通信情報保護學會論文誌
第 8 卷 第 4 號. 1998. 12

IDEA의 고속 암호칩 설계

이 상 덕*, 한 승 조**

Design of the High-Speed Encryption Chip of IDEA(International Data Encryption Algorithm)

Sang-Duck Lee*, Seung-Jo Han**

요 약

통신 및 컴퓨터 시스템의 처리 속도가 높아짐에 따라 정보 보호를 위해서 고속의 데이터처리가 반드시 요구되어진다. 따라서 본 논문에서는 국제 표준 암호알고리즘의 하나인IDEA(International Data Encryption Algorithm)를 고속 암호칩으로 설계한다. 먼저 고속 연산을 위하여 알고리즘을 분석하고 암호화 수행시간을 감소하기 위하여 파이프라인 처리를 하며, 서브키 생성시의 연산회수를 줄이기 위하여 서브키 블록을 EEPROM으로 구현하였다. 전체적인 시스템은 VHDL(VHSIC Hardware Description Language)을 사용하여 설계하였다. IDEA 알고리즘은 EDA tool인 Synopsys를 사용하여 Synthesis하였으며, Xilinx의 FPGA XC4052XL을 이용하여 One Chip화 시켰다. 입력 클럭으로 20Mhz를 사용하였을 때, data arrival time은 587.07ns였으며, 109.01 Mbps의 속도로 동작하였다.

Abstract

As the processing speed of communication and computer system has been improved, high speed data processing is required to protect information. This study aims at implementing a high speed encryption chip of IDEA(International Data Encryption Algorithm) which is one of international standard encryption algorithms. We analyze IDEA algorithm for the high speed operation at hardware and pipeline processing is performed to reduce the encryption time. To reduce the number of operations in generating subkey, subkey block is implemented with EEPROM. This system is designed by using VHDL(VHSIC Hardware Description Language). The IDEA algorithm is synthesized by using synopsys(EDA tool) and algorithm is made to one chip by means of FPGA XC4052XL of Xilinx. When 20Mhz is used as the input clock, data arrival time is 587.07ns and it is operated with the speed of 109.01Mbps.

이 논문은 1998년도 정보통신부 대학기초연구사업 지원의 일부에 의해서 연구되었음
* 조선대학교 전자공학과
** 조선대학교 전자 · 정보통신공학부(수송기계 부품공장 자동화 연구센터연구원)

1. 서 론

현대 정보화 사회에서 통신시스템은 급속적인 발전을 거듭하고 있으며, 과거 인위적인 의사소통에서 시작하여 최근에는 통신망을 이용한 화상통신까지 실용화되고 있다. 통신시스템의 발전과 더불어 개방된 통신망상에서 정보 보호가 중요시되기 시작하였으며, 암호의 활용은 정보화 사회의 인프라 구축을 위한 하나의 중요한 수단이 되었다.[1][2]

본 논문에서 H/W인 암호칩으로 구현하기 위한 IDEA((International Data Encryption Algorithm)는 52개의 서로 다른 서브키를 생성하여 암호화를 수행하며, 8라운드의 암호화 과정을 거쳐 출력을 생성하게 된다.[3][4] 키 사이즈가 56비트인 DES에 비해 128비트의 키를 사용하기 때문에 암호학적 강도가 DES에 비해 강할 뿐만 아니라 비트 대 비트의 XOR, mod(2^{16})의 덧셈, mod(2^{16}+1)의 곱셈 등의 3가의 연산을 조합하여 사용하므로 해독을 더욱 어렵게 만들며 MA 구조를 사용하여 Diffusion 및 Confusion 효과를 제공한다.[5] 그러나 mod 연산의 사용은 계산량을 증가시키므로 처리속도의 지연이 문제시된다.[6] 이에 대한 방안으로서 면적과 지연시간이 큰 모듈러 연산자의 수를 감소시키고, IDEA의 8라운드를 30클럭 동안 1024비트의 데이터를 처리하도록 파이프라인 방법을 제안하고 설계하고자 한다. 이 알고리즘은 VLSI 구현을 위해 VHDL[7]로 설계한 후 논리합성을 하였다. VHDL은 회로 설계의 효율화 및 설계 시간을 단축시킬 수 있으며, EDA(Electronic Design Automation) 소프트웨어나 ASIC Library에 상관없이 객관성 있게 설계를 할 수 있다. 또한 이것은 반도체 내부 회로에 대한 검증부터 시스템 설계까지 할 수 있으며, Synthesis 소프트웨어를 이용하여 실제 반도체 회로를 쉽게 얻을 수 있다는 장점을 가지고 있으므로 본 논문에서 설계 언어로서 채택하였다.

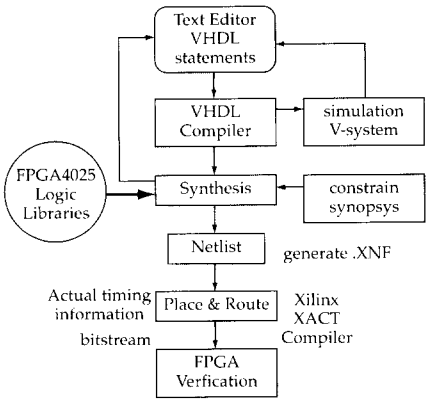

그림 1. IDEA의 설계 흐름도

설계 초기 단계에서 그림 1과 같이 VHDL 모델을 작성하여 함수 시뮬레이션[8]을 거쳐 알고리즘의 타당성을 검증하고, 이들 모델을 RTL(Register Transfer Level)로 구체화한 후, 논리합성을 수행함으로써 알고리즘을 회로로 구성하는 과정에서의 오류를 방지하였다. 합성이 끝난 회로는 node에서 node까지 도달하는데 걸리는 시간, critical path 등을 추출하여 Timing Simulation을 하였으며, 시뮬레이션을 거쳐 얻은 데이타는 hold time violation 등 시간에 관련된 논리 오류를 검사하는데 사용하였다. 타이미 시뮬레이션을 수행한 후 Xilinx FPGA(Field Programmable Gate Arrays) XC4052XL에 구현하였다.[9]

제 2장에서는 일반적인 IDEA 알고리즘의 구조를 살펴보았으며, 제 3장에서는 알고리즘의 모델링 방법 및 파이프라인 처리 등에 대하여 제안하였으며, 제 4장에서는 VHDL을 이용한 시뮬레이션과 합성과정 및 FPGA 실현에 대하여 기술하였으며, 제 5장에서는 암호칩의 성능을 분석하였으며, 제 6장에서 결론을 맺고자 한다.

2. IDEA의 이론적 배경

본 논문에서 활용하고자 하는 IDEA (International Data Encryption Algorithm)는 Xuejia Lai와 Hames

Massey에 의해서 개발된 블록 지향 암호 알고리즘이다.[4] IDEA는 64비트의 평문 블록을 암호화하기 위하여 128비트의 키를 사용하는 블록암호 방식이다.[5][6] 암호화의 전반적인 과정은 그림 2와 같이 암호화 함수에 암호화하고자 하는 평문과 키를 입력으로 사용하여 암호문 출력을 얻어낸다. IDEA 알고리즘은 마지막 변환까지 8개의 라운드로 구성되어 있으며 각 라운드의 출력은 다음 라운드 입력으로 사용된다. 8라운드의 동작이 끝나면 치환 과정을 거쳐 암호문 출력을 생성한다.

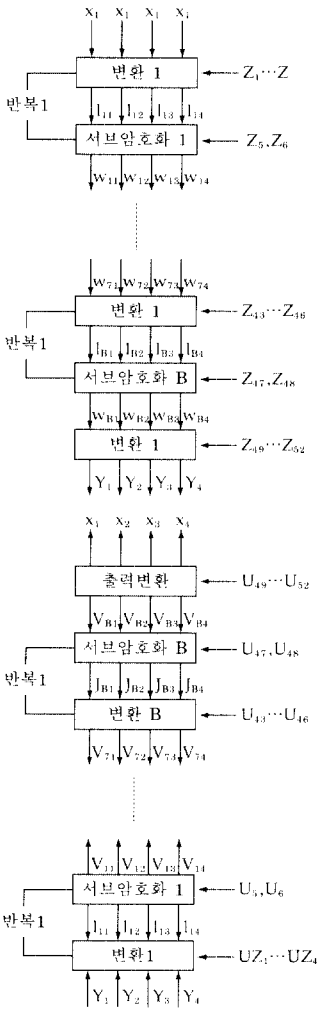

그림 2. IDEA의 암 · 복호화 과정

IDEA의 8라운드의 암호화와 출력변환까지는 52개의 16비트 서브키들이 필요하며, 서브키는 128비트의 key_data를 입력으로 사용하는 서브키 생성기로부터 생성된다. 서브키 생성방식은 128비트 입력을 16비트씩 분할하여 8개의 서브키를 만들어내고, 이 과정이 끝나면 25비트씩 왼쪽으로 쉬프트하여 나머지 서브키를 만들게 된다. 이러한 과정은 52개의 서브키들이 만들어질 때까지 반복된다. 25비트를 왼쪽으로 쉬프트시켜서 키를 추출하게되면 키가 서로 인접하지 않을 뿐만 아니라 키 비트를 더욱 다양화시킬 수 있다. 키 추출시 이런 현상이 나타나는 이유는 각 반복에서 단지 6개의 서브키들이 사용되어지는 반면 8개의 서브키들이 로테이션에 의해 추출되기 때문이다.[10][11]

일반적인 비밀 키 시스템과 마찬가지로 IDEA도 암호화와 복호화에 동일한 알고리즘을 사용하나 키 사용에서는 서로 다른 방식을 채택한다. 복호 반복 i라운드에서의 처음 네 개의 서브키는 암호 반복 $(10-i)$라운드에서의 처음 4개의 서브키로부터 유도되며, 처음과 네 번째에 사용된 복호 서브키는 첫 번째와 네 번째 암호 서브키에 대응하는 mod $(2^{16}+1)$에서의 곱셈의 역원을 취하여 사용하며, 두 번째와 세 번째 복호 서브키들은 세 번째와 두 번째 암호 서브키에 mod (2^{16})에서의 덧셈의 역원을 취하여 사용한다. 반복1과 9에 대해서 두 번째 세 번째 복호 서브키는 두 번째와 세 번째 암호 서브키의 법(2^{16})에서의 덧셈과 같다. 처음 8개의 반복에 대해서 복호 반복 i의 마지막 두 개의 서브키는 암호 반복$(9-i)$의 두 마지막 서브키와 같다. 이러한 과정을 거쳐 추출된 키들은 표 1과 같이 나타낼 수 있다.[4][11]

표 1. 암 · 복호화 키 스케줄 방식

	암호화	복호화
상태	선정	선정
반복 1	$Z_1Z_2Z_3Z_4Z_5Z_6$	$Z_{49}^{-1} - Z_{50} - Z_{51}Z_{52}^{-1}Z_{47}Z_{48}$
반복 2	$Z_7Z_8Z_9Z_{10}Z_{11}Z_{12}$	$Z_{43}^{-1} - Z_{45} - Z_{44}Z_{46}^{-1}Z_{41}Z_{42}$
반복 3	$Z_{13}Z_{14}Z_{15}Z_{16}Z_{17}Z_{18}$	$Z_{37}^{-1} - Z_{39} - Z_{38}Z_{40}^{-1}Z_{35}Z_{36}$
반복 4	$Z_{19}Z_{20}Z_{21}Z_{22}Z_{23}Z_{24}$	$Z_{31}^{-1} - Z_{33} - Z_{32}Z_{34}^{-1}Z_{29}Z_{30}$

반복 5	$Z_{25}Z_{26}Z_{27}Z_{28}Z_{29}Z_{30}$	$Z_{25}^{-1} - Z_{27} - Z_{26}Z_{28}^{-1}Z_{29}Z_{24}$
반복 6	$Z_{31}Z_{32}Z_{33}Z_{34}Z_{35}Z_{36}$	$Z_{19}^{-1} - Z_{21} - Z_{20}Z_{22}^{-1}Z_{17}Z_{18}$
반복 7	$Z_{37}Z_{38}Z_{39}Z_{40}Z_{41}Z_{42}$	$Z_{13}^{-1} - Z_{15} - Z_{14}Z_{16}^{-1}Z_{11}Z_{12}$
반복 8	$Z_{43}Z_{44}Z_{45}Z_{46}Z_{47}Z_{48}$	$Z_{7}^{-1} - Z_{9} - Z_{8}Z_{10}^{-1}Z_{5}Z_{6}$
변환	$Z_{49}Z_{50}Z_{51}Z_{52}$	$Z_{1}^{-1} - Z_{2} - Z_{3}Z_{4}^{-1}$

3. 고속 암호칩의 모델링

본 논문에서는 Top-Down 방식을 이용하여 IDEA 알고리즘을 Xilinx FPGA XC4052XL 상에 구현하며, 그림 3은 IDEA의 Top-level block의 schematic이다.

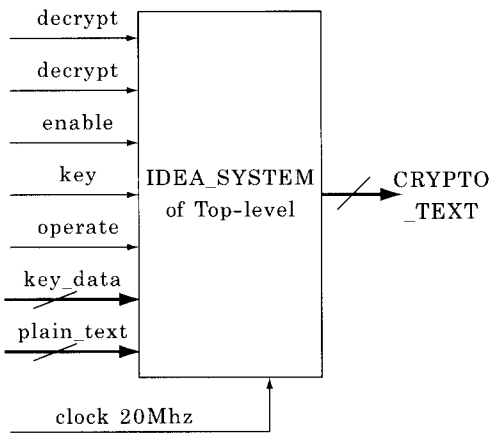

그림 3. IDEA의 Top-level Schematic

IDEA_SYSTEM은 64비트의 Plain_text와 128비트의 Key_data를 입력으로 받아 암호화 과정을 거쳐 64비트의 CRYPTO_TEXT를 출력하게 된다. decrypt signal에 의하여 암호화 또는 복호화 동작을 선택하게 되며, 외부 클럭으로 20Mhz를 사용하게 되고 key 신호와 operate 신호에 의해서 키 생성과정과 암호화 과정을 선택하게 된다.

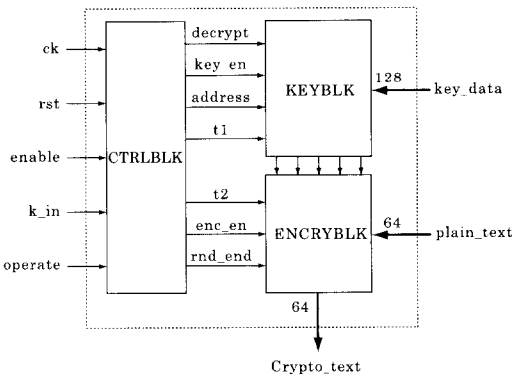

그림 4. Top-level의 Sub-block

IDEA_SYSTEM의 Top-block은 그림 4와 같이 CTRLBLK, KEYBLK, ENCRYBLK의 세 부분으로 나눌 수 있다. CTRLBLK는 KEYBLK와 ENCRYBLK의 전체적인 동작을 제어하는 역할을 한다. 외부로부터의 rst, enable signal은 전체적인 동작의 유무를 제어하며 k_in, operate signal은 KEYBLK와 ENCRYBLK block의 동작유무와 우선 순위를 결정하게 된다. KEYBLK는 t1의 클럭에 동기화 되어 동작을 하게 되며, 외부로부터 key_data를 입력으로 받아들여 암호화시 필요한 52개의 서브키를 생성한다. 이렇게 생성된 키들은 address signal에 의해 키들을 암호화 라운드에 제공하게 되며, ENCRYBLK의 입력으로 사용되어 연산을 거쳐 출력을 생성해낸다.

3.1 CRLBLK의 모델링

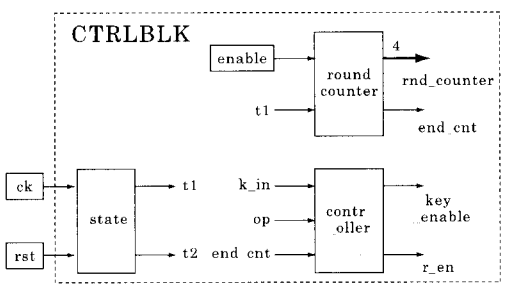

그림 5. CTRLBLK의 블럭도

CRLBLK는 그림 5와 같이 제어기, 라운드카운터, 클럭발생기로 구성되어 있다. 클럭발생기는 외부의 클럭을 입력으로 받아들여 t1, t2 두 개의 클럭을 발생하는 분주회로로 구성되어 있다. 분주된 두 개의 클럭 t1, t2는 KEYBLK, ENCRYBLK의 입력 클럭으로써 사용될 뿐 아니라, round_counter의 입력으로 사용되어 서브키 블록의 키를 발생시키는 부분과 동기를 맞추어 암호화시 필요한 서브키를 공급하게 한다. round_counter에는 종료신호인 end_cnt를 두어 전체 라운드의 동작이 종료되면 더 이상 동작을 하지 않도록 한다. controller는 KEYBLK, ENCRYBLK의 동작 우선 순위를 정하여 서로 유기적인 동작을 하도록 하는 역할을 한다. KEYBLK는 ENCRYBLK보다 빠르게 동작해야 하므로 가장 주기가 빠른 t1을 입력으로 사용하며, t1과 key_enable 신호가 동시에 활성화되면 동작하게 된다. ENCRYBLK는 t2와 r_en 신호가 활성화되면 동작하게 된다.

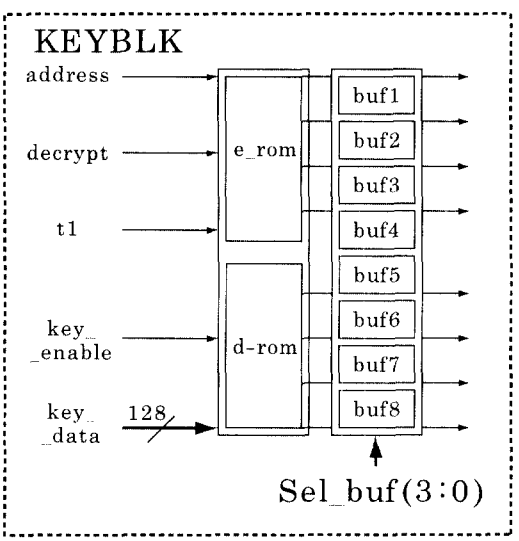

그림 6. KEYBLK의 블럭도

3.2 KEYBLK의 모델링

KEYBLK는 그림 6과 같이 128비트의 key_data를 입력으로 받아들여서 52개의 16비트 서브키를 생성한다. 서브키 생성방식은 16비트씩 키를 분할하여 8개의 서브키를 선택하고, shift_left하여 나머지 서브키를 생성하게 된다. 서브키 생성과정은 52개의 서브키들이 생성될 때까지 계속된다. 이러한 방식으로 encryption key를 생성하게 되고, decryption key는 각 서브키들의 역원을 구하여 사용하게 된다. 전체적인 동작의 유무는 key_enable 신호에 의해서 결정하게 된다. decrypt 신호가 활성화 되었을 때는 encryption_key를 생성하고, decrypt 신호가 비활성화 되었을 때는 decryption key를 생성하게 된다. 생성된 키는 e_rom과 d_rom에 저장하게 되고, address 신호에 의해 키를 호출하여 buffer단에 저장하게된다. 저장된 서브키는 4비트 sel_buf signal에 의해 8개의 buffer중 각 라운드에 필요한 buffer단을 선택하여 buffer 출력을 암호화 라운드에 서브키로 제공하게 된다. 이러한 서브키 생

```
*************************************************************
/* Keygen - 서브키를 생성하는 블록                          */
/*          key_data : 초기에 입력되는 키                   */
/*          ed_desig : 암호화, 복호화 선택 신호             */
/*          subkey1, subkey2, … , subkey52 : 1~8
            라운드의 52개의 서브키                          */
*************************************************************
PROCEDURE keygen(key_data, en_desig, subkey1,…
, subkey52)
    BEIGN
        I : = key_data
        if en_desig = 1 then
                subkey1 := I ;
                divid (subkey1) ;
                subkey2 := I ;
                divid (sybkey2) ;
                                ○
                                ○
                                ○
                subkey9 := I ;
```

```
            shift_left [divid (subkey9)]
                              ○
                              ○
                              ○
            subkey52 := 1 ;
            divid (subkey52) ;
            else
            subkey := subkey-1 ;
    end if ;
END ;
```

그림 7. IDEA의 키 생성 알고리즘

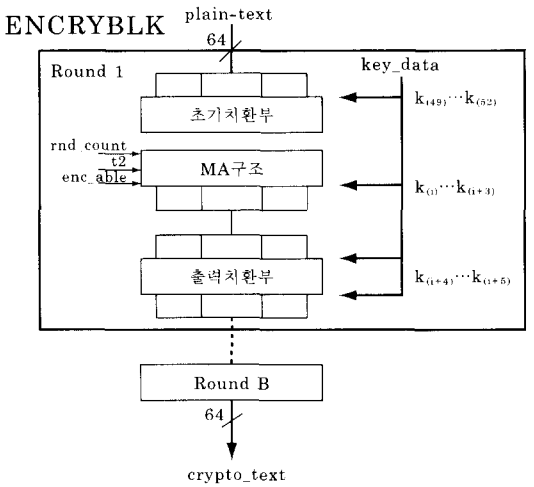

그림 8. ENCRYBLK의 블럭도

성과정을 알고리즘으로 표현하면 그림 7과 같다.

3.3 ENCRYBLK 모델링

ENCRYBLK는 Round block들의 정규적인 반복으로 이루어져 있으며, Round block의 내부는 치환과 MA구조의 반복으로 이루어져 있다. Round block의 내부구조는 그림 8과 같다. 각 Round block은 필요한 서브키들을 입력으로 받아 plain_text와 연산을 통하여 출력을 생성하고, 이 출력들은 다음 Round의 입력으로 다시 사용된다. Round-block은 enc_able 신호에 의해서 동작을 하게 되며, 내부는 XOR, mod(216)의 덧셈, mod(216+1)의 곱셈 등 세 가지 연산의 조합으로 이루어져 있다. XOR, mod(216)의 덧셈은 구현이 비교적 용이하나, mod(216+1)의 곱셈은 구현이 복잡하여 다음 식과 같이 기약 연산들로 구조화하여 구현한다.

$$
\begin{aligned}
&ab \bmod (2n+1) = \\
&\quad (ab \bmod 2n) - (ab \operatorname{div} 2n) \\
&\quad [if\ (ab \bmod 2n) \geq (ab \operatorname{div} 2n)] \\
&\quad (ab \bmod 2n) - (ab \operatorname{div} 2n) + 2n + 1 \\
&\quad [if\ (ab \bmod 2n) \leq (ab \operatorname{div} 2n)]
\end{aligned}
$$

동일한 Round를 8회를 거치게 되면 최종적인 암호화 출력을 얻게 되며, 마지막으로 이 출력은 자리바꿈을 하게 되는데, 이는 암호화와 복호화 시에 동일한 알고리즘을 사용하기 위해서이며, 암호화 과정에 대한 알고리즘은 그림 9과 같이 표현할 수 있다.

3.4 IDEA 알고리즘의 파이프라인 처리

IDEA의 ENCRYBLK는 초기치환부, MA구조, 출력치환부로 구성된 라운드 블럭의 반복으로 설계된다. 이러한 라운드 구조는 일반화하여 모두 같은 방식으로 처리가 가능하도록 구성할 수 있으며, 그림 10과 같이 8개의 클럭에 의해 암호화 과정을 수행하게 된다. 이와 같이 암호화를 수행하게 될 때 첫 번째 라운드가 클럭1에 의해 동작을 수행하게 되면 그 외의 라운드는 동작을 정지하며 입력을 대기하는 상태에 있게 된다. 클럭2가 입력으로 들어오면 첫 번째 블록은 라운드2의 동작을 수행하게 되며, 이때에 휴지상태에 있는 라운드1에 새로운 평문을 입력으로 사용하게 한다. 따라서 클럭2에서는 라운드2와 라운드1이 동시에 암호화 과정을 수행하고 있는 상태가 된다. 마찬가지로 클럭8에서는 8개의 라운드가 동작을 수행하게 된다. 이러한 과정을 통해 1회 암호화시키는데 8개의 클럭이 소요되고, 동시에 처리되는 반복순환8번째의 암호화 과정이 종료될 때까

지는 추가로 7개의 클럭이 더 소요되게 된다. 이러한 방식에 의해 암호기를 동작시키게 되면 15클럭 동안 512비트의 Plain_text를 처리하게 된다. 단일 암호화처리 과정에 비해 이러한 처리 방식은 1클럭의 소요시간을 더 가지면 64비트의 plain_text를 추가로 암호화시킬 수 있다.

```
*****************************************************************
/* Round(1~8) - 암호화 또는 복호화를 수행하는 블록 */
/*      Plain_text : 암호화 또는 복호화를 위한 데이터 */
/*      subkey1.. subkey52 : 1~8라운드의 서브키 */
/*      Crypto : 암호화 또는 복호화 된 데이터 */
*****************************************************************
PROCEDURE Round (Plain_text(1-4), MA(L), MA(R),
ini_permu, Subkey1,…, Subkey52, Crypto)
    BEGIN
        ini_permu(1) := Plain_text1 mod(216+1) subkey1
        ;
        ini_permu(2) := Plain_text2 add subkey2 ;
        ini_permu(3) := Plain_text3 add subkey1 ;
        ini_permu(4) := Plain_text4 mod(216+1) subkey2
        ;
        MA(L) := temp1 add temp2 ;
        MA(R) := temp2 ;
        temp1 := permu1 mod subkey5 ;
        temp2 := permu2 add temp1 mod subkey6 ;
        Crypto := MA(R) XOR ini_permu(1) & MA(R)
        XOR ini_permu(3) & MA(L) XOR ini_permu(2)
                & MA(L) XOR ini_permu(4) ;
    END ;

*****************************************************************
/*Encryption(1~8) : 암호화 또는 복호화 수행 블록의 결과*/
/*          Round : 암호화 또는 복호화를 수행하는 블록*/
*****************************************************************
PROCEDEUR Encryption (Round, Plain_text)
    BEGIN
        Encryption1 := Round[ plain_text ] ;
        Encryption2 := Round[ Encryption2 ] ;
                                ○
                                ○
                                ○
        Encryption8 := Round[ Encryption7 ] ;
END;
```

그림 9 IDEA의 암호화 알고리즘

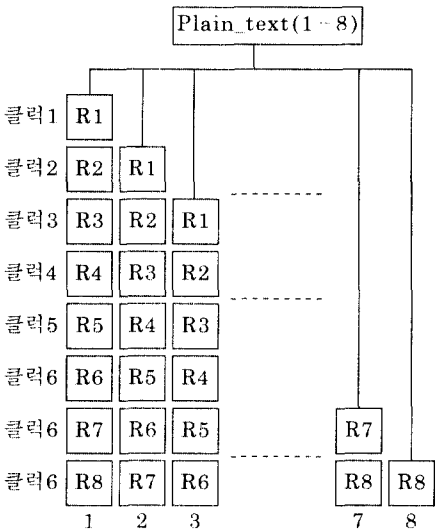

그림 10. 전체 라운드의 Pipeline처리

4. IDEA 시스템의 시뮬레이션 및 합성

암호칩을 구성하고 있는 블록의 모델링이 완료되면 각각의 블록들을 합하여 하나의 Top-Level 시스템을 구성하게 된다. Top-Level 시스템 상에서 논리검증을 위한 시뮬레이션을 수행한 결과 그림 11, 그림 12와 같이 입력된 데이터와 암·복호화되어 출력된 데이터가 동일함을 확인하여 암·복호화가 정상적으로 수행됨을 검증하였다. 데이터의 입력으로 사용되는 plain_text와 key_data는 각각 64비트와 128비트의 값을 무작위로 축출하여 사용하였으며, rst가 '1'이고 enable 신호가 '1'인 상태에서 활성화되었다. key 신호가 '1'일 때는 키를 생성하게 되고, operate 신호가 '1'일 때는 암호화 동작을 수행하게 된다. 아래의 그림에서 key 신호와 operate 신호가 모두 '1'인 상태이지만 모델링 과정에서 key 신호에 동작의 우선순위를 두었으며 서로 다른 클럭을 사용하였으므로 동작과정에서 오류를 일으키지 않았다. 외부에서 주어진 ck 신호는 서로 다른 세

개의 주기를 갖는 t1_dummy, t2_dummy, t3_dummy로 분주하였으며, 모든 블록들은 분주된 신호들의 상승 구간에서 동작하였다. count_dummy 신호는 현재의 동작중인 라운드를 카운트하는 역할을 하며 "0000"~"1001"까지 9라운드를 카운트하였다. IDEA는 8라운드 동작을 수행하게 되면 암호화된 출력을 얻게되지만 암·복호화에 동일한 알고리즘을 사용하기 위하여 출력변환을 하는 단계가 있으므로 "1001"을 한 번 더 카운트하도록 하였다. z1_dummy~z6_dummy 신호는 count_dummy에 맞추어 각각의 라운드에 해당되는 서브키를 출력하는 역할을 하며 암호화 과정에 필요한 52개의 서브키를 출력하는 것을 시뮬레이션을 통하여 확인하였다. 이러한 과정을 거쳐 9라운드의 동작을 수행하게 되면 암호화된 출력을 얻게 되고, 암호화된 데이터를 다시 입력으로 사용하게 되면 원래의 데이터를 얻게 된다.

함수 시뮬레이션 후 실제 회로를 얻기 위하여 Synopsys Design_Analyzer[12]를 사용하여 합성을 하였다. 합성은 FPGA 라이브러리에 포함된 셀 라이브러리에 맞추어 합성을 하였으며, 합성된 회로는 게이트 레벨의 회로로 변환하였다. report 파일을 참조로 하여 지연 및 타이밍 제약이나 게이트 수 제약의 범위 및 테스트 용이화를 고려하여 설계의 적합성을 확인하였다.[13][14] 합성이 끝난 회로로부터 XNF 파일을 생성해 낸 뒤 FPGA에 구현을 하게 되며, 합성이 끝난 각 블록과 Top-level의 합성결과는 그림 13~그림 16과 같다.

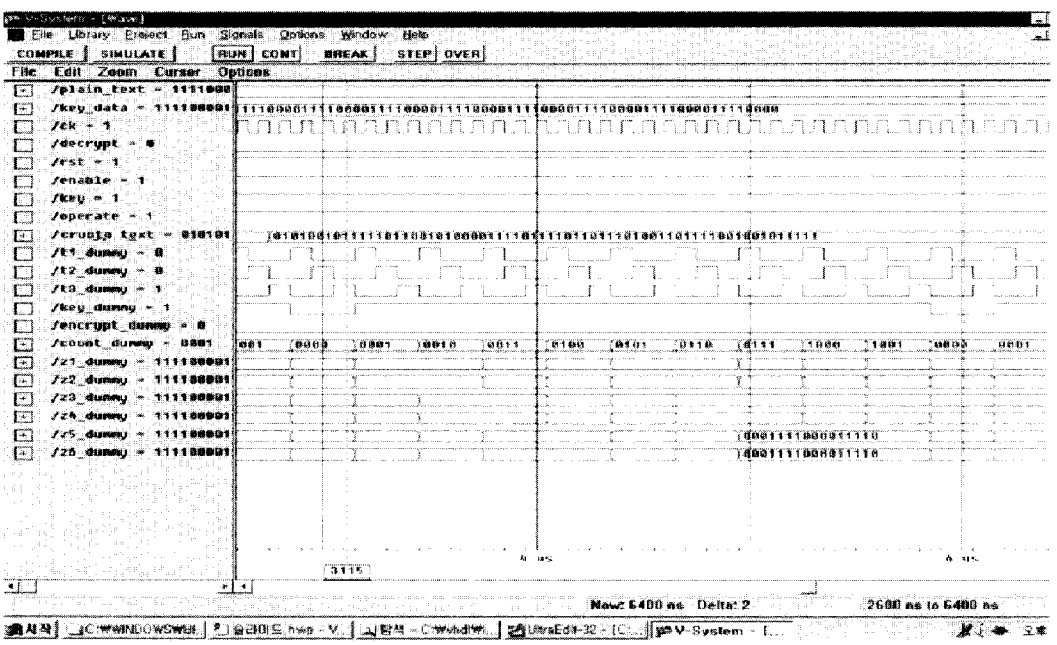

그림 11. 암호화 시뮬레이션 결과

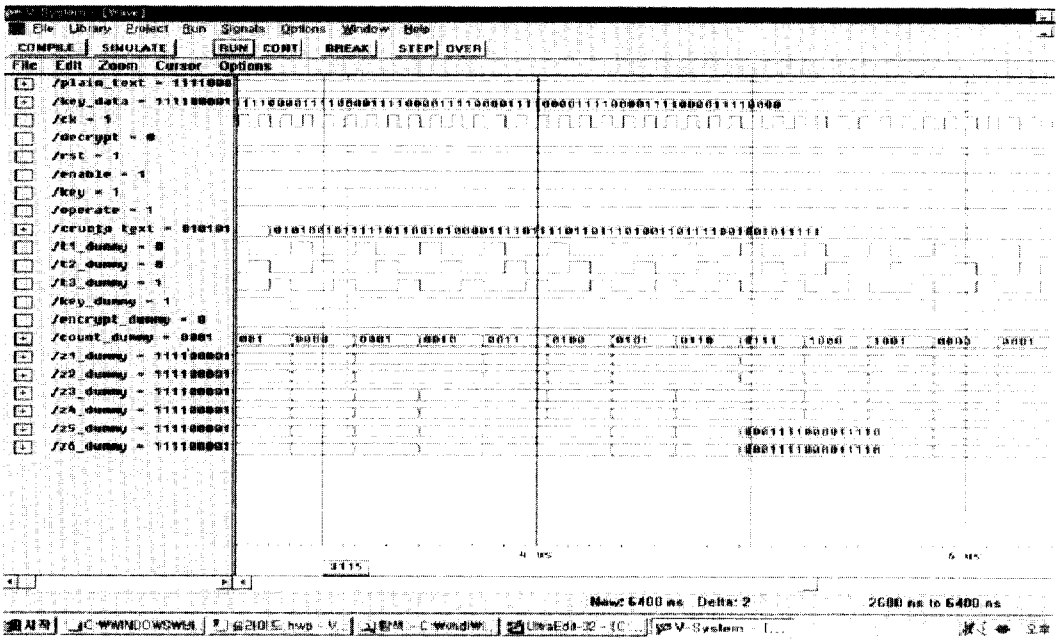

그림 12. 복호화 시뮬레이션 결과

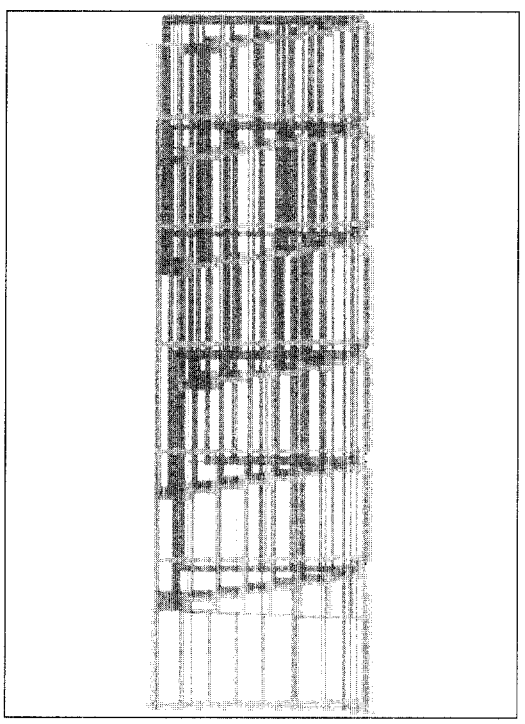

그림 13. CTRLBLK의 합성결과

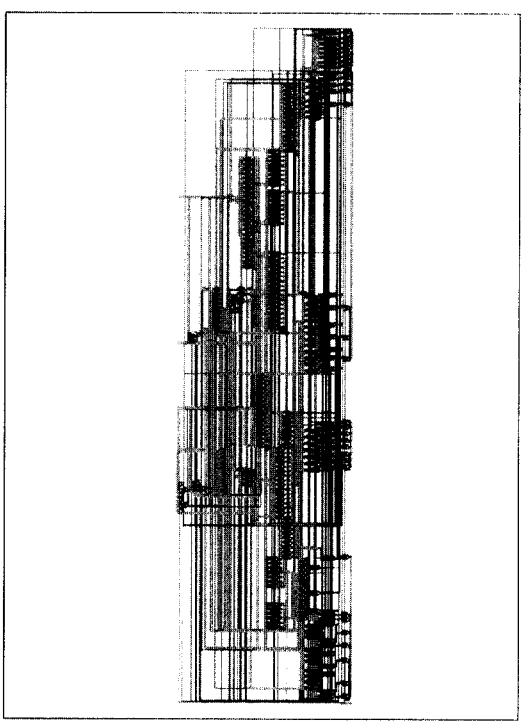

그림 15. ENCRYBLK의 합성결과

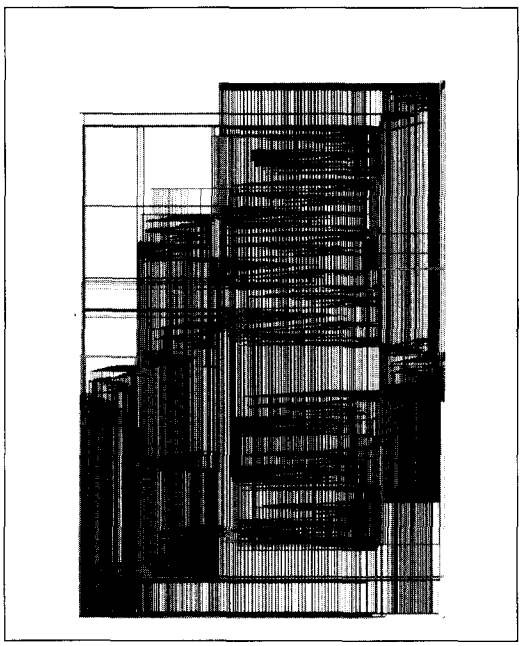

그림 16. IDEA-SYSTEM의 전체합성결과

5. 분석 및 고찰

FPGA를 이용한 구현은 미리 설계된 칩에 프로그램밍을 함으로써 원하는 기능의 회로를 구현하는데 목적이 있으며, IC 설계 제작에 비하여 소요되는 비용과 시간을 절감할 수 있을 뿐만 아니라, 원하는 회로에 대한 신속한 실험 제작(prototyping)이 가능하다는 장점이 있다. FPGA 칩은 조합 논리 회로와 메모리를 구현할 수 있는 CLB(Configurable Logic Block), 입출력을 위한 IOB(Input/Output Block), 그리고 CLB들과 IOB들을 서로 연결하여 회로를 구현하기 위한 배선 구조로 이루어져 있다.[14]

마지막 구현 단계로서 FPGA XC4052XL에 Place&Route를 하기 위하여 Xilinx의 XDM(XACT Design Manager)을 사용하였다. Place&Route후 back-annotated VHDL netlist와 SDF를 가지고 Xilinx vital 라이브러리를 이용하여 Setup time, Hold time, Minimum Pulse Width, Recovery Time 등 Timing Violation을 정확성을 검사하였다. 타이밍 시뮬레이션을 수행한 후 BIT 파일을 생성하고, Hardware debugger를 이용하여 FPGA XC4052XL에 구현하였다. FPGA XC4052는 표 2와 같이 1936개의 CLB를 사용할 수 있으며, 구현에 있어서는 실제로 86.72%에 해당되는 1697개의 CLB를 사용하였으며, 전체 입·출력 버퍼의 42.32%에 해당되는 149개의 입·출력 버퍼를 사용하였다. 입·출력 신호의 속도의 빠른 처리를 위해 사용되어지는 자체 CLB Flip Flops는 3872개중에 890개를 사용하였다. 이러한 CLB들은 표 3.과 같이 Leaf-gate로 나타낼 수 있다.

표 2. FPGA XC4052XLPG411-3의 사용 개수

	사용 개수	최대사용 가능개수	사용비율(%)
CLBs	1679	1936	86.72
Bonded IOBs	149	352	42.32
CLB Flip Flops	890	3872	22.98
4 Input LUTs	672	3872	17.35
3 Input LUTs	251	1936	12.96

표 3. FPGA XC4052XLPG411-3의 기본 Gate 사용수

Cell	Use	Cell	Use
and2	161	or3	2
and4	5	or4	85
fdc	50	xor2	145
fdce	32	ldc1	1
fdp	66	inv	572
nand2	2487	ld_1	192
nor2	4	add_sub	35
nor3	1	compare	16
nor4	80	inc_dec	1
or2	120	dw02_mult	14

6. 결 론

컴퓨터 통신망 상에서 정보보호를 위한 인증

성과 기밀성을 실현하기 위해 암호알고리즘을 S/W로 구현할 경우에는 방대한 계산량 때문에 실시간 처리가 불가능하다. 따라서 이를 위해서는 암호알고리즘은 반드시 H/W로 구현되어져야 하며, 특히 범용성 IC를 사용하여 구현하는 것보다는 전용 암호프로세서 칩을 사용하여 구현하는 것이 경제적이며 효율적이다. 따라서 본 논문에서는 국제 표준 암호 알고리즘인 IDEA를 암호칩으로 구현하기 위해서 이를 분석하고, 국내 최초로 VHDL을 이용하여 FPGA XC4052XL 칩에 구현하였다. 고속의 데이터 암호화를 위해서 전체적인 시스템의 복잡도를 최소화하였으며, 각 라운드를 파이프라인 처리하였다. 연산 시간이 많이 소요되는 KEYBLK 부분은 EEPROM으로 구현하여 지연을 최소화하였으며, 설계된 IDEA는 입력 클럭으로 20Mhz를 사용하여 64비트의 데이터를 암호화하는데 109.01Mbps의 속도로 동작하였다.

향후 과제로는 암호칩을 ASIC으로 설계하고 최적화하는 것과 단일 클럭을 사용하여 암호화부와 키를 생성하는 부분을 동시에 동작시켜 처리속도를 향상시키는 것이 필요하다. 또한 암호기 전체를 병렬 처리하여 데이터의 처리속도를 높이는 것이 남아있다.

참 고 문 헌

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

[2] Bruce Schneier, APPLIED CRYPTOGRAPHY, Hohn Siley & Sons, Inc. 1996.

[3] William Stallings, "Network and Internetwork Security Principles and Practice." IEEE PRESS, 1995.

[4] W. Meier, "On the security of the IDEA block cipher." In T. Helleseth, editor Advances in Cryptology-Eurocrypt' 93 LNCS 765, pp.371-385, Springer vertag, 1993.

[5] Kelsey, B. Schneier, and D. wagner, "Key-Schedule Crypt analysis of 3-WAY, IDEA, G-DES, RC4, SAFER, and Triple-DES." Advances in Cryptology--CRYPTO' 96, Springer-verlog, pp.273-251, August 1996.

[6] J. Borst, "Differential-Linear Cryptanalysis of IDEA," Technical Report ESAT-COSIC Report 96-2, Department of Electrical Engineering, Katholieke Universiteit Leuven, Feb. 1997.

[7] Joseph S. Lis and Daniel D. Gajski, "VHDL Synthesis Using Structured Modeling," 26th ACM/IEEE Design Automation, pp.746-749, June 1989.

[8] Steven S. Leung, "Behavioral Modeling of Transmission Gates in VHDL," 26th ACM/IEEE Design Automation, pp.746-749, June 1989.

[9] H. Shin and C. Kim, "Performance-Oriented Technology Mapping for LUT-Based FPGAs," IEEE Transactions on VLSI system, Vol.3, No.2 pp.323-327, June. 1995.

[10] J. Daemen, R. Govaerts, and J. Vandewalle, "Weak keys for IDEA," IN T. Helleseth, editor, Advances in Cryptology-Proc. Eurocrypt' 93, LNCS 773, pp.224-231, springer Verlag, 1994.

[11] J. Daemen, R. Govaerts, and J. Vandewalle, "Cryptanalysis of 2.5 rounds of IDEA," Techjnical Report ESAT-COSIC Report 94-1, Department of Electrical Engineering, Katholieke Universiteit Leuven, March 1994.

[12] Faul R. Jordan and Ronald D. Williams, "VCOMP : A VHDL Composition System," 26th ACM/IEEE Design Automation, pp.750-753, June 1989.

[13] G. Toacse, D. Nicula, "Circuite Integrate Digitale," Teora, 1996.

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

□ 著者紹介

이 상 덕

1997년 조선대학교 전자공학과 학사
1998년 조선대학교 대학원 전자공학과 석사과정

※ 주관심 분야 : 통신보안시스템설계, ASIC설계, 네트워크 보안

한 승 조

1980년 조선대학교 전자공학과 학사
1982년 조선대학교 대학원 전자공학과 석사
1994년 충북대학교 대학원 전자계산학과 박사
1997년 조선대학교 전자·정보통신공학부 교수
1986년 6월 ~ 1987년 3월 Univ. of New Oleans 객원 교수
1995년 2월 ~ 1996년 1월 Univ. of Texas 객원 교수

※ 주관심 분야 : 통신보안시스템설계, 네트워크 보안, ASIC 설계, 음성합성