MD5 및 HAS-160 해쉬 알고리즘을 통합한 면적 효율적인 설계에 관한 연구

손승일*

A Study on Area-Efficient Design of Unified MD5 and HAS-160 Hash Algorithms

Seungil, Sonh*

이 논문은 한신대학교 학술 연구비 지원에 의하여 연구되었음

요 약

본 논문에서는 MD5 및 HAS-160 해쉬 알고리즘을 통합한 하드웨어 설계에 대해 다룬다. MD5와 HAS-160 해쉬 알고리즘은 임의의 길이를 갖는 메시지를 512비트의 메시지 블록 단위로 처리하여 고정된 길이의 해쉬 코드를 출력한다. MD5의 해쉬 코드는 128비트이며, HAS-160 해쉬 코드는 160비트이다. 설계된 통합 해쉬 코어는 HAS-160 코어와 비교하여 32%의 슬라이스를 추가적으로 사용하지만, 고정된 메시지 버퍼 공간만을 사용한다. 단계당 1클럭에 수행되는 통합 해쉬 코어는 92MHz에서 동작하며, MD5 모드에서는 724Mbps로 HAS-160 모드에서는 581Mbps의 속도로 메시지를 다이제스트(Digest)하는 성능을 갖는다. 본 논문의 통합 해쉬 코어는 전자상거래, 데이터 무결성, 디지털 서명 등의 분야에서 응용이 가능할 것으로 사료된다.

ABSTRACT

This paper deals with hardware design which unifies MD5 and HAS-160 hash algorithms. Two algorithms get a message with arbitrary length and process message blocks divided into 512 bits each time and output a hash code with a fixed length. MD5 ouputs a hash code of 128 bits and HAS-160 a hash code of 160 bits. The unified hash core designed has 32% of slices overhead compared to HAS-160 core. However, there is only a fixed message buffer space used. The unified hash core which run a step in one clock cycle operates at 92MHz and has performance which digests a message in the speed of 724Mbps at MD5 and 581Mbps at HAS-160 hash mode. The unified hash core which is designed can be applicable to the areas such as E-commerce, data integrity and digital signature.

키워드

해쉬알고리즘, MD5, HAS-160, 데이터 무결성, 시큐리티

Key word

Hash algorithm, MD5, HAS-160, Data integrity, Security

* 종신회원 : 한신대학교 (교신저자 : saisonh@hs.ac.kr)

접수일자 : 2011. 10. 07
심사완료일자 : 2011. 11. 08

Open Access http://dx.doi.org/10.6109/jkiice.2012.16.5.1015

Ⅰ. 서 론

오늘날 정보 기술의 급속한 발전과 인터넷의 전 세계적인 보급 그리고 무선 기술에 기초한 유비쿼터스 컴퓨팅 기술의 확산으로 인해 정보 보호에 대한 관심은 지속적으로 증가하고 있다. 외부 공격자에 의한 보안 공격에 대해 안전하게 데이터를 보호할 수 있는 기술력의 확보를 위해 보안 관련 분야에 대한 투자가 증가하고 있으며 보안을 강화한 새로운 알고리즘이 지속적으로 발표되고 있다.

정보의 공유가 보편화된 21세기의 정보화 사회에서 정보의 해킹 혹은 크래킹의 정보화 역기능에 대처하기 위해 무결성(Confidentiality), 무결성(Integrity), 인증(Authentication), 부인 봉쇄(Non-repudiation) 등의 기술이 사용되고 있다[1,2].

입력 자료의 길이에 관계없이 고정된 길이의 값으로 전환시켜주는 해쉬 함수의 성질이 암호학 분야에서 암호학적 해쉬 함수라는 이름으로 인증 및 디지털 서명분야에서 응용되고 있다[3]. 해쉬 함수를 응용한 활용 분야는 전자 상거래, 데이터 무결성, 디지털 서명, 대규모 암호화, 고속 네트워킹 장비 및 무선 응용 분야에 걸쳐 다양한 것으로 알려져 있다[4].

해쉬 알고리즘은 임의의 길이의 비트 열을 고정된 길이의 출력 값인 해쉬 코드로 압축시키는 알고리즘으로써 다음의 성질을 만족해야 한다[5]. 첫째는 주어진 출력에 대하여 입력 값을 구하는 것이 불가능해야 한다. 둘째는 주어진 입력에 대하여 같은 출력을 내는 또 다른 입력을 찾아내는 것이 계산상 불가능해야 한다. 뿐만 아니라 암호학적 응용에 사용되는 대부분의 해쉬 알고리즘은 위의 두 가지 성질뿐만 아니라 이보다 강한 충돌 저항성(Collision-resistance)을 갖는 것이 요구된다. 즉 같은 출력을 내는 임의의 서로 다른 두 입력 메시지를 찾는 것이 계산상 불가능해야 한다. 이러한 암호학적 해쉬 알고리즘의 충돌 저항성은 전자 서명에서 송신자외의 제 3자에 의한 문서위조를 방지하는 부인 방지 서비스를 제공하기 위한 필수적인 조건이 되기 때문이다.

논문 [1]에서는 하드웨어 공유와 캐리 보존 덧셈을 이용한 MD5 해쉬 프로세서의 설계에 대해 연구하였으며, 참고문헌 [4]은 MD5 해쉬 함수 코어 IP에 대해 설명하고 있으며, 논문 [6]는 해쉬 함수를 반복적으로 수행하는 방안과 완전하게 펼친(unrolling) 기법으로 구현하는 방안에 대해서 논하고 있다. 그리고, 논문 [2]에서는 한국형 해쉬 함수 표준인 HAS-160 알고리즘의 구현 및 검증에 대해 다루고 있다. 또한 논문 [7]에서는 MD5와 SHA-1을 통합한 해쉬 알고리즘의 구현에 대해 설명하고 있다.

본 논문에서는 MD5 해쉬 알고리즘과 한국형 해쉬 표준 함수인 HAS-160 해쉬 알고리즘을 통합한 면적 효율적인 구현 방안에 대해 연구하였다. 2장에서는 MD5와 HAS-160 해쉬 함수의 특징에 대해서 설명하고, 3장에서는 각 해쉬 알고리즘에 사용하는 주요 연산 및 아키텍쳐에 대해 설명하며, 4장에는 통합 해쉬 알고리즘의 설계에 대해 설명할 것이다. 5장에서는 설계된 통합 해쉬 알고리즘의 성능 분석 및 고찰을 수행하며, 마지막으로 6장에서는 결론을 제시한다.

Ⅱ. MD5 및 HAS-160 해쉬 알고리즘의 특징

본 장에서는 일반적인 해쉬 함수의 개요 및 MD5 및 HAS-160 해쉬 알고리즘의 유사성 및 차이점에 대해 논할 것이다.

먼저 가변인 입력 메시지(m)의 길이가 총 K 비트가 주어지면 해쉬 함수를 적용하기 전에 해쉬 함수에 입력되는 전체 메시지의 길이가 512비트의 배수가 되도록 전처리를 수행해야 한다. 이 과정은 MD5와 HAS-160 해쉬 알고리즘에 공통적으로 적용되는 사항이다. 512비트로 분할된 마지막 메시지 M_{N-1}의 길이는 모듈로 512에서 448비트가 되도록 1000...000의 패딩 정보를 부착해야 하며, 마지막 64비트에는 원래 메시지의 총 비트 길이를 64비트로 표시하여 붙인다. 메시지의 길이 K 비트를 표시하는 방법은 마지막 64비트를 32비트씩 나누었을 때, 앞의 32비트에 먼저 총 비트 길이의 LSB를 저장하고, 메시지의 길이가 2^{32}을 넘을 경우에는 다음 번 32비트에 메시지의 MSB를 저장하게 된다. 이렇게 표기하는 이유는 하위 주소 위치에 LSB를 저장하고 상위 주소 위치에 MSB를 저장하는 인텔 방식의 리틀-엔디안(Little-endian) 데이터 표기 방식을 MD5와 HAS-160 해쉬 알고리즘이 채

용하고 있기 때문이다. 그림 1은 메시지 다이제스트(Digest : 요약)의 생성과정을 보여주고 있다.

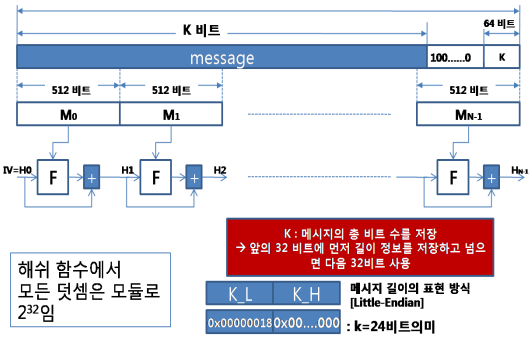


그림 1. 메시지 요약의 생성 과정
Fig. 1 Generation process of message digest

표1은 MD5와 HAS-160 해쉬 알고리즘의 특징을 요약한 것이다.

표 1. MD5와 HAS-160 해쉬 알고리즘의 특징
Table. 1 Characteristics of MD5 and HAS-160 hash algorithms

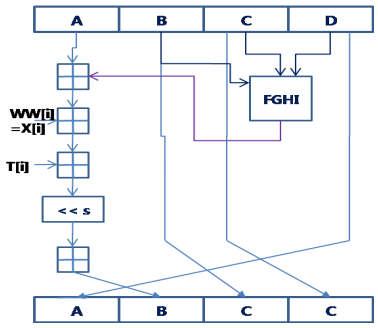
구분	MD5	HAS-160
라운드	4	4
라운드 당 단계	16 (총 64회 수행)	20 (총 80회 수행)
입력	512 비트	512 비트
출력	128 비트	160 비트
초기값	32비트 4개 (A, B, C, D)	MD5 + 32비트 1개 (A, B, C, D, E)
연산에 사용하는 입력	입력된 32비트 16개	입력된 32비트 16개 + 32비트 16개 추가 생성

MD5와 HAS-160 해쉬 알고리즘은 모두 4라운드로 구성되지만, 각 라운드의 수행 단계는 서로 다르다. MD5는 각 라운드는 16 단계로 이루어져 있지만, HAS-160은 20 단계로 이루어진다. 전에 언급한 것처럼 본 논문의 두 해쉬 알고리즘은 512비트 단위로 입력을 받아서 처리하지만, 최종 출력의 경우 MD5는 128비트를, HAS-160은 160비트를 출력하게 된다. 또한 초기 값의 경우에 출력 비트와 동일한 비트 수가 초기 값으로 사용될 수 있기 때문에 MD5는 128비트이지만, HAS-160은 160비트가 사용된다. 그런데 여기서 한 가지 특징은 32비트 4개의 기본적으로 제공되는 초기화 상수 값은 두 해쉬 함수에서 동일한 값을 사용한다. HAS-160의 경우에만 추가적인 32비트의 초기 값을 제공하고 있다.

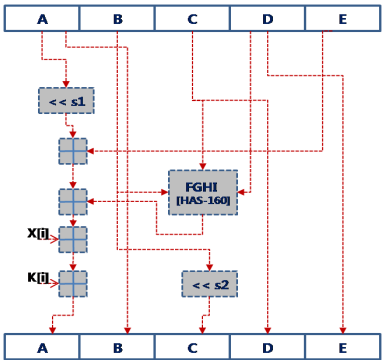
512비트는 16개의 32비트로 표현된다. 그리고 입력으로 제공되는 16개의 32비트 값이 해쉬 함수에서 액세스된다. 이 때 MD5 해쉬 함수는 입력된 16개 중에서 64단계를 수행하는 동안 랜덤하게 액세스된다. 그러나 HAS-160 해쉬 함수의 경우에는 각 라운드마다 추가적으로 4개의 입력 값을 생성하여 사용한다. 본 논문에서는 512비트 단위의 입력을 수신할 때 추가적으로 사용될 총 16개(4라운드 x 4개=16개)의 입력을 미리 생성하여 이중 포트 메모리에 저장하여 라운드 함수 수행시 제공하도록 함으로써 라운드 함수 구현의 복잡도를 줄였다.

Ⅲ. 두 해쉬 알고리즘의 주요 연산 및 아키텍쳐

그림 2는 MD5와 HAS-160 해쉬 알고리즘의 단일 단계 동작의 구성을 보여주고 있다. 여기서 X[i]는 입력된 16개의 32비트중에서 단계 연산에 제공되는 값을 의미하며, T[i]나 K[i]는 단계 연산에서 사용되는 상수 값을 의미한다. "<<amount" 연산은 amount 값만큼 좌측 회전한 값을 의미한다. 그리고 FGHI 함수 블록에서는 라운드 별로 서로 다른 논리 연산을 수행하게 된다. 아울러 내부의 모든 덧셈은 모듈로 2^{32} 연산을 수행한다.



(a)



(b)

그림 2. MD5와 HAS-160 해쉬 알고리즘의 단일 단계 동작 (a) MD5의 단일 단계 동작 (b) HAS-160의 단일 단계 동작
Fig. 2 Operations in a single step of MD5 and HAS-160 hash algorithm (a) Operations in a single step of MD5 (b) Operations in a single step of HAS-160

그림 2에 나타난 단계 연산을 정리하여 표현하면 표 2와 같이 나타낼 수 있다.

표 2. MD5와 HAS-160 알고리즘의 단계 연산 요약
Table. 2 Summary of step operation for MD5 and HAS-160 hash algorithms

구분	단계 연산
MD5	A <- D B <- B + ((A + FGHI(B, C, D) + X + T) << S) C <- B D <- C
HAS-160	A <- (A<<S1) + FGHI(B, C, D) + E + X + K B <- A C <- (B<<S2) D <- C E <- D

표 2에서 보면 알 수 있듯이 MD5와 HAS-160 해수 알고리즘은 FGHI()함수를 보유하고 있다. FGHI()함수는 라운드마다 서로 다른 함수를 사용하지만, MD5와 HAS-160 해쉬 알고리즘이 유사한 부울 함수를 사용함을 알 수 있다. 통합 해쉬 알고리즘에서는 8개의 부울 함수가 필요하지 않고, 공통된 부울 함수를 공유하면 5개의 부울 함수를 제공하면 된다.

표 3은 MD5와 HAS-160 해쉬 알고리즘의 FGHI() 부울 함수와 각 라운드에서 사용하는 상수 값에 대해 요약한 것이다. 상수 값의 경우에는 MD5의 경우에는 각 스텝마다 서로 다른 상수 값을 사용하기 때문 64개가 필요하며, HAS-160의 경우에는 라운드 당 같은 값을 사용하기 때문에 4개의 상수 값만을 필요로 함을 알 수 있다.

표 3. FGHI()부울 함수와 상수 값
Table. 3 FGHI() boolean function and Constants

MD5 해쉬 알고리즘			
라운드	단계	FGHI[j]	상수[T]
1	0~15	(B & C) | (~B & D)	단계별로 다른 값 사용
2	16~31	(B & D) | (C & ~D)	
3	32~47	(B ^ C ^ D)	
4	48~63	C ^ (B | ~D)	
HAS-160 해수 알고리즘			
라운드	단계	FGHI[j]	상수[K]
1	0~19	(B & C) | (~B & D)	00000000
2	20~39	(B ^ C ^ D)	5A827999
3	40~59	C ^ (B | ~D)	6ED9EBA1
4	60~79	(B ^ C ^ D)	8F1BBCDC

다음은 두 해쉬 알고리즘의 단일 단계 동작에서 사용되는 좌측 회전량을 정리한 것이다. 표 4에서 보는 바와 같이 MD5 해쉬 알고리즘에서 사용하는 좌측 회전량은 각 라운드에서 4개의 값을 사용하며 이는 반복적으로 사용된다. 그리고 HAS-160 해쉬 알고리즘에서는 S1의 경우에는 20개의 값이 4회 반복적으로 사용되고, S2의 경우에는 각 라운드마다 고정된 값을 사용한다.

표 4. 좌측 회전량 요약
Table. 4 Summary of left rotation amount

구분	명칭	단계	회전량	비고
MD5	S	0~15	7, 12, 17, 22	4회반복
		16~31	5, 9, 14, 20	“
		32~47	4, 11, 16, 23	“
		48~63	6, 10, 15, 21	“
HAS-160	S1	0~79	5, 11, 7, 15, 6, 13, 8, 14, 7, 12, 9, 11, 8, 15, 6, 12, 9, 14, 5, 13	4회반복
	S2	0~19	10	동일값
		20~39	17	“
		40~59	25	“
		60~79	30	“

마지막으로 각 알고리즘은 512비트 단위로 입력된 메시지에 대해 32비트씩 들어온 순서대로 X[0]부터 X[15] 주소를 할당받는다. MD5의 경우에는 32비트 16개의 데이터 X[i]에 대해 단계별로 랜덤 액세스가 이루어지지만, HAS-160의 경우에는 각 라운드마다 추가적으로 4개의 데이터가 생성되어진다. 결과적으로 4개의 라운드가 존재하므로 HAS-160의 경우에는 총 16개의 데이터가 추가적으로 생성하여 버퍼에 저장해 놓아야 한다. 물론 수행 과정에서 생성할 수도 있지만 제어의 편의 및 다음 메시지 블록 처리의 용이성을 위해서 메시지 입력시 계산된 값을 버퍼에 저장하도록 설계하였다.

표 5. 입력 X[i]의 접근 순서
Table. 5 Access order of input X[i]

구분	단계	각 단계 입력 X[i]	비고
MD5	0~15	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15	16개의 입력에 대해 순서를 변경하여 적용
	16~31	1, 6, 11, 0, 5, 10, 15, 4, 9, 14, 3, 8, 13, 2, 7, 12	
	32~47	5, 8, 11, 14, 1, 4, 7, 10, 13, 0, 3, 6, 9, 12, 15, 2	
	48~63	0, 7, 14, 5, 12, 3, 10, 1, 8, 15, 6, 13, 4, 11, 2, 9	
HAS-160	0~19	18, 0, 1, 2, 3, 19, 4, 5, 6, 7, 16, 8, 9, 10, 11, 17, 12, 13, 14, 15	각 단계는 16개의 입력과 생성된 4개의 데이터를 순서를 변경하여 적용
	20~39	18, 3, 6, 9, 12, 19, 15, 2, 5, 8, 16, 11, 14, 1, 4, 17, 7, 10, 13, 0	
	40~59	18, 12, 5, 14, 7, 19, 0, 9, 2, 11, 16, 4, 13, 6, 15, 17, 8, 1, 10, 3	
	60~79	18, 7, 2, 13, 8, 19, 3, 14, 9, 4, 16, 15, 10, 5, 0, 17, 11, 6, 1, 12	

Ⅳ. 통합 해쉬 알고리즘의 설계

본 장에서는 통합 해쉬 알고리즘의 설계에 대해 설명할 것이다. 먼저 본 설계에서 사용한 통합 해쉬 알고리즘의 구현 블록도는 그림 3과 같다. 속도를 높이기 위해서 각 스텝 연산은 그림 3에 나타난 바와 같이 스텝 중간에 레지스터를 삽입하여 2 클럭에 수행하는 방안과 스텝당 한 클럭에 수행한 방안을 모두 구현 보았다.

MD5는 128 비트 단위로 스텝 동작을 수행하며, HAS-160은 160비트 단위로 스텝 동작을 수행한다. 따라서 128비트(A~D)는 동일한 레지스터를 사용하지만, 나머지 32비트(E)는 HAS-160에서만 필요하기 때문에 점선으로 표현하였다. 그리고 전체적인 데이터 흐름은 MD5는 실선으로 HAS-160은 점선으로 표현하였다.

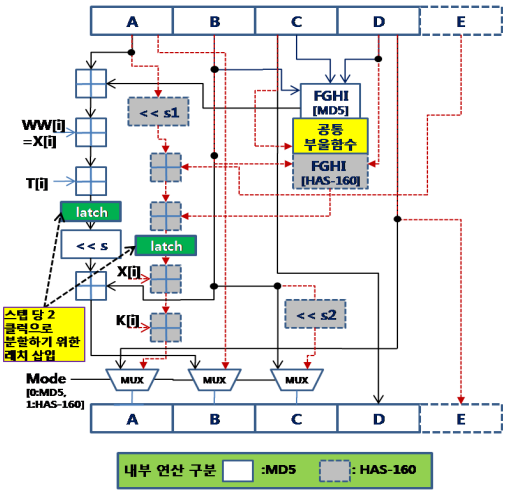


그림 3. 통합 해쉬 알고리즘의 단일 단계 동작 블록도
Fig. 3 Block diagram of a single step operation of unified MD5 and HAS-160 hash algorithm

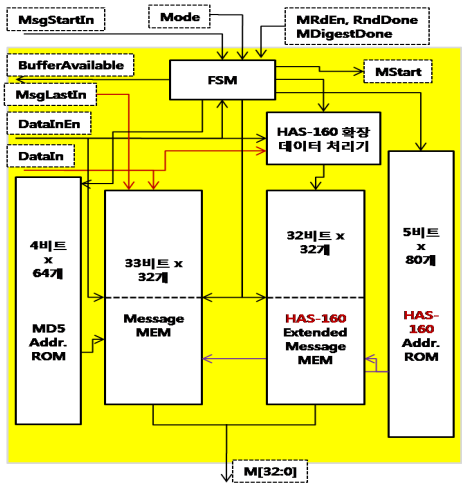


그림 4. 통합 해쉬 알고리즘의 메시지 버퍼 블록도
Fig. 4 Block diagram of message buffer of unified MD5 and HAS-160 hash algorithm

본 논문에서는 512비트 단위의 연속적인 메시지 블록을 처리할 수 있도록 메시지 버퍼를 설계하였다. 입력이 33비트인 메시지 메모리의 경우는 최상위 1비트가 “MsgLastIn” 신호를 래치하게 된다. 이는 메시지의 마지막을 판별하기 위해 추후에 상태머신에서 사용된다. 그리고 내부에 존재하는 상태머신인 FSM에서 유효한 512비트의 데이터를 보유하면 즉시 “MStart” 신호를 생성하며 라운드 함수 상태머신은 모드에 따라 라운드 함수를 수행하게 된다. 실제 필요한 메시지 버퍼 크기의 2배로 크기로 설계한 것은 메시지 요약을 수행하는 중에 다음에 수행될 메시지를 수신하여 현재 메시지 블록 처리 후 다음 번 메시지 블록을 즉시 처리할 수 있도록 하기 위함이다. 또한 그림에서 보면 알 수 있듯이 확장 메시지 버퍼 메모리를 제공하고 있는데, 이는 HAS-160 해쉬 알고리즘이 입력되는 데이터에 기반하여 추가적으로 16개의 데이터를 생성하기 때문이다. 최초 데이터 입력시 향후 사용될 추가 데이터를 생성하여 메모리에 저장함으로써 제어의 용이성을 추구하였다. 그리고 MD5와 HAS-160은 이중 포트 메모리를 액세스하기 위한 주소 ROM을 제공하고 있다.

표 6은 본 논문에서 설계된 통합 해쉬 코어(Core)의 입출력 신호와 각 신호에 대해 설명하고 있다.

표 6. 설계된 통합 해쉬 코어의 입출력 신호
Table. 6 I/O signals for unified hash core

신호명	I/O	설명
reset	I	리셋신호
clk	I	시스템 클럭
Mode	I	0: MD5, 1:HAS-160
LdIVEn	I	초기화벡터 로드 인에이블 신호
DataInEn	I	데이터입력 인에이블 신호
DataIn[31:0]	I	32비트 입력 [데이터와 초기화벡터 버스 공유]
MsgStartIn	I	메시지의 최초 데이터 전달시 활성화
MsgLastIn	I	메시지의 마지막 데이터 전달시 활성화
BufferAvailable	O	내부 메시지 버퍼에 512비트 공간이 존재함을 알림
MsgOut[31:0]	O	32비트 최종 다이제스트 결과
MsgOutEn	O	최종 다이제스트 결과 valid 신호 (MD5: 4회, HAS-160: 5회)

V. 통합 해쉬 알고리즘의 고찰

본 논문에서 설계한 통합 해쉬 알고리즘의 아키텍쳐는 먼저 C 언어를 통해 검증을 수행한 후, 이를 Verilog-HDL로 코딩하고, Xilinx사의 ISE7.1i에서 합성 및 검증하였다. 그림 MD5 해쉬 코어에 대한 시뮬레이션을 보여주고 있다. Mode 신호가 ‘0’이면 MD5 모드에서 동작하게 된다. 우선적으로 DataInEn신호가 활성화되어 있을 때, DataIn[31:0] 데이터가 내부의 메시지 버퍼에 담기며 16개의 워드가 쓰여지면 내부 상태머신은 즉시 MD5 해쉬 알고리즘을 수행한다. 메시지 블록에 대한 64라운드가 완료되면 RndDone신호가 한 클럭 활성화되고, 메시지 버퍼 M[32]를 통해서 전달된 값을 래치한 값을 보고 현재 메시지 블록이 최종 메시지 블록인지 여부를 판별한다. M[32]의 데이터 값이 ‘1’이면 최종 메시지 블록을 의미하면 이 때는 해쉬 수행 결과를 출력한 다음 내부 상태머신에서 MDigestDone 신호를 생성하여 통합 해쉬 코어가 초기 상태 모드에 진입하여 새로운 메시지 블록이 입력되기를 기다리게 된다. 시뮬레이션에 사용한 입력은 “abc”이며 최종 해쉬 결과는 “90 01 50 98 3C D2 4F B0 D6 96 3F 7D 28 E1 7F 72”이며, 4바이트씩 Little-endian 방식으로 변환하여 verilog 결과와 비교하면 동일함을 알 수 있다.

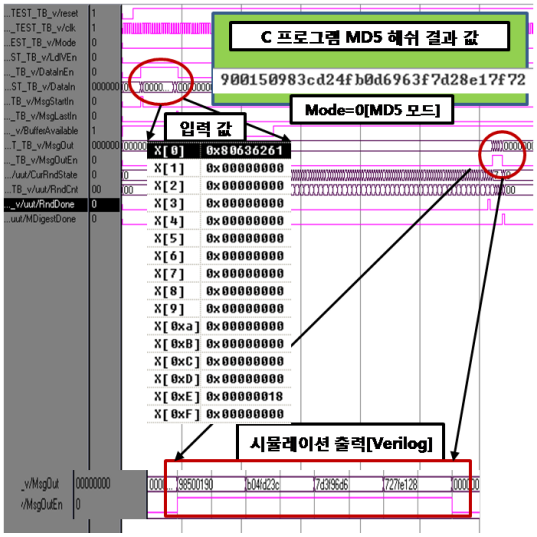


그림 5. MD5 해쉬 알고리즘의 시뮬레이션
Fig. 5 Simulation of MD5 hash algorithm

그림 6은 설계된 통합 해쉬 코어에서 Mode 신호가 1인 HAS-160 해쉬 알고리즘에 대한 시뮬레이션을 보여준다. 대부분의 신호는 MD5와 같지만, 출력의 경우 MD5는 128비트인데 비해 HAS-160은 160비트이다. 그리고 RndCnt 신호는 라운드와 단계을 표하는 레지스터로 하위 5비트는 20개의 단계를 의미하여 상위 2비트는 현재 진행중인 라운드를 의미한다. 다만, MD5 모드에서는 하위 4비트가 16개의 단계를 의미하고, 상위 2비트가 수행중인 라운드를 표시하도록 설계하였다. 시뮬레이션 결과 C 프로그램으로 수행한 결과와 시뮬레이터에서 수행한 결과가 동일함을 알 수 있다.

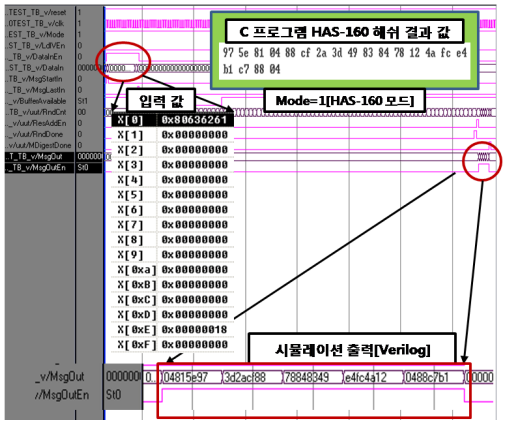


그림 6. HAS-160 해쉬 알고리즘의 시뮬레이션
Fig. 6 Simulation of HAS-160 hash algorithm

본 논문에서는 MD5와 HAS-160 해쉬 코어를 개별적으로 구현해 보고, 다시 최적화하여 통합 해쉬 알고리즘 코어를 Xilinx Vertex에서 구현해 보았다. 단계당 1클럭으로 구현한 통합 해쉬 코어는 1413개의 슬라이스를 사용하며 92MHz에서 동작하였으며, 단계당 2클럭으로 구현한 통합 해쉬 코어는 1428개이 슬라이스를 사용하고 133MHz에서 동작하였다. 메시지 버퍼에 대한 메모리를 두 모델 모두 33비트x 32개 버퍼와 32비트x 32개 버퍼를 사용한다. 이 때 사용되는 32비트x 32개의 버퍼는 한국형 표준 해쉬 알고리즘인 HAS-160에서 추가적으로 생성된 데이터를 저장하는 용도로 사용된다. 표 7은 본 논문에서 구현한 해쉬 알고리즘 코어를 정리한 것이다. 표에서 단일의 MD5와 HAS-160 코어는 면적 비교를 위해 구현한 것이다.

표 7. 설계 해쉬 코어의 비교
Table. 7 Comparison of designed hash core

구분	슬라이스 수	사용된 메모리	주파수 (MHz)
MD5 [각 스텝:2클럭]	789	33비트 x 32개 이중포트메모리	-
HAS-160 [각 스텝:2 클럭]	1105	33비트 x 32개와 32비트 x 32개 이중포트메모리	-
통합해쉬 (MD5+HAS-160) [각 스텝:1 클럭]	1413	33비트 x 32개와 32비트 x 32개 이중포트메모리	92
통합해쉬 (MD5+HAS-160) [각 스텝:2 클럭]	1428	33비트 x 32개와 32비트 x 32개 이중포트메모리	133

HAS-160 해쉬 코어를 기본으로 비교할 때 스텝 2클럭으로 구현된 통합 해쉬 코어는 슬라이스 수는 32% 증가하지만, 메모리는 HAS-160에서 구현된 것을 그대로 사용하므로 추가적인 면적이 증가하지 않음을 알 수 있다.

마지막으로 본 논문에서 구현된 통합 해쉬 코어의 성능을 비교한 것이다. 본 논문에서 설계된 통합 해쉬 코어는 인증, 디지털 서명, 무결성 등을 적용한 보안 분야에서 활용이 가능할 것으로 사료된다.

표 8. 해쉬 코어의 성능 비교
Table. 8 Performance comparison of hash core

구분	알고리즘	주파수 (MHz)	메시지블록당 해쉬 사이클	성능 (Mbps)
논문[1]	MD5	120	132	465
논문[2]	HAS-160	50	82	312
논문[6]	MD5	21	65	165
논문[7]	MD5	60	66	467
본 논문의 통합해쉬 [단계당 1클럭]	MD5	92	65	724
	HAS-160	92	81	581

Ⅵ. 결 론

본 논문에서는 MD5와 HAS-160 해쉬 알고리즘을 통합한 통합 해쉬 코어를 설계하였다. 스텝 당 1클럭에 수행되는 통합 해쉬 코어를 기준으로 비교할 때 HAS-160 해쉬 코어에 32%의 슬라이스를 추가하면 통합된 해쉬 코어를 구현할 수 있고, 추가적인 메시지 저장을 위한 메모리를 필요로 하지 않는 면적 효율적인 구조가 됨을 알 수 있다. 단계당 1클럭에 수행되는 통합 해쉬 코어는 92MHz에서 동작함을 확인하였고, MD5 해쉬 알고리즘 동작 모드에서는 최대 724Mbps의 메시지를 다이제스트(Digest)할 수 있으며, HAS-160 해쉬 알고리즘 동작 모드에서는 최대 581Mbps의 메시지를 다이제스트할 수 있다.

설계된 통합 해쉬 코어는 전자 상거래, 데이터 무결성, 디지털 서명, 대규모 암호화, 고속 네트워킹 장비 및 무선 응용 분야에 걸쳐 다양하게 응용될 수 있을 것으로 기대된다.

참고문헌

[1] 최병윤, 박영수, “하드웨어 공유와 캐리 보존 덧셈을 이용한 MD5 해쉬 프로세서의 설계”, 정보보호학회 논문지, 13권 제 4호, pp.139-149, Oct. 2003.

[2] 김해주, 전홍우, 신경욱, “해쉬 알고리즘 표준 HAS-160의 저면적 하드웨어 구현”, 한국해양정보통신학회논문지, 제14권3호, pp.715-722, Mar. 2009.

[3] 박창섭, 암호 이론과 보안, 대영사, 1999

[4] http://www.cast-inc.com/ip-cores/encryption/md5/cast_md5-a.pdf

[5] TTA(한국정보통신기술협회), 해쉬 표준 함수 - 제 2부 : 해쉬 함수 알고리즘 표준(HAS-160), Dec. 2005.

[6] Janaka Deepakumara, Howard Heys and R. Venkatesan, ″FPGA Implementation of MD5 Hash Algorithm″, Canadian Conference on Electrical and Computer Engineering, Vol.2, pp.13-16, May. 2001.

[7] Diez J. M., et al., ″Hash Algorithm for Cryptographic Protocols: FPGA Implementations″, 10th TELFOR‘ 2002, Nov. 2002.

저자소개

손승일(Seung-Il Sonh)

1989년 연세대학교 전자공학과(학사)
1991년 연세대학교 대학원 전자공학과(석사)
1998년 연세대학교 대학원 전자공학과(박사)
1998～2002년 호남대학교 컴퓨터공학과 조교수
2008～2009년 미국 미시간공과대학 방문교수
2002년～현재 한신대학교 정보통신학과 교수
※관심분야 : ATM 통신 및 보안, ASIC 설계