Rijndael 블록암호 알고리즘의 FPGA 구현

구본석, 이상한

국가보안기술연구소, 응용기술연구부

FPGA Implementation of Rijndael Algorithm

Bon-seok Koo, Sang-han Lee

Application Technology Development Department, NSRI.

요 약

본 논문에서는 차세대 표준 알고리즘(AES: Advanced Encryption Standard)인 Rijndael 알고리즘의 고속화를 FPGA로 구현하였다. Rijndael 알고리즘은 미국 상무부 기술 표준국(NIST)에 의해 2000년 10월에 차세대 표준으로 선정된 블록 암호 알고리즘이다. FPGA(Field Programmable Gate Array)는 아키텍쳐의 유연성이 가장 큰 장점이며, 근래에는 성능면에서도 ASIC에 비견될 정도로 향상되었다. 본 논문에서는 128비트 키 길이와 블록 길이를 가지는 암호화(Encryption)블럭을 Xilinx VirtexE XCV812E-8-BG560 FPGA에 구현하였으며 약 15Gbits/sec의 성능(throughput)을 가진다. 이는 현재까지 발표된 FPGA Rijndael 알고리즘의 구현 사례 중 가장 빠른 방법 중의 하나이다.

I. 서론

암호시스템은 크게 대칭키 암호 시스템과 비대칭키 암호 시스템으로 분류될 수 있다. 대칭키 암호 시스템은 다시 블록 암호 알고리즘 시스템과 스트림 암호 알고리즘 시스템으로 나누어 볼 수 있다. 블록 암호 알고리즘 중에는 1977년 미국의 국가 암호 표준으로 채택된 DES(Data Encryption Standard)와 Triple DES가 가장 널리 이용되고 있다. 그러나 1998년을 기점으로 DES는 표준 기한이 만료되었고, Triple DES는 2003년 만료될 예정이다. 이로 인해 새로운 표준 암호 알고리즘의 필요성이 제기되었고, 미국 NIST(National Institute of Standards and Technology)에서는 향후 정부와 상업계에서 사용할 수 있는 강한 암호화 알고리즘 표준 제정을 위해 AES(Advanced Encryption Standard) 개발 과제를 지난 3년여 동안 수행해왔다.[1] 그리고 지난 2000년 10월 Rijndael이 최종 AES 알고리즘으로 채택되었다.

본 논문에서는 FPGA를 이용한 Rijndael 알고리즘의 고속 구현기법을 설명한다. Rijndael 알고리즘에는 LUT(LookUp Table) 형태의 구조가 많이 사용되는데 이는 RAM이나 ROM 등으로 구현하기 적합하다. 따라서 내부 RAM 블록(BRAMs)을 최대 280개까지 가지는 Xilinx VirtexE 디바이스를 구현에 채택하였다. 현재까지 가장 높은 성능을 가지는 FPGA 구현 사례는 Chodowiec, Khuon 그리고 Gaj가 발표한 논문이다.[2] 이들은 3개의 Virtex XCV1000 FPGA를 이용하여 12160 Mbits/sec 속도를 가지는 시스템을 구현하였다. 단일 FPGA 칩을 이용한 사례로는 McLoone과 McCanny의 논문이 있으며, VirtexE XCV812E를 사용하여 7Gbits/sec의 속도를 가지도록 구현하였다.[3] 또한, 가장 빠른 소프트웨어 구현 사례는 Brian Gladman의 논문으로 팬티엄 III 933MHz에서 325MHz의 속도를 가진다.[5]

2장에서는 Rijndael 알고리즘의 특징과 구조에 대해 살펴보고 3장에서는 VirtexE 디바이스를 이용한 암호 알고리즘의 하드웨어 설계 및 구현을 설명하며, 4장에서는 성능 및 결과 고찰, 그리고 마지막 5장에서는 결론 순으로 설명한다.

II. Rijndael 알고리즘

1. Rijndael 알고리즘의 특징

Rijndael 알고리즘은 표 1에서 보는 바와 같이 가변 블록 길이와 가변 키 길이를 갖는 반복 구조

의 블록 암호 방식이며, 블록 길이와 키 길이는 128, 192, 256비트로 독립적으로 지정될 수 있다. 또한, 라운드 수는 블록길이와 키 길이에 의해 결정되며, 아래 표 1과 같다.

표 1 블록 길이와 키 길이에 따른 라운드 수

Nr(라운드 수)		블록 길이		
		128 비트 (Nb=4)	192 비트 (Nb=6)	256 비트 (Nb=8)
키길이	128비트 (Nk=4)	10	12	14
	192 비트 (Nk=6)	12	12	14
	256 비트 (Nk=8)	14	14	14

본 논문에서는 블록 길이와 키 길이가 128비트인 암호화 블록(Encryptor) 구현의 경우로 제한한다. 따라서 전체 라운드 10 라운드가 되며, 암호화 블록은 초기 데이터/키 덧셈, 9개의 라운드, 그리고 최종 라운드로 구성된다. 또한 키 스케줄 블록에서는 초기키 값을 확장하여 매 라운드마다 다른 키를 입력시킬 수 있도록 라운드 키를 생성한다. 이를 표현한 그림이 그림 1이다.

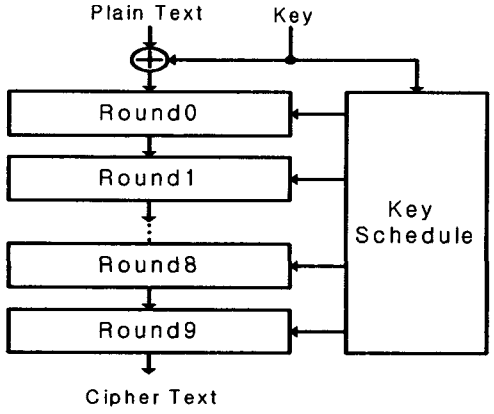

그림 1 Rijndael 알고리즘의 암호화 블록도

Rijndael 알고리즘의 매 라운드는 다음의 4가지 연산으로 구성된다.

- ByteSub: 바이트 단위의 고정된 치환을 수행하며, 각 바이트는 State의 다른 바이트값에 독립적으로 치환된다.
- ShiftRow: State의 값을 변경시키지 않으면서 State의 바이트들의 위치를 교환한다.
- MixColumn: State의 열에 대해 GF(2^8)상에서 정의되는 행렬 곱셈 연산을 수행한다.
- KeyAdd: State의 모든 바이트에 라운드 키를 더한다.

여기서 라운드 0부터 라운드 8까지는 그림 2와 같은 구성을 가진다. 하지만 마지막 라운드 에서는 MixColumn 연산이 제외된다. 또한 State는 4행 X Nb열(128비트인 경우 4) 바이트로 구성되는 블록 단위를 나타낸다.

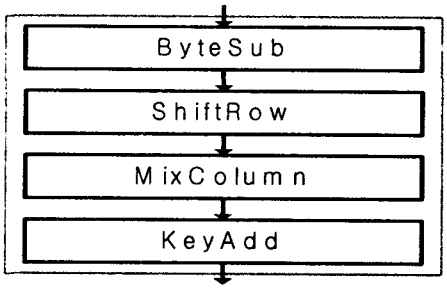

그림 2 라운드의 구성

2. ByteSubstitution

ByteSub는 State Byte들에 대한 각각의 독립적인 비선형 치환이다. 치환 테이블은 역변환이 가능하며 두개의 transformation으로 구성된다. 먼저 GF(2^8)에서의 곱셈의 역원을 구한 후, 식 (1)의 지정된 행렬을 이용하여 GF(2)상에서의 변환을 취한다. 이를 직접 연산기로 구현하면 회로가 복잡해지고 속도가 저하되므로 LUT(Look Up Table)을 이용하여 구현하게 된다. 이러한 LUT는 Xilinx VirtexE 디바이스 내부의 BRAM을 사용하여 구현하였다.

$$\begin{pmatrix} y0 \\ y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \\ y7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x0 \\ x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad (1)$$

3. ShiftRow

ShiftRow는 첫 번째 행을 제외한 나머지 행들을 각각 서로 다른 OffSet만큼 Byte단위로 Cyclic Shift를 취한다. 이를 나타내는 그림이 그림 3이

다. 그림에서 알 수 있는 것처럼 ShiftRow는 하드웨어적으로 매우 간단하게 구현가능하다. 또한 복호화(Decryptor) 블록의 구현을 위해서는 단지 반대 방향의 Shift를 취하기만 하면 된다.

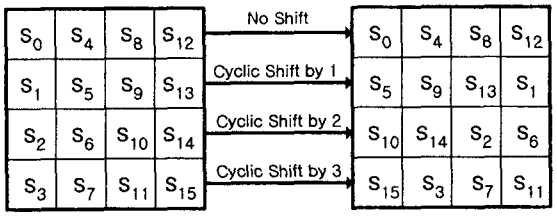

그림 3 ShiftRow 변환

4. MixColumn

MixColumn 연산은 State의 열 단위로 수행된다. 각 열은 GF(2^8)상의 다항식으로 처리되며, 다음 식 (2)와 같이 계산된다.

$$b(x) = a(x) \times c(x) \mod x^4+1, \quad \text{여기서 } c(x) = 3x^3 + x^2 + x + 2 \qquad (2)$$

MixColumn 연산의 하드웨어 구조는 그림 4와 같다. 그림 4에서 xtime은 $x \times a(x) \mod x^4+1$ 연산을 나타낸다. 즉 $a(x)$의 최상위 비트가 1이면 Left Shift를 한 다음 x^4+1을 가산하고, 최상위 비트가 0이면 단지 Left Shift만을 수행하게 된다.

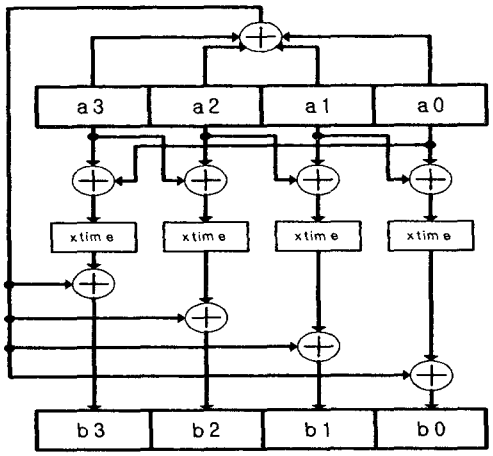

그림 4 MixColumn연산의 하드웨어 구조

5. Key Schedule

키 스케줄은 암호화 키의 확장(key expansion)과 각 라운드에 해당하는 키 선택(key selection)으로 이루어진다. 키 확장 함수는 다음과 같다

```
KeyExp(byte Key[4*Nk], word W[Nb*(Nr+1])
{
    for (i=0;i<Nk;i++)
    W[i] = (Key[4*i],Key[4*i+1],
            Key[4*i+2], Key[4*i+3]);
    for(i=Nk;i<Nb*(Nr+1);i++)
    {
        temp = W[i-1];
        if (i%Nk == 0)
            temp = SubByte(RotByte(temp))
                    ^Rcon[i/Nk];
        W[i] = W[i-Nk]^temp;
    }
}
```

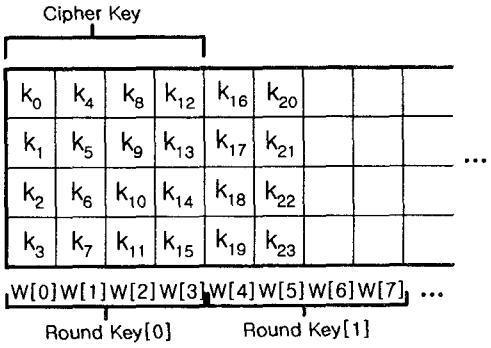

그림 5 Cipher Key의 확장

III. Rijndael의 FPGA구현

본 논문에서 제안하는 Rijndael 디자인 기법은 Sub-Pipelining 기법이다. Sub-Pipelining기법은 각 round 내부 블록들의 적당한 위치에 레지스터를 두는 기법이다. 이러한 방법은 latency를 떨어뜨리지만 Throughput을 최대화하는 데는 대표적인 방법이다. 본 논문에서는 라운드 함수 내부의 각 연산마다 레지스터를 삽입하여 4단 Pipelining, 전체적으로는 40단 Pipelining 구조로 디자인을 하였다.

앞서 설명한 바와 같이 ByteSub 연산은 LUT(LookUp Table)를 이용하였다. Rijndael의 각 라운드마다 8비트 입력/8비트 출력 LUT(LookUp Table)가 16개 필요하므로 전체 10라운드를 계산하는 데는 160개의 LUT가 필요하다. 실제에 있어

서는 Xilinx VirtexE 디바이스 내부에 있는 BlockSelectRAM(BRAM) 메모리로 이러한 LUT를 구현하였다. BRAM의 구조는 그림 6과 같다. 그림에서 보는 바와 같이 하나의 BRAM은 두개의 256 x 8 비트 RAM으로 사용될 수 있다. 따라서 매 라운드마다 8개의 BRAM이 사용되게 된다. 또한 키 스케줄 블록에서도 ByteSub연산이 필요한데 이 또한 BRAM으로 구현한다. 전체 10라운드의 확장키를 모두 계산하기 위해서는 총 40개의 8비트 입력/8비트 출력 LUT, 즉 20개의 BRAM이 필요하다. 하지만, 각 라운드 키를 해당 라운드가 수행될 때 병렬로 계산되도록 할 수 있으므로 2개의 BRAM만을 사용하여 키 스케줄을 구현할 수 있다. 따라서 본 논문의 디자인에서는 모두 82개의 BRAM이 사용되었으며, 그 중 80개는 10 라운드를 수행하는데 필요하며, 나머지 2개는 키 확장을 수행하는 데 사용하게 된다.

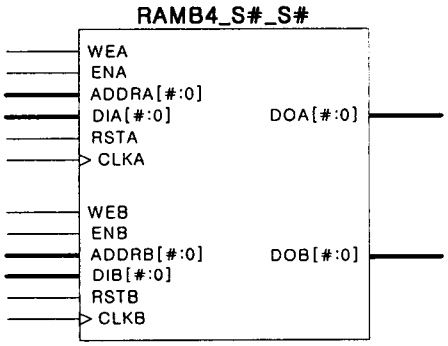

그림 6 VirtexE의 BRAM 구성

IV. 성능 및 결과 고찰

Xilinx Foundation Series 3.1i와 Verilog-XL 시뮬레이터를 이용하여 지금까지 설명한 Rijdnael 디자인을 FPGA에 구현하였다.

디바이스는 앞서 설명한 바와 같이 VirtexE XCV812E-8BG560를 사용하였으며, 128비트 블록 길이과 128비트 키 길이를 갖는 암호화 블록을 구현하는데 3084 CLB slices(32%)와 82 BRAMs (29%)가 사용되었다. 본 디자인은 시스템 클락이 최대 120MHz의 속도까지 동작하며 이는 최대 15Gbits/sec의 성능(throughput)을 가진다. 이러한 결과는 지금까지 발표된 FPGA Rijndael 알고리즘의 구현 사례 중 가장 높은 가장 빠른 방법 중의 하나이다. 이러한 높은 성능을 가질 수 있는 요인으로는 디자인을 Sub-Pipelining하여 Throughput을 최대한 높을 수 있게 한 점과 VirtexE 디바이스 내부의 BRAM을 사용하여 ByteSub 연산의 부담을 최소화한 점을 들 수 있다.

V. 결론

본 논문에서는 FPGA를 이용한 Rijndael 알고리즘의 효율적인 구현법을 제시하고 있다. 암호화 블록을 구현하여 15Gbits/sec의 데이터 처리 속도를 갖는 시스템을 구현하였으며, 이는 지금까지 소개된 FPGA구현 기법보다 2배 이상 빠르며, 소프트웨어보다는 40배 이상 빠른 성능을 나타낸다. Rijndael은 앞으로, IPSec 프로토콜이나 ATM 셀 암호화 등 다양한 분야에 있어서 DES를 대신하여 사용될 전망이다. 따라서 고속으로 동작하는 Rijndael 암호화/복호화 블록의 디자인에 대한 연구는 유/무선 실시간 통신 시스템의 개발에 중요한 요소라고 할 수 있다.

또한, 앞으로 CBC(Cipher Block Chaning), OFB(Output FeedBack), CFB(Cipher FeedBack) 등의 다양한 블록 암호 동작 방식에 대한 효율적 암/복호화 블록 디자인이 연구될 예정이다.

참고문헌

[1] J.Daemen, V.Rijmen, "The Rijndel Block Cipher: AES Proposal," *First AES Candidate*, pp. 20-22, Aug. 1998.

[2] p.Chodowiec, P.Khuon, K.Gaj, "Fast Implemetation of Secret-Key Block Ciphers Using Mixed Inner- and Outer-Pipelining," *FPGA 2001.*, pp. 11-13, Feb. 2001.

[3] M. McLoone, J.V McCanny, "High Perfomance Single-Chip FPGA Rijndael Algorithm Implementations," *CHES 2001*, pp. 68-79, May 2001.

[4] 이종수, 염동복, 이병윤, 박종서, "차세대 표준 알고리즘 RIJNDAEL의 고속 암호칩 설계," WISC 2001, pp. 52-63, 2001년 9월.

[5] Brian Gladman, "The AES Algorithm (Rijndael) in C and C++," URL: http://fp.gladman.plus.com/cryptography_technology/rijndael/index.htm, April 2001.