情報保護學會論文誌
第13券 第4號, 2003. 8

하드웨어 공유와 캐리 보존 덧셈을 이용한 MD5 해쉬 프로세서의 설계*

최 병 윤**, 박 영 수***

Design of MD5 Hash Processor with Hardware Sharing and Carry Save Addition Scheme

Byeong-Yoon Choi**, Young-Soo Park***

요 약

본 논문에서는 하드웨어 공유와 캐리 보존 덧셈 연산을 이용하여 MD5 알고리즘을 구현하는 면적 효율적인 해쉬 프로세서를 하드웨어로 설계하였다. 면적을 최소화하기 위해, MD5의 1 단계 동작을 2개의 부분 단계로 세분화하고, 각각의 부분 단계 동작을 동일 하드웨어로 구현하는 방식으로 하드웨어 공유를 극대화하였다. 그리고 MD5의 부분 단계를 구성하는 3개의 직렬 캐리 전달 덧셈 동작을 2개의 캐리 보존 덧셈과 1개의 캐리 전달 덧셈으로 변환하여 동작 주파수를 증가시켰다. MD5 해쉬 프로세서는 0.25μm CMOS 표준 셀 라이브러리로 합성한 결과 약 13,000개의 게이트 수로 구성되며, 타이밍 분석 결과 설계된 MD5 해쉬 프로세서는 120 Mhz의 동작 주파수에서 512 비트 입력 메시지에 대해 465 Mbps의 성능을 갖는다.

ABSTRACT

In this paper a hardware design of area-efficient hash processor which implements MD5 algorithm using hardware sharing and carry-save addition schemes is described. To reduce area, the processor adopts hardware sharing scheme in which 1 step operation is divided into 2 substeps and then each substep is executed using the same hardware. Also to increase clock frequency, three serial additions of substep operation are transformed into two carry-save additions and one carry propagation addition. The MD5 hash processor is designed using 0.25 μm CMOS technology and consists of about 13,000 gates. From timing simulation results, the designed MD5 hash processor has 465 Mbps hash rates for 512-bit input message data under 120 Mhz operating frequency.

keyword : *Hash Algorithm, MD5, SHA-1, HAS-160, IPsec, Multimedia security, Cryptographic processor*

Ⅰ. 서 론

정보화 사회는 인터넷을 넘어서 유비쿼터스 환경으로 발전하고 있지만, 정보화는 생활의 편이성과 정보 공유라는 긍정적인 측면과 함께 현금 카드 위·변조, 정보의 해킹 또는 크래킹의 정보화 역기능이 함께 존재한다. 이러한 정보화 역기능에 대처하기 위해 기밀성(secrecy, confidentiality), 무결성(integrity), 인증」

* 본 연구는 2003년도 동의대학교 교내 연구비와 한국 전자 통신 연구원 위탁 과제 사업 지원으로 이루어졌으며, 회로 구현에 IDEC 지원 장비를 사용하였습니다.
** 동의대학교 컴퓨터공학과(bychoi@dongeui.ac.kr)
*** 한국전자통신연구원 정보보호기술연구본부(yspark@etri.re.kr)

(authentication), 부인 봉쇄(non-repudiation) 등의 기술이 사용되고 있다. DES(Data Encryption Standard) 대칭형 암호 알고리즘 또는 RSA(Rivest Shamir Adleman) 공개키 암호 알고리즘은 정당한 송·수신자이외의 제3자로부터 메시지의 내용을 은닉시키기 위해서 사용하는 암호화 알고리즘으로 기밀성 보안기법이다. 그러나 유·무선 통신 채널은 해커나 크래커에게 노출될 수 있기 때문에 메시지의 변조, 즉 능동적인 공격에 취약하다. 따라서 사용자가 읽을 수 있는 일반 텍스트 문서가 아닌 음성 등의 데이터의 경우, 암호화되어 채널로 전송되는 데이터를 불법적인 제3자가 일부 데이터의 순서를 바꾸거나 변조하는 공격을 할 경우 수신자는 데이터가 변조되었는지를 구별하는 것이 불가능하다. 따라서 단순한 기밀성을 유지하기 위해 암호 알고리즘을 사용하는 것으로는 보안 서비스가 충분하지 않다. 수신자 입장에서는 수신된 메시지가 전송되어지는 과정에서 불법적인 제3자에 의해서 의도적으로 변조되지 않았다는 확신을 갖게 하기 위해, 암호 알고리즘을 통해 구현되는 메시지의 무결성 기법과 함께 수신된 메시지의 변조 여부를 확인하는 무결성 기술이 요구된다[1]. 이러한 메시지의 변조 여부를 확인하기 위해 메시지의 내용에서 다이제스트(digest)를 도출하는 해쉬(hash) 알고리즘이 사용된다. 이러한 해쉬 알고리즘은 디지털 서명, 개인 식별, 인터넷 보안, 웹 보안 등에 널리 사용되고 있다. 그동안 해쉬 알고리즘은 대부분 소프트웨어 방식으로 구현되고 있으나, IPSec[2], 유사 난수 발생기(PRNG : Pseudo Random Number Generator)등 네트워크 보안 및 PKI(Public Key Infrastructure) 분야에 사용되기 위해서 고속의 해쉬 프로세서 설계가 필요하게 되었다. 현재 대표적인 해쉬 알고리즘으로 MD5[3], SHA-1[4], RIPPEM-160[5], HAS-160[6] 등이 있다. 본 논문에서는 IPSec에서 표준 해쉬 알고리즘으로 사용되며, M.I.T. 대학의 Rivest 교수에 의해 개발된 MD5 해쉬 알고리즘을 하드웨어로 설계하고 성능을 분석하였다. MD5 알고리즘은 SHA-1과 RIPPEM-160과 유사한 구조를 갖고 있기 때문에 MD5 프로세서 개발은 다른 해쉬 알고리즘에도 쉽게 응용 가능하다.

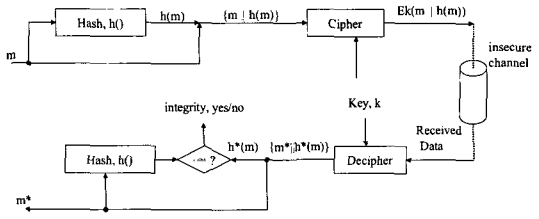
해쉬 알고리즘을 하드웨어로 구현한 사례를 살펴보면, 크게 하나의 라운드 하드웨어를 구성하고 이를 반복적으로 활용하는 iterative 방식[7,9], 몇 개의 라운드 동작을 하나의 블록으로 묶어 반복적으로 사용하는 loop unrolling기법[8], 그리고 라운드 하드웨어를 직렬체인으로 연결하는 파이프라인 기법으로 구현하는 방식으로 나누어 질 수 있다. 단, 파이프라인 방식의 경우 하드웨어 양이 너무 많이 요구되어 상용 프로세서에서는 실제 구현 예가 없다. 그리고 Loop unrolling 방식의 경우 반복 사이클 수는 줄일 수 있는 장점이 있지만, 클록 주파수가 상대적으로 감소하는 문제가 있기 때문에 면적의 증가에 비해 성능 개선 효과가 크지 않다. 따라서 대부분의 해쉬 하드웨어는 각 라운드 동작을 하나의 클록 사이클에 수행하는 iterative 방식을 채택하고 있다.

그런데 MD5 알고리즘은 각 라운드가 복잡한 직렬 덧셈 연산으로 구성되어 있기 때문에 각 라운드 동작을 단일 클록 시간 내에 구현하는 기존 iterative 방식의 경우, 단일 클록에 가산기가 동시에 사용되어야 하기 때문에 하드웨어 양이 너무 많이 필요하고, 내부 구조의 복잡성으로 동작 주파수가 크게 떨어지는 결점이 있다. 따라서 MD5를 면적 효율적인 응용분야에 적용하기 위해서, 자원 공유와 캐리 보존 덧셈 기법을 사용하여 면적 과 속도 측면에서 우수한 성능을 갖는 MD5 암호 프로세서 개발이 필요하다.

본 논문에서는 MD5 해쉬 알고리즘의 여러 가지 구현 방안을 비교하고, 면적과 속도 측면에서 효율적인 구조를 제안하고 이를 하드웨어로 설계한 후 성능을 분석하였다. 본 논문의 구성은 2장에서는 MD5 해쉬 알고리즘의 특징을 간단히 기술하고, 3장에서는 MD5 해쉬 프로세서의 설계를 다루며, 4장에서는 설계된 해쉬 프로세서의 검증과 성능 분석을 기술하였으며, 5장에서는 결론을 제시하였다.

II. MD5 해쉬 알고리즘

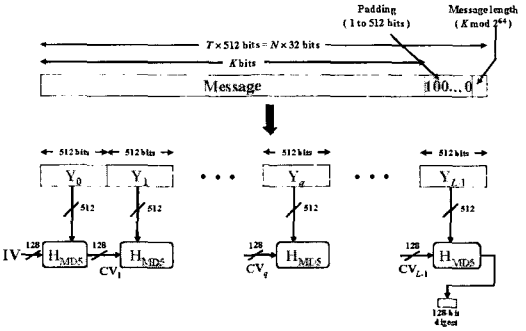
해쉬 함수는 가변적인 길이의 입력을 고정된 길이의 출력 값인 인증자(authenticator), 또는 해쉬 값(hash value)으로 압축하는 함수로 전자 서명과 송·수신되는 데이터의 변조 여부를 결정하는 무결성 확인에 주로 사용된다. [그림 1]은 메시지의 기밀성과 함께 무결성 확인 기능이 포함된 메시지 전달 개념을 나타낸다. 송신자의 경우 메시지, m과 함께 무결성을 위해 메시지에서 해쉬 값 h(m)을 생성한 후, 2개의 데이터를 암호 알고리즘을 통해 암호화(cipher)한 후 데이터 Ek(m ‖ h(m))을 전송한다. 여기서 ‖ 기호는 데이터를 연접(concatenation)시킴을 의미한다. 데이터는 불법적인 제3자에게 노출되는 안전하지 않은 채널로 전달되며, 수신자는 수신된 데이터를 복호화(decipher)후 메시지(m*)를 분리한 후 해쉬 값을 생성하여



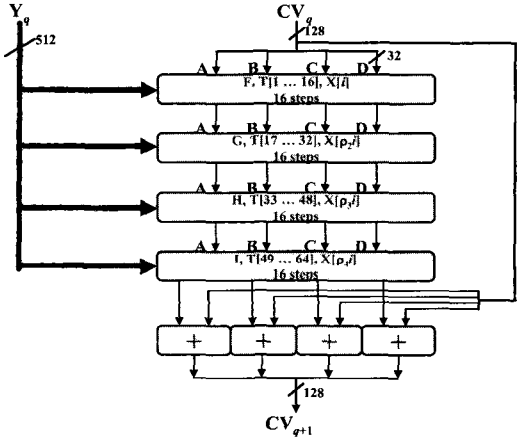
〔그림 1〕 메시지의 기밀성과 무결성이 보장되는 송·수신 개념

전송된 해쉬 값(h*(m))과 비교하여 변조 여부를 결정한다. 그림에서 첨자(*)의 의미는 불법적인 제 3자에 의한 메시지 변조를 고려하여 수신된 데이터를 송신 데이터와 구별해서 표현한 것으로, 변조되지 않은 경우 m과 m*는 동일한 값을 갖는다. [그림 1]에서 암호 및 복호 동작은 기밀성을 보장하며, 해쉬 함수는 무결성을 보장하는 역할을 한다.

해쉬 함수 h()는 임의 길이의 메시지를 입력 자료로 하여 고정된 길이의 비트 열을 출력하는 다대일(many-to-one) 함수이기 때문에 필연적으로 서로 다른 입력 자료에 대해 동일한 해쉬 값이 출력되는 충돌(collision)현상이 발생한다. 따라서 해쉬 함수가 출력하는 모든 m 비트 해쉬 값의 발생확률이 동일할 때, 무작위로 선정된 임의의 2개의 입력 자료에 대해서 동일한 해쉬 값을 발생할 확률은 2^{-m}이 된다. 암호학적 해쉬 함수는 해쉬 값 생성에 있어서 비밀키의 사용 여부에 따라 2가지 방식, 즉 MAC(Message Authentication Code) 해쉬 함수와 MDC(Modification Detection Code)로 구분된다. MAC 해쉬 함수는 입력 자료 m과 비밀키 k가 해쉬값 h(m, k)를 생성하는데 비해, MDC 해쉬 함수는 해쉬값 h(m)생성에 입력 자료 m만 사용된다. MD5, SHA, HAS-160 등은 MDC 계열의 해쉬 전용으로 제안된 알고리즘이며, 특히 HAS-160은 국내에서 독자적으로 정의한 해쉬 함수이다. MIT 대학의 Rivest 교수에 의해 제안된 MD5는 그림 2와 같이 가변적인 길이의 메시지를 입력으로 받아들여 128 비트의 해쉬 값을 출력하는 해쉬 알고리즘이다. 먼저 가변적인 길이(K bits)의 입력 메시지 m이 주어지면, 채우기(padding)와 메시지 길이 값을 삽입하는 사전 처리(pre-processing) 과정을 거쳐 L개의 512 비트 블록 Y_i로 변환한다. 사전 처리 단계를 거친 메시지는 512 비트(Y_i)씩 식 (1)과 같이 해쉬 알고리즘을 반복적으로 적용하여 128 비트 해쉬 값을 생성한다. IV 값은 표준안에 상수 값으로 정의되어 있다.



〔그림 2〕 MD5알고리즘을 사용한 사전처리 작업과 연산 과정



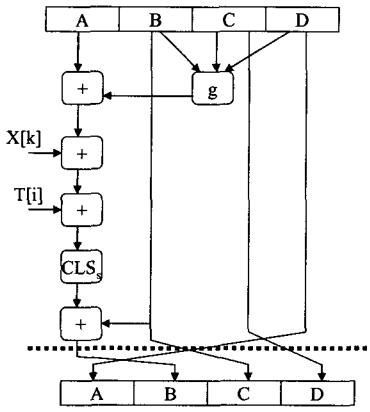
〔그림 3〕 사전 처리된 512 비트 메시지에 대한 MD5 해쉬 일고리즘

$$CV_0 = IV$$

$$CV_i = H_{MD5}(CV_{i-1}, Y_i),\ 0 \le i \le L-1 \quad (1)$$

$$\text{hash value} = CV_L$$

해쉬 알고리즘(H_{MD5})은 [그림 3]과 같이 4개의 라운드(F, G, H, I)와 추가의 4개의 모듈러(modular) 덧셈으로 구성된다. 그리고 각 라운드는 내부적으로 각각 16개의 단계(step) 동작으로 구성된다. 그림에서 CV는 [그림 2]에 보는 바와 같이 초기 벡터(IV, initial vector) 또는 이전 해쉬 동작의 결과(CV_i)를 나타낸다. MD5 알고리즘의 각 라운드를 구성하는 하나의 단계(step)는 [그림 4]와 같이 4개의 32 비트 워드(A, B, C, D)를 받아서, 4개의 덧셈 연산과 좌측 순환 이동(CLS, cyclic left shift) 동작을 한 후, 기존 4개의 32 비트 워드(A, B, C, D)를 갱신하는 동작으로 이루어져 있다. [그림 4]에서 함수 g는 [표 1]과 같이 각 라운드별로 구별된 논리 함수를 구현한다. 표 1에서 $\wedge$, $\vee$, $\oplus$ 는 각각 bitwise AND, bitwise



〔그림 4〕 MD5알고리즘의 단계(step) 연산

〔표 1〕 g 함수 기능

Round	Primitive function, g	g(b, c, d)
1	F(b,c,d)	(b∧c)∨(b'∧d)
2	G(b,c,d)	(b∧d)∨(c∧d')
3	H(b,c,d)	b⊕c⊕ d
4	I(b,c,d)	c⊕(b∨d')

OR, bitwise XOR 동작을 나타낸다. X[k]는 512 비트(16 워드) Y에서 선택된 워드를 의미하며 라운드 별로 사용하는 워드 순서가 다르다. T[i]는 2^{32} × *abs(sin(i))*의 정수 값을 가지며, 각 라운드는 64개의 T[i] 중에 16개를 사용한다. 자세한 단계별 동작은 MD5 표준안[3]에 기술되어 있다.

Ⅲ. MD5 알고리즘을 구현하는 해쉬 프로세서 설계

본 장에서는 MD5 해쉬 알고리즘을 효율적으로 구현하기 위해 고려한 설계 사양, 연산 기법과 하드웨어 설계를 기술한다.

3.1 설계 사양

MD5 해쉬 프로세서 설계 시 고려한 사항은 다음과 같다.

첫째, 암호 프로세서는 외부 호스트 프로세서에 대한 보조 프로세서 형태로 설계되어, 다양한 호스트 환경에 접속이 가능하며, 소프트(soft) IP(intellectual property) 형태로 활용할 수 있도록 한다.

둘째, 입력 메시지의 사전 처리 작업은 호스트 컴퓨터에서 처리하도록 하여, 본 연구의 해쉬 프로세서는 512 비트 사전 처리된 메시지에대해 128 비트 중간 결과 형태의 해쉬 값을 생성하는 역할을 담당한다.

셋째, 외부 입출력 인터페이스 방식으로 8, 16, 32 비트 병렬 인터페이스를 지원한다.

넷째, 입력 레지스터와 출력 레지스터를 별도로 두어 입출력 동작이 해쉬 동작과 병행적으로 이루어지도록 하여, 입출력 시간이 연산 시간에 포함되는 것을 배제하도록 하여 성능을 향상시켰다.

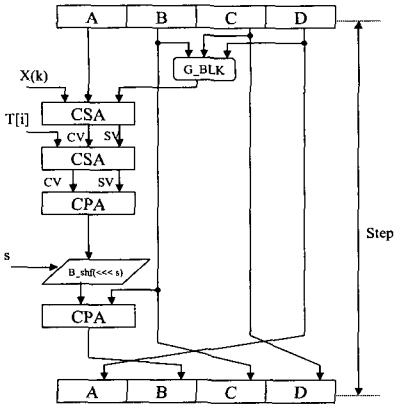
3.2 MD5 해쉬 프로세서의 성능 개선을 위한 연산 기법

3.2.1 캐리 보존 덧셈을 활용한 MD5 단계 동작의 지연 시간 최소화 기법

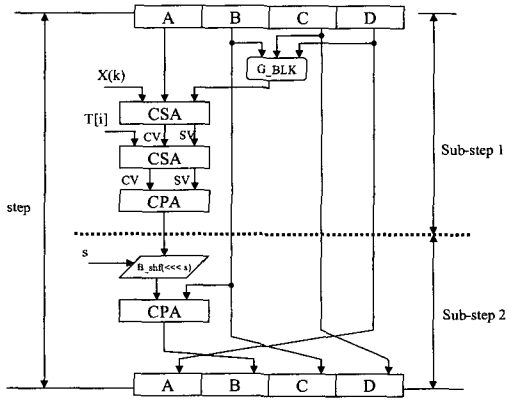
[그림 4]의 MD5 알고리즘의 단계(step) 동작을 분석해 보면, 단계 동작은 4개의 덧셈 연산과 1개의 좌측 순환 회전 동작으로 구성된다. 그런데 좌측 순환 이동 동작 전에 위치한 3개의 순차적인 캐리 전달 덧셈(CPA, carry propagate addition) 연산은 2개의 캐리 보존 덧셈(CSA, carry save addition)과 1개의 캐리 전달 덧셈(CPA)으로 대체가 가능하다. 이러한 대체 과정을 통해서 3개의 32 비트 캐리 전달 가산기 연산 시간은 2개의 전가산기와 1개의 32 비트 캐리 전달 덧셈 시간으로 대체가 가능하다. 단, 좌측 회전 이동 동작에 대해 캐리 전달(carry save) 개념을 적용하는 방안도 가능하지만, 이것은 복잡한 배선과 2개의 배럴 시프터가 필요하므로 배제하였다. [그림 5]는 MD5의 단계(step) 연산에 대해 캐리 전달 덧셈 기법을 적용한 경우를 나타낸다. [그림 5]의 경우 배럴 시프터 이전의 4개의 오퍼랜드 캐리 전달 덧셈 연산이, 2개의 캐리 전달 덧셈과 FA(full adder) 지연 시간을 갖는 2개의 캐리 보존 덧셈 연산 동작으로 대체되어 [그림 4]에 비해 동작 주파수가 약 2 배 향상될 수 있다.

3.2.2 MD5 단계 동작의 다중 사이클 처리 기법

[그림 5]의 회로의 경우 단일 단계를 1 개의 클록으로 구현할 경우 2개의 캐리 보존 덧셈 연산, 2개의 캐리 전달 덧셈, 1개의 순환 좌측 이동 동작이 필요하다. 따라서 본 연구에서는 하드웨어를 공유하고 동작 주파수를 증가시키기 위해서, 1개의 단계 연산 동작을 [그림 6]과 같이 2개의 부분 단계로 구분하는



〔그림 5〕 MD5 단계 연산에 캐리 보존 덧셈을 적용한 연산 기법

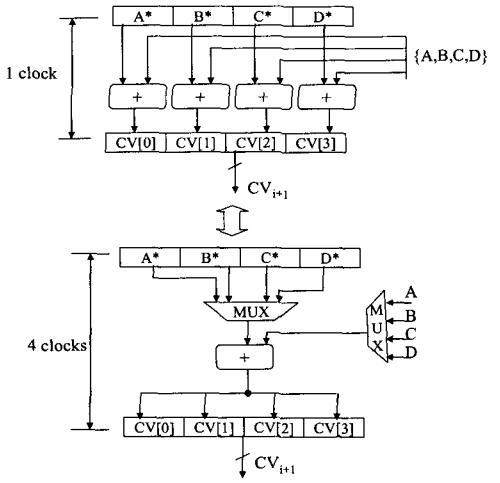


〔그림 6〕 단일 단계를 2개의 부분 단계로 분할하는 기법

기법을 채택하였다. 2개의 부분 단계간 지연 시간을 비슷하게 조정하기 위해서 순환 회전 이동 동작을 담당하는 배럴 시프터를 중심으로 2개의 부분 단계를 나누었다. 그리고 각각의 부분 단계에 대해 한 개의 클록을 배당하여 2개의 클록 시간 동안 하나의 단계 동작을 처리하도록 하였다. 이러한 기법을 채택할 경우 그림 5 방식에 비해 가산기를 공유할 수 있어서 단계 연산을 위한 면적은 약 1/2로 감소하며, 동작 주파수는 상대적으로 2배 만큼 향상될 수 있어서 연산 시간의 증가가 거의 발생치 않는다.

3.2.3 하드웨어 공유를 통한 모듈러 덧셈 하드웨어 감소 기법

[그림 3]의 MD5 알고리즘 동작을 보면 4개의 라운드 동작 후에 마지막으로 4개의 32 비트 모듈러 덧셈 동작이 필요하다. 그런데 이러한 동작을 위해 4개의 32 비트 가산기를 배당하는 것은 하드웨어 낭

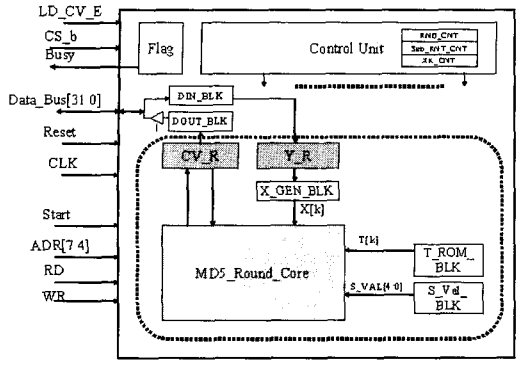


〔그림 7〕 모듈러 가산기의 하드웨어 공유

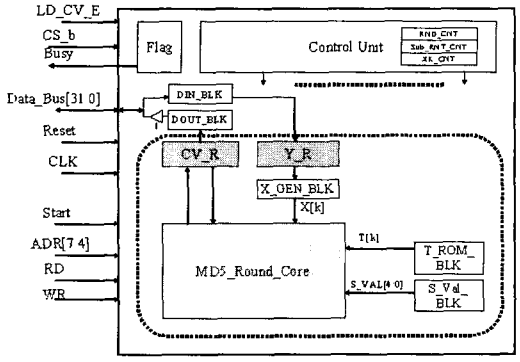
비가 심하다. 그리고 마지막 모듈러 덧셈 동작이 4개의 라운드 동작과 시간적으로 동시에 발생하지 않는다는 점을 고려하여, [그림 7]과 같이 32 비트 캐리 전달 가산기(CPA)를 4회 반복하여 사용하는 방안을 채택하였다. 단, 이러한 32 비트 가산기는 별도의 하드웨어가 아닌 [그림 6]의 기존 MD5 단계 연산에서 사용하는 32 비트 캐리 전달 가산기(CPA)를 활용하여 구현할 수 있다. 따라서 이러한 기법을 적용할 경우 하드웨어 공유를 사용하지 않는 방식이 129(=4×16×2+1) 클록 사이클이 필요한데 비해, 마지막 모듈러 덧셈을 하드웨어 공유 기법으로 구현할 경우 132(=4×16×2+4) 클록이 필요하므로, 기존 방식에 비해 MD5 해쉬 연산을 위해 3 클록이 추가로 필요하지만 가산기를 하드웨어 공유해서 활용할 수 있다는 장점이 있다. 그리고 추가로 필요한 3 사이클은 전체 연산 동작에 차지하는 비율이 작기 때문에 성능 저하는 크지 않다.

3.3 MD5 해쉬 프로세서의 하드웨어 설계

3.2절에서 설정한 설계 사양을 만족하며 단계 연산의 2 사이클 처리, 캐리 보존 덧셈과 하드웨어 공유 기법을 통해 동작 주파수와 면적을 개선한 해쉬 프로세서의 블록도는 [그림 8]과 같다. [표 2]는 입출력 핀의 기능을 나타낸다. 외부의 데이터 버스를 통해 사전 처리된 512 비트 입력 메시지를 Y 레지스터(Y-R)에 저장한다. SR(status register)는 외부 입출력 인터페이스의 데이터 크기를 제어한다. 해쉬 동작과 외부 입출력 동작을 동시에 수행할 수 있도록 하여



〔그림 8〕 MD5 해쉬 프로세서 블록도

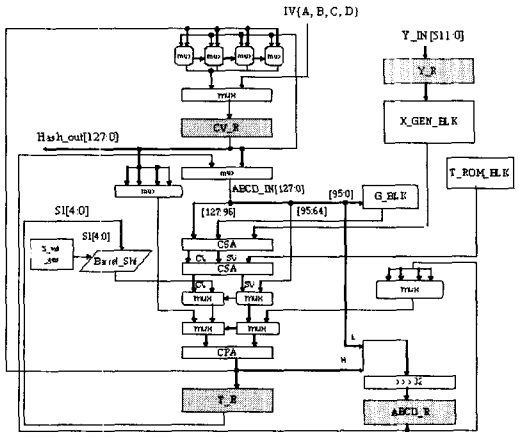


〔그림 8〕 MD5 해쉬 프로세서 블록도

〔표 2〕 MD5 해쉬 프로세서의 입출력 핀의 기능

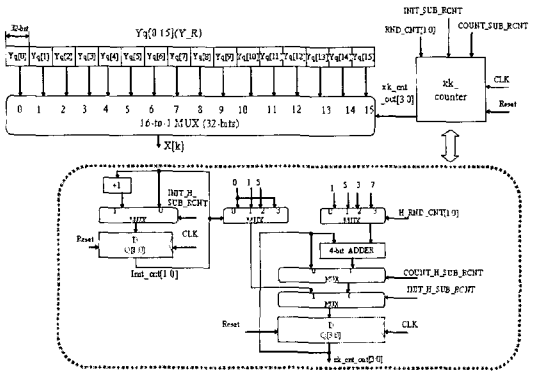
신 호	입·출력	기 능
CS_b	입력	칩 선택 신호
Busy	출력	해쉬 동작 진행 중 의미
Data_Bus[31:0]	양방향	데이터 버스
Reset	입력	초기화 신호
Clk	입력	시스템 클록
Start	입력	동작 개시신호
Adr[7:0]	입력	주소 버스
RD	입력	읽기 신호
WR	입력	쓰기 신호
LD_CV_E	입력	CV_R을 IV로 초기화

입출력 시간에 따른 성능 저하를 방지하기 위해, 입력 버퍼 레지스터(IBR)와 내부 해쉬 코어의 메시지 레지스터(Y-R)을 분리시켰다. 즉, 해쉬 동작 중에 입력 버퍼 레지스터를 통해 입력 동작을 수행할 수 있도록 하였으며, 새로운 512 비트 메시지에 대한 해쉬 동작에 대한 동작 개시(start) 신호 발생 시 IBR 레지



〔그림 9〕 MD5 해쉬 라운드 코어의 블록도

스터에 저장된 외부 입력 데이터를 메시지 레지스터(Y-R)로 이동시키는 역할을 수행한다. 해쉬 프로세서 내부 레지스터는 호스트 프로세서에서 제공하는 주소로 메모리 맵(memory-mapped)되어, 호스트 프로세서의 읽기와 쓰기 동작에 반응하도록 되어 있다. MD5 동작 중 가변 메시지의 첫 번째 512 비트 메시지(Y0)의 첫 번째 라운드의 경우 고정된(hardwired) IV(initial value)값을 사용하는데 비해, 다음에 이어지는 512 비트 메시지(Yi, i≠0)의 경우 CV 레지스터에 저장되어 있는 이전 메시지(Yi-1)의 해쉬 결과를 사용하므로, 이러한 두 동작의 초기 값 구별을 위해 Y0 메시지의 경우 CV 레지스터를 IV 값으로 초기화 시켜주는 작업이 필요한데, LD_CV_E 신호가 이러한 기능을 담당한다. 그리고 해쉬 동작 중에는 Busy 신호가 "1"이 되어, 해쉬 동작이 진행중임을 알려주게 된다. 해쉬 라운드 코어는 3장 2절에서 제안된 연산 기법을 바탕으로 [그림 9]의 구조를 갖고 있다. 단, 좌측 회전 이동을 구현하는 배럴 시프터는 퍼널 시프터(funnel shifter) 구조를 갖고 있다. 그림에서 32 비트 T_R 레지스터는 단계 연산의 중간 결과를 유지하며, ABCD_R은 단계 연산 결과와 라운드 연산 결과를 유지하며, CV_R은 최종 메시지 다이제스트(message digest), 즉 해쉬 값 과 Y0의 라운드 동작시 초기값 IV를 저장하는 역할을 수행한다. T_ROM_BLK, S_Val_Blk, X_gen_Blk 블록은 각각 단계 동작의 T(i), S(i), X(k)를 생성하는 블록이다. 특히 X_gen_Blk의 경우 16개의 워드로 구성된 Y값에서 적절한 워드 값을 선택해서 X(k) 값으로 제공하는 기능을 수행한다. [그림 10]은 X(k)값을 생성하는 X_gen_Blk 블록을 나타내며, Y_R에서 적절한 워드를 선택하는



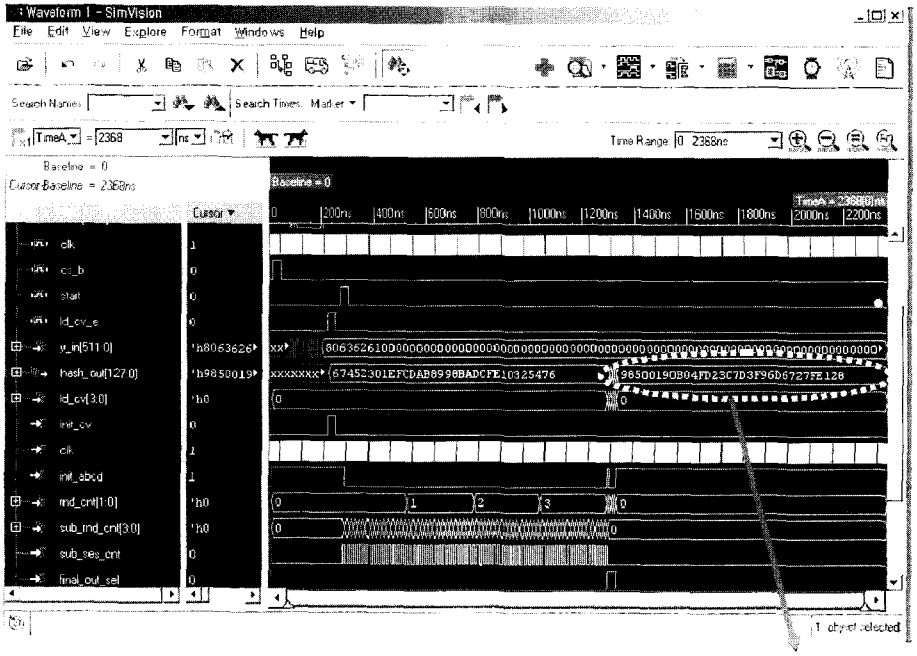
〔그림 10〕 X(k)를 생성하는 X gen_BLK의 블록도

16-to-1 Mux회로와 Mux를 제어하는 Xk 카운터로 구성된다. Xk 카운터는 크게 라운드 별로 구별되는 Y의 시작 워드 인덱스를 결정하는 카운터와 단계마다 워드 인덱스 값을 조정하는 모듈러 증가 기능을 갖는 카운터로 구성된다. 그리고 제어 회로 내부에는 라운드 동작과 각 단계에 따른 X(k)와 T(k) 생성 제어를 위한 3개의 카운터, 즉 라운드 카운터, 부분 라운드 카운터, X_K 카운터가 사용된다.

제어 회로는 필요한 입출력 동작을 수행하는 입출력 제어부 와 외부에서 제공하는 시작 신호를 받아서 해쉬 알고리즘을 구현하는 주 제어부로 나뉜다. 주 제어부는 FSM(finite state machine)부, 카운터부, 제어신호 생성부로 구성된다. 코어 제어 회로 설계는 데이터패스의 동작 흐름을 ASM(Algorithmic State Machine) 차트로 표현한 후 이를 제어 회로로 변환하는 방법을 사용하였다.

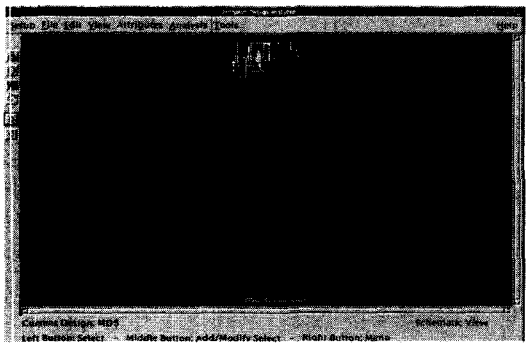
IV. 설계 검증과 성능 분석

MD5 알고리즘을 구현하는 프로세서를 설계하기 위해, 먼저 MD5 암호 알고리즘을 C 언어로 모델링한 후, 이를 Verilog HDL[11] 언어로 변환한 후, NC-Verilog 시뮬레이터[12]를 모의 검증을 수행하여 2가지 동작이 일치하는지 확인하는 과정을 사용하였다. 시뮬레이션을 통해 표준안에서 제공한 테스트 데이터를 만족함을 확인하였다. 이러한 기능 검증 후에 설계한 회로를 Synopsys사의 Design Analyzer[13] 소프트웨어와 0.25 μm CMOS 라이브러리[14]를 사용하여 합성하였다. 합성한 결과 해쉬 프로세서의 최장 동작 경로의 지연시간은 약 8.2ns로서, 최대 동작 주파수



Message Digest 결과

〔그림 12〕 MD5 프로세서에 대한 NC-Verilog 게이트 레벨 시뮬레이션 결과(테스트 입력 메시지 = "abc", CV_R에 담긴 big-endian 형식에 따른 결과 값 = 0x98500190b04fd23c7d3f96d6727fe128 little-endian 형식의 최종 결과 = 0x900150983cd24fb0d6963f7d28e17f72)



〔그림 11〕 Synopsys Design Compiler로 합성된 MD5 프로세서

는 120 Mhz이며 총 게이트 수는 약 13,000임을 확인하였다. [그림 11]은 Synopsys Design Analyzer로 합성한 결과를 나타낸다. 그리고 합성 과정에서 추출한 지연 정보를 포함한 게이트 수준 타이밍 시뮬레이션 결과는 [그림 12]와 같다. 입력메시지로 "abc"를 사용한 경우 MD5 해쉬 프로세서에 대한 시뮬레이션 결과로 MD5 표준안[3]에서 명시한 결과를 만족함을 확인하였다. 단, 시뮬레이션 과정에서 MD5 알고리즘이 데이터를 저장하는 방식으로 little-endian 방식을 따르므로, 하드웨어에 메시지 값을 할당할 때 주의가 필요하다. 즉 MD5 알고리즘이 채택하고 있는 little-endian 방식의 경우 32 비트 워드 내에 바이트가채워지는 순서가 가장 낮은 바이트부터 높은 바이트 순으로 저장하는 구조를 갖고 있는데 비해, 해쉬 라운드 코어의 내부 레지스터(Y_R, IBR, CV_R)는 big-endian 방식으로 상위 바이트에서 낮은 바이트 순으로 레지스터에 채워져서 내부 코어의 덧셈의 캐리 전달 동작에 적합하도록 되어 있다. 이러한 데이터

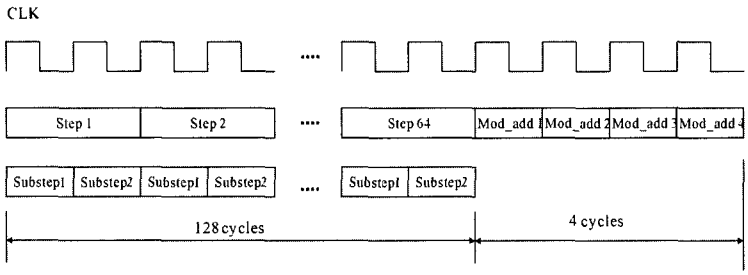
〔표 3〕 MD5 프로세서의 전기적 특성

지원 암호 알고리즘	MD5
단계 당 클록 수	2
전체 클록 수	132
동작 주파수	120 Mhz
게이트 수 (2-input NAND 기준)	약 13,000
입력 방식	background input
외부 인터페이스	8 / 16 / 32-bit
해쉬 생성율	465 Mbps
전력 소모	67.7 mW
공정 기술	0.25 um CMOS 표준셀
전원	2.5 Volt

저장 형식의 차이로 외부에서 제공하는 512 비트 입력 데이터는 해쉬 프로세서내 IBR에 저장될 때 바이트 단위의 주소값(Adr[7:0])을 조정하여 big-endian 형태로 상위 바이트가 워드 내에 최상위 위치에 저장된다. 그리고 유사하게 최종 결과 값으로 CV_R에 담겨지는 해쉬 값은 big-endian 형식으로 얻어지므로 호스트 프로세서에서 읽어 갈 때 주소 조정을 통해 little-endian 형태로 외부 메모리에 저장하게 된다. [그림 13]은 설계한 MD5 프로세서에 대한 타이밍 분석을 나타낸다. 최종 모듈로 덧셈이 라운드 하드웨어내의 캐리 전달 가산기(CPA)를 활용하기 때문에 4 사이클이 소요된다. [그림 13]에 따라 512 비트 메시지에 대한 MD5 해쉬 프로세서의 성능은 식 (2)과 같다. 식(2)에서 N이 132이고, f가 120 Mhz이므로 본 논문에서 설계한 해쉬 프로세서의 성능은 약 465 Mbps임을 알 수 있다.

〔표 4〕 MD5 해쉬 알고리즘의 구현 방식의 면적과 연산 시간 비교
(T=32 비트 캐리 전달 가산기 지연시간, 1/4 · T=배럴 시프터 지연 시간)

방 식	면적(CPA 수)		배럴 시프터수	성능 (클럭 수)	클록 주기	연산시간 (사이클 수 × 클록 주기)
	단계 처리부	최종 모듈러 덧셈부				
단계 당 1 클록 방식 (CPA 사용) ([8] / iterative)	4	4	1	65	4.25T	276T
단계 당 1 클록 방식 (CSA사용) [7]	2	4	1	65	2.25T	146T
Full Loop Unrolled+ CSA 방식 ([8] / Full Loop Unrolled)	2*64 =128	4	0 (hardwired)	1	128T	129T
단계 당 2 클록 방식 (CSA사용: 본 논문)	1	0	1	132	1.25T	165T



〔그림 13〕 MD5 프로세서의 타이밍 분석도

〔표 5〕 MD5 해쉬 프로세서의 성능 비교

	참고 문헌[7]	참고문헌[8] /Iterative	참고문헌[8] /Full Loop unrolling	참고문헌 [10]	참고문헌 [9]	본 논문
지원알고리즘	MD5 (SHA-1,HAS-160)	MD5	MD5	MD5	MD5	MD5
단계 연산당 클럭수	1	1	-	-	1	2
해쉬 동작에 필요한 사이클 수	65 @MD5	65	1	-	65	132
성능(Mbps)	142 @MD5	165	354	87~100	1,030	465
동작 주파수	18 Mhz @MD5	21 Mhz	0.714 Mhz	-	133 Mhz	120 Mhz
게이트수	16,604 logic element	161 IOB 2 RAM 880 slices	161 IOB 0 RAM 4,763 slices	소프트웨어 구현	19,200	13,000
사용공정	Altera EP20K1000 EBC652-3	Xilinx VirtexV1000 FG680-6	Xilinx VirtexV1000 FG680-6	DEC Alpha (190Mhz)	0.13um CMOS	0.25um CMOS

$$MD5 \text{ 해쉬 프로세서의 성능} = (512 \div (N)) \times f \quad (2)$$
여기서 f 주파수, N은 사이클 수

[표 3]은 설계한 해쉬 프로세서의 전기적 특성을 나타내며, [표 4]는 본 연구의 MD5 해쉬 프로세서에서 사용한 방식과 여러 가지 MD5 알고리즘의 구현 방안을 구조적인 측면에서 면적과 연산 시간을 결과이다. 단, 연산 시간을 정량화하기 위해 시뮬레이션 데이터를 기준으로 32 비트 CPA 연산을 T로, 배럴 시프터 지연 시간을 (1/4)T로 설정하였다. [그림 4]와 같이 단계 당 단일 클록을 사용하는 방식의 경우, 연속된 4개의 덧셈 연산으로 본 논문의 방식에 비해 지연 시간이 4 배이므로 많은 하드웨어에 비해 성능이 크게 떨어짐을 알 수 있다. 반면 그림 5와 같이 단계 당 단일 클록을 사용하며 캐리 보존 덧셈 연산을 하는 방식은 본 연구에 비해 반복 횟수(클럭 수)는 1/2로 감소되는데 비해, 2배에 가까운 면적과 클럭 주기를 갖게 된다. 따라서 본 연구에서 설계한 방식은 면적과 속도의 곱(AT)을 성능 척도로 사용할 경우 최적임을 알 수 있다. 그리고 이론적으로 가장 작은 연산 시간은 Full Loop Unrolling 방식과 CSA 기법이 결합될 경우 얻어 질 수 있지만, 지나치게 많은 면적이 필요하다는 문제가 있다. [표 5]는 본 논문에서 구현한 해쉬 프로세서와 기존에 발표된 해쉬 프로세서와의 성능 비교를 나타낸다. 참고 문헌 [8]의 경우 2가지 방식(iterative, full loop unrolling)으로 구현되어 있다. 참고 문헌 [10]에 따르면 DEC Alpha 워크스테이션에서 소프트웨어로 구현한 MD5 알고리즘은 87~100 Mbps 정도의 성능을 갖고 있으므로, 100 Mbps 이하의 낮은 성능이 필요할 경우에는, FPGA 구현 방식과 성능 차이가 크지 않으므로 소프트웨어 구현이 타당함을 알 수 있다. 참고 문헌 [8]의 Full Loop Unrolling 방식의 경우 64개의 단계를 하나의 클록으로 구현함에 따라 몇 가지 장점을 얻을 수 있다. 즉 상수 값을 내부 하드웨어에 고정배선할 수 있어서, 상수를 저장하기 위한 메모리가 제거되며, 배럴 시프터도 하드웨어가 아닌 고정 배선으로 구현 할 수 있어서, 참고 문헌 [8]의 iterative 구조에 비해 면적 증가량이 loop unrolling의 정도에 비례하지 않고, 높은 연산 성능을 얻을 수 있음을 보여 주

고 있다. 반면 참고 문헌 [9]의 경우 본 논문의 구현 방식에 비해 높은 성능을 보여 주고 있지만 상대적으로 우수한 공정을 사용하고 있음에 기인하며, 사용 공정의 2배 개선 효과를 고려 할때 본 논문의 MD5 프로세서는 참고 문헌 [9]와 유사한 성능을 가질 것으로 판단된다. 또한 면적 비교 척도가 될 수 있는 게이트 수는 본 연구 방식이 참고 문헌 [9]에 비해 1.5배 우수함을 알 수 있다. 이러한 특성을 고려할 때 본 연구의 해쉬 프로세서는 무결성 과 인증 모듈을 필요로 하는 보안 프로세서에 면적과 성능 효율적인 IP(Intellectual Property) 형태의 보조 프로세서로 장착될 수 있을 것으로 판단된다.

V. 결 론

본 논문에서는 면적과 속도 측면에서 효율적인 MD5 해쉬 프로세서를 설계하였다. 설계한 해쉬 보조 프로세서는 1 단계 동작을 2개의 부분 단계로 분할 처리, 캐리 보존 덧셈 연산의 최적 사용을 통한 최장 전달 경로의 지연시간을 최소화하고, 하드웨어 공유를 극대화시켜, 기존 라운드 당 1 클록 방식에 비해 약 1/2의 하드웨어 크기를 가지며 연산 시간은 유사한 특성을 갖고 있다. 따라서 면적과 속도의 곱(AT)을 성능 척도로 할 경우 본 연구에서 설계한 MD5 해쉬 프로세서는 효율적인 구조를 갖고 있음을 알 수 있다. 또한 외부 호스트 컴퓨터와 암호 프로세서사이의 입출력 동작과 암호 프로세서 동작을 병렬로 수행함으로써, 입출력 시간에 따른 성능 저하 문제를 제거하였다. 또한 외부 호스트 컴퓨터와의 인터페이스로 8, 16, 32 비트를 지원함에 의해서 다양한 데이터 버스를 갖는 호스트 프로세서에 접목이 가능한 특성을 갖고 있다. 설계한 해쉬 프로세서는 약 13,000개의 게이트로 구성되며, 0.25 μm CMOS 공정에서 약 120 Mhz의 동작 주파수를 가지고 있다. 그리고 128 비트 해쉬 값 생성에 132 클록이 소요되며, 512 비트 이하의 메시지의 경우 해쉬 생성율은 465 Mbps이다. 본 연구에서 설계한 해쉬 보조 프로세서는 적은 면적으로 상대적으로 높은 해쉬 생성율을 제공하므로, 해쉬 알고리즘이 적용되는 네트웍, 전자상거래 시스템 등의 IP 형태로 가공되어 보안 모듈로 사용될 수 있을 것으로 판단된다.

참 고 문 헌

[1] 박창섭, 암호 이론과 보안, 대영사, 1999.

[2] M. Mcloone and John V. McCanny, "A Single Chip IPSEC Cryptographic Processor", IEEE Workshop on Signal Processing System 2002(SIPS'02), pp.133~138, October 2002.

[3] R.L. Rivest. "The MD5 message digest algorithm", Request for Comments(RFC) 1320, Internet Activities Board, Internet Privacy Task Force, April, 1992.

[4] NIST, "Secure hash standard", FIPS 180-1, Washington, D.C. April, 1995.

[5] Bart Preneel, "The Cryptographic Hash Function RIPEMD-160", CryptoBytes, Vol.3, No.2, pp.9~14, 1997.

[6] 한국 정보 통신 기술 협회, "해쉬 함수 표준-제2부: 해쉬 함수 알고리즘 표준(HAS-160)", TTAS. KO-12.0011/R1, 2000

[7] Yong Kyu Kang, Dae Won Kim, Taek Won Kwon, and Jun Rim Choi, "An Efficient implementation of Hash Function Processor for IPSEC", Asia Pacific Conference on ASIC 2002, pp.93~96, August, 2002.

[8] Jonaka Deepakumara, "FPGA Implementation of MD5 Hash Algorithm", Canadian Conference on Electrical and Computer Engineering, vol.2, pp.13~16, May 2001.

[9] SCI-WORX, "MD5 High Speed MD5 Hash Engine", http://www.sci-worx.com/internet/homepage_internet_e.html?/internet/startpage_internet_e.html

[10] J. Touch, *Report on MD5 Performance,* RFC 1810, June 1995.

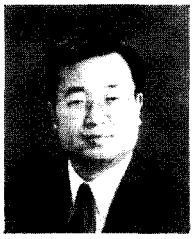
[11] Tomas and Moorby, *The Verilog Hardware Description Language,* 4th Edition, KAP, 1998.

[12] Cadence, NCLaunch User Guide, 2002

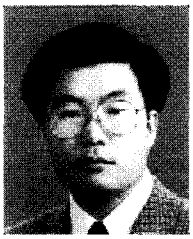
[13] IDEC 설계 교육 센터, *Synopsys Design Analyzer Tool, 교육 강좌 자료,* 1999.4.

[14] SEC ASIC, "MDL110 : 0.25μm 2.5V CMOS Standard Cell Library for Pure Logic/MDL Products", 1999

〈著者紹介〉



최 병 윤 (Byeong-Yoon Choi) 종신회원
1985년 2월 : 연세대학교 전자공학과 졸업
1987년 2월 : 연세대학교 전자공학과 석사
1992년 8월 : 연세대학교 전자공학과 공학 박사
1997년 8월~1998년 8월 : 일리노이 주립대, visiting professor
1993년 3월~현재 : 동의대학교 교수
<관심분야> RISC 및 Java 마이크로프로세서 설계, 암호 및 정보 통신용 VLSI 설계



박 영 수 (Young-Soo Park) 정회원
1990년~현재 : 한국 전자 통신 연구원 선임 연구원
<관심분야> CAD 및 VLSI 설계, 암호 프로세서 설계, IC 카드 설계