通信情報保護學會論文誌
第 11 券 第 1 號, 2001. 2

IDEA 알고리즘을 이용한 고속 암호 VLSI 설계

이 행 우*, 최 광 진**

A Design of the High-Speed Cipher VLSI Using IDEA Algorithm

Haeng-Woo Lee*, Kwang-Jin Choi**

요 약

본 논문은 IDEA 알고리즘을 사용한 고속 암호 IC의 설계에 관한 것이다. IDEA 알고리즘을 회로로 구현하기 위하여 전체 회로를 6개의 주요 기능블럭으로 분할하여 설계하였다. 주요 블록으로 암호키 및 복호키 생성부, 입력 데이터 처리부, 암호화 처리부, 출력 데이터 처리부, 그리고 동작모드 제어부 등이 있다. 서브키 생성회로는 연산속도보다 회로면적을 축소시키는 방향으로 설계한 반면, 암호화 처리부는 회로면적보다 연산속도를 증가시키는 방향으로 설계목표를 정했다. 따라서 반복연산에 적합한 파이프라인 구조와 연산속도를 향상시키는 모듈라 승산기를 채택하였다. 특히, 많은 연산시간이 소요되는 모듈라 승산기는 연산속도를 증가시키기 위하여 캐리선택 가산기 및 modified Booth 승산 알고리즘을 사용하여 한 클럭에 동작하도록 설계하였다. 또한, 입력 데이터 처리부는 데이터를 동작모드에 따라 8-bit, 16-bit, 32-bit 단위로 받아들이기 위하여 데이터 버퍼가 8-bit, 16-bit, 32-bit 씩 이동할 수 있도록 하였다. 0.25 μm 공정기술을 사용하여 시뮬레이션한 결과, 이 IC는 큰 면적을 요구하지 않으면서도 1Gbps 이상의 throughput을 달성하였으며, 회로구현에 약 12,000gates가 소요되었다.

ABSTRACT

This paper is on a design of the high-speed cipher IC using IDEA algorithm. The chip is consists of six functional blocks. The principal blocks are encryption and decryption key generator, input data circuit, encryption processor, output data circuit, operation mode controller. In subkey generator, the design goal is rather decrease of its area than increase of its computation speed. On the other hand, the design of encryption processor is focused on rather increase of its computation speed than decrease of its area. Therefore, the pipeline architecture for repeated processing and the modular multiplier for improving computation speed are adopted. Specially, there are used the carry select adder and modified Booth algorithm to increase its computation speed at modular multiplier. To input the data by 8-bit, 16-bit, 32-bit according to the operation mode, it is designed so that input buffer shifts by 8-bit, 16-bit, 32-bit. As a result of simulation by 0.25 μm process, this IC has achieved the throughput of 1Gbps in addition to its small area, and used 12,000gates in implementing the algorithm.

keyword : *IDEA algorithm, VLSI design*

Ⅰ. 서 론

오늘날 정보통신기술은 하루가 다르게 눈부신 속도로 발달하고 있다. 현대사회에 있어서 인터넷 등 정보통신의 이용은 필수적인 일이며 사용자들도 급격하게 확산되는 추세이다. 정보통신의 대중화와 함께 대두되는 문제는 정보의 보안성 확보이다. 안전하게 정보를 주고 받을 수 있느냐하는 문제는 통신에 있어서 가장 기본적으로 해결해야 할 사안이다.

암호화 기술을 이용하여 통신의 보안문제를 해결

* 남서울대학교 전자정보통신공학부(hwlee@nsu.ac.kr)
** (주)맥텔레콤 대표이사(mactel@mactel.co.kr)

하는 방법은 오래 전부터 아날로그 통신에서 사용되어 왔으나 오늘날 일반화되고 있는 디지털 통신에 있어서는 아직 대부분 사용되고 있지 않다. 현재 제안되어 있는 암호 알고리즘 가운데 대칭키 알고리즘으로 가장 널리 사용되고 있고 암호화 특성도 우수하다고 알려져 있는 IDEA 알고리즘이 있다.

IDEA 암호 알고리즘의 개념은 1990년 Xuejia Lai와 James Massey에 의해 PES(Proposed Encryption Standard)란 이름으로 발표[1]되었고, 1991년 각종 공격에 대한 방어능력을 강화한 Improved PES로 개선된 후 1992년 다시 IDEA로 개명되었다. 이 알고리즘은 강력한 이론적 토대를 갖고 있으며 현재 공개된 대칭키 알고리즘 가운데 가장 안전한 것으로 알려져 있다. IDEA 알고리즘은 DES 알고리즘과 함께 널리 사용되고 있는 블록 알고리즘이다. DES 알고리즘이 하드웨어 설계가 용이하도록 개발된 알고리즘이라면 IDEA 알고리즘은 프로그램 구현이 용이하도록 개발된 알고리즘이다. 그동안 이 알고리즘은 특성상 주로 소프트웨어를 이용하여 구현하였으나 통신의 고속화에 따른 처리속도 증가를 충분히 지원할 수 없게 되었다. 따라서 처리속도를 증대시키고자 하드웨어 구현[2,3]을 하게 되는데, 본 연구에서는 연산시간의 대부분을 차지하는 암호화 처리부의 효율적인 설계를 통해 IC 면적은 작으면서 데이터 처리능력을 강화시키는 방안을 모색하였다.

본 논문의 구성은 Ⅱ장에서 IDEA 알고리즘에 대해서 간단히 알아보고, Ⅲ장에서는 각 기능블럭의 회로설계 내용, Ⅳ장에서는 설계한 회로의 모의실험 및 검토, 그리고 Ⅴ장에서 결론을 맺었다.

Ⅱ. IDEA 알고리즘

IDEA 알고리즘은 DES 알고리즘과 마찬가지로 혼동성과 확산성을 주요 목적으로 하고 있지만 비트 단위의 대치나 치환보다는 주로 16비트 단위의 모듈러 연산을 사용하여 암호화의 비도를 증가시키는 것이 주요 특징이다. 본 알고리즘은 입력 데이터를 64비트 단위로 블록화하고 각 라운드마다 96비트 생성키를 사용하여 8라운드 동안 반복적으로 암호화를 수행한다. 그리고 복호화 과정 역시 복화키만 다른 값을 사용할 뿐 암호화 과정과 동일한 연산에 의해 수행된다. 원래 키는 128비트를 입력으로부터 받아들여 키스케줄링에 따라 각 라운드에 대한 96비트 길이의 암호키와 복호키를 생성한다. 본 알고리즘을 수행하기 위하여 필요한 연산에는 다음과 같은 것들이 있다.

- Exclusive OR
- Modulo-2^{16} 가산
- Modulo-2^{16} 가산역
- Modulo-$(2^{16}+1)$ 승산
- Modulo-$(2^{16}+1)$ 승산역
- 제산

이와 같은 연산들은 16비트 데이터를 갖고 수행하기 때문에 16비트 넓이의 프로세서에 적합한 알고리즘이다. 암호화는 이 연산들을 적절하게 조합하여 수행함으로서 데이터를 복잡하게 혼합하는 것이다.

2.1 서브키 생성방법

입력 키워드는 128비트로서 16비트 단위의 서브키 8개를 생성한다. 64비트 데이터를 암호화 또는 복호화하기 위하여 필요한 서브키는 총 52개이며 한 라운드 당 6개의 서브키를 사용한다. 먼저 암호용 서브키 생성방법을 알아보자. 일단 128비트 키워드를 사용하여 8개의 서브키를 생성하고 키워드를 좌측으로 25비트 순환이동시킨다. 그리고나서 다시 동일한 방법으로 연속해서 차례대로 사용될 16비트 단위의 8개 서브키를 생성한다. 이러한 방법으로 6회 순환이동하면서 서브키를 생성하면 52개의 서브키를 모두 얻을 수가 있다. 복호용 서브키는 생성된 암호용 서브키를 사용하여 얻어진다. 다만 암호용 서브키로부터 복호용 서브키를 생성하기 위해서는 암호용 서브키의 Modulo-2^{16} 가산역 또는 Modulo-$(2^{16}+1)$ 승산역을 필요로 한다. 이러한 연산을 사용하여 16비트 단위의 암호 서브키로부터 복호 서브키를 구하는 순서가 [표 1]에 표시되어 있다. 여기에서 복호

[표 1] IDEA 암호 및 복호 서브키

라운드	암호 서브키						복호 서브키					
1	K_{11}	K_{12}	K_{13}	K_{14}	K_{15}	K_{16}	K_{91}^{-1}	$-K_{92}$	$-K_{93}$	K_{94}^{-1}	K_{85}	K_{86}
2	K_{21}	K_{22}	K_{23}	K_{24}	K_{25}	K_{26}	K_{81}^{-1}	$-K_{83}$	$-K_{82}$	K_{84}^{-1}	K_{75}	K_{76}
3	K_{31}	K_{32}	K_{33}	K_{34}	K_{35}	K_{36}	K_{71}^{-1}	$-K_{73}$	$-K_{72}$	K_{74}^{-1}	K_{65}	K_{66}
4	K_{41}	K_{42}	K_{43}	K_{44}	K_{45}	K_{46}	K_{61}^{-1}	$-K_{63}$	$-K_{62}$	K_{64}^{-1}	K_{55}	K_{56}
5	K_{51}	K_{52}	K_{53}	K_{54}	K_{55}	K_{56}	K_{51}^{-1}	$-K_{53}$	$-K_{52}$	K_{54}^{-1}	K_{45}	K_{46}
6	K_{61}	K_{62}	K_{63}	K_{64}	K_{65}	K_{66}	K_{41}^{-1}	$-K_{43}$	$-K_{42}$	K_{44}^{-1}	K_{35}	K_{36}
7	K_{71}	K_{72}	K_{73}	K_{74}	K_{75}	K_{76}	K_{31}^{-1}	$-K_{33}$	$-K_{32}$	K_{34}^{-1}	K_{25}	K_{26}
8	K_{81}	K_{82}	K_{83}	K_{84}	K_{85}	K_{86}	K_{21}^{-1}	$-K_{23}$	$-K_{22}$	K_{24}^{-1}	K_{15}	K_{16}
9	K_{91}	K_{92}	K_{93}	K_{94}	*	*	K_{11}^{-1}	$-K_{12}$	$-K_{13}$	K_{14}^{-1}	*	*

서브키의 (-) 부호는 대응하는 암호 서브키의 가산 역을 나타내고 (-1)승 표현은 승산역을 나타낸다.

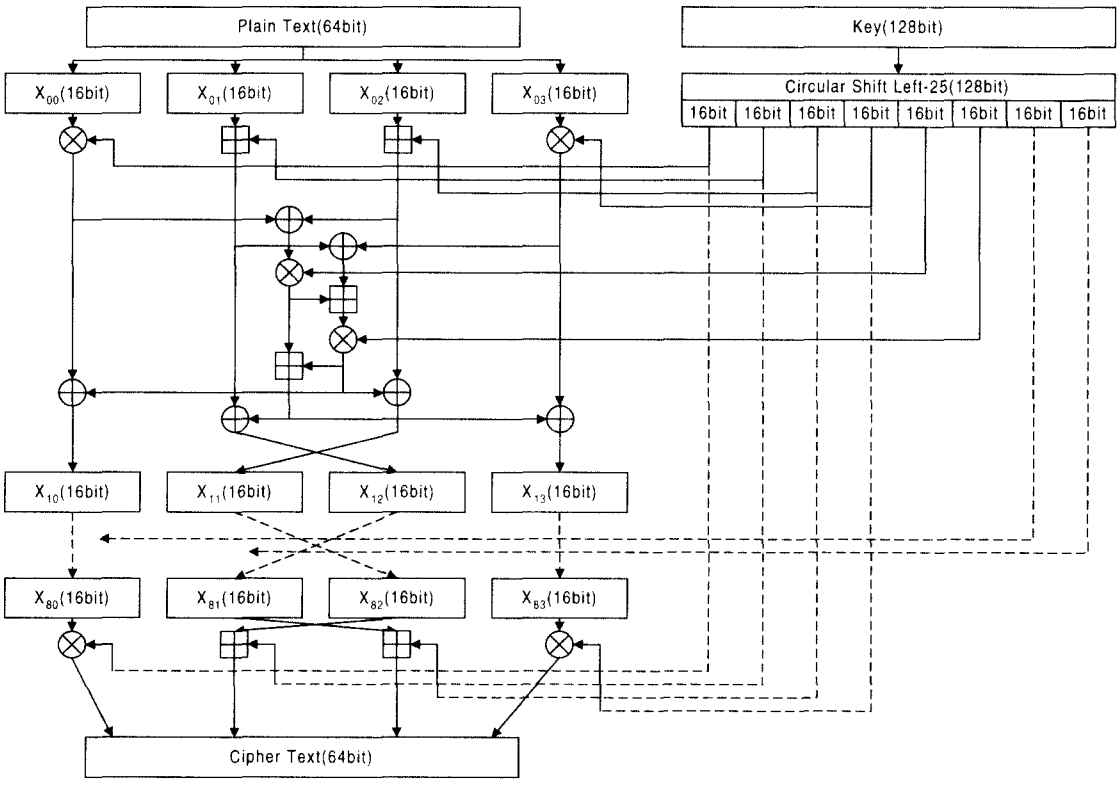
2.2 암호화 및 복호화 방법

64비트 데이터의 암호화와 복호화 과정은 사용하는 서브키만 다르고 동일한 하드웨어 구조를 사용하여 이루어진다. 일단 데이터는 16비트 단위의 4개 블록으로 분할된 다음 각 라운드마다 6개 서브키를 사용하여 정해진 과정에 따라 각 연산을 수행한다. 한 라운드를 마치게되면 다음 라운드의 서브키를 사용하여 동일한 연산과정을 반복하고, 이러한 동작을 8라운드 동안 반복하여 수행한다. 각 라운드에 대해 암호화에 필요한 연산과정은 다음과 같은 순서로 이루어진다.

① X0와 K1의 Modulo-$(2^{16}+1)$ 승산
② X1와 K2의 Modulo-2^{16} 가산
③ X2와 K3의 Modulo-2^{16} 가산
④ X3와 K4의 Modulo-$(2^{16}+1)$ 승산
⑤ 과정①과 ③의 결과를 Exclusive OR
⑥ 과정②와 ④의 결과를 Exclusive OR
⑦ 과정⑤의 결과와 K5의 Modulo-$(2^{16}+1)$ 승산
⑧ 과정⑥과 ⑦의 결과를 Modulo-2^{16} 가산
⑨ 과정⑧의 결과와 K6의 Modulo-$(2^{16}+1)$ 승산
⑩ 과정⑦과 ⑨의 결과를 Modulo-2^{16} 가산
⑪ 과정①과 ⑨의 결과를 Exclusive OR
⑫ 과정③과 ⑨의 결과를 Exclusive OR
⑬ 과정②와 ⑩의 결과를 Exclusive OR
⑭ 과정④와 ⑩의 결과를 Exclusive OR
⑮ 과정⑪의 결과를 X0 레지스터에 저장
⑯ 과정⑫의 결과를 X1 레지스터에 저장
⑰ 과정⑬의 결과를 X2 레지스터에 저장
⑱ 과정⑭의 결과를 X3 레지스터에 저장

이러한 과정을 8회에 걸쳐 반복한 다음 마지막으로 다음과 같은 연산을 수행한다

⑲ X0와 K1의 Modulo-$(2^{16}+1)$ 승산
⑳ X2와 K2의 Modulo-2^{16} 가산
㉑ X1와 K3의 Modulo-2^{16} 가산
㉒ X3와 K4의 Modulo-$(2^{16}+1)$ 승산



〔그림 1〕 IDEA 알고리즘 구조

이와 같은 연산을 수행한 다음, 그 결과로서 4개의 16비트 데이터를 연결하여 64비트 출력 데이터를 만듦으로서 암호화 과정을 완료한다. 복호화는 기본적으로 암호화 방법과 동일하게 이루어지는 바, 사용되는 서브키만 복호 서브키로 대체하고 암호화 회로와 동일한 회로를 사용하여 동일한 과정을 거치기 때문에 중복되는 설명은 생략하기로 한다.

2.3 Modular 연산법

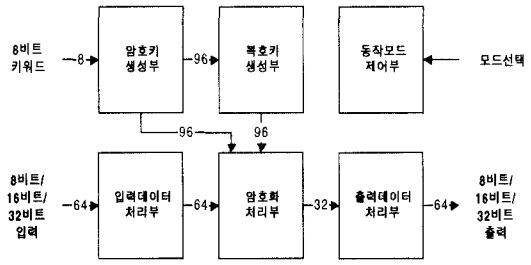
복호 서브키 생성에 사용되는 가산역 및 승산역 연산과 데이터 암호화 및 복호화에 사용되는 모듈라 가산 및 승산 연산방법은 다음 식과 같이 이루어진다.

- X, Y의 Modulo-2^{16} 가산
  $Z=(X+Y)\%\ 2^{16}$
- X, Y의 Modulo-$(2^{16}+1)$ 승산
  $Z=(X\times Y)\%\ (2^{16}+1)$
- X의 Modulo-2^{16} 가산역
  $Z=2^{16}-X$
- X의 Modulo-$(2^{16}+1)$ 승산역
  $(Z\times X)\%\ (2^{16}+1)=1$

위의 연산 중에서 어떤 값의 승산역을 구하는 방법은 쉽지 않기 때문에 그 해답을 구하는 알고리즘을 사용하는 바, 일반적으로 Euclid 알고리즘을 사용하여 승산역을 구한다.

Ⅲ. 회로 설계

IDEA 알고리즘을 회로로 구현하기 위하여 [그림 2]와 같이 전체 회로를 6개의 주요 기능블럭으로 분할하여 설계하였다. 주요 블록으로 암호키 및 복호키 생성부, 입력 데이터 처리부, 암호화 처리부, 출력 데이터 처리부, 그리고 동작모드 제어부 등이 있다. 암호화 처리는 미리 서브키를 준비해 놓은 상태에서 수행되며, 서브키 생성은 처음에 한번만 이루어지면 된다. 따라서, 서브키 생성에 있어서 연산속도는 그다지 중요한 문제가 아니므로 가능한 한 회로면적을 줄이는 방향으로 설계목표를 정했다. 그러나 동일한 연산을 8회에 걸쳐 수행하는 암호화 처리부는 연산속도가 매우 중요한 요소로 부각되어 회로면적의 증가에 상관없이 빠른 연산을 할 수 있도록 설계하였다. 특히, 연산속도를 높이는데 있어 가장 큰 장애



[그림 2] IDEA 회로 블록도

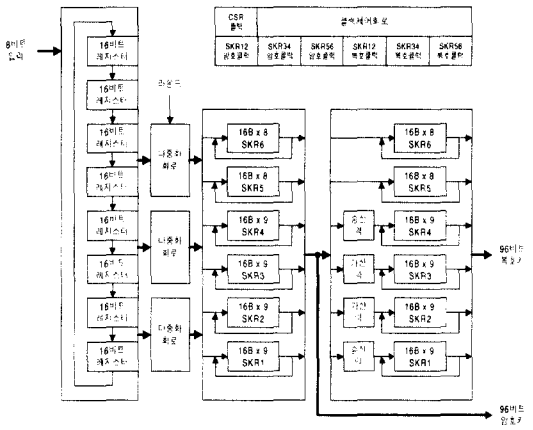
요인으로 작용하는 모듈라 승산기의 설계[4]에 많은 노력을 기울였다. 회로설계에 사용된 라이브러리는 Faraday Technology 사의 0.25μm standard cell 라이브러리인 FS9000A를 사용하였다.

3.1 암호키 생성부

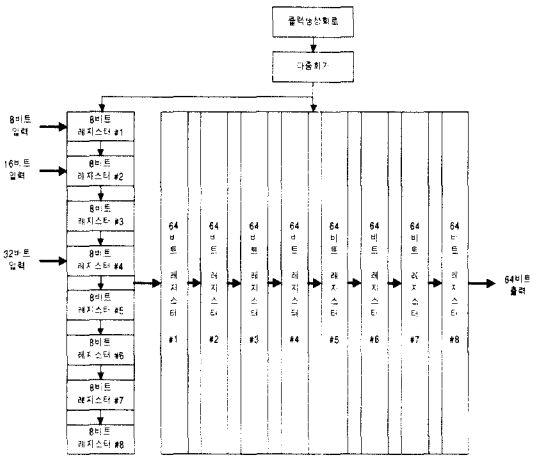
암호키 생성부는 128비트 키워드 레지스터와 16비트 서브키를 저장하기 위한 52개 메모리 공간 및 저장할 서브키를 선택하는 다중화기로 구성된다. 먼저 입력 핀으로부터 8비트, 16비트 또는 32비트 단위로 키워드를 받아들여 128비트 순환이동 레지스터에 저장한다. 이 레지스터에 저장된 데이터를 16비트씩 분할하여 8개의 서브키를 만들고 이중 시작부분의 6개 서브키를 1라운드용 서브키로 사용하기 위하여 해당 메모리에 저장한다. 나머지 2개 서브키는 2라운드의 시작부분 서브키로 사용한다. 다음에 키워드가 저장되어 있는 순환이동 레지스터를 좌측으로 25비트 순환이동시키고 다시 저장된 데이터를 16비트씩 분할하여 8개의 서브키를 만든다. 이중 시작부분의 4개 서브키를 2라운드의 끝부분 서브키로 사용하고 나머지 4개 서브키는 3라운드의 시작부분 4개 서브키로 사용하기 위하여 메모리에 저장한다. 이러한 일련의 과정이 나머지 동작에도 같은 방법으로 이루어져서 키워드 레지스터를 4회 순환이동하면 52개의 서브키를 얻을 수 있다.

3.2 복호키 생성부

복호키는 이미 생성되어 있는 암호키의 모듈라 연산역을 구함으로서 만들어진다. 회로는 복호키를 저장하기 위한 메모리와 가산역 및 승산역을 구하는 회로, 그리고 암호 서브키를 선택하는 다중화기 등으로 이루어진다. 복호키는 암호키 메모리로부터 다중화기를 거쳐 복호키 메모리로 입력된다. 가산역은



〔그림 3〕 암호키 및 복호키 생성회로



〔그림 4〕 입력 데이터 처리회로

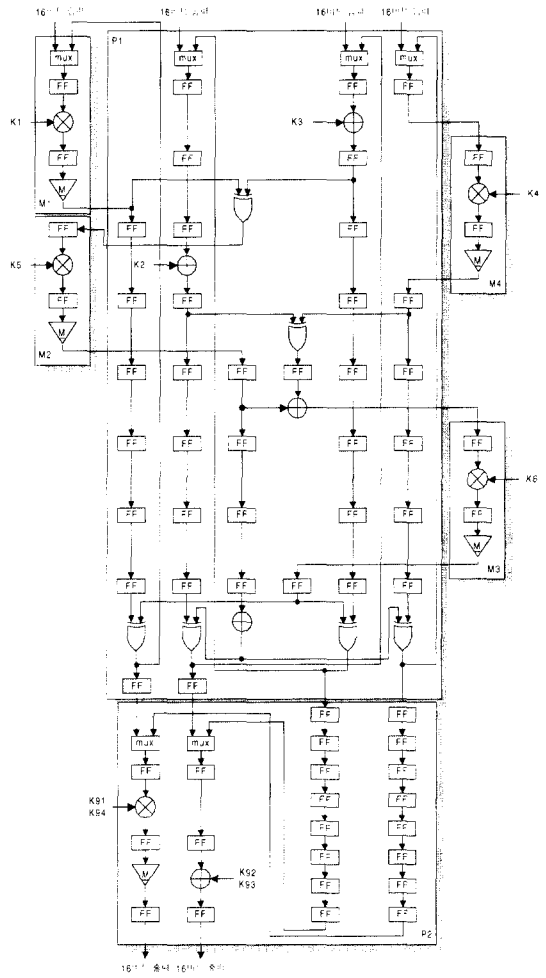
간단하게 구해지므로 암호 서브키의 가산역을 사용하는 둘째, 셋째 단의 복호 서브키는 이동하면서 직접 가산역이 구해진다. 이동하는 위치는 대략 암호키 메모리의 역순으로 진행되며 상세한 내용을 〔표 1〕에서 보여주고 있다. 복호키 메모리가 모두 채워진 후 첫째와 넷째 단에 있는 서브키의 승산역을 차례대로 구하기 위하여 1라운드와 9라운드가 연결된 순환이동으로 서브키를 꺼내 승산역을 구한 뒤 다시 원래 위치에 입력한다. 복호키를 생성하는 동작은 빠른 속도를 요구하지 않기 때문에 1개의 회로를 사용하여 모든 승산역을 구함으로서 회로면적을 최소화하고, 또한 이 회로에 포함된 승산기 및 제산기도 연산시간에 관계없이 면적을 최대한 줄일 수 있는 구조를 사용하여 설계한다. 따라서 복호키 생성용 모듈라 승산기 및 제산기는 직렬 구조로 설계하였다. 암호키 및 복호키 생성회로를 다음 그림에 도시하였다.

3.3 입력 데이터 처리부

입력 데이터 처리부는 입력핀으로부터 데이터를 받아들여 8라운드에 대한 64비트 프레임을 구성하고 다시 각 라운드에 대해서 16비트 단위의 4개 블록으로 분할하여 입력 데이터 메모리에 저장된다. 회로는 64비트 입력 데이터 버퍼와 클럭 생성기, 다중화기 및 64x8비트 크기의 입력 데이터 메모리로 구성된다. 입력 데이터는 동작모드에 따라 8비트 또는 16비트, 32비트 단위로 받아들일 수 있도록 입력 데이터 버퍼를 8비트, 16비트 또는 32비트 단위로 이동하도록 설계한다. 동작모드에 따라 사용되는 클럭이 다르기 때문에 클럭 생성기에서 모드에 맞는 클럭을 생성하고 다중화기를 통하여 공급한다. 입력 데이터 메모리는 64비트 데이터 단위로 메모리 위치를 이동하면서 입력 데이터 버퍼로부터 64비트 단위의 데이터를 일정한 클럭을 사용하여 연속적으로 받아들인다. 입력 데이터 메모리의 출력은 입력 클럭보다 8배 높은 클럭을 사용하여 8라운드에 해당하는 데이터를 1라운드가 동작하는 시간동안 차례로 이동 출력한다. 〔그림 4〕에서 입력 데이터 처리회로를 보여주고 있다.

3.4 암호화 처리부

IDEA 암호알고리즘의 핵심적인 비화과정은 서브키와의 모듈라 승산 및 가산과 XOR 연산으로 이루어진다. 이러한 연산을 효율적으로 결합하여 구성한 IDEA 알고리즘은 8라운드에 걸쳐 반복된 과정을 수행하기 때문에 연산속도와 회로면적을 고려하여 하드웨어 구현은 파이프라인 회로구조를 사용하였다. 연산속도를 높이고 회로면적을 최소화하기 위하여 파이프라인 구조[5]를 사용함으로서 회로설계에 가장 큰 비중을 차지하는 모듈라 연산기는 반복된 연산을 수행하기 위한 4개와 9번째 라운드 연산을 위한 1개 등 모두 5개로서 충분하다. 그리고 회로배치는 〔그림 5〕와 같이 파이프라인회로와 5개의 모듈라 승산기를 바탕으로 6개의 블록으로 분할하여 구성하였다. 그림에서 M1~4은 반복 연산에 사용되는 4개의 모듈라 승산기를 나타내며 승산 연산과 모듈라 연산이 구분되어 2클럭으로 이루어진다. 승산기의 연산시간이 회로의 연산속도를 좌우하기 때문에 최대한 속도를 높일 수 있도록 16비트 입력 및 출력데이터 버퍼도 같은 블록 내에 배치하였다. 또한, 캐

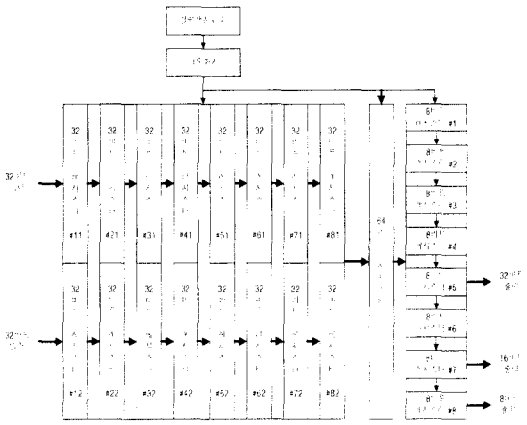


〔그림 5〕 암호화 처리부

리전달시간을 감소시키는 캐리선택 가산기(CSA)와 회로면적을 줄이는 modified Booth 승산 알고리즘을 사용하여 한 클럭에 동작이 완료되도록 설계하였다. 블록 P1은 파이프라인 구조로서 1라운드 연산이 8클럭으로 이루어진다. 1라운드에서 16비트 단위의 새로운 4개 입력 데이터블럭을 8개 받아들여 암호화 연산과정을 수행하고, 2라운드에서 8라운드까지는 출력을 입력측으로 궤환시켜 동일한 과정을 반복 수행한다. 전체 소요되는 클럭은 64클럭이고 연산 데이터량은 512비트로서, 동작주파수가 125MHz이므로 연산속도는 1Gbps이다. 8라운드 연산 후에는 P2 블록으로 전달되어 마지막 9라운드에 대한 연산을 수행한다. 9라운드에 대한 연산은 8개 라운드가 수행되는 연산시간 내에만 이루어지면 되므로, 입력 데이터를 동일한 연산을 하는 2개 블록으로 분할하여 한 블록은 연속해서 연산과정을 수행하고, 다른 블록은 순서대로 16비트 단위의 2개 블록이 직렬로 8개 연결된 플립플롭에 일단 저장하였다가 먼저 블록에 대한 연산이 끝나면 다음 차례로 연산을 수행한다. 이러한 과정을 거쳐 512비트에 대한 암호화 연산을 완료하고, 결과 데이터를 16비트 단위의 2개 블록형태로 출력데이터 처리기로 넘겨준다.

3.5 출력 데이터 처리부

출력 데이터 처리부는 암호화 처리부로부터 32비트씩 8라운드에 대한 데이터를 2번 받아들여 64비트 단위로 메모리에 저장한다. 다시 각 라운드에 대한 64비트 단위의 데이터를 출력버퍼로 이동한 다음 동작모드에 따라 8비트, 16비트, 또는 32비트 단위로 출력한다. 회로는 32×2×8비트 크기의 출력 데이터 메모리와 64×2비트 출력 데이터 버퍼 및 클럭 생성기, 다중화기 등으로 구성된다. 출력 데이터 메모리는 암호화 처리부로부터 32비트 단위로 8라운드에 대한 데이터를 2라운드 동작시간 동안 2번 받아들여 메모리 공간을 채운 다음, 64비트 데이터 단위로 메모리 위치를 차례로 이동하면서 출력 데이터 버퍼로 64비트 단위의 데이터를 일정한 클럭을 사용하여 7라운드 동작시간 동안 내보낸다. 이때 첫째 라운드 동작시간에서 출력버퍼로 이동하는 데이터는 2라운드에 해당하는 메모리 위치를 이동하도록 한다. 출력 데이터는 동작모드에 따라 8비트 또는 16비트, 32비트 단위로 내보낼 수 있도록 출력 데이터 버퍼를 8비트, 16비트 또는 32비트 단위로 이동하도록 설계한다. 동작모드에 따라 사용되는 클럭이 다르기 때문에 클럭 생성기에서 모드에 맞는 클



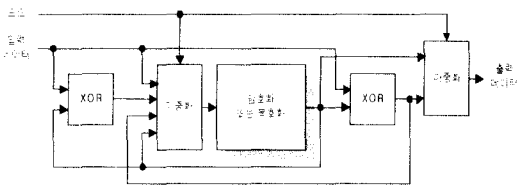
〔그림 6〕 출력 데이터 처리회로

력을 생성하고 다중화기를 통하여 공급한다. 출력 데이터 메모리의 입력은 출력 클럭보다 8배 높은 클럭을 사용하여 8라운드에 해당하는 데이터를 1라운드가 동작하는 시간동안 차례로 이동하면서 입력한다. [그림 6]에 출력 데이터 처리회로가 도시되어 있다.

3.6 동작모드 제어부

동작모드는 3가지 종류로 분류된다. 입출력 데이터의 크기와 암호화/복호화 방식 및 부가적인 4개의 암호화 동작방식 등으로 나누어진다. 입력 및 출력 데이터는 8비트, 16비트, 또는 32비트 크기로 입력 또는 출력될 수 있다. 이 동작모드를 선택함에 따라서 입력 및 출력 데이터 처리회로의 동작클럭이 달라진다. 동일한 회로를 사용하여 암호화 또는 복호화 동작을 수행할 수 있는데 이 모드를 선택하는 것에 따라서 암호화 처리회로에서 사용되는 서브키가 달라진다. 또한 내부 암호화 동작을 수행한 후 외부적으로 추가적인 암호화 동작을 위해 4개의 동작방식을 선택할 수 있다. 가장 일반적인 ECB를 비롯하여 일종의 블록 스트림 암호방식인 CBC, k비트 암호문을 궤환시켜 64비트가 아니라 k비트 단위로 암호화할 수 있는 CFB, 또는 k비트 단위로 출력을 궤환시키는 OFB 동작방식이 있다.

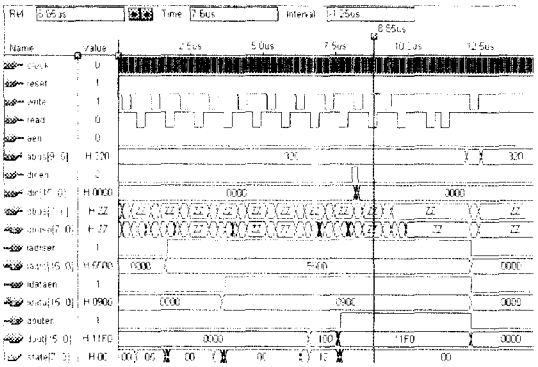
이 부가적인 암호화 처리회로는 외부적으로 설계되기 때문에 내부 암호화 처리회로는 아무런 변화가 없다. 이 모드를 포함한 전체 회로구조를 [그림 7]에서 보여주고 있다.



[그림 7] 동작모드 제어회로를 포함한 전체 회로구조

Ⅳ. 모의실험 및 검토

IDEA 알고리즘을 수행하는 기능별로 설계한 블록들을 통합하여 top 시뮬레이션을 실시하였다. 작업은 Mentor 사의 Quicksim 툴을 이용하였다. 시뮬레이션 수행방법은 랜덤 비트열을 사용하여 부호화한 후 출력파형을 파일로 저장하고, 이 파일을 복호화 과정의 입력으로 사용하여 시뮬레이션한 다음 그 결과를 부호화 과정의 입력과 비교하여 동일 여부를 확인한다. 이때 부호화 및 복호화 과정에서 연산 지연이 발생하는 바, 이 지연시간을 고려하여 입출력 비트를 비교하여야 한다.



[그림 8] 시뮬레이션 파형

회로는 먼저 부호화 모드로 동작하여 평문 비트를 암호문 비트로 변환한 후, 복호화 모드에서 다시 암호문 비트를 평문 비트로 복구한다. 이러한 시뮬레이션 과정을 4개의 동작모드, 즉 ECB, CBC, OFB, 그리고 CFB 등에 대해서 실시하였다. [그림 8]은 ECB 모드에서 암호화시 입출력 파형을 보여주고 있다.

125MHz master 클럭을 사용하여 시뮬레이션한 결과, 데이터의 암호화 및 복호화에 있어서 latency 시간이 ECB 모드에서는 1μs였으나 CBC 모드에서는 1.2μs, OFB 모드에서는 1.5μs, 그리고 CFB 모드에서는 1.6μs 등으로 나타났다. 이와 같이 궤환모드에서는 대체로 ECB 모드보다 많은 latency 시간이 소요되었다. 그러나 일반적으로 데이터 통신에서 이 정도의 시간 지연은 그다지 문제가 되지 않는다.

Ⅴ. 결 론

IDEA 알고리즘을 사용하여 고속 암호 IC를 설계하였다. 전체 회로는 6개의 주요 기능블럭으로 분할되는데 암호키 및 복호키 생성부, 입력 데이터 처리부, 암호화 처리부, 출력 데이터 처리부, 그리고 동작모드 제어부 등으로 구성되어 있다.

암호화 처리는 미리 서브키를 준비해 놓은 상태에서 수행되며, 서브키 생성은 처음에 한번만 이루어지면 된다. 따라서 서브키 생성에 있어서 연산속도는 그다지 중요한 문제가 아니므로 가능한 한 회로면적을 줄이는 방향으로 설계목표를 정했다. 그러나, 동일한 연산을 8회에 걸쳐 수행하는 암호화 처리부는 연산속도가 매우 중요하기 때문에 회로면적의 증가에 상관없이 빠른 연산을 할 수 있도록 설계하였다. 특히,

연산속도를 높이는데 있어 가장 큰 장애요인으로 작용하는 모듈라 승산기의 설계에 많은 노력을 기울였다.

IDEA 알고리즘은 8라운드에 걸쳐 반복된 과정을 수행하기 때문에 연산속도와 회로면적을 고려하여 파이프라인 회로구조를 사용하였다. 연산속도를 높이고 회로면적을 최소화하기 위하여 파이프라인 구조를 사용함으로서 회로설계에 가장 큰 비중을 차지하는 모듈라 연산기는 반복된 연산을 수행하기 위한 4개와 9번째 라운드 연산을 위한 1개 등 모두 5개로서 충분하다. 그리고 모듈라 승산기의 연산속도가 회로의 연산속도를 좌우하기 때문에 최대한 속도를 높일 수 있도록 16비트 입력 및 출력데이터 버퍼도 같은 블록 내에 배치하였다. 특히, 승산기는 캐리전달시간을 단축시키는 캐리선택 가산기와 회로면적 및 연산시간을 감소시키는 modified Booth 승산 알고리즘을 사용함으로서 연산시간이 8ns 이내가 되도록 설계하였다.

또한, 입력 데이터를 동작모드에 따라 8-bit, 16-bit, 32-bit 씩 받아들이기 위하여 입력 데이터버퍼가 8-bit, 16-bit, 32-bit 단위로 이동하도록 설계하였다.

Faraday Technology 사의 0.25μm standard cell 라이브러리인 FS9000A를 사용하여 회로설계하고 timing 시뮬레이션한 결과, 이 IC는 큰 면적을 요구하지 않으면서도 1Gbps 이상의 throughput을 달성하였으며, 회로구현에 약 12,000gates가 소요되었다. 따라서, 설계한 블럭암호 IC가 스마트카드 등 고속의 암호분야에 적용될 수 있을 것으로 사료된다.

참 고 문 헌

[1] X. Lai, J. L. Massey, "A Proposal for a New Block Encryption Standard", in Advances in Cryptology-EUROCRYPT '90, Berlin Germany : Springer-Verlag, pp. 389~404, 1990.

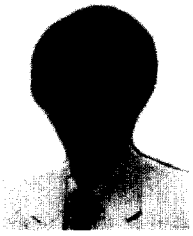
[2] R. Zimmermann, A. Curiger, H. Bonnenberg, H. Kaeslin, N. Felber, and W. Fichtner, "A 177 Mb/s VLSI Implementation of the International Data Encryption Algorithm", IEEE J. of Solid-State Circuits, Vol. 29, No. 3, pp. 303~307, March 1994.

[3] H. Bonnenberg, A. Curiger, N. Felber, H. Kaeslin, and X. Lai, "VLSI Implementation of a New Block Cipher", in ICCD '91 1991 IEEE Int. Conf. Computer Design : VLSI in Computer and Processors, Washington, DC : IEEE Computer Society Press, pp. 510~513, 1991.

[4] A. Curiger, H. Bonnenberg, and H. Kaeslin, "Regular VLSI Architectures for Multiplication Modulo (2^n+1)", IEEE Solid-State Circuits, Vol. 26, No. 7, pp. 990~994, July 1991.

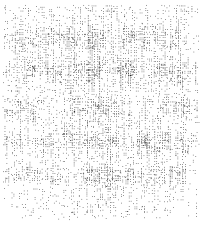
[5] 박태규, 황대준, "DES의 고속 암호화를 위한 파이프라인 구조", 통신정보보호학회논문지, 제3권 제2호, pp. 41~51, 1993.

〈著者紹介〉



이 행 우 (Haeng-Woo Lee)

1985년 2월 : 광운대학교 전자공학과(공학사)
1987년 2월 : 서강대학교 대학원 전자공학과(공학석사)
2001년 2월 : 전북대학교 대학원 전자공학과(공학박사)
1987년 2월~1998년 3월 : 한국전자통신연구원 선임연구원
1998년 4월~2001년 2월 : 벽성대학 정보통신과 교수
2001년 3월~현재 : 남서울대학교 전자정보통신공학부 교수
〈관심분야〉 암호IC 설계, 디지털통신, VLSI 설계



최 광 진 (Kwang-Jin Choi)

1991년 2월 : 광운대학교 대학원 컴퓨터공학과(공학석사)
1996년 2월 : 전북대학교 대학원 컴퓨터공학과 박사수료
2001년 현재 : 벽성대학 정보통신과 겸임교수
2001년 현재 : (주)맥텔레콤 대표이사
〈관심분야〉 암호 시스템, 정보통신 시스템