.

[11] James R. Armstrong & F. Gail, *Structured Logic Design with VHDL*, Prentice-Hall.