論文 96-21-5-24

수퍼스칼라 마이크로프로세서용 부동 소수점 승산기의 설계

正會員 崔 炳 允*, 李 文 基**

A Design of Floating-Point Multiplier for Superscalar Microprocessor

Byeong Yoon Choi*, Moon Key Lee** *Regular Members*

※이 논문은 1994년도 학술 진흥 재단 공모과제 연구비에 의해서 연구되었음

요 약

본 논문은 SD수 표현 체계에 바탕을 둔 기수-16 레코딩 기법과 새로운 반올림 및 정규화 기법을 조합한 수퍼스칼라 마이크로프로세서용 부동 소수점 승산기 회로(FMUL)에 대해 기술한다. 새로운 반올림 및 정규화 기법은 설계된 승산기 회로가 곱셈 동작과 스티키 비트 계산 동작을 병렬적으로 수행할 수 있도록 하며, 재정규화에 따른 시간 지연 문제를 해결할 수 있도록 한다. 그리고 SD 숫자 체계의 기수-16 레코딩 기법을 사용함으로써, 실리콘 면적과 계산 시간을 더욱 줄일 수 있다. 부동 소수점 승산기는 3단 파이프라인 구조를 통해 단정도와 배정도 부동 소수점 연산을 수행할 수 있고 IEEE 표준을 만족한다. 설계된 부동 소수점 승산기의 알고리즘과 구조는 Verilog HDL 모델에 의해 모의실험을 통해 성공적으로 검증되었다. 검증된 부동 소수점 승산기 회로는 0.6 m CMOS 표준 셀을 사용하여 하드웨어로 구현한 후 타이밍 특성을 분석한 결과, 동작 주파수는 67 Mhz이며, 최대 67 MFLOPS (Million Floating Point Operations Per Second)의 부동 소수점 곱셈 연산 능력을 갖고 있음이 확인되었다.

ABSTRACT

This paper presents a pipelined floating point multiplier(FMUL) for superscalar microprocessors that combines radix-16 recording scheme based on signed-digit(SD) number system and new rounding and normalization scheme. The new rounding and normalization scheme enables the FMUL to compute sticky bit in parallel with multiply operation and eliminate timing delay due to post-normalization. By expoliting SD radix-16 recording scheme, we can

*동의대학교 컴퓨터 공학과
Dongeui University, Dept. of Computer Eng.
**연세대학교 전자 공학과
Yonsei University, Dept. of Electronic Eng.
論文番號:96091-0314
接受日字:1996年 3月 14日

achieves further reduction of silicon area and computation time. The FMUL can execute single-precision or double-precision floating-point multiply operation through three-stage pipelined datapath and support IEEE standard 754. The algorithm and structure of the designed multiplier have been successfully verified through Verilog HDL modeling and simulation.

This multiplier achieves a peak performance of 67 MFLOPS double or single precision multiply operation with 67 Mhz clock, a latency of 3 cycles, and a throughput of 1 cycle under 0.6 m CMOS double-metal technology.

Ⅰ. 서 론

반도체와 컴퓨터 기술의 급속한 발달로, 최근에 개발되고 있는 명령어 축소형(reduced instruction set computer : RISC) 마이크로프로세서는 정수처리부, 캐쉬 메모리, 메모리 관리 회로및 부동 소수점 처리 장치를 단일 칩에 내장하는 추세이다. 그리고 성능을 극대화하기 위해, 수퍼 스칼라 처리 기법과 같은 명령어 수준의 병렬 처리 개념이 아키텍쳐에 구현되고 있다[1]. 이러한 수퍼 스칼라 처리 기법은 명령어의 동적인 스케쥴링에 바탕을 둔 병렬처리와 파이프라인 처리 개념을 사용하고 있다. 현재 이러한 RISC 마이크로프로세서를 빠른 컴퓨터 연산 능력이 요구되는 디지탈 신호처리, 그래픽과 같은 응용 분야에 사용되기 위해서는 고성능의 부동 소수점 처리 장치가 필수 불가결하다. 그러나 기존의 부동 소수점 처리 장치나 수치 보조 프로세서는 하드웨어와 칩 면적을 고려하여, 파이프라인 처리 구조를 사용하지 않고, 반복적인 동작으로 곱셈을 구현한다[2]. 따라서 이러한 구조는 최근의 수퍼스칼라 프로세서에 내장되기 위해서는 부동 소수점 연산 장치의 구조 변형이 요구된다. 그리고 1985년에 IEEE학회가 규정한 부동 소수점 데이타의 연산에 관한 표준안을 만족시키기 위해서는, 4가지 반올림 알고리즘과 5가지 예외(exception) 처리 회로가 필요하다[3].

본 논문에서는 고성능의 수퍼 스칼라 마이크로프로세서에 내장하기 위한 새로운 구조의 부동 소수점 승산기 회로를 제안하고, 이를 상용화된 셀 라이버러리를 사용하여 하드웨어로 구현한 후 성능을 분석하였다.

본 논문의 II장에서는 FMUL의 설계 사양을 설명하고, III장에서는 2진 SD(signed digit) 숫자 체계를 사용한 곱셈 알고리즘, 새로운 반올림 및 정규화 알고리즘 및 FMUL의 하드웨어 설계를 기술하고, IV장에서는 설계한 회로에 대한 모의 실험 결과 및 성능 분석에 대해 기술하고, V장에서는 결론을 맺는다.

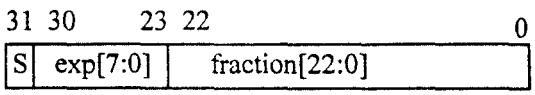
Ⅱ. 부동 소수점 승산기의 설계 사양

본 승산기 회로에서는 그림 1에 보이는 2가지 형태의 부동 소수점 데이타에 대한 곱셈만을 지원한다. IEEE 표준안에서 원래 3가지 형식(단정도, 배정도, 확장 정밀도)으로 나누어 진다. 확장 정밀도 데이타를 구현치 않은 것은 사용 빈도가 높은 연산은 하드웨어로 구현하고, 상대적으로 사용 빈도가 낮은 연산은 소프트웨어를 이용하는 RISC의 설계 기법을 채택했기 때문이다. 부동 소수점 형식의 수가 나타낼 수 있는 값 F_{val}은 식(1)과 같이 정해진다. 여기서 significand는 1 비트의 정수부와 p−1 비트의 분수부(fraction)로 표시된다.

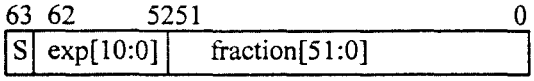
$$F_{val} = (-1)^S 2^E (1.f_1 f_2 \ldots f_{p-1}) \quad (1)$$

여기서 $s = 0$ 또는 1, $exp = E + bias$

$f_i = 0$ 또는 1



(a)



(b)

그림 1. 부동 소수점 데이타 형태
(a) 단정도 데이타 (b) 배정도 데이타

Fig. 1 Floating-point data formats
(a) single-precision data (b) double-precision data

IEEE 표준안은 일반 부동 소수점 데이타 이외에 비정규화(denormalized) 데이타, 무한대(∞)와 NAN(Not-A-Number)이라는 비 수치 데이타도 함께 나타낼 수 있다. 본 연구에서는 비정규화 데이타, NAN 데이타는 하드웨어로 지원치 않고 소프트 웨어로 처리하도록 하였다.

반올림(rounding)이란 무한대 정밀도(precision)의 결과를 최종 결과 형태가 요구하는 유한한 정밀도로 제한하는 방안으로, 버림되는 부분을 어떻게 최종 결과에 반영해주는가 하는 것이다. 현재 부동 소수점 데이타에 대한 IEEE 표준 반올림 알고리즘은 4가지 방식으로 세분된다. 본 연구에서는 4가지 반올림 알고리즘을 4가지 비트 L(LSB), G(guard), R(round), S(sticky) 비트를 이용하여 하드웨어로 구현하도록 하였다. 이외에도 IEEE 표준안은 비정상적인 상황 발생을 처리하기 위해 5가지 예외 처리조건(invaild operation, divide-by-zero, overflow, underflow, inexact)을 감지한 후 적절한 처리를 규정하고 있다. 본 연구에서는 예외처리에 대해서는 하드웨어적으로 예외 상황을 감지한 후, 이에 대한 처리는 예외 처리 루틴을 활용하는 방식을 사양으로 채택하였다.

Ⅲ. SD 숫자 체계를 사용한 부동 소수점 승산기 구조

본 연구에서 설계한 부동 소수점 승산기는 크게 분수부, 지수부, 부호부, 제어부, 입력 데이타 변환회로, 출력 데이타 변환회로로 구성된다. 설계된 회로는 단정도와 함께 배정도 데이타를 효율적으로 처리할수 있도록, 분수부(Frac unit)에 내장된 병렬 승산기 구조는 54-비트×54-비트 곱셈을 지원하는 구조를 갖고 있다. 따라서 단정도 부동 소수점 데이타에 대한 곱셈 동작이 동일 하드웨어상에서 구현되기 위해서는, 배정도 형태로 변형이 필요하다. 본 아키텍처에서는 입력 데이타 변환 회로(IN-FORMATTER)를 사용하여 내부 하드웨어에 적합한 형태로 입력 데이타가 변환된다. 마찬가지로 곱셈 연산후에는 곱셈 결과가 원하는 형태, 즉 배정도 혹은 단정도 형태로 변형되어야 하므로, 출력 변환 회로(OUT-FORMATTER)가 사용된다. 입력 데이타 변환 회로는 레지스터 화일에서 읽은 32 비트 또는 64 비트 데이타에 대해, 54-비트 분수부 비트, 11-비트 지수 비트 및 1 비트의 지수 비트를 생성한다. 즉 배정도 데이타의 경우 숨겨진(hidden) 비트가 포함됨과 동시에 수정된 Booth 알고리즘을 이용하여 곱셈 동작을 하기 위해, 0의 부호 비트(sign bit)가 추가된다. 단성도 데이타의 경우 분수부에 29개의 0이 삽입된다.

3.1 분수부 하드웨어 구조

FMUL의 분수부 데이타 패스는 그림 2에 보이는 바와 같이, 3단계 파이프라인 단계(ALU-1, ALU-2, ALU-3)를 거쳐 부동 소수점 곱셈을 수행하게 된다. ALU-1 단계는 54-비트×54-비트 분수부 곱셈을 수행하여, SD 수 표현 형태의 2개의 106-비트 결과(SS, SA)를 생성한다. 그림에서 zero & msb_one_det 블록은 0을 포함한 특수한 데이타가 입력에 존재하는지 결정하는데 필요한 정보를 감지하여 제어회로로 제공한다. 그리고 TOD(trailing one detector) 블록은 2개의 분수부 데이타의 끝의 0의 갯수를 계산하는 회로이다. ALU-2 단계에서는 SD 숫자 표현 형태의 결과에 대해, 뺄셈 동작을 수행하여 반올림이 되지 않은 결과를 만드는 동작과 함께 TOD 결과를 사용하여, S(sticky) 비트를 생성한다. ALU-3 단계에서는 반올림 동작, 정규화동작, 재정규화 동작과 함께, 예외처리 조건을 감지한다. 그리고 최종적으로 얻어진 결과를 출력 변환회로를 통해, 단정도 또는 배정도 형태의 결과를 얻게 된다.

3.2 SD 숫자 체계를 사용한 radix-16 곱셈 알고리즘

본 승산기에 사용한 곱셈 알고리즘은 기존의 2의 보수 숫자 체계를 이용한 수정형 Booth 알고리즘이 아닌, SD(signed digit) 숫자 체계를 사용한 수정형 곱셈 알고리즘이다. 이러한 SD 숫자 체계를 사용한 주된 이유는 SD 수를 더할때 캐리 전파 동작을 제거할 수 있다는 장점이 존재하기 때문이다[4]. 이러한 2진 SD 숫자 체계에서 각 디지트(digit)는 {1, 0, −1}을 가질수 있다. 이러한 1 디지트의 SD 수는 2개의 양수에 대한 뺄셈 결과로 해석될 수 있으므로, 2의 보수 형태에 대한 2개의 데이타도 약간의 조작으로 1개의 SD 숫자로 변형될 수 있다. 이러한 1 디지트의 SD 수를 하드웨어로 구현하기 위해서는 2-비트의 선이 필요하다. 이러한 SD 디지트를 나타내기 위한 2비트의

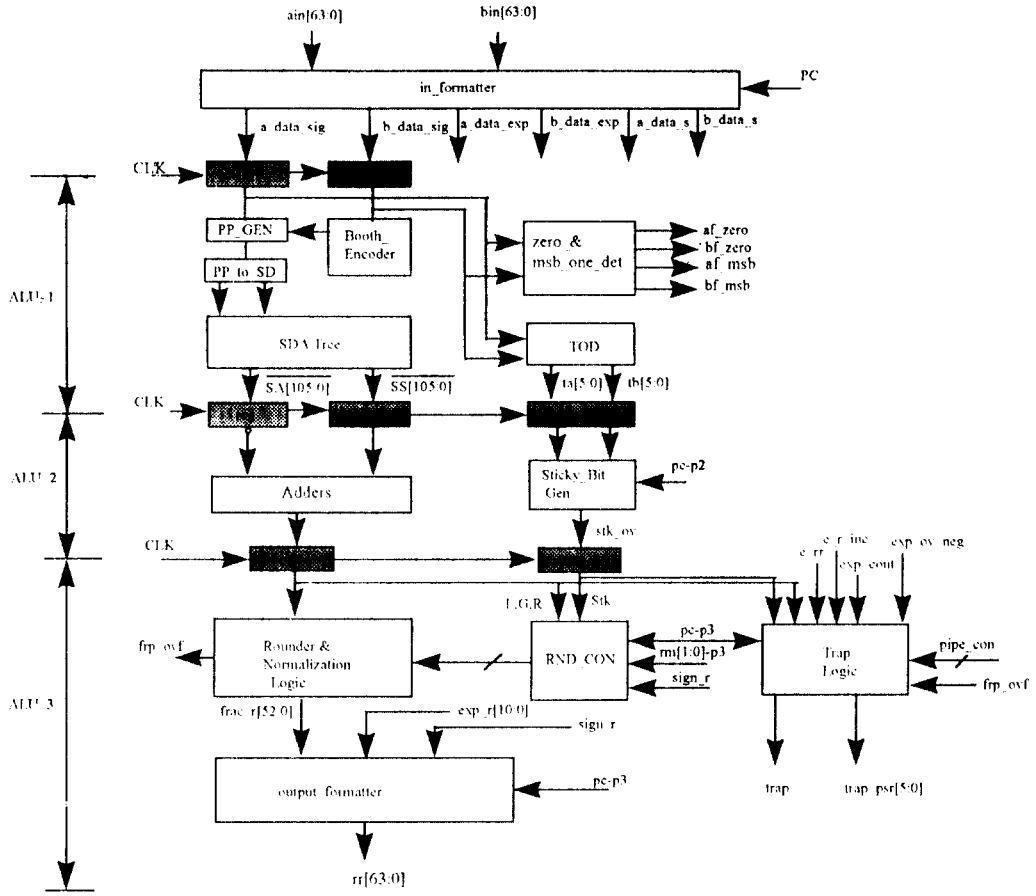


그림 2. 분수부 하드웨어의 블록도
Fig. 2. Block diagram of fraction unit

값 {Xs, Xa}와 SD 디지트사이의 관계는 표 1과 같다. 기존의 SD 숫자 체계를 사용한 승산기의 경우, 수정된 Booth 알고리즘에서 생성된 2의 보수 형태의 부분곱을 각각 SD 형태의 부분곱으로 변형한후, SD 가산기를 통해 덧셈을 수행하기 때문에 기존의 2의 보수 체계를 이용한 승산기 보다 실제로는 더 많은 하드웨어가 필요하다는 결점이 존재한다[5,6]. 기존 수정형 Booth알고리즘을 검토한 후, SD 숫자 표현 형태의 부분곱을 생성하는 기수-16의 승산 알고리즘을 제안한다. 수정형 Booth 알고리즘에서 승수(multiplier)의 레코딩 형태는 다음과 같다. 여기서 x_{-1}은 0이다.

$$X=\sum_{j=0}^{(n/2-1)}(x_{2j}+x_{2j-1}-2x_{2j+1})\,2^{2j} \qquad (2)$$

식 (2)에 따라 수정형 Booth 알고리즘에서는 승수값을 최하위 비트부터 3비트씩 중첩해서 레코딩함에 의해서 부분곱의 수를 n/2개로 만들수 있다[7]. 식 (2)에서 n의 개수가 4의 배수인 경우 식은 다음과 같이 변형될 수 있다.

$$X=\sum_{k=0}^{(n/4-1)}\{(x_{4k}+x_{4k-1}-2x_{4k+1})+(x_{4k+2}+x_{4k+1}-2x_{4k+3})2^2\}2^{4k} \qquad (3)$$

2개의 양의 정수 차로 하나의 SD수를 표현할 수 있다는, SD 수 표현 체계의 특성을 이용하기 위해서, 식 (3)을 다음과 같이 변형한다.

표 1. SD 디지트의 비트 할당
Table 1. bit assignment of SD digit

SD digit Xsd	2진 비트 할당 Xs	Xa
0	0	0
1	0	1
-1	1	0

$$X=\sum_{k=0}^{(n/4-1)}\{(x_{4k+2}+x_{4k+1}-2x_{4k+3})2^2 \quad (4)$$

$$-(2x_{4k+1}-x_{4k}-x_{4k-1})\}2^{4k}$$

$$=\sum_{k=0}^{(n/4-1)}\{B_{k,odd}2^2-B_{k,even}\}2^{4k}$$

$$X\cdot Y=\sum_{k=0}^{(n/4-1)}\{B_{k,odd}\cdot Y\cdot 2^2-B_{k,even}\cdot Y\}2^{4k} \quad (5)$$

$$=\sum_{k=0}^{(n/4-1)}\{PP_{k,odd}-PP_{k,even}\}2^{4k}$$

$$=\sum_{k=0}^{(n/4-1)}\{PP\_SD_k\}2^{4k}$$

여기서 $PP\_SD_k=PP_{k,odd}-PP_{k,even}$

식 (5)에 따라 SD 형태의 부분곱의 개수는 n/4로 될 수 있다. 단 n의 개수가 4의 배수가 아닌 2의 배수인 경우, 맨 상위 PP_SD_k의 $PP_{k,odd}$는 부호 확장 특성상 항상 0이므로, $PP_{k,even}$항만을 사용하여 PP_SD_k를 직접 구현할 수 있다. 단 2의 배수도 4의 배수도 아닌 경우는 승수를 먼저 2의 배수 또는 4의 배수로 부호 확장 후, 식 (5)를 적용한다. 식 (5)의 PP_SDk를 구현하기 위해서는 먼저 $PP_{k,odd}$와 $PP_{k,even}$을 생성한후, 이들을 하나의 PP_SDk로 만들어서, Wallace 트리[8] 구조의 SDA (signded digit adder)로 더하게 된다. 그림 3은 제안된 알고리즘의 레코딩 기법을 나타낸다. 즉 5 비트에 대한 레코딩이 이루어지므로 사실상 기수(radix)-16의 booth 곱셈과 유사하다. 그리고 표 2은 $B_{k,even}$과 $B_{k,odd}$에 대한 레코딩 동작을 나타낸다. $B_{k,odd}$는 수정된 Booth 알고리즘의 레코딩 동작과 동일하지만, $B_{k,even}$은 부호 값이 반대로 나타난다. $PP_{k,odd}$와 $PP_{k,even}$을 결합하여, PP_SD_k를 생성하는 동작은 비트단위로 병렬적으로 구현이 가능하다. 단, $PP_{k,odd}$와 $PP_{k,even}$은 2의 보수 형태이므로, 부호 비트는 다른 비트와 달리 음의 가중치를 갖고 있기 때문에 다른 비트와 다른 동작이 필요하다. 그림 4은 1 비트의 $PP_{k,odd}$와 $PP_{k,even}$을 사용하여 1 비트의 부분곱 PP_SD_k를 생성하는 회로를 나타낸다. 본 연구의 부동 소수점 승산기의 경우 54가 4의 배수가 아닌 2의 배수이므로, 마지막 PPk,even는 독립적으로 PP_SDk를 생성하는데, 이것은 추가의 게이트가 필요치 않고 단순한 배선으로 구현된다. 단, PPk,odd와 PPk,even생성시 -0, -Y 또는 -2Y 생성은 1의 보수 동작 후 추가로 1을 더하는 동작이 필요하다. 이러한 동작을 지시하는 신호는 Booth 인코더(Enc_e, Enc_o)의 출력인 neg_e 또는 neg_o인데, 각각의 neg_o신호들은 결합되어 부호 벡터(sv_o)를 생성하고, neg_e신호는 서로 결합되어, 또하나의 부호 벡터(sv_e)를 생성한다. 단, PP_SDk = PPk,odd - PPk,even이므로 sv_o 부호 벡터내의 1은 양의 가중치를 갖는데 비해, sv_e 부호 벡터내의 1은 음의 가중치를 갖고 있기 때문에, 2개의 벡터를 결합하면 하나의 PP_SDk를 생성할 수 있다. 따라서 제안된 SD 체계의 승산기 알고리즘에서 부분곱 PP_SDk의 개수 N은 부호 벡터에 기인하는 항을 포함해 다음과 같다.

$$N=\lfloor\frac{n}{4}\rfloor+1 \quad (6)$$

생성된 57 비트의 14개의 PP_SD:{xs, xa}는 SD체계의 숫자를 더하는 SDA가산기 트리를 통해 더해진다. 본 연구에 사용한 1-디지트 SD 가산기는 참고문헌[6]의 SD 가산기를 수정한 구조로 반전된 입력을 사용하여, 반전된 출력을 생성하는 구조를 갖고 있다. 그림 5는 SD 수 표현 형태의 부분곱을 병렬적으로 더하

표 2. $B_{k,odd}$와 $B_{k,even}$에 대한 레코딩 동작
Table 2. Recording table of $B_{k,odd}$ and $B_{k,even}$

X_{4k+3} (X_{4k+1})	X_{4k+2} (X_{4k})	X_{4k+1} (X_{4k-1})	$B_{k,odd}$	$B_{k,even}$
0	0	0	+0	-0
0	0	1	+1	-1
0	1	0	+1	-1
0	1	1	+2	-2
1	0	0	-2	+2
1	0	1	-1	+1
1	1	0	-1	+1
1	1	1	-0	+0

는 SDA 가산기 트리를 보여준다. 제안된 방식의 곱셈 알고리즘에서 필요한 SDA 트리의 단계수 R은 다음과 같다.

$$R = \lceil \log_2 N \rceil \quad (7)$$

SDA가산기 트리는 N이 14이므로 4 레벨로 구성된다. 기존 수정된 Booth 알고리즘을 사용하는 경우에 비해, 가산기 트리의 레벨 수가 훨씬 작게 된다.

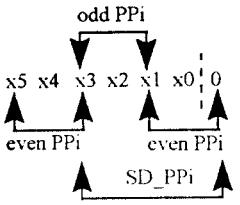


그림 3. 승수에 대한 SD 레코딩 방식
Fig. 3. SD recording scheme for multiplier

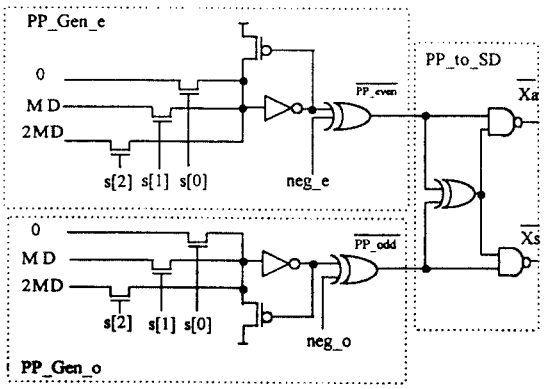


그림 4. 1 비트 부분곱 PP_SD_k:{xs, xa}를 생성하는 회로
Fig. 4. 1-bit partial product, PP_SD_k:{xs, xa} generator

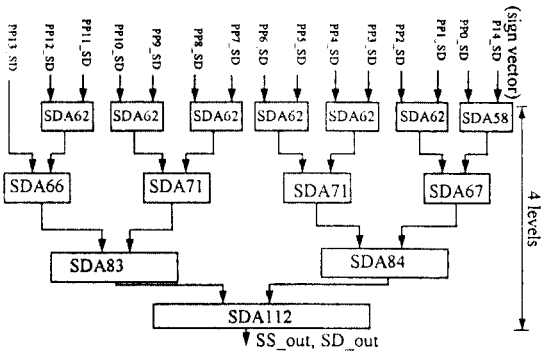


그림 5. 부분곱을 더하는 SDA 트리
Fig. 5. SDA tree to sum partial products

3.3 IEEE 반올림 및 정규화 회로 기법

부동 소수점 곱셈 동작에서는 분수부에 대한 곱셈 동작이 완료된 후, 정규화 동작, 반올림 동작, 재정규화 동작이 순차적으로 이루어져야 하므로 속도개선에 큰 난점이 되고 있다. 이러한 결점을 개선하기 위해서 3가지 동작을 동시에 처리하는 방안이 제안되고 있다[9,10,11]. 기존의 제안된 방식은 제한된 반올림 방식에만 적용될 수 있거나[9], 결과로 선택될 수 있는 값의 형태가 복잡하여 하드웨어로 구현이 어려우며[10], 또한 단일 하드웨어내에 여러 가지 방식의 곱셈이 수행되는 경우를 고려하지 않고 있다는 제한이 있다[11]. 본 논문에서는 반올림 방식에 관계없이 결과 선택의 종류를 2가지 경우로 제한하고, 추가의 재정규화를 위한 별도의 추가 동작이 필요없으며, 단일 하드웨어내에 여러 가지 곱셈이 처리되는 경우에도 올바로 동작하는 반올림 및 정규화 회로기법을 제안하였다. 그림 6은 제안한 반올림 및 정규화 방식을 n-bit × n-bit 부동 소수점 곱셈에 적용한 흐름도이다. 그림 6을 분석하면 n-비트 곱셈의 결과는 먼저 배열 또는 트리 구조에 의해, 2개의 2n-비트의 carry-save 형태로 표현된다. 그리고 나서 반올림 동작과 정규화 동작을 동시에 수행하기 위해 상위 (n+2)비트와 S (Sticky) 비트 생성과 관련된 하위 (n−2)비트를 분리한다. 하위 (n−2) 비트에 대한 캐리 출력 Cin은 고속의 가산기를 통해 생성한다. 상위 (n+2)비트는 carry-save기능을 갖는 반가산기를 통해, Cin을 위한 공간을 마련하고 새로운 합(sum)과 캐리(carry) 벡터를 생

성한다. 참고 문헌[9] 방식의 경우 하위 (n-1)비트의 값을 분석해서 반올림과 정규화가 수행된 최종 결과를 선택하는 기법을 사용하는데, 이러한 방식의 경우 모든 반올림 방식을 지원하기 위해서는 상위 (n+1)비트 결과에 대해 여러 가지 결과 형태가 필요하다. 따라서 이를 구현하려면, 다양한 기능의 다수개의 가산기가 필요하며 제어도 복잡하다. 본 연구에서는 상위 결과에 대한 선택 종류를 2가지로 제한하는 대신에 약간의 추가 처리 단계를 하위부에 삽입하는 방식을 사용한다. 즉 반가산기를 통해 얻어진 상위 (n+2)비트의 캐리와 합 벡터는 다시 상위 (n-1)비트와 하위 3비트로 구분한다. 여기서 하위 3비트에 대해 Cin 결과를 받아들여 캐리 전달 덧셈을 수행하여, 반올림 동작에 필요한 L, G, R값과 함께 반올림된 상위 (n-2)비트에 대한 결과 선택을 위한 캐리 신호(sel_cin)을 생성한다. 생성된 L, G, R은 외부에서 독립적으로 생성된 S(sticky)비트와 결합되어 반올림 신호(Rv, Rnv)를 생성한다. Rv는 분수부 오버플로우(V=1)가 발생하는 경우 반올림이 이루어지는 위치와 반올림값을 지시하고, Rnv는 분수부 오버플로우가 발생하지 않는 경우, 반올림이 이루어지는 위치와 반올림 값을 지시한다. 상위 (n-1)비트는 캐리 입력이 0인 경우와 1인 경우에 대한 2가지 형태의 출력(sum_0, sum_1)을 생성하는데, 이것은 기존 캐리 선택 가산기를 수정하여 구현할 수 있다. 하위 3비트에 대한 내부 형태는 carry-save 기능을 하는 반가산기의 동작 특성(0+0=00, 1+1=10, 0+1=01)을 고려하면, 4가지 형태(type 0-3)로 구분할 수 있다. 4가지 형태에 Cin=1을 더할 경우 결과는 2가지 그룹으로 구분될 수 있다. 그룹 0는 맨 상위 비트가 0이기 때문에 상위 2비트에 반올림 동작(Rv, 또는 Rnv와의 덧셈)이 수행되더라도 새로운 캐리 출력이 생성되지 않는 경우을 나타내고, 그룹 1은 새로운 캐리 출력이 생성될 수 있는 경우를 지시한다. 그런데 그룹 1은 Cin에 의해서는 캐리 출력이 생성되지 않은 경우에 해당된다. 위의 2가지 사실을 종합하면 Cin과 Rv, 또는 Rnv와 carry-save된 하위 3 비트와의 덧셈에서 캐리출력이 하나만 생성될 수 있다. 즉, Cin에 의한 캐리 출력(sel_cin)과 Rv, Rnv에 의한 캐리 출력(sel_rnd)은 서로 배타적인 관계가 있다. 따라서 최종 결과의 상위 (n-2) 비트는 sum_1과 sum_0의 2가지 형태로 제한될 수 있다. 단, 그림에서 Rv가 사용되어야 하는지 Rnv가 사용되어야 하는지 여부는, sel_cin에 의해, 반올림전 분수부 오버플로우 발생 여부를 지시하는 값, V에 의해서 결정된다. 그러나 하드웨어의 실제적인 구현시 타이밍 지연을 고려하여 병렬적인 처리기법을 활용한다. 즉, Rv와 Rnv를 반올림 회로(rnd_logic)에서 L, G, R, S 비트와 반올림 양식(rm[1:0]) 및 부호(sign) 비트를 사용하여 V값에 무관하게 동시에 결정한다. 그리고 나서 Cin이 더해진 3비트 결과(그룹 0 또는 1)에 Rv 또는 Rnv를 더하는 동작을 병렬로 수행한 후, V값을 사용하여 바람직한 sel_rnd를 결정한다. 최종 반올림 결과값에서 분수부 오버플로우(frp_ovf)가 발생하는지 여부에 따라, 최종 n-비트 결과는 (n-1) 비트 캐리 선택 가산기 출력과 하위 1-비트(rx[1])이거나, 캐리 선택 가산기 출력의 하위 (n-2)비트와 하위 2-비트(rx[1:0])로 결정된다. 제안된 방식은 기존 방식과는 달리 4가지 반올림 방식을 지원할 수 있으며, 상위 (n-1)비트에 대한 결과 선택의 종류가 2가지로 제한된다는 장점이 있다. 그림 6은 단일 데이타 형식의 곱셈에 대한 하드웨어 흐름도이므로, 본 연구에서 필요로 하는 단정도와 배정도 데이타에 대한 곱셈을 동시에 지원해주기 위해서, 단정도와 배정도 데이타의 특성을 분석하면 그림 7과 같다. 그림 8와 그림 9는 제안된 반올림 및 정규화 기법을 단정도 데이타와 배정도 데이타에 관계없이 동일 하드웨어내에 구현하기 위한 단계 2와 단계 3의 회로도를 나타낸다. 그림에서 PC=0는 단정도 연산, PC=1은 배정도 곱셈 동작을 지시한다. 최종 결과값은 sum_1 또는 sum_0와 함께 반올림 효과가 반영된 하위 1 비트(manv) 또는 2 비트(mann[1:0])로 결정된다. 일반적으로 분수부 오버플로우(frp_ovf)가 발생하는 경우, 하위 1 비트는 Rv가 더해진 결과(manv)를 선택하고, 발생치 않은 경우 하위 2 비트는 Rnv가 더해진 2 비트의 결과(mann[1:0])로 선택해야 하지만 예외적인 경우가 있다. 반올림전에는 분수부 오버플로우가 발생치 않았지만(vi=0), 반올림 동작에 의해 분수부 오버플로우(frp_ovf)가 발생한 경우는, 최종 결과치의 하위 1비트는 manv가 아닌 Rnv가 더해진 결과의 맨상위값, mann[1]로 선택되어야 한다. 이러한 예외적인 경우를 처리하기 위해 mann[1]과 manv를 입력으로 하는 하나의 mux를 사용하였다. 따라서 정규화, 반올림, 재정규화가 모두 반영된 최종

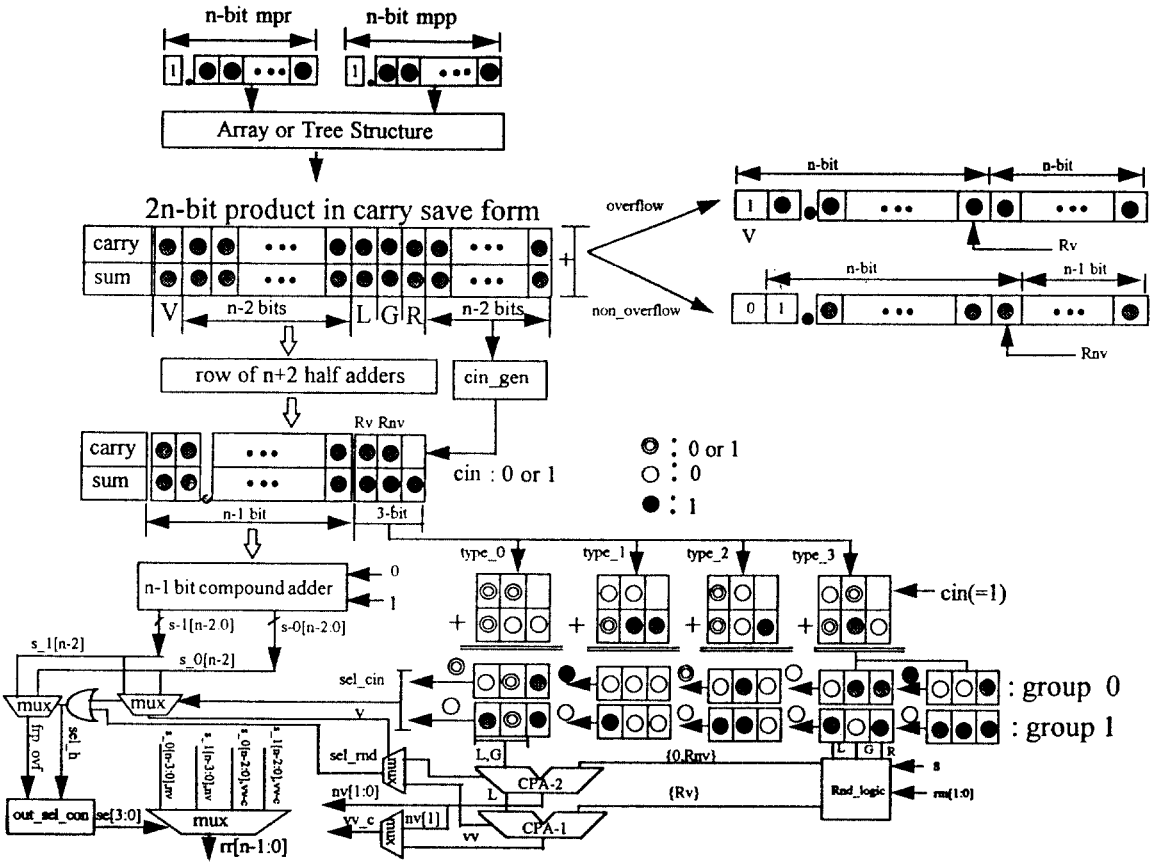


그림 6. n-비트 부동 소수점 곱셈에 대한 정규화 및 반올림 기법
Fig. 6. Normalization and rounding scheme for n-bit floating-point multiplication

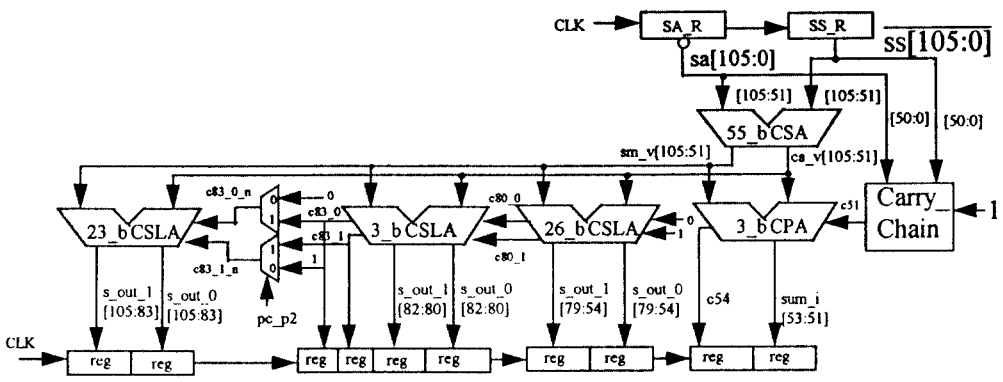


그림 7. 단정도와 배정도 데이타를 함께 지원하는 복합 가산기(분수부의 ALU-2단계 하드웨어)
Fig. 7. Compound adder for single and double-precision multiplication(ALU-2 stage hardware of fraction unit)

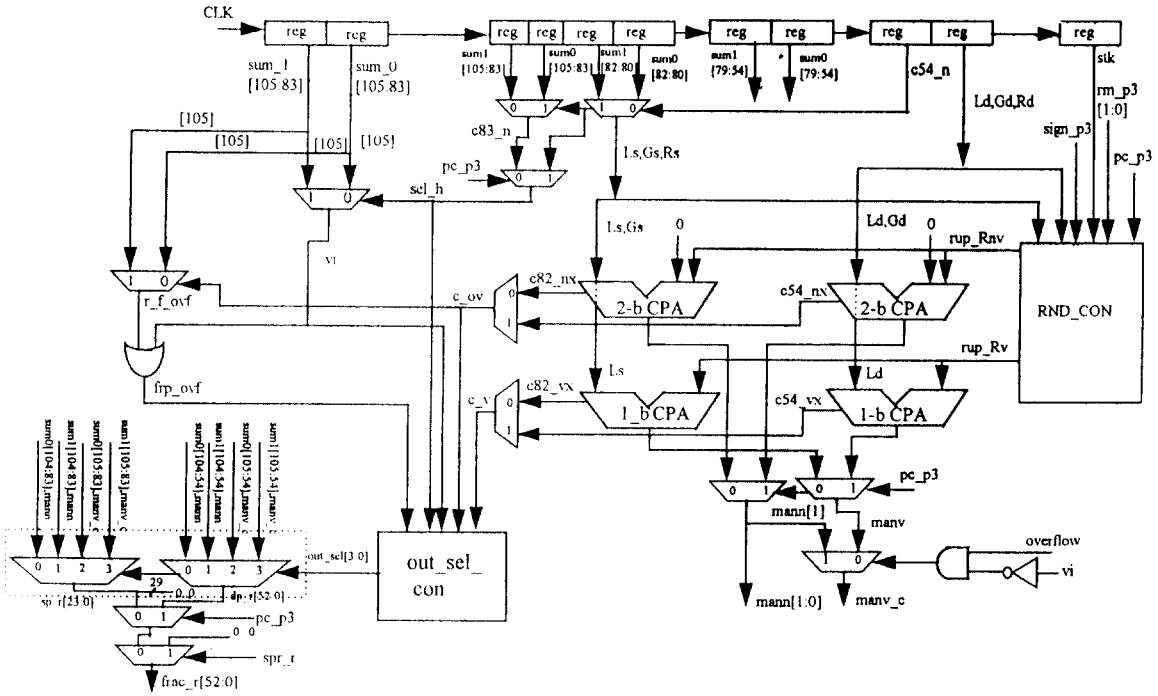


그림 8. 단정도 데이타와 배정도 데이타에 대한 정규화, 반올림 및 재정규화를 지원하는 회로(분수부의 ALU-3 단계 회로)

Fig. 8. Normalizarion, rounding, and post-normalization for single and double precision multiplication(ALU-3 stage hardware of fraction unit)

표 3. 결과 선택 동작

Table 3. Result selection operation

frp_ovf	sel_h	최종 결과(배정도)	최종 결과(단정도)
0	0	sum_0[104:54], mann	sum_0[104:83], mann
	1	sum_1[104:54], mann	sum_1[104:83], mann
1	0	sum_0[105:54], manv_c	sum_0[105:83], manv_c
	1	sum_1[105:54], manv_c	sum_1[105:83], manv_c

결과 선택 동작은 표 3에 나타나 있다.

3.4 sticky 비트 생성회로

본 연구에서 제안된 반올림 및 정규화 알고리즘이 효율적으로 동작하기 위해서는 스티키 비트의 빠른 계산이 필수적이다. 이러한 조건을 만족시키기 위해 이진수 곱셈 동작시 최종 결과의 하위비트 0의 갯수는 승수와 피승수 데이타에 존재하는 하위 비트 0의 갯수의 합과 같다[10]는 이론에 바탕을 두고, S(sticky) 비트를 생성하였다. 먼저 ALU-1단계에서 분수부에 존재하는 하위 0의 갯수를 결정하고, ALU-2단계에서 단정도와 배정도 곱셈에서 스티키 비트가 1이 되기 위해 1값이 놓일수 있는 최대 범위(51, 80)와 ALU-1 단계에서 얻어진 TOD 출력의 합과 비교동작을 통해 S 비트 값을 결정하는 방식을 사용하였다. 단, 분수부에 존재하는 하위 0의 갯수를 결정하는 TOD회로는 참고문헌[12]의 선행 0의 갯수 카운터(leading-zero counter)회로를 변형하여 사용하였다. 기존 부동 소수점 곱셈 회로의 경우 최종 연산 결과에서 스티키 비트를 생성시키는 방식을 채택함에 따라, S 비트 생성 회로가 최악 전달 지연회로(critical path)에 존재하는

데 비해, 본 아키텍처에서는 이러한 문제를 해결할 수 있다.

3.5 지수부, 부호부와 제어부

그림 9은 지수부에 대한 블럭도를 나타낸다. ALU-1 단계의 경우 지수 덧셈기와 bias 감산기로 구성된다. 2개의 캐리 전달 가산기는 carry-save 개념을 사용하는 하나의 3-입력 가산기와 하나의 캐리전달 가산기를 사용하여 등가적으로 구현하였다. 단, 11-비트가 아닌 13 비트로 결과를 유지한 이유는 오버플로우와 언더플로우 예외 조건을 ALU-3단계에서 용이하게 처리하기 위해서이다. ALU-2 단계에서는 분수부 오버플로우에 대비하기 위해 1만큼 증가된 지수값을 결정한다. 단, 단정도 데이타에 대한 오버플로우예외 조건을 배정도 데이타에 대한 오버플로우 예외조건과 동일 하드웨어로 처리하기 위해, 9번째 비트를 exp_ov_neg신호에 저장하도록 하였다. exp_cout = 1은 지수가 양수값임을 지시하며, exp_cout = 1 & exp_ov_neg = 1은 반올림에 무관하게 이미 오버플로우 예외조건이 발생했음을 지시한다. exp_cout = 0는 지수값이 음수로 반올림에 무관하게 이미 언더플로우 예외조건이 발생했음을 지시한다. 이러한 신호와 지수부값 e-rr, 1만큼 증가된 지수값 e_r_inc, 그리고 분수부 오버플로우값(frp_ovf)을 사용하여, 오버플로우와 언더플로우 예외 조건을 쉽게 결정할 수 있다. 그리고 추가적으로 0×0, 유한한 크기값×0 또는 유한한 크기값×무한대(∞) 등의 곱셈의 경우 회로에서 계산된 결과가 감추어진(hidden) 비트와 바이스화된 지수값에 기인하여 올바른 결과가 생성되지 않는다. 이러한 조건은 제어 회로에서 ALU-1 단계에서 감지한 특수 데이타

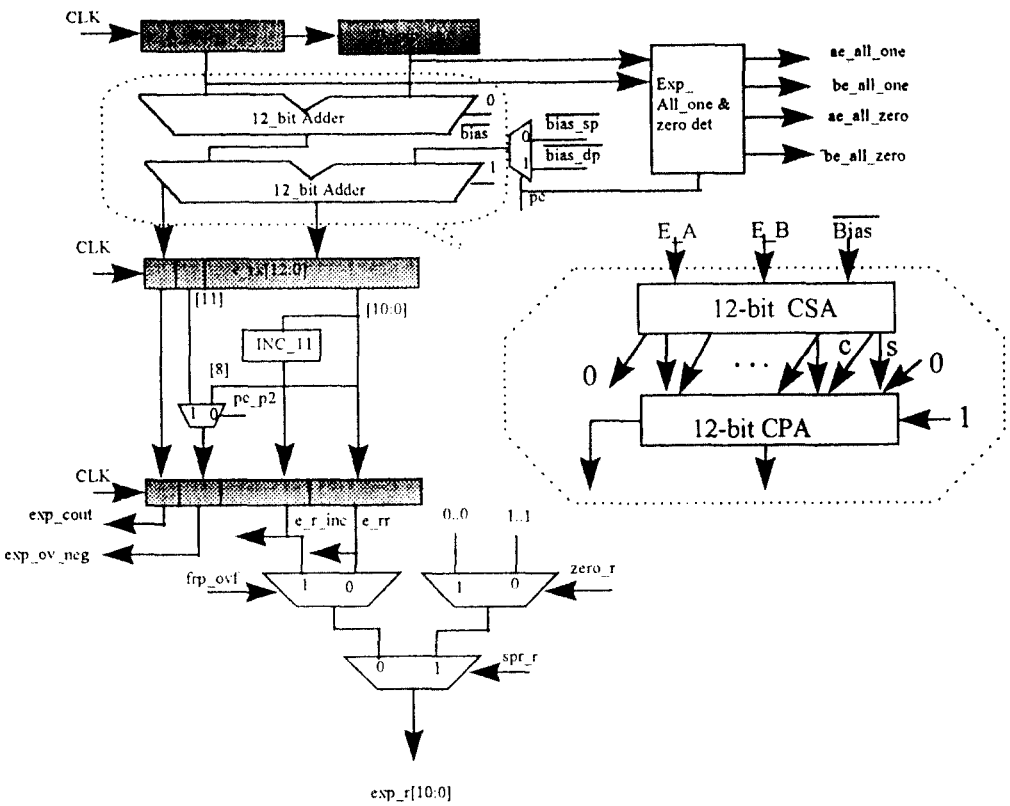


그림 9. 지수부
Fig. 9. Exponent unit

형 조건을 바탕으로 적절한 제어신호를 만들어, ALU-3 단계에서 무한대와 0값을 하드웨어적으로 제공한다. 부호부는 결과의 부호값을 결정하며, XOR 게이트와 3-단계의 레지스터 체인으로 구성된다. 제어부는 외부에서 곱셈 연산 종류(PC), 반올림 양식(RM[1:0]), 부동 소수점 곱셈 연산 수행 여부(no_fop) 그리고 내부에서 발생한 특별한 데이타 감지조건을 입력으로 받아서, 데이타 패스에 필요한 제어 신호를 적절한 타이밍에 맞추어 발생한다.

Ⅳ. 성능 분석과 설계 검증

설계된 FMUL는 Verilog HDL 시뮬레이터[13]를 사용하여 레지스터 전달 단계, 게이트 단계, 스위치 단계로 검증을 한 후, IEEE 754 부동 소수점 표준안을 만족하는지 검증하기 위해, 미 Berkeley 대학에서 개발한 테스트 벡터(test vector)[14]를 사용하였다. 그리고 나서 설계된 회로는 전기적인 동작 특성을 구하기 위해, 미국 VLSI사의 0.6 μm CMOS 표준셀을 사용하여 등가적으로 구현한후 성능을 분석해 보았다. TV (Timing Verifier)[15] 소프트 웨어를 통해 설계된 회로의 최악 지연 경로를 검증을 한 결과 ALU-1단계의 경우 14[ns], ALU-2 단계의 경우 10[ns], 그리고 ALU-3 단계는 6[ns]로 나타났다. 따라서 본 연구에서 설계한 승산기 회로는 대략 125,000의 트랜지스터로 구성되며, 67Mhz의 동작 주파수로 최대 67 MFLOPS의 성능을 가짐을 알수 있다. 위의 지연 특성을 고려할 때, 본 연구에서 제안한 승산기 회로의 경우 ALU-2와 ALU-3단계를 하나의 단계로 합쳐서 2단 파이프라인 구조로 변형하는 것도 가능하다. 그러나 본 연구에서 그러한 기법을 사용하지 않은 이유는, 본 논문의 승산기를 비정규화 데이타를 처리하고, 예외처리를 소프트웨어가 아닌 하드웨어로 구현하는 구조로 변형하는 방안을 검토하고 있기 때문이다. 이러한 추가되는 기능은 대부분 ALU-3단계에 왼쪽 이동(left-shift) 기능을 하는 배럴 시프터와 약간의 추가 하드웨어가 필요하다. 표 4는 본 연구의 승산기에 사용한 SD 숫자체계의 승산 알고리즘과 기존 수정형 Booth 알고리즘과의 비교 결과를 나타낸다. 단, 비교를 단순화하기 위해 승산기에서 부분곱을 더하는 가산기 트리 방식으로 Wallace 트리를 사용한다고 가정했다. 그리고 기존 승산기의 경우 각 부분곱의 2의 보수 구현시 발생하는 추가의 1을 묶어서 하나의 부호 벡터로 만들어, 이것을 부분곱으로 사용하는 것으로 가정했다. 그리고 기본 셀의 지연 특성을 정량적으로 비교하기 위해, 2-입력 NAND, NOR 게이트를 1의 기준 지연을 갖는 것으로 기준 설정하고, XOR, XNOR 게이트는 1.5, NOT 게이트는 0.5의 지연시간을 갖는 것으로

표 4. 기존 승산기 방식과 제안한 승산기 알고리즘 비교

Table 4. Comparisons between conventional multiplication scheme and proposed multiplication scheme

종류 / 특성	(3, 2)counter를 사용하는 수정형 Booth 알고리즘	(4, 2) counter를 사용하는 수정형 Booth 알고리즘	제안한 SD 곱셈 방식
부분곱의 개수 R	$\lceil n/2 \rceil + 1$	$\lceil n/2 \rceil + 1$	$\lceil n/4 \rceil + 1$
Wallace 트리의 단계수	$\lceil \log_{(3/2)}\{\lceil n/2 \rceil + 1\} \rceil$	$\lceil \log_2\{\lceil n/2 \rceil + 1\} \rceil$	$\lceil \log_2\{\lceil n/4 \rceil + 1\} \rceil$
기본셀의 트랜지스터 수	34[16]	68[16]	48
기본셀의 최악 지연시간	3	6	5.5

n : 승수의 비트 수, $\lceil x \rceil$: x 보다 같거나 큰 최소정수, 게이트 지연 시간 기준 : 2-입력 NAND, NOR 게이트

표 5. 부동 소수점 승산기의 성능 비교

Table 5. Performance comparisons between floating-point multipier

연 산	SPARC FPU[2]		Pentium[17]		Proposed architecture	
	latency	throughput	latency	throughput	latency	throughput
FMULs	5	1/5	3	1/2	3	1
FMULd	7	1/7	3	1/2	3	1

latency : numbers of clock cycle throughput : numbers of clock cycle

정량화 했다. 그리고 팬-인(fan-in)이 1만큼 증가할 때마다, 게이트당 0.5씩 지연값을 더하는 방식으로 게이트 지연시간을 정량화해서 기본셀의 지연 특성을 비교했다. 가산기 트리의 지연시간은 트리의 단계수와 기본셀의 지연시간의 곱으로 정량화 시킬 수 있는데, 제안된 방식이 PP_SDk를 만드는 과정을 하나의 단계로 판단하더라도 기존 방식에 비해 면적과 속도 측면에서 우수하다. 표 5는 상용화된 부동 소수점 승산기와 설계한 승산기를 아키텍쳐 측면에서 비교하였다. 표 5를 통해 본 승산기는 수퍼 스칼라 마이크로프로세서에 내장하기에 적합한 동작 주파수와 파이프라인 구조를 갖고 있음을 알수 있다.

V. 결 론

본 논문에서는 SD 수 표현 체계를 사용하며, 새로운 반올림 및 정규화 기법을 사용한 부동 소수점 승산기 회로를 제안하고 이를 하드웨어로 구현한 후 성능을 분석하였다. 제안한 승산기에서 사용한 알고리즘과 구조는 Verilog HDL로 표현하고 모의 실험을 통해 올바른 동작이 이루어짐을 확인하였다. 설계된 승산기 회로를 상용화된 0.6 μm CMOS 표준 셀로 구현한 결과, 약 125,000 트랜지스터로 구성되고 최악 조건에서 대략 67Mhz의 동작 주파수를 갖고 있음이 확인되었다. 설계된 회로는 trailing-zero 카운터을 사용한 효율적인 스티키-비트 발생회로와 반올림 결과 선택을 단순화시키는 새로운 반올림 및 정규화 회로를 사용하여, IEEE-754의 4가지 반올림 모드를 시간 지연 없이 효율적으로 구현할 수 있다. 그리고 기존의 수정된 Booth 알고리즘을 변형하여, 부분곱의 갯수를 비트수의 n/4로 감소시키는 SD 숫자체계의 부분곱을 구현함에 의해서, 면적감소와 함께 속도 개선을 얻을 수 있었다. 그리고 부동 소수점 승산기의 지수부에는 결과값을 13 비트로 유지하여, 오버플로우와 언더플로우 예외 처리 조건 감지 동작을 용이하도록 하였다. 그리고 SD 숫자 체계에 바탕을 둔 가산기 트리는 규칙적인 구조를 갖고 있기 때문에, 레이 아웃 측면에도 큰 이점이 존재한다. 본 승산기 회로는 67 Mhz의 클럭 주파수로 최대 67 MFLOPS의 단정도 및 배정도 곱셈 성능을 보여주며, 3단 파이프라인 구조를 갖고 있기 때문에, 고성능의 수퍼스칼라 마이크로프로세서에 쉽게 내장될 수 있다. 본 논문에서 설계한 회로는 향후 비정규화 데이타 처리와 함께 예외처리를 소프트웨어가 아닌 하드웨어로 처리하는 방안에 대한 연구가 필요하다.

참 고 문 헌

1. Nobuhiro Ide, etal, "A 320-MFLOPS CMOS Floating-Point Processing Unit for Superscalar Processors," IEEE Journal of Solid State Circuits, Vol, 28, No.3, pp.352-361, March 1993.
2. Cypress Semiconductor Inc, SPARC RISC User's Guide-CY7C602, 1990.
3. ANSI/IEEE Standard 754-1985 for binary Floating-Point Arithmetic, IEEE Computer Society Press, Los Alamitos, Calf., 1985.
4. A.Avizienis, "Signed digit number representation for the fast parallel arithmetic," IRE Trans. Electronic Computers, Vol. EC-10, pp.389-400, Sept. 1961.
5. N.Takagi et al, "High speed VLSI Multiplication Algorithm with redundant Binary Addition Tree," IEEE Trans. Computer, Vol, C-34, No.9, pp.789-796, Sept. 1985.
6. S.Kuminobu, and T.Taniguchi, "Design of High Speed MOS Multiplier and Divider Using Redundant Binary Representation," Proc. 8th Computer Arithmetic, pp.80-86, 1987.
7. L.P.Rubinfield, "A proof of the Modified Booth's Algorithm for multiplication," IEEE Trans on computers, Vol. C-24, pp.1014-1015, Oct, 1975.
8. C.S.Wallace, "A suggestion for fast Multipliers," IEEE Trans. Electronic Computer, Vol. EC-13, pp. 14-17, Feb, 1964.
9. N.T.Quach, N.Takaki, and M, J.Flynn, "On fast IEEE Rounding," Tech. Rep. CSL-TR-91-459, Stanford University, March, 1991.
10. Mark R. Santoro, Gary Bewick, and Mark A. Horowitz, "Rounding Algorithms for IEEE Multipliers," Proc. 9th Symposium on Computer Arithmetics, pp.176-183, 1989.

11. Robert K.Yu and Gregory B. Zyner, "167 Mhz radix-4 floating-point multiplier," Proc. 12th Symposium on Computer arithmetic, pp.149-154, 1995.
12. Vojin G. Oklobdzija, "An algorithm and Novel Design of a Leading Zero detector circuits: Comparison with Logic synthesis," IEEE Trans. on VLSI System, vol.2, no.1, pp.124-128, March, 1994.
13. Donald E. Thomas and Philip Moorby, The Verilog Hardware Description Language, Kluwer Publisher, 1991.
14. Univ. of California, Berkeley, A compact test suite for P754 arithmetic-ver.2.0, Computer Sci. Div., 1983.
15. VLSI Technology Inc, TV(Timinig Verifier) software tool manual, 1995.
16. Macro Annaratone, Digital CMOS Circuit Design, Kluwer Acadenic Publishiers, pp.179, 1986.
17. Brian Case, "Intel Reveals Pentium Implementation Details," Microprocessor Reports, pp.9-17, March, 2, 1993.

崔 炳 尤(Byeong Yoon Choi)정회원

1985년 2월 : 연세대학교 전자공학과 졸업(공학사)
1987년 2월 : 연세대학교 전자공학과 본대학원 졸업(공학 석사)
1992년 8월 : 연세대학교 전자공학과 본대학원 졸업(공학 박사)
1993년 3월~현재 : 동의대학교 컴퓨터공학과(조교수)
※주관심분야 : 수퍼스칼라 마이크로프로세서 설계, 신호처리 및 통신용 집적회로 설계

李 文 基(Moon Key Lee)정회원

1982년~현재 : 연세대학교 전자공학과 교수 및 연세대학교 부설 아식 설계 공동 연구소 소장
※주관심분야 : 마이크로프로세서 설계, 디지탈 영상처리칩 설계, 설계 자동화 등