논문 2012-07-15

단일 데이터패스 구조에 기반한 AES 암호화 및 복호화 엔진의 효율적인 통합설계

(Efficient Integrated Design of AES Crypto Engine Based on Unified Data-Path Architecture)

정 찬 복, 문 용 호*

(Chan-Bok Jeong, Yong-Ho Moon)

Abstract : An integrated crypto engine for encryption and decryption of AES algorithm based on unified data-path architecture is efficiently designed and implemented in this paper. In order to unify the design of encryption and decryption, internal steps in single round is adjusted so as to operate with columns after row operation is completed and efficient method for a buffer is developed to simplify the Shift Rows operation. Also, only one S-box is used for both key expansion and crypto operation and Key-Box saving expended key is introduced provide the key required in encryption and decryption. The functional simulation based on ModelSim simulator shows that 164 clocks are required to process the data of 128bits in the proposed engine. In addition, the proposed engine is implemented with 6,801 gates by using Xilinx Synthesizer. This demonstrate that 40% gates savings is achieved in the proposed engine, compared to individual designs of encryption and decryption engine.

Keywords : AES, Cryptography, Security, Verilog, Xilinx Synthesis

Ⅰ. 서론

정보통신기술의 발달은 기술적 측면 뿐만 아니라 사회적 측면에서도 많은 발전을 이루었다. 이러한 발전은 우리의 생활을 좀 더 편리하게 해주는 동시에 많은 긍정적 효과를 가져오고 있다. 하지만 정보통신의 발달은 정보의 유출, 위·변조를 증가시키는 부정적인 효과를 가져 온 것이 사실이다. 이 같은 현상은 모바일 기기, 항공기, 자동차 등등 정보처리가 필수적인 시스템에서 심각한 문제를 야기한다 [1]. 이러한 보안문제를 해결하기 위해 많은 방법이 제시되어져 왔다. 그 중 암호화를 통한 정보보안은 보안문제 해결을 위해 가장 보편적으로 사용되고 있는 방법이다 [2, 3].

AES(Advanced Encryption Standard) 알고리즘 [4]은 암호화 방법의 대표적인 알고리즘으로 다양한 분야에서 널리 사용되고 있다. AES는 기존의 대칭키 암호화 알고리즘인 DES(Data Encryption Standard) [5]의 취약점을 해결하고자 2001년 11월 미연방표준(FIPS 197)으로 선정되었다 [4]. AES 알고리즘은 소프트웨어는 물론 하드웨어로도 구현 가능하다. 소프트웨어 구현은 다른 시스템간의 이식성이 좋고 구현이 간단하며 유연하지만 동작속도가 느리며, 키의 노출 가능성이 높고 물리적 안정성이 보장되지 않는다는 단점이 있다. 반면 하드웨어로 구현할 경우 구현이 복잡하나 외부 침입에 의한 알고리즘의 노출 가능성이 매우 낮으며, 암호키 노출 및 조작 가능성이 낮아 물리적 안정성이 보장된다 [6].

일반적으로 AES 알고리즘은 암호화와 복호화의 구조가 동일하지 않기 때문에 암호화 및 복호화가 개별적으로 설계된다. 따라서 암호화와 복호화가 동

* 교신저자(Corresponding Author)

논문접수 : 2012. 01. 13., 수정일: 2012. 03. 06.
채택확정 : 2012. 04. 06.

정찬복, 문용호 : 경상대학교 정보과학과

※ 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 육성 지원사업의 연구결과로 수행되었음 (NIPA-2012-H0301-12-3003)

ISSN : 1975-5066

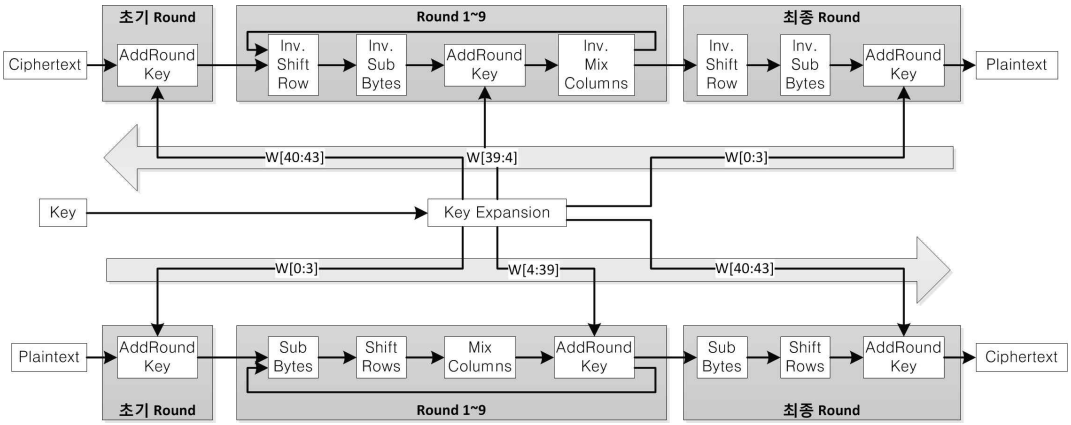

그림 1. 기존 AES 암호화 및 복호화 구성
Fig. 1. Organization of encryption and decryption for AES algorithm

시에 요구되는 모바일, 임베디드, 통신분야등에 있어서 AES 방식의 적용은 하드웨어 비용의 증가를 수반한다. 즉, 하드웨어의 비용 및 면적이 증가하고 구성이 복잡해 질 수 있다. 일반적으로 하드웨어의 비용과 면적 증가는 저전력, 초소형화의 추세에 있어서 극복해야 할 장애물이다. 이에 본 논문에서는 AES 알고리즘의 암호화 및 복호화 엔진을 단일 데이터패스 구조에 기반하여 보다 효율적으로 통합 설계하고 이를 FPGA로 구현한다.

Ⅱ. AES 알고리즘 개요

AES 알고리즘은 블록 암호화 방식으로 128bit 데이터가 초기 Round, Round 1~9, 최종 Round로 이루어지는 총 11번의 Round과정을 통하여 암호화 된다. 그리고 128bit의 입력 Key는 각 Round별로 확장되어 암호화 및 복호화에 사용된다. 그림 1은 이 같은 과정을 나타낸다. 그림 2는 AES의 Round 내부 구성을 개념적으로 나타낸 것이다.

- Sub Bytes
  그림 2-(a)와 같이 데이터를 Byte단위로 변환하는 기능을 수행한다. 암호화는 S-Box를, 복호화는 Inverse S-Box를 이용하여 Byte단위의 Data 치환이 이루어진다.
- Shift Rows
  그림 2-(b)와 같이 데이터를 행단위로 왼쪽(암호화) 또는 오른쪽(복호화)으로 순차적으로 이동시켜 데이터를 혼합하는 과정이다.
- Mix Columns
  그림 2-(c)와 같이 Galoa Field의 곱셈연산[7]을 통해 열안의 각 bit를 혼합하는 과정이다. 암호화에서는 {2,3,1,1}, 복호화에서는 {E,B,D,9}의 계수들이 이용된다.
- Add Round Key
  그림 2-(d)와 같이 Key Expansion을 통해 확장되어진 키와 입력 데이터를 32bit 블록별로 나누어 XOR연산을 수행한다.
- Key Expansion
  각 Round에 쓰일 Key 값을 확장하는 과정으로 한 번 확장된 Key는 입력된 Key 값이 바뀌지 않는 한 계속 유지 된다.

Ⅲ. 제안하는 AES Crypto 엔진 설계

AES 알고리즘의 암호화 및 복호화에 대한 단일 데이터 패스를 설계하기 위해서는 그림 1에서 제시된 서로 다른 Round 내부구조가 통일되어야 한다. 그리고 확장된 Key를 Round에 제공하기 위해서는 별도의 Key 저장 공간이 필요하다.

1. Round내부 순서 변경을 통한 단일 데이터 패스 설계

암호화와 복호화를 단일 데이터패스에서 동작하게 하려면 무엇보다도 먼저 Round의 내부구조가 통일되어야 한다. 본 논문에서는 다음과 같은 특성을 이용하여 Round의 순서를 그림 3의 (b)와 같이 통

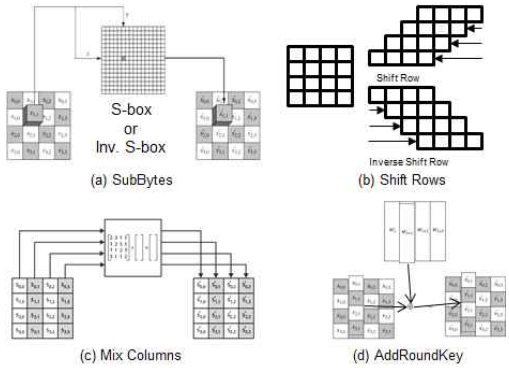

그림 2. AES 알고리즘의 단일 Round 구성요소 [7]
Fig. 2. Key elements composing of single round in AES algorithm

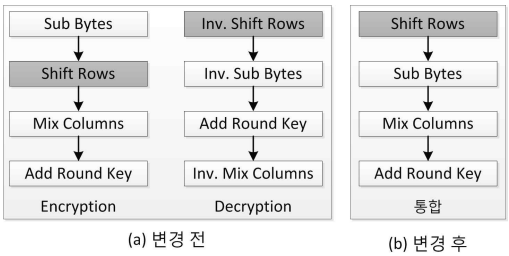

그림 3. 암호화 및 복호화를 위한 Round 내부 순서 변경
Fig. 3. Modification of internal steps in single round for integrating encryption and encryption in AES algorithm

일하였다.

그림 2와 3의 (a)를 보면 암호화 및 복호화의 Sub Bytes와 Shift Rows의 순서가 서로 다른 것을 확인할 수 있다. 단일 데이터 패스 설계를 위해서는 이 순서를 통일해야 하는데 이 두 과정은 어떤 것이 먼저 처리되어도 항상 동일한 결과를 가져 온다. 왜냐하면 Byte치환이 Shift Rows의 행 이동의 순서에 영향을 주지 않기 때문이다. 마찬가지로 Shift Rows에 의한 행 이동이 Sub Bytes에서의 Byte 치환에 영향을 미치지 않는다.

Shift Rows에서 요구되는 액세스가 가능하도록 Buffer를 제어한다면 Sub Bytes는 물론 Mix Columns과 Add Round Key의 열 단위 연산에 알맞은 순서로 값을 출력할 수 있다. Shift Rows를 위한 buffer 액세스는 그림 4와 같이 표현될 수 있다. 만약 암호화를 한다면 Buffer의 Index 15, 10, 5, 0 순으로 값을 가져오게 된다. 위의 순서로 값을 가져오기 위해 i와 j, 그리고 Round Count를 사용한다. i는 행을 j는 열을 가리킨다. Round Count는 하나의 Round가 처리되는데 걸리는 Clock을 Count하며 i와 j의 값을 만들어 내는데 사용하며 15부터 매 Clock마다 1씩 감소한다. i는 Round Count를 4로 나눈 나머지 값으로 행을 가리키며, j는 Round Count를 4로 나눈 나머지 값이 3일 경우 그 몫을 취하고 그 외에는 감소(암호화), 증가(복호화)를 수행한다. 이렇게 생성된 i와 j를 4bit의 Index에 삽입(Index[j:i])하면 해당 인덱스의 값을 Shift Rows연산에 맞게 가져올 수 있다.

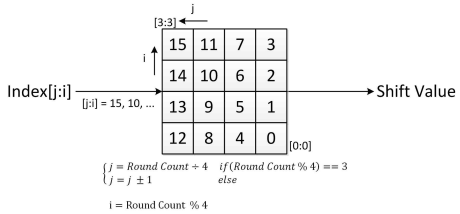

그림 4. Shift Rows를 위한 buffer 액세스
Fig. 4 Accessing a buffer for shift rows

그림 3의 (a)에서 볼 수 있듯이 Mix Columns과 Add Round Key 또한 암호화와 복호화의 순서가 다르다. 그러나 암호화와 같은 순서로 동작하게 할 수 있다. 이를 위해서는 Key Expansion에 확장된 Key만을 위한 별도의 Mix Columns이 추가적으로 삽입되어야 한다. 즉, 암호화 되고 있는 128bit의 데이터와 확장된 키값 각각에 Mix Columns을 적용시킬 수 있다면 Mix Columns과 Add Round Key의 순서를 동일하게 통일 시킬 수 있다.

2. Key-Box 설치 및 단일 S-Box 사용

암호화와 복호화에 있어서 확장된 키의 사용 순서가 동일하지 않다. 이를 해결하기 위해서는 Key Expansion과정에서 확장된 Key를 별도의 공간에 저장하는 것이 필요하다. 이 별도의 공간은 암호화를 수행하느냐, 복호화를 수행하느냐에 따라 저장되는 내용이 달라진다. 즉, 암호화 및 복호화 각각의 순서에 맞게 확장된 Key를 Key-Box에 저장해 두고 Key가 필요할 때 반환한다. 또한 Key-Box는 Key Expansion의 수행을 최소화하기 위해서 필수적이다.

S-Box는 256개의 Byte가 저장되어 있는 큰 공간을 필요로 한다. S-Box의 사용을 최소화 하는 것은 하드웨어의 면적을 줄일 수 있는 기초적인 방법

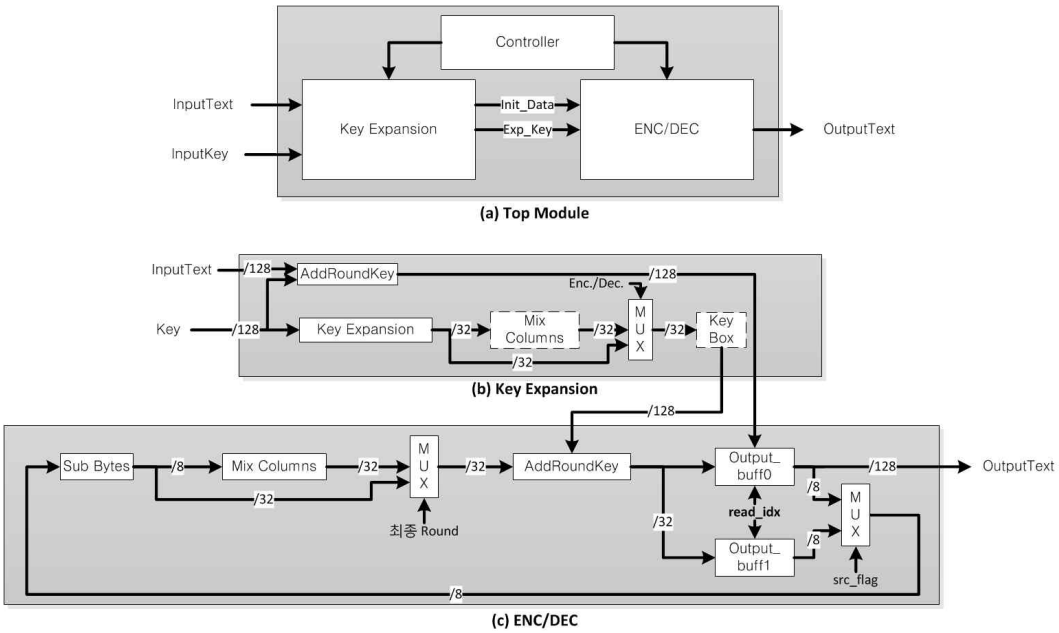

그림 5. 제안하는 AES Crypto Engine의 Data Path
Fig. 5. Data path of the proposed AES crypto engine

으로 Key Expansion의 수행 후 암호화 및 복호화가 수행되도록 함으로 써 S-Box를 하나만 사용한다.

IV. 제안하는 AES Crypto 엔진 구현

1. AES Crypto 엔진 구현

그림 5는 본 논문에서 제안하는 통합 AES Crypto Engine의 구조를 보여준다. 그림 5-(a)는 Top Module을 나타낸 것으로 키 확장을 위한 Key Expansion과 암호화 및 복호화를 수행하는 ENC/DEC Module, 그리고 이들을 제어하는 Control Module로 구성되어 있다.

그림 5-(b)는 Key Expansion Module로 암호화 및 복호화의 초기 Round에서 요구되는 Add Round Key 단계를 포함하고 있다. 따라서 초기 Round로부터 생성된 결과가 ENC/DEC Module로 전달된다. 그리고 Key Expansion후에 적용되는 Key Expansion Module의 Mix Columns은 복호화 시에 사용되는 복호화 Key를 생성한다. 이것은 단일 데이터 패스 구조의 Mix Columns과 Add Round Key의 순서변경으로 인하여 요구되는 추가적인 연산을 보상하기 위한 것이다. 이렇게 확장된 Key는 Key-Box에 저장되어 각 Round마다 요구되는 필요한 Key가 전달된다.

그림 5-(c)는 ENC/DEC Module로 암호화 및 복호화가 이루어지는 모듈이다. 그림3에서 제시된 바와 같이 Shift Rows, Sub Bytes, Mix Columns,

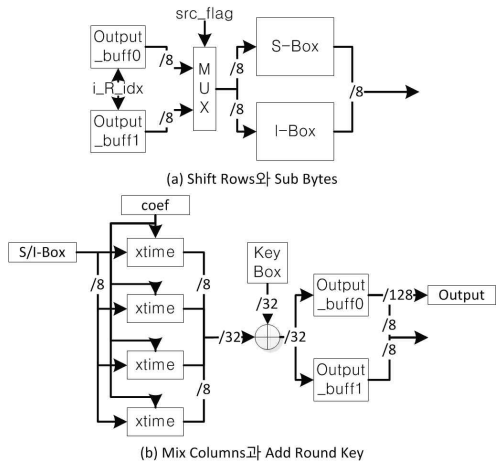

그림 6. Round 내부 Data Path
Fig. 6. Data path in round

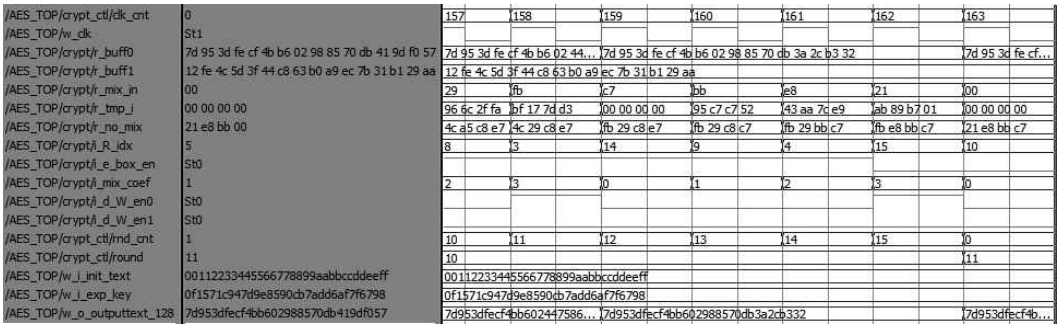

그림 7. ModelSim 시뮬레이터에 의한 기능적 시뮬레이션 결과
Fig. 7. Functional simulation results based on ModelSim simulator

Add Round Key의 순서로 연산이 수행된다.

2. Round 내부 연산 구현

그림 6은 Round 내부의 연산을 구현한 Data Path이다. 그림 6-(a)는 Shift Rows와 Sub Bytes 동작을 수행하는 구조도이다. 먼저 Round에 맞는 Output_buff가 선택된다. 선택된 buffer에서는 매 Clock마다 1Byte씩 치환되어져야 하는 값이 출력된다. 이 때 Shift Rows 순서에 맞도록 Index값이 i_R_idx를 통하여 Buffer로 입력되어 특정 Byte가 출력된다. 이렇게 주어지는 값은 암호화 및 복호화 단계에 따라 S-Box 또는 I-Box의 입력주소로 동작하여 해당되는 Byte가 출력됨으로써 Sub Bytes가 이루어지게 된다.

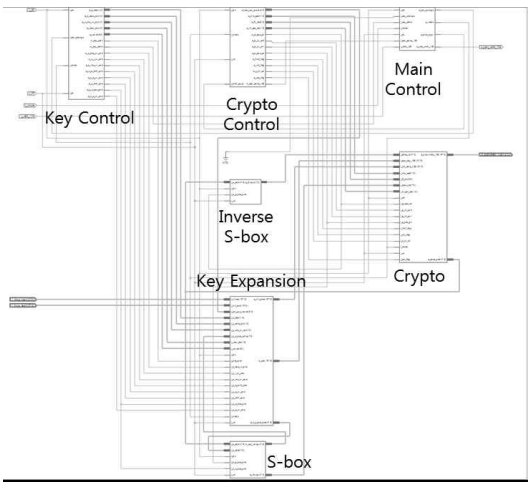

그림 8. 제안된 Engine의 FPGA Systhesis 결과
Fig. 8. FPGA systhesis results for the proposed engine

그림 6-(b)는 Mix Columns과 Add Round Key의 연산과정이다. 우선 Mix Columns은 S-Box 또는 I-Box의 결과 값을 입력받고 암호화 계수(2,3,1,1) 또는 복호화 계수(E,B,D,9)에 따라 Galoa Field 곱셈연산을 하게 되며 계산의 결과는 4X4의 행렬 곱셈이 된다. Mix Columns로부터 얻어진 결과는 Add Round Key연산을 통하여 32Bit단위로 확장된 Key와 XOR연산을 수행한다. Add Round Key연산이 완료되면 최종 결과는 buffer에 저장되어 다음 Round를 위한 입력으로 사용된다.

이 때 Buffer가 하나일 경우, 현재 Buffer에 저장된 데이터는 Shift Rows순서로 출력되고 있기 때문에 최종결과를 Buffer에 저장하면 처리되지 않는 유효데이터가 변질될 수 있다. 이를 방지하기 위해 계산된 결과가 저장되는 Buffer와 Shift Rows Index순으로 액세스되는 Buffer를 분리하여 동작하게 하였다.

Ⅴ. FPGA 구현 및 검증

본 논문에서는 그림 5와 같이 설계된 Engine을 Verilog로 구현하고 그 동작을 ModelSim 시뮬레이터를 이용하여 검증하였다. 하나의 Round 동작을 위해서는 16Clock이 소요 된다. Shift Rows와 Sub Bytes, Mix Columns 연산이 매 Clock마다 동작하게 하였다. 이는 암호화와 복호화 모두 동일하다.

그림 7은 ModelSim 시뮬레이션에서 얻어진 결과로써 예상한 타이밍 다이어그램과 동일한 결과가 얻어짐을 확인할 수 있었다. 그리고 128bit의 암호화된 결과 값이 나오는데 164Clock이 소요됨을 확인할 수 있었다. 이 또한 암호화와 복호화 모두 동일하며 암호화와 복호화를 동시에 수행 할 수는 없기 때문에 위 두 과정을 차례로 수행시 328Clock이

소요된다.

Verilog로 구현된 Engine을 Xilinx ISE 9.1i Simulator, Xilinx Spartan3 XC3S2000 Chipset을 사용하여 합성하였다. 그림 8은 Xilinx ISE 9.1i Simulator를 이용한 합성의 결과로 각 Module을 제어하기 위한 Control Module과 ENC/DEC Module, Key Expansion Module이 정상적으로 합성되는 것을 확인하였다.

표 1은 합성시 생성된 Module별 Unit의 개수와 Gate의 개수를 정리한 것이다. 동작 Clock은 100MHz로 설정하였다. 통합 구현의 효과를 나타내기 위해 암호화 및 복호화 통합 설계, 암호화 단일, 복호화 단일 설계 각각에서 소요되는 Unit 및 Gate 개수를 정리하였으며 표의 마지막행에 합성에서 사용된 Unit 및 Gate 개수의 합을 정리하였다. 표 1을 보면 암호화 및 복호화를 각각 구현한 경우가 통합 구현한 경우보다 약 1.7배 많은 Gate를 사용하고 있음을 알 수 있다. 즉 제안한 바와 같이 AES 알고리즘을 통합 구현하는 것이 하드웨어의 면적과 복잡도를 감소시켜 더욱 효과적이라는 것을 확인할 수 있다.

표 1. 합성시 생성 된 Unit 및 Gate Count 비교
Table 1. Comparison of the number of gates and units created from synthesis

Module	Unit	통합 설계	개별적 설계 (암호화+복호화)
control	counter	6	12(6+6)
	D-filp-flop	18	32(16+16)
	Adder/ Subtractor	2	4(2+2)
	Comparator	2	4(2+2)
Crypto	D-flip-flop	491	944(470+474)
	Multiplexer	48	96(48+48)
	Xor	352	704(352+352)
S-box	D-flip-flop	8	16(8+8)
I-box	D-flip-flop	8	8(0+8)
control key	D-filp-flop	28	49(22+27)
	Adder/ Subtractor	5	9(4+5)
	Comparator	11	20(9+11)
Expan-sion	D-flip-flop	2216	4288(2072+2216)
	Comparator	5	9(4+5)
	Multiplexer	168	304(136+168)
	Xor	40	48(8+40)
Control AES	D-filp-flop	5	8(4+4)
합계		3413	6555(3163+3392)
2-Input NAND Gate		6801	11313(5614+5699)

Ⅵ. 결론

본 논문에서는 단일 데이터 패스 구조에 기반한 AES Crypto Engine을 설계하였다. 이를 위해 AES의 암호화 및 복호화의 Round내부 구조를 동일하게 하였으며, Index를 효과적으로 생성하여 효과적으로 Shift Rows를 구현하였다. 그리고 Key 사용 순서를 고려한 Key-Box를 설치하였고 S-Box의 사용을 최소화 하여 하드웨어의 비용을 감소시켰다. 시뮬레이션 및 합성결과 최대 동작 가능 주파수는 200MHz이었으며, 128bit의 Data를 암호화 하는데 각 Round별로 16Clock이 소요 되었고, 총 164Clock이 소요 되는 것을 확인할 수 있었다. 또한 통합 구현 시 생성된 6801개의 Gate는 개별적 구현의 60%에 해당하는 것으로 약 40%의 하드웨어 감소효과를 가져왔다.

참고문헌

[1] H. JingDe, S. Gang, "Substitution Encryption Algorithm Study for Embedded Mobile Code Protection," Proceedings on International Conference on Communication Software and Networks, pp.645-649, 2009.

[2] 조승훈, 서정배, 문용호, "항공기에서 보안 강화된 음성 데이터 저장 방식," 대한임베디드공학회 논문지 Vol. 6, No. 4, pp.255-261, 2011.

[3] [3] 호정일, 이강, 조윤석, "저비용 FPGA를 이용한 AES 암호프로세서 설계 및 구현," 한국정보과학회 학술발표논문집, Vol. 31, No. 1, pp. 934-936, 2004.

[4] US National Institute of Standards and Technology, "Advanced Encryption Standard," Federal Information Processing Standards Publication 197, pp.1-47, 2001.

[5] US National Institute of Standards and Technology, "Data Encryption Standard," Federal Information Processing Standards Publication 46-3, pp.1-22, 1999.

[6] 최병윤, 박영수, 전성의, "차세대 대칭키 암호 알고리즘 AES의 하드웨어 구현 기술," 한국통신학회지, Vol. 19, No. 8, pp. 1262-1272, 2002.

[7] B.A. Forouzan, “암호학과 네트워크 보안,” McGraw-Hill Korea, 서울, 2008.

저 자 소 개

정 찬 복

2012년 경상대 정보과학과 학사.
현재, 경상대 일반대학원 정보과학과 석사 과정
관심분야: 임베디드 소프트웨어, 코드압축, SoC
관심분야: 정보보안, 보안프로토콜, 암호 H/W
Email: nirk@paran.com

문 용 호

1992년 부산대 전자공학과 학사.
1994년 부산대 일반대학원 전자공학과 석사.
1998년 부산대 일반대학원 전자공학과 박사.
1998년~2001년 삼성전자 DM연구소 책임연구원.
현재, 경상대 정보과학과 부교수.
관심분야: 동영상압축, 영상처리, 임베디드 시스템, SoC
Email: yhmoon5@gnu.ac.kr