通信情報保護學會論文誌
第 8 卷 第 3 號. 1998. 9

VHDL을 이용한 고속 DES 암호칩 설계 및 구현

한 승 조*

Design and Implementation of High Speed Encryption Chip of DES using VHDL

Seungjo Han*

요 약

본 논문에서는 컴퓨터 시스템에서 정보보호를 위해 가장 많이 사용하고 있는 DES(Data Encryption Standard) 암호알고리즘을 시스템 설계 기술언어인 VHDL(Vhsic Hardware Description Language)로 설계하고 이것을 칩으로 합성하여 하드웨어에서 차지하는 면적과 속도를 비교 분석하였다. 설계방법에 있어서는 구현하는 방법에 따라 전 라운드 구현형, S-box 공유형 그리고 단일 라운드 반복형 등의 방식을 이용하였으며 그 중에서 면적을 가장 작게 차지하는 단일 라운드 반복형을 범용성을 갖도록 하여 FPGA로 구현한다.

본 논문에서 구현한 단일 라운드 반복형 설계는 Synopsys의 EDA 툴을 이용하여 시뮬레이션 및 합성을 하였고, Xilinx사의 xdm을 이용하여 XC4052XL 칩에 구현하였다. 그 결과 입력 클록 50MHz 상에서 100Mbps의 암·복호화 속도를 갖는 범용성 암호칩을 설계 및 구현한다.

Abstract

This paper seeks not only to design DES(Data Encrytion Standard) algorithm, a standard encrytion algorithm by VHDL (Vhsic Hardware Description) which is a designed system-description language but also to synthesize a chip. By doing so, we could analyze the size and speed of the hardware. In designing the method of comparing objects, we adopted the implemental method and designed a method by using full-round implemental method, S-box sharing method, and simple-round iterative method: thus, using the simple-round iterative method which has the smallest space as well as modifying it, we implemented it into FPGA chip.

The simple-round iterative design which we tried in this project used Synopsys' EDA tool for simulation and synthesis and could implement XC4052XL by using Xilinx's xdm. As a result, we could design the general- purpose high speed encryption chip of 100Mbps at the input pulse of 50Mhz.

이 논문은 1998년도 정보통신연구관리단의 산·학·연 공동기술개발사업 지원에 의해 연구되었음.

* 조선대학교 전자·정보통신 공학부 교수(조선대학교 수송기계부품 공장자동화 연구센터 연구원)

1. 서 론

컴퓨터의 성능이 급속도로 발전하고 컴퓨터 통신망이 전세계로 확대, 개방화됨에 따라 다가오는 21세기 정보화 사회에서는 정보가 재화(財貨)의 가치로 인정받고 국가경제 발전의 성쇠를 좌우하는 중요한 변수로 작용하게 되었다.

따라서 전세계적으로 중요한 정보를 보다 안전하게 보호하기 위한 다양한 방법들이 연구되어지고 있으며 그 중에서 암호화를 이용한 정보의 보호방법이 가장 많이 사용되고 있다. 암호화를 위해서는 암호알고리즘을 사용하는데, 현재 가장 보편적으로 실용화되어 사용되는 암호알고리즘이 IBM의 Lucifer 알고리즘을 기반으로 개발한 DES이다. DES는 암호화, 복호화 알고리즘이 대칭적이며 치환과 대치 그리고 S-Box로 구성된 블록암호화 시스템이다[1].

일반적으로 어떤 시스템에 암호화를 적용할 경우 암호화 처리는 각종 컴퓨터 시스템이나 통신시스템에서 부가적인 처리로 적용되어지기 때문에 느린 암호화 처리가 시스템의 속도 지연을 발생시키게 된다[2]. 따라서 고속의 컴퓨팅이나 고속통신에 있어서 이에 맞는 고속의 암호화는 필수적으로 해결되어야 할 과제이다. 이에 처리속도의 향상을 위한 연구가 다각도로 이루어지고 있으나 국내에서는 아직도 정보보호를 위한 시스템들의 대부분이 소프트웨어로 구현되고 있는 실정이며, 특히 DES를 소프트웨어로 구현하게 되면 계산의 복잡성과 동작환경의 제약에 따라 10Mbps 미만의 암호화 속도를 가지게 되며, 최근의 고속 컴퓨터 시스템에서도 60Mbps 미만의 암호화 속도를 갖는다[3][4][5]. 그러나 실시간 통신이나 100Mbps 이상의 초고속 통신망에서 암·복호화 동작이 시스템에 미치는 부하 등을 최소화하기 위해서는 암호화 시스템 또한 100Mbps 이상으로 동작하여야만 한다[6][7]. 그러므로 실시간 통신이나 고속통신시스템에 암호화를 적용하기 위해서는 반드시 하드웨어로 구현하여야 한다[8][9][10].

따라서 본 논문에서는 널리 알려진 단일키를 사용하는 블록 암호화 방식인 DES를 하드웨어로 설계하기 위해서 전 라운드 구현형, S-box 공유형 그리고 단일 라운드 반복형 등의 3가지 방식을 적용하여 설계하고, 그 중에서 크기와 면적이 가장 작은 단일 라운드 반복형을 FPGA칩으로 구현한다.

2. 이론적 배경

2.1. DES의 이론적 배경

DES는 대치와 치환이 반복되는 알고리즘으로 원문의 한 블록을 64비트씩 읽어들여, 이를 2개의 32비트 서브 블록(L, R)으로 나눈 후 16개의 암호라운드를 거쳐 출력하게 된다. 이때 각 암호라운드마다 암호함수를 적용한다[1].

DES의 암호화 과정은 크게 세 단계로 진행된다. 첫째, 64비트 평문이 치환된 입력을 생성하기 위해 비트열의 순서를 바꾸는 초기순열(IP: initial permutation), 둘째, 순열과 치환이 포함된 동일함수의 16회 반복, 셋째, 64비트 암호문 생성을 위한 역초기순열(IP-1)등이 그것이다. 또한 부분키(sub-key)의 생성도 크게 세 단계로 나누어 진행되는데, 먼저 키가 순열함수를 통과한 후, 16회에 걸쳐 좌측 순환이동(left circular shift)을 수행하며, 이때 순환이동 후 생성된 값들이 순열함수를 통과하여 sub_key(Ki)로 출력된다.

순열함수는 각 반복 과정에서 동일하지만 키 비트의 반복적인 이동으로 각각 다른 서브키가 생성된다. 복호화는 암호화와 동일한 구조로 이루어지며, 다만 서브키의 입력이 암호

화에는 sub_key1부터 sub_key16 순으로 입력이 되고 복호화에는 역순으로 sub_key16부터 sub_key1까지를 입력하면 된다.

암·복호화에 적용되는 단일 라운드를 식으로 표현하면 다음과 같다.

- 암호화 : $L_i = R_{i-1}$

$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

- 복호화 : $R_{i-1} = L_i$

$L_{i-1} = R_i \oplus f(R_{i-1}, K_i)$

$= R_i \oplus f(L_i, K_i)$

여기에서

$f(R_{i-1}, K_i) = P(S_1(B_1),\cdots\cdots,S_8(B_8))$

$B_1, B_2, B_3,\cdots\cdots,B_8 = E(R_{i-1}) \oplus K_i$

이다.

초기순열(IP)과 16개의 단일 라운드(simple round) 그리고 역초기순열(IP-1)로 구성된 DES를 블록도로 나타내면 그림 1과 같다. 여기서 점선으로 묶여 있는 블록이 단일 라운드로서 16개의 라운드에 동일한 구조로 이루어져 있다.

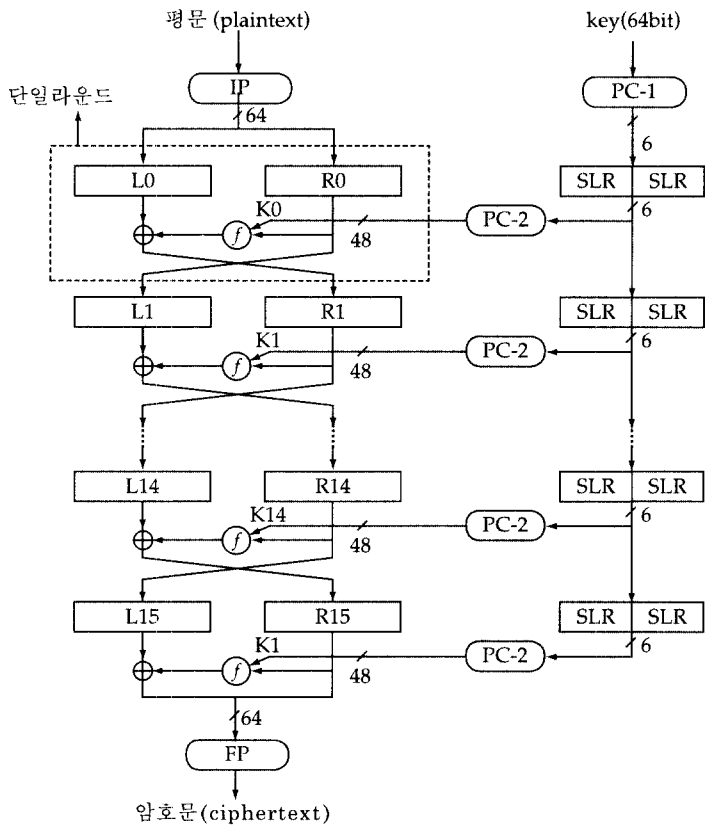


그림 1. DES의 블록 다이어그램

3. 칩의 설계

본 논문에서는 DES의 하드웨어 설계를 위해 3가지의 설계방식을 제안한다. 첫째, 순열과 치환이 포함된 함수의 16라운드 전부를 직렬로 연결하여 데이터의 파이프라인 처리를 적용한 전 라운드 구현형, 둘째, 16라운드를 직렬로 연결하되 하드웨어적으로 면적을 가장 많이 차지하는 S-box를 1개만 구현한 후 16개의 라운드에서 공유하게 하는 S-box 공유형, 마지막으로 DES의 16라운드를 1개의 단일 라운드 프로세서에 의해서 반복 수행하게 하는 단일 라운드 반복형 등이다.

앞에서 제안한 3가지 방식으로 암호칩을 구현하기 위하여 다음과 같이 3개의 기능별로 분류를 하여 탑다운 설계를 하였다.

- 블록단위(64비트)로 데이터를 암호화시키는 암호처리부 (Round Block)
- 칩의 전체적인 흐름을 제어하고, 데이터의 입출력을 제어하는 제어부 (Control Block)
- 데이터를 암호화시키기 위해 각 라운드에 필요한 키 값을 생성시키는 키생성부 (Key Block)

암호처리부, 제어부 그리고 키생성부는 하나의 칩으로 구현하게 되며, 외부에서 제어신호에 의해 데이터의 입출력 및 암·복호화가 이루어지게 된다.

3.1. 전 라운드 구현형

전 라운드 구현형은 16회 반복되는 단일 라운드를 일렬로 연결하는 구조로서 파이프라인 처리가 가능한 특성을 갖는다. Key Block은 16개 라운드의 서브키를 동시에 Round Block에 제공해 줌으로써 Round Block에 있는 16개의 단일 라운드들이 동시에 동작할 수 있는 요건을 갖춘다. Key Block에 제공되는 키는 데이터의 암호화 전에 먼저 제공되게 되고, 이 Key Block에서는 제공된 키를 가지고 16개의 서브키를 생성하게 된다.

먼저, 데이터가 입력되면 초기순열 테이블을 거쳐서 첫 번째 라운드의 입력으로 들어간다. 첫 번째 라운드에서 암호화가 수행된 후 출력은 두 번째 라운드의 입력으로 연결되어진다. 이렇게 16라운드가 수행되고 나면 역 초기순열 테이블을 거쳐서 암호화된 데이터를 출력하게 된다. 키의 생성은 입력된 키를 1차 순열 테이블과 로테이트 함수 그리고 2차 순열 테이블을 거쳐 1개의 서브키를 생성한다. 이때 1라운드의 서브키 생성에서 로테이트 함수를 거쳐 나온 값을 다시 로테이트 함수와 2차 순열 테이블을 통과시켜 다음 라운드의 서브키를 생성한다. 이 과정을 16회 반복하여 16개의 서브키를 생성해 놓게 된다. 여기에서 16개의 단일 라운드들은 동시에 동작을 하게 되며, 이 과정을 동작 알고리즘으로 기술하면 그림 2와 같다.

```
/* Key Block - 서브키를 생성하는 블록                    */
/*      Key : 초기에 입력되는 키                         */
/*      Enc__Dec : 암호화 또는 복호화 선택               */
/*      SK1__Out .. SK16__Out : 1~16라운드의 서브키 */
Procedure Key__Block (Key, Enc__Dec, SK1__Out, SK2__Out, ..., SK16__Out)
  Begin
      K := PC__1[ Key ];
      Call Rotate ( K, Enc__Dec, Shift__Tbl[1] );
      SK1__Out := PC__2[ K ];
```

```
Call Rotate ( K, Enc__Dec, Shift__Tbl[2] );
SK2__Out := PC__2[ K ];



Call Rotate ( K, Enc__Dec, Shift__Tbl[16] );
SK16__Out := PC__2[ K ];
End;

/* Round Block - 암호화 또는 복호화를 수행하는 블록        */
/*      In__Data : 암호화 또는 복호화를 위한 데이터        */
/*      SK1 .. SK16 : 1~16라운드의 서브키                  */
/*      Out__Data : 암호화 또는 복호화 된 데이터           */
Procedure Round__Block (In__Data, SK1, SK2, ···, SK16, Out__Data)
   Begin
Data1 := Ip__Table[ In__Data ];

      Data2(Left) := Data1(Right);
      Data2(Right) := P1[ S1[ E1[ Data1(Right) ] XOR SK1 ] ] XOR Data1(Left);
      Data3(Left) := Data2(Right);
      Data3(Right) := P2[ S2[ E2[ Data2(Right) ] XOR SK2 ] ] XOR Data2(Left);
                  .
                  .
                  .
      Data16(Left) := Data15(Right);
      Data16(Right) := P16[ S16[ E16[ Data15(Right) ] XOR SK16 ] ] XOR Data15(Left);

      Out__Data := Fp__Table[ Data16 ];
End;
```

그림 2. 전 라운드 구현형의 동작 알고리즘

3.1.1 Control Block

전 라운드 구현형은 단일 라운드를 일렬로 연결한 형태이기 때문에 입력 데이터가 출력되기까지는 16개의 라운드를 거쳐야 한다. 따라서 마지막 입력데이터의 잔류 문제 등을 제어하기 위해 그림 3과 같이 round counter가 필요하며(16라운드가 끝나면 _rnd_end가 Active), 기타 각 블록간의 동기화를 유지하기 위해 controller가 존재해야 한다. controller는 입력되는 제어신호선이 키의 입력을 나타낼 때는 Round Block은 비활성화하고 Key Block은 활성화(_k_en을 Active)시켜서 16개의 서브키를 생성하게 한다. 또 입력된 제어신호선이 데이터의 입력을 의미할 때에는 Key Block은 비활성화하고(_r_en을 Active) Round Block은 활성화시켜서 16회의 라운드를 동작시키는 역할을 하게 된다.

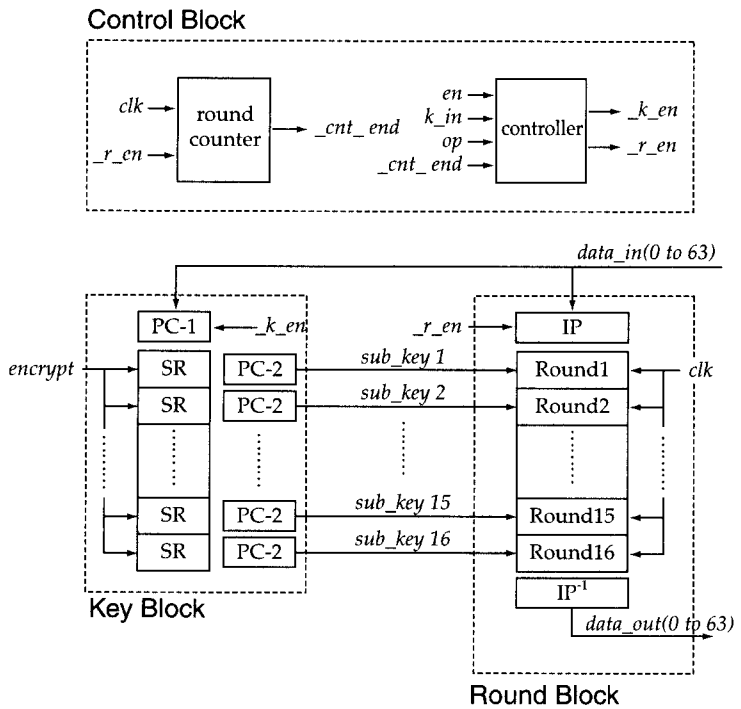


그림 3. 전 라운드 구현형의 블록 다이어그램

3.1.2 Key Block

Key Block은 16개의 단일 라운드에서 필요로 하는 서브키(sub_key1~sub_key16)를 생성하는 블록으로 Control Block의 제어 하(_k_en이 Active이면 Key Block이 동작)에서 Key의 입력을 받고 이것을 PC-1과 SR(shift rotate) 그리고 PC-2를 거쳐 Round Block에 제공해 준다. 이때 전단계의 SR의 출력은 다음 단계의 SR의 입력이 된다. 각각의 SR은 encrypt 시그널에 의해 암호화 또는 복호화의 key를 생성하게 되는데, 암호화 때에는 (encrypt가 active) SR이 left shifter로 동작하고, 복호화 때에는(encrypt가 not active) SR이 right shifter로 동작한다.

복호화는 암호화의 sub_key1~sub_key16이 각 라운드에 필요한 것과는 반대로 sub_key16~sub_key1이 필요하므로 left shifter 대신 right shifter를 수행함으로써 복호화 동작시에 필요로 하는 서브키를 생성하는 회로를 대신한다.

3.1.3 Round Block

Round Block은 그림 3과 같이 IP와 16개의 simple round processor 그리고 IP^{-1}(FP)가 직렬로 연결되어 있는 구조로 1 time clock동안에 1 round processor씩 처리하게 된다. 따라서 입력된 데이터가 암호화되어서 출력되는데는 16 time clock이 필요하다. 하지만 16개의 round processor를 파이프라인으로 처리하면 64비트를 암호화하는데 소요된 클럭은 1 time clock에 근사하게 된다.

3.2. S-box 공유형

DES 알고리즘의 실질적인 비도에 영향을 주는 부분은 치환연산을 수행하는 S-box로써 이것을 하드웨어로 구현하게 될 경우 ROM으로 대치할 수 있다. 그러나 전 라운드 구현형으로 S-box를 ROM으로 구현하게 되면 32Kbit의 용량을 필요로 하게 되며, 또한 1280개 이상의 net를 필요로 한다. 따라서 DES칩의 면적과 net를 줄이기 위해서 전 라운드 구현형에서 16개의 라운드가 1개의 S-box를 공유하도록 설계하였다.

데이터가 입력되어 처리되는 과정은 전 라운드 구현형과 동일하나 각각의 단일 라운드들은 1개의 S-Box를 공유하며, 따라서 16개의 단일 라운드들이 동시에 동작하지는 못하게 된다. 이 과정을 동작 알고리즘으로 표현하면 그림 4와 같다.

```
Procedure Key_Block (Key, Enc_Dec, SK_Out)
 Begin
      K := PC_1[ Key ];
      Call Rotate ( K, Enc_Dec, Shift_Tbl[1] );
      SK1_Out := PC_2[ K ];
      Call Rotate ( K, Enc_Dec, Shift_Tbl[2] );
      SK2_Out := PC_2[ K ];
                ·
                ·
                ·
      Call Rotate ( K, Enc_Dec, Shift_Tbl[16] );
      SK16_Out := PC_2[ K ];
      End;

Procedure Round_Block (In_Data, SK1, SK2, ···, SK16, Out_Data)
 Begin
      Data1 := Ip_Table[ In_Data ];

      Data2(Left) := Data1(Right);
      Data2(Right) := P1[ S[ E1[ Data1(Right) ] XOR SK1 ] ] XOR Data1(Left);
      Data3(Left) := Data2(Right);
      Data3(Right) := P2[ S[ E2[ Data2(Right) ] XOR SK2 ] ] XOR Data2(Left);
                ·
                ·
                ·
      Data16(Left) := Data15(Right);
      Data16(Right) := P16[ S[ E16[ Data15(Right) ] XOR SK16 ] ] XOR Data15(Left);

      Out_Data := Fp_Table[ Data16 ];
End;
```

그림 4. S-Box 공유형의 동작 알고리즘

3.2.1 Control Block

그림 5에서와 같이 S-box 공유형에서 Control Block의 가장 중요한 동작은 각 라운드가 동작을 할 때 S-box의 치환 테이블을 각 라운드와 연결을 해주는 타이밍에 있다고 할 수 있다. S-box는 동시에 2개 이상의 라운드에

치환 테이블을 제공해 줄 수는 없기 때문에 16개의 라운드를 동시에 병렬처리하지는 못한다는 단점이 있으나 각 라운드를 실행하는데 1개의 time clock만을 필요로 하기 때문에 16개의 라운드를 수행하는데 16 time clock만으로 충분하다. 또한 S-box를 제외한 대치나 확장, XOR 부분들은 전후 S-box와 병렬로 동작할 수가 있다.

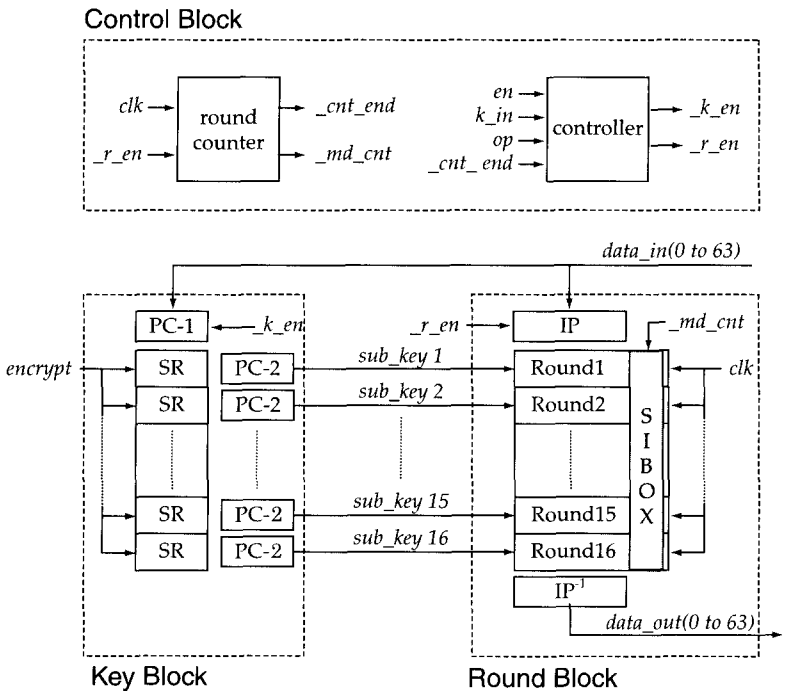


그림 5. S-box 공유형의 블록 다이어그램

3.2.2 Key Block

S-box 공유형 설계에서의 Key Block은 전라운드 구현형의 Key Block과 동일하여 16라운드에 필요한 서브키를 미리 생성하여 서브키 버퍼에 저장해 놓고 Round Block에서 참조할 수 있도록 해 놓는다.

3.2.3 Round Block

Round Block 내의 S-box는 1개만 존재하며 16개 라운드에 공통으로 연결되어 각 라운드에 순차적으로 치환 테이블을 제공해 주어야 한다. 각 단일 라운드는 매 time clock동안 위에서부터 순차적으로 동작하게 되며, 이때 필요한 치환 테이블은 공유된 S-box로부터 제공받게 된다.

S-box 공유형의 Round Block은 S-box를 1개로 줄여 전 라운드 구현형에 비해 칩이 차지하는 면적이 90%이상 줄어들며, 처리시간이 절반으로 감소된다.

3.3. 단일 라운드 반복형

단일 라운드 반복형은 매 라운드의 동작과 동시에 키를 생성하게 함으로써 Key Block 내

의 16개의 서브키를 저장하는데 사용한 서브키 버퍼를 없애고, 암호화 16라운드를 단일 라운드 processor만 16회 반복하여 구현하게 함으로써 16개의 S-box를 1개로 감소시키는 방식이다.

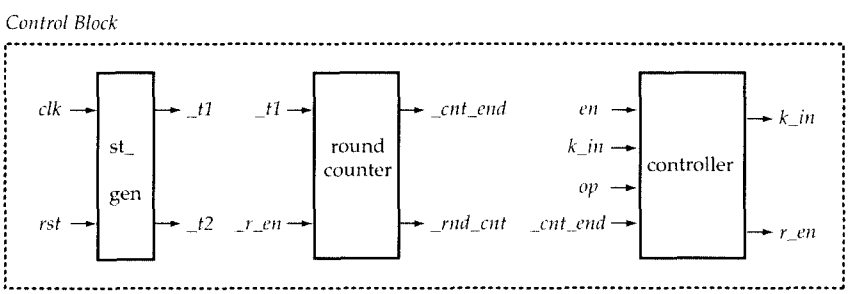
먼저 입력된 데이터를 초기순열 테이블을 통과시킨 후 단일 라운드의 입력으로 보낸다. 단일 라운드 프로세서는 입력을 제어신호의 동기에 맞추어 암호화를 수행 후 출력으로 내보낸다. 이때 단일 라운드의 출력을 다시 입력으로 피드백 시킨다. 이 과정을 16회 반복한 후 역초기순열 테이블을 통해 암호화된 데이터를 출력시키며 이러한 과정에 대한 동작 알고리즘은 그림 6과 같다.

본 논문에서는 단일 라운드를 반복시키기 위해 입력되는 클록을 _t1, _t2로 이등분을 시켰으며 Key Block 내에 초기 입력된 키를 위한 Key Register를 두었다. key는 미리 생성해 놓지 않고 필요한 때에 생성하게 된다.

그림 7은 단일 라운드 반복법으로 암호칩을 설계한 블록 다이어그램이다.

```
Procedure Key_Block (Key, Enc_Dec, SK_Out)
 Begin
      K := PC_1[ Key ];
      For Round = 1 To 16 Do
       Begin
            Call Rotate ( K, Enc_Dec, Round );
            SK_Out := PC_2[ K ];
       End;
 End;
 Procedure Round_Block (In_Data, SK, Out_Data)
  Begin
      Data := Ip_Table[ In_Data ];
      For Round = 1 To 16 Do
       Begin
            Data(Left) := Data(Right);
            Data(Right) := P[ S[ E[ Data(Right) ] XOR SK ] ] XOR Data(Left);
       End;
    Out_Data := Fp_Table[ Data ];
End;
```

그림 6. 단일 라운드 반복법의 동작 알고리즘



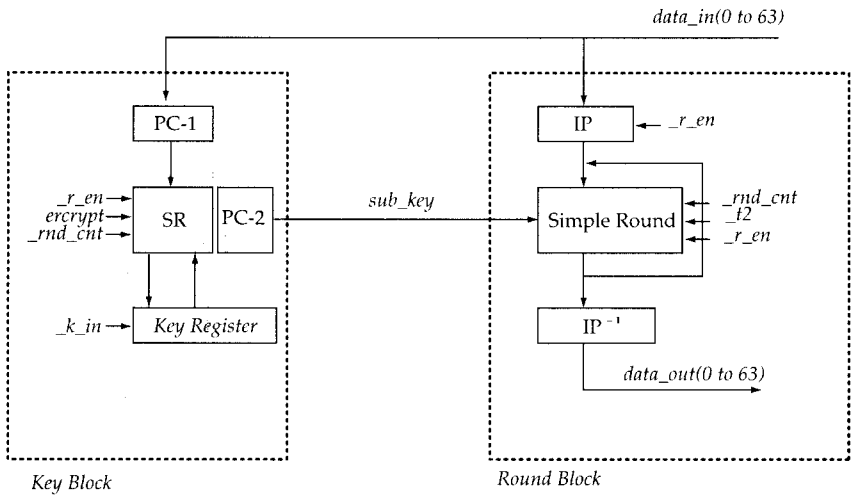


그림 7. 단일 라운드 반복법 블록 다이어그램

3.3.1 Control Block

그림 7에서와 같이 Control Block은 외부에서 들어오는 제어신호를 분석하여 각 블록들의 동작을 제어하는 controller, 수행 라운드를 카운트하는 round counter 그리고 칩의 전체적인 동기를 유지하기 위해 입력 클록을 이등분하는 state generator 등으로 구성된다.

단일 라운드 반복형은 키를 생성할 때 SR이 쉬프트 할 횟수를 지정하기 위해 현재 수행중인 라운드를 인식 할 필요가 있으며, round processor가 16회 반복하기 위한 반복횟수를 보관하고 있어야 한다. round counter는 현재 수행중인 라운드가 몇 라운드인지를 나타내는 4비트의 신호선을 출력으로 갖으며, 암·복호화 시작신호인 _r_en이 활성화 될 때 0부터 15까지 카운트하고, _cnt_end 출력신호를 활성화 시키면서 동작을 정지한다.

단일 라운드를 반복 수행할 때는 구성되어 있는 각각의 블록들이 서로 동기를 유지하며 유기적인 동작을 갖어야 한다. 예를 들면 라운드가 동작중일 때 현재 라운드에 맞는 서브키를 생성해서 제공해 주어야 하고 16라운드가 수행이 완료된 후에는 출력된 결과를 칩 외부로 내보내고 새로운 데이터를 받을 준비를 하여야 한다. 또한 busy 신호선을 통하여 칩 외부로 현재의 작업 완료 여부를 알린다. 그러나 이러한 동작의 순차는 외부의 제어와 관계되어야 한다.

controller는 상기와 같이 외부의 제어신호에 따라 칩 내부의 블록들에게 제어신호를 전송해주는 부분이다. 또한 이때 이루어지는 암호화 16라운드의 모든 동작은 2 state 사이클 단위로 반복되어진다. 암·복호화는 2사이클 동안에 한 라운드씩 총 32사이클 동안 수행하게 되며, state generator는 이 2사이클을 구분하기 위해 입력되는 클록을 __t1, _t2로 분리한다.

3.3.2. Key Block

Key Block은 입력된 64비트의 데이터를 가지고 16개의 서브키를 생성하는 블록으로 전 라운드 구현형이 동시에 서브키를 생성해 놓고 수행하는 것과는 달리 단일 라운드 반복형은 암호라운드의 수행 바로 전에 서브키를 생성하는 방식으로서 서브키 버퍼를 필요로 하

지 않는다.

Key의 입력은 Key-in 제어 신호선에 의해서 외부에서 입력받아 PC-1을 거쳐 Key Register에 저장해 놓는다. 그 후 암호라운드가 동작하게 되면 라운드 카운터(_rnd_cnt)가 지시하는 라운드의 서브키를 생성하여 Round Block에 제공한다.

이때 encrypt 제어신호선이 '1'이면 서브키를 sub_key1~sub_key16 순으로 생성하고, '0'이면 서브키를 sub_key16~sub_key1 순으로 생성한다.

3.3.3. Round Block

Round Block은 1개의 단일 라운드의 수행블록(round processor)으로 구성되어 있으며 이 단일 라운드를 16회 반복하여 DES를 수행한다.

IP-box, FP-box 그리고 단일 라운드 등으로 구성되어 있는 round processor는 _r_en이 '1'(활성상태)일 때 __t1과 __t2에 따라 단일 라운드를 수행한다. 초기 데이터는 라운드 카운터(_rnd_cnt)가 "0000"일 때 IP-box를 거쳐 내부 라운드 버퍼로 전달되며, 라운드 카운터의 증가에 따라 16회 반복 수행하고, 마지막 라운드를 수행한 후에는 FP-box를 거쳐서 출력하게 된다.

내부적으로 단일 라운드는 _t2의 상승에지에서 동작하게 되며, _t1의 상승에지에서 round counter의 카운트 증가와 함께 전 단계의 단일 라운드의 출력을 다시 다음 단계의 단일 라운드의 입력으로 피드백 시켜 16라운드를 반복한다.

4. 칩의 합성

VHDL을 이용하여 칩을 설계하게 되면 스키메틱으로 칩을 설계하는 것과는 달리 설계된 회로를 시뮬레이션을 거쳐서 회로를 검증하고 검증된 회로를 게이트 레벨로 변환하는 합성의 과정을 거쳐야 한다. 이 합성의 과정에서 게이트의 최적화(optimize)도 함께 이루어진다[11][12][13][14].

본 논문에서는 설계된 암호칩을 VHDL 코드로 기술하고 Synopsys tool을 이용하여 합성을 하였으며, 칩의 면적과 속도를 비교하기 위해 FPGA의 cell 개수와 소요 time clock을 사용하여 비교 분석하였다.

4.1. 전 라운드 구현형

전 라운드 구현형으로 칩을 합성하게 되면 칩의 면적을 가장 많이 차지하는 부분은 Key Block과 Round Block내의 S-box로 나타났다. Key Block은 16라운드에 사용되는 서브키를 저장하기 위한 버퍼가 차지하는 공간이 크기 때문이다. 이는 FPGA로 RAM을 구현하게 되는 결과로 인해 면적이 증가한다. Round Block 내의 S-box는 S-box 내의 치환 테이블이 ROM처럼 쓰여졌기 때문에 차지하는 면적이 많게 나타났다.

표 1은 전 라운드 구현형으로 칩을 구현하여 합성한 결과 나타난 cell 면적과 소요 time clock이며, 그림 8은 합성 결과에 따른 최적화된 스키메틱 회로이다.

표 1. 전라운드 구현형의 합성결과에 따른 cell 면적과 소요 time clock

	total cells (CLBs+cell)	1회 암호화 소요 time clock
Control Block	7	≈1 time clock
Key Block	386	
Round Block	5024	
Total	5417	

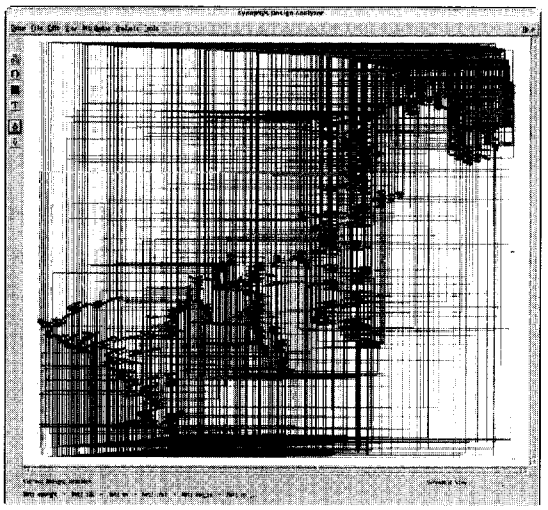


그림 8. 전 라운드 구현형의 합성결과

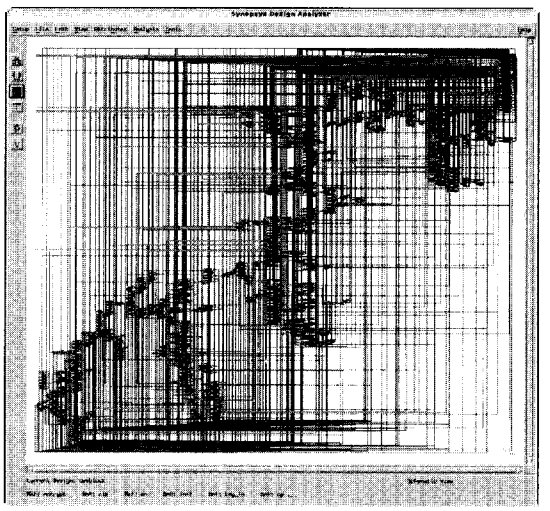


그림 9. S-box 공유형의 합성결과

4.2. S-box 공유형

전 라운드 구현형에서는 칩의 면적 중 가장 많은 영역을 차지하는 부분이 S-box이므로 이러한 S-box를 16라운드에서 공유하도록 설계하면 그렇지 않을 경우에 비해 암호칩의 면적을 90%이상 줄일 수 있다.

따라서 표 2는 Control Block과 Key Block이 전 라운드 구현형과 동일한 결과를 나타내고 있고 Round Block 부분만 크게 감소되어 있는 것을 보인다. 그림 9는 S-box를 공유하는 방식을 적용한 합성 결과에 따른 최적화된 스키메틱 회로이다.

표 2. S-box 공유형의 합성결과에 따른 cell 면적과 소요 time clock

	total cells (CLBs+cell)	1회 암호화 소요 time clock
Control Block	7	16 time clock
Key Block	386	
Round Block	1064	
Total	1457	

4.3. 단일 라운드 반복형

칩의 면적을 가장 많이 차지하는 부분인 Key Block내의 서브키 버퍼와 Round Block내의 S-box를 1개씩만 가지고 16회 반복을 하여 DES를 구현하는 방법인 단일 라운드 반복형으로 합성한 칩의 면적이 전 라운드 구현형보다 약 93.75%정도 작게 나타났다. 그러나 1개의 단일 라운드 processor를 16회 반복하게 됨으로써 파이프라인처리 같은 병렬처리에 제약이 따르게 되었고 따라서 전 라운드 구현형보다는 동작 속도 면에서 제한을 받게 되었다.

따라서 표 3에서 보인 바와 같이 Key Block과 Round Block이 전 라운드 구현형이나 S-box 공유형에 비해 크기가 크게 감소되었음을 보인다.

그림 10은 단일 라운드 반복형으로 칩을 구현한 합성 결과에 따른 최적화된 스키메틱 회로이다.

표 3. 단일 라운드 반복형의 합성결과에 따른 cell 면적과 소요 time clock

	total cells (CLBs+cell)	1회 암호화 소요 time clock
Control Block	58	32 time clock
Key Block	169	
Round Block	386	
Total	613	

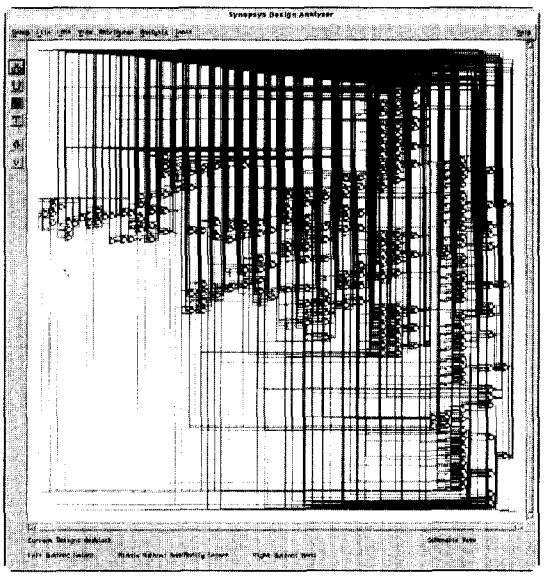


그림 10. 단일 라운드 반복형의 합성결과

5. 분석 및 고찰

본 논문에서 제안한 3가지 설계방식에 의한 합성결과 전 라운드 구현형의 16개 서브키가 차지하는 부분과 16개의 S-box가 차지하는 면적을 단일 라운드 반복형에서는 1개씩만 이용함으로써 전체적으로 전 라운드 구현형에 비해 11.3%의 크기로 감소시킬 수 있었다. 표 4는 이러한 3가지 설계방식에 대한 cell 면적과 소요시간을 비교하였다.

표 4. 3가지 설계 방식에 따른 합성결과

	전라운드 구현형	S-box 공유형	단일라운드 반복형
Subkey Register+S-box	5305	1210	468
Total Cells	5417	1457	613
소요 시간 (Clock)	1	16	32

실제로 FPGA 칩에 전 라운드 구현형을 설계하는 경우 서브키와 S-box를 별도의 고속 RAM과 ROM을 사용한다면 S-box의 경우 net가 각 라운드 당 80개(입력 48개, 출력 32개)씩 16라운드에 총 1280개 이상 필요하게 되기 때문에 전 라운드 구현형을 FPGA 칩으로 구현하기가 어렵다. 따라서 본 논문에서는 앞에서 설계된 3가지 방식의 암호칩 중에서 면적이 가장 적게 차지하는 단일 라운드 반복형을 Xilinx사의 FPGA 칩에 구현하였다.

먼저 단일 라운드 반복형으로 칩을 구현하기에 앞서 설계하고자 하는 칩이 범용성을 갖도록 다음과 같이 2가지를 고려하여 설계한다.

첫째, 데이터의 입력과 출력은 공통버스를 사용하고 버스의 크기를 8비트로 줄였다. 따라서 64비트의 데이터의 암호화시 암호칩은 8회 걸쳐 8비트 데이터를 입력받으며, 암호화된 결과를 8회에 걸쳐 출력하게 된다.

둘째, Control Block에 버퍼를 사용하므로서 8비트의 외부 입출력버스와 64비트의 내부버스 사이의 데이터 크기를 정합하도록 한다. 이에 따라 2가지를 고려하여 회로를 설계하여 내부 타이밍에서 데이터 입출력시 64비트의 내부 데이터 폭을 맞추기 위한 8비트의 8회 반복입력에 필요한 8 cycle time을 지연시켜서 단일 라운드 반복형 범용 암호칩을 구현 및 합성하였다.

추가된 회로 및 암호화 회로의 논리검증을 위해 다음과 같은 조건하에서 시뮬레이션 하였다.

- Key : "Infocomm" = (49 6E 66 6F 63 6F 6D 6D)$_{16}$
- Data : "DES-FPGA" = (44 45 53 2D 46 50 47 41)$_{16}$
- Simulator : Synopsys
- Running time : 1100ns
- clock : 50 MHz

위와 같은 조건에서 시뮬레이션 한 결과 그림11, 그림12와 같이 입력된 데이터가 암·복호화되어 출력된 데이터와 동일하므로서 암·복호화가 정상적으로 수행되었다. 입력의 64비트 데이터가 8비트 씩 8회에 걸쳐 입력되며 내부에서는 64비트 단위로 암호화가 이루어진다. 64비트 데이터의 암호화를 위해 소요된 암호화 수행시간은 입력주파수가 50MHz 일 때 총 640ns가 소요되었으며, 암호화 속도는 100Mbps 이었다.

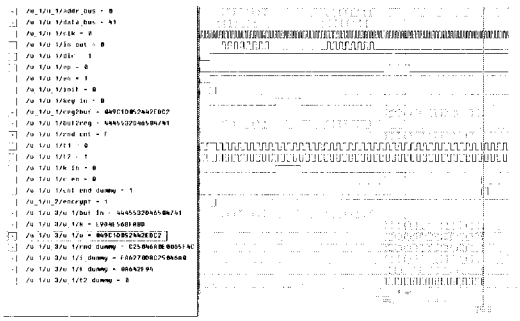


그림 11. 암호화 과정의 시뮬레이션

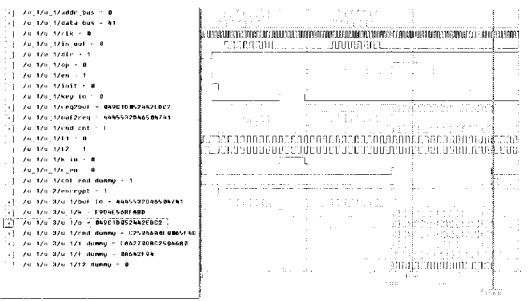


그림 12. 복호화 과정의 시뮬레이션

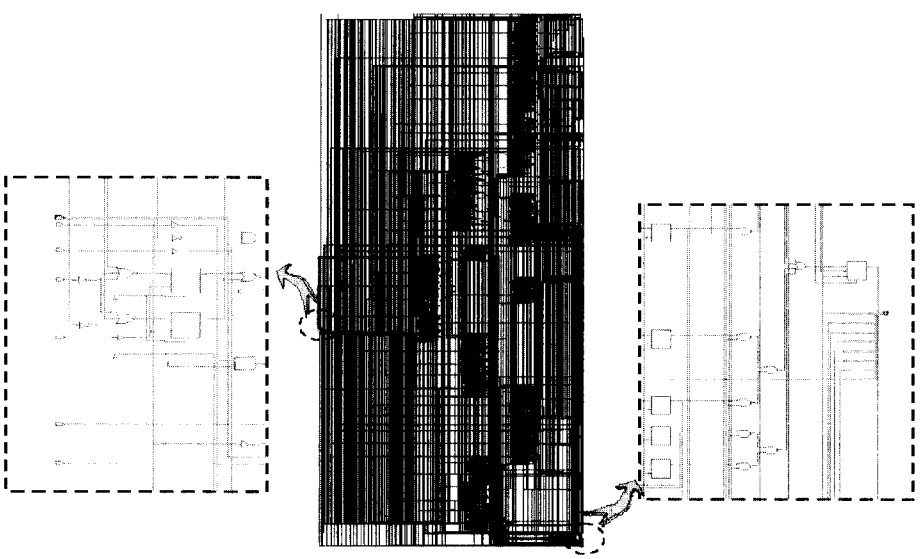


그림 13. 범용 암호칩의 합성결과

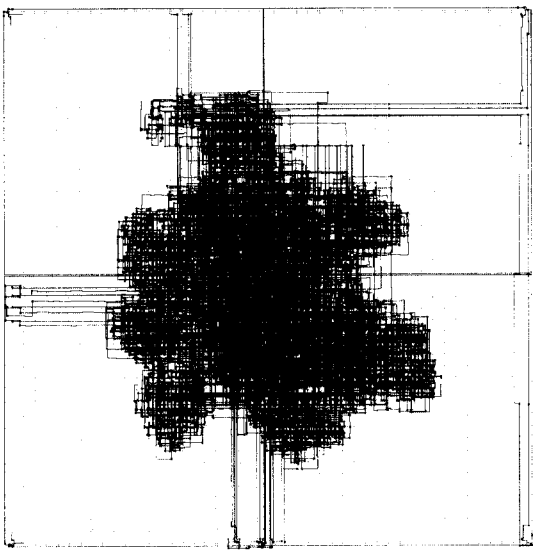


그림 14. 단일라운드 반복형을 XC4052XL에 Floorplanning한 결과

그림 13은 Synopsys tool을 이용하여 FPGA cell로 합성한 결과이다. 그림에서 왼쪽 부분은 범용으로 수정된 입력의 제어신호선에 대한 세부 스키메틱 회로를 보이고 있으며, 오른쪽은 양방향 데이터에 대한 입출력을 수행하는 8비트 크기의 데이터버스 스키메틱 회로이다.

그림 13의 스키메틱 회로를 Xilinx의 xdm을 이용하여 XC4052XL칩에 Floorplanning 한 결과는 그림 14에서 보여지고 있다.

6. 결 론

일반적으로 어떤 시스템에 암호화를 적용할 경우 암호화 처리는 각종 컴퓨터 시스템이나 통신시스템에서 부가적인 처리로 적용되어지기 때문에 느린 암호화 처리가 시스템의 속도 지연을 발생시키게 된다. 따라서 정보화시대인 오늘날에는 고속의 컴퓨팅이나 고속통신에 있어서 고속의 암호화는 필수적으로 해결되어야 할 과제이다.

이를 위하여 본 논문에서는 DES 암호화 알고리즘을 3가지의 설계방식을 이용하여 VHDL로 설계 및 합성하였고 수행 속도와 칩면적을 비교 분석하였다. 그 결과 단일 라운드 반복형에서는 cell 수가 613개였으며, 입력주파수가 50Mhz 일 때 수행시간은 640ns였다. 따라서 가장 작은 면적을 차지하는 단일 라운드 반복형을 범용성을 갖도록 개선하여 Xilinx FPGA에 Floorplanning함으로써 암호화를 고속으로 수행할 수 있는 칩을 구현하였다.

본 알고리즘을 주문형 반도체로 제작하게 되면 서브키에 사용된 레지스터나 버퍼를 고속의 RAM으로, S-box를 고속의 ROM으로 대체할 수 있다. 따라서 전 라운드를 전부 구현하는 것이 가능해지기 때문에 데이터의 파이프라인 처리 등, 단일 라운드 반복법에 비해 암호칩의 처리속도를 높이는 것이 가능하다. 따라서 암호칩을 ASIC으로 설계하고 최적화하는 것은 앞으로의 연구과제이다.

참 고 문 헌

[1] NBS, Data Encryption Standard, Federal Information Processing Standard Pub. 46, 1977

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

[3] Andreas Pfitzmann and Ralf Abmann, "Efficient Software Implementations of (Generalized)DES," SECURICOM '90, 1990

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

[5] Deborah Williams and Harvey J. Hindin, "Can software do encryption job?," Electronics, 1980.7

[6] R. Zimmermann, A. Curiger, H. Kaeslin, N. Felber, W. Fiehtner, "A 177Mb/s

VLSI Implementation of the International Data Encryption Algorithm," IEEE Journal of Solid State Circuit, Vol.29 No.3 March. 1994.

[7] Hans Eberle, "A High-speed DES Implementation for Network Applications," CRYPTO '92, 1992

[8] Frank Hoornart, Jo Goubert & Yvo Desmedt, "Efficient hardware implementation of the DES," Journal of Cryptology, Vol.4, no.1, pp.148-151.

[9] Ingrid Verbauwhede, Frank Hoornaert, Joos Vandewalle, and Hugo DeMan, "Security Considerations in the Design and Implementation of a new DES chip," Eurocrypt '87, 1987

[10] Z. Navadi, "Using VHDL for modeling 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.

[12] David R. Coelho, The VHDL Handbook, Kluwer Academic Publishers, Norwell, Massachusetts, 1989.

[13] Roger Lipsett, Carl Schaefer, Cary Ussery, VHDL : Hardware Description and Design, Kluwer Academic Publishers, Norwell, Massachusetts, 1989.

[14] Petra Michel, Ulrich Lauther, Peter Duzy, The Synthesis Approach to Digital System Design, Kluwer Academic Publishers, Norwell, Massachusetts, 1989.

□ 著者紹介

한 승 조

1980년 조선대학교 전자공학과 학사
1982년 조선대학교 대학원 전자공학과 석사
1994년 충북대학교 대학원 전자계산학과 박사
1998년 현재 조선대학교 전자·정보통신공학부 교수
1986년 6월 ~ 1987년 3월 Univ. of New Oleans 객원 교수
1995년 2월 ~ 1996년 1월 Univ. of Texas 객원교수
1998년 - 현재 조선대학교 전자정보통신공학부 교수

※ 주관심 분야 : 통신보안, ASIC, 음성합성