논문 99-24-3A-11 한국통신학회논문지 '99-3 Vol.24 No.3A

DSP 기능을 갖춘 RISC 마이크로프로세서용 누적 동작 승산기의 VLSI 설계

정회원 최 병 윤

VLSI DESIGN OF MULTIPLIER-ACCUMULATOR MACRO FOR DSP-ORIENTED RISC MICROPROCESSOR

Byeong-Yoon Choi *Regular Member*

요 약

본 논문은 디지털 신호 처리와 제어 분야 응용을 위해 개발된 32 비트 RISC 마이크로프로세서의 누적 동작 승산기 회로에 대해 기술한다. 고속 곱셈 동작과 면적 효율성의 상반 관계를 만족시키기 위해, 32-비트 * 8-비트 곱셈 구조를 반복 활용하여 32-비트 곱셈 또는 누적 동작 곱셈의 carry-save 결과를 생성하고, RISC 칩 내부의 ALU를 활용하여 carry-save된 곱셈 결과를 더하여 최종 결과를 생성하는 방식을 채택하였다. 승산 동작에 사용되는 데이터의 크기가 대부분 작다는 특징을 고려하여, 승산 동작시 승수 데이타의 크기가 작을 경우 연산의 조기 종료가 가능하도록 하는 기능을 내장하였다. 설계된 승산기는 수정된 Booth 승산 알고리즘을 사용하며, 부호 비트의 과부하 문제를 제거한 부호-확장 알고리즘을 사용하였다. 승산기 회로는 0.6 ㎛ CMOS 공정으로 설계되었으며, 대략 9,100개의 게이트로 구성되며 최악 지연 시간은 13.8ns이다. 그리고 누적 동작 승산기의 레이아웃 면적은 2.16mm * 1.34 mm이다.

ABSTRACT

This paper describes a on-chip multiplier-accumulator macro for 32-bit RISC microprocessor which is designed for control and DSP applications. To satisfy trade-off between fast multiplication and area-efficient hardware, multiplication scheme is adopted which can generate 64-bit carry-save results for 32-bit by 32-bit multiplication(MUL) or multiplication & accumulation(MAC) with repeated use of 32-bit by 8-bit dedicated multiplier hardware and then add carry-save results with on-chip 32-bit ALU. To consider characteristics that most data used in multiplication are small, early-termination hardware which has data-dependent multiplication cycles is included. This multiplier architecture uses modified booth's algorithm and adopts efficient sign-extension scheme which eliminate over-loading problem of sign-bit signal. This multiplier is designed with 0.6㎛ triple metal CMOS technology and consists of about 9,100 gates and its worst case delay is about 13.8ns and its layout size is 2.16mm * 1.34mm.

I. 서 론

1980년 중반부터 32-비트 마이크로프로세서가 보편화되면서 내장형 및 디지털 신호처리(Digital Signal Processing) 응용 분야에 32 비트 마이크로프로세서의 필요성이 증가하였다. 이는 8비트, 16비트 내장형 프로세서가 새로운 응용 분야의 성능에 대한 요구를 만족시켜주지 못하기 때문이다. DSP 응용 분야는 빠른 곱셈 및 곱셈과 덧셈이 연결되는

* 동의대학교 컴퓨터공학과(bychoi@hyomin.dongeui.ac.kr)

※ 이 논문은 과학 재단이 지원한 1997년도 전반기 해외 Post-Doc. 연수 연구비에 의한 결과임
논문번호 : 98452-1014, 접수일자 : 1998년 10월 14일

MAC(multiplication & accumation) 기능이 중요하다[1]. 본 논문에서는 내장형 응용 분야[2]와 DSP 응용 RISC 프로세서에 내장하기에 적합한 누적(accumulation) 기능을 갖는 승산기 회로를 설계하였다. 일반적으로 마이크로프로세서와 부동 소수점 연산장치에 내장되는 승산기의 구조는 배열(array) 구조와 반복 연산(shift & add) 구조로 크게 나눌수 있다[3-7]. 그런데 32-비트 * 32-비트 곱셈을 수행하기 위해 배열(array) 구조[6]를 사용하는 경우 하드웨어 양이 너무 크기 때문에, 면적상 큰 문제가 된다. 반면에 수정된 Booth 알고리즘을 사용한 반복 연산 구조의 승산기[7]의 경우, 최대 16 사이클이 필요로 하므로 DSP 응용으로는 속도가 너무 떨어지는 문제가 있다. 따라서 본 논문에서 설계한 누적 동작 승산기는 무부호(unsigned) 와 부호화된(signed) 이진 곱셈(MUL) 및 누적 기능 곱셈(MAC)을 지원하는 것을 설계사양으로 채택하였다. 또한 마이크로프로세서에 내장하기에 적합하도록 면적 과 연산 시간의 상반 관계를 고려하여, 32-비트 * 8-비트 곱셈을 4 번 반복하여 32-bit * 32-비트 곱셈 또는 누적 기능 곱셈(MAC)을 등가 구현하는기법이 사용되었으며, 이를 하드웨어로 구현하기 위해 배열 구조와 반복 연산(shift-&-add) 구조가 결합된 형태를 갖는다. 또한 승산 동작으로 얻어진 64-비트 carry-save 된 곱셈 결과를 더하기 위해, 별도의 64-비트 가산기 회로를 사용하지 않고 RISC 마이크로프로세서 내부의 32-비트 ALU를 2번 반복 사용하여 최종 결과를 레지스터 파일에 저장하도록 하는 방안을 채택하여, 전용 승산기의 RISC 내장에 따른 면적 부담을 최소화하였으며, 승산기가 RISC의 클럭 주파수를 좌우하는 최악 전달 경로가 되지 않도록 하였다. 본 논문의 2장에서는 승산기 설계사양 과 MAC 기능을 지원하는 승산 알고리즘을 기술하였으며, 3장에서는 누적 동작 승산기 하드웨어 설계를, 4장에서는 설계한 회로의 성능을 분석하고, 5장에서는 결론을 맺었다.

Ⅱ. 승산기 설계 사양과 누적 동작 승산 알고리즘

1. 승산기 설계 사양

본 논문의 승산기가 적용될 RISC 마이크로프로세서는 기존 제어용 RISC[7]를 개선한 구조로 DSP 응용에 적합하도록 전용 승산기를 하드웨어 내부에 내장하고 있다. RISC 마이크로프로세서는 그림 1과 같이 크게 7가지 블럭, 즉 명령어 prefetch buffer, ALU, 배럴 시프터, 승산기, 레지스터 화일, 주소 발생 블럭, 제어부, field extractor 등으로 구성된다.

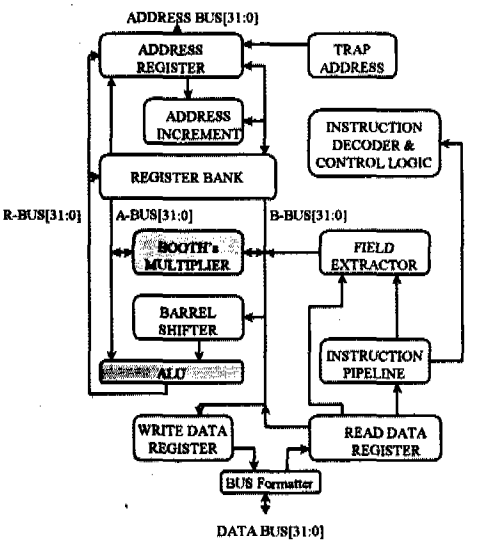


그림 1. DSP용 RISC 마이크로프로세서의 블록도
Fig. 1 Block diagram of DSP-oriented RISC microprocessor

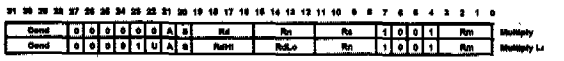


그림 2. 승산 명령어 형식
Fig. 2 Multiplication instruction format

그림에서 음영으로 처리된 부분(Booth's Multiplier, ALU)이 승산 동작과 관련된 주요 하드웨어를 나타낸다. 본 논문의 승산기 설계시 DSP 응용에 적합하도록, 기존 RISC[7-8]에 내장된 반복 연산(shift-&-add) 구조 승산기와 달리 허용된 면적 범위(칩 면적의 20 %내)에서 최대한 고속으로 동작하는 승산 하드웨어를 내장하는 것을 기본적인 설계목표로 설정하였다. 그리고 마이크로프로세서에 대한 컴파일러는 변수를 메모리 할당할 때, byte, halfword, word 등으로 정의하지만, RISC는 load-store 아키텍처로서 메모리에 있는 데이터를 먼저 내부 32 비트 레지스터로 가져와서 승산 동작을 한다. 따라서 본 연구의 승산 명령은 32 비트 데이터에 대한 승산 동작만을 지원한다. 표 1은 본 논문의 누적 동작 승산기가 지원하는 데이터형과 연산 형태를 나타낸다. 표 1에서 Rs, Rm, Rd, Rdhi, Rdlo는 모두 32비트 레지스터를 나타낸다. 그림 2는 승산과 누적 기능 승산과 관련된 명령어 형식을 나타낸다. Multiply 명령 형식은 32 비트 곱셈 결과를 얻는 경우를 나

타내며, A 비트값에 따라 승산과 누적 동작 승산(MAC)으로 구분된다. Multiply Long 형식은 64-비트 곱셈 결과를 생성하는 경우로, U 비트값에 따라 데이터형이 무부호(unsigned) 데이터와 부호(signed) 데이터 승산으로 구분된다.

표 1. 승산기의 설계 사양
Table 1. Specification of multiplier

데이터형	unsigned 32-bit data
	signed 32-bit data
곱셈(MUL) 형태	Rd <- Rs * Rm
	{Rdhi, Rdlo} <- Rs * Rm
누적 동작 곱셈(MAC) 형태	Rd <- Rs * Rm + Rd
	{Rdhi, Rdlo}<-Rs * Rm +{Rdhi, Rdlo}
면적	칩 면적의 20 % 이내

2. 승산 및 누적 승산 알고리즘

승산 알고리즘 및 승산기 구조 결정시 고려한 3가지 조건은 다음과 같다.

첫째, DSP 응용에 적합하도록 고속의 승산 동작이 가능해야 한다.

둘째, RISC CPU에 내장하기에 적합한 면적을 가져야한다.

셋째, 승산에 사용되는 대부분의 데이터 크기가 32 비트보다 작은 크기를 갖는다[9].

이러한 3가지 조건을 고려하여 본 논문에서는 그림 3(a)와 같이 32-비트 * 8-비트 승산기를 반복 사용하여 32-비트 * 32-비트 승산 동작을 구현하는 방안을 채택하였다. DSP 용 승산기에 사용되는 데이터는 대부분 16비트 이하이므로, 32-비트 * 8 비트 승산기에 조기 종료 기능을 내장 시킬 경우 성능상 커다란 저하가 발생치 않는다. 이것은 사용 빈도가 많은 기능을 명령어화하고 빈도가 낮은 동작은 채택한 명령어의 조합으로 구현하는 RISC 철학과 유사하다. 그런데 본 연구의 승산기는 누적 기능을 필요로 하므로, 그림 3(b)와 같이 승산 결과의 누적 덧셈이 필요하다. 그런데 이러한 덧셈을 캐리 전달 가산기(carry propagation adder : CPA)를 사용하여 구현하는 경우 기존 RISC의 동작 주파수를 감소시키는 문제가 있다. 따라서 본 연구에서는 RISC의 동작 주파수에 영향을 주지 않도록 하기 위해, 곱셈의 중간 결과를 carry-save 형태로 유지하는 방식[10]을 채택하였다. 그림 3을 직접적으로 구현할 경우 8개의 부분적(partial product)이 발생하여 이를 더하는데 많은 지연시간이 야기될 수 있다.

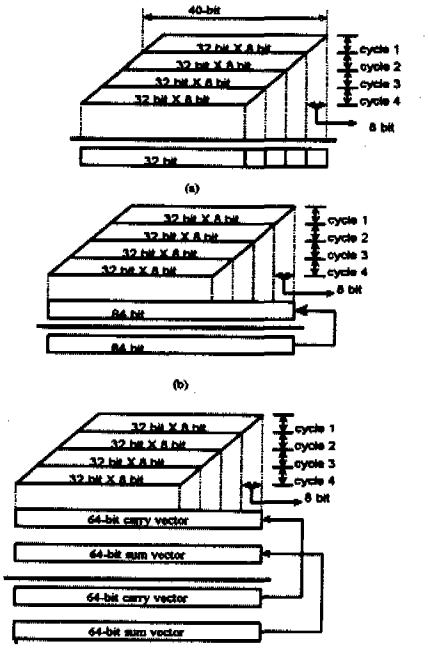


그림 3. 누적 동작 승산 알고리즘
a. 32-비트 * 8-비트를 사용하는 32-비트 * 32-비트 승산 알고리즘
b. 누적기능 승산 알고리즘
c. carry-save 출력 형태를 갖는 승산 알고리즘

Fig. 3. Multipliy-&-accumulation algorithm
a. 32-bit by 32-bit multiplication algorithm based on repeated use of 32-bit by 8-bit multiplication algorithm b.conventional multiply-&-accumulation algorithm with nonredundant output
c. multiply-&-accumulation algorithm with carry-save outputs

이러한 승산 알고리즘의 하드웨어 구현시 부분곱의 수를 줄이는 방안은 승수(multiplier) 필드에 대하여 리코딩(recoding)을 하는 방식으로 대표적인 것이 수정된 Booth 알고리즘[11]이다. 이러한 방식을 적용하는 경우 직접적인 곱셈 구현 방식에 비해 부분곱의 수를 1/2로 감소시킨다. 기본적인 수정된 Booth 리코딩은 그림 4와 같이 최하위 승수 비트 아래에 추가된 0을 포함해 3비트 단위로 중첩하여 리코딩된다. 그런데 본 연구의 승산기는 32 비트 * 8 비트 승산을 반복하는 구조로 4개의 그룹으로 구현된다. 단, 이때 고려해야 할 사항은 무부호(unsigned) 데이터의 곱셈의 경우, 수정된 Booth 곱셈 알고리즘을 적용하기 위해, 부호 비트(sign bit)

로 0이 추가 되어야 하므로, 33 비트에 대해 리코딩 동작이 이루어져야 하므로, 최상위 비트 그룹(그룹 4)은 다른 그룹과 달리 5개의 부분곱(Patrial Product) 생성이 필요하다. 이러한 동작을 위해 그룹 4를 처리하기에 적합한 하드웨어를 준비해두고, 다른 그룹 동작의 경우 필요하지 않은 기능은 마스킹(masking) 하는 방식을 채택하였다. 본 승산기에 누적 기능을 구현하기 위해, 그림 4와 같이 승산 동작 결과에 누적 데이터를 더하는 구조[12]와 누적 데이터를 먼저 누산기(SV-R)에 초기화 시킨 후, 승산 동작과 누적 동작을 통합하는 구조를 검토하였다. 그림 4(a) 방식의 경우 궤환(feedback)되는 데이터의 비트수가 작다는 잇점이 있으나, 누적 동작을 위한 별도의 CSA (carry save adder) 회로가 필요하다는 단점이 있다. 그림 4(b) 방식의 경우 초기에 누적 데이터 값을 누적 레지스터에 누적함으로서, 궤환되는 데이터가 64-비트가 되는 단점이 있으나, 별도의 CSA가 필요없으므로, 기존 RISC의 ALU를 최대한 활용할 수 있다는 잇점이 있다. 따라서 본 연구에서는 면적 효율성과 하드웨어 자원 공유 특성을 고려하여 그림 4(b)의 방식을 채택하였다.

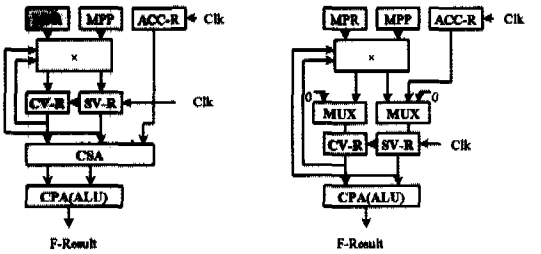


그림 4. MAC 연산 처리 방식
a. 기존 MAC 회로
b. 제안한 승산기의 MAC회로
Fig. 4 MAC operation scheme
a. conventional scheme
b. proposed scheme

Ⅲ. MAC 기능을 지원하는 승산기 회로 설계

그림 5는 그림 3과 그림 4에서 제시한 누적 동작 승산 알고리즘을 하드웨어로 구현한 블록도를 나타낸다. 누적 기능 승산기는 크게 입력 레지스터(MPR, MPP), Booth Encoder, Early Terminator, 부분곱 생성 회로(PP Generator), carry-save array, Rotator, 초기화 MUX와 carry-save된 중간 결과를 저장하기 위한 누산기 레지스터(CV-R, SV-R), 결과를 선택하여 ALU로 보내는 역할을 수행하는 결과 선택 회로(Aligner) 등으로 구성된다.

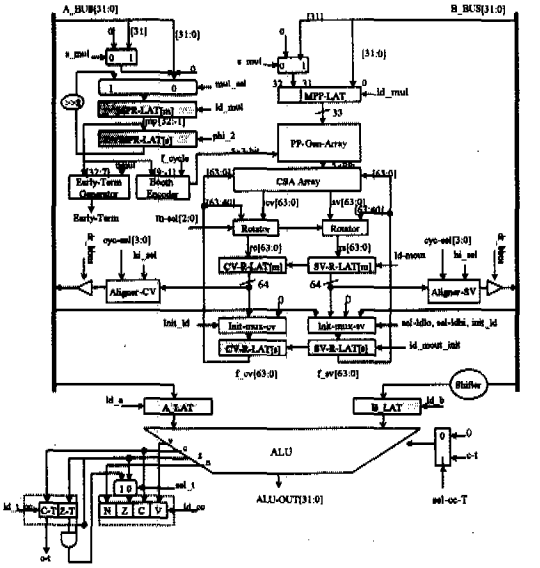


그림 5. 승산기 하드웨어 블록도
Fig. 5 Block diagram of multiplier-&-accumulator

1. Booth Encoder와 부분곱 생성회로 설계

무부호 데이터 곱셈의 경우, 수정된 Booth 알고리즘을 사용하기 위해서 32 비트 승수(multiplier)의 맨위에 양수(positive number)를 지시하는 0의 부호 비트(sign bit)가 추가해야 하므로, 실제로는 33-비트 곱셈이 필요하다. 따라서 이러한 동작을 지원하기 위해서, booth encoder 회로는 처음 3 사이클은 8 비트 리코딩을 수행하고 마지막 사이클은 10 비트 리코딩을 필요로 한다. 단, 하드웨어의 공유를 구현하기 위해, 10 비트 리코딩 회로를 준비하고, 처음 3 사이클의 경우 상위 3 비트를 0으로 처리하여 회로 동작에 영향을 주지 않도록 하였다. 승수 레지스터(MPR)에 8 비트 산술 시프터(arithmetic shifter) 기능을 내장하도록 하여 매사이클마다 항상 하위 11비트를 조사하여 booth 인코딩 동작이 가능하도록 하였다. 또한 부분곱(partial products) 제어에 필요한 제어선 수를 최소화하는 구조를 갖고 있다. 부분곱을 생성하는 회로의 경우 2의 보수 구현을 위해 1의 보수 결과에 1을 더하는 동작이 필요한데, 별도의 덧셈 동작을 배제하기 위해, 추가의 1 신호(neg-con 신호)의 덧셈은 carry-save array 내부에서 처리할 수 있도록 하였다.

2. carry-save array 설계

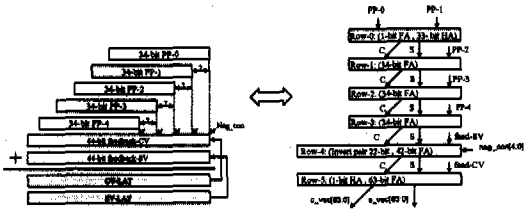
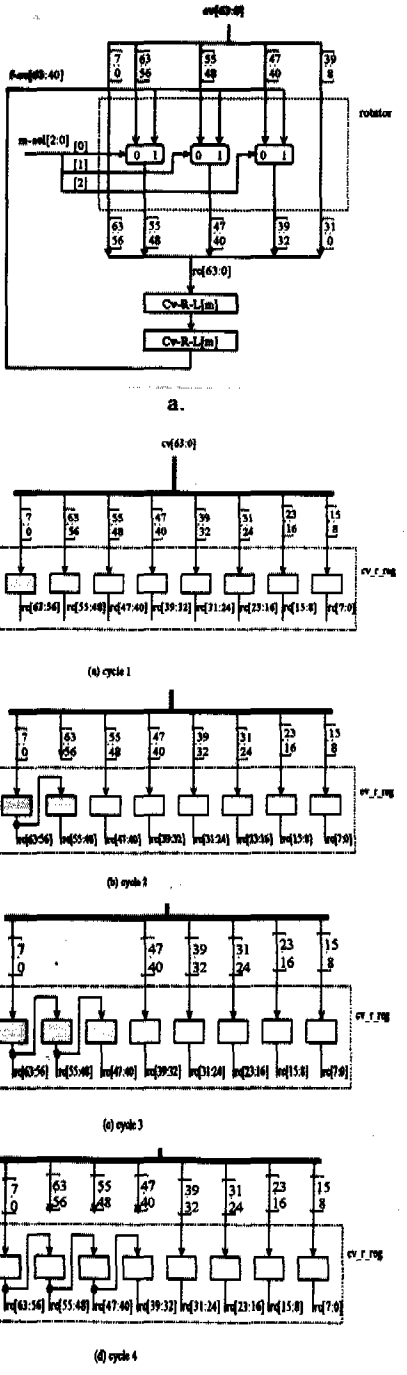


그림 6. carry-save 배열 회로
Fig. 6 carry-save array 회로

carry-save array부는 그림 6와 같이 carry-save된 64-비트 중간 결과(feed-SV, feed-CV)와 10-비트 승수 리코딩에 의해 새로이 생성된 5개의 부분곱이 배열구조로 결합된다. 배열 구조로 부분곱 결합시 부호 비트의 부호 확장 문제는 기존 알고리즘[13]을 사용하여 제거하였다. 단 결과값은 그림 7에 보여주는 바와 같이 우측으로 8-비트만큼 rotate 되면서 저장된다. 그림 3의 곱셈 알고리즘을 보면 매사이클에서 생성되는 하위 8비트 결과는 다음 사이클 동작과 무관함을 알수 있다. 또한 누적 동작 곱셈의 경우 누적된 중간 결과(CV-R, SV-R)이 누적 동작에 참여하는 비트수가 첫 번째 사이클에는 64-비트, 두번째 사이클에는 56-비트 등으로 매 사이클마다 8 비트씩 감소됨을 알수 있다. 따라서 누적 동작에 참여하지 않는 누적 레지스터(CV-R, SV-R) 위치에 매사이클 새로이 생성된 하위 8-비트 결과값을 저장하도록 하면 하드웨어 공유가 가능해지게 된다. 단 이러한 곱셈 동작(MUL)과 누적기능 곱셈(MAC) 동작의 초기화 과정에서 누적 레지스터를 0 또는 누적값(Rd, 또는 {Rdhi, Rdlo})으로 초기화 시켜, 누적 동작 승산 지원과 함께 하드웨어와 연산 시간 단축 효과를 얻을 수 있다.

3. 조기 종료기 및 결과 선택 회로 설계

Booth 곱셈 알고리즘은 리코딩에 참여하는 비트 수가 모두 동일 할 경우 단순한 이동(shift) 동작으로 구현된다. 그리고 승수 레지스터가 매 곱셈 사이클마다 8 비트 만큼 산술 시프트를 수행하므로, Booth 인코더에 관여하는 비트 위치가 고정된다. 이러한 특성을 이용하면 부부호(unsigned) 데이터 곱셈의 경우 부호 비트가 항상 0 이므로, 승수 레지스터의 [32:7] 비트값이 모두 0 일 때 조기 종료(early-termination)가 가능하며, 부호(signed) 데이터 곱셈의 경우 승수 레지스터의 [32:7] 비트 값이 모두 0 또는 1인 경우 조기 종료가 가능하다.



a.

b.

그림 7. rotator 회로와 동작
a. 회로 b. 사이클별 동작
Fig. 7 rotator circuit and its operation
a. circuit b. its operation during multiply cycle

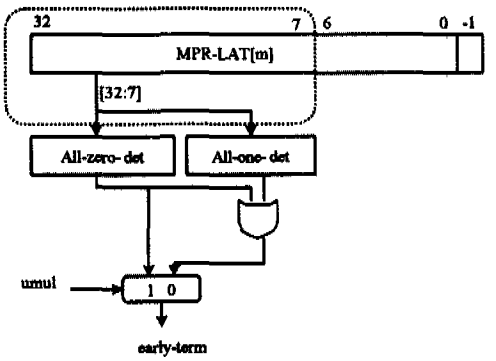


그림 8. 조기 종료 조건 감지 회로
Fig. 8 Early-terminator

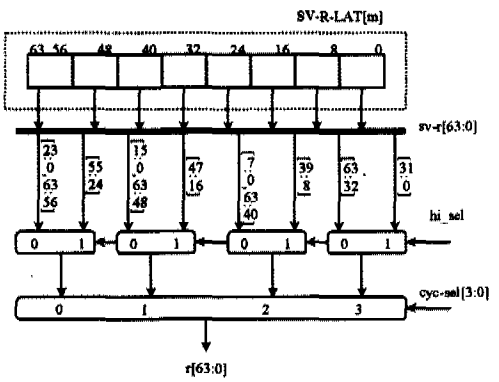


그림 9. 결과 선택 회로
Fig. 9 Result aligner

그림 8은 조기 종료 감지 회로도(early terminator)를 나타낸다. 단 이때 고려해야 할 사항은 그림 7에 보는바와 같이 종료 시점에 따라 결과값의 위치가 다르므로 결과 선택 회로(Aligner)가 필요하게 된다. 그림 9과 표 2은 각각 결과 정렬회로 와 결과 선택 동작을 나타낸다. 단, 결과값이 64-비트인데 비해 ALU를 거쳐서 레지스터 파일로 결과를 전송하기 위한 경로는 32-비트이므로, Carry-Vector 레지스터의 하위 32-비트는 A-bus로, Sum-Vector 레지스터의 하위 32-비트는 B-bus를 통해 ALU에서 더해지며, 이어서 상위 32-비트가 마찬가지 방식으로 ALU를 통해 레지스터 파일에 저장된다. 이와 같이 하위 비트를 먼저 전송하는 이유는 carry-save 형태의 곱셈 또는 누적 곱셈 결과의 하위 32-비트의 덧셈 과정에서 발생하는 Cout, Zero flag를 임시 조건 코드 레지스터(C_t, Z_t)에 유지되었다가, 상위 32-비트값을 더할 때 활용하기 위해서이다. Z-t는 상위 32-비트 덧셈시 발생하는 Z flag와 AND 게이트로 결합되어 최종 64-비트 결과에 대한 Z flag를 발생하게 된다. 결과 선택회로의 Hi-sel 신호는 상위 32 비트 결과가 ALU로 전송되는 시점을 제어한다. 조기 종료 동작에 따라 1 사이클에 종료된 경우, MUL-CNT[1:0]는 00을 갖고, 2 사이클만에 종료되는 경우는 01, 3 사이클만에 종료되는 경우 10, 4 사이클만에 종료되는 경우 11을 갖는다. cyc-sel[3:0]는 MUL-CNT[1:0]의 디코딩값을 나타낸다

표 2. 결과 선택 회로의 동작
Table 2. Operation of result aligner

cyc-sel[3:0]	hi-sel	output
0001	0	sv-r[23:0, 63:56]
	1	sv-r[55:24]
0010	0	sv-r[15:0, 63:48]
	1	sv-r[47:16]
0100	0	sv-r[7:0, 63:40]
	1	sv-r[39:8]
1000	0	sv-r[63:32]
	1	sv-r[31:0]

4. 제어회로 설계

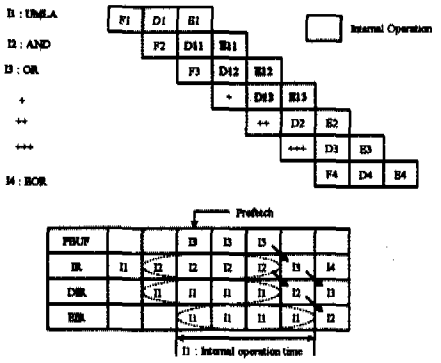


그림 10. 내부 연산 동작을 갖는 승산 명령의 파이프라인 동작
Fig. 10 Pipeline of multiply instruction with internal operations

본 연구의 승산기가 적용되는 RISC 마이크로프로세서는 3 단 파이프라인 구조를 갖고 있다. 그러나 shift-&-add 동작을 하나의 명령으로 정의하고 이를 조합하여 승산 동작을 구현하는 기존 RISC[8] 방식의 경우, 코드 길이가 늘어나는 결점이 있으며 또한 조기 종료 조건을 감지하기 위해서는 별도의 비교 및 분기 명령이 필요하다는 단점이 있다. 따라

서 본 연구의 누적 승산기는 그림 2의 명령어 형식에 보는 바와 같이 승산을 위한 단일 명령을 정의하고, 상태도에 바탕을 둔 다중 사이클(multi-cycle)로 구현하였다. 승산 명령은 다른 명령과 달리 단일 사이클로 구현되지 않으므로, 3단 파이프라인에 내장시키기 위해 그림 10에 보이는 바와 같이 진행중인 명령의 상태를 지시하는 제어회로내의 사이클 카운터(CYC-ST[1:0])와 승산 동작 사이클을 지시하는 MUL-CNT[1:0]를 사용하여, 승산 명령이 각 사이클마다 구분된 내부 연산(internal operation)을 수행하도록 하였다.

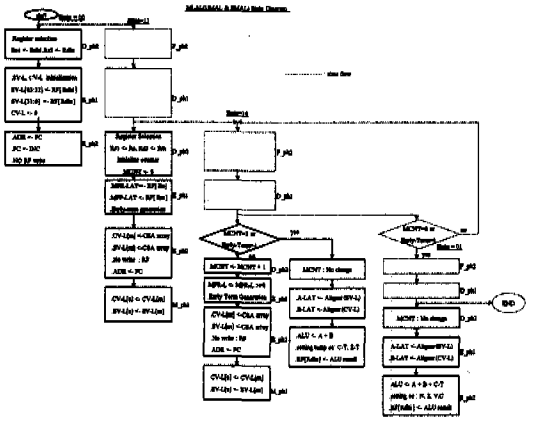


그림 11. 누적 기능을 갖는 승산 명령에 대한 명령어 파이프라인 상태도
Fig. 11 Instruction pipeline state diagram of multiply-&-accumulation

그림 11는 기존 흐름도에 의한 명령어 제어 신호 생성 방안[14]을 파이프라인 처리에 적합도록 수정하여, 본 승산기의 누적 승산 명령어 파이프라인 동작을 기술한 상태도를 나타낸다. 기존 디지털 시스템의 상태도와는 달리 각 상태가 하나의 명령어 파이프라인을 나타낸다. 단 새로운 상태 변화값은 현재 상태의 D 단계 phase 2에서 생성되지만, 그 상태가 명령 디코더에 사용되는 시점은 1 사이클 지난뒤에 이루어진다. 그리고 그림 12는 승산 명령이 처리되기 위한 RISC 내부 제어 회로의 블록도를 나타낸다. 제어회로는 명령어 buffer, 데이터 정적 파이프라인 제어(data stationary pipeline control) 부와 명령 사이클 제어부(Cycle counter), 그리고 승산 사이클을 제어하는 카운터부(Multiplier Counter) 등으로 구성된다. 그리고 조기 종료 출력 Early-Term 값이 승산 동작의 조기 종료를 제어하기 위해 명령어 디코더에 사용된다.

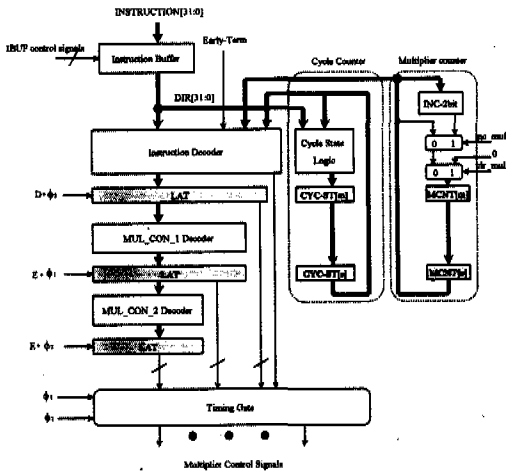


그림 12. 승산기 제어회로의 블록도
Fig. 12 Block diagram of multiplier controller

Ⅳ. 성능 분석 및 검증

설계된 승산기의 성능을 그림 13과 같이 오퍼랜드 크기와 승산 명령의 분포에 따라 분석하였다. 그림 13에서 k, l, m, n은 승수 오퍼랜드의 크기가 각각 8비트, 16비트, 24비트, 32비트인 분포를 나타낸다. 그리고 t는 전체 명령어중 곱셈 명령어의 동적 빈도수를 나타내다. 본 연구의 32 비트 RISC의 검증에 사용된 benchmark 프로그램의 경우, 승수 오퍼랜드의 크기가 16비트이하인 경우가 90%이상으로 나타났다. 그런데 명령어 분포와 달리 오퍼랜드 크기 분포는 프로그래머가 사용한 데이터에 크게 의존하는 문제가 있었다. 따라서 보다 보편적인 결과를 얻기 위해 그림 13에서는 1,000개의 명령어를 수행하는 5가지 경우에 대해 비교하였다. 단 분석을 단순화 하기 위해 곱셈 명령을 제외한 다른 명령의 경우는 CPI(clock per instruction)를 1로 가정하였다. 그림 13에서 오퍼랜드 크기가 8 비트인 경우 3 사이클에 수행되는 것으로 나타낸 것은, 그림 11에서 보는 바와 같이 carry-save된 64-비트 승산 결과가 32비트 ALU를 통해, 32비트씩 나뉘어서 레지스터 파일에 저장되는 동작 특성을 반영하여 2 사이클이 추가되었다. 그림 13을 분석하면 오퍼랜드의 크기가 작고, 승산 명령어의 분포가 높을수록, 조기 종료 기능을 갖는 승산기의 성능이 크게 증가함을 알 수 있다. 단, 곱셈 동작을 1 사이클에 처리하는 전용 DSP 프로세서의 경우 병렬 승산기 사용으로 clock cycle 수는 감소하지만, 클럭 주파수가 1/4이

하로 감소하며, 오퍼랜드 크기가 대부분 작다는 특성을 활용하지 못하는 문제가 있다. 설계된 RISC의 승산기는 Verilog HDL(hardware description language)[15]을 사용하여 논리 및 behavioral 시뮬레이션을 행하였다. HDL 수준에서 검증된 프로세서는 ASIC 합성기를 사용하여 게이트 수준으로 합성되었다. ASIC Synthesizer를 이용해 얻은 회로에 대해 타이밍 시뮬레이션을 행하였다. 현재 설계한 승산기 회로의 최악 전달 경로는 Booth Encoder, PP-gen, CSA(carry save array)로 구성되며, 타이밍 검증 결과 그림 14과 같이 13.8ns의 최악 지연 시간을 가짐이 알 수 있었다. 따라서 승산기 자체의 최대 동작 가능 주파수는 70Mhz 이상이므로, 승산기가 내장되는 RISC의 클럭 주파수(현재 40 Mhz 동작)에 영향을 주지 않는다, 그리고 사용한 게이트 수는 약 9100 gates(2-input NAND기준)으로 전체 RISC 하드웨어에서 차지하는 비율은 약 18 %이다. 합성된 회로는 0.6 ㎛ 삼중 금속 단일 실리콘 CMOS 공정을 이용한 표준셀 방식으로 레이아웃 되었다. 승산기의 레이아웃 면적은 2.16mm * 1.34 mm으로 전체 칩에서 차지하는 면적은 15 %이하이다. 따라서 본 승산기는 2장의 설계사양으로 정한 RISC 프로세서 20 % 내의 면적 요구 조건을 만족함을 알수 있다. 표 3와 그림 15은 합성된 승산기의 전기적 특성과 레이아웃을 나타낸다. 내장형 제어 및 DSP용 RISC 승산기는 대략 50 Mhz이상의 동작 주파수를 필요로 하므로 본 연구의 승산기는 이에 적합한 사양을 갖고 있다고 판단된다. 또한 32-비트 * 8-비트 승산 동작을 반복 활용하여 32-비트 승산기를 구현하므로 면적상 효율적이며, carry-save된 결과를 더하기 위해 별도의 가산기를 사용하지 않고, RISC내의 ALU를 활용함으로서 면적의 부담을 최소화 하였다.

표 3. 전기적 특성
Table 3. Electrical characteristics

사용 공정	0.6 ㎛ 삼중 금속 단일 실리콘 CMOS 공정
게이트 수 개수 (2-input NAND 기준)	약 9,100 (RISC칩의 18 %)
최대 동작 주파수	70 Mhz
지원 명령 형태	Multiply, MAC
지원하는 데이터 형태	unsigned, signed
레이아웃 면적	2.16mm * 1.34 mm (RISC 칩의 15 %이하)

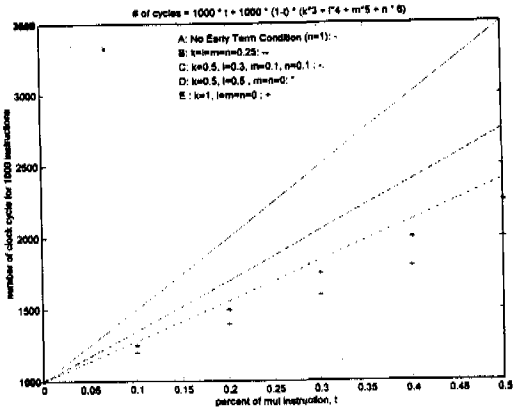


그림 13. 제안한 승산기의 오퍼랜드 크기에 따른 성능 분석 (k, l, m, n : 승수 오퍼랜드 크기가 8비트, 16비트, 24비트, 32비트인 분포)
Fig. 13. Performance evaluation of proposed multiplier for various operand sizes.

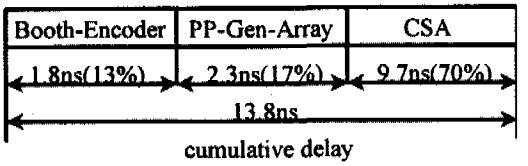


그림 14. 최대 경로 지연에 대한 게이트 수준 시뮬레이션
Fig. 14 Gate level simulation of critical delay path

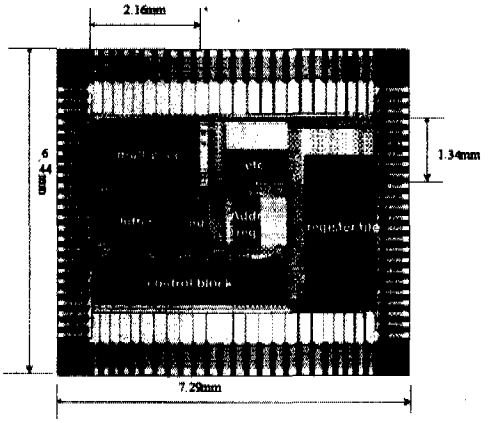


그림 15. 레이아웃
Fig. 15. Layout

V. 결 론

본 논문에서는 DSP 응용을 위한 RISC 마이크로프로세서에 적합한 누적 기능을 갖는 승산기 회로를 설계하였다. 승산기 구조 결정시 DSP 응용에 적

합하도록 부호(signed) 및 무부호(unsigned) 데이터의 승산 동작 이외에 누적 승산(MAC)을 효율적으로 처리할 수 있도록 하였다. 또한 배열 구조 승산기의 면적 문제와 반복 연산(shift-&-add 동작) 승상기의 속도 문제의 상반 관계를 고려하여, 조기 종료 기능을 갖는 32-비트 * 8-비트 승산기 하드웨어를 내장하고, 이를 반복 사용하여 32 비트 누적 동작 승산을 구현하는, 배열 구조와 반복 연산 구조가 결합된 승산기 구조를 채택하였다. 또한 승산기의 carry-save된 결과를 더하기 위해 별도의 가산기를 사용하지 않고, RISC 내의 ALU를 활용함으로서, 면적의 부담을 최소화 하였다. 그리고 승산기에 사용하는 데이터의 크기가 16비트 이하인 경우가 많으므로, 조기 종료 기능과 결합될 경우, 채택한 승산기 회로는 병렬 구조 승산기에 비해 성능 저하는 극히 작다. 설계된 승산기 회로는 32-bit × 32-bit 승산 동작과 MAC(multiply & accumulation) 연산를 최대 70 Mhz로 1 - 4 clock cycle의 latency를 갖고 처리할 수 있다. 본 연구의 누적 기능 승산기는 약 9,100 게이트로 구성되며, 0.6 ㎛ 삼중 금속 단일 실리콘 CMOS 공정을 사용하여 설계되었으며, 레이아웃 면적은 2.16mm * 1.34 mm으로 전체 RISC 칩에서 차지하는 면적은 15 %이하이다. 따라서 본 연구에서 설계한 누적 기능 승산기 회로는 면적 측면에서 효율적이며, 조기 종료 기능과 결합하여 고속 승산 및 누적 승산 동작을 지원하므로, 고속의 MAC 동작이 필요한 디지털 신호 처리(DSP)와 통신 분야에 효율적으로 사용 가능하다.

참 고 문 헌

[1] Tony Baker, "Maximizing Hardware/Software Tradeoffs in Real-Time RISC-Based Systems," Intel Corp., 1989.

[2] Ron Cates, " Processor Architecture Considerations for Embedded Controller Applications", IEEE Micro, pp. 28-34, June 1988.

[3] Michael Dolle and Manfred Schlett, "A Cost-Effective RISC/DSP Microprocessor for Embedded Systems," IEEE Micro, pp. 32-40, October, 1995.

[4] Kwak Seung Ho Kwak, Byeong Yoon Choiand Moon Key Lee, "A 32-bit Low Power RISC Core for Embedded Application," 1995 IEEE TENCON, pp.488~491, 1995.

[5] C.R. Baugh and B.A. Wooley, "A two's complement parallel array multiplication algorithm," IEEE Trans. Computer, Vol.C-22, pp.1045-1047, Dec. 1073.

[6] C.S. Wallace, "A suggestion for fast multipliers," IEEE Trans. Electronic Computer, vol. EC-13, pp.14-17, Feb. 1964.

[7] 곽 승 호, 최 병 윤, 이 문기, "PDA를 위한 RISC 코어의 설계," 한국 통신학회 논문지, 제 22 권 10호, pp2136 - 2149, 1997.

[8] Paul Chow, The MIPS-X Microprocessor, KAP, 1989. 9. Veliko Milutinovic and Mark Bettings, "Multiplier/Shifter Tradeoffs in a 32-bit GaAs Microprocessor," in Gallium Arsenide Computer Design, pp.318-327, 1988.

[10] Tobias G. Noll, "Carry-save Architectures for High Speed Digital Signal Processing," Journal of VLSI Signal Processings, Vol.3, pp. 121-140, 1991.

[11] L. P. Rubinfield, "A proof of the modified Booth's algorithm for multiplication," IEEE Trans. Computer., vol. C-24, pp.1014~1015, Oct. 1975.

[12] Hiroaki Murakami and Naoka Yano, " A Multiplier-Accumulator Macro for a 45 MIPS Embedded RISC Processor," IEEE JSSC Vol. 31, No. 7, pp.1067 - 1071, 1996.

[13] O. Salomon, J-M Green and H. Klar, " General Algorithm for a Simplified Addition of 2's Complement Numbers", IEEE JSSC, Vol. 30. No. 7, pp. 839~844, July, 1995.

[14] Nick Tredenick, Microprocessor Logic Design -The flowchart Method-digital press, 1987

[15] Eliezer Sternheim, *Digital Design with Verilog HDL*, Automota Publishing Co., 1989.

최 병 윤(Byeong-Yoon Choi) 정회원

1985년 2월 : 연세대학교 전자공학과 졸업(공학사)
1987년 2월 : 연세대학교 대학원 전자공학과 졸업 (공학석사)
1992년 8월 : 연세대학교 대학원 전자공학과 졸업 (공학박사)
1990년 3월~1993년 2월 : ASIC 설계 연구소 선임연구원
1993년 3월~1995년 2월 : 동의대학교 컴퓨터공학과 전임강사
1997년 8월~1998년 7월 : University of Illinois at Urbana-Champaign, Visiting Scholars
1995년 3월~현재 : 동의대학교 컴퓨터공학과 조교수
<주관심 분야> RISC 마이크로프로세서 설계, 디지털 신호처리 및 통신용 VLSI 설계