부동 소수점 유닛의 고속처리를 위한 가산기 모듈의 설계 및 검증

정명수, 손승일

한신대학교 정보통신학과

Design and Verification of Adder Module for Fast Floating-Point Unit

Myung Su Jung, Seung Il Sonh

Dept. of Information and Communication HanShin University

E-mail : redpunki@hotmail.com

요 약

1970년대 말까지 초창기에 출시된 컴퓨터들은 부동 소수점을 표현하기 위한 자신들의 내부적 표현 방식을 사용하였다. 따라서 각 컴퓨터마다 부동 소수점 연산에 대한 계산 결과가 약간씩 차이가 나기도 하였다. 이러한 문제점을 해결하기 위해 IEEE에서는 부동 소수점에 대한 표준안을 제안하였다. 이는 서로 다른 컴퓨터 간에 부동 소수점 데이터의 교환이 가능하게 할 뿐만 아니라 하드웨어 설계자들에게도 정확한 모델을 제공하는 것이 목적이었다. 이 당시 제정된 부동 소수점 표준안은 IEEE Standard 754 부동 소수점이며, 오늘날 인텔 CPU 기반의 PC, 매킨토시 및 대부분의 유닉스 플랫폼에서 컴퓨터 상의 실수를 표현하기 위해 사용하는 가장 일반적인 표현 방식으로 발전하였다. 본 논문에서는 부동 소수점의 기본적인 표현방식에 대해 연구하고, 이 중 32 bit 단일 정밀도 부동 소수점 가산기를 Microsoft Visual C++ 6.0을 이용해 시뮬레이션하고 이를 VHDL로 구현한다.

Ⅰ. 서론

우주과학, 그래픽 이미지 처리, 통계 등 오늘날의 복잡한 연산은 모두 컴퓨터가 처리를 한다. 이러한 연산에서는 분수(fraction)와 지수(exponent)를 사용하는 부동 소수점 표현 방식을 널리 사용되고 있다. 부동 소수점 연산기는 반도체 분야의 발전으로 칩(Chip)의 집적도가 증가함에 따라서 중앙처리 장치(Central Processing Unit)와 함께 한 칩에 내장되어 발전되면서 주 연산기의 중요한 요소로 등장하고 있다. 이처럼 부동 소수점 연산기는 보조적인 기능을 뛰어넘어 주 연산기의 중요한 요소로 사용된다[2].

본 논문에서는 IEEE Standard 754에서 지정한 부동 소수점 표준안의 기본적인 표현방식에 대해 연구하고, 이 중 32 bit 단일 정밀도 부동 소수점 가산기를 Microsoft Visual C++6.0을 이용해 시뮬레이션한다. 그리고 하드웨어 설계언어인 VHDL로 구현하고 최종적으로 Model Sim6.0a를 이용하여 파형을 통하여 연산결과를 검증하고자 한다.

Ⅱ. 본론

2.1 IEEE 754

부동소수점 수의 표현 방식의 통일을 위하여 IEEE에서 정의한 표준이다. 컴퓨터 제조회사들이 부동 소수점을 사용하기 위하여 각기 다른 표현 방법을 사용함으로써 표준화가 요구되자, IEEE 위원회는 1985년 부동 소수점을 위하여 IEEE 754라는 표준을 제정하였다. 현재 대부분의 PC와 워크스테이션에서 IEEE 표준을 이용하고 있다[1].

2.2 부동 소수점 표현 방법

2.2.1 단일정밀도

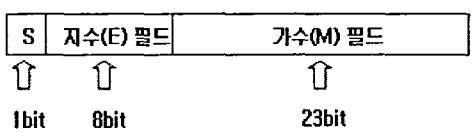

그림 1. IEEE 표준 단일정밀도 형식

단일정밀도는 1비트의 부호 비트와 8비트의 지수 비트 그리고 23비트 유효자릿수로 표현된다. 유효자릿수는 소수점의 왼쪽에 1이 있는 것으로 하며, 정규화된 유효자릿수는 1에서 2사이의 값을 갖는다. 지수는 127의 바이어스 사용하는데 그 범

위는 '-126 ~ 127'이며, 지수값 중 0000 0000 (-127)과 1111 1111 (128) 은 예약되어 있다.

$$N = (-1)^S \, 2^{E-127} \, (1.M) \quad \text{(식 1)}$$

부동 소수점 N은 식 1과 같이 나타낼 수 있으며 지수는 $1.M \times 2^E$의 형태를 가지며, 소수점 아래 M 부분만 가수 필드에 저장된다. 여기서 소수점 왼쪽의 표현되지 않는 1을 hidden bit라고 지칭한다[1][3][4].

2.2.3 IEEE 표현의 예

부동 소수점 N을 -13.625라고 하였을 때, 식 1과 같이 표현하면 식 2과 같이 나타낼 수 있다.

$$13.625 = 1101.101 = 1.101101 \times 2^3 \quad \text{(식 2)}$$

식 2을 이용하여 단일정밀도 형식에 맞게 표현하면 그림2과 같다.

부호(S) 비트 = 1 (음수)

지수 E = 00000011 + 01111111 = 10000010
(바이어스 127를 더함.)

가수 M = 10110100000000000000000
(소수점 좌측의 1은 제외.)

그림 2. 단일정밀도 형식 표현

그림2의 내용을 나타낸 것이 그림3이다.

0	10000010	10110100000000000000000

그림 3. "-13.625" 의 단일정밀도 형식

2.3 오버플로우와 언더플로우

그림1, 그림2에 표현된 지수와 유효자리 수의 크기는 컴퓨터의 산술이 매우 큰 수의 표현범위를 갖게 한다. 매우 큰 수를 표현할 수 있다고는 하지만 이것은 무한대와는 큰 차이가 있다. 따라서 부동 소수점으로 표현된 수가 이 범위보다 더 커질 가능성이 있다. 그러므로 오버플로우(overflow) 인터럽트는 정수연산에서 뿐만 아니라 부동소수점 연산에서도 발생할 수 있다. 여기서 오버플로우라는 것은 지수의 값이 지수부분에 표현될 수 없을 만큼 클 때를 말한다.

단일정밀도의 경우 표시할 수 있는 가장 작은 숫자는 $N_{min}=\pm 2^{-127}$(배정밀도의 경우는 $N_{min}=\pm 2^{-1023}$)이며, 이보다 더 작은 숫자를 나타내기 위해서는 비정규(denormalized) 숫자(그림4의 언더플로우(underflow)에 해당)를 사용해야 한다. 비정규로 표시할 수 있는 가장 작은 숫자는 단일정밀도의 경우 $N_{min}=\pm 2^{-149}$이다[1][3].

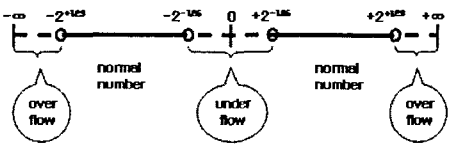

그림 4. 오버프로우와 언더플로우의 범위

2.4 Rounding

부동 소수점 산술 유닛을 통해 얻은 결과의 정확도는 산술 유닛에서 계산된 중간 결과가 정확하다고 할지라도 제한되게 된다. 계산된 수의 자리수가 IEEE 형식에서 허용한 전체 자리수를 초과할 경우가 있을 수 있다. 일반적으로 최종적인 결과 값이 사용자가 액세스할 수 있는 레지스터나 혹은 메모리에 저장되기 전에 초과한 자리수를 처리해야 한다. 반올림 모드는 비교 및 나머지 연산을 제외한 모든 산술 연산의 결과에 영향을 미친다. 부동 소수점 관련 IEEE 표준안 754에서 규정한 반올림 모드는 다음의 4가지가 있다[1].

1 . Round to Nearest
2 . Direct rounding
  (1) Round to Zero(0)
  (2) Round to Positive Infinity(+∞)
  (3) Round to Negative Infinity(-∞)

이 중 Round to nearest을 일반적으로 기본 반올림 모드로 사용하고 있다.

2.4.1 Round to Nearest

가장 가까운 쪽으로 반올림 혹은 반내림하는 방법으로 양쪽이 똑같이 가까운 경우 LSB(Least Significant Bit)가 0이 되도록 하는 방법이다. 이 방법이 가장 많이 사용되는 방법으로 이렇게 하기 위해서는 배럴 쉬프터(barrel shifer)에서 guard, round, sticky 비트를 검출해야 한다.

2.4.2 Direct Rounding

+∞, -∞ 또는 0을 향해서 반올림 또는 반내림하는 방법이다.

2.5 Guard, Round, Sticky

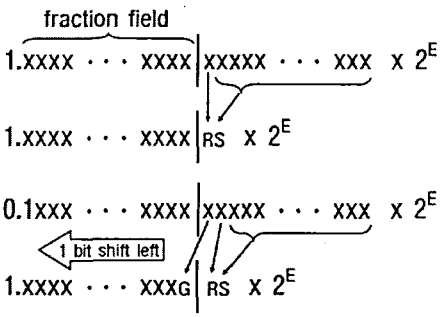

그림 5. Guard, Round, Sticky 비트의 정의

G(Guard) 비트는 가수의 표현 범위를 벗어난 첫 번째 비트를 의미하며, R(Round) 비트는 G비트 다음에 오는 비트를 의미하며, S(Sticky) 비트는 R비트 다음에 오는 나머지 전체 비트를 논리합(OR)하여 얻는다. 반올림을 수행할 때, 실제로 R, S 2비트만 필요한데, 때에 따라 가수 표현 범위에서 벗어났던 1비트가 다시 표현 범위로 이동하는 경우를 대비해서 G 비트를 포함해서 3비트를 사용한다. S 비트는 어느 쪽으로 반올림하여도 그 오차가 동일한 경우를 검출하기 위하여 필요하다[5][6][7].

III 시뮬레이션

3.1 Visual C++을 이용한 시뮬레이션

그림6은 Microsoft Visual c++ 6.0을 이용해서 두개의 실수를 단일 정밀도의 부동소수점 형식으로 가산하는 결과를 보여주고 있다.

반올림 모드로 Round to Nearest Even과 Round to Positive Infinity, Round to Negative Infinity를 구성하였다. 실수 A와 B를 입력하고 덧셈 버튼을 클릭하면 실수가 내부적으로 연산되어 부동소수점 형식의 부호(1 bit)와 지수(8 bit), 가수(32 bit)로 분류하여 비트열로써 출력한다. G는 Guard bit이고 R은 Round bit, S는 Sticky bit를 나타낸다.

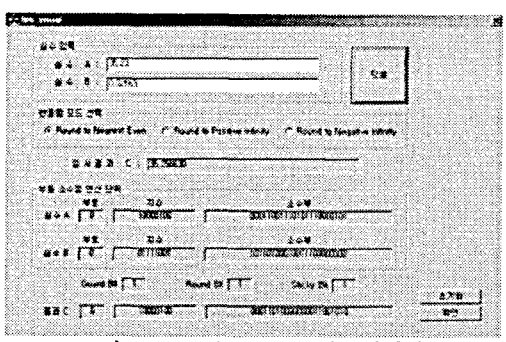

그림 6. Visual c++ 6.0 시뮬레이션

IV. FPA 설계

4.1 부동소수점 가산기의 연산과정

부동소수점연산의 기본적인 과정은 그림7과 같은 순서로 진행된다.

두 수의 입력을 받아서 지수를 비교해서 작은 수를 쉬프트시키고 가산을 해서 정규화를 시킨다. 그리고 반올림 모드에 따라 반올림의 여부를 결정하고 최종적으로 다시 정규화한다.

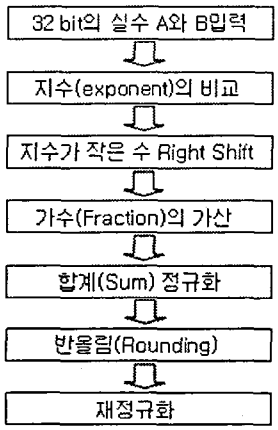

그림 7. 부동소수점 가산기의 연산과정

4.2 부동소수점 가산기 설계구성

그림8에서 볼 수 있듯이 부동소수점 연산과정이 4 step으로 구성되어져 있다.

A와 B 각각의 수로부터 32비트를 입력받으면 step1에서 AbsDiff모듈로부터 지수의 차가 연산되고 Compar모듈에서 부호를 제외한 절대값의 대소비교를 통해 큰 수를 A로 작은 수를 B로 스왑한다.

*step2*에서는 *ShiftR*모듈로 절대값이 작은수를 받아 지수의 차만큼을 오른쪽 쉬프트하고 Guard/Round/Sticky bit를 가지게 된다. Decod 모듈은 두 수의 절대값의 대소비교에 따라 지수값을 마스크를 통하여 셋팅해준다.

step3에서는 ZLPC모듈을 통해 앞에 0의 개수가 몇 개인지 정규화가 되었는지에 대한 정보를 알려준다. CompoundAdder모듈은 부호비트와 가수비트 그리고 나머지비트들을 가지고 알고리즘에 의해서 가산과 반올림 연산을 하게된다.

마지막으로 step4에서는 ShiftL모듈에서 재정규화를 시켜주고 AddEx모듈에서 결과에 따라 지수를 가산시켜준다.

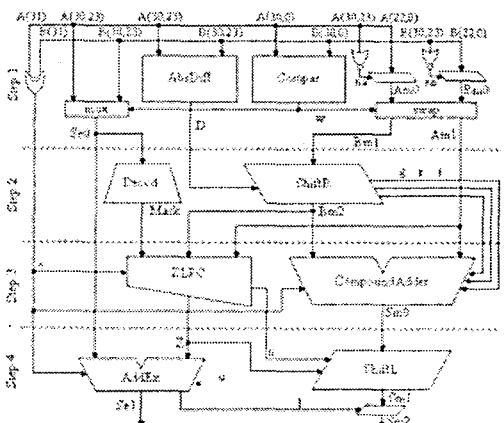

그림 8. 부동소수점 가산기 전체 블록도

4.3 Model Sim을 이용한 검증

그림9는 VHDL언어를 사용하여 얻은 부동소수점 가산기의 출력파형을 보여주고 있다. valid 신호가 활성화 되면 두개의 32bit 실수를 입력받아 4step의 클럭을 거쳐 최종 연산결과가 출력된다.

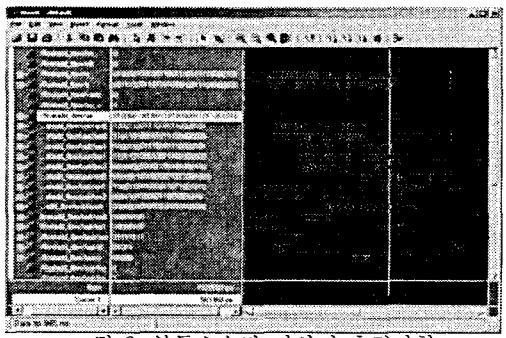

그림 9. 부동소수점 가산기 출력파형

4.4 합성 결과

본 논문에서 설계한 부동소수점 가산기 모듈의 각 등가게이트와 사용 슬라이스 수 그리고 타이밍 결과가 표1과 같이 나타났다.

표 1. 설계 합성 결과

Num of Slices	Num of Gates	Timing	Target Device
679	9,993	Mininum Period: 13.809ns Maximum Frequency: 72.417MHz	xc2s100-6 -pq208

V. 결론

부동 소수점 가산기는 FPU 설계의 핵심 유닛으로 사용되며 데이터의 수치연산을 하는 일에 다양하게 사용되어 진다. 연산량이 큰 데이터 처리에 주로 이용되기 때문에 불필요한 처리를 되도록 줄여서 설계해야 한다.

본 논문에서는 부동소수점 연산처리에 중요한 요소인 저전력과 고속 처리를 위해 step마다 제어신호를 두고 현재 step이 아니면 연산하지 않도록 하고있다. 먼저 부동소수점 형식에 대해 연구하고 연산과정대로 Visual c++로 시뮬레이션을 해서 최종적으로 VHDL언어를 이용하여 하드웨어로 구현하였다.

설계된 부동소수점 가산기 모듈은 FPU의 다른 모듈들과 함께 단일 코어로 최적화되어서 칩으로 설계되어 DSP분야와 그래픽 프로세싱 분야는 물론 여러 과학 연산 분야 등에서 활용될 수 있다.

참고문헌

[1] ANSI/IEEE Standard 754-1985 for binary Floating-Point Arithmetic, IEEE Computer Society Press, Los Alamitos, Calf., 1985.

[2] 박우찬, 정철호, 양진기, 한탁돈, "IEEE 반올림과 덧셈을 동시에 수행하는 부동 소수점 곱셈 연산기 설계", 전자공학회논문지, vol.34-C, no.11, 1997.11.

[3] 이용석, "60MHz Clock 주파수의 IEEE 표준 Floating Point ALU", 전자공학회논문지, vol.28-A, no.11, 1991.11

[4] Loucas Louca, Todd A. Cook, William H. Johnson "Implementation of IEEE Single Precision Floating Point Addition and Multiplication on FPGAs", Rutgers University, 1996.9

[5] 박우찬, 이시화, 권오영, 김신동, 한탁돈, "Floating Point Adder / Subtractor Performing IEEE Rounding and Addition / Subtraction in Parallel", IEICE TRANS.INF. & SYST., vol.E79-D, no.4, April 1996

[6] Nhon Quach and Michael Flynn, "Design and Implementation of the SNAP Floating-Point Adder", Technical Report CSL-TR-91-501, Stanford University, Dec. 1991

[7] Numerical Computation Guide, Sun Microsystems, 1996.