CSA를 사용한 고속 MD5 프로세서 구현

윤희진*, 정용진*
*광운대학교 전자통신공학과
e-mail : hijiny@explore.gwu.ac.kr

Implementation of high speed MD5 processor using CSA

Hee-Jin Yoon*, Yong-Jin Jeong*
*Dept. of Electronic Communication Engineering, Kwangwoon University

요 약

본 논문에서는 해쉬 함수를 바탕으로 한 메시지 인증 코드 중의 하나인 MD5 를 하드웨어로 설계하였다. MD5 는 block-chained digest 알고리즘으로 64 단계의 동일한 단계 연산 구조를 가지므로 가장 기본적인 연산 한 단계를 구현하여 반복적으로 수행하는 구조로 설계하였다. 단계 연산구조 내에서는 연속된 32bit 덧셈 연산이 이루어지는데 기존의 CLA(carry-lookahead-adder)만을 사용하여 구현한 구조 대신 본 논문에서는 CSA(carry-save-adder)와 CLA 를 혼용하였다. 덧셈연산의 결과는 순서와 상관없기 때문에 연산자의 덧셈 순서를 리스케쥴링 하였으며, 이는 기존의 CLA 만을 이용한 방법과 비교하여 최장지연 경로를 15% 줄여 훨씬 빠르게 연산을 수행하고, 전체 면적도 30%를 줄일 수 있었다. 결과적으로 본 논문에서 제안하는 구조는 지금까지 나온 어떤 MD5 프로세서 보다 작고 빠른 프로세서를 구현 할 수 있을 것으로 판단된다.

1. 서론

정보화 사회가 점점 발달함에 따라 그 만큼의 많은 편리함이 우리에게 주어졌지만 그에 따른 많은 위협들이 존재 하고 있다. 특히 요즘과 같이 컴퓨터 앞에서 인터넷으로 금융거래, 전자상거래, 카드결제가 빈번히 이루어지는 상황에서 인터넷 보안상의 사고나 전산 시스템의 범죄들은 우리를 가장 많이 위협하는 요소다. 인터넷과 전산 시스템 상에서 보내고자 하는 정보를 안전하게 보내는 것은 매우 중요하다. 정보보안에서 메시지인증은 받는 사람의 입장에서 보낸 사람의 메시지인지를 증명하고 또한 그 보낸 메시지가 손상되지 않은 것임을 확신시켜주는 것은 중요한 과정이다. 인증의 핵심적인 요소로는 메시지 인증 코드(MAC : Message Authentication Code)이다. 그 중에서도 해쉬함수를 바탕으로 해서 메시지 인증 코드를 생성하는 방법을 HMAC 이라고 한다.[2]

해쉬 알고리즘이란 임의의 길이의 비트 열을 고정된 길이의 출력값인 해쉬코드로 압축시키는 알고리즘이다.[1] 그 특징은 첫째, 주어진 출력에 대하여 입력값을 구하는 것이 계산상 불가능하며 둘째, 주어진 입력에 대하여 같은 출력을 내는 또 다른 입력을 찾아내는 것이 계산상 불가능해야 한다. 암호학적 응용에 사용되는 대부분의 해쉬알고리즘은 위의 두 성질뿐만 아니라 이보다 강한 충돌저항성을 지니도록 요구된다. 암호학적 해쉬알고리즘의 충돌 저항성은 전자서명에서 송신자 외의 제 3 자에 의한 문서위조를 방지하는 부인방지서비스를 제공하기 위한 필수적인 요구조건이 된다.[1] 이러한 용도로 사용되는 대표적인 해쉬알고리즘이 MD5 이다.

MD5 는 미국 MIT 대학의 Ron Rivest 교수에 의해 개발된 메시지 압축 알고리즘이다. MD2 를 초기버전으로 시작하여 MD4, MD5 로 점차 알고리즘을 개발시켰다. MD5 는 이전버전 MD4 보다 계산속도는 느리지만 훨씬 안전성이 뛰어나 외부 공격의 위협에 강하도록 고안되었다. 또한 IPSEC, IVv4, IPv6 등에서 인증 알고리즘으로 가장 많이 사용 하고 있는 해쉬 알고리즘이다. MD5 알고리즘은 32 비트 단위로 연산이 수행

이 논문은 광운대학교 반도체 설계 교육 센터(IDEC)의 자원으로 연구되었음.

되기 때문에 32 비트 시스템에서 잘 작동되도록 설계되었고, 중간 계산과정 중에 어떤 치환 작용도 일어나지 않는다.[4]

MD5 는 임의의 길이의 메시지를 입력 받아 128 비트의 고정된 길이를 출력 시키는 함수로 임의의 입력 메시지를 512 비트 단위로 처리한다. 메시지를 입력 받아 출력을 내는 과정은 다음과 같다.[1]

⊙ 첫번째 과정: 패딩(padding) 및 덧붙이기 : 입력 메시지는 512 비트 단위로 처리된다. 마지막 메시지 블록은 블록의 길이가 448 비트(=512-64)가 되도록 "1" 다음에 필요한 개수의 "0"으로 패딩한 후, 마지막 64 비트는 덧붙이기 전 메시지 길이를 2^{64} 법 연산 값으로 채운다.

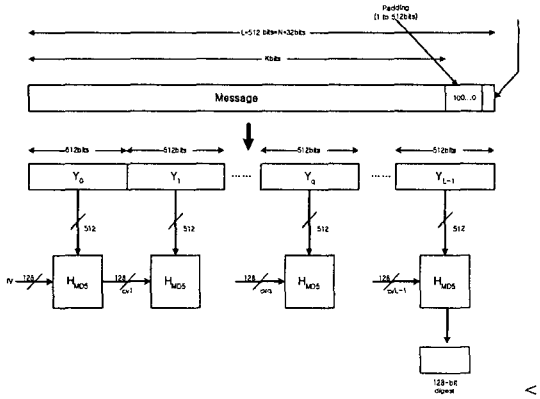


그림 1> MD5 의 해쉬과정

⊙ 두번째 과정: MD 버퍼 초기화 : 128 비트 버퍼는 MD5 를 수행하는 동안의 중간 값과 최종 결과 값을 저장하기 위해 사용된다. 128 비트 버퍼는 32 비트 레지스터 ABCD 로 표시되며 초기값은 표 1 과 같다.

A	B	C	D
67452301	efcdab89	98bacdfe	10325476

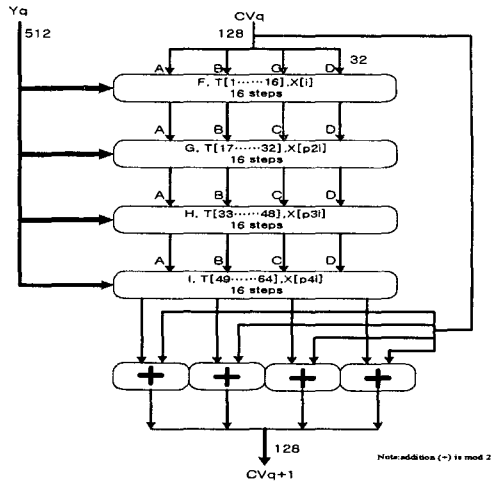
<표 1> 초기값

⊙ 세번째 과정: 512 비트(16-word)블록 처리 과정 : MD5 알고리즘의 핵심은 <그림 2>에서 보여주듯이 4 개의 라운드에서 처리되는 digest 과정이라고 할 수 있다. 4 개의 라운드 계산 과정은 비슷한 구조를 가지고 있지만 각 라운드에서 사용되는 논리 연산함수 F, G, H, I 는 다르다.

라운드	함수	논리연산
1	F(X, Y, Z)	(X ∧ Y) ∨ (~X ∧ Y)
2	G(X, Y, Z)	(X ∧ Z) ∨ (Y ∧ ~Z)
3	H(X, Y, Z)	X ⊕ Y ⊕ Z
4	I(X, Y, Z)	Y ⊕ (X ∨ ~Z)

<표 2> 논리연산 함수

각 라운드에서는 MD5 를 수행하고자 하는 512 비트 데이터 y_q 와 128 비트 버퍼 ABCD 에 저장된 값, 사인 함수로 구해진 64 개의 32 비트 값이 입력으로 처리된다. 네번째 라운드를 지나고 나온 결과값은 첫 번째 라운드의 입력값(CV_q)과 더해지고 MD5 를 수행한 값(CV_{q+1})을 생성한다.[4]



<그림 2 >H_{MD5} 처리 과정

덧셈은 modulo 2^{32} 덧셈을 사용하고 중간에 수행하는 순환비트 연산 값은 단계연산마다 다른 값들을 가지고 비트가 순환된다. 단계연산 과정은 다음과 같다.[4]

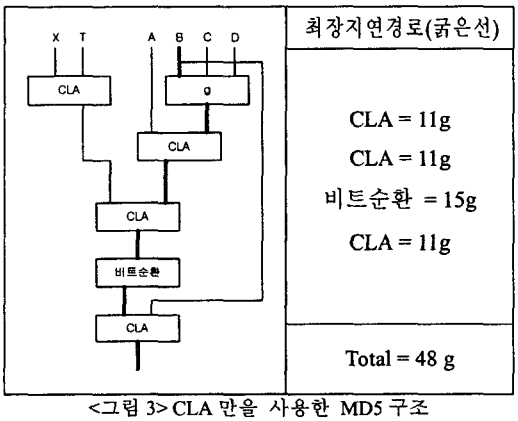
A <- D
B <- B+((A+Func(B,C,D)+X[k]+T[i])<<s)
C <- B
D <- C

2. MD5 해쉬 프로세서 구현

해쉬함수 알고리즘은 소프트웨어로 구현하기 쉽고 업그레이드, 이동성 그리고 이식성이 하드웨어보다 용이하기 때문에 소프트웨어 기반에서 많이 구현되어왔다. 그러나 하드웨어로 구현할 경우 소프트웨어보다 물리적인 공격에 있어서 안전성이 훨씬 뛰어나다. 또한 소프트웨어는 시스템의 플랫폼에 따라 성능이 제한되지만 하드웨어는 그러한 제약 없이 계산할 수 있다. 따라서 본 논문에서는 대표적인 해쉬함수 중 하나인 MD5 를 하드웨어로 설계하였다. MD5 알고리즘은 block-chained 알고리즘이기 때문에 현재 블록에서 수행하려는 H_{MD5}(전체 MD5 계산과정 중 한 블록계산)값은 현재 블록에 입력으로 들어오는 512 비트 메시지 값과 이전 블록에서 수행된 128 비트 H_{MD5} 값에 영향을 받는다.

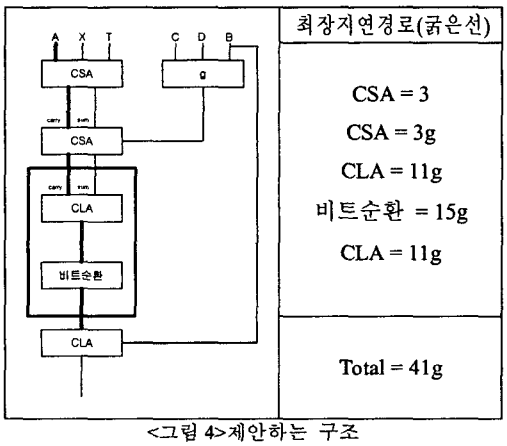
한 블록의 연산과정은 4 번의 라운드를 거쳐야 하고 한 라운드는 총 16 번의 기본적인 단계 연산을 거친다. 기본적인 한 단계연산과정에서는 4 번의 덧셈, 1 번의 논리 연산, 1 번의 비트순환 연산이 일어난다. 기본적인 단계 연산 과정은 4 번의 덧셈과정을 수행한다. 그 중 세번의 덧셈은 비트순환에 앞서 연속적으로 이

루어진다. 이 때 기존의 구현 예들은 CLA 만을 사용하였기 때문에 3 개의 CLA 만큼의 계산 시간이 필요하다. 하지만 MD5 의 내부연산을 고려하여 CSA 와 CLA 를 혼용한다면 더 빠른 속도로 계산 할 수 있다. 또한 CSA 는 CLA 보다 적은 하드웨어 리소스를 필요하기 때문에 하드웨어 사이즈가 작아지는 장점이 있다. 3 번의 덧셈 뒤에 이루어지는 비트순환은 각 연산마다 다른 값을 쉬프트 하기 때문에 MUX 를 이용한 일반적인 배럴쉬프터 구조로 설계하였다. <그림 3>은 CLA 만으로 구성했을 때의 구조를 보여주고 있다. 참고한 논문[2]에서 구현한 구조도 이와 비슷하다.



<그림 3> CLA 만을 사용한 MD5 구조

본 논문에서는 앞에서 언급했듯이 CSA 와 CLA 를 적절하게 구성하여 최장지연경로를 줄이고 면적에서도 gate 수를 줄일 수 있는 구조를 제안한다. 제안하는 구조는 <그림 4>와 같다.



<그림 4>제안하는 구조

제안한 구조에서는 CSA 2 개, CLA 2 개를 사용하였다. 특히 비트 순환이 일어나기 전에 CSA 를 사용함으로써 덧셈 연산이 빨리 이루어지도록 하였다. 입력 단에서 A 버퍼 값, 메시지 X[k]값, T[i]값을 받아 CSA 를 수행하고, 그 결과로 나온 carry 와 sum 은 버퍼 B, C, D 의 논리연산과정을 거쳐서 나온 값과 다시 CSA 를 수행한다. 그리고 최종적으로 나온 carry 와 sum 을 CLA 를 통해 연속된 덧셈 값을 출력한다. 최장지연경로를 비교하면 7g(g 는 게이트 한개의 지연경로)만큼 빠름을 볼 수 있다. 사이즈 측면에서 비교하면 <표 3>과 같다.

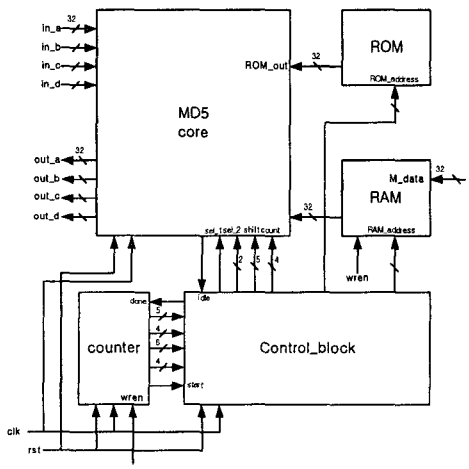
게이트	CLA	CSA	CLA 만 사용 (CLA 4 개)	제안한 구조 (CLA2 개, CSA2 개)
2input AND	137 개	64 개	548 개	402 개
2input OR	130 개	32 개	520 개	324 개
2input XOR	64 개	64 개	256 개	256 개

<표 3> 사이즈 비교

3. 하드웨어 설계

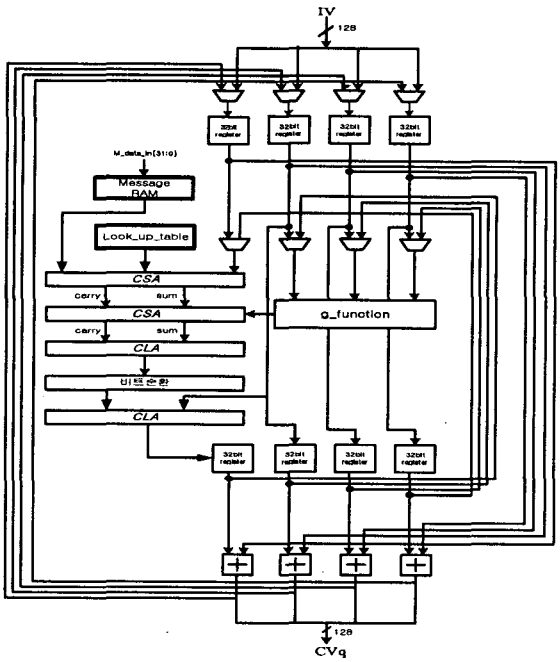
본 논문에서 설계한 MD5 전체 구조는 <그림 5>에 보이는 바와 같이 몇 개의 서브 블록으로 나누어져 있다. MD5 핵심블럭은 알고리즘의 가장 기본적인 단계연산과정을 구현한 것으로 속도를 빠르게 할 뿐만 아니라 면적 측면에서도 많은 이점이 있는 반복적 구조로 설계하였다. 제어블럭에서는 핵심블럭에서 사용할 MUX 제어 신호와 비트순환값, ROM 과 RAM 주소를 생성한다. 카운터 블럭은 제어블럭에 필요한 신호들과 MD5 를 수행하려는 512 비트의 메시지 값을 RAM 에 저장한 후 단계연산과정을 시작하는 start 신호를 제어블럭으로 보낸다.

MD5 를 수행하려는 512 비트 메시지가 32 비트씩 램에 저장이 되어지면 카운터 블럭에서는 start 신호를 제어블럭에 전달하고 각 단계연산 count 가 증가 됨에 따라 필요한 신호들이 MD5 핵심블럭, ROM, RAM 에 전달되어 64 번의 단계연산과정을 거친다. 단계연산 동안에는 핵심블럭에서 idle 신호를 보내 단계연산과정 중에 있음을 나타낸다. 64 번째 단계연산이 끝나면 제어블럭은 done 신호를 내보내고 출력으로는 H_{MD5} 과정을 거친 128 비트의 MD5 결과값이 출력된다.



<그림 5> MD5 전체 블록 다이어그램

MD5 코어블럭의 내부 구조는 <그림 6>과 같다. 핵심블럭은 이전 블록에서 수행된 MD5 값(CV_q)과 초기 벡터값(IV)을 선택하기 위한 입력단의 MUX, 초기값과 데이터의 귀환값을 선택하는 MUX, 각 라운드별로 논리 연산을 수행하는 함수블록, 3 개의 operand 를 갖는 32 비트 CSA, CSA 를 수행하고 난 이후 발생한 carry 와 sum 을 더해주는 32 비트 CLA, 각 단계 연산 과정마다 다른 비트 순환이 일어나기 때문에 크기가 크지만 가장 보편적이 안정적인 32 비트 배럴 쉬프트, 그리고 중간값을 저장하기 위한 32 비트 레지스터, 마지막 최종적인 MD5 값을 얻기 위한 초기값과 64 번의 단계연산을 거치고 나온 값을 더하기 위한 32 비트 CLA 로 구성되어있다.



<그림 6> MD5 핵심블럭

4.성능분석

본 논문에서 ALTERA APEX20k EP20K400EBC652-3 디바이스를 타겟으로 합성하였고, 그 결과 1032 개의 로직 셀을 사용하였다. 시뮬레이션을 통해 나온 결과값은 RFC1321 에서 제공하는 테스트 벡터 값과 일치함을 확인하였다.[4] MD5 를 한번 수행하는하는데 65 클럭이 걸렸으며 이는 기본적인 단계연산 과정을 한 클럭에 수행하고 마지막 덧셈을 수행하는데 걸리는 클럭수다. 설계한 MD5 프로세서의 최대 지연경로 시간은 약 37.8ns 로, 최대동작주파수 26.43Mhz 까지 가능하고 결과적으로 512 비트의 블록을 H_{MD5} 하는데 걸리는 시간은 총 2457ns 가 걸리고, 208Mbps 의 성능을 가진다. 이는 아래 <표 4>에서 볼 수 있듯이 지금까지 구현한 소프트웨어와 하드웨어보다 가장 빠른 성능을 가짐을 알 수 있다.

	효율	최대동작주파수
DEC Alpha(10Mhz) [3]	87Mbps	
Intel Pentium/90 NeXTStep[3]	44Mbps	
Sun SPARC-20/71 SunOS 4.1.3[3]	57Mbps	
참고논문[2]	165Mbps	21Mhz
제안한 구조	208Mbps	26.43Mhz

<표 4> 성능비교

5. 결론

본 논문에서는 해쉬 함수를 바탕으로 한 메시지 인증 코드 중의 하나인 MD5 를 하드웨어로 설계하였다. 다양한 플랫폼상에서 소프트웨어로 구현한 사례는 많지만 하드웨어로 구현한 예는 드물다. MD5 알고리즘에서 발생하는 연속된 덧셈을 효과적으로 처리하기 위해 CSA 와 CLA 를 같이 사용하였고, 덧셈 연산자의 계산 순서를 리스케쥴링 하였다. 덧셈 연산에 CSA 를 사용함으로 CLA 만을 사용했을 때보다 최장지연경로를 15% 줄일 수 있었으며 또한 면적에서도 30%를 줄임으로 성능 향상을 가져 올 수 있었다. 향후로는 ASIC 으로 구현하거나 파이프라인을 시키면 더 빠른 속도로 계산 과정을 수행할 것으로 기대된다.

6. 참고문헌

[1] 해쉬함수 표준 - 제 2 부 : 해쉬함수 알고리즘 (HAS-160), 한국정보통신기술협회, 1998 년 11 월.

[2] Janaka Deepakumara, Howard M. Heys and R. Venkatesan " *FPGA implementation of MD5 hash algorithm,*" , Electrical and Computer Engineering, 2001. Canadian Conference on , Volume: 2 , pp. 919 - 924, 2001.

[3] J. Touch, " *Report on MD5 performance* " , RFC 1810, June, 1995.

[4] R. Rivest, " *The MD5 Message-Digest Algorithm* " , RFC 1321, MIT LCS & RSA Data Security Inc., April ,1992.

[5] W. Stallings, " *Ctyptography and Network Security* " , Second edition. Prentice Hall, 1997.

[6] Joseph D. Touch, " *Performance Analysis of MD5* " ,Sigcomm ' 95, Boston MA.

[7] Cetin Kaya Koc, " *RSA Hardware Implementation* " , RSALaboratories RSA Data Security, Inc., August 1995.

[8] Behrooz Parhami, " *Computer Arithmetic Algorithms and Hardware Designs* " , Oxford, 2000.