(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2011년01월18일
(11) 등록번호 10-1009095
(24) 등록일자 2011년01월11일

(51) Int. Cl.
G06T 1/00 (2006.01) *G06F 7/50* (2006.01)
G06F 3/14 (2006.01)
(21) 출원번호 10-2008-0124099
(22) 출원일자 2008년12월08일
심사청구일자 2008년12월08일
(65) 공개번호 10-2009-0060207
(43) 공개일자 2009년06월11일
(30) 우선권주장
11/952,858 2007년12월07일 미국(US)
(56) 선행기술조사문헌
US5778247 A
JP2003223316 A
JP06059862 A
KR1020010050800 A
전체 청구항 수 : 총 23 항

(73) 특허권자
엔비디아 코포레이션
미국 캘리포니아 95050 산타 클라라 산 토마스 익스프레스웨이 2701
(72) 발명자
오버맨, 스튜어트 에프.
미국 95050 캘리포니아주 산타 클라라 산 토마스 익스프레스웨이 2701
시우, 밍 와이.
미국 95050 캘리포니아주 산타 클라라 산 토마스 익스프레스웨이 2701
탄넨바움, 데이비드 씨.
미국 95050 캘리포니아주 산타 클라라 산 토마스 익스프레스웨이 2701
(74) 대리인
양영준, 백만기

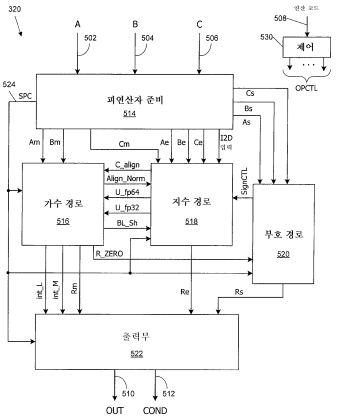
심사관 : 권성호

(54) 다목적 배정도 기능 유닛을 구비한 그래픽 프로세서

(57) 요 약

기능 유닛이 그래픽 프로세서에 추가되어, 렌더링을 위해 이용되는 단정도(single-precision) 기능 유닛 이외에도, 배정도 산술(double-precision arithmetic)에 대한 직접적인 지원을 제공한다. 배정도 기능 유닛은 적어도 배정도 폭인 데이터 경로 및/또는 로직 회로를 이용하여 배정도 입력에 대해, 결합형 곱셈 덧셈을 포함하는 다수의 상이한 연산들을 실행할 수 있다. 배정도 및 단정도 기능 유닛들은 공유된 명령 발행 회로에 의해 제어될 수 있으며, 코어에 포함된 배정도 기능 유닛의 복사본의 수는 단정도 기능 유닛의 복사본의 수보다 작을 수 있어, 칩 영역 상에서 배정도에 대한 지원을 추가하는 것의 영향을 감소시킨다.

대 표 도 - 도5



특허청구의 범위

청구항 1

그래픽 프로세서에 있어서,

이미지 데이터를 생성하도록 구성된 렌더링 파이프라인(rendering pipeline) - 상기 렌더링 파이프라인은 복수의 동시 발생적인 스레드들을 실행하도록 구성된 처리 코어를 포함하고, 단정도 피연산자들(single-precision operands)을 연산함 - 을 포함하고,

상기 처리 코어는 배정도(double-precision) 입력 피연산자들의 세트에 대해 복수의 배정도 연산들 중 하나를 선택가능하게 실행하도록 구성된 다목적 배정도 기능 유닛을 더 포함하고, 상기 다목적 배정도 기능 유닛은 적어도 하나의 산술 로직 회로를 포함하며,

상기 배정도 기능 유닛의 모든 산술 로직 회로들은 배정도로 연산할 수 있는 그래픽 프로세서.

청구항 2

제1항에 있어서,

상기 배정도 기능 유닛은 상기 복수의 배정도 연산들 각각이 동일한 수의 클럭 사이클들 안에 완료되도록 더 구성되는 그래픽 프로세서.

청구항 3

제2항에 있어서,

상기 배정도 기능 유닛은 오버플로우 또는 언더플로우 상태가 발생하는지의 여부에 관계없이 상기 복수의 배정도 연산들 각각이 동일한 수의 클럭 사이클들 안에 완료되도록 더 구성되는 그래픽 프로세서.

청구항 4

제3항에 있어서,

상기 배정도 기능 유닛은 오버플로우 또는 언더플로우 상태가 발생되는 경우에 부동 소수점 산술 표준(floating-point arithmetic standard)에 따르는 오버플로우 또는 언더플로우 결과를 생성하고, 상기 오버플로우 또는 언더플로우 상태가 발생되었는지 여부를 표시하도록 출력 상태 플래그를 설정하도록 더 구성되는 그래픽 프로세서.

청구항 5

제1항에 있어서,

상기 배정도 기능 유닛은 상기 복수의 배정도 연산들 중 임의의 하나를 완료하는데 필요한 시간이 부동 소수점 예외에 의해 영향을 받지 않도록 더 구성되는 그래픽 프로세서.

청구항 6

제1항에 있어서,

상기 복수의 배정도 연산들은,

두 개의 배정도 피연산자들을 더하는 덧셈 연산;

두 개의 배정도 피연산자들을 곱하는 곱셈 연산; 및

제1 배정도 피연산자 및 제2 배정도 피연산자의 곱을 계산한 후, 제3 배정도 피연산자를 상기 곱에 더하는 결합형 곱셈 덧셈 연산(fused multiply-add operation)을 포함하는 그래픽 프로세서.

청구항 7

제6항에 있어서,

상기 복수의 배정도 연산들은, 제1 피연산자와 제2 피연산자에 대한 비교 검사를 수행하고, 상기 비교 검사가 만족되었는지의 여부를 나타내는 불린 결과(Boolean result)를 생성하는 DSET(double-precision comparison) 연산을 더 포함하는 그래픽 프로세서.

청구항 8

제6항에 있어서,

상기 복수의 배정도 연산들은,

두 개의 배정도 입력 피연산자들 중 보다 큰 배정도 입력 피연산자를 반환하는 DMAX(double-precision maximum) 연산; 및

두 개의 배정도 입력 피연산자들 중 보다 작은 배정도 입력 피연산자를 반환하는 DMIN(double-precision minimum) 연산을 더 포함하는 그래픽 프로세서.

청구항 9

제6항에 있어서,

상기 복수의 배정도 연산들은 피연산자를 배정도 포맷에서 비-배정도 포맷(non-double-precision format)으로 변환하는 적어도 하나의 포맷 변환 연산을 더 포함하는 그래픽 프로세서.

청구항 10

제6항에 있어서,

상기 복수의 배정도 연산들은 피연산자를 비-배정도 포맷에서 배정도 포맷으로 변환하는 적어도 하나의 포맷 변환 연산을 더 포함하는 그래픽 프로세서.

청구항 11

그래픽 프로세서에 있어서,

이미지 데이터를 생성하도록 구성된 렌더링 파이프라인 - 상기 렌더링 파이프라인은 복수의 동시 발생적인 스레드들을 실행하도록 구성된 처리 코어를 포함함 - 을 포함하고,

상기 처리 코어는,

하나 이상의 단정도 피연산자들에 대해 산술 연산을 실행하도록 구성된 단정도 기능 유닛을 포함하고,

배정도 입력 피연산자들의 세트에 대해 결합형 곱셈 덧셈 연산을 실행하여 배정도 결과를 제공하도록 구성된 DFMA(double-precision fused multiply-add) 기능 유닛을 더 포함하며,

상기 DFMA 기능 유닛은 데이터 경로들을 갖는 DFMA 파이프라인을 포함하며, 상기 데이터 경로들은 상기 결합형 곱셈 덧셈 연산을 상기 DFMA 파이프라인을 통해서 한 번의 통과로 수행할 수 있는, 그래픽 프로세서.

청구항 12

제11항에 있어서,

상기 DFMA 기능 유닛은,

단일의 반복으로 두 개의 배정도 가수(double-precision mantissas)의 곱을 계산하도록 구성된 곱셈기; 및

단일의 반복으로 두 개의 배정도 가수의 합을 계산하도록 구성된 덧셈기를 포함하는 그래픽 프로세서.

청구항 13

제11항에 있어서,

상기 DFMA 기능 유닛은 한 쌍의 배정도 입력 피연산자들에 대해 곱셈 연산을 실행하여 배정도 결과를 제공하도록 더 구성되는 그래픽 프로세서.

청구항 14

제13항에 있어서,

상기 곱셈 연산 및 상기 결합형 곱셈 덧셈 연산은 동일한 수의 클럭 사이클들 안에 각각 완료되는 그래픽 프로세서.

청구항 15

제11항에 있어서,

상기 DFMA 기능 유닛은 한 쌍의 배정도 입력 피연산자들에 대해 덧셈 연산을 실행하여 배정도 결과를 제공하도록 더 구성되는 그래픽 프로세서.

청구항 16

제15항에 있어서,

상기 덧셈 연산 및 상기 결합형 곱셈 덧셈 연산은 동일한 수의 클럭 사이클들 안에 각각 완료되는 그래픽 프로세서.

청구항 17

제16항에 있어서,

상기 DFMA 기능 유닛은 한 쌍의 배정도 입력 피연산자들에 대해 곱셈 연산을 실행하여 배정도 결과를 제공하도록 더 구성되고,

상기 결합형 곱셈 덧셈 연산, 상기 덧셈 연산 및 상기 곱셈 연산은 오버플로우 또는 언더플로우 상태가 발생하는지의 여부에 관계없이 동일한 수의 클럭 사이클들 안에 각각 완료되는 그래픽 프로세서.

청구항 18

제17항에 있어서,

상기 DFMA 기능 유닛은 오버플로우 또는 언더플로우 상태가 발생되는 경우에 부동 소수점 산술 표준에 따르는 오버플로우 또는 언더플로우 결과를 생성하고, 상기 오버플로우 또는 언더플로우 상태가 발생되었는지 여부를 표시하도록 출력 상태 플래그를 설정하도록 더 구성되는 그래픽 프로세서.

청구항 19

제11항에 있어서,

상기 처리 코어는 병렬로 연산하도록 구성된 제1 기능 유닛의 다수(P개)의 복사본 및 상기 DFMA 기능 유닛의 다수(N개)의 복사본을 포함하는 그래픽 프로세서.

청구항 20

제19항에 있어서,

상기 P개는 상기 N개보다 큰 그래픽 프로세서.

청구항 21

제20항에 있어서,

상기 N개는 1인 그래픽 프로세서.

청구항 22

제21항에 있어서,

상기 처리 코어는 상기 DFMA 기능 유닛에 대한 P개의 세트의 배정도 입력 피연산자들을 모아서, 상기 P개의 세트의 배정도 피연산자들의 상이한 피연산자들을 상이한 클록 사이클들에 상기 DFMA 기능 유닛으로 전달하도록

구성된 입력 관리자 회로를 더 포함하는 그래픽 프로세서.

청구항 23

제22항에 있어서,

상기 입력 관리자 회로는 상기 제1 기능 유닛에 대한 P개의 세트의 단정도 입력 피연산자들을 모아서, 상기 P개의 세트의 단정도 피연산자들의 상이한 피연산자를 상기 제1 기능 유닛의 P개의 복사본들 각각에 대해 병렬로 전달하도록 더 구성되는 그래픽 프로세서.

명 세 서

발명의 상세한 설명

기 술 분 야

[0001] 본 발명은 일반적으로 그래픽 프로세서에 관한 것으로, 특히 그래픽 프로세서에 대한 배정도 결합형 곱셈 덧셈 기능 유닛에 관한 것이다.

배 경 기 술

[0002] 그래픽 프로세서들은 흔히 2차원 또는 3차원 지오메트리 데이터로부터 이미지들의 렌더링을 가속시키기 위해 컴퓨터 시스템들에서 사용된다. 그러한 프로세서들은 일반적으로 높은 수준의 병렬성(parallelism) 및 높은 처리율을 갖도록 설계되기 때문에 수천개의 프리미티브들이 병렬로 처리되어 복잡하고 사실적이고 살아있는 이미지들을 실시간으로 렌더링할 수 있다. 고성능의 그래픽 프로세서들은 전형적인 CPU들보다 많은 계산 능력을 제공한다.

[0003] 보다 최근에는, 이미지 렌더링에 관련되지 않는 다양한 계산들을 가속시키기 위해 그래픽 프로세서들의 능력을 레버리지(leverage)하는 것에 관심이 있다. "범용" 그래픽 프로세서는 과학, 금융, 비지니스 및 다른 분야에서 계산을 수행하는 데 사용될 수 있다.

[0004] 범용 계산들에 그래픽 프로세서들을 적응시키는데 있어서 한가지 어려움은 그래픽 프로세서들이 보통 비교적 낮은 수치 정밀도(precision)로 설계된다는 점이다. 고품질 이미지들은 32비트("단정도") 또는 심지어 16비트("반정도(half-precision)") 부동 소수점 값들을 이용하여 렌더링될 수 있고, 기능 유닛 및 내부 파이프라인들은 이러한 데이터 폭들을 지원하도록 구성된다. 대조적으로, 많은 범용 연산들은 보다 높은 수치의 정밀도, 예를 들어 64 비트("배정도")를 요구한다.

[0005] 보다 높은 정밀도를 지원하기 위해, 몇몇 그래픽 프로세서들은 기계 명령어들의 시퀀스 및 32비트 또는 16비트 기능 유닛들을 이용하여 배정도 계산들을 실행하는 소프트웨어 기술들을 이용한다. 이러한 접근법은 처리율을 감소시키는데, 예를 들어, 단일 64비트 곱셈 연산을 완료하기 위해 백개 이상의 기계 명령어들이 요구될 수 있다. 그러한 긴 시퀀스들은 그래픽 프로세서의 배정도 처리율을 상당히 감소시킬 수 있다. 하나의 대표적인 경우에, 그래픽 프로세서는 최고급 듀얼 코어 CPU 칩에 의해 가능한 처리율의 약 1/5에서 배정도 계산들을 완료할 것으로 추정된다. (비교로서, 동일한 그래픽 프로세서는 그 듀얼 코어 CPU의 처리율의 약 15-20배에서 단정도 계산들을 완료할 수 있다.) 소프트웨어 기반 솔루션들은 훨씬 느리기 때문에, 기존의 그래픽 프로세서들은 배정도 계산들에 거의 사용되지 않는다.

[0006] 다른 솔루션은 간단히 그래픽 프로세서의 산술 회로들 모두를 배정도 피연산자들을 처리할 만큼 충분히 넓게 만드는 것이다. 이것은 배정도 연산들에 대한 그래픽 프로세서의 처리율을 증가시켜 단일 스피드 처리율에 매치시킬 것이다. 그러나, 그래픽 프로세서들은 전형적으로 병렬 연산을 지원하기 위해 각 산술 회로의 수십개의 복사본들을 가지며, 각각의 그러한 회로의 크기를 증가시킴으로써 실질적으로 칩 면적, 비용 및 전력 소비를 증가시킬 것이다.

[0007] 공동 소유의 계류중인 미국 특허 출원 번호 제11/359,353호(2006년 2월 21자 출원)에 기재되어 있는 또 다른 솔루션은 단정도 산술 회로들을 레버리지하여 배정도 연산들을 수행하는 것이다. 이러한 접근법에서는, 단정도 기능 유닛에 포함된 특수 하드웨어를 이용하여 배정도 연산을 반복적으로 수행한다. 이러한 접근법은 소프트웨어 기반 솔루션들보다 상당히 빠르지만(처리율이 예를 들어 ~100의 인수만큼 보다는, 단정도 처리율에 비해 4의 인수만큼 감소될 수 있다), 칩 설계를 상당히 복잡하게 할 수 있다. 또한, 단정도 및 배정도 연산들 사이에 동

일한 기능 유닛을 공유함으로써 너무 많은 명령어들이 동일한 기능 유닛을 요구하는 경우에 유닛은 파이프라인에서 병목된다.

발명의 내용

해결 하고자하는 과제

[0008] 본 발명의 실시예들은 그래픽 프로세서에서 배정도 산술을 직접적으로 지원한다. 렌더링에 사용되는 단정도 기능 유닛들 이외에 다중 목적의 배정도 기능 유닛이 제공된다. 배정도 기능 유닛은 적어도 배정도 폭인, 데이터 경로들 및/또는 로직 회로들을 이용하는 배정도 입력들에 대해, 결합형 곱셈 덧셈을 포함하는 다수의 상이한 연산들을 실행할 수 있다. 배정도 및 단정도 기능 유닛들은 공유 명령어 발행 회로에 의해 제어될 수 있고, 코어 내에 포함된 배정도 기능 유닛의 복사본들의 수는 단정도 기능 유닛들의 복사본들의 수보다 작을 수 있기 때문에 칩 면적에 대한 배정도 지원을 추가하는 영향을 감소시킬 수 있다.

과제 해결수단

[0009] 본 발명의 일 양상에 따르면, 그래픽 프로세서는 이미지 데이터를 생성하도록 구성된 렌더링 파이프라인을 갖는다. 단정도 피연산자들에 대해 동작하는 렌더링 파이프라인은 다수의 동시 스레드를 실행하도록 구성된 처리 코어를 포함한다. 처리 코어는 배정도 입력 피연산자들의 세트에 대해 다수의 배정도 연산들 중 하나를 선택가능하게 실행하도록 구성된 다중 목적의 배정도 기능 유닛을 포함한다. 다중 목적의 배정도 기능 유닛은 적어도 하나의 산술 로직 회로를 포함하고, 배정도 기능 유닛의 산술 로직 회로들 모두는 배정도로 동작할만큼 충분히 넓다. 일부 실시예들에서, 배정도 기능 유닛은 배정도 연산들이 동일한 클럭 사이클 수에서 완료하도록 구성되며, 그 유닛은 또한 배정도 연산들 중 어느 하나를 완료하기 위해 요구되는 시간(예를 들어, 클럭 사이클 수)이 언더플로우 또는 오버플로우 상태에 의해 영향을 받지 않도록 구성될 수 있다.

[0010] 각종 연산들 및 배정도 연산들의 조합들이 지원될 수 있다. 일 실시예에서, 배정도 연산들은 2개의 배정도 피연산자들을 더하는 덧셈 연산; 2개의 배정도 피연산자들을 곱하는 곱셈 연산; 및 제1 배정도 피연산자 및 제2 배정도 피연산자의 곱을 계산하고 나서 그 곱에 제3 배정도 피연산자를 더하는 결합형 곱셈 덧셈 연산을 포함한다. 지원될 수 있는 다른 배정도 연산들은 제1 피연산자 및 제2 피연산자에 대한 비교 테스트를 수행하고 비교 테스트가 만족되는지를 나타내는 불린(Boolean) 결과를 생성하는 배정도 비교(DSET) 연산, 2개의 배정도 입력 피연산자들 중 큰 것을 반환하는 배정도 최대(DMAX) 연산, 또는 2개의 배정도 입력 피연산자들 중 작은 것을 반환하는 배정도 최소(DMIN) 연산을 포함한다. 또한, 배정도 포맷의 피연산자를 비 배정도 포맷으로 변환(또는 그 반대)하는 포맷 변환 연산들도 지원될 수 있다.

[0011] 본 발명의 다른 양상에 따르면, 그래픽 프로세서는 이미지 데이터를 생성하도록 구성된 렌더링 파이프라인을 포함한다. 렌더링 파이프라인은 다수의 동시 스레드를 실행하도록 구성된 처리 코어를 포함한다. 처리 코어는 하나 이상의 단정도 피연산자들에 대해 산술 연산을 실행하도록 구성된 단정도 기능 유닛 및 배정도 입력 피연산자들의 세트에 대해 결합형 곱셈 덧셈 연산을 실행하고 배정도 결과를 제공하도록 구성된 배정도 결합형 곱셈 덧셈(DFMA) 기능 유닛을 포함한다. DFMA 기능 유닛은 DFMA 파이프라인을 통해 단일 패스로 결합형 곱셈 덧셈 연산을 수행하기에 충분한 데이터 경로들 폭을 갖는 DFMA 파이프라인을 포함하는 것이 바람직하다. 예를 들어, DFMA 기능 유닛은 2개의 배정도 가수(mantissas)의 곱을 단일 반복으로 계산하도록 구성된 곱셈기 및 2개의 배정도 가수의 합을 단일 반복으로 계산하도록 구성된 덧셈기를 포함한다.

[0012] DFMA 기능 유닛은 또한 다른 연산들을 실행하도록 구성될 수 있다. 예를 들어, 일부 실시예들에서 DFMA는 배정도 입력 피연산자들의 쌍에 대해 곱셈 연산을 실행하고 배정도 결과를 제공하도록 구성된다. 일부 실시예들에서, 곱셈 연산 및 결합형 곱셈 덧셈 연산은 각각 동일한 클록 사이클 수에서 완료된다. 마찬가지로, DFMA 기능 유닛은 배정도 입력 피연산자들의 쌍에 대해 덧셈 연산을 실행하고 배정도 결과를 제공하도록 구성될 수 있다. 일 실시예에서, 덧셈 연산 및 결합형 곱셈 덧셈 연산은 각각 동일한 클록 사이클 수에서 완료될 수 있다.

[0013] 일부 실시예들에서, 처리 코어는 병렬로 동작하도록 구성된 제1 기능 유닛의 복사본들의 수(P) 및 DFMA 기능 유닛의 복사본들의 수(N)를 포함하며, 수 P는 수 N보다 크다. 일 실시예에서, 수 N은 1이다.

[0014] 처리 코어는 DFMA 기능 유닛에 대한 배정도 입력 피연산자들의 P 세트를 모으고 배정도 피연산자들의 P 세트 중 상이한 것들을 상이한(예를 들어 연속적인) 클록 사이클에서 DFMA 기능 유닛에 보내도록 구성된 입력 관리자 회로를 포함할 수 있다. 입력 관리자 회로는 제1 기능 유닛에 대한 단정도 입력 연산자들의 P 세트를 모으고, 병렬로, 단정도 피연산자들의 P 세트 중 상이한 것들을 제1 기능 유닛의 P 복사본들의 각각에 보내도록 구성될 수

도 있다.

[0015] 첨부 도면과 함께 이하의 상세한 설명은 본 발명의 특징 및 이점을 가장 잘 이해하도록 할 것이다.

발명의 실시를 위한 구체적인 내용

[0016] 본 발명의 실시예들은 전용 배정도(예를 들어, 64 비트) 기능 유닛을 포함하는 그래픽 프로세서를 제공한다. 일 실시예에서, 배정도 기능 유닛은 덧셈, 곱셈 및 결합형 곱셈 덧셈 연산들 뿐만 아니라 배정도 비교 및 배정도 포맷들에 대한 포맷 변환을 수행할 수 있다.

[0017] I.시스템 개관

[0018] A.컴퓨터 시스템 개관

[0019] 도 1은 본 발명의 실시예에 따른 컴퓨터 시스템(100)의 블록도이다. 컴퓨터 시스템(100)은 CPU(102) 및 메모리 브리지(105)를 포함하는 버스 경로를 통해 통신하는 시스템 메모리(104)를 포함한다. 메모리 브리지(105)는 예를 들어 종래의 노스브리지 칩일 수 있는데, 버스 또는 다른 통신 경로(106)(예를 들어, 하이퍼트랜스포트 링크)를 통해 I/O(입출력) 브리지(107)에 접속된다. I/O 브리지(107)는 예를 들어, 종래의 사우스브리지 칩일 수 있는데, 하나 이상의 사용자 입력 장치(108)(예를 들어, 키보드, 마우스)로부터 사용자 입력을 수신하고 그 입력을 버스(106) 및 메모리 브리지(105)를 통해 CPU(102)에 전송한다. 버스 또는 다른 통신 경로(113), 예를 들어, PCI-E(PCI Express) 또는 AGP(Accelerated Graphics Port) 링크를 통해 메모리 브리지(105)에 결합되는 그래픽 서브시스템(112)의 제어 하에서 동작하는 픽셀 기반 디스플레이 장치(110)(예를 들어, 종래의 CRT 또는 LCD 기반 모니터) 상에 시각적 출력이 제공된다. 또한 시스템 디스크(114)가 I/O 브리지(107)에 접속된다. 스위치(116)는 I/O 브리지(107)와 네트워크 어댑터(118) 및 각종 애드인 카드(120, 121)와 같은 다른 컴포넌트들 사이에 접속을 제공한다. USB 또는 다른 포트 커넥션들, CD 드라이브들, DVD 드라이브들 등을 포함하는 다른 컴포넌트들(특별히 도시되지 않음)도 I/O 브리지(107)에 접속될 수 있다. 각종 컴포넌트들 사이의 버스 접속들은 PCI(Peripheral Component Interconnect), PCI-E, AGP, 하이퍼트랜스포트, 또는 임의의 다른 버스나 점대점 통신 프로토콜(들)과 같은 버스 프로토콜을 이용하여 구현될 수 있고, 상이한 장치들 사이의 접속들은 공지되어 있는 상이한 프로토콜들을 이용할 수 있다.

[0020] 그래픽 처리 서브시스템(112)은 그래픽 처리 유닛(GPU)(122) 및 그래픽 메모리(124)를 포함하며, 그래픽 메모리(124)는 예를 들어 프로그램가능 프로세서들, ASIC(application specific integrated circuit) 및 메모리 장치들과 같은 하나 이상의 집적 회로 장치를 이용하여 구현될 수 있다. GPU(122)는 메모리 브리지(105) 및 버스(113)를 통해 CPU(102) 및/또는 시스템 메모리(104)에 의해 공급되는 그래픽 데이터로부터 픽셀 데이터를 생성하고 그래픽 메모리(124)와 상호작용하여 픽셀 데이터를 저장 및 업데이트 하는 등과 관련된 각종 작업을 수행하도록 구성될 수 있다. 예를 들어, GPU(122)는 CPU(102)에서 실행되는 각종 프로그램들에 의해 제공되는 2차원 또는 3차원 장면 데이터로부터 픽셀 데이터를 생성할 수 있다. GPU(122)는 수신된 픽셀 데이터를 추가의 처리를 하거나 하지 않고 메모리 브리지(105)를 통해 그래픽 메모리(124)에 저장할 수도 있다. GPU(122)는 또한 그래픽 메모리(124)로부터의 픽셀 데이터를 디스플레이 장치(110)에 전달하도록 구성된 스캔아웃 모듈을 포함한다.

[0021] GPU(122)는 또한 그래픽 어플리케이션들에 관한 작업(예를 들어, 비디오 게임을 위한 물리 모델링 등)뿐만 아니라 그래픽 어플리케이션들과 관련 없는 작업을 포함하는 데이터 처리 작업에 대한 범용 계산들을 수행하도록 구성가능하다. 범용 계산에 있어서, GPU(122)는 시스템 메모리(104) 또는 그래픽 메모리(124)로부터의 입력 데이터를 판독하고, 하나 이상의 프로그램을 실행하여 그 데이터를 처리하고 시스템 메모리(104) 또는 그래픽 메모리(124)에 다시 출력 데이터를 기입하는 것이 바람직하다. GPU(122)는 범용 계산들에서 사용되는 하나 이상의 배정도 결합형 곱셈 덧셈 유닛(도 1에 도시되지 않음) 이외에도 렌더링 동작 동안 사용되는 기타 단정도 기능 유닛들을 포함하는 것이 바람직하다.

[0022] CPU(102)는 시스템(100)의 마스터 프로세서로서 동작하여 다른 시스템 컴포넌트들의 동작들을 제어 및 조정한다. 특히, CPU(102)는 GPU(122)의 동작을 제어하는 커맨드를 발행한다. CPU(102)는 GPU(122)에 대한 커맨드들의 스트림을 그래픽 메모리(124) 또는 CPU(102) 및 GPU(122) 모두에 액세스가능한 다른 저장 위치에 존재할 수 있는 커맨드 버퍼에 기입한다. GPU(122)는 커맨드 버퍼로부터의 커맨드 스트림을 판독하고 CPU(102)의 동작과 비동기적으로 커맨드들을 실행한다. 커맨드들은 이미지들을 생성하기 위한 종래의 렌더링 커맨드뿐만 아니라 이미지 생성과 관련 없을 수 있는 데이터 처리용의 GPU(122)의 계산 능력을 레버리지하기 위해 CPU(102)

상에서 어플리케이션이 실행되게 하는 범용 계산 커맨드를 포함할 수 있다.

[0023] 본원에 도시된 시스템은 예시적이고 변경 및 수정이 가능함을 이해할 것이다. 브리지의 수와 구성을 포함하는 버스 토폴로지는 원하는 대로 변경될 수 있다. 예를 들어, 일부 실시예들에서, 시스템 메모리(104)는 브리지를 통해서 보다는 직접 CPU(102)에 접속되고, 다른 장치들은 메모리 브리지(105) 및 CPU(102)를 통해 시스템 메모리(104)와 통신한다. 다른 대안의 토폴로지에서, 그래픽 서브시스템(112)은 메모리 브리지(105)보다는 I/O 브리지(107)에 접속된다. 또 다른 실시예에서, I/O 브리지(107) 및 메모리 브리지(105)는 단일 칩으로 통합될 수 있다. 본원에 도시된 특정 컴포넌트들은 선택적이며, 예를 들어, 임의 수의 애드인 카드 또는 주변 장치들이 지원될 수 있다. 일부 실시예들에서, 스위치(116)는 제거되고, 네트워크 어댑터(118) 및 애드인 카드(120, 121)는 I/O 브리지(107)에 직접 접속된다.

[0024] GPU(122)를 시스템(100)의 나머지에 접속시키는 것은 변경될 수도 있다. 일부 실시예들에서, 그래픽 시스템(112)은 시스템(100)의 확장 슬롯으로 삽입될 수 있는 애드인 카드로서 구현된다. 다른 실시예들에서, GPU는 단일 칩 상에 메모리 브리지(105) 또는 I/O 브리지(107)와 같은 버스 브리지로 통합된다. 또 다른 실시예들에서, GPU(122)의 일부 또는 모든 요소들은 CPU(102)와 통합될 수 있다.

[0025] GPU는 로컬 메모리를 포함하지 않고 임의의 양의 로컬 그래픽 메모리가 제공될 수 있으며, 로컬 메모리와 시스템 메모리를 임의로 조합하여 사용할 수 있다. 예를 들어, UMA(unified memory architecture) 실시예에서, 전용 그래픽 메모리 장치가 제공되지 않고 GPU는 배타적으로 또는 거의 배타적으로 시스템 메모리를 사용한다. UMA 실시예에서, GPU는 버스 브리지 칩으로 통합될 수 있고 또는 GPU를 브리지 칩 및 시스템 메모리에 접속시키는 고속 버스(예를 들어 PCI-E)를 갖는 개별적인 칩으로서 제공될 수 있다.

[0026] 또한, 예를 들어 다수의 GPU를 단일 그래픽 카드 상에 포함시키거나 다수의 그래픽 카드를 버스(113)에 접속함으로써 임의 수의 GPU가 시스템에 포함될 수 있음을 이해해야 한다. 다수의 GPU가 병렬로 동작하여 상이한 디스플레이 장치들 또는 동일한 디스플레이 장치들에 대한 이미지들을 생성할 수 있고, 또는 하나의 GPU가 이미지들을 생성하도록 동작할 수 있는 반면 다른 GPU는 후술하는 배정도 계산들을 포함하는 범용 계산들을 수행한다.

[0027] 부가적으로, 본 발명의 양상들을 구체화하는 GPU는 범용 컴퓨터 시스템, 비디오 게임 콘솔 및 다른 특수 컴퓨터 시스템, DVD 플레이어, 이동 전화 또는 PDA와 같은 핸드헬드 장치 등을 포함하는 각종 장치들로 통합될 수 있다.

[0028] B. 렌더링 파이프라인 개관

[0029] 도 2는 본 발명의 실시예에 따라 도 1의 GPU(122)에 구현될 수 있는 렌더링 파이프라인(200)의 블록도이다. 이 실시예에서, 레더링 파이프라인(200)은 임의의 적용가능한 그래픽 관련 프로그램(예를 들어, 버텍스 셰이더, 지오메트리 셰이더, 및/또는 픽셀 셰이더) 및 범용 계산 프로그램들이 본원에서 "멀티 스레디드 코어 어레이"(202)라고 지칭되는 동일한 병렬 처리 하드웨어를 이용하여 실행되는 아키텍쳐를 이용하여 구현된다.

[0030] 멀티 스레디드 코어 어레이(200) 이외에, 렌더링 파이프라인(200)은 프론트 엔드(204) 및 데이터 어셈블러(206), 셋업 모듈(208), 래스터화기(210), 컬러 어셈블리 모듈(212) 및 래스터 동작 모듈(ROP)(214)을 포함하며, 이들 각각은 종래의 집적 회로 기술 또는 다른 기술을 이용하여 구현될 수 있다.

[0031] 렌더링 동작에 있어서, 프론트 엔드(204)는 상태 정보(STATE), 커맨드(CMD) 및 지오메트리 데이터(GDATA)를 예를 들어 도 1의 CPU(102)로부터 수신한다. 일부 실시예들에서, 지오메트리 데이터를 직접 제공하기보다는 CPU(102)는 지오메트리 데이터가 저장되어 있는 시스템 메모리(104) 내의 위치들에 대한 참조를 제공하고, 데이터 어셈블러(206)는 시스템 메모리(104)로부터 데이터를 검색한다. 렌더링 동작에 있어서, 상태 정보, 커맨드 및 지오메트리 데이터는 일반적으로 종래의 성질을 가질 수 있고 장면의 지오메트리, 라이팅(lighting), 셰이딩, 텍스쳐, 모션 및/또는 카메라 파라미터를 포함하는 원하는 렌더링 이미지 또는 이미지들을 정의하기 위해 사용될 수 있다.

[0032] 상태 정보 및 렌더링 커맨드는 렌더링 파이프라인(200)의 각종 스테이지에 대한 처리 파라미터 및 동작을 정의한다. 프론트 엔드(204)는 상태 정보 및 렌더링 커맨드들을 제어 경로(명확히 도시하지 않음)를 통해 렌더링 파이프라인(200)의 다른 컴포넌트에 보낸다. 공지되어 있는 바와 같이, 이들 컴포넌트들은 처리 동안 액세스되는 각종 제어 레지스터들 내의 값들을 저장하거나 업데이트함으로써 수신된 상태 정보에 응답할 수 있고 파이프라인에서 수신된 데이터를 처리함으로써 렌더링 커맨드에 응답할 수 있다.

[0033] 프론트 엔드(204)는 지오메트리 데이터를 데이터 어셈블러(206)에 보낸다. 데이터 어셈블러(206)는 지오메트리

데이터를 포매팅하고 멀티 스레디드 코어 어레이(202) 내의 지오메트리 모듈(218)로 전달하기 위해 지오메트리 데이터를 준비한다.

[0034] 지오메트리 모듈(218)은 프로그램가능 처리 엔진(명확히 도시하지 않음)을 멀티 스레디드 코어 어레이(202)로 보내어 버텍스 데이터 상에서 버텍스 및/또는 지오메트리 셰이더 프로그램들을 실행하는데, 그 프로그램들은 프론트 엔드(204)에 의해 제공되는 상태 정보에 응답하여 선택된다. 버텍스 및/또는 지오메트리 셰이더 프로그램들은 공지되어 있는 렌더링 어플리케이션에 의해 특정될 수 있고, 상이한 셰이더 프로그램들이 상이한 버텍스 및/또는 프리미티브에 적용될 수 있다. 일부 실시예들에서, 버텍스 셰이더 프로그램들 및 지오메트리 셰이더 프로그램들은 멀티 스레디드 코어 어레이(202) 내에서 동일한 프로그램가능 처리 코어들을 이용하여 실행된다. 따라서, 특정 시간에, 주어진 처리 코어가 버텍스 셰이더로서 동작하여, 버텍스 프로그램 명령어들을 수신 및 실행할 수 있고, 다른 시간에 동일한 처리 코어가 지오메트리 셰이더로서 동작하여 지오메트리 프로그램 명령어들을 수신 및 실행할 수 있다. 처리 코어는 멀티 스레디드될 수 있고, 상이한 유형의 셰이더 프로그램들을 실행하는 상이한 스레드들은 멀티 스레디드 코어 어레이(202) 내에서 동시에 진행될(in flight) 수 있다.

[0035] 버텍스 및/또는 지오메트리 셰이더 프로그램들이 실행된 후에, 지오메트리 모듈(218)은 처리된 지오메트리 데이터(GDATA')를 셋업 모듈(208)로 보낸다. 셋업 모듈(208)은 일반적으로 종래의 설계일 수 있는데, 각 프리미티브의 클립 공간 또는 스크린 공간 좌표들로부터 에지 방정식들을 생성하며, 바람직하게는 그 에지 방정식들은 스크린 공간 내의 포인트가 프리미티브 내부 또는 외부에 있는지를 결정하기 위해 이용가능하다.

[0036] 셋업 모듈(208)은 각 프리미티브(PRIM)를 래스터화기(210)에 제공한다. 래스터화기(210)는 일반적으로 종래의 설계일 수 있는데, 예를 들어 종래의 스캔 변환 알고리즘들을 이용하여, 프리미티브에 의해 어떤 픽셀들(존재하는 경우)이 커버되는지를 결정한다. 본원에서 사용되는 바와 같이, "픽셀"(또는 "프래그먼트")는 일반적으로 단일 컬러 값이 결정되는 2차원 스크린 공간 내의 영역을 지칭하며, 픽셀들의 수 및 구성은 렌더링 파이프라인(200)의 구성가능한 파라미터일 수 있고 특정 디스플레이 장치의 스크린 해상도와 함께 연관될 수 있거나 연관되지 않을 수 있다.

[0037] 어떤 픽셀이 프리미티브에 의해 커버되는지를 결정한 후에, 래스터화기(210)는 프리미티브에 의해 커버되는 픽셀들의 스크린 좌표(X,Y)의 리스트와 함께 프리미티브(PRIM)를 컬러 어셈블리 모듈(212)에 제공한다. 컬러 어셈블리 모듈(212)은 그 프리미티브 및 래스터화기(21)로부터 수신된 커버리지 정보를 프리미티브의 버텍스의 속성들(예를 들어, 컬러 성분, 텍스쳐 좌표, 표면 법선)과 연관시키고 그 속성들 중 일부 또는 전부를 스크린 좌표 공간 내의 위치의 함수로서 정의하는 평면 방정식(또는 다른 적합한 방정식)을 생성한다.

[0038] 바람직하게는, 이들 속성 방정식들은 프리미티브 내의 임의의 위치에서 속성에 대한 값을 계산하기 위해 픽셀 셰이더 프로그램에서 사용가능하며, 종래의 기술들은 그 방정식들을 생성하는 데 사용될 수 있다. 예를 들어, 일 실시예에서, 컬러 어셈블리 모듈(212)은 각 속성 U에 대해 U=Ax+By+C의 형태의 평면 방정식에 대한 계수들 A, B, C를 생성한다.

[0039] 컬러 어셈블리 모듈(212)은 커버된 픽셀들의 스크린 좌표들(X,Y)의 리스트 및 픽셀의 적어도 하나의 샘플링 위치를 커버하는 각 프리미티브에 대한 속성 방정식(EQS, 이것은 예를 들어 평면 방정식 계수들 A, B, C를 포함할 수 있음)을 멀티스레디드 코어 어레이(202) 내의 픽셀 모듈(224)에 제공한다. 픽셀 모듈(224)은 프로그램가능 처리 엔진(명확히 도시되지 않음)을 멀티 스레디드 코어 어레이(202)로 보내서 프리미티브에 의해 커버되는 각 픽셀에 대해 하나 이상의 픽셀 셰이더 프로그램들을 실행하는데, 프로그램(들)은 프론트 엔드(204)에 의해 제공되는 상태 정보에 응답하여 선택된다. 버텍스 셰이더 프로그램들 및 지오메트리 셰이더 프로그램들로써, 렌더링 어플리케이션들은 임의의 주어진 픽셀들의 세트에 대해 사용되는 픽셀 셰이더 프로그램을 특정할 수 있다.

[0040] 픽셀 셰이더 프로그램들은 바람직하게는 버텍스 및/또는 지오메트리 셰이더 프로그램들을 실행하는 동일한 프로그램가능 처리 엔진들을 이용하여 멀티 스레디드 코어 어레이(202)에서 실행된다. 따라서, 특정 시간에, 주어진 처리 엔진은 버텍스 셰이더로서 동작하여 버텍스 프로그램 명령어들을 수신 및 실행할 수 있고, 다른 시간에 동일한 처리 엔진이 지오메트리 셰이더로서 동작하여 지오메트리 프로그램 명령어들을 수신 및 실행할 수 있고, 또 다른 시간에 동일한 처리 엔진이 픽셀 셰이더로서 동작하여 픽셀 셰이더 프로그램 명령어들을 수신 및 실행할 수 있다.

[0041] 일단 픽셀 또는 픽셀들의 그룹에 대한 처리가 완료되면, 픽셀 모듈(224)은 처리된 픽셀(PDATA)을 ROP(214)에 제공한다. ROP(214)는 일반적으로 종래의 설계일 수 있는데, 픽셀 모듈(224)로부터 수신된 픽셀 값들을, 예를 들어 그래픽 메모리(124) 내에 위치될 수 있는 프레임 버퍼(226) 내에 구성되는 이미지의 픽셀들과 통합한다. 일

부 실시예들에서, ROP(214)는 픽셀들을 마스크하거나 새로운 픽셀들을 렌더링된 이미지에 미리 기입된 픽셀들과 혼합할 수 있다. 또한 심도 버퍼들, 알파 버퍼들 및 스텐실 버퍼들을 사용하여 렌더링된 이미지로의 각각의 인입 픽셀의 기여(존재하는 경우)를 결정할 수 있다. 각 인입 픽셀 값과 임의의 이전에 저장된 픽셀 값의 적절한 조합에 대응하는 픽셀 데이터 PDATA'가 프레임 버퍼(226)에 다시 기입된다. 이미지가 완성되면, 프레임 버퍼(226)는 디스플레이 장치 외부에 스캔되거나 및/또는 추가로 처리될 수 있다.

[0042] 범용 계산을 위해, 멀티 스레디드 코어 어레이는 픽셀 모듈(224)에 의해 (또는 지오메트리 모듈(218)에 의해) 제어될 수 있다. 프론트 엔드(204)는 예를 들어 도 1의 CPU(102)로부터 상태 정보(STATE) 및 처리 커맨드(CMD)를 수신하고 그 상태 정보 및 커맨드들을 제어 경로(명확히 도시하지 않음)를 통해 예를 들어 컬러 어셈블리 모듈(212) 또는 픽셀 모듈(224)로 통합될 수 있는 작업 분배 유닛으로 전달한다. 작업 분배 유닛은 멀티 스레디드 코어 어레이(202)를 이루는 처리 코어들 사이에서 처리 작업들을 분배한다. 각종 작업 분배 알고리즘들이 사용될 수 있다.

[0043] 각 처리 작업은 바람직하게는 다수의 처리 스레드를 실행하는 단계를 포함하며, 각 스레드는 동일한 프로그램을 실행한다. 프로그램은 바람직하게는 "글로벌 메모리"(예를 들어, 시스템 메모리(104), 그래픽 메모리(124) 또는 GPU(122) 및 CPU(102) 모두에 액세스 가능한 임의의 다른 메모리)로부터의 입력 데이터를 판독하고, 그 입력 데이터에 대한 적어도 일부의 배정도 연산들을 포함하는 각종 연산들을 수행하여 출력 데이터를 생성하고, 그 출력 데이터를 글로벌 메모리에 기입하라는 명령어들을 포함한다. 특정 처리 작업들은 본 발명에 중요하지 않다.

[0044] 본원에 기술된 렌더링 파이프라인은 예시적인 것이며 변형 및 수정이 가능하다는 것을 알 것이다. 파이프라인은 도시된 것과 상이한 유닛들을 포함할 수 있으며, 처리 이벤트들의 순서는 본원에 기술된 것과 다를 수 있다. 또한, 본원에 기술된 모듈들 중 일부 또는 전부의 다수 예들이 병렬로 동작될 수 있다. 하나의 이러한 실시예에서, 멀티 스레디드 코어 어레이(202)는 2개 이상의 지오메트리 모듈들(218) 및 병렬로 동작하는 동일한 수의 픽셀 모듈들(224)을 포함한다. 각각의 지오메트리 모듈 및 픽셀 모듈은 멀티 스레디드 코어 어레이(202) 내의 상이한 처리 엔진의 서브세트를 공동으로 제어한다.

[0045] C. 코어 개관

[0046] 멀티 스레디드 코어 어레이(202)는 유리하게, 대다수의 처리 스레드의 병렬 실행을 위해 적응된 하나 이상의 처리 코어를 포함하며, 여기서 "스레드"라는 용어는 특정 입력 데이터 세트에 대해 실행하는 특정 프로그램의 예를 가리킨다. 예를 들어, 스레드는 단일 버텍스의 속성들에 대해 실행하는 버텍스 셰이더 프로그램의 예, 주어진 프리미티브(primitive) 및 픽셀에 대해 실행하는 픽셀 셰이더 프로그램, 또는 범용 계산 프로그램의 예일 수 있다.

[0047] 도 3은 본 발명의 실시예에 따른 실행 코어(300)의 블록도이다. 예를 들어, 전술한 멀티 스레디드 코어 어레이(202)에서 구현될 수 있는 실행 코어(300)는 다양한 계산들을 수행하기 위한 임의의 명령어 시퀀스를 실행하도록 구성된다. 일부 실시예들에서, 범용 계산 프로그램뿐 아니라, 버텍스 셰이더, 지오메트리 셰이더, 및/또는 픽셀 셰이더 프로그램을 포함하여 그래픽 렌더링의 모든 측면에서 셰이더 프로그램들을 실행하기 위해 동일한 실행 코어(300)를 사용할 수 있다.

[0048] 실행 코어(300)는 페치 및 디스패치 유닛(302), 발행 유닛(304), DFMA(double-precision fused multiply-add) 유닛(320), 다수(N)의 다른 기능 유닛들(FU)(322), 및 레지스터 파일(324)을 포함한다. 각각의 기능 유닛(320, 322)은 특정 동작들을 수행하도록 구성된다. 일 실시예에서, DFMA 유닛(320)은 유리하게, 후술한 바와 같이 다른 배정도 연산들 이외에 DFMA 연산을 구현한다. 임의 수의 DFMA 유닛(320)이 코어(300)에 포함될 수 있다.

[0049] 다른 기능 유닛들(322)은 일반적으로 통상의 설계로 되어 있을 수 있으며, 단정도(single-precision) 덧셈, 곱셈, 비트 논리 연산, 비교 연산, 포맷 변환 연산, 텍스처 필터링, 메모리 액세스(예를 들어, 로드 및 저장 연산들), 선험적 함수의 근사, 보간법 등과 같은 다양한 연산을 지원할 수 있다. 기능 함수들(320, 322)은 파이프라인으로 될 수 있어, 이 기술분야에 알려진 바와 같이, 이전 명령어가 완료되기 전에 새로운 명령어가 발행될 수 있게 한다. 임의 조합의 기능 유닛들이 제공될 수 있다.

[0050] 실행 코어(300)의 동작 동안, 페치 및 디스패치 유닛(302)은 명령어 저장소(도시되지 않음)로부터 명령어들을 취득하고, 그 명령어들을 디코딩하고, 그 명령어들을 연관된 피연산자 레퍼런스들 또는 피연산자 데이터를 갖는 연산 코드로서 발행 유닛(304)에 디스패치한다. 각각의 명령어에 대하여, 발행 유닛(304)은 예를 들어, 레지스터 파일(324)로부터 임의의 레퍼런스된 피연산자를 취득한다. 명령어에 대한 모든 피연산자가 준비되어 있을

때, 발행 유닛(304)은 DFMA 유닛(320) 또는 다른 기능 유닛(322)에 연산 코드 및 피연산자들을 보냄으로써 명령어를 발행한다. 발행 유닛(304)은 유리하게, 연산 코드를 사용하여 적절한 기능 유닛을 선택하고 주어진 명령어를 실행한다. 페치 및 디스패치 유닛(302) 및 발행 유닛(304)은 종래의 마이크로프로세서 아키텍처들 및 기술들을 이용하여 구현될 수 있고, 상세한 설명은 본 발명의 이해에 중요한 것이 아니기 때문에 생략한다.

[0051] DFMA 유닛(320) 및 다른 기능 유닛(322)은 연산 코드 및 연관된 피연산자들을 수신하고 피연산자들에 대해 특정 연산을 수행한다. 데이터 전송 경로(326)를 통해 레지스터 파일(324)(또는 다른 목적지)에 전달될 수 있는 결과값들의 형태로 결과 데이터가 제공된다. 레지스터 파일(324)은 일부 실시예들에서 데이터가 스레드들 간에 공유될 수 있게 하는 글로벌 레지스터 파일뿐만 아니라 특정 스레드들에 할당된 섹션들을 갖는 로컬 레지스터 파일을 포함한다. 레지스터 파일(324)은 프로그램 실행 동안 입력 데이터, 중간 결과들 등을 저장하는 데 이용될 수 있다. 레지스터 파일(324)의 특정 구현은 본 발명에 대해 중요하지 않다.

[0052] 일 실시예에서, 코어(300)는 멀티스레드되고 예를 들어, 각 스레드와 연관된 현재 상태 정보를 유지하는 것에 의해 최대 수(예를 들어, 384, 768)의 스레드들까지 동시에 실행할 수 있다. 코어(300)는 유리하게, 한 스레드에서 다른 스레드들과 신속하게 스위치되도록 설계되어, 예를 들어, 버텍스로부터의 프로그램 명령어가 한 클록 사이클에 발행될 수 있고, 그 다음에 상이한 버텍스 스레드로부터의 또는 지오메트리 스레드 또는 픽셀 스레드 등과 같은 상이한 유형의 스레드로부터의 프로그램 명령어가 발행될 수 있다.

[0053] 도 3의 실행 코어는 예시적인 것이며 변형 및 수정이 가능하다는 것을 알 것이다. 임의 수의 코어가 프로세서 내에 포함될 수 있고, 임의 수의 기능 유닛이 코어 내에 포함될 수 있다. 페치 및 디스패치 유닛(302) 및 발행 유닛(304)은 순차적(in-order) 또는 비순차적(out-of-order) 명령어 발행의 스칼라(scalar), 수퍼스칼라(super scalar) 또는 벡터 아키텍처들; 추론적 실행 모드; SIMD(single-instruction, multiple data) 명령어 발행 등을 포함한 임의의 원하는 마이크로아키텍처를 필요에 따라 구현할 수 있다. 일부 아키텍처들에서, 발행 유닛은 다수의 기능 유닛을 위한 연산 코드들 및 피연산자들 또는 하나의 기능 유닛을 위한 연산 코드들 및/또는 피연산자들을 포함하는 긴 명령어 워드를 수신 및/또는 발행할 수 있다. 일부 아키텍처들에서, 실행 코어는 예를 들어, SIMD 명령어들의 실행을 위해 병렬로 동작가능한 각각의 기능 유닛의 복수 예들을 포함할 수 있다. 실행 코어는 또한 한 스테이지에서 기능 유닛들로부터의 결과들이 레지스터 파일에 직접 보내지기 보다는 이후 스테이지들에서의 기능 유닛들에 보내지는 일련의 파이프라인의 기능 유닛들을 포함할 수 있고, 이러한 구성의 기능 유닛들은 단일의 긴 명령어 워드 또는 별도의 명령어들에 의해 제어될 수 있다.

[0054] 또한 본 교시들을 접한 이 기술분야의 통상의 기술자들이면, DFMA 유닛(320)이 그래픽 프로세서들 또는 임의의 특정 프로세서 또는 실행 코어 아키텍처로 한정되지 않고, 임의의 마이크로프로세서 내의 기능 유닛으로서 구현될 수 있다는 것을 인식할 것이다. 예를 들어, DFMA 유닛(320)은 범용 병렬 처리 유닛 또는 CPU에 구현될 수 있다.

[0055] C. DFMA 유닛 개관

[0056] 본 발명의 일 실시예에 따르면, 실행 코어(300)는 세가지 연산 클래스들, 즉, 배정도 산술, 비교 연산들, 및 배정도와 다른 포맷들 간의 포맷 변환들을 실행하는 DFMA 유닛(320)을 포함한다.

[0057] DFMA 유닛(320)은 유리하게, 배정도 부동 소수점 포맷으로, 그리고 변환 연산들의 경우, 다른 부동 소수점 및 고정 소수점 포맷들로 입력들 및 출력들을 조정하고, 상이한 연산들에 대한 피연산자들이 상이한 포맷들로 될 수 있다. DFMA 유닛(320)의 실시예를 기술하기 전에, 대표적인 포맷들이 정의될 것이다.

[0058] 본원에서 이용된 바와 같이, "Fp32"는, 정상 부동 소수점 수가 단일 비트, 8 지수 비트, 및 23 유효 숫자 비트로 표현되는 표준 IEEE 754 단정도 부동 소수점 포맷을 가리킨다. 지수는 127만큼 위로 바이어스되어 범위 2^{-126} 내지 2^{127}의 범위의 지수들이 1 내지 254의 정수를 이용하여 표현된다. "정상" 수들의 경우, 23 유효 숫자 비트들이 정수 부분으로서 암시적 1을 갖는 24 비트 가수의 분수 부분으로서 해석된다. ("유효 숫자"라는 용어는 선두 1(leading 1)이 암시적일 때 본원에서 이용되고, "가수"는 적용가능한 경우, 선두 1이 명시적으로 되었음을 나타내기 위해 이용된다.)

[0059] 본원에서 이용된 바와 같이, "Fp64"는, 정상 부동 소수점 수가 부호 비트, 11 지수 비트, 및 52 유효 숫자 비트로 표현되는 표준 IEEE 754 배정도 부동 소수점 포맷을 가리킨다. 지수는 1023만큼 위로 바이어스되어 범위 2^{-1022} 내지 2^{1023}의 범위의 지수들이 1 내지 2046의 정수를 이용하여 표현된다. "정상" 수들의 경우, 52 유효 숫자

비트들이 정수 부분으로서 암시적 1을 갖는 53 비트 가수의 분수 부분으로서 해석된다.

[0060] 본원에서 이용된 바와 같이, "Fp16"은, 정상 부동 소수점 수가 부호 비트, 5 지수 비트, 및 10 유효 숫자 비트로 표현되는, 그래픽에서 보통 사용되는 "반정도(half-precision)" 부동 소수점 포맷을 가리키다. 지수는 15만큼 위로 바이어스되어 범위 2^{-14} 내지 2^{15}의 범위의 지수들이 1 내지 30의 정수를 이용하여 표현된다. "정상" 수들의 경우, 10 유효 숫자 비트들이 정수 부분으로서 암시적 1을 갖는 11 비트 가수의 분수 부분으로서 해석된다.

[0061] fp16, fp32 및 fp64 포맷들에서, 지수 비트들이 모두 0인 수들은 비정규화된 수들(또는 "비정규화 수들(denorms)")로 지칭되고 가수에 암시적 선두 1을 갖지 않는 것으로 해석되며, 이러한 수들은 예를 들어, 계산에서 언더플로우(underflow)를 나타낼 수 있다. 지수 비트들이 모두 1이고 유효 숫자 비트들이 모두 0인 (플러스 또는 마이너스) 수는 (플러스 또는 마이너스) INF를 지칭하며, 이 수는 예를 들어, 계산에서 오버플로우(overflow)를 나타낼 수 있다. 지수 비트들이 모두 1이고 유효 숫자 비트들이 0이 아닌 수는 NaN(Not a Number)을 지칭하며 예를 들어, 정의되지 않은 값을 나타내는 데 이용될 수 있다. 0은 또한 특수수(special number)로 고려되며, 0으로 설정되어 있는 지수 및 유효 숫자 비트들의 전부에 의해 표현된다. 0은 어느 한쪽의 부호를 가질 수 있으며, 따라서 플러스와 마이너스 둘다의 0이 허용된다.

[0062] 고정 소수점 포맷들은 포맷이 부호가 있는지 또는 부호가 없는지를 나타내는 머리글자 "s" 또는 "u" 및 비트들의 총 수(예를 들어, 16, 32, 64)를 나타내는 수에 의해 본원에 특정되어 있으며, 따라서 s32는 부호 있는 32 비트 포맷을 지칭하고, u64는 부호 없는 64 비트 포맷을 지칭한다. 부호 있는 포맷의 경우, 2의 보수 표현이 유리하게 이용된다. 본원에 이용된 모든 포맷에서, MSB(most significant bit)는 비트 필드의 좌측에 있고, LSB(least significant bit)는 비트 필드의 우측에 있다.

[0063] 이 포맷들은 예시를 위해 본원에 정의되고 언급되어 있으며, DFMA 유닛이 본 발명의 범위로부터 벗어나지 않고 이들 포맷 또는 상이한 포맷들의 임의의 조합을 지원한다는 것을 이해할 것이다. 특히, "단정도" 및"배정도"는 현재 정의된 표준들에 한정되지 않고 임의의 2개의 상이한 부동 소수점 포맷을 가리킬 수 있으며, 배정도 포맷(예를 들어, fp64)은 더 큰 범위의 부동 소수점 수를 표현하기 위해 및/또는 더 높은 정밀도로 부동 소수점 값들을 표현하기 위해 관련 단정도 포맷(예를 들어, fp32)보다 많은 수의 비트를 이용하는 임의의 포맷을 가리킨다. 마찬가지로, "반정도"는 일반적으로 더 작은 범위의 부동 소수점 수를 표현하기 위해 및/또는 더 낮은 정밀도로 부동 소수점 수들을 표현하기 위해 관련 단정도 포맷보다 적은 수의 비트를 이용하는 포맷을 가리킬 수 있다.

[0064] 이제 본 발명에 따른 DFMA 유닛(320)의 실시예가 설명될 것이다. 도 4는 DFMA 유닛(320)의 본 실시예에 의해 수행될 수 있는 배정도 산술, 비교 연산들, 및 포맷 변환 연산들을 열거한 표(400)이다.

[0065] 섹션(402)은 산술 연산들을 열거한다. 덧셈(DADD)은 2개의 fp64 입력들 A 및 C를 더하고, fp64 합 A+C를 반환한다. 곱셈(DMUL)은 2개의 fp64 입력들 A 및 B를 곱하고 fp64 곱 A*B를 반환한다. 결합형(fused) 곱셈 덧셈(DFMA)은 3개의 fp64 입력들 A, B, C를 수신하고 A*B+C를 계산한다. 이 연산은 곱 A*B가 C에 더해지기 전에 라운딩되지(rounded) 않고; 정확한 값 A*B를 이용하여 정확도를 향상시키고 부동 소수점 산술을 위한 이번의 IEEE 754R 표준에 따른다는 점에서 "결합형(fused)"이다.

[0066] 섹션(404)은 비교 연산들을 열거한다. 최대 연산(DMAX)은 fp64 피연산자들 A 및 B 중 더 큰 것을 반환하고, 최소 연산(DMIN)은 둘 중 더 작은 것을 반환한다. 바이너리 테스트 연산(DSET)은 배정도 피연산자들 A 및 B에 대해 다수의 바이너리 관계 테스트들 중 하나를 수행하고, 그 테스트가 만족되는지 여부를 나타내는 불린(Boolean) 값을 반환한다. 본 실시예에서, 테스트될 수 있는 바이너리 관계들은 보다 더 큼(A>B), 보다 더 작음(A<B), 동일함(A=B), 및 순서없음(unordered)(A?B, 이것은 A 또는 B가 NaN인 경우 참임), 그리고 부정(negation)(예를 들어, A≠B) 및 다양한 조합 테스트들(예를 들어, A≥B, A<>B, A?=B 등)을 포함한다.

[0067] 섹션(406)은 포맷 변환들 및 라운딩 연산들을 열거한다. 본 실시예에서, DFMA 유닛(320)은 fp64 포맷 수들을 다른 64 비트 또는 32 비트 포맷들의 수들로 및 그 반대로 변환할 수 있다. D2F 연산은 피연산자 A를 fp64로부터 fp32로 변환하고, F2D는 피연산자 A를 fp32로부터 fp64로 변환한다. D2I 연산은 피연산자 A를 fp64로부터 s64, u64, s32 및 u32 포맷들 중 임의의 하나로 변환하고, 별개의 연산 코드들이 타겟 포맷을 식별하는 데 이용될 수 있다는 것을 이해할 것이다. I2D 연산은 정수 피연산자 C를 s64, u64, s32 및 u32 포맷들 중 임의의 하나로부터 fp64 포맷으로 변환하며, 또한 별개의 연산 코드들이 소스 포맷을 식별하는 데 이용될 수 있다는 것을 이해할 것이다. 본 실시예에서, 배정도 포맷으로의 또는 배정도 포맷으로부터의 모든 변환들은 DFMA 유닛(32

0)에 의해 지원되고, 다른 기능 유닛들이 다른 포맷 변환들(예를 들어, fp32와 fp16 간, fp32와 정수 포맷들 간 등)을 수행할 수 있다.

[0068] D2D 연산은 IEEE 라운딩 모드들과 같은 라운딩 연산들을 fp64 피연산자에 적용하는 데 이용된다. 이 연산들은 fp64 피연산자를 fp64 포맷으로 표현된 정수 값으로 라운딩한다. 일 실시예에서, 지원되는 D2D 연산들은 절단(0을 향해 라운딩), 실링(+INF를 향해 라운딩), 플로어(-INF를 향해 라운딩), 및 니어리스트(가장 근접한 정수로 라운딩 업 또는 다운(up or down))를 포함한다.

[0069] 본 실시예에서, DFMA 유닛(320)은 나눗셈, 나머지(remainder), 또는 제곱근과 같은 더 진보한 수학적 함수들을 위해 직접 하드웨어 지원을 제공하지 않는다. 그러나, DFMA 유닛(320)은 이 연산들의 소프트웨어 기반 구현들을 가속시키는 데 이용될 수 있다. 예를 들어, 나눗셈에 대한 하나의 공통 접근법이 몫(q=a/b)을 추정한 다음, t=q*b-a를 이용하여 그 추정값을 테스트한다. t가 0이면, 몫 q는 정확하게 결정되었다. 그렇지 않으면, t의 크기는 추정된 몫 q를 수정하는 데 이용되고, t가 0이 될 때까지 테스트는 반복된다. 각각의 반복에서의 테스트 결과 t는 단일 DFMA 연산(A=q, B=b, C=-A)을 이용하여 정확하게 계산될 수 있다. 유사하게, 제곱근의 경우, 하나의 공통 접근법은 $r=a^{1/2}$를 추정하고 나서, t=r*r-a를 계산하여 그 추정값을 테스트하고 t가 0이 아니면 r을 수정한다. 또한, 각각의 반복에서의 테스트 결과 t는 단일 DFMA 연산(A=B=r, C=-a)을 이용하여 정확하게 계산될 수 있다.

[0070] 섹션 II 및 III은 도 4에 도시된 연산들 전부를 수행할 수 있는 DFMA 유닛(320)을 기술하고 있다. 섹션 II은 DFMA 유닛(320)에 대한 회로 구조를 기술하고 있으며, 섹션 III은 그 회로 구조가 도 4에 열거된 연산들을 실행하는 데 어떻게 이용될 수 있는지를 기술하고 있다. 본원에 기술된 DFMA 유닛(320)은 예시적인 것이며, 다른 또는 상이한 기능 조합들이 적절한 회로 블록 조합을 이용하여 지원될 수 있다는 것을 이해할 것이다.

[0071] II. DFMA 유닛 구조

[0072] 도 5는 도 4에 도시된 모든 연산들을 지원하는 본 발명의 실시예에 따른 DFMA 유닛(320)의 간략한 블록도이다. 본 실시예에서, DFMA 유닛(320)은 모든 연산들에 대해 이용되는 멀티-스테이지 파이프라인을 구현한다. 각 프로세서 사이클에서, DFMA 유닛(320)은 (예를 들어, 도 3의 발행 유닛(304)으로부터) 피연산자 입력 경로들(502, 504, 506)을 통해 3개의 새로운 피연산자(A0, B0, C0) 및 연산 코드 경로(508)를 통해 수행될 연산을 나타내는 연산 코드를 수신할 수 있다. 본 실시예에서, 연산은 도 4에 도시된 임의의 연산일 수 있다. 연산 이외에, 연산 코드는 유리하게, 피연산자들을 위한 입력 포맷 및 결과에 대해 이용하기 위한 출력 포맷을 나타내며, 이 출력 포맷은 입력 포맷과 동일할 수도 동일하지 않을 수도 있다. 도 4에 도시된 연산이 그와 연관된 다수의 연산 코드를 가질 수 있다는 것에 주목해야 하며, 예를 들어, s64 출력을 갖는 D2I를 위한 하나의 연산 코드 및 s32 출력을 갖는 D2I를 위한 다른 연산 코드 등이 있을 수 있다.

[0073] DFMA 유닛(320)은 각각의 연산을 그의 파이프라인 스테이지들 전부를 통해 처리하고, 신호 경로(510)에 64 비트(또는 특정 포맷-변환 연산들을 위한 32 비트) 결과값(OUT)을 생성한다. 이 신호들은 아키텍처에 따라, 예를 들어, 도 3에 도시된 바와 같은 레지스터 파일(324), 발행 유닛(304), 또는 프로세서 코어의 다른 요소들에 전달될 수 있다. 일 실시예에서, 파이프라인 스테이지는 프로세서 사이클에 대응하며, 다른 실시예들에서, 스테이지는 다수의 프로세서 사이클을 포함할 수 있다. 또한, 파이프라인 내의 상이한 경로들은 유리하게, 병렬로 동작한다.

[0074] 섹션 II.A는 DFMA 파이프라인의 개요를 제공하고, 섹션 II.B-I는 각 섹션의 회로 블록들을 상세하게 기술하고 있다.

[0075] A. DFMA 파이프라인

[0076] DFMA 연산 동안 회로 블록들이 어떻게 이용되는지를 참조하여 파이프라인의 처음 이해가 이루어질 수 있다. 피연산자 준비 블록(514)은 피연산자 포맷팅(fp64 포맷으로 아직 준비되지 않은 피연산자들의 경우) 및 특수수 검출을 수행하고, 피연산자 준비 블록(514)은 또한 입력 fp64 피연산자들로부터 가수(Am, Bm, Cm), 지수(Ae, Be, Ce), 및 부호 비트들(As, Bs, Cs)을 추출한다. 일 실시예에서, 불법적인 피연산자 조합들이 없고, 특정 연산에 이용되지 않는 임의의 피연산자들은 간단하게 무시될 수 있다.

[0077] 가수 경로(516)는 가수들 Am 및 Bm의 곱을 계산한다. 병렬로, 지수 경로(518)는 지수들 Ae 및 Be을 사용하여 곱 A*B와 피연산자 C 간 상대 정렬(relative alignment)을 결정하고, 가수 경로(516)에 피연산자 C에 대한 정렬된 가수(C_align)를 공급한다. 가수 경로(516)은 곱 Am*Bm에 C_align을 더하고 나서, 그 결과를 정규화한다.

그 정규화에 기초하여, 가수 경로(516)는 정렬 신호(ALIGN_NORM)를 지수 경로(518)에 제공하고, 지수 경로(518)는 지수들 Ae, Be 및 Ce와 함께 ALIGN_NORM 신호를 이용하여 그 최종 결과에 대한 지수를 결정한다.

[0078] 부호 경로(520)는 피연산자 준비 블록(514)로부터 부호 비트들 As, Bs 및 Cs를 수신하고 그 결과의 부호를 결정한다. 가수 경로(516)는 그 결과가 0인 경우들을 검출하고 0 결과(R_ZERO)를 부호 경로(520)로 보낸다.

[0079] 출력부(522)는 가수 경로(516)로부터 결과 가수 Rm을, 지수 경로(518)로부터 결과 지수 Re를, 그리고 부호 경로(520)로부터 결과 부호 Rs를 수신한다. 출력부(522)는 또한 피연산자 준비 블록(514)으로부터 특수수 신호들(special number signals; SPC)을 수신한다. 이 정보에 기초하여, 출력부(522)는 출력 경로(510)에 전달을 위한 최종 결과(OUT)를 포맷팅하고, 출력 경로(512)에 상태 코드(COND)를 생성한다. 유리하게, 그 결과보다 적은 비트를 포함하는 상태 코드는 그 결과의 특징에 관한 일반적인 정보를 나른다. 예를 들어, 상태 코드는 그 결과가 플러스, 마이너스, 0, NaN, INF, 비정규화 수 등인지를 나타내는 비트들을 포함할 수 있다. 이 기술분야에 알려져 있는 바와 같이, 상태 코드가 제공되는 경우, 그 결과의 후속 컨슈머(consumer)가 때때로 그의 처리 시에 결과 자체보다는 상태 코드를 이용할 수 있다. 일부 실시예들에서, 상태 코드는 연산의 실행 동안 예외 또는 다른 이벤트의 발생을 나타내는 데 이용될 수 있다. 다른 실시예들에서, 상태 코드는 완전히 생략될 수 있다.

[0080] "가수 경로" 및 "지수 경로"와 같은 명칭들은 각 경로의 다양한 회로 블록에 의해 특정 연산들(예를 들어, DFMA) 동안 수행되는 기능들을 시사하고 있지만, 내부 데이터 경로들 중 임의의 것을 따르는 회로 블록들이 연산에 따라 다양한 용도로 레버리지될 수 있다는 것을 이해해야 한다. 예들은 후술된다.

[0081] 데이터 경로들 이외에, DFMA 유닛(320)은 또한 제어 블록(530)에 의해 도 5에 표시된 제어 경로를 제공한다. 제어 블록(530)은 연산 코드를 수신하고, 파이프라인을 통한 데이터 전달과 동기하여 각 회로 블록에 전달될 수 있는 "OPCTL"로서 본원에 일반적으로 표시된 다양한 연산 코드 종속 제어 신호들을 생성한다. (OPCTL 신호들의 다양한 회로 블록들로의 접속은 도 5에 도시되어 있지 않다.) 후술하는 바와 같이, OPCTL 신호들은 상이한 연산들이 동일한 파이프라인 요소들을 이용하여 수행될 수 있도록 연산 코드에 응답하여 DFMA 유닛(3220)의 다양한 회로 블록들의 연산을 인에이블(enable), 디스에이블(disable), 및 제어하는 데 이용될 수 있다. 본원에서 참조된 다양한 OPCTL 신호들은 예를 들어, 제어 블록(530)에 구현된 결합 로직에 의해 연산 코드 그 자체 또는 그 연산 코드로부터 유도된 일부 다른 신호를 포함할 수 있다. 일부 실시예들에서, 제어 블록(530)은 몇몇 파이프라인 스테이지들에서 다수의 회로 블록들을 이용하여 구현될 수 있다. 주어진 연산 동안 상이한 블록들에 제공된 OPCTL 신호들은 동일한 신호 또는 상이한 신호들일 수 있다는 것을 이해해야 한다. 본 개시에 비추어 볼 때, 이 기술분야의 전문가들은 적절한 OPCTL 신호들을 구성할 수 있을 것이다.

[0082] 주어진 스테이지에 대한 회로 블록들은 상이한 처리 시간량을 요구할 수 있고, 특정 스테이지에서 요구되는 시간은 연산마다 다를 수 있다는 것에 주목해야 한다. 따라서, DFMA 유닛(320)은 또한 한 파이프라인에서 다음 파이프라인으로의 상이한 경로들에서의 데이터의 전달을 제어하기 위해 다양한 타이밍 및 동기화 회로들(도 5에 도시되지 않음)을 포함할 수 있다. 임의의 적절한 타이밍 회로(예를 들어, 래치들, 전송 게이트들 등)가 이용될 수 있다.

[0083] A. 피연산자 준비

[0084] 도 6은 본 발명의 실시예에 따른 피연산자 준비 블록(514)의 블록도이다. 피연산자 준비 블록(514)은 입력 피연산자들 A, B, C를 수신하고 가수 경로(516)에 가수 부분(Am, Bm, Cm)을, 지수 경로(518)에 지수 부분(Ae, Be, Ce)을, 그리고 부호 경로(520)에 부호 비트(As, Bs, Cs)를 제공한다.

[0085] 피연산자 A, B, C는 각각의 NaN 검출 블록들(612, 614, 616) 및 각각의 절대값/부정 블록들(618, 620, 622)에서 수신된다. 각각의 NaN 검출 블록(612, 614, 616)은 수신된 피연산자가 NaN(지수 비트들이 모두 1이고 유효숫자 비트들이 0이 아닌 수)인지를 결정하고 대응하는 제어 신호를 생성한다.

[0086] 절대값/부정 블록들(618, 620, 622)은 OPCTL 신호들(명시적으로 도시되지 않음)에 응답하여 피연산자들 중 부호 비트를 인버트(invert)하는 데 이용할 수 있다. 예를 들어, 도 4에 열거된 연산들 중 임의의 연산은 피연산자의 마이너스 또는 피연산자의 절대값이 이용될 것임을 특정할 수 있다. 블록들(618, 620, 622)은 부호 비트를 인버트하여 피연산자를 부정할 수 있거나, 또는 부호 비트를 마이너스가 아닌 상태(IEEE 754 포맷들을 위한 0)로 되게 할 수 있다. 입력 피연산자가 NaN인 경우, 적절한 절대값/부정 블록(618, 620, 622)은 또한 (예를 들어, 유효 숫자의 선두 비트를 1로 설정하는 것에 의해) NaN을 "콰이어트(quiet)"함으로서, 부호 비트를 유지한다. 절대값/부정 블록들(618, 620, 622)은 그들 각각의 출력들을 피연산자 선택 다중화기(632, 634, 636)로 제

공한다.

[0087] 배정도 산술의 경우, 절대값/부정 블록(618)에 의해 생성된 피연산자들 A, B, C이 직접 이용될 수 있다. 비교 연산들의 경우, A/B 비교 회로(624)는 피연산자 A와 B를 비교한다. 일 실시예에서, 절대값/부정 회로(620)는 피연산자 B를 부정하고 A/B 비교 회로(624)는 그것들이 고정 소수점 수라면 A와 -B의 합을 계산한다. 결과가 플러스이면, A는 B보다 크고, 결과가 마이너스이면, A는 B보다 작으며, 결과가 0이면, A는 B와 같다. A/B 비교 회로(624)는 또한 NaN 검출 회로들(612, 614)로부터 NaN 정보를 수신할 수 있다(이 경로들은 도 6에 명시적으로 도시되어 있지 않다). A 또는 B(또는 둘다)가 NaN이면, A와 B는 "순서없음(unordered)"이다. 결과 정보는 제어 로직(630)으로 제공된다. 제어 로직(630)은 결과 정보를 신호 R_TEST로서 출력부(522)에 제공하고, 또한 제어 신호들을 피연산자 선택 다중화기(632, 634, 636)로 제공한다.

[0088] 포맷 변환 피연산자들의 경우, 입력은 fp64 포맷으로 되어 있지 않을 수 있다. fp32 추출 회로(626)는 F2D 연산들 동안 활성이다. fp32 추출 회로(626)는 피연산자 A를 수신하고 비정상적인(non-normal) fp32 입력들에 대한 모든 테스트를 수행한다. fp32 추출 회로(626)는 또한 (예를 들어, 트레일링(trailing) 0들을 더하는 것에 의해) 23 비트로부터 52 비트로 수신된 피연산자의 유효 숫자 필드를 확장한다. fp32 추출 회로(626)는 8 비트 fp32 지수를 11 비트로 확장하고, 지수 바이어스를 127로부터 1023으로 증가시킨다(예를 들어, fp32 지수에 896을 더하는 것에 의해).

[0089] 부호없음/부호있음(unsigned/signed; U/S) 추출 회로(628)는 I2D 연산들 동안 활성이다. U/S 추출 회로(628)는 u32, s32, u64 또는 s64 포맷들 중 임의의 것으로 고정 소수점 피연산자 C를 수신하고 그것을 fp64로의 변환을 위해 준비한다. U/S 추출 회로(628)는 고정 소수점 피연산자를 1의 보수(또는 2의 보수) 형태로부터 부호-크기(sign-magnitude) 형태로 변환하고, 유효 숫자 필드에 피연산자를 정렬하기 위해 0들을 프리펜딩(prepending) 또는 첨가한다. U/S 추출 회로(628)는 그의 출력을 피연산자 선택 다중화기(636)로 제공하고, 또한 I2D 입력 신호로서 지수 경로(518)로 제공한다.

[0090] 피연산자 선택 다중화기(632, 634, 636)는 제어 로직(630)으로부터의 신호들에 응답하여 피연산자들 A, B, C를 선택한다. 피연산자 선택 다중화기(632)는 절대값/부정 회로(618)로부터의 피연산자 A와 상수값 0.0 및 1.0(fp64 포맷으로 표현됨) 사이에서 선택한다. DMUL 및 DFMA 연산들의 경우, 피연산자 A가 선택된다. DMIN(DMAX) 연산들의 경우, A<B (A>B)이면 피연산자 A가 선택되고 그렇지 않으면 1.0이 선택된다. DADD 및 I2D 연산들의 경우, 0.0이 선택된다.

[0091] 피연산자 선택 다중화기(634)는 절대값/부정 회로(620)로부터의 피연산자 B와 상수값 0.0 및 1.0(fp64 포맷으로 표현됨) 사이에서 선택한다. DMUL 및 DFMA 연산들의 경우, 피연산자 B가 선택된다. DMIN(DMAX) 연산들의 경우, B<A (B>A)이면 피연산자 B가 선택되고 그렇지 않으면 1.0이 선택된다. DADD 및 I2D 연산들의 경우, 0.0이 선택된다.

[0092] 피연산자 선택 다중화기(636)는 절대값/부정 회로(622)로부터의 피연산자 C, fp32 추출 회로(626)로부터의 추출된 fp32 값, U/S 추출 회로(628)로부터의 추출된 부호없음 또는 부호있음 정수 값, 및 상수값 0.0(fp64 포맷으로 표현됨) 중에서 선택한다. DADD 및 DFMA 연산들의 경우, 피연산자 C가 선택된다. DMUL 및 비교 연산들의 경우, 상수 0.0이 선택된다. F2D 연산들의 경우, fp32 추출 회로(626)로부터의 추출된 fp32 값이 선택되고, I2D 연산들의 경우, U/S 추출 회로(628)로부터의 추출된 u/s 값이 선택된다.

[0093] 선택 다중화기(632, 634, 636)에 의해 선택된 피연산자들 A, B, C이 특수수 검출 회로(special number detection circuit)(638, 640, 642)에 제공된다. fp64 피연산자들의 경우, 특수수 검출 회로(638, 640, 642)는 비정규화 수들, NaN, INF 및 0을 포함한 모든 특수수 상태들을 검출한다. F2D 연산들의 경우, 특수수 검출 회로(642)는 fp32 추출 회로(626)로부터 경로(644)를 통해 fp32 특수수 정보를 수신한다. 각각의 특수수 검출 회로(638, 640, 642)는 피연산자가 특수수인지, 그렇다면 어떤 타입인지를 나타내는 특수수 신호(special number signal; SPC)를 생성한다. 특수수 신호들(SPC)은 도 5에 도시된 바와 같이, 신호 경로(524)를 통해 출력부(522)에 제공된다. 일반적으로 통상적 설계로 된 특수수 검출 로직이 이용될 수 있다. 하나의 대안적인 실시예에서, (회로들(612, 614, 616)에 의해 수행되는) NaN 검출은 회로들(638, 640, 642)에서 반복되지 않으며, 그 대신, 각각의 특수수 검출 회로(638, 640, 642)는 NaN 검출 회로들(612, 614, 616) 중 대응하는 하나의 회로로부터 NaN 신호를 수신하고, 그 신호를 사용하여 피연산자가 NaN인지를 결정한다.

[0094] 임의의 특수수들이 검출되는지 여부에 관계없이, 특수수 검출 회로(638, 640, 642)는 피연산자들을 가수, 지수 및 부호 비트들로 분리한다. 특수수 검출 회로(638)는 피연산자 A의 가수 부분(Am)을 가수 경로(516)(도 5)로,

피연산자 A의 지수 부분(Ae)을 지수 경로(518)로, 그리고 부호 비트(As)를 부호 경로(520)로 제공한다. 특수수 검출 회로(640)는 피연산자 B의 가수 부분(Bm)을 가수 경로(516)에 제공하고, 피연산자 B의 지수 부분(Be)를 지수 경로(518)에 제공하고, 부호 비트(Bs)를 부호 경로(520)에 제공한다. 특수수 검출 회로(642)는 피연산자 C의 가수 부분(Cm) 및 지수 부분(Ce)을 지수 경로(518)에 제공하고 부호 비트(Cs)를 부호 경로(520)에 제공한다. 소정의 실시예에서, 특수수 검출 회로(638, 640, 642)는 (그 수가 비정규화 수인 경우를 제외하고) 가수들(Am, Bm, Cm)에 선두 1들을 첨가한다.

[0095] B. 지수 경로

[0096] 도 7은 본 발명의 실시예에 따른 지수 경로(518)의 블록도이다.

[0097] 지수 계산 회로(702)는 피연산자 준비 블록(514)(도 5)으로부터 지수 비트들 Ae, Be 및 Ce을 수신하고 DFMA 결과 A*B+C에 대한 블록 지수를 계산한다. 종래의 지수 계산 회로들이 사용될 수 있다. 일 실시예에서, 만약 모든 피연산자가 정상수(normal number)들이라면, 지수 계산 회로는 Ae와 Be를 더하고 fp64 지수 바이어스(1023)를 빼서 곱 A*B에 대한 지수를 결정하고, 그 후 곱 지수와 지수 Ce 중 큰 것을 DFMA 결과에 대한 블록 지수(BLE)로서 선택한다. 블록 지수 BLE는 다운스트림 최종 지수 계산 회로(704)에 제공된다. 만약 하나 이상의 피연산자가 (특수수 신호들 SPC 에 의해 표시되는 것처럼) 비정상적이라면, 적절한 로직이 블록 지수 BLE를 결정하는 데 사용될 수 있다. 다른 실시예들에서, 특수수들을 수반하는 연산들에 대한 지수 결정은 출력부(522)에서 아래에 기술되는 것과 같이 취급된다.

[0098] 더욱이, 지수 계산 블록(702)은 곱 Am*Bm과 Cm을 정렬하기 위해 피연산자 C의 가수가 좌측으로 또는 우측으로 어느 정도 이동되어야 효율적인지를 결정한다. 이러한 양은 Sh_C 제어 신호로서 이동 회로(706)에 제공된다. 이러한 제어 신호는 유리하게, Cm의 초과 패딩(extra padding)을 없이하여, 효율적인 좌측으로의 이동 또는 우측으로의 이동이 Cm의 우측 이동(right-shifting)에 의하여 달성되도록 한다.

[0099] 가수 Cm은 만약 C와 곱 A*B 사이에 상대적인 마이너스 부호가 있으면 조건적으로 (예를 들어 1의 보수 부인(1's complement negation)을 이용하여) Cm을 부인하는 부인 회로(708)에 제공된다. 상대적인 마이너스 부호는 아래에서 기술되는 것과 같이 부호 경로(520)에서 검출되고, 부호 제어 신호 SignCTL는 상대적인 마이너스 부호가 존재하는지의 여부를 표시한다. 부인 회로(708)의 출력(Cm 또는 ~Cm 중 어느 하나)은 이동 회로(706)에 제공된다.

[0100] 일 실시예에서, 이동 회로(706)는 54 비트의 가수 Cm을 157비트까지 우측으로 이동시킬 수 있는 217 비트 배럴 시프터(barrel shifter)이고; Sh_C 신호는 Cm이 어느 정도 우측 이동되는지를 결정한다. 가수 Cm은 유리하게, Cm이 필요한만큼 우측 이동되도록 조절되어 시프터에 입력된다. (가드 비트(guard bit)와 라운드 비트(round bit)를 더한) 53 비트의 가수 Cm이 106 비트의 곱 A*B(그 곱에 대한 가드 비트와 라운드 비트를 더하여)의 완전히 좌측 또는 완전히 우측으로 정렬되는 데 충분한 공간을 제공하도록 217 비트 크기가 선택되고, 그것은 217 비트 필드의 MSB의 우측에 대해 정렬된 55비트가 될 것이다. 배럴 시프터의 외부로 우측 이동된 임의의 비트는 폐기될 수 있다. 다른 실시예들에서, 배럴 시프터의 외부로 우측 이동된 모든 비트들이 "1"인지의 여부를 계속해서 추적하는 데 플래그 비트가 사용되고, 이러한 정보는 아래에서 기술되는 라운딩 연산들에서 사용될 수 있다.

[0101] 대안적인 실시예에서, 곱 Am*Bm과 Cm 사이에서 더 큰 피연산자를 선택하고, 그 후 더 작은 피연산자를 우측 이동시키는 데 종래의 스왑 다중화기들이 사용될 수 있다.

[0102] D2D 연산들에서, 가수 Cm이 D2D 로직 회로(710)에 제공될 수도 있다. D2D 로직 회로(710)는 가수 Cm, 지수 Ce 및 부호 Cs를 수신하고 정수-라운딩 규칙(integer-rounding rule)들을 적용한다. 일 실시예에서, D2D 로직 회로(710)는 지수 Ce에 기초하여 이진 소수점의 위치를 결정하고 그 후 OPCTL 신호에 기초하여, 선택된 라운딩 규칙을 적용한다(명백하게 도시 안됨). 라운딩 모드들을 구현하기 위해 종래의 로직이 사용될 수 있고, 라운딩 모드들의 임의의 조합 - 절단(truncation), 실링(ceiling), 플로어(floor) 및 니어리스트(nearest) 모드들을 포함하지만 그것으로 제한되지 않는 - 이 지원될 수 있다.

[0103] 선택 다중화기(712)는 이동된 가수 C_Shift, D2D 로직 회로의 출력 및 U/S 추출 회로(628)(도 6)로부터의 I2D 입력을 수신하고, OPCTL 신호에 기초하여, 이들 입력 중 하나의 입력을 가수 경로(516)에 공급될 정렬된 가수 C_align으로서 선택한다. 배정도 산술(double-precision arithmetic) 및 비교 연산들에 대하여, 피연산자 C_Shift가 선택된다. 포맷 변환들 D2D 또는 I2D를 위해 적절한 대안적인 입력이 선택된다.

[0104] 언더플로우 로직(713)이 fp64 및 fp32 결과들에서 잠재적인 언더플로우들을 검출하도록 구성된다. D2F 연산들

이 아닌 연산들에 대하여, 언더플로우 로직(713)은 11 비트 fp64 블록 지수 BLE가 0인지 또는 충분히 0에 가까워서 비정규화된 결과가 가능한지의 여부를 결정한다. 블록 지수에 기초하여, 언더플로우 로직(713)은 지수가 0에 도달하기 전에 가수가 좌측으로 이동될 수 있는 최대 비트수를 결정한다. 이 수는 가수 경로(516)에 8비트 언더플로우 신호 U_fp64로서 제공된다(도 8 참조). D2F 연산들에 대하여, 지수는 8비트 fp32 지수로서 취급되고, 언더플로우 로직(713)은 허용되는 최대 좌측 이동을 결정한다. 이 수는 가수 경로(516)에 8비트 언더플로우 신호 U_fp32로서 제공된다.

[0105] 지수 경로(518)는 최종 지수 계산 로직 회로(704)도 포함한다. 블록 지수 BLE는 뺄셈 회로(720)에 제공된다. 가수 경로(516)로부터의 블록 이동(BL_Sh) 신호도 뺄셈 회로(720)에 제공된다. 아래에서 기술되는 것과 같이, BL_Sh 신호는 곱 Am*Bm이 피연산자 C_align에 더해질 때 MSB들의 소거 효과를 반영한다. 뺄셈 회로(720)는 BLE에서 BL_Sh를 빼서 차 EDIF를 결정한다. 언더플로우/오버플로우 회로(722)는 뺄셈 결과 EDIF에서 언더플로우 또는 오버플로우를 검출한다. 1을 더하는 회로(Plus-1 circuit)(724)는 결과 EDIF에 1을 더하고, 언더플로우/오버플로우 상태에 기초하여, 다중화기(720)는 EDIF와 EDIF+1 신호들 사이에서 결과 지수 Re로서 선택한다. 결과 Re 및 언더플로우/오버플로우 신호(U/O)는 출력부(522)에 제공된다.

[0106] C. 가수 경로

[0107] 도 8은 본 발명의 실시예에 따른 가수 경로(516)의 블록도이다. 가수 경로(516)는 피연산자들 A, B 및 C의 가수들에 대한 곱셈 연산 및 덧셈 연산을 수행한다.

[0108] 53x53 곱셈기(802)는 피연산자 준비 블록(514)으로부터 가수들 Am 및 Bm을 수신하고 106 비트의 곱 Am*Bm을 계산한다. 곱은 168 비트 덧셈기(804)에 제공되고 덧셈기(804)는 정렬된 가수 C_align도 수신한다. 배럴 시프터(706)에 의해 사용된 217 비트 필드의 트레일링 비트(trailing bit)들이 폐기될 수 있거나, 또는 트레일링 비트들이 0이 아니었는지의 여부를 나타내는 플래그 비트들 또는 모든 1이 유지될 수 있다. 덧셈기(804)는 Sum과 ~Sum(합의 2의 보수) 출력을 생성한다. 다중화기(806)는 합의 MSB(부호 비트)에 기초하여 Sum 및 ~Sum 사이에서 선택한다. 선택된 합(S)은 제로 검출 회로(814)에 제공되고 좌측 이동 회로(816)에 제공된다. 제로 검출 회로(814)는 선택된 합 S가 0인지의 여부를 결정하고 대응하는 R_ZERO 신호를 부호 경로(520)에 제공한다.

[0109] 가수 경로(516)는 합 S를 정규화하기도 한다. 선두 0의 검출(leading zero detection)이 LZD 회로들(808, 810)을 이용하여 Sum과 ~Sum 둘 다에 대하여 병렬로 수행된다. 각 LZD 회로(808, 810)는 그것의 입력의 선두 0들의 개수를 나타내는 LZD 신호(Z1, Z2)를 생성한다. LZD 다중화기(812)는 합의 MSB(부호 비트)에 기초하여 적절한 LZD 신호(Z1 또는 Z2)를 선택한다. 만약 Sum이 다중화기(806)에 의해 선택되면 Z2가 선택되고, 만약 ~Sum이 다중화기(806)에 의해 선택되면 Z1이 선택된다. 선택된 LZD 신호는 지수 경로(518)에 블록 이동 신호 BL_Sh로서 제공되어, 위에서 기술된 것과 같이 결과 지수를 조정하는 데 사용된다.

[0110] 정규화 로직(818)은 합 S에 대한 정규화 이동을 결정하는 좌측 이동량 Lshift을 선택한다. 정상수(normal-number) 결과들에 대하여, 좌측 이동량은 유리하게, 선두 1을 가수 필드의 외부로 이동시켜 (가드 비트 및 라운딩 비트를 더하여) 52비트의 유효 숫자(significand)를 남기기에 충분하도록 크다. 그러나 소정의 경우들에서, 결과는 fp64 또는 fp32 비정규화 수로서 표현되어야 하는 언더플로우이다. 일 실시예에서, D2F가 아닌 연산들에 대하여, 출력 BL_Sh가 언더플로우 신호 U_fp64보다 커서 정규화 로직(818)이 U_fp64를 좌측 이동량으로서 선택하는 경우가 아니라면 정규화 로직(818)은 LZD 다중화기(812)에서 출력 BL_Sh를 선택한다. D2F 연산들에 대하여, 정규화 로직(818)은 fp32 언더플로우 신호 U_fp32를 사용하여 좌측 이동량 Lshift를 제한한다.

[0111] 좌측 이동 회로(816)는 합 S를 Lshift만큼 좌측으로 이동시킨다. 결과 Sn이 라운딩 로직(820), 1을 더하는 덧셈기(822) 및 가수 선택 다중화기(824)에 제공된다. 라운딩 로직(820)은 유리하게, IEEE 표준 산술에 대해 정의된 4개의 라운딩 모드(니어리스트, 플로어, 실링 및 절단)를 구현하고, 상이한 모드들은 상이한 결과들을 선택할 수도 있다. OPCTL 신호 또는 다른 제어 신호(도시 안됨)는 라운딩 모드들 중 하나의 모드를 특정하는 데 사용될 수 있다. 라운딩 모드 및 정규화된 합 Sn에 기초하여, 라운딩 로직(820)은 결과 Sn을 선택할지 1을 더하는 덧셈기(822)에 의해 계산된 Sn+1을 선택할지를 결정한다. 선택 다중화기(824)는 적절한 결과(Sn 또는 Sn+1)을 선택함으로써 라운딩 로직(820)으로부터의 제어 신호에 응답한다.

[0112] 다중화기(824)에 의해 선택된 결과는 포매팅 블록(826)으로 전해진다. 부동 소수점 출력(floating-point output)을 갖는 연산들에 대해 블록(826)은 출력부(522)에 가수 Rm를 제공한다. 합 S는 유리하게, 적어도 (정수 연산들을 지원하는) 64비트 폭이고, 외부에 발생한(extraneous) 비트들이 포매팅 블록(826)에 의하여 제거될 수도 있다. (정수 출력들을 갖는) D2I 연산들에 대해 포매팅 블록(826)은 결과를 LSB들을 포함하는 52 비트의

int_L 필드 및 MSB들을 포함하는 11 비트의 int_M 필드로 분리한다. Rm, int_L 및 int_M은 출력부(522)로 전달된다.

[0113] D. 부호 경로

[0114] 도 9는 본 발명의 실시예에 따른 부호 경로(520)의 블록도이다. 부호 경로(520)는 피연산자 준비 블록(514)로부터 피연산자들 As, Bs 및 Cs의 부호들을 수신한다(도 5). 부호 경로(520)는 피연산자 준비 블록(514)로부터의 특수수 신호들 SPC뿐만 아니라, 가수 경로(516)로부터의 제로 결과 신호 R_Zero, 진행 중인 연산의 유형을 표시하는 OPCTL 신호도 수신한다. 이러한 정보에 기초하여, 부호 경로(520)는 결과의 부호를 결정하고 부호 비트 Rs를 생성한다.

[0115] 더욱 구체적으로는, 부호 경로(520)는 곱/합 회로(902) 및 최종 부호 회로(904)를 포함한다. 곱/합 회로(902)는 피연산자 준비 블록(514)로부터의 피연산자들 A, B 및 C의 부호 비트들 As, Bs 및 Cs를 수신한다. 곱/합 회로(902)는 부호 비트들 As와 Bs 및 종래의 부호 로직 규칙들을 이용하여 곱 A*B의 부호(Sp)를 결정하고, 그 후 곱의 부호를 부호 비트 Cs와 비교하여 곱과 피연산자 C가 동일한 부호를 갖는지 반대 부호를 갖는지를 결정한다. 이러한 결정에 기초하여, 곱/합 회로(904)는 SignCTL 신호를 어서트하거나 디어서트하고, SignCTL 신호는 최종 부호 회로(904)와, 지수 경로(518)(도 7)의 부정 블록(708)으로 전달된다. 또한, 만약 곱과 피연산자 C가 동일한 부호를 갖는다면, 최종 결과도 그 부호를 가질 것이고; 만약 곱과 피연산자 C가 반대 부호를 갖는다면, 결과는 어느 것이 더 큰 지에 따라 결정된다.

[0116] 최종 부호 회로(904)는 최종 부호를 결정하는 데 필요한 모든 정보를 수신한다. 특히, 최종 부호 회로(904)는 부호 비트들 As, Bs 및 Cs 뿐만 아니라, 곱/합 회로(902)로부터의 곱의 부호 Sp 및 SignCTL 신호를 포함하는 부호 정보를 수신한다. 최종 부호 회로(904)는 가수 경로(516)로부터 제로 검출 신호 R_ZERO 및 피연산자 준비 블록(514)으로부터의 특수수 신호 SPC도 수신한다. 최종 부호 회로(904)는 합이 플러스였는지 마이너스였는지를 나타내는, 가수 경로(516)의 덧셈기(804)로부터의 합의 MSB도 수신한다.

[0117] 이러한 정보에 기초하여, 최종 부호 회로(904)는 종래의 부호 로직을 채용하여 결과에 대한 부호 비트 Rs를 결정할 수 있다. 예를 들어, DFMA 연산들에 대해, 부호 비트들 Sp 및 Cs가 동일하다면, 결과도 그 부호를 가질 것이다. 만약, Sp와 Cs가 반대 부호라면, 가수 경로(516)의 덧셈기(804)는 (Am*Bm)-C_align을 계산한다. 만약 Am*Bm이 C_align보다 크다면, 덧셈기(804)는 플러스의 결과 Sum을 계산할 것이고 곱의 부호 Sp가 선택되어야 하고; 만약 Am*Bm이 C_align보다 작다면, 덧셈기(804)는 마이너스의 결과 Sum을 계산할 것이고 부호 Cs가 선택되어야 한다. 덧셈기(804)의 Sum 출력의 MSB는 결과의 부호를 표시하고 선택을 구동하는 데 사용될 수 있다. 만약 결과 Sum이 0이면, R_ZERO 신호가 어서트되고, 최종 부호 회로(904)가 적절한 때 (fp64 포맷에서 0은 플러스 또는 마이너스 중 어느 하나일 수 있다) 어느 하나의 부호를 선택할 수 있다. 다른 연산들에 대해, 최종 부호 회로(904)는 하나의 피연산자 또는 또다른 피연산자의 부호를 최종 부호로서 통과시킬 수 있다.

[0118] E. 출력부

[0119] 도 10은 본 발명의 실시예에 따른 DFMA 유닛(320)에 대한 출력부(522)의 블록도이다.

[0120] 출력 다중화기 제어 로직(1002)은 지수 경로(518)(도 7)로부터의 언더플로우/오버플로우(U/O) 신호, 피연산자 준비 블록(514)(도 6)으로부터의 SPC 신호들 및 R_test, 및 진행 중인 연산의 유형을 표시하는 OPCTL 신호를 수신한다. 이러한 정보에 기초하여, 출력 다중화기 제어 로직(1002)은 유효 숫자 선택 다중화기(1004) 및 지수 선택 다중화기(1006)에 대한 선택 제어 신호들을 생성한다. 출력 다중화기 제어 로직(1002)은 예를 들면, 오버플로우 또는 언더플로우, NaN 또는 다른 상태들을 표시할 수 있는 컨디션 코드 신호(condition code signal, COND)들도 생성한다. 소정의 실시예에서, 컨디션 코드는 DSET 연산들 동안 불린 결과(Boolean result)를 신호하는 데 사용될 수도 있다.

[0121] 유효 숫자 선택 다중화기(1004)는 다수의 특수값 뿐만 아니라 가수 경로(516)로부터의 결과 유효 숫자 Rm 및 (D2I 연산들 동안 사용되는) 최대 52 비트의 정수 출력을 수신한다. 일 실시예에서 특수값들은: (D2I 연산들에 대해 최대의 64 비트 정수를 표현하는 데 사용되는) 52 비트 필드의 1들; (결과가 0.0 또는 1.0인 경우에 사용되는) 52 비트 필드의 0들; (D2I 연산들에 대해 최소의 32 비트 정수를 표현하는 데 사용되는) 52 비트 필드 0x0_0000_8000_0000; (내부적으로 생성된 콰이어트(quiet) NaN을 표현하는 데 사용되는) 선두 1을 갖는 52 비트 필드; 예를 들면(D2I 연산들에 대해 최대의 32 비트 정수를 표현하는 데 사용되는) 0x7fff_ffff_ffff_ffff인 max_int32 값; (피연산자 준비 블록(514)으로부터의 NaN인 입력 피연산자를 통과시키는 데 사용되는) 콰이어트된(quieted) NaN 값; 및 (언더플로우들에 대해 사용되는) 예를 들면, 마지막 비트 위치의 1인 min_denorm 값을

포함한다. 연산에 따라 그리고 피연산자들 중 임의의 것 또는 결과가 특수수인지에 따라 입력들 중 임의의 것이 선택될 수 있다.

[0122] 지수 선택 다중화기(1006)는 다수의 특수값 뿐만 아니라, 지수 경로(518)로부터 결과 지수 Re를 수신하고 최대 11 정수 비트(정수 포맷 출력들에 대한 MSB)들을 수신한다. 일 실시예에서 특수값들은: 0x3ff(fp64의 1.0에 대한 지수); 0x000(비정규화 수들 및 0.0에 대한 지수), 0x7fe(정상수들에 대한 최대의 fp64 지수) 및 0x7ff(fp64 NaN 또는 INF 결과들에 대한)을 포함한다. 연산에 따라 그리고 피연사자들 중 임의의 것 또는 결과가 특수수인지에 따라 입력들 중 임의의 것이 선택될 수 있다.

[0123] 연결 블록(concatenation block, 1008)은 부호 비트 Rs를 수신하고, 다중화기(1004)에 의해 선택된 유효 숫자 비트들 및 다중화기(1006)에 의해 선택된 지수 비트들을 수신한다. 연결 블록(1008)은 결과를 (예를 들면, IEEE 754 표준에 따라 부호, 지수, 유효 숫자 순서로) 포매팅하고 64 비트 출력 신호 OUT을 제공한다.

[0124] F. 피연산자 우회 또는 통과 경로

[0125] 소정의 실시예에서 DFMA 유닛(320)은 피연산자들이 수정되지 않은 채로 다양한 회로 블럭을 통하여 전파되는 것을 허용하는 우회 또는 통과 경로들을 제공한다. 예를 들면, 소정의 연산들 동안, 곱셈기(802)는 입력(예를 들면, Am)을 1.0으로 곱하여 입력 Am을 사실상 통과시킨다. Am을 1.0으로 곱하기보다는, 오히려 입력 Am에 대한 우회 경로가 곱셈기(802)의 주위에 제공될 수 있다. 우회 경로는 유리하게, 곱셈기(802)와 동일한 개수의 클럭 사이클을 소비하여, Am이 정확한 시간에 덧셈기(804)로의 입력에 도착한다. 그러나 곱셈기(802)는 우회될 때, 비활성이나 저전력 상태로 설정될 수 있고, 그에 의해 회로 면적이 조금 증가하는 대신 전력 소비를 감소시킬 수 있다. 유사하게, 소정의 연산들 동안, 덧셈기(804)는 입력(예를 들면, C_align)에 0을 더하는 데 사용되어, 입력 C_align을 사실상 통과시킨다. C_align에 0을 더하기 보다는, 오히려 입력 C_align에 대한 우회 경로가 덧셈기(804) 주위에 제공될 수 있고, 특히 덧셈기(804)의 Sum과 ~Sum 출력 중 어느 것이 다중화기(806)에 의해 선택되어야 하는지가 미리 알려지는 연산들에 대해; 입력 C_align이 Sum 과 ~Sum 경로 중 정확한 경로 상으로 우회될 수 있다. 다시, 우회 경로는 유리하게, 덧셈기(804)와 동일한 개수의 클럭 사이클을 소비하여, 타이밍에는 영향이 없지만, 덧셈기(804)를 우회하는 연산에 대하여 덧셈기(804)가 비활성 또는 저전력 상태로 될 수 있기 때문에 전력 소비가 감소될 수 있다.

[0126] 그리하여, 섹션 Ⅲ(아래)에서의 연산적인 기술들은 특정한 회로 블록에 대하여 우회되거나 통과되는 다양한 피연산자들을 참조할 수 있으며; 사이에 끼인 임의의 회로 블록에 의하여 피연산자에게 영향을 주지 않는 연산(예를 들면, 0을 더하거나 1.0을 곱하기)을 수행하여 블록의 입력이 출력으로서 통과되도록 제어하거나 우회 경로를 이용함으로써 중 어느 하나에 의해 달성될 수 있다는 것이 이해되어야 한다. 또한, 소정의 회로 블록들 주위의 이하의 우회 또는 통과 경로는 반드시 후속의 회로 블럭들에서 우회 경로를 계속해서 따를 필요는 없다. 게다가, 하나의 회로 블록에서 수정된 값이 후속 회로 블록 주위의 우회로를 따를 수 있다. 특정한 회로 블록이 연산 동안 우회되는 경우, 회로 블록은 비활성화 상태로 설정되어 전력 소비를 감소시키거나 예를 들면, 선택 다중화기들 또는 다른 회로 요소들의 사용을 통하여 출력이 무시되면서 정상적으로 동작될 수 있다.

[0127] 본 명세서에서 기술된 DFMA 유닛은 예시적이며 변형들 및 변경들이 가능하다는 것이 이해될 것이다. 본 명세서에 기술된 화로 블록들 중 다수는 종래의 기능들을 제공하고 본 기술분야에서 알려진 기술들을 사용하여 구현될 수 있기 때문에; 따라서, 이들 블록들의 상세한 설명은 생략되었다. 연산 회로의 블록들로의 분할은 변경될 수 있고, 블록들은 조합되거나 변화될 수 있다. 게다가, 파이프라인 스테이지들의 수 및 특정한 회로 블록들 또는 연산들의 특정한 파이프라인 스테이지들로의 할당은 수정되거나 변화될 수 있다. 특정한 구현들에 대한 회로 블록들의 선택 및 배치는 지원될 연산들의 세트에 따라 결정될 것이며, 본 기술분야의 통상의 기술자들은 본 명세서에서 기술된 블록이 가능한 연산들의 조합 모두에 대하여 모두 필요하지는 않다는 것을 인식할 것이다.

[0128] Ⅲ. DFMA 유닛 연산들

[0129] DFMA 유닛(320)은 유리하게, 면적 효율적인 방식(area-efficient manner)으로 도 4에 열거된 모든 연산들을 지원하도록 전술된 회로 블록들을 레버리지한다. 따라서, 적어도 소정의 태양에서 DFMA 유닛(320)의 연산은 어떤 연산이 실행되는 지에 의해 결정된다. 다음의 섹션들은 도 4에 열거된 연산들 각각을 수행하기 위한 DFMA 유닛(320)의 사용을 기술한다.

[0130] (예를 들면, 오버플로우 또는 언더플로우 상태들을 포함하는) 부동 소수점 예외들이 추가의 처리 사이클들을 필요로 하지 않고 DFMA 유닛(320) 내에서 처리된다는 것을 주의해야 한다. 예를 들어, 입력 피연산자가 NaN이거

나 다른 특수수인 경우 연산들은 도 5의 피연산자 준비 블록(514)에서 검출되고, 적절한 특수수 출력이 출력부(522)에서 선택된다. NaN, 언더플로우, 오버플로우 또는 다른 특수수들이 연산 동안에 발생하는 경우, 상태가 검출되고 적절한 특수수 출력이 출력부(522)에서 선택된다.

[0131] A. 결합형 곱셈 덧셈(DFMA)

[0132] DFMA 연산들에 대하여, DFMA 유닛(320)은 fp64 포맷인 피연산자들 A0, B0 및 C0와, DFMA 연산이 수행되어야 한다는 것을 표시하는 연산 코드(opcode)를 수신한다. NaN 회로들(612, 614, 616)은 임의의 선택된 피연산자가 NaN인지의 여부를 결정한다. 절대값/부정 회로들(618, 620, 622)는 각 피연산자에 대하여 적절할 때 부호 비트를 부정한다 (또는 부정하지 않는다). 피연산자 선택 다중화기들(632, 634 및 636)은 절대값/부정 회로들(618, 620, 622)의 개별적인 출력을 선택하고 이들 출력을 특수수 검출 회로들(638, 640, 642)에 제공한다. 특수수 검출 회로들(638, 640 및 642)은 각 피연산자가 특수수인지의 여부를 결정하고 적절한 특수수 SPC 신호들을 경로(524) 상에 생성한다. 특수수 검출 회로들(638, 640 및 642)는 (정상수들에 대해서는 첨가된 선두 1과 함께, 그리고 비정규화 수들에 대해서는 선두 0과 함께) 가수들 Am, Bm 및 Cm을 가수 경로(516)에 제공하고, 지수들 Ae, Be 및 Ce를 지수 경로(518)에 제공하고, 부호 비트들 As, Bs 및 Cs를 부호 경로(520)에 제공한다.

[0133] A/B 비교 회로(624), fp32 추출 회로(626) 및 U/S 정수 추출 회로(628)은 DFMA 연산들을 위해 사용되지 않으며, 이들 회로는 원한다면 비활성화 또는 저전력 상태로 설정될 수 있다.

[0134] 부호 경로(520)에서, 곱/합 회로(902)는 부호 비트들 As 및 Bs로부터 곱 A*B가 플러스인지 마이너스인지의 여부를 결정하고 곱 Sp의 부호를 부호 비트 Cs와 비교한다. 만약 곱과 Cs가 반대 부호를 갖는다면, SignCTL 신호가 어서트되어 반대 부호를 표시하고; 만약 곱과 Cs가 동일한 부호를 갖는다면 SignCTL 신호는 디어서트된다.

[0135] 지수 경로(518)(도 7)에서, 지수 계산 블록(702)은 지수 Ae, Be 및 Ce를 수신한다. 지수 계산 블록(702)은 지수 Ae와 Be를 더하여 곱 A*B에 대한 블록 지수를 결정하고, 그 후 곱 블록 지수와 지수 Ce 중에서 더 큰 것을 결과 블록 지수 BLE로서 선택한다. 지수 계산 블록(702)는 또한 곱 블록 지수와 지수 Ce 중에서 더 작은 것을 둘 중에서 더 큰 것에서 빼서, 대응하는 이동 제어 신호 Sh_C를 생성한다. 언더플로우 로직(713)은 블록 지수 BLE가 언더플로우 또는 잠재적인 언더플로우에 대응하는지의 여부를 검출하고 언더플로우 신호 U_fp64를 생성한다. (U_fp32 신호는 DFMA 연산들 동안 사용되지 않는다.)

[0136] 부정 블록(708)은 피연산자 준비 블록(514)으로부터 가수 Cm을 수신하고 부호 경로(520)로부터 SignCTL 신호를 수신한다. 만약 SignCTL 신호가 어서트되면, 부정 블록(708)은 가수 Cm을 인버트시켜 상대적인 마이너스 부호를 제거하고 인버트된 Cm을 이동 회로(706)에 제공한다. 그렇지 않으면, 부정 블록(708)은 이동 회로(706)에 수정 없이 Cm을 제공한다.

[0137] 이동 회로(706)는 부정 블록(708)에 의해 제공된 가수 Cm을 이동 제어 신호 Sh_C에 대응하는 양만큼 우측으로 이동시키고 이동된 C_Shift 가수를 선택 다중화기(712)에 제공한다. 선택 다중화기(712)는 이동된 가수 C_Shift를 선택하고 그 이동된 가수를 피연산자 C_align으로서 가수 경로(516)에 제공한다.

[0138] 가수 경로(516)(도 8)에서, 곱셈기(802)는 106 비트 곱 Am*Bm을 계산하고 그 곱을 168비트 덧셈기(804)에 제공한다. 곱셈기(802)의 연산은 지수 계산 블록(702)의 연산과 병렬로 일어날 수 있다.

[0139] 덧셈기(804)는 지수 경로(518)의 선택 다중화기(712)로부터 피연산자 C_align을 수신하고 입력들 Am*Bm과 C_align을 더하여 Sum과 ~Sum을 결정한다. Sum의 MSB에 기초하여, 다중화기(806)는 출력들 중 하나의 출력을 최종합으로 선택한다. 만약 Sum이 플러스라면(MSB 0) 선택되고, Sum이 마이너스라면(MSB 1) ~Sum이 선택된다. LZD 회로들(808 및 810)은 ~Sum과 Sum 각각에 있는 선두 0들의 개수를 결정한다; 다중화기(812)는 LZD 출력들 중 하나를 선두 0들의 개수로서 선택하고 선두 0 신호 BL_Sh를 지수 경로(518) 및 정규화 로직(818)에 제공한다.

[0140] 다중화기(806)에 의하여 선택된 최종합 S도 0 검출 회로(814)에 제공된다. 만약 최종합이 0이면, 0 검출 회로(814)가 R_ZERO 신호를 부호 경로(520)에 어서트할 것이고; 그렇지 않으면 R_ZERO 신호는 어서트되지 않는다.

[0141] 정규화 로직(818)은 U_fp64 신호가 언더플로우를 표시하지 않는 한, 선두 0 신호를 정규화 신호 Lshift로서 선택하고, 그 경우에, 단지 1의 지수에 대응하는 점으로 가수가 이동되어 결과가 정규화되지 않은(non-normalized) 형태로 표현된다. 이동 회로(816)는 선택된 합 S를 Lshift 신호에 응답하여 좌측으로 이동시켜 정규화된 합 Sn을 생산한다. 1을 더하는 덧셈기(822)는 1을 정규화된 합 Sn에 더한다. 라운딩 로직(820)은 (OPCTL 신호에 의하여 특정되는) 라운딩 모드 및 (경로(821) 상의) 정규화된 합 Sn의 LSB들을 사용하여 정규화

된 합이 라운딩 업되어야 하는지의 여부를 결정한다. 만약 그렇다면, 라운딩 로직(820)은 선택 다중화기(824)를 제어하여 덧셈기(822)로부터 출력 Sn+1을 선택한다; 그렇지 않으면; 선택 다중화기(824)는 정규화된 합 Sn을 선택한다. 선택 다중화기(824)는 선택된 결과 Rm을 출력부(522)에 제공한다. 소정의 실시예에서, 선택 다중화기(824)는 결과 가수로부터 선두 비트(정상수들에 대하여 1)를 드롭(drop)한다.

[0142] 라운딩 연산들과 병렬로, 지수 경로(518)(도 7)는 결과 지수 Re를 계산한다. 특히, 뺄셈 블록(720)은 지수 계산 블록(702)으로부터 블록 지수 BLE를 수신하고, 가수 경로(516)로부터 블록 이동 신호 BL_Sh를 수신한다. 뺄셈 블록(720)은 그것의 두 입력들을 뺄셈하고 결과 EDIF를 언더플로우/오버플로우 로직(722), 1을 더하는 덧셈기(724) 및 선택 다중화기(726)에 제공한다. 언더플로우/오버플로우 로직(722)은 결과의 MSB들을 사용하여 언더플로우가 발생했는지 또는 오버플로우가 발생했는지를 결정하고 언더플로우 또는 오버플로우의 존재 혹은 부재를 반영하는 U/O 신호들을 생성한다. U/O 신호들에 기초하여, 선택 다중화기(726)는 뺄셈 결과 EDIF와 1을 더하는 덧셈기(724)의 출력 사이에서 선택한다. 선택된 값은 결과 지수 Re로서 U/O 신호들과 함께 출력부(522)에 제공된다.

[0143] 라운딩 연산들과 병렬로, 부호 경로(520)(도 9)의 최종 부호 회로(904)는 곱/합 회로(902)에 의하여 결정된 부호, 가수 경로(516)로부터 수신된 합의 MSB와 R_ZERO 신호, 및 피연산자 준비 블록(514)으로부터 수신된 특수수 SPC 신호들에 기초하여 최종 부호 Rs를 결정한다.

[0144] 출력부(522)(도 10)는 피연산자 준비 블록(514)로부터의 특수수 SPC 신호 및 지수 경로(518)로부터의 U/O 신호들뿐만 아니라, 가수 경로(516)로부터 결과 가수 Rm을 수신하고, 지수 경로(518)로부터 결과 지수 Re를 수신하고, 부호 경로(520)로부터 결과 부호 Rs를 수신한다. SPC 및 U/O 신호들에 기초하여, 출력 다중화기 제어 로직(1002)은 유효 숫자 다중화기(1004)에 대한 제어 신호 및 지수 다중화기(1006)에 대한 제어 신호를 생성한다. 출력 다중화기 제어 로직(1002)도 예를 들어, 결과가 오버플로우인지, 언더플로우인지 또는 NaN인지를 표시하는 다양한 상태 코드들(COND)를 생성한다.

[0145] 유효 숫자 다중화기(1004)는 정상수들과 비정규화 수들에 대하여 유효 숫자 Rm을 선택한다. 언더플로우들에 대하여, 라운딩 모드에 따라 0 또는 min_denorm 유효 숫자가 선택된다. 오버플로우들(INF)에 대하여, 유효 숫자 0x0_0000_0000_0000가 선택된다. 임의의 입력 피연산자가 NaN인 경우, 콰이어트된 NaN 유효 숫자가 선택된다. 만약, NaN이 연산 동안 생성되었다면, 내부(콰이어트) NaN 가수 0x8_0000_0000이 선택된다.

[0146] 지수 다중화기(1006)는 정상수들에 대하여 결과 지수 Re를 선택한다. 비정규화 수들 및 언더플로우들에 대하여, 지수 0x000이 선택된다. INF 또는 NaN에 대하여, 최대 지수 0x7ff가 선택된다.

[0147] 연결 블록(1008)은 선택된 유효 숫자와 지수 및 부호 Rs를 수신하고 최종 fp64 결과 OUT을 생성한다. 상태 코드들이 원하는 대로 설정될 수 있다.

[0148] DFMA 유닛(320)이 오버플로우들 또는 언더플로우들에 관계없이 모든 DFMA 연산들을 동일한 개수의 사이클들에서 완료한다는 것을 주의해야 한다. DFMA 유닛(320)은 또한 IEEE 754 표준에 따르는 부동 소수점 산술에 대하여 기대되는 기본 오버플로우/언더플로우 동작을 구현한다: 적절한 결과 OUT이 반환되고, (상태 코드 COND의) 상태 플래그가 오버플로우/언더플로우 상태를 표시하도록 설정된다. 소정의 실시예에서, 사용자 정의된 트랩들이 이러한 상태들을 처리하기 위해서 구현될 수 있다; 상태 코드 COND는 트랩이 발생해야 될지의 여부를 결정하는 데 사용될 수 있다.

[0149] B.곱셈

[0150] 곱셈(DMUL)은 피연산자 C가 0으로 설정된, 전술된 DFMA 연산과 동일하게 구현될 수 있고; 그러면 DFMA 유닛(320)은 A*B+0.0을 계산한다. 일 실시예에서, 선택 다중화기(636)(도 6)는 연산 코드가 DMUL 연산을 표시할 때 입력 피연산자 C를 fp64 0 값으로 대체하는 데 사용될 수 있다.

[0151] C. 덧셈

[0152] 덧셈(DADD)은 피연산자 B가 1.0으로 설정된, 전술된 DFMA 연산과 동일하게 구현될 수 있고, 그 후 DFMA 유닛(320)은 A*1.0+C를 계산한다. 일 실시예에서, 선택 다중화기(634)(도 6)는 연산 코드가 DADD 연산을 표시할 때, 입력 피연산자 B를 fp64 1.0값으로 대체하는 데 사용될 수 있다.

[0153] D. DMAX 및 DMIN

[0154] DMAX 또는 DMIN 연산들에 대해, 피연산자 준비 블록(514)(도 6)은 피연산자들 A 및 B를 수신한다. NaN 회로들

(612 및 614)은 선택된 피연산자들 중 하나 또는 둘 다 NaN인지 여부를 결정한다. 절대값/부정(negation) 회로들(618, 620)은 적절하게 부호 비트를 부정한다 (또는 부정하지 않는다).

[0155] A/B 비교 회로(624)는 절대값/부정 회로들(618, 620)로부터 피연산자들 A 및 B를 수신하여 예를 들어, 피연산자들이 정수들인 것처럼 A에서 B를 뺌으로써 비교를 수행한다. 뺄셈에 기초하여, A/B 비교 회로(624)는, A가 B보다 큰지, 작은지 또는 같은지를 나타내는 COMP 신호들을 생성한다. COMP 신호들은 제어 로직(630)에 제공되고, 제어 로직은 대응하는 R_Test 신호들을 생성하고, 또한 선택 다중화기들(632, 634 및 636)에 대한 선택 신호들을 생성한다.

[0156] 특히, DMAX 연산들에 대해, 피연산자 A 다중화기(632)는 A가 B보다 크면 피연산자 A를 선택하고, A가 B보다 작으면 피연산자 1.0을 선택하는 한편, 피연산자 B 다중화기(634)는 B가 A보다 크면 피연산자 B를 선택하고, B가 A보다 작으면 피연산자 1.0을 선택한다. DMIN 연산들에 대해, 피연산자 A 다중화기(632)는 A가 B보다 작으면 피연산자 A를 선택하고, A가 B보다 크면 피연산자 1.0을 선택하는 한편, 피연산자 B 다중화기(634)는 B가 A보다 작으면 피연산자 B를 선택하고, B가 A보다 크면 피연산자 1.0을 선택한다. DMAX 및 DMIN 둘 다에 대해, A=B인 특별한 경우는 다중화기(634)가 피연산자 1.0을 선택하면서 피연산자 A를 선택하도록 다중화기(632)를 제어하거나 또는 다중화기(634)가 피연산자 B를 선택하면서 피연산자 1.0을 선택하도록 다중화기(632)를 제어함으로써 처리될 수 있다. 임의의 경우, 피연산자 C 다중화기(636)는 피연산자 0.0을 선택하도록 유리하게 연산된다.

[0157] 특수수(special number) 검출 회로들(638, 640 및 642)은 피연산자들이 특수수들인지를 결정하고 경로(524)에 대해 적합한 특수수 SPC 신호들을 생성한다. 특수수 검출 회로들(638, 640 및 642)은 가수 경로(516)에 (정상수들에 부가된 1을 유도하고 비정규화 수들에 대해 0을 유도하는) 가수(mantissas) Am, Bm 및 Cm을, 지수 경로(518)에 지수들 Ae, Be 및 Ce을, 및 부호 경로(520)에 부호 비트들 As, Bs 및 Cs를 제공한다.

[0158] Fp32 추출 회로(626) 및 부호없음/부호있음(unsigned/signed) 정수 추출 회로(628)는 DMAX 또는 DMIN 연산들에 대해 사용될 수 없고, 이러한 회로들은 원한다면 비활성 또는 저전력 상태로 설정될 수 있다.

[0159] 가수 경로(516), 지수 경로(518) 및 부호 경로(520)는 DFMA 연산들에 대해 전술된 바와 같이 연산한다. DMAX 연산들에 대해, 가수 경로(516), 지수 경로(518) 및 부호 경로(520)는 max(A,B)*1.0+0.0을 계산하고, DMIN 연산들에 대해, 가수 경로(516), 지수 경로(518) 및 부호 경로(520)는 min(A,B)*1.0+0.0을 계산한다. 따라서, 정상수들에 대한 Rm, Re 및 Rs는 원하는 결과의 가수, 지수 및 부호에 대응한다.

[0160] 출력부(522)(도 10)는 특수수들을 처리한다. 구체적으로, DMAX 및 DMIN 연산들의 결과들은 NaN 피연산자들에 대해 정의되지 않고, 그 결과는 NaN 값으로 설정될 수 있다. 출력 다중화기 제어 로직(1002)은 결과가 NaN이어야 하는지를 결정하기 위해 특수수 SPC 신호들을 사용하고, 만일 그렇다면, 유효 숫자(significand) 다중화기(1004)는 지수 다중화기가 0x7ff를 선택하는 동안 콰이어트된(quieted) NaN 입력을 선택한다. 그렇지 않으면, 결과 Rm 및 Re가 선택된다. 상태 코드들은 원하는 대로 설정될 수 있다.

[0161] 대안적인 실시예에서, 가수 경로(516), 지수 경로(518) 및 부호 경로(520)의 일부 또는 모든 컴포넌트들은 우회될 수 있고, 우회된 컴포넌트들은 저전력 상태로 놓여질 수 있다. 우회 경로는, 가장 긴 가수 경로(516), 지수 경로(518) 및 부호 경로(520)와 동일한 수의 파이프라인 스테이지들을 점유하기 위해, 다양한 지연 회로들(래치들 등)을 포함할 수 있다. 이것은, DFMA 유닛(320)에서의 모든 연산들이 완료를 위해 동일한 수의 사이클들을 요구하도록 보증하며, 그것은 명령 발행 로직(instruction issue logic)을 단순화한다.

[0162] E. DSET

[0163] DMAX 및 DMIN과 같이, DSET 연산들은 피연산자 준비 블록(514)(도 6)의 A/B 비교 회로(624)를 사용한다. DMAX 및 DMIN과 다르게, DSET은 입력 피연산자들 중 하나를 반환하지 않고, 대신 테스트 조건이 만족되는지를 나타내는 불린 값(Boolean value)을 반환한다.

[0164] DSET 연산들에 대해, 피연산자 준비 블록(514)(도 6)은 피연산자 A 및 B를 수신한다. NaN 회로들(612 및 614)은 선택된 피연산자들 중 하나 또는 둘다 NaN인지 결정한다. 절대값/부정 회로들(618, 620)은, 적합하다면 부호 비트를 부정한다.

[0165] A/B 비교 회로(624)는 절대값/부정 회로들(618, 620)로부터 피연산자들 A 및 B를 수신하고, 예를 들어, 피연산자들이 정수인 것처럼 A에서 B를 뺌으로써 비교를 수행하고 그들 각각의 부호 비트들을 고려한다. 뺄셈에 기초하여, A/B 비교 회로(624)는, A가 B보다 큰지, 작은지 또는 같은지를 나타내는 COMP 신호들을 생성한다. COMP 신호들은 제어 로직(630)에 제공되고, 제어 로직은 대응하는 R_Test 신호들을 생성하고, 또한 A-다중화기(632),

B-다중화기(634) 및 C-다중화기(636)에 대한 선택 신호들을 생성한다. 일 실시예에서, DSET 연산의 결과가 불린이므로, 세 다중화기들(632, 634 및 636) 모두는 0 피연산자를 선택한다. 다른 실시예에서, 다중화기들(632, 634)은 피연산자들 A 및 B를 선택하고, 특수수 검출 회로들(638 및 640)은 피연산자들이 특수수들인지를 결정하고 경로(524)에 대해 적합한 특수수 SPC 신호들을 생성한다.

[0166] Fp32 추출 회로(626) 및 부호없음/부호있음 정수 추출 회로(628)는 DSET 연산들에 대해 사용되지 않고, 이러한 회로들은 원한다면, 비활성화 또는 저전력 상태로 설정될 수 있다.

[0167] 가수 경로(516), 지수 경로(518) 및 부호 경로(520)는 DFMA 연산들에 대해 상술된 바와 같이 연산하거나 부분적으로 또는 전체적으로 우회될 수 있다. 임의의 우회된 컴포넌트들은 저전력 상태로 놓여질 수 있다. 상기 언급된 바와 같이, 우회 경로는, 경로가 가장 긴 가수 경로(516), 지수 경로(518) 및 부호 경로(520)와 동일한 수의 파이프라인 스테이지들을 점유하도록, 다양한 지연 회로들(래치들 등)을 포함할 수 있다. 이것은, DFMA 유닛(320)에서의 모든 연산들이 완료를 위해 동일한 수의 사이클들을 요구하도록 보증하며, 그것은 명령 발행 로직을 단순화한다.

[0168] 출력부(522)(도 10)는 특수수들을 처리한다. 구체적으로, IEEE 754 표준 하에서, A 및 B는 A 또는 B(또는 둘 다)가 NaN이면 정렬되지 않는다(unorder). 출력 다중화기 제어 로직(1002)은 A가 B보다 큰지, 작은지 또는 같은지를 나타내는 R_Test 신호들을 수신하고, 특수수 SPC 신호들은 A 또는 B가 NaN인지를 나타내고, OPCTL 신호는 요청된 특정 테스트 연산을 나타낸다. 출력 다중화기 제어 로직(1002)은 요청된 테스트가 만족되는지 결정하기 위해 R_Test 및 SPC 신호들을 사용한다. 일 실시예에서, DSET 연산의 결과는 상태 코드로서 제공되고, 결과 OUT은 무시된다; 이 경우, 출력 다중화기 제어 로직(1002)은 그 결과를 나타내도록 상태 코드 COND를 설정하고, 출력 OUT에 대한 유효 숫자 및 지수를 임의로 선택할 수 있다. 다른 실시예에서, 출력 OUT은 테스트 결과를 반영하도록 설정될 수 있고, 이 경우, 출력 다중화기 제어 로직(1002)은 테스트가 만족되면 논리 TRUE에 대응하는 64비트 값을, 또는 테스트가 만족되지 않으면 논리 FALSE에 대응하는 64비트 값을 선택하도록, 유효 숫자 다중화기(1004) 및 지수 다중화기(1006)를 연산한다.

[0169] F. 포맷 변환

[0170] 일부 실시예들에서, DFMA 유닛(320)은 배정도 포맷으로/로부터 포맷 변환 연산들을 지원하기도 한다.

[0171] 1. fp32 내지 fp64(F2D)

[0172] F2D 연산들에 대해, fp32 입력 피연산자 A는 대응하는 fp64 수로 변환된다. 특수수 입력들은 예를 들어, fp32 INF 또는 NaN이 fp64 INF 또는 NaN으로 변환되게 적절히 처리된다. 모든 fp32 비정규화 수들은 fp64 정상수들로 변환될 수 있다.

[0173] 피연산자 준비 블록(514)(도 6)은 fp32 피연산자 A를 수신한다. 절대값/부정 회로(618)는 수정 없이 피연산자 A를 fp32 추출 블록(626)으로 전달한다. Fp32 추출 블록(626)은 피연산자 A의 fp64 포맷으로의 최초 업-변환(up-conversion)을 수행한다. 특히, fp32 추출 블록(626)은 8비트 지수를 추출하고 fp64 포맷에 대해 정확한 바이어스를 갖는 11비트 지수를 생성하기 위해 1023-127-896을 추가한다. 23비트 가수는 트레일링 제로(trailing zero)들로 패딩된다(pad). Fp32 추출 블록(626)은 또한 피연산자 A가 fp32 특수수(예를 들어, INF, NaN, 0 또는 비정규화 수)인지 결정하고, 경로(644)를 통해 특수수 검출 회로(642)에 그 정보를 제공한다. Fp32 추출 블록(626)은 또한 피연산자에 절대값을 부정하거나 또는 적용할 수 있다.

[0174] 피연산자 C 다중화기(636)는 fp32 추출 블록(626)에 의해 제공된 업-변환 피연산자를 선택하고, 피연산자 A 다중화기(632) 및 피연산자 B 다중화기(634)는 0 연산자들을 선택한다. 특수수 검출 회로(642)는 피연산자가 fp32 비정규화 수가 아니라면 선두(leading) 1을 가수에 대해 프리펜딩한다(prepend). 특수수 검출 회로(642)는 또한, (모든 fp32 비정규화 수들이 fp64의 정상수들로서 표현될 수 있기 때문에) fp32 비정규화 수들이 정상수들로서 식별되는 것을 제외하고, 그의 특수수 SPC 신호들로서 fp32 추출 블록(626)에 의해 제공된 특수수 정보를 사용한다.

[0175] 가수 경로(516) 및 지수 경로(518)은 fp64 포맷으로 0.0*0.0+C를 계산하기 위해 DFMA 연산들에 대해 전술된 바와 같이 연산된다. 가수 경로(516) 및 지수 경로(518)의 정규화 구성요소(normalization element)는 업-변환된 fp64 피연산자를 정규화한다. 대안적인 실시예에서, 도 8을 참조하면, 지수 경로(518)로부터 정렬된 가수 C_align은 가수 경로(516)의 덧셈기(804) 주변에서 다중화기(806)의 Sum 입력으로 우회될 수 있고, 곱셈기(802) 및 덧셈기(804)는 저전력 상태로 놓일 수 있다. 유리하게, 부호 경로(520)는 부호 비트 Cs를 전달한다.

[0176] 출력부(522)(도 10)에서, 특수수 SPC 신호들이 입력 피연산자가 fp32 INF, NaN 또는 0이었던 것을 나타내지 않는다면, 정규화된 fp64 결과(Rm, Rs, Re)가 선택된다. 입력 피연산자가 fp32 INF였다면, 출력 다중화기 제어 로직(1002)은 유효 숫자 다중화기(1004)가 fp64 INF 유효 숫자(0x0_0000_0000_0000)를 선택하고, 지수 다중화기(1006)가 fp64 INF 지수(0x7ff)를 선택하도록 연산한다. 입력 피연산자가 fp32 NaN이었다면, 출력 다중화기 제어 로직(1002)은 유효 숫자 다중화기(1004)가 fp64 콰이어트된 NaN 유효 숫자를 선택하고, 지수 다중화기(1006)가 fp64 NaN 지수(0x7ff)를 선택하도록 연산한다. 입력 피연산자가 fp32 0이었다면, 출력 다중화기 제어 로직(1002)은 유효 숫자 다중화기(1004)가 fp64 0 유효 숫자(0x0_0000_0000_0000)를 선택하고, 지수 다중화기(1006)가 fp64 0 지수(0x000)를 선택하도록 연산한다. 상태 코드들은 원하는 대로 설정될 수 있다.

[0177] 2. 정수 내지 fp64(I2D)

[0178] I2D 연산들에 대해, 정수(u64, s64, u32 또는 s32 포맷)는 fp64 포맷으로 변환된다. 피연산자 준비 블록(514)(도 6)은 64비트 정수 피연산자 C를 수신한다. 32비트 정수 포맷들에 대해, 32 선두 제로들은 프리펜딩될 수 있다. 절대값/부정 회로(622)는 수정 없이 피연산자 C를 U/S 추출 블록(628)으로 전달한다. U/S 추출 블록(628)은 피연산자 C의 fp64 포맷의 최초 업-변환을 수행한다. 특히, 추출 블록(628)은 (예를 들어, 우선순위 인코더를 사용하여) 피연산자 C의 선두 1의 위치를 결정한다. 11비트 지수는 지수 필드를 (2^{63}에 대응하는) 1086으로 초기화함으로써 결정된다. 32비트 입력 포맷들에 대해, 선두 1이 드롭(drop)되고, 가수는 52비트 유효 숫자를 생성하기 위해 트레일링 제로들로 패딩된다. 64비트 입력 포맷들에 대해, 필요하다면 가수는 53비트로 끝수를 절단하고(truncate) 선두 1이 드롭된다. 가드 비트 및 라운드 비트는 원한다면 유지될 수 있다.

[0179] U/S 추출 블록(628)은 또한 입력 피연산자가 0인지 결정하고, 특수수 검출 회로(642)에 대한 대응하는 제어 신호를 생성한다. 다른 특수수들(비정규화 수들, INF 및 NaN)은 I2D 연산들 동안 발생하지 않고, 검출될 필요도 없다.

[0180] 피연산자 C 다중화기(636)는 U/S 추출 블록(628)에 의해 제공된 업-변환 피연산자를 선택하고, 피연산자 A 다중화기(632) 및 피연산자 B 다중화기(634)는 각각 0 피연산자를 선택한다. 특수수 검출 회로(642)는 입력 피연산자가 0인지를 나타내는 특수수 SPC 신호를 생성하기 위해 U/S 추출 블록(628)에 의해 제공된 0 정보를 사용한다.

[0181] 가수 경로(516) 및 지수 경로(518)는 0.0*0.0+C를 계산하기 위해 DFMA 연산들에 대해 전술된 바와 같이 연산된다. 가수 경로(516) 및 지수 경로(518)의 정규화 구성요소들은 업-변환된 fp64 피연산자를 정규화한다. 대안적인 실시예에서, 지수 경로(518)로부터 정렬된 가수 C_align은 가수 경로(516)의 덧셈기(804) 주변에서 다중화기(806)의 Sum 입력으로 우회될 수 있고, 곱셈기(802) 및 덧셈기(804)는 저전력 상태로 놓일 수 있다. 유리하게, 부호 경로(520)는 부호 비트 Cs를 전달한다.

[0182] 출력부(522)(도 10)에서, 특수수 SPC 신호들이 입력 피연산자가 정수 0이었다는 것을 나타내지 않는다면 정규화된 fp64 결과(Rm, Rs, Re)가 선택된다. 입력 피연산자가 정수 0이었다면, 출력 다중화기 제어 로직(1002)은 유효 숫자 다중화기(1004)가 fp64 0 유효 숫자(0x0_0000_0000_0000)를 선택하고, 지수 다중화기(1006)가 fp64 0 지수(0x000)를 선택하도록 연산한다. 상태 코드들은 원하는 대로 설정될 수 있다.

[0183] 3. fp64 내지 fp32(D2F)

[0184] fp64는 fp32보다 큰 범위의 부동소수점 값들을 커버하기 때문에, fp64로부터 fp32(D2F)로의 변환은 fp32 값의 오버플로우(overflow)들 및 언더플로우(underflow)들의 검출을 요구한다.

[0185] D2F 연산들에 대해, 피연산자 C는 피연산자 준비 블록(514)(도 6)에 제공된다. 절대값/부정 회로(622)는 원하는 대로 절대값 또는 피연산자 부정을 수행하고 특수수 검출 회로(642)에 제공될 피연산자 C를 선택하는 피연산자 C 다중화기(636)에 피연산자 C를 전달한다. 특수수 검출 회로(642)는 fp64 비정규화 수, 0, INF 또는 NaN을 검출하고, 대응하는 SPC 신호들을 출력부(522)에 제공한다. 선택 다중화기들(632 및 634)은 0.0 피연산자들을 선택한다.

[0186] 지수 경로(518)(도 7)에서, 지수 계산 블록(702)은 대응하는 fp32 지수를 결정하기 위해 fp64 지수를 897만큼 다운하여 바이어싱한다. fp32 지수가 언더플로우하면, 지수 계산 블록(702)는 언더플로우를 제거하기 위해 C의 가수를 오른쪽 이동할 Sh-C 신호를 생성한다(만일 217비트보다 더 많은 이동이 요구되면, C의 가수는 0이 될 것임). 이동 회로(706)는 Sh_C 신호에 따라 C의 가수를 오른쪽 이동한다. 그 결과는 다중화기(712)에 의해 선택되고, 정렬된 가수 C_align으로서 가수 경로(516)에 제공된다. 언더플로우 로직(713)은 fp32 언더플로우를 검

출하고 U_fp32 신호를 생성한다.

[0187] 가수 경로(516)(도 8)에서, 곱셈기(802)는 곱(product) 0.0*0.0을 계산한다(또는 우회된다). 곱(0)은 덧셈기(804)에 의해 가수 C_align에 추가된다. Sum 결과는 (입력이 부호/크기 형태이므로) 다중화기(806)에 의해 선택된다. 0 결과는 회로(814)에 의해 검출되고, 0이 아닌 결과들은 DFMA 연산들의 맥락으로 상기 기술된 바와 같이 정규화된다. 라운딩 로직(820)은 라운딩 업(round up)의 여부를 결정하는 데 사용될 수 있다. 그 결과가 23비트 fp32 가수가 됨에 따라, 1을 더하는 덧셈기(822)가 (53번째가 아닌) 24번째 비트 위치에 1을 추가할 필요가 있다는 것을 주목해야 한다.

[0188] 출력부(522)(도 10)는 그 결과를 집합시킨다. 23비트 fp32 유효 숫자가 52비트 필드 Rm에 제공된다. 출력 다중화기 제어 로직(1002)은, 결과가 fp32 정상수가 아니지 않는 한 Rm을 선택하기 위해 유효 숫자 다중화기(1004)를 제어한다. fp32 0 또는 INF에 대해, 0 가수 0x0_0000_0000_0000이 선택되고, fp32 NaN에 대해, 콰이어트된 fp32 NaN 가수가 선택된다. fp32 비정규화 수들에 대해, Rm이 사용될 수 있다.

[0189] 8비트 fp32 지수가 11비트 지수 필드에 제공된다. 출력 다중화기 제어 로직(1002)은 결과가 fp32 정상수가 아니지 않는 한 Re를 선택하기 위해 지수 다중화기(1004)를 제어한다. fp32 비정규화 수 또는 0에 대해, 0 지수 0x000이 선택된다. fp32 INF 또는 NaN에 대해, 최대 fp32 지수 0x7ff가 선택된다.

[0190] 연결(concatenation) 블록(1008)은 Rm 및 Re를 64비트 출력 필드의 31비트에 패킹하고 부호 비트 Rs를 추가한다. 52비트 유효 숫자의 29 LSB들과 같이, 11비트 지수의 3 MSB들이 드롭된다. fp32 결과는 원하는 대로, 예를 들어, 64비트 필드의 MSB들 또는 LSB들에서 정렬될 수 있다. 상태 코드들은 원하는 대로 설정될 수 있다.

[0191] 4. fp64 내지 정수(D2I)

[0192] D2I 연산들에 대해, 오버플로우들 및 언더플로우들이 검출된다. 오버플로우들은 최대 정수 값으로 설정되고, 언더플로우들은 0으로 설정된다.

[0193] 변환될 피연산자는 fp64 포맷의 피연산자 C로서 제공된다. 절대값/부정 회로(622)는 원하는 대로 절대값 또는 피연산자 부정을 수행하고, 특수수 검출 회로(642)에 제공될 피연산자 C를 선택하는 피연산자 C 다중화기(636)에 피연산자 C를 전달한다. 특수수 검출 회로(642)는 fp64 비정규화 수, 0, INF 또는 NaN을 검출하고, 대응하는 SPC 신호들을 출력부(522)에 제공한다. 선택 다중화기들(632 및 634)은 0.0 피연산자들을 선택한다.

[0194] 지수 경로(518)(도 7)에서, 지수 계산 블록(702)은 정수 위치에서 이진 소수점을 정렬하기 위해 Cm이 이동되어야 하는 양을 결정하기 위해 지수 Ce를 사용하고 대응하는 Sh_C 신호를 생성한다. 일 실시예에서, 지수 계산 블록(702)은 지수 바이어스를 제거하고 유효 숫자의 폭, 사용될 정수 포맷 및 32비트 포맷들에 대한 결과들이 (예를 들어, 32MSB들 또는 32LSB들을 사용하여) 64비트 필드로 표현되는 방법을 기술한다. 지수 Ce는 그 결과가 타겟 정수 포맷에서 오버플로우일지 또는 언더플로우일지 결정하는 데 사용될 수도 있다. 만일 그렇다면, 대응하는 오버플로우 또는 언더플로우 신호(정확히 도시되지 않음)는 출력부(522)(도 10)의 출력 다중화기 제어 로직(1002)에 유리하게 송신된다.

[0195] 이동 회로(706)는 C_Shift 만큼 Cm을 이동시키고, C_Shift 신호는 다중화기(712)에 의한 C_align 신호로서 선택된다.

[0196] 가수 경로(516)(도 8)에서, 곱셈기(802)는 덧셈기(804)에 0.0 결과를 제공한다. 덧셈기(804)는 C_align에 0.0을 추가하고 C가 플러스이거나 또는 마이너스인지에 따라 Sum 또는 ~Sum을 선택한다. 이동기(816)는 유리하게 그 결과를 이동시키지 않는다. 정수 포맷 블록(826)은 그 결과를 11비트 MSB 필드 int_M 및 53비트 LSB 필드 int_L로 분리한다.

[0197] 출력부(522)(도 10)에서, 출력 다중화기 제어 로직(1002)은, 오버플로우, 언더플로우 또는 특수수 피연산자의 경우를 제외하고, int_L 및 int_M 결과들을 각각 선택하기 위해, 유효 숫자 다중화기(1004) 및 지수 다중화기(1006)을 제어한다. 오버플로우들에 대해, 출력 포맷(u32, s32, u64 또는 s64)의 최대 정수가 선택되고, 언더플로우들에 대해, 0이 선택된다. 상태 코드들은 원하는 대로 설정될 수 있다.

[0198] Ⅳ. 추가 실시예들

[0199] 본 발명은 특정 실시예들에 대해 기술되었지만, 본 기술분야의 통상의 기술자는, 다수의 수정들이 가능하다는 것을 이해할 것이다. 예로서, DFMA 유닛은 결합(combination)의 더 많은, 더 적은 또는 상이한 기능들을 지원

하고, 임의의 포맷 또는 포맷들의 결합의 피연산자들 및 결과들을 지원하도록 구현될 수 있다.

[0200] 본원에 기술된 다양한 우회 경로들 및 통과(pass-throughs)는 변화될 수도 있다. 일반적으로, 임의의 회로 블록 주변에 우회 경로가 기술되는 경우, 그 경로는 동일(identity) 연산(즉, 0을 추가하는 것과 같은 그의 피연산자에 대해 영향이 없는 연산)에 의해 대체될 수 있다. 회로 블록은, 주어진 연산이 유휴 상태(예를 들어, 감소된 전력 상태)로 위치되는 동안 우회되거나, 또는 예를 들어 선택 다중화기들 또는 다른 회로들의 연산을 통해 다운스트림 블록들에 의해 무시되는 그의 결과를 이용하여 정상적으로 연산된다.

[0201] DFMA 파이프라인은 임의의 수의 스테이지들로 분할될 수 있고, 각각의 스테이지에서의 컴포넌트들의 결합은 원하는 대로 변화될 수 있다. 본원의 특정 회로 블록들에 속하는 기능은 파이프라인 스테이지들을 통해 분리될 수 있고, 예를 들어, 곱셈기 트리(tree)는 곱셈기 스테이지들을 차지할 수 있다. 다양한 블록들의 기능이 수정될 수도 있다. 일부 실시예들에서, 예를 들어, 상이한 덧셈기 회로들 또는 곱셈기 회로들이 사용될 수 있다.

[0202] 부가적으로, DFMA 유닛은 이해를 수월하게 하기 위해 회로 블록들의 관점에서 기술되었고, 본 기술분야의 기술자들은, 블록들이 다양한 회로 컴포넌트들 및 레이아웃들을 사용하여 구현될 수 있고, 본원에 기술된 블록들이 특정 컴포넌트들의 세트 또는 물리적 레이아웃으로 제한되지 않는다는 것을 이해할 것이다. 블록들은 원하는 대로 물리적으로 결합되거나 또는 분리될 수 있다.

[0203] 프로세서는 실행 코어에 하나 이상의 DFMA 유닛들을 포함할 수 있다. 예를 들어, 슈퍼스칼라 명령 발행(즉, 사이클 당 하나보다 많은 명령을 발행함) 또는 SIMD 명령 발행이 요구되는 경우, 다중 DFMA 유닛들이 구현될 수 있고, 상이한 DFMA 유닛들은 기능들의 상이한 결합들을 지원할 수 있다. 프로세서는 다중 실행 코어들을 포함할 수도 있고, 각각의 코어는 그 자신의 DFMA 유닛(들)을 가질 수 있다.

[0204] 일부 실시예들에서, 실행 코어가 SIMD 명령 발행을 지원하는 경우, 단일 DFMA 유닛은 적합한 입력 시퀀싱 및 출력 컬렉션 로직과의 결합에 사용되어, 다중 데이터 세트들이 단일 DFMA 파이프라인에서 순차적으로 처리되도록 할 수 있다.

[0205] 도 11은 본 발명의 실시예에 따른 DFMA 기능 유닛(1102)을 포함하는 실행 코어(1100)의 블록도이다. DFMA 유닛(1102)은 상술된 DFMA 유닛(320)과 유사하거나 또는 동일할 수 있다. 코어(1100)는, 단정도 피연산자들의 P개의 상이한 세트들을 갖는 동일한 명령이 P 단정도 SIMD 유닛들(1104)의 세트와 병렬로 발행될 수 있다는 것을 의미하는 SIMD 명령들을 발행한다. 각각의 SIMD 유닛(1104)은 동일한 연산 코드(opcode) 및 피연산자의 상이한 세트를 수신하고, P SIMD 유닛들(1104)은 P 결과들을 생성하기 위해 병렬로 연산한다. P-웨이(P-way) SIMD 명령들은 일련의 P 단일 명령, 단일 데이터(SISD) 명령들로서 DFMA 유닛(1102)에 발행된다.

[0206] (명령 발행 유닛의 일부일 수 있는) 입력 관리자(1106)는 SIMD 명령들에 대한 피연산자들을 집합시키고, SIMD 명령에 대한 피연산자들의 모든 P 세트들이 집합될 때, P SIMD 유닛들(1104) 또는 DFMA 유닛(1102) 중 하나에 피연산자들 및 적용가능한 연산 코드를 전달한다. 출력 컬렉터(1008)는 SIMD 유닛들(1104) 또는 DFMA 유닛(1102)으로부터 그 결과들을 모아 결과 버스(1110)를 통해 그 결과를 레지스터 파일(도 11에 명확히 도시되지 않음)에 전달한다. 일부 실시예들에서, 결과 버스(1110)는 또한, 레지스터 파일로의 전달과 병렬로 다음 명령과의 사용을 위해 결과들이 입력 관리자(1106)에 전달될 수 있도록, 입력 관리자(1106)에 우회 경로를 제공한다. 하나의 DFMA 유닛(1102)을 사용하는 SIMD 동작의 출현을 위해, 입력 관리자(1106)는 예를 들어, P 연속(consecutive) 클록 사이클들 각각에 대해 피연산자들의 상이한 세트로 동일한 연산 코드를 발행함으로써, DFMA 유닛(1102)에 명령들의 발행을 유리하게 연속화한다.

[0207] 도 12는 본 발명의 실시예에 따른 DFMA 유닛(1102)에 대한 연속화된 명령 발행을 나타내는 블록도이다. 도 11의 입력 관리자(1106)에 포함될 수 있는 입력 피연산자 컬렉션 유닛(1202)은 2개의 컬렉터들(1204, 1206)을 포함한다. 각각의 컬렉터(1204, 1206)는 단정도 피연산자들 A, B 및 C의 P 트리플릿(triplet)을 위한 충분한 공간을 제공하는 32비트 레지스터들의 배열이고, 다른 말로, 각각의 컬렉터(1204, 1206)는 단일 SIMD 명령에 대한 모든 피연산자들을 저장할 수 있다. 입력 피연산자 컬렉션 유닛(1202)은 예를 들어, 도 3의 레지스터 파일(324) 및/또는 도 11의 결과 버스(1110)로부터 피연산자들을 획득하고, 태그(tag)들 또는 다른 종래 기술들은 피연산자들이 주어진 명령에 대해 집합되는지를 결정하는 데 사용될 수 있다. 주어진 명령에 대한 피연산자들이, 그 명령이 발행되기 전의 여러 클록 사이클들에 집합되도록, 충분한 컬렉터들(1206)이 제공된다.

[0208] 단정도 명령들에 대해, 하나의 컬렉터(예를 들어, 컬렉터(1204))는 하나의 명령을 실행하기 위해 P SIMD 유닛들(1104)에 요구되는 모든 피연산자들을 이용하여 로딩된다. 명령이 P SIMD 유닛들(1104)에 발행될 때, 전체 컬렉터(1204)는 SIMD 유닛들(1104)의 각각에 전달되는 상이한 A, B, C 피연산자 트리플릿과 병렬로 유리하게 판독

된다.

[0209] DFMA 유닛(1102)으로의 명령들에 대해, 피연산자들은 배정도(예를 들어, 64비트)이다. 각각의 피연산자들은 컬렉터들(1204, 1206) 둘 다에서의 대응하는 레지스터들을 사용하여 저장될 수 있고, 예를 들어, 컬렉터(1204)의 레지스터(1208)는 피연산자 A의 일례의 32 MSB들(예를 들어, 부호 비트, 11 지수 비트들 및 유효 숫자의 20 MSB들)을 저장할 수 있는 반면, 컬렉터(1206)의 레지스터(1210)는 동일한 피연산자의 32 LSB들(예를 들어, 유효 숫자의 남아있는 32비트)을 저장한다. 따라서, P-웨이 배정도 SIMD 명령에 요구되는 피연산자 트리플릿 A, B, C 모두는 2개의 단정도 컬렉터들(1204, 1206)을 사용하여 집합될 수 있다.

[0210] 코어(1100)는 단지 하나의 DFMA 유닛(1102)을 갖고, P 연산자 세트들은, 둘 다 카운터(1216)에 의해 제어되는 출력 다중화기들(muxs)(1212, 1214)을 사용하여 순차적으로 유리하게 전달된다. 다중화기들(1212 및 1214)은 카운터(1216)에 응답하여 각각의 컬렉터들(1204 및 1206)로부터 피연산자 트리플릿의 MSB들 및 LSB들을 선택한다. 예를 들어, 나타낸 데이터 경로들에서, 다중화기(1212)는 컬렉터(1204)의 레지스터(1208)로부터 피연산자 A의 32 MSB들을 선택할 수 있는 한편, 다중화기(1214)는 컬렉터(1206)의 레지스터(1210)로부터 동일한 피연산자 A의 32 LSB들을 선택할 수 있다. 64 비트들이 배정도 폭 경로 상에서 DFMA 유닛(1102)에 전달된다. 유사하게, (레지스터들(1220 및 1222)로부터의) 피연산자 B 및 (레지스터들(1224 및 1226)로부터의) C는 동일한 카운터(1216)에 의해 제어된 대응하는 다중화기들(명확히 도시되지 않음)을 사용하여 DFMA 유닛(1102)에 전달될 수 있다. 다음 클록 사이클에 대해, 컬렉터들(1204 및 1206)에 있는 레지스터들의 다음 세트로부터의 피연산자들 A, B 및 C는 피연산자들의 모든 P 세트들이 전달될 때까지 DFMA 유닛(1102)에 전달될 수 있다.

[0211] 컬렉터들(1204 및 1206)과 함께 다중화기들(1212 및 1214)은 비록 감소된 처리량(throughput)이지만, DFMA 유닛(1102)에 대해 SIMD 실행의 외관(external appearance)을 제공한다. 따라서, 코어(1100)에 대한 프로그래밍 모델은, P-웨이 SIMD 실행이 배정도 명령들을 포함하는 모든 명령들에 대해 이용가능할 수 있다는 것을 가정할 수 있다.

[0212] 본원에 기술된 피연산자 컬렉션 및 시퀀싱 로직은 예시적이고, 변경들 및 수정들이 가능하다는 것이 이해될 것이다. SIMD 가능한 코어에서, 임의의 수의 DFMA 유닛들이 제공될 수 있고, 명령들이 임의의 수의 DFMA 유닛들에 병렬로 발행될 수 있다. 일부 실시예들에서, 단정도 연산들에 대한 배정도 연산들의 처리량은 DFMA 유닛들의 수에 따라 조절된다(scale). 예를 들어, P SIMD 유닛들 및 N DFMA 유닛들이 존재하면, 배정도 처리량은 단정도 처리량의 N/P일 것이다. 일부 실시예들에서, N은 P와 최적으로 동일하고, 다른 실시예에서, 다른 요소(factor)들 - 예를 들어, 레지스터 파일과 기능 유닛들 사이의 내부 데이터 경로들의 폭 - 은 존재하는 DFMA 유닛들의 수와 관계없이 단정도 처리량보다 작게 배정도 처리량을 제한할 수 있다. 그 경우에, N은 다른 제한 요소들이 허용하는 것보다 최적으로 더 크지 않다.

[0213] DFMA 유닛은, 단정도 기능 유닛들과 구별되기 때문에, 사용되지 않을 때, 예를 들어, 그래픽 프로세서 또는 코어가 배정도를 요구하지 않는 다른 계산들 또는 프로세스들을 렌더링하기 위해 배타적으로 사용될 때, 전력 다운될 수 있다는 것도 주목해야 한다. 또한, DFMA 유닛은 다른 회로 컴포넌트들의 연산에 영향 없이 집적 회로 설계에서 제거될 수 있다. 이는 상이한 칩들이 배정도 연산들에 상이한 레벨의 지원을 제공하는 제품 패밀리들의 설계를 용이하게 한다. 예로서, GPU 패밀리는 적어도 하나의 DFMA 유닛을 각각 포함하는 많은 코어들을 갖는 최고급 GPU, 및 하드웨어 기반의 배정도 지원 및 DFMA 유닛들이 없는 저급 GPU를 포함할 수 있다.

[0214] 또한, 본 발명은 그래픽 프로세서에 관해 기술되었지만, 본 기술분야의 통상의 기술자들은, 본 발명의 양태들이 매스 코프로세서들(math co-processors), 벡터 프로세서들 또는 범용 프로세서와 같은 다른 프로세서들에 채용될 수도 있다는 것을 이해할 것이다.

[0215] 따라서, 본 발명이 특정 실시예들에 관해 기술되었지만, 본 발명은 다음의 특허청구범위의 영역 내의 모든 수정들 및 균등물을 커버하고자 의도함을 이해할 것이다.

도면의 간단한 설명

[0216] 도 1은 본 발명의 실시예에 따른 컴퓨터 시스템의 블록도.

[0217] 도 2는 본 발명의 실시예에 따른 그래픽 처리 유닛에서 구현될 수 있는 렌더링 파이프라인의 블록도.

[0218] 도 3은 본 발명의 실시예에 따른 실행 코어의 블록도.

[0219] 도 4는 본 발명의 실시예에 따른 배정도 기능 유닛에 의해 수행될 수 있는 배정도 산술, 비교 연산들, 및 포맷

변환 연산들을 열거하는 도면.

[0220] 도 5는 본 발명의 실시예에 따른 배정도 기능 유닛의 단순 블록도.

[0221] 도 6은 도 5의 배정도 기능 유닛에 대한 피연산자 준비 블록의 블록도.

[0222] 도 7은 도 5의 배정도 기능 유닛에 대한 지수 경로의 블록도.

[0223] 도 8은 도 5의 배정도 기능 유닛에 대한 가수 경로의 블록도.

[0224] 도 9는 도 5의 배정도 기능 유닛에 대한 부호 경로의 블록도.

[0225] 도 10은 도 5의 배정도 기능 유닛에 대한 출력부의 블록도.

[0226] 도 11은 본 발명의 일 실시예에 따른 실행 코어의 블록도.

[0227] 도 12는 본 발명의 실시예에 따른 배정도 기능 유닛에 대한 피연산자 시퀀싱을 도시하는 블록도.

[0228] <도면의 주요 부분에 대한 부호의 설명>

[0229] 102: CPU

[0230] 104: 시스템 메모리

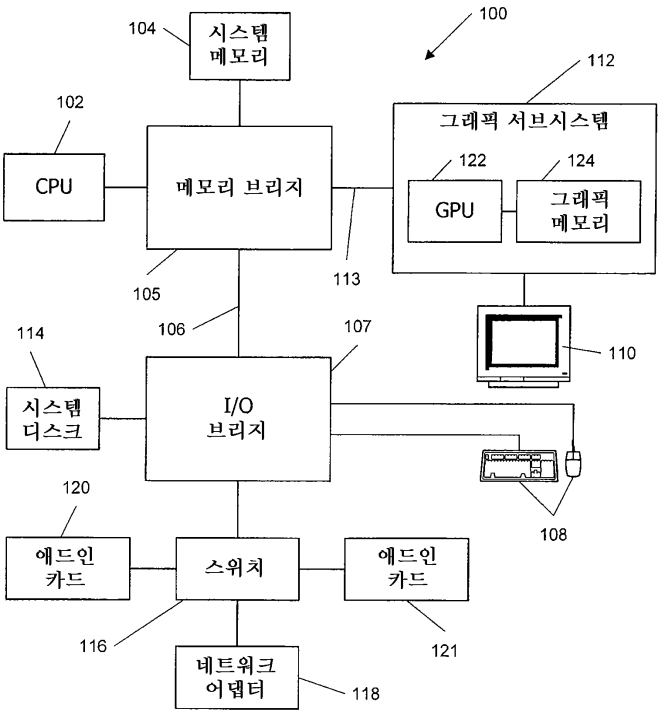
[0231] 105: 메모리 브리지

[0232] 107: I/O 브리지

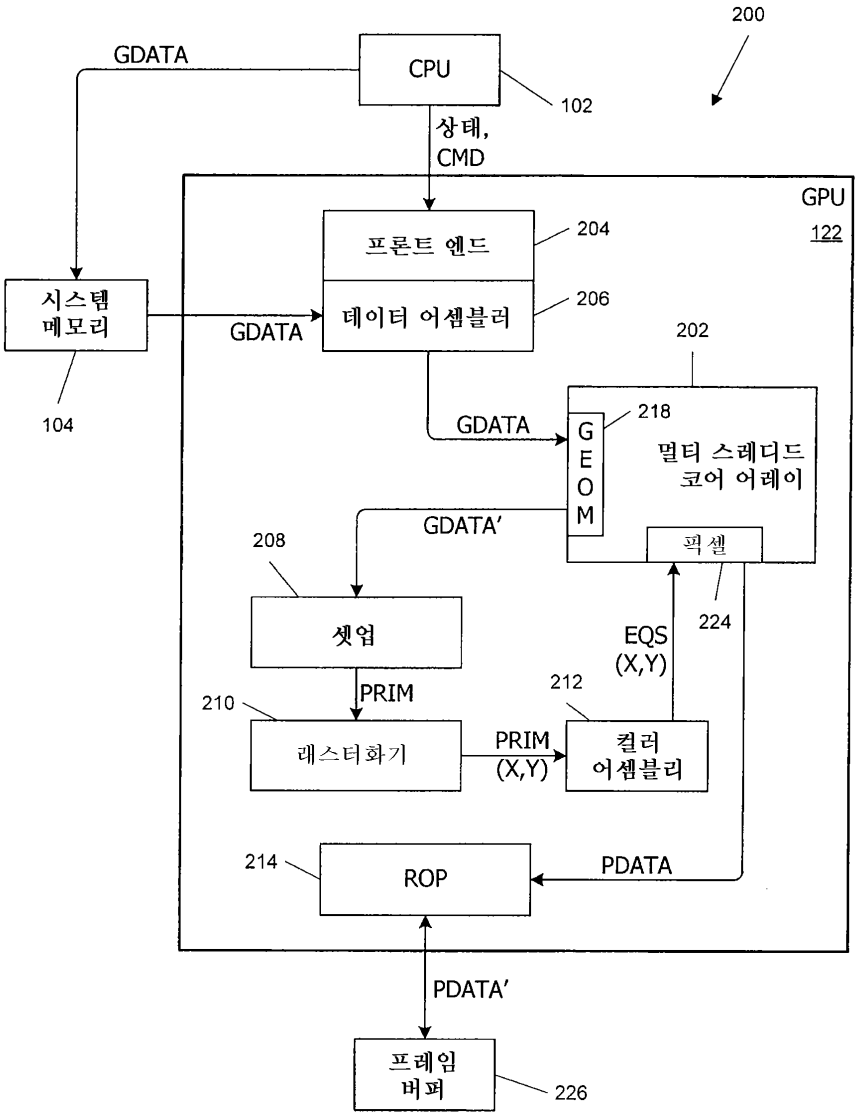
[0233] 116: 스위치

도면

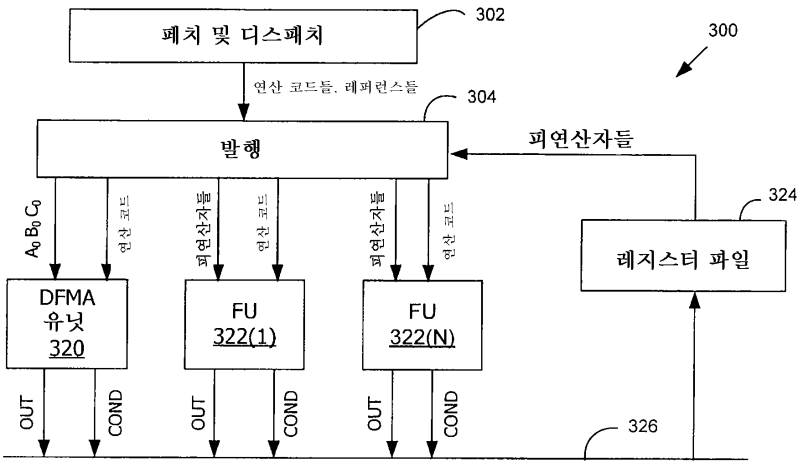
도면1



도면2


200
GDATA
CPU
102
상태,
CMD
GPU
122
프론트 엔드
204
시스템
메모리
데이터 어셈블러
206
GDATA
202
104
GDATA
218
G
E
O
M
멀티 스레디드
코어 어레이
GDATA'
픽셀
208
셋업
224
EQS
(X,Y)
PRIM
212
210
래스터화기
PRIM
(X,Y)
컬러
어셈블리
214
ROP
PDATA
PDATA'
프레임
버퍼
226

도면3


페치 및 디스패치
302
300
연산 코드들, 레퍼런스들
304
발행
피연산자들
324
$A_0 B_0 C_0$
연산 코드
피연산자들
연산 코드
피연산자들
연산 코드
레지스터 파일
DFMA
유닛
320
FU
322(1)
FU
322(N)
OUT
COND
OUT
COND
OUT
COND
326

도면4

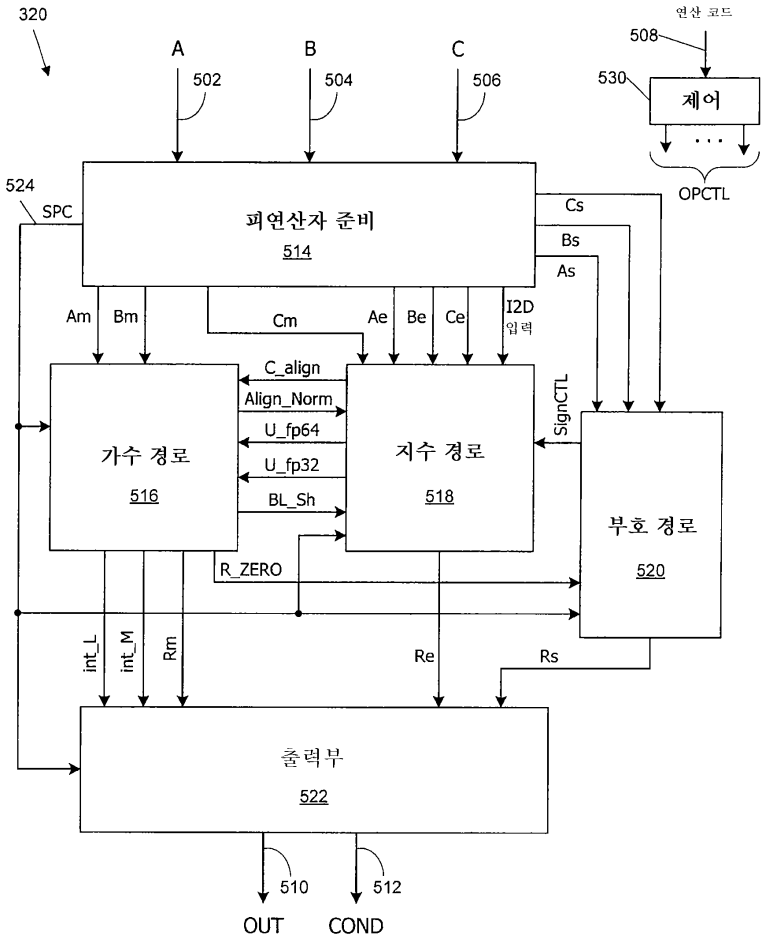
400

배정도 산술		402
이름	입력들	결과
DADD	A,C: fp64	A+C
DMUL	A,B: fp64	A*B
DFMA	A,B,C: fp64	A*B+C: fp64

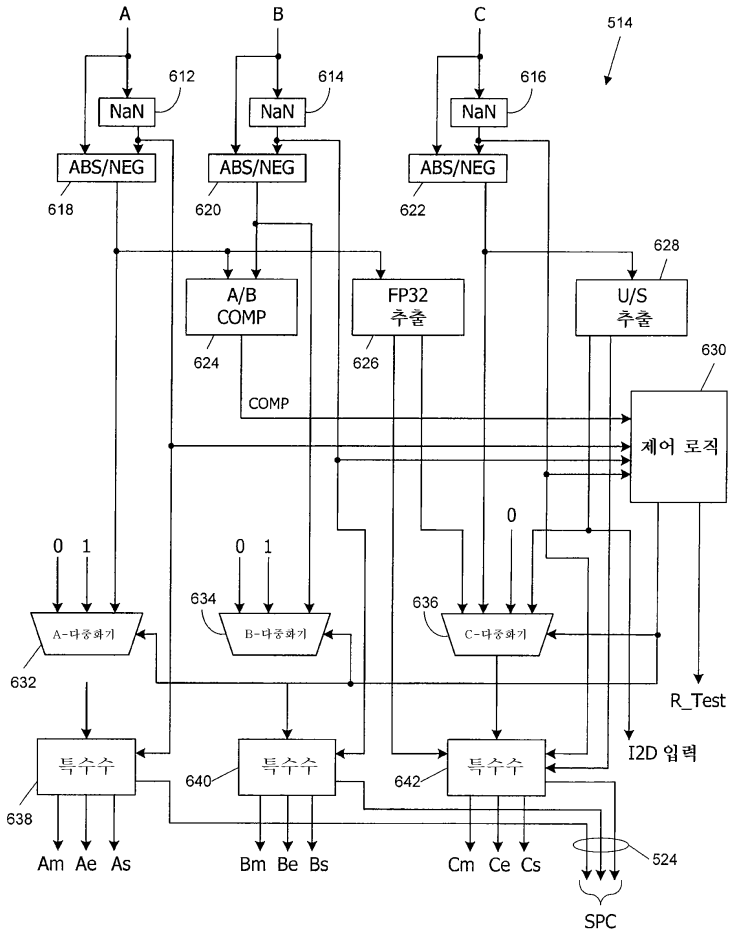
배정도 비교		404
이름	입력들	결과
DMAX	A,B: fp64	max(A,B)
DMIN	A,B: fp64	min(A,B)
DSET	A,B: fp64	R: boolean

포맷 변환 및 라운딩		406
이름	입력들	결과
D2F	A: fp64	A': fp32
F2D	A: fp32	A': fp64
D2I	A: fp64	A': u/s64 or u/s32
I2D	C: u/s64 or u/s32	C': fp64
D2D	A: fp64	A': fp64

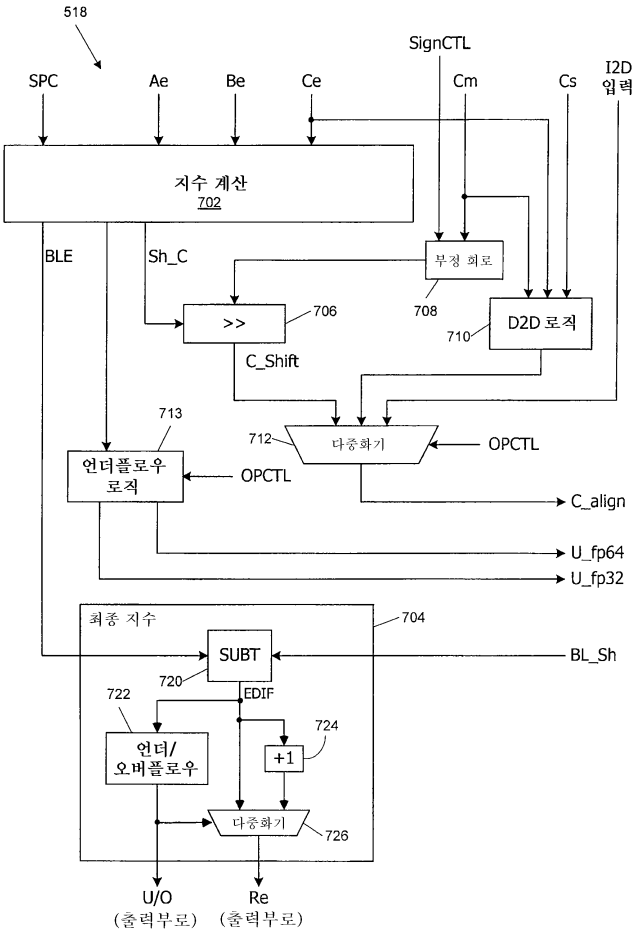
도면5


320
A
B
C
502
504
506
연산 코드
508
530
제어
OPCTL
524
SPC
피연산자 준비
514
Cs
Bs
As
Am
Bm
Cm
Ae
Be
Ce
I2D
입력
C_align
Align_Norm
U_fp64
U_fp32
BL_Sh
가수 경로
516
지수 경로
518
SignCTL
부호 경로
520
R_ZERO
int_L
int_M
Rm
Re
Rs
출력부
522
510
512
OUT
COND

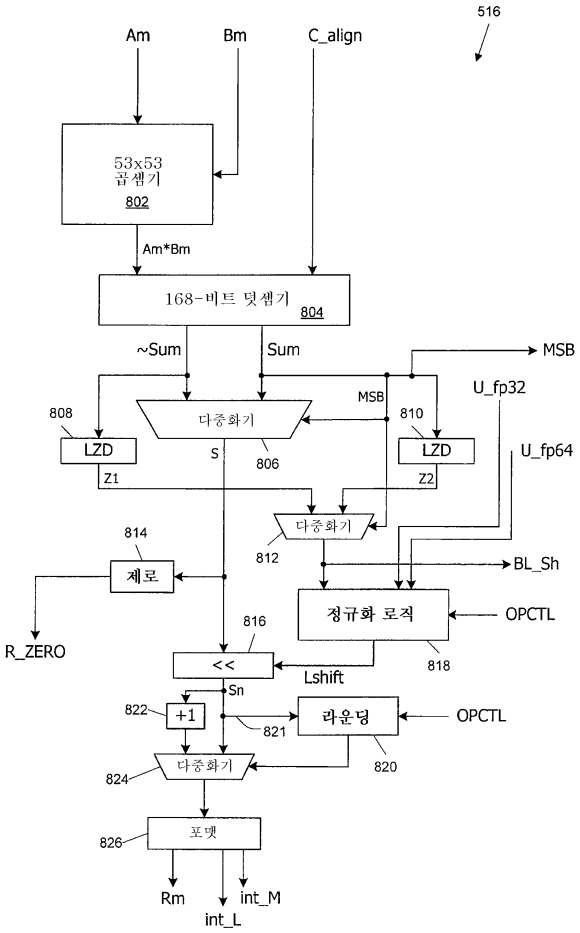
도면6


A
B
C
514
612
614
616
NaN
NaN
NaN
ABS/NEG
ABS/NEG
ABS/NEG
618
620
622
628
A/B
COMP
FP32
추출
U/S
추출
624
626
630
COMP
제어 로직
0
1
0
1
0
634
636
A-다중화기
B-다중화기
C-다중화기
632
R_Test
I2D 입력
특수수
특수수
특수수
640
642
638
Am Ae As
Bm Be Bs
Cm Ce Cs
524
SPC

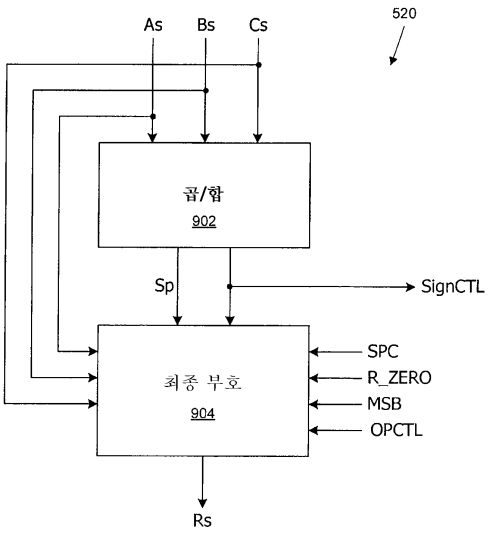
도면7


518
SPC
Ae
Be
Ce
SignCTL
Cm
Cs
I2D 입력
지수 계산
702
BLE
Sh_C
부정 회로
708
>>
706
D2D 로직
710
C_Shift
713
712
다중화기
OPCTL
언더플로우 로직
OPCTL
C_align
U_fp64
U_fp32
최종 지수
704
SUBT
BL_Sh
720
EDIF
722
724
언더/ 오버플로우
+1
다중화기
726
U/O
(출력부로)
Re
(출력부로)

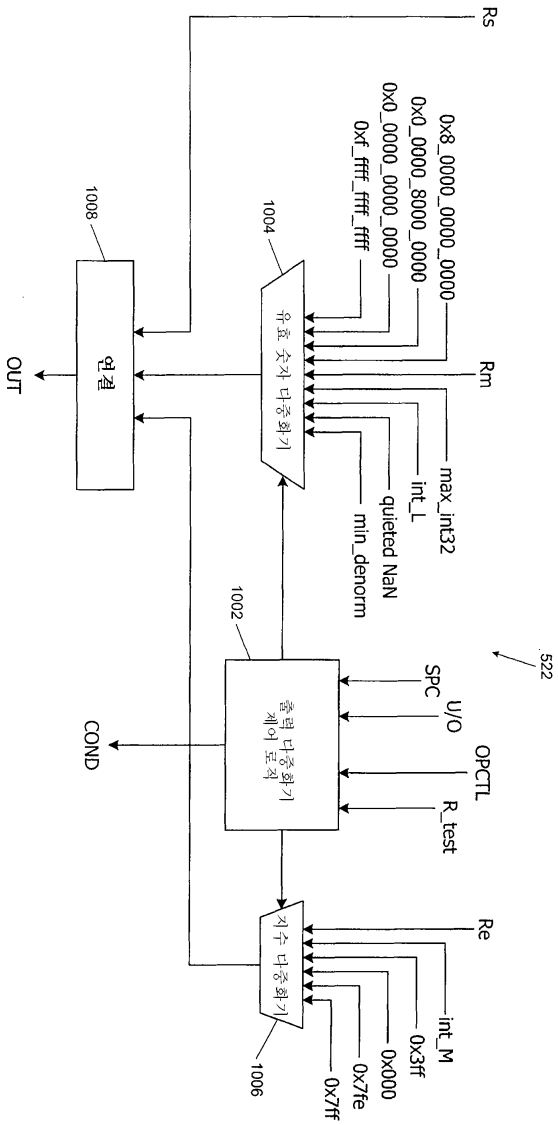
도면8


516
Am
Bm
C_align
53x53
곱셈기
802
Am*Bm
168-비트 덧셈기
804
~Sum
Sum
MSB
U_fp32
808
다중화기
MSB
810
LZD
S
806
LZD
U_fp64
Z1
Z2
다중화기
814
812
BL_Sh
제로
정규화 로직
OPCTL
R_ZERO
816
<<
818
Lshift
Sn
822
+1
821
라운딩
OPCTL
다중화기
824
820
포맷
826
Rm
int_M
int_L

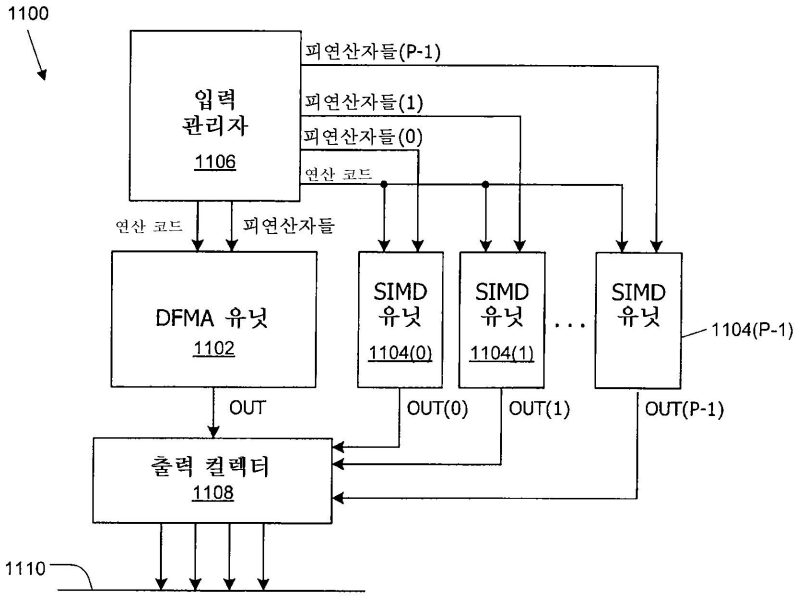
도면9


520
As
Bs
Cs
곱/합
902
Sp
SignCTL
최종 부호
904
SPC
R_ZERO
MSB
OPCTL
Rs

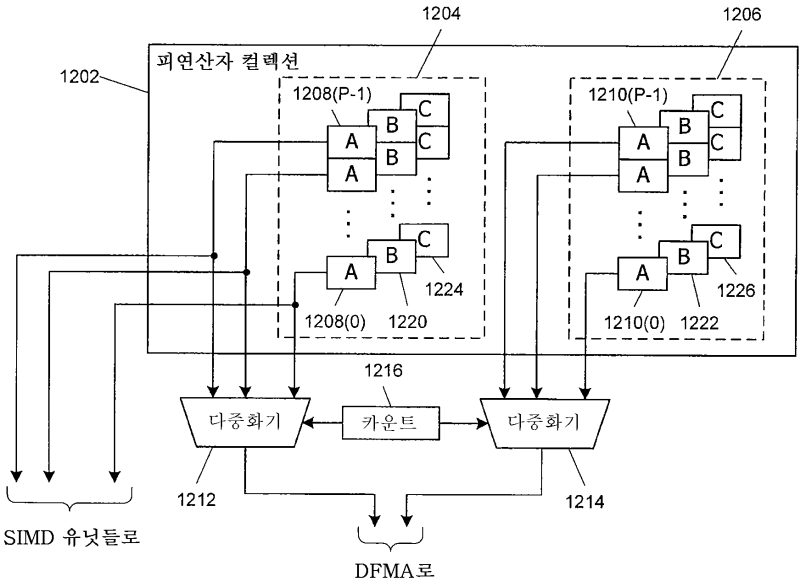
도면10


Rs
0x8_0000_0000_0000
0x0_0000_8000_0000
0x0_0000_0000_0000
0xf_ffff_ffff_ffff
1004
유효 숫자 다중화기
Rm
max_int32
int_L
quieted NaN
min_denorm
1008
연결
OUT
1002
출력 다중화기 제어 로직
522
SPC
U/O
OPCTL
R_test
COND
지수 다중화기
Re
int_M
0x3ff
0x000
0x7fe
0x7ff
1006

도면11


1100
입력
관리자
1106
피연산자들(P-1)
피연산자들(1)
피연산자들(0)
연산 코드
연산 코드
피연산자들
DFMA 유닛
1102
SIMD
유닛
1104(0)
SIMD
유닛
1104(1)
SIMD
유닛
1104(P-1)
OUT
OUT(0)
OUT(1)
OUT(P-1)
출력 컬렉터
1108
1110

도면12


1204
1206
1202
피연산자 컬렉션
1208(P-1)
1210(P-1)
A
B
C
1224
1226
1208(0)
1220
1210(0)
1222
1216
다중화기
카운트
다중화기
1212
1214
SIMD 유닛들로
DFMA로