HAS-160 해쉬 프로세서의 VLSI 설계

현 주 대, 최 병 윤
동의대학교 컴퓨터공학과

VLSI Design of HAS-160 Algorithm

Ju-Dai HYUN, Byeong-Yoon CHIO.
Dept. of Computer Engineering, Dong-Eui University
E-mail : highfreq@dongeui.ac.kr

요 약

본 논문에서는 한국형 디지털 서명 표준인 KCDSA에서 사용할 목적으로 개발된 국내 해쉬 함수 표준인 HAS-160 알고리즘을 VLSI 설계하였다. 하나의 단계연산을 하나의 클럭에 동작하고 단계연산의 핵심이 되는 4개의 직렬 2^{32} 모듈러 가산기를 CSA(Carry Save Adder)로 구현하여 캐리 전파시간을 최소로 하고 HAS-160 해쉬 알고리즘의 특징인 메시지 추가생성을 사전에 계산하여 지연시간을 줄이는 설계를 하였다. 설계된 해쉬 프로세서를 0.25 um CMOS 스탠다드 셀 라이브러리에서 합성한 결과 총 게이트 수는 약 21,000개이고 최대 지연 시간은 5.71 ns로 최대 동작 주파수 약 175 MHz서 약 1,093 Mbps의 성능을 얻을 수 있었다.

1. 서론

현재 사회는 통신 기술의 발달로 인한 정보화 시대를 맞이하여 편리함을 누리고 있으나 그 이면에는 정보의 유출과 같은 보안 사고가 뒤따르고 있다.

메시지 내용 공개나 트래픽 분석과 같은 수동적 공격은 DES나 AES 같은 대칭키 암호 알고리즘, 그리고 RSA와 같은 비대칭키 암호 알고리즘으로 대처할 수 있다. 하지만 위장, 재전송 그리고 메시지 불법 수정과 같은 능동적 공격에 대해서는 단순히 메시지의 암호화만으로는 대처할 수가 없으므로 이런 능동적 공격에 대처하기 위한 방법으로 해쉬 함수를 이용한 메시지 인증과 무결성이 요구된다[1]. 무결성이란 송신자가 보내는 메시지의 내용을 수신자가 메시지의 내용이 공격자에 의해 변조되지 않았음을 확인 할 수 있는 것이며 인증이란 수신자가 수신된 메시지가 정당한 송신자로부터 송신되었음의 확인이다. 해쉬 알고리즘은 데이터 무결성 및 메시지 인증 등에서 사용할 수 있는 함수로써 정보보호의 여러 메커니즘에서 이용되는 핵심 요소기술이다.

해쉬 알고리즘이란 임의의 길이의 비트열을 고정된 길이의 출력 값인 해쉬코드로 압축시키는 함수이며, 암호학적 응용에 사용되는 대부분의 해쉬함수는 강한 충돌저항성을 지닐 것이 요구된다. 암호학적 해쉬 알고리즘의 충돌 저항성은 디지털 서명에서 송신자 외의 제 3자에 의한 문서위조를 방지하는 부인방지 서비스를 제공하기 위한 필수적인 요구 조건이 된다. 해쉬 알고리즘은 크게 DES와 같은 블록암호 알고리즘에 기초한 해쉬 알고리즘과 전용 해쉬 알고리즘으로 나눌 수 있다. 블록암호를 이용한 해쉬 알고리즘은 이미 구현되어 사용되고 있는 블록암호를 사용할 수 있다는 이점이 있으나, 대부분의 블록암호 알고리즘의 속도가 그리 빠르지 않을뿐더러 이를 기본함수로 이용한 해쉬 알고리즘의 경우 블록암호보다도 훨씬 더 속도가 떨어지므로 현재는 대부분의 응용에서 전용 해쉬 알고리즘이 주로 이용된다[2]. 전용 해쉬 알고리즘으로는 RIPEMD-160, SHA-1, MD5, HAS-160등이 있다[1][3][4].

본 논문에서는 한국형 디지털 서명 표준인 KCDSA에서 사용할 목적으로 개발된 국내 해쉬 함수 표준인 HAS-160 알고리즘을 VLSI 설계하였다.

본 논문의 구성은 2장에서 HAS-160 해쉬 알고리즘을 소개하고, 3 장에서 HAS-160 해쉬 프로세서의 VLSI 설계를 기술하였고, 4 장에서 성능을 분석하

며, 5 장에서 결론을 제시하였다.

2. 해쉬함수 알고리즘 표준(HAS-160)

해쉬 알고리즘은 임의의 길이의 비트열을 고정된 길이의 출력 값인 해쉬 코드로 압축시키는 알고리즘으로써 같은 출력을 내는 임의의 서로 다른 두 입력 메시지를 찾는 것이 계산상 불가능한 충돌 저항성을 가질 것이 요구된다.

HAS-160은 한국형 디지털 서명 표준인 KCDSA에서 사용할 목적으로 개발되었으며, 1998년 10월의 국내 단체표준화 (TTAS.KO-12.0011)를 거쳐 2000년 12월에 개정되었다. HAS-160은 메시지를 512비트 블록단위로 처리하여 160비트의 해쉬코드를 출력하는 Little endian 구조의 32비트 마이크로 프로세서를 기본으로 설계된 충돌저항성의 해쉬 함수이다. HAS-160 표준 해쉬 알고리즘의 내부 연산은 32비트 정수간 연산이 사용된다. 따라서 비트 열 - 워드열간의 변환규칙이 필요하다.

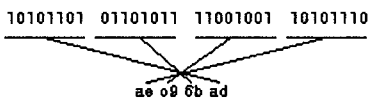

그림 1 비트열 - 워드열 변환

HAS-160 알고리즘은 임의의 길이를 가지는 입력 메시지를 512비트 블록 단위로 처리하여 160비트의 출력을 가진다. 512비트 단위 블록은 4 라운드, 80 단계로 처리하여 압축 함수를 생성한다. 각 단계연산에서 사용되는 연쇄변수는 5개이다. 각 라운드에 적용될 메시지는 512비트 입력블록으로부터 생성된 16 워드와 이로부터 추가로 생성되는 4개의 워드를 포함하여 20개가된다. 임의의 길이의 메시지 M이 입력으로 들어오면 이 M을 덧붙이기 과정을 통해 512의 배수로 만든 후, 512비트 블록단위로 각각 연속해서 단계연산에서 적용되며 각 단계에서 연쇄변수를 갱신하여 해쉬 코드를 생성한다. 각 라운드마다 적용되는 상수는 다음과 같다.

$$
\begin{aligned}
K_j &= 0000000 \ (0 \le 19) \\
K_j &= 5a827999 \ (20 \le 39) \\
K_j &= 6ed9eba1 \ (40 \le 59) \\
K_j &= 8f1bbcdc \ (60 \le 79)
\end{aligned} \qquad (1)
$$

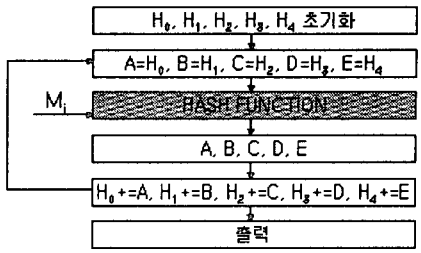

그림 2 HAS-160 해쉬 알고리즘

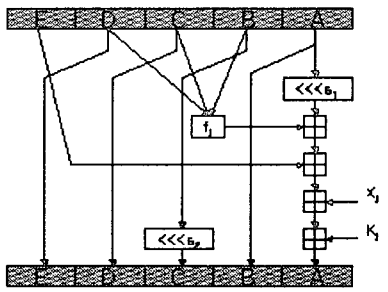

그림 3 HASH FUNCTION의 단계연산

각 단계에 사용되는 순환이동의 양(S)의 순서는 다음과 같이 주어진다.

$$
\begin{aligned}
S_1 &= 10,11,7,15,6,13,8,14,7,12,9,11,8,15,6,12 \\
&\quad 9,14,5,13 \ (\text{각라운드}) \\
S_2(j) &= 10 \ (0 \le j \le 19) \\
S_2(j) &= 17 \ (20 \le j \le 39) \\
S_2(j) &= 25 \ (40 \le j \le 59) \\
S_2(j) &= 30 \ (60 \le j \le 79)
\end{aligned} \qquad (2)
$$

각 단계에 적용되는 메시지 워드의 순서(K)는 다음과 같이 주어진다.

$$
\begin{aligned}
X(j) &= 18,0,1,2,3,19,4,5,6,7,16,8,9, \\
&\quad 10,11,17,12,13,14,15 \ (0 \le j \le 19) \\
X(j) &= 18,3,6,9,12,19,15,2,5,8,16,11,14, \\
&\quad 1,4,17,7,10,13,0 \ (20 \le j \le 39) \\
X(j) &= 18,12,5,14,7,19,0,9,2,11,16,4,13, \\
&\quad 6,15,17,8,1,10,3 \ (40 \le j \le 59)
\end{aligned}
$$

$$X(j) = 18,7,2,13,8,19,3,14,9,4,16,15,10,$$
$$5,0,17,11,6,1,12 \ (60 \le j \le 79) \quad (3)$$

각 단계에서는 3개의 불 함수가 사용되며 각각의 라운드에서 사용되는 함수는 다음과 같다.

$$f_j(x,y,z) = x \cdot y + \overline{z} \quad round\ 1$$
$$f_j(x,y,z) = x \oplus y \oplus z \quad round\ 2,4$$
$$f_j(x,y,z) = y \oplus (x + \overline{z}) \quad round\ 3 \quad (4)$$

하나의 512비트 메시지 블록에 대해 4라운드, 80단계의 압축함수 연산이 끝나면 그 결과 A,B,C,D,E를 연쇄변수 H_0,H_1,H_2,H_3,H_4에 더하여 H_0,H_1,H_2,H_3,H_4를 갱신한다.

$$H_0 += A, \quad H_1 += B, \quad H_2 += C,$$
$$H_3 += D, \quad H_4 += E \quad (5)$$

모든 512비트 메시지 블록을 처리한 후의 연쇄변수 H_0,H_1,H_2,H_3,H_4가 각각 $H_i = h_{i3}h_{i2}h_{i1}h_{i0}$과 같이 주어질 때 워드열 - 비트열 변환규칙에 의해 문자열로 변환된 후 다음과 같이 해쉬 코드로 변환된다.

$$H_i = h_{i0}h_{i1}h_{i2}h_{i3} \quad (6)$$

3. HAS-160 알고리즘 프로세서 설계

3.1 HAS-160 프로세서 설계 사양

본 논문의 HAS-160 프로세서는 호스트프로세서의 보조 프로세서로 설계되었으며 외부 인터페이스는 32 비트 인터페이스를 지원한다. 입력 메시지의 사전처리는 호스트프로세서가 담당하도록 하여 HAS-160 프로세서는 512 비트 입력에 160 비트의 출력을 가지도록 한다.

3.2 전체구조

해쉬 프로세서의 전체구조는 그림 4 와 같이 컨트롤 유닛, 512 비트 메시지를 입력받아 4개의 워드를 추가 생성하는 메시지 제너레이터 부분, HAS-160의 각 단계연산을 수행하는 HAS-160 STEP 부분, 그리고 입력된 메시지와 해쉬 코드의 결과 값 H 레지스터, 각 단계에서 갱신되는 연쇄변수 A,B,C,D,E 레지스터로 구성되어 있다.

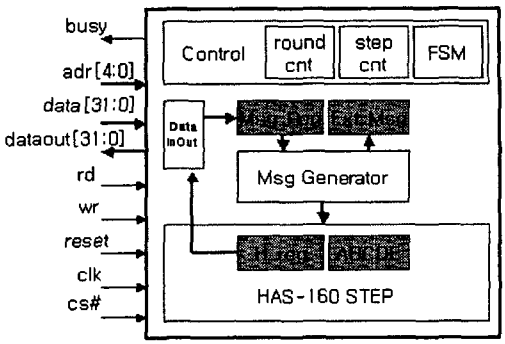

그림 4 HAS-160 해쉬 프로세서 전체구조

3.3 단계 연산

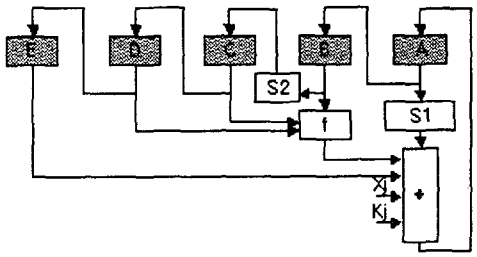

그림 5 단계연산

HAS-160의 단계 연산은 그림 3 과 같이 2개의 순환이동, 1 개의 불 함수 식, 그리고 4개의 2^{32} 모듈러 덧셈으로 구성된다. 각 라운드 별로 다른 불식이 사용되는 f 블록과 각 라운드별 일정한 순환 이동을 하는 S1, 각 단계마다 일정한 순환 이동을 하는 S2의 순환 이동의 양은 각각 제어 회로의 round_counter와 step_counter의 출력으로 선택적으로 순환이동을 할 수 있도록 설계하였고 단계연산의 핵심이 되는 4개의 2^{32} 모듈러 덧셈은 연산수행의 성능을 높이기 위해 CLA(Carry Look-ahead Adder)와 CSA(Carry Save Adder)로 설계하였다.

3.4 Adder

SHA-160 해쉬 알고리즘의 단계 연산중 가장 연산시간이 많이 소요되는 부분은 4개의 가산기 모듈이다 이 가산기 모듈을 구현하는 방법은 여러 가지가

있다. 먼저 하나의 32 비트 캐리 전파 가산기를 두고 4번을 반복하는 방법이 있다. 이 방법은 최소의 하드웨어를 이용하는 효율적인 구조이나 32 비트 캐리를 전파하는데 소요되는 시간이 매우 많이 소모되므로 수행 성능 면에서는 바람직하지 않은 구조이다. 본 논문에서는 캐리 전파 시간을 최소로 하여 성능을 높이는 CSA(Carry Save Adder)로 다음과 같이 구현하였다.

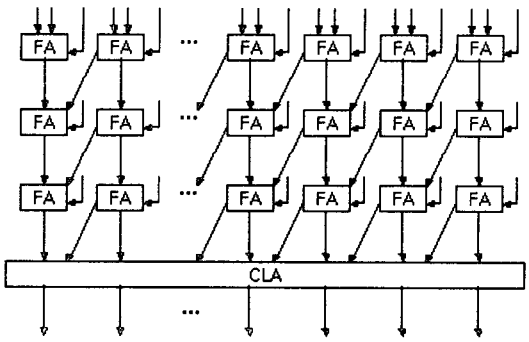

그림 6 Carry Save Adder 구조

3.4 메지지 입력

주어지는 16개의 32비트 메시지워드에서 각 라운드마다 서로 다른 4개의 메시지 워드를 추가로 생성하는 방법은 수식 (7) 과 같이 XOR 하여 생성한다.

$$
\begin{aligned}
X_i[16] &= X_i[I(j+1)] \oplus X_i[I(j+2)] \\
&\quad \oplus X_i[I(j+3)] \oplus X_i[I(j+4)] \\
X_i[17] &= X_i[I(j+6)] \oplus X_i[I(j+7)] \\
&\quad \oplus X_i[I(j+8)] \oplus X_i[I(j+9)] \\
X_i[18] &= X_i[I(j+11)] \oplus X_i[I(j+12)] \\
&\quad \oplus X_i[I(j+13)] \oplus X_i[I(j+14)] \\
X_i[19] &= X_i[I(j+16)] \oplus X_i[I(j+17)] \\
&\quad \oplus X_i[I(j+18)] \oplus X_i[I(j+19)] \qquad (7)
\end{aligned}
$$

3.5 제어 회로

제어회로의 구조는 그림 4 와 같이 각 라운드와 단계를 지시하는 카운터와 입출력 제어부, 그리고 FSM(Finite State Machine)으로 구성된다. 입출력 제어부는 데이터 입출력을 제어하고 시작신호를 감지해서 FSM을 구동한다. FSM의 상태는 start 신호를 감지하기 전 state, 메시지워드를 생성하는 state, 단계연산을 수행하는 state 3가지 state로 구분된다.

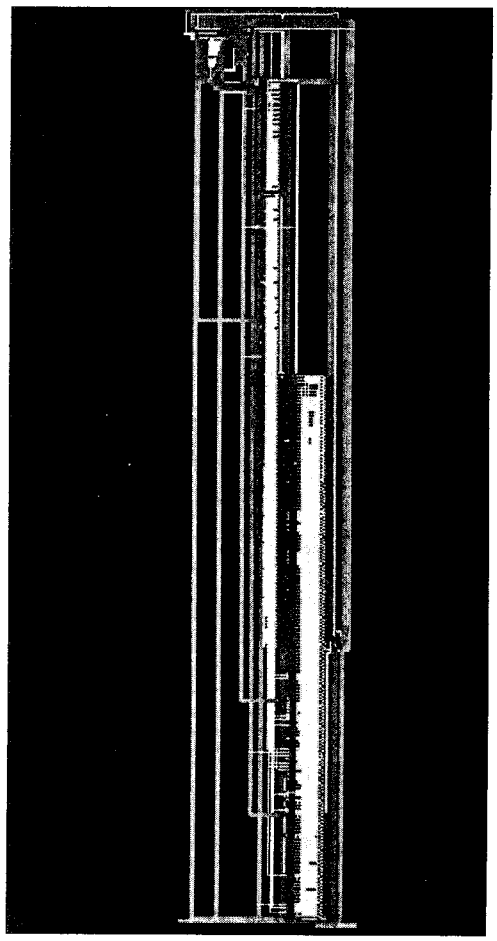

그림 7 SHA-160 해쉬 프로세서 합성결과

수식 (1), (2), (3), (4), (7)과 같이 HAS-160 해쉬 알고리즘은 각 라운드 혹은 각 단계별 각각의 동작을 하므로 라운드 카운터와 스텝 카운터를 두고 이 결과를 디코딩해서 제어 신호를 생성한다. 1 clock에 한 단계의 연산을 수행하므로 4라운드 각각 20 단계, 총 80단계 연산을 수행하는데 80 clock을 사용하고, 첫 번째 단계 연산에서 메시지워드 X[18]을 사용한다. X[18]은 입력되는 메시지워드 X[0~15]에서 추가로 생성하는 워드이므로 최초 단계에서 사용하기 위해서 단계 연산 이전에 추가의 메시지워드를 생성하는 1 clock을 할당하고 모든 단계연산 종료 후 초기값 H레지스터와 더해져서 최종 해쉬코드를 생성한다. 따라서 512비트 메시지로부터 160비트 해쉬코드를 생성하는데 소요되는 총 clock cycle 수는 82 clock이다.

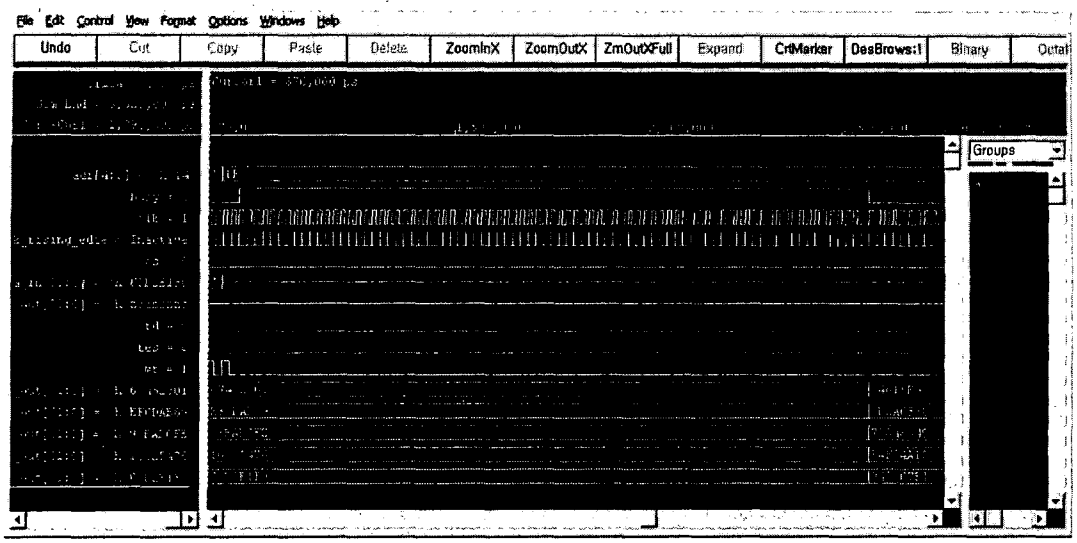

그림 8 시뮬레이션 결과

4. 성능 분석

본 논문에서 설계한 해쉬 프로세서는 Verilog - HDL을 이용하여 설계하였고 Verilog - XL[5] 시뮬레이터로 시뮬레이션한 결과와 한국 정보 통신 기술 협회(TTA)의 표준안과 비교 일치하는가를 확인하였다. 설계된 해쉬 프로세서는 0.25 um CMOS 라이브러리를 사용하여 Synopsys의 Design_Analyzer 소프트웨어를 이용하여 합성하였다. 합성 결과 본 논문에서 설계한 해쉬 프로세서의 총 게이트 수는 약 21,000이고, 최대 지연 시간은 5.71 ns이므로 최대 동작 주파수는 약 175 MHz이다. 따라서 약 1,093Mbps의 성능을 가진다.

지원 알고리즘	HAS-160
단계 당 clock 수	1
전체 clock 수	82
최대 동작 주파수	175 MHz
게이트 수	약 21,000
외부 인터페이스	32 bit
성능	1093 Mbps
사용 공정	0.25 um

표 1 성능 분석

5. 결론

본 논문에서는 고속의 해쉬 프로세서를 설계하였다. 한 단계를 한 클럭으로 처리하고 단계연산은 CSA를 사용하여 고속으로 처리하는 구조로 설계하였다. 면적보다는 속도를 우선한 설계로 게이트 수가 많으나 다른 해쉬 알고리즘을 추가로 구현할 시 메시지, 내부변수, 해쉬코드 래지스터와 가산기등 규모가 큰 모듈들을 공유하여 효율적으로 사용할 수가 있으므로 추가로 소요되는 게이트는 크지 않다. 본 논문에서 설계한 해쉬 프로세서는 1,093Mbps의 고성능이므로 고속을 요구하는 네트웍, 전자상거래 등에서 사용할 수 있을 것으로 판단된다.

[참고문헌]

[1] William Staillings, Cryptography and Network Security, Principles and Practice, 1999.

[2] 한국 정보 통신 기술 협회, "해쉬 함수 표준 - 제 2 부 : 해쉬 함수 알고리즘 표준 (HAS-160), TTAS.KO-12.0011/R1, 2000.12.

[3] Douglas R. Stinson, CRYPTOGRAPHY Theory and Practice, CRC press

[4] 김철, 암호학의 이해, 영풍문고, 1996.10.

[5] CADENCE, Verilog - XL Reference Manual Volume1-2, 1991.3.