JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 21, No. 6 : 1075~1082 Jun. 2017

SHA-3과 SHAKE256 알고리듬을 지원하는 해쉬 프로세서의 하드웨어 설계

최병윤*

Efficient Hardware Design of Hash Processor Supporting SHA-3 and SHAKE256 Algorithms

Byeong-Yoon Choi*

Department of Computer Engineering, Dongeui University, Busan 47340, Korea

요 약

본 논문에서는 새로운 해쉬 알고리듬인 SHA-3과 출력 길이 확장함수인 SHAKE256을 구현하는 해쉬 프로세서를 설계하였다. 해쉬 프로세서는 성능을 극대화하기 위해 Padder 블록, 라운드 코어 블록, 출력 블록이 블록 단계에서 파이프라인 구조로 동작한다. Padder 블록은 가변길이의 입력을 여러 개의 블록으로 만들고, 라운드 코어 블록은 on-the-fly 라운드 상수 생성기를 사용하여 SHA-3와 SHAKE256에 대응하는 해쉬 및 출력 확장 결과를 생성하며, 출력 블록은 결과 값을 호스트로 전달하는 기능을 수행한다. 해쉬 프로세서는 Xilinx Virtex-5 FPGA에서 최대 동작 속도는 220 MHz이며, SHA3-512의 경우 5.28 Gbps의 처리율을 갖는다. 프로세서는 SHA-3 와 SHAKE-256 알고리듬을 지원하므로 무결성, 키 생성, 난수 생성 등의 암호 분야에 응용이 가능하다.

ABSTRACT

This paper describes a design of hash processor which can execute new hash algorithm, SHA-3 and extendable-output function (XOF), SHAKE-256. The processor that consists of padder block, round-core block and output block maximizes its performance by using the block-level pipelining scheme. The padder block formats the variable-length input data into multiple blocks and then round block generates SHA-3 message digest or SHAKE256 result for multiple blocks using on-the-fly round constant generator. The output block finally transfers the result to host processor. The hash processor that is implemented with Xilinx Virtex-5 FPGA can operate up to 220-MHz clock frequency. The estimated maximum throughput is 5.28 Gbps(giga bits per second) for SHA3-512. Because the processor supports both SHA-3 hash algorithm and SHAKE256 algorithm, it can be applicable to cryptographic areas such as data integrity, key generation and random number generation.

키워드 : 무결성, 키 생성, 해시 알고리듬, SHA-3, SHAKE256,

Key word : Data Integrity, Key Generation, Hash Algorithm, Sha-3, Shake256

Received 24 January 2017, **Revised** 03 February 2017, **Accepted** 21 February 2017

* **Corresponding Author** Byeong-Yoon Choi(E-mail:bychoi@deu.ac.kr, Tel:+82-51-890-1706)
Department of Computer Engineering, Dongeui University, Busan 47340, Korea

Open Access https://doi.org/10.6109/jkiice.2017.21.6.1075 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

정보 보호 분야에서 암호화 알고리듬만큼 중요하다고 평가되는 암호학적 해쉬 알고리듬(cryptographic hash function)은 통신 시스템과 저장 매체에 담긴 정보의 위・변조 여부를 확인하는 기술이다[1,2].

90년대부터 널리 사용되고 있는 대표적인 암호학적 해쉬 알고리듬인 Secure Hash Algorithm(SHA-1)은 MD4 알고리듬을 기반으로 미국 표준 기술 연구소인 National Institute of Standards and Technology(NIST)에서 개발하였다. 2002년에 NIST는 SHA-1을 수정하여 4개의 해쉬 출력 길이(224, 256, 384, 512 비트)를 갖는 새로운 표준(SHA-2)을 발표하였다. 그런데 기존에 알려진 280보다 훨씬 적은 269의 연산 횟수로 SHA-1를 공격할 수 있는 방안이 발표되었다[3]. 이에 NIST는 SHA-2가 위험해지는 경우를 대비하여 이들 대신할 수 있는 차세대 해쉬함수 SHA-3을 제정하기로 하였다[4]. NIST는 2007년 SHA-3에 대한 요구조건을 공개하여 전 세계 암호 연구 기관으로부터 해쉬 알고리듬을 접수받은 후, 5년 간 3 라운드에 걸친 평가 과정을 거쳐서 2012년 10월 이탈리아와 벨기에 연구팀이 공동으로 개발한 Keccak 알고리듬을 SHA-3(FIPS PUB 202)로 선정하였다. NIST는 공청회와 표준화 과정을 거쳐 영역 파라미터를 입력 메시지에 추가하는 최종 문서를 2015년 8월에 발표하였다[5]. SHA-3 표준안은 출력 해쉬값의 길이에 따라 SHA3-224, SHA3-256, SHA3-384, SHA3-512로 구성된 SHA-3 해쉬 알고리듬과 출력 길이를 확장할 수 있는 extendable-output function (XOF)인 SHAKE128, SHAKE256로 구성된다. SHA-3 알고리듬은 스폰지 함수(sponge function)를 사용하며 매개 변수에 따라 처리 성능과 암호 강도를 적절히 조절할 수 있고 가변 길이의 해쉬 출력을 생성할 수 있다. SHA-3 알고리듬은 반복 구조, 단일 클록에 여러 라운드 동작을 처리하는 루프 펼침 구조, 단일 라운드의 파이프라인 구조 등으로 구현이 이루어지고 있다[6-9]. NIST의 홈페이지에서는 비트 단위 길이의 테스트 벡터를 사용하는 예제를 제공하고 있는데[10], SHA-3에 대한 대부분의 하드웨어 및 소프트웨어 구현이 컴퓨터 구조의 바이트 처리에 적합하게 바이트 배수 길이의 입력 메시지만 처리하는 한계가 있다. 본 논문에서는 입력 메시지 길이 제한 문제를 해결하기 위해 임의의 비트 길이 입력 메시지를 처리할 수 있는 SHA-3 해쉬 프로세서를 설계하였다.

논문의 구성은 2장에서 SHA-3 표준 알고리듬을 기술하고, 3장에서는 알고리듬을 구현하는 프로세서의 하드웨어 설계를 기술하였으며, 4장에서는 설계 검증 및 성능 분석을 하였으며, 마지막으로 결론을 기술하였다.

Ⅱ. SHA-3 해쉬 알고리듬

2.1. 스폰지 함수

SHA-3 해쉬 알고리듬의 핵심인 그림 1의 스폰지 함수(Z=sponge[f, pad, r](N, d))는 4개의 주요 매개변수로 정의된다. 여기서 f()는 각 블록을 처리하는 내부 함수, r은 입력 블록의 길이, pad는 패딩 알고리듬, N은 입력 메시지, d는 출력의 길이를 나타낸다.

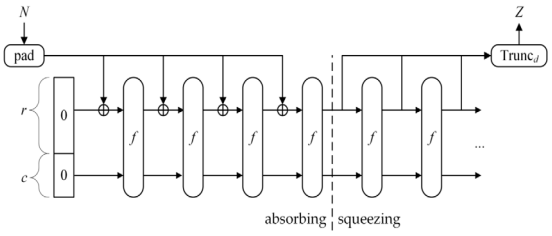

Fig. 1 Sponge function[5]

SHA-3에 사용하는 스폰지 함수는 s=r+c=1600 비트인 상태 변수 s에 의해 동작하는데, 초기 상태 변수는 0으로 초기화되며, 반복 동작 시 새로운 값으로 갱신된다. SHA-3의 경우 r 비트가 증가할수록 처리 성능은 증가하고 보안 수준은 감소한다. 가변 길이인 입력 메시지는 블록 크기 r의 정수배로 조정되는데, 마지막 블록의 경우 패딩 규칙에 따라 패딩이 되어, P_0, P_1, ..., P_k의 r-비트 길이의 블록 들을 생성한다. 각 P_i블록은 하위에 c개의 0을 추가하여, 이전 상태 값과 비트 단위의 XOR 연산을 한 후 f() 함수에 입력된다. f() 함수의 결과는 다음 반복 동작의 입력 상태가 된다. 이러한 반복 동작은 메시지의 모든 블록이 소진될 때까지 반복되는데, 이러한 동작을 흡수 단계(absorbing phase)라 한다. 반면 출력 짜내기 동작(squeezing phase)은 흡수 단계 출력의 상위 r-비트의 메시지를 종자로 사용하여 f()

함수의 반복 동작으로 원하는 길이의 출력을 생성한다. 이러한 동작은 출력 확장 함수인 SHAKE128과 SHAKE256에서 사용되며, 키 생성과 난수 생성기로 활용된다. SHA-3의 표준안의 경우 최대 해쉬 출력 길이가 d=512비트로 항상 r-비트 보다 작으므로 출력 짜내기 단계가 필요하지 않다.

2.2. SHA-3 알고리듬

표 1은 SHA-3 표준 알고리듬에서 해쉬 출력 길이 d와 r, c, s의 길이 관계를 나타낸다. SHAKE256은 SHA3-256과 유사하지만 출력에서 256-비트가 아닌 r-비트를 취해서 출력 짜내기 동작을 한다. SHA-3의 경우 c의 길이는 출력 길이의 2배가 된다.

Table. 1 Length of parameters in SHA-3 algorithm

SHA-3 algorithm	hash output length, d / output length (bit)	block length, r (bit)	capacity, c (bit)	state length,s (bit)
SHA3-224	224 / -	1152	448	1600
SHA3-256	256 / -	1088	512	1600
SHA3-384	384 / -	832	768	1600
SHA3-512	512 / -	576	1024	1600
SHAKE128	1344 / any	1344	256	1600
SHAKE256	1088 / any	1088	512	1600

SHA-3의 반복 함수인 f()는 입력으로 5×5 64-b 워드 행렬, $a[x][y][z](0 \le x,y < 5, 0 \le z < 64)$로 정의된 1600-비트의 상태 변수를 받아서 총 24 라운드 처리를 한다. 여기서 $a[x][y]$는 64-비트 lane을 나타낸다. 각 라운드는 식 (1)과 같이 5개의 세부 단계($\theta, \rho, \pi, \chi, \iota$)로 구성되며 +는 GF(2) 상의 덧셈인 XOR 연산을 나타낸다. SHA-3의 라운드 동작은 3차원 배열 구조에 대해 행과 열 간의 위치 조정, 비트 단위의 논리 연산 동작과 회전 이동 동작 등으로 구성된다. 마지막 ι 단계는 라운드별로 다른 라운드 상수 $RC[i_r]$을 사용하여 라운드 동작 간에 차이를 만든다. Keccak 알고리듬(Keccak[c](M))이 SHA-3 알고리듬으로 표준화되는 과정에 식(2)과 식(3)과 같이 SHA-3와 SHAKE 기능에서 메시지 M 뒤에 각각 2-비트(01_2)와 4-비트(1111_2)의 영역 파라미터가 추가되는 방식으로 약간 변형되었다.

$$
\begin{aligned}
&\theta : a[x][y][z] \leftarrow a[x][y][z] + \sum_{y'=0}^{4} a[x-1][y'][z] \\
&\qquad\qquad + \sum_{y'=0}^{4} a[x+1][y'][z-1] \qquad (1)\\
&\rho : a[x][y][z] \leftarrow a[x][y][z-(t+1)(t+2)/2] \\
&\quad with\ 0 \le t < 24, \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}, GF(5)^{2\times 2}, \\
&\quad \text{or } t = -1 \text{ if } x = y = 0 \\
&\pi : a[x][y] \leftarrow a[x'][y'], with \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} \\
&\chi : a[x] \leftarrow a[x] + (a[x+1]+1)\, a[x+2] \\
&\iota : a \leftarrow a + RC[i_r] \\
&\quad RC[i_r][x][y] = 0, \text{ if } x \neq 0 \text{ or } y \neq 0, \\
&\qquad\qquad \text{for } 0 \le i_r \le 23 \\
&\quad RC[i_r][x][y][2^j - 1] = rc[j + 7i_r], \\
&\qquad\qquad \text{for } 0 \le j \le 6 \\
&\quad rc[t] = (x^t
\end{aligned}
$$

$$
\begin{aligned}
SHA3-224(M) &= Keccak[448](M \parallel 01, 224) \\
SHA3-256(M) &= Keccak[512](M \parallel 01, 256) \\
SHA3-384(M) &= Keccak[768](M \parallel 01, 384) \\
SHA3-512(M) &= Keccak[1088](M \parallel 01, 512)
\end{aligned} \qquad (2)
$$

$$
\begin{aligned}
SHAKE128(M, d) &= Keccak[256](M \parallel 1111, d) \\
SHAKE256(M, d) &= Keccak[512](M \parallel 1111, d)
\end{aligned} \qquad (3)
$$

Ⅲ. 알고리듬의 하드웨어 설계

3.1. 설계 사양

본 논문의 해쉬 프로세서는 구현 시 다음과 같은 설계 사양을 적용하였다.

첫째, 해쉬 응용과 키, 난수 생성 등에 응용할 수 있도록 4가지 함수로 구성된 SHA-3과 SHAKE256 알고리듬을 하나의 하드웨어로 구현한다.

둘째, SHA-3의 핵심 내부 f() 함수를 구현하는 라운드 코어 구조는 반복 구조로 구현하지만, padder 블록, 라운드 코어블록, 출력 블록 간에는 블록 수준의 파이프라인 처리를 적용하여 성능을 높이도록 한다.

셋째, 메시지 길이가 바이트의 배수가 아닌 경우도 처리할 수 있도록 하여, SHA-3의 표준안에서 제시된 모든 테스트 벡터를 만족하도록 한다.

넷째, SHA-3의 최종 표준안에 도입된 영역 파라미터는 메시지의 일부로 처리하여, 호스트 단에서 메시지의 맨 뒤에 추가해서 입력하도록 한다.

3.2. 프로세서의 전체 구조와 입출력 인터페이스

해쉬 프로세서는 그림 2와 같이 크게 padder 블록, round core 블록, output 블록으로 구성되며, 내부적으로 datapath 블록과 control 블록으로 나뉜다. 각 블록의 연산 시간이 알고리듬 별로 다르므로, 블록간 파이프라인 처리를 할 수 있도록 블록 간에 Busy 신호와 Start 신호가 교환된다.

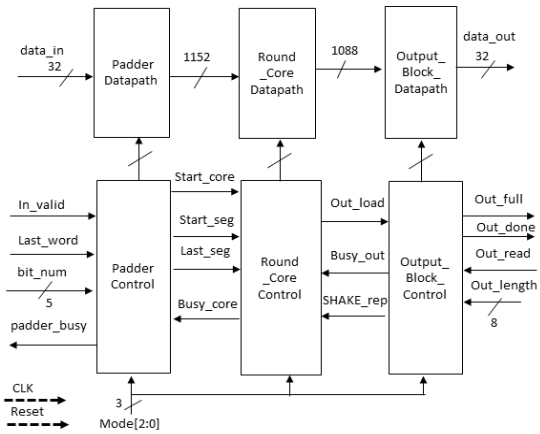

Fig. 2 Architecture of hash processor

3.3. Padder 블록 구조

호스트 프로세서는 Padder 블록이 데이터를 받을 준비가 된 경우 data-in 데이터를 32-비트 단위로 순차적으로 전송한다. 단, 해쉬 알고리듬이 가변 길이의 입력을 제공하므로 입력 메시지의 마지막을 알리는 last_word =1 신호를 data-in과 함께 전송한다. 마지막 워드가 32-비트가 아닌 경우 bit_num[4:0]로 타당한 비트 수를 알려 준다. last_word가 0인 경우는 마지막 워드가 아닌 경우로 항상 32-비트 전송을 의미한다. 단, Padding 제어 프로토콜을 단순화하기 위해 마지막 워드가 32-비트가 모두 타당한 경우는 last_word=1을 발생시키지 않고, 대신 bit_num[4:0]=0, last_word=1인 추가적인 데이터 입력 사이클을 호스트에서 발생한다. 입력 메시지 길이가 1개의 블록 길이를 초과하는 경우, 1개의 블록이 채워지면 Padder 블록은 Padder_busy 신호를 1로 만들어 호스트 프로세서가 추가의 데이터를 전송하는 것을 막도록 한다. 채워진 블록은 라운드 코어에 전달한 후 다시 Padder_busy 신호를 0으로 만든다. Padder 블록은 호스트 프로세서에서 블록을 채우는 동작과 함께 마지막 블록의 경우 패딩 처리를 지원한다. SHA-3의 패딩 규칙 {10*1}에서 처음 1과 마지막 1은 항상 존재해야 하는 필수적인 비트이며, 가운데 0*은 0개 이상의 0의 반복 비트를 나타낸다. 따라서 수신된 메시지 블록의 길이가 r-4, r-2비트인 경우 패딩 되는 값은 그림 3(a)에 따라 각각 $\{1001_2\}$, $\{11_2\}$가 된다. 그림 3(b)과 같이 메시지의 마지막 블록의 길이가 r-1 비트인 경우 패딩에 필요한 최소한의 공간인 2-비트가 남아 있지 않아서, 1개의 1을 채워 블록을 완성한 후, (r-1)개의 0과 1개의 1로 구성된 새로운 블록을 추가로 생성하여 라운드 코어에 전송해야 한다. 이러한 추가의 패딩 블록 생성은 마지막 입력 블록이 r-비트인 경우에도 필요하다. 패딩 동작은 해쉬 프로세서에서 가장 복잡한 동작으로 내부 제어회로가 입력 받은 메시지의 길이를 유지하고 있다가 요구사항에 맞는 제어신호를 생성한다. Padder 블록에서 Round_core 블록으로 전달되는 블록의 r-비트가 가변 길이 입력 메시지의 첫 번째 블록, 중간 블록, 마지막 블록인지 여부에 따라 round-core의 입력 처리가 다르므로 첫 번째와 마지막 블록을 나타내는 start_seg과 last_seg 신호가 사용된다. 2개의 신호가 모두 0인 경우는 중간 단계 메시지 블록을 나타낸다.

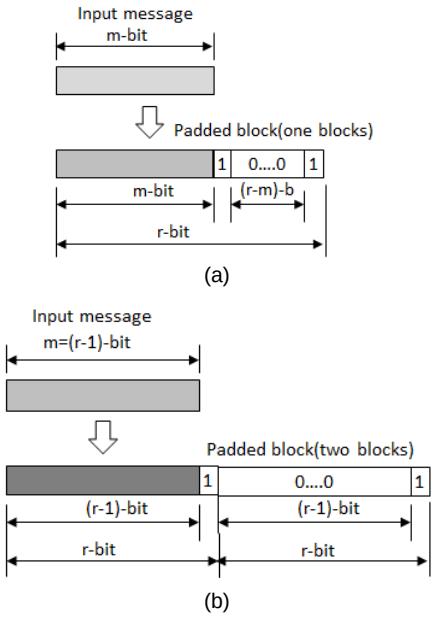

Fig. 3 Padding rule
(a) m ≤ (r-2)-bit (b) m=(r-1)-bit

SHA-3의 패당 규칙을 지원하는 padder 블록의 구조는 그림 4와 같다. SHA-3의 4가지 모드와 SHAKE-256은 다른 블록 길이(r)를 가지고 있으므로, 모드별로 다른 초기 값을 갖는 pad-counter를 사용하며, 가장 긴 블록 길이를 갖는 SHA3-224의 r=1152-b에 맞추어 36개의 32-비트로 구성된 쉬프트 레지스터 구조를 구현한다. LW_padder회로는 마지막 32-비트 워드인 경우 bit_num[4:0]과 last_q를 사용하여 그림 3의 32-비트 패딩 결과를 생성하며, MUX는 last_q=0,1에 따라 $\{0...00_2\}$ 혹은 $\{0...01_2\}$의 패딩 결과를 생성한다. 블록 출력에 존재하는 MUX는 SHA-3 모드별로 PB_BUF에 담긴 블록을 선택하고 좌측 정렬한 후 적절한 수(c-bit)의 0을 추가하여 라운드 코어에 제공한다.

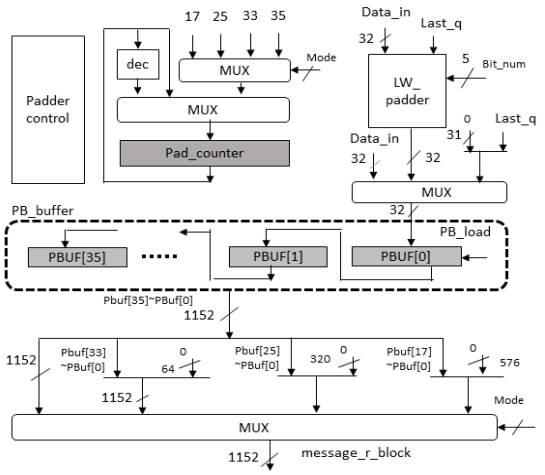

Fig. 4 Padder block

3.4. Round-Core 블록 구조

그림 5의 라운드 코어 블록은 식(1)의 5개의 세부 단계($\theta, \rho, \pi, \chi, \iota$)를 사용하여 24번의 라운드를 구현하는 기능을 하며, 매 클록마다 1개의 라운드를 처리하는 구조를 갖는다. 식 (1)의 5개의 세부 단계 동작 중 theta(θ) 단계는 인접 열사이에 11개의 비트의 조합 함수로 구현하였다. Rho(ρ) 단계는 64-비트로 구성된 각 lane을 행과 열의 위치 (x,y)에 따라 결정되는 오프셋 길이만큼 회전 이동하는 동작으로 구현하였다. pi(π) 단계는 행과 열의 좌표 (x,y)에 따라 64-비트 lane의 위치를 조정하는 동작으로 고정 배선으로 구현된다. chi(χ) 단계는 인접한 행의 다른 비트 간에 비선형 동작을 구현한다.

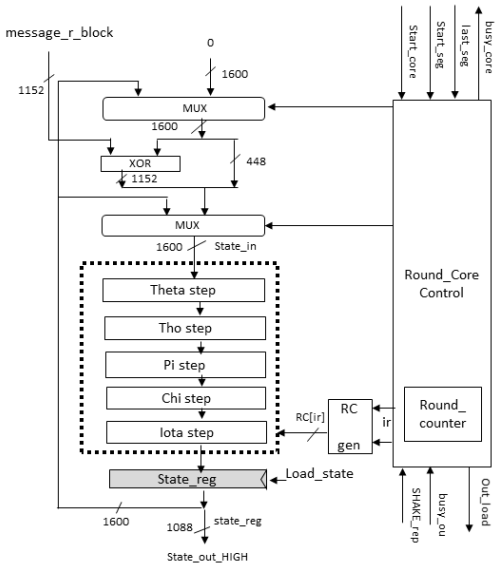

Fig. 5 Round-Core block

마지막 iota(ι) 단계는 (x,y)=(0,0)인 lane에 대해 라운드 상수(RC)와 XOR 동작으로 구현된다. 라운드 상수 생성 방안은 룩업 테이블로 구현하는 방안과 하드웨어로 구현하는 방안으로 나눌 수 있는데, 본 논문에서는 속도향상을 위해 그림 6(b) 구조와 같은 on-the-fly 방식의 라운드 상수 발생 회로를 설계하였다. update_logic은 그림 6(a)의 bit-serial GF divider[11]을 병렬 연산 구조로 수종해서 매 클록마다 t=0, 7, 14, ... , 161로 7만큼 떨어진 간격으로 $rc[7i_r]$, 즉 $x^t \bmod (x^8+x^6+x^5+x^4+1)$을 계산하는 회로이다. 단, 첫 번째 라운드에 대응하는 $x^0 \bmod (x^8+x^6+x^5+x^4+1)$ 값은 별도로 계산하지 않고 정해진 값을 MUX로 바로 제공한다. LFSR_R 값을 입력으로 사용하는 GF_MUL_reduction 회로는 $rc[7i_r]$에 대해 j=0~6에 대응하는 $rc[j+7i_r]$를 생성한다. LFSR_R 값에 $x^i\,(i=0,...6)$을 곱한 후 기약 다항식으로 나눈 후, 추가적으로 mod x 동작을 하므로 유한체 곱셈을 하지 않고 상수로 결정되므로 7개의 결과는 각각 2개 이하의 XOR 연산으로 구현되어 룩업 테이블을 사용하는 방식에 비해 속도 측면과 면적 측면에서 훨씬 효율적이다.

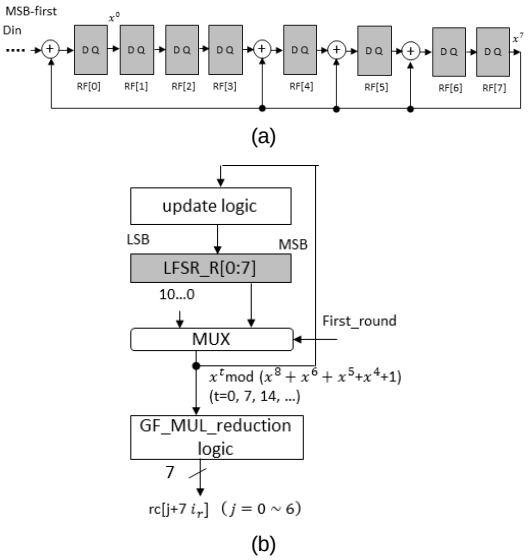

Fig. 6 On-the-fly Round constant generator (a) Bit-serial Galois field divider (b) Round constant generator

3.5. Output 블록 구조

Output 블록은 Padder 블록의 반대 동작을 한다. 라운드 코어에서 계산한 SHA-3와 SHAKE-256의 결과를 호스트로 전송하는 역할을 한다. 블록 간에 독립성을 제공하고 블록 파이프라인을 구현하기 위해 라운드 코어의 state 레지스터의 상위 1088-비트를 쉬프터 레지스터 구조의 출력 레지스터(out_R[0] ~ out_R[33])에 병렬로 저장한 후 Mode[2:0]에 따라 32-비트 단위로 좌측 이동하면서 호스트에 결과를 전달한다. 단, SHAKE-256의 경우 원하는 길이의 출력을 생성할 때까지 그림 1의 출력 짜내기 단계를 수행하도록 라운드 코어에 SHAKE_rep 신호로 지시한다. SHAKE-256 모드 동작 시 10만 비트 이상의 난수를 지원하도록 출력의 블록 수를 지시하는 8 비트의 SHAKE_out_length를 호스트에서 제공된다.

Ⅳ. 검증 및 성능 분석

그림 7은 설계한 해쉬 프로세서의 SHA-3의 4가지 동작 모드에 대한 블록 수준 파이프라인 동작을 나타낸다. 단, 메시지가 3개의 블록으로 표현되는 경우이다.

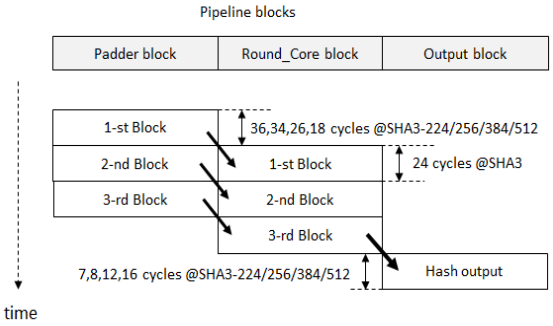

Fig. 7 Block-level pipeline for SHA3 algorithm (message length = three blocks)

두 번째 입력 블록이 Padder 블록에 입력되는 시점에 이전에 입력된 블록이 라운드 코어 블록으로 전달되어 라운드 동작을 한다. Padder 블록과 라운드 코어 블록간의 블록 수준 파이프라인 동작은 모든 메시지 블록이 Padder 블록에 채워질 때까지 반복된다. 마지막 블록이 라운드 코어에서 처리가 완료된 후 출력 블록으로 전달되어 해쉬 결과를 32-비트 단위로 호스트로 전달한다. SHA-3의 4가지 동작 모드별로 블록 길이(r-비트)와 출력 길이(d-비트)가 다르므로 Padder 블록에서 32-비트 단위로 블록을 입력하는 경우 필요한 사이클 수가 모드별로 다르다. SHA3-224의 경우 Padder 블록에 최대 36 사이클이 필요하다. 반면 라운드 코어의 경우 24 사이클로 고정이며, Output 블록에 필요한 사이클 수는 SHA3-512의 경우가 최대 16 사이클이 필요하다. SHA-3의 경우 블록 파이프라인의 성능을 결정하는 블록은 Padder 블록이 된다. 전체 메시지가 1개의 블록 이하인 경우 입력 메시지 처리 성능은 블록 파이프라인 동작으로 병목 부분 블록의 사이클 수에 의해 결정되며, 처리율이 식(4)과 같이 표현될 수 있다.

$$throughput = \frac{input\,block\,length \times frequency}{Cycles} \quad (4)$$

그림 8은 입력 메시지가 2개의 블록으로 구성되고, 출력 길이가 3개의 블록인 경우 SHAKE-256에 대한 블록 수준 파이프라인 동작을 나타낸다. SHAKE-256의 경우 출력 비트 생성에 대한 처리율은 식(5)과 같다.

$$throughput = \frac{output\,block\,length \times frequency}{cycles} \quad (5)$$

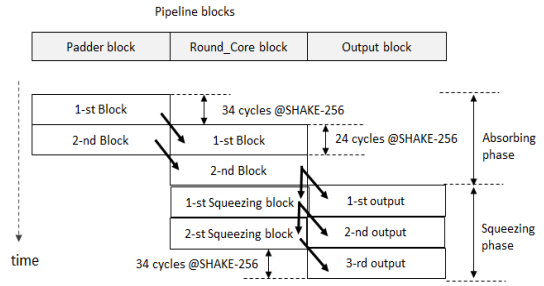

Fig. 8 Block-level pipeline for SHAKE-256 algorithm (message length = two blocks)

설계한 해쉬 프로세서는 Verilog-2001 HDL로 모델링하였으며, 동작 검증에 NIST에서 제공한 테스트 벡터와 Modelsim 시뮬레이터를 사용하였다[10]. 그림 9는 입력 길이가 1630-비트인 경우 SHA3-256에 대한 Modelsim 검증 파형을 나타낸다.

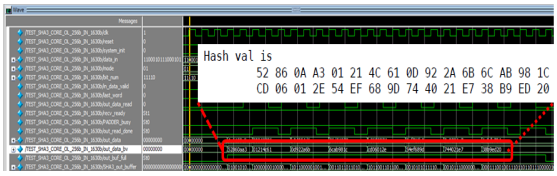

Fig. 9 Modelsim simulation waveform for SHA3-256 with 1630-bit message

표 2는 해쉬 프로세서의 특성을 요약한 결과이다. 해쉬 프로세서를 Xilinx Virtex5 칩을 사용하여 구현한 결과 최대 동작 주파수가 220 MHz로 평가되었다. 식 (4)과 식(5)을 적용한 결과 SHA3-512와 SHAKE-512는 각각 5.28 Gbps와 7.04 Gbps의 처리율을 갖는다. 표 3은 문헌에서 발표한 SHA3 프로세서와 비교한 결과이다. 본 논문의 해쉬 프로세서는 5개의 기능 지원으로 하드웨어의 복잡도 증가로 최장 경로의 지연 시간이 늘어나 한 개의 라운드 코어를 기준으로 한 처리율은 기존 문헌에 비해 떨어지지만, 블록 수준 파이프라인 동작으로 여러 개의 연속된 메시지에 대한 해쉬 처리의 경우 성능이 높아, 시스템 측면에서는 다른 프로세서에 비해 우수하다고 판단된다. 향후 시뮬레이션 단계의 성능 평가 다음에 FPGA 환경에서 USB 인터페이스로 구현을 검증할 예정이다.

Table. 2 Summary of Hash processor

Hash algorithms supported	SHA3-224, SHA3-256, SHA3-384, SHA3-512	
XOF algorithm	SHAKE-256	
operation scheme for round core	iterative operations (1 round per clock)	
system-level operation scheme	block pipeline scheme	
message format supported	bit-level	
RC generation scheme	on-the-fly hardware	
Throughput @220Mhz and 1-block message (SHA3)	SHA3-512	5.28 Gbps
	SHA3-384	7.04 Gbps
	SHA3-256	7.04 Gbps
	SHA3-224	7.04 Gpbs
Throughput @220 Mhz (SHAKE-256)	7.04 Gbps	
Max. Output length (SHAKE-256)	278,528-bit	
FPGA	Xilinx Virtex5 xc5vlx50	
Max. clock frequency	220 MHz	

Table. 3 Comparison of SHA-3 hash processors

	[7]	[9]	this paper
Algorithms supported	SHA3-224/ 256/384/512	SHA3-512	SHA3-224/ 256/384/512, SHAKE-256
Message format	byte-level	-	bit-level
Round core structure	iterative round core (128-bit I/O)	iterative round core (2-stage)	iterative round-core
Padding scheme	hardware	no	hardware
# of processing cycles for a block	25	48	24
Throughput [Gbps]	5.70 @SHA3-512	25.4 @SHA3-512	5.28 @SHA3-512
Technology	FPGA Xilinx Virtex5 xc5vsx240t	TSMC 65-nm standard cell	FPGA Xilinx Virtex5 xc5vlx50
Hardware Resources	2573 slices	49.1 K [GE]	3887 slices
Max. Freq.	285 MHz	1.192 GHz	220 MHz

Ⅴ. 결 론

본 논문에서는 미국 표준 해쉬 알고리듬으로 채택된 SHA-3 알고리듬과 출력 확장 함수인 SHAKE-256을 처리하는 해쉬 프로세서를 하드웨어로 설계한 후 검증하였다. 해쉬 프로세서는 padder 블록, round core 블록, output 블록으로 구성되며, 블록 파이프라인 처리를 통해 성능을 최적화한다. 라운드 코어는 매 클록당 1개의 라운드를 처리하는 반복 구조를 갖고 있으며, Padder 블록은 바이트 배수가 아닌 임의의 비트 길이 메시지 길이를 지원하며 SHA-3의 모든 표준 벡터를 만족한다. 또한 라운드 동작의 세부 iota(ι) 단계에서 라운드 상수를 테이블 룩업 방식이 아닌 on-the-fly 하드웨어 방식으로 구현함에 의해서 면적과 속도 향상을 얻을 수 있었다. 설계된 해쉬 프로세서는 Virtex5 FPGA 환경에서 최대 220 MHz의 클록 주파수를 가지며, SHA3-512의 경우 최대 5.28Gbps의 처리율을 가지며, SHAKE-256의 경우 7.04 Gbps의 난수 생성율을 제공한다. 다만 향후 암호 분야에 폭넓게 응용하기 위해 SHAKE-256 지원 확장과 회로의 신뢰성 향상 및 부채널 공격에 대한 대처 연구가 필요하다. 본 논문의 해쉬 프로세서가 지원하지 않은 SHAKE128의 경우 Padder 블록과 Output 블록의 확장으로 쉽게 구현이 가능하다.

설계한 해쉬 프로세서는 임의의 비트 길이 메시지 지원, 블록 파이프라인에 따른 높은 시스템 레벨 처리율, SHA-3와 SHAKE-256 알고리듬 지원으로 무결성, 디지털서명, 키 생성, 난수, HMAC 등의 다양한 암호 분야에 효율적으로 응용이 가능하다.

ACKNOWLEDGMENTS

The IDEC CAD tools were partially used to verify operation of circuit and algorithm.

REFERENCES

[1] W. Stalling, *Cryptography and Network Security-Principle and Practices*, 5th ed., Essex, England: Pearson, 2013.

[2] I. S. Janik, “High Level Synthesis and Evaluation of the Secure Hash Standard for FPGAs,” Ms. D. dissertation, University of Windsor, Windsor, Ontario, Canada, 2015.

[3] X. Wang, Y. Yin, and H. Yu, “Finding collisions in the full SHA-1,” in *Advances in Cryptology(Crypto-2005) Lecture Notes in Computer Science, vol. 3621*, Berlin, Heidelberg: Springer-Verlag, 2005, pp. 17-36.

[4] W. Stalling, “Inside SHA-3,” *IEEE Potentials*, vol. 22, no. 6, pp. 26-31, Nov.-Dec. 2013.

[5] FIPS PUB 202, *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*, NIST, Gaithersburg, MD, Aug. 2015.

[6] B. Baldwin, A. Byrne, and L. Lu, “A Hardware Wrapper for the SHA-3 Hash Algorithms,” in *IET Irish Signals and Systems Conference*, Cork, Ireland, pp.1-6, 2010.

[7] G. Provelengios et al, “FPGA-Based Design Approaches of Keccak Hash Function,” in *15th Euromicro Conference on Digital System Design*, Izmir, Turkey, pp.648-653, 2012.

[8] A. Arshad et al “Compact Implementation of SHA3-512 on FPGA,” in *2014 Conference on Information Assurance and Cyber Security(CIACS)*, Rawalpindi, Pakistan, pp.29-33, 2014.

[9] S. Bayat-Sarma et al, “Efficient and Concurrent Reliable Realization of the Secure Cryptographic SHA-3 Algorithm,” *IEEE Transactions on CAD of Integrated Circuit and Systems*, vol. 33, no. 7, pp.1105-1109, July 2014.

[10] NIST, “SHA-3 example: Test vector,” [Internet]. Available: http://csrc.nist.gov/groups/ST/hash/sha-3/fips202_standard_2015.html.

[11] K. K. Saluja, “Linear Feedback Shift Registers Theory and Applications,” [Internet]. Available: http://homepages.cae.wisc.edu/~ece553/handouts/LFSR-notes.PDF.

최병윤(Byeong-yoon Choi)

1985년 2월 : 연세대학교 전자공학과 졸업
1992년 8월 : 연세대학교 공학 박사
1993년 3월~현재 : 동의대학교 교수
※관심분야 : RISC 프로세서 설계, 정보통신 알고리즘의 SoC 설계, 임베디드 시스템 응용 등