고성능 가산기의 최적화 연구

준회원 허 석 원, 김 문 경, 정회원 이 용 주, 이 용 석

Study of Optimization for High Performance Adders

Seok-Won Heo, Moon-Gyung Kim *Associate Members*
Yong-Joo Lee, Yong-Surk Lee *Regular Members*

요 약

본 논문에서는 단일 클락 사이클과 다중 클락 사이클에 수행되는 여러 가산기를 구현하고 area와 time을 비교한다. 가산기의 크기를 64, 128, 256-비트로 다양화 시키면서, 특히 하이브리드 구조의 가산기는 소그룹을 4, 8, 16-비트로 나누어서 group / ungroup으로 합성을 하여 비교하였다. 제안된 가산기들은 Verilog-HDL을 이용하여 하향식 설계 방법으로 구현되었다. Cadence의 Verilog-XL을 이용하여 설계된 가산기와 behavioral model을 이용한 가산기의 출력이 일치하는지를 비교하여 검증하였다. 검증된 모델은 삼성 0.35um 3.3(V) CMOS standard cell 라이브러리를 이용하여 합성되었으며, 최악 조건 2.7(V), 85(℃)에서 동작하였다. 스마트 카드 IC의 Crypto-Processor에 사용할 수 있는 최적화된 가산기는 64-비트를 기준으로 할 때, group으로 합성된 16-비트 캐리 예측 가산기를 기반으로 하는 리플 캐리 가산기(RCA_CLA)이다. 이 가산기는 198(MHz)의 속도로 동작하며, 게이트 수는 nand2 게이트 기준으로 약 967개이다.

ABSTRACT

In this paper, we implement single cycle and multi cycle adders. We can compare area and time by using the implemented adders. The size of adders is 64, 128, 256-bits. The architecture of hybrid adders is that the carry-out of small adder groups can be interconnected by utilizing a carry propagate unit. The size of small adder groups is selected in three formats - 4, 8, 16-bits. These adders were implemented with Verilog HDL with top-down methodology, and they were verified by behavioral model. The verified models were synthesized with a Samsung 0.35(um), 3.3(V) CMOS standard cell library while a using Synopsys Design Compiler. All adders were synthesized with group or ungroup. The optimized adder for a Crypto-processor included Smart Card IC is that a 64-bit RCA based on 16-bit CLA. All small adder groups in this optimized adder were synthesized with group. This adder can operate at a clock speed of 198 MHz and has about 967 gates. All adders can execute operations in this worst case conditions of 2.7 V, 85 ℃.

Key Word : 가산기, 단일 클락 사이클, 다중 클락 사이클, area, cycle time, 합성, HDL

Ⅰ. 서 론

일반적으로 자주 이용되는 모듈이 전체 성능에 큰 영향을 준다. 통계적으로 80X86 마이크로프로세서에서 SPECint92 프로그램으로 벤치마킹을 하였을 경우 명령어의 83%가 가산기(adder)를 사용하고 있다. 전체 명령어 중에 가산(혹은 감산) 명령어는 13%, 메모리 접근(memory-access) 명령어는 34%, 분기

*연세대학교 전기전자공학과 프로세서연구실(comace@dubiki.yonsei.ac.kr)
논문번호 : 030411-0917, 접수일자 : 2003년 9월 17일
※ 본 연구는 하이닉스반도체의 "Flexible Cryptographic Engine" 프로젝트 지원으로 수행되었음

(branch) 명령어는 36%가 가산 명령어를 사용하고 있다[1]. 그러므로 가산기의 성능이 RISC 마이크로프로세서의 성능에 매우 중요한 영향을 준다고 할 수 있다.

현재까지 가산기에 관한 연구는 다양한 방법으로 연구되었다.

비동기식 가산기들의 최악 지연 시간(worst delay)과 평균 지연 시간(average delay)에 관한 연구가 발표되었다. 이 논문에서 하이브리드(hybrid) 구조의 캐리 선택 가산기(Carry Select Adder)가 리플 캐리 가산기(Ripple Carry Adder)보다 32-비트 비동기식 RISC 마이크로프로세서에서 17%, 64-비트 마이크로프로세서에서 23%의 성능 향상을 보였다[2].

하이브리드 구조를 이용한 멀티 레벨 캐리 예측 가산기(carry look-ahead adder)와 캐리 스킵 가산기(carry skip adder)는 각각의 소그룹 블락 크기가 다른 가산기가 같은 가산기보다 15~25% 성능이 뛰어난 것으로 발표되었다[3][4][5].

여러 종류의 수퍼파이프라인(superpipelined) 가산기들을 설계하고, stage 개수와 latch, gate의 개수를 분석한 논문에서는 파이프라인 stage가 적을수록 설계 비용(design cost)과 면적이 작아지는 것을 보였다[6][7].

현재까지 다양한 구조와 방법으로 가산기의 area 및 delay가 연구되었다. 그러나 단일 클락 사이클과 다중 클락 사이클 가산기에서 소그룹의 크기 차이와 group 및 ungroup 합성에 따른 time 및 area가 비교된 적이 없었다.

그러므로, 본 논문에서는 단일 클락 사이클과 다중 클락 사이클 가산기들의 성능 및 면적을 분석하며, 디지털 시스템에 최적화된 가산기 구조를 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 단일 클락 사이클 구조와 다중 클락 사이클 구조의 차이점에 관하여 비교한다. 3장에서는 본 논문에서 여러 가산기의 알고리듬을 설명 및 구현한다. 4장에서는 구현된 가산기를 합성 및 비교한다. 5장에서는 디지털 시스템에 최적화된 가산기를 제안하며, 6장에서 결론을 짓는다.

Ⅱ. 단일 클락 사이클 구조와 다중 클락 사이클 구조 비교

1. 단일 클락 사이클 구조

단일 클락 사이클 구조는 한 클락 사이클 내에 모든 작업이 처리되는 구조이다. 가장 긴 명령어에 의하여 수행 시간이 결정되므로 성능이 감소되고 연산 모듈을 중복해야 하므로 하드웨어를 비효율적으로 사용하는 단점을 가진다. 그러나 구현이 간단하고, 임시 데이터를 저장할 저장 회로(latch)가 필요 없다. 연산 결과를 한 클락 사이클에 이용할 수 있으므로 연산 결과가 나오지 않아서 다른 모듈이 기다릴 필요가 없다.

2. 다중 클락 사이클 구조

다중 클락 사이클 시스템은 연산 과정을 여러 개의 작은 단계로 나누어서 연산을 가속화하는 방법이다. 한 번의 클락 사이클에 연산을 실행하는 큰 모듈을 설계하는 대신 여러 번의 클락 사이클에 연산을 실행하는 작은 모듈을 설계하는 구조이다. 여러 번의 클락 사이클에 연산하기 위하여 임시 데이터를 저장하는 latch가 필요하다.

Ⅲ. 제안된 가산기의 알고리듬 및 구현

1. 단일 클락 사이클 구조의 가산기

단일 클락 사이클 구조의 가산기는 직렬 구조, 트리 구조, 하이브리드 구조 등 크게 3가지로 분류할 수 있다. 직렬 구조 가산기는 최악 조건 지연 시간이 O(n)일 경우에 면적이 O(n)이 되는 특징을 가진다. 즉, 최악 조건 지연 시간과 면적이 정비례하는 특징을 가진다. 트리 구조 가산기는 최악 조건 지연 시간이 크게 감소되는 고성능 가산기 구조이다. 이 구조의 가산기는 O(logn)의 지연 시간을 가지는 장점을 가지고 있지만, O(n logn)의 큰 면적을 갖는 단점을 가지고 있다. 하이브리드 구조의 가산기는 직렬 구조와 트리 구조의 중간 성능을 갖는다. 하이브리드 구조는 n-비트 블락으로 나누어서 병렬적으로 연산한다. 하이브리드 구조의 가산기는 면적이 O(n)이지만, 지연 시간은 $O(\sqrt{n})$ 이 된다[1]. 면적은 직렬 가산기와 동일하지만, 지연시간이 더 빠른 장점을 가진다.

단일 클락 사이클 구조의 가산기는 7가지 구조를 제안하였다. 직렬 구조의 리플 캐리 가산기와(RCA) 트리 구조의 캐리 예측 가산기(CLA)를 제안하였다. 하이브리드 구조의 캐리 예측 가산기를 기반으로 하는 리플 캐리 가산기(RCA_CLA), 리플 캐리 가산기를 기반으로 하는 캐리 예측 가산기(CLA_RCA), 캐리 예측 가산기를 기반으로 하는 캐리 선택 가산기

(CSA_CLA), 리플 캐리 가산기를 기반으로 하는 캐리 선택 가산기(CSA_RCA)를 제안하였다.

1) 직렬 구조 가산기

리플 캐리 가산기(Ripple Carry Adder)

리플 캐리 가산기는 가장 기본적인 방법으로 전가산기(full adder)를 직렬적으로 연결하여 구성하는 것이다[8].

전가산기는 세 개의 입력(X_i, Y_i, Ci(carry- in))과 두 개의 출력(S_i, C_{i+1}(carry-out))을 가진다.

$$S_i = X_i \oplus Y_i \oplus C_i \quad (1)$$

$$C_{i+1} = X_i \cdot Y_i + Y_i \cdot C_i + C_i \cdot X_i \quad (2)$$

이 가산기는 쉽게 구현이 가능하고, 면적이 작은 장점을 가지지만, 결과를 연산하는데 필요한 시간이 선형적으로 증가하므로, 고성능 하드웨어에 이용할 수 없다. 리플 캐리 가산기는 전가산기를 직렬 연결하여 구현을 하였다. 그림 1은 1-비트 전가산기와 4-비트 리플 캐리 가산기의 구조이다.

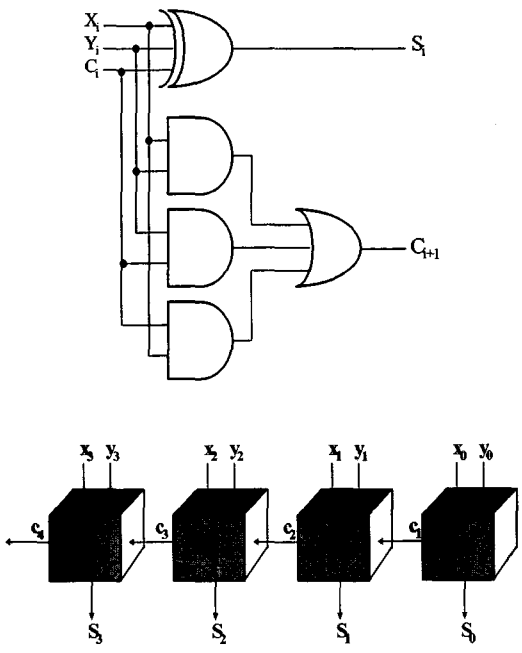


그림 1. 1-비트 전가산기와 4-비트 리플 캐리 가산기
Fig. 1. A Full adder and 4-bit Ripple Carry Adder

2) 트리 구조 가산기

캐리 예측 가산기(Carry Look-ahead Adder)

캐리 예측 가산기는 하위 비트의 캐리 출력을 기다릴 필요 없이 입력이 결정되면 빠른 시간에 연산의 결과를 얻을 수 있다는 장점을 가지고 있다. 그러나 입력의 비트가 클 경우 fan-in이 많아지고 면적이 커지는 단점을 가지고 있다. 이러한 단점을 극복하기 위하여, 가산기를 여러 그룹으로 나눈 후에 그 그룹이 더 작은 그룹을 갖는 멀티 레벨 구조를 갖는 가산기가 많이 이용되고 있다[9][10].

캐리 생성 신호(G_i)를 $G_i = X_i \cdot Y_i$, 캐리 전달 신호(P_i)를 $P_i = X_i + Y_i$ 라고 정의하면,

$$\begin{aligned} C_{i+1} &= X_i \cdot Y_i + Y_i \cdot C_i + C_i \cdot X_i \\ &= X_i \cdot Y_i + C_i \cdot (X_i + Y_i) \\ &= G_i + C_i \cdot P_i \end{aligned} \quad (3)$$

(3)식에 $C_i = G_{i-1} + C_{i-1} \cdot P_{i-1}$ 을 대입하면 다음과 같이 나타낼 수 있다.

$$C_{i+1} = G_i + G_{i-1} \cdot P_i + C_{i-1} \cdot P_{i-1} \cdot P_i \quad (4)$$

$$C_{i+1} = G_i + G_{i-1} \cdot P_i + G_{i-2} \cdot P_{i-1} \cdot P_i + \cdots + C_0 \cdot P_0 \cdots P_i \quad (5)$$

캐리 예측 가산기는 두가지 방법으로 구현을 하였다. 첫 번째 방법은 fan-in의 제한을 하지 않은 방법이고, 두 번째 방법은 fan-in을 4개로 제한한 방법이다. 일반적인 게이트에서는 복잡성 때문에 fan-in을 8개 이하로 제한하고 있다[11]. 그림 2는 4-비트 캐리 예측 가산기의 구조이다.

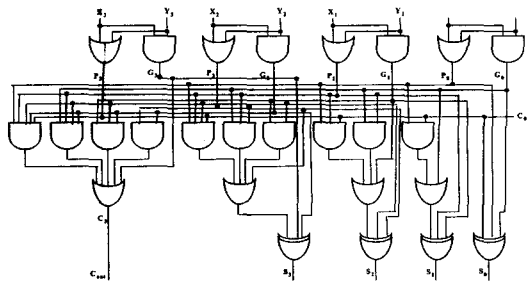


그림 2. 4-비트 캐리 예측 가산기
Fig. 2. A 4-bit Carry Look-ahead Adder

3) 하이브리드 구조 가산기

캐리 선택 가산기(Carry Select Adder)

캐리 선택 가산기는 입력되는 캐리의 값이 '0' 또는 '1'의 값을 가지므로 이 두 가지 경우에 대해 덧셈을 미리 수행한다. 최하위 블락의 캐리 출력이 결정되면, 이 결과를 이용하여 미리 계산한 결과들 중의 하나를 선택한다. 캐리 선택 가산기는 병렬적으로 연산을 하기 때문에, 리플 캐리 가산기보다 약 2배의 면적을 사용하지만, 약 4배의 속도를 향상시킬 수 있다[7]. 캐리 선택 가산기는 멀티플렉서를 통과하는 시간을 고려하여 마지막 그룹으로 갈수록 많은 비트를 할당할 수 있다. 본 논문에서는 그룹별 비트가 다를 경우 너무 많은 경우의 수가 발생할 수 있으므로 각 그룹별 비트수를 동일하게 처리하였다. 그림 3은 8-비트 캐리 선택 가산기의 구조이다.

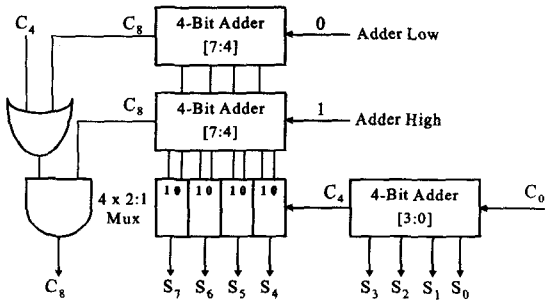


그림 3. 8비트 캐리 선택 가산기
Fig. 3. A 8-bit Carry Select Adder

하이브리드 구조 가산기 구현

하이브리드 구조 가산기는 소그룹의 캐리 출력을 캐리 전달 유닛으로 연결하여 연산한 후 캐리 전달 유닛의 출력을 다음 소그룹의 캐리 입력으로 연결하는 구조이다. 소그룹은 리플 캐리 가산기, 캐리 예측 가산기의 두 가지 구조이며, 캐리 전달 유닛은 리플 캐리 가산기, 캐리 예측 가산기, 캐리 선택 가산기 등 세 가지 구조를 가지고 있다. 소그룹은 4, 8, 16-비트로 나누어서 연산을 할 수 있다. 그림 4는 하이브리드 구조의 4개 소그룹으로 나누어진 16-비트 가산기의 구조이다.

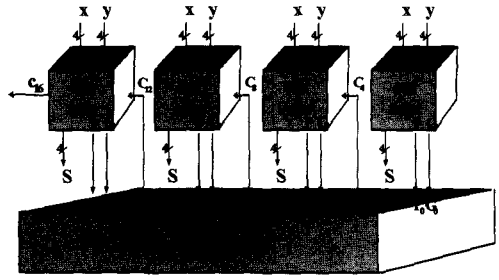


그림 4. 16-비트 하이브리드 구조 가산기
Fig. 4. A 16-bit Adder using hierarchical structure

4) Behavioral model

Verilog HDL의 '+' operator를 이용하여 구현하였다

2. 다중 클락 사이클 구조의 가산기

다중 클락 사이클 구조의 가산기는 latch를 두어서 임시 연산 결과를 저장하는 구조를 가지고 있다.

다중 클락 사이클 구조는 입력 비트가 클 경우 사용할 수 있다.

다중 클락 사이클 구조의 가산기는 6가지 종류로 제안하였다. 클락(clock)이 발생할 때, 캐리 저장 가산기(Carry Save Adder)에 임시 출력이 저장되는 구조를 갖는 리플 캐리 가산기를 기반으로 하는 캐리 저장 가산기(CSA_RCA), 캐리 예측 가산기를 기반으로 하는 캐리 저장 가산기(CSA_CLA)를 설계하였다. 가산기의 면적을 줄이기 위하여 멀티플렉서와 디멀티플렉서를 이용한 가산기와 멀티플렉서와 쉬프터를 이용한 가산기를 설계하였다.

1) 캐리 저장 가산기를 이용한 가산기

캐리가 모든 전가산기마다 발생하는 것이 아니라 일정한 크기의 리플 캐리 가산기(캐리 예측 가산기)마다 발생하도록 만든 구조이다. 예를 들어 만약 리플 캐리 가산기를 이용한다면 몇 개의 전가산기를 묶어서 리플 캐리 가산기를 만든 후 클락이 입력될 때마다 하나의 리플 캐리 가산기에서 하나의 캐리를 발생시키는 구조이다. 이러한 구조는 모든 전가산기마다 캐리를 발생시키는 기존의 캐리 저장 가산기 구조보다 발생할 수 있는 캐리가 적어 보다 적은 클락으로 캐리를 처리할 수 있다[12]. 그림 5는 캐리 저장 가산기를 이용한 가산기의 구조이다.

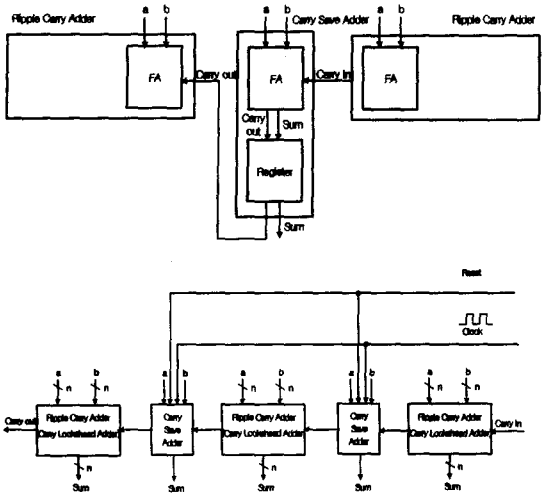


그림 5. 캐리 저장 가산기를 이용한 가산기
Fig. 5. A multi-cycle adder using Carry Save Adder

2) 멀티플렉서와 디멀티플렉서를 이용한 가산기

캐리 저장 가산기를 이용한 가산기는 가산기가 클락 사이클 수만큼 사용된다. 이러한 단점을 극복하기 위하여 본 가산기는 가산기를 한 번만 사용하도록 설계되었다. 멀티플렉서(multiplexor)를 이용하여 n/a 비트를 가산기로 보낸다. (n은 전체 입력 비트, a는 클락 사이클 수이다.) 가산기에서 연산을 수행한 후, 디멀티플렉서(demultiplexor)를 이용하여 전체 출력의 하위 비트 부분을 출력한다. 그 후에 클락 카운터(clock counter)가 입력 후 클락의 개수를 세서 멀티플렉서의 선택 신호를 보낸다. 멀티플렉서에서 다음 입력을 가산기로 보내고 가산기에서 연산을 하여 디멀티플렉서로 출력하는 구조이다. 본 구조는 가산기의 면적이 작아지는 장점을 가진다. 그러나 디멀티플렉서에서 레지스터를 가져야 하므로 저장 공간의 면적이 커지는 단점을 가진다. 그림 6은 멀티플렉서와 디멀티플렉서를 이용한 가산기의 구조이다.

3) 멀티플렉서와 쉬프터를 이용한 가산기

본 가산기 구조는 앞의 구조에서 디멀티플렉서를 쉬프터(shifter)로 대체한 구조이다. 디멀티플렉서와 쉬프터는 레지스터의 크기는 같다. 그러나 디멀티플렉서는 각 레지스터의 wire가 연결되어야 한다. 따라서 클락 사이클 수가 많아질수록 컨트롤러(controller)가 복잡해지고, wire가 많아지는 단점을 가진다. 쉬프터는 가산기에서 연산된 비트만큼 오른쪽으로 쉬프트하면 되므로 컨트롤러가 간단해지고, wire가 작아지는 장점을 가진다. 그림 7은 멀티플렉서와 쉬프터를 이용한 가산기의 구조이다.

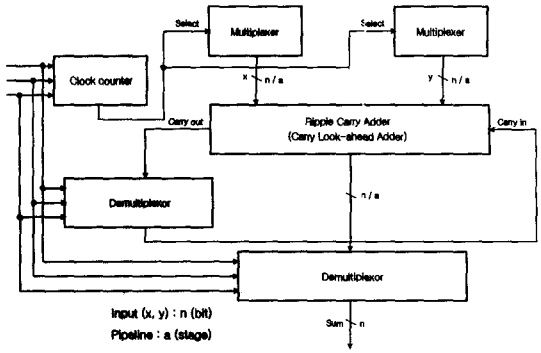


그림 6. 멀티플렉서와 디멀티플렉서를 이용한 가산기
Fig. 6. A multi-cycle adder using multiplexor and demultiplexor

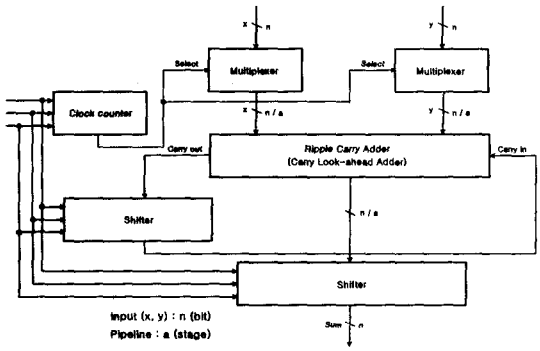


그림 7. 멀티플렉서와 디멀티플렉서를 이용한 가산기
Fig. 7. A multi-cycle adder using multiplexor and demultiplexor

Ⅳ. 합성 결과

가산기를 두가지 방법으로 합성하였다. 첫 번째 방법은 하위 모듈들을 group을 한 후에 합성을 하였다. 두 번째 방법은 ungroup을 한 후에 합성을 하였다.

모든 가산기는 Verilog-HDL을 이용하여 하향식 설계 방식(top-down methodology)으로 설계되었다. 가산기의 검증은 Cadence의 Verilog-XL을 이용하여 설계된 가산기의 출력과 '+' operator를 이용한 가산기의 출력이 일치하는지를 테스트 벡터 1,000,000개로 진행하였다. 삼성 0.35um 3.3(V) CMOS standard cell 라이브러리를 이용하여 group / ungroup으로 합성을 하였다. 모든 가산기는 2.7(V), 85(℃) 최악 조건에서 동작한다. 면적은 nand2 gate 기준으로 report하였다.

표 1. 단일 사이클 구조의 가산기 비교 (Group으로 합성)
Table 1. Single-cycle Adder Comparison by Group

adder	base (bit)	64 (bit)		128 (bit)		256 (bit)	
		time (ns)	area (nand2)	time (ns)	area (nand2)	time (ns)	area (nand2)
RCA	-	22.28	749	44.56	1551	88.46	2966
CLA	-	3.59	1091	4.56	1962	4.85	4401
'+' operator	-	3.47	1168	3.79	2625	2.97	5534
RCA_CLA	4	12.66	856	23.33	1708	50.55	3344
	8	7.99	846	14.59	1744	26.49	3491
	16	5.05	967	7.97	1782	13.76	3596
CLA_RCA	4	22.37	801	42.96	1483	50.09	2856
	8	21.85	799	43.39	1466	86.24	2914
	16	21.89	759	43.46	1471	86.45	2943
CSA_CLA	4	7.78	1983	14.49	3878	29.14	7933
	8	4.96	1895	8.29	3989	15.51	8454
	16	3.95	1976	5.55	3993	9.16	8250
CSA_RCA	4	8.38	1330	15.11	2657	29.62	5757
	8	6.45	1507	9.98	2724	17.11	5533
	16	9.07	1531	10.98	3188	14.97	5782

표 2. 단일 사이클 구조의 가산기 비교 (Ungroup로 합성)
Table 2. Single-cycle Adder Comparison by Ungroup

adder	base (bit)	64 (bit)		128 (bit)		256 (bit)	
		time (ns)	area (nand2)	time (ns)	area (nand2)	time (ns)	area (nand2)
RCA	-	3.34	1567	4.28	2251	6.19	5267
CLA	-	3.46	1403	4.26	2505	5.07	4557
'+' operator	-	3.47	1168	3.79	2625	2.97	5534
RCA_CLA	4	8.84	991	5.38	2192	6.82	4283
	8	3.67	1387	5.16	2517	5.45	4985
	16	3.38	1441	4.35	2576	5.64	4551
CLA_RCA	4	3.37	1506	4.79	2543	5.12	5147
	8	3.58	1498	6.87	2198	7.78	4699
	16	6.62	1193	9.23	2329	6.62	4936
CSA_CLA	4	3.52	1116	4.26	2057	4.88	4226
	8	3.91	1101	4.54	2228	5.47	3989
	16	3.68	1163	4.00	2489	6.58	5030
CSA_RCA	4	5.10	1158	8.01	2164	13.71	4335
	8	6.34	1195	9.10	2266	15.30	4070
	16	6.04	1371	6.76	2555	11.51	4054

RCA 리플 캐리 가산기
CLA 캐리 예측 가산기
RCA_CLA 캐리 예측 가산기 기반의 리플 캐리 가산기
CLA_RCA 리플 캐리 가산기 기반의 캐리 예측 가산기
CSA_CLA 캐리 예측 가산기 기반의 캐리 선택 가산기
CSA_RCA 리플 캐리 가산기 기반의 캐리 선택 가산기

1. 단일 사이클 구조의 가산기 합성 결과

합성한 결과 중에서 ungroup은 cycle time을 감소(최대 93.00% 감소)시키고 area를 증가(최대 80.25%)시켰다. ungroup 결과 특히 리플 캐리 가산기(RCA)와 리플 캐리 가산기 기반의 캐리 예측 가산기(CLA_RCA)는 상당한 cycle time 감소를 보였다(최대 93.00% 감소, 최소 69.76% 감소).

그러나 캐리 선택 가산기(CSA)를 사용한 가산기는(CSA_RCA, CSA_CLA) ungroup 시에 면적이 감

표 3. 다중 사이클 구조의 가산기 비교 (Group으로 합성)
Table 3. Multi-cycle Adder Comparison by Group

adder	base (bit)	64 (bit)				128 (bit)				256 (bit)			
		stage	time[1]	time[2]	area	stage	time[1]	time[2]	area	stage	time[1]	time[2]	area
CSA_RCA	4	16	1.97	31.52	762	32	2.05	65.60	1600	64	2.11	135.04	3136
	8	8	3.12	24.96	745	16	3.30	52.80	1415	32	3.28	104.96	2867
	16	4	6.12	24.48	739	8	6.18	49.44	1487	16	6.27	100.32	2692
CSA_CLA	4	16	1.62	25.92	915	32	1.60	51.20	1786	64	1.65	105.60	3535
	8	8	2.17	17.36	1188	16	2.29	36.64	1993	32	2.27	72.64	3943
	16	4	2.82	11.28	1148	8	3.06	24.48	2197	16	3.22	51.52	4233
DMX_RCA	4	16	5.70	91.20	1395	32	8.24	263.68	2847	64	15.91	1018.24	5419
	8	8	6.01	48.08	1275	16	6.96	111.36	2433	32	9.87	315.84	4722
	16	4	7.58	30.32	1118	8	8.53	68.24	2236	16	10.41	166.56	4102
DMX_CLA	4	16	6.43	102.88	1444	32	9.13	292.16	2833	64	16.79	1074.56	5872
	8	8	5.00	40.00	1288	16	6.30	100.80	2322	32	7.59	242.88	4664
	16	4	4.98	19.92	1175	8	5.63	45.04	2251	16	6.65	106.40	4546
SFT_RCA	4	16	4.77	76.32	1206	32	5.23	167.36	2059	64	5.51	352.64	3980
	8	8	5.81	46.48	974	16	6.10	97.60	2073	32	6.39	204.48	3686
	16	4	7.66	30.64	1065	8	8.24	65.92	1937	16	8.57	137.12	3376
SFT_CLA	4	16	4.92	78.72	1222	32	5.32	170.24	2151	64	5.64	360.96	3922
	8	8	5.15	41.20	1205	16	5.44	87.04	1910	32	5.68	181.76	3768
	16	4	5.27	21.08	1144	8	6.01	48.08	2065	16	5.85	93.60	3611

표 4. 다중 사이클 구조의 가산기 비교 (Ungroup으로 합성)
Table 4. Multi-cycle Adder Comparison by Ungroup

adder	base (bit)	64 (bit)				128 (bit)				256 (bit)			
		stage	time[1]	time[2]	area	stage	time[1]	time[2]	area	stage	time[1]	time[2]	area
CSA_RCA	4	16	1.77	28.32	1054	32	1.73	55.36	2089	64	1.78	113.92	4175
	8	8	2.34	18.72	1240	16	2.35	37.60	2147	32	2.50	80.00	4602
	16	4	2.71	10.84	1304	8	3.01	24.08	2437	16	3.19	51.04	4982
CSA_CLA	4	16	1.45	23.20	1107	32	1.60	51.20	2193	64	1.61	103.04	4475
	8	8	1.90	15.20	974	16	2.44	39.04	2070	32	2.52	80.64	4224
	16	4	2.66	10.64	1021	8	2.73	21.84	2227	16	2.78	44.48	4464
DMX_RCA	4	16	5.91	94.56	1401	32	7.33	234.56	2899	64	17.40	1113.60	5432
	8	8	5.24	41.92	1451	16	7.51	120.16	2377	32	7.54	241.28	4711
	16	4	5.30	21.20	1351	8	5.86	46.88	2348	16	6.78	108.48	4327
DMX_CLA	4	16	5.72	91.52	1367	32	10.43	333.76	2576	64	14.69	940.16	5767
	8	8	5.08	40.64	1239	16	5.65	90.40	2621	32	7.65	244.80	4508
	16	4	4.89	19.56	1377	8	5.56	44.48	2405	16	6.08	97.28	4643
SFT_RCA	4	16	4.33	69.28	1225	32	4.27	136.64	1808	64	5.06	323.84	4334
	8	8	5.27	42.16	1217	16	5.84	93.44	2192	32	5.62	179.84	3838
	16	4	5.16	20.64	1276	8	9.08	72.64	2178	16	5.61	89.76	4176
SFT_CLA	4	16	4.06	64.96	1423	32	4.22	135.04	1947	64	4.90	313.60	4334
	8	8	4.44	35.52	1274	16	4.73	75.68	2188	32	5.16	165.12	3716
	16	4	4.81	19.24	1293	8	5.09	40.72	2213	16	5.33	85.28	4124

CSA_RCA 리플 캐리 가산기 기반의 캐리 저장 가산기
CSA_CLA 캐리 예측 가산기 기반의 캐리 저장 가산기
DMX_RCA 멀티플렉서와 디멀티플렉서를 이용한 리플 캐리 가산기
DMX_CLA 멀티플렉서와 디멀티플렉서를 이용한 캐리 예측 가산기
SFT_RCA 멀티플렉서와 쉬프터를 이용한 리플 캐리 가산기
SFT_CLA 멀티플렉서와 쉬프터를 이용한 캐리 예측 가산기

time[1] cycle time
time[2] execution time

소된 것이 보였다(최대 52.82% 감소, 최소 10.45% 감소). 대부분의 가산기에서 ungroup시 cycle time이 감소하고 area가 증가하였다. 그러나 캐리 선택 가산기(CSA)를 사용한 가산기는 ungroup시 cycle time과 area 모두 감소하였다.

소그룹이 캐리 예측 가산기인 경우(RCA_CLA, CSA_CLA) 소그룹의 크기가 4-비트인 경우 ungroup

이 group보다 cycle time의 급격한 감소를 보이다가 8, 16-비트로 가면서 비슷해지는 현상을 보였다.

'+' operator는 area와 cycle time의 결과가 캐리 예측 가산기와 비슷하게 나타났다.

캐리 예측 가산기(CLA)는 fan-in을 4개로 제한한 구조나 fan-in을 제한하지 않은 구조에서 비슷한 결과가 발생하였다. 따라서 본 논문에서는 fan-in을 제한하지 않은 구조로 결과를 제시하였다.

리플 캐리 가산기(RCA)는 ungroup으로 합성할 경우 캐리 예측 가산기(CLA)와 비슷한 area와 cycle time을 갖는다.

합성시 ungroup을 사용하면 대부분의 가산기에서 비슷한 cycle time과 면적이 측정되었다.

그림 8과 9는 파이프라인이 되지 않은 가산기의 cycle time과 area를 비교한 그림이다.

1) Group 합성 결과

합성한 결과 중에서 '+' operator가 가장 빠른 결과를 나타내었다. 모든 가산기가 입력이 증가함에 따라서 cycle time이 증가하였음에도 불구하고 '+' operator는 cycle time이 입력이 64-비트에서 128-비트로 증가함에 따라서 22% 증가, 128-비트에서 256-비트로 증가하는 것에 21.64% 감소하였다. area가 가장 작은 가산기는 64-비트에서 리플 캐리 가산기(RCA)와 128, 256-비트에서 리플 캐리 가산기 기반의 캐리 예측 가산기(CLA_RCA)로 나타났다. 리플 캐리 가산기(RCA)와 리플 캐리 가산기 기반의 캐리 예측 가산기(CLA_RCA)는 대부분 같은 area를 보였지만, 256-비트에서 소그룹의 크기를 4-비트로 하면 리플 캐리 가산기 기반의 캐리 예측 가산기(CLA_RCA)가 리플 캐리 가산기(RCA)보다 큰 cycle time의 감소(43.38%)를 얻을 수 있었다.

소그룹이 캐리 예측 가산기(CLA)인 경우 상대적으로 cycle time이 빠르게 측정되었다(최대 84.08% cycle time 감소). 그리고 소그룹의 비트 크기가 커질수록 cycle time이 크게 감소되었다.

그러나 소그룹이 리플 캐리 가산기(RCA)인 경우 area가 작았다(최대 34.56% 면적 감소, 최소 6.43% 면적 감소).

소그룹이 캐리 선택 가산기(CSA)인 경우 중복된 구조로 인하여 가산기 중에 가장 큰 area를 가졌다.

2) Ungroup 합성 결과

Ungroup 합성 결과는 area보다 cycle time에 최적화된 결과가 나타났다.

캐리 예측 가산기를 기반으로 하는 리플 캐리 가산기(RCA_CLA)는 소그룹의 크기가 증가할수록 cycle time이 감소되고, area가 증가되는 결과가 나타났다. 그러나 리플 캐리 가산기를 기반으로 하는 캐리 예측 가산기는 소그룹의 크기가 증가할수록 cycle time이 증가하고, area가 감소되는 상반된 결과가 나타났다. 캐리 선택 가산기가 포함된 가산기(CSA_RCA, CSA_CLA)는 소그룹이 8-비트인 경우에 cycle time이 가장 큰 결과가 발생하였다. 이러한 결과는 입력의 크기가 64, 128-비트에서는 적용되지만 256-비트가 되면 위의 결과가 적용되지 않는다.

2. 다중 사이클 구조의 가산기 합성 결과

쉬프터를 이용한 가산기(SFT_RCA, SFT_CLA)나 캐리 저장 가산기(CSA)를 이용한 가산기(CSA_RCA, CSA_CLA)는 입력 비트 크기에 따른 cycle time의 변화가 거의 없었다.

그러나 디멀티플레서(DMX)를 사용한 가산기(DMX_RCA, DMX_CLA)는 입력 비트 크기에 따른 cycle time의 변화가 크다. 특히 소그룹이 4-비트인 경우 cycle time이 크게 증가하고, 소그룹의 크기가 8, 16 비트로 증가할수록 cycle time이 감소한다. 이러한 결과는 입력 비트가 커질수록 cycle time의 증가 감소 차이가 크게 나타난다.

소그룹이 리플 캐리 가산기(RCA)인 것에서 크기가 4-비트인 것은 입력이 256-비트로 증가하면 cycle time이 증가하는 것을 볼 수 있다.

리플 캐리 가산기 기반의 캐리 저장 가산기(CSA_RCA)는 ungroup이 area가 급격하게 증가한다(최대 85.08 % 증가, 최소 38.38% 증가).

캐리 예측 가산기 기반의 캐리 저장 가산기(CSA_CLA)는 소그룹의 크기가 4에서 16으로 증가함에 따라서 group과 ungroup의 area 차이가 점차 감소한다.

그림 10과 11은 다중 사이클 구조 가산기의 cycle time과 area를 비교한 그림이다.

연산의 execution time은 식 (6)과 같다[13].

$$\text{execution time} = \text{clock cycle} \times \text{clock cycle} \quad (6)$$

파이프라인 가산기의 execution time을 비교하면, 소그룹의 크기가 4-비트일 때 입력 비트가 증가함에 따라서 execution time이 증가하는 것을 알 수 있다.

특히 디멀티플렉서(DMX)를 사용하는 가산기는 execution time이 매우 크게 증가한다(64-비트 ⇒ 128-비트 : 최대 122.44%, 128-비트 ⇒ 256-비트 :

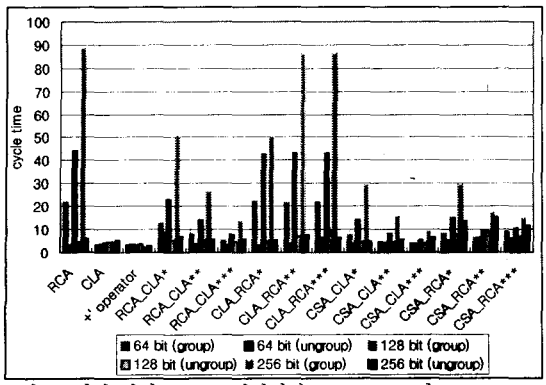


그림 8. 단일 사이클 구조 가산기의 cycle time 비교
Fig. 8. Cycle time comparison in single-cycle adder

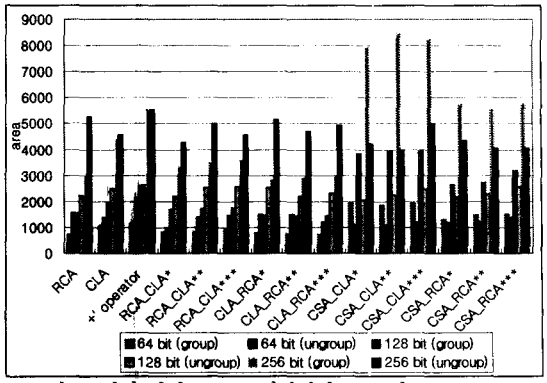


그림 9. 단일 사이클 구조 가산기의 area 비교
Fig. 9. Area comparison in single-cycle adder

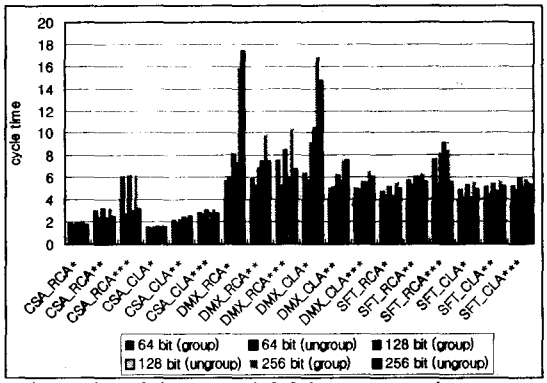


그림 10. 다중 사이클 구조 가산기의 cycle time 비교
Fig. 10. Cycle time comparison in multi-cycle adder

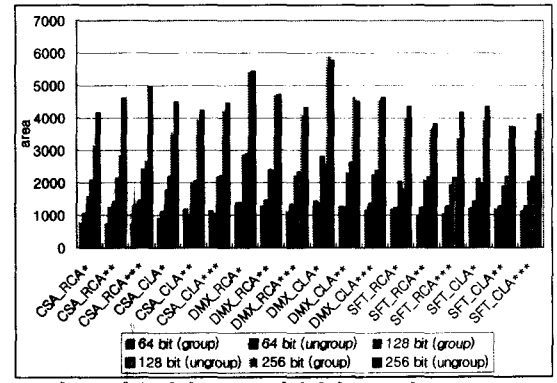


그림 11. 다중 사이클 구조 가산기의 area 비교
Fig. 11. Area comparison in multi-cycle adder

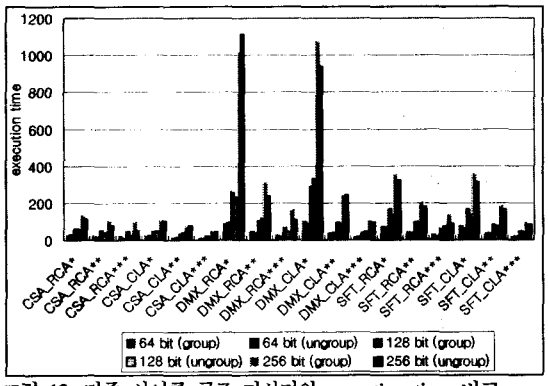


그림 12. 다중 사이클 구조 가산기의 execution time 비교
Fig. 12. Execution time comparison in multi-cycle adder

최대 374.76%). 그러나 캐리 저장 가산기(CSA)를 이용한 가산기는 입력 비트의 크기와 소그룹 크기의 관계없이 비슷한 execution time을 가지는 것을 알 수 있다.

그림 12는 다중 사이클 구조 가산기의 execution time을 비교한 그림이다.

* base 4 비트
** base 8 비트
*** base 16 비트

1) Group 합성 결과

디멀티플렉서(DMX)를 사용한 가산기는 group으로 합성할 때 소그룹의 크기가 작아질수록 cycle time과 area가 증가하는 현상이 보인다. 이것은 소그룹의 크기가 감소할수록 가산기의 area가 감소되지만 디멀티플렉서의 area가 증가되기 때문이다.

쉬프터(SFT)를 사용한 가산기는 group으로 합성할 때 소그룹의 크기가 작아질수록 cycle time이 감소되고 area가 증가한다. 소그룹의 크기가 작아질수록 쉬프터의 area가 증가하지만 가산기의 area가 감소되기 때문이다.

위의 두 결과를 비교하면 디멀티플렉서(DMX)를 사용한 소그룹의 크기가 작아질수록 cycle time이 증가하고, 쉬프터(SFT)를 사용한 가산기는 cycle time이 감소하는 현상을 볼 수 있다. 이것을 보면

가산기는 쉬프터가 디멀티플렉서보다 제어 부분이 간단한 것을 알 수 있다.

2) Ungroup 합성 결과

리플 캐리 가산기를 기반으로 하는 캐리 저장 가산기(CSA_RCA)는 소그룹의 크기가 작아질수록 cycle time과 area가 감소되는 것이 보인다. 이것은 group으로 합성할 때 소그룹의 크기가 작아질수록 cycle time은 감소하지만 area가 증가하는 것과 대비되는 결과이다.

캐리 예측 가산기를 기반으로 하는 캐리 저장 가산기(CSA_CLA)는 소그룹의 크기가 8인 경우에 area가 최소화되었다.

디멀티플렉서(DMX)를 사용한 가산기는 소그룹의 크기가 16-비트일 때 cycle time과 area가 가장 작았다. 그러나 쉬프터(SFT)를 사용한 가산기는 그룹의 크기가 4-비트일 때 cycle time이 가장 작았다.

V. 최적화된 가산기의 적용 사례

본 논문에서 제시한 합성 결과는 가산기를 사용하는 많은 디지털 시스템에 사용될 수 있다.

1. 암호화 프로세서(Crypto-processor)

하이닉스 반도체에서 구현한 스마트 카드 IC에는 ARM 프로세서, Crypto 프로세서(ECC 모듈, RSA 모듈, 레지스터 파일) 등이 사용된다. ARM 프로세서는 40 MHz의 클락 주파수를 가지며 RSA 모듈 및 ECC 모듈도 같은 속도의 클락 주파수를 가져야 한다. ARM 프로세서가 한 번 실행될 때, RSA 모듈 안에 Big Montgomery 프로세서 가산기는 네 번 실행이 되어야 한다. 그러므로 가산기는 160 MHz 이상의 클락 주파수를 가져야 한다. 위의 클락 주파수의 조건을 갖춘 가산기 중에서 제조 원가를 줄이기 위하여 area가 가장 작은 가산기를 선택해야 한다. 그 결과 Big Montgomery 프로세서 안에서 사용될 최적화된 가산기는 64-비트를 기준으로 할 때, group으로 합성된 16-비트 캐리 예측 가산기를 기반으로 하는 리플 캐리 가산기(RCA_CLA)이다. 이 가산기는 198 MHz의 속도로 동작하며, nand2 게이트 기준으로 약 967의 area를 갖는다.

그림 13은 최적화된 가산기가 적용될 Crypto 프로세서와 RSA 모듈의 구조이다.

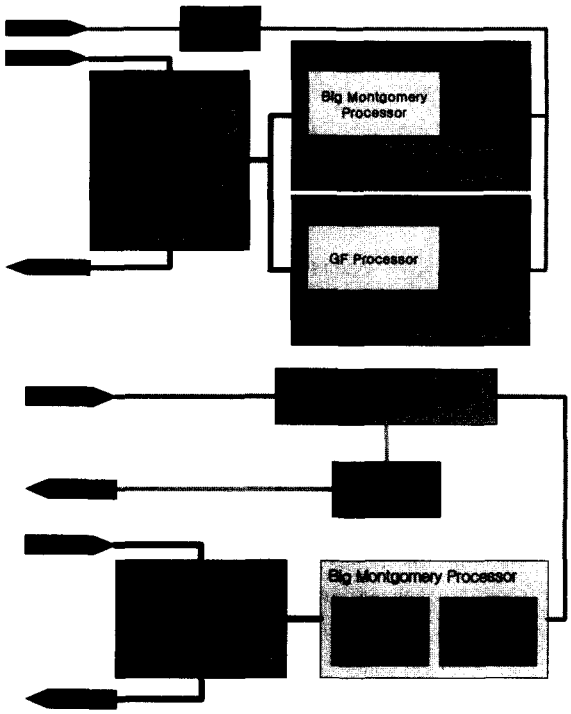


그림 13. Crypto 프로세서와 RSA module 구조
Fig. 13. Crypto-Processor and RSA module Architecture

Ⅳ. 결 론

본 논문에서는 단일 사이클 구조 가산기와 다중 사이클 구조 가산기를 여러 가지 알고리듬으로 구현하였다.

단일 사이클 구조 가산기에서 ungroup으로 합성한 결과는 cycle time을 감소시키고 area를 증가시켰다. 그러나 캐리 선택 가산기(CSA)를 사용한 가산기는 ungroup으로 합성할 경우 cycle time과 area 모두 감소하였다.

다중 사이클 구조 가산기는 쉬프터를 이용한 가산기나 캐리 저장 가산기를 이용한 가산기는 입력 비트 크기에 따른 cycle time의 변화가 거의 없다. 그러나 디멀티플레서를 사용한 가산기는 입력 비트 크기에 따른 cycle time의 변화 폭이 크게 나타났다.

본 논문은 제안된 가산기의 cycle time과 area를 비교함으로써 최적화된 가산기를 여러 디지털 시스템에 응용할 수 있을 것이다.

참 고 문 헌

[1] J. L. Hennessy and D. A. Patterson, "*Computer Architecture : A Quantitative Approach, third edition*", Morgan Kaufmann Publishers, CA, 2003.

[2] Mark. A. Franklin and Tienyo Pan, "Performance Comparison of Asynchronous Adders", *Advanced Research in Asynchronous Circuits and Systems, 1994. Proceedings of the International Symposium on 3-5*, pp. 117-125, Nov. 1994.

[3] P. K. Chan, M. D. F. Schlag, C.D. Thomborson and V. G. Oklobdzija, "Delay Optimization of Carry-Skip Adders and Block Carry-Lookahead Adders", *Proc. 10th Symp. on Computer Arithmetic*, pp. 154- 164, 1991

[4] M. Lehman and N. Burla, "Skip Techniques for High-Speed Carry-Propagation in Binary Arithmetic Units", *IRE Transactions on Electronic Computers, vol. EC-10*, pp. 691-698, Dec. 1961.

[5] S. Majerski, "On Determination of Optimal Distribution of Carry Skips in Adders", *IEEE Transactions on Electronic Computers, vol. EC-16*, pp. 45-48, Feb. 1967.

[6] Ishaq H. Unwala, Earl E. Swartzlander, Jr, "Superpipelined Adder Designs", *Circuits and Systems, 1993., ISCAS '93, 1993 IEEE International Symposium on, 3-6 May 1993*, pp. 1841-1844 vol.3

[7] Peter M. Kogge, "*The Architecture of Pipelined Computers*", New York : Hemisphere, 1981.

[8] Israel Koren, "*Computer Arithmetic Algorithms*", A. K. Peters, Natick, MA, 2002, 2nd Edition

[9] John P. Hayes, "*Introduction to Digital Logic Design*", Addison Wesley Publishing Company, 1993.

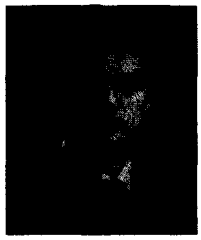
[10] 경종민, 박인철 외 공저, "*고성능 마이크로프로세서 구조 및 설계 방법*", 대영사, pp. 316-338

[11] Sajjan, G.Shiva, "*Computer Design and architecture*", Little Brown, Boston, 1985

[12] Jae-Cheol Ha, Sang-Jae Moon, "Efficient Architectures for Modular Exponentiation Using Montgomery Multiplier", *Korea Institute of Information Security & Cryptography*, 11-5, Oct. 2001

[13] J. L. Hennessy and D.A. Patterson, "*Computer Organization & Design : The Hardware / Software Interface, second edition*", Morgan Kaufmann Publishers, CA, 1998.

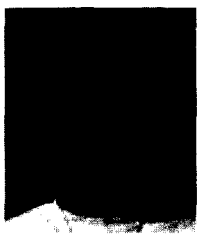
허 석 원(Seok-Won Heo) 준회원



2002년 2월 : 한양대학교 전자컴퓨터공학부 학사
2004년 2월 : 연세대학교 전기전자공학과 석사
2004년 2월~현재 : 삼성전자 반도체총괄 연구원

<관심분야> Flash Card Controller, Smart Card IC, Microprocessor, FPU, Crypto-processor

김 문 경(Moon-Gyung Kim) 준회원



1997년 2월 : 연세대학교 전자공학과 학사
1999년 2월 : 연세대학교 전자공학과 석사
1999년 3월~현재 : 연세대학교 전기전자공학과 박사과정

<관심분야> Microprocessor, Network Processor, Crypto-processor

이 용 주(Yong-Joo Lee) 정회원

1999년 8월 : 연세대학교 전자공학과 학사
2001년 8월 : 연세대학교 전자공학과 석사
2001년 9월~현재 : 연세대학교 전기전자공학과 박사과정

<관심분야> Network-processor, Crypto-processor, FPU, DSP, Wireless LAN

이 용 석(Yong-Surk Lee) 정회원

1973년 2월 : 연세대학교 전기공학과 학사
1977년 2월 : University of Michigan, Ann Arbor 석사
1981년 2월 : University of Michigan, Ann Arbor 박사
1993년~현재 : 연세대학교 전기전자공학과 교수

<관심분야> Microprocessor, Network-processor, Crypto-processor, Wireless LAN