ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 18-04-01
j.inst.Korean.electr.electron.eng.Vol.22,No.4,886~895,December 2018
http://dx.doi.org/10.7471/ikeee.2018.22.4.886

SHA-3 해시 함수의 최적화된 하드웨어 구현

An Optimized Hardware Implementation of SHA-3 Hash Functions

김 동 성*, 신 경 욱*★

Dong-Seong Kim*, Kyung-Wook Shin*★

Abstract

This paper describes a hardware design of the Secure Hash Algorithm-3 (SHA-3) hash functions that are the latest version of the SHA family of standards released by NIST, and an implementation of ARM Cortex-M0 interface for security SoC applications. To achieve an optimized design, the tradeoff between hardware complexity and performance was analyzed for five hardware architectures, and the datapath of round block was determined to be 1600-bit on the basis of the analysis results. In addition, the padder with a 64-bit interface to round block was implemented in hardware. A SoC prototype that integrates the SHA-3 hash processor, Cortex-M0 and AHB interface was implemented in Cyclone-V FPGA device, and the hardware/software co-verification was carried out. The SHA-3 hash processor uses 1,672 slices of Virtex-5 FPGA and has an estimated maximum clock frequency of 289 Mhz, achieving a throughput of 5.04 Gbps.

요 약

본 논문에서는 NIST에서 발표한 Secure Hash Algorithm(SHA) 표준의 최신 버전인 SHA-3 해시 함수의 하드웨어 구현과 함께 보안 SoC 응용을 위한 ARM Cortex-M0 인터페이스 구현에 대해 기술한다. 최적화된 설계를 위해 5 가지 하드웨어 구조에 대해 하드웨어 복잡도와 성능의 교환조건을 분석하였으며, 분석 결과를 토대로 라운드 블록의 데이터패스를 1600-비트로 결정하였다. 또한, 라운드 블록과 64-비트 인터페이스를 갖는 패더를 하드웨어로 구현하였다. SHA-3 해시 프로세서, Cortex-M0 그리고 AHB 인터페이스를 집적하는 SoC 프로토타입을 Cyclone-V FPGA 디바이스에 구현하여 하드웨어/소프트웨어 통합 검증을 수행하였다. SHA-3 프로세서는 Virtex-5 FPGA에서 1,672 슬라이스를 사용하였으며, 최대 289 Mhz의 클록 주파수로 동작하여 5.04 Gbps의 처리율을 갖는 것으로 예측되었다.

Key words : Secure Hash Algorithm-3, SHA-3, Hash function, KECCAK, Integrity, Digital Signature

---

* School of Electronic Engineering, Kumoh National Institute of Technology
★ Corresponding author
E-mail : kwshin@kumoh.ac.kr, Tel : +82-54-478-7427
※ Acknowledgment
- This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (No. 2017R1D1A3B03031677)
- This work was supported by KIAT(Korea Institute for Advancement of Technology) grant funded by the Korea Government (MOTIE : Ministry of Trade, Industry and Energy) (No. N0001883, HRD Program for Intelligent semiconductor Industry)
- Authors are thankful to IDEC for supporting EDA softwares.

Manuscript received Dec. 19, 2018; revised Dec. 22, 2018; accepted Dec. 22, 2018

Ⅰ. 서론

정보보안 시스템은 대칭키 암호, 공개키 암호, 해시 함수, 난수 발생 등 다양한 보안 관련 알고리듬을 기반으로 구현된다. 대칭키 암호는 정보의 기밀성을 보장하기 위해 데이터를 암호화하는 목적으로 사용되며 AES(Advanced Encryption Standard) [1], ARIA(Academy, Research Institute, Agency) [2] 등의 블록암호 알고리듬이 널리 사용되고 있다. 공개키 암호 방식으로는 RSA(Rivest, Shamir, Adleman) [3], ECC(Elliptic Curve Cryptography) [4] 등이 있으며, 키 교환이나 전자서명과 같은 보안 서비스를 제공하기 위해 사용된다. 해시(hash) 함수는 가변 길이의 데이터를 고정된 길이의 다이제스트(digest)로 변환하는 암호학적 함수를 의미하며, HAS-160 [5], SHA-1(Secure Hash Algorithm-1), SHA-2 [6], Whirlpool [7] 등이 사용되고 있다. 그러나 보안 공격이 날로 정교해짐에 따라 기존에 사용되고 있는 해시 함수들의 수학적 결함이 발견되면서 새로운 해시 함수의 필요성이 제기되었다. 이에 미국 국립표준기술연구소(NIST)는 새로운 해시 함수 후보 알고리듬을 공개적으로 모집하였고, 2012년에 KECCAK 알고리듬이 새로운 해시 함수인 SHA-3로 선정되어 2015년 8월에 표준안이 공개되었다. SHA-3 KECCAK 알고리듬은 현재 SHA-2가 사용되는 응용 프로그램에 직접적으로 대체가 가능하며, 보다 높은 수준의 보안을 제공하기 때문에 해시 함수 알고리듬의 공식 권장 표준이 되었다. SHA-3 KECCAK은 하드웨어 구현 시 고속 연산이 가능하므로 다양한 분야에 적용이 가능할 것으로 예상된다.

정보통신 기술의 발달로 인해 무선 네트워크 기반의 장치와 시스템이 급속히 증가하는 추세에 따라 다양한 형태의 보안 위협도 함께 증가하고 있으며, 이를 방지하기 위한 보안서비스의 필요성이 날로 증가하고 있다. 보안 서비스는 하드웨어에 소프트웨어가 결합되는 형태를 기본으로 하며, 강력한 보안기능을 내장한 보안 SoC는 보안 안전성과 함께 저전력 동작이 가능하여 IoT 장치용 보안 시스템에 핵심 요소가 된다. SoC는 마이크로프로세서를 기반으로 구성되며, AVR(Advanced Virtual RISC), ARM(Advanced RISC Machines) CPU가 널리 사용된다. ARM 아키텍처는 임베디드 기기에 가장 폭넓게 사용되고 있으며, 저전력으로 동작하여 모바일 장치에 적합한 것으로 평가된다. 최신 ARM 아키텍처인 Cortex는 Cortex-A, Cortex-R, Cortex-M으로 나뉜다. Cortex-M은 저전력으로 동작하며, 적은 면적을 차지하므로 IoT 모바일 장치에 더욱 적합하다. 특히 Cortex-M0는 Cortex-M 프로세서 제품군 중 가장 적은 면적을 차지하며 저렴하고 초저전력으로 동작한다는 특징이 있다.

본 논문에서는 차세대 해시 함수 표준으로 채택된 SHA-3 해시 알고리듬을 최적 하드웨어 IP(intellectual property)로 구현하고, Cortex-M0 슬레이브 인터페이스와 FPGA 구현을 통해 하드웨어 동작을 검증하였다. II장에서는 SHA-3 해시 알고리듬에 대해 간략히 기술하고, III장에서는 하드웨어 설계를 위한 최적 조건 분석 결과에 대해 설명한다. IV장에서는 도출된 최적 설계조건을 적용한 SHA-3 해시 프로세서의 하드웨어 구현과 Cortex-M0 인터페이스에 대해 기술하고, V장에서는 HW/SW 통합 검증 및 성능 분석에 대해 설명하며, VI장에서 결론을 맺는다.

Ⅱ. SHA-3 해시 알고리듬 [8]

1. SHA-3 해시 알고리듬

NIST에서 채택한 SHA-3 표준은 해시 함수와 출력 확장 함수(extendable output function; XOF)로 구성된다. 해시 함수는 다이제스트 길이에 따라 SHA3-224, SHA3-256, SHA3-384, SHA3-512로 구분되며, XOF는 SHAKE128, SHAKE256로 구분된다. XOF는 출력되는 해시 값을 임의의 길이로 확장할 수 있다. 해시 함수의 경우 함수 이름 뒤의 숫자는 출력되는 해시 값의 길이를 나타내며, XOF의 숫자 128, 256은 보안강도를 나타낸다. 각각의 함수들은 KECCAK의 기초가 되는 순열을 기반으로 구성되며, 이 순열은 $KECCAK-p$ 계열의 수학적 순열로 규정된다.

$KECCAK-p$ 순열은 너비(width) b와 라운드를 나타내는 n_r로 정의하고 $KECCAK-p[b, n_r]$로 표기한다. b는 {25, 50, 100, 200, 400, 800, 1600}의 집합 중 하나의 원소로 결정되며, $KECCAK-p$ 순열 변환이 수행되는 데이터 단위의 크기를 나타낸다. b에 의해 결정되는 변수 w, l은 SHA-3 KECCAK 알고리듬에서 $KECCAK-p$ 순열을 규정하기 위

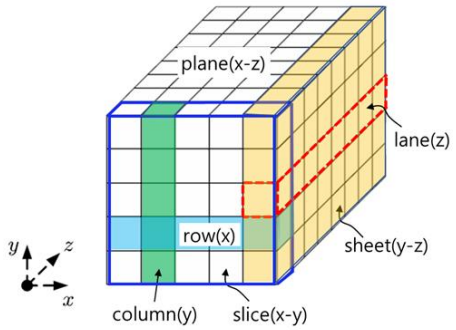


Fig. 1. 3-D array representation of state.
그림 1. 3차원 배열로 표현된 state

한 중요한 파라미터가 되며, $w=b/25$와 $l=\log_2 w$로 정의된다. n_r은 라운드 연산의 반복횟수를 의미한다. $KECCAK-p$ 순열은 5개의 변환 단계로 구성되며, 각 단계를 스텝(step)이라고 한다. 스텝을 통한 순열 변환을 스텝 매핑(step mapping)이라고 하며, state라는 b-비트 크기의 $5\times5\times w$ 3차원 배열을 기반으로 하는 블록단위로 진행된다. 그림 1은 $KECCAK-p$ 순열 변환에 사용되는 3차원 배열인 state의 구성을 보인 것이다. 5단계의 스텝 매핑을 통한 변환 과정을 라운드 연산으로 정의하며, 라운드 연산을 n_r회 반복 연산함으로써 하나의 블록에 대한 변환이 완료된다.

$$\begin{aligned}&Theta(\theta)\ step:\\&A'[x,y,z]=\\&A[x,y,z]\oplus\sum_{y'=0}^{4}A[(x-1)\bmod 5,y',z]\\&\oplus\sum_{y'=0}^{4}A[(x+1)\bmod 5,y',(z-1)\bmod w]\end{aligned}\quad(1)$$

$$\begin{aligned}&Rho(\rho)\ step:\\&A'[x,y,z]=A[x,y,(z+offset(x,y))\bmod w]\end{aligned}\quad(2)$$

$$\begin{aligned}&Pi(\pi)\ step:\\&A'[y,(2x+3y)\bmod 5]=A[x,y]\end{aligned}\quad(3)$$

$$\begin{aligned}&Chi(\chi)\ step:\\&A'[x,y,z]=\\&A[x,y,z]\oplus((A[(x+1)\bmod 5,y,z]\oplus 1)\\&\bullet\ A[(x+2)\bmod 5,y,z])\end{aligned}\quad(4)$$

$$\begin{aligned}&Iota(\iota)\ step:\\&A'[0,0]=A[0,0]\oplus RC\end{aligned}\quad(5)$$

$KECCAK-p$ 순열을 구성하는 스텝 매핑은 θ(theta), ρ(rho), π(pi), χ(chi), ι(iota)의 5단계로 이루어지며, 각 스텝은 식 (1)~식 (5)와 같이 정의된다. θ 스텝에서는 state의 각 비트와 인접한 두 column의 패리티와 XOR 연산을 수행한다. 기준 비트의 좌표를 $[x,y,z]=[x_0,y_0,z_0]$라고 할 때, 두 개의 column은 $[x,z]=[(x_0-1)\bmod 5,z_0]$, $[x,z]=[(x_0+1)\bmod 5,(z_0-1)\bmod w]$로 표현된다. ρ 스텝에서는 lane 단위로 순환 시프트 연산이 수행된다. π 스텝은 ρ 스텝과 마찬가지로 lane 단위로 연산이 수행되며, 해당 lane의 위치를 x, y 좌표에 따라 재배치 한다. χ 스텝은 동일한 row에 속한 데이터들의 연산이다. 다른 두 개의 비트를 입력으로 하는 비선형 함수의 결과 값과 XOR 연산이 수행된다. ι 스텝은 $x=0$, $y=0$에 속하는 lane과 라운드 상수의 XOR 연산이 수행된다.

2. 스폰지 함수

SHA-3 KECCAK 알고리듬은 그림 2와 같이 스폰지 (sponge) 함수 $SPONGE[f,pad,r](N,d)$를 기반으로 한다. 함수 f는 $KECCAK-f[b]$를 의미하며, $KECCAK-p[b,n_r]$ 순열에서 $n_r=12+2l$인 경우이다. 따라서 KECCAK 알고리듬에서 n_r의 값은 b의 개수 만큼인 7가지 값으로 정해진다. pad는 패딩 (padding) 알고리듬을 의미하며, N은 입력 메시지, 그리고 d는 출력되는 해시 값의 길이를 의미한다. r은 state에 포함되는 입력 메시지의 길이를 나타내며 입력 메시지는 r-비트 단위로 블록 연산이 수행된다. c는 *capacity*를 의미하는데 c의 크기는 state의 크기인 b-비트에서 입력 메시지가 차지하는 r-비트를 제외한 나머지 데이터의 길이를 나타낸다. 따라서 변수 b, r, c는 $b=r+c$의 관계를 만족하며, r이 작을수록 동일한 길이의 메시지 처리에 시간이 많이 소요되지만 보안 강도는 높아진다.

스폰지 함수는 임의의 길이의 메시지 데이터가 함수 f를 통해 고정된 길이의 데이터로 변환되는 흡수 (absorbing) 단계와 임의의 길이로 출력 값을 생성하는 짜내기 (squeezing) 단계로 나누어진다. 흡수 단계에서 첫 번째 블록의 경우 r과 c를 모두 0으로 초기화한 상태에서 연산이 시작된다. 다음

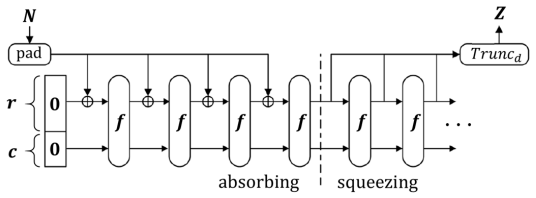


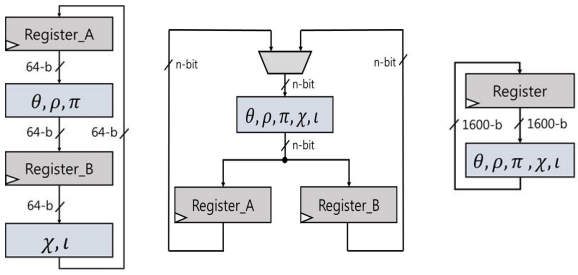
Fig. 2. Sponge construction of SHA-3.
그림 2. SHA-3의 스폰지 구조

블록 연산부터는 이전 블록 연산의 결과와 입력 메시지를 XOR한 결과가 함수 f의 입력으로 사용된다. 짜내기 단계는 흡수 단계에서 생성된 state의 상위 r-비트를 입력으로 받아 원하는 길이의 해시 값을 생성한다. 이러한 짜내기는 XOF에서만 수행되고, SHA-3 KECCAK 표준에 제시된 해시 알고리듬의 경우에는 출력 해시 값의 길이가 r-비트보다 작기 때문에 흡수 과정이 진행된 결과의 상위 비트를 해시 값으로 사용하여 짜내기 과정이 필요하지 않다.

III. SHA-3 해시 프로세서의 설계조건 분석

본 논문에서는 SHA-3 KECCAK 알고리듬을 저면적 하드웨어 구현에 적합한 하드웨어 구조를 탐색하기 위해 여러 가지 구조를 구상하고, 이를 비교, 분석하였다. 하드웨어 구조의 비교를 위해 SHA3-512 해시 함수를 선택했고, state의 크기는 b=1600로 결정하였다.

$KECCAK-p$ 순열 변환을 구성하는 5 가지의 스텝 매핑을 처리할 때, 각 스텝에서 연산되는 데이터 단위를 고려하여 라운드 블록의 데이터패스 크기를 결정하였다. 두 번째, 세 번째 스텝인 ρ, π의 경우, lane 단위로 연산되기 때문에 64-비트 크기로 데이터패스를 구현하면 χ 스텝을 제외한 모든 연산에 적합할 것으로 판단하였다. χ 스텝의 경우 row 단위의 연산 회로 5개를 병렬로 배치하여 plane 단위로 연산이 가능하다. θ 스텝은 연산되는 비트를 기준으로 인접한 column의 패리티 계산이 필요하다. 따라서 state에서 연산이 수행되는 sheet의 x-좌표를 x_0라고 하면, 해당 sheet에 속한 데이터를 계산하기 위해서 x-좌표가 $(x_0-1) \bmod 5$, $(x_0+1) \bmod 5$에 해당하는 sheet의 패리티를 계산해야 한다. 패리티 계산만을 위해서는 640-비트의 연산 처리가 필요하고, 이 sheet에 속한 데이터를 같은 클록 사이클에 매핑하기 위해서는 960-비트의 연산 처리량을 필요로 한다. 연산 처리량이 320-비트만큼 늘어날 때마다 한 개의 sheet에 포함된 데이터의 패리티를 더 계산할 수 있으므로 데이터패스를 320-비트씩 늘릴 수 있다. 따라서 데이터패스를 320-비트의 배수로 선정하는 것도 합리적인 구상이라고 판단했다. 따라서 state에 속한 모든 데이터를 한 번에 연산할 수 있는 1600-비트 데이터패스를 포함하여 64-비트, 320-비트, 640-비트, 960-비트의 5 가지 데이터패스 구조를 대상으로 비교, 분석하였다. [9]



(a) 64-bit (b) 320-/640-/960-bit (c) 1600-bit

Fig. 3. Architectures of SHA-3 hash processor depending on datapath bit-width.

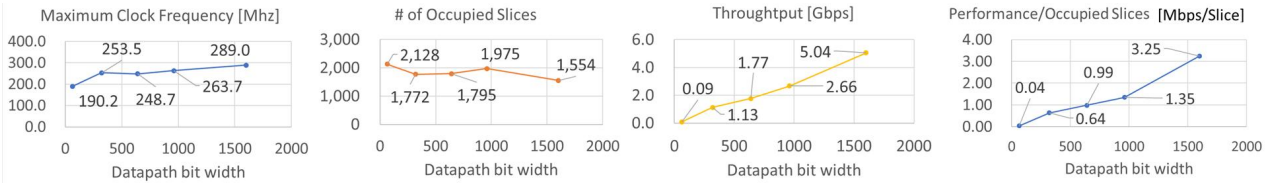
그림 3. 데이터패스 크기에 따른 SHA-3 해시 프로세서의 구조

하드웨어 구현을 위해 loop unrolling, pipelining 등 다양한 구조를 고려할 수 있다. loop unrolling 구조가 pipelining에 비해 처리율은 낮지만 적은 면적으로 구현할 수 있기 때문에 본 논문에서는 loop-unrolling 구조를 채택하였다. 선정된 데이터패스를 하드웨어로 구현하기 위해 그림 3과 같은 3 가지 구조를 구상하였다.

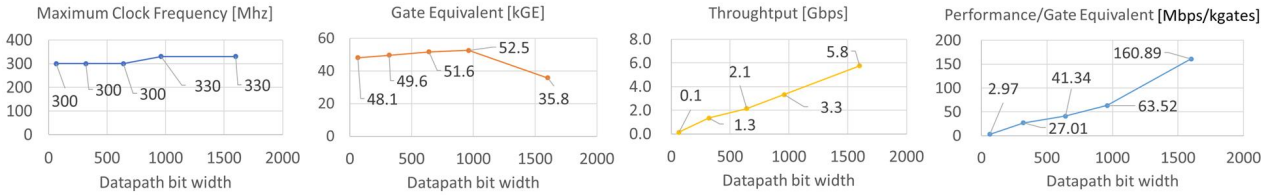
그림 3-(a)는 64-비트 데이터패스로 설계된 SHA-3 해시 프로세서의 라운드 블록 구조이며, state의 lane 단위로 연산을 수행하기 때문에 ρ 스텝, π 스텝 매핑에 유리하다. 하지만 χ 스텝 매핑은 row 단위로 연산을 수행하기 때문에 중간 결과를 저장하는 레지스터가 필요하여 1600-비트 레지스터 두 개가 포함된다. 이 구조는 라운드 연산을 수행하는데 50 클록 사이클이 소요되며, 한 블록 연산에 1200 클록 사이클이 소요된다.

그림 3-(b)는 320-비트, 640-비트, 960-비트 데이터패스로 설계된 라운드 블록 구조이다. 320-비트 단위로 매핑을 수행하는 구조이며, 640-비트, 960-비트 데이터패스는 320-비트 단위로 병렬 연산을 수행한다. 이 구조는 1600-비트 레지스터 두 개를 포함하며, 라운드 연산이 끝날 때마다 중간 결과 값이 두 개의 레지스터에 번갈아 저장된다. 320-비트, 640-비트, 960-비트 데이터패스 구조에서는 한 블록을 연산하는데 각각 120, 78, 48 클록 사이클이 소요된다.

그림 3-(c)는 1600-비트 데이터패스로 설계된 라운드 블록 구조이다. state에 포함된 모든 비트의



(a) In the case of synthesis using Xilinx Virtex-5 FPGA device



(b) In the case of synthesis using 0.18 µm CMOS standard cell library

Fig. 4. Performance analysis results for determining SHA-3 hash processor architecture.
그림 4. SHA-3 해시 프로세서의 아키텍처 결정을 위한 성능분석 결과

데이터를 1 클록 사이클에 연산하므로, 1600-비트 레지스터 하나만 사용되며, 한 블록 연산에 24 클록 사이클이 소요된다.

그림 3의 5 가지 데이터패스 구조로 SHA-3 해시 프로세서를 구현하고, Virtex-5 FPGA 디바이스로 합성하여 성능과 면적 사이의 관계를 분석한 결과는 그림 4와 같다. 최대 동작 주파수는 1600-비트 데이터패스로 설계된 경우가 289 MHz로 가장 높은 것으로 예측되었다. 소요된 슬라이스 수는 그림 3-(c) 구조의 1600-비트로 설계된 경우가 1,554개로 가장 적은 것으로 평가되었으며, 이는 다른 데이터패스 구조에 비해 1600-비트 레지스터가 한 개만 사용되는 것에 기인한다. 1600-비트 데이터패스를 갖는 프로세서가 5.04 Gbps로 가장 높은 처리율을 가지며 면적대비 성능 비도 가장 높게 나타났다.

그림 4-(b)는 5가지 데이터패스 구조로 구현된 SHA-3 해시 프로세서를 0.18 µm CMOS 셀 라이브러리로 합성한 결과를 토대로 성능 및 면적을 비교 분석한 결과이다. 최대 동작주파수는 1600-비트, 960-비트의 데이터패스를 갖는 경우가 330 MHz로 가장 높게 나타났으며, 1600-비트 데이터패스로 설계된 경우가 35,794 GE로 가장 적은 면적을 차지하였다. 처리율은 1600-비트 데이터패스의 구조가 5.7 Gbps로 가장 높았으며, 면적대비 성능 비도 가장 높았다. 이와 같은 하드웨어 구조에 따른 면적, 성능 분석결과로부터, SHA-3 해시 알고리듬의 최적 하드웨어 구현을 위해서는 loop-unrolling 구조의 1600-비트 데이터패스로 설계하는 것이 가장 효율적인 것으로 판단된다.

Ⅳ. SHA-3 해시 프로세서의 하드웨어 구현

그림 5는 설계된 SHA-3 해시 프로세서의 최상위 구조이며, Cortex-M0 인터페이스를 위한 레지스터 맵이 포함된 Reg_map 블록, 패딩 알고리듬에 따라 메시지 패딩을 처리하는 Padder 블록, 순열변환 연산을 수행하는 라운드 블록, 그리고 각 모듈의 동작을 제어하는 제어블록으로 구성된다. 라운드 블록은 그림 3-(c)의 1600-비트 데이터패스로 구현되었고, 각 블록 간 데이터 이동은 64-비트로 설계되었으며, Reg_map → Padder → Round → Reg_map 순으로 데이터가 처리된다. Reg_map 블록은 Cortex-M0 인터페이스를 위한 레지스터 맵을 포함하며, SHA-3 해시 프로세서의 동작 제어를 위한 컨트롤 레지스터와 연산될 데이터를 저장하

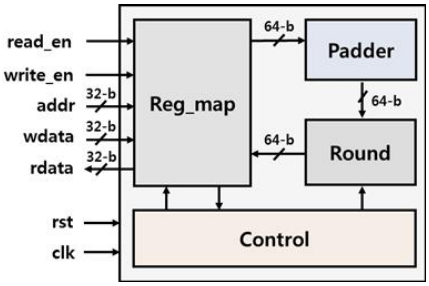


Fig. 5. Architecture of SHA-3 processor.
그림 5. SHA-3 프로세서의 구조

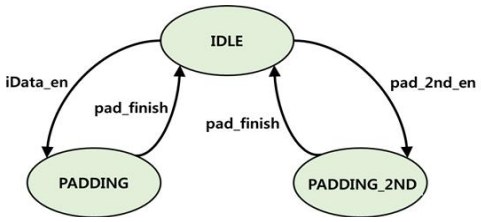


Fig. 6. FSM of Padder block.
그림 6. Padder 블록의 유한상태머신

는 데이터 레지스터로 구성된다. Cortex-M0로부터 받는 입력 메시지는 데이터 레지스터에 저장하고, Cortex-M0에 의해 컨트롤 레지스터의 값이 바뀌면 그에 따라 필요한 신호들이 생성된다. 연산이 완료된 해시 값은 데이터 레지스터에 저장된다.

Padder 블록으로 입력되는 메시지는 가변 길이의 메시지이고, 첫 부분부터 64-비트 단위로 순차적으로 data_in 포트를 통해 입력된다. 패딩 동작은 그림 6에 표현된 유한상태머신(FSM)을 기반으로 동작한다. IDLE 상태에서 iData_en 또는 pad_2nd_en 신호에 따라 각각 PADDING 상태와 PADDING_2ND 상태로 천이된다. PADDING 상태에서는 입력되는 데이터를 라운드 블록으로 전송한다. 패딩 비트를 추가할 때에는 계수기 출력이 $(r/64-1)$이면서 valid_data가 0x7인 경우를 제외한 모든 경우에 대해서 패딩이 수행된다. 계수기의 출력이 $(r/64-1)$이면서 valid_data가 0x7인 경우는 먼저 PADDING 상태로 천이되어 데이터를 $r/64$ 클록 사이클 동안 라운드 블록으로 전송하고 IDLE 상태로 돌아가며, 라운드 블록에서 입력된 블록에 대한 연산이 끝날 때까지 대기한다. 라운드 블록에서 연산이 끝나면 PADDING_2ND 상태로 천이되어 패딩 비트를 추가하고 라운드 블록으로 전송한다.

라운드 블록은 그림 7과 같이 1600-비트 레지스터와 $KECCAK-p$ 순열변환을 위한 조합회로로 구성된다. 첫 번째 블록의 연산이 수행되기 전에 1600-비트 레지스터의 값은 전부 0으로 초기화된다. Padder 블록을 거쳐 처리된 64-비트 데이터가 라운드 블록으로 입력되면, 1600-비트 레지스터의 상위 64-비트부터 순차적으로 저장된다. 저장된 데이터의 크기가 r-비트가 되면, 1600-비트의 하위 $(1600-r)$-비트의 *capacity*와 함께 라운드 연산이 시작된다. 연산된 데이터는 다시 1600-비트 레지스터로 저장되고 다음 라운드 연산이 수행되며, 24회 라운드 연산을 거쳐 한 블록의 연산이 완료된다. 마지막 라운드 연산이 끝나면, 다음 블록의 연산을 위해서 입력되는 데이터와 1600-비트 레지스터에 저장된 값이 64-비트씩 순차적으로 XOR되어 r-비트만큼 1600-비트 레지스터에 저장되고, 하위 $(1600-r)$-비트를 *capacity*로 하여 다음 블록의 첫 번째 라운드 연산에 사용된다. 마지막 블록에 대한 연산 결과가 1600-비트 레지스터에 저장되면, 상위 64-비트부터 순차적으로 d-비트만큼 출력된다.

$KECCAK-p$ 순열변환 블록은 그림 8과 같이 θ 스텝 블록, ρ와 π 스텝 블록, 그리고 χ와 ι 스텝 블록으로 분할하여 구현하였다. θ 스텝 블록은 내부에 5개의 하위블록으로 구성되며, 이 하위블록들은 하나의 sheet에 포함된 5개의 lane과 매핑에 필요한 2개의 패리티를 다른 블록으로부터 입력받는

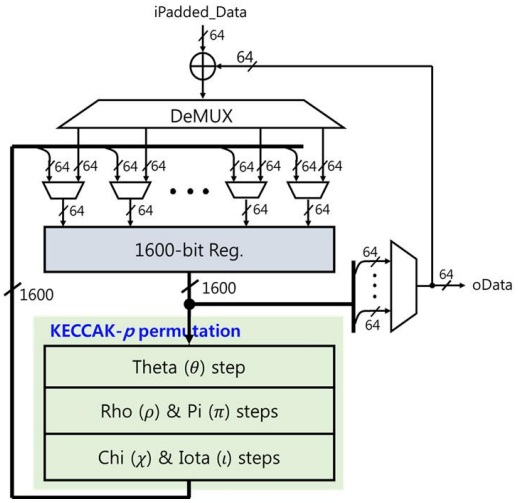


Fig. 7. Round block.
그림 7. 라운드 블록

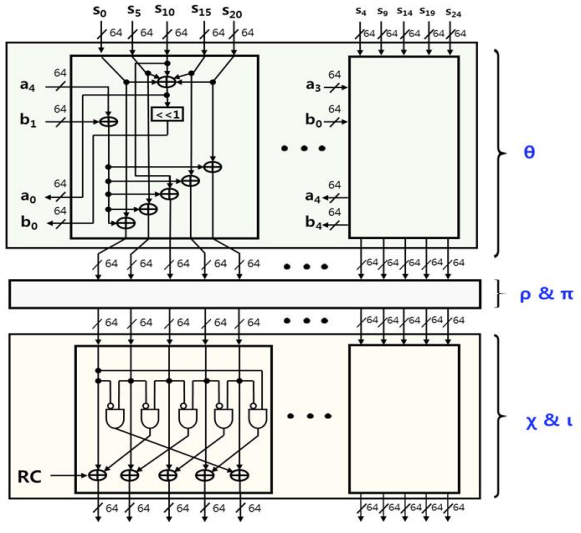


Fig. 8. KECCAK-p permutation block.
그림 8. KECCAK-p 순열 변환 블록

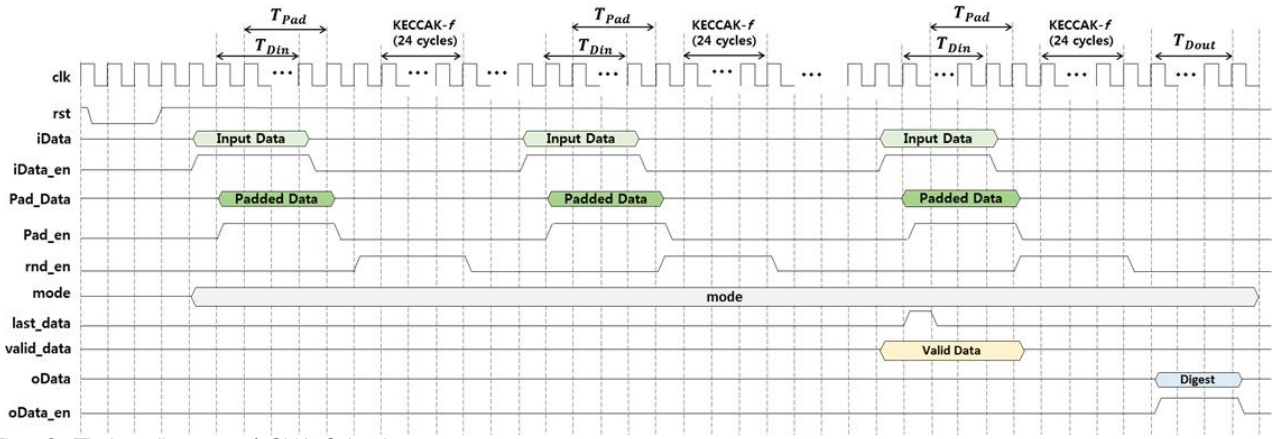


Fig. 9. Timing diagram of SHA-3 hash processor.
그림 9. SHA-3 해시 프로세서의 타이밍 도

다. 그리고 각 블록에서는 다른 블록에 사용되는 2개의 패리티를 계산하여 출력으로 내보내고, sheet 단위로 320-비트 데이터가 매핑된다. state는 5개의 sheet 표현될 수 있으므로 θ 스텝은 5개 하위블록의 병렬로 구성되고, 각 블록에 입력되는 sheet 단위의 데이터가 매핑된다.

ρ 스텝과 π 스텝 매핑이 수행되는 부분에서의 연산은 lane 단위로 독립적으로 수행된다. ρ 스텝 매핑을 위해 비트열이 위치하는 x, y 좌표에 따라 할당된 오프셋만큼 순환시프트 연산을 수행하는 회로와 π 스텝 매핑을 위해 lane 단위로 재배치하는 회로로 구현된다.

χ 스텝과 ι 스텝 매핑이 수행되는 부분의 연산은 하위블록 5개의 병렬로 구성되어 320-비트 단위로 수행되며, 각 하위블록의 연산은 plane에 포함된 5개의 row를 입력으로 받는다. 하위블록의 연산은 독립적으로 수행되고, 하위블록 내부는 XOR, AND, NOT 게이트로 구성된다. ι 스텝 매핑은 χ 스텝을 통해 매핑 된 데이터들 중 $x=0$, $y=0$ 좌표에 해당하는 lane에 대해서 라운드 상수와 XOR 연산되도록 구현된다. ι 스텝에서 사용되는 라운드 상수는 문헌 [11]의 방법을 적용하여 8-비트 LUT로 구현하였다. 64-비트 LUT로 구현하는 직접적인 방식과 비교하여 LUT의 면적이 1/8로 감소되고 8-비트에 대한 XOR 연산만 필요하므로, 경량화 구현이 가능하다.

그림 9는 SHA-3 해시 프로세스의 동작 타이밍도이다. T_{Din} 클록 사이클 동안 Reg_map 블록으로부터 iData 포트를 통해 r-비트만큼의 데이터가 입력된다. T_{Pad} 클록 사이클 동안은 Padder 블록을 통해 처리된 Padded_Data가 생성되며, 이 때 소요되는 클록 사이클은 T_{Din}과 동일하다. Padded_en 신호가 1로 유지되는 동안 라운드 블록 내부의 1600-비트 레지스터에 Padded_Data가 순차적으로 저장되고, T_{Pad} 클록 사이클 이후에 rnd_en 신호가 1로 되고, 24 클록 사이클 동안 $KECCAK-f[1600]$ 순열변환이 진행된다. 모든 메시지 블록의 처리가 완료될 때까지 위의 동작이 반복되고, 마지막 $KECCAK-f[1600]$ 순열변환이 완료되면, T_{Dout} 클록 사이클 동안 1600- 비트 레지스터에 저장된 해시 값이 64-비트 단위로 d-비트만큼 출력되며, 출력되는 해시 값을 Reg_map 블록 내부의 레지스터에 저장된다.

설계된 SHA-3 해시 프로세서는 그림 10과 같이 Cortex-M0에 슬레이브로 인터페이스 되어 동작한다. SHA3_ahb_slave는 SHA-3 해시 프로세서와 AHB 프로토콜에 따라 Cortex-M0와 데이터를 송수신하는 AHB_slave_IF 블록으로 구성된다. AHB_slave_IF는 AHB 프로토콜에 의해 Cortex- M0로부터 전송되는 데이터를 SHA-3 해시 프로세서 내부의 레지스터 맵에 저장하고, Cortex-M0로부터 해시 값 출력을 요청 받으면 레지스터 맵에 저장된 해시 값을 Cortex-M0로 전송한다.

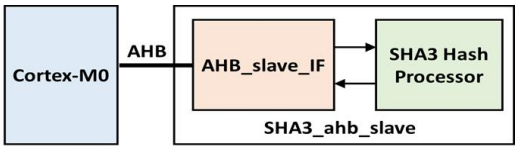


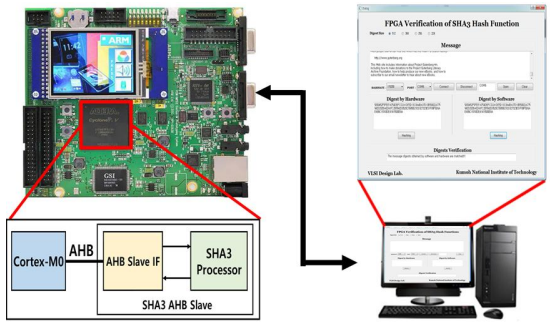
Fig. 10. Cortex-M0 interface of SHA-3 slave.
그림 10. SHA-3 슬레이브의 Cortex-M0 인터페이스

V. 통합 검증 및 비교

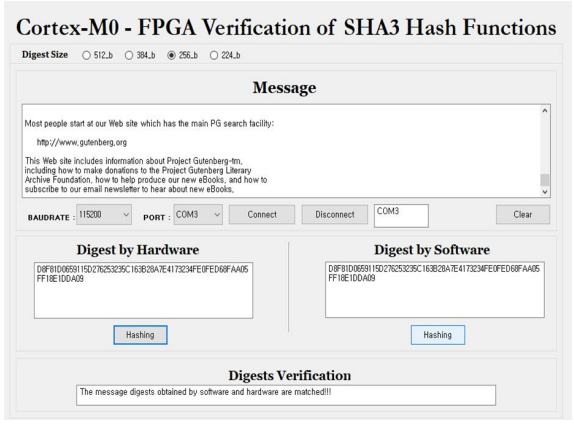
설계된 SHA-3 해시 프로세서는 RTL 시뮬레이션을 통한 기능검증을 수행한 후, FPGA 구현을 통해 하드웨어 동작을 검증하였다. SHA-3 해시 프로세서와 UART 블록 그리고 버퍼 회로를 Xilinx Virtex-5 FPGA에 구현했으며, Python을 이용하여 검증용 GUI와 SHA-3 해시 함수의 소프트웨어 버전을 구현하였다. NIST FIPS 180-2 표준 문서의 참조 구현 값을 사용하여 검증한 결과, 소프트웨어 결과와 FPGA에 구현된 SHA-3 해시 프로세서의 결과가 일치함을 확인하였다.

그림 10의 Cortex-M0 인터페이스 설계를 검증하기 위해 SHA-3 해시 코어와 AHB_slave_IF 블록으로 구성된 슬레이브를 BFM(bus function model) 시뮬레이션으로 검증하여 데이터의 송수신이 AHB 규격에 맞게 동작함을 확인하였다.

Cortex-M0와 인터페이스된 SHA-3 해시 프로세서를 HW/SW 통합 검증하기 위해 그림 11-(a)의 검증 플랫폼을 사용하였다. Cyclone-V FPGA 디바이스가 탑재된 V2M-MPS2 보드를 사용하였으며, Cortex-M0와 SHA3_ahb_slave를 포함하는 그림 10의 회로를 FPGA에 구현하였다. 검증 플랫폼은 UART 포트를 통해 PC와 연결되고, 검증 과정을 제어하는 소프트웨어가 Cortex- M0에 프로그래밍 되며, PC의 GUI 화면을 통해 검증 결과를 확인한다. 그림 11-(b)는 설계된 SHA-3 해시 프로세서의 HW/SW 통합검증 결과 화면이다. 입력 메시지의 길이는 163,809-바이트이며 4가지 SHA-3 해시 함수를 검증했다. 그림 11-(b)는 SHA3-256 해시 함수의 검증결과를 보인 것이며, 이와 같은 HW/SW 통합검증을 통해 SHA-3 해시 프로세서가 Cortex-M0에 슬레이브로 연결되어 정상적으로 동작함을 확인하였다.



(a) HW/SW co-verification platform



(b) Screenshot of SHA3-256 mode

Fig. 11. HW/SW co-verification of SHA-3 processor and Cortex-M0 with FPGA implementation.

그림 11. FPGA 구현에 의한 SHA-3 프로세서와 Cortex-M0의 하드웨어/소프트웨어 통합 검증

표 1은 설계된 SHA-3 해시 프로세서의 성능을 문헌에 발표된 사례와 비교한 결과이다. 처리량 측면에서는 문헌 [12]의 약 88% 정도이나, 문헌 [14]와 비슷한 것으로 분석되었다. 하드웨어 자원소모에 있어서는 문헌 [12]의 64%, 문헌 [14]의 43% 정도만 사용되어 경량화 하드웨어로 구현되었다. 또한, 최대 동작 주파수도 문헌 [12], [14]에 비해 높은 것으로 평가되었다. 문헌 [13]의 사례와는 처리량과 하드웨어 자원에 대해서 직접적으로 비교할

Table. 1. Comparison of SHA-3 hash processors.

표 1. SHA-3 해시 프로세서 비교

	[12]	[13]	[14]	This paper
Algorithms supported	SHA3-224/256/384/512	SHA3-512	SHA3-224/256/384/512, SHAKE-256	SHA3-224/256/384/512
Message format	byte-level	-	bit-level	byte-level
Round structure	iterative round (128-b I/O)	iterative round (2-stage)	iterative round	iterative round
Padding scheme	hardware	no	hardware	hardware
# of cycles for a block	25	48	24	33
Troughput [Gbps]	5.70 @SHA3-512	25.4 @SHA3-512	5.28 @SHA3-512	5.04 @SHA3-512
Technology	Virtex5 xc5vsx240t	TSMC 65-nm standard cell	Virtex5 xc5vlx50	Virtex5 xc5vsx95t
HW Resources	2,573 slices	49.1k [GE]	3,887 slices	1,672 slices
Max. Freq.	285 Mhz	1.192 GHz	220 MHz	289 MHz

수 없지만, 본 논문의 결과는 문헌 [13]에 비해 많은 해시 함수를 지원하고, Padder 블록이 하드웨어로 구현되었다는 점에서 유용성이 우수한 것으로 평가된다.

Ⅵ. 결론

NIST FIPS 202에 표준으로 제시된 SHA-3 해시 프로세서의 하드웨어 구현을 위한 최적 설계조건을 분석하고, 이를 토대로 경량화 하드웨어로 구현하였다. 저면적 구현을 위해 loop-unrolling 구조를 채택하였고, 면적대비 성능이 가장 우수한 1600-비트 데이터패스로 구현하였다. 0.18 μm CMOS 셀 라이브러리로 합성한 결과, 클록 주파수 100 Mhz에서 35,753 GE로 합성되었으며, 최대 동작 주파수는 330 Mhz로 예측되었다.

SHA-3 해시 프로세서를 Cortex-M0에 슬레이브로 인터페이스하고, FPGA에 구현하여 HW/SW 통합검증을 통해 정상적으로 동작함을 확인하였다. 본 논문의 SHA-3 해시 프로세서는 저면적 구현을 장점으로 가지며, 따라서 하드웨어 자원이 제한된 IoT 분야와 보안 SoC(system-on-chip)에 IP로 이용이 가능할 것으로 평가된다.

References

[1] NIST Std. FIPS-197, Advanced Encryption Standard, National Institute of Standard and Technology (NIST), 2001.

[2] KS X 1213, 128 bit Block Encryption Algorithm ARIA, Korean Agency for Technology and Standards, 2004.

[3] R. Rivest, A. Shamir and L. Adleman, “A method for obtaining Digital Signatures and Public-Key Crypto-system,” *Communications of Association for Computing Machinery (ACM)*, vol.21, no.2, pp.120-126, 1978.
DOI:10.1145/359340.359342

[4] NIST Std. FIPS PUB 186-2, Digital Signature Mechanism with Appendix (Part 3) Korean Certificate-based Digital Signature Algorithm using Elliptic Curve, Telecommu-nications Technology Association, 2012.

[5] TTA std. TTAK.KO-12.0011/R1, Hash Function Standard-Part2: Hash Function Algorithm Standard (HAS-160), Telecommu-nications Technology Association (TTA), 2000.

[6] NIST std. FIPS 180-2, Secure Hash Standard (SHS), National Institute of Standard and Technology (NIST), 2001.

[7] P. Barreto and V. Rijmen, “The Whirlpool Hashing Function,” pp.1-20, 2003.

[8] National Institute of Standards and Technology. FIPS PUB 202 - SHA3 Standard: Permutation-Based Hash and Extendable-Output function, 2015.

[9] D. S. Kim and K. W. Shin, “Analysis of Optimal Design conditions for SHA3-512 Hash Function,” *Proceedings of 2018 2nd Conference of the Korea Institute of Information and Communication Engineering*, vol.22, no.2, pp.187-189, Oct. 2018.

[10] D. S. Kim and K. W. Shin, “A Hardware Implementation of SHA3 Hash Processor using Cortex-M0,” *Proceedings of 2019 International Conference on Electronics, Information, and Communication (ICEIC 2019)*, Accepted.

[11] M. M. Wong, J. Haj-Yahya, S. Sau and A. Chattopadhyay,″ A New High Throughput and Area Efficient SHA-3 Implementation,″ *Proceedings of 2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, Florence, pp.1-5, 2018.

[12] G. Provelengios et al, “FPGA-Based Design Approaches of Keccak Hash Function,” *Proceedings of 15th Euromicro Conference on Digital System Design*, Izmir, Turcky, pp.648-653, 2012.
DOI:10.1109/DSD.2012.63

[13] 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 System*, vol.33, no.7, pp.1105-1109, 2014. DOI:10.1109/TCAD.2014.2307002

[14] B. Y. Choi, “Efficient Hardware Design of Hash Processor Supporting SHA-3 and SHAKE256 Algorithms,” *Journal of the Korea Institute of Information and Communication Engineering*, vol.21, no.6, pp.1075-1082, 2017.
DOI:10.6109/jkiice.2017.21.6.1075

BIOGRAPHY

Dong-Seong Kim (Member)

2018 : BS degree in School of Electronic Engineering, Kumoh National Institute of Technology.
2018~: Graduate student in School of Electronic Engineering, Kumoh National Institute of Technology

Kyung-Wook Shin (Member)

1984 : BS degree in Electronic Engineering, Korea Aerospace University
1986 : MS degree in Electronic Engineering, Yonsei University
1990 : Ph. D. degree in Electronic Engineering, Yonsei University
1990~1991 : Senior Researcher in Semiconductor Research Center, Electronics and Telecommunication Research Institute (ETRI)
1991~Present : Professor in School of Electronic Engineering, Kumoh National Institute of Technology
1995~1996 : University of Illinois at Urbana- Champaign (Visiting Professor)
2003~2004 : University of California at San Diego (Visiting Professor)
2013~2014 : Georgia Institute of Technology (Visiting Professor)