fullcustom 0.35㎛ CMOS 공정을 이용한 16*16 bit 고속 승산기의 설계

박현규*, 신현철**,김종진***,
*,**,***부경대학교 전자공학과

Design of fast 16-bit multiplier with 0.35㎛ CMOS technology

(HYUN-KYU PARK* , HYUN-CHOL SHIN** JONG-JIN KIM***)
*,**,***Dept. of Electronics Engineering, PUKYONG national UNIV.

abstract

각종 범용 컴퓨터 및 디지탈 신호처리에서 중요한 역할을 하는 16비트 정수형, 2의 보수 형태의 곱셈연산을 수행하기 위한 고속 승산기구조를 설계하고 시뮬레이션 하였다. 부분곱을 합하는 부분은 일반적으로 전체 곱셈기 처리 지연시간의 절반정도를 차지하므로 이 부분의 설계방법이 곱셈기의 궁극적인 속도향상에 직접적인 영향을 미친다. 부분곱의 개수를 줄이기 위하여 Booth encoder를 사용하였고, partial product(부분곱)의 덧셈시간을 줄이기 위하여 4:2 CSA(carry save adder)와 3:2 CSA로 CSA tree를 구성 하였으며, 최종결과는 carry look- ahead tree로 얻어진다. Hyundai CMOS 0.35㎛ 1-poly 4-metal 공정으로 layout하여 설계하였으며, 곱셈시간은 2.7ns(tipical case)이하로 측정되었다.

Ⅰ. 서 론

오늘날 컴퓨터를 비롯한 디지탈 통신 및 신호처리 기술의 급속한 발전에 따라 데이타를 실시간 처리할 수 있는 고성능 연산장치의 필요성이 증대되고 있으며, 이의 구현을 위해서는 VLSI(Very Large Scale Integration) 기술을 바탕으로 한 고성능 병렬 승산기 등의 연산회로 설계기술이 필수적이다. 일반적으로 VLSI system의 설계에서 시스템의 처리속도를 증가시키는 것과 함께 면적을 줄이는 것이 중요한 요소로 되고 있다. 여기에 휴대용 멀티미디어 시스템 등에 응용하기 위하여 저전력 소모의 필요성도 증대되고 있다. 기본적으로 speed, area, power 는 모두 서로 상반된 관계, 즉 한쪽의 성능을 향상시키면 다른 한쪽은 손실을 보게 되는 관계에 있으므로 응용분야에 맞도록 이에 대한 적절한 trade off가 필요하다.

그림.1 에 보인 바와 같이 곱셈기는 크게 세 부분의 functional block으로 나눌 수 있는데, 입력으로 부터 들어오는 피승수(multiplicand)와 승수(multiplier)로 부터 partial product들을 만들어 내는 인코더(encoder) 부분과 그것들을 모두 더하여 두 줄의 partial product를 만들어 내는 adder array 부분과, 최종 결과를 만들어 내는 final adder 부분이 그것이다.

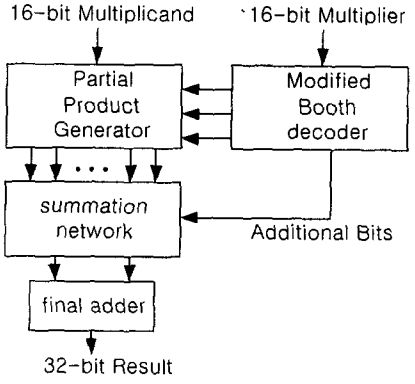

그림 1. 일반적인 병렬 승산기의 블록도

기본적으로는 N-bit word 끼리의 곱셈에서는 N에 비례하는 개수의 부분곱들이 만들어지고 이들을 순차적으로 더하기 위하여 N에 비례하는 덧셈 수행시간이 걸린다. 현재까지 알려진 곱셈기 중 가장 빠른 것으로 알려진 곱셈기의 형태는 부분곱들을 생성하는 인코더에는 modified Booth algorithm을 사용하고 부분곱들을 더하는 adder array에는 Wallae tree를 이용하는 방식이다.

modified Booth algorithm 방법을 사용하면 부분곱들의 개수를 $\lfloor \frac{N}{2} \rfloor$ 개로 줄일 수 있다. 이러한 경우에는 CSA stage 개수가 한 단계 줄어들기 때문에 속도 및 interconnection에 소요되는 면적을 많이 줄일 수 있다. 또한 이와 동시에 2의 보수로 표현된 word 들의 곱셈도 자동적으로 가능해진다는 장점이 있다. CSA tree는 부분곱 생성기에서 나오는 부분곱들의 덧셈을 최대한 병렬로 수행하여 N-bit word의 곱셈을 위하여 $O(\log_2 N)$ 에 비례하는 수행시간을 가진다. 즉, CSA를 사용하여 출력 비트의 개수는 $\log_2 N$개로 줄어드는 원리를 이용한다.

II. 16비트 승산기의 설계

일반적으로 Partial product들의 합을 계산 할 때, 2's complement number의 경우 각 partial product 의 sign bit를 연산에 필요한 최대 자리수 까지 확장한 후 계산해야 한다. 이때 필요한 부가적인 adder를 sign generate method를 이용하면 각 partial product 에 2bits 확장으로 같은 결과를 얻을 수 있다. 설계된 승산기는 partial product의 수를 줄이기 위한 modified Booth algorithm과 2's complement 계산을 위한 sign extension 부분을 대신하여 최소한의 hardware로 구현할 수 있는 부호생성기법(sign generation method)를 적용하였다. 그림.2는 설계한 승산기 회로에 적용시킨 radix-4 Booth encoding table과 부호확장 방법을 설명한다. partial product selector 회로의 좀더 자세한 블록 다이아 그램을 그림.3 에 보였다. 그림.3 에서 Booth Decoder가 많은 수의 MUX회로를 구동시켜야 함을 알 수 있는데, 이럴 경우 fan-out 문제가 발생하게 된다. fan-out이 커지게 되면 추가 전력소모와 delay를 발생시키게 된다. 이를 해결하기 위하여 Booth decoder와 MUX 사이에 시뮬레이션을 통하여 적절한 구동능력을 가지는 버퍼를 추가하였다.

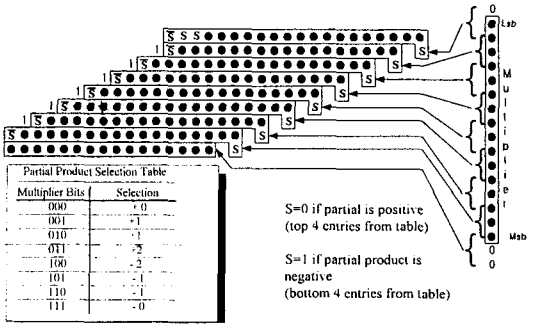

그림 2. 16bit radix-4 Booth multiplication

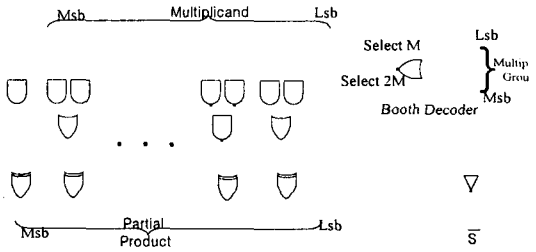

그림 3. partial product selection logic

partial product 생성기를 통하여 나온 부분곱들을 두 줄의 부분곱으로 만들기 위한 summation network의 구조는 CSA tree를 적용하였다. CMOS에서 주된 전력소모는 신호의 dynamic transition에 의해서 일어나므로 신호의 변화가 잦을 수록 전력 소모가 커진다. CSA tree 구조를 기준으로 했을 때, array type 구조는 속도가 느릴 뿐 아니라 transition 횟수에서 5배 이상 차이가 나는 것을 table.1[참고]에서 확인 할 수 있다. 때문에 저전력 소모를 위해서는 CSA tree 구조가 적합함을 알 수 있다.

Type	Delay	Number of Gates	Transition/Mult.
array	3.14	1.05	5.06
CSA tree	1	1	1

table 1

CSA tree 구조는 대개의 경우 매우 불규칙적인 상호결선구조로 되어 있다. 이 불규칙성은 구현을 복잡하게 하고 area 사용 효율을 떨어뜨리는

데, level수가 적을수록 불규칙성이 줄어들게 된다. tree에서 level의 수는 3:2보다 높은 reduction rate를 사용함으로서 낮아질 수 있다. 본 논문에서는 배치 및 배선의 복잡도를 최대한 줄일 수 있게 부분곱을 재 정렬하고, 속도를 떨어뜨리지 않으면서 비교적 면적을 적게 차지하게 만들어진 4-2 CSA와 3-2 CSA를 조합하여 수정된 형태의 CSA tree 구조를 제안하였다. (4,2) counter를 사용하면 (3,2) counter에 비하여 더 많은 gate가 소요되지만, 2:1의 reduction rate 를 얻을 수 있고, 좀더 규칙적인 layout을 할 수 있다. 설계에 사용된 (4,2) counter는 그림.4 에 나타내었다.

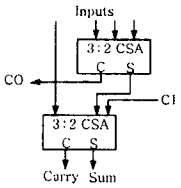

그림 4.a 블록도

l_1+l_2	l_3+l_4	Cout	SUM	Carry
0	0	l_1	Cin	l_4
0	1	l_1	Cin	Cin
1	0	l_3	Cin	Cin
1	1	l_3	Cin	l_4

그림 4.b 진리표

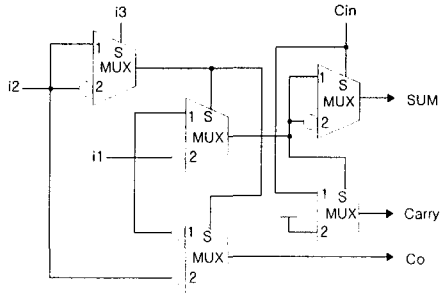

그림 4.c 회로도

부분곱 생성기에 만들어진 아홉 줄의 partial product를 두 줄의 final product로 만들어 내기 위하여 CSA routing에 사용된 방법은 부분곱들의 처음 여덟 줄은 8-2 compressor를 구성하고, 이것의 출력과 마지막 줄 을 3-2 CSA 및 HA(half-adder)로 묶었다. 이렇게 연결한 결과, 약 145개의 FA(full adder)가 쓰였고, summation network의 critical path는 단지 10개의 MUX만 거치게 하여 속도향상을 얻을 수 있었다.

final adder로 Ripple carry adder를 사용하면 비트 수에 비례하여 carry에 의한 delay가 매우 커지게 되므로, 다음 단의 carry를 미리 계산 해주는 CLA(Carry- Look ahead Adder)를 사용하였다. CLA 는 비트 수가 커질수록 large fan-in 문제가 발생하므로, 본 설계에서는 8개의 4비트 CLU(carry look-ahead unit)와 한 개의 8 비트 CLU로 구성된 CLA tree를 사용하였다.

그림.5 에는 알테라 MAX-plusII 를 사용하여 설계된 승산기의 functional simulation을 결과를 보였다.

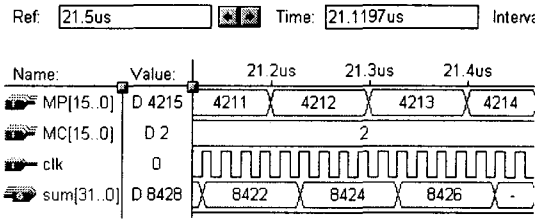

그림 5. 승산기의 functional simulation

Ⅳ. 결론

본 논문에서는 16비트 정수형, 2의 보수 곱셈 연산을 수행하기 위한 고속 승산기 구조를 설계하고 시뮬레이션 하였다. 승산기의 구조는 계산 시간과 면적을 줄이기 위하여, radix-4 modified Booth 알고리즘과 CSA tree 및 CLA를 이용하여 설계되었다. Hspice 시뮬레이션으로 측정한 곱셈계산의 최대 지연시간은 2.64ns 이다. Cadence 로 layout 한 결과 core layout이 차지하는 면적은 약 1m×1m(chip area: 4m×4m) 정도이고, transistor 개수는 9702개 이다. 이 승산기의 core Layout은 그림.6 에 보였다.

설계된 승산기는 고속동작 특성을 가지면서도 area가 크지 않으므로, 32비트 혹은 54비트로의 확장이 가능하며, 이러한 고속 곱셈기는 향후 실시간 data 처리가 요구되는 다양한 DSP 응용에 특히 유용하게 사용될 수 있을 것으로 기대된다. 앞으로 Chip의 속도를 떨어뜨리지 않으면서 면적을 적게 차지하는 회로를 대체 함으로서 면적 및 전력 소모를 더욱 줄이는 일이 남아 있으며 이러한 새로운 설계기법의 개발 및 이에 수반되는 문제점을 줄이는 방법에 대한 연구가 지속적으로 필요하며 이러한 연구 주제를 향후 해야 할 일로 남기고 본 논문의 결론을 맺는다.

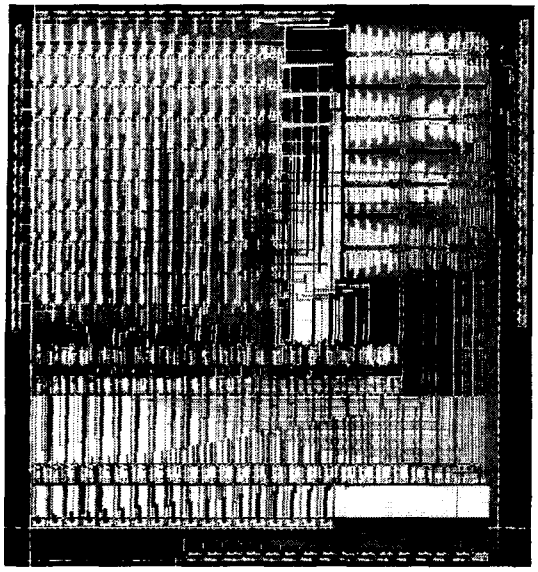

그림 6. multiplier chip core layout

참고 문헌

[1] Gary W.Bewick "fast multiplication: algorithms and implementation", stanford univ. press

[1] Patrik Larsson and Chris J. Nicol, "Transition Reduction in Carry-Save Adder Trees," *International Symposium on Low Power Electronics and Design*, 1996.

[2] Akilesh Parameswar, Hirotuki Hara, and Takayasu Sakurai, "A High Speed, Low Power, Swing Restored Pass-Transistor Logic Based Multiply and Accumulate Circuit for Multimedia Applications," *IEEE Custom Integrated Circuit Conference*, 1994.

[3] Issam S. Abu-Khater, Abdellatif Belaouar, and M. I. Elmasry, "Circuit Techniques for CMOS Low-Power High-Performance Multipliers," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 10, pp. 1535-1546, Oct. 1996.

[4] Paul J. Song and Giovanni De Micheli, " Circuit and Architecture Trade-offs for High-Speed Muliplication," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 9, pp. 1184-1198, September 1991.

[5] Koren, Israel "COMPUTER ARITHMETIC ALGORITHMS", PRENTICE HALL, 1993.

[6] THOMAS A. DeMASSA and ZACK CICCONE "DIGITAL INTEGRATED CIRCUITS", WILEY, 1996.