파이프라이닝을 이용한 AES 암호화 알고리즘의 FPGA 구현

(FPGA Implementation of the AES Cipher Algorithm by using Pipelining)

김 방 현 † 김 태 규 † 김 종 현 ††
(Bang-Hyun Kim) (Tae-Kyu Kim) (Jong-Hyun Kim)

요 약 본 연구에서는 최근 미국표준기술연구소(NIST)에 의해 암호화 표준 알고리즘으로 채택된 AES 알고리즘을 Altera FLEX10KE 계열의 하드웨어로 구현하는 여러 가지 방법들에 대하여 VHDL 설계를 이용하여 전반적으로 분석하였다. 구현 방법들로는 기본 구조, 루프 언롤링, 라운드 내부 파이프라이닝, 라운드 외부 파이프라이닝, 그리고 S-box의 자원 공유 등을 사용하였다. 이 연구에서 VHDL 설계 및 시뮬레이션은 Altera 사의 MaxPlus2 9.64를 이용하였으며, FPGA는 Altera 사의 FLEX10KE 계열을 사용하였다. 결과에 따르면, 4-단계 라운드 내부 파이프라이닝 구현 방법이 성능대가격비 면에서 가장 우수한 것으로 나타난 반면에, 루프 언롤링 방법이 가장 뒤떨어지는 것으로 나타났다.

키워드 : AES 알고리즘, VHDL 설계, FPGA, 파이프라이닝

Abstract In this study, we analyze hardware implementation schemes of the AES(Advanced Encryption Standard-128) algorithm that has recently been selected as the standard cypher algorithm by NIST(National Institute of Standards and Technology). The implementation schemes include the basic architecture, loop unrolling, inner-round pipelining, outer-round pipelining and resource sharing of the S-box. We used MaxPlus2 9.64 for VHDL design and simulations and FLEX10KE-family FPGAs produced by Altera Corp. for implementations. According to the results, the four-stage inner-round pipelining scheme shows the best performance vs. cost ratio, whereas the loop unrolling scheme shows the worst.

Key words : AES algorithm, VHDL design, FPGA, pipelining

1. 서 론

초고속 정보통신망을 비롯한 정보 인프라의 구축이 확대되면서 다양한 종류의 정보 서비스들이 활성화되고 있다. 그러한 서비스들 중에는 송수신되는 정보의 기밀성이 반드시 보장되어야 하는 경우가 많지만, 아직 정보보호에 대한 대책은 미비한 실정이다. 다시 표현하면, 그와 같이 다양한 응용 서비스들을 안전하게 제공하기 위해서는 정보의 신뢰성을 보장해주기 위한 정보보호 기술이 필수적으로 개발되어야 한다. 그에 따라 최근 정보 암호화(information encryption)와 전자 서명 및 인증(electronic signature and authentication) 기술에 대한 관심과 수요가 급증하고 있다.

정보를 암호화하는 암호시스템(cryptosystem)은 키(key) 관리 형태에 따라 비밀키 암호시스템(secret-key cryptosystem)과 공개키 암호시스템(public-key cryptosystem)으로 분류될 수 있다. 대칭 암호시스템(symmetric cryptosystem) 혹은 단일키 암호시스템(one-key cryptosystem)이라고도 불리는 비밀키 암호시스템은 암호화(encryption)와 복호화(decryption)에 같은 키를 사용하는 방식이다. 예를 들면, 송신자는 전송하고자 하는 평문(plaintext)을 키와 암호 알고리즘을 이용하여 암호문(ciphertext)으로 변환하여 수신자에게 전송한다

· 이 연구는 2001년도 정보통신부 대학기초연구지원사업의 지원에 의해 이루어졌음
† 학생회원 : 연세대학교 전산학과
legnamai@chol.com
windcry@korea.com
†† 종신회원 : 연세대학교 전산학과 교수
jhkim@dragon.yonsei.ac.kr
논문접수 : 2000년 5월 25일
심사완료 : 2002년 8월 6일

수신자는 동일한 키를 복호 알고리즘에 적용하여 원래의 평문을 만들어 낸다. 이때, 송신자와 수신자는 암호화 통신을 하기 전에 안전하게 키를 교환해야 하며, 암호 통신을 도청하려는 제3자는 그 키가 없는 한 원래의 평문을 판독할 수가 없게 된다. 이 방식의 단점인 키 관리의 어려움을 해결하기 위하여 공개키 암호시스템이 개발되었지만, 처리 시간이 길고 하드웨어로 구현하는 경우에 매우 복잡해진다는 문제가 있다.

최근에는 전송되는 정보의 크기가 증가하면서 암호화 속도가 점차 더 중요해지고 있기 때문에 비밀키 암호시스템이 널리 활용되고 있다. 비밀키 암호시스템은 다시 블록 암호(block cipher) 방식과 스트림 암호(stream cipher) 방식으로 나누어진다. 블록 암호 방식은 평문을 블록 단위로 암호화시키는 것을 말하며, 스트림 방식은 선형 쉬프트 레지스터를 이용하여 평문을 한 번에 한 비트씩 암호화시키는 암호시스템을 말한다. DES(Data Encryption Standard)[1]는 지난 20년 동안 세계적인 표준으로 사용되어 온 64 비트 블록 암호 알고리즘으로서, 56 비트의 키를 사용하여 블록을 암호화하며, 1976년 미국 연방 표준으로 채택된 후 매 5년마다 안전성 평가를 통하여 안전성을 인정받아 왔다. 그러나 근래에 들어 컴퓨터 속도의 비약적인 발전으로 인하여 안전하다고 알려져 왔던 DES 등의 암호가 전사적 공격(brutal attack)을 통하여 짧은 시간 내에 해독될 수 있다는 것이 여러 실험을 통하여 입증되어 왔다. 그러한 위협에 대응하기 위하여 1997년 1월 미국표준기술연구소(NIST: National Institute of Standards and Technology)는 DES를 대체하기 위한 새로운 표준 블록 암호화 알고리즘인 AES(Advanced Encryption Standard)를 개발하기 위한 계획을 발표하였다. 이 계획안에 따라 세계 각국을 대상으로 암호 알고리즘을 모집하였고, 후보 알고리즘들을 대상으로 2년여 동안의 심사 과정을 통하여 2000년 10월경에 최종적으로 벨기에의 Joan Daemen과 Vincent Rijmen이 제안한 Rijndael 알고리즘[2]을 채택하였다. 이 알고리즘은 기존에 알려진 암호 공격법에 대하여 안전하며 다양한 시스템 환경에서 암호화 속도와 복호화 속도, 및 소스 코드의 크기가 합리적이라는 평가를 받았다. 그에 따라 NIST는 2001년 2월에 연방정보처리표준출판물(FIPS PUBS)를 통하여 Rijndael 알고리즘이 AES로 선정되었음을 공식적으로 발표하였다[3].

새로운 AES 알고리즘은 대칭적 블록 암호화 방식으로서 데이타 블록은 128 비트이고, 암호 키는 128 비트, 192 비트 혹은 256 비트 중에서 선택할 수 있도록 하였다. 이와 같이 암호 키가 128 비트 이상이기 때문에 불법적인 암호 해독은 거의 불가능하며, S/W 혹은 각종 H/W 칩으로 구현하였을 때 처리 속도와 비용이 기존의 다른 암호 알고리즘들에 비하여 상대적으로 우수하다는 것이 선정 과정에서 입증되었다. 특히 미국 표준에 이어 ISO에서도 세계 표준으로 채택하기 위한 검토도 이루어지고 있기 때문에, AES 알고리즘은 앞으로 각종 정보보안에 가장 널리 사용될 것이 거의 확실하다. 따라서 각국의 정보보안업체들과 대학들은 이 새로운 알고리즘을 구현하여 실제 제품화하기 위한 많은 노력을 기울이고 있으나, 불과 수개월 전에 발표되었기 때문에 아직 발표된 결과는 없는 상태이다.

암호화 알고리즘의 구현은 소프트웨어와 하드웨어 모두 가능하다. 소프트웨어로 구현하는 경우의 장점은 사용이 용이하고, 포팅(porting)과 업그레이드(upgrade)가 쉬우며 융통성이 높다는 점이다. 그러나 소프트웨어 구현은 키 관리상의 안전성이 떨어지고 처리 속도가 낮다는 단점을 가지고 있다. 반면에 하드웨어로 구현하면 비용이 더 들지만, 외부 공격에 의해 내용을 읽거나 수정하기가 어렵기 때문에 안전성이 높다. 하드웨어 구현 방식들 중에서도 ASIC(Application-Specific Integrated Circuit)을 이용하는 방법은 알고리즘과 파라미터 변경에 대하여 융통성이 부족하기 때문에, 기능이 고정되어 있지 않고 시스템 내에서 프로그래밍이 가능한 FPGA(Field Programmable Gate Array)가 널리 사용되고 있다[4,5].

AES가 선정되는 과정에서 B. Weeks 등[6]과 A. J. Elbirt 등[7]이 AES 최종 후보 알고리즘들을 대상으로 VHDL을 사용하여 FPGA로 설계하고 성능을 비교하였다. 이 연구들에서는 Xilinx Virtex 계열의 XCV1000 BG560-4 FPGA를 기반으로 하였으며, VHDL 설계 및 시뮬레이션은 Synopsys를 사용하였다. [6]의 연구는 외부에서 미리 생성된 키가 FPGA의 내부 레지스터에 저장되어 있고, 하나의 블록이 1 ns 주기의 클록 사이클에 의해 처리된다는 가정 하에서 5 개의 AES 후보 알고리즘들을 VHDL로 설계하고 평가하였다. 그러나 이 연구는 기본 구조의 반복 라운드와 10 단계의 라운드 외부 파이프라이닝만을 대상으로 하였고, 이상적인 조건을 가정하여 얻어진 이론적 결과만 보여주고 있다. 반면에 [7]의 연구는 실질적인 FPGA의 구현을 통해서 5개의 최종 AES 후보 알고리즘들을 평가하였는데, 여러 가지 구현 방법들에 따른 평가 결과를 제시하고 있다. 이 연구에서는 성능의 척도인 처리율(throughput)을 높이기 위하여 16개의 S-box를 조합회로로 FPGA에 함

께 구현하였는데, 이것은 FPGA가 대용량일 경우에만 가능하다. 또한 라운드 내부 파이프라인에 대한 연구는 이루어지지 않았기 때문에 이에 대한 효과를 파악할 수가 없다.

이 연구들에서 사용된 FPGA는 고성능, 대용량으로서 개당 가격이 일백만원을 넘는 고급 FPGA이다. 그리고 데이타와 키 블록의 입출력 폭을 각각 128 비트로 하였는데, 이것은 기본적으로 384개의 핀을 사용한다는 것을 의미한다. 따라서 제시된 평가 결과는 AES를 최상의 FPGA로 구현하였을 때 얻을 수 있는 최상의 결과를 보여준다고 할 수 있다. 그렇지만 실제 AES의 FPGA 구현에 있어서는 성능과 비용 문제가 함께 고려되어야 한다. 즉 어떠한 처리율을 요구하는 AES 응용이 있다면, 그 성능을 만족시키는 한도 내에서 최소 비용을 갖는 구현 방법에 대한 연구가 필요하다. 그리고 저가의 FPGA를 사용하는 경우에는 적은 수의 인터페이스 핀을 가지고 있기 때문에 블록을 나누어 처리해야 되므로, 인터페이스 폭의 변경에 따른 성능 분석도 필요하다.

본 연구에서는 AES 암호화 알고리즘을 대규모 데이타를 신속하게 암호화할 수 있는 파이프라인 구조 기반으로 설계하고, VHDL 설계를 이용하여 FPGA로 구현한다. 그리고 다양한 내부 구조들에 대한 성능을 분석하고, 성능대가격비(performance vs. cost ratio)를 극대화할 수 있는 최적의 구현 방법을 찾는 것을 목표로 한다. 본 연구에서는 Altera FLEX10KE 계열의 FPGA를 사용하였으며, VHDL 설계 및 시뮬레이션은 Altera 사의 MaxPlus2 9.64를 이용하였다.

본 논문의 구성은 2장에서 AES 블록 암호화 알고리즘에 대하여 개략적으로 살펴본 다음에, 3장에서 이 알고리즘의 FPGA 구현에 대하여 설명한다. 그리고 4장에서는 파이프라이닝 및 인터페이스 폭의 변화에 따른 성능을 비교 분석하고 비교한 결과를 제시하고, 마지막으로 5장에서 결론을 정리하였다.

2. AES 블록 암호화 알고리즘

1장에서 언급한 바와 같이 NIST에 의하여 최종 AES 표준 알고리즘으로 선정된 Rijndael 암호화 알고리즘은 암호화와 복호화에 필요한 키(key)가 동일한 대칭 블록 암호 알고리즘으로서 128, 192, 256 비트의 블록 크기와 키 길이를 지원한다. 또한 암호화 키 길이와 암호화의 기본 단위인 데이타 블록의 크기를 128, 192, 256 비트 중에서 선택할 수 있다. 따라서 이 알고리즘에서는 키와 블록의 크기를 조합한 아홉 가지의 다양한 선택이 가능하지만, 표준에서는 데이타 블록의 크기는 128 비트로 고정하였고, 키의 길이는 128, 192, 256 비트 중에서 선택할 수 있도록 제한하였다[7]. 그리고 각 키 길이에 대하여 알고리즘을 "AES-128", "AES-192" 및 "AES-256"이라 명명하였다. 표 1은 각 키 길이에 대하여 암호화 과정에서 필요한 반복 라운드(round) 수를 보여주고 있다. 본 연구는 이들 중에서 AES-128을 대상으로 하였으며, 결과적으로 연산 과정에서 라운드가 10번 반복 수행된다.

표 1 키와 블록 길이에 따른 반복 라운드 수

AES 알고리즘	키 길이	데이타 블록 길이	라운드 수
AES-128	128	128	10
AES-192	192	128	12
AES-256	256	128	14

AES-128에서는 128 비트의 데이타 블록을 16개의 바이트 단위로 분할하고, 그들을 네 개의 행과 네 개의 열로 배열하여 연산을 수행한다. 암호화 연산은 그림 1에서 보는 바와 같이 ByteSub 변환, ShiftRow 변환, MixColumn 변환 및 AddRoundKey 변환의 순서로 구성된 라운드를 10번을 반복 수행함으로써 이루어진다. 그림 1의 전체 과정을 개략적으로 살펴보면, 먼저 입력 데이타 블록인 평문(plaintext)과 원래의 비밀 키 사이에 AddRoundKey 변환이 이루어진 다음에, 네 가지 변환들이 순서대로 수행된다. 이 과정은 처음 9번의 라운드에 대하여 동일하게 반복되며, AddRoundKey 변환에서는 각 라운드를 위해 생성된 라운드 키를 이용하게 된다. 그리고 마지막 10번째 라운드에서는 MixColumn 변환이 수행되지 않는다. 그림 2는 복호화 과정의 흐름도를 보여주고 있는데, 암호화 과정과 반대 순서로 이루어진다.

각 라운드에서 수행되는 변환들에 대하여 좀 더 자세히 살펴보면 다음과 같다.

(1) ByteSub 변환

이 변환은 그림 3에서 보는 바와 같이 S-box를 이용하여 각 바이트에 독립적으로 작용하는 비선형적 바이트 치환(nonlinear byte substitution)이다. S-box는 각 바이트에 대하여 일대일 치환을 수행하기 위한 비선형 치환 테이블로서, 16×16 형태의 256 바이트로 구성된다[1,2]. 암호화와 복호화에 사용되는 S-box는 서로 대칭이며, 복호화에 사용되는 S-box를 inverse S-box라 한다.

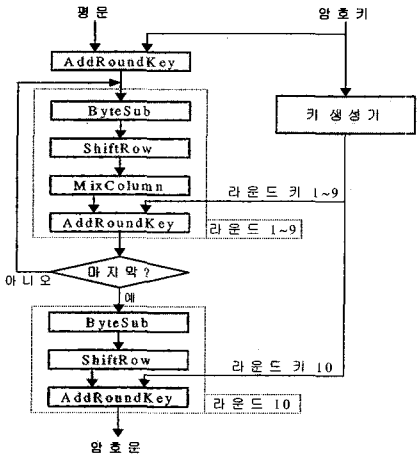


그림 1 암호화 과정의 흐름도

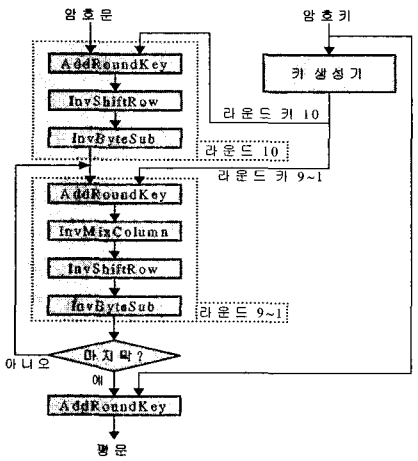


그림 2 복호화 과정의 흐름도

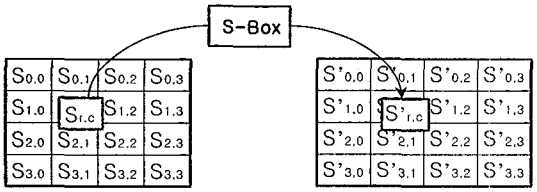


그림 3 ByteSub 변환

(2) ShiftRow 변환

이 변환에서는 행(row) 단위로 순환 쉬프트(circular shift) 연산이 이루어진다. 행 번호가 0인 첫 행은 쉬프트 되지 않고, 행 번호가 1인 행은 한번 좌측으로 쉬프트 되며, 행 번호가 2인 행은 두 번 쉬프트 된다. 그리고 행 번호가 3인 행은 세 번 쉬프트 된다. 이것은 같은 행에 있는 바이트들이 열의 번호가 낮은 위치로 이동하는 결과를 가져오며, 열 번호가 가장 낮은 위치의 바이트는 최상위의 열의 위치로 가게 된다. 그림 4는 Shift Row 변환 과정을 보여주고 있다.

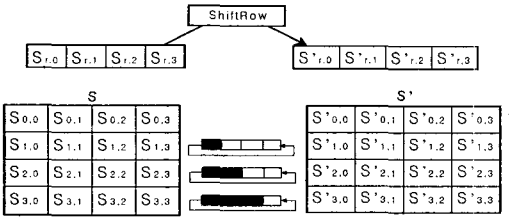


그림 4 ShiftRow 변환

(3) MixColumn 변환

이 변환은 각 열(column)을 4차 다항식으로 취급하여, 고정된 다항식 a(x)와 곱함으로써 이루어진다. 이것을 행렬의 곱셈 형식으로 나타내면 그림 5와 같다. 마지막 라운드에서는 이 변환이 수행되지 않는다.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

그림 5 MixColumn의 변환식

(4) AddRoundKey 변환

이 변환은 키 생성기에 의해 생성된 라운드 키(round key)와 데이타 블록 사이에 XOR 연산을 수행함으로써 이루어진다. 각 라운드에서 사용되는 라운드 키는 처음 입력된 128 비트의 암호 키에 키 생성 알고리즘을 적용하여 생성된다. AddRoundKey 변환은 첫 라운드 시작 전에 한번 수행되고, 다시 각 라운드마다 실행되기 때문에 하나의 데이타 블록을 처리할 때 11번 수행된다.

3. AES-128의 FPGA 구현

암호화 알고리즘을 소프트웨어로 처리하면 구현이 용이하지만, 처리 속도가 느리기 때문에 실시간 통신이나 고속 통신시스템과 연계시키는 데 어려움이 있다. 반면에 하드웨어로 구현하는 경우에는 속도가 높아지지만, 구현 방법에 따라 하드웨어 복잡도가 증가하게 되면 부

품의 수가 많아지고 크기(면적)가 커져 비용이 많이 들게 된다. 따라서 하드웨어로 구현할 경우에는 응용 목적에 따라 속도와 크기간의 적절한 조정(tradeoff)이 필요하다.

AES-128을 하드웨어로 구현하는 방법은 크게 비궤환 방식(non-feedback mode)과 궤환 방식(feedback mode)으로 나눌 수 있다[8]. 비궤환 방식은 10 라운드의 모든 과정을 하나의 조합회로로 구현하는 방식이다. 따라서 하나의 데이타 블록을 처리할 때 하드웨어를 한 번만 통과하면 되므로 처리 속도가 매우 높지만, 하드웨어가 매우 복잡하여 비용이 가장 많이 드는 방법이다.

반면에 궤환 방식은 동일한 연산을 수행하는 하드웨어는 부분적으로만 구현하고, 데이타 블록 처리 과정에서 그 하드웨어 모듈을 두 번 혹은 그 이상 반복 사용하도록 하는 방식으로 하드웨어의 크기를 줄이는 방법이다. 이 경우에는 하나의 데이타 블록을 처리하는데 그 반복 회수만큼의 클록이 소요된다. 이러한 반복 방식에는 기본 구조(basic architecture), 루프 언롤링(loop unrolling), 라운드 내부 파이프라이닝(inner-round pipelining), 라운드 외부 파이프라이닝(outer-round pipelining) 및 S-box의 자원 공유(resource sharing) 방법 등이 있다[8].

3.1 기본 구조

그림 6은 기본 구조를 이용한 구현의 개념도를 보여주고 있는데, 한 라운드를 처리하는데 필요한 회로만 하드웨어 모듈로 구현하고 제어를 통하여 필요한 회수만큼 연산을 반복 수행시킨다. 즉, 첫 번째 라운드의 처리 결과를 다시 회로에 입력시켜 처리하고, 그러한 과정을 아홉 번 더 반복하는 것이다. 이 구현 방법은 하드웨어 크기가 가장 작아지는 경우로서, 처리 속도는 다른 구조들에 비해 가장 느리지만 비용이 최소화되는 장점이 있다.

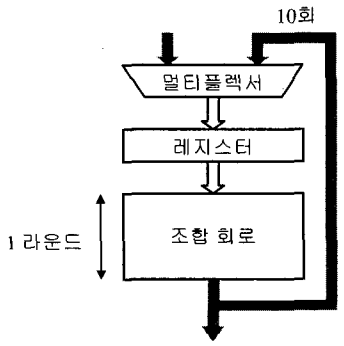


그림 6 기본 구조

본 연구에서는 이 기본 구조의 구현에 있어서 성능대 가격비를 최적화 할 수 있는 방법을 찾기 위하여 데이타 블록 및 비밀 키 입출력을 위한 외부 인터페이스 핀의 수와 내부 모듈간의 버스 폭을 조정하는 분석도 고려하였다. FPGA에서 사용되는 외부 인터페이스 핀의 수와 내부 모듈간의 버스 폭에는 적절한 선택이 필요하다. 외부 인터페이스 핀의 수가 많으면 데이타 블록을 입출력하는데 걸리는 시간이 단축되어 처리 속도는 빨라지지만 하드웨어 복잡도가 증가하여 비용이 높아질 뿐 아니라 FPGA의 주변회로들도 복잡해진다. 따라서 본 연구에서는 기본 구조를 외부 인터페이스 핀의 수와 내부 모듈간의 버스 폭을 8, 16, 32, 64 및 128 비트로 변경하여 구현하면서 하드웨어 복잡도와 속도간의 관계를 분석하였다. 또한 하나의 칩에 암호화 회로와 복호화 회로가 모두 포함되도록 구현하고, 외부 인터페이스 핀의 수를 줄이기 위하여 데이타 블록과 키 블록이 동일한 핀들을 이용하여 입출력되도록 하였다. 그리고 S-box는 Altera FLEX10KE 계열에 내장된 배열 블록(EAB: Embedded Array Block)을 이용하였다.

그림 7은 인터페이스 핀의 수와 내부 모듈간의 버스 폭이 각각 32 비트인 경우에 대하여 본 연구에서 설계한 AES-128 암호화 칩의 내부 구조를 보여주고 있다. 내부 모듈은 동작 모드(암호화혹은 복호화)와 입력(데이타 혹은 키)을 선택해주는 AES 제어기, 키 생성기, 버퍼, S-box 및 그림 6의 구조를 실제 구현한 AES 코어로 이루어진다.

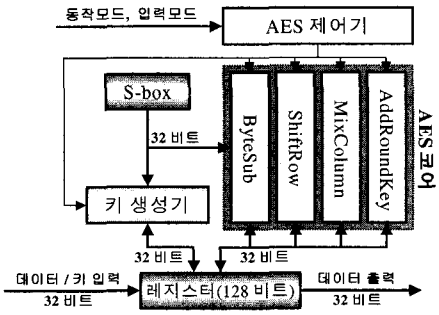


그림 7 AES 암호화 칩의 내부 구조

3.2 루프 언롤링

이 구현 방법에서는 그림 8에서 보는 바와 같이 k번의 라운드를 처리하는 하드웨어를 하나의 조합 회로로 구현한다. 따라서 전체 연산은 입력 데이타가 조합회로에 의해 10/k 번 반복 처리됨으로써 수행될 수 있다. k

값은 1, 2, 5 혹은 10이 될 수 있는데, k=1이면 기본 구조와 동일하고 k=10은 비궤환 방식에 해당한다.

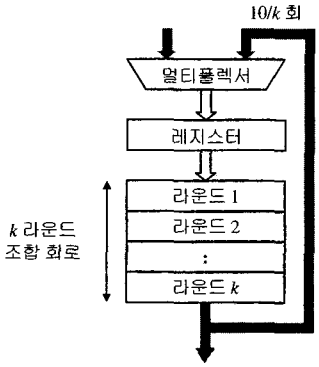


그림 8 루프 언롤링의 구성도

3.3 라운드 내부 파이프라이닝

이 방법은 그림 9와 같이 각 라운드에서 처리되는 각 변환을 별도의 하드웨어 모듈로 분리하여 파이프라인 단계(pipeline stage)로 구현하는 방법으로서, 2장에서 설명한 바와 같이 변환 과정이 모두 네 가지이므로 최대 네 단계까지 가능하다.

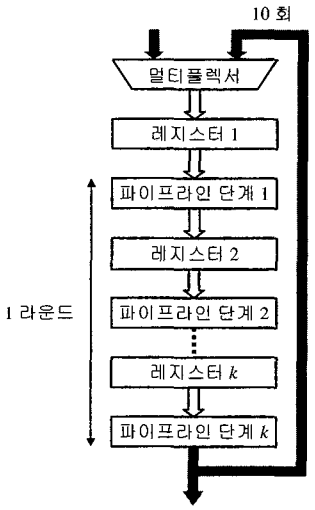


그림 9 라운드 내부 파이프라이닝의 구성도

3.4 라운드 외부 파이프라이닝

이 방법은 루프 언롤링 구조를 기본으로 하여 그림 10과 같이 각 라운드 연산을 위한 하드웨어 모듈을 파이프라인 단계로 하고, 각 단계들 사이에는 레지스터를 둔다. 파이프라이닝 단계의 수는 전체 라운드 수가 10개이므로 1개, 2개, 5개 혹은 10개로 할 수 있다.

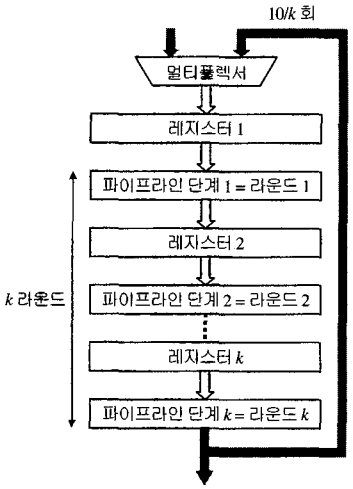


그림 10 라운드 외부 파이프라이닝의 구성도

3.5 S-box 구현

S-box는 기본적으로 한 번에 한 바이트씩만 처리할 수 있다. 따라서 AES-128의 128비트 데이타 블록을 변환하기 위해서는 S-box를 16번 반복하여 액세스해야 하는 문제점이 있는데, 이것을 해결하기 위한 방법이 자원 공유(resource sharing)이다. 여기에는 멀티플렉서를 사용하여 하나의 S-box를 시간차를 이용하여 액세스하는 방법과 두개 이상의 S-box를 두어 병렬로 액세스하는 방법이 있다. 본 연구에서는 후자의 경우를 선택하여 FPGA에 내장된 RAM인 EAB에 여러 개의 S-box들을 저장하였다.

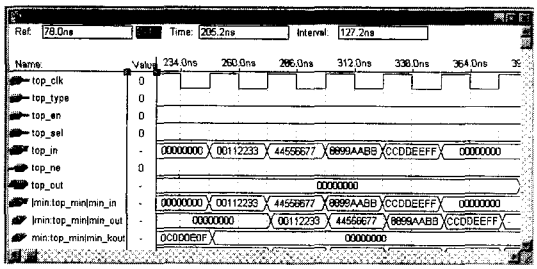
4. 성능 분석

본 연구에서 VHDL 설계 및 시뮬레이션은 Altera 사의 MaxPlus2 9.64를 이용하였으며, FPGA는 Altera 사의 FLEX10KE 계열을 사용하였다. 이 장에서는 3장에서 설명한 구현 방법들에 대한 성능 분석 결과를 요약하고 비교하였다.

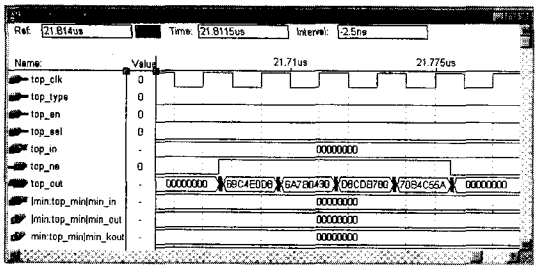
4.1 기본 구조

이 구조에 대한 성능 분석에서는 외부 인터페이스 폭(핀의 수)과 내부 모듈들간의 버스 폭을 동일하게 하였고, 그 폭을 8, 16, 32, 64 및 128 비트로 변경시키면서 성능 분석을 수행하였다. S-box는 복호화에 필요한

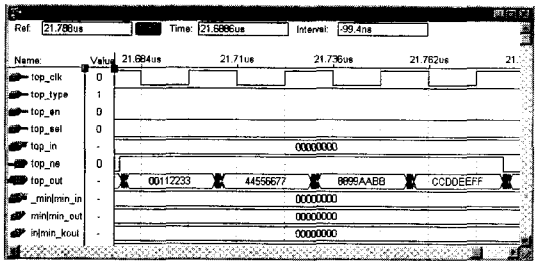
inverse S-box도 필요하므로 각각의 경우에 2, 4, 8, 16 개를 포함시켰고, 인터페이스 폭이 128 비트일 때에는 EAB의 용량에 한계가 있기 때문에 32 개가 아닌 16 개만 포함시켜 실험하였다. 먼저 인터페이스 및 내부 버스 폭이 32 비트인 경우에 대하여 암호화 및 복호화 과정을 시뮬레이션으로 수행시킨 결과는 그림 11과 같다. 이 시뮬레이션에서 입력 데이타 블록은 "00112233445566778899AABBCCDDEEFF", 키 값은 "000102030405060708090A0B0C0D0E0F"를 사용하였으며, 그림 11에서 (a)는 평문 입력, (b)는 암호화 결과, (c)는 복호화 결과를 각각 나타낸다.



(a) 평문 입력



(b) 암호화된 출력



(c) 복호화된 출력

그림 11 암호화 및 복호화 과정의 시뮬레이션 결과

시뮬레이션에 의한 성능 분석 결과에 따르면, 처리율 [Mbits/s]은 그림 12와 같이 인터페이스의 폭이 증가할수록 선형적으로 높아졌는데, 그 이유는 FPGA 외부의 데이타 입출력과 내부 모듈간의 데이타 전송에 소요되는 시간이 단축되기 때문이다. 예를 들어, 8 비트인 경우에 데이타 입출력을 위하여 16 클록이 필요하지만, 32 비트인 경우에는 4 클록, 그리고 128 비트인 경우에는 한 클록만 소요된다.

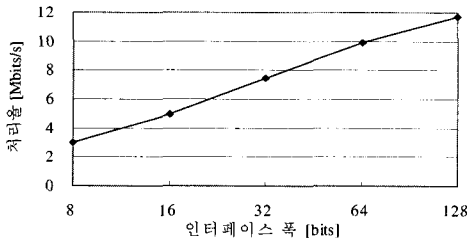


그림 12 인터페이스 폭에 따른 처리율

그러나 그림 13에서 보는 바와 같이 하드웨어 복잡도는 인터페이스의 폭이 증가할수록 점차 감소하다가 32 비트일 때 최소가 되고, 그 이후에는 다시 증가하는 결과를 보였다. 이것은 64 비트나 128 비트를 나누어 처리할 때, 카운트와 쉬프트를 처리하는 부가 회로에 필요한 논리 셀(logic cells)의 수와 인터페이스에 필요한 논리 셀(synthesized logic cells)의 수의 상대적인 비율에 따라 나타나는 현상이다. 즉, 인터페이스 폭이 32 비트 이상으로 증가하게 되면 부가 회로에 필요한 논리 셀의 감소보다 인터페이스에 필요한 셀의 증가가 더 크기 때문이다. 결과적으로 인터페이스의 폭이 증가할수록 처리율은 높아지지만, FPGA 내부와 외부의 하드웨어 복잡도를 함께 고려하였을 때 가장 효율적인 인터페이스 폭은 32 비트라고 할 수 있다. 따라서 본 연구에서는 이후의 구현 방법들에 대하여 인터페이스 폭을 32 비트로 적용하였다.

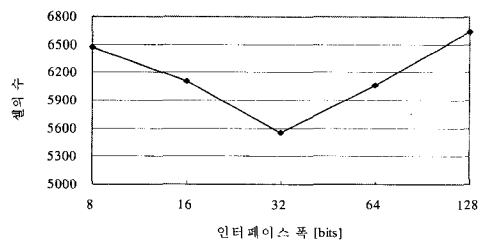


그림 13 인터페이스 폭에 따른 하드웨어 복잡도

4.2 루프 언롤링

루프 언롤링에서는 k 라운드를 처리하는데 필요한 회로들을 하나의 조합회로로 구현하기 때문에 하드웨어 복잡도가 한 라운드를 처리하는 회로의 k 배가 된다. 본 연구에서 사용한 시뮬레이션 도구가 지원하는 FPGA로는 k가 2 이상인 경우에는 하나의 칩에 합성시킬 수가 없었기 때문에, 각 모듈별로 나누어서 시뮬레이션을 한 후에 산술적으로 실험 결과를 얻었다. 실험 결과는 그림 14에서 보이는 바와 같이 k가 증가함에 따라 하드웨어 복잡도가 크게 증가하는 반면에, 처리율의 향상은 매우 적었다. 이것은 조합회로를 통과하는 데 걸리는 시간이 증가하여 기본 클록 주기가 길어지기 때문이다. 이러한 결과는 [7]에서도 거의 동일하게 나타났다.

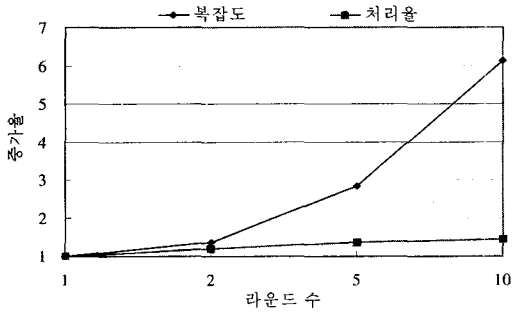


그림 14 루프 언롤링에서 처리율과 하드웨어 복잡도 증가율간의 상관관계

4.3 라운드 내부 파이프라이닝

이 구현에 대한 분석은 파이프라인 단계의 수가 1, 2, 3 및 4인 경우에 대하여 수행하였다. 인터페이스 폭은 하나의 FPGA에 구현하기 위하여 32 비트로 하였고, 데이타 블록과 키 블록의 입력은 같은 핀을 이용하였으며, S-box는 네 개를 EAB에 내장하여 사용하였다. 실험 결과에 따르면, 그림 15에서 보는 바와 같이 파이프라이인의 단계가 증가함에 따라 처리율은 거의 선형적으로 증가하였으며, 그에 비하여 하드웨어 복잡도의 증가율은 상대적으로 낮았다. 그 이유는 파이프라인 단계가 증가함에 따라 하드웨어 복잡도가 증가되는 요인이 기본 처리 모듈의 증가가 아니라 파이프라인 단계간의 레지스터 수와 파이프라인 제어 회로의 증가이기 때문이다.

표 2는 라운드 내부 파이프라이닝에서 단계의 수에 따른 분석 결과들을 보여주고 있는데, 단계의 수가 4인 경우에는 파이프라인을 하지 않았을 경우에 비하여 약 3.13배의 성능 향상을 얻을 수 있었으며, 하드웨어 복잡

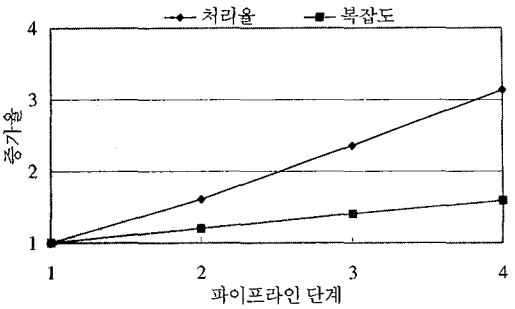


그림 15 라운드 내부 파이프라이닝에서 처리율과 하드웨어 복잡도 증가율간의 비교

도는 약 1.59배가 증가하였다. 이 표에서 TPC (Throughput Per Cell)은 처리율을 논리 셀(logic cell)의 수로 나눈 값으로서, 처리율과 하드웨어 복잡도(비용)간의 비율을 나타내는 척도이다.

표 2 라운드 내부 파이프라이닝의 분석 결과

파이프라인 단계	1	2	3
입/출력 핀의 수	68	68	68
논리 셀의 수	4685	5632	6567
암호화 시간[us]	42.8	25.7	17.2
처리율[Mbit/s]	7.5	12.1	17.7
TPC[Bit/Cell]	1598	2142	2693
속도 향상	1	1.31	2.36
복잡도 증가율	1	1.20	1.40

또한, 4-단계 라운드 내부 파이프라인에 대하여 인터페이스 폭을 32 비트에서 128 비트로 확장하여 추가적으로 실험한 결과를 보면, 처리율이 23.5 [Mbits/s]에서 38.89 [Mbits/s]까지 향상되어 약 1.65 배의 처리율 향상을 얻을 수 있었으며, 이때 하드웨어 복잡도는 약 1.16 배만 증가하였다.

4.4 라운드 외부 파이프라이닝

이 구현 방법도 루프 언롤링과 마찬가지로 본 연구에서 사용한 시뮬레이션 도구가 합성할 수 없는 하드웨어 크기를 가지기 때문에, 각 모듈별로 나누어서 시뮬레이션을 한 후에 산술적으로 실험 결과를 얻었다. 실험 결과(그림 16)를 보면, 루프 언롤링과는 달리 파이프라인 단계 수가 증가할수록 처리율이 두드러지게 증가하였지만, 하드웨어 복잡도가 유사한 비율로 크게 증가하기 때문에 실제 FPGA로 구현하기는 어렵다.

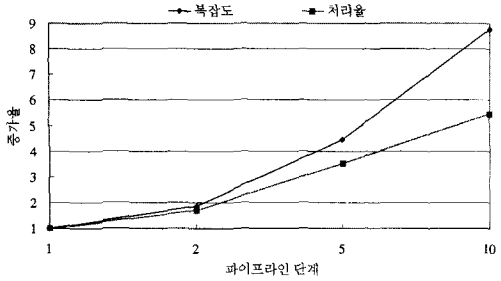


그림 16 라운드 외부 파이프라이닝에서 처리율과 하드웨어 복잡도 증가율간의 비교

4.5 S-box의 자원 공유

이 분석에서는 FPGA의 EAB에 S-box를 1개, 2개, 4개, 8개 혹은 16개를 넣은 상태에서 각각 실험을 수행하여 그림 17과 같은 결과를 얻었다. S-box는 키 생성과 ByteSub 변환에서 사용되는 데, S-box의 수가 많아질수록 S-box를 이용하여 데이타 블록을 처리하는 시간이 짧아지기 때문에 결과적으로 처리 시간이 단축된다. 하드웨어 복잡도는 S-box가 FPGA의 EAB에 의해 구현되기 때문에 별다른 증가를 보이지 않았다. S-box의 자원 공유 방법은 앞의 4.1절부터 4.4절까지의 모든 방법에 병행하여 사용될 수 있다.

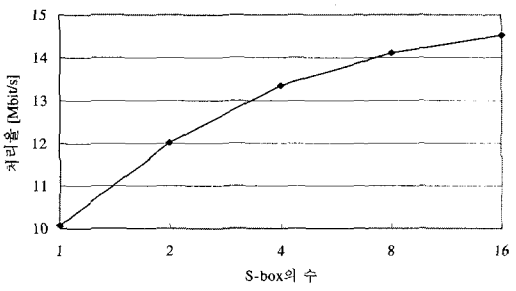


그림 17 S-box 자원 공유에 따른 처리율

4.6 구현 방법들에 대한 성능 비교

그림 18은 위의 분석에서 얻은 결과들 중에서 각 구현 방법 중 가장 효율적인 경우들에 대한 성능대가격비를 나타내는 TPC를 비교해 보여주고 있다. 그림에서 "Basic"은 128 비트의 인터페이스 폭을 가지는 기본 구조, "LU-10"은 k=10인 루프 언롤링, "IP-4"는 4-단계 라운드 내부 파이프라이닝, 그리고 "OP-10"은 10-단계 라운드 외부 파이프라이닝을 나타낸다. 결과에 따르면, 4-단계 라운드 내부 파이프라이닝(IP) 구현의 TPC가 가장 높은 것으로 나타난 반면에, 루프 언롤링(LU) 방법이 가장 낮은 것으로 나타났다. 즉, 그림 18에서 보는 바와 같이 IP-4 구현의 TPC가 OP-10 구현에 비하여 40%, 기본 구조보다는 73% 더 높게 나타났다. LU-10 구현의 경우에는 처리율이 기본 구조와 유사하였으나, 하드웨어 복잡도가 매우 높아지기 때문에 TPC가 그 절반에도 미치지 못하였다. 이러한 결과를 통하여, 기본 클록 주기가 길어지는 큰 조합회로를 구현하는 것보다는 파이프라이닝을 함으로써 기본 클록이 짧게 동작하도록 작은 조합회로로 구현하면 더 높은 성능을 얻을 수 있다는 것을 알 수 있었다. 이러한 성능 향상은 어떤 종류의 칩에서 구현하더라도 같은 비율로 나타날 수 있다.

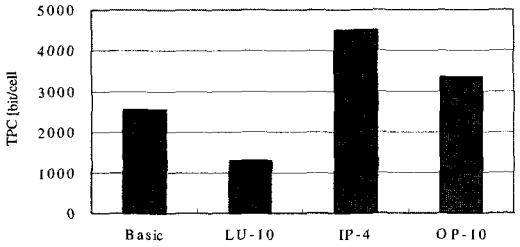


그림 18 구현 방법에 따른 TPC의 비교

5. 결 론

본 연구에서는 AES 암호화 알고리즘을 Altera FLEX10KE 계열의 FPGA로 구현하는 여러 가지 방법들에 대하여 VHDL를 이용하여 설계하고, 성능대가격비를 측정하여 비교 분석하였다. AES 알고리즘은 최근에 미국표준기술연구소(NIST)에 의해 암호화 표준 알고리즘으로 채택되었기 때문에 아직 구현 사례에 대한 발표가 거의 없는 실정이다. 따라서 본 연구에서는 표준 선정 과정에서 후보 알고리즘들에 대한 하드웨어 구현을 시도하여 비교한 연구들[6,7]의 결과와 비교하였다. 그 연구들에서는 기본 구조를 중심으로 분석하였으나, 본 연구에서는 라운드 내부 파이프라이닝 및 라운드 외부 파이프라이닝 등 다양한 구조들을 모두 구현하여 성능을 비교하였다.

분석 결과에 따르면, 4-단계 라운드 내부 파이프라이닝 구현 방법이 성능대가격비를 나타내는 TPC 면에서 다른 방법들에 비하여 40% 이상 높아 가장 우수한 것으로 나타난 반면에, 루프 언롤링 방법은 가장 뒤떨어지는 것으로 나타났다. 이러한 결과를 통하여, 기본 클록 주기가 길어지는 큰 조합회로를 구현하는 것보다는 기

본 클록이 짧게 동작하도록 작은 조합회로를 이용하여 파이프라이닝으로 구현하면 더 높은 성능을 얻을 수 있다는 것을 알 수 있었다. 또한 기존의 유사 연구들에서는 데이타와 키 블록의 인터페이스 폭을 각각 128 비트로 고정시켰는데, 그 경우에는 구현되는 칩의 핀 수가 384개로 매우 많아진다. 그러나 실제 AES 응용에 있어서 그와 같이 많은 수의 핀을 가진 칩을 사용하면 주변 하드웨어의 복잡도가 높아지는 문제가 있다. 따라서 본 연구에서는 인터페이스 폭을 8 비트부터 128 비트까지 다양하게 변화시키면서 처리율을 비교하고, 가장 좋은 결과를 보여준 32 비트 폭을 선택하여 모든 실험들을 수행하였다. 그리고 FPGA의 EAB를 이용하여 여러 개의 S-box들을 구현함으로써 병렬처리를 통한 처리시간 단축 효과를 얻을 수 있었다. 이와 같이 본 연구에서는 AES 알고리즘을 FPGA로 구현하는 경우에 사용할 수 있는 여러 가지 구현 방법들에 대한 분석 결과를 다양하게 제공하였으며, 이들은 원하는 성능대가격비에 따라 적절한 구현 방법을 결정하기 위한 기초 자료로 유용하게 사용될 수 있을 것이다.

참 고 문 헌

[1] National Bureau of Standards. FIPS PUB 46: Data Encryption Standard, January 1997.

[2] J. Daemon and V. Rijmen, AES Proposal: Rijndael, http://csrc.nist.gov/publications/drafts/dfips-AES.pdf

[3] National Institute of Standards and Technology. FIPS PUB: Advanced Encryption Standard (AES), March 2001.

[4] J. Kaps and C. Paar, "Fast DES Implementation for FPGAs," Proc. of Annual Workshop on SAC, Vol. LNCS 1556, Springer-Verlag, August 1998.

[5] Kazumaro Aoki and Helger Lipma, "Fast Implementations of AES Candidates," Proceedings of the third Advanced Encryption Standard (AES3) Candidate Conference, April 2000.

[6] B. Weeks, et al., "Hardware Performance Simulations of Round 2 Advanced Encryption Standard Algorithms," National Security Agency white paper, May 2000.

[7] A. J. Elbirt, et al.., "An FPGA-Based Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists," IEEE Transactions on VLSI Systems, Vol 9, No. 4, August 2001.

[8] K. Gaj and P. Chodowiec, "Comparison of the hardware performance of the AES candidates using reconfigurable hardware," in proc. 3rd Advanced Encryption Standard Candidate Conference, New York, Apr. 13-14, 2000, pp. 40-54. National Inst. of Standard and Technology.

김 방 현

1996년 연세대학교 전산학과 졸업. 2001년 연세대학교 전산학과 석사학위 취득 2002년 ~ 현재 연세대학교 전산학과 박사과정. 관심분야는 정보보안, RTOS, 병렬처리

김 태 규

2002년 연세대학교 전산학과 졸업. 2002년 ~ 현재 연세대학교 전산학과 석사과정. 관심분야는 정보보안, 병렬처리

김 종 현

1976년 연세대학교 전기공학과 졸업 1981년 연세대학교 대학원 전기공학과 석사학위 취득. 1988년 Arizona State University 전기 및 컴퓨터공학과 박사학위 취득. 1976년 ~ 1982년 국방과학연구소 연구원. 1988년 ~ 1990년 한국전자통신연구소 실장 역임. 1990년~현재 연세대학교 전산학과 교수. 1996년 Oregon State University 전산학과 방문교수. 관심분야는 컴퓨터구조, 병렬처리, 정보보안, 실시간 시스템 등