(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2019년05월17일
(11) 등록번호 10-1979697
(24) 등록일자 2019년05월13일

(51) 국제특허분류(Int. Cl.)
G06F 12/08 (2016.01)
(52) CPC특허분류
G06F 12/084 (2013.01)
G06F 12/0811 (2013.01)
(21) 출원번호 10-2016-7005327
(22) 출원일자(국제) 2014년10월03일
심사청구일자 2016년02월26일
(85) 번역문제출일자 2016년02월26일
(65) 공개번호 10-2016-0041950
(43) 공개일자 2016년04월18일
(86) 국제출원번호 PCT/US2014/059130
(87) 국제공개번호 WO 2015/048826
국제공개일자 2015년04월02일
(56) 선행기술조사문헌
US20070282928 A1*
US20080005504 A1*
US20090172284 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
인텔 코포레이션
미합중국 캘리포니아 95054 산타클라라 미션 칼리지 블러바드 2200
(72) 발명자
리우, 옌-정
미국 97229 오리건주 포틀랜드 노스웨스트 피리미노 애비뉴 5240
파힘, 바하
미국 95135 캘리포니아주 산 호세 비올레타 코트 412
(뒷면에 계속)
(74) 대리인
양영준, 김연송, 백만기

전체 청구항 수 : 총 21 항 심사관 : 임정묵

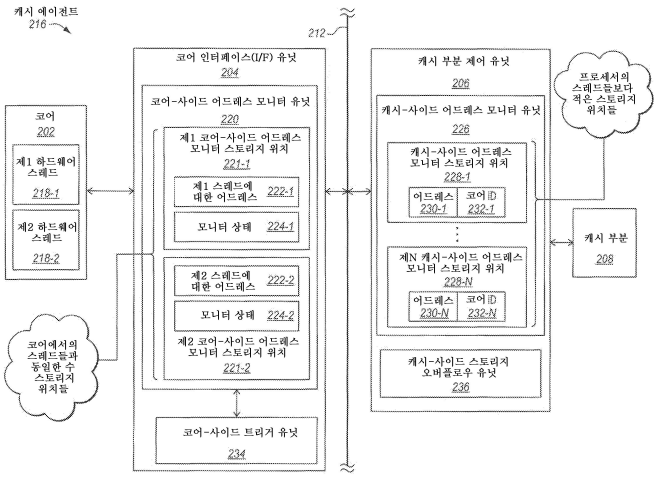
(54) 발명의 명칭 **어드레스로의 기입들을 모니터링하는 명령어를 구현하는 스케일가능형 메커니즘**

(57) 요 약

프로세서는, 분산형 캐시의 제1 캐시 부분에 대응하고, 프로세서의 논리 프로세서들의 총 수보다 적은 총 수의 캐시-사이드 어드레스 모니터 스토리지 위치들을 갖는 캐시-사이드 어드레스 모니터 유닛을 포함한다. 각각의 캐시-사이드 어드레스 모니터 스토리지 위치는 모니터링될 어드레스를 저장하기 위한 것이다. 코어-사이드 어드

(뒷면에 계속)

대 표 도



레스 모니터 유닛은 제1 코어에 대응하고, 제1 코어의 논리 프로세서들의 수와 동일한 수의 코어-사이드 어드레스 모니터 스토리지 위치들을 갖는다. 각각의 코어-사이드 어드레스 모니터 스토리지 위치는 어드레스 및 제1 코어의 상이한 대응 논리 프로세서에 대한 모니터 상태를 저장하기 위한 것이다. 캐시-사이드 어드레스 모니터 스토리지 오버플로우 유닛은, 제1 캐시 부분에 대응하고, 어떠한 미사용된 캐시-사이드 어드레스 모니터 스토리지 위치도 모니터링될 어드레스를 저장하는데 이용가능하지 않을 때 어드레스 모니터 스토리지 오버플로우 정책을 시행하기 위한 것이다.

(52) CPC특허분류

G06F 12/0833 (2013.01)

G06F 12/0846 (2013.01)

G06F 2212/1016 (2013.01)

(72) 발명자

할노르, 에릭 지.

미국 97007 오리건주 비버톤 사우스웨스트 166번 플레이스 6415

챔버레인, 제프리 디.

미국 95377 캘리포니아주 트레이시 케이프 브리톤 레인 465

반 도렌, 스테픈 알.

미국 97229 오리건주 포틀랜드 노스웨스트 바소로뮤 드라이브 8927

후안, 안토니오

스페인 이-08005 바르셀로나 2에이 1번 호안 미로 7

명 세 서

청구범위

청구항 1

프로세서로서,

적어도 일부의 회로를 포함하고, 분산형 캐시의 제1 캐시 부분에 대응하고 또한 연결되며, 상기 프로세서의 논리 프로세서들의 총 수보다 적은 총 수의 캐시-사이드 어드레스 모니터 스토리지 위치들을 갖는 캐시-사이드 어드레스 모니터 - 상기 분산형 캐시는 어드레스들의 중첩하지 않는 범위에 각각 매핑되는 동작 동안 복수의 캐시 부분을 포함하고, 각각의 캐시-사이드 어드레스 모니터 스토리지 위치는 상기 캐시-사이드 어드레스 모니터가 기입들을 모니터링하는 어드레스를 저장하고, 상기 캐시-사이드 어드레스 모니터 스토리지 위치들은 상기 분산형 캐시의 일부가 아님 -;

적어도 일부의 회로를 포함하고, 제1 코어에 대응하고 또한 연결되며, 상기 제1 코어의 하나 이상의 논리 프로세서들의 수와 동일한 수의 코어-사이드 어드레스 모니터 스토리지 위치들을 갖는 코어-사이드 어드레스 모니터 - 각각의 코어-사이드 어드레스 모니터 스토리지 위치는 상기 코어-사이드 어드레스 모니터가 기입들을 모니터링하는 어드레스 및 상기 제1 코어의 상이한 대응 논리 프로세서에 대한 모니터 상태를 저장함 -;

어떠한 미사용된 캐시-사이드 어드레스 모니터 스토리지 위치도 모니터링될 추가적 어드레스를 저장하는데 이용가능하지 않을 때 어드레스 모니터 스토리지 오버플로우 정책을 시행하는, 적어도 일부의 회로를 포함하고 상기 제1 캐시 부분에 대응하고 또한 연결되며 상기 캐시-사이드 어드레스 모니터에 연결되는 캐시-사이드 어드레스 모니터 스토리지 오버플로우 유닛; 및

적어도 일부의 회로를 포함하고, 상기 제1 코어에 대응하고 또한 연결되며 상기 코어-사이드 어드레스 모니터와 연결되는 코어-사이드 트리거 유닛

을 포함하고, 상기 코어-사이드 트리거 유닛은, 대응하는 코어-사이드 어드레스 모니터 스토리지 위치가 트리거할 준비가 된 모니터 상태를 갖고 트리거 이벤트가 검출될 때, 상기 제1 코어의 논리 프로세서를 트리거하는 프로세서.

청구항 2

삭제

청구항 3

제1항에 있어서,

적어도 일부의 회로를 포함하고 상기 캐시-사이드 어드레스 모니터와 연결되는, 동일한 모니터 어드레스에 대한 상이한 논리 프로세서들로부터의 모니터 요청들을 공통 캐시-사이드 어드레스 모니터 스토리지 위치에 기록하기 위한 캐시-사이드 어드레스 모니터 스토리지 위치 재사용 유닛을 더 포함하는 프로세서.

청구항 4

제3항에 있어서,

상기 공통 캐시-사이드 어드레스 모니터 스토리지 위치는 상기 동일한 모니터 어드레스에 대해 상기 모니터 요청들을 제공한 상기 상이한 논리 프로세서들을 기록하는 구조를 포함하는 프로세서.

청구항 5

제1항에 있어서,

상기 프로세서는 40개보다 많은 하드웨어 스레드들을 갖고, 상기 제1 캐시 부분에 대응하는 상기 캐시-사이드 어드레스 모니터의 상기 캐시-사이드 어드레스 모니터 스토리지 위치들의 총 수는 적어도 20개의 캐시-사이드 어드레스 모니터 스토리지 위치들이지만 상기 40개보다 많은 하드웨어 스레드들의 총 수보다 적은 프로세서.

청구항 6

삭제

청구항 7

제1항에 있어서,

모니터링될 제1 어드레스를 표시하는 명령어에 응답하여, 상기 캐시-사이드 어드레스 모니터는 상기 제1 어드레스를 캐시-사이드 어드레스 모니터 스토리지 위치에 저장하고, 상기 코어-사이드 어드레스 모니터는 상기 제1 어드레스를 코어-사이드 어드레스 모니터 스토리지 위치에 저장하는 프로세서.

청구항 8

제1항에 있어서,

상기 제1 코어의 상기 하나 이상의 논리 프로세서들은 하드웨어 스레드들을 포함하는 프로세서.

청구항 9

제1항에 있어서,

상기 캐시-사이드 어드레스 모니터 스토리지 오퍼플로우 유닛은 판독 트랜잭션들이 공유된 상태를 사용하도록 강요하는 것을 포함하는 상기 어드레스 모니터 스토리지 오버플로우 정책을 시행하는 프로세서.

청구항 10

제1항에 있어서,

상기 캐시-사이드 어드레스 모니터 스토리지 오버플로우 유닛은 코어 식별자들이 저장되는 코어들의 서브세트에만 무효화 요청들을 보내는 것을 포함하는 상기 어드레스 모니터 스토리지 오버플로우 정책을 시행하는 프로세서.

청구항 11

제10항에 있어서,

상기 캐시-사이드 어드레스 모니터 스토리지 오버플로우 유닛은 오버플로우 구조를 확인하여 상기 코어들의 서브세트를 결정하는 프로세서.

청구항 12

명령어들을 처리하는 시스템으로서,

인터커넥트;

상기 인터커넥트와 연결되는 프로세서 - 상기 프로세서는:

적어도 일부의 회로를 포함하고, 분산형 캐시의 제1 캐시 부분에 대응하고 또한 연결되며 상기 프로세서의 하드웨어 스레드들의 총 수보다 적은 총 수의 어드레스 모니터 스토리지 위치들을 갖는 제1 어드레스 모니터를 포함하는 캐시 부분 제어 유닛 - 상기 분산형 캐시는 어드레스들의 중첩하지 않는 범위에 각각 매핑되는 동작 동안 복수의 캐시 부분을 포함하고, 각각의 어드레스 모니터 스토리지 위치는 상기 캐시 부분 제어 유닛이 기입들을 모니터링하는 어드레스를 저장하고, 상기 어드레스 모니터 스토리지 위치들은 상기 분산형 캐시와 상이함 -;

적어도 일부의 회로를 포함하고, 제1 코어에 대응하고 또한 연결되고, 상기 제1 코어의 하나 이상의 하드웨어 스레드들의 수와 동일한 수의 어드레스 모니터 스토리지 위치들을 갖는 제2 어드레스 모니터를 포함하는 코어 인터페이스 유닛 - 상기 제2 어드레스 모니터의 각각의 어드레스 모니터 스토리지 위치는 상기 코어 인터페이스 유닛이 기입들을 모니터링하는 어드레스 및 상기 제1 코어의 상이한 대응 하드웨어 스레드에 대한 모니터 상태를 저장함 -;

상기 제1 어드레스 모니터의 모든 어드레스 모니터 스토리지 위치들이 이용되고 이들 중 어느 것도 모

니터 요청에 대한 어드레스를 저장하는데 이용가능하지 않을 때 어드레스 모니터 스토리지 오버플로우 정책을 구현하는, 적어도 일부의 회로를 포함하고 상기 제1 어드레스 모니터에 연결되는 상기 캐시 부분 제어 유닛의 어드레스 모니터 스토리지 오버플로우 유닛; 및

적어도 일부의 회로를 포함하고, 상기 제1 코어에 대응하고 또한 연결되며 상기 제2 어드레스 모니터와 연결되는 코어-사이드 트리거 유닛

을 포함하고, 상기 코어-사이드 트리거 유닛은, 대응하는 어드레스 모니터 스토리지 위치가 트리거할 준비가 된 모니터 상태를 갖고 트리거 이벤트가 검출될 때, 상기 제1 코어의 하드웨어 스레드를 트리거함 -;

상기 인터커넥트와 연결되는 다이나믹 랜덤 액세스 메모리;

상기 인터커넥트와 연결되는 무선 통신 디바이스; 및

상기 인터커넥트와 연결되는 이미지 캡처 디바이스

를 포함하는 시스템.

청구항 13

제12항에 있어서,

상기 어드레스 모니터 스토리지 오버플로우 유닛은,

판독 트랜잭션들이 공유된 상태를 사용하도록 강요하는 것; 및

코어 식별자들이 저장될 코어들의 서브세트에만 무효화 요청들을 보내는 것

을 포함하는 상기 어드레스 모니터 스토리지 오버플로우 정책을 구현하는 시스템.

청구항 14

제12항에 있어서,

상기 프로세서는 40개보다 많은 하드웨어 스레드들을 갖고, 상기 제1 어드레스 모니터의 상기 어드레스 모니터 스토리지 위치들의 총 수는 적어도 20개이지만 상기 프로세서의 상기 40개보다 많은 하드웨어 스레드들의 총 수보다 적은 시스템.

청구항 15

제12항에 있어서,

상기 프로세서는 동일한 모니터 어드레스에 대한 상이한 하드웨어 스레드들로부터의 모니터 요청들을 공통 어드레스 모니터 스토리지 위치에 기록하기 위한, 적어도 일부의 회로를 포함하는 상기 캐시 부분 제어 유닛의 어드레스 모니터 스토리지 위치 재사용 유닛을 더 포함하는 시스템.

청구항 16

프로세서에서의 방법으로서,

어드레스를 표시하고, 멀티-코어 프로세서의 제1 코어의 제1 논리 프로세서에서 상기 어드레스로의 기입들에 대해 모니터할 것을 표시하는 제1 명령어를 수신하는 단계; 및

상기 제1 명령어에 응답하여:

상기 제1 명령어에 의해 표시되는 상기 어드레스를 상기 제1 코어에 대응하는 복수의 코어-사이드 어드레스 모니터 스토리지 위치들 중 제1 코어-사이드 어드레스 모니터 스토리지 위치에 저장하는 단계 - 상기 복수의 코어-사이드 어드레스 모니터 스토리지 위치들의 수는 상기 제1 코어의 논리 프로세서들의 수와 동일함 -;

상기 제1 명령어에 의해 표시되는 상기 어드레스를 어드레스들의 중첩하지 않는 범위에 각각 매핑되는 복수의 캐시 부분을 포함하는 분산형 캐시의 제1 캐시 부분에 대응하는 복수의 캐시-사이드 어드레스 모니터 스토리지 위치들 중 제1 캐시-사이드 어드레스 모니터 스토리지 위치에 저장하는 단계 - 상기 복수의 캐시-사이드 어드레스 모니터 스토리지 위치는 상기 분산형 캐시의 일부가 아니고, 상기 복수의 캐시-사이드 어드레스 모니터 스

토리지 위치들의 총 수는 상기 멀티-코어 프로세서의 논리 프로세서들의 총 수보다 적음 -;

상기 프로세서가 상기 어드레스로의 기입들에 대한 모니터링을 활성화하게 하는 단계; 및

모니터 상태를 추정 상태(speculative state)로 변경하는 단계

를 포함하고,

상기 방법은,

상기 어드레스로의 기입을 검출하는 단계; 및

코어-사이드 트리거 유닛으로부터 상기 제1 논리 프로세서에 웨이크업 신호를 보내는 단계

를 더 포함하는 방법.

청구항 17

제16항에 있어서,

상기 어드레스를 또한 표시하고 제2 코어의 제2 논리 프로세서에서 상기 어드레스로의 기입들에 대해 모니터할 것을 표시하는 제2 명령어를 수신하는 단계; 및

상기 제2 코어에 대한 상기 어드레스에 대한 모니터 요청을 상기 제1 캐시-사이드 어드레스 모니터 스토리지 위치에 기록하는 단계

를 더 포함하는 방법.

청구항 18

제17항에 있어서,

상기 제2 코어에 대한 상기 어드레스에 대한 상기 모니터 요청을 상기 제1 캐시-사이드 어드레스 모니터 스토리지 위치에 기록하는 단계는 상기 멀티-코어 프로세서의 각각의 코어에 대응하는 상이한 비트를 갖는 코어 마스크에서의 비트를 변경하는 단계를 포함하는 방법.

청구항 19

제16항에 있어서,

제2 어드레스를 표시하고, 상기 제1 논리 프로세서에서 상기 제2 어드레스로의 기입들에 대해 모니터할 것을 표시하는 제2 명령어를 수신하는 단계;

상기 제1 캐시 부분에 대응하는 상기 복수의 캐시-사이드 어드레스 모니터 스토리지 위치들 중에 이용가능한 캐시-사이드 어드레스 모니터 스토리지 위치들이 없는 것으로 결정하는 단계; 및

캐시-사이드 어드레스 모니터 스토리지 위치 오버플로우 모드에 진입할 것을 결정하는 단계

를 더 포함하는 방법.

청구항 20

제19항에 있어서,

상기 캐시-사이드 어드레스 모니터 스토리지 위치 오버플로우 모드에 있는 동안,

상기 제1 캐시 부분에 대응하는 모든 판독 트랜잭션들이 공유된 캐시 코히어런시 상태를 사용할 것을 강요하는 단계; 및

하나 이상의 계류중인 모니터 요청들을 갖는 상기 멀티-코어 프로세서의 코어들의 서브세트에만 상기 제1 캐시 부분에 대응하는 무효화 요청들을 보내는 단계

를 더 포함하는 방법.

청구항 21

제16항에 있어서,

상기 제1 논리 프로세서에서 상기 어드레스를 표시하는 제2 명령어를 수신하는 단계; 및

상기 제2 명령어에 응답하여, 상기 모니터 상태를 트리거 대기 상태로 변경하는 단계

를 더 포함하는 방법.

청구항 22

제16항에 있어서,

상기 복수의 코어-사이드 어드레스 모니터 스토리지 위치들의 수는 상기 제1 코어의 하드웨어 스레드들의 수와 동일한 방법.

청구항 23

삭제

청구항 24

프로세서로서,

집적 회로;

상기 집적 회로 상에 집적되고 분산형 캐시의 제1 캐시 부분에 대응하고 또한 연결되며 상기 프로세서의 논리 프로세서들의 총 수보다 적은 총 수의 캐시-사이드 어드레스 모니터 스토리지 위치들을 갖는 캐시 부분 제어 유닛 - 상기 분산형 캐시는 어드레스들의 중첩하지 않는 범위에 각각 매핑되는 동작 동안 복수의 캐시 부분을 포함하고, 각각의 캐시-사이드 어드레스 모니터 스토리지 위치는 상기 캐시 부분 제어 유닛이 기입들을 모니터링하는 어드레스를 저장하고, 상기 캐시-사이드 어드레스 모니터 스토리지 위치들은 상기 분산형 캐시와 상이함 -;

상기 집적 회로 상에 집적되고 제1 코어에 대응하고 또한 연결되며 상기 제1 코어의 하나 이상의 논리 프로세서들의 수와 동일한 수의 코어-사이드 어드레스 모니터 스토리지 위치들을 갖는 코어 인터페이스 유닛 - 각각의 코어-사이드 어드레스 모니터 스토리지 위치는 상기 코어 인터페이스 유닛이 기입들을 모니터링하는 어드레스 및 상기 제1 코어의 상이한 대응 논리 프로세서에 대한 모니터 상태를 저장함 - ; 및

적어도 일부의 회로를 포함하고, 상기 제1 코어에 대응하고 또한 연결되며 상기 코어 인터페이스 유닛과 연결되는 코어-사이드 트리거 유닛

을 포함하고, 상기 코어-사이드 트리거 유닛은, 대응하는 코어-사이드 어드레스 모니터 스토리지 위치가 트리거할 준비가 된 모니터 상태를 갖고 트리거 이벤트가 검출될 때, 상기 제1 코어의 논리 프로세서를 트리거하는 프로세서.

청구항 25

삭제

발명의 설명

기 술 분 야

[0001] 본 명세서에 설명되는 실시예들은 프로세서들에 관련된다. 특히, 본 명세서에 설명되는 실시예들은 어드레스로의 기입을 모니터링하는 명령어를 수행하도록 동작될 수 있는 프로세서들에 일반적으로 관련된다.

배 경 기 술

[0002] 반도체 처리 및 로직 설계에서의 발달들은 프로세서들 및 다른 집적 회로 디바이스들에 포함될 수 있는 로직의 양에서의 증가를 가능하게 해주었다. 그 결과, 많은 프로세서들은 이제 단일 집적 회로 또는 다이 상에 모놀리식으로 집적되는 다수의 많은 코어들을 갖는다. 다수의 코어들은 다수의 소프트웨어 스레드들 또는 다른 작업 부하들이 동시에 수행되게 하는 것을 일반적으로 돕고, 이는 실행 처리율을 증가시키는 것을 일반적으로

돕는다.

[0003] 이러한 다수의 코어 프로세서들에서의 하나의 도전과제는, 메모리로부터 데이터 및/또는 명령어들을 캐시하는데 사용되는 캐시들 상에 더 많은 요구사항들이 종종 놓인다는 점이다. 한 가지를 들면, 이러한 캐시들에서 데이터를 액세스하기 위해 더 높은 인터커넥트 대역폭에 대한 꾸준히 증가하는 요구사항이 존재하는 경향이 있다. 캐시들에 대해 인터커넥트 대역폭을 증가시키는 것을 돕는 하나의 기술은 분산형 캐시를 사용하는 것을 포함한다. 이러한 분산형 캐시는 다수의 물리적으로 분리되거나 또는 분산된 캐시 슬라이스들 또는 다른 캐시 부분들을 포함할 수 있다. 이러한 분산형 캐시는 공유된 인터커넥트를 통해 캐시의 상이한 분산형 부분들에 대해 병렬 액세스를 허용할 수 있다.

[0004] 이러한 다수의 코어 프로세서들에서의 다른 도전과제는 공유된 메모리에 관하여 스레드 동기화를 제공하는 능력이다. 운영 체제들은 공유된 메모리에 관하여 스레드 동기화를 다루기 위해 통상적으로 유휴 루프들(idle loops)을 구현한다. 예를 들어, 일 세트의 메모리 위치들을 사용하는 여러 바쁜 루프들이 존재할 수 있다. 제1 스레드가 루프에서 대기하고 대응 메모리 위치를 폴링(poll)할 수 있다. 예를 들어, 이러한 메모리 위치는 제1 스레드의 작업 큐를 나타낼 수 있고, 제1 스레드는 수행할 가용 작업이 존재하는지 결정하기 위해 작업 큐를 폴링할 수 있다. 공유된 메모리 구성에서, 바쁜 루프로부터의 이탈들은 메모리 위치와 관련되는 상태 변화로 인해 종종 발생한다. 이러한 상태 변화들은 다른 컴포넌트(예를 들어, 다른 스레드 또는 코어)에 의한 메모리 위치로의 기입들에 의해 통상 트리거된다. 예를 들어, 다른 스레드 또는 코어는 제1 스레드에 의해 실행될 작업을 제공하기 위한 메모리 위치에서의 작업 큐에 기입할 수 있다.

[0005] 특정 프로세서들(예를 들어, 캘리포니아주 산타 클라라의 인텔사로부터 입수가능한 것들)은, 공유된 메모리에 관하여 스레드 동기화를 달성하는데 MONITOR 및 MWAIT 명령어들을 사용할 수 있다. 하드웨어 스레드 또는 다른 논리 프로세서는, 모니터 유닛에 의해 모니터링될 선형 어드레스 범위를 셋 업하고, 모니터 유닛을 준비(arm) 또는 활성화하는데 MONITOR 명령어를 사용할 수 있다. 이러한 어드레스는 범용 레지스터를 통해 제공될 수 있다. 이러한 어드레스 범위는 일반적으로 라이트-백(write-back) 캐싱 타입이다. 이러한 모니터 유닛은 어드레스 범위 내의 어드레스로의 저장들/기입들을 모니터링하고 검출할 것이며, 이는 모니터 유닛을 트리거할 것이다.

[0006] MWAIT 명령어는, 프로그램 순서로 MONITOR 명령어를 뒤따를 수 있고, 하드웨어 스레드 또는 다른 논리 프로세서가, 명령어 실행을 정지하고 구현-종속 상태에 진입하게 허용하는 힌트로서 역할을 할 수 있다. 예를 들어, 이러한 논리 프로세서는 절전 소비 상태에 진입할 수다. 논리 프로세서는 MONITOR 명령어와 관련되는 일 세트의 자격부합 이벤트들(qualifying events) 중 하나의 검출까지 그 상태로 유지될 수 있다. 선행 MONITOR 명령어에 의해 준비되는 어드레스 범위에서의 어드레스로의 기입/저장은 하나의 이러한 자격부합 이벤트이다. 이러한 경우들에서, 논리 프로세서는, 그 상태를 이탈하고, 프로그램 순서로 MWAIT 명령어를 뒤따르는 명령어로 실행을 재개할 수 있다.

발명의 내용

도면의 간단한 설명

[0007] 본 발명은, 실시예들을 도시하는데 사용되는 첨부 도면들 및 이하의 설명을 참조하여 최상으로 이해될 수 있다. 이러한 도면들에서:

도 1은 프로세서의 일 실시예의 블록도이다.

도 2는 캐시 에이전트의 일 실시예의 블록도이다.

도 3은 모니터 유한 상태 머신의 일 실시예의 상태들을 도시하는 도면이다.

도 4는 모니터 요청들이 동일한 어드레스를 표시할 때 다수의 하드웨어 스레드들 및/또는 코어들에 대해 단일 캐시-사이드 어드레스 모니터 스토리지 위치를 재사용하도록 동작될 수 있는 오버플로우 회피 로직의 일 실시예의 블록도이다.

도 5는, 오래된/구형의(stale/outdated) 캐시-사이드 어드레스 모니터 스토리지 위치들을 확인하고, 어떠한 이러한 오래된/구형의 스토리지 위치들도 발견되지 않을 때 오버플로우 모드에 진입하는 것에 의해, 오버플로우 모드를 회피하려고 옵션으로 시도하는 방법의 일 실시예의 블록 흐름도이다.

도 6은 오버플로우 구조의 일 실시예의 블록도이다.

도 7a는 본 발명의 실시예들에 따른 예시적인 순차적(in-order) 파이프라인과 예시적인 레지스터 리네이밍, 비순차적(out-of-order) 발행/실행 파이프라인 양자 모두를 도시하는 블록도이다.

도 7b는 본 발명의 실시예들에 따른 프로세서에 포함될 순차적 아키텍처 코어의 예시적인 실시예와 예시적인 레지스터 리네이밍, 비순차적 발행/실행 아키텍처 코어 양자 모두를 도시하는 블록도이다.

도 8a는, 본 발명의 실시예들에 따른 단일 프로세서 코어를, 온-다이(on-die) 인터커넥트 네트워크로의 그 접속 및 레벨 2(L2) 캐시의 그 로컬 서브세트와 함께 도시하는 블록도이다.

도 8b는 본 발명의 실시예들에 따른 도 8a에서의 프로세서 코어의 일부의 확대도이다.

도 9는 본 발명의 실시예들에 따른 하나보다 많은 코어를 가질 수 있고, 통합된 메모리 제어기를 가질 수 있으며, 통합된 그래픽스(integrated graphics)를 가질 수 있는 프로세서의 블록도이다.

도시된 도 10은 본 발명의 일 실시예에 따른 시스템의 블록도이다.

도시된 도 11은 본 발명의 실시예에 따른 제1의 보다 구체적인 예시적인 시스템의 블록도이다.

도시된 도 12는 본 발명의 실시예에 따른 제2의 보다 구체적인 예시적인 시스템의 블록도이다.

도시된 도 13은 본 발명의 실시예에 따른 SoC의 블록도이다.

도 14는 본 발명의 실시예들에 따라 소스 명령어 세트에서의 바이너리 명령어들을 타겟 명령어 세트에서의 바이너리 명령어들로 변환하는 소프트웨어 명령어 변환기의 사용을 대비하는 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0008] 어드레스로의 기입들을 모니터링하는 명령어를 스케일가능하게 구현하는 방법들, 장치, 및 시스템들이 본 명세서에 개시된다. 이하의 설명에서는, 수많은 특정 상세사항들이 제시된다(예를 들어, 특정 명령어들, 명령어 기능들, 프로세서 구성들, 마이크로아키텍처 상세사항들, 일련의 동작들 등). 그러나, 실시예들은 이러한 특정 상세사항들 없이도 실시될 수 있다. 다른 경우들에서, 본 설명의 이해를 불명료하게 하는 것을 회피하기 위해서 잘 알려진 회로들, 구조들 및 기술들은 상세하게 제시되지 않았다.

[0009] 도 1은 프로세서(100)의 일 실시예의 블록도이다. 이러한 프로세서는 물리 프로세서, 집적 회로, 또는 다이를 나타낸다. 일부 실시예들에서, 프로세서는 범용 프로세서(예를 들어, 데스크톱, 랩톱, 및 유사한 컴퓨터들에 사용되는 타입의 범용 마이크로프로세서)일 수 있다. 대안적으로, 프로세서는 특수 목적 프로세서일 수 있다. 적합한 특수 목적 프로세서들의 예들은, 몇 가지 예를 들자면, 네트워크 프로세서들, 통신 프로세서들, 암호화 프로세서들, 그래픽 프로세서들, 코-프로세서들, 임베디드 프로세서들, DSP들(digital signal processors), 및 제어기들(예를 들어, 마이크로제어기들)을 포함하지만, 이에 제한되는 것은 아니다. 프로세서는 다양한 CISC(complex instruction set computing) 프로세서들, 다양한 RISC(reduced instruction set computing) 프로세서들, 다양한 VLIW(very long instruction word) 프로세서들, 이들의 다양한 하이브리드들, 또는 다른 타입들의 프로세서들 전부 중 임의의 것일 수 있다.

[0010] 이러한 프로세서는 다수의 프로세서 코어들(102)을 갖는 멀티-코어 프로세서이다. 도시된 예시적인 실시예에서, 프로세서는 코어 0(102-0) 내지 코어 7(102-7)(집합적으로는 코어들(102))을 포함하는 8개 코어들을 갖는다. 다른 실시예들에서라도, 프로세서는, 예를 들어, 2개 내지 수백개, 종종 2개 내지 수십개 정도(예를 들어, 약 5개 내지 약 100개)의 임의의 다른 원하는 수의 코어들을 포함할 수 있다. 코어들 각각이 단일 하드웨어 스레드, 다수의 하드웨어 스레드들을 가질 수 있거나, 또는 일부 코어들은 단일 하드웨어 스레드를 가질 수 있는 반면에 다른 코어들은 다수의 하드웨어 스레드들을 가질 수 있다. 일 예시적인 실시예에서, 코어들 각각은 적어도 2개의 하드웨어 스레드들을 가질 수 있지만, 본 발명의 범위가 이에 제한되는 것은 아니다.

[0011] 코어라는 용어는 종종 독립적인 아키텍처 상태(예를 들어, 실행 상태)를 유지할 수 있는 집적 회로 상에 위치되는 로직을 지칭하며, 여기서 독립적으로 유지되는 아키텍처 상태는 전용 실행 리소스들과 관련된다. 대조적으로, 하드웨어 스레드라는 용어는 종종 독립적인 아키텍처 상태를 유지할 수 있는 집적 회로 상에 위치되는 로직을 지칭하며, 여기서 독립적으로 유지되는 아키텍처 상태는 그가 사용하는 실행 리소스들에 대한 액세스를 공유한다. 특정 리소스들이 아키텍처 상태에 의해 공유되고, 다른 것들이 그 아키텍처 상태에 전용일 때, 코어와 하드웨어 스레드 사이의 경계는 덜 명확하다. 그럼에도 불구하고, 코어 및 하드웨어 스레드는 종종 운영 체제

에 의해 개별 처리 엘리먼트들 또는 논리 프로세서들로서 간주된다. 이러한 운영 체제는 일반적으로 코어들, 하드웨어 스레드들 또는 다른 논리 프로세서들 또는 처리 엘리먼트들 각각 상의 동작들을 개별적으로 스케줄링 할 수 있다. 환언하면, 처리 엘리먼트 또는 논리 프로세서는, 일 실시예에서, 실행 리소스들이 전용이거나 또는 공유되거나 또는 이들의 일부 조합이건 말건, 소프트웨어 스레드, 운영 체제, 애플리케이션, 또는 다른 코드와 같은, 코드와 독립적으로 관련될 수 있는 임의의 온-다이 프로세서 로직을 나타낼 수 있다. 하드웨어 스레드들 및 코어들 이외에도, 논리 프로세서들 또는 처리 엘리먼트들의 다른 예들은, 스레드 유닛들, 스레드 슬롯들, 처리 유닛들, 맥락들, 및/또는 상태를 유지할 수 있고 독립적으로 코드와 관련될 수 있는 임의의 다른 로직을 포함하지만, 이에 제한되는 것은 아니다.

[0012] 코어들(102)은 하나 이상의 온-다이 인터커넥트들(112)에 의해 함께 연결된다. 이러한 인터커넥트는 코어들 사이에 메시지들 및 데이터를 전달하는데 사용될 수 있다. 많은 상이한 타입들의 인터커넥트들이 적합하다는 점이 이해될 것이다. 일 실시예에서는, 링 인터커넥트가 사용될 수 있다. 대안적인 실시예들에서는 망(mesh), 원환체(torus), 크로스바(crossbar), 하이퍼큐브(hypercube), 다른 인터커넥트 구조, 또는 이러한 인터커넥트들의 하이브리드 또는 조합이 사용될 수 있다.

[0013] 각각의 코어는, 예를 들어, 하나 이상의 하위 레벨들의 캐시(도시되지 않음)와 같은, 로컬 명령어 및/또는 데이터 스토리지를 포함할 수 있다. 예를 들어, 각각의 코어는, 코어들에 가장 가까운 대응 최하위-레벨 또는 레벨 1(L1) 캐시, 및 옵션으로 코어들에 그 다음 가까운 중간-레벨 또는 레벨 2(L2) 캐시를 포함할 수 있다. 이러한 하나 이상의 하위 레벨들의 캐시는 이들이 상위 레벨 캐시(들)(예를 들어, 이하 논의되는 분산형 캐시(108))보다 이들의 대응 코어들에 물리적으로 및/또는 논리적으로 더 가깝기 때문에 하위 레벨로 지칭된다. 하나 이상의 레벨의 캐시 각각은 데이터 및/또는 명령어들을 캐시할 수 있다.

[0014] 코어들(102)은 분산형 상위 레벨 캐시(108)를 또한 공유할 수 있다. 이러한 분산형 상위 레벨 캐시는 캐시의 물리적으로 분산된 메모리들 또는 부분들을 나타낼 수 있다. 도시된 예시적인 실시예에서, 분산형 캐시는 다수의(예를 들어, 이 경우에 8개) 물리적으로 분산된 캐시 부분들(108-0 내지 108-7)(집합적으로는 캐시 부분들(108))을 포함하고, 이는 종종 캐시 슬라이스들로서 지칭된다. 다른 실시예들에서, 분산형 캐시는 더 적거나 또는 더 많은 캐시 부분들(예를 들어, 프로세서의 코어들의 수와 동일한 수의 분산형 캐시 부분들)을 포함할 수 있다. 이러한 분산형 캐시 부분들은 상이한 코어들 및/또는 스레드들에 의해 공유될 수 있다. 도시되는 바와 같이, 각각의 캐시 부분은, 각 코어와 더욱 관련될 수 있고/있거나, 옵션으로 각 코어와 함께 다이 상에 더 가깝게 물리적으로 위치될 수 있다(예를 들어, 함께 위치됨). 예를 들어, 캐시 부분(108-0)은, 다른 코어들에 비해, 코어 0(102-0)와 더 관련될 수 있고/있거나, 코어 0(102-0)와 함께 다이 상에 더 가깝게 물리적으로 위치될 수 있다(예를 들어, 함께 위치됨).

[0015] 일부 실시예들에서, 각각의 캐시 부분은 메모리 어드레스들의 상호 배타적이거나 또는 중첩하지 않는 범위에 대응할 수 있거나 또는 이에 맵핑될 수 있다. 예를 들어, 캐시 부분(108-0)은 관련된 제1 세트의 어드레스들을 가질 수 있고, 캐시 부분(108-1)은 관련된 상이한 제2 세트의 어드레스들 가질 수 있으며, 등등이다. 이러한 어드레스 범위들은 다양한 상이한 방식들로(예를 들어, 상이한 해시 함수들 또는 다른 알고리즘들을 사용하여) 분산형 캐시의 상이한 캐시 부분들 사이에 분할되거나 또는 배분될 수 있다. 일부 실시예들에서, 상위 레벨 공유된 캐시는 데이터 및/또는 명령어들을 저장하도록 동작될 수 있는 LLC(last level cache)를 나타낼 수 있지만, 이것이 요구되는 것은 아니다. 일부 실시예들에서, 분산형 캐시 (예를 들어, LLC)는, 캐시 계층에서 모든 하위 레벨들의 캐시를 포함할 수 있거나, 또는 캐시 계층에서 최상위 레벨의 캐시 다음의 것(예를 들어, L2 캐시)을 포함할 수 있지만, 이것이 요구되는 것은 아니다. 일부 실시예들에서, 코어들은 초기에 데이터 및/또는 명령어들에 대해 하나 이상의 하위 레벨 캐시들을 확인할 수 있다. 청해진 데이터 및/또는 명령어들이 하나 이상의 하위 레벨 캐시들에서 발견되지 않으면, 코어들은 공유된 분산형 상위 레벨 캐시를 확인하는 것으로 진행할 수 있다.

[0016] 도시되는 바와 같이, 일부 실시예들에서, 코어 인터페이스(I/F) 유닛(104)은 각각의 대응 코어(102)와 연결될 수 있다. 각각의 코어 인터페이스 유닛은 또한 인터커넥트(112)와 연결될 수 있다. 각각의 코어 인터페이스 유닛은, 대응 코어와 다른 코어들 사이에서, 뿐만 아니라, 대응 코어와 분산형 캐시 부분들 사이에서 매개체로서의 역할을 하도록 동작가능할 수 있다. 더욱 도시되는 바와 같이, 일부 실시예들에서, 대응 캐시 제어 유닛(106)은 각각의 캐시 슬라이스 또는 다른 부분(108)과 연결될 수 있다. 일부 실시예들에서, 각각의 캐시 제어 유닛은 대응 캐시 슬라이스 및 대응 코어와 대략 물리적으로 함께 위치될 수 있다. 각각의 캐시 제어 유닛은 인터커넥트(112)와 연결될 수 있다. 각각의 캐시 제어 유닛은 대응 분산형 캐시 부분에 대해 캐시 코히어런시를 제공하는 것을 제어하고 돕도록 동작가능할 수 있다. 코어 인터페이스 유닛(104) 및 캐시 제어 유닛(106)의

각각 대응 쌍은, 대응 코어 및 대응 캐시 부분을 인터커넥트에 및/또는 다른 코어들에 인터페이스하도록 동작될 수 있는 코어-캐시 부분 인터페이스 유닛을 집합적으로 나타낼 수 있다. 이러한 코어 인터페이스 유닛들 및 캐시 제어 유닛들은 하드웨어(예를 들어, 집적 회로, 회로들, 트랜지스터들 등), 펌웨어(예를 들어, 불휘발성 메모리에 저장되는 명령어들), 소프트웨어, 또는 이들의 조합으로 구현될 수 있다.

[0017] 프로세서는, 프로세서를 제1 메모리(도시되지 않음)와 연결하는 제1 캐시 코히어런시 인식 메모리 제어기(110-1), 및 프로세서를 제2 메모리(도시되지 않음)와 연결하는 제2 캐시 코히어런시 인식 메모리 제어기(110-2)를 또한 포함한다. 일부 실시예들에서, 각각의 캐시 코히어런시 인식 메모리 제어기는 캐시 코히어런시를 수행하도록 동작될 수 있는 홈 에이전트 로직 및 메모리와 상호작용하도록 동작될 수 있는 제2 메모리 제어기 로직을 포함할 수 있다. 단순성을 위해, 본 설명에서는, 이러한 홈 에이전트 및 메모리 제어기 기능성들이 캐시 코히어런시 인식 메모리 제어기로서 지칭될 것이다. 다른 실시예들은 더 적거나 또는 더 많은 캐시 코히어런시 인식 메모리 제어기들을 포함할 수 있다. 더욱이, 도시된 실시예에서는, 캐시 코히어런시 인식 메모리 제어기들이 온-다이 또는 온-프로세서인 반면, 다른 실시예들에서는 이들이 그 대신에 (예를 들어, 하나 이상의 칩셋 컴포넌트들로서) 오프-다이 또는 오프-프로세서일 수 있다.

[0018] 프로세서는 본 명세서에서 다양한 실시예들을 이해하는데 필요하지 않은 다른 컴포넌트들을 또한 포함할 수 있다는 점이 이해되어야 한다. 예를 들어, 프로세서는 입력 및/또는 출력 디바이스로의 인터페이스, 시스템 인터페이스, 소켓-투-소켓 인터커넥트 등 중 하나 이상을 옵션으로 포함할 수 있다.

[0019] 위에 언급된 바와 같이, 특정 프로세서들(예를 들어, 인텔사로부터 입수가능한 것들)은 공유된 메모리에 관하여 스레드 동기화를 달성하는데 MONITOR 및 MWAIT 명령어들을 사용할 수 있다. 하드웨어 스레드 또는 다른 논리 프로세서는 모니터 유닛에 의해 모니터링될 선형 어드레스 범위를 셋 업하고, 모니터 유닛을 준비 또는 활성화하는데 MONITOR 명령어를 사용할 수 있다. 이러한 어드레스는 범용 레지스터(예를 들어, EAX)를 통해 제공될 수 있다. 이러한 어드레스 범위는 일반적으로 라이트-백(write-back) 캐싱 타입이다. 이러한 모니터 유닛은 어드레스 범위 내의 어드레스로의 저장들/기입들을 모니터링하고 검출할 것이며, 이는 모니터 유닛을 트리거할 것이다. 다른 범용 레지스터들(예를 들어, ECX 및 EDX)이 모니터 유닛에 다른 정보를 통신하는데 사용될 수 있다. MWAIT 명령어는, 프로그램 순서로 MONITOR 명령어를 뒤따를 수 있고, 하드웨어 스레드 또는 다른 논리 프로세서가, 명령어 실행을 정지하고 구현-종속 상태에 진입하게 허용하는 힌트로서 역할을 할 수 있다. 예를 들어, 이러한 논리 프로세서는 슬립 상태, 전력 C-상태, 또는 다른 절전 소비 상태에 진입한다. 논리 프로세서는 MONITOR 명령어와 관련되는 일 세트의 자격부합 이벤트들 중 하나의 검출까지 그 상태로 유지될 수 있다. 선행 MONITOR 명령어에 의해 준비되는 어드레스 범위에서의 어드레스로의 기입/저장은 하나의 이러한 자격부합 이벤트이다. 이러한 경우들에서, 논리 프로세서는, 그 상태를 이탈하고, 프로그램 순서로 MWAIT 명령어를 뒤따르는 명령어로 실행을 재개할 수 있다. 범용 레지스터들(예를 들어, EAX 및 ECX)이 모니터 유닛에 다른 정보(예를 들어, 진입하는 상태에 관한 정보)를 통신하는데 사용될 수 있다.

[0020] 도 2는 캐시 에이전트(216)의 일 실시예의 블록도이다. 일부 실시예들에서, 캐시 에이전트는 도 1의 프로세서에 사용될 수 있다. 그러나, 도 2의 캐시 에이전트가 도 1의 것과는 상이한 프로세서들과 함께 사용될 수 있다는 것이 이해되어야 한다.

[0021] 캐시 에이전트(216)는 코어(202) 및 캐시 부분(208)을 포함한다. 일부 실시예에서, 이러한 코어는 멀티-코어 프로세서의 다수의 코어들 중 하나일 수 있다. 일부 실시예들에서, 이러한 캐시 부분은 분산형 캐시(예를 들어, 분산형 LLC)의 다수의 캐시 슬라이스들 또는 다른 캐시 부분들 중 하나일 수 있다. 캐시 에이전트는 또한 코어 인터페이스 유닛(204)과 캐시 부분 제어 유닛(206)을 포함한다. 코어는 코어 인터페이스 유닛을 통해 인터커넥트(212)와 연결된다. 캐시 부분은 캐시 부분 제어 유닛을 통해 인터커넥트와 연결된다. 코어 인터페이스 유닛은 코어와 캐시 부분 제어 유닛 사이에 연결된다. 캐시 부분 제어 유닛은 코어 인터페이스와 캐시 부분 사이에 연결된다. 이러한 코어, 캐시 부분, 코어 인터페이스 유닛, 및 캐시 부분 제어 유닛은 옵션으로, 도 1의 대응하게 명명되는 컴포넌트들과 유사하거나 동일할 수 있다. 이러한 특정 예에서, 코어는 제1 하드웨어 스레드(218-1) 및 제2 하드웨어 스레드(218-2)를 포함하는 멀티-스레드형 코어이지만, 본 발명의 범위가 이에 제한되는 것은 아니다. 다른 실시예들에서, 코어는 단일 스레드형일 수 있거나 또는 2개보다 많은 하드웨어 스레드들을 가질 수 있다.

[0022] 캐시 에이전트(216)는 하나 이상의 어드레스들(예를 들어, MONITOR 명령어에 의해 표시되는 어드레스 범위)로의 기입에 대해 모니터링하는데 사용되는 모니터 명령어(예를 들어, MONITOR 명령어)를 구현하도록 동작될 수 있는 모니터 메커니즘을 포함한다. 이러한 메커니즘은 기존 캐시 코히어런시 메커니즘을 이용할 수 있거나 또는 이

를 레버리징할 수 있다(예를 들어, 캐시 코히어런시 메커니즘을 통해 전달되는 어드레스에 기입할 의도의 통신을 이용할 수 있다). 도시된 실시예에서, 모니터 메커니즘은 캐시-사이드 어드레스 모니터 유닛(226), 코어-사이드 어드레스 모니터 유닛(220), 코어-사이드 트리거 유닛(234), 및 캐시-사이드 스토리지 오버플로우 유닛(236)을 포함한다. 본 명세서에 사용되는 바와 같이, "코어-사이드"라는 용어는, 인터커넥트(212)의 코어(202)와 동일한 사이드에 있는 것 및/또는 코어와 인터커넥트 사이에 배치되는 것 및/또는 캐시 부분보다 코어에 논리적으로 더 가까운 것을 지칭한다. 마찬가지로, "캐시-사이드"란 용어는, 인터커넥트(212)의 캐시 부분(208)과 동일한 사이드에 있는 것 및/또는 캐시 부분과 인터커넥트 사이에 배치되는 것 및/또는 코어보다 캐시 부분에 논리적으로 더 가까운 것을 지칭한다.

[0023] 도시된 실시예에서, 캐시-사이드 어드레스 모니터 유닛(226) 및 캐시-사이드 스토리지 오버플로우 유닛(236)은 양자 모두 캐시 부분 제어 유닛(206)에 구현되지만, 이것이 요구되는 것은 아니다. 다른 실시예들에서, 이러한 유닛들 중 하나 이상은 분리된 캐시-사이드 컴포넌트로서 구현될 수 있다(예를 들어, 캐시 제어 유닛과 및/또는 캐시 부분과 연결됨). 유사하게, 도시된 실시예에서, 코어-사이드 어드레스 모니터 유닛(220) 및 코어-사이드 트리거 유닛(234)은 양자 모두 코어 인터페이스 유닛(204)에 구현되지만, 이것이 요구되는 것은 아니다. 다른 실시예들에서, 이러한 유닛들 중 하나 이상은 분리된 코어-사이드 컴포넌트로서 구현될 수 있다(예를 들어, 코어 인터페이스 유닛과 및/또는 코어와 연결됨).

[0024] 캐시-사이드 어드레스 모니터 유닛(226)은 캐시 부분(208)에 대응하며, 이는 분산형 캐시의 슬라이스 또는 다른 부분이다. 캐시-사이드 어드레스 모니터 유닛은 다수의 상이한 캐시-사이드 어드레스 모니터 스토리지 위치들(228)을 갖는다. 도시된 바와 같이, 각각의 캐시-사이드 어드레스 모니터 스토리지 위치는 기입들에 대해 모니터링될 어드레스(230)를 저장하는데 사용될 수 있다. 일부 실시예들에서, 각각의 캐시-사이드 어드레스 모니터 스토리지 위치는 어드레스에 관련되는 코어의 표시(예를 들어, 코어 식별자, 각각의 상이한 코어에 대응하는 상이한 비트를 갖는 코어 마스크 등)를 또한 저장할 수 있다. 예를 들어, 이러한 스토리지 위치들은 하드웨어 구현되는 표에서의 상이한 엔트리들을 나타낼 수 있다. 도시된 바와 같이, 도시된 실시예에는, 제1 캐시-사이드 어드레스 모니터 스토리지 위치(228-1) 내지 제N 캐시-사이드 어드레스 모니터 스토리지 위치(228-N)가 존재할 수 있으며, 여기서 N은 특정 구현에 적합한 수일 수 있다.

[0025] 일부 실시예들에서, 캐시 부분에 대응하는 캐시-사이드 어드레스 모니터 유닛에서의 캐시-사이드 어드레스 모니터 스토리지 위치들의 총 수는 프로세서 및/또는 프로세서가 구현되는 소켓의 하드웨어 스레드들(또는 다른 논리 프로세서들)의 총 수보다 작을 수 있다. 일부 실시예들에서, 각각의 하드웨어 스레드(또는 다른 논리 프로세서)는 단일 어드레스 또는 어드레스들의 단일 범위를 모니터링하는데 모니터 명령어(예를 들어, MONITOR 명령어)를 사용하도록 동작가능할 수 있다. 일부 경우들에서, 이러한 모니터 명령어를 사용한 후, 하드웨어 스레드는 슬립(sleep)에 놓일 수 있거나 또는 다른 절전 소비 상태에 놓일 수 있다. 하나의 가능한 접근방식은 모니터링될 어드레스를 저장하기에 충분한 캐시-사이드 어드레스 모니터 스토리지 위치들(228)을 각각의 하드웨어 스레드(또는 다른 논리 프로세서)에 대해 제공하는 것일 것이다. 그러나, 분산형 캐시가 사용될 때, 각각의 어드레스는 단일 대응 캐시 슬라이스 또는 다른 캐시 부분에만 해시(hash)할 수 있거나 그렇지 않으면 이에 맵핑할 수 있다. 예를 들어, 이러한 어드레스의 해시는 특정 해시 함수에 따라 그 어드레스에 대응하는 단일 대응 캐시 슬라이스를 선택할 수 있다. 따라서, 이러한 분산형 캐시가 사용될 때, 하드웨어 스레드들(또는 다른 논리 프로세서들) 모두에 대해 모니터링될 어드레스들 모두가 동일한 단일 캐시 슬라이스에 모두 해시될 수 있거나 또는 그렇지 않으면 이에 매핑될 수 있는 기회가 있지만, 일반적으로 매우 적은 기회이다.

[0026] 이러한 가능성을 허용하기 위해, 하나의 가능한 접근방식은 각각의 캐시 부분에 프로세서 및/또는 소켓의 하드웨어 스레드들(또는 다른 논리 프로세서들)의 총 수와 동일한 수의 캐시-사이드 어드레스 모니터 스토리지 위치들(228)을 공급하는 것일 것이다. 예를 들어, 각각의 코어가 2개의 하드웨어 스레드들을 갖는 8개의 코어 프로세서에서는, 총 16개의 캐시-사이드 어드레스 모니터 스토리지 위치들 (즉, 코어들의 수 곱하기 코어 당 스레드들의 수)이 8개의 각 캐시 슬라이스들에 대해 제공될 수 있다. 예를 들어, 하드웨어 스레드들의 총 수와 동일한 수의 엔트리들을 갖는 하드웨어 구현되는 표가 포함될 수 있다. 일부 경우들에서, 각각의 스토리지 위치는 대응 하드웨어 스레드에 대한 고정된 대응관계 또는 할당을 가질 수 있다. 이는 모든 하드웨어 스레드마다 모니터링될 어드레스를 저장하게 할 수 있고, 이러한 어드레스들 모두가 동일한 캐시 부분에 매핑할 수 있고 이에 따라 그 캐시 부분에 대해 로컬하게 저장될 필요가 있을 것인 가능성을 허용할 수 있다. 이러한 접근방식은 본질적으로 최악의 경우 시나리오에 대해 설계를 하고, 이는 일반적으로 상당히 가능성이 없지만, 지금까지는 무시될 수 없는데, 그 이유는 이것이 정말 발생하게 되면 어떠한 접근방식도 이러한 시나리오를 다룰 수 없었기 때문이다.

[0027] 이러한 접근방식에 대한 하나의 단점은 하드웨어 스레드들(또는 다른 논리 프로세서들)의 수 및/또는 캐시 부분들의 수가 증가함에 따라 상대적으로 스케일불가능하게(un-scalable) 되는 경향이 있다는 점이다. 하드웨어 스레드들의 수를 증가시키는 것은 각각의 캐시 부분에 대해 필요한 스토리지 위치들의 수를 증가시킨다. 더욱이, 캐시 부분들의 수를 증가시키는 것은 각각의 추가적 캐시 부분에 대해 이러한 스토리지 위치들의 추가적 세트를 추가하는 것을 포함한다. 프로세서들은, 몇몇 예들을 들자면, 32개 스레드들, 36개 스레드들, 40개 스레드들, 56개 스레드들, 128개 스레드들, 또는 256개 스레드들보다 많이 가질 수 있다. 이러한 많은 수들의 스레드들이 사용될 때 스토리지의 양이 상당히 중요하게 될 수 있다는 점을 쉽게 알 수 있다. 스토리지의 이러한 상당한 양들은 프로세서의 제조 비용, 스토리지를 공급하는데 필요한 다이 상의 영역(area on-die)의 양, 및/또는 스토리지에 의해 초래되는 전력 소비를 증가시키는 경향이 있다.

[0028] 대안적인 접근방식으로서, 일부 실시예들에서, 캐시 부분(208)에 대응하는 캐시-사이드 어드레스 모니터 유닛(226)에서의 캐시-사이드 어드레스 모니터 스토리지 위치들(228)의 총 수는 프로세서 및/또는 소켓의 하드웨어 스레드들(또는 다른 논리 프로세서들)의 총 수보다 적을 수 있다. 어드레스 모니터 스토리지 오버플로우의 가능성을 완전히 회피하는데 엄격하게 필요한 수보다 더 적은 어드레스 모니터 스토리지 위치들이 존재할 수 있다. 일부 실시예들에서, 각각의 캐시 부분은 대부분의 시간 오버플로우를 회피하기에 충분한 어드레스 모니터 스토리지 위치들의 수를 그와 관련하여 가질 수 있지만, 이는 이러한 오버플로우를 모든 인스턴스들에서 완전히 방지하기에는 불충분하다. 일부 실시예들에서, 캐시 부분 당 캐시-사이드 어드레스 모니터 스토리지 위치들의 총 수는, 오버플로우의 가능성이, 원하는 바에 따라, 약 십만분의 일, 약 백만분의 일, 또는 약 천만분의 일보다 크지 않은 프로세서의 하드웨어 스레드들의 총 수에 대해 충분할 수 있다. 일부 실시예들에서, 프로세서는 약 40개보다 많은 하드웨어 스레드들을 가질 수 있고, 캐시 부분 당 캐시-사이드 어드레스 모니터 스토리지 위치들의 총 수는 40개보다 적을 수 있다(예를 들어, 약 20개 내지 약 38개). 일부 실시예들에서, 프로세서는 50개보다 많은 하드웨어 스레드들을 가질 수 있고, 캐시 부분 당 캐시-사이드 어드레스 모니터 스토리지 위치들의 총 수는 약 50개보다 적을 수 있다(예를 들어, 약 20개 내지 약 45개, 또는 약 25개 내지 약 40개, 또는 약 30개 내지 약 40개). 일부 실시예들에서는, 캐시-사이드 어드레스 모니터 스토리지 위치들을 특정 하드웨어 스레드들에 지정하거나 할당하는 것 대신에, 스토리지 위치들이 임의의 특정 하드웨어 스레드에 대응하지 않을 수 있지만, 오히려 임의의 스토리지 위치가 임의의 하드웨어 스레드에 의해 사용될 수 있다. 유리하게는, 프로세서 및/또는 소켓의 하드웨어 스레드들(또는 다른 논리 프로세서들)의 총 수보다 적은 캐시 부분에 대응하는 캐시-사이드 어드레스 모니터 유닛에서의 캐시-사이드 어드레스 모니터 스토리지 위치들의 총 수를 사용하는 것이, 모니터 명령어들(예를 들어, MONITOR 명령어)를 구현하는데 보다 스케일가능한 해결책을 제공하는 것을 잠재적으로 도울 수 있다. 그러나, 본 명세서에 개시되는 실시예들은, 하드웨어 스레드들 및/또는 코어들의 수, 및/또는, 스토리지의 총량이 큰가 적은가에 무관하게, 유용성을 갖는다는 점이 이해되어야 한다.

[0029] 다시 도 2를 참조하면, 캐시 에이전트는 코어-사이드 어드레스 모니터 유닛(220)을 포함하고, 이는 코어(202)에 대응한다. 이러한 코어-사이드 어드레스 모니터 유닛은 대응 코어의 하나 이상의 하드웨어 스레드들의 수와 동일한 수의 코어-사이드 어드레스 모니터 스토리지 위치들을 갖는다. 도시된 실시예에서, 제1 코어-사이드 어드레스 모니터 스토리지 위치(221-1)는 제1 하드웨어 스레드(218-1)에 대해 고정된 대응관계를 갖고, 제2 코어-사이드 어드레스 모니터 스토리지 위치(221-2)는 제2 하드웨어 스레드(218-2)에 대해 고정된 대응관계를 갖는다. 다른 실시예들에서는, 스레드들과 스토리지 위치들의 다른 수들이 사용될 수 있다. 각각의 코어-사이드 어드레스 모니터 스토리지 위치는 대응 코어의 대응 하드웨어 스레드(218-1, 218-2)에 대해 모니터링될 어드레스(222-1, 222-2)를 저장하도록 동작가능할 수 있다. 이러한 고정된 대응관계가 존재할 때, 어드레스를 스토리지 위치에 저장하는 것은 어드레스를 하드웨어 스레드와 대응하는 하드웨어와 관련시킬 수 있다. 다른 실시예들에서, 스토리지 위치들과 하드웨어 스레드들 사이에 고정된 대응관계가 존재하지 않으면, 각각의 스토리지 위치는 모니터링될 어드레스에 대응하는 하드웨어 스레드의 표시(예를 들어, 하드웨어 스레드 식별자)를 저장하는데 사용될 수 있다. 일부 실시예들에서, 각각의 코어-사이드 어드레스 모니터 스토리지 위치는 대응 코어의 대응 하드웨어 스레드(218-1, 218-2)에 대해 모니터 상태(224-1, 224-2)를 저장하도록 또한 동작가능할 수 있다. 일부 실시예들에서, 각각의 모니터 상태는 모니터 FSM(finite state machine)을 나타낼 수 있다. 일부 실시예들에서, MONITOR 명령어의 경우에, 이러한 모니터 상태는 유휴 상태, 추정(예를 들어, 모니터가 로딩된) 상태, 및 트리거 준비(예를 들어, wait2trigger) 상태 중 임의의 하나일 수 있지만, 본 발명의 범위가 이에 제한되는 것은 아니다.

[0030] 일부 실시예들에서, 캐시-사이드 어드레스 모니터 유닛(226) 및 코어-사이드 어드레스 모니터 유닛(220)은 하나 이상의 어드레스들(예를 들어, MONITOR 명령어에 의해 보여진 어드레스 범위의 어드레스)로의 기입들에 대해 모

니터링하기 위해 협력하거나 함께 작업할 수 있다. 특정 개념들을 더 예시하기 위해, 모니터 메커니즘이 MONITOR 및 MWAIT 명령어들을 수행하는 방법의 일 예를 고려하자. 제1 하드웨어 스레드(218-1)는 MONITOR 명령어를 수행할 수 있다. MONITOR 명령어는 기입에 대해 모니터링될 어드레스를 표시할 수 있다. 제1 하드웨어 스레드는 표시된 모니터 어드레스에 대해 대응 모니터 요청을 발행한다. 이러한 모니터 요청은 제1 코어-사이드 어드레스 모니터 유닛(220)로 하여금 표시된 모니터 어드레스(222-1)를 제1 코어-사이드 어드레스 모니터 스토리지 위치(221-1)에 저장하게 할 수 있다. 모니터 상태(224-1)는 추정된 또는 모니터 로드된 상태로 설정될 수 있다. MONITOR 요청은 표시된 모니터 어드레스에 대응하는 데이터를 저장하는 것으로 가정하기에 적절한 분산형 캐시 부분(208)에게 인터커넥트(212) 상에서 라우팅될 수 있다. 특정 표시된 모니터 어드레스에 의존하여 이는 매핑에 대해 사용되는 해시 함수 또는 다른 알고리즘에 기초하는 임의의 분산형 캐시 부분들일 수 있다는 점이 주목된다. 캐시-사이드 어드레스 모니터 유닛은 표시된 모니터 어드레스를 캐시-사이드 어드레스 모니터 스토리지 위치(230)(예를 들어, 위치들 230-1 내지 230-N 중 임의의 이용가능한 것)에 저장할 수 있다. 제1 하드웨어 스레드(218-1)를 갖는 코어(202)를 식별하는 코어 식별자가 코어 식별자(ID)(232)로서 캐시-사이드 어드레스 모니터 스토리지 위치(230)에 또한 저장될 수 있다. 일부 실시예들에서, 이러한 코어 식별자는 코어들 중 하나를 식별하는 일 세트의 비트들일 수 있다. 다른 실시예들에서는, 모니터되는 동일한 어드레스에 대해 다수의 코어들에 의해 단일 스토리지 위치가 공유될 수 있도록 코어 마스크가 옵션으로 사용될 수 있다.

[0031] 제1 스레드(218-1)는 모니터된 어드레스를 또한 표시할 수 있는 MWAIT 명령어를 후속하여 수행할 수 있다. 제1 하드웨어 스레드는 표시된 모니터 어드레스에 대해 대응 MWAIT 신호를 발행한다. 이러한 MWAIT 신호에 응답하여 코어 사이드 어드레스 모니터 유닛(220)은 모니터 상태(224-1)를 트리거 준비 상태(예를 들어, 트리거 대기 상태)로 설정할 수 있다. 제1 하드웨어 스레드는, 예를 들어, 슬립 또는 다른 절전 소비 상태와 같은, 상이한 상태에 옵션으로 놓일 수 있다. 대표적으로, 제1 스레드는 스레드가 슬립으로 가야 하고 그리고 나서 슬립으로 가야 하면 맥락에 그 상태를 저장할 수 있다.

[0032] 후속하여, 표시된 모니터 어드레스에 기입하려는 의도(예를 들어, 소유권 요청에 대한 판독, 표시된 모니터 어드레스를 보여주는 스누프 무효화, 공유된 상태로부터 배타적인 상태로 변경하는 어드레스와 관련되는 상태 천이 등)가 존재할 때, 캐시-사이드 어드레스 모니터 유닛은 어드레스에 기입하려는 이러한 의도를 검출할 수 있다. 어드레스는 그 스토리지 위치들 중 하나에서의 어드레스들 중 하나와 일치할 수 있다. 스토리지 위치에 대응하는 하나 이상의 코어들은, 예를 들어, 캐시-사이드 어드레스 모니터 스토리지 위치에 저장되는 코어 식별자 또는 코어 마스크에 의해, 결정될 수 있다. 캐시-사이드 어드레스 모니터 유닛은 표시된 모니터 어드레스를 저장하는데 사용되는 캐시-사이드 어드레스 모니터 스토리지 위치를 클리어할 수 있다. 이는, 예를 들어, 대응 코어(들)에 스누프 무효화를 보내는 것에 의해, 대응 코어(들)에 신호를 보낼 수 있다. 캐시-사이드 어드레스 모니터 유닛은, 어드레스에 기입하려는 의도를 (예를 들어, 소유권에 대한 요청 또는 스누프 무효화를 통해) 그 어드레스를 모니터링하고 있는 것으로 알려진 하나 이상의 코어들을 향해서만 선택적으로 직접 통지하는 것을 돕는 일종의 개선된 필터로서 역할을 할 수 있다. 이러한 통지들은 그 어드레스를 모니터링하는 코어들의 서브세트에 선택적으로 제공되는 "힌트들(hints)"을 나타낼 수 있다. 유리하게도, 이는 그 어드레스를 모니터링하고 있지 않은 코어들에게 통지하는 것을 회피하는데 도움이 될 수 있고, 이는 잘못된 웨이크업들을 회피하는 것 및/또는 인터커넥트 상의 트래픽을 감소시키는 것에 도움이 될 수 있다.

[0033] 신호를 받은 코어(들)에서의 코어-사이드 어드레스 모니터 유닛(220)은, 그 신호를 수신할 수 있고, 그 신호에 (예를 들어, 스누프 무효화에) 표시된 어드레스를 그 코어-사이드 어드레스 모니터 스토리지 위치들에서의 모니터 어드레스들과 비교할 수 있다. 이는 그 신호의 어드레스가 제1 하드웨어 스레드(218-1)에 대응하는 제1 코어-사이드 모니터 어드레스 스토리지 위치(221-1)에서의 모니터 어드레스(222-1)에 일치하는지 결정할 수 있다. 코어-사이드 어드레스 모니터 유닛은 제1 하드웨어 스레드가 모니터되고 있는 어드레스에 대응하는지 알 수 있다. 코어-사이드 어드레스 모니터 유닛은, 모니터된 어드레스에 기입하려는 의도가 관찰되었다는 신호를 코어 사이드 트리거 유닛(234)에 보낼 수 있다. 이는 제1 코어-사이드 어드레스 모니터 스토리지 위치를 클리어할 수 있고, 모니터 상태(224-1)을 유휴로 변경할 수 있다. 코어-사이드 트리거 유닛은 제1 하드웨어 스레드에 트리거 신호(예를 들어, 경보, 통지, 또는 웨이크 신호)를 제공하도록 동작가능할 수 있다. 이러한 실시예에서, 코어-사이드 트리거 유닛은 코어-사이드이고, 이는 단순히 로직에 도움이 될 수 있지만, 이는 옵션으로 캐시-사이드에 제공될 수도 있다. 제1 하드웨어 스레드는, 슬립이었으면, 웨이크될 수 있다.

[0034] 일부 실시예들에서는, 캐시-사이드 어드레스 모니터 스토리지 위치들이 오버플로우될 가능성이 있다. 예를 들어, 새로운 모니터 요청이 캐시-사이드 어드레스 모니터 유닛에서 수신될 수 있지만, 캐시-사이드 어드레스 모니터 스토리지 위치들 모두가 현재 사용 중이어서, 새로운 모니터 요청의 어드레스를 저장하기 위한 비어있는/

이용가능한 캐시-사이드 어드레스 모니터 스토리지 위치가 존재하지 않을 수 있다. 도시된 바와 같이, 일부 실시예들에서, 캐시-사이드 어드레스 모니터 유닛은 캐시 부분에 대응하는 캐시-사이드 어드레스 모니터 스토리지 오버플로우 유닛(236)과 연결될 수 있다. 일부 실시예들에서, 이러한 캐시-사이드 어드레스 모니터 스토리지 오버플로우 유닛은, 새로운 모니터 요청의 어드레스를 저장할 수 있는 비어있는/이용가능한/미사용된 캐시-사이드 어드레스 모니터 스토리지 위치들이 존재하지 않을 때, 어드레스 모니터 스토리지 오버플로우 정책을 시행하거나 구현하도록 동작가능할 수 있다.

[0035] 언급된 바와 같이, 일부 실시예에서, 코어-사이드 어드레스 모니터 유닛은 그 대응 코어에서의 하드웨어 스레드들의 수와 동일한 수의 코어-사이드 어드레스 모니터 스토리지 위치들을 가질 수 있다. 유사하게, 일부 실시예들에서 다른 코어들의 코어-사이드 어드레스 모니터 유닛들은 그들의 대응 코어들에서의 하드웨어 스레드들의 수와 동일한 수의 코어-사이드 어드레스 모니터 스토리지 위치들을 가질 수 있다. 집합적으로 이러한 코어-사이드 어드레스 모니터 스토리지 위치들은 프로세서의 하드웨어 스레드들(또는 다른 논리 프로세서들)의 총 수만큼 많은 코어-사이드 어드레스 모니터 스토리지 위치들의 한 세트를 나타낼 수 있다. 유리하게도, 심지어 캐시-사이드 어드레스 모니터 스토리지 위치들의 오버플로우가 있을 때에도, 코어-사이드 어드레스 모니터 유닛들은 모든 하드웨어 스레드들(또는 다른 논리 프로세서들)에 대해 모니터되는 어드레스들 모두를 저장하기에 충분한 코어-사이드 어드레스 모니터 스토리지 위치들을 여전히 갖는다.

[0036] 도 3은 MONITOR 명령어 및 MWAIT 명령어를 구현하는데 적합한 모니터 FSM(finite state machine)(347)의 일 실시예의 상태들을 도시하는 도면이다. 실행 스레드로부터 어드레스에 대한 모니터 요청을 수신하면, 모니터 FSM은 유휴 상태(340)로부터 추정 상태(341)로의 천이(343)를 행할 수 있다. 모니터 FSM이 추정 상태에 있는 동안, 해당 어드레스에 대응하는 데이터를 저장하여야 할 캐시 부분이 그 어드레스에 일치하는 기입 요청을 수신하면, 또는 모니터 클리어 요청이 실행 스레드로부터 제공되면, 모니터 FSM은 유휴 상태(340)로 돌아가는 천이(344)를 행할 수 있다. 다른 모니터 요청이 그 동일한 실행 스레드로부터 제공되면, 모니터 FSM은 추정 상태(341)로 돌아가는 천이(343)를 행할 수 있고, 모니터된 어드레스는 적절하면 조절될 수 있다. 반면에, 추정 상태(341)에 있는 동안 그 실행 스레드로부터 MWAIT 요청이 제공되면, 모니터 FSM은 트리거 대기 상태(342)로의 천이(345)를 행할 수 있다. 심지어 MWAIT 요청을 수신하기 이전에도, 모니터 요청이 수신되는 시간으로부터 어드레스를 추적하는 동안, 가장 최근에 모니터된 어드레스에 대해서만 모니터-웨이크 이벤트들이 보내지는 것을 보장하는 것에 이러한 추정 상태가 도움이 될 수 있다. 모니터 FSM이 트리거 대기 상태에 있는 동안, 해당 어드레스에 대응하는 데이터를 저장하여야 할 캐시 부분이 모니터된 어드레스에 일치하는 기입 요청을 수신하면, 실행 스레드에 모니터-웨이크 이벤트가 보내질 수 있다. 한편, 모니터 FSM이 트리거 대기 상태(342)에 있는 동안 모니터 클리어 요청이 실행 스레드로부터 제공될 수 있다. 이러한 경우에 모니터 요청은 그 실행 스레드에 대해 소거될 수 있고 어떠한 모니터-웨이크 이벤트도 실행 스레드에 보내질 필요가 없지만, 이러한 2가지 경우들 중 어느 하나에서, 모니터 FSM은 유휴 상태(340)로 돌아가는 천이(346)를 행할 수 있다.

[0037] 도 4는 모니터 요청들이 동일한 어드레스를 표시할 때 다수의 하드웨어 스레드들 및/또는 코어들에 대해 단일 캐시-사이드 어드레스 모니터 스토리지 위치(428)를 재사용하도록 동작될 수 있는 오버플로우 회피 로직(460)의 일 실시예의 블록도이다. 이러한 로직은 캐시-사이드 어드레스 모니터 스토리지 위치(428)와 연결되는 캐시-사이드 어드레스 모니터 스토리지 위치 재사용 유닛(464)을 포함한다. 이러한 캐시-사이드 어드레스 모니터 스토리지 위치 재사용 유닛은 동일한 어드레스를 표시하는 상이한 하드웨어 스레드들 및/또는 코어들로부터 모니터 요청들(462)을 수신할 수 있다. 하나의 가능한 접근방식은 이러한 동일한 어드레스의 상이한 사본들을 상이한 캐시-사이드 어드레스 모니터 스토리지 위치들에(예를 들어, 상이한 엔트리들을 하드웨어 구현되는 표에) 저장하는 것일 것이다. 그러나, 이것은 다수의, 또는 일부 경우들에서 많은, 캐시-사이드 어드레스 모니터 스토리지 위치들을 소비할 수 있다.

[0038] 대안적 접근방식으로서, 일부 실시예들에서는, 모니터링될 어드레스(430)를 저장하고 상이한 하드웨어 스레드들로부터의 모니터 요청들을 나타내는데 단일 캐시-사이드 어드레스 모니터 스토리지 위치(428)가 사용될 수 있다. 일부 실시예들에서, 모니터링될 어드레스와 다수의 코어들을 관련시킬 수 있는 구조(432)가 또한 캐시-사이드 어드레스 모니터 스토리지 위치(428)에 저장된다. 일 예에서, 이러한 구조는 코어 마스크 구조(432)를 포함할 수 있다. 이러한 코어 마스크는 프로세서의 코어들의 총 수와 동일한 수의 비트들을 가질 수 있고, 코어 마스크의 각각의 비트는 상이한 코어에 대해 고정된 대응관계를 가질 수 있다. 하나의 가능한 협약에 따르면, 각각의 비트는, 대응 코어가 어드레스에 대해 계류중인 모니터 요청을 갖지 않는 것을 표시하는 제1 값(예를 들어, 바이너리 0으로 클리어됨), 또는 대응 코어가 어드레스에 대해 계류중인 모니터 요청을 갖는 것을 표시하는 제2 값(예를 들어, 바이너리 1로 설정됨)을 가질 수 있다. 반대의 협약이 또한 가능하다. 대응 코어에

대한 비트는, 모니터 요청이 캐시-사이드 어드레스 모니터 스토리지 위치에 저장된 어드레스에 대한 그 코어로부터 수신된 것을 표시하도록 설정될 수 있거나, 또는 어드레스로의 기입이 관찰되어 코어-사이드 로직에 보고될 때 클리어될 수 있다. 캐시-사이드 어드레스 모니터 스토리지 위치가 스레드 식별자에 의해서가 아니라 어드레스에 의해 추적된다는 점을 주목하자. 유리하게도, 이러한 방식에서, 상이한 코어들로부터의 동일한 어드레스에 대한 모니터 요청들은 동일한 단일 캐시-사이드 어드레스 모니터 스토리지 위치 내로 모일 수 있다(collapsed). 상이한 스레드들/코어들로부터의 다수의 요청들에 대한 스토리지 위치의 이러한 재사용은 캐시-사이드 어드레스 모니터 스토리지 위치 오버플로우를 회피하는데 도움이 될 수 있다.

[0039] 위에 언급된 바와 같이, 제한된 수의 캐시-사이드 어드레스 모니터 스토리지 위치들을 오버플로우시키는 것이 일부 경우들에서 가능하다. 일부 실시예들에서, 정책들의 오버플로우 모드 또는 세트는 모니터 메커니즘이 오버플로우의 경우에도 올바르게 동작하게 하도록 제공될 수 있다.

[0040] 도 5는 오래된/구형의(stale/outdated) 캐시-사이드 어드레스 모니터 스토리지 위치들을 확인하고, 어떠한 이러한 오래된/구형의 스토리지 위치들도 발견되지 않을 때 오버플로우 모드에 진입하는 것에 의해, 오버플로우 모드를 회피하려고 옵션으로 시도하는 방법(570)의 일 실시예의 블록 흐름도이다. 일부 실시예들에서, 도 5의 동작들 및/또는 방법은 도 1의 프로세서 및/또는 도 2의 캐시 에이전트에 의해 및/또는 이들 내에서 수행될 수 있다. 도 1의 프로세서 및 도 2의 캐시 에이전트에 대해 본 명세서에 설명되는 컴포넌트들, 특징들, 및 특정 옵션의 상세사항들은 도 5의 동작들 및/또는 방법에 옵션으로 또한 적용된다. 대안적으로, 도 5의 동작들 및/또는 방법은 유사하거나 상이한 프로세서 및/또는 캐시 에이전트에 의해 및/또는 이들 내에서 수행될 수 있다. 또한, 도 1의 프로세서 및/또는 도 2의 캐시 에이전트는 도 5의 것들과 동일하거나, 유사하거나 또는 상이한 동작들 및/또는 방법들을 수행할 수 있다.

[0041] 이러한 방법은, 블록 571에서, 수신된 모니터 요청을 다루는데 이용가능한/미사용된 캐시-사이드 어드레스 모니터 스토리지 위치들이 존재하지 않는다는 것을 결정하는 단계를 포함한다. 예를 들어, 모니터 요청이 캐시-사이드 어드레스 모니터 유닛(예를 들어, 캐시-사이드 어드레스 모니터 유닛(226))에서 수신될 수 있고, 이러한 캐시-사이드 어드레스 모니터 유닛은 이러한 모니터 요청을 다루는데 이용가능한/미사용된 캐시-사이드 어드레스 모니터 스토리지 위치가 존재하지 않는다는 것을 결정할 수 있다. 예를 들어, 모든 캐시-사이드 어드레스 모니터 스토리지 위치들이 모니터링될 어드레스를 현재 저장할 수 있다.

[0042] 이러한 방법은, 블록 572에서, 오래된/구형의 캐시-사이드 어드레스 모니터 스토리지 위치가 존재하는지 결정하는 단계를 옵션으로 포함하고, 최근에 수신된 모니터 요청을 다루는데 사용될 수 있다. 일부 실시예들에서, 캐시-사이드 어드레스 모니터 유닛은 어드레스를 갖는 엔트리를 선택하여 그것이 오래된 및/또는 구형의 것인지를 결정할 수 있다. 예를 들어, 오래된/구형의 어드레스는 스토리지 위치에 여전히 저장된 어드레스를 나타낼 수 있지만 그 어드레스에 대해 어떠한 유효한 계류중인 모니터 요청들도 현재 존재하지 않는다. 예를 들어, 설정되고 있지만 준비되지 않은 모니터로 인해, 잘못된 모니터 요청들의 경우들이 존재할 수 있다. 이러한 엔트리는, 랜덤으로, 그 엔트리의 연령에 기초하여, 유효성의 예측에 기초하여, 또는 다른 방식으로 선택될 수 있다. 일부 실시예들에서는, 스토리지 위치가 오래된/구형의 것인지 확인하기 위해, 캐시-사이드 어드레스 모니터 유닛이, 관련된 어드레스에 대한 스누프 요청을, 그 어드레스에 대한 모니터 요청들을 갖는 것으로 표시되는 하나 이상의 코어들에 보낼 수 있다(예를 들어, 스토리지 위치에 저장되는 코어 식별자 또는 코어 마스크에 기초하여 결정됨). 이러한 스누프 요청을 수신하는 코어(들)에 대한 하나 이상의 코어-사이드 어드레스 모니터 유닛(들)은 그 어드레스가 저장된 것인지를 결정하기 위해 그들의 대응 코어-사이드 어드레스 모니터 스토리지 위치들을 확인할 수 있다. 그리고 나서 하나 이상의 코어-사이드 어드레스 모니터 유닛(들) 각각은 캐시-사이드 어드레스 모니터 유닛에 그 어드레스가 여전히 유효한지(예를 들어, 대응 코어로부터 유효한 모니터 요청에 여전히 대응함)를 표시하는 응답을 다시 보낼 수 있다. 하나 이상의 코어-사이드 어드레스 모니터 유닛들로부터의 응답들이 그 어드레스에 대해 유효한 여전히 계류중인 모니터 요청들을 표시하면, 이러한 어드레스 및/또는 스토리지 위치는 오래된/구형의 것이 아닌 것으로 결정될 수 있다. 그렇지 않고, 어떠한 코어-사이드 어드레스 모니터 유닛도 그 어드레스에 대해 유효한 여전히 계류중인 모니터 요청을 보고하지 않으면, 그 어드레스 및/또는 스토리지 위치는 오래된/구형의 것으로 결정될 수 있다. 일부 실시예들에서는, 단일 스토리지 위치 및/또는 어드레스만이 이러한 접근방식을 사용하여 확인될 수 있다. 대안적으로, 다수의 스토리지 위치들 및/또는 어드레스들이 이러한 접근방식을 사용하여 확인될 수 있다.

[0043] 도 5를 다시 참조하면, 블록 572에서 최근에 수신된 모니터 요청을 다루는데 사용될 수 있는 이러한 오래된/구형의 캐시-사이드 어드레스 모니터 스토리지 위치가 존재한다고 결정되면(즉, 블록 572에서의 결정이 "예"임), 이러한 방법은 옵션으로 블록 573으로 진행할 수 있다. 블록 573에서, 오래된/구형의 캐시-사이드 어드레스 모

니터 스토리지 위치는 최근에 수신된 모니터 요청을 다루는데 옵션으로 사용될 수 있다. 유리하게도, 이 경우에 오버플로우 모드는 오래된/구형의 스토리지 위치를 이용하는 것에 의해 이 지점에서 회피될 수 있다.

[0044] 대안적으로, 블록 572에서 이러한 오래된/구형의 캐시-사이드 어드레스 모니터 스토리지 위치가 존재하지 않는다고 결정되면(즉, 블록 572에에서의 결정이 "아니오"임), 이러한 방법은 블록 574로 진행할 수 있다. 블록 574에서, 이러한 방법은 오버플로우 모드에 진입할 수 있다. 오버플로우 모드에 진입하는 단계는 오버플로우 정책들을 시행하거나 또는 구현하는 단계를 포함할 수 있다. 오버플로우에서 모드에서 성능은 다소 저하될 수 있다. 그러나, 종종 오버플로우 모드는 드물게 그리고 보통은 오버플로우 조건이 완화될 때까지의 비교적 짧은 주기들의 시간에 대해서만 구현될 필요만 있다.

[0045] 하나의 오버플로우 정책으로서, 블록 575에서, 이러한 방법은 모든 판독 트랜잭션이 공유된 캐시 코히어런시 상태를 사용하도록 강요하는 단계를 포함할 수 있다. 개념적으로 이것은 모든 판독 트랜잭션을 모니터 요청으로서 처리하는 것으로서 간주될 수 있다. 오버플로우 모드에 진입하면, 캐시-사이드 어드레스 모니터 유닛은 더 이상 전용 스토리지로 모니터 요청들/어드레스들을 추적할 수 없다. 따라서, 어떠한 코어도 캐시라인의 배타적 사본을 갖는 것이 허용되지 않을 수 있다. 예를 들어, 캐시-사이드 어드레스 모니터 유닛에 의해 수신되는 임의의 판독 동작은 공유된 상태 응답에 의해 다루어질 수 있다. 이러한 판독 트랜잭션들이 공유된 상태를 사용하도록 강요하는 것은, 대응 어드레스로의 기입 의도가, 스누프 또는 브로드캐스트로 하여금, 그 어드레스를 캐시하였을 수 있는 모든 코어들에게 제공되게 할 것이라는 점을 보장하게 하는데 도움이 될 수 있다.

[0046] 다른 오버플로우 정책으로서, 블록 576에서, 이러한 방법은 계류중인 모니터 요청들을 가질 가능성이 있는 모든 코어들에게 임의의 무효화 요청을 보내는 단계를 포함한다. 일부 실시예들에서, 이는 (예를 들어, 자신의 요청을 무효화하는 판독, 스누프 무효화 요청 등의 검출을 통해) 임의의 무효화 요청이 검출될 때 계류중인 모니터 요청들을 가질 가능성이 있는 프로세서의 및/또는 동일한 소켓 내의 모든 코어들을 무효화하는 스누프를 포함할 수 있다. 오버플로우 모드에 진입하면, 캐시-사이드 어드레스 모니터 유닛은 더 이상 전용 스토리지로 모니터 요청들/어드레스들을 추적할 수 없다. 따라서, 계류중인 모니터 요청들을 가질 가능성이 있는 모든 코어들은 모든 무효화 요청에 대해 정보를 받아야 한다. 이러한 스누프는, 모든 이러한 코어들의 코어-사이드 어드레스 모니터 유닛들에 도달할 수 있고, 관련된 어드레스에 대해 유효한 계류중인 모니터 요청이 있는 임의의 코어들에 적합할 때 모니터 트리거들을 제공할 수 있다.

[0047] 프로세서의 모든 코어들에게 통지하는 것이 엄격히 요구되지는 않고, 오히려 계류중인 모니터 요청들을 가질 가능성이 있는 모든 코어들에게만 그러하다는 점은 주목할 만한 가치가 있다. 일부 실시예들에서, 일 구조는 오버플로우가 발생할 때 계류중인 모니터 요청들을 가질 가능성이 있는 모든 코어들의 추적을 유지하는데 옵션으로 사용될 수 있다. 이러한 구조의 일 예는 옵션의 오버플로우 구조이다. 이러한 오버플로우 구조는 오버플로우가 발생할 때 어느 코어들이 계류중인 모니터 요청들을 가질 가능성이 있는지 표시할 수 있다. 일 예에서, 오버플로우 구조는 프로세서의 코어들의 총 수와 동일한 수의 비트들을 가질 수 있고, 각각의 비트는 상이한 대응 코어에 대해 고정된 대응관계를 가질 수 있다. 하나의 가능한 협약에 따르면, 각각의 비트는, 오버플로우가 발생할 때 대응 코어가 계류중인 모니터 요청을 가질 가능성이 있는 것을 표시하는 제1 값(예를 들어, 바이너리 1로 설정됨)을 가질 수 있거나, 또는 오버플로우가 발생할 때 대응 코어가 계류중인 모니터 요청을 가질 가능성이 없다는 것을 표시하는 제2 값(예를 들어, 바이너리 0으로 클리어됨)을 가질 수 있다.

[0048] 일 실시예에서, 오버플로우 구조는 그 자체로 오버플로우가 발생할 때 계류중인 모니터 요청들을 가질 가능성이 있는 코어들 모두를 반영할 수 있다. 예를 들어, 오버플로우가 발생할 때 오버플로우 구조는 캐시-사이드 어드레스 모니터 스토리지 위치들에 현재 저장된 임의의 하나 이상의 어드레스들에 대응하는 모든 코어들을 반영하도록 수정될 수 있다. 다른 실시예에서, 캐시-사이드 어드레스 모니터 스토리지 위치들과 조합된 오버플로우 구조는 오버플로우가 발생할 때 계류중인 모니터 요청들을 가질 가능성이 있는 코어들 모두를 반영할 수 있다. 예를 들어, 오버플로우가 발생할 때, 캐시-사이드 어드레스 모니터 스토리지 위치가 최근에 수신된 모니터 요청에 의해 중복기입(overwrite)되거나 또는 소비될 때마다, 중복기입되거나 또는 소비되는 어드레스들과 관련되는 코어들이 오버플로우 구조에서 반영될 수 있다. 즉, 오버플로우 구조는 계류중인 모니터 요청들을 가질 가능성이 있는 코어들에 대한 정보를 포착하기 위해 스토리지 엘리먼트가 중복기입될 때마다 업데이트될 수 있다. 이러한 실시예들에서, 오버플로우가 발생할 때 코어들이 계류중인 모니터 요청들을 가질 가능성이 있는지에 관한 정보는, 캐시-사이드 어드레스 모니터 스토리지 위치들과 오버플로우 구조 사이에서 분할된다.

[0049] 이러한 오버플로우 구조 또는 관련된 구조가 사용되는 실시예들에서는, 모든 코어들에게 임의의 수신된 무효화 요청을 보낼 것이 요구되지는 않고, 오히려 오버플로우 벡터에 의해 표시되는 코어들 및/또는 계류중인 모니터

요청들을 가질 가능성이 있는 스토리지 위치들에게만 그러하다. 일부 코어들은 오버플로우 벡터 및/또는 스토리지 위치들에 표시되지 않을 수 있고, 따라서 오버플로우가 발생할 때 어떠한 계류중인 모니터 요청들도 가질 가능성이 없고, 따라서 무효화 요청들이 보내질 필요가 없다. 그러나, 이러한 오버플로우 구조의 사용은 옵션이며 요구되는 것은 아니다.

[0050] 다시 도 5를 참조하면, 오버플로우 모드는 이용가능한 스토리지 위치들이 존재하지 않는 한 필요에 따라 블록들 575 및 576을 반복하는 것에 의해 계속될 수 있다. 그러나, 시간이 지나면서 오래된/구형의 어드레스들 및/또는 스토리지 위치들은 블록 576에서 계류중인 모니터 요청들을 가질 가능성이 있는 모든 코어들에게 임의의 무효화 요청들을 스누핑하거나 또는 다른 방식으로 보내는 것에 의해 적극적으로 제거될 수 있다. 코어-사이드 어드레스 모니터링 유닛들이 이러한 스누프들에 대해 유효한 계류중인 모니터 요청들 또는 무효화 요청을 가지고 있지 않으면, 이들은 이것에 관해 되돌려 보고할 수 있고, 이는 캐시-사이드 어드레스 모니터 유닛이, 코어가 어드레스를 모니터링하는데 관심이 없다는 것을 반영(예를 들어, 코어 마스크를 업데이트함)하게 할 수 있거나, 또는 다른 코어들이 어드레스에 관심이 없다면 스토리지 위치를 클리어하게 할 수 있다. 다양한 실시예들에서, 오래된/구형의 스토리지 위치들의 제거는, 특정 어드레스, 특정 캐시 부분, 특정 코어 등에 기초하여 수행될 수 있다. 오버플로우 마스크는 오래된/구형의 스토리지 위치들 또는 어드레스들의 클리닝 업(cleaning up)을 반영하도록 또한 수정될 수 있다. 예를 들어, 더 이상 계류중인 모니터 요청들을 갖지 않는 코어들은 오버플로우 마스크에서 1들 대신에 0들로 업데이트될 수 있다. 이러한 방식으로 블록 576에서 스누프들 또는 무효화 요청들은, 오버플로우 모드가 일부 지점에서 이탈될 수 있도록 시간이 지나면서 오래된/구형의 스토리지 엘리먼트들 또는 어드레스들을 클린 업하는데 도움이 될 수 있다. 블록 577에 도시되는 바와 같이, 이러한 오버플로우 모드는 이탈될 수 있다.

[0051] 이는 단지 하나의 예시적인 실시예이다. 이러한 실시예에 대한 많은 변경들이 고려된다. 예를 들어, 블록 572에서의 결정은 옵션이며 요구되는 것은 아니다. 다른 실시예들에서, 오버플로우 모드는 가능한 오래된 엔트리/어드레스에 대한 확인없이 자동으로 진입될 수 있다.

[0052] 도 6은 오버플로우 구조(680)의 일 실시예의 블록도이다. 이러한 오버플로우 구조는, 캐시-사이드 어드레스 모니터 스토리지 위치들의 세트 홀로 또는 이와 조합하여, 오버플로우가 발생할 때 어느 코어들이 계류중인 모니터 요청들을 가질 가능성이 있는지를 표시하는데 사용될 수 있다. 이러한 실시예에서, 오버플로우 구조는 N+1 코어들(예를 들어, 코어0 내지 코어 N) 중 상이한 하나에 대해 고정된 대응관계를 각각 갖는 N+1 비트들을 포함한다. 하나의 가능한 협약에 따르면, 각각의 비트는, 오버플로우가 발생할 때 대응 코어가 계류중인 모니터 요청을 가질 가능성이 있는 것을 표시하는 제1 값(예를 들어, 바이너리 1로 설정됨)을 가질 수 있거나, 또는 오버플로우가 발생할 때 대응 코어가 계류중인 모니터 요청을 가질 가능성이 없다는 것을 표시하는 제2 값(예를 들어, 바이너리 0으로 클리어됨)을 가질 수 있다. 예를 들어, 도면에서, 코어0에 대응하는 가장 좌측의 비트는 코어0이 어떠한 계류중인 모니터 요청들도 갖지 않는다는 것을 표시하는 바이너리 제로(즉, 0)를 갖고, 코어1에 대응하는 그 다음 가장 좌측의 비트는 코어1이 계류중인 모니터 요청을 갖는다는 것을 표시하는 바이너리 원(즉, 1)을 가지며, 코어N에 대응하는 가장 우측의 비트는 코어N이 계류중인 모니터 요청들을 갖지 않는다는 것을 표시하는 바이너리 제로(즉, 0)를 갖는다. 이는 적합한 오버플로우 구조의 단지 하나의 예시적인 예이다. 다른 구조들이 정보의 동일하거나 또는 유사한 타입들의 정보를 전달하는데 사용될 수 있다는 점이 이해되어야 한다. 예를 들어, 다른 실시예에서는, 계류중인 모니터 요청들을 갖는 코어 ID들의 리스트가 구조 등에 저장될 수 있다.

[0053] 본 명세서에 개시되는 모니터 메커니즘들, 뿐만 아니라, 이들의 유닛들 또는 컴포넌트들 중 임의의 것은, 하드웨어(예를 들어, 집적 회로, 트랜지스터들 또는 다른 회로 엘리먼트들 등), 펌웨어(예를 들어, ROM, EPROM, 플래시 메모리 또는 다른 영구적 또는 불휘발성 메모리 및 마이크로코드, 그 안에 저장되는 마이크로명령어들, 또는 다른 하위-레벨 명령어들), 소프트웨어 (예를 들어, 메모리에 저장되는 상위-레벨 명령어들), 또는 이들의 조합(예를 들어, 펌웨어 및/또는 소프트웨어 중 하나 이상과 잠재적으로 조합되는 하드웨어)으로 구현될 수 있다.

[0054] 도 1, 3, 4 및 6 중 임의의 것에 대해 설명된 컴포넌트들, 특징들, 및 상세사항들은 도 2 및 5 중 임의의 것에서 또한 옵션으로 사용될 수 있다. 또한, 장치 중 임의의 것에 대해 본 명세서에서 설명된 컴포넌트들, 특징들, 및 상세사항들은, 실시예들에서 이러한 장치에 의해 및/또는 이러한 장치를 이용하여 수행될 수 있는, 본 명세서에 설명된 방법들 중 임의의 것에서 또한 옵션으로 사용될 수 있다.

[0055] 예시적인 코어 아키텍처들, 프로세서들 및 컴퓨터 아키텍처들

[0056] 프로세서 코어들은 상이한 방식들로, 상이한 목적들을 위해, 상이한 프로세서들에서 구현될 수 있다. 예를 들어, 이러한 코어들의 구현들은: 1) 범용 컴퓨팅을 대상으로 하는 범용 순차적 코어; 2) 범용 컴퓨팅을 대상으로 하는 고 성능 범용 비순차적 코어; 3) 그래픽 및/또는 과학적 (쓰루풋) 컴퓨팅을 주로 대상으로 하는 특수 목적 코어를 포함할 수 있다. 상이한 프로세서들의 구현들은: 1) 범용 컴퓨팅을 대상으로 하는 하나 이상의 범용 순차적 코어들 및/또는 범용 컴퓨팅을 대상으로 하는 하나 이상의 범용 비순차적 코어들을 포함하는 CPU; 및 2) 그래픽 및/또는 과학적 (쓰루풋) 컴퓨팅을 주로 대상으로 하는 하나 이상의 특수 목적 코어들을 포함하는 코프로세서를 포함할 수 있다. 이러한 상이한 프로세서들은 상이한 컴퓨터 시스템 아키텍처들로 이어지며, 이는: 1) CPU와는 별개인 칩 상의 코프로세서; 2) CPU와 동일한 패키지 내의 별개의 다이 상의 코프로세서; 3) CPU와 동일한 다이 상의 코프로세서(이 경우에, 이러한 코프로세서를 때때로 통합 그래픽 및/또는 과학적 (쓰루풋) 로직 등의 특수 목적 로직이라고 하거나, 또는 특수 목적 코어들이라고 함); 및 4) 설명된 CPU(때때로 애플리케이션 코어(들) 또는 애플리케이션 프로세서(들)라고 함), 위에 개시된 코프로세서, 및 부가적인 기능성을 동일한 다이 상에 포함할 수 있는 시스템 온 칩(system on a chip)을 포함할 수 있다. 예시적인 코어 아키텍처들이 다음에 개시되고, 예시적인 프로세서들 및 컴퓨터 아키텍처들의 개시들이 후속된다.

[0057] ***예시적인 코어 아키텍처들***

[0058] **순차적 및 비순차적 코어 블록도**

[0059] 도 7a는 본 발명의 실시예들에 따라 예시적인 순차적 파이프라인 및 예시적인 레지스터 리네이밍, 비순차적 발행/실행 파이프라인 양자 모두를 도시하는 블록도이다. 도 7b는 본 발명의 실시예들에 따라 프로세서에 포함될 순차적 아키텍처 코어 및 예시적인 레지스터 리네이밍, 비순차적 발행/실행 아키텍처 코어 양자 모두의 예시적인 실시예를 도시하는 블록도이다. 도 7a-b에서 실선 박스들은 순차적 파이프라인 및 순차적 코어를 도시하는 한편, 점선 박스들의 선택적 추가는 레지스터 리네이밍, 비순차적 발행/실행 파이프라인 및 코어를 도시한다. 순차적 양상이 비순차적 양상의 서브세트라는 점을 고려하여, 비순차적 양상이 설명될 것이다.

[0060] 도 7a에서, 프로세서 파이프라인(700)은 페치 스테이지(702), 길이 디코드 스테이지(704), 디코드 스테이지(706), 할당 스테이지(708), 리네이밍 스테이지(710), (디스패치 또는 발행으로도 알려진) 스케줄링 스테이지(712), 레지스터 판독/메모리 판독 스테이지(714), 실행 스테이지(716), 라이트 백(write back)/메모리 기입 스테이지(718), 예외 처리 스테이지(722) 및 커미트(commit) 스테이지(724)를 포함한다.

[0061] 도 7b는 실행 엔진 유닛(750)에 연결되는 프론트 엔드 유닛(730)을 포함하는 프로세서 코어(790)를 도시하며, 이들 양자 모두는 메모리 유닛(770)에 연결된다. 코어(790)는 RISC(Reduced Instruction Set Computing) 코어, CISC(Complex Instruction Set Computing) 코어, VLIW(Very Long Instruction Word) 코어, 또는 하이브리드 또는 대안적인 코어 타입일 수 있다. 또 다른 옵션으로서, 코어(790)는, 예를 들어 네트워크 또는 통신 코어, 압축 엔진, 코프로세서 코어, GPGPU(General Purpose computing Graphics Processing Unit) 코어, 그래픽 코어 또는 이와 유사한 것 등의 특수 목적 코어일 수 있다.

[0062] 프론트 엔드 유닛(730)은 명령어 캐시 유닛(734)에 연결되는 분기 예측 유닛(732)을 포함하고, 명령어 캐시 유닛(734)은 명령어 TLB(Translation Lookaside Buffer)(736)에 연결되고, 명령어 TLB(736)는 명령어 페치 유닛(738)에 연결되고, 명령어 페치 유닛(738)은 디코드 유닛(740)에 연결된다. 디코드 유닛(740)(또는 디코더)은 명령어들을 디코딩할 수 있으며, 오리지널 명령어들로부터 디코딩되거나, 또는 그렇지 않으면 이들을 반영하거나, 또는 이들로부터 유도되는, 하나 이상의 마이크로-동작들, 마이크로-코드 엔트리 포인트들, 마이크로명령어들, 다른 명령어들 또는 다른 제어 신호들을 출력으로서 생성할 수 있다. 디코드 유닛(740)은 여러가지 상이한 메커니즘들을 사용하여 구현될 수 있다. 적합한 메커니즘들의 예들은, 이에 제한되는 것은 아니지만, 룩-업 테이블들, 하드웨어 구현들, PLA들(Programmable Logic Arrays), 마이크로코드 ROM(Read Only Memory)들 등을 포함한다. 일 실시예에서 코어(790)는 (예를 들어, 디코드 유닛(740)에 또는 그렇지 않으면 프론트 엔드 유닛(730) 내에) 특정 매크로 명령어들에 대한 마이크로코드를 저장하는 마이크로코드 ROM 또는 다른 매체를 포함한다. 디코드 유닛(740)은 실행 엔진 유닛(750)에서의 리네임/할당자 유닛(752)에 연결된다.

[0063] 실행 엔진 유닛(750)은 리타이어먼트 유닛(754) 및 하나 이상의 스케줄러 유닛(들)(756)의 세트에 연결되는 리네임/할당자 유닛(752)을 포함한다. 스케줄러 유닛(들)(756)은 예약 스테이션들, 중앙 명령어 윈도우 등을 포함하는 임의 수의 상이한 스케줄러들을 나타낸다. 스케줄러 유닛(들)(756)은 물리적 레지스터 파일(들) 유닛(들)(758)에 연결된다. 물리적 레지스터 파일(들) 유닛들(758) 각각은 하나 이상의 물리적 레지스터 파일들을 나타내고, 이들 중 상이한 것들은 스칼라 정수, 스칼라 부동 소수점, 팩킹된 정수, 팩킹된 부동 소수점, 벡터 정수, 벡터 부동 소수점, 상태(예를 들어, 실행될 다음 명령어의 어드레스인 명령어 포인터) 등의 하나 이상의

상이한 데이터 타입들을 저장한다. 일 실시예에서, 물리적 레지스터 파일(들) 유닛(758)은 벡터 레지스터 유닛, 기입 마스크 레지스터 유닛 및 스칼라 레지스터 유닛을 포함한다. 이러한 레지스터 유닛들은 아키텍처의 벡터 레지스터들, 벡터 마스크 레지스터들 및 범용 레지스터들을 제공할 수 있다. 물리적 레지스터 파일(들) 유닛(들)(758)은, 레지스터 리네이밍 및 비순차적 실행이 (예를 들어, 재배열 버퍼(들) 및 리타이어먼트 레지스터 파일(들)을 사용하여; 미래 파일(들), 이력 버퍼(들) 및 리타이어먼트 레지스터 파일(들)을 사용하여; 레지스터 맵들 및 레지스터들의 풀(pool)을 사용하여 등) 구현될 수 있는 다양한 방식들을 도시하도록 리타이어먼트 유닛(754)에 의해 오버랩된다. 리타이어먼트 유닛(754) 및 물리적 레지스터 파일(들) 유닛(들)(758)은 실행 클러스터(들)(760)에 연결된다. 실행 클러스터(들)(760)는 하나 이상의 실행 유닛들(762)의 세트 및 하나 이상의 메모리 액세스 유닛들(764)의 세트를 포함한다. 실행 유닛들(762)은 다양한 타입의 데이터(예를 들어, 스칼라 부동 소수점, 팩킹된 정수, 팩킹된 부동 소수점, 벡터 정수, 벡터 부동 소수점)에 대해 다양한 동작들(예로서, 시프트, 가산, 감산, 승산)을 수행할 수 있다. 일부 실시예들은 특정 펑션들이나 펑션들의 세트들에 전용의 다수의 실행 유닛들을 포함할 수 있지만, 다른 실시예들은 단 하나의 실행 유닛, 또는 모두가 모든 펑션들을 수행하는 다수의 실행 유닛을 포함할 수 있다. 스케줄러 유닛(들)(756), 물리적 레지스터 파일(들) 유닛(들)(758) 및 실행 클러스터(들)(760)는 복수 개일 수 있는 것으로 도시되는데, 그 이유는 특정 실시예들이 특정 타입들의 데이터/동작들에 대해 개별 파이프라인들(예를 들어, 자신들의 스케줄러 유닛, 물리적 레지스터 파일(들) 유닛 및/또는 실행 클러스터를 각각 갖는 스칼라 정수 파이프라인, 스칼라 부동 소수점/팩킹된 정수/팩킹된 부동 소수점/벡터 정수/벡터 부동 소수점 파이프라인 및/또는 메모리 액세스 파이프라인 - 그리고 개별 메모리 액세스 파이프라인의 경우, 이러한 파이프라인의 실행 클러스터만이 메모리 액세스 유닛(들)(764)을 갖는 특정 실시예들이 구현됨)을 생성하기 때문이다. 개별 파이프라인들이 사용되는 경우, 이들 파이프라인들 중 하나 이상은 비순차적 발행/실행일 수 있고 나머지는 순차적일 수 있다는 점도 이해되어야 한다.

[0064] 메모리 액세스 유닛들(764)의 세트는, 레벨 2(L2) 캐시 유닛(776)에 연결되는 데이터 캐시 유닛(774)에 연결되는 데이터 TLB 유닛(772)을 포함하는 메모리 유닛(770)에 연결된다. 예시적인 일 실시예에서, 메모리 액세스 유닛들(764)은 로드 유닛, 저장 어드레스 유닛 및 저장 데이터 유닛을 포함할 수 있으며, 이들 각각은 메모리 유닛(770) 내의 데이터 TLB 유닛(772)에 연결된다. 명령어 캐시 유닛(734)은 메모리 유닛(770) 내의 레벨 2(L2) 캐시 유닛(776)에 더 연결된다. L2 캐시 유닛(776)은 하나 이상의 다른 레벨들의 캐시에 그리고 궁극적으로 메인 메모리에 연결된다.

[0065] 예를 들어, 예시적인 레지스터 리네이밍, 비순차적 발행/실행 코어 아키텍처는 다음과 같이 파이프라인(700)을 구현할 수 있다: 1) 명령어 페치(738)는 페치 및 길이 디코딩 스테이지들(702 및 704)을 수행하고; 2) 디코드 유닛(740)은 디코드 스테이지(706)를 수행하고; 3) 리네임/할당자 유닛(752)은 할당 스테이지(708) 및 리네이밍 스테이지(710)를 수행하고; 4) 스케줄러 유닛(들)(756)은 스케줄 스테이지(712)를 수행하고; 5) 물리적 레지스터 파일(들) 유닛(들)(758) 및 메모리 유닛(770)은 레지스터 판독/메모리 판독 스테이지(714)를 수행하고; 실행 클러스터(760)는 실행 스테이지(716)를 수행하고; 6) 메모리 유닛(770) 및 물리적 레지스터 파일(들) 유닛(들)(758)은 라이트 백/메모리 기입 스테이지(718)를 수행하고; 7) 다양한 유닛들이 예외 처리 스테이지(722)에 관련될 수 있고; 8) 리타이어먼트 유닛(754) 및 물리적 레지스터 파일(들) 유닛(들)(758)은 커미트 스테이지(724)를 수행한다.

[0066] 코어(790)는, 본 명세서에서 개시되는 명령어(들)를 포함하는, 하나 이상의 명령어 세트들(예를 들어, (보다 새로운 버전들과 함께 추가된 일부 확장들을 갖는) x86 명령어 세트; 캘리포니아 서니베일의 MIPS 테크놀로지스의 MIPS 명령어 세트; 캘리포니아 서니베일의 ARM 홀딩스의 (NEON 등의 선택적 추가 확장들을 갖는) ARM 명령어 세트)을 지원할 수 있다. 일 실시예에서, 코어(790)는 팩킹된 데이터 명령어 세트 확장(예를 들어, AVX1, AVX2)을 지원하는 로직을 포함하며, 따라서 많은 멀티미디어 애플리케이션들에 의해 사용되는 동작들이 팩킹된 데이터를 사용하여 수행되는 것을 허용한다.

[0067] 코어는 (2 이상의 병렬 세트들의 동작이나 스레드들을 실행하는) 멀티스레딩을 지원할 수 있고, 시분할 멀티스레딩(time sliced multithreading), (단일의 물리적 코어가, 물리적 코어가 동시에 멀티스레딩할 수 있는 스레드들 각각에 대해 논리적 코어를 제공하는) 동시 멀티스레딩, 또는 이들의 조합(예를 들어, Intel® Hyperthreading 기술에서 등의 시분할 페칭 및 디코딩과 그 이후의 동시 멀티스레딩)을 포함하는 다양한 방식으로 멀티스레딩을 지원할 수 있다는 점이 이해되어야 한다.

[0068] 레지스터 리네이밍이 비순차적 실행의 정황에서 설명되었지만, 레지스터 리네이밍은 순차적 아키텍처에서 사용될 수도 있다는 점이 이해되어야 한다. 도시된 프로세서의 실시예는 또한 개별 명령어 및 데이터 캐시 유닛들(734/774) 및 공유 L2 캐시 유닛(776)을 포함하지만, 대안적인 실시예들은, 예를 들어, 레벨 1(L1) 내부 캐시

또는 다수 레벨들의 내부 캐시 등, 명령어들 및 데이터 양자 모두에 대해 단일 내부 캐시를 가질 수 있다. 일부 실시예들에서, 시스템은 내부 캐시와, 코어 및/또는 프로세서에 대해 외부에 있는 외부 캐시의 조합을 포함할 수 있다. 대안적으로, 모든 캐시는 코어 및/또는 프로세서에 대해 외부에 있을 수 있다.

[0069] **특정 예시적인 순차적 코어 아키텍처**

[0070] 도 8a-b는, 코어가 칩 내의 (동일 타입 및/또는 상이한 타입들의 다른 코어들을 포함하는) 여러 논리 블록 중 하나인, 보다 구체적인 예시적인 순차적 코어 아키텍처의 블록도를 도시한다. 논리 블록들은 애플리케이션에 따라 일부 고정된 펑션 로직, 메모리 I/O 인터페이스들 및 다른 필요한 I/O 로직과 고-대역폭 인터커넥트 네트워크(예를 들어, 링 네트워크)를 통해 통신한다.

[0071] 도 8a는, 본 발명의 실시예들에 따른, 싱글 프로세서 코어의 블록도로, 온-다이(on-die) 인터커넥트 네트워크(802)에 대한 접속, 및 레벨 2(L2) 캐시(804)의 로컬 서브세트와 함께 보여준다. 일 실시예에서, 명령어 디코더(800)는 팩킹된 데이터 명령어 세트 확장을 갖는 x86 명령어 세트를 지원한다. L1 캐시(806)는 스칼라 및 벡터 유닛들 내로의 캐시 메모리에 대한 저-지연(low-latency) 액세스들을 허용한다. 일 실시예에서는 (설계를 단순화하기 위해) 스칼라 유닛(808) 및 벡터 유닛(810)이 별개의 레지스터 세트들(각각, 스칼라 레지스터들(812) 및 벡터 레지스터들(814))을 사용하고 이들 간에 이동되는 데이터는 메모리에 기입된 다음 레벨 1(L1) 캐시(806)로부터 다시 판독되지만, 본 발명의 대안적 실시예들은 상이한 접근방식을 사용할 수 있다(예를 들어, 단일 레지스터 세트를 사용하거나, 또는 기입 및 다시 판독되지 않고 2개의 레지스터 파일들 사이에서 데이터가 이동되는 것을 허용하는 통신 경로를 포함함).

[0072] L2 캐시(804)의 로컬 서브세트는, 프로세서 코어 당 하나씩인 개별 로컬 서브세트들로 분할되는 글로벌 L2 캐시의 일부이다. 각각의 프로세서 코어는 L2 캐시(804)의 자신의 로컬 서브세트에 대한 직접 액세스 경로를 갖는다. 프로세서 코어에 의해 판독된 데이터는 자신의 L2 캐시 서브세트(804)에 저장되며, 다른 프로세서 코어들이 그들 자신의 로컬 L2 캐시 서브세트들에 액세스하는 것과 병렬로 빠르게 액세스될 수 있다. 프로세서 코어에 의해 기록된 데이터는 자신의 L2 캐시 서브세트(804)에 저장되며, 필요한 경우에는, 다른 서브세트들로부터 제거된다. 링 네트워크는 공유 데이터에 대한 코히어런시(coherency)를 보장한다. 링 네트워크는 양-방향성이어서, 프로세서 코어들, L2 캐시들 및 다른 논리 블록들 등의 에이전트들이 칩 내에서 상호 통신하는 것을 허용한다. 각각의 링 데이터-경로는 방향 당 1012-비트 폭이다.

[0073] 도 8b는 본 발명의 실시예들에 따른 도 8a에서의 프로세서 코어의 부분 확대도이다. 도 8b는, L1 캐시(804)의 L1 데이터 캐시(806A) 부분은 물론, 벡터 유닛(810) 및 벡터 레지스터들(814)에 관한 보다 많은 상세를 포함한다. 구체적으로, 벡터 유닛(810)은 정수, 단일 정밀도 부동 및 이중 정밀도 부동 명령어들 중 하나 이상을 실행하는 16-폭 VPU(Vector Processing Unit)(16-폭 ALU(828) 참조)이다. VPU는, 스위즐(swizzle) 유닛(820)에 의한 레지스터 입력들의 스위즐링, 수치 변환 유닛들(822A-B)에 의한 수치 변환 및 메모리 입력에 대한 복제 유닛(824)에 의한 복제를 지원한다. 기입 마스크 레지스터들(826)은 결과적인 벡터 기입들을 서술하는 것(predicating)을 허용한다.

[0074] ***통합 메모리 제어기 및 그래픽들을 갖는 프로세서***

[0075] 도 9는, 본 발명의 실시예들에 따라, 둘 이상의 코어를 가질 수 있고, 통합 메모리 제어기를 가질 수 있고, 및 통합 그래픽을 가질 수 있는 프로세서(900)의 블록도이다. 도 9의 실선 박스들은 싱글 코어(902A), 시스템 에이전트(910), 하나 이상의 버스 제어기 유닛들(916)의 세트를 갖는 프로세서(900)를 도시하는 한편, 옵션인 점선 박스들의 추가는 다수의 코어들(902A-N), 시스템 에이전트 유닛(910) 내의 하나 이상의 통합 메모리 제어기 유닛(들)(914)의 세트, 및 특수 목적 로직(908)을 갖는 대안적인 프로세서(900)를 도시한다.

[0076] 따라서, 프로세서(900)의 상이한 구현들은: 1) 통합 그래픽 및/또는 과학적 (쓰루풋) 로직(하나 이상의 코어들을 포함할 수 있음)인 특수 목적 로직(908) 및 하나 이상의 범용 코어들(예를 들어, 범용 순차적 코어들, 범용 비순차적 코어들, 이 두 가지의 조합)인 코어들(902A-N)을 갖는 CPU; 2) 그래픽 및/또는 과학적 (쓰루풋) 컴퓨팅을 주로 대상으로 하는 다수의 특수 목적 코어들인 코어들(902A-N)을 갖는 코프로세서; 및 3) 다수의 범용 순차적 코어들인 코어들(902A-N)을 갖는 코프로세서를 포함할 수 있다. 따라서, 프로세서(900)는 범용 프로세서, 코프로세서 또는 특수 목적 프로세서, 예를 들어 네트워크 또는 통신 프로세서, 압축 엔진, 그래픽 프로세서, GPGPU(General Purpose Graphics Processing Unit), 하이-쓰루풋 MIC(Many Integrated Core) 코프로세서(30개 이상의 코어를 포함함), 임베디드 프로세서, 또는 이와 유사한 것 등일 수 있다. 프로세서는 하나 이상의 칩들 상에 구현될 수 있다. 프로세서(900)는, 예를 들어, BiCMOS, CMOS, 또는 NMOS 등의 다수의 프로세스 기술들 중

임의의 것을 사용하여 하나 이상의 기판들의 일부가 될 수 있고 및/또는 이들 기판 상에 구현될 수 있다.

[0077] 메모리 계층구조는 코어들 내의 하나 이상의 레벨들의 캐시, 하나 이상의 공유 캐시 유닛들(906)의 세트, 및 통합 메모리 제어기 유닛들(914)의 세트에 연결되는 외부 메모리(도시되지 않음)를 포함한다. 공유 캐시 유닛들(906)의 세트는, 예를 들어 레벨 2(L2), 레벨 3(L3), 레벨 4(L4) 또는 다른 레벨의 캐시 등의 하나 이상의 중간 레벨 캐시들, 최종 레벨 캐시(LLC) 및/또는 이들의 조합들을 포함할 수 있다. 일 실시예에서는 링 기반 인터커넥트 유닛(912)이 통합 그래픽 로직(908), 공유 캐시 유닛들(906)의 세트 및 시스템 에이전트 유닛(910)/통합 메모리 제어기 유닛(들)(914)을 인터커넥트하지만, 대안 실시예들은 이러한 유닛들을 인터커넥트하는 임의 수의 공지된 기술들을 이용할 수 있다. 일 실시예에서, 하나 이상의 캐시 유닛들(906)과 코어들(902A-N) 사이에는 코히어런시가 유지된다.

[0078] 일부 실시예들에서, 코어들(902A-N) 중 하나 이상은 멀티-스레딩이 가능하다. 시스템 에이전트(910)는 코어들(902A-N)을 조정 및 조작하는 컴포넌트들을 포함한다. 시스템 에이전트 유닛(910)은 예를 들어 PCU(Power Control Unit) 및 디스플레이 유닛을 포함할 수 있다. PCU는 코어들(902A-N) 및 통합 그래픽 로직(908)의 전력 상태를 조절하는 데 필요한 로직 및 컴포넌트들이거나 이들을 포함할 수 있다. 디스플레이 유닛은 하나 이상의 외부 접속되는 디스플레이들을 구동하기 위한 것이다.

[0079] 코어들(902A-N)은 아키텍처 명령어 세트와 관련하여 동종 또는 이종일 수 있다; 즉, 코어들(902A-N) 중 둘 이상은 동일 명령어 세트를 실행할 수 있는 반면, 다른 코어들은 그 명령어 세트의 서브세트 또는 상이한 명령어 세트만을 실행할 수 있다.

[0080] ***예시적인 컴퓨터 아키텍처***

[0081] 도 10-13은 예시적인 컴퓨터 아키텍처들의 블록도들이다. 랩톱들, 데스크톱들, 핸드헬드 PC들, 퍼스널 디지털 어시스턴트들, 엔지니어링 워크스테이션들, 서버들, 네트워크 디바이스들, 네트워크 허브들, 스위치들, 임베디드 프로세서들, 디지털 신호 프로세서들(DSPs), 그래픽 디바이스들, 비디오 게임 디바이스들, 셋-톱 박스들, 마이크로 제어기들, 셀 폰들, 휴대용 미디어 플레이어들, 핸드헬드 디바이스들 및 다양한 다른 전자 디바이스들에 대한 기술분야에 알려진 다른 시스템 설계들 및 구성들도 적합하다. 일반적으로, 본 명세서에 개시된 바와 같은 프로세서 및/또는 다른 실행 로직을 통합할 수 있는 매우 다양한 시스템들 또는 전자 디바이스들이 일반적으로 적합하다.

[0082] 이제 도 10을 참조하면, 본 발명의 일 실시예에 따른 시스템(1000)의 블록도가 도시된다. 시스템(1000)은 하나 이상의 프로세서들(1010, 1015)을 포함할 수 있고, 이는 제어기 허브(1020)에 결합된다. 일 실시예에서, 제어기 허브(1020)는 GMCH(Graphics Memory Controller Hub)(1090) 및 IOH(Input/Ouput Hub)(1050)(개별 칩들 상에 존재할 수 있음)를 포함하고; GMCH(1090)는 메모리(1040) 및 코프로세서(1045)에 연결되는 메모리 및 그래픽 제어기들을 포함하고; IOH(1050)는 I/O(Input/Output) 디바이스들(1060)을 GMCH(1090)에 연결한다. 대안적으로, 메모리 및 그래픽 제어기들 중 하나 또는 양자 모두는 (본 명세서에서 개시되는 바와 같이) 프로세서 내에 통합되고, 메모리(1040) 및 코프로세서(1045)는 프로세서(1010) 및 IOH(1050)와 단일 칩에 있는 제어기 허브(1020)에 직접 연결된다.

[0083] 추가적인 프로세서들(1015)의 옵션적 속성이 도 10에 파선들로 표시된다. 각각의 프로세서(1010, 1015)는 본 명세서에 개시되는 처리 코어들 중 하나 이상을 포함할 수 있고, 프로세서(1000)의 일부 버전일 수 있다.

[0084] 메모리(1040)는, 예를 들어, DRAM(Dynamic Random Access Memory), PCM(Phase Change Memory), 또는 이 둘의 조합일 수 있다. 적어도 하나의 실시예에 대해, 제어기 허브(1020)는 FSB(Front Side Bus), QPI(QuickPath Interconnect) 등의 지점-대-지점 인터페이스, 또는 유사한 접속(1095) 등의 멀티-드롭 버스를 통해 프로세서(들)(1010, 1015)와 통신한다.

[0085] 일 실시예에서, 코프로세서(1045)는 예를 들어 하이-스루풋 MIC 프로세서, 네트워크 또는 통신 프로세서, 압축엔진, 그래픽 프로세서, GPGPU, 임베디드 프로세서 등의 특수 목적 프로세서이다. 일 실시예에서, 제어기 허브(1020)는 통합 그래픽 가속기를 포함할 수 있다.

[0086] 아키텍처, 마이크로 아키텍처, 열, 전력 소비 특성들 등을 포함하는 장점의 다양한 메트릭들과 관련하여 물리적 리소스들(1010, 1015) 사이에는 다양한 차이점들이 존재할 수 있다.

[0087] 일 실시예에서, 프로세서(1010)는 일반적인 타입의 데이터 처리 작업들을 제어하는 명령어들을 실행한다. 명령어들 내에는 코프로세서 명령어들이 내장될 수 있다. 프로세서(1010)는 이러한 코프로세서 명령어들을 부속된

코프로세서(1045)에 의해 실행되어야 하는 타입의 것으로 인식한다. 따라서, 프로세서(1010)는 이러한 코프로세서 명령어들(또는 코프로세서 명령어들을 나타내는 제어 신호들)을 코프로세서 버스 또는 다른 인터커넥트 상에서 코프로세서(1045)에 발행한다. 코프로세서(들)(1045)는 수신된 코프로세서 명령어들을 수락 및 실행한다.

[0088] 이제, 도 11을 참조하면, 본 발명의 일 실시예에 따른 제1의 보다 구체적인 예시적인 시스템(1100)의 블록도가 도시된다. 도 11에 도시된 바와 같이, 멀티프로세서 시스템(1100)은 지점-대-지점 인터커넥트 시스템이며, 지점-대-지점 인터커넥트(1150)을 통해 연결되는 제1 프로세서(1170) 및 제2 프로세서(1180)를 포함한다. 프로세서들(1170, 1180) 각각은 일부 버전의 프로세서(900)일 수 있다. 본 발명의 일 실시예에서, 프로세서들(1170, 1180)은 각각 프로세서들(1010, 1015)이고, 코프로세서(1138)는 코프로세서(1045)이다. 다른 실시예에서는, 프로세서들(1170, 1180)이 각각 프로세서(1010) 및 코프로세서(1045)이다.

[0089] 프로세서들(1170, 1180)은 각각 IMC(Integrated Memory Controller) 유닛들(1172, 1182)을 포함하는 것으로 도시된다. 프로세서(1170)는 또한 그의 버스 제어기 유닛들의 일부로서 P-P(Point-to-Point) 인터페이스들(1176, 1178)을 포함한다; 유사하게 제2 프로세서(1180)는 P-P 인터페이스들(1186, 1188)을 포함한다. 프로세서들(1170, 1180)은 P-P 인터페이스(Point-to-Point) 회로들(1178, 1188)을 이용하여 P-P 인터페이스(1150)를 통해 정보를 교환할 수 있다. 도 11에 도시된 바와 같이, IMC들(1172 및 1182)은 프로세서들을 각자의 메모리, 즉 메모리(1132) 및 메모리(1134)에 연결하며, 이들 메모리는 각 프로세서에 국부적으로 부속되는 메인 메모리의 일부일 수 있다.

[0090] 프로세서들(1170, 1180)은 포인트 투 포인트 인터페이스 회로들(1176, 1194, 1186, 1198)을 사용하여 개별 P-P 인터페이스들(1152, 1154)을 통해 칩셋(1190)과 정보를 각각 교환할 수 있다. 칩셋(1190)은 고-성능 인터페이스(1139)를 통해 코프로세서(1138)와 정보를 선택적으로 교환할 수 있다. 일 실시예에서, 코프로세서(1138)는 예를 들어 하이-쓰루풋 MIC 프로세서, 네트워크 또는 통신 프로세서, 압축 엔진, 그래픽 프로세서, GPGPU, 임베디드 프로세서 등 특수 목적 프로세서이다.

[0091] 공유된 캐시(도시되지 않음)는 어느 한 프로세서에 포함되거나, 양자 모두의 프로세서의 외부이지만 여전히 P-P 인터커넥트을 통해 프로세서들과 접속될 수 있어서, 프로세서가 저 전력 모드에 놓이는 경우 어느 한쪽 또는 양자 모두의 프로세서의 로컬 캐시 정보가 공유된 캐시에 저장될 수 있다.

[0092] 칩셋(1190)은 인터페이스(1196)를 통해 제1 버스(1116)에 연결될 수 있다. 일 실시예에서, 제1 버스(1116)는 PCI(Peripheral Component Interconnect) 버스일 수 있거나, 또는 PCI 익스프레스 버스 또는 다른 3세대 I/O 인터커넥트 버스 등의 버스일 수 있지만, 본 발명의 범위가 이에 제한되는 것은 아니다.

[0093] 도 11에 도시된 바와 같이, 다양한 I/O 디바이스들(1114)이 제1 버스(1116)에 연결될 수 있으며, 이와 함께 버스 브릿지(1118)가 제1 버스(1116)를 제2 버스(1120)에 연결한다. 일 실시예에서는, 코프로세서들, 하이-쓰루풋 MIC 프로세서들, GPGPU들, 가속기들(예를 들어, 그래픽 가속기 또는 DSP(Digital Signal Processing) 유닛 등), 필드 프로그래머블 게이트 어레이들 또는 임의의 다른 프로세서 등 하나 이상의 추가적인 프로세서(들)(1115)가 제1 버스(1116)에 연결된다. 일 실시예에서, 제2 버스(1120)는 LPC(Low Pin Count) 버스일 수 있다. 일 실시예에서는, 예를 들어 키보드 및/또는 마우스(1122), 통신 디바이스들(1127) 및 명령어들/코드 및 데이터(1130)를 포함할 수 있는 디스크 드라이브 또는 기타 대용량 저장 디바이스 등의 저장 유닛(1128)을 포함하는 다양한 디바이스들이 제2 버스(1120)에 연결될 수 있다. 또한, 오디오 I/O(1124)가 제2 버스(1120)에 연결될 수 있다. 다른 아키텍처들도 가능하다는 점에 주의한다. 예를 들어, 도 11의 지점-대-지점 아키텍처 대신에, 시스템은 멀티-드롭 버스 또는 다른 그러한 아키텍처를 구현할 수 있다.

[0094] 이제, 도 12를 참조하면, 본 발명의 일 실시예에 따른 제2의 보다 구체적인 예시적인 시스템(1200)의 블록도가 도시된다. 도 11 및 12에서 동일한 엘리먼트들은 동일한 참조 번호들을 가지며, 도 11의 특정 양상들은 도 12의 다른 양상들을 모호하게 하는 것을 회피하기 위해 도 12로부터 생략되었다.

[0095] 도 12는 프로세서들(1170, 1180)이 각각 통합 메모리 및 I/O 제어 로직("CL")(1172, 1182)을 포함할 수 있다는 점을 도시한다. 따라서, CL(1172, 1182)은 통합 메모리 제어기 유닛들을 포함하며, I/O 제어 로직을 포함한다. 도 12는 메모리들(1132, 1134)이 CL(1172, 1182)에 연결될 뿐만 아니라, I/O 디바이스들(1214) 또한 제어 로직(1172, 1182)에 연결된다는 것을 도시한다. 레거시 I/O 디바이스들(1215)은 칩셋(1190)에 연결된다.

[0096] 이제, 도 13을 참조하면, 본 발명의 일 실시예에 따른 SoC(1300)의 블록도가 도시된다. 도 9에서의 유사한 엘리먼트들은 동일한 참조 번호를 갖는다. 또한, 점선 박스는 더욱 개선된 SoC들에 관한 선택적 특징들이다. 도 13에서, 인터커넥트 유닛(들)(1302)은: 하나 이상의 코어들(202A-N)의 세트 및 공유 캐시 유닛(들)(906)을 포함

하는 애플리케이션 프로세서(1310); 시스템 에이전트 유닛(910); 버스 제어기 유닛(들)(916); 통합 메모리 제어기 유닛(들)(914); 통합 그래픽 로직, 이미지 프로세서, 오디오 프로세서 및 비디오 프로세서를 포함할 수 있는 하나 이상의 코프로세서들(1320)의 세트; SRAM(Static Random Access Memory) 유닛(1330); DMA(Direct Memory Access) 유닛(1332); 및 하나 이상의 외부 디스플레이들에 연결하기 위한 디스플레이 유닛(1340)에 연결된다. 일 실시예에서, 코프로세서(들)(1320)는, 예를 들어, 네트워크 또는 통신 프로세서, 압축 엔진, GPGPU, 하이-쓰루풋 MIC 프로세서, 임베디드 프로세서 등의 특수 목적 프로세서를 포함한다.

[0097] 본 명세서에 개시되는 메커니즘들의 실시예들은 하드웨어, 소프트웨어, 펌웨어 또는 이러한 구현 접근방식들의 조합으로 구현될 수 있다. 본 발명의 실시예들은 적어도 하나의 프로세서, 스토리지 시스템(휘발성 및 불휘발성 메모리 및/또는 스토리지 엘리먼트들을 포함함), 적어도 하나의 입력 디바이스, 및 적어도 하나의 출력 디바이스를 포함하는 프로그래머블 시스템들 상에서 실행되는 컴퓨터 프로그램들 또는 프로그램 코드로서 구현될 수 있다.

[0098] 도 11에 도시된 코드(1130) 등의 프로그램 코드는 본 명세서에 개시되는 평션들을 수행하고 출력 정보를 생성하기 위한 입력 명령어들에 적용될 수 있다. 출력 정보는 알려진 방식으로 하나 이상의 출력 디바이스에 적용될 수 있다. 본 출원의 목적으로, 처리 시스템은, 예를 들어, DSP(Digital Signal Processor), 마이크로제어기, ASIC(Application Specific Integrated Circuit) 또는 마이크로프로세서 등의 프로세서를 갖는 임의의 시스템을 포함한다.

[0099] 프로그램 코드는 하이 레벨 절차적 또는 객체 지향적 프로그래밍 언어로 구현되어 처리 시스템과 통신할 수 있다. 프로그램 코드는, 또한, 요구되는 경우, 어셈블리 또는 기계 언어로 구현될 수 있다. 사실상, 본 명세서에 개시되는 메커니즘들이 임의의 특정 프로그래밍 언어로 범위가 제한되는 것은 아니다. 어느 경우에나, 언어는 컴파일되거나 또는 해석되는 언어일 수 있다.

[0100] 적어도 일 실시예의 하나 이상의 양상은, 머신에 의해 판독될 때 머신으로 하여금 본 명세서에서 개시되는 기술들을 수행하는 로직을 제조하게 하는, 프로세서 내의 다양한 로직을 표현하는, 머신 판독-가능 매체 상에 저장되는 대표적인 명령어들에 의해 구현될 수 있다. "IP 코어들"로서 알려진 그러한 표현들은 유형의 머신 판독가능 매체 상에 저장될 수 있으며, 다양한 고객들 또는 제조 설비에 공급되어, 로직 또는 프로세서를 실제로 제작하는 제조 머신들 내에 로드될 수 있다.

[0101] 이러한 머신-판독가능 스토리지 매체들은, 하드 디스크들, 플로피 디스크들, 광 디스크들, CD-ROM들(Compact Disk Read-Only Memories), CD-RW들(Compact Disk ReWritable's) 및 광자기 디스크들 포함하는 임의의 다른 타입의 디스크들, ROM들(Read-Only Memories), DRAM들(Dynamic Random Access Memories), SRAM들(Static Random Access Memories) 등의 RAM들(Random Access Memories), EPROM들(Electrically Erasable Programmable Read-Only Memories), 플래시 메모리들, EEPROM들(Electrically Erasable Programmable Read-Only Memories), PCM(Phase Change Memory) 등의 반도체 장치, 자기 또는 광학 카드, 또는 전자적 명령어들을 저장하기에 적합한 임의의 다른 타입의 매체와 같은 스토리지 매체를 포함하는 머신 또는 디바이스에 의해 제조되거나 또는 형성되는 물품들의 비-일시적이고 유형인 배열들을 포함할 수 있고, 이에 제한되는 것은 아니다.

[0102] 따라서, 본 발명의 실시예들은, 또한, 명령어들을 포함하거나, 또는 본 명세서에 개시되는 구조들, 회로들, 장치들, 프로세서들 및/또는 시스템 특징들을 정의하는, HDL(Hardware Description Language) 등의 설계 데이터를 포함하는 비-일시적이고 유형인 머신 판독가능 매체를 포함한다. 이러한 실시예들은 또한 프로그램 제품들이라고 할 수 있다.

[0103] ***에뮬레이션(바이너리 번역, 코드 모핑 등을 포함함)***

[0104] 일부 경우에는, 명령어 변환기가 소스 명령어 세트로부터 타겟 명령어 세트로 명령어를 변환하는데 사용될 수 있다. 예를 들어, 명령어 변환기는 코어에 의해 처리될 하나 이상의 다른 명령어들로 명령어를 (예를 들어, 정적 바이너리 번역, 동적 컴필레이션을 포함하는 동적 바이너리 번역을 이용하여) 번역하거나, 모프하거나, 에뮬레이트하거나, 또는 다른 방식으로 변환할 수 있다. 명령어 변환기는 소프트웨어, 하드웨어, 펌웨어, 또는 그의 조합으로 구현될 수 있다. 명령어 변환기는 온 프로세서(on processor), 오프 프로세서(off processor), 또는 부분 온 및 부분 오프 프로세서(part on and part off processor)일 수 있다.

[0105] 도 14는 본 발명의 실시예들에 따라 소스 명령어 세트 내의 바이너리 명령어들을 타겟 명령어 세트 내의 바이너리 명령어들로 변환하는 소프트웨어 명령어 변환기의 사용을 대조하는 블록도이다. 도시된 실시예에서, 명령어 변환기는 소프트웨어 명령어 변환기이지만, 대안적으로 명령어 변환기가 소프트웨어, 펌웨어, 하드웨어, 또는

이들의 다양한 조합들로 구현될 수 있다. 도 14는 하이 레벨 언어(1402)의 프로그램을 x86 컴파일러(1404)를 사용하여 컴파일하여, 적어도 하나의 x86 명령어 세트 코어를 갖는 프로세서(1416)에 의해 선천적으로 실행될 수 있는 x86 바이너리 코드(1406)를 생성할 수 있다는 것을 도시한다. 적어도 하나의 x86 명령어 세트 코어를 갖는 프로세서(1416)는, 적어도 하나의 x86 명령어 세트 코어를 갖는 인텔 프로세서와 실질적으로 동일한 결과를 달성하기 위해서, (1) 인텔 x86 명령어 세트 코어의 명령어 세트의 상당 부분 또는 (2) 적어도 하나의 x86 명령어 세트 코어를 갖는 인텔 프로세서 상에서 실행되는 것을 목적으로 하는 오브젝트 코드 버전들의 애플리케이션들 또는 다른 소프트웨어를 호환가능하게 실행하거나 또는 다른 방식으로 처리함으로써, 적어도 하나의 x86 명령어 세트 코어를 갖는 인텔 프로세서와 실질적으로 동일한 펑션을 수행할 수 있는 임의의 프로세서를 나타낸다. x86 컴파일러(1404)는 추가적인 링크 처리(linkage processing)를 갖거나 갖지 않고서 적어도 하나의 x86 명령어 세트 코어를 갖는 프로세서(1416) 상에서 실행될 수 있는 x86 바이너리 코드(1406)(예를 들어, 오브젝트 코드)를 생성하도록 작동될 수 있는 컴파일러를 나타낸다. 유사하게, 도 14는 하이 레벨 언어(1402)의 프로그램을 대안적인 명령어 세트 컴파일러(1408)를 사용하여 컴파일하여, 적어도 하나의 x86 명령어 세트 코어를 갖지 않는 프로세서(1414)(예를 들어, 캘리포니아주 서니베일의 MIPS 테크놀로지스의 MIPS 명령어 세트를 실행하고/실행하거나 캘리포니아주 서니베일의 ARM 홀딩스의 ARM 명령어 세트를 실행하는 코어들을 갖는 프로세서)에 의해 선천적으로 실행될 수 있는 대안적인 명령어 세트 바이너리 코드(1410)를 생성할 수 있다는 점을 도시한다. 명령어 변환기(1412)는 x86 바이너리 코드(1406)를, x86 명령어 세트 코어(1414)를 갖지 않는 프로세서에 의해 선천적으로 실행될 수 있는 코드로 변환하는데 사용된다. 이러한 변환된 코드는 대안적인 명령어 세트 바이너리 코드(1410)와 동일할 가능성이 없는데, 그 이유는 이를 행할 수 있는 명령어 변환기를 제조하기 어렵기 때문이다; 그러나, 변환된 코드는 일반적인 작업을 달성할 것이며, 대안적인 명령어 세트로부터의 명령어들로 이루어질 것이다. 따라서, 명령어 변환기(1412)는, 에뮬레이션, 시뮬레이션 또는 임의의 다른 프로세스를 통해 x86 명령어 세트 프로세서 또는 코어를 갖지 않는 프로세서 또는 다른 전자 디바이스가 x86 바이너리 코드(1406)를 실행하는 것을 허용하는 소프트웨어, 펌웨어, 하드웨어 또는 이들의 조합을 나타낸다.

[0106] 설명 및 청구범위에서, "연결되는(coupled)" 및/또는 "접속되는(connected)"이라는 용어들이, 이들의 파생어와 함께, 사용될 수 있다. 이들 용어가 상호 동의어로서 의도되는 것은 아니라는 점이 이해되어야 한다. 오히려, 실시예들에서, "접속되는"이란 2 이상의 엘리먼트들이 상호 직접적인 물리적 및/또는 전기적 접촉을 이루고 있다는 점을 나타내는데 사용될 수 있다. "연결되는"이란 2 이상의 엘리먼트들이 직접적인 물리적 및/또는 전기적 접촉을 이루고 있다는 점을 의미할 수 있다. 그러나, "연결되는"이란 2 이상의 엘리먼트들이 상호 직접적인 접촉을 이루고 있지는 않지만, 여전히 상호 협력하거나 상호작용하고 있다는 점을 또한 의미할 수 있다. 예를 들어, 코어는 하나 이상의 중간 컴포넌트들을 통해 캐시 부분과 연결될 수 있다. 도면들에서, 화살표들은 접속들 및 연결들을 보여주는데 사용된다.

[0107] 설명 및 청구범위에서, "로직(logic)", "유닛(unit)", "모듈(module)", 또는 "컴포넌트(component)"라는 용어가 사용되었을 수 있다. 이들은, 하드웨어, 펌웨어, 소프트웨어 또는 이들의 조합을 포함할 수 있다는 점이 이해되어야 한다. 이들의 예들은, 집적 회로, 주문형 집적 회로들, 아날로그 회로들, 디지털 회로들, 프로그램 논리 디바이스들, 명령어들을 포함하는 메모리 디바이스들 등을 포함한다. 일부 실시예들에서, 이들은 잠재적으로 트랜지스터들 및/또는 게이트들 및/또는 다른 회로 컴포넌트들을 포함할 수 있다.

[0108] 위의 설명에서는 설명의 목적으로 본 발명의 실시예들의 충분한 이해를 제공하기 위해 다양한 구체적 상세사항들이 설명되었다. 그러나, 다른 실시예들이 이들 구체적 상세사항의 일부 없이 실시될 수 있다. 본 발명의 범위는 위에 제공되는 특정 예들에 의해서가 아니라 청구항들에 의해서만 결정되어야 한다. 다른 경우들에서, 잘 알려진 회로들, 구조들, 디바이스들, 및 동작들은 설명의 이해를 불명료하게 하는 것을 회피하기 위해 블록도 형태로 또는 상세사항 없이 도시되었다. 다수의 컴포넌트들이 도시되고 설명되는 경우들에서, 일부 경우들에서 이들은 단일 컴포넌트로서 함께 통합될 수 있다. 단일 컴포넌트가 도시되고 설명되는 다른 경우들에서, 일부 경우들에서 이것은 2개 이상의 컴포넌트들로 분할될 수 있다.

[0109] 다양한 동작들 및 방법들이 설명되었다. 방법들 중 일부는 흐름도에서 비교적 기본적인 형태로 설명되었지만, 동작들은 옵션으로 방법들에 추가될 수 있거나 및/또는 방법들에서 제거될 수 있다. 또한, 흐름도들이 실시예들에 따른 동작들의 특정 순서를 도시하지만, 그 특정 순서는 예시적이다. 대안적인 실시예들은 옵션으로 상이한 순서로 동작들을 수행할 수 있고, 특정 동작들을 조합할 수 있고, 특정 동작들을 중복할 수 있는 등등이다.

[0110] 특정 동작들은 하드웨어 컴포넌트들에 의해 수행될 수 있거나, 또는 명령어들로 프로그램된 머신, 회로 또는 하드웨어 컴포넌트(예를 들어, 프로세서, 프로세서의 일부, 회로 등)가 동작들을 수행하는 것을 야기시키고/시키거나 초래하는데 이용될 수 있는 머신 실행가능 또는 회로 실행가능 명령어들로 구현될 수 있다. 또한, 동작들

은 하드웨어와 소프트웨어의 조합에 의해 선택적으로 수행될 수 있다.

[0111] 일부 실시예들은 비-일시적 머신-판독가능 스토리지 매체를 포함하는 제조 물품(예를 들어, 컴퓨터 프로그램 제품)을 포함한다. 이러한 비-일시적 머신-판독가능 스토리지 매체는 일시적 전파되는 신호를 포함하지 않는다. 이러한 비-일시적 머신-판독가능 스토리지 매체는 머신에 의해 판독가능한 형태로 정보를 저장하는 메커니즘을 포함할 수 있다. 이러한 머신-판독가능 스토리지 매체는 머신에 의해 실행되면 및/또는 실행될 때 머신으로 하여금 본 명세서에 개시되는 하나 이상의 동작들, 방법들, 또는 기술들을 수행하게 하고 및/또는 수행하는 결과가 머신에 생기게 하도록 동작가능한 명령어 또는 명령어들의 시퀀스를 저장할 수 있다. 적합한 머신들의 예들은, 이에 제한되는 것은 아니지만, 프로세서들 및 이러한 프로세서들을 갖는 컴퓨터 시스템들 또는 다른 전자 디바이스들을 포함한다. 다양한 예들로서, 비-일시적 머신-판독가능 스토리지 매체는, 플로피 디스켓, 광 스토리지 매체, 광 디스크, CD-ROM, 자기 디스크, 광자기 디스크, ROM(Read Only Memory), PROM(Programmable ROM), EPROM(Erasable-and-Programmable ROM), EEPROM(Electrically-Erasable-and-Programmable ROM), RAM(Random Access Memory), SRAM(Static-RAM), DRAM(Dynamic-RAM), 플래시 메모리, 상 변화 메모리, 상 변화 데이터 스토리지 재료, 불휘발성 메모리, 불휘발성 데이터 스토리지 디바이스, 비-일시적 메모리, 비-일시적 데이터 스토리지 디바이스를 포함할 수 있다.

[0112] 본 명세서 전반에 걸쳐서 예를 들어, "일 실시예", "실시예", "하나 이상의 실시예들", "일부 실시예들"에 대한 언급은, 예를 들어, 특정한 특징이 본 발명의 실시에 포함될 수 있지만 반드시 그럴 필요는 없다는 것을 나타낸다. 유사하게, 본 개시내용을 간소화하고 다양한 본 발명의 양상들의 이해를 도울 목적으로, 설명에서는 다양한 특징들이 때때로 단일 실시예, 도면, 또는 그의 설명에서 함께 그룹화된다. 그러나, 이러한 개시의 방법이, 본 발명은 각 청구범위에 명백하게 기재된 것보다 더 많은 특징들을 요구하는 의도를 반영하는 것으로서 해석되어서는 안 된다. 오히려, 이하 청구범위들이 반영하는 바에 따라, 본 발명의 양상들은 단일 개시된 실시예의 모든 특징들보다 적게 놓일 수 있다. 따라서, 상세한 설명에 후속하는 청구범위들은 이에 의해 본 상세한 설명에 명백하게 통합되고, 각 청구범위는 본 발명의 개별 실시예로서 자립한다.

[0113] **예시적인 실시예들**

[0114] 이하 예들은 추가적 실시예들에 관련된다. 이러한 예들에서의 상세사항들은 하나 이상의 실시예들 어디에서나 사용될 수 있다.

[0115] 예 1은 분산형 캐시의 제1 캐시 부분에 대응하고, 프로세서의 논리 프로세서들의 총 수보다 적은 총 수의 캐시-사이드 어드레스 모니터 스토리지 위치들을 갖는 캐시-사이드 어드레스 모니터 유닛을 포함하는 프로세서이다. 각각의 캐시-사이드 어드레스 모니터 스토리지 위치는 모니터링될 어드레스를 저장하기 위한 것이다. 이러한 프로세스는, 제1 코어에 대응하고, 제1 코어의 하나 이상의 논리 프로세서들의 수와 동일한 수의 코어-사이드 어드레스 모니터 스토리지 위치들을 갖는 코어-사이드 어드레스 모니터 유닛을 또한 포함한다. 각각의 코어-사이드 어드레스 모니터 스토리지 위치는 모니터링될 어드레스 및 제1 코어의 상이한 대응 논리 프로세서에 대한 모니터 상태를 저장하기 위한 것이다. 이러한 프로세서는 어떠한 미사용된 캐시-사이드 어드레스 모니터 스토리지 위치도 모니터링될 추가적 어드레스를 저장하는데 이용가능하지 않을 때 어드레스 모니터 스토리지 오버플로우 정책을 시행하는, 제1 캐시 부분에 대응하는 캐시-사이드 어드레스 모니터 스토리지 오버플로우 유닛을 또한 포함한다.

[0116] 예 2는 임의의 선행 예의 프로세서를 포함하고, 제1 코어에 대응하고 코어-사이드 어드레스 모니터 유닛과 연결되는 코어-사이드 트리거 유닛을 옵션으로 포함한다. 이러한 코어-사이드 트리거 유닛은, 대응하는 코어-사이드 어드레스 모니터 스토리지 위치가 트리거할 준비가 된 모니터 상태를 갖고 트리거 이벤트가 검출될 때, 제1 코어의 논리 프로세서를 트리거하기 위한 것이다.

[0117] 예 3은 임의의 선행 예의 프로세서를 포함하고, 캐시-사이드 어드레스 모니터 유닛과 연결되고, 동일한 모니터 어드레스에 대한 상이한 논리 프로세서들로부터의 모니터 요청들을 공통 캐시-사이드 어드레스 모니터 스토리지 위치에 기록하는 캐시-사이드 어드레스 모니터 스토리지 위치 재사용 유닛을 옵션으로 포함한다.

[0118] 예 4는 예 3의 프로세서를 포함하고, 공통 캐시-사이드 어드레스 모니터 스토리지 위치는 동일한 모니터 어드레스에 대한 모니터 요청들을 제공한 상이한 논리 프로세서들을 기록하는 구조를 포함한다.

[0119] 예 5는 임의의 선행 예의 프로세서를 포함하고, 이러한 프로세서는 40개보다 많은 하드웨어 스레드들을 갖고, 제1 캐시 부분에 대응하는 캐시-사이드 어드레스 모니터 유닛의 캐시-사이드 어드레스 모니터 스토리지 위치들의 총 수는 적어도 20개의 캐시-사이드 어드레스 모니터 스토리지 위치들이지만, 40개보다 많은 하드웨어 스레

드들의 총 수보다 적다.

[0120] 예 6은 임의의 선행 예의 프로세서를 포함하고, 캐시-사이드 어드레스 모니터 유닛의 캐시-사이드 어드레스 모니터 스토리지 위치들의 총 수는 캐시-사이드 어드레스 모니터 스토리지 위치들의 오버플로우의 가능성이 십만분의 일을 넘지 않을 만큼 프로세서의 논리 프로세서들의 총 수에 비해 충분하다.

[0121] 예 7은 임의의 선행 예의 프로세서를 포함하고, 모니터링될 제1 어드레스를 표시하는 명령어에 응답하여, 캐시-사이드 어드레스 모니터 유닛은 제1 어드레스를 캐시-사이드 어드레스 모니터 스토리지 위치에 저장하기 위한 것이다. 또한, 코어-사이드 어드레스 모니터 유닛은 제1 어드레스를 코어-사이드 어드레스 모니터 스토리지 위치에 저장하기 위한 것이다.

[0122] 예 8은 임의의 선행 예의 프로세서를 포함하고, 논리 프로세서들은 하드웨어 스레드들이다.

[0123] 예 9는 임의의 선행 예의 프로세서를 포함하고, 캐시-사이드 어드레스 모니터 스토리지 오퍼플로우 유닛은 판독 트랜잭션들이 공유된 상태를 사용하도록 강요하는 것을 포함하는 어드레스 모니터 스토리지 오버플로우 정책을 시행하기 위한 것이다.

[0124] 예 10은 임의의 선행 예의 프로세서를 포함하고, 캐시-사이드 어드레스 모니터 스토리지 오버플로우 유닛은 계류중인 모니터 요청을 가질 가능성이 있는 모든 코어들에게 무효화 요청들을 보내는 것을 포함하는 어드레스 모니터 스토리지 오버플로우 정책을 시행하기 위한 것이다.

[0125] 예 11은 예 10의 프로세서를 포함하고, 캐시-사이드 어드레스 모니터 스토리지 오버플로우 유닛은 오버플로우 구조를 확인하여 어느 코어들이 계류중인 모니터 요청을 가질 가능성이 있는지 결정하기 위한 것이다.

[0126] 예 12는 인터커넥트, 및 인터커넥트와 연결되는 프로세서를 포함하는 명령어들을 처리하는 시스템이다. 이러한 프로세서는 분산형 캐시의 제1 캐시 부분에 대응하고, 프로세서의 하드웨어 스레드들의 총 수보다 적은 총 수의 어드레스 모니터 스토리지 위치들을 갖는 캐시 부분 제어 유닛의 제1 어드레스 모니터 유닛을 포함한다. 각각의 어드레스 모니터 스토리지 위치는 모니터링될 어드레스를 저장하기 위한 것이다. 이러한 프로세서는 제1 코어에 대응하고, 제1 코어의 하나 이상의 하드웨어 스레드들의 수와 동일한 수의 어드레스 모니터 스토리지 위치들을 갖는 코어 인터페이스 유닛의 제2 어드레스 모니터 유닛을 또한 포함한다. 제2 어드레스 모니터 유닛의 각각의 어드레스 모니터 스토리지 위치는 모니터링될 어드레스 및 제1 코어의 상이한 대응 하드웨어 스레드에 대한 모니터 상태를 저장하기 위한 것이다. 이러한 프로세서는 제1 어드레스 모니터 유닛의 모든 어드레스 모니터 스토리지 위치들이 사용되고 이들 중 어느 것도 모니터 요청에 대한 어드레스를 저장하는데 이용가능하지 않을 때 어드레스 모니터 스토리지 오버플로우 정책을 구현하는, 캐시 부분 제어 유닛의 어드레스 모니터 스토리지 오버플로우 유닛을 더 포함한다. 이러한 시스템은 인터커넥트와 연결되는 다이나믹 랜덤 액세스 메모리, 인터커넥트와 연결되는 무선 통신 디바이스, 및 인터커넥트와 연결되는 이미지 캡처 디바이스를 또한 포함한다.

[0127] 예 13은 예 12의 시스템을 포함하고, 어드레스 모니터 스토리지 오버플로우 유닛은, 판독 트랜잭션들이 공유된 상태를 사용하도록 강요하는 것; 및 계류중인 모니터 요청을 가질 가능성이 있는 모든 코어들에게 무효화 요청들을 보내는 것을 포함하는 어드레스 모니터 스토리지 오버플로우 정책을 구현하기 위한 것이다.

[0128] 예 14는 예 12-13 중 임의의 것의 시스템을 포함하고, 프로세서는 40개보다 많은 하드웨어 스레드들을 갖고, 제1 어드레스 모니터 유닛의 어드레스 모니터 스토리지 위치들의 총 수는 적어도 20개이지만 프로세서의 40개보다 많은 하드웨어 스레드들의 총 수보다 적다.

[0129] 예 15는 예 12-14 중 임의의 것의 시스템을 포함하고, 프로세서는 동일한 모니터 어드레스에 대한 상이한 하드웨어 스레드들로부터의 모니터 요청들을 공통 어드레스 모니터 스토리지 위치에 기록하는 캐시 부분 제어 유닛의 어드레스 모니터 스토리지 위치 재사용 유닛을 더 포함한다.

[0130] 예 16은 어드레스를 표시하고, 멀티-코어 프로세서의 제1 코어의 제1 논리 프로세서에서 어드레스로의 기입들에 대해 모니터할 것을 표시하는 제1 명령어를 수신하는 단계를 포함하는 프로세서에서의 방법이다. 제1 명령어에 응답하여, 본 방법은, 제1 명령어에 의해 표시되는 어드레스를 제1 코어에 대응하는 복수의 코어-사이드 어드레스 모니터 스토리지 위치들 중 제1 코어-사이드 어드레스 모니터 스토리지 위치에 저장하는 단계를 포함한다. 복수의 코어-사이드 어드레스 모니터 스토리지 위치들의 수는 제1 코어의 논리 프로세서들의 수와 동일하다. 본 방법은 제1 명령어에 의해 표시되는 어드레스를 분산형 캐시의 제1 캐시 부분에 대응하는 복수의 캐시-사이드 어드레스 모니터 스토리지 위치들 중 제1 캐시-사이드 어드레스 모니터 스토리지 위치에 저장하는 단계를 또한 포함한다. 복수의 캐시-사이드 어드레스 모니터 스토리지 위치들의 총 수는 멀티-코어 프로세서의 논리 프

로세서들의 총 수보다 적다. 본 방법은 모니터 상태를 추정 상태로 변경하는 단계를 더 포함한다.

[0131] 예 17은 예 16의 방법을 포함하고, 어드레스를 또한 표시하고 제2 코어의 제2 논리 프로세서에서 어드레스로의 기입들에 대해 모니터할 것을 표시하는 제2 명령어를 수신하는 단계, 및 제2 코어에 대한 어드레스에 대한 모니터 요청을 제1 캐시-사이드 어드레스 모니터 스토리지 위치에 기록하는 단계를 옵션으로 포함한다.

[0132] 예 18은 예 17의 방법을 포함하고, 제2 코어에 대한 어드레스에 대한 모니터 요청을 제1 캐시-사이드 어드레스 모니터 스토리지 위치에 기록하는 단계는 멀티-코어 프로세서의 각각의 코어에 대응하는 상이한 비트를 갖는 코어 마스크에서의 비트를 변경하는 단계를 포함한다.

[0133] 예 19는 임의의 선행 예의 방법을 포함하고, 제2 어드레스를 표시하고 제1 논리 프로세서에서 제2 어드레스로의 기입들에 대해 모니터할 것을 표시하는 제2 명령어를 수신하는 단계, 제1 캐시 부분에 대응하는 복수의 캐시-사이드 어드레스 모니터 스토리지 위치들 중에 이용가능한 캐시-사이드 어드레스 모니터 스토리지 위치들이 없는 것으로 결정하는 단계, 및 캐시-사이드 어드레스 모니터 스토리지 위치 오버플로우 모드에 진입할 것을 결정하는 단계를 옵션으로 포함한다.

[0134] 예 20은 예 19의 방법을 포함하고, 캐시-사이드 어드레스 모니터 스토리지 위치 오버플로우 모드에 있는 동안, 제1 캐시 부분에 대응하는 모든 판독 트랜잭션들이 공유된 캐시 코히어런시 상태를 사용할 것을 강요하는 단계, 및 하나 이상의 계류중인 모니터 요청들을 가질 가능성이 있는 멀티-코어 프로세서의 모든 코어들에게 제1 캐시 부분에 대응하는 무효화 요청들을 보내는 단계를 옵션으로 포함한다.

[0135] 예 21은 임의의 선행 예의 방법을 포함하고, 제1 논리 프로세서에서 어드레스를 표시하는 제2 명령어를 수신하는 단계, 및 제2 명령어에 응답하여, 모니터 상태를 트리거 대기 상태로 변경하는 단계를 옵션으로 포함한다.

[0136] 예 22는 예 16-21 중 임의의 것을 방법을 수행하는 프로세서 또는 다른 장치를 포함한다.

[0137] 예 23은 예 16-21 중 임의의 것의 방법을 수행하는 수단을 포함하는 프로세서 또는 장치를 포함한다.

[0138] 예 24는 예 16-21 중 임의의 것의 방법을 수행하는 집적 회로 및/또는 로직 및/또는 유닛들 및/또는 컴포넌트들 및/또는 모듈들, 및/또는 수단, 또는 이들의 임의의 조합을 포함하는 프로세서를 포함한다.

[0139] 예 25는 머신에 의해 실행되면 및/또는 실행될 때 머신으로 하여금 예 16-21 중 임의의 것의 방법을 수행하게 하는 하나 이상의 명령어들을 옵션으로 저장하거나 또는 다른 방식으로 제공하는 옵션으로 비-일시적 머신-판독가능 매체를 포함한다.

[0140] 예 26은 인터커넥트, 인터커넥트와 연결되는 프로세서, DRAM, 그래픽 칩, 무선 통신 칩, 상 변화 메모리, 및 비디오 카메라 중 적어도 하나를 포함하는 컴퓨터 시스템을 포함하고, 이러한 적어도 하나는 인터커넥트와 연결되며, 프로세서 및/또는 컴퓨터 시스템은 예 16-21 중 임의의 것의 방법을 수행하기 위한 것이다.

[0141] 예 27은 본 명세서에 설명되는 바와 같은 하나 이상의 동작들 또는 임의의 방법을 실질적으로 수행하는 프로세서 또는 다른 장치를 포함한다.

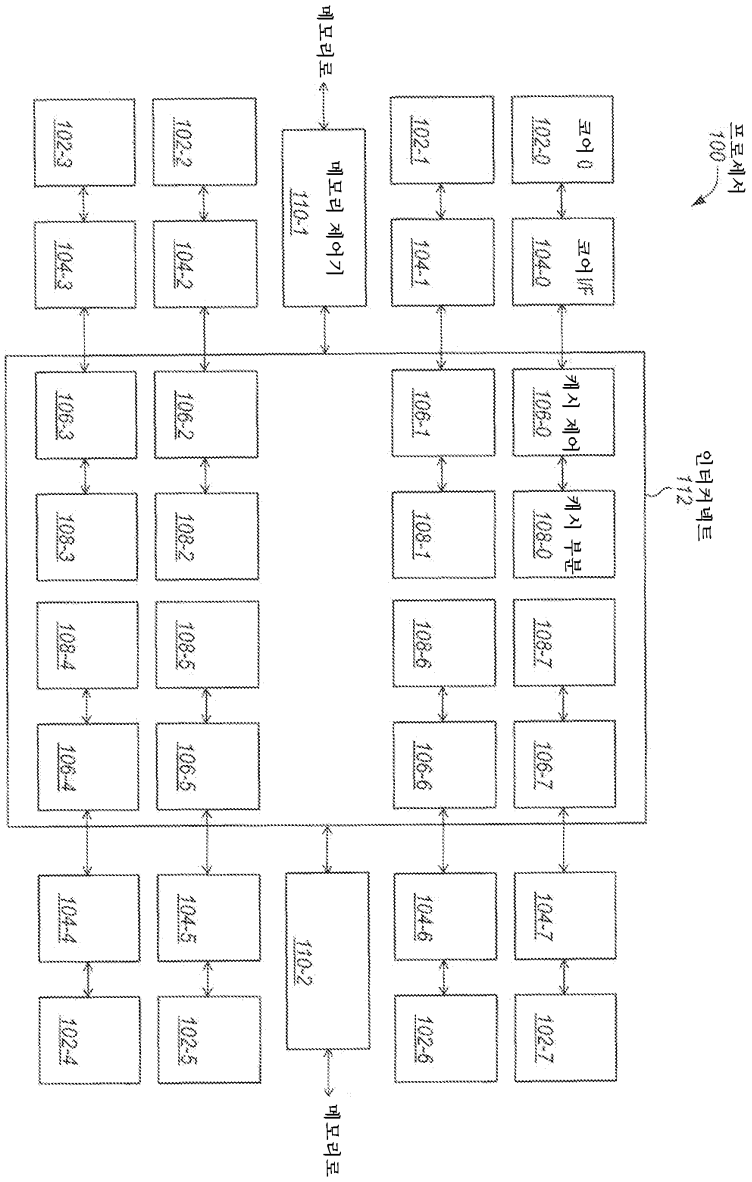
[0142] 예 28은 본 명세서에 설명되는 바와 같은 하나 이상의 동작들 또는 임의의 방법을 실질적으로 수행하는 수단을 포함하는 프로세서 또는 다른 장치를 포함한다.

[0143] 예 29는 본 명세서에 설명되는 바와 같은 명령어를 실질적으로 수행하는 프로세서 또는 다른 장치를 포함한다.

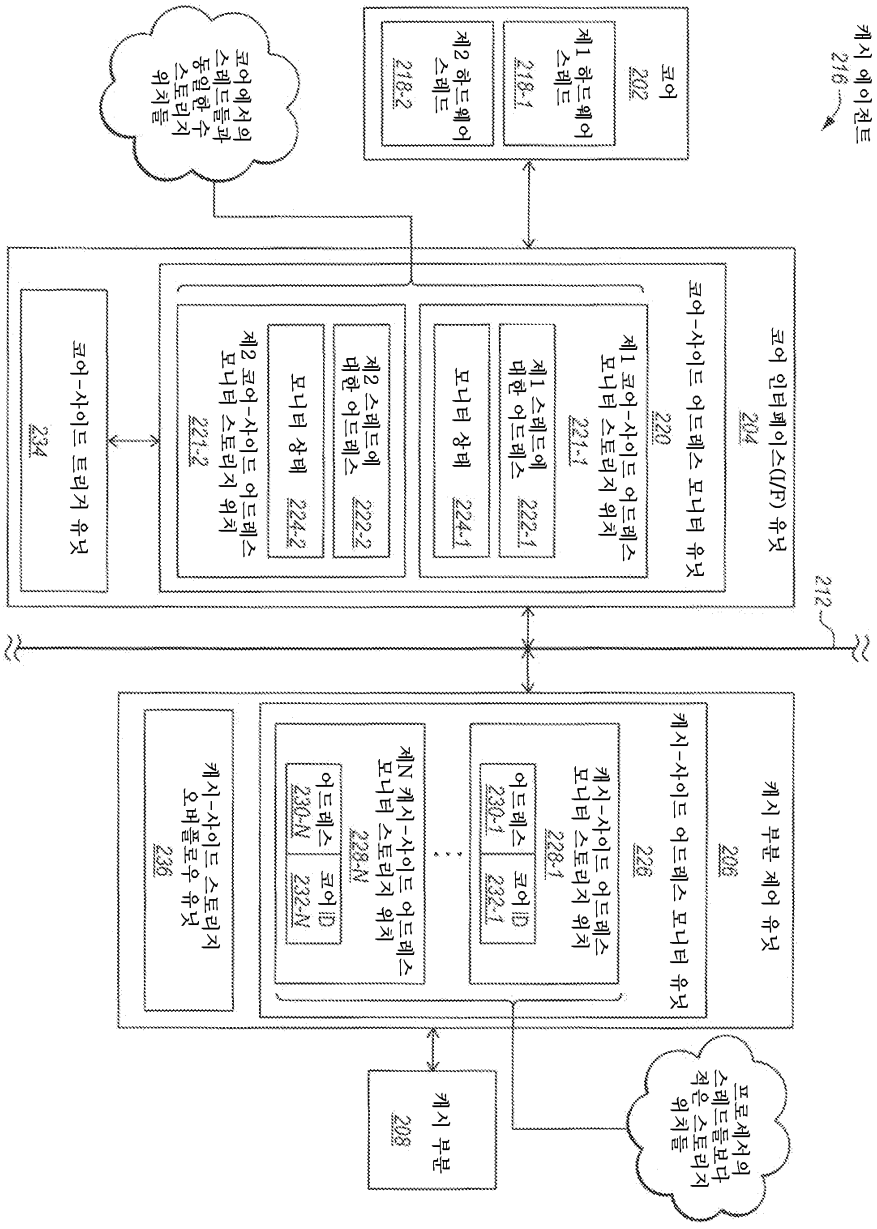
[0144] 예 30은 본 명세서에 설명되는 바와 같은 명령어를 실질적으로 수행하는 수단을 포함하는 프로세서 또는 다른 장치를 포함한다.

도면

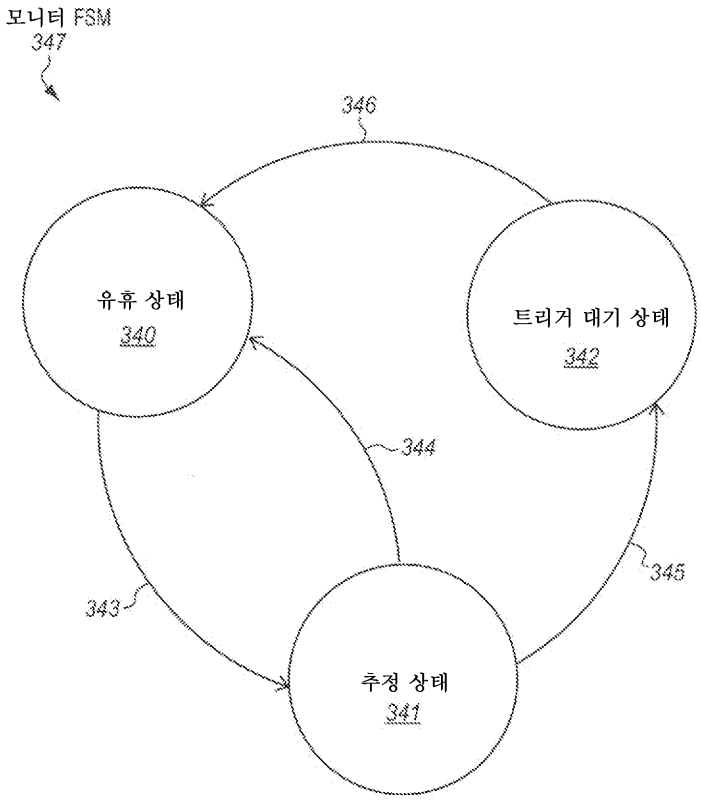
도면1


프로세서
100
코어 0
102-0
코어 I/F
104-0
캐시 제어
106-0
캐시 뷰분
108-0
인터커넥트
112
메모리로
메모리 제어기
110-1
102-1
104-1
106-1
108-1
102-2
104-2
106-2
108-2
102-3
104-3
106-3
108-3
108-4
106-4
108-5
106-5
104-4
102-4
104-5
102-5
110-2
메모리로
108-6
106-6
104-6
102-6
108-7
106-7
104-7
102-7

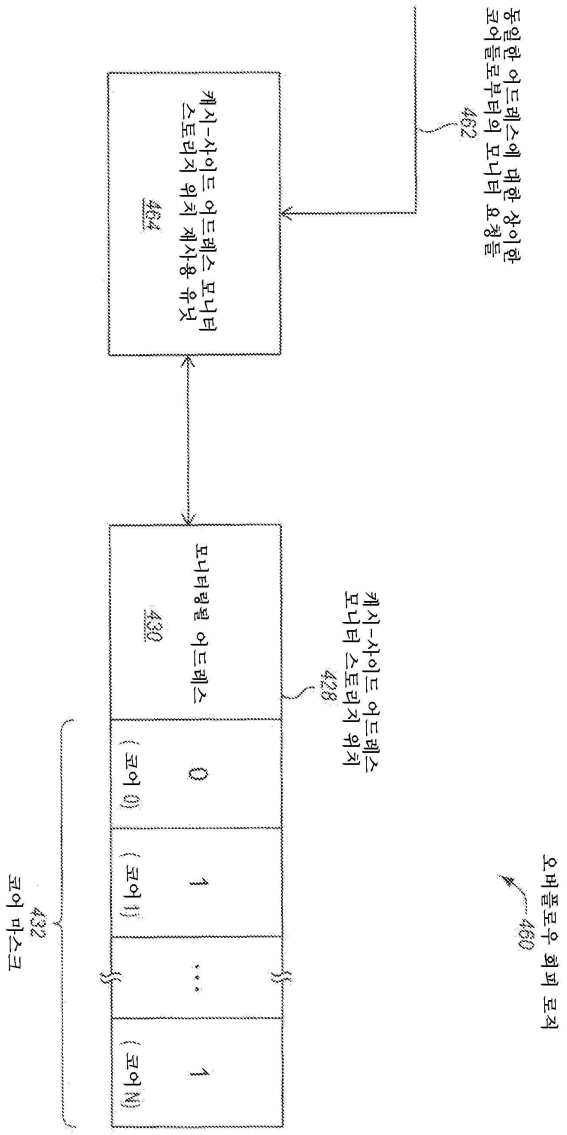
도면2


케시 에이전트
216
코어
202
제1 하드웨어 스레드
218-1
제2 하드웨어 스레드
218-2
코어에서의 스레드들과 동일한 수 스토리지 위치들
코어 인터페이스(I/F) 유닛
214
코어-사이드 어드레스 모니터 유닛
220
제1 코어-사이드 어드레스 모니터 스토리지 위치
221-1
제1 스레드에 대한 어드레스
222-1
모니터 상태
224-1
제2 스레드에 대한 어드레스
222-2
모니터 상태
224-2
제2 코어-사이드 어드레스 모니터 스토리지 위치
221-2
코어-사이드 트리거 유닛
234
212
케시 부분 제어 유닛
206
케시-사이드 어드레스 모니터 유닛
226
케시-사이드 어드레스 모니터 스토리지 위치
228-1
어드레스
230-1
코어 ID
232-1
제N 케시-사이드 어드레스 모니터 스토리지 위치
228-N
어드레스
230-N
코어 ID
232-N
케시-사이드 스토리지 오버플로우 유닛
236
케시 부분
208
프로세서의 스레드들보다 적은 스토리지 위치들

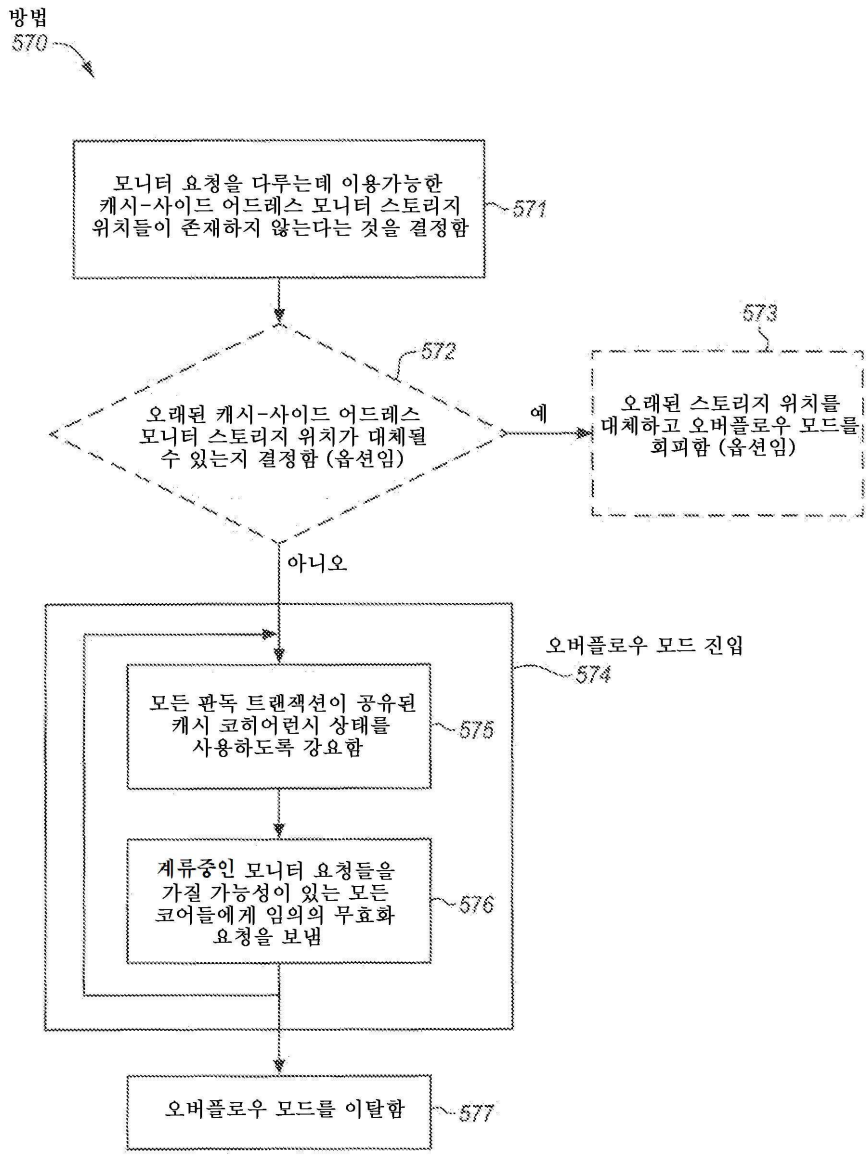
도면3



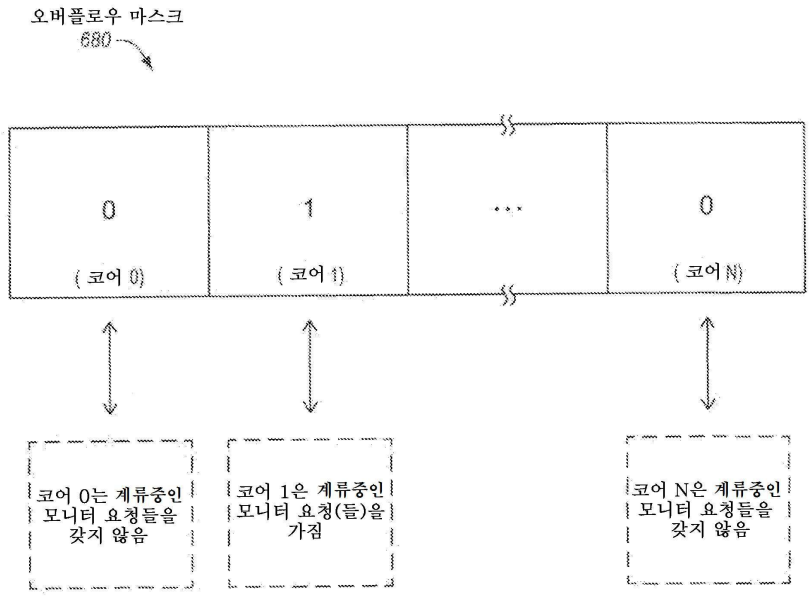
도면4


동일한 어드레스에 대한 상이한 코어들로부터의 모니터 요청들
462
캐시-사이드 어드레스 모니터 스토리지 위치 제거용 유닛
464
캐시-사이드 어드레스 모니터 스토리지 위치
428
모니터링될 어드레스
430
0
1
...
1
(코어 0)
(코어 1)
(코어 N)
코어 마스크
432
오버플로우 회피 로직
460

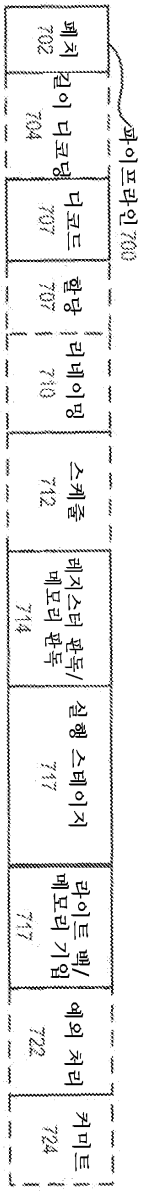
도면5


방법
570
모니터 요청을 다루는데 이용가능한 캐시-사이드 어드레스 모니터 스토리지 위치들이 존재하지 않는다는 것을 결정함
571
572
오래된 캐시-사이드 어드레스 모니터 스토리지 위치가 대체될 수 있는지 결정함 (옵션임)
예
573
오래된 스토리지 위치를 대체하고 오버플로우 모드를 회피함 (옵션임)
아니오
오버플로우 모드 진입
574
모든 판독 트랜잭션이 공유된 캐시 코히어런시 상태를 사용하도록 강요함
575
계류중인 모니터 요청들을 가질 가능성이 있는 모든 코어들에게 임의의 무효화 요청을 보냄
576
오버플로우 모드를 이탈함
577

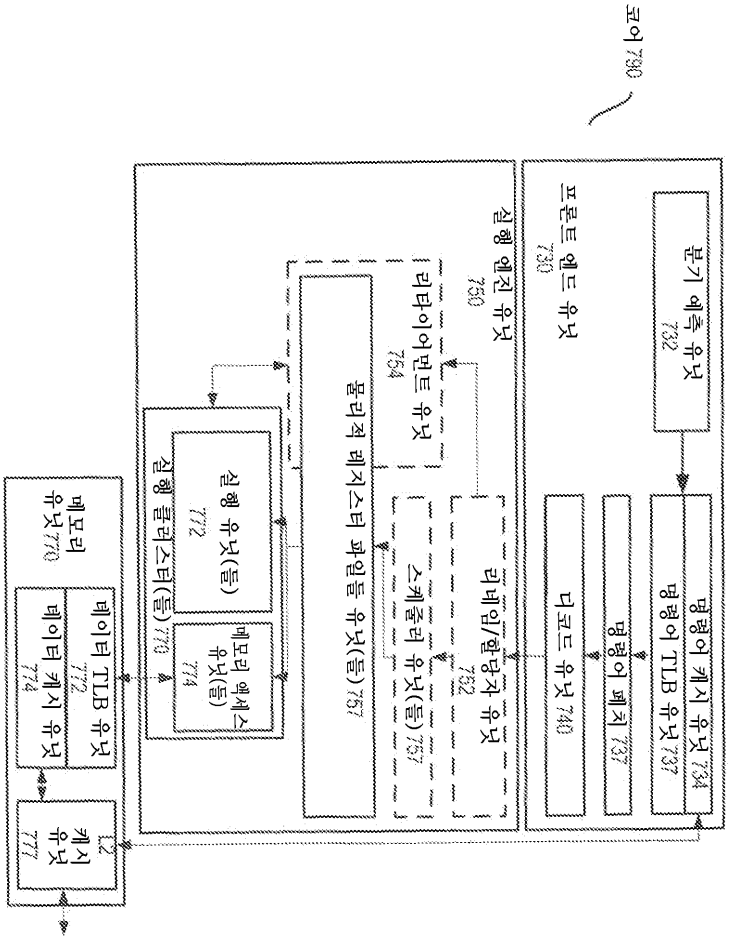
도면6


오버플로우 마스크
680
0
(코어 0)
1
(코어 1)
...
0
(코어 N)
코어 0는 계류중인 모니터 요청들을 갖지 않음
코어 1은 계류중인 모니터 요청(들)을 가짐
코어 N은 계류중인 모니터 요청들을 갖지 않음

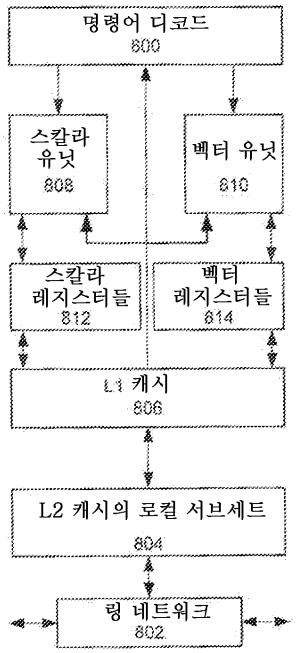
도면7a


파이프라인 700
페치 702
길이 디코딩 704
디코드 707
할당 707
리네이밍 710
스케줄 712
레지스터 판독/ 메모리 판독 714
실행 스테이지 717
라이트 백/ 메모리 기입 717
예외 처리 722
커미트 724

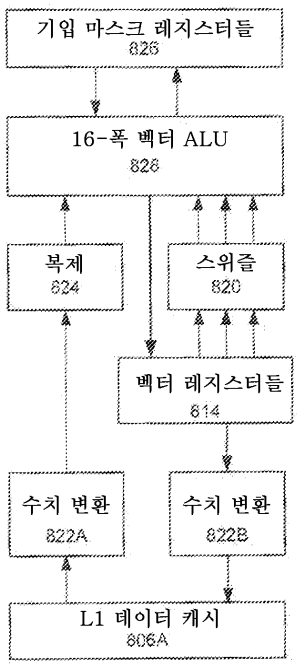
도면7b


코어 790
분기 예측 유닛 732
명령어 캐시 유닛 734
명령어 TLB 유닛 737
프론트 엔드 유닛 730
명령어 페치 737
디코드 유닛 740
실행 엔진 유닛 750
리네임/할당자 유닛 752
스케줄러 유닛(들) 756
리타이어먼트 유닛 754
물리적 레지스터 파일들 유닛(들) 758
실행 클러스터(들) 760
실행 유닛(들) 762
메모리 액세스 유닛(들) 764
메모리 유닛 770
데이터 TLB 유닛 772
데이터 캐시 유닛 774
L2 캐시 유닛 776

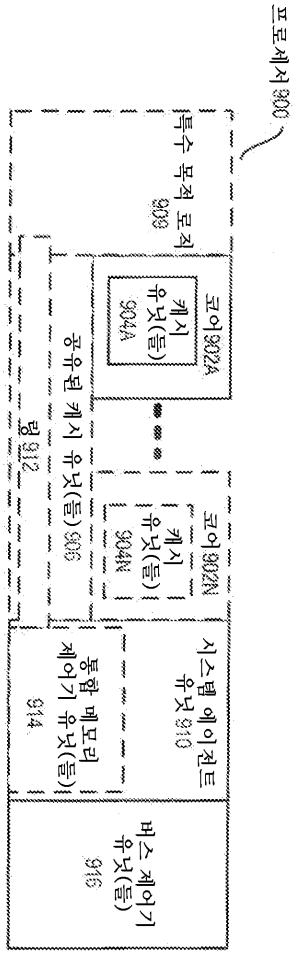
도면8a


명령어 디코드
800
스칼라 유닛
808
벡터 유닛
810
스칼라 레지스터들
812
벡터 레지스터들
814
L1 캐시
806
L2 캐시의 로컬 서브세트
804
링 네트워크
802

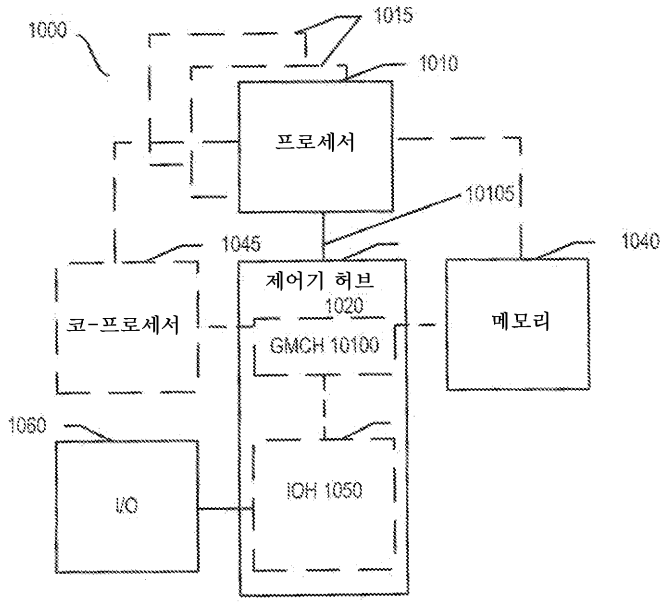
도면8b


기입 마스크 레지스터들
826
16-폭 벡터 ALU
828
복제
824
스위즐
820
벡터 레지스터들
814
수치 변환
822A
수치 변환
822B
L1 데이터 캐시
806A

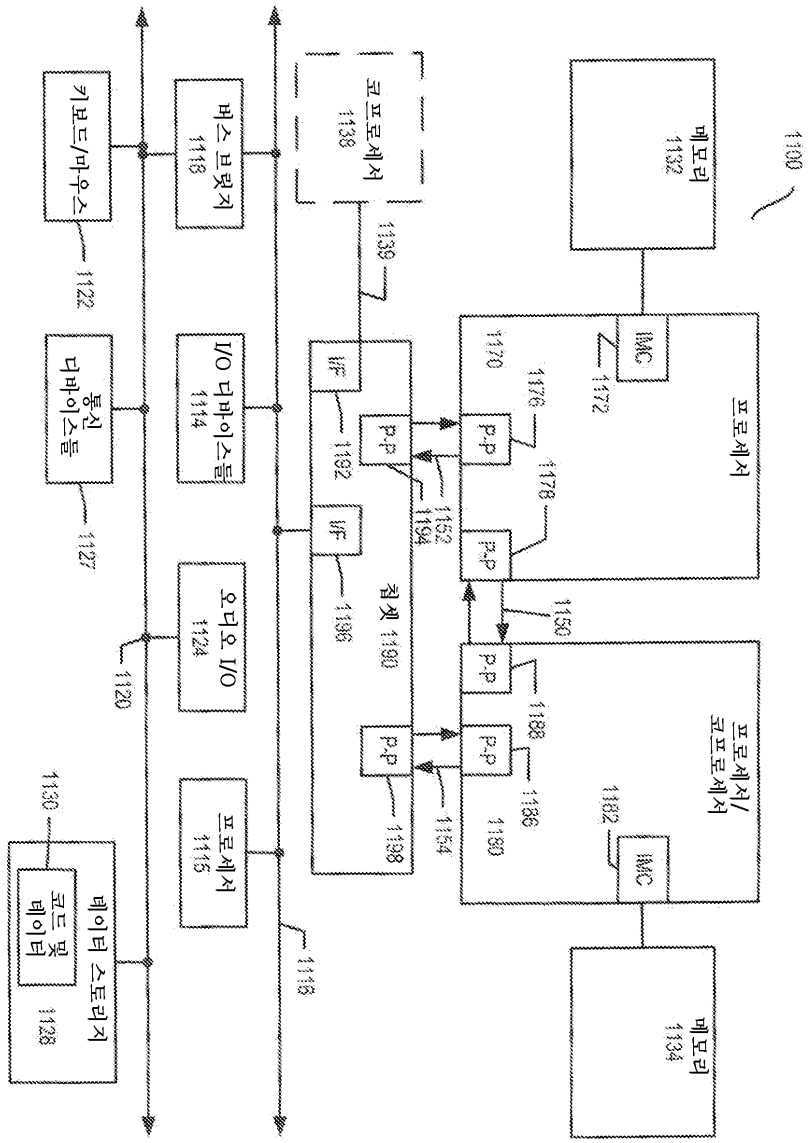
도면9


프로세서 900
특수 목적 로직
909
코어 902A
캐시 유닛(들)
904A
코어 902N
캐시 유닛(들)
904N
공유된 캐시 유닛(들) 906
링 912
시스템 에이전트 유닛 910
통합 메모리 제어기 유닛(들)
914
버스 제어기 유닛(들)
916

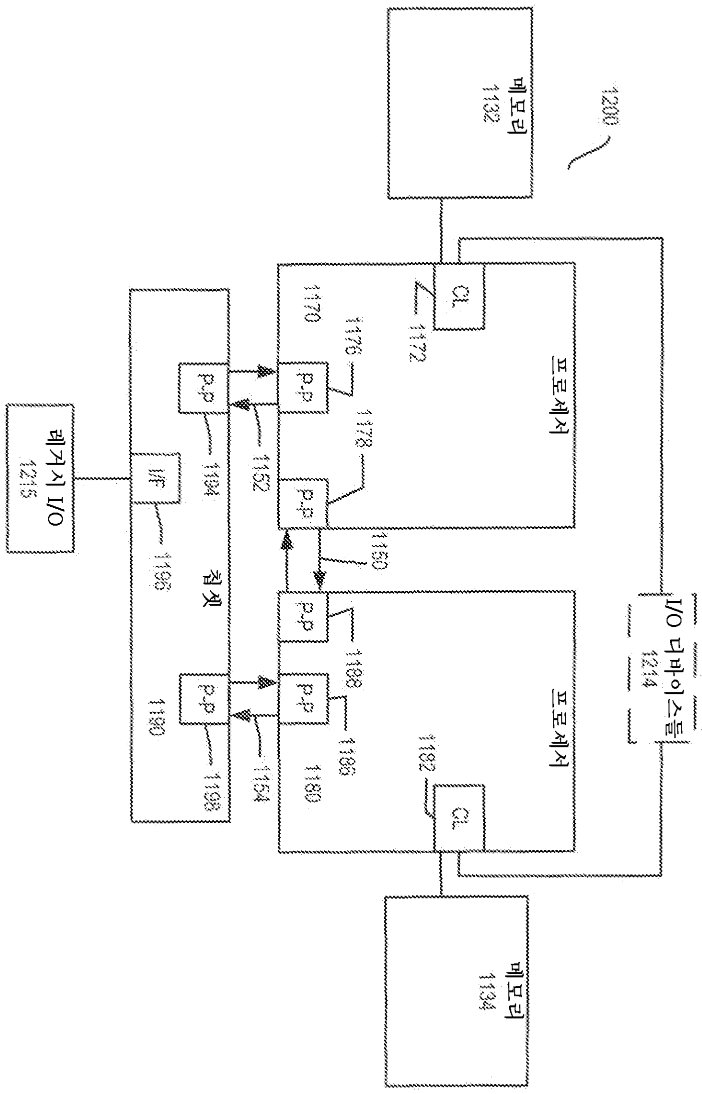
도면10


1000
1015
1010
프로세서
10105
1045
1040
제어기 허브
1020
코-프로세서
GMCH 10100
메모리
1060
I/O
IOH 1050

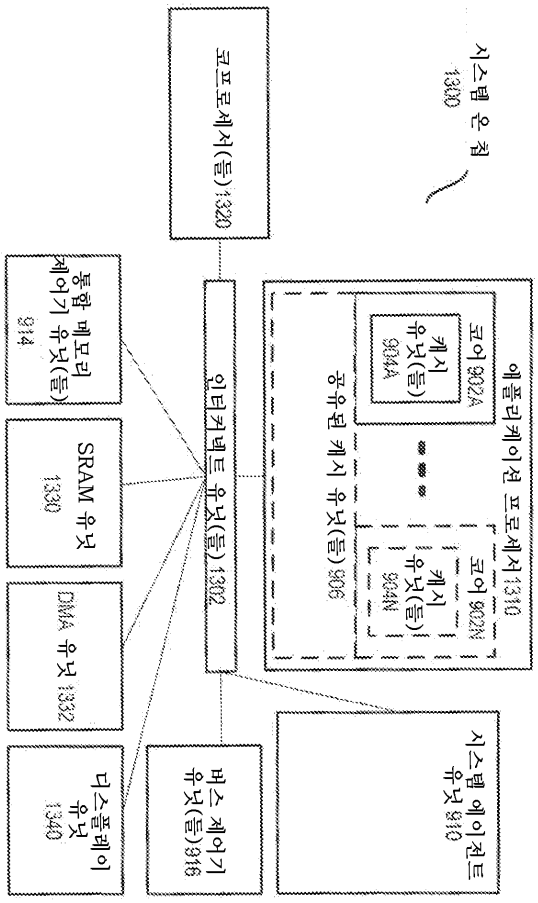
도면11


1100
메모리
1132
IMC
1172
프로세서
1170
P-P
1176
P-P
1178
1150
P-P
1188
P-P
1186
프로세서/
코프로세서
1180
1182
IMC
메모리
1134
1152
1154
칩셋 1190
P-P
1194
P-P
1198
IF
1192
IF
1196
코프로세서
1138
1139
버스 브릿지
1118
I/O 디바이스들
1114
오디오 I/O
1124
프로세서
1115
키보드/마우스
1122
통신
디바이스들
1127
데이터 스토리지
1128
코드 및
데이터
1130
1120
1116

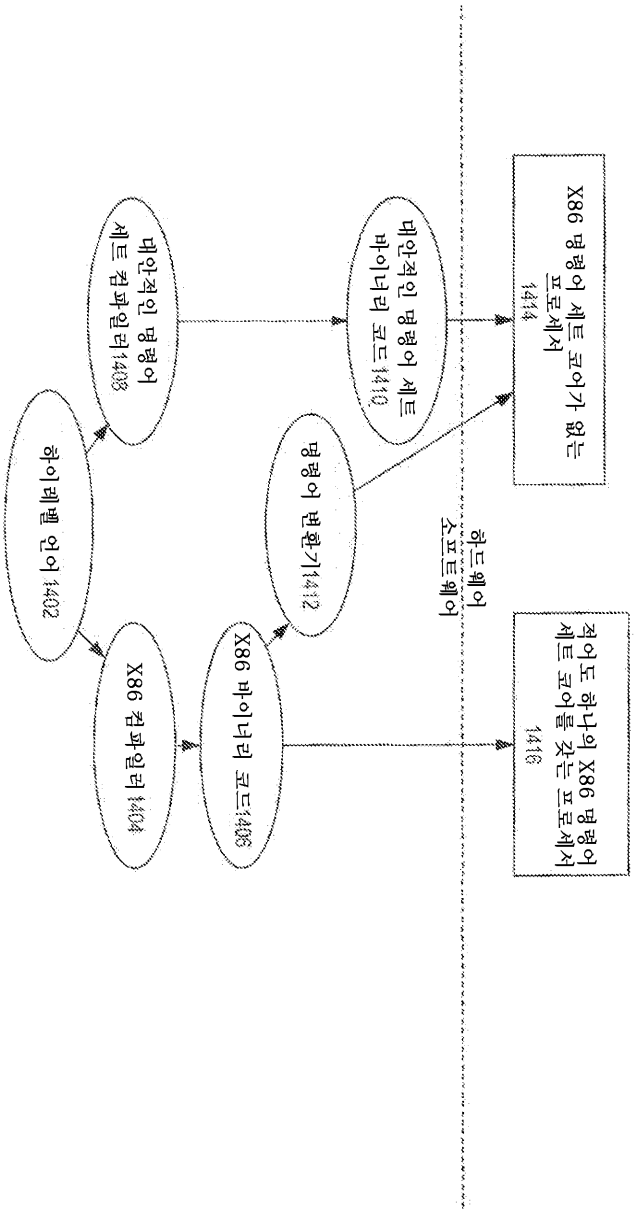
도면12


메모리
1132
1200
프로세서
CL
1172
1170
1176
P-P
1178
P-P
1150
P-P
1188
P-P
1186
1180
CL
1182
프로세서
메모리
1134
I/O 디바이스들
1214
칩셋
P-P
1194
1152
I/F
1196
P-P
1190
1198
1154
레거시 I/O
1215

도면13


시스템 온 칩
1300
코프로세서(들)1320
에플리케이션 프로세서1310
코어902N
캐시 유닛(들) 904N
코어902A
캐시 유닛(들) 904A
공유된 캐시 유닛(들)906
인터커넥트 유닛(들)1302
통합 메모리 제어기 유닛(들) 914
SRAM 유닛 1330
DMA 유닛 1332
디스플레이 유닛 1340
버스 제어기 유닛(들)916
시스템 에이전트 유닛 910

도면14


X86 명령어 세트 코어가 없는 프로세서 1414
대안적인 명령어 세트 바이너리 코드 1410
대안적인 명령어 세트 컴파일러 1403
하이레벨 언어 1402
명령어 변환기 1412
X86 바이너리 코드 1406
X86 컴파일러 1404
하드웨어
소프트웨어
적어도 하나의 X86 명령어 세트 코어를 갖는 프로세서 1416