GOST 암호화 알고리즘의 구현 및 분석

Design and Analysis of the GOST Encryption Algorithm

류승석*, 정연모**
Seung-Suk, Ryu, Yunmo, Chung

Abstract

Since data security problems are very important in the information age, cryptographic algorithms for encryption and decryption have been studied for a long time. The GOST(Gosudarstvennyi Standard or Government Standard) algorithm as a data encryption algorithm with a 256-bit key is a 64-bit block algorithm developed in the former Soviet Union.

In this paper, we describe how to design an encryption chip based on the GOST algorithm. In addition, the GOST algorithm is compared with the DES(Data Encryption Standard) algorithm, which has been used as a conventional data encryption algorithm, in modeling techniques and their performance. The GOST algorithm whose key size is relatively longer than that of the DES algorithm has been expanded to get better performance, modeled in VHDL, and simulated for implementation with an CPLD chip.

* 삼성전자 LSI 개발팀
** 경희대학교 공과대학 전자공학과

1. 서 론

21세기 정보화 시대를 맞이하여 컴퓨터 및 통신상에서 모든 정보의 보호에 대한 중요성은 더해가고 있다. 정보처리 및 전송 시에 암호화(encryption) 및 복호화(decryption) 과정을 이용하면 타인에 불법적인 해독이 어려워지므로 보안성을 향상시킬 수 있다. 암호화 알고리즘과 복호화 알고리즘을 포함한 알고리즘을 암호알고리즘(cryptographic algorithm)이라 한다. 지금까지 암호알고리듬의 이용은 주로 CPU를 이용한 소프트웨어 구현이 대부분이고, 하드웨어를 이용한 구현은 많지 않은 것이 사실이다.

최근에 인터넷을 이용한 전자 상거래가 많아지면서 사용자들 간에 암호화된 데이터의 전송이 필요한 경우가 많아진다. 또한 컴퓨터뿐만 아니라 정보 단말기 등을 이용한 정보의 전달이 다양하게 이루어지고 있음을 고려하면 하드웨어로 구현된 암호알고리듬의 필요성이 더욱 절실하다 [3, 5,9,10].

미국 정부가 암호화의 표준으로 사용하고 있는 DES(Data Encryption Standard) 알고리즘[4,5,8]은 안정성은 있으나 56비트의 작은 key 값을 사용하므로 불법적인 해독에 취약하다. 따라서 DES가 발표된 후 이러한 문제점을 보완하려는 노력이 계속 되었다[4,5,7]. 최근에는 key의 크기를 늘이고 불법적인 해독에 대비할 수 있는 알고리듬의 개발에 대한 연구가 활발하며 또한 암호화 알고리듬을 하드웨어 칩으로 구현하기 위한 연구가 활발하다.

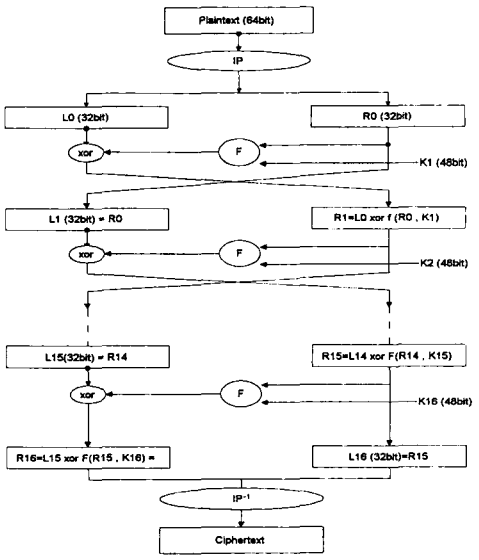
본 논문에서는 DES 알고리즘과 유사한 구조를 가지면서 key의 길이가 더 큰 GOST (Government Standard) 알고리즘[4]을 칩으로 구현하는 방법을 제시한다. 또한 구현 시에 차지하는 면적과 동작 속도에 대해서 DES 알고리듬의 경우와 비교 및 분석하였다. GDES(General DES)[7]구조를 기본 GOST 알고리즘에 적용하여 한번에 처리할 수 있는 데이터 양을 증가시켰다. 즉 GOST에서 64비트 단위로 처리하던 것을 128비트로 처리할 수 있는 구조로 변환하였다. 하드웨어화하는 과정에서 GOST 알고리듬의 안정성을 희생하지 않는다. 최종적으로 이 구조를 Altera사의 설계 도구인 MAX+Plus II를 이용하여 시뮬레이션을 통해 검증하였다[1].

2. 암호화 알고리즘

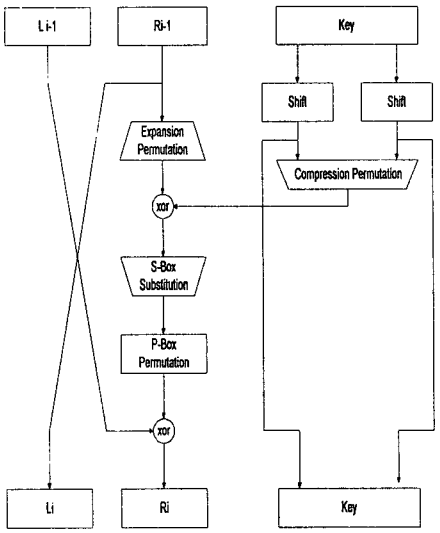
2.1 DES 알고리즘

비밀키 암호화 알고리듬의 하나인 DES 는 64비트로 이루어진 블록을 한꺼번에 암호화하는 블록 암호화 방식 중의 하나이다. 64비트 평문(plaintext)을 56비트(64비트 중에서 최상위 비트 8비트는 패리티 검사를 위해 사용되므로 제외)의 key를 사용하여 암호화한다. 암호를 해독하기 위해서는 key를 알아야하는데 이를 위해서는 이론적으로 최악의 경우에 2^{56} 만큼의 입력 벡터값의 입력해 보아야한다. 따라서 불법적으로 암호를 해독하기 위해서 많은 시간을 필요로 한다.

DES 알고리즘의 전체적인 순서도와 핵심부분인 round의 순서도는 각각 <그림 1> 및 <그림 2>와 같다



<그림 1> DES 전체 블록도



<그림 2> 한 round 의 블록도

<그림 1>의 전체 블록도에서 보는 바와 같이 DES는 총 16개의 round로 구성되어 있다. <그림 2>는 DES의 한 round를 나타낸다. 여기서는 오른쪽 32비트 데이터는 48비트로 변환하는 Expansion Permutation 과정을 거치며 다른 한 편으로 key는 Shift 및 Compression Permutation 과정을 거쳐서 48비트로 변환되어 데이터와 XOR 연산을 수행한다. 이 결과를 다시 32비트로 축소하는 S-BOX Substitution 과정이 있다. 또 왼쪽 32비트 블록과 XOR 연산을 수행하기 전에 P-Box Permutation 과정을 거친다. DES 알고리듬의 각 round에서 수행되는 각 연산에 대해서 자세히 설명한다.

1) IP(Initial Permutation)

이 과정은 암호화에 직접적으로 영향을 주지 않으며 바이트 단위로 구성된 평문이나 암호문을 쉽게 읽어 들이기 위해서 사용된다. 일반적으로 소프트웨어로 구현할 때는 이 과정을 생략한다. 만일 64비트의 입력 블록에 대해 이 과정을 수행하였으면 반드시 암호화가 끝난 후에 역과정을 수행하여야 한다.

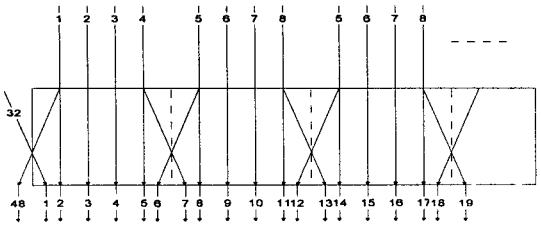
2) Key Transformation

먼저 64비트의 DES key에서 패리티 비트를 제외한 56비트로 줄이는 과정으로 시작한다. 이 56비트 key를 왼쪽 28비트와 오른쪽 28비트로 나눈 뒤에 매 round 마다 미리 지정된 비트 수만큼 왼쪽으로 Shift 연산을 수행한다. 그 결과를 다음 round의 입력 key로 사용한다.

또한 이 Shift 수행 결과인 56 비트의 두 블록을 이용하여 48비트를 만들어내는 Compression Permutation 과정을 거친다. 여기서는 매 round 마다 암호화를 위해 사용되는 다른 48비트 블록을 생성해 준다.

3) Expansion Permutation

이 과정은 64비트 중에서 오른쪽 32 비트의 값을 subkey값과 xor 연산을 하기 위해 48비트로 확장하는 과정이다. <그림 3>에서처럼 4비트 input 중 첫 번째와 네 번째 비트를 이웃 블록에 더해 줌으로써 4비트를 6비트로 확장시켜준다.



<그림 3> Expansion Permutation

4) S-box substitution

Key 값과 xor 연산된 오른쪽 블록 값 48비트를 6비트씩 8개로 나누어 수행한다. 하나의 S-box는 6비트 입력을 4비트 출력으로 줄여주는 역할을 한다. 즉 6비트의 입력을 차례로 b1, b2, b3, b4, b5, b6 라고 가정했을 때 비트 b1과 b6을 상위 비트로 하고 나머지 4비트를 하위 비트로 조합(b1b6b2b3b4b5)하여 미리 주어진 테이블 값에 따라 변환한다.

5) P-box substitution

여기서는 S-box 에 출력된 값을 받아서 각

입력 비트를 출력 위치로 치환하는 과정이다. 이 과정은 추가되거나 삭제되는 비트가 없으므로 Straight Permutation 또는 Permutation이라고도 한다.

2.2 GOST 알고리즘

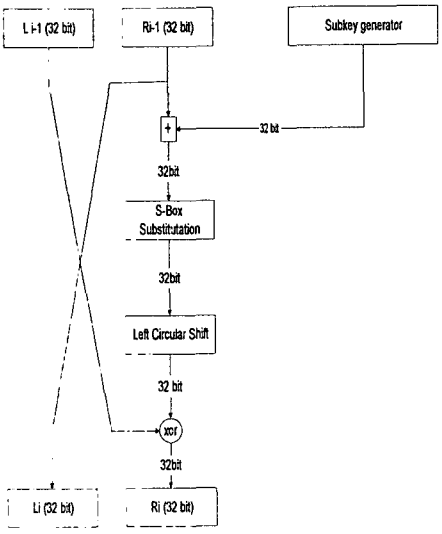
DES 알고리즘은 블록 암호화 알고리즘의 가장 기본 되는 표준으로서 어느 정도 안전성에 대해서는 인정받고 있다. 하지만 방대한 양의 자료를 처리해야 하는 경우에 속도를 고려하고 불법적인 해독을 방지하기 위해 key의 크기를 증가해야 하는 경우에는 한 쪽을 희생해야 되는 경우가 발생한다. 따라서 DES 알고리즘이 발표된 이후에 속도를 빠르게 하면서 보안 측면의 강화에 대한 연구가 활발히 진행되어 왔다.

GOST 알고리즘은 구 소련의 정부 표준으로서 28147-89라는 표준 번호를 가지고 있다[3,4]. GOST 알고리즘은 64비트로 이루어진 블록을 한꺼번에 암호화하는 블록 암호화 방식 중의 하나로서 256비트의 key값을 사용하여 암호화한다.

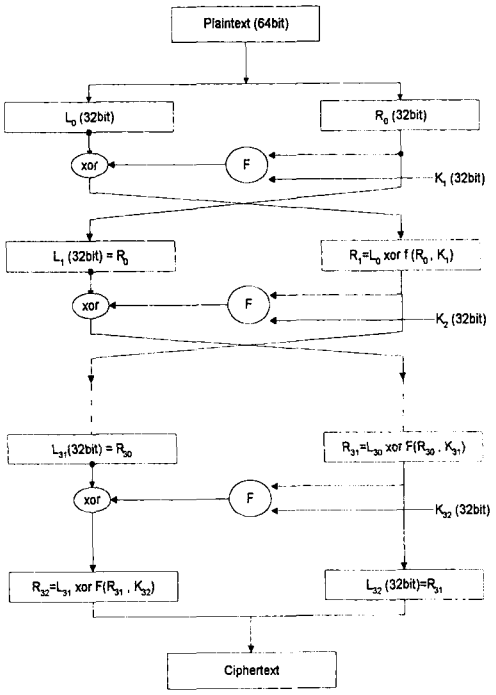
<그림 4>와 같이 연산과정의 round가 32회 반복되는데 이 알고리즘은 DES보다 key 값이 상대적으로 많지만, round 부분이 덜 복잡하다.

64비트의 입력 값을 32비트씩 둘로 나누어 왼쪽(L_{i-1})과 오른쪽(R_{i-1})에 제공한다. 오른쪽 32비트 R_{i-1}과 256비트의 key값 중 32비트를 이용한 덧셈연산을 수행한다. 덧셈연산 결과 중 carry는 무시하고 다음 과정에 32비트를 제공한다. 앞으로 carry를 무시한 덧셈 연산과정을 Modulo 2^{32} Adder라 부른다. 또 256비트의 key값에서 32비트를 선정하는 과정을 Subkey Generator라 칭한다. Modulo 2^{32} Adder과정을 거친 다음 불규칙적인 배치를 하는 S-Box Substitution이라는 과정을 거친 후 32비트의 값을 왼쪽으로 회전시키는 Left Circulate Shift과정을 거친다. 최종적으로 shift후 결과 값과 L_{i-1}과 xor 과정을 실행하여 R_i에 저장하고 L_i에는 R_{i-1}값을 저장한다.

GOST알고리즘의 전체 구성도는 <그림 5>과 같다. 각 round에서 수행되는 연산에 대해 자세히 설명하면 다음과 같다.



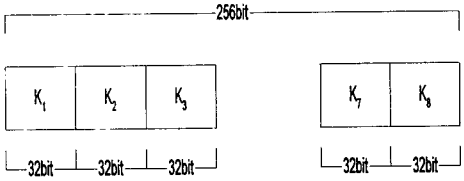
<그림 4> GOST 알고리즘의 round



<그림 5> GOST 전체 블록도

1) Subkey Generator

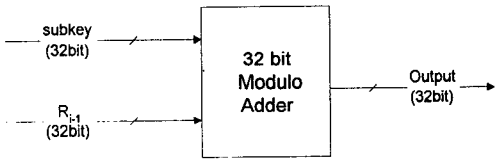
256비트의 입력된 key값을 이용해서 각 round 마다 32비트의 subkey를 공급해주는 역할을 하는 부분이다. 일단 256 비트를 <그림 6>처럼 8개의 32비트로 나눈 뒤에 각각 차례대로 K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8에 저장한다. 미리 저장된 key값을 각 round마다 차례대로 공급해 준다. 복호화의 경우에는 각 round에 공급된 key의 역 값을 제공해 주면 된다.



<그림 6> 256 비트 key 값의 배분

2) Modulo 2^{32} Adder

<그림 7>과 같이 32비트의 subkey 값과 32비트의 블록 값을 입력받아 덧셈 연산을 수행한 후 carry는 무시하고 sum의 32비트 값만을 출력한다. 즉, 덧셈 연산 후 2^{32} 값으로 나누어 나머지 값만 출력한다.



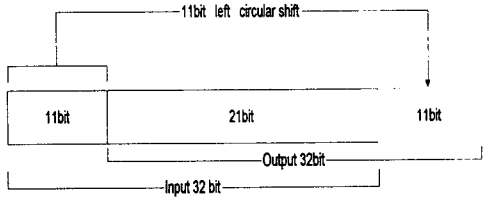
<그림 7> Modulo 2^{32} adder

3) S-box Substitution

여기서는 4비트의 입력 값을 받아서 4비트를 출력하고 입력된 데이터 값을 불규칙적으로 흩어지게 하는 역할을 하며 총 8개의 S-box로 구성되어 있다. S-box 각각의 내용은 알고리즘에 의해서 미리 정의된다.

4) Left Circular Shift

DES 알고리즘의 P-box 역할을 하는 것으로서 <그림 8>과 같이 32비트의 입력 중에서 11비트 왼쪽으로 rotation 하여 32비트 값을 출력한다. 이것은 8개의 S-box로부터 받은 값을 재배열하는 역할을 한다



<그림 8> 11-bit left circular shift

2.3 GOST알고리즘의 비교

DES와 GOST 알고리듬의 기본적인 구조는 다음과 같은 차이점을 가지고 있다.

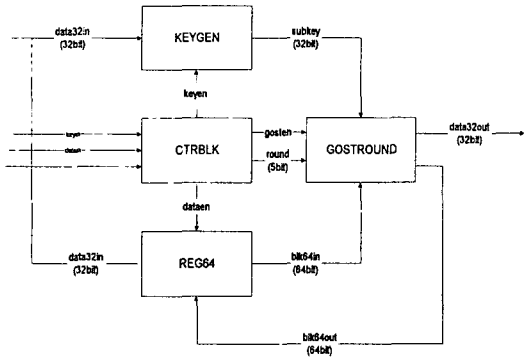
(1) DES는 입력된 key 값에서 subkey를 생성하는데 복잡한 과정을 통해 얻어지는 반면 GOST는 간단한 과정을 통해 생성된다.
(2) DES는 56비트의 key 값을 가진 반면 GOST는 256비트의 key 값을 가진다.
(3) DES에서 S-box는 6비트를 입력하여 4비트를 출력하는데 비하여 GOST의 S-box는 4비트를 입력받아 4비트를 출력한다.
(4) DES는 P-box라 불리는 불규칙적 permuta-tion 과정을 거치지만 GOST는 11비트 왼쪽으로 rotation하는 과정이 있다.
(5) DES는 16 개의 round를 거치지만 GOST는 총 32 round를 통해서 결과 값이 나온다.

3. 일반화된 GOST 칩

여기서는 앞에서 설명한 GOST 알고리즘을 각 블록으로 나누어 칩으로 설계한다. 또 GDES의 구조를 이용하여 기본적인 GOST구조를 확장하여 한번에 처리할 수 있는 데이터 양을 증가시킨 구조를 구현한다.

3.1 기본적인 GOST 구조

GOST 칩은 <그림 9>처럼 크게 CTRBLK 블록, REG64 블록, KEYGEN 블록, GOST ROUND 블록으로 4개로 나눌 수 있다. 칩 외부



<그림 9> GOST 칩의 구성

로부터 클럭을 입력해 주는 clock(1비트), 암호화와 복호화를 알리는 EnDec(1비트), 칩 전체를 초기화하는 reset(1비트), key값과 데이터 값을 입력받는 data32in(32비트), 64비트의 값이 key값임을 알리는 keyin(1비트), 64 비트의 값이 데이터 값임을 알리는 datain(1비트)의 입력 값과 최종적으로 암호화 또는 복호화 값을 내보내는 data 32out(32비트)의 출력 값 그리고 데이터의 출력을 알리는 dataout(1비트)을 포함해서 총 70개의 핀으로 구성되어있다. 64비트의 데이터 값은 data32in으로부터 2번에 나누어 입력받고 256비트의 key값은 data32in으로부터 8번에 나누어 입력받는다. 또 암호화/복호화 된 출력 값은 data32out으로 2번에 나누어 출력된다. 각 블록에 대한 자세한 기능은 다음과 같다.

1) CTRBLK 블록

CTRBLK은 KEYGEN 와 REG64 블록에 데이터를 저장할 것인지 읽어올 것인지를 여부를 제어하고 매 round마다 GOSTROUND에 제어신호를 제공하는 역할을 한다.

2) REG64 블록

REG64 블록은 외부로부터 읽어들인 64비트의 데이터 값을 저장하고 매 round마다 값을 GOSTROUND 블록에 데이터 값을 공급해주고 매 round를 거쳐 나온 값을 저장한다.

3) KEYGEN 블록

외부로부터 읽어들인 256비트의 key값을 저장하고 매 round마다 32비트의 key값을 공급한다.

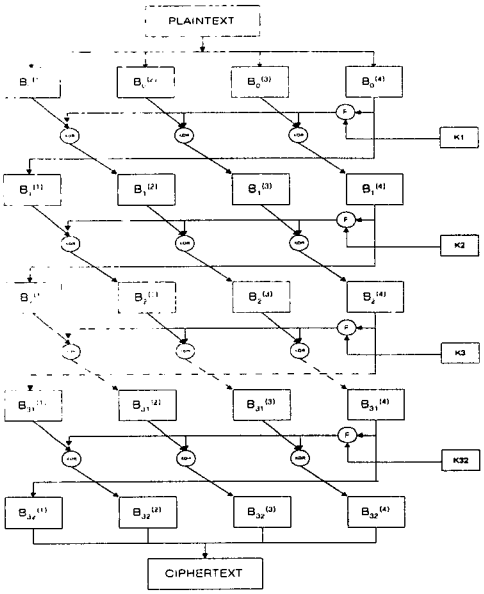
즉, keyin 값이 enable인 경우는 32 비트씩 8번에 나누어 key값을 저장하고, gosten 값이 enable 인 경우에는 CTRBLK에서 round<4:0>값을 받아 각 round에 해당하는 값을 전송한다

4) GOSTROUND 블록

이 블록은 GOST 암호화 칩의 핵심으로 실질적인 암호화 과정의 부분이다. 64비트의 데이터 값 blk64in<63:0>을 REG64로부터 받고 KEYGEN부터 32비트의 subkey 값 subkey<31:0>을 입력받아 암호화를 한다. 또 CTRBLK 블록의 rd 신호를 이용하여 새로운 값을 읽어오고 결과 값을 REG64에 보낸다.

3.2 일반화시킨 GOST 구조

<그림 10>은 앞에서 구현한 구조를 기초로 하여 한번에 처리할 수 있는 데이터 처리 양을 두 배로 증가시킨 구조이다. 즉 입력 데이터 값을 64비트에서 128비트로 증가시켜 한번에 처리할 수 있는 능력을 두 배로 향상하였다. 단순히 GOST를 나란히 두 번 배열하는 경우는 GOST ROUND 부분이 2개가 필요하지만 여기에서는 GOSTROUND 부분을 하나로 구성하고 하나의 32 비트 블록에만 적용하였다. 즉, 128비트 값을 32비트씩 네 개로 나누어 가장 우측에 있는 32비트만을 F(Modulo 2^{32} Adder -> S-Box -> Left Circular Shift)연산 과정에 적용시킨 형태이다. 이 과정에 의해 나온 결과 값을 각각의 나머지 32비트 블록에 XOR 연산하여 하나의 round를



<그림 10> 128비트 입력으로 일반화 시킨 GOST 구조

처리한다. 이런 방법을 이용해서 계속적으로 입력 데이터를 증가시키면 처리량을 증가되지만 32비트 블록에 적용되는 GOSTROUND 회수가 적어지므로 상대적으로 데이터를 불규칙적으로 흩어지게 하는 round 과정이 줄어 암호화가 약해지기도 한다. 이것을 이용하면 필요에 따라서 빨리 처리하길 원하는 경우에 활용할 수 있다.

4. 합성 및 시뮬레이션

여기서는 일반화된 GOST알고리즘 구조를 VHDL로 기술하고 Synopsys Analyzer를 이용하여 합성한 후 최종적으로 Altera사의 설계 도구인 Max+plus II를 이용하여 시뮬레이션을 하여 검증하였다.

4.1 합성결과

Samsung KG75000 SOG cell library를 이용하여 합성하였다. 합성결과 <표 1>과 같은 결과 값을 얻을 수 있었다. 전체적으로 볼 때 KEYGEN 부분이 256 비트의 저장공간이 필요하여 가장 많은 면적을 차지함을 볼 수 있으며 그리고 다음으로 핵심인 GOSTROUND 부분이 가장 많은 면적을 차지함을 볼 수 있다.

<표 1> 합성된 GOST 칩의 결과 값

(단위 : nand = 1)

	Cell Area	Combination	Non-combination
CTRBLK	165	88	77
GOSTROUND	1603	803	800
KEYGEN	2415	1135	1280
REG128	1376	736	640
Total	5626	2765	2861

4.2 Round블록의 비교

DES와 GOST의 암호화 핵심부분인 round 블록부분을 합성한 것을 비교하면 <표 2>와 같다. 한 개의 round를 구성하는데 면적 측면에서 본다면 GOST(64비트 입력)가 DES보다 50% 정도 적게 사용됨을 볼 수 있다. 또 128 비트 입력인 경우 면적이 64비트를 두개 사용한 것의 75% 정도만 사용했다. 지연적 측면에서 GOST(128비트)의 지연과 GOST(64비트)를 비교해 보면 거의 차이가 없음을 알 수 있다. 즉 속도 지연 없이 한번에 처리할 수 있는 양을 증가시킬 수 있음을 볼 수 있다.

<표 2> DES와 GOST의 round 비교

	DES	GOST (64비트)	GOST (128비트)
cell Area (nand = 1)	2084	1080	1610
data arrival time (ns)	9.07	19.95	20.28

또 round 부분을 software로 구현했을 때 지연시간을 비교한 것은 <표 3>과 같다.

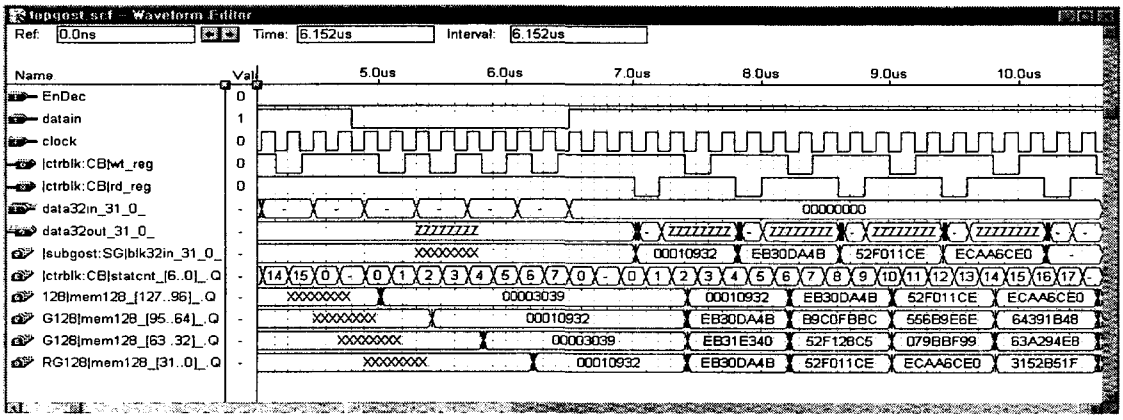
<표 3> DES와 GOST의 software 구현비교 (단위 : μs)

	DES	GOST(64비트)	GOST (128비트)
지연 시간	15	6.875	11.85

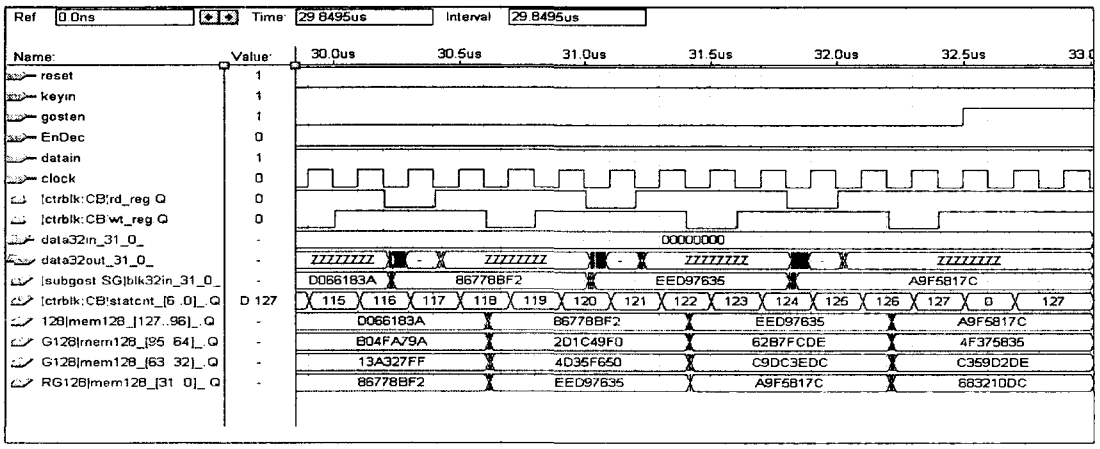
<표 2>와 <표 3>를 단순히 지연시간만을 비교했을 때 GOST(64비트)는 하드웨어로 구현했을 때 350배 정도 빠르게 동작함을 알 수 있었고, DES는 1500배 이상 빠르게 동작함을 볼 수 있었다.

4.3 시뮬레이션 결과

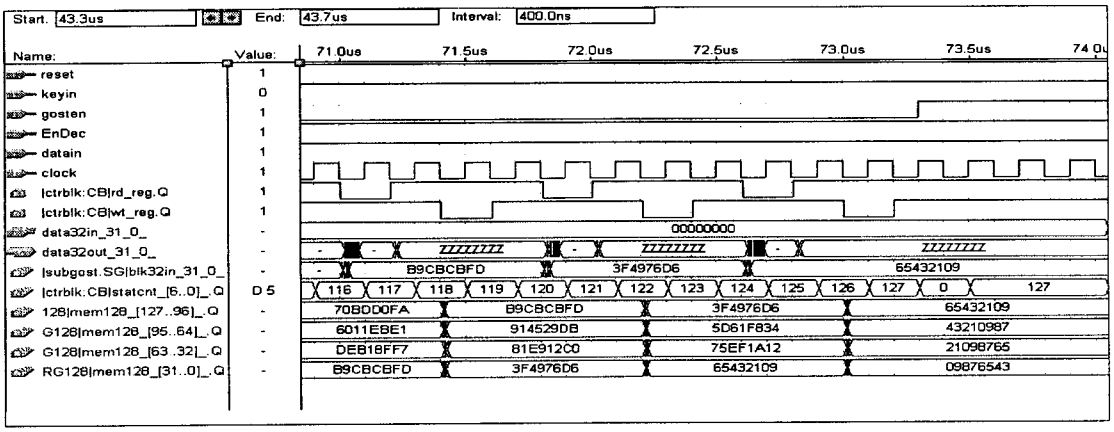
본 논문에서 제시한 구조를 <그림 11>와 같이 256비트의 key(12345678 90123456 78901234 56789012 34567890 12345678 90123456 78901234 : HEX)와 128비트의 데이터(09876543 21098765 43210987 65432109 : HEX)를 입력하면 <그림 12>과 같이 128비트의 출력값 데이터(683210DC



<그림 11> 설계된 GOST 칩의 입력



<그림 12> 암호화된 결과



<그림 13> 복호화 결과

C359D2DE 4F375835 A9F5817C : HEX)값을 얻을 수 있었다. 이것을 다시 입력하여 복호화 하면 <그림 13>과 같이 암호화 이전에 입력된 값 (09876543 21098765 43210987 65432109 :HEX)을 얻어 제시된 구조가 정확하게 수행되고 있음을 확인할 수 있었다.

5. 구 현

본 논문에서 제시한 구조를 Altera사의 CPLD 칩을 사용하여 구현하였다. <그림 14>는 실제로 구현을 위해서 사용한 CPLD 칩을 나타내었다.

<그림 14> 구현한 CPLD 칩

6. 결 론

암호화에 대한 연구가 계속되는 것에 비례하여 암호를 불법적으로 해독하려는 기술 또한 발전하였다. 해독하려는 데이터가 시간과 비용을 투자하여 그 이상의 가치를 얻을 수 있는가 하는 것이 관건이다. 앞에서 언급한 비밀 key 암호화 방식으로는 DES 알고리즘 그리고 공개 key 암호화 알고리즘으로는 RSA 알고리즘을 가장 널리 사용되고 있다[4,5]. DES 알고리즘은 보안성 면에서는 어느 정도 인정받고 있지만 앞으로 계속 사용될 경우 key 크기 면에서 불안한 요소가 있다.

본 논문에서는 key의 크기가 DES사용되는 것보다 4배가 큰 GOST 알고리즘을 이용하여 칩을 구성하여 보안 적인 면을 강화하였다. 또 기본 GOST 알고리즘을 개선하여 한번에 처리할 수 있는 데이터 양을 두 배(128 비트)로 처리할 수 있는 구조를 제시하고 Altera사의 설계 도구인 MAX+Plus II를 이용하여 Flex 8000 시리즈 칩에 적용하였을 때 기본 GOST 알고리즘과 비교 분석하였다.

구현 결과에 따르면 DES보다 round 측면에서 칩 크기를 반정도로 줄일 수 있어 GOST로 칩을 구현하면 크기면 에서 효과를 볼 수 있었다. 또 확장된 구조가 기본 GOST 구조와 지연

측면에 차이 없이 한번에 처리할 수 있는 양을 두 배로 증가시킬 수 있었음을 알 수 있었다. 또 이 구조를 이용하면 처리 능력을 같이 했을 때 확장된 구조가 기본 구조보다 40% 정도 줄일 수 있었다.

앞으로의 연구방향은 기존의 GOST 알고리듬의 각 round에서 사용된 S-box의 table 값을 보다 불규칙적으로 유도하고 공개 key 암호화 알고리즘인 RSA 알고리즘과 연계하여 공개 key 암호화 알고리즘과 비밀 key 암호화 알고리즘을 하나의 칩에 효율적으로 구현하는 것에 대해 연구하고자 한다. 또한 암호화 칩의 다양한 이용에 대해서도 연구하고자 한다.

참고문헌

[1] Altera, *Max+plus II getting started*, Altera corporation, 1995.

[2] Altera, *Data Book*, Altera corporation, 1996.

[3] Artech House Boston · London, *Smart Cards*, Artech House Inc.

[4] Bruce Schneier, *Applied Cryptography*, John Wiley & Sons Inc, 1996.

[5] Man Young Rhee, *Cryptography and Secure Communications*, McGraw-Hill, 1994.

[6] GOST, Gosudarstvennyi Standard 28147-89, *Cryptographic Protection for Data Processing Systems*, Government Committee of the USSR for Standards 1989.

[7] Schaumuller-Bichl, *"On the Design and Analysis of New Cipher System Related to the DES"*, Technical Report, Linz University, 1983.

[8] Warwick Ford, *Computer Communications Security*, Prentice-Hall International Inc. 1994.

[9] 류승석 · 정연모, "DES 알고리즘을 이용한 암호화 칩 설계", 경희대학교 산학협력기술연구논문집, 제2집, pp195-199. 1996.

[10] 류승석 · 오재곤 · 정연모, "GOST 알고리즘을 이용한 암호화 칩 설계에 관한 연구", 대한전자공학회 CAD 및 VLSI 설계 연구회 학술발표회 논문집, pp 49-54, 1997.

[11] 류승석, 암호화 칩의 설계, 경희대학교 대학원 석사학위논문, 1997.

[12] 원치선 "Digital TV에서의 데이타 보호 및 응용".

저자소개

류승석
1995 경희대 전자공학과 학사
1997 경희대 전자공학과 석사
현재 삼성전자 LSI 개발팀 주임연구원
관심분야: DES, RSA coprocessor, Smart Card

정연모
1980 경북대학교 이학사
1982 KAIST 공학석사
1982~미국 미시간주립대학교 공학박사
현재 경희대학교 전자정보학부 부교수
관심분야: CAD 및 VLSI, 암호화 알고리듬, 인터넷 응용