(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2012-0025492
(43) 공개일자 2012년03월15일

(51) 국제특허분류(Int. Cl.)
G06F 9/38 (2006.01) *G06F 11/16* (2006.01)
(21) 출원번호 10-2011-7028603
(22) 출원일자(국제) 2010년04월27일
심사청구일자 없음
(85) 번역문제출일자 2011년11월29일
(86) 국제출원번호 PCT/US2010/032514
(87) 국제공개번호 WO 2010/126868
국제공개일자 2010년11월04일
(30) 우선권주장
12/432,146 2009년04월29일 미국(US)

(71) 출원인
어드밴스드 마이크로 디바이시즈, 인코포레이티드
미국 캘리포니아 94088-3453 서니베일 원 에이엠디 플레이스 메일 스톱68
(72) 발명자
수드하카르 란가나탄
미국 캘리포니아 95054 산타클라라 이스트 리버 파크웨이 806
콱 논 티.
미국 캘리포니아 95120 산호세 파이퍼 랜치 로드 6522
(74) 대리인
박장원

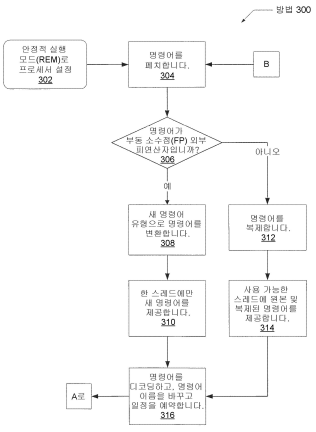
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 **SMT 기계에서 비교 및 전달 명령어를 사용한 안정적 실행**

(57) 요 약

동시 멀티스레딩 기계에서 효율적인 안정적 실행을 위한 시스템 및 방법. 프로세서는 중요한 소프트웨어 애플리케이션 실행 중 가능한 오류를 찾아내기 위해 안정적 실행 모드(REM)에 놓인다. 이 모드에서 작동할 스레드는 두 가지만 구성할 수 있다. 부동 소수점 저장 및 정수 전송 단항 명령어를 새 이진 명령어로 변환할 수 있다. 각각의 새 명령어는 두 개의 소스 피연산자를 가지며, 다른 스레드에 대응하는 각 피연산자에는 원래 단항 명령어의 단일 소스 피연산자와 동일한 논리 레지스터 번호가 지정된다. 다른 모든 명령어는 복제된다. 여기서 원래 명령어와 그 복제본이 서로 다른 스레드에 할당된다. 인스턴스화된 정수 독립적 클러스터와 함께 새 명령어를 사용하여 통신할 때 동시 멀티스레드(SMT) 부동 소수점 로직만 록스텝 실행을 제공하도록 할 수 있다. 새 명령어는 두 소스 피연산자가 모두 준비될 때까지 시작할 수 없다. 그러한 피연산자는 불일치 또는 오류 판별을 위해 순차적으로 비교된다.

대 표 도 - 도3a



특허청구의 범위

청구항 1

시스템으로서,

컴퓨터 프로그램의 명령어들을 저장하는 캐시와; 그리고

프로세서를 포함하며,

상기 프로세서는,

원래(original) 명령어가 제 1 명령어 유형에 대응하는지를 판별함에 응답하여, 상기 명령어들의 원래 명령어를 복제하여 원래 명령어의 복사본을 생성하고;

원래 명령어가 제 1 기능 유닛에 의해 실행될 제 2 명령어 유형에 대응하는지를 판별함에 응답하여, 상기 명령어들의 원래 명령어를 복제하지 않으며 -상기 제 1 기능 유닛의 M개의 복사본이 있으며 각 복사본은 복수의 스레드들 중 2개 이상의 스레드에 대응함- ; 그리고

소스 피연산자(source operand)가 제 1 스레드로부터 준비되고 그리고 소스 피연산자가 상기 복수의 스레드들 중 하나 이상의 다른 스레드로부터 준비되는지를 판별함에 응답하여, 상기 제 2 명령어 유형의 원래 명령어를 실행하도록 된 것을 특징으로 하는 시스템.

청구항 2

제1항에 있어서,

상기 제 2 명령어 유형의 원래 명령어를 실행하는 것에 응답하여,

상기 프로세서는 또한, 상기 제 1 기능 유닛으로부터 상기 제 1 스레드에 대응하는 제 2 기능 유닛과 상기 복수의 스레드들 중 다른 하나의 스레드에 대응하는 제 3 기능 유닛으로 단일 결과를 전달(convey)하도록 된 것을 특징으로 하는 시스템.

청구항 3

제2항에 있어서,

상기 원래 명령어가 제 1 명령어 유형에 대응하는지를 판별함에 응답하여,

상기 프로세서는 또한,

상기 원래 명령어를 복수의 스레드들 중 상기 제 1 스레드에 할당하고; 그리고

상기 원래 명령어의 상기 복사본을 상기 제 1 스레드와는 다른 하나 이상의 스레드에 할당하도록 된 것을 특징으로 하는 시스템.

청구항 4

제2항에 있어서,

상기 제 2 명령어 유형의 원래 명령어를 실행하고 그리고 각각의 소스 피연산자가 준비되는지를 판별함에 응답하여,

상기 프로세서는 또한, 각각의 소스 피연산자의 값들을 서로 비교하도록 된 것을 특징으로 하는 시스템.

청구항 5

제4항에 있어서,

상기 프로세서는 또한, 상기 제 1 명령어 유형의 원래 명령어의 결과와 상기 원래 명령어의 적어도 하나의 복사본의 결과를 비교하도록 된 것을 특징으로 하는 시스템.

청구항 6

제5항에 있어서,

상기 결과들의 비교, 및 소스 피연산자들의 비교 중 하나 이상으로부터 불일치(mismatch)를 검출하는 것에 응답하여, 상기 프로세서는 또한, 오류 정보를 전달하도록 된 것을 특징으로 하는 시스템.

청구항 7

제6항에 있어서,

상기 제 1 기능 유닛은 정수 실행 유닛이고 그리고 상기 제 2 기능 유닛은 부동-소수점 유닛(FPU)인 것을 특징으로 하는 시스템.

청구항 8

제7항에 있어서,

상기 제 1 명령어 유형은,

부동-소수점 저장 명령어(floating-point store instruction) 및 부동-소수점 정수-전달 명령어(floating-point integer-transfer instruction) 중 적어도 하나인 것을 특징으로 하는 시스템.

청구항 9

원래 명령어가 제 1 명령어 유형에 대응하는지에 응답하여, 명령어들 중 상기 원래 명령어를 복제하여 원래 명령어의 복사본을 생성하는 단계;

상기 원래 명령어가 제 1 기능 유닛에 의해 실행될 제 2 명령어 유형에 대응하는지에 응답하여, 상기 명령어들 중 상기 원래 명령어를 복제하지 않는 단계 -상기 제 1 기능 유닛의 M개의 복사본이 있으며 각 복사본은 복수의 스레드들 중 2개 이상의 스레드에 대응함- ; 그리고

소스 피연산자가 제 1 스레드로부터 준비되고 그리고 상기 복수의 스레드들 중 하나 이상의 다른 스레드로부터 준비되는지에 응답하여, 상기 제 2 명령어 유형의 원래 명령어를 실행하는 단계

를 포함하며,

각각의 소스 피연산자는 상기 원래 명령어의 동일 레지스터 번호에 의해 지정되는 것을 특징으로 하는 방법.

청구항 10

제9항에 있어서,

상기 제 2 명령어 유형의 원래 명령어를 실행하는 것에 응답하여, 하나의 제 1 기능 유닛으로부터 상기 제 1 스레드에 대응하는 제 2 기능 유닛과 상기 복수의 스레드들 중 다른 하나의 스레드에 대응하는 하나 이상의 다른 제 2 기능 유닛으로 단일 결과를 전달하는 단계를 더 포함하며,

상기 제 2 기능 유닛의 N개의 복사본이 있으며, 각 복사본은 상기 복수 스레드들 중 서로 다른 하나의 스레드(single different thread)에 대응하며, N은 M보다 큰 정수인 것을 특징으로 하는 방법.

청구항 11

제10항에 있어서,

상기 원래 명령어가 제 1 명령어 유형에 대응하는지에 응답하여,

상기 방법은,

상기 원래 명령어를 상기 복수의 스레드들 중 상기 제 1 스레드에 할당하는 단계와; 그리고

상기 원래 명령어의 상기 복사본을 상기 제 1 스레드와는 다른 하나 이상의 스레드에 할당하는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 12

제10항에 있어서,

상기 제 2 명령어 유형의 원래 명령어를 실행하고 그리고 각각의 소스 피연산자가 준비되는지에 응답하여,

상기 방법은,

각각의 소스 피연산자의 값들을 서로 비교하는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 13

제12항에 있어서,

상기 제 1 명령어 유형의 원래 명령어의 결과와 상기 원래 명령어의 적어도 하나의 복사본의 결과를 비교하는 단계를 더 포함하며,

상기 결과들은 동일한 파이프라인 스테이지에서 계산되는 것을 특징으로 하는 방법.

청구항 14

제13항에 있어서,

상기 결과들의 비교, 및 소스 피연산자들의 비교 중 하나 이상으로부터 불일치(mismatch)를 검출하는 것에 응답하여, 오류 정보를 전달하는 단계를 더 포함하는 것을 특징으로 하는 방법.

청구항 15

제14항에 있어서,

상기 제 1 기능 유닛은 정수 실행 유닛이고 그리고 상기 제 2 기능 유닛은 부동-소수점 유닛(FPU)인 것을 특징으로 하는 방법.

청구항 16

제15항에 있어서,

상기 제 1 명령어 유형은,

부동-소수점 저장 명령어(floating-point store instruction) 및 부동-소수점 정수-전달 명령어(floating-point integer-transfer instruction) 중 적어도 하나인 것을 특징으로 하는 방법.

청구항 17

부동-소수점(FP) 코프로세서(coprocessor)로서,

명령어 패치 유닛, 캐시 및 두 개 이상의 정수 실행 클러스터(IEC)와 통신하는 인터페이스 -각각의 IEC는 복수의 스레드들 중 서로 다른 하나의 스레드에 대응하며- 와;

저장 요소와; 그리고

회로를 포함하며,

상기 회로는,

원래 명령어가 제 1 명령어 유형에 대응하는지에 응답하여, 명령어들 중 상기 원래 명령어를 복제하여 원래 명령어의 복사본을 생성하고;

상기 원래 명령어가 제 1 기능 유닛에 의해 실행될 제 2 명령어 유형에 대응하는지에 응답하여, 상기 명령어들 중 상기 원래 명령어를 복제하지 않으며 -상기 제 1 기능 유닛의 M개의 복사본이 있으며 각 복사본은 상기 복수의 스레드들 중 2개 이상의 스레드에 대응함- ; 그리고

소스 피연산자가 제 1 스레드로부터 준비되고 그리고 상기 복수의 스레드들 중 하나 이상의 다른 스레드로부터 준비되는지에 응답하여, 상기 제 2 명령어 유형의 원래 명령어를 실행하도록 되며,

각각의 소스 피연산자는 상기 원래 명령어의 동일 레지스터 번호에 의해 지정되는 것을 특징으로 하는 부동-소수점 코프로세서.

청구항 18

제17항에 있어서,

상기 제 2 명령어 유형의 원래 명령어를 실행하는 것에 응답하여,

상기 회로는 또한,

하나의 제 1 기능 유닛으로부터 상기 제 1 스레드에 대응하는 IEC와 상기 복수의 스레드들 중 다른 하나의 스레드에 대응하는 하나 이상의 다른 IEC로 단일 결과를 전달하며,

상기 IEC의 N개의 복사본이 있으며, 각 복사본은 상기 복수 스레드들 중 서로 다른 하나의 스레드에 대응하며, N은 M보다 큰 정수인 것을 특징으로 부동-소수점 코프로세서.

청구항 19

제18항에 있어서,

상기 원래 명령어가 제 1 명령어 유형에 대응하는지에 응답하여,

상기 회로는 또한,

상기 원래 명령어를 상기 복수의 스레드들 중 상기 제 1 스레드에 할당하고; 그리고

상기 원래 명령어의 상기 복사본을 상기 제 1 스레드와는 다른 하나 이상의 스레드에 할당하는 것을 특징으로 하는 부동-소수점 코프로세서.

청구항 20

제18항에 있어서,

상기 제 2 명령어 유형의 원래 명령어를 실행하고 그리고 각각의 소스 피연산자가 준비되는지에 응답하여,

상기 회로는 또한,

각각의 소스 피연산자의 값들을 서로 비교하는 것을 특징으로 하는 부동-소수점 코프로세서.

명 세 서

기 술 분 야

[0001] 본 발명은 컴퓨팅 시스템과 관련이 있으며, 구체적으로는 동시 멀티스레딩 시스템에서 효율성과 안정성을 높인 실행을 위한 것이다.

배 경 기 술

[0002] 중요 업무용 소프트웨어 응용 프로그램에는 높은 안정성이 요구된다. 이러한 응용 프로그램의 예로는 재무 및 금융 소프트웨어, 데이터베이스, 군사 응용 프로그램 등이 있다. 여러 가지 소프트웨어 테스트 방법을 사용하여 소프트웨어 응용 프로그램이 미리 정해진 품질 수준에 부합되는지 확인하고 검증할 수 있다. 그러나 응용 프로그램을 실행하기 위해서 이용하는 마이크로프로세서와 같은 하드웨어 플랫폼이 문제를 유발할 수도 있다. 마이크로프로세서를 테스트한 결과, 미리 정해진 요건을 충족하는 것으로 확인되었더라도 소프트웨어와 마찬가지로 초기 상태 등의 전제 조건과 입력의 모든 조합 아래에서 테스트 결과는 불확실하다. 기능적 오류 외에, 고착 고장 (stuck-at faults) 등의 하드웨어 오류와 스토리지 노드에서 발생하는 방사선 유발 오류와 같은 소프트웨어 오류도 최신 마이크로프로세서에서는 발생할 수 있다.

[0003] 다양한 차세대 신형 프로세서가 등장하면서 노드 정전용량 및 공급 전압이 점차 감소하고, 노드에 저장된 전하량도 감소하고 있다. 이러한 사실 때문에 노드는 우주 광선, 알파 입자, 중성자 등의 높은 에너지 입자가 원인이 되는 방사선 유발 오류에 더욱 취약하다. 방사선이 트랜지스터의 소스 및 드레인 영역에서 소스 및 드레인 다이오드에 의해 이송되는 소수 캐리어를 생성한다. 세대의 진행에 따라 감소하고 있는 총 전하량과 비교한 노

드에 저장된 전하량의 변화율은 회로의 노이즈 여유를 능가하며 노드의 저장 상태를 바꾸기에 충분한 크기의 백분율에 도달할 수 있다. 회로가 방사선에 의해 영구히 손상되지는 않더라도 로직 오류가 발생할 수는 있다.

[0004] 이상과 같은 이유로 SRAM (Static Random Access Memory) 과 같은 메모리는 오류 수정 코드 (Error Correcting Code, ECC) 를 사용해 소프트 오류를 검출하여 수정한다. 플립플롭 등의 순차 요소는 소프트 오류에 대응하기 위하여 정전용량을 높인 노드나 중복 래치를 설계에 사용하기도 한다. 하지만 정수 및 부동 소수점 기능 유닛과 같은 조합 로직 내 노드 또한 소프트 오류에 취약하다. 따라서 기능 오류와 하드 오류를 검사하는 테스트에서 조합 로직이 소프트 오류에 안전함이 증명되지 않았으며, 결국 중요 업무용 응용 프로그램에는 허용되지 않을 수도 있다.

[0005] 기능적 장애, 하드 오류, 소프트 오류 등, 오류의 유형에 관계없이, 중요 업무용 응용 프로그램은 오류 발생에 대한 허용도가 낮고, 특정 데이터 세트를 사용한 반복적 실행을 허용하지 않을 수 있다. 특정 하드웨어에서 응용 프로그램의 올바른 실행을 유지하고 오류를 검출하기 위해서 검사점을 적용한 응용 프로그램의 두 가지 병렬 실행을 수행할 수 있다. 그리고 각 검사점에서 각 실행 결과 데이터를 비교할 수 있다. 두 결과는 동일해야 한다. 따라서 동시 실행은 록스텝 방식으로 수행된다. 검사점에서 비교 결과 확인된 모든 차이는 오류로 표시할 수 있다. 동시 실행 작업은 두 실행을 모두 성공한 이전 검사점으로 롤백할 수 있고, 해당 검사점에서 병렬 실행을 다시 수행할 수 있다. 또한 플래그나 경고를 사용자에게 보고하기도 한다. 결과 데이터가 일치하지 않음이 확인될 때, 사용자는 문제가 있는 검사점에서 차이가 다시 나타나는지 검사하거나 응용 프로그램을 디버깅할 수 있다.

[0006] 하지만 효율적인 병렬 록스텝 실행을 수행하기 어려운 경우도 있다. 예를 들어 마이크로프로세서 두 개를 사용하는데 각 마이크로프로세서가 동시에 응용 프로그램 복사본을 하나씩 실행하고 다른 마이크로프로세서와 동시에 실행을 시작하는 경우, 동일하지 않은 DMA(Direct Memory Access) 시간과 동일하지 않은 새로 고침 작업 등과 같은 이유로 록스텝 실행이 불가능할 때도 있다. 따라서 여러 하드웨어 및 기능 유닛 복사본을 가진 마이크로프로세서 한 개를 사용하는 것이 더 유리할 수 있다.

[0007] 많은 최신 마이크로프로세서가 멀티스레딩 연산을 실행하기 위해서 여러 복사본 코어를 이용하며, 이때 각 코어는 다른 코어와 동시에 별도 소프트웨어 스레드에서 독립적으로 작동할 수 있다.

[0008] 중요 업무용 응용 프로그램의 록스텝 실행에 성공하는 한 가지 방법은 한 마이크로프로세서 내의 두 가지 복사본에서 응용 프로그램 원본과 복사본을 동시에 실행하는 것이다. 첫째 코어가 원래의 명령어를 수신하고 둘째 코어가 복사본 명령어를 수신하는 경우와 같이 각 코어가 동일한 명령어를 수신하면 각 클럭 사이클에서 관련 데이터 비교를 수행할 수 있다. 따라서 중요 업무용 응용 프로그램의 록스텝 실행을 달성할 수 있다.

[0009] 그러나 다른 여러 가지 요소가 이 록스텝 실행을 중단시킬 수 있다. 예를 들어 한 마이크로프로세서 내에서 멀티스레딩 연산을 수행하기 위해서 모든 하드웨어 리소스가 복사되지는 않는다. 온다이 공간을 많이 점유하는 복잡한 로직이 부동 소수점 유닛 (Floating-Point Unit, FPU) 에 포함된다. 또한 FPU 연산이 때로 수행되지 않는다. 그러므로 설계자가 다이에서 값비싼 독립적 부동 소수점 로직 복사본을 여러 개 생성하는 것은 바람직하지 않다. 대신에 동시 멀티스레딩 (Simultaneous Multi-Threading, SMT) 을 통하여 FPU 및 온다이의 다른 하드웨어 리소스에 대한 멀티스레딩 연산을 수행할 수 있다.

[0010] 멀티스레딩과 마찬가지로 SMT에서 둘 이상의 스레드에서 나오는 명령어들이 한꺼번에 주어진 파이프라인 단계에서 실행될 수 있으며, 메모리 지연을 숨기고 사용된 하드웨어 용량당 컴퓨팅 처리량을 증가시키는 데 사용될 수 있다. 그러나 SMT는 주 실행 리소스는 복제하는 것이 아니라 아키텍처 상태를 저장하는 섹션과 같은 특정 프로세서 섹션을 복제하여 연산된다. 결과적으로 호스트 운영 체제에는 SMT 탑재 프로세서 한 개가 두 개의 "논리" 프로세서로 보이게 된다. 따라서 운영 체제가 두 개 이상의 스레드를 예약하거나 동시에 처리할 수 있다. 비 SMT 가능 프로세서의 실행 리소스가 현재 스레드에 사용되지 않는 경우, 특히 캐시 누락, 분기 예측 오류 및 기타 사유로 인해 프로세서가 중단될 때, SMT 탑재 프로세서가 그러한 실행 리소스를 사용하여 또 하나의 예정된 스레드를 실행할 수 있다.

발명의 내용

해결하려는 과제

[0011] FPU와 같은 SMT 하드웨어는 두 가지 스레드 작업을 록스텝 방식으로 수행하지 않는다. 따라서 두 가지 독립적 정수 클러스터 복사본과 같은 여러 하드웨어 복사본과 통신이 두 개의 정수 클러스터 내에서 록스텝 실행을 중

단시킬 수 있다. 또한 안정적 실행 모드로 작동하지 않을 때 마이크로프로세서의 성능 저하를 막기 위해서 스케줄러를 수정하고 로직 이름을 바꾸거나 FPU로부터 수신된 비 록스텝 신호를 동기화하기 위해 정수 클러스터 사이에서 신호를 라우팅하는 것은 바람직하지 않다.

[0012] 이상의 관점에서 동시 멀티스레딩 기계에서 효율적이며 안정적 실행을 위한 효율적인 방법과 메커니즘이 필요하다.

과제의 해결 수단

[0013] 동시 멀티스레딩 기계에서 효과적이며 안정적 실행을 위한 시스템과 방법이 고려되고 있다. 한 실시 예에서 캐시와 프로세서로 구성되는 컴퓨터 시스템을 제공한다. 프로세서가 동시 멀티스레드 (SMT) 부동 소수점 유닛 (FPU) 한 개와 인스턴스화된 독립적 정수 클러스터 여러 개로 구성될 수 있으며, 이때 각 클러스터는 다른 스트랜드 또는 하드웨어 스레드에 해당한다. 안정적 실행 모드 (Reliable Execution Mode, REM) 에서는 두 개의 스레드에서만 작동하도록 프로세서를 구성할 수 있다. 중요 업무용 소프트웨어 응용 프로그램의 명령어가 프로세서에서 실행될 수 있다. FPU 로부터 정수 클러스터로 통신되는 부동 소수점 외부 연산이 아닌 각 명령어가 복제된다. 원래의 명령어는 첫째 스레드에 할당된다. 복사본 명령어는 둘째 스레드에 할당된다.

[0014] 프로세서 프론트 엔드 및 정수 클러스터는 록스텝 방식으로 두 스레드의 명령어를 실행할 수 있다. 설계 요구사항에 따라 실행 완료 또는 중지 시 두 결과의 비교를 포함하는 검사점이 발생할 수 있다. 불일치가 발생하면 프로세서는 사용자가 검사할 수 있도록 캐시에서 미리 정한 위치로 오류 정보를 전달할 수 있다.

[0015] FPU 로부터 정수 클러스터로 통신되는 부동 소수점 외부 연산인 각 명령어가 새 명령어로 변환된다. 일반적으로 부동 소수점 외부 연산은 부동 소수점 저장 명령어 또는 부동 소수점 정수 전달 명령어와 같은 단항(unary) 명령어이다. REM 연산 동안 두 스레드가 연산되는 경우 단항 명령어가 이진 명령어로 변환된다. 두 소스 피연산자는 각 연산 스레드에 해당하며 첫째 명령어의 단일 소스 피연산자와 동일한 논리 레지스터 번호에 의해 지정된다.

[0016] 새 명령어를 실행하는 동안 FPU는 두 소스 피연산자가 준비되기를 기다려야 하는데, 이는 부동 소수점 스케줄러의 정상적인 작동이다. 두 소스 피연산자의 비교 결과는 일치해야 한다. 불일치가 발견되면, 앞에서 설명했듯이 사용자가 검사할 수 있도록 프로세서가 캐시에서 미리 정한 위치로 오류 정보를 전달한다. 원하는 설계에 따라 저장 또는 정수 전달 작업을 계속 진행하거나 실행을 중지할 수도 있다. 불일치가 발생하면 FPU가 정수 클러스터들로 동시에 결과를 전달한다. 따라서 부동 소수점 연산은 비동기식으로 실행될 수 있지만 FPU로부터 정수 클러스터로의 통신은 동기식 작업이다. 프로세서는 록스텝 방식으로 실행을 계속한다.

[0017] 또 다른 실시 예에서는 여러 인스턴스화된 독립적 정수 클러스터와 마찬가지로 명령어 페치 단위로부터 명령어를 수신하는 SMT FPU가 제공된다. 프로세서가 REM 연산 모드에 있을 때 위에서 설명한 모든 단계를 수행하도록 FPU 내 로직이 구성된다.

도면의 간단한 설명

[0018] 도1은 예시적인 마이크로프로세서의 실시 예를 보여주는 일반화된 블록 다이어그램이다.

도2는 비순차적 실행을 수행하는 범용 프로세서 코어의 실시 예를 보여주는 일반화된 블록 다이어그램이다.

도3A는 동시 멀티스레딩 프로세서에서 효율적이며 안정성 높은 실행 방법의 실시 예를 보여주는 흐름도이다.

도3B는 동시 멀티스레딩 프로세서에서 효율적이며 안정성 높은 실행 방법의 실시 예를 보여주는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0019] 본 발명은 다양한 수정과 대체 형태가 가능하지만 몇 가지 구체적인 실시예들을 도면에서 예시하며 본 명세서에서 자세히 설명한다. 하지만 도면과 자세한 설명은 공개된 특정 형태로 본 발명을 제한하기 위한 것이 아니며, 반대로 첨부한 청구항에 정의된 대로 본 발명의 정신과 범위에 속하는 모든 수정, 등가물 및 대체물이 본 발명에 포함된다는 사실을 인식해야 한다.

[0020] 다음의 설명에서, 본 발명의 완전한 이해를 돕기 위해 많은 구체적인 설명을 제시한다. 하지만 기술 분야에서 평균 수준의 기량이면 구체적인 설명 없이도 본 발명을 실현할 수 있다. 일부 예에서, 본 발명의 왜곡을 피하기 위하여 널리 알려진 회로, 구조 및 기법은 상세히 제시하지 않았다.

[0021] 도1 에서는 모범적 마이크로프로세서 100의 실시예를 보여준다. 마이크로프로세서 100 에는 메모리 컨트롤러 120, 인터페이스 로직 140, 그리고 하나 이상의 프로세서 코어 112 와 해당하는 캐시 메모리 서브시스템 114 를 포함할 수 있는 하나 이상의 처리 장치 115, 크로스바 상호연결 논리 116, 공유 캐시 메모리 서브시스템 118 이 포함될 수 있다. 한 실시 예에서 제시된 마이크로프로세서 100의 기능은 단일 집적 회로에 통합된다.

[0022] 인터페이스 140 은 일반적으로 마이크로프로세서 100 에서 공유 캐시 메모리 서브시스템 118 과 처리 장치 115 에 입/출력 (I/O) 장치용 인터페이스를 제공한다. 여기에 사용된 대로, 참조 숫자와 그 뒤의 문자로 참조된 요소들을 숫자만으로 총칭할 수도 있다. 예를 들어 처리 장치 115a-115b 를 처리 장치 115 또는 장치 115 로 총칭할 수 있다. I/O 장치에는 프린터, 키보드, 모니터, 카메라, 카드 판독기, 하드 또는 플로피 디스크 드라이브 또는 드라이브 컨트롤러, 네트워크 인터페이스 카드, 비디오 가속기, 오디오 카드, 모뎀, 그 밖에 범용 인터페이스 버스 (GPIB) 등의 다양한 데이터 획득 카드 또는 필드 버스 인터페이스 카드 등이 포함될 수 있다. 또한, 인터페이스 140 은 다른 마이크로프로세서 및/또는 다른 처리 노드와 통신에 사용할 수 있다. 일반적으로 인터페이스 로직 140 은 해당 링크로부터 패킷 수신용 버퍼와 해당 링크에서 전송될 패킷 버퍼링용 버퍼로 구성될 수 있다. 마이크로프로세서 100 과 패킷 송수신에는 적절한 흐름 제어 메커니즘을 사용할 수 있다.

[0023] 마이크로프로세서 100 은 각각의 메모리 컨트롤러 120 을 통하여 각 메모리에 결합될 수 있다. 메모리는 적합한 메모리 장치들로 구성될 수 있다. 예를 들어 하나 이상의 RAMBUS 동적 랜덤 액세스 메모리 (Dynamic Random Access Memory, DRAMs), 동기 (SDRAM, Synchronous DRAM), DRAM, 정적 RAM 등으로 메모리가 구성될 수 있고, 여러 메모리 사이에서 마이크로프로세서 100 의 주소 공간을 나눌 수 있다. 마이크로프로세서 100 으로 구성된 처리 노드 또는 마이크로프로세서 100 자체 각각에 메모리 맵이 포함될 수 있다. 메모리 맵은 메모리로 매핑되고 결과적으로 특정 주소에 대한 메모리 요청이 라우팅되는 마이크로프로세서 100 또는 처리 노드로 매핑되는 주소를 결정하는 데 사용된다. 한 실시 예에서, 주소의 일관성 유지 지점 (coherency point) 은 주소에 해당하는 메모리 저장 바이트에 결합되는 메모리 컨트롤러 120이다. 메모리 컨트롤러 120은 메모리 인터페이스용 제어 회로로 구성할 수 있다. 메모리 요청들이 대기할 요청 큐도 메모리 컨트롤러 120 에 포함될 수 있다.

[0024] 일반적으로 크로스바 상호연결 로직 116 은 인터페이스 140 에 결합된 링크에서 수신된 제어 패킷에 응답하고, 프로세서 코어 112 및/또는 캐시 메모리 서브시스템 114 에 응답하여 제어 패킷을 생성하고, 서비스를 위해 메모리 컨트롤러 120 에 의해 선택된 트랜잭션에 응답해 프로브 명령을 생성하여 패킷에 응답하며, 마이크로프로세서로 구성되는 중간 노드의 패킷을 인터페이스 논리 140 을 통하여 다른 노드로 라우팅하도록 구성한다. 인터페이스 논리 140 에는 패킷을 수신하여 크로스바 상호연결 논리 116 에 사용된 내부 클럭과 동기화하는 논리가 포함된다. 크로스바 상호연결 논리 116 은 프로세서 코어 112 에서 보낸 메모리 요청을 공유 캐시 메모리 서브시스템 118 또는 메모리 컨트롤러 120 과 이하 수준 메모리 서브시스템으로 전달하도록 구성할 수 있다. 또한, 크로스바 상호연결 논리 116 은 더 낮은 수준 메모리로부터 수신된 메모리 라인과 제어 신호를 메모리 컨트롤러 120 을 통하여 프로세서 코어 112 및/또는 캐시 메모리 서브시스템 114 와 118 로 전달할 수 있다. 크로스바 상호연결 논리 116, 메모리 컨트롤러 120, 인터페이스 140 과 프로세서 장치 115 사이 상호연결 버스 구현은 적절한 기술로 구성할 수 있다.

[0025] 캐시 메모리 서브시스템 114 와 118 은 데이터 블록을 저장하도록 구성된 고속 캐시 메모리로 구성할 수 있다. 캐시 메모리 서브시스템 114 를 각각의 프로세서 코어 112 내에 통합할 수 있다. 또는 뒷면 캐시 구성 또는 인라인 구성 중 원하는 구성의 프로세서 코어 112 에 캐시 메모리 서브시스템 114 를 결합할 수도 있다. 더 나아가 캐시 메모리 서브시스템 114 를 캐시 계층 구조로 구현할 수 있다. 원하면 계층 구조 내에서 프로세서 코어 112 에 더 가까운 캐시를 프로세서 코어 112 에 통합할 수도 있다. 한 실시 예에서, 캐시 메모리 서브시스템 114 각각은 L2 캐시 구조를 나타내고, 공유 캐시 서브시스템 118 은 L3 캐시 구조를 나타낸다.

[0026] 캐시 메모리 서브시스템 114와 공유 캐시 메모리 서브시스템 118 에는 해당 캐시 컨트롤러에 결합된 캐시 메모리가 포함될 수 있다. 프로세서 코어 112 에는 미리 정의한 범용 명령어 세트에 따라 명령어를 실행하는 회로가 포함된다. 예를 들어 x86 명령어 세트 아키텍처를 선택할 수 있다. 그렇지 않으면 Alpha, PowerPC 또는 기타 범용 명령어 세트 아키텍처를 선택할 수도 있다. 일반적으로 프로세서 코어 112 는 각각 데이터와 명령어를 위해 캐시 메모리 서브시스템에 액세스한다. 요청된 블록을 캐시 메모리 서브시스템 114 또는 공유 캐시 메모리 서브시스템 118 에서 찾지 못하면, 읽기 요청을 생성하여 누락 블록이 매핑되는 위치를 경유하여 메모리 컨트롤러 120 으로 전송한다.

[0027] 도2 는 비순차적 실행을 수행하는 범용 프로세서 코어 200 의 실시 예를 보여준다. 명령어 캐시 (i-cache) 와 해당 TLB(Translation Lookaside Buffer) 202 는 명령어에 액세스하기 위하여 소프트웨어 응용 프로그램에 대

한 명령어와 주소를 저장할 수 있다. 캐시 202 와 222 는 모두 도 1의 캐시 메모리 서브시스템 114 에서와 같이 외부에 배치하거나 프로세서 코어 200 내에 배치하거나 할 수 있다. 명령어 페치 장치 (Instruction Fetch Unit, IFU) 204 는 i-cache 손실이 없을 때 i-cache 202 로부터 나오는 여러 명령어를 클럭 사이클 당 페치할 수 있다. IFU 204 에는 i-cache 202 에 페치될 다음 명령어의 주소를 가리키는 포인터가 있는 프로그램 카운터가 포함될 수 있다. 이 주소는 i-TLB 의 주소에 비교될 수 있다. 또한 나중에 파이프라인 단계에서 실체 출력을 결정하는 실행 장치에 앞서 조건 명령어의 출력을 예측하는 분기 예측 장치도 IFU 204 에 포함될 수 있다.

[0028] 디코더 유닛 206 은 페치된 여러 명령어의 OP 코드를 디코딩하고, 재정렬 버퍼 208 등의 순차 중지 큐에 항목을 할당할 수 있다. 항목들은 재정렬 버퍼 208 내에서 할당 이후 정수 큐 212 와 부동 소수점 큐 216 에 할당될 수 있다. 로드/저장 장치 220 내의 입력 큐에 있는 항목들은 필요에 따라 실행 장치 210 내에서 할당 이후에 또는 동시에 할당될 수 있다. 실행 장치 210 및/또는 로드/저장 장치 220 에서 항목 할당은 디스패치로 간주된다. 실행 장치 210 에 대해서는 메모리 액세스에 대한 설명 이후에 설명한다.

[0029] 로드 및 저장 작업과 같은 메모리 액세스가 로드/저장 장치 220 으로 발행된다. 로드/저장 장치 220에는 메모리 액세스 명령어를 실행하기 위한 큐와 로직이 포함될 수 있다. 또한 로드 명령어가 올바른 최신 저장 명령어로부터 전달된 데이터를 수신하도록 하기 위한 검증 로직도 로드/저장 장치 220에 상주시킬 수 있다. 로드/저장 장치 220은 메모리 액세스 요청 224를 칩에 있는 하나 이상 수준의 데이터 캐시(d-cache) 222로 전송할 수 있다. 캐시의 각 수준마다 메모리 요청 224와 주소 비교를 위한 자체 TLB를 가질 수 있다. 캐시 222의 각 수준은 직렬 또는 병렬 방식으로 검색할 수 있다. 요청된 메모리 라인을 캐시 222에서 찾지 못하면 메모리 요청 224가 캐시 114나 118과 같이 낮은 수준 캐시 메모리로 또는 오프칩 메모리로 전송된다. 직렬 또는 병렬 검색, 가능한 다른 메모리 요청, 요청된 메모리 라인의 도달 대기에는 상당한 수의 클럭 사이클이 요구될 수 있다.

[0030] 실행 장치 210과 로드/저장 장치 220에서 나오는 결과는 공통 데이터 버스 230에 나타날 수 있고, 재정렬 버퍼 208로 전송될 수 있다. 한 실시 예에서, 재정렬 버퍼 208은 프로그램 순서에 따라 명령의 순차 중지를 보장하는 선입선출(FIFO) 방식일 수 있다. 여기서 그 결과를 수신하는 명령어가 중지 대상으로 표시된다. 해당 명령어가 큐의 맨 앞에 있으면 결과를 해당 아키텍처 레지스터 파일로 전송시킬 수 있다. 큐 212 및 216은 각각 해당하는 아키텍처 레지스터 파일을 보유할 수 있다. 아키텍처 레지스터 파일은 프로세서 코어 220의 범용 레지스터의 아키텍처 상태를 보유할 수 있다. 그런 다음, 재정렬 버퍼에 있는 명령어가 순서대로 중지되고 큐의 맨 처음 포인터가 프로그램 순서의 다음 명령어를 가리키도록 조정된다.

[0031] 결과를 기다리는 명령어의 피연산자로 값을 전달하기 위하여 공통 데이터 버스 230의 결과를 실행 장치 210으로 전송할 수 있다. 예를 들어 산술 명령어가 이전 산술 명령어의 결과에 좌우되는 피연산자를 갖거나 로드 명령어에 기능 유닛 214의 주소 생성 장치(Address Generation Unit, AGU)에 의해 계산된 주소가 필요할 수 있다. 이러한 대기 명령어가 피연산자에 대한 값을 갖고 명령어 실행에 하드웨어 리소스를 사용할 수 있을 때, 큐 212 및 216 각각으로부터 기능 유닛 214와 216에 있는 적절한 리소스 또는 로드/저장 장치 220으로 비순차적으로 발행될 수 있다.

[0032] 커밋되지 않거나 중지되지 않은 메모리 액세스 명령어는 로드/저장 유닛 220에 항목들을 갖는다. 커밋되지 않은 최신의 이전 명령어로부터 진행 중 또는 커밋되지 않은 로드 명령어의 전달된 데이터 값은 공통 데이터 버스 230에 배치되거나 간단히 로드/저장 유닛 220 내 로드 버퍼에 있는 해당 항목으로 라우팅될 수 있다.

[0033] 실행 장치 210으로 복귀될 때 큐 212 및 216은 피연산자의 사용이 가능하게 되길 기다리는 해당 정수 및 부동 소수점 명령어를 저장할 수 있다. 레지스터 이름 명령과 실행 일정 예약은 큐 212 및 216 내에서 또는 표시되지 않은 주변의 해당 회로 내에서 일어날 수 있다. 피연산자와 하드웨어 리소스를 모두 사용할 수 있게 되면, 큐 212 및 216 각각으로부터 정수 기능 유닛 214, 부동 소수점 기능 유닛 218 및/또는 로드/저장 장치 220으로 비순차적 명령이 발행될 수 있다. 각 정수 기능 유닛 214 세트에는 더하기, 빼기 등의 정수 계산, 주소 생성 및 분기 조건 명령어의 결과 판별을 위한 산술 논리 연산장치(Arithmetic Logic Unit, ALU)가 포함될 수 있다. 부동 소수점 기능 유닛 218에는 부동 소수점 더하기, 빼기, 곱하기, 나누기, 제곱근 계산, 정수를 부동 소수점으로 변환, 부동 소수점을 정수로 변환 등의 같은 연산 작업을 수행하기 위한 회로가 포함될 수 있다.

[0034] 한 실시 예에서, 마이크로프로세서 명령어 처리량을 증가시키기 위해 여러 정수 기능 유닛 214를 인스턴스화할 수 있다. 예를 들어 실행의 소프트웨어 스레드를 여러 개 실행하도록 프로세서 코어 200을 구성할 수 있다. 마이크로프로세서 100용 운영 체제와 해당 프로세서 코어 200이 소프트웨어 응용 프로그램용 메모리 영역을 할당한다. 컴파일된 소프트웨어 응용 프로그램은 프로세스 여러 개로 구성될 수 있다. 이러한 실시 예에서 프로세스마다 메모리 이미지 등의 자체 리소스 또는 응용 프로그램 실행 전 명령어 인스턴스 및 데이터를 보유할 수 있

다. 또한, 각 프로세스는 코드, 데이터, 힙 및 스택의 주소를 지정하는 주소 공간과 같은 프로세스 관련 정보, 데이터의 변수, 스택 포인터, 일반 및 부동 소수점 레지스터 등의 제어 레지스터, 프로그램 카운터 및 기타, 그리고 stdin, stdout 등의 운영 체제 서술자, 프로세서 소유자, 프로세스 권한 세트와 같은 보안 속성들로 구성될 수 있다.

[0035] 소프트웨어 응용 프로그램의 각 프로세스는 소프트웨어 스레드로 세분할 수 있다. 많은 최신 마이크로프로세서는 두 개 이상의 소프트웨어 스레드를 동시에 실행하도록 구성된다. 이러한 마이크로프로세서는 운영 체제 커널에서 병렬 실행 및 할당에 사용할 수 있는 하드웨어 스레드 또는 스트랜드를 두 개 이상 가질 수 있다. 예를 들어 멀티스레드 마이크로프로세서의 경우, 각 정수 기능 유닛 214가 유닛 내에서 하드웨어 리소스의 가용성에 따라 클럭 사이클 당 특정 스레드의 정수 명령어를 두 개 이상 수신할 수 있다. 각 장치의 하드웨어 복제 없이 프로세서 코어 200의 스트랜드 두 개 이상을 관리하도록 명령어 페치 장치 204부터 정수 큐 212 사이 회로를 수정할 수 있다. 예를 들어 스레드의 아키텍처 상태를 보유하고 있는 레지스터는 복제할 수 있지만 실행 하드웨어 리소스는 복제할 수 없다.

[0036] 한 실시 예에서, 프로세서 코어 200의 멀티스레드 실행은 클러스터 한 개가 단일 스레드 실행에 사용되는 방식의 전체 클러스터 복사본의 인스턴스화를 포함할 수 있고, 다른 구현에서는 디코더 유닛 206부터 명령어 기능 유닛 214 사이 회로로 구성된다. 이러한 구현에 사용된 회로를 정수 실행 클러스터라고 할 수 있다. 또 다른 실시 예에서, 부동 소수점 보조 프로 세서에는 디코더 유닛 206부터 부동 소수점 기능 유닛 218 사이 회로가 포함될 수 있다. 이러한 부동 소수점 보조 프로세서 인스턴스화 개수는 정수 실행 클러스터의 인스턴스화 개수보다 작을 수 있다. 이러한 실시 예에서, 각 부동 소수점 보조 프로세서는 두 개 이상의 스레드에 해당되는 반면 각 정수 실행 클러스터는 한 개의 다른 스레드에 해당된다. 명령어 페치 장치 204에는 코어 200의 여러 스레드를 관리하고 특정 스레드의 사용 가능한 해당 명령어를 한 개 이상 각 클러스터에 공급하는 회로가 포함된다. 이 경우에 각 클러스터는 스트랜드 또는 하드웨어 스레드이다.

[0037] 회로 로직 관점에서 볼 때 정수 기능 유닛 214의 각 복사본을 수반하기 위해서 부동 소수점 기능 유닛 218의 여러 복사본 인스턴스화가 좋은 방법일 수 있지만 하드웨어 구현 관점에서는 그리 바람직하지 않을 수 있다. 예를 들어 온다이 공간을 많이 점유하는 복잡한 로직이 부동 소수점 기능 유닛(FPU)에 포함될 수 있다. 또한, 부동 소수점 연산이 때로 프로그램 코드에서 수행되지 않기도 한다. 그러므로 하드웨어 설계자가 다이에 부동 소수점 기능 유닛 218의 값비싼 독립적 복사본을 생성하는 것은 바람직하지 않다.

[0038] 마이크로프로세서 100과 해당하는 프로세서 코어 200은 연산의 안정적 실행 모드(REM)로 구현될 수 있다. 중요 업무용 소프트웨어 응용 프로그램을 실행하는 동안 하드웨어 소프트 오류 등과 같은 실행 오류의 발생을 검출하는 데 이 모드를 이용할 수 있다. 한 실시 예에서, 독립적 정수 실행 클러스터 2개 또는 독립적 정수 기능 유닛에서 독립적인 소프트웨어 스레드 실행 2개를 지원하기 위하여 프로세서 코어 200의 부동 소수점 큐 216과 부동 소수점 기능 유닛 218은 동시 멀티스레딩(SMT) 모드로 작동할 수 있다. 여기서 두 가지 구현은 독립적인 스트랜드이다. 한 실시 예에서 IFU 204를 두 개의 스트랜드로 명령어를 공급하는 SMT 회로로 구현할 수 있다. 디코더 유닛 206부터 정수 기능 유닛 214 사이 회로를 다이에 하나의 정수 스트랜드로 인스턴스화할 수 있다. 첫째 스트랜드의 복사본인 둘째 정수 스트랜드도 다이에 인스턴스화할 수 있다.

[0039] 두 개의 정수 스트랜드는 IFU 204의 출력부터 각각의 정수 기능 유닛 214의 출력 사이에서 록스텝 방식으로 작동할 수 있다. 예를 들어 소프트웨어 스레드의 명령어가 첫째 정수 스트랜드로 제공되고 이 스레드의 복사본이 둘째 정수 스트랜드로 제공되면, 두 스트랜드의 각 파이프라인이 각각의 클럭 사이클 끝에서 동일한 결과를 출력할 것이다. 각 클럭 사이클 끝에서 비교를 수행하기 보다 여러 검사점 명령어를 프로그램 코드에 삽입하는 방법이 있다. 각 검사점 동안, 두 개의 스트랜드에 대해 지정된 레지스터들의 내용 비교를 수행할 수 있다. 해당하는 값들이 일치하면 소프트 오류가 검출되지 않으므로 실행을 계속할 수 있다. 한 실시 예에서, 불일치가 발견되면 성공한 마지막 검사점으로 실행을 롤백하고 프로그램 코드에서 이 위치부터 실행을 진행할 수 있다. 한 구현에서 운영 체제(OS)로 경고를 보고할 수 있다. 다른 실시 예에서는 사용자가 자세히 확인할 수 있도록 메모리에 미리 정한 위치에서 로그 파일에 기록할 수 있다.

[0040] 또 다른 실시 예에서는 디코더 유닛 206부터 정수 큐 212 사이 회로를 복사하지 않고 SMT 장치로 구현할 수 있다. 정수 기능 유닛 214는 복사할 수 있다. 다시 한 번, 한 실시 예에서 두 개의 정수 스트랜드 실행은 IFU 204의 출력부터 각각의 정수 기능 유닛 214의 출력 사이에서 록스텝 방식으로 작동할 수 있다. 검사와 비교는 위에서 설명한 대로 수행될 수 있다. 스트랜드를 두 개 사용하는 것은 설명을 위해서이며, 더 많은 스트랜드 이용이 가능하며 고려되고 있다. 예를 들어 다른 실시 예에서, 위에서 프로세스에 관해 설명한 대로 하드웨어 스레드

또는 스트랜드를 두 개 이상 필요로 하는 두 개 이상의 소프트웨어 스레드를 실행하도록 마이크로프로세서의 회로를 구성할 수 있다. 따라서 한 구현에서 세 개의 정수 스트랜드 실행은 IFU 204의 출력부터 각각의 정수 기능 유닛 214의 출력 사이에서 록스텝 방식으로 작동할 수 있다.

[0041] 위에 설명된 어느 실시 예에서든 부동 소수점 큐 216, 부동 소수점 기능 유닛 218, 로드/저장 장치(LSU) 220 및 캐시 222는 별도로 인스턴스화되고 정수 큐 212와 인스턴스화된 두 개의 별도 정수 기능 유닛 214로부터 입력을 수신할 수 있다. 위에서 설명한 대로, 복사본 두 개를 각 정수 스트랜드에서 한 개씩 록스텝 방식으로 실행함으로써 높은 안정성을 갖고 한 소프트웨어 응용 프로그램을 실행할 수 있다. 그러나 부동 소수점 큐 216과 부동 소수점 기능 유닛 218에는 이 기법을 적용할 수 없다. 그 이유는 여러 다른 스레드로부터 작업이 록스텝 방식으로 실행된다는 보장이 없기 때문이다. 부동 소수점 기능 유닛 218과 같은 칩의 SMT 부분에는 두 개의 소프트웨어 스레드로부터 보내는 명령어를 동시에 실행할 하드웨어 자원이 충분하지 않을 수 있다. 따라서 두 개의 소프트웨어 스레드로부터 보내는 명령어가 록스텝이 아닌 비동기식으로 실행될 수 있다.

[0042] 이 문제에 대한 한 가지 해결책은 SMT 부동 소수점 회로에서 두 소프트웨어 스레드를 록스텝 방식으로 실행시키는 것이다. 그러나 이 방법은 복잡성, 온다이 공간, 추가적인 온다이 배선 라우팅에서 엄청난 비용을 치르며, REM 연산이 필요하지 않을 때 성능이 떨어지기도 한다. 또 다른 해결책은 두 소프트웨어 스레드 모두의 부동 소수점 명령어가 부동 소수점 기능 유닛 218에서 독립적으로 실행되어 비동기식으로 결과를 산출할 수 있도록 하는 방법이다. 그런 다음 두 소프트웨어 스레드 중 나중 스레드가 결과를 계산할 때까지 각 작업에 대해 두 소프트웨어 스레드의 결과에 대한 교차 검사를 지연시켜야 한다. 지연 정도가 수백 클럭 사이클 등과 같이 매우 길며 코드 스트림의 모든 작업을 검사해야 하기 때문에 설계자는 모두 안전하게 검사할 수 있을 때까지 두 스레드의 결과를 캡처할 매우 큰 버퍼를 할당해야 한다.

[0043] 대신에 거의 대부분의 시간에 SMT 부동 소수점 회로에서 두 개의 "별도" 소프트웨어 스레드(즉, 원본 소프트웨어 스레드와 그 복사본)의 원래의 명령어와 그 복사본의 비동기식 실행을 허용하는 기법을 REM 연산에 대해 사용할 수 있다. 부동 소수점 도메인을 걸쳐 결과 전달을 시도할 때에만 두 스레드가 동기화 및 비교되어 차이를 확인하므로 두 정수 클러스터에 동기화된 자극 전달이 보장된다. 그 결과 정수 클러스터가 부동 소수점 회로에서 나오는 모든 통신에서 두 스레드로부터 동일한 응답을 보게 되므로 록스텝 방식으로 유지될 수 있다.

[0044] 이제 도3A을 참조하면, 동시 멀티스레딩 프로세서에서 효율적이며 안정적 실행을 위한 실시 예로 방법 300이 제시된다. 설명 목적으로 이 실시 예의 단계들을 순차적으로 표시한다. 그러나 일부 단계는 도면에 나온 것과 다른 순서로 일어날 수도 있고, 또 일부 단계는 동시에 수행되기도 하고, 일부 단계는 다른 단계에 결합될 수도 있고, 일부 단계는 다른 실시 예에서 빠지기도 한다. 블록 302에서 마이크로프로세서 내의 멀티스레딩 프로세서 코어가 안정적 실행 모드(REM)로 설정된다. 이 작업은 프로그램 코드 명령어, 시스템 마이크로코드 명령어, 지정된 핀에서 외부 제어에 의해 인가되거나 인가 취소되는 신호 또는 기타에 의해 수행된다.

[0045] 마이크로프로세서의 명령어 페치 장치가 블록 304에 지정된 i-cache의 주소에서 나온 명령어를 페치한다. 한 실시 예에서, REM 연산 동안 마이크로프로세서가 두 스트랜드 또는 하드웨어 스레드만의 실행을 허용할 수 있다. 페치된 각 명령어는 선택적으로 복제될 수 있다. 예를 들어 부동 소수점 회로(조건부 블록 306) 밖에서 통신하는 부동 소수점 명령어(예: 부동 소수점 저장 명령어, 부동 소수점 정수 전달 명령어 또는 기타)가 검출되지 않는 경우, 다른 연산의 이러한 명령어는 블록 312에 복제된다. 원래의 명령어는 첫째 소프트웨어 스레드의 명령어로 취급되어 첫째 스트랜드로 전송된다. 명령어 복사본은 둘째 소프트웨어 스레드의 명령어로 취급되어 블록 314의 둘째 스트랜드로 전송된다.

[0046] 명령어 검출과 명령어 복제는 모두 디코더 유닛 206, 재정렬 버퍼 208, 실행 큐 212 및 216 또는 설계자가 선택한 다른 프론트엔드 장치에서 수행될 수 있다. 이러한 부동 소수점 외부 작업을 위한 선행 디코딩 회로를 디코더 유닛 206 앞에 배치할 수 있다. 디코더 유닛 206부터 정수 기능 유닛 214 사이 회로가 통합된 코어 200에 정수 코어가 복사되는 경우에 이 기법을 사용할 수 있다. 또는 디코더 유닛 206의 출력 신호를 이러한 작업을 검출하는 데 사용하고, 복제는 디코딩 이후에 진행될 수 있다. 디코더 유닛 206부터 실행 큐 212 및 216 사이 회로가 SMT 회로로 구현될 때 이 기법을 사용할 수 있다. 이어서, 원래의 명령어와 복제 명령어는 모두 블록 314의 별도 스트랜드에 할당되고, 블록 316에서 실행을 위해 이름을 바꾸고 일정이 예정된다.

[0047] 그러나 부동 소수점 회로(조건부 블록 306) 밖에서 통신하는 부동 소수점 명령어(예: 부동 소수점 저장 명령어, 부동 소수점 정수 전달 명령어 또는 기타)가 검출되면, 이 부동 소수점 외부 연산 또는 명령어가 복제되지 않을 수 있다. 이러한 명령어는 부동 소수점 기능 유닛 218부터 정수 기능 유닛 214 간 통신 경로 또는 회선 라우트를 필요로 한다. 기능 유닛 218에 부동 소수점 로직 복사본이 없기 때문에 이들 명령어로 인하여 모든 기능 유

닛 214 및 218에서 록스텝 실행이 차단될 것이다. 따라서 이 명령어 유형이 블록 308에서 새로운 명령어 유형으로 변환될 수 있다. 그런 다음, 복제되지 않는 새 명령어 유형이 사용 가능한 스트랜드 중 하나에 할당된다. 예를 들어 새 명령어 유형이 원본 소프트웨어 스레드의 명령어로 취급되어 프로세서 200의 해당 스트랜드에 할당될 수 있다.

[0048] 마이크로프로세서 내에 두 스트랜드를 사용하는 한 실시 예에서, 첫째 스트랜드에 32개의 부동 소수점(FP) 레지스터 32개(f0부터 f31까지)가 있고 둘째 스트랜드도 유사한 경우를 예로 들 수 있다. 큐 216과 장치 218의 회로로 구성되거나 디코더 유닛 206부터 큐 216 사이 작업용 자체 회로로 구성될 수 있는 FPU가 두 스트랜드의 상태를 유지하기 위해 필요할 수 있다. FPU에는 두 스트랜드의 확약된 아키텍처 상태를 보유하는 논리 레지스터가 64개(예: L0부터 L63까지) 이상 필요할 수 있다. L0-L63에서 Strand0.FP0-Strand0.FP31 및 Strand0.FP1-Strand0.FP31로 매핑은 임의 방식일 수 있다. 한 실시 예에서, Strand0.FP0에 L0 할당과 더불어 Strand0.FP31에 L31 할당, 그리고 Strand1.FP0에 L32 할당과 더불어 Strand1.FP31에 L63 할당이 매핑에 포함될 수 있다.

[0049] 비 SMT 기계에서는 L32~ L63의 할당이 없을 수 있다. 이제 한 실시 예에서, 발송 스트랜드(Strand0 또는 Strand1)에 따라서 FMUL L0, L1 --> L2 또는 FMUL L32, L33 --> L34 중 하나에만 FMUL FP0, FP1 --> FP2와 같은 간단한 FP 연산을 매핑할 수 있다. REM/록스텝 모드에서 소프트웨어 스레드 실행이 한 개뿐인데 이 소프트웨어 스레드가 하드웨어 스트랜드 Strand0과 Strand1 모두에서 실행되도록 할 수 있다. 위에 나온 부동 소수점 곱하기, FMUL, 명령어를 FMUL L0, L1 --> L2 및 FMUL L32, L33 -> L34에 매핑하면 된다. 이는 이미 설명한 복제 작업이다.

[0050] 따라서 FPU 프론트엔드에서 위의 매핑을 수행하고 나면, 논리 레지스터 수가 모든 종속물을 전달하기 충분하기 때문에 여러 작업 사이에 종속성을 실현하기 위해 작업에 대한 스트랜드 ID를 구분할 필요가 없다. 한 실시 예에서, Strand0에서 나온 작업은 L0 ~ L31 레지스터만 참조하고, Strand1에서 나온 작업은 L32 ~ L63 레지스터만 참조한다. 비 REM 모드에서는 두 세트 간 혼합이 가능하지 않다. 그래서 작업이 매핑된 후에는 마이크로프로세서가 록스텝 모드로 작동하는지 여부에 관계없이 작업에 대해 스트랜드 ID가 필요하지 않을 수 있다.

[0051] REM 모드의 한 실시 예에서, 복제 작업이 수행된 후에 복제된 두 작업을 생성할 수 있고, 두 작업 모두 일정 예약 목적으로 스트랜드 ID를 갖지 않는다. 스트랜드 ID가 필요하지만 다른 목적용이다.

[0052] 한 실시 예에서, FSTO FP0 --> [mem]([mem]은 대상 메모리 위치를 가리킴) 등의 부동 소수점 저장 명령어와 같은 단항 작업은 복제되지 않을 수 있다. 대신에 FCMPSTO L0, L32 --> [mem] 형태의 이진 논리 명령어 한 개로 단항 명령어를 매핑할 수 있다. 이 논리 명령어에는 할당된 스트랜드 ID가 없을 수 있다. 그 보다는 모든 스트랜드 ID가 할당될 수 있는데, 그 이유는 사용 가능한 모든 스트랜드로부터 유도된 논리 레지스터들의 혼합 세트를 사용하기 때문이다.

[0053] 이와 같이 특이한 매핑 명령어는 두 스트랜드로부터 소스 피연산자를 가져오고 L0과 L32의 마지막 생성자 명령어에 종속될 수 있다. 정의에 따르면 한 실시 예에서 생성자 L0과 L32는 각각 Strand0과 Strand1에서 비롯된다. 두 생성자는 부동 소수점 곱하기 명령어 FMUL와 유사한 작업에서 복제될 수 있다. 복제된 작업들은 L0와 L32가 모두 생성되고 서로 경쟁하지 않을 때와 같이 서로 다른 시기에 실행될 수 있다. 그런 다음, 실행 목적으로 FCMPSTO의 일정 예약을 진행할 수 있다.

[0054] 더 구체적인 예로, 부동 소수점 외부 작업은 다음과 같은 부동 소수점 저장 명령어일 수 있다.

[0055] FDIV FP_Reg_Result, FP_Reg_Src1, FP_Reg_Src2 /* 라인 1 */

[0056] STF <address>, FP_Reg_Result /* 라인 2 */

[0057] 한 실시 예에서, 위 라인 1(첫째 줄)의 FDIV(부동 소수점 나누기 명령어)는 FP_Reg_Src1에 의해 지정된 부동 소수점 레지스터의 64비트 배정밀도 부동 소수점 소스 피연산자와 FP_Reg_Src2에 의해 지정된 부동 소수점 레지스터의 64비트 배정밀도 부동 소수점 소스 피연산자를 나눌 수 있다. 나머지는 보존되지 않을 수 있다. 프로세서 200이 REM 모드로 작동될 때 FDIV 명령어가 복제될 수 있다. 원래 FDIV 명령어는 스트랜드 0에 할당되고 복제된 FDIV 명령어는 스트랜드 1에 지정될 수 있다.

[0058] 라인 2(둘째 줄)의 STF(부동 소수점 저장 명령어)는 원본 부동 소수점 레지스터 FP_Reg_Result에서 지정된 유효 주소 <address>로 미리 정해진 바이트 수를 저장할 수 있다. 정수 전달 작업과 마찬가지로 부동 소수점 저장 작업은 단항 연산 작업이므로 소스 피연산자가 한 개뿐이다. 이러한 특수한 단항 작업은 두 아키텍처 원본(첫째 스레드에서 나온 원본 논리 레지스터 번호와 둘째 스레드에서 나온 동일한 논리 레지스터 번호)을 갖는 새로운

명령어 유형을 생성하여 단항에서 이진 작업으로 변환시킬 수 있다. 예를 들어 프로세서 코어 200 작업이 REM 모드에서 두 개의 스트랜드만 실행하는 경우, 위 부동 소수점 저장 명령어는 다음과 같이 변환된다.

[0059] STF <address>, FP_Reg_Result, FP_Reg_Result /* 라인 3 */

[0060] <strand 0> <strand 1>

[0061] 따라서 부동 소수점 저장 명령어가 부동 소수점 저장-비교 명령어로 변환된다. 이름 변경 레지스터 파일 회로는 재정렬 버퍼 208 또는 실행 큐 212 및 216에 상주할 수 있다. 한 실시 예에서, 매핑 장치가 명령 레지스터 번호(Instruction Register Number, IRN)를 논리 레지스터 번호(Logical Register Number, LRN)로 변환할 수 있다. 그런 다음, 변경 레지스터 파일이 물리적 레지스터 번호(Physical Register Number, PRN)에 LRN을 매핑할 수 있다. 이때, 물리적 레지스터 수는 IRN의 직접 사용에 의해 주소 지정 가능한 수보다 크다. 위 라인 3에서 변환된 작업의 피연산자 FP_Reg_Result <strand 0> 및 FP_Reg_Result <strand 1>은 평상시 대로 PRN에 매핑된 후, 부동 소수점 큐 216 내 부동 소수점 스케줄러로 전송된다. 이 부동 소수점 외부 연산은 복제되지 않으며, 위 라인 3에 나온 명령어는 strand 0에 할당되고 비 op(no-op) 명령어는 방법 300의 블록 310의 strand 1에 할당될 수 있다.

[0062] 변환된 명령어는 당연히 프로세서 코어 200의 두 스트랜드에서 실행 중인 두 소프트웨어 스레드의 데이터 흐름 사이에서 동기화 점 또는 검사점의 기능을 수행한다. 실행 장치 210에서 수정되지 않은 부동 소수점 스케줄러 내에서 변환된 작업만이 유효하여 PRN 원본 2개가 모두 준비될 때 선택될 수 있다. 이는 부동 소수점 나누기 연산을 실행하는 일련의 연산 체인이 SMT 부동 소수점 기능 유닛 218의 정상적인 데이터 흐름 방식으로 실행을 완료한 후 변환된 작업의 두 원본 레지스터, 즉 FP_Reg_Result <strand 0> 및 FP_Reg_Result <strand 1>에 결과를 제공할 때 자연스럽게 발생할 수 있다.

[0063] 이제 도3B를 참조하면, 동시 멀티스레딩 프로세서에서 효율적이며 안정적 실행을 지속하기 위한 실시 예로 방법 330이 제시된다. 설명 목적으로 이 실시 예의 단계들을 순차적으로 표시한다. 그러나 일부 단계는 도면에 나온 것과 다른 순서로 일어날 수도 있고, 또 일부 단계는 동시에 수행되기도 하고, 일부 단계는 다른 단계에 결합될 수도 있고, 일부 단계는 다른 실시 예에서 빠지기도 한다. 프로세서 코어 200은 블록 332에서 REM 모드로 명령어를 실행하고 있다.

[0064] 중요 업무용 소프트웨어 응용 프로그램을 실행하는 동안 소프트 오류 등과 같은 오류가 발생했는지 주기적으로 확인하기 위해 프로그램 실행에서 동기화 점을 선택할 수 있다. 한 실시 예에서, 하나 이상의 특정 파이프라인 단계 또는 특정 조합 로직의 출력을 조건부 블록 334의 동기화 점으로 선택할 수 있다. 이 선택은 병렬 실행 스레드가 록스텝 방식으로 실행 중인 경우에만 가능할 수 있다. 록스텝 실행이 가능하면 특정 레지스터 출력, 조합 로직 출력 및/또는 기타 지정된 값과 같은 특정 레지스터 출력이 블록 340의 두 스트랜드 사이 각 클럭 사이클에서 비교된다. 한 실시 예에서, 더 잦은 검사점을 원할 경우에 소프트웨어 프로그램 스트림의 어느 위치에든 FMOV FP0 --> FP0 형태의 더미 단항 이동 명령을 삽입할 수 있다. 그러면 원본 프로그램 명령어 스트림에 정규 단항 저장이 없을 경우에도 임의 주기로 검사를 수행할 수 있는 이진 비교-이동 명령어로 FMOV 명령어가 변환될 수 있다.

[0065] 중요 업무용 소프트웨어 응용 프로그램의 명령어 복제와 한 실시 예에서 프로세서의 인스턴스화된 정수 클러스터 복사본들로 구성된 두 개의 별도 스트랜드에 원본 및 동일한 복사본이 놓임으로 정수 록스텝 실행이 가능해진다. 따라서 한 실시 예에서 각 정수 명령어 및 동일한 복사본이 동일한 클록 사이클 내에서 명령어 실행이 완료될 때 서로 비교된 각각의 결과를 가질 수 있다(조건부 블록 334). 이 경우에 명령어는 정수 명령어이다(조건부 블록 336).

[0066] 블록 340에서 비교를 통하여 한 실시 예에서 록스텝 실행으로 인해 동일한 클럭 사이클에서 결과가 동일해야 하는 두 스레드가 일치하지 않는 상황을 검출할 수 있다. 불일치가 발견되면(조건부 블록 342), REM 컨트롤러의 로직에 보고되며 블록 344에서 미리 정해진 후속 작업이 일어날 수 있다. 그러한 조치에는 마이크로프로세서에서 출력된 일정한 신호의 어설션, 응용 프로그램의 추가 실행 중단, 미리 정해진 버퍼에 분기 명령어의 주소 저장 및/또는 기타 조치가 포함될 수 있다.

[0067] 불일치가 발견되지 않으면(조건부 블록 342), 록스텝 방식으로 정상적으로 실행이 계속되며 방법 330의 제어 흐름이 제어 블록 B로 이동되었다가 방법 300의 블록 304로 복귀된다. 또 다른 실시 예에서, 소프트웨어 프로그램이나 컴파일러에 의해 프로그램 코드에 삽입될 수 있는 검사점 명령어를 사용하여 동기화 검사점(조건부 블록 334)을 제공할 수 있다. 한 실시 예에서, 모든 록스텝 실행 스레드에서 추적할 명령어 개수를 작게 유지하기 위

해서 코드의 모든 기본 블록 내에 그러한 명령어를 삽입할 수도 있다.

[0068] 또 다른 동기화 검사점은 변환된 부동 소수점 외부 연산(조건부 블록 336)의 실행일 수 있다. 실행하는 동안 프로세서의 프론트엔드, 즉 디코더 유닛 206 또는 재정렬 버퍼 208 등이 정상 처리량의 1/2에 해당하는 명령어를 부동 소수점 큐 216으로 전송할 수 있다. 이는 클럭 사이클 당 최대 두 개의 부동 소수점 명령어를 전송하거나 클럭 사이클을 하나씩 걸러서 최대 네 개의 부동 소수점 명령어를 전송하여 수행할 수 있다. 이 예에서 프로세서 코어 200은 4-wide 명령어 발행 기계로 간주한다. 다른 실시 예도 가능하며 고려되고 있다.

[0069] 방법 300의 블록 312에 대해 설명한 대로, 한 실시 예에서 프로세서 프론트-엔드의 로직이 이러한 부동 소수점 명령어를 선택적으로 복제할 수 있다. 이 로직이 부동 소수점 저장 및 정수 전달 작업과 같은 부동 소수점 외부 작업은 복제하지 않는다. 오히려 이러한 부동 소수점 명령어 유형은 위에서 방법 300의 블록 308에 대해 설명한 대로 변환된다.

[0070] 한 실시 예에서, 프로세서 코어 200 내의 두 스트랜드가 REM 연산 동안 사용된다. 한 실시 예에서, 원본 소프트웨어 스레드로부터 부동 소수점 명령어의 동일한 복사본을 생성하기 위해서 부동 소수점 외부 작업이 아닌 부동 소수점 명령어를 복제하는 로직을 프로세서 프론트엔드에 상주시킨다. 이때, 코어 200의 둘째 스트랜드에 있는 두 번째 소프트웨어 스레드에서 복사본이 실행된다. 두 명령어는 모두 부동 소수점 기능 유닛 218 내의 논리를 사용하여 실행되지만 하드웨어 리소스 부족으로 인해 동시에 실행되지는 않는다. 그러면 위 라인 3에 나온 것과 같은 변환된 부동 소수점 외부 명령어를 실행하는 동안, 블록 338에서 모든 소스 피연산자가 준비될 때 부동 소수점 작업에 해당하는 실행 장치 210 내의 다른 로직이 블록 340의 두 스트랜드에서 번호가 같은 레지스터의 내용을 비교할 수 있다. 불일치(조건부 블록 342)는 블록 344의 적절한 작업을 수행하도록 REM 컨트롤러에 플래그 지정된다.

[0071] REM 연산 도중, 부동 소수점 로직 216 및 218을 벗어나 통신하는 유일한 명령어인 복제되지 않고 변환된 부동 소수점 외부 연산을 실행하면 부동 소수점 로직 내에서 동시에 실행하고 록스텝 방식으로 정수 로직 212 및 214 신호를 발생시키는 것으로 나타난다. 다른 모든 부동 소수점 명령어는 실제로 다른 소프트웨어 스레드의 복제본 명령어와 관련하여 록스텝을 벗어난 방식으로 실행될 수 있다. 그 이유는 그러한 명령어는 정수 로직이 록스텝을 벗어난 방식으로 실행되게 만들 수 있는 부동 소수점 로직을 벗어난 스티뮬러스를 전송하기 때문이다. 변환된 부동 소수점 외부 명령어는 블록 338의 해당 소스 피연산자를 읽는다. 이 블록은 이미 SMT 코어에 존재하는 로직을 지원하며 수정이 필요하지 않다. 또한 정상 기능과 블록 340의 피연산자를 비교하여 균등한지 확인한다. 불일치(조건 블록 342) 사실은 블록 344의 REM 컨트롤러로 보고될 수 있다. 불일치(조건 블록 342)가 있는 경우 명령어는 단순히 동일 사이클의 정수 모두로 정보를 전송하므로, 정수 기능 유닛 214는 부동 소수점 로직의 록스텝 스티뮬러스만 인식하며 동기화 상태를 벗어나지 않는다.

[0072] 지금까지 설명한 구조를 살펴보면, 두 클러스터에서 중지하면 동기화 상태를 벗어나게 될 수 있다. 한 실시 예에서, 프로세서의 두 스트랜드에서 동작하는 두 소프트웨어 스레드의 교차 검사에는 중지 시 비교가 아닌 결과의 시그너처를 사용한 실행 후 곧바로 값을 비교하는 작업이 포함된다. 중지 시 비교를 수행하면 물리 레지스터 파일에 읽기 포트가 추가로 필요해진다. 그러나 지금까지 설명한 구조는 두 가지 상황에서 문제점을 제기한다.

[0073] 첫째는 동반 스레드에서 나타나지 않는 로직 오류 때문에 한 스레드에서 내부 예외가 발생할 가능성이다. 그러한 예외로 인해, 한 스레드에서는 발생하지 않는 리디렉션과 플러시가 다른 스레드에서는 유발된다. 이 상황은 두 스레드 모두의 리디렉션을 유발하는 프로세서 프론트 엔드 로직과 디스패치 로직에 로직을 추가하는 방식으로 처리할 수 있다.

[0074] 두 번째 상황은 비동기 외부 인터럽트이다. 그러한 인터럽트는 중지 상태와 관련하여 두 스레드 모두 동기 상태에 있지 않는 한 록스텝 방식으로 실행할 수 없다. 이를 위해 인스턴스화된 정수 로직이 복사하고 부동 소수점 로직이 두 스레드 모두에서 미해결 명령어를 모두 중지할 때까지 외부 인터럽트를 디스패치를 중지해야 한다. 이 동일한 메커니즘을 위에서 설명한 내부 예외 사례를 처리하는 데도 사용할 수 있다.

[0075] 다른 실시 예에서는 저성능 구조가 EPC를 통한 실행 유닛 대신 중지 큐로부터 명령어에 대한 완료 상태를 전송한다. 이 구조를 사용할 경우 EPC는 더 이상 클러스터 ROBS로 상태를 전송하지 않고 대신 상태를 FPRET 블록으로 전송한다. FPRET 내의 로직은 두 스레드 모두로부터 완료된 op를 추적한 다음 두 스레드 모두에서 실행을 완료한 모든 op에 대해 록스텝 방식으로 두 클러스터로 완료 상태를 전송한다.

[0076] 또 다른 실시 예에서, 프로세서 코어 200이 3개 이상의 사용 가능한 스트랜드 또는 하드웨어 스레드를 갖는 경우 부동 소수점 외부 연산의 변환이 단항 명령어를 제3의 명령어로 변환할 수도 있다. 라인 2에서 단항 명령어

를 변환하는 예가 아래에 다시 설명되어 있다.

[0077] FDIV FP_Reg_Result, FP_Reg_Src1, FP_Reg_Src2 /* 라인 1 */

[0078] STF <address>, FP_Reg_Result /* 라인 2 */

[0079] 프로세서 코어 200이 REM 연산 도중 세 개의 스트랜드를 연산하면 위 부동 소수점 저장 명령어가 다음과 같이 변환될 수 있다.

[0080] STF <address>, FP_Reg_Result, FP_Reg_Result, FP_Reg_Result /* 라인 4 */

[0081] <strand 0> <strand 1> <strand 2>

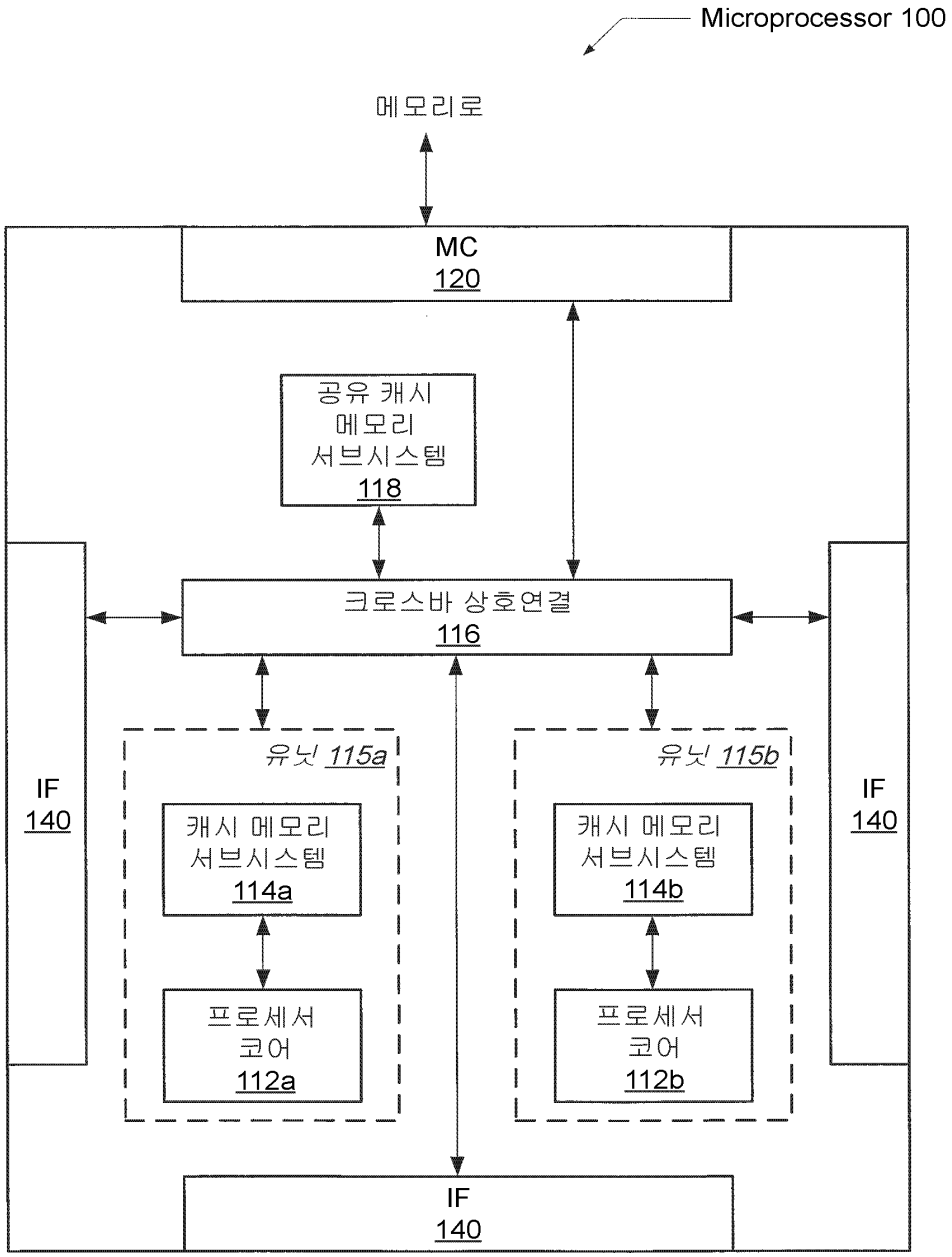
[0082] 따라서 부동 소수점 저장 명령어는 three-operand floating-point store-and-compare(세 피연산자 부동 소수점 저장 및 비교) 명령어로 변환된다. 이 실시 예의 이점은 비교 단계에 있는데, 두 개의 스트랜드(스트랜드 0과 1)의 값이 일치하지만 세 번째 스트랜드(예: 스트랜드 2)의 값이 불일치하는 경우 방법 330의 블록 344에서의 불일치를 보고하기 위해 적절한 신호가 REM 컨트롤러로 전송될 수 있지만 스트랜드 0과 1의 일치하는 값을 사용하여 실행은 계속될 수 있다. 부동 소수점 유닛 218은 이 일치 값을 인스턴스화된 정수 로직 212와 214로 전송할 수 있다. 그러나 세 가지 스트랜드 모두가 다른 값을 가지면 블록 344에서 REM 컨트롤러의 로직으로 보고되고 앞에서 설명한 것처럼 사전 결정된 후속 작업이 발생할 수 있다. 그러한 작업에는 마이크로프로세서의 특정 신호 출력 어설션, 애플리케이션의 추가 실행 중단, 미리 결정된 버퍼에 분기 명령어의 주소 저장 등이 포함될 수 있다.

[0083] 다양한 실시 예에 추가로 컴퓨터 액세스 가능한 매체에 위 설명에 따라 구현된 데이터와 명령어의 수신, 전송 또는 저장이 포함될 수 있다. 일반적으로, 컴퓨터 액세스 가능한 매체에는 자기 또는 광학 매체와 같은 스토리지 매체나 메모리 매체(예: 디스크 또는 DVD/CD-ROM), RAM(예: SDRAM, DDR, RDRAM, SRAM), ROM 등 휘발성 또는 비휘발성 매체가 포함될 수 있다. 컴퓨터 액세스 가능한 매체에는 또한 네트워크 및/또는 무선 링크와 같은 통신 매체를 통해 전송되는 전기, 전자기 또는 디지털 신호 등의 전송 매체나 신호도 포함될 수 있다.

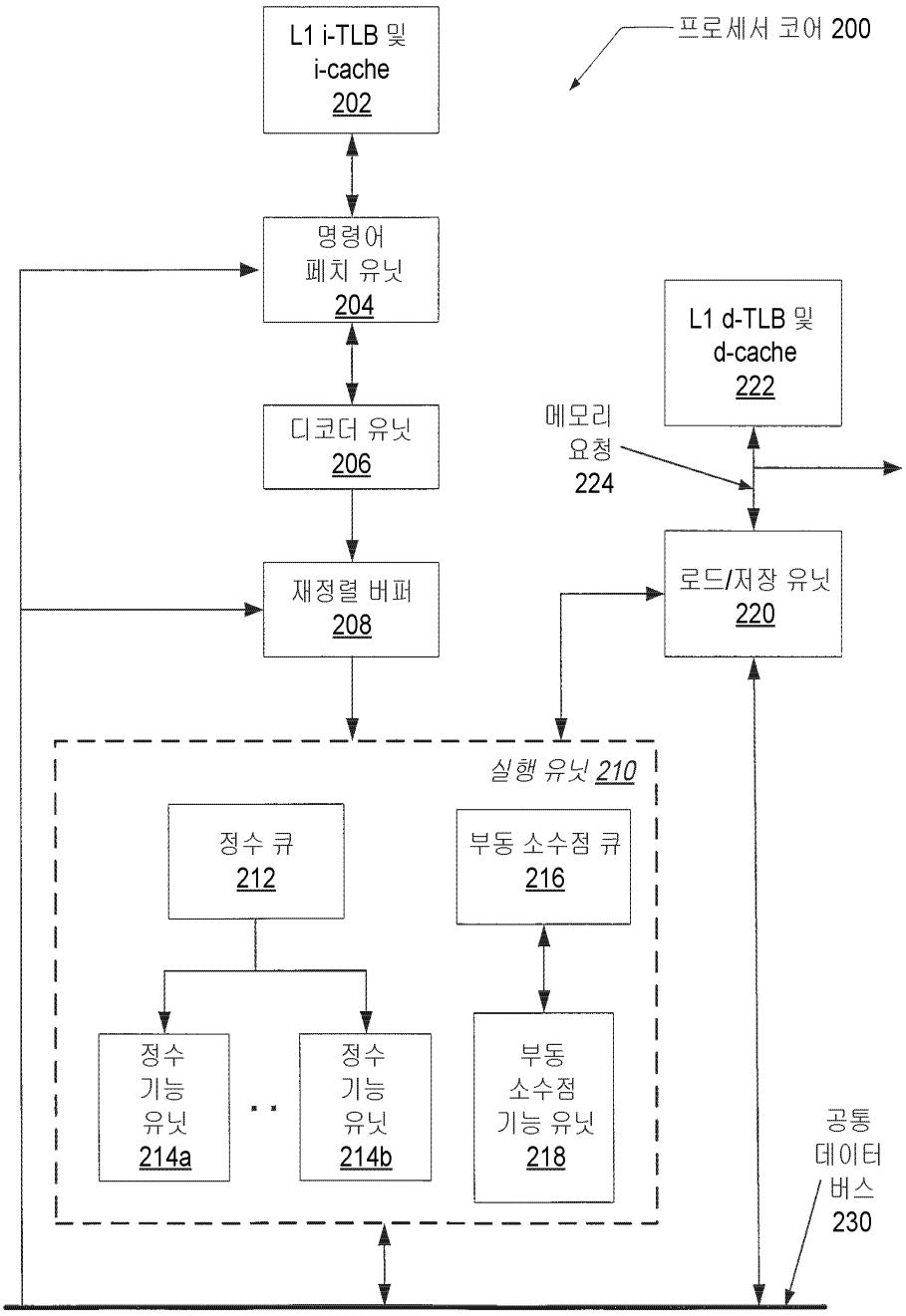
[0084] 비록, 실시예를 상당히 자세하게 설명했지만 본 개시내용이 완전히 이해되면 다양한 변형예들 및 수정예들이 해당 기술분야의 당업자에게 자명해질 것이다. 따라서 다음의 청구범위는 그러한 변형예와 수정예를 모두 포괄하는 것으로 해석되어야 한다.

도면

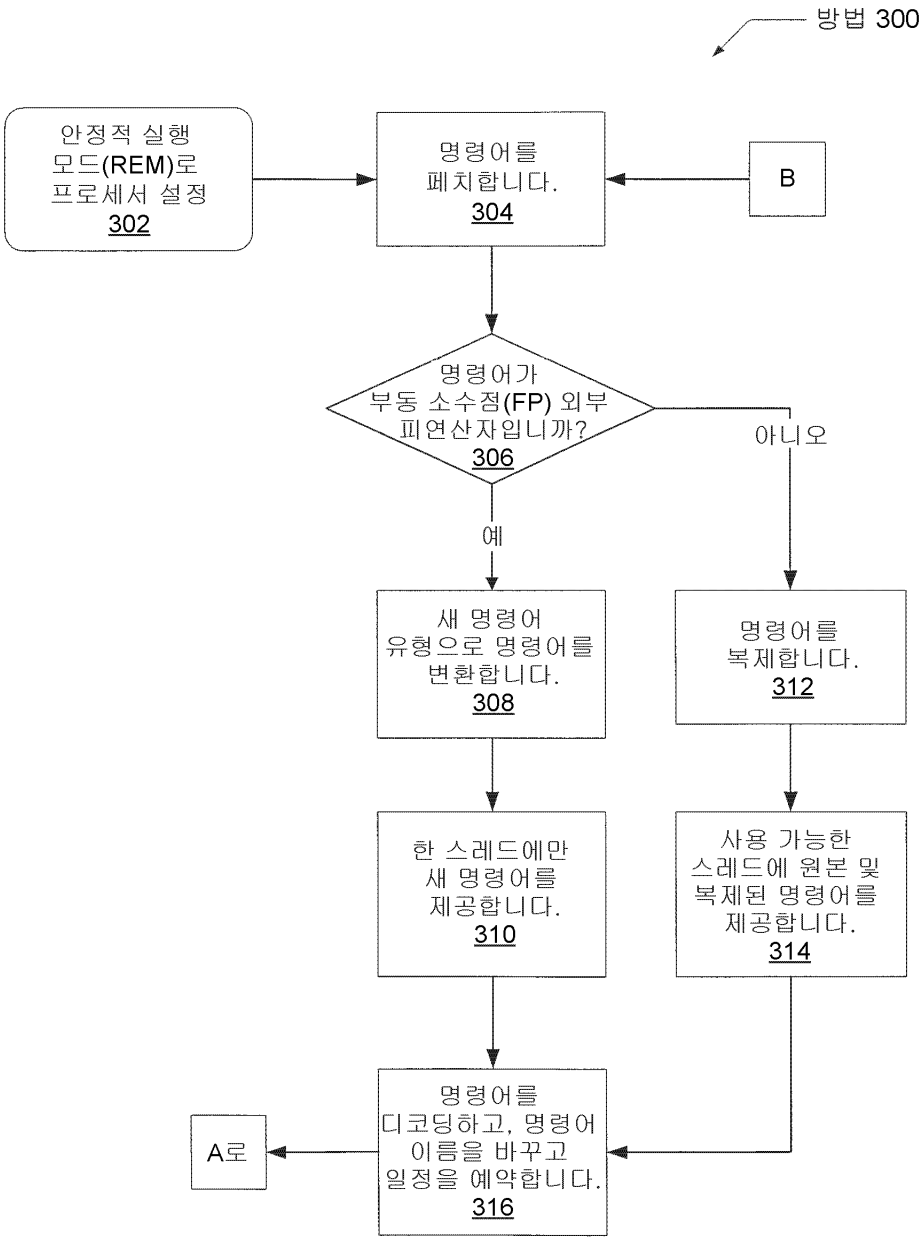
도면1


Microprocessor 100
메모리로
MC
120
공유 캐시
메모리
서브시스템
118
크로스바 상호연결
116
IF
140
유닛 115a
캐시 메모리
서브시스템
114a
프로세서
코어
112a
유닛 115b
캐시 메모리
서브시스템
114b
프로세서
코어
112b
IF
140
IF
140

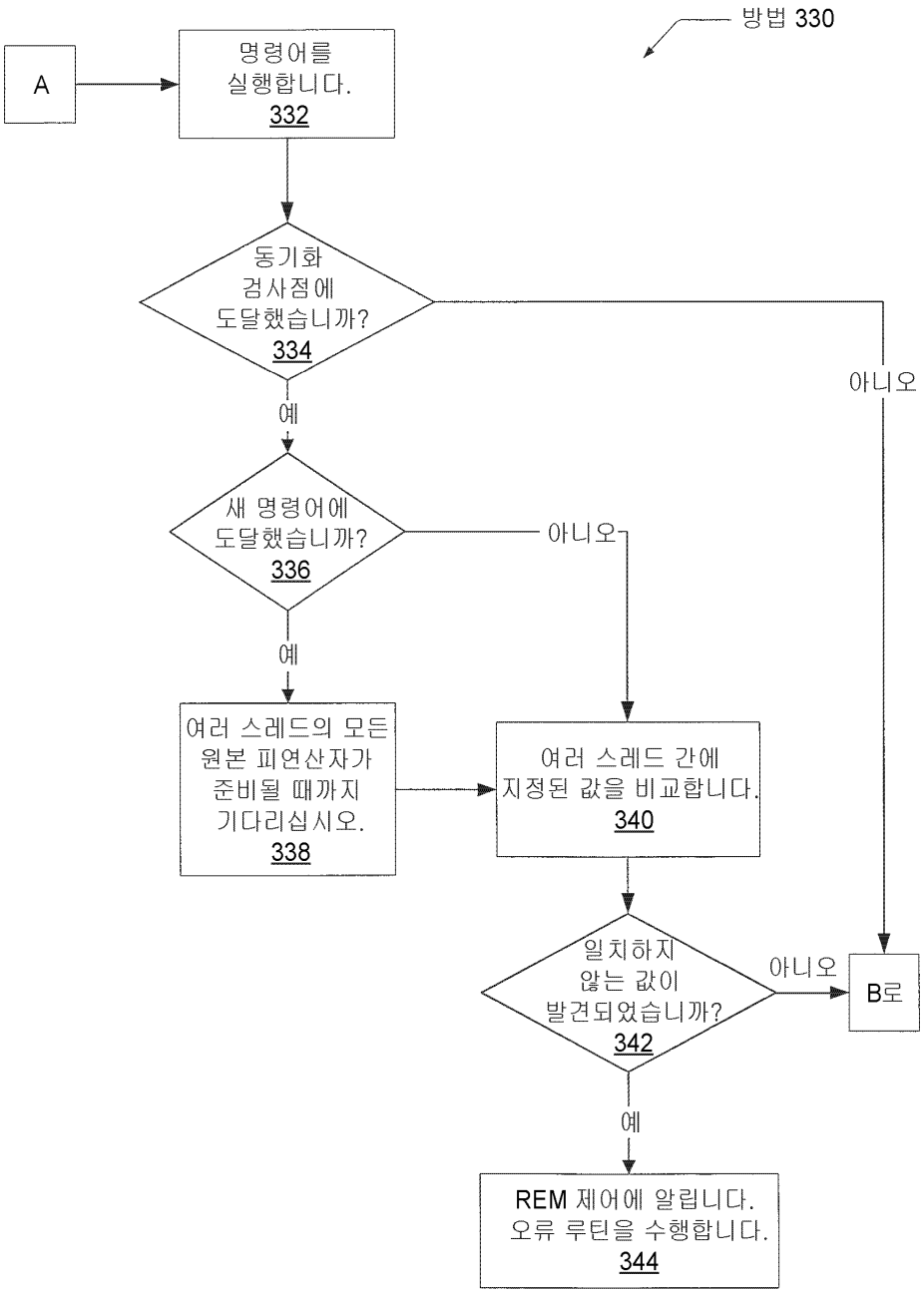
도면2


L1 i-TLB 및
i-cache
202
프로세서 코어 200
명령어
페치 유닛
204
디코더 유닛
206
재정렬 버퍼
208
L1 d-TLB 및
d-cache
222
메모리
요청
224
로드/저장 유닛
220
실행 유닛 210
정수 큐
212
부동 소수점 큐
216
정수
기능
유닛
214a
정수
기능
유닛
214b
부동
소수점
기능 유닛
218
공통
데이터
버스
230

도면3a


방법 300
안정적 실행 모드(REM)로 프로세서 설정 302
명령어를 페치합니다. 304
B
명령어가 부동 소수점(FP) 외부 피연산자입니까? 306
아니오
예
새 명령어 유형으로 명령어를 변환합니다. 308
명령어를 복제합니다. 312
한 스레드에만 새 명령어를 제공합니다. 310
사용 가능한 스레드에 원본 및 복제된 명령어를 제공합니다. 314
명령어를 디코딩하고, 명령어 이름을 바꾸고 일정을 예약합니다. 316
A로

도면3b


방법 330
A
명령어를
실행합니다.
332
동기화
검사점에
도달했습니까?
334
아니오
예
새 명령어에
도달했습니까?
336
아니오
예
여러 스레드의 모든
원본 피연산자가
준비될 때까지
기다리십시오.
338
여러 스레드 간에
지정된 값을 비교합니다.
340
일치하지
않는 값이
발견되었습니까?
342
아니오
B로
예
REM 제어에 알립니다.
오류 루틴을 수행합니다.
344