저전력 승산기 보조 프로세서 설계

이 창 호, 곽 승 호, 이 문 기
연세대학교 전기전자 공학과
전화 : 02-2123-4731 / 핸드폰 : 016-349-2434

A Low-power Multiplier Co-processor Design

Chang-Ho Lee, Sung-Ho Kwak, Moon-Key Lee
Dept. of Electrical and Electronic Engineering, Yonsei University
E-mail : chlee@spark.yonsei.ac.kr

Abstract

This paper describes a fast and low-power multiplier co-processor architecture for digital signal processing applications and real-time control systems and its use as a multiplier co-processor for a 32-bit RISC microprocessor utilizing its one of the 16 co-processor interfaces. Its architecture adopts various algorithms to reduce the dynamic power and the area as well.

The designed multiplier performs 32x32 bit multiplication, and was designed using verilog HDL and 0.35um, 3V, 4M CMOS standard cell library. Its target operating speed is 40MHz, area lower than 10000 gate counts, and 10mW/MHz of power.

I. 서론

1980년대 중반부터 32비트 마이크로프로세서가 보편화되면서 내장형 및 디지털 신호처리 응용분야와 각종 실시간 제어 응용분야에서 32비트 마이크로프로세서의 필요성이 증가하였다. 이는 8비트 및 16비트의 내장형 프로세서들이 급속히 변화하는 응용분야의 성능에 대한 요구를 만족 시켜주지 못하기 때문이다. 이러한 관점에서 볼 때 비트수가 커질수록 다른 연산에 비해 긴 연산시간을 갖는 승산동작을 빠르게 해주는 승산블록은 필수적이다. 아울러 날로 발전하는 각종 휴대형 기기들이 공통적으로 요구하는 낮은 전력소모와 면적 효율성을 만족시키는 승산기의 설계는 매우 중요하다.

본 논문에서는 승산과정 중간에 생성되는 부분 곱의 수를 줄이고 부분 곱 노드에서의 스위칭 확률을 줄여주는 저전력을 위한 변형된 Booth 알고리즘을 적용하여 승산기 블록을 설계하였고, 이렇게 설계된 승산기 블록을 사용하여 32x32 비트 승산동작을 한번에 처리하지 못하는 32비트 RISC 마이크로프로세서의 보조 프로세서 인터페이스를 사용하여 승산 전용 보조 프로세서로 동작하게 함으로써 기존 프로세서를 사용한 시스템의 전체적인 성능향상을 이루도록 하였다.

II. 승산기 동작 알고리즘

2.1 원형의 Booth 알고리즘과 문제점

승산 동작은 크게 두 가지 동작으로 나누어 볼 수 있다. 하나는 승수의 비트별로 피승수의 부분 곱을 생성하는 것이고 다른 하나는 생성된 부분 곱들을 모두 더해 최종결과를 얻는 동작이다. 즉, 승산기의 성능향상을 위해서는 생성되는 부분 곱의 수를 줄이고 생성된 부분 곱들을 빠른 속도로 더해 주어야 한다. 부분 곱의 수를 줄이는 알고리즘으로 널리 사용되는 것이 Booth의 알고리즘이다[1]. 1951년에 발표된 이 원형의 알고리즘은 연속되는 0이나 1에 대해서는 원래 필요한

수만큼의 부분 곱보다 적은 수의 부분 곱만이 필요하다는 원리에 근거하고 있다. 예를 들어, 승수에 연속된 n개의 1이 나타난다고 하였을 때 이는 n개의 부분 곱 대신 단지 2개의 부분 곱으로 대치 가능하다. 이러한 동작을 승수를 SD(signed digit) code로 recoding 한다고 하며 표 1에 Booth 알고리즘의 recoding 동작을 요약하였다.

표 1. 원형 Booth 알고리즘의 recoding 동작.

X_i	X_{i-1}	동작	설명	Y_i
0	0	shift only	string of zeros	0
1	1	shift only	string of ones	0
1	0	sub and shift	beginning of a string of ones	$\bar{1}$
0	1	add and shift	end of a string of ones	1

하지만 이 알고리즘은 실제로 적용하기에는 몇 가지 문제점을 가지고 있는데 이는 다음과 같다.

1. 연속되는 0 또는 1의 개수를 판단하기 어려움
2. 동기식 시스템으로 적용이 어려움
3. 생성하는 부분 곱의 종류가 일정치 않음
4. 반복되어 나타나는 0과 1에 대해 비효율적임

2.2 수정된 4진 Booth 알고리즘

위와 같은 문제점을 해결한 것이 바로 수정된 Booth 알고리즘이다[2]. 수정된 Booth 알고리즘에서는 승수를 recoding 시에 두 비트가 아닌 한 비트씩 겹치는 3비트 단위로 끊어서 한다. 그림 1에서 16비트의 경우에 대한 이들 비트 그룹을 나타내었고 표 2에서는 수정된 4진 Booth 알고리즘의 동작을 요약하였다.

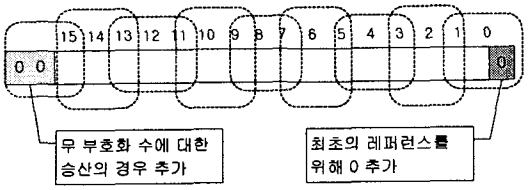

그림 1. 16비트 승수에 대한 비트 그룹 예

이러한 수정은 승수의 비트에 따라 생성되는 부분 곱의 개수를 일정하게 하여 원형의 알고리즘의 문제점을 해결할 수 있다.

2.3 저전력을 위해 변형된 알고리즘

저전력을 위해 변형된 알고리즘은 부분 곱의 노드에서 0의 확률을 높여 줌으로써 부분 곱 노드에서의 스

표 2. 수정된 4진 Booth 알고리즘 동작

X_i X_{i-1}	X_{i-2}	Y_i Y_{i-1}	동작	설명
0 0	0	0 0	+0	string of zeros
0 1	0	0 1	+A	a single 1
1 0	0	$\bar{1}$0	-2A	beginning of 1's
1 1	0	0 $\bar{1}$	-A	beginning of 1's
0 0	1	0 1	+A	end of 1's
0 1	1	1 0	+2A	end of 1's
1 0	1	0 $\bar{1}$	-A	a single 0
1 1	1	0 0	+0	string of 1's

위칭이 일어날 확률을 줄여주는 방법을 취하고 있다[3].표 2에서 볼 수 있듯이 수정된 4진 Booth 알고리즘에서 레코딩된 승수의 패턴을 두 가지로 나누어 볼 수 있다. 그 첫 번째는 다음과 같이 두 인접한 비트가 서로 상반된 부호를 가지고있고 변형시 캐리의 전달이 필요 없는 경우이고 나머지는 이러한 경우가 아닌 경우이다.

$$01\,\bar{1}0 \rightarrow 0010$$

$$0\,\bar{1}10 \rightarrow 00\,\bar{1}0$$

이러한 방법으로 나타나는 두 가지 경우에 대해서 레코딩을 변경하여 준다면 표 3에 나타난 것처럼 0의 확률을 증가시켜 결과적으로 부분 곱 노드에서의 스위칭 활동을 3.75% 감소시키게 된다.

표 3. 레코딩된 승수의 확률분포 비교

P(s)	Traditional	Improved
P(-2)	0.125	0.125
P(-1)	0.250	0.219
P(0)	0.250	0.313
P(1)	0.250	0.219
P(2)	0.125	0.125

III. 보조 프로세서 동작

최근에는 각종 멀티미디어 기기와 휴대 및 내장형 시스템에서의 프로세서에 대한 수요가 급증하면서 속도와 전력소모 및 가격적인 측면에서도 우수한 프로세서가 두각을 나타내고 있는데 이러한 것으로는 ARM 프로세서를 들 수 있다. 본 논문은 현재 연세대학교 VLSI & CAD 연구실에서 자체 보유하고 있는 ARM7과 명령어 레벨에서 호환되는 32비트 RISC MCU 코어를 목표로 삼아 설계되었다.

3.1 보조 프로세서 인터페이스

적용된 RISC 코어는 최고 16개의 보조 프로세서에 의해 성능이 향상되어질 수 있으며 보조 프로세서와의 인터페이스는 다음의 3개의 신호로써 이루어진다.

(1) nCPI(Not Co-processor Instruction)

nCPI 신호는 코어에서 보조 프로세서 명령어를 수행시킬 때 발생시키며 이때 nCPI 신호는 LOW가 된다. 이후에는 보조 프로세서의 반응(CPA, CPB)을 기다리며 적절하게 대응하게 된다.

(2) CPA(Co-processor Absent)

코어에서 nCPI 신호가 발생하게 되면 해당되는 보조 프로세서는 즉시 CPA신호를 LOW로 하게 된다. 만일 nCPI 신호가 발생한 클럭 구간이 끝날 때까지 CPA신호가 HIGH를 유지한다면 코어는 보조 프로세서 명령을 취소하고 미정의 명령어 트랩을 취하고, LOW로 유지한다면 코어는 다시 CPB가 LOW일 때까지 기다리게 된다.

(3) CPB(Co-processor Busy)

해당되는 보조 프로세서의 CPB 신호가 HIGH로 유지되는 경우 그 보조 프로세서는 당장 명령을 수행할 수 없는 경우이다. 코어는 nCPI 신호가 LOW인 페이즈1의 매번 끝마다 CPB 신호를 체크하게되고 보조 프로세서는 다시 명령어를 수행할 상태가 되면 CPB신호를 LOW로 하게 된다. 그림 2에서 이러한 인터페이스 신호의 타이밍을 나타내었다.

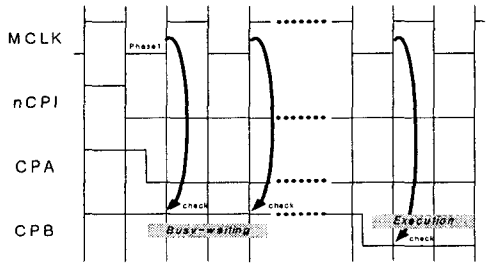

그림 2. 인터페이스 신호 타이밍

3.2 보조 프로세서 명령어

적용된 코어의 보조 프로세서 명령은 데이터 전송 명령어, 레지스터 전송 명령어, 데이터 연산 명령어의 3가지 형태로 구분된다. 모든 보조 프로세서 명령어는 상태 필드가 참일 경우에만 수행된다. 데이터 전송 명령은 보조 프로세서와 메모리간의 직접 데이터 전송을 위해 사용되는데 본 논문에서 설계된 승산기는 데이터 전송은 지원하지 않도록 하였다.

(1) 데이터 연산 명령

데이터 연산 명령은 보조 프로세서가 실제 데이터 연산을 수행하는 명령어이다. 이 명령은 내부에서 연산이 수행되고 결과 역시 내부에 저장된다. 지원하는 명령으로는 부호 및 무 부호화 32비트*32비트 곱셈과 64비트 곱셈을 지원한다. 표 4에서는 데이터 연산 명령의 종류를 나타내었고 그림 3에서는 명령어 필드를 나타내었다.

표 4. 데이터 연산 명령어

지원 명령어 니모닉	CPOpc [21:20]	L	S	Cp [6:5]	동작 정의	오퍼레이션
MSS	0 0	0	1	00	부호화 승산 32비트	MRd := MRn * MRm
MUS	0 1	0	0	00	무부호화 승산 32비트	MRd := MRn * MRm
MSL	1 0	1	1	00~11	부호화 승산 64 비트	MRd := MRn * MRm
MUL	1 1	1	0	00~11	무부호화 승산 64 비트	MRd := MRn * MRm

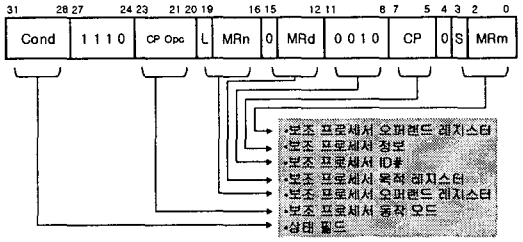

그림 3. 데이터 연산 명령어 필드

(2) 레지스터 전송 명령

레지스터 전송 명령은 코어의 내부 레지스터와 보조 프로세서 레지스터간의 데이터 전송을 위해 사용된다. 지원하는 명령은 LOAD/STORE 두 가지 종류로써 L 비트에 의해 결정된다. 표 5에서는 명령어의 종류를 나타내었고 그림 4에서는 명령어 필드를 나타내었다.

표 5. 레지스터 전송 명령

지원 명령어 니모닉	L	동작 정의	오퍼레이션
MRC	1	보조 프로세서 레지스터에서 ARM7 레지스터로 전송	MRn := Rd
MCR	0	ARM7 레지스터에서 보조 프로세서 레지스터로 전송	Rd := MRn

Ⅳ. 보조 프로세서 설계 및 검증

설계된 승산기 블록은 크게 Booth 인코더 블록과 부분 곱 생성 블록, CSA(Carry-Save Adder) Tree 블록, CPA(Carry-Propagation Adder) 블록으로 나눌 수 있다. Booth 인코더 블록과 부분 곱 생선 블록은 부호

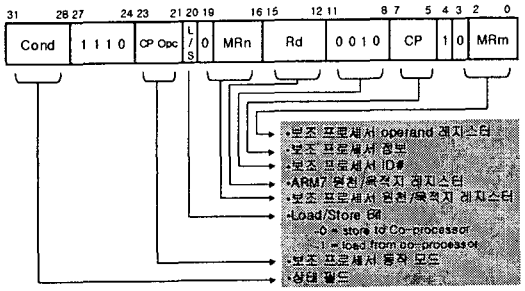

그림 4. 레지스터 전송 명령어 필드

및 무 부호화 수에 대한 승산동작을 위해 s_mul 신호를 이용하여 최상위 비트에 대한 부호 처리를 하여 준다. CSA Tree 블록에서는 생성된 총 17개의 34비트 데이터를 더해주는데 부호 확장 시 발생되는 부호비트의 과부하 문제를 해결하는 알고리즘을 적용하였고 각 부분 곱의 최하위 비트에는 WCSA cell 구조를 적용하여 최종결과를 만들어내는 CPA 블록의 비트 수를 감소하여 캐리의 전달시간이 줄어들게 하였다[4,5]. 그림 5에서는 각 승산기 블록에 대한 면적과 소요되는 delay에 대한 백분율을 나타내었다. 이 그림에서 보면 부분 곱 노드와 관련된 블록이 전체의 약 80%를 차지하는 것을 알 수 있다. 저전력을 위한 변형된 Booth 알고리즘을 적용 시에 인코더 부분에서 게이트 delay가 늘어나는데 전체의 delay중 인코더 부분이 차지하는 부분은 약 3.4%로써 그 비중이 크지 않다는 것을 알 수 있다.

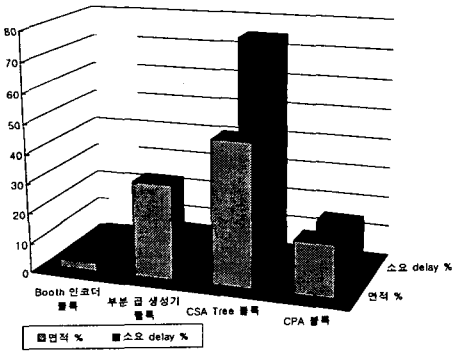

그림 5. 승산기 블록의 면적 및 소요 delay 백분율

그림 6에서는 설계된 승산기 보조 프로세서의 전체 블록 도를 나타내고 있다.

V. 결론

본 논문에서는 디지털 신호처리 응용분야와 제어 응용분야에 적용할 수 있는 저전력 승산기를 설계하였고

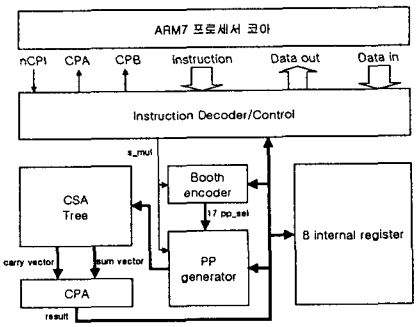

그림 6. 승산기 보조프로세서 블록도

이를 사용하여 32비트 RISC MCU 코어에서 사용할 수 있는 보조 프로세서를 설계하였다. 저전력 동작을 위하여 전체의 80%에 해당하는 부분 곱 관련 노드에서의 스위칭 활동을 3.75% 감소시켜주는 알고리즘을 사용하였고 면적과 속도를 고려하여 부호 비트의 과부하를 제거하고 WCSA cell 구조를 적용하여 CPA의 캐리 전달을 감소시켰다. 설계된 보조 프로세서는 verilog HDL을 사용하여 설계되었고 0.35um 3V 4층 금속 CMOS 표준 라이브러리를 사용하여 합성되었다. 설계된 승산기는 약 40MHz 의 동작속도와 약 10000개의 게이트 수, 그리고 약 10mW/MHz의 전력소모를 갖는다.

참고문헌(또는 Reference)

[1] A. D. Booth, "A signed binary multiplication technique," Quarter. J. Mech. Appl. Math, vol.4, Part2, 1951

[2] C.R. Baugh, B.A. Wooley, "A two's complement parallel array multiplication algorithm," IEEE Trans. Computer, Vol.C-22, pp1045-1047, Dec. 1973

[3] Kei-Yong Khoo, Zhan Yu, Willson, A.N., Jr., "Improved-Booth encoding for low-power multipliers," Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on , Volume: 1 , 1999, pp62 -65 vol.1

[4] Salomon, O. Green, J.-M. Klar, H., "General algorithms for a simplified addition of 2's complement numbers, Solid-State Circuits," IEEE Journal of , Volume: 30 Issue: 7 , July 1995, pp839 -844

[5] Bong-Il Park, In-Cheol Park, Chong-Min Kyung, "A regular layout structured multiplier based on weighted carry-save adders," Computer Design, 1999. (ICCD '99) International Conference on , 1999, pp243 -248