Whirlpool 해쉬 함수의 효율적인 하드웨어 구현

박진철* · 신경욱**

*국립 금오공과대학교

An Efficient Hardware Implementation of Whirlpool Hash Function

Jin-Chul Park* · Kyung-Wook Shin**

*Kumoh National Institute of Technology

E-mail : 20060593@kumoh.ac.kr

요 약

본 논문에서는 ISO/IEC 10118-3의 표준인 Whirlpool 해쉬 함수의 효율적인 하드웨어 설계와 FPGA 검증에 대해 기술한다. Pipelined small LUT를 이용하여 동작 타이밍을 최적화하였으며, Whirlpool 블록암호와 key schedule을 병렬로 사용하여 throughput을 개선하였다. 키 스케쥴에서 키 덧셈부분에 rom과 xor 게이트를 사용하지 않고 인버터와 mux로 구현하여 면적을 최적화하였다. Virtex5-XC5VSX50T를 사용하여 FPGA 검증을 하였고 최대 동작 주파수는 약 151MHz이며, 약 950Mbps의 성능을 가진다.

ABSTRACT

This paper describes an efficient hardware implementation of Whirlpool hash function as ISO/IEC 10118-3 standard. Optimized timing is achieved by using pipelined small LUTs, and Whirlpool block cipher and key schedule have been implemented in parallel for improving throughput. In key schedule, key addition is area-optimized by using inverters and muxes instead of using rom and xor gates. This hardware has been implemented on Virtex5-XC5VSX50T FPGA device. Its maximum operating frequency is about 151MHz, and throughput is about 950Mbps.

키워드

Whirlpool, hash function, block cipher, integrity, non-Feistel, Miyaguchi-Preneel

Ⅰ. 서 론

평문의 무결성을 필요로 하는 메시지 인증 코드(Message Authentication Code)로 사용될 뿐만 아니라 디지털 서명에도 사용되는 Whirlpool 해쉬 함수는 Vincent Rijmen과 Paulo S. L. M. Barreto가 설계하였다.

또한, Whirlpool 해쉬 함수는 초기의 S-box의 단점을 보완하고 수정한 후에 ISO/IEC 10118-3로 표준이 되었고 2003년 2월에 NESSIE(New European Schemes for Signatures, Integrity and Encryption)가 승인하였다.[1][2] Whirlpool 해쉬 함수는 블록암호인 AES의 구조와 유사하며 non-Feistel 구조로서 해쉬 알고리듬에 사용되는 블록암호를 기반으로 설계된 것을 알 수 있다. Whirlpool 해쉬 함수는 가변길이의 평문을 입력으로 받지만, Miyaguchi-Preneel 구조이므로 한 블록당 512비트 길이의 고정된 입력을 가진다. Miyaguchi-Preneel 구조는 평문, 암호문 그리고 암호 키를 모두 xor 연산을 하여 메시지 다이제스트를 생성하므로 외부 공격에 대한 강인한 방어능력을 가진다.[3]

본 논문에서는 Whirlpool 해쉬 함수의 효율적인 하드웨어 설계에 대해 기술하며 저면적과 고성능을 가지는 최적화된 구조를 제안한다

II. 본 론

2.1 메시지 패딩

Whirlpool 알고리듬은 가변 길이의 평문을512 비트의 고정 길이로 분할하여 처리한다 이를 위해 그림 1에 나타낸 것처럼 패딩을 해야 한다. 평문에 붙이는 패딩의 첫 번째 비트는‘1’이고 나머지 비트들은 ‘0’으로 이루어진다. 패딩된 평문의 길이는 256비트의 홀수 배수가 된다. 그리고 평문의 길이를 정의하는 256비트의 평문 길이 필드가 추가되어서 평문의 최대길이는2^{256}이고 메시지는 512비트의 배수가 된다. 따라서 메시지는 512비트 블록 단위로 쪼개어져 Whirlpool 해쉬 함수 입력으로 들어간다. (전부 패딩된 평문을 메시지라고 부른다.)

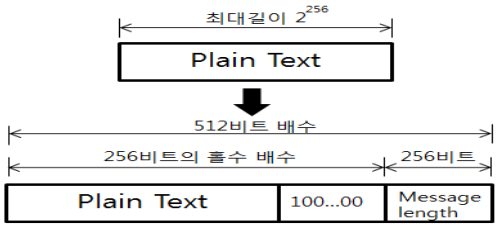


그림 1. 평문 패딩
Fig 1. Plain Text Padding

2.2 블록 암호 W[K]

Whirlpool 해쉬 함수는 초기 키 가산 $\sigma[k_0]$를 갖는 pre-round와 non-linear layer γ, cyclical permutation π, linear diffusion layer θ, 그리고 키 가산 $\sigma[k]$를 라운드 함수로 갖는 10 라운드 블록암호이다. 각 입출력 길이가 512비트이고 11개의 라운드 키를 사용한다 블록암호는 그림 2와 같다.

512비트 메시지 블록 (행 매트릭스)이 블록암호로 들어와서 512비트 메시지 state(8x8 정방행렬)로 변환되고 γ의 입력으로 들어간다. 입력으로 들어온 메시지 state는 S-box를 통해 연산되어 새로운 메시지 state로 대체된다. S-box는 full LUT(Look Up Table)로 구현될 수 있고 혹은 4개의 작은non-linear layer LUT, 하나의 의사난수 LUT 그리고 xor 게이트로 구현될 수 있다. 또한 4개의 작은 LUT 그리고 하나의 의사난수 LUT을 부울함수로 표현할 수 있다. S-box에 의해 연산된 메시지 state는 π의 입력으로 들어가고 state의 각 열을 바이트 단위로 아래쪽으로 순환 shift를 한다. 첫 번째 열은 그대로 두고 두 번째 열은 1-바이트, 세 번째 열은 2-바이트 ... 여덟 번째 열은 7-바이트만큼 순환 shift를 한다. π는 state에서 어떤 연산을 구현하지 않고 바이트의 재배치만으로 구현할 수 있지만 본 논문에서는 64비트 데이터 패스를 가지므로 mux와 레지스터가 필요하다. 그런 다음 메시지 state는 θ의 입력으로 들어와서 state의 한 행과 상수행렬

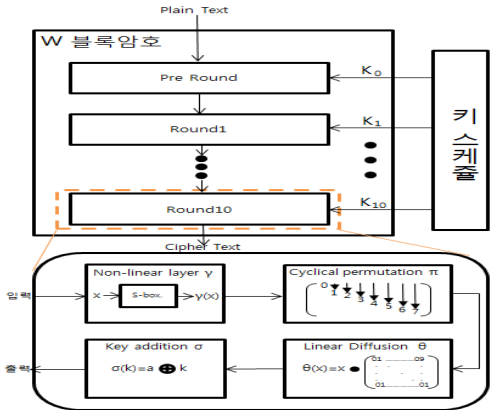


그림 2. 블록 암호 W
Fig 2. Block cipher W

의 한 행을 GF(2^8)상에서 0x11D를 모듈로 곱셈 연산을 수행한다. 라운드 함수의 마지막 과정으로써 $\sigma[k]$는 θ의 출력과 해당되는 라운드 키 K와 바이트 단위로 xor 연산을 수행한다.

2.3 키 스케줄

Whirlpool에서 라운드 키를 생성하기 위해서 블록 암호의 라운드 함수와 동일한 알고리듬을 사용한다. 키 스케줄은 블록 암호의 라운드 함수와 마찬가지로 non-linear layer γ, cyclical permutation π, linear diffusion layer θ, 그리고 키 가산 $\sigma[k]$를 라운드 함수로 갖는다. 라운드 과정을 거쳐 10개의 라운드 키 $K^0,\ldots,K^{10}$를 생성한다. K^0는 초기의 키 값으로서 블록 암호의 pre-round를 수행할 때 사용한다. 열개의 라운드 키인 $K^1,\ldots,K^{10}$는 해당되는 각 라운드의 키 가산 과정에서 사용된다. 블록 암호의 라운드 함수와 마찬가지로 키 스케줄에서도 키 가산 과정이 필요하다. 가상의 라운드 키인 10개의 라운드상수로 키 가산 과정을 수행한다. 라운드 상수의 값을 구하는 식은 다음과 같다.

$$c_{0j}^r = S[8(r-1)+j], 0 \le j \le 7, \qquad (1)$$
$$c_{ij}^r = 0, \qquad 1 \le i \le 7, 0 \le j \le 7$$

위 식을 보면 라운드 상수행렬은 첫 번째 행만 S-box를 이용해 만들어지고 나머지 행들은 모두 ‘0’의 값을 가진다는 것을 알 수 있다. 그리고 각 라운드마다 8x8 정방행렬인 S-box의 full LUT 값을 사용하는 것을 알 수 있다. 첫 번째 라운드에서 키 스케줄의 라운드 키 값은 c_{0j}^1=[18 23 C6 E8 87 B8 01 4F]이고 나머지 행은 ‘0’들로 이루어지고 두 번째 라운드에서는 c_{0j}^2=[36 A6 D2 F5 79 6F 91 52]이고 나머지 행은 ‘0’들로 이루어진다. 이런 방법으로 10개의 키 스케줄의 라운드 키 값을 생성한다.

Ⅲ. Whirlpool Hardware 설계

본 논문에서 설계된 Whirlpool 해쉬 함수의 하드웨어 구조에 대해 알아본다 설계된 하드웨어 전체 구조는 그림 3과 같으며, 하드웨어를 제어하는 Control Logic, 제어를 위해 라운드를 계수하는 Round Counter과 클럭을 계수하는 Clk Counter, 메시지 다이제스트를 생성하기 위한 블록암호와 키 스케쥴 그리고 Plain Store와 Key Store로 구성된다. 설계된 블록암호 구조는 데이터를 모으고 비선형 변환 그리고 치환을 수행하는 Buf-Shift-Sub(BSS) Manager, 비트를 확산시키는 Diffusion 그리고 키 덧셈을 하는 키 가산으로 구성된다. 설계된 키 스케쥴는 블록암호와 동일한 구조이다.

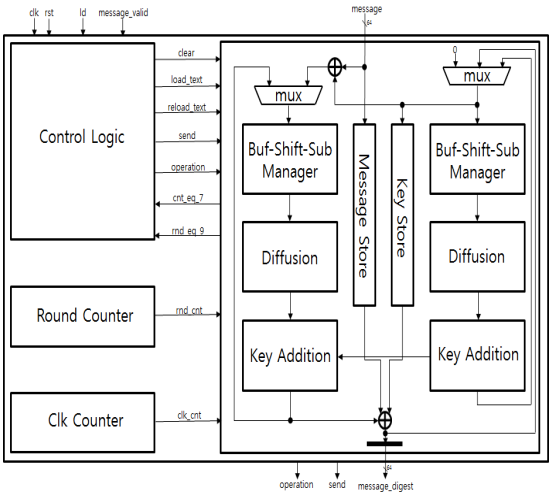


그림 3. 설계된 Whirlpool 해쉬 프로세서 구조
Fig. 3. Architecture of Whirlpool hash processor

본 논문에서는 면적을 최소화하기 위해 4개의 작은 non-linear layer LUT 그리고 하나의 의사난수 LUT와 xor 연산 구조를 채택하여 설계하였다. mux와 레지스터 그리고 S-box로 구성된 non-linear layer와 cyclical permutation 그리고 버퍼 역할을 하는 BSS 구조는 그림 4에 나타내었다. 설계된 하드웨어는 64비트 데이터 패스를 가지므로 mux와 레지스터가 필수적이다. 이 구조 내부에 S-box를 추가하여 파이프라인 레지스터 구조로 구현하여 타이밍을 최적화하였다.

linear diffusion layer는 그림 5에 구조를 나타낸 것과 같고 분배법칙과 결합법칙을 사용하여 곱셈 연산의 수가 감소되도록 설계하였다. 곱셈은 xor 게이트들로 구성된다. xor 게이트들로 구성된 linear diffusion layer에 공유 기법을 적용하여 면적을 최적화하였다. 아래의 식은 메시지에 상수행렬을 64비트 곱셈하는 과정을 보여준다.[4]

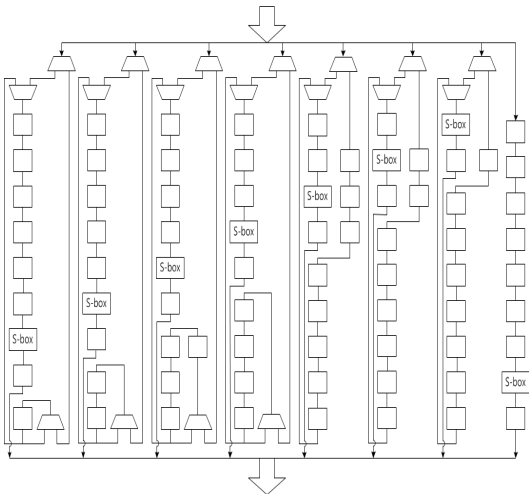


그림 4. 설계된 BSS manager 블록
Fig. 4. Designed BSS manager block

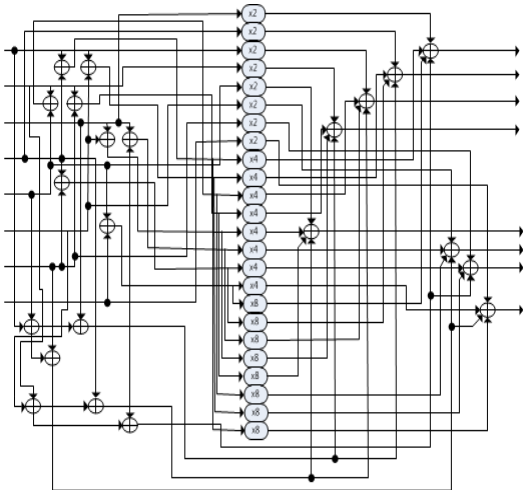


그림 5. 설계된 diffusion 블록
Fig. 5. Designed diffusion block

$$y_{0,0} = x_{0,0} \oplus (x_{0,1} \bullet 09_x) \oplus (x_{0,2} \bullet 02_x) \oplus (x_{0,3} \bullet 05_x) \oplus (x_{0,4} \bullet 08_x) \oplus x_{0,5} \oplus (x_{0,6} \bullet 04_x) \oplus x_{0,7} \quad (2)$$

식 (2)는 아래의 식으로 다시 유도된다.

$$y_{0,0} = x_{0,0} \oplus x_{0,1} \oplus x_{0,3} \oplus x_{0,5} \oplus x_{0,7} \oplus (x_{0,2} \bullet 02_x) \oplus ((x_{0,3} \oplus x_{0,6}) \bullet 04_x) \oplus ((x_{0,1} \oplus x_{0,4}) \bullet 08_x) \quad (3)$$

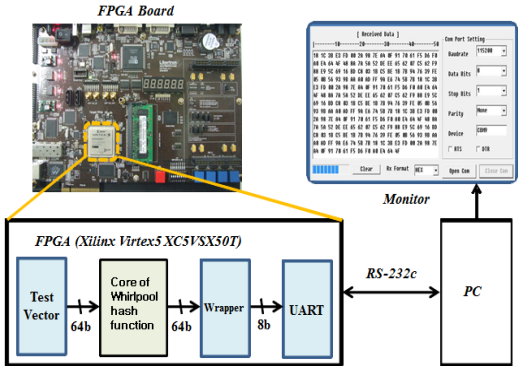
식 (3)을 보면 곱셈 연산의 수가 감소했음을 알 수 있다.

키 스케쥴의 키 가산을 구현하기 위해서는 rom과 xor 게이트가 필요하다. 그러나 본 논문에서는 상수와의 xor 연산이므로 mux와 인버터로 구현하여 면적을 최적화하였다. 예를 들면 $c^1_{0,j}$ =[18 23 C6 E8 87 B8 01 4F]을 이진수로 나타내면 [00011000 00100011 11000110

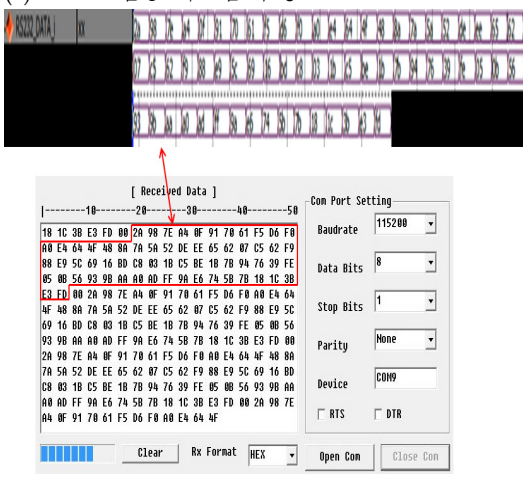
11101000 10000111 10111000 00000001 00101111] 이고 여기서 2진수로 바꾼 상수행렬 '1'인 부분은 데이터의 해당 비트를 반전하고 '0'인 부분은 그대로 두어 mux와 인버터로 구현하였다.

Ⅳ. 설계검증 및 성능평가

Whirlpool 해쉬 함수 프로세서는 Verilog HDL로 설계되었으며, 표준 문서에서 주어진 테스트 벡터를 이용하여 시뮬레이션과 검증을 수행하였다. 설계된 하드웨어는 그림 6-(a)의 방법으로 FPGA 구현을 통해 하드웨어 동작을 검증하였다 테스트 벡터를 설계된 하드웨어의 입력 데이터로 사용한 FPGA 검증결과는 그림 6-(b)와 같으며, 생성된 메시지 다이제스트가 테스트 벡터의 결과와 일치하여 설계된 하드웨어의 정상 동작을 확인하였다. FPGA 합성결과는 표 1과 같으며, 최대 동작 주파수는 약 151MHz이고 약 950Mbps의 성능을 가진다.



(a) FPGA 검증 시스템 구성도



(b) FPGA 검증 결과

그림 6. Whirlpool 해쉬 함수의 FPGA 검증
Fig. 6. FPGA verification of Whirlpool hash function

표 1. 설계된 Whirlpool 해쉬 함수의 FPGA 합성 결과
Table 1. FPGA synthesis results of designed Whirlpool hash function

	사용개수	허용개수
device	XC5vsx50t-1ff1136	
Slice Register	1283	32640
Slice LUTs	1874	32640
LUT-FFpair	2259	26391
BUFGCTRLs	1	32
동작속도	151MHz	

Ⅴ. 결 론

설계된 Whirlpool 해쉬 함수 하드웨어의 키 스케쥴에서 키 가산을 rom과 xor로 구현하지 않고 인버터와 mux로 구현하여 면적을 최적화하였으며, small LUT을 BSS의 내부에 구현하여 마치 파이프라인 레지스터를 가진 효과로 타이밍을 개선하였다. 또한 Diffusion을 구성하는 xor 게이트에 공유기법을 적용하여 면적을 최적화하였다

참고문헌

[1] Paulo S.L.M. Barreto and Vincent Rijmen, "The WHIRLPOOL Hashing Function", pp1-20, May. 2003.

[2] Akashi Satoh, "ASIC Hardware Implementations for 512-bit Hash Function Whirlpool", Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on, pp2917-2920, 2008.

[3] Behrouz A. Forouzan,"암호학과 네트워크 보안", McGraw-Hill Korea, pp1-833, Jan. 2007.

[4] Maire Mcloone and Ciaran Mcivor, "High-speed&Low Area Hardware Architectures of the Whirlpool Hash Function", Journal of VLSI Signal Processing. 47, pp47-57, 2007.