Pipeline 구조의 SEED 암호화 프로세서 구현 및 설계

채 봉 수 ,*김 기 용, 조 용 범
건국대학교 전자공학과, *뮤즈텔

Hardware Implementation for SEED Cipher Processor of Pipeline Architecture

Bong-Soo Chae, Ki-Yong Kim, Yong-Beom Cho
Dept. of Electronics, Konkuk University
E-mail : cherry@konkuk.ac.kr

Abstract

This paper designed a cipher process, which used SEED-Algorithm that is totally domestic technique. This cipher processor is implemented by using SEED-cipher-Algorithm and pipeline scheduling architecture. The cipher is 16-round Feistel architecture but we show just 8-round Feistel architecture for brevity in this thesis. Of course, we can get the result of the 16-round processing by addition of control part simply. Furthermore, it has pipelined architecture, so the speed of cipher process is the faster than others when we performed a cipher a lot of data. The schedule-function can performed the two-cipher process simultaneously, such as using two-cipher processors.

I. 서론

정보의 전송에 있어서 중요한 요소 중의 하나가 정보의 보안성이다. 보안성을 유지하기 위해서는 여러 가지 방법이 사용되고 있으나, 가장 보편적인 방법은 전송하려는 정보에 암호화를 취해, 암호화된 정보를 전송하는 것이다.[1] 암호화란 데이터를 전혀 새로운 값으로 바꾸어 보냄으로써 그 해석을 임의로 할 수 없도록 하는 것을 말한다. 암호화를 소프트웨어로 구현하면, 실행 속도가 느리기 때문에 하드웨어로 구현해 왔으며, 이미 수많은 암호화 알고리즘들이 하드웨어로 구현되어 왔다. 여태까지 구현된 암호화 알고리즘의 대부분이 모두 외국에서 만들어진 암호화 알고리즘이었으니, SEED 암호화 알고리즘은 우리 나라에서 만들어진 암호화 알고리즘이다.[2][3]

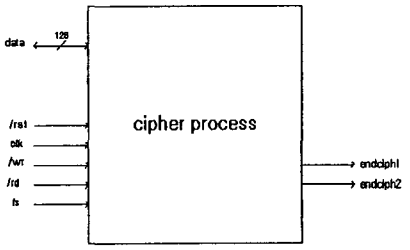
본 논문에서는 이 SEED 암호화 알고리즘을 이용하여 암호화 프로세서를 구현하였다. 암호화의 수행에는 16 라운드가 필요하나, 본 논문에서는 4 라운드 씩 4개의 구조로 나누고, 나눈 구조들을 pipeline 구조로 구성하여 전체 구조를 단순화하였을 뿐만 아니라 처리 속도도 향상시켰다.[4][5]

II. Pipeline구조의 cipher processor

암호화 프로세서는 처음에 모두 4개의 128bit 데이터를 입력 받는다. 그 값은 암호화키값 2개와 두개의 데이터를 입력 받는데, 동일한 데이터를 사용하여, 서로 다른 암호화 키를 가지고, 동시에 2개의 암호화가 진행되는 것을 보이기 위해서 두 번 입력되는 데이터는 동일 한 값을 입력하기로 한다. /wr신호에 의해서 이 값들이 입력을 받게 되며, /wr신호가 disable되면 바로 암호화를 시작하게 된다. 첫번째 암호화 키를 key1, 두 번째 암호화 키를 key2, 첫번째 데이터를 data1, 두 번째 데이터를 data2라 하자. 암호화과정은 먼저, key1과 data1에 의해서 암호화가 진행 된다. 암호화가 직접 진행되는 부분은 round부이다. 이곳에 데이터가 입력되고, 각 라운드에 입력되는 라운드 키를 생성하는 key generator에 암호화 키가 입력된다. 다음 클럭에서 key1과 data1에 의해서 암호화가 끝난 암호화 데이터가 저장됨과 동시에 key2와 data2가

round와 key generator에 입력된다. 첫번째 암호화가 끝난 데이터는 암호화 과정 중 4라운드를 거친 결과이다. 이 결과 데이터는 다시 schedule 함수내의 레지스터에 저장된다. Key2와 data2에 의해서 암호화가 끝난 암호화 데이터가 저장될 때 동시에 첫번째 4라운드 과정을 끝낸 암호화 데이터가 입력된다. 이제 이 과정에서 첫번째 암호화가 완전히 끝났다는 endciph1신호를 내보내면서 첫번째 암호화가 끝이 나고, 바로 뒤이어서 두 번째 암호화가 시작되고, 마찬가지고, 두 번째 암호화가 끝나면서 endciph2신호가 출력되면서, 한 개의 데이터로 두개의 암호화를 진행하는 과정이 끝이 난다. 이 과정을 반복하면서 암호화 프로세서로 동작을 하는 것이다.

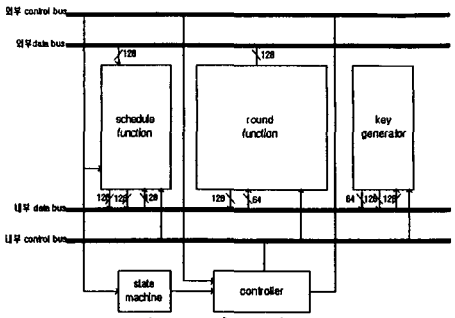
2.1 Cipher processor architecture [6][7]



[그림 1] Cipher process 블록도

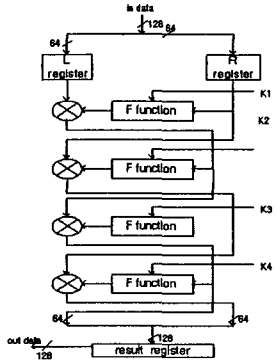
입출력 핀으로 128bit의 data가 존재하고, 입출력 제어 신호가 존재한다. 여기서 fs라는 제어신호는 일종의 tail flag로써 지금 입력하고 있는 것이 key1인지 key2인지 data1인지 data2인지를 구별해 주는 제어신호이다.

내부 블록은 크게 3부분으로 나누어져 있다. Schedule function, round function, key generator로 나눌 수 있다. 우선 key generator부는 암호화 키를 입력 받아서 각각의 라운드에 해당하는 라운드 키를 생성한다. 이 라운드 키는 라운드부에 입력되고, 라운드부는 라운드 키와 입력되는 데이터를 사용하여, 암호화를 실제로 진행한다. Schedule부는 단순히 암호화될 데이터를 출력하는 부분인데, 상황에 따라 암호화될 데이터를 출력하여, 2개의 암호화가 동시에 진행되도록 해준다.



[그림 2] 내부 블록도

2.2 Round Block

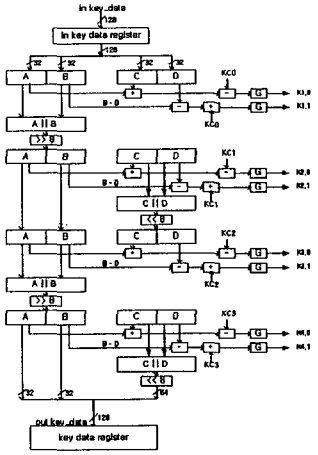


[그림 2] Round 블록

파이프라인 구현 시 나타나는 구조적 해저드를 해결하기위해 입출력 단에 데이터를 저장하는 레지스터를 두었다. 모두 4라운드로 이루어져 있다.

2.3 Key generator

[그림 4]는 key generator부분이다. 4개의 라운드 키를 생성하는데, 확장된 라운드 키를 생성하기 위해서 각각의 라운드에 들어가는 라운드 상수는 따로 제어신호에 의해서 다른 값을 입력하게 된다.



[그림 3] Key generator

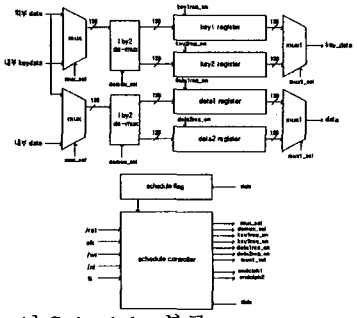
2.4 Schedule Block

[그림 5]Schedule은 전체 프로세서 과정과는 다르게 자체적으로 controller를 가지고 있으며, 현재의 진행 상태에 따라서 데이터를 출력해야 하므로, 현재의 상태를 나타내어 주는 schedule flag를 가지고 있다.

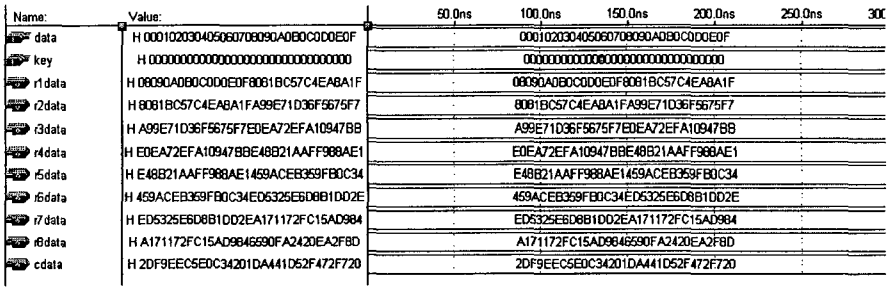
III. 시뮬레이션

암호화의 정확한 동작을 보이기 위해 SEED 알고리즘에 근거한 functional 한 암호화 프로세서를 설계하여, 그 결과값을 비교하였다. [그림 6]과 [그림 7]은 키 값 128 bit 로 "00000000000000000000000000000000"을 넣었을 때의 결과를 보여주는 시뮬레이션이다. 입력 데이터 입력값 "000102030405060708090A0B0C0D0E0F"을 넣었을 때 결과값이 [그림 6]과 [그림 7] 의 'r8data' 와 'outdata' 에서 볼 수 있듯이 "A171172fc15AD9846590FA2420EA2F8D" 값을 가져 두 시뮬레이션의 결과가 일치함을 볼 수 있다.

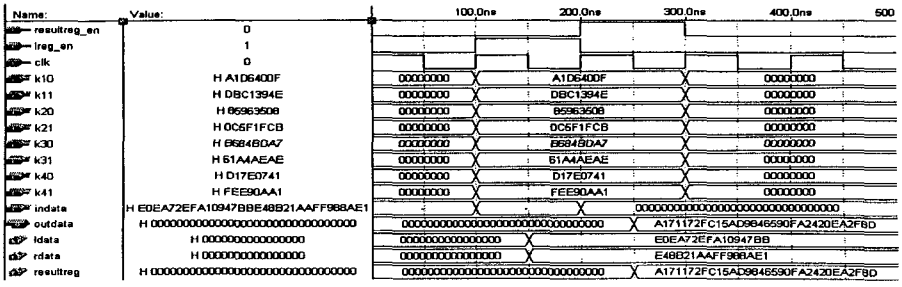
[그림 8]에 전체 시뮬레이션을 보인다. 두 개의 키값과 한 개의 데이터를 입력 했을 때 첫번째 데이터가 암호화 된 뒤 바로 두 번째 암호화가 끝나는 것을 볼 수 있다.



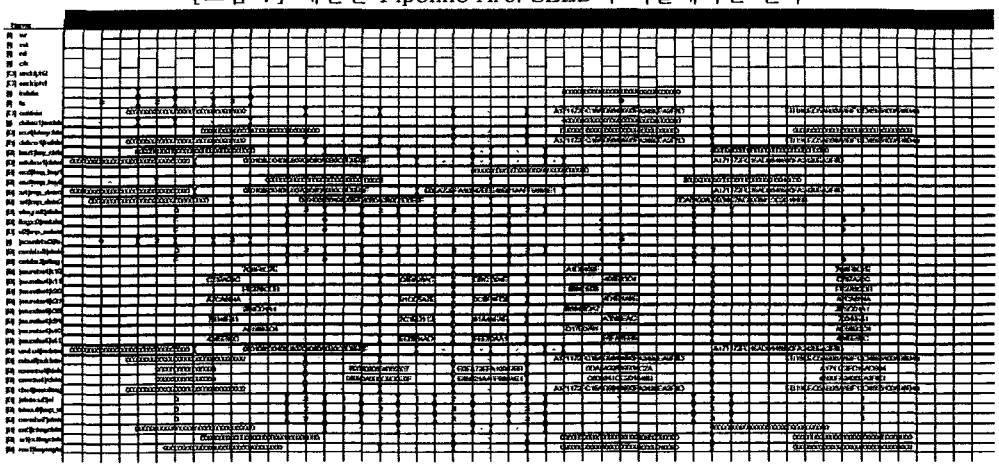
[그림 4] Schedule 블록



[그림 6] functional 한 SEED 알고리즘 구현 하드웨어



[그림 7] 제안한 Pipeline Arc. SEED의 시뮬레이션 결과



[그림 8] 전체 시뮬레이션

IV. 결론

본 논문에서 제시한 암호화 프로세서는 크게 두 가지 장점을 가지고 있다. 첫째로 파이프라인 구조를 사용하여, 매 클럭 마다 암호화가 이루어 질 수 있어, 빠른 수행속도를 나타내고, Round 과정을 4단계로 줄이면서 칩의 면적을 1/4 수준으로 줄일 수 있다. 두 번째로, 특별한 구조인 schedule이라는 블록을 두어서 입력되는 데이터를 schedule해 줌으로써 거의 동시에 2개의 암호화가 진행되어, 2개의 암호화 프로세서를 사용한 것과 같은 효과를 볼 수 있었다. 이러한 구조를 사용하면, 3개,4개의 암호화를 동시에 수행할 수 있으며, 일대 다의 멀티 통신에 있어서 매우 하게 사용 될 수 있다.

참고문헌

[1] Martin P.J. Kratz, *Information systems security*, Van Nostrand Reinhold, 1993

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

[3] 박종서외, *"암호화 알고리즘 소개 : SET을 기반으로"*, 한국통신학회지, 1999.11

[4] Kai Hwang, *Advanced Computer Architecture*, McGraw-Hill, 1996

[5] David A. Patterson, John L. Hennessy, *Computer Organization and Design : The Hardware/Software Interface*, Morgan Kaufmann Publishers, 1998

[6] 한국정보통신기술협회, *128-bits 블록 암호화 알고리즘(SEED)*, 한국정보통신기술협회, 1999. 6

[7] Behrooz Parhami, *Computer Arithmetic : Algorithms and Hardware Designs*, Oxford University Press, 2000