JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 22, No. 1 : 109~116 Jan. 2018

IoT 보안을 위한 SHA-256 해시 프로세서의 면적 효율적인 설계

이상현 · 신경욱*

An Area-efficient Design of SHA-256 Hash Processor for IoT Security

Sang-Hyun Lee · Kyung-Wook Shin*

School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, Kyungbuk 39177, Korea

요 약

전자서명, 인증 코드, 키 생성 알고리듬 등의 보안 프로토콜에 사용되는 SHA-256 해시 함수를 면적 효율적으로 설계하였다. 설계된 SHA-256 해시 프로세서는 입력 메시지에 대한 패딩 및 파싱 기능을 수행하는 패더 블록을 포함하여 프리프로세싱을 위한 소프트웨어 없이 동작하도록 구현하였다. 라운드 연산을 16-비트 데이터 패스로 구현하여 64 라운드 연산이 128 클록 주기에 처리되도록 하였으며, 이를 통해 저면적 구현과 함께 성능 대비 하드웨어 복잡도 (area per throughput; APT)를 최적화 하였다. 설계된 SHA-256 해시 프로세서는 Virtex5 FPGA로 구현하여 정상 동작함을 확인하였으며, 최대 116 MHz 클록 주파수로 동작하여 337 Mbps의 성능을 갖는 것으로 평가되었다. ASIC 구현을 위해 0.18-μm CMOS 셀 라이브러리로 합성한 결과, 13,251 GE로 구현되었으며, 최대 동작주파수는 200 MHz로 예측되었다.

ABSTRACT

This paper describes an area-efficient design of SHA-256 hash function that is widely used in various security protocols including digital signature, authentication code, key generation. The SHA-256 hash processor includes a padder block for padding and parsing input message, so that it can operate without software for preprocessing. Round function was designed with a 16-bit data-path that processed 64 round computations in 128 clock cycles, resulting in an optimized area per throughput (APT) performance as well as small area implementation. The SHA-256 hash processor was verified by FPGA implementation using Virtex5 device, and it was estimated that the throughput was 337 Mbps at maximum clock frequency of 116 MHz. The synthesis for ASIC implementation using a 0.18-μm CMOS cell library shows that it has 13,251 gate equivalents (GEs) and it can operate up to 200 MHz clock frequency.

키워드 : Secure Hash Algorithm (SHA), SHA-256, 해시 프로세서, 메시지 다이제스트, 정보보안, IoT 보안

Key word : Secure Hash Algorithm (SHA), SHA-256, hash processor, message digest, information security, IoT security

Received 15 September 2017, **Revised** 31 October 2017, **Accepted** 09 November 2017

* **Corresponding Author** Kyung-Wook Shin(E-mail:kwshin@kumoh.ac.kr, Tel:+82-54-478-7427)
School of Electronic Engineering, Kumoh National Institute of Technology, Gumi, Kyungbuk 39177, Korea

Open Access http://doi.org/10.6109/jkiice.2018.22.1.109 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

정보통신기술의 발전에 따라 우리 주변의 다양한 사물(장치)들이 지능화되고 네트워크화 되는 사물인터넷 (IoT; Internet of Things) 기술이 빠르게 확산되고 있다. 2017년 현재 수십억 개 이상의 IoT 장치가 인터넷에 연결되어 있으며, 앞으로 그 수가 기하급수적으로 증가하여 2020년에는 500억 개 이상의 장치가 연결되어 새로운 서비스 및 시장이 창출될 것으로 예측되고 있다. IoT 기기와 서비스가 빠른 속도로 확대됨에 따라 개인정보 침해, 모바일 악성코드 등 보안 위협이 나타날 것으로 예상되며, 이에 대한 대비가 중요하다[1, 2]. 다양한 보안 위협으로부터 정보를 안전하게 보호하기 위해서는 기밀성 (confidentiality), 인증 (authentication), 무결성 (integrity) 등의 보안 서비스를 제공하기 위한 보안 알고리듬의 하드웨어 및 소프트웨어 구현이 필요하다.

보안 알고리듬에는 대칭키 암호, 공개키 암호, 해시 함수 등 다양한 알고리듬이 존재한다. 해시 함수는 주어진 메시지(정보)를 고정된 길이의 다이제스트(digest)로 압축하는 기능을 가지며, 정보의 위조나 변조 여부를 확인할 수 있는 무결성 제공과 함께 전자서명, 인증, 키교환, 의사난수 생성 등의 정보보안 목적으로 폭넓게 사용되고 있다. 해시 함수는 미국 표준기술연구소 (NIST; National Institute of Standards and Technology)에서 개발된 SHA (Secure Hash Algorithm)[3], 국내에서 개발된 HAS-160[4] 등 다양한 알고리듬들이 있다.

SHA 해시 함수는 인터넷 뱅킹, 전자서명, 메시지 인증 코드, 키 교환 알고리듬, 키 생성 알고리듬 등 다양한 분야의 보안 프로토콜에 사용되고 있다. SHA-1은 TLS, SSL, PGP, SSH 등 많은 보안 프로토콜과 프로그램에 사용되어 왔으나, 2005년에 SHA-1에 대한 충돌 쌍 공격[5]이 중국의 연구팀에 의해 발표됨에 따라 NIST에서는 해시 함수의 모든 응용 프로그램에 대해 최소한 SHA-256을 사용하는 것을 권장하였다[6].

본 논문에서는 SHA-256 해시 함수를 저면적 하드웨어로 설계하여 하드웨어 리소스가 제한된 IoT 환경에 적합하도록 구현하였다. Xilinx ISE를 통해 시뮬레이션 검증을 하였고, Virtex5 FPGA, UART 통신, 소프트웨어를 이용하여 하드웨어 동작을 검증하였다. Ⅱ장에서는 SHA-256 해시 함수에 대해 간략히 설명하고, Ⅲ장에서는 SHA-256 해시 프로세서 설계에 대해 설명한다. 설계된 회로의 기능 검증 및 FPGA 구현에 대해 Ⅳ장에서 기술하며, Ⅴ장에서 결론을 맺는다.

Ⅱ. SHA-256 해시 함수[3]

Merkle-Damgard 구조를 갖는 SHA는 입력 메시지 블록에 압축함수를 반복 적용하여 일정한 크기의 다이제스트를 출력하는 해시 함수 이다. 입력 메시지에 메시지 길이와 패딩을 붙이고, 압축함수 크기로 분할하여 메시지 블록을 만든 후, 메시지 블록과 다이제스트가 압축함수를 통해 새로운 다이제스트로 생성된다. 맨 처음 압축함수에는 SHA 알고리듬에 따라 정해진 초기 다이제스트가 사용된다. 이와 같은 과정은 크게 나누어 프리프로세싱과 압축함수로 구성된다. 표 1은 SHA 해시 알고리듬에 따른 파라미터를 나타내었다[7].

2.1. 프리프로세싱

프리프로세싱은 메시지 패딩 (padding), 메시지 파싱 (parsing), 초기 다이제스트 (digest) 설정으로 구성된다. 메시지를 입력받아 메시지 패딩 및 메시지 파싱을 통해 다이제스트 길이의 2배인 512-비트의 메시지 블록을 만든다. 메시지 패딩은 그림 1과 같이 구성되며, 메시지, 길이, 패딩 3가지 영역으로 구성되고, 512-비트의 정수

Table. 1 Parameters of SHA hash function

SHA Hash algorithm	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message size [bit]	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size [bit]	512	512	512	1024	1024
Digest size [bit]	160	224	256	384	512
Number of rounds[N]	80	64	64	80	80
Word size [bit]	32	32	32	64	64

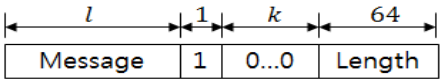


Fig. 1 Message padding

배가 되어야 한다. 패딩 영역은 첫 번째 비트가 1이고 나머지는 0으로 구성되며, 길이 영역은 64-비트로 고정되어 있고 메시지의 길이 값이 저장된다. 메시지의 길이를 l이라고 할 때, 식 (1)을 통해 패딩 구역에서 0의 개수인 k 값이 구해진다. 메시지 파싱은 패딩된 메시지가 512-비트씩 압축함수로 입력되도록 분할하고, 초기 다이제스트 설정은 첫 번째 압축함수의 연산을 위해 정해진 다이제스트가 입력되는 과정이다.

$$l + 1 + k \equiv 448 \bmod 512 \tag{1}$$

2.2. 압축함수

압축함수는 워드 단위로 연산되며, SHA-256의 워드 크기는 32-비트이다. 프리프로세싱을 거쳐 압축함수로 입력되는 메시지 블록은 16개의 워드로 구성되고, 64 라운드 동안 반복 연산을 통해 다이제스트가 생성되므로 64개의 워드가 필요하다. 16개의 워드 블록을 64개의 워드로 확장하기 위해 식 (2)의 워드 확장이 적용된다. 워드 확장에 사용되는 함수 σ_0, σ_1는 식 (3)과 같이 오른쪽 순환 이동 ROTR, 오른쪽 시프트 SHR, 그리고 XOR 연산으로 구성된다.

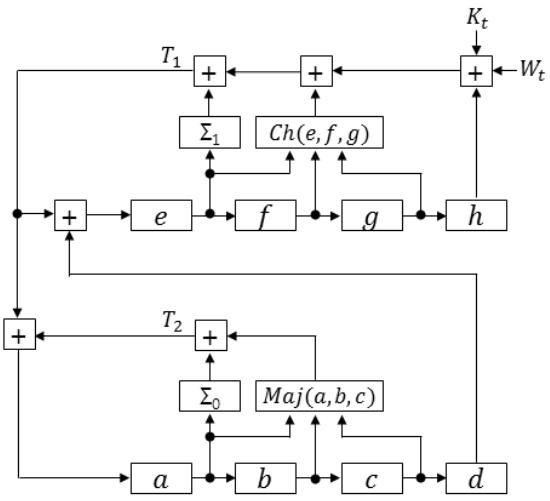


Fig. 2 Round structure of SHA-256 hash function

$$W_t = \begin{cases} M_t & (0 \le t \le 15) \\ \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16} & (16 \le t \le 63) \end{cases} \tag{2}$$

$$\begin{aligned} \sigma_0(x) &= ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x) \\ \sigma_1(x) &= ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x) \end{aligned} \tag{3}$$

압축함수의 라운드 구조는 그림 2와 같이 4가지 함수 Σ_0, Σ_1, *Maj*, *Ch*와 변수 a, b, c, d, e, f, g, h 그리고 가산기로 구성되며, 32-비트의 워드 W_t, 상수 K_t, 8개의 변수를 연산하여 새로운 8개의 변수를 생성한다. 식 (4)에서 보는 바와 같이, 함수 Σ_0, Σ_1은 오른쪽 순환 이동 ROTR과 XOR 연산으로 구성되고, 함수 $Ch(e,f,g)$ $Maj(a,b,c)$는 비트 AND ($\wedge$), NOT ($\neg$) 연산 그리고 XOR 연산으로 구성된다.

$$\begin{aligned} \Sigma_0(x) &= ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x) \\ \Sigma_1(x) &= ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x) \\ Maj(a,b,c) &= (a \wedge b) \oplus (a \wedge c) \oplus (b \wedge c) \\ Ch(e,f,g) &= (e \wedge f) \oplus (\neg e \wedge g) \end{aligned} \tag{4}$$

매 라운드 마다 변수 값을 생성하고 64번의 라운드 연산 후에 변수 값과 초기 다이제스트를 가산하여 최종 다이제스트를 생성한다. 메시지 블록이 두 개 이상인 경우에는 이전 블록의 다이제스트는 다음 메시지 블록에 대한 라운드 연산의 초기 다이제스트로 사용된다.

Ⅲ. SHA-256 해시 프로세서 설계

3.1. 전체 구조

SHA-256 해시 프로세서는 그림 3과 같이 패더 블록, 라운드 블록, 다이제스트 레지스터 그리고 제어 블록으로 구성된다. 입력 메시지에 대한 패딩 및 파싱 기능을 수행하는 패더 블록을 하드웨어로 구현하여 프리프로세싱을 위한 소프트웨어 없이 동작하도록 구현 하였다. 그림 2의 라운드 함수를 연산하여 다이제스트를 생성하는 라운드 블록은 저면적 구현을 위해 16-비트 데이터패스로 설계되었다. 다이제스트 레지스터는 다이제스트 초기 값과 생성되는 중간 결과 값을 저장한다.

3.2. 패더 블록

메시지 패딩과 메시지 파싱을 수행하는 패더 블록은

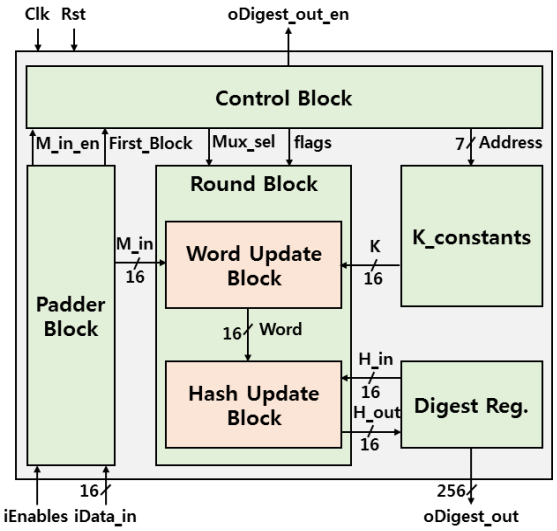


Fig. 3 Architecture of SHA-256 hash processor having 16-bit datapath

그림 4와 같이 IDEL, DATA, LENGTH의 세 개 상태로 구성되는 유한상태머신 (FSM)으로 설계되었다.

IDLE 상태에서는 데이터의 길이 값을 입력받아 해당 레지스터에 저장하고, data_in_en 및 last_in_en 신호에 의해 다음 상태로 천이된다. data_in_en 신호에 의해 메시지가 패더 블록으로 입력되고, 메시지 블록의 개수가 2개 이상이고 마지막 블록의 패딩 값이 메시지를 포함하지 않는 경우에 last_in_en 신호에 의해 다음 상태로 천이된다.

DATA 상태에서는 메시지와 “1”, “0”이 패딩된 값을 패더 레지스터에 입력한다. 메시지가 16-비트씩 입력되므로 메시지 길이 레지스터에 저장된 길이 값을 16씩 빼서 16미만이 되면 메시지가 끝났음을 알린다. 이 신호가 발생할 때 메시지 뒤에 “1”과 “0”을 붙여 패딩하고

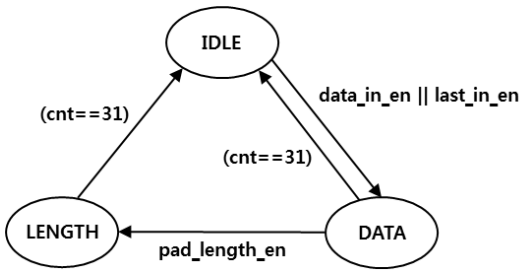


Fig. 4 FSM of Padder block

나머지는 “0”을 패딩하여 패더 레지스터에 저장한다. 한 블록의 메시지가 448-비트 미만일 경우 메시지 길이 값을 패딩하기 위해 pad_length_en 신호가 발생되어 다음 상태로 천이된다. 이 신호가 발생하지 않으면 패딩을 완료 한 후, IDLE 상태로 천이되어 다음 메시지 블록의 입력을 기다린다.

LENGTH 상태에서는 메시지 길이 값을 패딩하여 패더 레지스터에 입력한다. 패딩이 완료되면 IDLE 상태로 천이된다.

3.3. 라운드 블록

라운드 블록은 워드 확장을 통해 워드를 생성하는 워드 업데이트 블록과 라운드 연산을 통해 다이제스트를 생성하는 해시 업데이트 블록으로 구성된다. 패더 블록으로부터 메시지가 입력되면 64 라운드 (128 클록 주기) 동안 연산을 반복 처리하여 새로운 해시 값을 생성하고, 그 이후 생성된 새로운 해시 값과 초기 해시 값을 더하여 다이제스트를 16 클록동안 생성한다.

3.3.1. 워드 업데이트 블록

그림 5의 워드 업데이트 블록은 워드 확장을 통해 워드 W_t를 생성하며, 워드 값을 저장하는 32개의 16-비트 워드 업데이트 레지스터 W0~W31, 연산 함수 σ_0, σ_1, 16-비트 가산기, 캐리 레지스터 등으로 구성된다.

M_in 포트를 통해 32 클록 동안 메시지가 입력되어 워드 업데이트 레지스터 W0~W31에 초기 값으로 저장된다. 이후에는 가산기의 결과 값이 레지스터 W31에 저장된다. 워드 업데이트는 다음의 과정으로 이루어진다. 홀수 번째 클록에서 레지스터 W3, W2, W1의 값이

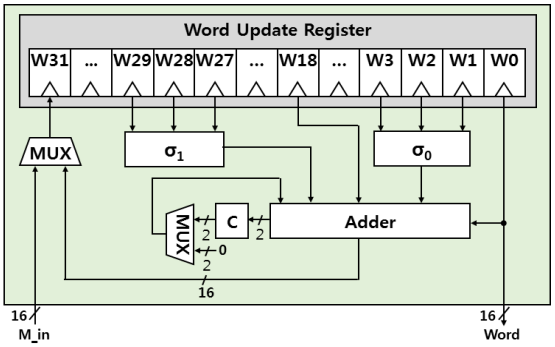


Fig. 5 Word update block

함수 σ_0을 통해 연산된 값, 레지스터 W29, W28, W27의 값이 함수 σ_1을 통해 연산된 값, 레지스터 W0, W18의 값이 가산된다. 짝수 번째 클록에서는 캐리 레지스터에 저장된 값도 함께 가산된다. 가산결과 값은 레지스터 W31에 저장되고 캐리 값은 캐리 레지스터에 저장된다. 동시에 워드 업데이트 레지스터에 저장되어 있던 값은 워드 단위로 오른쪽 시프트 된다. 따라서 한 라운드의 워드 업데이트 연산에 두 클록주기가 소요되며, 생성된 워드 값은 128 클록주기 동안 해시 업데이트 블록으로 입력된다.

3.3.2. 해시 업데이트 블록

압축함수의 라운드 연산을 수행하는 해시 업데이트 블록은 그림 6과 같이 변수 값을 저장하는 16개의 16-비트 해시 업데이트 레지스터 H15~H0 , 연산 함수 Σ_0, Σ_1, Maj, Ch , 16-비트 가산기, 캐리 레지스터 등으로 구성된다. 256-비트의 다이제스트 초기 값은 16-비트 단위로 16 클록동안 입력되어 해시 업데이트 레지스터 H0~ H15에 저장된다. 해시 업데이트는 다음의 과정으로 이루어진다. 홀수 번째 클록에서는 T_1 값과 H8 값이 가산되어 H7에 저장되고, T_1값과 T_2값을 가산하여 H15에 저장된다. 함수 $\Sigma_1(H7, H6, H5)$의 결과 값, 함수 $Ch(H6, H4, H2)$의 결과 값, 상수 K, 워드 W을 가산하여 T_1이 생성된다. 함수 $\Sigma_0(H15, H14, H13)$의 결과 값, 함수 $Maj(H14, H12, H10)$의 결과 값을 가산하여 T_2가 생성된다. 캐리 레지스터에 저장된 값도 함께 가산된다. 동시에 해시 업데이트 레지스터에 저장된 값은 워드 단위로 H7→H6→. . . →H1→H0, H15→H14→. . . →H9→H8로 시프트된다. 한 라운드의 해시 업데이트 연산에는 두 클록 주기가 소요되며, 생성된 해시 값은 128 클록주기 동안 다이제스트 레지스터로 입력된다.

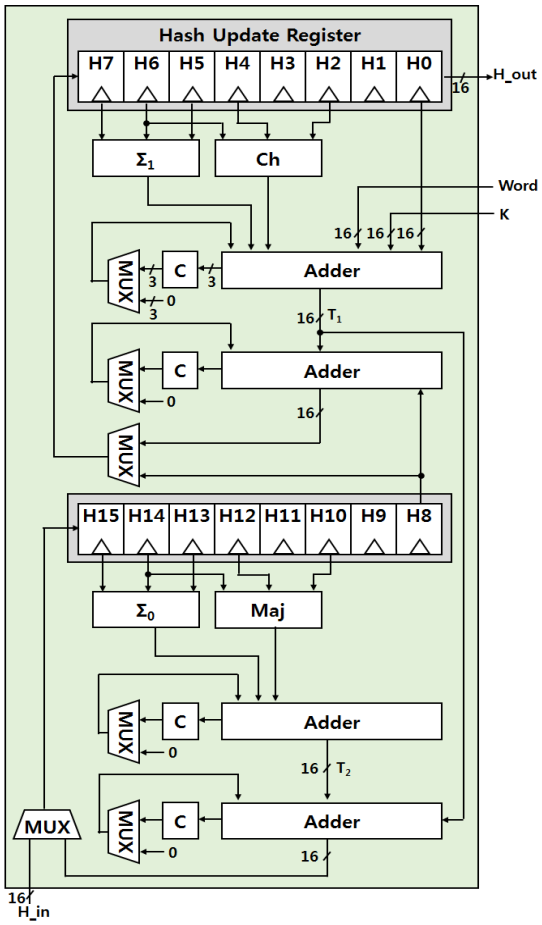


Fig. 6 Hash update block

3.3.3. 연산 함수

연산 함수 σ와 Σ는 워드와 다이제스트를 생성하기 위해 사용된다. 연산 함수 σ는 그림 7-(a)와 같이 오른쪽 순환 이동 POTR, 오른쪽 시프트 SHR, 멀티플렉서, XOR 게이트로 구성되고, 연산 함수 Σ는 그림 7-(b)와 같이 오른쪽 순환 이동 POTR , 멀티플렉서, XOR 게이트로 구성된다.

Ⅳ. 기능검증 및 FPGA 구현

Verilog HDL로 설계된 SHA-256 해시 프로세서의 기능검증 결과는 그림 8과 같다. 입력벡터는 표준문서[3]에 제시된 값을 사용하였고, 그림 8은 448-비트로 구성되는 메시지 “6162 6364 6263 6465 6364 6566 6465 6667 6566 6768 6667 6869 6768 696A 6869 6A6B 696A 6B6C 6A6B 6C6D 6B6C 6D6E 6C6D 6E6F 6D6E 6F70 6E6F 7071”을 사용하여 시뮬레이션한 결과이다. 설계된 SHA-256 해시 프로세서를 통해 256-비트의 다이제스트 “248D 6A61 D206 38B8 E5C0 2693 0C3E 6039 A33C E459 64FF 2167 F6EC EDD4 19DB

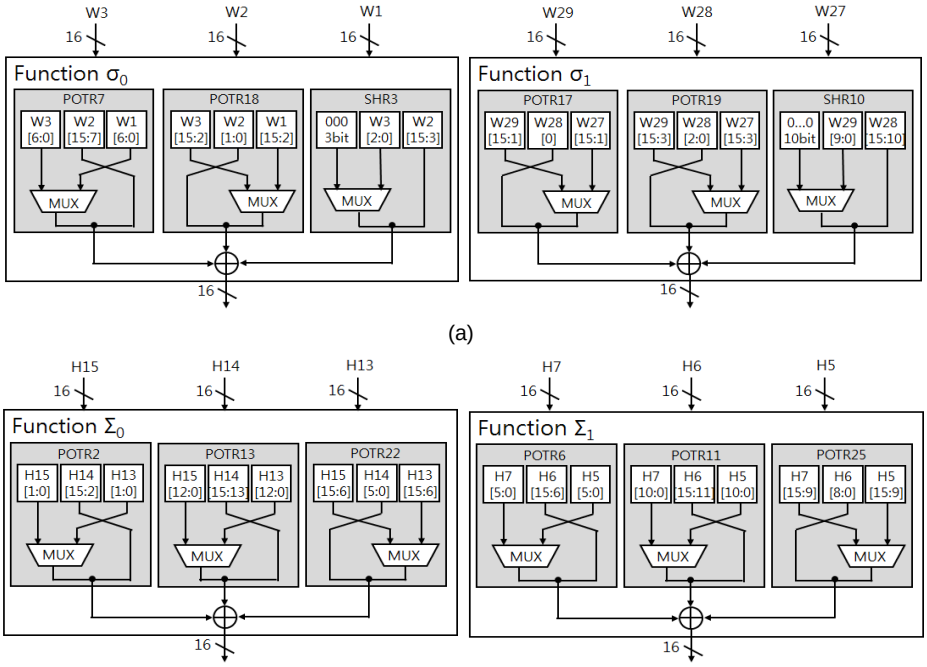


Fig. 7 Functions of round computation (a) Function σ (b) Function Σ

06C1"이 출력되어 표준문서 [3]의 참조 값과 일치하는 것을 확인하였으며, 설계된 SHA-256 해시 프로세서의 논리기능이 올바로 동작함을 확인하였다.

설계된 SHA-256 해시 프로세서는 FPGA 구현을 통해 하드웨어 동작을 검증하였다. FPGA 검증 시스템은

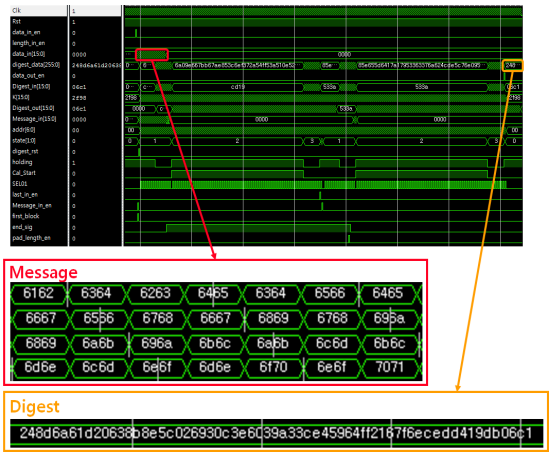
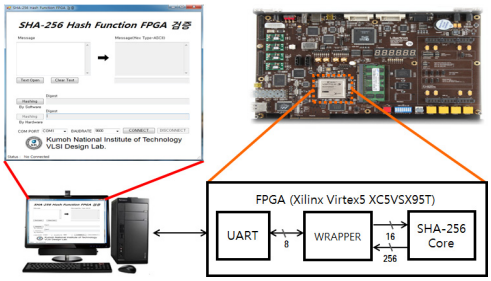


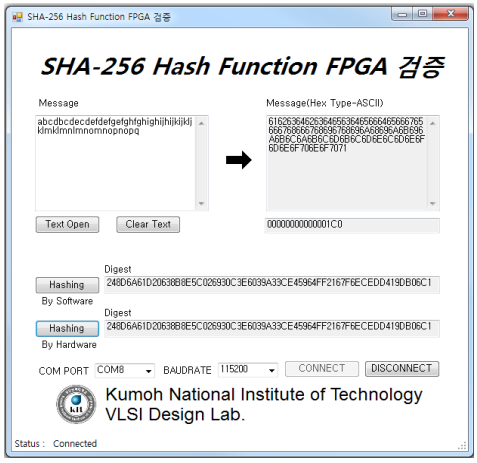
Fig. 8 Simulation results of SHA-256 hash processor

그림 9-(a)와 같이 FPGA 보드, UART I/F, 구동 소프트웨어로 구성되며, Virtex5 XC5VSX95T 디바이스가 사용되었다. PC에서 입력된 메시지와 메시지 길이 데이터가 RS232C 통신을 통해 FPGA로 전송된 후, 설계된 SHA-256 해시 프로세서를 통해 계산된 다이제스트 값이 다시 PC로 전송되어 화면에 표시된다. 그림 9-(b)는 FPGA 검증 결과의 화면 캡처를 보이고 있다. 왼쪽 상단부에 메시지 데이터를 입력하면 오른쪽 상단부에 메시지 길이가 표시되고 메시지 데이터는 ASCII 값으로 변환된다. FPGA에서 출력되는 다이제스트 값과 소프트웨어로 계산된 다이제스트 값이 일치하여 FPGA에 구현된 SHA-256 해시 프로세서가 올바르게 동작함을 확인하였다.

표 2는 본 논문에서 설계된 SHA-256 해시 프로세서와 문헌에 발표된 사례들을 비교한 것이다. 문헌 [8-10]의 SHA-256 해시 프로세서는 32-비트 데이터 패스로 설계되었으며, 본 논문의 SHA-256 해시 프로세서는 16-비트 데이터 패스로 설계되었다. 표 2에서 APT (Area Per Throughput)는 처리율 대비 하드웨어 복잡도를 나



(a)



(b)

Fig. 9 FPGA verification results of SHA-256 hash processor (a) FPGA verification setup (b) screen shot of FPGA verification results

타내며, 작을수록 우수한 설계를 의미한다. 문헌 [10]의 사례는 APT가 2.33으로 가장 크며, 문헌 [9]의 사례는 APT가 0.68로 우수하지만, 본 논문의 설계에 비해 약 2.5배의 슬라이스가 사용되므로 작은 하드웨어가 요구되는 환경에는 적합하지 않다. 문헌 [8]의 사례는 APT

Table. 2 Comparison of SHA-256 hash processors

	Our	Ref[8]	Ref[9]	Ref[10]
Data-path [bits]	16	32	32	32
Padder	H/W	NA	H/W	NA
FPGA Device	Virtex5	Virtex5	Virtex6	Virtex4
Number of Slices	300	1,203	736	1,994
Max. Freq. [MHz]	116	170	135	43
Throughput [Mbps]	337	1,359	1,081	856
APT	0.89	0.88	0.68	2.33

가 0.88로 본 논문의 설계와 유사하나 4배 많은 슬라이스를 필요로 한다. 본 논문의 SHA-256 해시 프로세서는 APT가 작고, 적은 수의 슬라이스로 설계되어 저면적 구현이 필요한 IoT 정보보안 분야에 적합하다.

Ⅴ. 결 론

본 논문에서는 SHA-256 해시 프로세서를 설계하고, FPGA 구현을 통해 하드웨어 동작을 검증했다. 설계된 SHA-256 해시 프로세서는 하드웨어 패더 블록을 포함하여 프리프로세싱을 위한 소프트웨어가 필요 없으며, 데이터 패스를 16-비트로 설계하여 저면적으로 구현하였다. 설계된 SHA-256 해시 프로세서는 Xilinx으로 합성한 결과 300개 슬라이스로 구현되었으며, 최대 동작 주파수는 116 MHz로 평가되었다. 0.18-μm CMOS 셀 라이브러리로 합성한 결과 13,251 GE (gate equivalent)로 구현되었으며, 최대 200 MHz의 클록 주파수로 동작할 수 있다. 설계된 SHA-256 해시 프로세서는 하드웨어 리소스가 제한된 IoT 디바이스의 정보보안 코어로 활용이 가능하다.

ACKNOWLEDGEMENTS

- This work was supported by Korea Institute for Advancement of Technology (KIAT) grant funded by the Korean government (Ministry of Trade, Industry & Energy, HRD Program for Software- SoC convergence) (No. N0001883)
- 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)
- Authors are thankful to IDEC for supporting EDA S/W

REFERENCES

[1] D.H. Kim, S.W Yoon, and Y.P Lee, “Security for IoT Service,” *Journal of The Korean Institute of Communication Sciences*, vol. 30, no.8, pp.53-59, July 2013.

[2] J.U. Kim and S.H. Jin., “Internet (IoT) Security Technology for Security Threats in Hyper-Connection Environment,” *Journal of The Korean Institute of Communication Sciences*, vol. 34, no.3, pp.57-64, Feb. 2017.

[3] NIST std. FIPS 180-4, *Secure Hash Standard(SHS)*, National Institute of Standard and Technology(NIST), Gaithersburg, M.D., March 2012.

[4] TTA std. TTAK.KO-12.0011/R1, *Hash Function Standard - Part 2 : Hash Function Algorithm Standard (HAS-160)*, Telecommunications Technology Association(TTA), Dec. 2000.

[5] X. Wang, Y.L. Yin, and H. Yu., “Finding collisions in the full SHA-1,” in *Annual International Cryptology Conference*, Springer Berlin Heidelberg, pp. 17-36, Aug. 2005.

[6] NIST Policy on Hash Functions [Internet], Available: http://csrc.nist.gov/groups/ST/hash/policy.html/.

[7] J.C. Jeon, K.J. Seo, and K.W Kim, “"Hardware complexity of SHA-1 and SHA-256 based on area and time analysis,” *The International Conference on Information Network 2012 (ICOIN)*, Bali, pp. 557-561, Feb. 2012.

[8] A. Mohamed and A. Nadjia, “SHA-2 hardware core for virtex-5 FPGA,” *2015 IEEE 12th International Multi-Conference on Systems, Signals & Devices (SSD15)*, pp. 1-5, Mahdia, Mar. 2015.

[9] M.D. Rote, V.N, D. Selvakumar, “High performance SHA-2 core using the Round Pipelined Technique,” *2015 IEEE International Conference on Electronics, Computing and Communication Technologies (CONNECT)*, Bangalore, pp. 1-6, July 2015.

[10] J.W. Cha and C.H. Kim., “Design of FPGA Hardware Accelerator for Information Security System,” *Journal of the Korea Industrial Information Systems Research*, vol. 18, no. 2, pp. 1-12, Apr. 2013.

이상현(Lee-Sang Hyub)

2017년 2월 금오공과대학교 (공학사)
2017년 3월~현재 금오공과대학교 대학원 전자공학과 석사과정 재학 중
※관심분야 : 통신 및 신호처리용 반도체 IP 설계, 정보보호용 반도체 IP 설계

신경욱(Kyung-Wook Shin)

1984년 2월 한국항공대학교 전자공학과(공학사)
1986년 2월 연세대학교대학원 전자공학과(공학석사)
1990년 8월 연세대학교대학원(공학박사)
1990년 9월~1991년 6월 한국전자통신연구소 반도체연구단(선임연구원)
1991년 7월~현재 금오공과대학교 전자공학부(교수)
1995년 8월~1996년 7월 University of Illinois at Urbana-Champaign(방문교수)
2003년 1월~2004년 1월 University of California at San Diego(방문교수)
2013년 2월~2014년 2월 Georgia Institute of Technology(방문교수)
※관심분야 : 통신 및 신호처리용 SoC 설계, 정보보호 SoC 설계, 반도체 IP 설계