ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

경량 블록암호 LEA용 암·복호화 IP 설계☆

Design of Encryption/Decryption IP for Lightweight Encryption LEA

손 승 일1*
Seungil Sonh

요 약

LEA(Lightweight Encryption Algorithm)는 2013년 국가보안연구소(NSRI)에서 빅데이터 처리, 클라우드 서비스 및 모바일 환경에 적합하도록 개발되었다. LEA는 128비트 메시지 블록 크기와 128비트, 192비트 및 256비트 키(Key)에 대한 암호화 방식을 규정하고 있다. 본 논문에서는 128비트 메시지를 암호화하고 복호화할 수 있는 LEA 블록 암호 알고리즘을 Verilog-HDL을 사용하여 설계하였다. 설계된 LEA 암.복호화 IP는 Xilinx Vertex5 디바이에서 약 164MHz에서 동작하였다. 128비트 키 모드에서 최대 처리율은 874Mbps이며, 192비트 키 모드에서는 749Mbps 그리고 256비트 키 모드에서는 656Mbps이다. 본 논문에서 설계된 암호 프로세서 IP는 스마트 카드, 인터넷 뱅킹, 전자상거래 및 IoT (Internet of Things) 등과 같은 모바일 분야의 보안 모듈로 응용이 가능할 것으로 사료된다.

☞ 주제어 : LEA, 대칭형블록암호, 암호화, 복호화, HDL설계

ABSTRACT

Lightweight Encryption Algorithm(LEA) was developed by National Security Research Institute(NSRI) in 2013 and targeted to be suitable for environments for big data processing, cloud service, and mobile. LEA specifies the 128-bit message block size and 128-, 192-, and 256-bit key sizes. In this paper, block cipher LEA algorithm which can encrypt and decrypt 128-bit messages is designed using Verilog-HDL. The designed IP for encryption and decryption has a maximum throughput of 874Mbps in 128-bit key mode and that of 749Mbps in 192 and 656Mbps in 256-bit key modes on Xilinx Vertex5. The cryptographic IP of this paper is applicable as security module of the mobile areas such as smart card, internet banking, e-commerce and IoT.

☞ keyword : LEA(Lightweight Encryption Algorithm), Symmetric Block Cipher, Encryption, Decryption, HDL Design

1. 서 론

암호는 과거에는 군사적인 용도 등의 비밀 통신을 위해 주로 사용하였지만, 오늘날은 인터넷 기반의 사회, 경제 활동의 안정성, 신뢰성, 사생활 보호 등을 위한 핵심 기술로서 메일전송, 사용자 인증, 전자상거래, 인터넷 뱅킹 등에서 다양하게 사용되고 있다[1,2]. 또한 IoT(Internet of Things) 서비스의 발전은 스마트 기기, 센서 등 다양한 단말 및 이종 네트워크, 애플리케이션을 활용하므로, 발생 가능한 보안 위협도 한층 증가할 것으로 예측되고 있다[3].

대칭형 암호 시스템은 암호화를 위한 키와 복호화를 위한 키가 동일한 시스템으로 DES, IDEA, SKIPJACK, MISTY, Camellia 및 AES(Advanced Encryption Standard) 알고리즘 등 다양하게 발표되었다[4].

우리나라에서 주관하여 개발된 대칭형 블록 암호는 SEED, ARIA, HIGHT 및 LEA가 있으며, 미국 NIST에서 주관한 AES 암호는 세계적으로 널리 알려진 암호 알고리즘이다[2-7]. 특히 AES 블록 암호 알고리즘은 강력한 보안 성능과 다양한 암호 운영 모드의 장점으로 인해 무선 랜(Wireless LAN), Zigbee 등 다양한 유·무선 통신 시스템의 보안 알고리즘으로 폭 넓게 활용되고 있는 것으로 알려졌다[3,7].

본 논문에서 설계할 LEA 블록 암호는 국내의 국가보안기술연구소(NSRI)에서 2013년도에 개발한 128비트 경량 블록 암호 알고리즘으로 라운드 함수는 32비트 단위의 ARX(Addition, Rotation, XOR) 연산만으로 구성하여, 이를 지원하는 범용 32비트 소프트웨어 플랫폼에서 고속으로 동작한다. 또한 라운드 함수 내부의 ARX 연산 배치는 충분한 안정성을 보장하는 것과 동시에 S-box의 사용을 배제하여 경량 구현이 가능하도록 하였다[8]. 아울러

1 Division of Information and Telecommunications, Hanshin Univ., Osan 447-791, Korea
* Corresponding author (saisonh@hs.ac.kr)
[Received 18 Feburary 2017, Reviewed 25 July 2017, Accepted 28 August 2017]
☆ 본 연구는 한신대학교 학술연구비 지원에 의하여 연구되었음.

LEA는 128비트 데이터 블록을 암호화하는 알고리즘으로 128, 192, 256비트 비밀키(Secret key)를 사용하는 것을 규정하고 있으며, 키의 사용은 요구되는 안정성 기준에 따라 사용자가 선택할 수 있도록 하였다[8].

특히 본 논문의 LEA 암호 알고리즘은 소프웨어 및 IoT 분야에 적용이 확대되고 있다. 이스트소프트는 압축 SW인 “알집(버전 10.5)”에 암호화 기능으로 적용하였다. 또한 스마트그리드 지능형 전력계량(AMI) 사업에도 LEA를 채용하였으며, 이니텍사는 IoT 보안 플랫폼인 ISoT(Internet Security of Things)에 채용하였다[9].

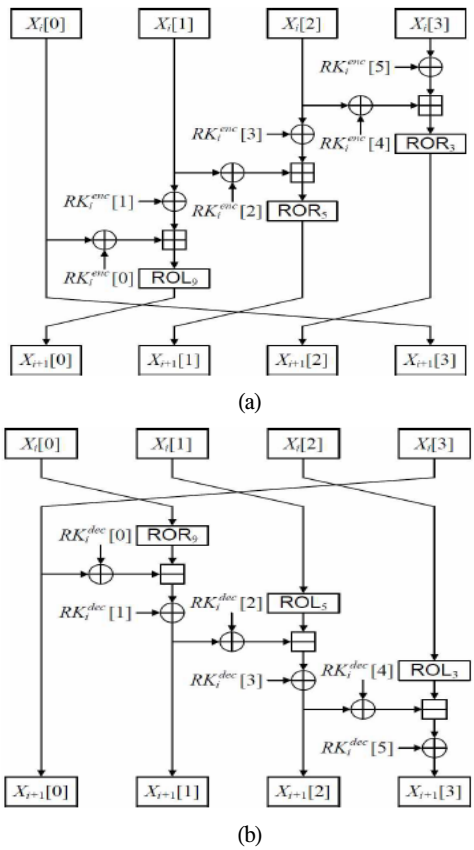
본 논문은 다음과 같이 구성된다. 2장에서는 LEA 암호 알고리즘의 개념에 대해 소개하고, 3장에서는 본 논문에서 제안하는 LEA 블록 암호의 설계에 대해 다루며, 4장에서는 LEA 암호의 성능 향상을 위한 설계 방법을 다루며, 5장에서는 설계된 LEA 블록 암호의 검증 및 성능 분석을 수행하고, 마지막으로 6장에서 결론을 맺는다.

2. LEA 암호 알고리즘의 개념[8]

LEA 암호 알고리즘은 대부분의 암호화 알고리즘과 마찬가지로 키 스케줄링 부분(Key Scheduling Part)과 데이터 랜덤화를 수행하는 라운드 연산 블록(Round Operation Block) 부분 나눌 수 있다. LEA 암호 알고리즘에 대한 설명은 기존의 논문에 자세히 언급되어 있다[3,6,8]. 핵심적인 몇 가지만 소개하고자 한다.

LEA 블록 암호 알고리즘은 앞에서 언급한 바와 같이 128비트 메시지(평문 혹운 암호문)에 대해 128, 192, 256비트 매스터 키를 사용하여 128비트의 암호문이나 평문을 도출한다. 이 때 매스터 키의 길이에 따라 라운드 연산 블록에서 수행되는 라운드 연산 횟수가 다를 수 있다. 128비트 매스터 키를 사용하면 24라운드 연산을 수행하고, 192비트 매스터 키를 사용하면 28라운드 연산을 수행하고, 256비트 매스터 키를 사용할 경우에는 32라운드 연산을 수행하게 된다.

(그림 1)의 (a)는 LEA 블록 암호에서 암호화를 위해 한 라운드 연산을 수행하는 과정을 보여준다. $X_i[0]$부터 $X_i[3]$은 각각 32비트로 된 메시지를 의미하는데, 128비트 메시지를 사용하므로 32비트로 분할된 4개가 사용된다. 그리고 i 번째 라운드에서 사용되는 라운드 키는 $RK_i[0]$부터 $RK_i[5]$까지 모두 32비트로 된 6개가 사용되어, 각 라운드 연산에 사용되는 라운드 키는 192비트가 된다. 각 라운드에서 사용되는 연산은 기존에 언급한 바와 같이 XOR, 32비트 모듈로 덧셈 및 회전 연산만을 사용하는 것을 볼 수 있다. 각 라운드의 마지막에 저장되는 중간 값이 32비트 좌측 방향 회전하여 저장된다.



(그림 1) LEA 암호의 (a)암호화와 (b)복호화에 대한 라운드 함수

(Figure 1) Round function for (a) encryption and (b) decryption of LEA cryptography

(그림 1)의 (b)는 LEA 블록 암호에서 복호화를 위한 한 라운드 연산을 수행하는 과정을 보여주고 있다. 복호화 과정은 암호화 과정의 역과정이다. 예를 들어 암호화 시 XOR 연산 후 모듈로 덧셈 연산을 하고 마지막으로 우측 회전을 하였다면, 복호화 시에는 좌측 회전 연산을 수행한 후 모듈로 뺄셈 연산 후 XOR 연산을 수행해야 한다. 또한 각 라운드 마지막에서 암호화 시에는 32비트 좌측 회전을 수행하였기 때문에, 복호화 시에는 32비트 우측 회전을 수행해야 한다.

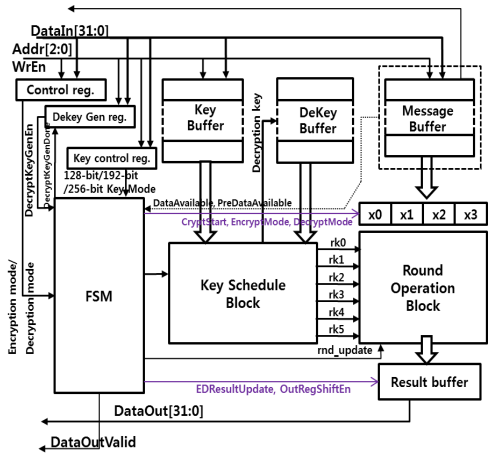
키 스케줄의 경우에는 각 라운드 마다 192비트의 라운드 키가 적용된다. 128비트의 매스터 키 모드를 사용할 경우에는 키를 확장해서 사용해야 하며, 192비트 매

스터 키 모드를 적용할 경우에는 각 라운드마다 매스터 키 전체가 사용된다. 마지막으로 256비트 매스터 키 모드를 사용할 경우에는 각 라운드 키 생성에 참여하는 매스터 키는 전체가 사용되는 것이 아니라 192비트만 사용된다. 따라서 각 라운드 키가 생성될 때마다 매스터 키 관련 내용은 암호화 시에는 64비트 좌측 회전을 한 후 다음 라운드 키를 생성하며, 복호화 시에는 64비트 우측 회전을 한 후 다음 라운드 키를 생성하도록 하여 모든 키가 공평하게 라운드 키 생성에 참여하도록 고안되었다.

마지막으로 키 스케줄 시에 라운드 키를 생성할 때 사용되는 델타(Delta) 상수 값이다. 모두 8개의 델타 값이 존재하며, 각각의 값은 δ[0] = c3efe9db, δ[1] = 44626b02, δ[2] = 79e27c8a, δ[3] = 78df30ec, δ[4] = 715ea49e, δ[5] = c785da0a, δ[6] = e04ef22a, δ[7] = e5c40957으로 정의된다. 128비트 매스터 키 모드에서는 δ[0]부터 δ[3]까지 4개의 값이 사용되며, 192비트 매스터 키 모드에서는 δ[0]부터 δ[5]까지 6개의 값이 사용되고, 256비트 매스터 키 모드에서는 δ[0]부터 δ[7]까지 8개의 값이 사용되어 라운드 키 생성에 사용된다. 앞에 언급한 델타 값은 암호화 시에 사용되는 초기 값인데, 복화시에는 위에 언급한 델타 값을 사용할 수 없다. 복호화는 암호화 과정의 역순이므로 암호화 과정의 마지막에 사용된 변경된 델타 값이 복화시에 사용되는 델타 값의 초기값이 된다. 이러한 델타 값들은 암•복호화 시에 서로 직렬로 순환 연결되어 있으며, 각 라운드마다 좌측 1비트 순환 이동 후 앞쪽으로 전달된다. 즉 좌측 1비트 회전된 δ[1] 값이 δ[0] 로 이동되고, 좌측 1비트 회전된 δ[2] 값은 δ[1] 으로 이동된다. 좌측 1비트 회전된 δ[0] 의 경우에는 δ[i]로 이동되는 데, 여기서 i 값은 128비트 매스터 키 모드에서는 i=3이며, 192비트 매스터 키 모드에서는 i=5가 되고, 256비트 매스터 키 모드에서는 i=7이 된다. 그리고 라운드 키 생성에 참여하는 델타 값은 항상 δ[0]가 되도록 하면 구현이 용이해진다. 복호화 시의 델타 값 사용은 먼저 초기 값이 다르고 매 라운드 마다 각 델타 값은 우측 1비트 회전을 수행해야 하고, 1비트 회전된 δ[0] 값은 암호화와 반대로 δ[1]으로 전달되어야 한다.

3. LEA 암호 알고리즘의 설계

(그림 2)는 본 논문에서 설계한 LEA 블록 암호 알고리즘 IP의 아키텍쳐를 보여준다. 본 논문의 LEA 블록 암호 알고리즘은 "128비트 블록암호 LEA 규격서"인 참



(그림 2) LEA 암호화/복호화 IP의 아키텍쳐

(Figure 2) Architecture of LEA block encryption/decryption IP

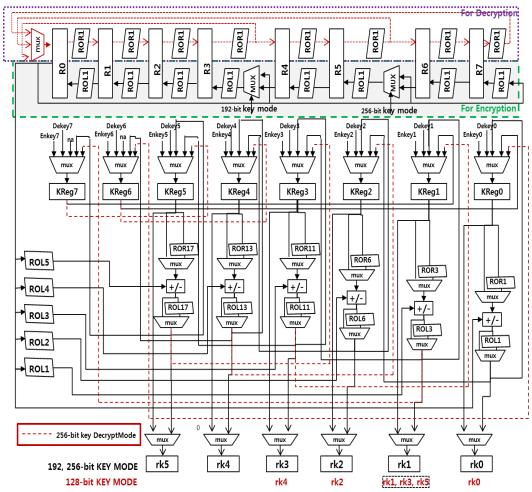
고문헌 [8]에서 규정하고 있는 암호화 및 복호화에 대한 모든 사양을 지원하도록 설계되었다. 따라서 128비트 단위의 메시지 입력에 대해 128비트, 192비트 및 256비트 매스터 키가 적용되어 암호화 혹은 복호화된 128비트 결과를 출력한다.

(그림 2)의 메시지 버퍼(Message Buffer)는 32비트로 된 4개의 레지스터로 구성되어 있는데, 라운드 연산 수행시 필요한 128비트 메시지와 32비트 외부 인터페이스 간 데이터 크기 불일치에 의한 성능 저하를 제거하기 위한 용도이다[2]. 현재 처리중인 메시지 블록 다음에 수행할 128비트 데이터를 현재 메시지 블록에 대한 암•복호화 수행동안 미리 메시지 버퍼에 버스트(Burst)로 저장한 후, 현재 수행중인 메시지 블록의 암•복호화가 완료되는 즉시 다음 번 메시지 블록을 메시지 버퍼에서 x0, x1, x2 및 x3에 동시에 제공해 줌으로써 IP의 최대 Throughput 성취를 돕는 버퍼이다.

메시지 버퍼는 키 값 및 제어 레지스터 등에 대한 설정을 완료한 후에 입력을 수신해야 하며, 4개의 완전한 입력 데이터가 수신되는 즉시 LEA 암·복호화 모듈의 설정값에 따라 암호화나 복호화를 수행하게 된다. 키 버퍼(Key Buffer)는 설정된 키 사이즈 정보에 따라 128비트, 192비트 및 256비트를 저장한다. Control 레지스터는 암호화 모드 혹은 복호화 모드를 결정하는 제어 비트를 저장하고 있다. 그리고 Key Control 레지스터는 사용할 키의 길이 모드 값을 저장한다. 128비트 매스터 키 모

드, 192비트 매스터 키 모드 혹은 256비트 매스터 키 모드 중에서 사용하고자 하는 매스터 키 모드로 레지스터를 설정하면 된다. 그림을 보면 DeKey 버퍼가 존재하는데, 이는 복호화 모드에서 사용할 키 값을 저장하고 있다. 본 IP를 사용하는 사용자가 처음부터 복호화 모드로 사용하고자 한다면, 먼저 키 스케줄 블록(Key Schedule Block)에서 암호화 키를 생성하는 과정을 수행하여 최종 라운드에서 사용되는 라운드 키 값을 생성한다. 이 값은 복호화 시에는 첫 번째 사용되는 키가 되기 때문에, DeKey 버퍼에 저장하여 복화시에 사용하도록 설계하였다. 이처럼 DeKey를 생성하여 저장할 수 있도록 설정하는 레지스터가 DeKey Gen 레지스터이다. 외부 버스를 통해 DeKey 생성을 활성화하도록 세팅하면 설계된 IP는 즉시 키 스케줄 블록을 사용해 매스터 키 모드에 해당되는 복호화용 첫 번째 키를 생성하여 DeKey 버퍼에 저정한다. 저장된 복호화용 키는 복호화시에는 자동적으로 키 스케줄 블록에 제공하여 복호화용 라운드 키를 생성하기 위해 사용된다. DeKey Gen 레지스터의 복호화 시작 키 생성 활성화 비트는 복호화 시작 키를 생성하여 DeKey 버퍼를 저장하면, 즉시 내부적으로 '0'으로 초기화되도록 설계하였다. 따라서 DeKey Gen 레지스터에 관련 레지스터 비트를 설정할 때만 1회에 한해 복호화용 키를 생성하도록 하였다. 그리고 추후 새롭게 변경된 복호화용 키를 사용할 필요가 있을 경우에는 다시 DeKey Gen 레지스터를 세팅하면 된다. DeKey Gen 레지스터에 대한 세팅은 복호화가 필요할 경우에만 사용한다. 암호화 모드로 운영하고자 할 경우에는 DeKey Gen 레지스터를 사용하지 않는다.

내부 레지스터에 대한 세팅을 종료한 이후에는 메시지 버퍼를 통해 메시지 데이터가 입력되게 된다. 32비트 버스를 통해 4개의 데이터를 수신하면 메시지 버퍼의 내용이 라운드 연산을 수행할 수 있도록 x0, x1, x2, x3 레지스터로 전달된다. 그런데 현재 라운드 연산 블록(Round Operation Block)이 암호화나 복호화를 수행중에 있을 수 있기 때문에 FSM(Finite State Machine)은 새로운 메시지를 x0, x1, x2, x3에 수신할 수 있을 때 CryptoStart 신호를 생성하여 x0~x3 레지스터에 저장하도록 설계하였다. 메시지 버퍼가 3번째 32비트 데이터가 수신되면 PreDataAvailable 신호를 FSM에 전달하여 키 스케줄 블록이 라운드 키를 생성하기 위한 초기 값 설정을 하도록 구현하였는데, 이는 라운드 키를 라운드 연산 블록에 전달할 때 키 스케줄 블록에서 수행하는 ARX(Addition, Rotation, Xor) 연산에 따른 지연이 라운



(그림 3) 설계된 키 스케줄 블록
(Figure 3) Designed key schedule block

드 연산 블록까지 전달되어 설계된 IP의 클럭 속도를 저하시키는 것을 방지하기 위해 레지스터에 버퍼링한 라운드 키를 제공하기 위해서 라운드 연산 블록보다 한 클럭 먼저 키 스케줄 블록에 동작하도록 설계하였기 때문이다. 라운드 연산 블록은 FSM으로부터 암호화 모드 혹은 복호화 모드 여부를 알려주는 신호인 EncryptMode, DecryptMode 신호와 라운드 갱신 신호인 rnd_update를 신호를 사용하여 암호화와 복호화에 해당하는 연산을 수행하고 매 라운드 마다 x0~x3 레지스터를 갱신한다.

마지막으로 암호화와 복호화의 마지막 라운드에서 FSM은 EDResultUpdate 신호를 생성하여 Result 버퍼에 최종 결과를 저장하고, 이후 OutRegShiftEn를 신호를 활성화하여 저장된 결과 값을 외부로 전달하도록 한다. 이때는 DataOutValid 신호를 생성하여 결과 값이 유효함을 알리도록 설계하였다.

LEA 암호 알고리즘의 설계에 있어서 가장 복잡한 부분이 키 스케줄 블록이다. (그림 3)은 본 논문에서 설계한 암·복화 키 생성이 가능한 키 스케줄 블록을 보여주고 있다.

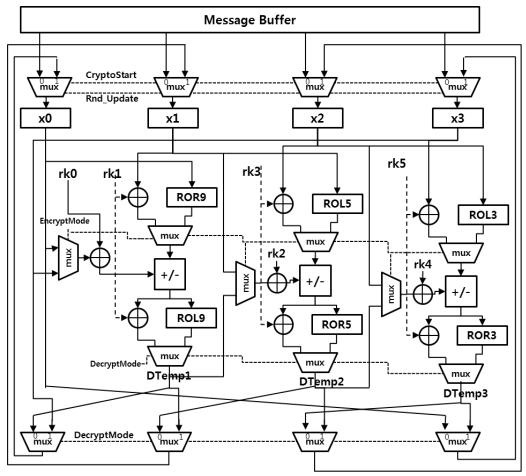
먼저 (그림 3)의 상단에 위치한 R0~R7 레지스터는 델타 값을 관리하는 레지스터이다. 라운드 키를 생성하기 위해 사용되는 레지스터는 항상 R0가 되도록 설계하였으며, 초기 델타 값의 설정은 암호화 시에는 참고 문헌 [8]에서 제시된 8개의 값을 이용하면 된다. 그러나 복호화 시에는 사용하는 매스터 키의 크기에 따라 다른 값으로 설정해 주어야 한다. 이러한 값은 LEA 프로그램

분석을 통해서 얻은 값을 사용하였다. 그리고 델타 레지스터간 연결 방법 및 회전 방향이 암호화와 복호화시에 정 반대 방향으로 구현된다. $KReg_{i[i=0부터7사이]}$ 레지스터의 입력 중에서 EnKey0~EnKey7은 암호화 시에 제공되는 초기 키 값인데, 이는 Key 버퍼로부터 제공받으며, DeKey0~DeKey7은 복호화 시에 사용되는 초기 키 값인데, 이는 DeKey 버퍼로부터 제공받아 사용된다. 128비트 매스터 키 모드에서는 KReg0~KReg3까지 4개의 레지스터만 사용하여 라운드 키를 생성하는데, 192비트 라운드 키에 맵핑시키기 위해 rk1 레지스터는 rk1, rk3 및 rk5로도 사용되며, rk4 레지스터에 저장된 값은 rk3으로 변형되어 사용된다. 192비트 매스터 키 모드에서는 rk0~rk5에 저장되는 값이 라운드 키 r0~rk5로 일대일 맵핑된다. 256비트 매스터 키 모드에서는 제공되는 키는 256비트지만, 생성되는 라운드 키는 192비트이므로 처음에는 KReg0~KReg5가 라운드 키 생성에 사용되지만, 다음 클럭에는 64비트 좌측 회전한 후 라운드 키를 생성한다. 따라서 각 라운드 키를 생성할 때마다 64비트 좌측 회전된 KReg0~KReg5가 라운드 키 생성에 사용되어 256비트가 모두 공평하게 라운드 키 생성에 사용되게 된다. 256비트 매스터 키 복호화 모드에서는 라운드 키 생성에 사용되는 레지스터는 KReg0~KReg5까지 동일하지만, 암호화 경우와 반대로 우측 64비트 회전한 후 새로운 라운드 키 생성을 하게 된다.

키 스케줄 블록의 내부 연산은 암호화와 복호화를 합체하여 구현하였다. 암호화 시의 경로는 XOR 연산을 수행한 후 모듈로 덧셈 연산을 수행하고 마지막으로 회전 연산을 수행한다. 복호화 시에는 회전 연산을 먼저 수행한 후 모듈로 뺄셈 연산을 수행하고 마지막으로 XOR 연산을 수행한다. 따라서 적절한 멀티플레서의 사용과 덧셈과 뺄셈을 동시에 수행할 수 있는 모듈을 사용하여 하드웨어 사용을 줄일 수 있도록 설계하였다.

마지막으로 라운드 연산 블록의 설계에 대해 설명하고자 한다. (그림 4)는 설계된 라운드 연산 블록을 보여주고 있다.

메시지 버퍼에 128비트 데이터가 도착하면 라운드연산 블록이 암·복호화를 수행중이 아니면 즉시 CryptoStart 신호가 활성화되어 메시지 버퍼에서 x0~x3 레지스터에 전달되며 매스터 키 모드와 암·복호화 모드에 따라 라운드 연산을 수행한다. 물론 암·복호화를 수행중이라면, 마지막 라운드 종료 시점에 x0~x3 레지스터에 전달되어 라운드 연산을 수행하도록 설계하였다. 그리고 x0~x3레지스터는 매 라운드 연산이 종료될 때마다



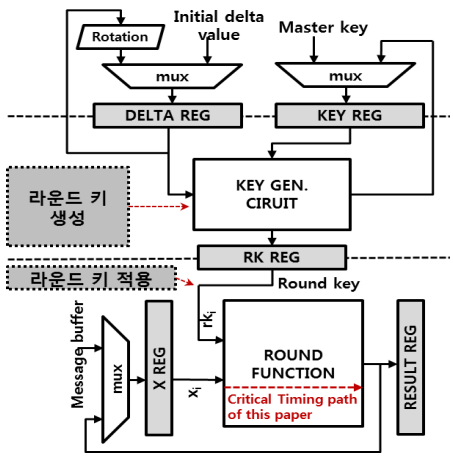
(그림 4) 제안하는 라운드 연산 블록의 구조
(Figure 4) The design of round operation block

rnd_update 신호가 활성화되어 갱신된다. 내부 라운드 연산은 암호화와 복호화를 동시에 수용할 수 있는 구조로 설계하였다.

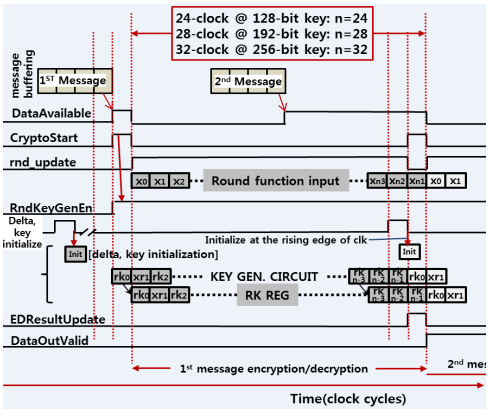
암호화 시의 라운드 연산은 XOR연산 수행 후 모듈로 덧셈 연산을 수행하고 회전 연산을 수행한 다음 {DTemp1, DTemp2, DTemp3, x0}로 정렬되어 x0~x3 레지스터에 저장한다. 복호화 시의 라운드 연산은 암호화 과정과 반대로 {x3, x0, x1, x2, x3}로 먼저 정렬되고 회전 연산을 수행한 후 모듈로 뺄셈 연산을 수행하며, XOR 연산을 마친 값이 x0~x3에 저장된다. 따라서 32비트 덧셈기와 뺄셈기를 사용하지 않고 덧셈 및 뺄셈을 제공하는 가감산기를 사용하여 하드웨어 자원을 효율적으로 관리할 수 있도록 하였다.

4. LEA 암호의 성능 향상을 위한 설계

LEA 암호 알고리즘에 대해 암호화 및 복호화를 동시에 지원하는 고속의 IP에 관한 논문은 발표되지 않았고, 저면적의 저속 IP는 논문 [3]에서 발표되었다. 고속의 LEA 알고리즘 구현 논문 [6]은 복호화 기능이 없고, 암호화 기능만을 지원하고 있다. 따라서, 본 논문은 고속을 목표로 한 논문 [6]보다 우수한 성능의 암호화와 복호화 및 3개의 매스터 키 모드를 동시에 지원하고 IP 구현을 목표로 하였다. 이를 위해 먼저 (그림 5)에 단순화한 LEA 아키텍처와 핵심 타이밍도를 제시하고, 성능 향상을 위해 적용한 디자인 기법을 설명할 것이다.



(a)



(b)

(그림 5) 단순화한 LEA 구조 및 타이밍
(Figure 5) Simplified LEA structure and timing: (a)Simplified LEA structure, (b)timing

(그림 5)에서 회색의 배경색이 있는 DELTA REG, KEY REG, RK REG, X REG 및 RESULT REG는 레지스터를 의미한다. 모든 레지스터는 클럭의 상승 에지에서 값이 변경되도록 설계되었다. 논문 [6]을 분석한 결과 IP의 성능에 큰 영향을 미치는 최악지연경로(Worst-case Critical Path)가 KEG REG와 DELTA REG의 출력 값부터 조합회로인 키 생성 회로(KEY GEN CIRCUIT) 블록의 거쳐, 다시 라운드 함수(ROUND FUNCTION) 블록까지의 경로가 단일 사이클에 이루진 구조였다.

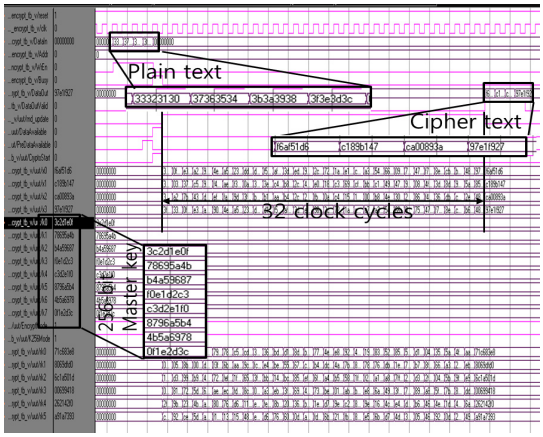
이는 신호의 지연 경로가 너무 길어서 결과적으로 동작 주파가 낮아져 고성능을 얻기에는 적절하지 않다. 따라서, 본 논문에서는 (그림 5)의 (a)에서 볼 수 있듯이 RK REG 레지스터를 키 생성 블록의 말단에 삽입하여, 키의 생성과 키의 사용을 분리시켜 최악지연경로가 RK REG 및 X REG의 출력부터 라운드 함수 블록에 한정하도록 하였다. 이를 통해 256비트 키의 암호화만을 지원하는 논문 [6]과 비교해 암호화와 복호화를 지원하고 128, 192, 256 비트 키 모두를 지원하는 훨씬 복잡한 구조인 본 논문의 IP 성능이 약 30%이상 향상되었다.

(그림 5)의 (b)는 LEA IP의 성능 향상을 위해 본 논문에서 제안한 방식을 구현하기 위한 타이밍도를 보여준다. 먼저 LEA IP의 기본 설정이 완료된 후 매스터 키 정보가 로딩되면 델타 값과 KEY REG 레지스터에 대한 초기화 진행된다. 이후 메시지 버퍼에 128비트의 데이터가 도착하면 CryptoStart 신호가 활성화되면서 다음 클럭의 상승 에지에 X REG에 메시지가 전달된다. 한편 키 스케줄 블록 쪽에서는 키 생성 회로 블록에서 첫 번째 라운드 키가 계산되고, 클럭의 상승 에지에서 RK REG에 첫 번째 라운드 키가 전달되어 첫 번째 라운드의 X 값과 라운드 키 값의 동기가 맞게 된다. 즉, RK REG 값은 키 생성 회로 블록의 결과 값을 클럭의 상승 에지에서 전달받은 결과가 된다. 새로운 메시지 블록에 대한 연속적인 처리를 위해 키 생성 회로 블록의 마지막 라운드 키 생성 시간에 델타 및 키 초기화 신호가 활성화되어야 한다. 이는 클럭의 상승 에지에서 생성된 라운드 키는 KEY REG로 전달되고, 동시에 DELTA REG와 KEY REG는 새로운 암·복호화를 위한 초기 값으로 초기화되어야 한다. 이러한 제어 신호를 생성하면 연속적인 메시지 블록의 처리가 가능하게 된다.

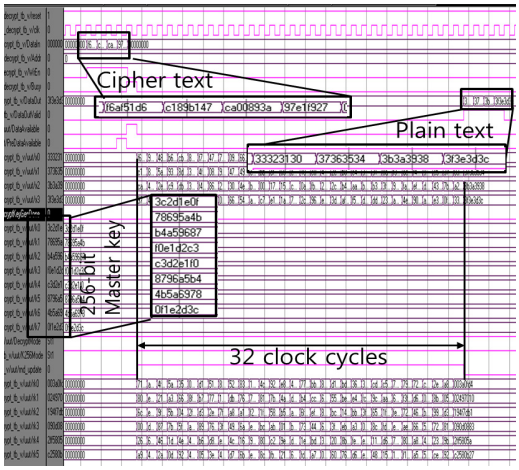
5. 설계된 LEA 암호의 검증 및 성능 분석

구현한 LEA 암호 알고리즘의 검증을 위해 먼저 참고문헌 [8]에서 제공하는 테스트 케이스 등을 참고하였다. Verilog HDL을 사용하여 LEA IP를 설계하였으며, 제시된 테스트 케이스의 입출력 값을 설계된 IP에서 입력에 대한 출력 값이 서로 일치하는지를 비교하여 검증을 수행하였다. 본 논문에서 128비트, 192비트 256비트의 매스터 키를 지원하도록 설계하고 검증하였는데, 검증 자료는 256비트 매스터 키 모드에 대해서만 설명할 것이다.

(그림 6)은 256비트 매스터 키를 채용한 암호화 시뮬레이션 파형을 보여주고 있다. 128비트 평문 입력 “33323130”, “37363534”, “3b3a3938”, “3f3e3d3c:”에 대해 256비트 매스터 키 “3c2d1e0f”, “78695a4b”, “b4a59687”,



(그림 6) LEA 블록암호의 암호화 시뮬레이션
(Figure 6) simulation of encryption for LEA cipher



(그림 7) LEA 블록암호의 복호화 시뮬레이션
(Figure 7) simulation of decryption for LEA cipher

"f0e1d2c3", "c3d2e1f0", "8796a5b4", "4b5a6978", "0f1e2d3c"를 적용하여 암호화를 수행하면 최종적인 암호 출력은 "f6af51d6", "c189b147", "ca00893a", "97e1f927"으로 테스트 케이스에서 제시된 값과 일치함을 확인하였다.

(그림 7)은 LEA 암호 알고리즘의 복호화 과정에 대한 시뮬레이션을 보여주고 있다. 복호화 과정은 암호화 과정의 반대 과정으로 암호화된 입력 값을 사용하여 평문을 얻는 과정이다.

표 1은 본 논문에서 설계된 LEA 암호 알고리즘의 설계 결과를 요약하고 비교한 것이다. 본 논문에서 설계한 LEA 암호 알고리즘은 128, 192 및 256비트의 키를 사용한 암·복호화를 지원하며 높은 데이터 처리율을 목표로 하고 있다.

(표 1) 제안하는 LEA 블록 암호와 타 구현의 비교
(Table 1) Comparison of LEA block ciphers

Items	Designed IP	Paper [3]	Paper [6]
Block	128-bit	128-bit	128-bit
Key size	128, 192, 256-bit	128, 192, 256-bit	256-bit only
Encryption/ decryption	All support	All support	Encryption only
Device	Vertex5	Vertex5	Vertex5
Frequency	164MHz	113MHz	126MHz
Maximum throughput	874Mbps@128-bit key 749Mbps@192-bit key 656Mbps@256-bit key	181Mbps@128-bit key 162Mbps@192-bit key 109Mbps@256-bit key	496Mbps@256-bit key

논문 [3]은 3가지 유형의 매스터 키를 지원하는 암·복호화를 지원하지만 면적을 최소화한 구현을 목표로 하고 있다. 논문 [6]은 면적 및 속도 측면에서 여러 개의 모듈을 설계하여 비교 분석하고 있지만 128비트, 192비트 및 256비트를 모두 통합한 모듈 설계가 없으며 각 모듈별로 면적 측면과 속도 측면에 따른 모듈별 설계를 하여 비교 분석하였으며, 복호화를 지원하지 않는다. 표 1에서는 논문 [6]의 256비트 고속 모델인 "LEA-256-SPEED" 모듈과 비교를 수행하였다.

특히 본 논문에서는 LEA 코어의 성능을 향상시키기 위해 라운드 연산 블록에 on-the-fly 방식으로 라운드 키를 제공할 때 한 클럭 먼저 동작하여 레지스터에 래치된 안정된 값을 제공하여 라운드 키 생성 과정에서 발생하는 지연시간이 라운드 연산 블록의 수행시간에 영향을 주지 않도록 하였다. 이러한 설계 방식의 채택은 256비트 매스터 키만 지원하고 암호화만을 수행하는 논문 [6]과 비교하여도 30% 이상의 클럭 주파수의 향상을 얻었다.

본 논문에서 설계한 LEA 암호 알고리즘 IP는 verilog-HDL로 설계하여 Xilinx Vertex5에 구현한 결과 약 164MHz에서 동작하는 것을 확인하였다. 따라서 본 논문에서 설계된 IP의 최대 암·복화율은 128비트 키 모드에 대해서는 874Mbps, 192비트 키 모드에 대해서는 749Mbps, 그리고 256비트 키 모드에서는 656Mbps의 처리 능력을 보여주었다. 256비트의 매스터를 사용한 암호화만 지원하는 논문 [6] "LEA-256-SPEED" 모델의 최대 암호화율은 496Mbps였으며, 면적 최소화에 중점을

두고 3가지 유형의 매스터 키에 대한 암•복화를 지원하는 논문 [3]은 128비트 키 모드에 대해서는 181Mbps, 192비트 키 모드에 대해서는 162Mbps, 그리고 256비트 키 모드에서는 109Mbps의 성능을 보였다. 설계된 LEA IP는 Xilinx Vertex5 상에서 836개의 슬라이스(Slice) 자원을 사용하고, 1638개의 플립플롭 및 2615개의 슬라이스 LUT를 사용하였다.

6. 결 론

본 논문에서는 128비트 입력에 대해 128비트, 192비트 및 256비트 키를 지원하며 높은 데이터 처리율을 가지는 LEA 암•복호화 IP를 설계하였다. 설계된 암•복호화 알고리즘은 Xilinx Vertex5에서 164MHz에서 동작하며, 초당 최대 암•복화율은 128비트 키 모드에서는 최대 874Mbps, 192비트 및 256비트 키 모드에서는 최대 749Mbps 및 656Mbps의 데이터 처리율을 가진다. 기존에 발표된 LEA 블록 암호 논문들과 비교하였을 경우에도 우수한 성능을 보이는 것으로 확인되었다. 이러한 LEA 암호 알고리즘은 인터넷 뱅킹, 위성 방송, IP 보안, 데이터 보안, IoT 및 영상통화 보안 등 기타 다양한 분야에서 활용이 기대된다.

참 고 문 헌(Reference)

[1] Sungjoo Ha, Jongho Lee, “Design of fast encryption/decryption for block cipher ARIA,” Institute of korean electrical and electronics engineers, Vol. 57 No. 9, pp.1652-1659, Sep. 2008. http://www.ndsl.kr/ndsl/commons/util/ndslOriginalView.do

[2] Seungil Sonh, “Design of Encryption /Decryption Core for Block Cipher HIGHT,” JKIICE, Vol.16 No. 4, pp.778-784, April 2012. http://dx.doi.org/10.6109/jkiice.2012.16.4.778

[3] Mi-ji Sung, Kyung-wook Shin, ″An Efficient Hardware Implementation of Lightweight Block Cipher LEA-128/192/256 for IoT Security Applications,″ JKIICE, Vol.19, No.7, pp1609-1616, Jul. 2015. http://www.ndsl.kr/ndsl/commons/util/ndslOriginalView.do http://dx.doi.org/10.6109/jkiice.2015.19.4.888

[3] Seungil Sonh, Byeongyoon Choi, Mingoo Kanag, “Technology Trend of Cipher Chips,” KSII, Vol.1 No. 2, pp1491-1500, Oct. 2000.

[5] Byeongyoon Choi, Jinil Kim, ″CPLD Implementation of SEED Cryptographic Coprocessor,″ JISPS, Vol.1, No.1-2, pp.177-185, Oct. 2001. http://www.ndsl.kr/ndsl/commons/util/ndslOriginalView.do

[6] Donggeon Lee et al., ″Efficient Hardware Implementation of the Lightweight Block Encryption Algorithm LEA,″ Sensors, pp.975-994, 2014. http://www.mdpi.com/1424-8220/14/1/975

[7] FIPS Publication 197, ″Advanced Encryption Algorithm(AES),″ U.S. Doc/NIST. https://doi.org/10.6028/NIST.FIPS.197

[8] Telecommunications Technology Association, ″128-Bit Block Cipher LEA,″ TTA Standard, TTAKKO-12.0223, 2013. http://seed.kisa.or.kr/

[9] Electronic Times, http://v.media.daum.net/v/ 20160801170007878

◐ 저 자 소 개 ◑

손 승 일(Seungil Sonh)

1989년 연세대학교 전자학과(공학사)
1991년 연세대학교 대학원 전자공학과(공학석사)
1998년 연세대학교 대학원 전자학과(공학박사)
1998년~2002년 호남대학교 컴퓨터공학과 조교수
2008년~2009년 미국미시간공대 방문교수
2014년~2015년 미국조지아공대 방문교수
1998년~현재 한신대학 정보통신학부 교수
관심분야 : 암호, 프로세서 설계, etc.
E-mail : saisonh@hs.ac.kr