16-비트 데이터 패스를 이용한 SHA-256 해시함수의 경량화 구현

이상현* · 신경욱*

*금오공과대학교

Lightweight Implementation of SHA-256 Hash Function using 16-bit Datapath

Sang-Hyun Lee* · Kyung-Wook Shin*

*Kumoh National Institute of Technology

E-mail : lpp1124@kumoh.ac.kr

요 약

본 설계에서는 임의의 길이의 메시지를 256-비트의 해시 코드로 압축하는 해시 알고리듬인 SHA-256(Secure Hash Algorithm-256) 해시함수를 경량화 구현 설계 하였다. 미국 표준 기술연구소 NIST에서 발표한 표준문서 FIPS 180-4에 정의16된 32-비트의 데이터 패스를 16-비트로 설계하여 경량화 구현하였다. Verilog HDL로 설계된 SHA-256 해시함수는 Xilinx ISim를 사용하여 시뮬레이션 검증을 하였다. CMOS 표준 셀 라이브러리로 합성한 결과 100MHz 동작주파수에서 18,192 GE로 구현되었으며, 192MHz의 최대 동작주파수를 갖는다.

키워드

SHA-256, Hash function, Lightweight, Datapath

Ⅰ. 서 론

해시함수는 임의의 길이를 갖는 메시지를 입력받아 고정된 길이의 해시값을 출력하는 함수이며, 무결성을 제공하는 목적으로 주로 사용될 뿐만 아니라 다른 보안 기법 및 프로토콜과 함께 사용되어 정보 원천에 대한 인증, 디지털 서명, 키 분배 등을 제공한다.

SHA(Secure Hash Algorithm)는 미국 표준기술연구소(NIST)에 의하여 출판되었으며, 단방향 해시함수이다. 2005년 중국의 연구팀에 의해 SHA-1에 대한 충돌쌍 공격이 발표됨[1]에 따라 NIST는 해시 함수의 모든 응용 프로그램에 대해 최소한 SHA-256을 사용하는 것을 권장하고 있다[2].

본 설계에서는 데이터 패스를 줄이므로써 하드웨어 면적이 경량화된 SHA-256 해시함수를 설계하고, Xilinx ISim을 이용하여 시뮬레이션 검증을 하였다.

Ⅱ. SHA 알고리듬

SHA는 Pre-processing, Hash computation 2가지 단계로 진행된다. Pre-processing은 메시지 패딩, 메시지 파싱, 해시 초깃값 설정으로 구성되며, Hash compuation은 라운드 연산, 다이제스트 생성으로 구성된다.

2.1 메시지 패딩

SHA-256은 2^{64}-비트 이하의 메시지를 메시지(M), “1”, “0”, 메시지 길이(l)로 패딩한다. 그림 1은 메시지 패딩 구조를 나타낸 것이며, 메시지의 끝에 “1”, 하위 64-비트에는 메시지 길이, 나머지는 k개의 “0”을 패딩한다. 식 (1)을 통해 k를 구하고 패딩 된 메시지는 512-비트의 정수배가 된다. 패딩 된 메시지는 512-비트씩 나누어 해시값을 계산하는 Hash computation으로 입력된다.

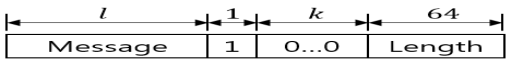


Fig. 1. Message padding

$$l + 1 + k \equiv 448 \bmod 512 \qquad (1)$$

2.2 Hash computation

Hash computation은 512-비트의 메시지 블록을 입력받아 라운드 연산을 통해 최종 해시값을 생성하고, 해시 초기값과 더하여 다이제스트를 생성한다.

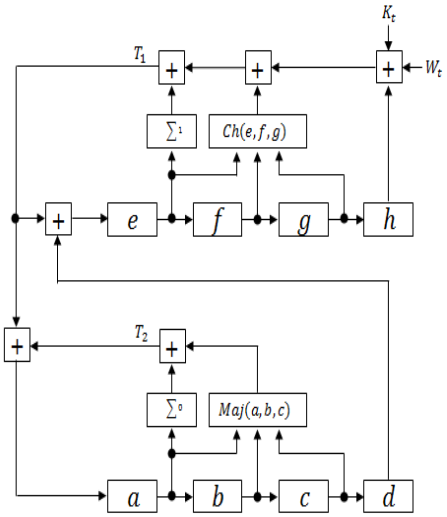


Fig. 2. Structure of SHA-256

그림 2는 SHA-256의 라운드 구조를 나타낸 것이며, 8개의 32-비트 레지스터와 연산 함수들로 구성된다. 라운드 마다 정해진 상수 K_t와 메시지 스케줄 W_t를 입력받아 라운드를 통해 해시값을 생성한다.

SHA-256은 64번의 라운드를 반복하여 최종 해시값을 생성하고, 최종 해시값과 초기 해시값을 더해 다이제스트를 생성한다. 메시지 블록이 2개 이상일 경우 다이제스트가 다음 블록의 해시 초기값이 된다.

III. SHA-256 설계

본 설계에서는 16-비트 데이터 패스를 이용하여 SHA-256을 경량화 구현하였다. 그림 3은 설계된 하드웨어의 전체 구조이며, 하드웨어를 제어하기 위한 Control 블록, 메시지를 패딩하는 Padder 블록, 해시를 생성하기 위한 Round computation으로 구성된다. Round computation은 메시지 스케줄을 하는 Word update 블록과 연산을 통해 해시값을 생성하는 Hash update 블록으로 구성되어 있으며, Digest register에 다이제스트를 저장하여 256-비트로 출력한다.

Padder 블록은 64-비트 메시지 길이 레지스터, 512-비트 패더 레지스터, 멀티플렉서로 구성된다. 메시지 길이 레지스터에 메시지 길이를 저장하고 메시지가 입력될 때마다 16씩 감소하여 그 값에 따라 메시지, "1", "0", 메시지 길이를 멀티플렉서로 제어하여 512-비트 레지스터에 입력한다. 메시지 길이 레지스터는 메시지 길이를 패딩하기 위해 유지하는 것과 멀티플렉서를 제어하는 것으로 2가지가 존재한다.

Word update 블록과 Hash update 블록은 오른쪽 회전 시프트, 오른쪽 시프트, XOR, 덧셈 연산으로 구성되어 있으며, 32-비트 데이터 패스와 동일한 값을 얻기 위해 캐리 레지스터와 멀티플렉서가 추가된다. 패딩 된 메시지가 입력되고 1라운드에 2 클록 씩 소비하여 128 클록 동안 연산을 한다.

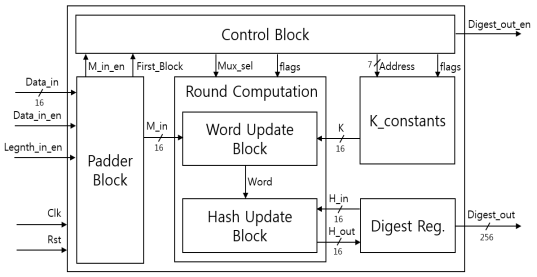


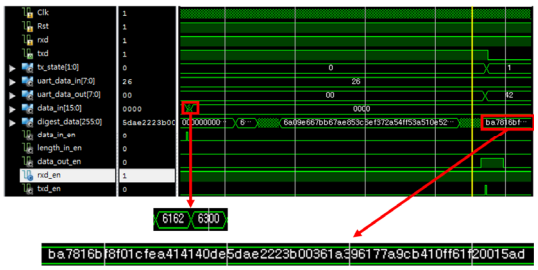
Fig. 3. Architecture of SHA-256 using 16-bit datapath

Digest register는 Round computation에서의 연산 동안 초기 해시 값을 유지하고 최종 해시값이 생성되면 16 클록 동안 해시 초기값과 더하여 다이제스트를 생성한다.

IV. 기능검증

SHA-256 해시 프로세서는 Verilog HDL을 이용하여 설계되었고, Xilinx ISim을 사용한 시뮬레이션으로 기능검증 하였다.

그림 4는 시뮬레이션 검증한 결과값과 테스트 벡터값을 보여준다. 테스트 벡터는 NIST FIPS 180-2[3]에 정의된 값을 사용하였다. 그림 4-(a)에서 24-비트의 16진수 "616263"가 입력되었을 때 256-비트의 16진수 "ba7816bf 8f01cfea 414140de 5dae2223 b00361a3 96177a9c b410ff61 f20015ad"가 출력되는 것을 확인하였다. 이는 그림 4-(b)의 테스트 벡터값과 정확히 일치함을 확인할 수 있다.



(a) Simulation result of SHA-256 processor

Let the message, M, be the 24-bit ($\ell = 24$) ASCII string "abc", which is equivalent to the following binary string:

01100001 01100010 01100011.

The resulting 256-bit message digest is

ba7816bf 8f01cfea 414140de 5dae2223 b00361a3 96177a9c b410ff61 f20015ad.

(b) Reference data[3]

Fig. 4. Simulation verification of SHA-256 processor

V. 결 론

미국 표준 기술연구소 NIST에서 발표한 표준 문서 FIPS 180-2[3]에 정의된 SHA-256의 데이터 패스 32-비트를 16-비트로 줄여 경량화 하였다. 연산이 되는 블록에서 기존과 동일한 연산을 얻기 위해 캐리 레지스터와 멀티플렉서가 추가되어 게이트 수가 증가하지만 레지스터의 크기가 줄어 감소되는 게이트 수가 더 많기 때문에 경량화 설계되었다. CMOS 표준 셀 라이브러리로 합성한 결과 100MHz 동작주파수에서 18,192 GE로 구현되었으며, 192MHz의 최대 동작주파수를 갖는다.

ACKNOWLEDGMENTS

· This work was supported by the Industrial Core Technology Development Program (1004 9009, Development of Main IPs for IoT and Image-Based Security Low-Power SoC) funded by the Ministry of Trade, Industry & Energy.
· The authors are thankful to IDEC for EDA software support.

참고문헌

[1] WANG, Xiaoyun; YIN, Yiqun Lisa; YU, Hongbo. Finding collisions in the full SHA-1. In: Annual International Cryptology Conference. Springer Berlin Heidelberg, 2005. p. 17-36.
[2] NIST Hash Policy : http://csrc.nist.gov/groups/ST/hash/policy.html.
[3] STANDARD, Secure Hash. FIPS PUB 180-2. National Institute of Standards and Technology, 2002.