임베디드 시스템에 적용이 용이한 Booth 알고리즘 방식의 곱셈기 설계 (Design of a Booth's Multiplier Suitable for Embedded Systems)

문상국

목원대학교 정보전자영상공학부

Sangook Moon

Mokwon University, Division of Information-Electronics-Image Engineering

E-mail : smoon@mokwon.ac.kr

요 약

본 연구에서는 두 개의 17비트 오퍼랜드를 radix-4 Booth's algorithm을 이용하여 곱셈 연산을 수행하는 곱셈기를 설계하였다. 속도를 빠르게 하기 위하여 2단 파이프라인 구조로 설계하였고 Wallace tree 부분의 레이아웃을 규칙적으로 하기 위해서 4:2 덧셈기를 사용하였다. 회로를 평가하기 위해 Hynix 0.6-um CMOS 공정으로 MPW 칩을 제작하였다. 회로를 효율적으로 테스트하기 위한 방법을 제안하고 고장 시뮬레이션을 수행하였다. 설계된 곱셈기는 9115개의 트랜지스터로 구성되며 코어 부분의 레이아웃 면적은 약 1135*1545 mm2 이다. 칩은 전원전압 5V에서 24-MHz의 클럭 주파수로 동작하였음을 확인하였다.

ABSTRACT

In this study, we implemented a 17*17b binary digital multiplier using radix-4 Booth's algorithm. Two stage pipeline architecture was applied to achieve higher throughput and 4:2 adders were used for regular layout structure in the Wallace tree partition. To evaluate the circuit, several MPW chips were fabricated using Hynix 0.6um 3M N-well CMOS technology. Also we proposed an efficient test methodology and did fault simulations. The chip contains 9115 transistors and the core area occupies about 1135*1545 mm2. The functional tests using ATS-2 tester showed that it can operate with 24 MHz clock at 5.0 V at room temperature.

키워드

Booth, multiplier, Wallace tree

I. 서 론

지속적으로 정보화 사회로 변해가는 과정과 함께 각종 정보통신기기들의 소형화가 병행하게 된다. 이러한 각종 정보통신기기에는 각각의 시스템을 제어할 수 있는 uC(microcontroller)가 존재하게 되고 시스템이 고성능화가 될수록 DSP를 처리할 수 있는 하드웨어가 필수 불가결한데 이러한 DSP 처리 유닛의 핵심은 곱셈기이다.

이제까지 수많은 연구가 곱셈기의 효율적인 구현이라는 주제를 가지고 행해진 바 있는데, 그 많은 연구 중에서도 Booth 알고리즘을 응용한 곱셈

기의 구조가 아직까지 가장 효율적인 구조라고 알려져 있다. 중요한 것은, 그 효율적인 Booth 알고리즘을 어떻게 또한 효율적으로 구현하느냐에 대한 방법론적인 문제가 되는 것이다. 본 연구에서는 확장 Booth 알고리즘을 사용하여 곱셈기 구조를 실제로 칩으로 구현해 보고 이에 대한 예상되는 칩의 성능과 실제 구현한 칩으로 측정한 결과를 비교해 보았다. 실제로 풀커스텀 구조로 레이아웃을 수행하였고, 4:2 덧셈기까지 커스텀 구조로 디자인하여 최적의 성능을 보일 수 있도록 하였다.

II. 곱셈기의 구조

1. Block diagram

설계된 곱셈기의 전체 블록 다이어그램은 다음 그림 1과 같다.전체 블록은 크게 나누어 Booth encoding을 하여 Wallace tree까지 연산하는 부분과 최종 합 벡터와 캐리 벡터를 처리하는 덧셈기 단으로 나누어 볼 수 있다. 곱셈기 입력으로 받는 임시 저장소는 면적을 줄이기 위해서 플립플롭이 아닌 래치로 설계하였다.

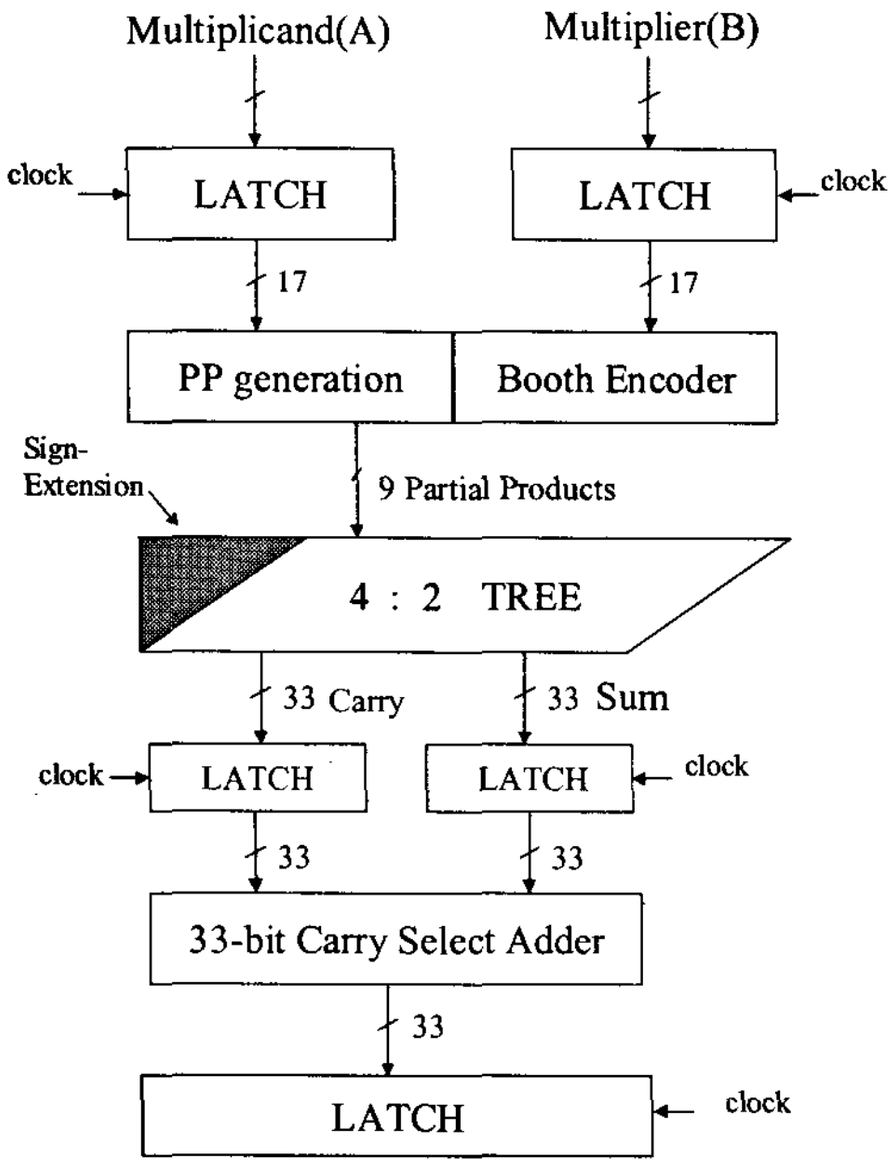

그림 1. 17b*17b 곱셈기의 전체 블록 다이어그램
Fig. 1. 17b*17b multiplier block diagram

전체 동작은 2단 파이프라인으로 구성되는데, 클럭 시작과 함께 입력 A와 입력 B가 래치에 저장되어 입력 A는 booth encoder 블록으로 들어가고 입력 B는 booth selector 블록으로 들어가게 된다. 17비트 입력A에 대한 부분곱은 radix-4 booth's algorithm에 의해 입력 B의 시퀀스에 따라 결정되어 각각 9개의 18비트 부분곱이 생성된다 [1]. 생성된 부분곱은 4:2 덧셈기로 구성된 wallace tree로 보내어져 33비트 합 벡터와 33비트 캐리 벡터로 표현된 다음 두번째 플립플롭 단으로 넘겨진다. 여기서 sign-extension 때문에 생기는 과도한 계산량을 줄이기 위해서 sign-generation이라는 방법을 사용하여 불필요한 하드웨어를 사용하지 않도록 하였다 [2].

최종적으로 생성된 캐리 벡터와 합 벡터는 33비트 CSA(Carry-Select Adder)에서 처리되는데 기본 덧셈기 블록으로 캐리 체인 덧셈기를 사용하였다 [3].

2. Booth encoder

Booth's algorithm에 따르면 입력 B의 시퀀스에 따라서 입력 A의 형태에 대한 +2A, +1A, 0A, -1A, -2A 중 하나를 택해야 하는데 이러한 선택이 이루어지는 블록이다. 입력이 17비트의 쌍이기 때문에 9개의 부분곱을 구해야 하는데 첫번째부터 8번째 부분곱을 구하는 회로까지는 B 입력의 이전 비트와 현재 두 비트를 참고하기 때문에 8개의 인코더가 동일하지만 9번째 부분곱에 대한 인코더는 표 1에서 보듯이 b15와 b16만 참고하면 가상적으로 sign-extension을 시켜 b17을 알 수 있기 때문에 회로를 보다 간단하게 할 수 있다.

3. 부분곱 생성 블록

입력 A가 첫번째 플립플롭 단을거친 다음 입력 B의 시퀀스에 따라서 부분곱을 생성하는 블록이다. 1에서 8번째 부분곱을 생성하는 블록에서는 5:1 mux를 사용하고 9번째 부분곱을 생성하는 블록에서는 3:1 mux를 사용하였다. mux의 구조는 그림 2에서와 같이 회로의 면적을 줄이기 위하여 weak pull-up PMOS를 사용하였다 [3].

표 1. 가상적으로 생성되는 b17비트의 값
Table 1. Presumably generated value of b17 bit

b17	b16	b15	select
0	0	0	0A
1	1	0	-1A
0	0	1	+1A
1	1	1	0A

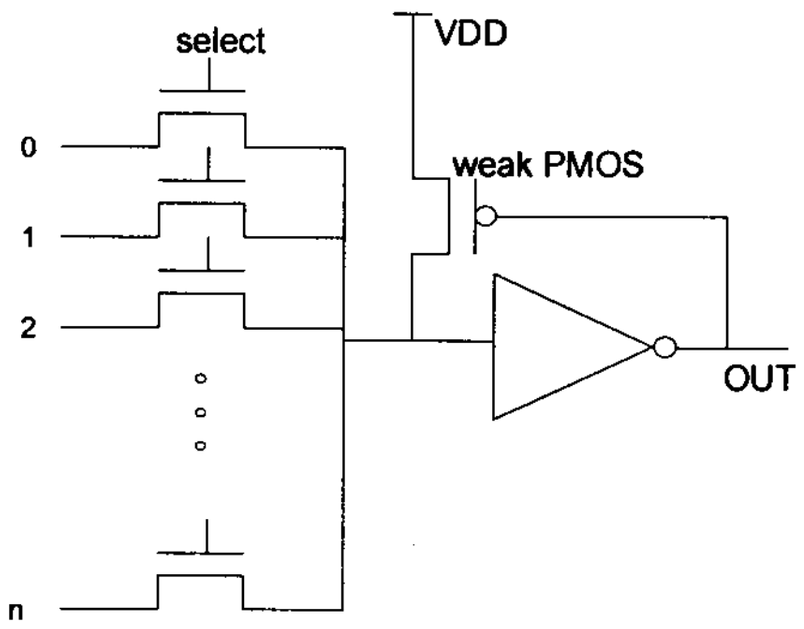

그림 2. Weak PMOS를 사용한 mux 구조
Fig. 2. Pull-up network design using weak PMOS

입력 B의 시퀀스에 따라서 +2A, +1A, 0A, -1A, -2A중 어느것을 선택할 것인지가 정해지면 select 비트 중 하나만이 일정 시간에 활성화 되고 그에따른 입력 A의 값들이 각각에 해당하는 형태로 쉬프트 되거나 보수화가 되어서 출력되어 wallace tree로 들어가게 된다.

4. Wallace 트리

이전 단에서 생성된 9개의 부분곱들을 고속으로 처리하기 위하여 CSA(Carry Save Adder) 구조를 사용하는데 3:2 덧셈기를 사용한 트리의 형태는 역삼각형 구조를 가지는 것에 비해 4:2 덧셈기를 사용한 형태가 레이아웃 상에서 규칙적인 구조를 이루기 때문에 본 논문에서는 4:2 덧셈기를 사용하였다. Wallace tree 안에서는 부호비트 확장 때문에 생기는 불필요한 계산을 하지 않기 위해서 sign-generation method [2]을 사용하여 면적을 최소화 하고자 하였다.

5. 캐리-선택 덧셈기 (carry-select adder)

Wallace tree에서 계산되어 나온 캐리 벡터와 합 벡터는 일단 플립플롭에 담겨진다. 플립플롭은 간단한 형태의 마스터-슬레이브 형태로 구성하였다. 다음 사이클에서 플립플롭의 출력들은 33비트 CSA로 들어가게 되는데 덧셈기의 기본 셀로는 캐리 체인 덧셈기 [3]를 사용하였다.

덧셈기의 최종 출력은 원하는 결과값이 되어 출력 버퍼를 통해 패드로 전해지게 된다.

III. 설계 결과 및 고찰

전체 설계의 front-end는 CADENCE의 Composer와 HSPICE로 작업을 수행하였다. 레이아웃은 전원과 diffusion을 가능한 한 공유하도록 하여 회로 면적의 최소화를 꾀하였다. 전체 회로의 레이아웃은 그림 3과 같다.

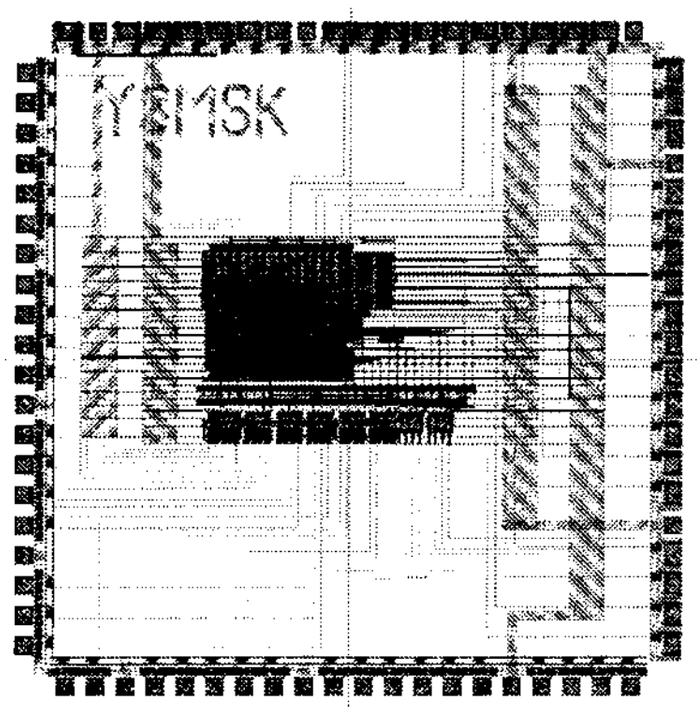

그림 3. 칩 마스크 레이아웃
Fig. 3. Chip mask layout

레이아웃은 CADENCE의 Virtuoso editor를 사용하였다. 회로를 추출 하기 전까지의 예상되는 지연 경로는 다음과 같다.

입력 B -> 플립플롭 -> booth encoder -> partial product generation -> wallace tree -> 두번째 플립플롭단의 입력

위와 같은 경로로 해서 예상해본 지연시간은 다음과 같았다.

플립플롭(1.3ns) + booth encoder(0.7ns) + partial product generation(1.2ns) + wallace tree(2.1ns*3) = 9.5ns

위의 예상되는 지연시간은 각각의 회로에 출력단에 해당하는 로드의 커패시턴스를 예상하여 스파이스 시뮬레이션을 수행한 값이다. Wallace tree 단에서의 계산치는 4:2 덧셈기의 지연시간을 2.1ns라고 측정한 다음 최대 지연 경로가 4:2 덧셈기를 3단으로 통과한다고 생각했을 때의 예상치이다.

다음으로, LVS를 통과한 레이아웃에서 커패시턴스와 레지스턴스를 추출 한 다음 간단한 테스트 패턴을 넣고 시뮬레이션 하여 보았다. 그 결과로 얻은 예상 결과는 그림 4와 같았다. 최대 지연시간의 저항치와 커패시턴스를 모두 고려한 예상치는 14.8ns였다. 이를 주파수로 환산하면 약 67MHz가 된다. 예상했던 약 100MHz 주파수와 차이가 나는 이유는 pp generation block에서 wallace tree로 값을 전달하는 wire의 길이가 실제 레이아웃 상에서 예상했던 것 보다 상당히 길게 그려졌기 때문이다.

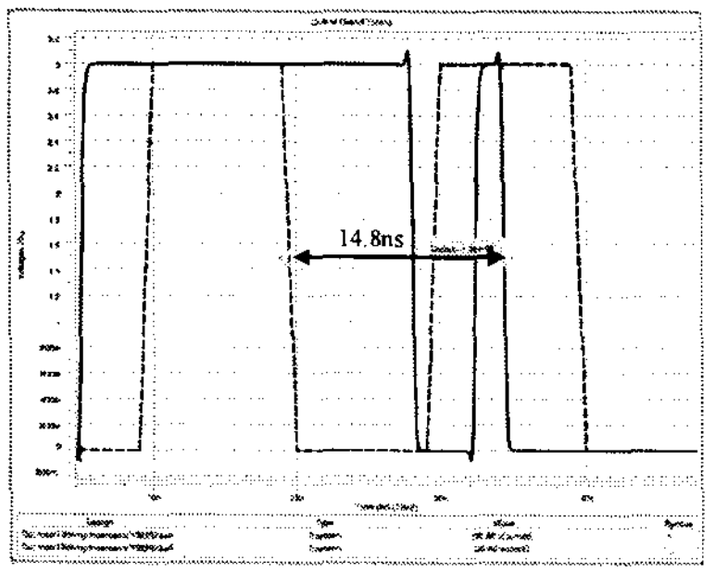

그림 4. HSPICE를 이용한 post-simulation 결과 최대 출력 지연 시간
Fig. 4. HSPICE result of worst-case propagation delay after the post-simulation

III. 결론

설계된 회로를 LG 0.6-um 3-metal 1-poly CMOS공정으로 제작하였다. 제작된 칩은 5.0V에서 약 24.5MHz의 주파수로 동작하는 것을 확인할 수 있었다. 그림의 결과는 앞의 고장시뮬레이션을 통해 얻은 37개의 테스트 패턴을 포함하여 약 2만개의 테스트 패턴을 입력으로 하여 테스트한 결과이다. C 프로그램을 이용하여 얻은 결과와 칩 테스터에서 결과로 출력된 값을 비교하여 논리 검증을 할 수 있었다.

칩의 특성을 살펴보면 전체적으로 전원 전압을 충분히 공급하는데 문제가 발생한 것을 알 수 있다. 예를 들어 전원 전압을 7.0V 까지 올렸을 경우에는 약 45MHz의 주파수에서 동작할 수 있지만 5.0V보다 약간만 낮은 전압을 가하면 주파수 특성이 현저히 떨어지는 현상을 보였다. 이에 대한 원인으로서는 우선 partial product generator에서 Wallace tree의 입력으로 연결되는 wire의 길이를 약 500um으로 보고 인버터의 크기를 결정하였었는데 실제 레이아웃을 하다 보니 몇몇 부분은 800um가 넘는 wire로 연결할 수밖에 없었고 이에 따른 인버터의 오버로딩 영향을 나중에야 알 수 있었다. 두번째로 칩에서의 사용가능한 핀 수가 80개인데 클럭을 포함한 입력이 35비트, 출력이 33비트이기 때문에 전원에 연결되는 핀 수를 vdd 6개, gnd 6개까지밖에는 할당할 수가 없었고 이는 전원 전압을 공급하기에 충분하지 못했다는 것을 보여준다.

또한 5.0V에서 전체 동작 주파수는 24.5MHz라고 측정되었지만 이 수치는 파이프라인의 첫번째 단인 booth encoder를 거쳐 부분곱을 생성한 다음 wallace tree를 거친 시간에 해당하는데, 파이프라인의 두번째 단인 두번째 플립플롭에서부터 덧셈기를 거친 출력핀까지의 지연 시간을 측정해 보았더니 6.6ns밖에 걸리지 않았다. 이는 주파수로 환산하면 약 150MHz에 해당하고 이는 파이프라인의 균형이 잘 맞지 않았다는 사실을 보여준다.

만일 차후에 다시 칩을 제작할 경우 입력A, B를 내부에서 멀티플렉싱하는 방법을 사용하고 사용하는 외부 핀 수를 줄여 전원 전압을 충분하게 공급해 줄 수 있다면 5.0V근방에서 45MHz 정도의 주파수 특성을 보여줄 것으로 생각된다. 또한, 파이프라인의 균형을 잘 맞추어 줄 경우에는 100MHz 이상의 동작 주파수를 얻을 수 있을 것이다.

IV. 참고문헌

[1] ISRAEL KOREN, "Computer Arithmetic Algorithms", University of Massachusetts, Amherst

[2] M. Annaratone, W.Z Shen, "The Design of a Booth Multiplier : nMOS vs. CMOS Technology"

[3] Yong S. Lee, "A Secondary Cache Controller Design for a High-End Microprocessor", IEEE Journal of Solid-State Circuits, Vol.27, No.8, pp1141-1146, August 1992

[4] Verifault-XL Reference Manual, CADENCE, 1997