論 文
57-9-26

블록 암호 ARIA를 위한 고속 암호기/복호기 설계

Design of High Speed Encryption/Decryption Hardware for Block Cipher ARIA

河 成 珠* · 李 鍾 浩†
(Seong-Ju Ha · Chong-Ho Lee)

Abstract - With the increase of huge amount of data in network systems, ultimate high-speed network has become an essential requirement. In such systems, the encryption and decryption process for security becomes a bottle-neck. For this reason, the need of hardware implementation is strongly emphasized. In this study, a mixed inner and outer round pipelining architecture is introduced to achieve high speed performance of ARIA hardware. Multiplexers are used to control the lengths of rounds for 3 types of keys. Merging of encryption module and key initialization module increases the area efficiency. The proposed hardware architecture is implemented on reconfigurable hardware, Xilinx Virtex2-pro. The hardware architecture in this study shows that the area occupied 6437 slices and 128 BRAMs, and it is translated to throughput of 24.6Gbit/s with a maximum clock frequency of 192.9MHz.

Key Words : ARIA, Cryptography, FPGA, Pipeline, Encrypt

1. 서 론

암호는 과거에는 군사적인 용도 등의 비밀통신을 위하여 주로 사용하였으나 현재는 인터넷 기반의 사회, 경제 활동의 안전성, 신뢰성, 프라이버시 보호 등을 위한 핵심기술로서 메일 전송, 사용자 인증, 전자상거래 등에 널리 사용되고 있다. 점차 암호 기술은 특정분야에서 사용하는 특수기술에서 차세대 정보 환경의 기반기술로 변화하고 있으며 중요성이 증대되고 있다[1]. 초고속 네트워크 기반의 전자정부 시스템을 비롯해 앞으로 다가올 다양한 정보보호 환경을 대비하여 개발된 차세대 국가 암호 알고리즘 ARIA가 지난 2004년 12월 30일 한국산업규격 KS 표준으로 제정되었다. ARIA는 민간 암호 알고리즘 SEED와 함께 전자정부의 대국민 행정서비스용으로 보급되고 있으며, 스마트 카드 등의 초경량 환경 및 고성능 서버 환경 등에서 SEED에 비하여 상대적으로 장점을 가지고 있다[2].

ARIA 알고리즘이 발표된 이후 시스템의 요구에 맞는 다양한 구현 방법들이 연구되었다. 32-비트, 64-비트 범용 프로세서에 적합한 소프트웨어 구현과 분석이 이루어졌다[3][4]. 또한 1cycle/round 구조의 하드웨어 구현을 비롯해서 스마트 카드나 RFID와 같은 휴대용 장치에 적합한 소면적, 저전력 위주의 하드웨어 구현이 이루어 졌다[5][6][7][8].

한편, 공정기술의 발달로 재구성 하드웨어의 생산단가가 낮아지고, 기능과 성능이 우수해지면서 단일칩으로 시스템 레벨의 설계 및 구현이 가능해졌다. 특히, 재구성 하드웨어는 소프트웨어와 같이 사용자의 요구나 상황에 알맞게 내부 회로를 쉽게 재구성 할 수 있다[9][10]. 따라서 시장에서의 그 수요가 점점 증가하고 있으며, 활용 범위가 넓어지고 있다. 네트워크에서 데이터의 전송과정에서 정보는 쉽게 노출되기 때문에 비밀 정보에 대한 보안 기술이 요구된다. 네트워크에서 보안을 위한 암호화/복호화 과정은 병목현상의 요인으로 작용하므로 빠른 처리를 위해서 하드웨어화가 필요한데, 보안 시스템에 적용되는 알고리즘과 표준안은 수시로 변화하기 때문에, 그 변화에 따라 시스템 변경이 필요하다. 이런 점을 고려했을 때, 재구성 하드웨어가 보안시스템의 구현에 적합하다.

본 논문에서는 기존 ARIA 하드웨어 구현에서 다루지 않았던 대용량 데이터의 고속처리에 초점을 맞추어 재구성 하드웨어를 사용한 각 모듈의 구현 방법을 분석하였다. 그리고 구현은 Virtex2-pro에서 하였으며, 그 결과를 동급의 경쟁 알고리즘인 AES의 하드웨어 구현 결과와 성능을 비교하였다.

2. ARIA 알고리듬[11]

ARIA는 암호의 난이도에 따라서 128, 192, 256비트 길이의 암호키를 선택할 수 있으며, 128비트 데이터 블록에 대해 암호화(복호화)를 수행한다. 키의 길이에 따라서 암호화(복호화)는 라운드 함수가 12, 14, 16번 반복 실행하며, 각각 ARIA-128, ARIA-192, ARIA-256으로 구분한다. 라운드 함수에 필요한 라운드키는 암호키로부터 키 확장을 통해서 생성한다.

* 學生會員 : 仁荷大 工大 情報通信工學科 碩士課程
† 교신저자, 選任會員 : 仁荷大 工大 情報通信工學科 敎授 · 工博
E-mail : chlee@inha.ac.kr
接受日字 : 2008年 5月 23日
最終完了 : 2008年 8月 7日

2.1 표기 및 용어

- Involution 구조: 암호화 과정과 복호화 과정이 같은 구조
- SPN 구조: Substitution-Permutation-Networks 구조로 치환과 확산이 반복적으로 사용되는 구조
- Feistel 구조: 데이터를 두 블록으로 나누어 좌·우부분에 교대로 비선형 변환을 적용시키는 구조
- $W^{\lll k}$: W의 각 비트를 왼쪽으로 k비트씩 순환이동
- $W^{\ggg k}$: W의 각 비트를 오른쪽으로 k비트씩 순환이동

2.2 라운드 함수

ARIA의 라운드 함수는 라운드키 덧셈(AddRoundKey), 치환 계층(SubstLayer), 확산 계층(DiffLayer) 세부분으로 구성된 SPN 구조를 가진다. 라운드 함수는 홀수 라운드(F_o), 짝수 라운드(F_e)에 따라 다른 유형의 치환 계층이 사용되며, 마지막 라운드(F_f)에서 확산 계층을 라운드키 덧셈으로 대치한다. 라운드키 덧셈은 128-비트 라운드키와 라운드 입력 128-바트와 비트별 XOR 한다. 그리고 치환 계층은 8비트 입/출력을 가지는 S-box S_1, S_2과 그 역치환 S_1^{-1}, S_2^{-1}으로 구성되어 있으며, S_1, S_2은 식(1)과 식(2)로 구해진다. (S_1, S_2, S_1^{-1}, S_2^{-1})를 LT라고 하면, 홀수 라운드의 치환계층은 (LT, LT, LT, LT) 짝수 라운드의 치환계층은 (LT^{-1}, LT^{-1}, LT^{-1}, LT^{-1})로 이루어져 involution 구조를 이룬다. 마지막으로 확산 계층은 16×16 involution 이진 행렬을 사용하며, 입력을 $(x_0, x_1, \ldots, x_{15})$라 하고 출력을 $(y_0, y_1, \ldots, y_{15})$라 하면 식(3)의 행렬의 곱으로 표현된다.

$$S_1 : x \rightarrow A \cdot x^{-1} \oplus a \tag{1}$$

$$S_2 : x \rightarrow B \cdot x^{-1} \oplus b \tag{2}$$

$$A=\begin{bmatrix}1&0&0&0&1&1&1&1\\1&1&0&0&0&1&1&1\\1&1&1&0&0&0&1&1\\1&1&1&1&0&0&0&1\\1&1&1&1&1&0&0&0\\0&1&1&1&1&1&0&0\\0&0&1&1&1&1&1&0\\0&0&0&1&1&1&1&1\end{bmatrix} \text{ and } a=\begin{bmatrix}1\\1\\0\\0\\0\\1\\1\\0\end{bmatrix}, \quad B=\begin{bmatrix}0&1&0&1&1&1&1&0\\0&0&1&1&1&1&0&1\\1&1&0&1&0&1&1&1\\1&0&0&1&1&1&0&0\\0&0&1&0&1&1&0&1\\0&0&0&0&1&0&0&1\\1&0&0&0&0&0&0&1\\0&1&0&1&1&0&1&1\end{bmatrix} \text{ and } b=\begin{bmatrix}0\\1\\0\\0\\0\\1\\1\\1\end{bmatrix}$$

$$\begin{pmatrix}y_0\\y_1\\y_2\\y_3\\y_4\\y_5\\y_6\\y_7\\y_8\\y_9\\y_{10}\\y_{11}\\y_{12}\\y_{13}\\y_{14}\\y_{15}\end{pmatrix}=\begin{bmatrix}0&0&0&1&1&0&1&0&1&1&0&0&0&1&1&0\\0&0&1&0&0&1&0&1&1&1&0&0&1&0&0&1\\0&1&0&0&1&0&1&0&0&0&1&1&1&0&0&1\\1&0&0&0&0&1&0&1&0&0&1&1&0&1&1&0\\1&0&1&0&0&1&0&0&1&0&0&1&1&0&0&1\\0&1&0&1&1&0&0&0&0&1&1&0&0&0&1&1\\1&0&1&0&0&0&0&1&0&1&1&0&1&1&0&0\\0&1&0&1&0&0&1&0&1&0&0&1&1&1&0&0\\1&1&0&0&1&0&0&1&0&0&1&0&0&1&0&1\\1&1&0&0&0&1&1&0&0&0&0&1&1&0&1&0\\0&0&1&1&0&1&1&0&1&0&0&0&0&1&0&1\\0&0&1&1&1&0&0&1&0&1&0&0&1&0&1&0\\0&1&1&0&0&0&1&1&0&1&0&1&1&0&0&0\\1&0&0&1&0&0&1&1&1&0&1&0&0&1&0&0\\1&0&0&1&1&1&0&0&0&1&0&1&1&0&0&1\\0&1&1&0&1&1&0&0&1&0&1&0&0&0&0&1\end{bmatrix}\bullet\begin{pmatrix}x_0\\x_1\\x_2\\x_3\\x_4\\x_5\\x_6\\x_7\\x_8\\x_9\\x_{10}\\x_{11}\\x_{12}\\x_{13}\\x_{14}\\x_{15}\end{pmatrix} \tag{3}$$

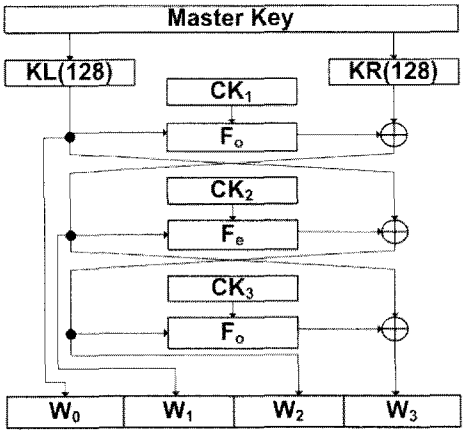

그림 1 ARIA 키 초기화 과정
Fig. 1 ARIA key initialization process

2.3 키 확장

ARIA의 키 확장은 키 초기화 과정과 라운드키 생성 과정의 두 부분으로 나뉜다.

2.3.1 키 초기화 과정

키 초기화 과정에서는 그림 1과 같이 3라운드 Feistel 암호를 이용하여, 암호키(MK)로부터 네 개의 128-비트 값 W_0, W_1, W_2, W_3를 생성한다. Feistel 암호는 (KL, KR)로 구성된 256-비트 입력이 필요하다. 키의 길이가 128-비트인 경우에는 KL을 암호키로 채우고 KR은 0으로 채운다. 키의 길이가 192-비트인 경우에는 KL은 암호키의 상위 128-비트로 채우고, KR은 암호키의 하위 64-비트와 64-비트의 0으로 채운다. 키의 길이가 256-비트인 경우 KL과 KR은 암호키의 상위, 하위 128-비트로 채운다.

Feistel 암호의 128-비트 라운드키 CK_i는 π^{-1}의 유리수 부분의 128-비트 상수($C1$, $C2$, $C3$)를 이용하여 암호키의 길이에 따라 표 1과 같이 결정된다.

$C1$ = 0x517cc1b727220a94fe13abe8fa9a6ee0
$C2$ = 0x6db14acc9e21c820ff28b1d5ef5de2b0
$C3$ = 0xdb92371d2126e9700324977504e8c90e

표 1 암호키 길이에 따른 초기화 상수
Table 1 Initial constant according to the length of key

암호키 길이	CK_1	CK_2	CK_3
128-비트	$C1$	$C2$	$C3$
192-비트	$C2$	$C3$	$C1$
256-비트	$C3$	$C1$	$C2$

2.3.2 라운드키 생성 과정

라운드키 생성 과정에서는 네 개의 128-비트 W_0, W_1, W_2, W_3를 조합하여 암호화 라운드키 ek_i와 복호화 라운드

키 dk_i를 생성한다. 암호화 라운드키는 다음과 같다.

$$ek_1 = (W_0) \oplus (W_1^{\gg 19}), \quad ek_2 = (W_1) \oplus (W_2^{\gg 19}),$$
$$ek_3 = (W_2) \oplus (W_3^{\gg 19}), \quad ek_4 = (W_3) \oplus (W_0^{\gg 19}),$$
$$ek_5 = (W_0) \oplus (W_1^{\gg 31}), \quad ek_6 = (W_1) \oplus (W_2^{\gg 31}),$$
$$ek_7 = (W_2) \oplus (W_3^{\gg 31}), \quad ek_8 = (W_3) \oplus (W_0^{\gg 31}),$$
$$ek_9 = (W_0) \oplus (W_1^{\gg 67}), \quad ek_{10} = (W_1) \oplus (W_2^{\gg 67}),$$
$$ek_{11} = (W_2) \oplus (W_3^{\gg 67}), \quad ek_{12} = (W_3) \oplus (W_0^{\gg 67}),$$
$$ek_{13} = (W_0) \oplus (W_1^{\ll 97}), \quad ek_{14} = (W_1) \oplus (W_2^{\ll 97}),$$
$$ek_{15} = (W_2) \oplus (W_3^{\ll 97}), \quad ek_{16} = (W_3) \oplus (W_0^{\ll 97}),$$
$$ek_{17} = (W_0) \oplus (W_1^{\ll 109})$$

복호화 라운드키는 암호화 라운드키와 다르며 암호화 라운드키로부터 유도된다. 먼저 키의 순서가 바뀌고 처음과 마지막 라운드키를 제외하고 암호키를 입력으로 하는 식 (3)의 확산 함수(A)의 출력이 복호화 라운드키가 된다. 라운드 수가 n일 때, 복호화 라운드키는 다음과 같다.

$$dk_1 = ek_{n+1},\ dk_2 = A(ek_n),\ dk_3 = A(ek_{n-1}),$$
$$\cdot\ \cdot\ \cdot,\ dk_n = A(ek_2),\ dk_{n+1} = ek_1$$

3. ARIA 하드웨어

ARIA는 라운드 함수로 이루어진 암호화 모듈과 키 초기화 모듈, 라운드키 생성 모듈로 이루어진다.

3.1 암호화 모듈

3.1.1 순환 구조

ARIA 알고리즘은 라운드 함수의 반복으로 이루어지므로 그림 2(a)와 같이 단일 라운드 feedback 구조가 가능하다. 최종 라운드의 경우 확산계층 대신 라운드키 덧셈을 하므로 라운드 입력은 3대1 멀티플렉서와 128-비트 레지스터가 필요하다. 그리고 홀수 라운드와 짝수 라운드에 따라 치환계층의 데이터 패스를 조정하기 위해 치환계층의 앞뒤로 멀티플렉서가 필요하다. 단일 라운드 구조로써 면적 면에서 효율이 높은 구조이지만, 입력에 대해 라운드를 반복하는 동안 새로운 입력을 받을 수 없다.

3.1.2 비순환: outer round pipelining 구조

라운드 반복에 의한 시간 지연을 막기 위해서는 그림 2(b)와 같은 라운드 non-feedback 파이프라인 구조가 필요하다. 기본 구조에 비해서는 면적이 라운드 길이 배만큼 늘어나지만, 연속적으로 입력을 병렬로 처리함에 따라 고속 처리에 유리하다.

3.1.3 비순환: mixed inner and outer round pipelining 구조

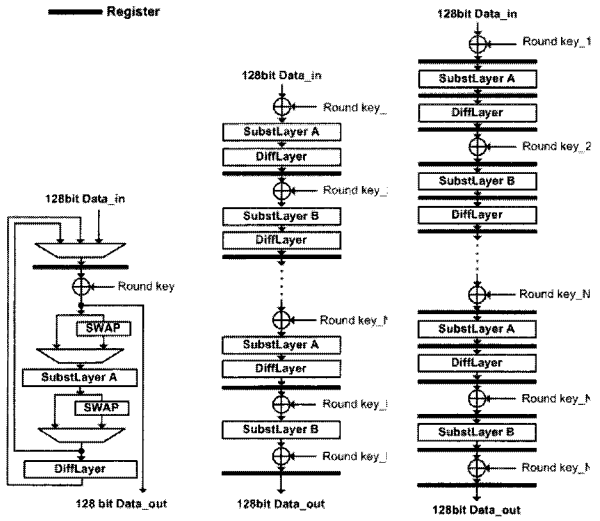

그림 2 암호화 모듈 구조, (a) 순환 구조, (b) outer round pipelining 구조, (c) mixed inner and outer round pipelining 구조

Fig. 2 Encryption module architecture, (a) Iteration architecture, (b) outer round pipelining architecture, (c) mixed inner and outer round pipelining architecture

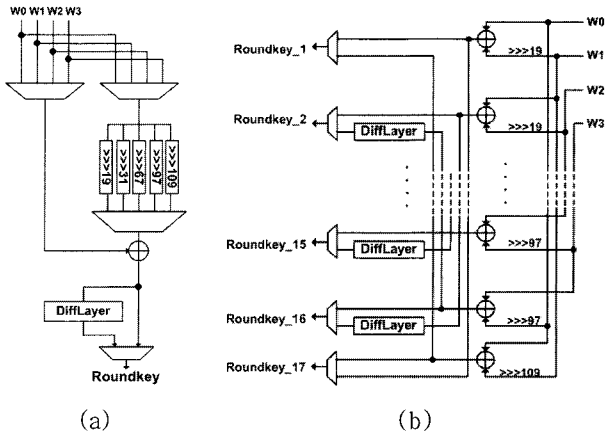

그림 3 라운드키 생성 모듈 구조

Fig. 3 Round key generation module architecture

그림 2(c)는 라운드 내부에도 파이프라인을 적용하여 레지스터 사이의 critical path를 줄임으로써 최대 동작 주파수를 outer round pipelining 구조보다 높일 수 있으므로 고속 처리에 가장 유리한 구조이다.

3.2 키 초기화 모듈

키 초기화 과정은 그림 1과 같이 페이스텔(Feistel) 구조로써 라운드 함수와 XOR 연산의 3회 반복으로 이루어지므

로 암호화 모듈과 마찬가지로 feedback 구조와 non-feedback 구조가 가능하다.

3.3 라운드키 생성 모듈

암호화 모듈에 따라 라운드키 생성 모듈의 구현은 달라진다. feedback 구조는 라운드키가 하나씩 순차적으로 필요하므로 그림 3(a)와 같은 구조가 적합하다. non-feedback 구조는 ARIA-256의 경우 17개의 라운드키가 동시에 필요하므로 그림 3(b)와 같은 구조가 적합하다.

3.4 기존 연구

ARIA 하드웨어 구현에 관한 기존 연구는 표 2과 같다. 최초로 [5]에서 FPGA에서 1cycle/1round feedback 구조를 제안하여 면적은 1490 slices, 16BRAMs를 차지하고, 437Mbps 성능을 내었다. 그리고 다른 연구들을 보면 소면적, 저전력 위주의 ASIC에서 feedback 구조의 구현이 이루어졌다. [6], [7], [8]에서는 데이터 버스를 줄여 필요한 S-box를 최소한으로 구성하였으며, [7], [8]에서는 면적을 줄이기 위해서 치환계층의 S-box를 테이블로 구성하지 않고, 조합논리 회로로 설계하였다. 데이터 버스를 줄이는 만큼 라운드 처리에는 더 많은 clock이 소모되어 Throughput이 높지 않다. 이와 같이 모든 기존 연구에서 feedback 구조를 갖는 형태로 구현되었다.

표 2 기존 연구 결과
Table 2 Related works

	Platform	Data Bus (bit)	AREA	Max. Freq. (MHz)	Throughput (Mbps)
박진섭[5]	XCV1600E-8	128	1491 slices 16 BRAMs	46.5	496
Park[6]	0.35μm CMOS	32	13893 gates	71	25
Yang[7]	0.18μm CMOS	16	6076 gates	15	4.8
유권호[8]	0.25μm CMOS	32	11301 gates	467	215
Koo[12]	0.25μm CMOS	128	15496 gates	-	816

4. 고속 처리 ARIA 하드웨어

본 논문에서는 기존의 연구에서 다룬 것과 달리 고속 비순환 구조 ARIA에 관한 하드웨어 구현을 제안하였다. 먼저 각각의 구현에 대한 특징들을 구분하기 위해 다음과 같은 표기법으로 구분하도록 한다.

예) ARIA-N-VAR-ED-B

처음 부분은 암호 알고리즘의 종류를 말하며, ARIA 알고리즘은 ARIA-로 시작하고, AES 알고리즘의 경우 AES-로 시작한다. 두 번째 부분은 암호의 구조를 말하며, N의 경우 non-feedback 라운드 구조를 말하며, F는 feedback 라운드 구조를 뜻한다. 세 번째는 지원하는 키 길이를 말하며, VAR은 128, 192, 256 모두 가능함을 뜻하며, 128-비트 키 길이의 경우 128로 표기한다. 네 번째 부분은 암호화, 복호화 지원유무를 말하며, ED의 경우는 둘다, E는 암호화만, D는 복호화만을 지원함을 뜻한다. 마지막 부분은 BRAM를 사용하였을 때만 표기하도록 한다.

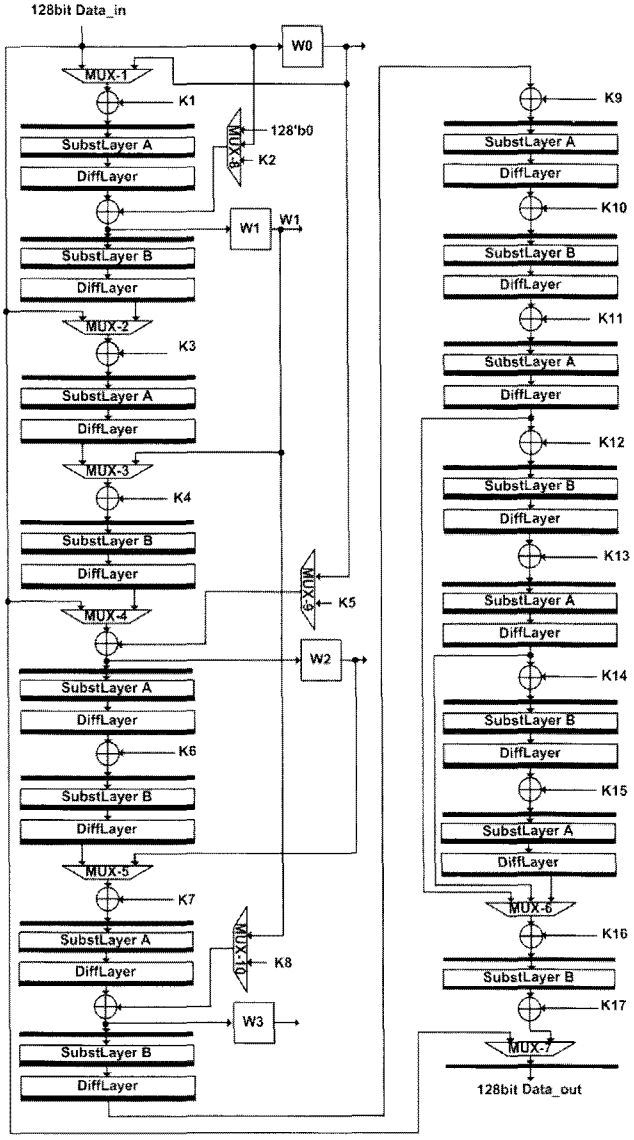

그림 4 제안하는 암호화 모듈 구조
Fig. 4 Proposed encryption module architecture

4.1 고속 처리 ARIA 하드웨어 구현

본 논문에서는 고속 처리에 적합한 블록 암호 ARIA 하드웨어 구현을 위해서 mixed inner and outer round pipelining 구조를 적용하였다. 그리고 면적을 줄이기 위해서 암호화 모듈과 키 초기화 모듈을 공유하는 구조를 제안하였다. 또한, 컨트롤 신호(Mode_A, Mode_B)를 통해 ARIA-128,

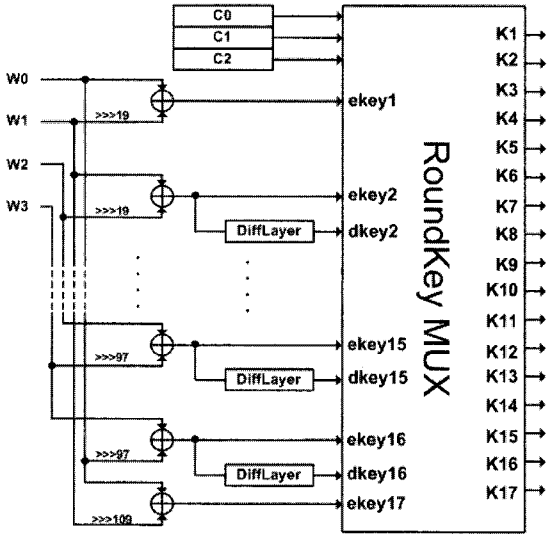

그림 5 제안하는 라운드키 생성 모듈 구조

Fig. 5 Proposed roundkey generation module architecture

ARIA-192, ARIA-256 모드가 모두 가능한 ARIA-N-VAR-ED를 제안한다.

4.1.1 암호화 모듈 구현

블록 암호 ARIA 알고리즘은 키 길이에 따라 라운드 함수의 반복 횟수가 달라진다. 그림 2(a)와 같은 feedback 구조의 경우에는 멀티플렉서를 이용하여 라운드의 출력을 다시 라운드의 입력으로 넣어 원하는 만큼 반복하면 된다. 따라서 feedback 구조는 키 길이에 따라서 구조가 바뀌지 않는다. 반면 non-feedback 구조는 키 길이에 따라 필요한 라운드 수만큼의 라운드가 요하다. 따라서 ARIA-128, ARIA-192, ARIA-256은 각각 12, 14, 16개의 라운드가 필요하며, 각각은 다른 키 길이에 대해 처리할 수 없다.

그래서 본 논문에서는 non-feedback 구조에서 키 길이에 따라 가변적으로 라운드 길이를 조정할 수 있도록 그림 4와 같은 구조를 제안하다. 라운드 수가 가장 많이 필요한 ARIA-256에 맞게 16개의 라운드로 구성하고 최종 라운드 앞에 3대1 멀티플렉서(MUX-6)를 두어 ARIA-128에는 11번째 라운드의 출력을, ARIA-192에는 13번째 라운드의 출력을, ARIA-256에는 15번째 라운드의 출력을 최종 라운드로 입력되도록 하였다.

복호화의 경우에도 암호화와 마찬가지로 MUX-6을 이용하여 라운드 길이를 조정할 수 있지만, 라운드키 생성 모듈의 복잡도를 낮추기 위해서 MUX-2, MUX-4를 추가하였다. ARIA-128에는 MUX-4를 이용하여 입력을 5번째 라운드에 넣어 최종라운드까지 순차적으로 전달함으로써 복호화가 이루어진다. ARIA-192에는 MUX-2을 이용하여 3번째 라운드에 입력이 들어가도록 하였고, ARIA-256의 경우에는 첫 번째 라운드로 입력이 들어가도록 하였다.

키 초기화 모듈은 라운드키 덧셈, 치환계층, 확산계층, 라운드키 덧셈의 4가지 함수를 3회 반복함으로써 키 초기화 값이 생성된다. 한편 암호화 과정이 라운드키 덧셈, 치환계층, 확산계층으로 이루어진 라운드 함수의 반복으로 이루어지므로 필요한 내부 함수가 동일하다. 따라서 그림 4에서 MUX-1, MUX-2, MUX-8, MUX-9를 이용하여 ARIA 코어로부터 키 초기화에 필요한 데이터 패스를 만들 수 있다. 그리고 키 초기화 값을 저장하는 *W0, W1, W2, W3*는 표 4와 같이 제어된다.

표 3 컨트롤 신호

Table 3 Control signal

Mode_A	Description	Mode_B	Description
00	Key Initialization	00	ARIA-128 / KR
01	Encryption	01	ARIA-192 / KL
10	Decryption	10	ARIA-256 / KL
11	Bypass	11	-

표 4 키 초기화 레지스터

Table 4 Key initial register

	Key Initialization			Encryption	Decryption	Bypass
	ARIA-128	ARIA-192	ARIA-256			
W0_en	1	0	0	0	0	0
W1_en	1	1	1	0	0	0
W2_en	1	1	1	0	0	0
W3_en	1	1	1	0	0	0

표 5 암호화 모듈 멀티플렉서

Talbe 5 Multiplexer of encryption module

	Key Initialization			Encryption			Decryption			Bypass
ARIA-	128	192	256	128	192	256	128	192	256	
mode	key128	key192	key256	Enc128	Enc192	Enc256	Dec128	Dec192	Dce256	
MUX-1	0	1	1	0	0	0	-	-	0	-
MUX-2	-	-	-	1	1	1	-	0	1	-
MUX-3	1	1	1	0	0	0	-	0	0	-
MUX-4	1	1	1	1	1	1	0	1	1	-
MUX-5	0	0	0	0	0	0	0	0	0	-
MUX-6	-	-	-	0	1	2	2	2	2	-
MUX-7	1	1	1	1	1	1	1	1	1	0
MUX-8	0	1	1	2	2	2	2	2	2	-
MUX-9	0	0	0	1	1	1	1	1	1	-
MUX-10	0	0	0	1	1	1	1	1	1	-

표 6 라운드키 생성

Table 6 Roundkey generation

	Key Initialization			Encryption			Decryption		
ARIA-	128	192	256	128	192	256	128	192	256
K1	C0	C1	C2	ekey1	ekey1	ekey1	-	-	ekey17
K2	-	-	-	ekey2	ekey2	ekey2	-	-	dkey16
K3	-	-	-	ekey3	ekey3	ekey3	-	ekey15	dkye15
K4	C1	C2	C0	ekey4	ekey4	ekey4	-	dkey14	dkey14
K5	-	-	-	ekey5	ekey5	ekey5	ekey13	dkey13	dkey13
K6	-	-	-	ekey6	ekey6	ekey6	dkey12	dkey12	dkey12
K7	C2	C0	C1	ekey7	ekey7	ekey7	dkey11	dkey11	dkey11
K8	-	-	-	ekey8	ekey8	ekey8	dkey10	dkey10	dkey10
K9	-	-	-	ekey9	ekey9	ekey9	dkey9	dkey9	dkey9
K10	-	-	-	ekey10	ekey10	ekey10	dkey8	dkey8	dkey8
K11	-	-	-	ekey11	ekey11	ekey11	dkey7	dkey7	dkey7
K12	-	-	-	-	ekey12	ekey12	dkey6	dkey6	dkey6
K13	-	-	-	-	ekey13	ekey13	dkey5	dkey5	dkey5
K14	-	-	-	-	-	ekey14	dkey4	dkey4	dkey4
K15	-	-	-	-	-	ekey15	dkey3	dkey3	dkey3
K16	-	-	-	ekey12	ekey14	ekey16	dkey2	dkey2	dkey2
K17	-	-	-	ekey13	ekey15	ekey17	ekey1	ekey1	ekey1

4.1.2 라운드키 생성 모듈 구현

ARIA-N-VAR-ED에 필요한 라운드키를 생성하기 위한 모듈 구조는 그림 5와 같으며, RoundKey MUX는 각 모드

에 따라 표 6와 같이 컨트롤 신호에 따라 라운드키를 출력한다.

4.2 동작

Mode_A 와 Mode_B 신호에 따라 표 3과 같이 모드가 정해지며, 각 모드에 따라 암호화 모듈의 멀티플렉서는 표 5와 같이 동작하며, 라운드키 생성 모듈은 표 6에 따라 라운드키를 출력한다.

4.2.1 키 초기화

키 초기화는 키 길이가 128-비트일 경우 표 5의 Key128 모드에 따라 멀티플렉서를 조정하여, 키 입력을 12 클록 동안 유지하여야 키 초기화 레지스터(*W0, W1, W2, W3*)가 업데이트 한다. 키 길이가 192, 156-비트일 경우, 입력 인터페이스가 128-비트이므로 상위 128-비트를 먼저 저장하기 위해서 Key128모드로 먼저 설정하여 상위 128-비트를 *W0* 레지스터에 저장하고 나서, 키 길이에 따라 Key192, Key256모드로 설정하여 하위 64, 128-비트 정보를 입력하여 12 클록 동안 유지하여 나머지 키 초기화 레지스터(*W1, W2, W3*)를 업데이트 한다.

4.2.2 암호화

암호화는 키 길이에 따라서 표 5의 Enc128, Enc192, Enc256모드에 맞게 멀티플렉서 신호를 조정한다. 멀티플렉서 신호에 따라서 Enc128의 경우에 11번째 라운드의 출력이 최종라운드로 들어가며, Enc192의 경우에는 13번째 라운드의 출력이 최종라운드로 들어간다. Enc256일 때는 15번째 라운드의 출력이 최종라운드로 전달된다.

4.2.3 복호화

복호화는 키 길이에 따라서 표 5의 Dec128, Dec192, Dec256모드에 맞게 멀티플렉서 신호를 조정한다. 멀티플렉서 신호에 따라서 Dec128의 경우에 데이터 입력은 5번째 라운드로 들어가며, Dec192의 경우에는 3번째 라운드로 들어간다. Dec256일 때는 1번째 라운드로 전달된다.

5. 성능 비교 분석

고속처리를 위한 ARIA 하드웨어는 Xilinx Virtex2-pro의 XC2VP30-7을 이용하여 구현하였다. 구현 결과는 6437 slices와 128 BRAMs의 면적을 차지하고 최대 동작 주파수 192.9 MHz에서 24.6 Gbit/s의 데이터 처리율을 보였다. FPGA에서 feedback 라운드 구조의 기존의 박진섭[5]의 연구결과와 비교해 보았을 때, slice 면적은 4.3배, BRAM의 면적은 8배가 더 필요하지만 데이터 처리율에 있어서 50배의 성능 향상이 있다. 따라서 기존의 feedback 라운드 구조에 비해 제안한 ARIA-N-VAR-ED-B가 고속처리에 더 적합한 것을 알 수 있다.

또한, 안정성 면에서 동급으로 평가받는 AES 알고리듬의

표 7 ARIA 성능표
Table 7 ARIA performance

	ARIA-	Platform	Slices	BRAMs	Max. Freq. (MHz)	Throughput (Gbps)
박진섭[5]	F-128-EB-B	XCV 1600E-8	1491	16	46.5	0.496
Proposed	**N-VAR-ED-B**	**XC2 VP30-7**	**6437**	**128**	**192.9**	**24.6**

표 8 AES 성능표
Table 8 AES performance

	AES-N-128-	Device	slices	BRAMs	Max. Freq. (MHz)	Throughput (Gbps)
Kotturi[13]	E	XC2VP20	5408	200	232.6	29.7
	ED	-	-	-	125.6	16.7
Hodjat[14]	E	XC2VP20 -7	9446	-	169.1	21.6
Fu[15]	E	XC2V1000	17887	-	212.5	27.1
GMU[16]	ED	XCV1000E -8	9199	80	134.5	16.0
Rizk[17]	E	Virtex4	18855	-	222.6	28.5
	D	Virtex4	20155	-	180.3	23.0

연구결과를 조사하여 그 성능과 비교를 하였다. AES는 1997년 미국의 NIST에서 기존 암호 DES를 대체하기 위해서 새로 채택한 알고리즘이다. ARIA와 마찬가지로 128-비트의 데이터 블록을 대상으로 암호화, 복호화를 하며, 128, 192, 256-비트의 키를 사용할 수 있다. 키 길이에 따라서 라운드를 10, 12, 14회 반복하는 구조로 되어 있다. 표 13의 AES의 하드웨어 구현을 살펴보면 대부분 키 길이 128-비트의 암호화 전용으로 이루어져 있다. 반면 제안하는 ARIA 하드웨어는 128, 192, 256-비트의 키를 사용할 수 있으며, 암호화, 복호화가 모두 이루어진다. 따라서 동등한 성능 비교를 위해서 다음을 고려해야 한다. ARIA의 경우 암호화와 복호화가 동일한 라운드 구조를 이용할 수 있으나 AES의 경우에는 그렇게 할 수 없어, 복호화가 가능하기 위해서는 2배의 면적이 필요하다. 그리고 AES는 키 길이 128-비트를 사용할 경우 10라운드를 반복해야 하고, 256-비트를 사용할 때는 14라운드를 반복해야 한다. 따라서 키 길이가 256-비트인 AES 하드웨어는 키 길이가 128-비트인 경우 보다 40%의 면적이 추가적으로 증가한다. 한편 제안한 ARIA-N-VAR-ED-B는 치환계층을 BRAM으로 구현하였는데, 정량적으로 비교하기 위해서 BRAM을 LUT로 변환하면, 8개의 BRAM이 1024 slices로 구현되기 때문에 전체 면적은 22778 slices 추정할 수 있다. 따라서 Hodjat[20]의 경우 AES-256에 맞게 암호화, 복호화 모듈을 함께 구현했을 때 추정되는 면적은 9446 slices의 280%인 26448 slices 임으로 제안하는 ARIA 하드웨어에 비해 1.16배가 크다. Rizk[24]의 경우에는 단순히 암호화, 복호화 모듈의 면적을 더해보면 39010 slices를 차지하므로 제안하는 ARIA 하드웨어에 비해 1.7배의 면적을 차지한다.

Throughput의 경우 kotturi[19], Fu[21], Rizk[24]의 구현 결과에 비해 암호화 과정만 사용할 때 제안하는 구조의 Throughput이 떨어지는 것으로 나타난다. 그러나 kotturi[13]의 경우 복호화 모듈을 포함하여 구현하였을 때 최대 주파수가 125.6MHz로 떨어졌으며, Rizk[24]의 경우 복호화 모듈은 암호화 모듈에 비해 동작 속도가 느려지기 때문에, 암호화를 해서 보낸 정보를 복호화로 받아본다면 실제 전체 시스템은 복호화의 동작 속도를 따를 것이다. 그리고 본 논문에서 구현한 ARIA가 여러 가지 모드를 지원하기 위한 멀티플렉서로 인해 전파시간이 길어짐으로 인한 손실을 감안해야 한다. 대등한 성능비교를 위하여 키 길이를 고정하거나, 암호화만 지원하는 구조로 한다면 제안한 구조의 성능은 월등할 것이다.

6. 결 론

본 논문에서는 대용량 데이터의 고속처리를 위한 국내 표준 블록 암호인 ARIA의 하드웨어 구현을 위해 비순환형 mixed inner and outer round pipelining 구조를 적용하였다. 그리고 ARIA-128, ARIA-192, ARIA-256의 암호화, 복호화가 모두 가능하도록 선택적 컨트롤 신호를 적용함으로써 활용도를 높혔다. 면적을 줄이기 위한 방법으로 키 초기화 모듈을 암호화 모듈의 라운드 함수와 공통 사용되도록 하였다.

하드웨어 실제 구현은 Virtex2-pro에서 하였으며, 그 결과 면적은 6437 slices, 128 BRAMs를 차지하였고, 최대 동작 주파수 192.9MHz에서 24.6Gbit/s의 데이터 전송률을 나타내었다. 이 결과는 동급의 경쟁 알고리즘인 AES의 하드웨어 연구와 비교하였을 때, 면적 면에서 AES가 암호화, 복호화 모듈을 개별적으로 설계해야 하는데 비해서 ARIA는 암호화 모듈의 라운드 구조를 이용하여 복호화가 가능하기 때문에 효율적인 것을 확인하였다. 그리고 속도 면에서 여러 가지의 AES의 하드웨어와 대체적으로 대등한 수준임을 확인하였다.

네트워크에서 보안을 위한 암호화/복호화 과정은 고속 처리에 있어서 병목현상의 요인으로 작용하므로 빠른 처리를 위해서 전용 하드웨어화가 필요하다. 따라서 본 논문에서 제안하는 ARIA 하드웨어는 네트워크에서 라우터와 같은 대량의 데이터를 고속으로 처리해야 하는 장치에 보안성을 높이기 위한 암호 시스템으로 유용할 것이다.

참 고 문 헌

[1] 임용진, 홍진, 지성택, "스트림 암호의 발전 방향" 한국정보과학회, 정보과학회지 제23권 제1호, pp. 40-45, 2005. 1.

[2] http://www.nsri.re.kr/kor/aria.html

[3] Daesung Kwon, Jaesung Kim, Sangwoo Park et al. "New block cipher: ARIA", In Proc. Information Security and Cryptology (ICISC'03), Seoul, Korea, LNCS 2971, Springer-Verlag, pp. 432-445, November 27~28, 2003.

[4] 장환석, 이호정, 구본욱, 송정환, "64비트 마이크로프로세서에 적합한 블록암호 ARIA 구현방안", 한국정보보호학회, 정보보호학회지 제16권 제3호, pp. 63-74, 2006. 6.

[5] 박진섭, 윤연상, 김용대, 양상운, 장태주, 유영갑, "ARIA 암호 알고리즘의 하드웨어 설계 및 구현", 대한전자공학회, 전자공학회논문지 제42권 SD편 제4호, pp. 29-36, 2005. 4.

[6] Jinsub Park, Young-Dae Kim, Sangwoon Yang, Younggap You, "Low power compact design of ARIA block cipher", Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on, May 2006.

[7] S. Yang, J. Park, and Y. You, "The smallest ARIA module with 16-bit architecture", ICISC 2006, LNCS 4296, pp. 107-117 , 2006.

[8] 유권호, 구본석, 양상운, 장태주, "경량화된 확산계층을 이용한 32-비트 구조의 소형 ARIA 연산기 구현", 한국정보보호학회, 정보보호학회논문지 제16권 제6호, pp. 15-24, 2006. 12.

[9] Yeong-Jae Oh, Hanho Lee, Chong-Ho Lee, "Dynamic Partial Reconfigurable FIR Filter Design," Reconfigurbale Computing: Architectures and Applications (ARC 2006), LNCS3985, Mar. 2006. (SCIE).

[10] Chang-Seok Choi and Hanho Lee, "A Self-Reconfigurable Adaptive FIR Filter System on Partial Reconfiguration Platform," IEICE Transactions on Information and Systems, vol. E90-D, no. 12, pp. 1932-1938, Dec. 1. 2007. (SCIE).

[11] NSRI: ARIA Algorithm Specification, http://www.nsri.re.kr/ARIA/doc/ARIA-specification.pdf, 2004 (in Korea).

[12] Bonseok Koo, Gwonho Ryu, Taejoo Chang, Sangjin Lee, "Design and Implementation of Unified Hardware for 128-Bit Block Ciphers ARIA and AES", 한국전자통신연구원, ETRI Journal 제29권 제6호, 2007. 12.

[13] D. Kotturi, S.M. Yoo, and J. Blizzard, "AES Crypto Chip Utilizing High-Speed Parallel Pipelined Architecture," IEEE Int'l Symp. on Circuits and Systems(ISCAS-05), Kobe, Japan, pp.4653-4656, May 2005.

[14] A. Hodjat, I. Verbauwhede, A 21.54 Gbits/s fully pipelined AES processor on FPGA, in: IEEE Symposium on Field-Programmable Custom Computing Machines, 2004.

[15] Yongzhi Fu, Lin Hao, Xuejie Zhang and Rujin Yang, "Design of an extremely high performance counter mode AES reconfigurable processor", Embedded Software and Systems, 2005. Second International Conference on Embedded Software and Systems(ICESS'05), 16-18 Dec. 2005.

[16] George Mason University, Hardware IP Cores of

Advanced Encryption Standard AES Rijndael, http://ece.gmu.edu/crytpo/rijndael.htm

[17] M.R.M. Rizk, M. Morsy, "Optimized Area and Optimized Speed Hardware Implementatoins of AES on FPGA" International Design and Test Workshop, 2007 2nd, pp. 207-217, 16-18 Dec. 2007.

저 자 소 개

하 성 주 (河 成 珠)

1982년 11월 10일생. 2006년 인하대 항공우주공학과 졸업. 2006년~현재 동 대학원 정보통신공학과 석사과정

Tel : 032-860-7396

E-mail : melon@inhaian.net

이 종 호 (李 鍾 浩)

1953년 4월 14일생. 1976년 서울대 전기공학과 졸업. 1978년 동 대학원 전기공학과 졸업(공석). 1986년 미국 아이오와 주립대 전기 및 컴퓨터 공학과 졸업(공박). 1986년 ~ 1989년 미국 노틀담대학교 조교수. 1997년 ~ 1998년 인하대 집적회로 설계센터 소장. 2004년 ~ 2005년 브라운대학교 두뇌 및 신경회로망 연구소 방문교수. 1989년 ~ 현재 인하대학교 정보통신공학과 교수. 2000년 ~ 현재 수퍼지능기술연구소 소장

Tel : 032-860-7396

E-mail : chlee@inha.ac.kr