(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년05월24일
(11) 등록번호 10-1735222
(24) 등록일자 2017년05월04일

(51) 국제특허분류(Int. Cl.)
G06F 12/08 (2016.01)
(52) CPC특허분류
G06F 12/0815 (2013.01)
G06F 12/0811 (2013.01)
(21) 출원번호 10-2015-7004705
(22) 출원일자(국제) 2013년08월05일
심사청구일자 2016년02월29일
(85) 번역문제출일자 2015년02월24일
(65) 공개번호 10-2015-0040946
(43) 공개일자 2015년04월15일
(86) 국제출원번호 PCT/US2013/053626
(87) 국제공개번호 WO 2014/025691
국제공개일자 2014년02월13일
(30) 우선권주장
61/680,201 2012년08월06일 미국(US)
(뒷면에 계속)
(56) 선행기술조사문헌
US20130262775 A1
(뒷면에 계속)
전체 청구항 수 : 총 48 항

(73) 특허권자
퀄컴 인코포레이티드
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
(72) 발명자
리칠릭 보후슬라프
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
쳉 충 렌
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
(뒷면에 계속)
(74) 대리인
특허법인코리아나

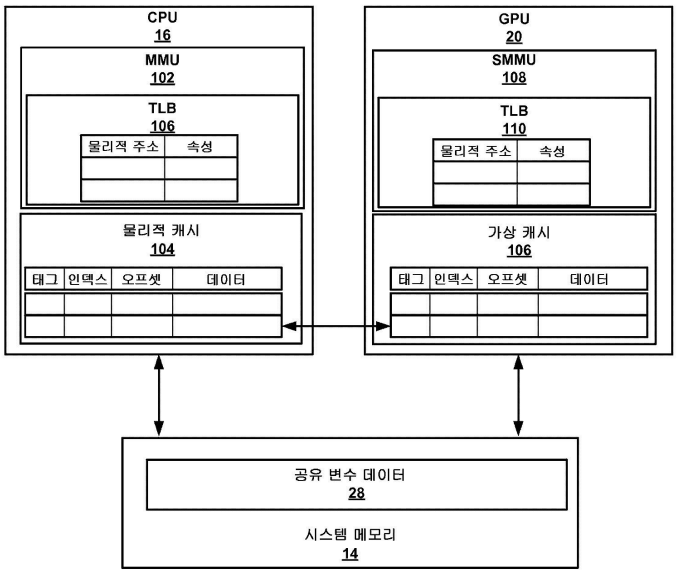
심사관 : 임정묵

(54) 발명의 명칭 **해제 일관성 메모리 순서화 모델을 갖는 멀티-코어 컴퓨트 캐시 코히어런시**

(57) 요 약

제 1 프로그램가능 프로세서로, 제 1 프로세서의 제 1 캐시의 캐시 라인들에 공유 변수 데이터를 저장하는 것을 포함하는 방법이다. 그 방법은,제 1 프로그램가능 프로세서로, 해제와-함께-저장 (store-with-release) 동작을 실행하는 것, 제 2 프로그램가능 프로세서로, 획득과-함께-로드 (load-with-acquire) 동작을 실행하는 것; 및 제 2 프로그램가능 프로세서로, 제 2 프로그램가능 프로세서의 캐시로부터 공유 변수 데이터의 값을 로드하는 것을 더 포함한다.

대 표 도



(52) CPC특허분류
G06F 12/0837 (2013.01)
G06F 12/0891 (2013.01)
G06F 2212/302 (2013.01)

(72) 발명자
그루버 앤드류 에반
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
부르드 알렉세이 브이
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
샤프 콜린 크리스토퍼
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
디머스 에릭
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775

(56) 선행기술조사문헌
US20060026371 A1
US5553266 B1
JP2010044599 A*
WO2011031969 A1*
JP2011128803 A*
*는 심사관에 의하여 인용된 문헌

(30) 우선권주장
61/800,441 2013년03월15일 미국(US)
13/958,399 2013년08월02일 미국(US)

명 세 서

청구범위

청구항 1

프로그램가능 그래픽 프로세서에 의한 실행을 위한 명령들이 상기 프로그램가능 그래픽 프로세서 및 프로그램가능 프로세서에 의해 액세스가능한 공유 변수 데이터에 대한 메모리 동작을 포함하는지 여부를 결정하는 단계;

상기 메모리 동작 후에 제 1 동기화 동작 및 제 2 동기화 동작을 포함하도록 상기 명령들을 변경하는 단계로서, 상기 제 1 및 제 2 동기화 동작들 각각은 메모리 코히어런시 동작을 포함하고, 상기 제 1 동기화 동작은 해제와-함께-저장 동작이고 상기 제 2 동기화 동작은 획득과-함께-로드 동작인, 상기 명령들을 변경하는 단계;

상기 메모리 동작을 실행하는 단계;

상기 제 1 동기화 동작을 실행하는 단계; 및

상기 제 2 동기화 동작을 실행하는 단계를 포함하고,

상기 제 2 동기화 동작을 실행하는 단계는 또한,

상기 프로그램가능 그래픽 프로세서로, 상기 공유 변수 데이터와 연관된 캐시 라인들을 무효화하는 단계;

상기 프로그램가능 그래픽 프로세서로, 상기 공유 변수 데이터와 연관된 상기 캐시 라인들을 채우는(fill) 단계; 및

상기 프로그램가능 그래픽 프로세서로, 상기 제 2 동기화 동작이 실행을 완료하기까지 후속 명령들의 발행을 방지하는 단계를 포함하는, 방법.

청구항 2

제 1 항에 있어서,

상기 제 1 동기화 동작을 실행하는 단계는 또한,

상기 프로그램가능 그래픽 프로세서로, 상기 공유 변수 데이터와 연관된 캐시 라인들을 플러시하는 단계; 및

임의의 이전의 저장들이 완료하기를 기다리는 단계를 포함하는, 방법.

청구항 3

삭제

청구항 4

제 1 항에 있어서,

상기 프로그램가능 그래픽 프로세서로, 상기 프로그램가능 프로세서의 캐시를 스누프하는 단계;

상기 프로그램가능 그래픽 프로세서로, 상기 공유 변수 데이터의 업데이트된 값과 연관된 캐시 히트를 검출하는 단계; 및

상기 프로그램가능 그래픽 프로세서로, 상기 공유 변수 데이터와 연관된 상기 캐시 히트의 검출에 응답하여 상기 프로그램가능 그래픽 프로세서의 캐시에 상기 공유 변수 데이터의 상기 업데이트된 값을 쓰는 단계를 더 포함하는, 방법.

청구항 5

제 1 항에 있어서,

상기 제 1 동기화 동작을 실행하는 단계는 또한,

상기 프로그램가능 그래픽 프로세서의 제 1 캐시보다 높은 레벨을 갖는 상위 레벨 캐시, 시스템 메모리, 상기 프로그램가능 프로세서의 캐시, 및 상기 프로그램가능 그래픽 프로세서의 상기 제 1 캐시 중 하나에 상기 공유 변수 데이터를 쓰는 단계를 포함하는, 방법.

청구항 6

제 1 항에 있어서,

상기 프로그램가능 프로세서는 중앙 프로세싱 유닛 (CPU) 을 포함하는, 방법.

청구항 7

제 1 항에 있어서,

상기 프로그램가능 프로세서의 모든 캐시 라인들 및 상기 프로그램가능 그래픽 프로세서의 모든 캐시 라인들은 공유되며,

상기 제 1 동기화 동작을 실행하는 단계는 또한, 상기 프로그램가능 프로세서의 상기 캐시 라인들의 모두를 플러시하는 단계를 포함하고,

상기 제 2 동기화 동작을 실행하는 단계는 또한, 상기 프로그램가능 그래픽 프로세서의 상기 캐시 라인들의 모두를 무효화하는 단계를 포함하는, 방법.

청구항 8

제 1 항에 있어서,

상기 프로그램가능 프로세서의 모든 캐시 라인들의 서브세트가 공유되고, 상기 프로그램가능 그래픽 프로세서의 모든 캐시 라인들의 서브세트가 공유되며,

상기 제 1 동기화 동작을 실행하는 단계는 또한, 상기 프로그램가능 프로세서의 상기 캐시 라인들의 공유된 서브 세트만을 플러시하는 단계를 포함하고,

상기 제 2 동기화 동작을 실행하는 단계는 또한, 상기 프로그램가능 그래픽 프로세서의 상기 캐시 라인들의 공유된 서브 세트만을 무효화하는 단계를 포함하는, 방법.

청구항 9

제 8 항에 있어서,

상기 프로그램가능 프로세서의 상기 캐시 라인들의 상기 공유된 서브 세트의 적어도 하나의 공유된 라인이 제 1 공유성 (shareability) 속성에 의해 나타내어지고,

상기 프로그램가능 그래픽 프로세서의 상기 캐시 라인들의 상기 공유된 서브 세트의 각각의 공유된 라인이 제 2 공유성 속성에 의해 나타내어지는, 방법.

청구항 10

제 9 항에 있어서,

캐시 필 동작을 수행하는 단계; 및

상기 캐시 필 동작의 수행에 응답하여, 페이지 테이블로부터 상기 제 1 공유성 속성 및 상기 제 2 공유성 속성 중 적어도 하나를 읽는 단계를 더 포함하는, 방법.

청구항 11

제 9 항에 있어서,

컴파일러에 의해, 로드 명령 및 저장 명령 중 적어도 하나를 발행하는 단계를 더 포함하며,

상기 로드 명령 및 상기 저장 명령 중 상기 적어도 하나는 상기 제 1 공유성 속성 및 상기 제 2 공유성 속성 중

적어도 하나를 나타내는, 방법.

청구항 12

제 11 항에 있어서,

상기 로드 명령은 공유된 로드 명령을 포함하고, 상기 저장 명령은 공유된 저장 명령을 포함하는, 방법.

청구항 13

제 9 항에 있어서,

상기 제 1 공유성 속성 및 상기 제 2 공유성 속성 중 적어도 하나를 나타내는 복수의 주소 포인터들의 하나 이상의 비트들을 읽는 단계를 더 포함하는, 방법.

청구항 14

제 9 항에 있어서,

상기 제 1 공유성 속성 및 상기 제 2 공유성 속성 중 적어도 하나는, 상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나가 비공유 상태에 있다는 것을 나타내며,

상기 방법은,

상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나에 대해 캐시 동작을 수행하는 단계; 및

상기 캐시 동작의 수행에 응답하여, 상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나가 공유 상태에 있다는 것을 나타내기 위해 상기 제 1 공유성 속성 및 상기 제 2 공유성 속성 중 상기 적어도 하나를 변경하는 단계를 더 포함하는, 방법.

청구항 15

제 9 항에 있어서,

상기 제 1 공유성 속성 및 상기 제 2 공유성 속성 중 적어도 하나는, 상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나가 공유 상태에 있다는 것을 나타내며,

상기 방법은,

상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나에 대해 캐시 동작을 수행하는 단계; 및

상기 캐시 동작의 수행에 응답하여, 상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나가 비공유 상태에 있다는 것을 나타내기 위해 상기 제 1 공유성 속성 및 상기 제 2 공유성 속성 중 상기 적어도 하나를 변경하는 단계를 더 포함하는, 방법.

청구항 16

제 1 항에 있어서,

상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나의 주소 지역을 결정하기 위해 하나 이상의 레지스터에 저장된 주소와 상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 상기 적어도 하나의 주소를 비교하는 단계; 및

상기 주소 지역이 공유되는지 또는 비공유되는지 여부를 결정하는 단계를 더 포함하는, 방법.

청구항 17

제 1 항에 있어서,

상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하

나의 더티 바이트들을 결정하는 단계;

캐시 퇴거 (eviction) 및 캐시 플러시 중 적어도 하나를 수행하는 단계; 및

상기 캐시 퇴거 및 상기 캐시 플러시 중 상기 적어도 하나 동안 상기 더티 바이트들만을 캐시 또는 시스템 메모리에 쓰는 단계를 더 포함하는, 방법.

청구항 18

제 1 항에 있어서,

상기 제 1 동기화 동작을 실행하는 단계는 또한,

인터럽트, 레지스터 읽기, 레지스터 쓰기, 및 프로그램가능 지연 중 적어도 하나를 수신하는 것에 응답하여 캐시를 플러시하는 단계를 포함하는, 방법.

청구항 19

제 1 항에 있어서,

상기 제 1 동기화 동작을 실행하는 단계는 또한,

캐시 압력, 타임아웃, 및 명시적 소프트웨어 캐시 유지보수 중 적어도 하나에 응답하여 캐시를 플러시하는 단계를 포함하는, 방법.

청구항 20

제 1 항에 있어서,

상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나는 가상적으로 태깅되는, 방법.

청구항 21

제 1 항에 있어서,

상기 제 1 동기화 동작을 실행하는 단계는 또한,

캐시에 저장된 가상 주소를 물리적 주소로 번역하는 단계; 및

상기 물리적 주소에 기초하여 상기 가상 주소를 상기 물리적 주소로 번역한 후 상기 캐시의 캐시 라인들을 플러시하는 단계를 포함하는, 방법.

청구항 22

제 1 항에 있어서,

상기 제 2 동기화 동작을 실행하는 단계는 또한,

캐시에 저장된 가상 주소를 물리적 주소로 번역하는 단계; 및

상기 물리적 주소에 기초하여 상기 가상 주소를 상기 물리적 주소로 번역한 후 상기 캐시의 캐시 라인들을 무효화하는 단계를 포함하는, 방법.

청구항 23

프로그램가능 그래픽 프로세서에 의한 실행을 위한 명령들이 상기 프로그램가능 그래픽 프로세서 및 프로그램가능 프로세서에 의해 액세스가능한 공유 변수 데이터에 대한 메모리 동작을 포함하는지 여부를 결정하는 하드웨어 수단;

상기 메모리 동작 후에 제 1 동기화 동작 및 제 2 동기화 동작을 포함하도록 상기 명령들을 변경하는 하드웨어 수단으로서, 상기 제 1 및 제 2 동기화 동작들 각각은 메모리 코히어런시 동작을 포함하고, 상기 제 1 동기화 동작은 해제와-함께-저장 동작이고 상기 제 2 동기화 동작은 획득과-함께-로드 동작인, 상기 명령들을 변경하는 하드웨어 수단;

상기 메모리 동작을 실행하는 하드웨어 수단;

상기 제 1 동기화 동작을 실행하는 하드웨어 수단; 및

상기 제 2 동기화 동작을 실행하는 하드웨어 수단을 포함하고,

상기 제 2 동기화 동작을 실행하는 수단은 또한,

상기 공유 변수 데이터와 연관된 캐시 라인들을 무효화하는 수단;

상기 공유 변수 데이터와 연관된 상기 캐시 라인들을 채우는 (fill) 수단; 및

상기 제 2 동기화 동작이 실행을 완료하기까지 후속 명령들의 발행을 방지하는 수단을 포함하는, 장치.

청구항 24

제 23 항에 있어서,

상기 제 1 동기화 동작을 실행하는 수단은 또한,

상기 공유 변수 데이터와 연관된 캐시 라인들을 플러시하는 수단; 및

임의의 이전의 저장들이 완료하기를 기다리는 수단을 포함하는, 장치.

청구항 25

삭제

청구항 26

제 23 항에 있어서,

상기 프로그램가능 프로세서의 캐시를 스누프하는 수단;

상기 공유 변수 데이터의 업데이트된 값과 연관된 캐시 히트를 검출하는 수단; 및

상기 공유 변수 데이터와 연관된 상기 캐시 히트의 검출에 응답하여 상기 프로그램가능 그래픽 프로세서의 캐시에 상기 공유 변수 데이터의 상기 업데이트된 값을 쓰는 수단을 더 포함하는, 장치.

청구항 27

장치로서,

프로그램가능 프로세서; 및

프로그램가능 그래픽 프로세서를 포함하며,

상기 장치는,

상기 프로그램가능 그래픽 프로세서에 의한 실행을 위한 명령들이 상기 프로그램가능 그래픽 프로세서 및 상기 프로그램가능 프로세서에 의해 액세스가능한 공유 변수 데이터에 대한 메모리 동작을 포함하는지 여부를 결정하는 것;

상기 메모리 동작 후에 제 1 동기화 동작 및 제 2 동기화 동작을 포함하도록 상기 명령들을 변경하는 것으로서, 상기 제 1 및 제 2 동기화 동작들 각각은 메모리 코히어런시 동작을 포함하고, 상기 제 1 동기화 동작은 해제와-함께-저장 동작이고 상기 제 2 동기화 동작은 획득과-함께-로드 동작인, 상기 명령들을 변경하는 것;

상기 메모리 동작을 실행하는 것;

상기 제 1 동기화 동작을 실행하는 것; 및

상기 제 2 동기화 동작을 실행하는 것을 하도록 구성되고,

상기 제 2 동기화 동작을 실행하기 위해, 상기 장치는,

상기 프로그램가능 그래픽 프로세서로, 상기 공유 변수 데이터와 연관된 캐시 라인들을 무효화하는 것;

상기 프로그램가능 그래픽 프로세서로, 상기 공유 변수 데이터와 연관된 상기 캐시 라인들을 채우는 (fill) 것; 및

상기 프로그램가능 그래픽 프로세서로, 상기 제 2 동기화 동작이 실행을 완료하기까지 후속 명령들의 발행을 방지하는 것을 하도록 추가로 구성되는, 장치.

청구항 28

제 27 항에 있어서,

상기 제 1 동기화 동작을 실행하기 위해, 상기 장치는,

상기 프로그램가능 그래픽 프로세서로, 상기 공유 변수 데이터와 연관된 캐시 라인들을 플러시하는 것; 및

임의의 이전의 저장들이 완료하기를 기다리는 것을 하도록 추가로 구성되는, 장치.

청구항 29

삭제

청구항 30

제 27 항에 있어서,

상기 장치는,

상기 프로그램가능 그래픽 프로세서로, 상기 프로그램가능 프로세서의 캐시를 스누프하는 것;

상기 프로그램가능 그래픽 프로세서로, 상기 공유 변수 데이터의 업데이트된 값과 연관된 캐시 히트를 검출하는 것; 및

상기 프로그램가능 그래픽 프로세서로, 상기 공유 변수 데이터와 연관된 상기 캐시 히트의 검출에 응답하여 상기 프로그램가능 그래픽 프로세서의 캐시에 상기 공유 변수 데이터의 상기 업데이트된 값을 쓰는 것을 하도록 추가로 구성되는, 장치.

청구항 31

제 27 항에 있어서,

상기 제 1 동기화 동작을 실행하기 위해, 상기 장치는,

상기 프로그램가능 그래픽 프로세서의 제 1 캐시보다 높은 레벨을 갖는 상위 레벨 캐시, 시스템 메모리, 상기 프로그램가능 프로세서의 캐시 및 상기 프로그램가능 그래픽 프로세서의 상기 제 1 캐시 중 하나에 상기 공유 변수 데이터를 쓰는 것을 하도록 추가로 구성되는, 장치.

청구항 32

제 27 항에 있어서,

상기 프로그램가능 프로세서는 중앙 프로세싱 유닛 (CPU) 을 포함하는, 장치.

청구항 33

제 27 항에 있어서,

상기 프로그램가능 프로세서의 모든 캐시 라인들 및 상기 프로그램가능 그래픽 프로세서의 모든 캐시 라인들은 공유되며,

상기 제 1 동기화 동작을 실행하기 위해, 상기 장치는, 상기 프로그램가능 프로세서의 상기 캐시 라인들의 모두를 플러시하도록 추가로 구성되고,

상기 제 2 동기화 동작을 실행하기 위해, 상기 장치는, 상기 프로그램가능 그래픽 프로세서의 상기 캐시 라인들의 모두를 무효화하도록 추가로 구성되는, 장치.

청구항 34

제 27 항에 있어서,

상기 프로그램가능 프로세서의 모든 캐시 라인들의 서브세트가 공유되고, 상기 프로그램가능 그래픽 프로세서의 모든 캐시 라인들의 서브세트가 공유되며,

상기 제 1 동기화 동작을 실행하기 위해, 상기 장치는, 상기 프로그램가능 프로세서의 상기 캐시 라인들의 상기 공유된 서브 세트만을 플러시하도록 추가로 구성되고,

상기 제 2 동기화 동작을 실행하기 위해, 상기 장치는, 상기 프로그램가능 그래픽 프로세서의 상기 캐시 라인들의 상기 공유된 서브 세트만을 무효화하도록 추가로 구성되는, 장치.

청구항 35

제 34 항에 있어서,

상기 프로그램가능 프로세서의 상기 캐시 라인들의 상기 공유된 서브 세트의 적어도 하나의 공유된 라인이 제 1 공유성 (shareability) 속성에 의해 나타내어지고,

상기 프로그램가능 그래픽 프로세서의 상기 캐시 라인들의 상기 공유된 서브 세트의 각각의 공유된 라인이 제 2 공유성 속성에 의해 나타내어지는, 장치.

청구항 36

제 35 항에 있어서,

상기 장치는,

캐시 필 동작을 수행하는 것; 및

상기 캐시 필 동작의 수행에 응답하여, 페이지 테이블로부터 상기 제 1 공유성 속성 및 상기 제 2 공유성 속성 중 적어도 하나를 읽는 것을 하도록 추가로 구성되는, 장치.

청구항 37

제 35 항에 있어서,

상기 장치는,

컴파일러에 의해, 로드 명령 및 저장 명령 중 적어도 하나를 발행하는 것을 하도록 추가로 구성되며,

상기 로드 명령 및 상기 저장 명령 중 상기 적어도 하나는 상기 제 1 공유성 속성 및 상기 제 2 공유성 속성 중 적어도 하나를 나타내는, 장치.

청구항 38

제 37 항에 있어서,

상기 로드 명령은 공유된 로드 명령을 포함하고, 상기 저장 명령은 공유된 저장 명령을 포함하는, 장치.

청구항 39

제 35 항에 있어서,

상기 장치는 상기 제 1 공유성 속성 및 상기 제 2 공유성 속성 중 적어도 하나를 나타내는 복수의 주소 포인터들의 하나 이상의 비트들을 읽도록 추가로 구성되는, 장치.

청구항 40

제 35 항에 있어서,

상기 제 1 공유성 속성 및 상기 제 2 공유성 속성 중 적어도 하나는, 상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나가 비공유 상태에 있다는 것을 나타내

며,

상기 장치는,

상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나에 대해 캐시 동작을 수행하는 것; 및

상기 캐시 동작의 수행에 응답하여, 상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나가 공유 상태에 있다는 것을 나타내기 위해 상기 제 1 공유성 속성 및 상기 제 2 공유성 속성 중 상기 적어도 하나를 변경하는 것을 하도록 추가로 구성되는, 장치.

청구항 41

제 35 항에 있어서,

상기 제 1 공유성 속성 및 상기 제 2 공유성 속성 중 적어도 하나는, 상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나가 공유 상태에 있다는 것을 나타내며,

상기 장치는,

상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나에 대해 캐시 동작을 수행하는 것; 및

상기 캐시 동작의 수행에 응답하여, 상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나가 비공유 상태에 있다는 것을 나타내기 위해 상기 제 1 공유성 속성 및 상기 제 2 공유성 속성 중 상기 적어도 하나를 변경하는 것을 하도록 추가로 구성되는, 장치.

청구항 42

제 27 항에 있어서,

상기 장치는,

상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나의 주소 지역을 결정하기 위해 하나 이상의 레지스터에 저장된 주소와 상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 상기 적어도 하나의 주소를 비교하는 것; 및

상기 주소 지역이 공유되는지 또는 비공유되는지 여부를 결정하는 것을 하도록 추가로 구성되는, 장치.

청구항 43

제 27 항에 있어서,

상기 장치는,

상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나의 더티 바이트들을 결정하는 것;

캐시 퇴거 (eviction) 및 캐시 플러시 중 적어도 하나를 수행하는 것; 및

상기 캐시 퇴거 및 상기 캐시 플러시 중 상기 적어도 하나 동안 상기 더티 바이트들만을 캐시 또는 시스템 메모리에 쓰는 것을 하도록 추가로 구성되는, 장치.

청구항 44

제 27 항에 있어서,

상기 제 1 동기화 동작을 실행하기 위해, 상기 장치는,

인터럽트, 레지스터 읽기, 레지스터 쓰기, 및 프로그램가능 지연 중 적어도 하나를 수신하는 것에 응답하여 캐시를 플러시하는 것을 하도록 추가로 구성되는, 장치.

청구항 45

제 27 항에 있어서,

상기 제 1 동기화 동작을 실행하기 위해, 상기 장치는,

캐시 압력, 타임아웃, 및 명시적 소프트웨어 캐시 유지보수 중 적어도 하나에 응답하여 캐시를 플러시하는 것을 하도록 추가로 구성되는, 장치.

청구항 46

제 27 항에 있어서,

상기 프로그램가능 프로세서와 연관된 캐시 및 상기 프로그램가능 그래픽 프로세서와 연관된 캐시 중 적어도 하나는 가상적으로 태깅되는, 장치.

청구항 47

제 27 항에 있어서,

상기 제 1 동기화 동작을 실행하기 위해, 상기 장치는,

캐시에 저장된 가상 주소를 물리적 주소로 변역하는 것; 및

상기 물리적 주소에 기초하여 상기 가상 주소를 상기 물리적 주소로 번역한 후 상기 캐시의 캐시 라인들을 플러시하는 것을 하도록 추가로 구성되는, 장치.

청구항 48

제 27 항에 있어서,

상기 제 2 동기화 동작을 실행하기 위해, 상기 장치는,

캐시에 저장된 가상 주소를 물리적 주소로 번역하는 것; 및

상기 물리적 주소에 기초하여 상기 가상 주소를 상기 물리적 주소로 번역한 후 상기 캐시의 캐시 라인들을 무효화하는 것을 하도록 추가로 구성되는, 장치.

청구항 49

저장된 명령들을 갖는 비일시적 컴퓨터 판독가능 저장 매체로서,

상기 명령들은, 실행되는 경우, 하나 이상의 프로세서들로 하여금,

프로그램가능 그래픽 프로세서에 의한 실행을 위한 명령들이 상기 프로그램가능 그래픽 프로세서 및 프로그램가능 프로세서에 의해 액세스가능한 공유 변수 데이터에 대한 메모리 동작을 포함하는지 여부를 결정하게 하고;

상기 메모리 동작 후에 제 1 동기화 동작 및 제 2 동기화 동작을 포함하도록 상기 명령들을 변경하게 하는 것으로서, 상기 제 1 및 제 2 동기화 동작들 각각은 메모리 코히어런시 동작을 포함하고, 상기 제 1 동기화 동작은 해제와-함께-저장 동작이고 상기 제 2 동기화 동작은 획득과-함께-로드 동작인, 상기 명령들을 변경하게 하며;

상기 메모리 동작을 실행하게 하고;

상기 제 1 동기화 동작을 실행하게 하며; 및

상기 제 2 동기화 동작을 실행하게 하고,

상기 하나 이상의 프로세서들로 하여금, 상기 제 2 동기화 동작을 실행하게 하는 명령들은, 실행되는 경우, 상기 하나 이상의 프로세서들로 하여금,

상기 프로그램가능 그래픽 프로세서로, 상기 공유 변수 데이터와 연관된 캐시 라인들을 무효화하게 하며;

상기 프로그램가능 그래픽 프로세서로, 상기 공유 변수 데이터와 연관된 상기 캐시 라인들을 채우게(fill) 하며; 그리고

상기 프로그램가능 그래픽 프로세서로, 상기 제 2 동기화 동작이 실행을 완료하기까지 후속 명령들의 발행을 방지하게 하는 명령들을 더 포함하는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 50

제 49 항에 있어서,

상기 하나 이상의 프로세서들로 하여금, 상기 제 1 동기화 동작을 실행하게 하는 명령들은, 실행되는 경우, 상기 하나 이상의 프로세서들로 하여금,

상기 프로그램가능 그래픽 프로세서로, 상기 공유 변수 데이터와 연관된 캐시 라인들을 플러시하게 하며; 그리고

임의의 이전의 저장들이 완료하기를 기다리게 하는 명령들을 더 포함하는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 51

삭제

청구항 52

제 49 항에 있어서,

실행되는 경우, 상기 하나 이상의 프로세서들로 하여금,

상기 프로그램가능 그래픽 프로세서로, 상기 프로그램가능 프로세서의 캐시를 스누프하게 하며;

상기 프로그램가능 그래픽 프로세서로, 상기 공유 변수 데이터의 업데이트된 값과 연관된 캐시 히트를 검출하게 하며; 그리고

상기 프로그램가능 그래픽 프로세서로, 상기 공유 변수 데이터와 연관된 상기 캐시 히트의 검출에 응답하여 상기 프로그램가능 그래픽 프로세서의 캐시에 상기 공유 변수 데이터의 상기 업데이트된 값을 쓰게 하는 명령들을 더 포함하는, 비일시적 컴퓨터 판독가능 저장 매체.

발명의 설명

기 술 분 야

[0001] 본 출원은 2012년 8월 6일자로 출원된 미국 가출원 제61/680,201호, 및 2013년 3월 15일자로 출원된 미국 가출원 제61/800,441호를 우선권 주장하며, 그것들의 전체 내용들은 참조로 본원에 통합된다.

[0002] 기술 분야

[0003] 본 개시물은 이종 컴퓨팅에 관계된 기법들에 관한 것이고, 더 상세하게는, 이종 컴퓨팅에서 캐시 코히어런시 (cache coherency) 와 관계된 기법들에 관한 것이다.

배 경 기 술

[0004] 최근에 다수의 프로세서들이 하나의 시스템으로 상호접속되는 이른바 이종 컴퓨팅 쪽으로 이동하고 있다. 어떤 경우들에서, 태스크들이 프로세서들 간에 공유될 수도 있다. 작업들 중 일부는 특히 어떤 유형의 프로세서, 이를테면 중앙 프로세싱 유닛 (CPU), 그래픽 프로세싱 유닛 (GPU), 또는 디지털 신호 프로세서 (DSP) 에 더 적합할 수도 있다. 이종 태스크들의 수행에 관계된 각각의 프로세서가 하나 이상의 캐시들을 가질 수도 있다. 각각의 캐시는 다수의 프로세서들 간에 공유되는 변수 데이터의 복사본을 포함할 수도 있다. 캐시들은 프로세서의 하나 이상의 실행 유닛들에 대해 캐싱된 데이터를 더욱 빠르게 액세스할 수 있게 함으로써 프로세서들이 프로그램 실행을 더 빠르게 하는 것을 허용할 수도 있다. 하나의 프로세서가 시스템의 프로세서들의 둘 이상의 캐시들 간에 공유되는 공유 변수 데이터에 액세스하는 경우, "캐시 코히어런시 시스템"이라고 지칭되는 메커니즘이, 프로세서 캐시들에 저장된 공유 변수 데이터의 복사본들이 시스템의 프로세서들의 모두 간에 일관성이 있는 것과, 공유 변수 데이터에 대한 변경들이 이들 변수들을 시기적절하고 일관된 방식으로 공유하는 모든 프로세서들에게 관측가능하게 되는 것을 보장한다.

발명의 내용

과제의 해결 수단

[0005] 대체로, 본 개시물에서 설명된 기법들은, 멀티-코어, 예컨대, 어쩌면 상이한 유형들의 다수의 프로세서 코어들을 포함하는 이종, 컴퓨팅 시스템들에서, 캐시 코히어런시, 즉, 다수의 캐시들 및 시스템 메모리 간의 데이터 일관성을 유지하는 기법들을 위한 것이다. 본 개시물의 기법들 중 하나에 따르면, 중앙 프로세싱 유닛 (CPU), 그래픽 프로세싱 유닛 (GPU), 디지털 신호 프로세서 (DSP), 또는 다른 유형의 프로세싱 코어가 멀티-코어 컴퓨팅 시스템의 프로세서들의 모두 사이에서 공유된 변수에 쓰거나 또는 그 변수를 획득하는 경우 명령들을 실행할 수도 있다. 프로세서 코어 ("프로세서"라고 지칭됨) 가 공유 변수 데이터에 액세스를 시도하는 경우, 프로세서 코어는, 공유 변수 데이터에 적용된 임의의 동작들이 순서화된 코히어런트 방식으로 일어나는 것을 보장하기 위하여, 동기 동작들, 이를테면 획득과-함께-로드 (load-with-acquire) 및/또는 해제와-함께-저장 (store-with-release) 동작들을 실행할 수도 있다.

[0006] 멀티-코어 시스템에서의 프로세서들의 일부 또는 전부가 캐시를 포함할 수도 있다. 캐시는 빈번하게 사용되는 데이터의 로컬 작업 세트를 포함할 수도 있다. 일부 경우들에서, 캐시는 멀티-코어 시스템의 하나 이상의 다른 프로세서들 간에 공유되는 공유 변수 데이터의 로컬 복사본을 포함할 수도 있다. 본 개시물의 기법들에 따라, 다른 프로세서들의 각각이 공유 변수 데이터의 로컬 복사본을 포함하는 캐시를 마찬가지로 포함할 수도 있다.

[0007] 다수의 프로세서들, 예컨대, 제 1 프로세서 및 제 2 프로세서를 갖는 멀티-코어 시스템에서, 제 1 프로세서는 일부 공유 변수 데이터의 값을 변경할 수도 있다. 제 1 프로세서는 그 다음에 공유 변수 데이터의 값을 플러시 (flush) 할 수도 있다. 공유 변수 데이터를 플러시하는 것은 제 1 프로세서로 하여금, 업데이트된 공유 변수 데이터 값을, 예컨대, 시스템 메모리에 쓰도록 할 수도 있다. 다른 프로세서들이 공유 변수 데이터의 업데이트된 값을 보는 것을 보장하기 위해, 캐시 플러시에 응답하여, 제 1 또는 제 2 프로세서는 제 2 프로세서의 캐시에서의 공유 변수 데이터에 대응하는 캐시 엔트리를 무효화할 수도 있다. 제 2 프로세서가 공유 변수 데이터의 값을 이전에 포함했던 캐시 라인을 읽는 경우, 캐시 라인은 무효 (invalid) 로서 마킹된다. 제 2 프로세서의 캐시 라인이 무효로 마킹되기 때문에, 제 2 프로세서는 제 2 프로세서의 캐시로부터 공유 변수 데이터의 기한이 지난 값을 읽는 대신, 시스템 메모리로부터 공유 변수 데이터의 최신 값을 취출하고, 공유 변수 데이터에 대응하는 제 2 프로세서의 캐시를 업데이트한다. 이런 방식으로, 본 개시물의 기법들은 공유 변수 데이터가 캐시 무효화 및 플러시들의 사용을 통해 코히어런트가 되는 것을 보장한다.

[0008] 하나의 예에서, 방법이, 제 1 프로그램가능 프로세서로, 제 1 프로세서의 제 1 캐시의 캐시 라인들에 공유 변수 데이터를 저장하는 것, 제 1 프로그램가능 프로세서로, 해제와-함께-저장 (store-with-release) 동작을 실행하는 것, 제 2 프로그램가능 프로세서로, 획득과-함께-로드 (load-with-acquire) 동작을 실행하는 것, 및 제 2 프로그램가능 프로세서로, 제 2 프로그램가능 프로세서의 캐시로부터 공유 변수 데이터의 값을 로드하는 것을 포함한다.

[0009] 다른 예에서, 본 개시물은, 제 1 캐시를 갖는 제 1 프로그램가능 프로세서, 제 2 캐시를 갖는 제 2 프로그램가능 프로세서를 포함하는 디바이스를 설명하는데, 그 디바이스는, 제 1 프로그램가능 프로세서로, 제 1 프로세서의 제 1 캐시의 캐시 라인들에 공유 변수 데이터를 저장하며, 제 1 프로그램가능 프로세서로, 해제와-함께-저장 동작을 실행하며, 제 2 프로그램가능 프로세서로, 획득과-함께-로드 동작을 실행하고, 제 2 프로그램가능 프로세서로, 상기 제 2 프로그램가능 프로세서의 캐시로부터 상기 공유 변수 데이터의 값을 로드하도록 구성된다.

[0010] 다른 예에서, 본 개시물은, 제 1 프로그램가능 프로세서로, 제 1 프로세서의 제 1 캐시의 캐시 라인들에 공유 변수 데이터를 저장하는 수단, 제 1 프로그램가능 프로세서로, 해제와-함께-저장 동작을 실행하는 수단, 제 2 프로그램가능 프로세서로, 획득과-함께-로드 동작을 실행하는 수단, 및 제 2 프로그램가능 프로세서로, 제 2 프로그램가능 프로세서의 캐시로부터 공유 변수 데이터의 값을 로드하는 수단을 포함하는 디바이스를 설명한다.

[0011] 다른 예에서, 본 개시물은, 실행 시, 프로그램가능 프로세서로 하여금, 제 1 프로그램가능 프로세서로, 제 1 프로세서의 제 1 캐시의 캐시 라인들에 공유 변수 데이터를 저장하게 하며, 제 1 프로그램가능 프로세서로, 해제와-함께-저장 동작을 실행하게 하며, 제 2 프로그램가능 프로세서로, 획득과-함께-로드 동작을 실행하게 하고, 제 2 프로그램가능 프로세서로, 제 2 프로그램가능 프로세서의 캐시로부터 공유 변수 데이터의 값을 로드하게 하는 명령들을 저장하고 있는 비일시적 컴퓨터 판독가능 저장 매체를 설명한다.

[0012] 하나 이상의 예들의 세부사항들은 첨부 도면들 및 다음의 설명에서 언급된다. 다른 특징들, 목적들, 및 이점들은 상세한 설명 및 도면들로부터, 그리고 청구항들로부터 명확하게 될 것이다.

도면의 간단한 설명

[0013] 도 1은 본 개시물에서 설명되는 하나 이상의 예들에 따른 멀티-코어 컴퓨팅 시스템의 일 예를 도시하는 블록도이다.

도 2는 그래픽 프로세싱 유닛 (GPU) 이 온-디맨드 캐시 코히어런스를 지원하는 것을 허용하는 명령들을 실행할 수 있는 그래픽 프로세싱 파이프라인을 구현할 수도 있는 GPU의 다른 예를 도시하는 블록도이다.

도 3a는 캐시 무효화 및 플러시 명령들을 포함할 수도 있는 GPU 커널에서의 동기 동작들을 수행하도록 구성된 GPU를 예시하는 개념도이다.

도 3b는 동기 동작들을 위한 명령들을 GPU 커널에 삽입하는 컴파일러/드라이버를 예시하는 개념도이다.

도 4a는 오래된 (stale) 공유된 변수를 포함하는 캐시를 예시하는 개념도이다.

도 4b는 무효로서 마킹된 캐시 라인으로부터 읽는 것으로 인해 메인 메모리로부터 캐시를 업데이트하는 것을 예시하는 개념도이다.

도 5a는 오래된 공유된 변수를 포함하는 메인 메모리를 예시하는 개념도이다.

도 5b는 캐시 플러시를 수행한 후의 시스템 메모리의 상태를 예시하는 개념도이다.

도 6은 본 개시물의 기법들에 따른 GPU 및 CPU의 캐시 및 메모리 관리 유닛들을 예시하는 개념도이다.

도 7은 본 개시물의 기법들에 따른 속성 비트들을 구현하는 캐시 라인을 예시하는 개념도이다.

도 8은 본 개시물의 기법들에 따른 GPU 캐시를 예시하는 개념도이다.

도 9는 본 개시물에서 설명되는 하나 이상의 예들에 따른 멀티-코어 시스템에서 캐시 코히어런시를 유지하는 프로세스를 예시하는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0014] 최근에 다수의 프로세서들 또는 프로세서 코어들 ("코어들") 이 하나의 시스템으로 상호접속되는 이른바 이종 컴퓨팅 쪽으로 이동하고 있다. 그러나, 이종 컴퓨팅 시스템들에 연관된 많은 도전과제들이 있다. 특정한 코어들이 특정 태스크들에 적합할 수도 있다. 일 예로서, CPU들은 상당한 양들의 조건부 로직 (예컨대, 분기 (branch) 들 및 점프들) 으로 프로그램들을 실행하는 것에 더 적합할 수도 있다. 다른 프로세서들, 이를테면 그래픽 프로세싱 유닛 (Graphics Processing Unit; GPU) 들은 대규모 병렬 동작들, 이를테면 벡터 프로세싱, 및 부동 소수점 연산들을 수행하는 것에 더 적합할 수도 있다. 또 다른 프로세서들, 이를테면 디지털 신호 프로세서들 (DSP들) 이 특정 수학적 연산들을 수행하는 것을 포함할 수도 있는 특정 애플리케이션들, 이를테면 디지털 신호 프로세싱에 적합할 수도 있다. 이러한 동작들은 고속 푸리에 변환들 (FFT들), 및 이산 코사인 변환들 (DCT들) 을 포함할 수도 있다.

[0015] 점점 더, I/O 프로세싱 코어들, 이를테면 GPU들의 능력들은, 더 많은 CPU들의 기능성과 유사한 기능성을 포함하도록 증가되었다. 많은 GPU들은 조건부 로직, 이를테면 분기들 및 점프들을 포함하는 더 많은 범용 애플리케이션들을 실행하는 능력을 포함한다. 덧붙여, 범용 컴퓨팅 언어들, 이를테면 Khronos Group에 의한 OpenCL 또는 Microsoft에 의한 DirectCompute는, 애플리케이션 프로그래밍 인터페이스들 (API들) 을 제공하는데, 이 API들은 프로그램이 이러한 범용 컴퓨팅 언어들과 호환하는 임의의 프로세싱 코어 상에서 실행할 수 있도록 프로그래머가 단일 프로그램을 하나의 언어로 작성하는 것을 허용할 수도 있다.

[0016] 프로그래밍 언어들에서의 진보들과 GPU들 및 DSP들과 같은 프로세서들의 프로세싱 능력들에서의 증가들에도 불구하고, 이종 멀티-코어 컴퓨팅은 여전히 다수의 도전과제들을 제기한다. 하나의 이러한 도전과제는 이종 컴퓨팅 시스템의 다수의 코어들 간에 공유되는 데이터, 이를테면 공유 변수 데이터가, 프로세서들 간에 코히어런트를 유지하는 것을 보장하는 것이다. 본 개시물의 기법들은 공유된 변수에 대한 변경들이 시스템의 프로세서들의 모두 간에 적시에 및 순서화된 방식으로 전파되는 것을 보장하는 캐시 코히어런시 시스템을 위한 것이다.

[0017] 도 1은 본 개시물에서 설명되는 하나 이상의 예들에 따른 멀티-코어 컴퓨팅 시스템의 일 예를 도시하는 블록도이다. 도 1은 CPU (16), GPU (20), 및 DSP (24) 를 포함하는 이종 멀티-코어 시스템 (2) 을 예시한다. 비록 CPU (16), GPU (20), 및 DSP가 비분할 (indivisible) 유닛들인 것으로 예시되었지만, CPU (16), GPU

(20), 및 DSP (24) 의 각각은 다수의 프로세서들 및/또는 다수의 프로세서 코어들을 나타낼 수도 있다. 일 예로서, 멀티-코어 시스템 (2) 이 다수의 CPU들, GPU들, DSP들, 및/또는 다른 유형들의 프로세서들을 가질 수도 있다. CPU (16) 는 캐시 (18) 를 포함하며, GPU (20) 는 캐시 (22) 를 포함하고, DSP (24) 는 캐시 (26) 를 포함한다 (총칭하여 "캐시들 (18, 22, 및 26)"). 캐시들 (18, 22, 및 26) 은 시스템 메모리 (14) 로부터 명령들 및/또는 데이터를 취출하는 것에 비하여 명령들 및/또는 데이터의 빠른 취출을 위해 빈번하게 사용되는 명령들 및/또는 데이터를 각각 저장할 수도 있다. 비록 캐시들 (18, 22, 및 26) 이 명령들 및/또는 데이터를 포함할 수도 있지만, 본 개시물의 기법들은 공유된 명령들이 아니라, 공유된 데이터의 캐시 코히어런시 쪽을 겨냥하고 있다. 따라서, 본 개시물은 캐시들 (18, 22, 및 26) 로부터의 공유된 변수들과 같은 데이터의 취출 및 저장만을 언급한다. 캐시들 (18, 22, 및 26) 의 각각은 하나 이상의 "레벨들"의 캐시, 이를테면 레벨 1 ("L1") 캐시, 및 레벨 2 ("L2") 캐시를 포함할 수도 있다. L1 캐시가 L2 캐시에 비하여 많은 데이터를 유지하지 못할 수도 있지만, 프로세서가 L2 캐시보다는 더 빠르게 (즉, 더 낮은 액세스 레이턴시를 가지고) L1 에 쓰고 그 L1으로부터 읽을 수도 있다.

[0018] CPU (16), GPU (20), 및 DSP (24) 는 시스템 메모리 (14) 에, 예를 들어, 시스템 버스를 통해 접속된다. 시스템 메모리 (14) 는 일부 공유 변수 데이터 (28) 를 포함할 수도 있는 동적 랜덤 액세스 메모리 (DRAM) 를 포함할 수도 있다. 공유 변수 데이터 (28) 는, 예를 들어, 공유된 명령 워드들, 변수들, 버퍼들, 데이터 구조들, 작업 (job) 들 등을 포함할 수도 있다. 비록 CPU (16), GPU (20), 및 DSP (24) 가 비분할 유닛들인 것으로서 예시되었지만, 시스템 메모리 (14) 는 하나 이상의 물리적 집적 회로들 (또한 "칩들"이라고 지칭됨) 을 포함할 수도 있다. 시스템 메모리 (14) 는, 시스템 메모리 (14) 로부터 분리되는 그리고 각각의 CPU (16), GPU (20), 및 DSP (24) 간에 공유되는 (즉, 그것들에 의해 액세스가능한) 다른 레벨의 캐시, 이를테면 레벨 3 ("L3") 캐시를 또한 포함할 수도 있다.

[0019] 본 개시물의 기법들은, 이종 코어들 및/또는 프로세서들 (예컨대, CPU (16), GPU (20), 및 DSP (24)) 로 구성될 수도 있는 멀티-코어 시스템 (2) 이 멀티-코어 시스템 (2) 의 코어들 전체에 걸쳐 캐시 코히어런시를 유지하기 위하여 동기 동작들에 응답하여 캐시 관리를 수행하는 것을 가능하게 한다. 동기 동작들은 이른바 명시적 전체 및 부분적 배리어들, 펜스들, 뿐만 아니라 전체 및 부분적 배리어들과 로드들 및 저장 (store) 들의 결합된 형태들을 포함할 수도 있다. 일부 예들에서, 동기 동작들은 해제와-함께-저장 및 획득과-함께-로드 동작들을 포함할 수도 있다.

[0020] 본 개시물의 기법들은, 다양한 복사본들이 캐시 (18), 캐시 (22), 및 캐시 (26) 중 하나 이상에 저장될 수도 있는 공유 변수 데이터 (28) 에 대한 변경들이, 멀티-코어 시스템 (2) 의 다른 프로세서들 (즉, CPU (16), GPU (20), 및 DSP (24)) 의 캐시들로 전파되는 것을 가능하게 한다. CPU (16), GPU (20), 및 DSP (24) 가 캐시 코히어런스 모드를 구현하는 것을 가능하게 하는 것은 "온-디맨드 캐시 코히어런스 모델"이라고 지칭된다. 온-디맨드 캐시-코히어런스 모델에서, 공유 변수 데이터 (28) 를 읽거나 또는 쓰는 프로세서, 예컨대, GPU (20) 는, 공유 변수 데이터 (28) 에 연관된 해제와-함께-저장 동작 및/또는 획득과-함께-로드 동작의 실행과 같은 동기 동작을 수행할 수도 있다. 멀티-코어 시스템 (2) 의 다른 프로세서들, CPU (16), 및 DSP (24) 는, 공유 변수 데이터 (28) 를 읽는 것 또는 쓰는 것에 연관되는 경우 해제와-함께-저장 동작 및/또는 획득과-함께-로드 동작들을 또한 실행할 수도 있다.

[0021] 해제와-함께-저장 및 획득과-함께-로드 동작들은 메모리 순서화 모델을 정의하는데 사용될 수도 있는 원자적 동작들 (또한 "동기 동작들"이라고 지칭됨) 이다. 원자적 동작들은 컴퓨팅 시스템의 나머지 (예컨대, 코어들, 프로세서들, 메모리 등) 에게 마치 그것들이 즉각적으로 일어난 것처럼 보이는 동작들이다. 획득과-함께-로드 및 해제와-함께-저장 동기 동작들을 실행하는 경우, 프로세서가 획득/해제 메모리 모델 시맨틱스를 표현하기 위해 부분적 배리어들과 로드들 및 저장들을 결합한다. C++11 표준은 획득/해제 메모리 모델의 하나의 구현예를 정의하는 프로그래밍 언어의 하나의 예이다.

[0022] 획득과-함께-로드 및 해제와-함께-저장 동작들에 연관된 시맨틱스 및 제약들은 제 1 스레드에서의 공유된 변수, 이를테면 공유 변수 데이터 (28) 로부터의 읽기가, 제 2 스레드에서의 동일한 공유된 변수로의 쓰기가 완료된 후에 발생하는 것을 보장하기 위해, 즉, 공유된 변수의 읽기 및 쓰기 사이의 의존성 또는 순서화를 강제하기 위해 사용될 수도 있다. 특히, 획득/해제 모드는, 획득과-함께-로드 동작을 사용한 제 1 스레드에서의 공유된 변수에 대한 읽기가 제 2 스레드에서의 동일한 공유된 변수에 대한 해제와-함께-저장 동작을 실행한 후에 발생하도록 읽기/쓰기 의존성을 강제한다.

[0023] 본 개시물의 온-디맨드 캐시 코히어런시 기법들은, 다수의 프로세서들이 획득/해제 메모리 순서화 모델을 구현

하는 동기 동작들, 예컨대, 획득과-함께-로드 및 해제와-함께-저장 동작들을 실행하는 것을 허용하기 위해, 예컨대 이종 멀티-코어 시스템 (2) 에서 이용될 수도 있는 캐시 코히어런시 기법들을 설명한다. 본 개시물의 온-디맨드 코히어런시 모델은, 제 1 프로세서 코어, 예컨대, GPU (20) 가, 공유 변수 데이터 (28) 를 읽고 동기 동작을 수행할 때마다, GPU (20) 또는 다른 프로세서가 공유 변수 데이터 (28) 의 값을 포함하는 제 2 코어의 캐시의 적어도 일부 ("캐시 라인"이라 지칭됨), 예컨대, CPU (16) 의 캐시 (18) 의 하나 이상의 라인들을 무효화할 수도 있도록 기능을 한다. GPU (20) 는 동기 동작을 실행하는 것의 일부로서 캐시 (18) 의 캐시 라인의 무효화를 자동으로 호출할 수도 있다.

[0024] 대안으로, 프로그램 코드가 획득과-함께-로드 및 해제와-함께-저장 동작들을 포함하는 경우, 컴파일러, 이를테면 컴파일러 (30) 는, 캐시 (22) 의 무효화를 자동으로 호출할 수도 있다. 컴파일러 (30) 는 드라이버 및/또는 컴파일러를 포함할 수도 있고, CPU (16), GPU (20), 및 DSP (24) 중 하나 이상을 위한 코드를 생성, 컴파일, 및/또는 수정할 수도 있다. 일부 예들에서, 컴파일러 (30) 는 CPU (16), GPU (20), 및 DSP (24) 중 하나 이상을 실행하는 운영 체제의 애플리케이션 또는 프로세스일 수도 있다. 컴파일러 (30) 는 동기 동작들을 포함하는 코드에서의 로케이션들에서 컴파일 시간에 또는 런 타임에 로우 (low) -레벨 캐시 무효화 명령을 프로그램 코드에 삽입함으로써 캐시 무효화를 자동으로 호출할 수도 있다. 제 2 코어는 그 다음에 캐시 무효화 명령들을 포함하는 수정된 코드를 실행한다. 일단 제 1 코어가 공유 변수 데이터 (28) 를 포함하는 캐시 (18) 의 캐시 라인들을 무효화하면, CPU (16) 는 시스템으로부터 (예컨대, 시스템 메모리 (14), 하이 레벨 캐시, 또는 다른 프로세서의 캐시로부터) 공유된 데이터의 새로운 복사본을 획득하며, 이에 의해 CPU (16) 가 읽는 공유 변수 데이터 (28) 의 값이 "오래된 (stale)" 또는 기한이 지난 (outdated) 것이 아닌 것을 보장한다.

[0025] 프로세서, 이를테면 CPU (16) 가, 획득과-함께-로드 동작을 실행하는 경우, CPU (16) 는, 공유 변수 데이터 (28) 를 포함하는, 예컨대, 캐시 (18) 의 임의의 캐시 라인들을 무효화할 수도 있다. CPU (16) 가 무효 캐시 라인들로부터 읽는 것을 시도하는 경우, 캐시 미스 (miss) 가 발생하고, 그러면 CPU (16) 는 무효화된 캐시 라인들의 새로운 값들을 캐시 (18) 에 입력하고 공유 변수 데이터 (28) 에 대해 임의의 더티 (dirty) 값들을 병합한다. 이런 방식으로, 획득과-함께-로드 동작은, 획득과-함께-로드 동작을 실행하는 일부로서 로드된 임의의 값들이 메모리 계층구조, 예컨대 시스템 메모리 (14), 캐시 (18), 캐시 (22), 및 캐시 (26) 에서 현재 가장 최신 값들이라는 것을 보장한다.

[0026] 위에서 설명된 바와 같이, 획득과-함께-저장 (store-with-acquire) 동작이 제 2 프로세서가 공유된 변수의 로드를 실행하기 전에 제 1 프로세서가 공유된 변수 값의 저장의 실행을 완료하는 것을 보장하는 획득과-함께-로드 동작과 쌍을 이룰 수도 있다. 로드가 저장 뒤에 실행하는 것을 보장하기 위하여, CPU (16) 또는 다른 프로세서는 해제와-함께-저장 동작에 후속하는 명령, 이를테면 GPU (20) 가 실행하는 획득과-함께-로드 명령에 동기화 비트를 첨부할 수도 있다. 동기화 비트를 후속하는 명령에 첨부하는 것은 CPU (16) 가 배리어 (barrier) 를 실행하는 것을 효과적으로 유발하는데, 이 후속하는 명령은 CPU (16) 가 실행을 시작했던 모든 이전의 저장들이 완료되기까지 발행되지 않는다.

[0027] 프로세서가 온-디맨드 캐시 코히어런스를 구현하는 프로세서와 연동하기 위해 온-디맨드 캐시 코히어런스를 구현할 필요는 없다. 오히려, I/O 코히어런스, 즉, 캐시 코히어런스의 일부 형태를 지원하는 임의의 프로세서가, 획득과-함께-로드 및 해제와-함께-저장 명령들을 사용하여 구현될 수도 있는 온-디맨드 캐시 코히어런스를 구현하는 프로세서들과 상호작용할 수도 있다. 일 예로서, 시스템 (2) 의 일부 프로세서들은 온-디맨드 캐시 코히어런스를 지원할 수도 있는 반면, 다른 것들은 더욱 전통적인 캐시 코히어런시 프로토콜들, 이를테면 MESI (Modified, Exclusive, Shared, Invalid), 및 MOESI (Modified, Owned, Exclusive, Shared, Invalid) 프로토콜들을 지원할 수도 있다. 일부 예들에서, 임의의 캐시들이 없는 프로세서들은 온-디맨드 캐시 코히어런트 프로세서들과 연동할 수도 있다.

[0028] 일 예로서, 프로세서, 이를테면 CPU (16) 는, 본 개시물의 기법들에 따른 온-디맨드 캐시 코히어런시를 구현하지 않을 수도 있지만, I/O 코히어런스 또는 캐시 코히어런스를 구현할 수도 있다. CPU (16) 는 캐싱되지 않을 수도, 즉, 캐시를 포함하지 않을 수도 있거나, 또는 다른 캐시 코히어런스 프로토콜, 이를테면 MESI, 또는 MOESI를 구현할 수도 있다. GPU (20) 및/또는 DSP (24) 는 온-디맨드 캐시 코히어런스를 구현할 수도 있고, CPU (16) 가 일부 형태의 I/O 코히어런스를 지원하는 한 CPU (16) 와 연동할 수도 있다.

[0029] 본 개시물의 기법들에 따라, 제 1 프로세서와 제 2 프로세서가, 획득/해제 메모리 순서화 모델을 구현하기 위하여, 온-디맨드 캐시 코히어런시라고 지칭되는 다양한 캐시 관리 기법들을 수행할 수도 있다. 본 개시물의 기법들에 따른 일 예로서, 제 1 프로그램가능 프로세서, 예컨대 CPU (16) 는, 공유 변수 데이터를 캐시 (18) 의

캐시 라인들에 저장할 수도 있다. CPU (16) 는 그 다음에 해제와-함께-저장 동작을 실행할 수도 있다. 제 2 프로세서, 예컨대 GPU (20) 는, 획득과-함께-로드 동작을 실행할 수도 있다. 제 2 프로세서는 그 다음에 GPU (20) 의 캐시, 예컨대 캐시 (22) 로부터 공유 변수 데이터의 값을 로드할 수도 있다.

[0030] 다양한 다른 예들에서, 해제와-함께-저장 동작은 캐시 플러시를 유발하지 않을 수도 있다. 오히려, 컴파일러 또는 드라이버가 캐시 플러시 명령을 프로그램의 실행에 삽입할 수도 있는데, 이는 CPU (16) 가 공유 변수 데이터 (28) 의 업데이트된 값에 대응하는 캐시 라인을 플러시하는 것을 유발한다. CPU (16) 는 캐시 압력, 타임아웃, 및 명시적 소프트웨어 캐시 유지보수 중 적어도 하나에 응답하여 캐시 (18) 를 또한 플러시할 수도 있다. 다른 예에서, CPU (16) 는 인터럽트, 레지스터 읽기, 레지스터 쓰기, 및 프로그램가능 지연 중 적어도 하나를 수신하는 것에 응답하여 해제와-함께-저장 동작을 실행할 수도 있다.

[0031] 일부 예들에서, 해제와-함께-저장 동작은 다른 프로세서의 캐시 라인들 속으로 스누프 (snoop) 하고 공유 변수 데이터 (28) 를 포함하는 캐시 라인들을 업데이트함으로써 CPU (16) 또는 다른 프로세서가 공유 변수 데이터 (28) 의 기한이 지난 값들을 포함하는, 다른 프로세서의 하나 이상의 캐시 라인들, 예컨대, 캐시 (22) 또는 캐시 (26) 를 무효화하는 것을 추가로 유발할 수도 있다. 컴파일러 또는 드라이버가 또한, CPU (16) 또는 GPU (20) 가 공유 변수 데이터 (28) 의 기한이 지난 값들에 대응하는 캐시 (22) 의 라인들을 무효화하는 것을 유발하는 캐시 무효화 명령을 삽입할 수도 있다.

[0032] 일부 예들에서, 멀티-코어 시스템 (2) 에서의 일부 프로세서들이 다른 프로세서들에 가시적인 캐시들을 가지지 않을 수도 있다. 그럼에도 불구하고, 이들 프로세서들은 가시적 캐시들을 가지는 프로세서들로부터 공유된 변수의 최근 값을 읽는 능력을 포함할 수도 있다. 또한, 가시적 캐시들을 가지지 않는 이들 프로세서들은 쓰기 동작을 통해 새로운 값을 공유된 변수에 제공할 수도 있다. 본 개시물의 기법들에 따라, 비-가시적 캐시를 이용한 프로세서의 쓰기들은 이미 관측된 공유 변수 데이터 (28) 에 대한 시스템에서의 모든 쓰기들 뒤에 스케줄링될 수도 있다. 가시적 캐시들을 가지지 않는 프로세서들이 위에서 언급된 방식으로 읽는 것과 쓰는 것을 가능하게 하는 메커니즘은 보통 IO 캐시 코히어런시라고 지칭된다.

[0033] 대안적 예에서, 제 1 프로세서는, 해제와-함께-저장에 연관된 쓰기 동작으로부터의 값을 일시적으로 버퍼링 또는 캐싱하고 올바른 시맨틱스를 여전히 유지하면서도 나중에 그 값을 시스템 메모리 (예컨대, 상위 레벨의 캐시, 다른 프로세서의 캐시, 또는 시스템 메모리) 에 쓰는 것을 또한 선택할 수 있다.

[0034] 본 개시물의 기법들에 따라, CPU (16) 는 공유 변수 데이터 (28) 를 캐시 (18) 의 캐시 라인들에 저장하도록 구성될 수도 있다. CPU (16) 는 해제와-함께-저장 동작을 실행하도록 추가로 구성될 수도 있다. 제 2 프로그램가능 프로세서, 이를테면 GPU (20) 는 획득과-함께-로드 동작을 실행하도록 구성될 수도 있다. GPU (20) 는 GPU (20) 의 캐시 (22) 로부터 공유 변수 데이터 (28) 의 값을 추가로 로드할 수도 있다.

[0035] 동기 동작, 예컨대, 해제와-함께-저장 동작에 응답하여, CPU (16) 는 공유 변수 데이터 (28) 에 연관된 캐시 (18) 의 캐시 라인들을 플러시할 수도 있다. 제 2 프로그램가능 프로세서, 예컨대, 제 2 캐시 (22) 를 갖는 GPU (20) 는 그 다음에 획득과-함께-로드 동작을 실행할 수도 있는데, 이는 획득과-함께-로드 동작이 실행을 완료한 후에 임의의 로드 명령들이 발생하는 것을 보장한다. 획득과-함께-로드 동작은 공유 변수 데이터 (28) 에 대응하는 캐시 (22) 의 캐시 라인들을 무효화하고 다른 캐시 또는 시스템 메모리 (14) 로부터 캐시 필 (cache fill) 을 유발할 수도 있다. 획득과-함께-로드 동작의 실행에 후속하는 임의의 로드들이 GPU (20) 의 캐시 (22) 로부터 공유 변수 데이터 (28) 의 최신 값을 읽을 것이다.

[0036] 도 2는 본 개시물에서 설명되는 하나 이상의 예들에 따른, 그래픽 프로세싱 유닛 (GPU) 이 온-디맨드 캐시 코히어런스를 지원하는 것을 허용하는 명령들을 실행할 수 있는 그래픽 프로세싱 파이프라인을 구현할 수도 있는 GPU의 다른 예를 도시하는 블록도이다. 도 2는 GPU (20), 시스템 메모리 (14), 및 CPU (16) 를 포함하는 멀티-코어 시스템 (2) 을 예시하는데, CPU는 추가로 캐시 (18) 를 포함한다. 멀티-코어 시스템 (2) 을 포함할 수도 있는 디바이스들의 예들은, 모바일 무선 전화기들, 비디오 디스플레이들을 구비한 비디오 게이밍 콘솔들, 모바일 비디오 회의 유닛들, 랩톱 컴퓨터들, 데스크톱 컴퓨터들, 태블릿 컴퓨터들, 텔레비전 셋톱 박스들 등을 포함하지만 그것들로 제한되지는 않는다.

[0037] CPU (16) 는 다양한 유형들의 애플리케이션들을 실행할 수도 있다. 애플리케이션들의 예들은, 웹 브라우저들, 이메일 애플리케이션들, 스프레드시트들, 비디오 게임들, 또는 디스플레이를 위한 볼 수 있는 오브젝트들, 뿐만 아니라 범용 컴퓨팅 작업들을 생성하는 다른 애플리케이션들을 포함한다. 하나 이상의 애플리케이션들의 실행을 위한 명령들이 시스템 메모리 (14) 내에 저장될 수도 있다. CPU (16) 는 프로세싱 작업들, 뿐만

아니라 생성된 볼 수 있는 오브젝트들의 그래픽 데이터를 추가의 프로세싱을 위해 GPU (20) 로 송신할 수도 있다.

[0038] 예를 들어, GPU (20) 는 대규모 병렬 프로세싱을 허용하는 전문화된 하드웨어일 수도 있는데, 이는 그래픽 데이터를 프로세상하기 위해, 뿐만 아니라 병렬화가능 태스크들, 이를테면 다양한 시뮬레이션들 및 수학적으로 강력한 작업들에 대해 잘 기능을 한다. 이런 식으로, CPU (16) 는 GPU (20) 에 의해 더 양호하게 핸들링되는 프로세싱을 분담 (offload) 시킬 수도 있다. CPU (16) 는 특정 API에 따라 GPU (20) 와 통신할 수도 있다. 이러한 API들의 예들은 Microsoft®에 의한 DirectX® API와 Khronos group에 의한 OpenGL®을 포함하지만, 본 개시물의 양태들은 DirectX 및 OpenGL API들로 제한되지 않고, 개발되어 있거나, 현재 개발하고 있거나, 또는 미래에 개발될 다른 유형들의 API들로 확장될 수도 있다.

[0039] GPU (20) 가 CPU (16) 로부터 그래픽 데이터를 수신하는 방식을 정의하는 것에 더하여, API들은 GPU (20) 가 구현하는 특정 그래픽 프로세싱 파이프라인을 정의할 수도 있다. 파이프라인 셋업의 특정한 양태들이 CPU (16) 상에서 실행하는 GPU 드라이버 및 셰이더 코드 컴파일러, 이를테면 컴파일러 (30) 에 의해 핸들링될 수도 있다. GPU 드라이버는 CPU (16) 상에서 실행하는 애플리케이션과는 별개일 수도 있다. 드라이버는 운영 체제, 이를테면 Windows® 또는 Linux 사이의 상호작용을 핸들링할 수도 있는데, 이는 통상적으로 CPU (16), 및 GPU (20) 상에서 실행된다. 셰이더 코드 컴파일러는 언어, 이를테면 OpenCL 또는 DirectCompute로 작성된 코드를 GPU (20) 가 해석하고 실행할 수도 있는 어셈블리 명령들로 컴파일할 수도 있다. GPU (20) 는, 도 1 에서, Direct3D 11 API에 의해 정의된 그래픽 프로세싱 파이프라인을 예시한다. 비록 GPU (20) 의 파이프라인이 DirectX 파이프라인에 관해 설명되지만, GPU (20) 는 OpenGL 4.x 파이프라인 또는 다른 그래픽스 파이프라인에 따라 또한 구현될 수도 있다.

[0040] CPU (16) 및 GPU (20) 의 예들은, 디지털 신호 프로세서 (DSP), 범용 마이크로프로세서, 주문형 집적회로 (ASIC), 필드 프로그램가능 로직 어레이 (FPGA), 또는 다른 동등한 집적 또는 개별 로직 회로를 포함하지만 그것들로 제한되지는 않는다. 일부 예들에서, GPU (20) 는 그래픽 프로세싱에 적합한 대규모 병렬 프로세싱 능력들을 GPU (20) 에 제공하는 통합된 및/또는 개별 로직 회로를 포함하는 전문화된 하드웨어일 수도 있다. 어떤 경우들에서, GPU (20) 는 범용 프로세싱을 또한 포함할 수도 있고, 범용 GPU (GPGPU) 라고 지칭될 수도 있다. 본 개시물에서 설명된 기법들은 GPU (20) 가 GPGPU인 예들에 또한 적용가능할 수도 있다.

[0041] 시스템 메모리 (14) 는 하나 이상의 컴퓨터 판독가능 저장 매체를 포함할 수도 있다. 시스템 메모리 (14) 의 예들은, 랜덤 액세스 메모리 (RAM), 판독 전용 메모리 (ROM), 전기 소거가능 프로그램가능 판독 전용 메모리 (EEPROM), 플래시 메모리, 또는 소망의 프로그램 코드를 명령들 및/또는 데이터 구조들의 형태로 저장하는데 사용될 수 있고 컴퓨터 또는 프로세서에 의해 액세스될 수 있는 임의의 다른 매체를 포함하지만 그것들로 제한되지는 않는다. 위에서 설명된 바와 같이, 시스템 메모리 (14) 는 하나 이상의 레벨들의 캐시를 또한 포함할 수도 있다.

[0042] 일부 양태들에서, 시스템 메모리 (14) 는 CPU (16) 및/또는 GPU (20) 가 본 개시물에서 CPU (16) 및 GPU (20) 에 기인한 기능들을 수행하는 것을 유발하는 명령들, 이를테면 공유 변수 데이터 (28) 에 연관된 동기 동작들을 수행하기 위한 명령들을 포함할 수도 있다. 일부 예들에서, 그 명령들은 획득과-함께-로드 및 해제와-함께-저장 동작들을 포함할 수도 있는데, 그것들은 캐시, 예컨대, 캐시 (18) 또는 캐시 (22) 가 캐시 무효화 및/또는 캐시 플러싱을 수행하는 것을 유발할 수도 있다. 따라서, 시스템 메모리 (14) 는 하나 이상의 프로세서들, 예컨대, CPU (16) 또는 GPU (20) 가, 다양한 기능들을 수행하는 것을 유발하는 명령들을 포함하는 컴퓨터 판독가능 저장 매체일 수도 있다.

[0043] 시스템 메모리 (14) 는, 일부 예들에서, 비일시적 저장 매체로서 간주될 수도 있다. 용어 "비일시적 (non-transitory)"은 저장 매체가 반송파 또는 전파되는 신호로 실시되지 않음을 나타낼 수도 있다. 그러나, 용어 "비일시적"은 시스템 메모리 (14) 가 이동가능하지 않은 것을 의미하도록 해석되지 않아야 한다. 하나의 예로서, 시스템 메모리 (14) 는 멀티-코어 시스템 (2) 의 디바이스로부터 제거되고, 다른 디바이스로 이동될 수도 있다. 다른 예로서, 시스템 메모리 (14) 에 실질적으로 유사한 시스템 메모리가, 멀티-코어 시스템 (2) 의 디바이스 속에 삽입될 수도 있다. 특정한 예들에서, 비일시적 저장 매체는 (예컨대, RAM 내에) 시간 경과에 따라 변화할 수 있는 데이터를 저장할 수도 있다.

[0044] CPU (16) 상의 애플리케이션들의 실행은 CPU (16) 가 GPU (20) 상의 실행을 위해 작업들, 프로세스들 또는 스레드들을 생성하는 것을 유발한다. 작업은 GPU (20) 의 하나 이상의 코어들 또는 실행 유닛들 상에서 실행할 수도 있다. 덧붙여서, GPU (20) 는 볼 수 있는 내용을 함께 형성하는 복수의 프리미티브 (primitive) 들을

또한 렌더링할 수도 있다. 프리미티브들의 예들은 점들, 선들, 삼각형들, 사각형들, 또는 임의의 다른 유형의 다각형을 포함한다. CPU (16) 는 이들 프리미티브들을 그것들의 개별 정점들에 의해 정의할 수도 있다.

예를 들어, CPU (16) 는 그 정점들에 대한 좌표들 및 컬러 값들을 정의할 수도 있다. 좌표 값들은 3차원 (3D) 좌표들 또는 2D 좌표들일 수도 있다.

[0045] GPU (20) 는 그래픽 출력을 렌더링하는 것보다는 GPGPU 프로그램 또는 "커널"을 실행하기 위하여 GPU (20) 의 기능성 유닛들, 예컨대, 부동 소수점 유닛들, 셰이더들, 캐시들 등을 이용할 수도 있다. 도 2에 도시된 그래픽 파이프라인은 3D 그래픽 프로세싱 파이프라인이다. 그러나, 본 개시물의 기법들은 GPGPU 파이프라인, 또는 GPU (20) 상에서 실행하는 애플리케이션을 포함하는 임의의 유형의 파이프라인 상에서의 사용에 적용가능하다. 그래픽 프로세싱 파이프라인은 GPU (20) 상에서 실행하는 소프트웨어 또는 펌웨어에 의해 정의된 바와 같은 기능들을 수행하는 것과 특정 기능들을 수행하기 위해 하드와이어드된 고정 기능 유닛들에 의한 기능들을 수행하는 것을 포함한다. GPU (20) 상에서 실행하는 소프트웨어 또는 펌웨어는 셰이더들이라고 지칭될 수도 있고, 셰이더들은 GPU (20) 의 하나 이상의 셰이더 코어들 상에서 실행할 수도 있다. 셰이더들은 기능적 유연성을 사용자들에게 제공하는데, 사용자가 소망의 태스크들을 임의의 생각할 수 있는 방식으로 수행하도록 셰이더들을 설계할 수 있기 때문이다. 고정-기능 유닛들은, 그러나, 고정-기능 유닛들이 태스크들을 수행하는 방식을 위해 하드와이어드된다. 따라서, 고정-기능 유닛들은 많은 기능적 유연성을 제공하지 못할 수도 있다.

[0046] 정점 셰이더 (38), 도메인 셰이더 (44), 및 기하구조 셰이더 스테이지 (46), 뿐만 아니라 도 2에 그림이 없는 다른 셰이더 유닛들을 포함할 수도 있는 셰이더들은, 프로그램가능 유닛들일 수도 있는 반면, GPU (20) 의 다른 스테이지들 또는 유닛들 (도 2에서 그림이 없음) 은 프로그램가능하지 않을 수도 있다. 본 개시물의 기법들에 따라 온-디맨드 캐시 코히어런스를 지원하기 위해, GPU (20) 의 셰이더들은 획득과-함께-로드 및 해제와-함께-저장 동작들을 포함할 수도 있는 부가적인 동작들을 지원하도록 구성될 수도 있다. 일부 예들에서, 획득과-함께-로드 동작들 및 해제와-함께-저장 동작들은 GPU (20) 가 캐시 무효화들 및/또는 플러시들을 수행하는 것을 유발할 수도 있다. GPU (20) 는 캐시 (22) 의 거동을 제어할 수도 있는 별개의 명시적인 캐시 관리 명령들을 또한 지원할 수도 있다. 캐시 관리 명령들은 캐시 플러시 명령과 캐시 무효화 명령을 포함할 수도 있다. 캐시 플러시 명령 및/또는 해제와-함께-저장 동작은 GPU (20) 가 캐시 (22) 에 저장된, 예컨대, 공유 변수 데이터 (28) 의 값들을 시스템 메모리 (14) 에, 하이 레벨 캐시에, 또는 CPU (16) 의 캐시 (18) 에 쓰는 것을 유발할 수도 있다. 캐시 무효화 명령 및/또는 획득과-함께-로드 동작은 GPU (20) 가 시스템 메모리 (14) 로부터 데이터 값들을 읽는 것과 그 값들을 캐시 (22) 에 저장하는 것을 유발할 수도 있다.

[0047] 일부 예들에서, 캐시 관리 명령들과, 획득과-함께-로드 및 해제와-함께-저장 동작들은, GPU (20) 가 전력 및/또는 면적 관점에서 값이 비쌀 수도 있는 전용 캐시 코히어런시 하드웨어를 필요로 하는 일 없이 캐시 코히어런시를 유지하는 것을 허용할 수도 있다. 드라이버, 컴파일러 (예컨대, 컴파일러 (30)), 또는 다른 애플리케이션이 셰이더 프로그램 ("커널"이라고 지칭됨) 의 코드를 수정할 수도 있다. 컴파일러 또는 드라이버는, 획득과-함께-로드 동작 또는 해제와-함께-저장 동작이 본 개시물의 기법들에 따라 멀티-코어 시스템 (2) 의 다른 프로세서들과의 캐시 코히어런시를 유지하기 위해 발생하는 포인트들에 코드를 자동으로 삽입할 수도 있다. 컴파일러 또는 드라이버에 의한 코드의 삽입은 도 3a 및 도 3b에서 더 상세히 도시된다.

[0048] 커널이 공유 변수 데이터 (28) 에 액세스할 것임을 컴파일러가 결정하면, 커널 컴파일러는 동기 동작들, 예컨대, 획득과-함께-로드 또는 해제와-함께-저장 동작들을 삽입할 수도 있다. 획득과-함께-로드 동작은 GPU (20) 가 공유된 캐시 라인들 (예컨대, 공유 변수 데이터 (28) 에 연관된 캐시 라인들) 을 무효화하며, 캐시 필을 수행하는 것을 유발할 수도 있고, 읽기들이 획득과-함께-로드 동작의 완료 전에 실행하도록 재순서화되지 않을 수 있는 것을 보장한다. 해제와-함께-저장 동작은 GPU (20) 가 공유 변수 데이터 (28) 를 포함하는 임의의 캐시 라인들을 플러시하는 것을 유발할 수도 있고, 현재 스레드에서의 쓰기들이 해제와-함께-저장 동작 후에 실행하기 위해 재순서화될 수 없다는 것을 보장한다. 하나의 프로세서에 의해 실행되는 해제와-함께-저장 동작에 획득과-함께-로드 동작이 뒤따르는 경우, 공유 변수 데이터 (28) 에 대한 변경들은 순서화되고 일관된 방식으로 발생한다. 획득과-함께-로드 및 해제와-함께-저장 동작들은 일부 예들에서 커널/셰이더 프로그래밍 언어의 일부일 수도 있다. 그러나, 일부 예들에서, 동기 동작, 예컨대, 획득과-함께-로드 또는 해제와-함께-저장 동작을 수행하기 위한 필요성이 컴파일 시간에 알려지지 않을 수도 있는데, 이들 동기 동작들의 명령들에 대한 필요성이, 임의의 공유된 데이터 또는 변수들 (또한 "공유된 표면들"이라고 지칭됨) 을 사용하여 커널이 실행될 지의 여부에 의존할 수도 있기 때문이다.

[0049] GPU (20) 가 공유된 변수를 읽고 있는 경우의 캐시 코히어런시를 보장하기 위해, 컴파일러 또는 드라이버는 공

유 변수 데이터 (28) 로부터 읽는 및/또는 그 데이터에 쓰는 경우, 셰이더 코드를 분석할 수도 있고 동기 동작들, 예컨대, 획득과-함께-로드 및 해제와-함께-저장 동작들을 위한 명령들을 셰이더 코드에 삽입할 수도 있다.

동기 동작들은, GPU (20) 가 시스템 메모리 (14) 로부터 공유 변수 데이터 (28) 의 값을 읽는 것과, 공유 변수 데이터 (28) 의 값을 읽기 전에 캐시 (22) 에 공유 변수 데이터 (28) 의 값을 저장하는 것을 유발하는 캐시 무효화 명령들을 포함할 수도 있다. 캐시 코히어런시를 보장하기 위해 GPU (20) 가 캐시 (22) 에 값을 되 (back) 쓰는 경우, 컴파일러 또는 드라이버는 셰이더 코드를 분석하고 공유 변수 데이터 (28) 에 대한 변경들이 순서화된 방식으로 전파되는 것을 보장하는 동기 동작, 예컨대, 획득과-함께-로드 또는 해제와-함께-저장 동작들을 삽입한다.

[0050] 일부 예들에서, 해제와-함께-저장 동작과 같은 동기 동작은, GPU (20) 가 공유 변수 데이터 (28) 에 대응하는 캐시 (18) 의 하나 이상의 캐시 라인들을 무효화하는 것과 공유 변수 데이터 (28) 의 새로운 값을 페치하는 것을 유발할 수도 있고, 공유 변수 데이터 (28) 의 새로운 값을 캐시 (22) 에 저장할 수도 있다. 일부 예들에서, GPU (20) 는 공유 변수 데이터 (28) 의 가장 최신 값을 획득하기 위해 다른 프로세서의 캐시, 예컨대, CPU (16) 의 캐시 (18) 를 "스누프" 또는 읽을 수도 있다. 다른 예들에서, GPU (20) 는 시스템 메모리 (14) 또는 상위 레벨 캐시로부터 공유 변수 데이터 (28) 의 값을 읽을 수도 있다.

[0051] 컴파일러 또는 드라이버는 동기 동작들, 예컨대, 해제와-함께-저장 및 획득과-함께-로드 동작들을 커널 속으로 다수의 상이한 방법들로 삽입할 수도 있다. 하나의 예에서, 컴파일러는 공유 변수 데이터 (28) 에 대한 로드 및 저장 명령들에 입각하여 해제와-함께-저장 및 획득과-함께-로드 동작들을 삽입할 수도 있다. 다른 예에서, 컴파일러는 공유 변수 데이터 (28) 에 대한 읽기들 및 쓰기들 (로드들 및 저장들) 에 입각하여 NOP (No Operation) 명령들을 커널 속으로 삽입할 수도 있다. 커널의 바인드 시간에, 드라이버는 커널이 공유 변수 데이터 (28) (예컨대, 공유된 표면들) 와 쌍을 이루는지의 여부를 결정할 수도 있다. 커널이 공유 변수 데이터 (28) 와 쌍을 이루는지의 여부에 기초하여, 드라이버는 본 개시물의 기법들에 따라 커널 바이너리를 수정하고 NOP 명령들을 해제와-함께-저장 및 획득과-함께-로드 동작들로 교체할 수도 있다. 그리고 또 다른 예에서, GPU (20) 는, 커널을 런칭하기 전에 드라이버가 설정할 수도 있는 GPU (20) 의 레지스터에 저장된 일정한 값에 기초하여 실행 시간에 조건부로, 해제와-함께-저장 및 획득과-함께-로드 동작들을 커널 속으로 컴파일할 수도 있다.

[0052] 캐시 (22) 의 공유된 변수들을 플러시 또는 무효화하기 위해, 드라이버는 커널 코드에서의 공유된 변수들의 리스트를 바인드 시간에 그리고 커널을 런칭하기 전에 생성할 수도 있다. 커널을 실행하는 동안, GPU (20) 는, 동기 동작이 특정 변수에 대해 실행되는 경우에는 언제든지, 공유된 변수들 및 플러시들의 리스트를 "걷거나 (walk)" 또는 트래버스 (traverse) 하고 리스트에서의 각각의 공유된 변수에 연관된 캐시 (22) 의 영역을 플러시 또는 무효화하는 루틴을 실행할 수도 있다. 공유된 변수들의 리스트가 비어 있으면, GPU (20) 는 해제와-함께-저장 및 획득과-함께-로드 동작들을 실행하지 않을 수도 있다.

[0053] 위에서 나타낸 바와 같이, 도 2에 예시된 그래픽 프로세싱 파이프라인은 실질적으로는 Direct3D 11에 의해 정의된 바와 같은 그래픽 프로세싱 파이프라인이다. 이 예에서, GPU (20) 는 하나 이상의 정점 (vertex) 셰이더 스테이지 (38), 헐 (hull) 셰이더 스테이지 (40), 도메인 셰이더 스테이지 (44), 및 기하구조 (geometry) 셰이더 스테이지 (46) 를 포함할 수도 있다. GPU (20) 는 예시된 것들보다 많은 스테이지들을 포함할 수도 있고, 일부 예들에서, GPU (20) 는 예시된 스테이지들의 모두를 반드시 포함하지 않을 수도 있다. 또한, 스테이지의 특정 순서화는 예시의 목적으로 제공되고 제한하는 것으로 간주되지 않아야 한다.

[0054] 정점 셰이더 스테이지 (38) 는 정점들을 프로세싱할 수도 있고 변환들, 스키닝 (skinning), 모핑 (morphing), 및 정점 당 라이팅 (per-vertex lighting) 과 같은 정점 당 동작들을 수행할 수도 있다. 정점 셰이더 스테이지 (38) 는 셰이더일 수도 있다. 헐 셰이더 스테이지 (40) 는 패치의 제어 포인트들을 정점 셰이더 스테이지 (38) 에 의해 프로세싱된 것으로서 수신하며, 제어 포인트들을 프로세싱하고, 프로세싱된 패치 (patch) 에 대한 제어 포인트들을 출력한다. 다르게 말하면, 헐 셰이더 스테이지 (40) 는 입력 패치를, 정점 셰이더 스테이지 (38) 에 의해 프로세싱된 것으로서 수신하며, 입력 패치를 프로세싱하고, 출력 패치를 출력한다. 헐 셰이더 스테이지 (40) 는 입력 패치를 프로세싱하기 위해 다양한 기능들을 수행할 수도 있다. 예를 들어, 헐 셰이더 스테이지 (40) 는 제어 포인트들의 로케이션들을 변경하기 위해 패치의 제어 포인트들의 좌표들을 수정할 수도 있거나, 또는 심지어 패치의 제어 포인트들을 추가 또는 삭제할 수도 있다.

[0055] 덧붙여서, 헐 셰이더 스테이지 (40) 는 얼마나 많은 프리미티브들이 정점 셰이더 스테이지 (38) 에 의해 생성된 패치 (즉, 출력 패치) 에 추가될 것인지를 나타내는 값들을 결정할 수도 있다. 헐 셰이더 스테이지 (40) 는

얼마나 많은 프리미티브들이 패치에 추가될 것인지를 결정하기 위해 다양한 기준을 이용할 수도 있다. 아래에서 설명되는 것은 얼마나 많은 프리미티브들이 패치에 추가될 것인지를 결정하는데 헐 셰이더 스테이지 (40) 가 이용할 수도 있는 2 개의 예의 기준들이다. 그러나, 본 개시물의 양태들은 그렇게 제한되지 않고, 헐 셰이더 스테이지 (40) 는 얼마나 많은 프리미티브들이 패치에 추가되어야할 지를 결정하는데 임의의 기준들을 이용할 수도 있다. 얼마나 많은 프리미티브들이 추가되어야 하는지의 결정에 기초하여, 헐 셰이더 스테이지 (40) 는 얼마나 많은 프리미티브들이 패치에 추가될 것인지를 나타내는 값들과 도메인 유형을 출력할 수도 있다.

[0056] 도메인 셰이더 스테이지 (44) 는 테셀레이션 (tessellation) 스테이지 (그림에는 없음) 로부터 정점 좌표들의 세트를 수신할 수도 있다. 도메인 셰이더 스테이지 (44) 는 테셀레이션 스테이지 (그림에는 없음) 에 의해 출력된 각각의 정점 좌표에 대해 실행할 수도 있다. 헐 셰이더 스테이지 (40) 로부터의 패치의 제어 포인트들의 좌표들로, 도메인 셰이더 스테이지 (44) 는 패치 상에서, 테셀레이션 스테이지에 의해 출력된 바와 같이, 정점의 로케이션을 결정할 수도 있다. 기하구조 셰이더 스테이지 (46) 는 도메인 셰이더 스테이지 (44) 에 의해 패치에 추가된 프리미티브들의 정점들을 수신하고, 심지어 더 높은 해상도를 추가하기 위해 프리미티브들에 대한 부가적인 정점들을 추가로 생성할 수도 있다.

[0057] 멀티-코어 시스템 (2) 은 CPU (16), 및 GPU (20) 간에 캐시 코히어런시를 구현하기 위해 다양한 기법들을 구현할 수도 있다. 본 개시물의 기법들에 따라, GPU (20) 는 공유 변수 데이터 (28) 를 GPU (20) 의 캐시 (22) 의 캐시 라인들에 저장하도록 구성될 수도 있다. GPU (20) 는 해제와-함께-저장 동작을 실행하도록 구성될 수도 있다. CPU (16) 는 그 다음에 획득과-함께-로드 동작을 실행하고, CPU (16) 의 캐시 (18) 로부터 공유 변수 데이터 (28) 의 값을 로드할 수도 있다.

[0058] 도 3a는 캐시 무효화 및 플러시 명령들을 포함할 수도 있는 GPU 커널에서의 동기 동작들을 수행하도록 구성된 GPU를 예시하는 개념도이다. 도 3a는 CPU (16), GPU (20), 및 시스템 메모리 (14) 를 예시한다. CPU (16) 는 운영 체제 (50), GPU 드라이버 (51), 및 GPU 컴파일러 (52) 를 실행한다. 운영 체제 (50) 는 태스크들, 이를테면 스레드들의 스케줄링을 수행할 수도 있고, GPU 드라이버 (51) 및 GPU 컴파일러 (52) 의 실행을 또한 관리할 수도 있다. GPU 드라이버 (51) 는 CPU (16) 및 GPU (20) 간의 상호작용을 관리할 수도 있고, 일부 예들에서, 본 개시물의 기법들에 따라 캐시 관리 명령들을 포함하기 위해 GPU (20) 의 명령들 (54A) 을 수정할 수도 있다. GPU 컴파일러 (52) 는 GPGPU 프로그램을 위한 코드를 컴파일할 수도 있고, 일부 예들에서, 본 개시물의 기법들에 따라 캐시 관리 명령들을 포함하기 위해 컴파일 시간에 코드를 수정할 수도 있다.

[0059] 도 3a에서, GPU (20) 는 명령들 (54A) 을 실행하려고 한다. GPU 컴파일러 (52) 는 드라이버 (51) 가 GPU (20) 의 메모리 속으로 로드했을 수도 있는 이전에 생성된 명령들 (54A) 을 가질 수도 있다. 명령들 (54A) 은 공유 변수 데이터 (28) 에 대한 일련의 읽기들 및 쓰기들을 포함한다. 이들 예들에서, 공유 변수 데이터 (28) 는 GPU (20) 및 다른 프로세서, 이를테면 CPU (16) 간에 공유될 수도 있고, GPU (20) 의 캐시 (22) 에 그리고 CPU (16) 의 캐시 (18) 에 저장될 수도 있다.

[0060] 명령들 (54A) 로 예시된 바와 같이, CPU (16) 는 공유 변수 데이터 (28) 의 값을 변경하는 쓰기를 실행할 수도 있으며, 그 뒤에, GPU (20) 는 그 값을 읽는다. 컴파일러 (52) 는, GPU (20) 가 명령들 (54A) 중 읽기 명령을 실행하는 경우, 런 타임에, 예컨대, 운영 체제 (50) 또는 드라이버 (51) 에 의해, CPU (16) 에 의한 공유 변수 데이터 (28) 에 대한 쓰기가 GPU (20) 로 코히어런트 방식으로 전파되는 것을 보장하는 동기 동작들로 교체될 수도 있는 쓰기 및 읽기 명령들에 입각하여 일련의 NOP (no operation) 명령들을 삽입할 수도 있다.

[0061] 하드웨어 명령 또는 소프트웨어 루틴, 이를테면 획득과-함께-로드 루틴 또는 명령 또는 해제와-함께-저장 루틴을 실행하는 프로세서가 인터럽트되지 않을 수도 있다. 그 루틴이 인터럽트되지 않을 수도 있는 하나의 이러한 방법이 임계 섹션 (critical section) 이라고 지칭되는 프로그래밍 구성 (programming construct) 을 사용하는데, 이는 하나의 프로세서가 특정 시간에 획득과-함께-로드 루틴을 실행하는 것만을 허용한다. 다른 예에서, 멀티-코어 시스템 (2) 에서의 인터럽트들을 일시적으로 가능하지 않게 함으로써 그 루틴이 인터럽트되지 않는 것을 프로세서가 보장할 수도 있다.

[0062] GPU 컴파일러 (52) 가 명령들 (54A) 을 생성한 후, GPU 컴파일러 (52) 또는 GPU 드라이버 (51) 는 본 개시물의 기법들에 따라 캐시 코히어런시를 유지하기 위하여 캐시 플러시들 및/또는 무효화들을 수행할 수도 있는 동기 동작들, 이를테면 획득과-함께-로드 동작들을 포함하기 위해 명령들 (54A) 을 삽입하거나 또는 수정할 수도 있다. 캐시 관리 명령들을 포함할 수도 있는, 동기 동작들에 대한 명령들 (54A) 을 수정한 결과는 도 3b에 예시되어 있다.

[0063] 도 3b는 동기 동작들을 위한 명령들을 GPU 커널에 삽입하는 컴파일러/드라이버를 예시하는 개념도이다. GPU 컴파일러 (52) 가 명령들 (54A) 을 생성한 후 (도 3a), GPU 컴파일러 (52) 또는 GPU 드라이버 (51) 는 동기 동작들을 포함하기 위해 명령들, 이를테면 캐시 관리 명령들을 수정할 수도 있다. GPU 컴파일러 (52) 또는 GPU 드라이버 (51) 는 명령들 (54A) 을 그 명령들이 공유된 변수를 액세스하는지의 여부에 기초하여 수정할 수도 있다. 이 예에서, 명령들 (54A) 은 공유 변수 데이터, 예컨대, 공유 변수 데이터 (28) 의 읽기가 뒤따르는 쓰기를 포함한다. GPU 컴파일러 (52) 또는 GPU 드라이버 (51) 는 쓰기 명령 후에 획득과-함께-로드 동작을 삽입함으로써 그리고 해제와-함께-저장 동작 후이지만 GPU (20) 에 의한 공유 변수 데이터 (28) 의 읽기 전에 해제와-함께-저장 동작들을 삽입함으로써 명령들 (54A) 을 수정할 수도 있다. 해제와-함께-저장 동작은 제 1 프로세서, 이를테면 GPU (20) 가 공유 변수 데이터 (28) 에 연관된 캐시 라인들을 플러시하는 것을 유발할 수도 있고 해제와-함께-저장 동작 후에 쓰기가 발생하지 않는 것을 보장한다. 획득과-함께-로드 동작은 GPU (20) 가 공유 변수 데이터 (28) 의 임의의 복사본들을 유지하는 CPU (16) 의 캐시 (18) 의 라인들을 무효화하는 것과, 예컨대, 시스템 메모리 (14) 로부터 공유 변수 데이터 (28) 의 값을 스누프하거나 또는 읽음으로써 캐시 필을 수행하는 것을 유발할 수도 있다. 획득과-함께-로드 동작은 추가로, 획득과-함께-로드 동작이 실행을 완료한 후에 읽기가 발생하는 것을 보장하며, 이에 의해, GPU (20) 에 의해 읽힌 공유 변수 데이터 (28) 의 값이 가장 최신 값인 것을 보장한다. 캐시 플러시들 및 무효화 명령들의 효과는 도 4a 및 도 4b에 관해 아래에서 설명된다.

[0064] GPU 컴파일러 (52) 또는 GPU 드라이버 (51) 는 공유된 변수들이 컴파일 시간에 또는 바인드 시간에 알려지는지의 여부에 기초하여 명령들 (54A) 을 수정할 수도 있다. 공유된 변수들이 컴파일 시간에 알려지는 경우에, GPU 컴파일러 (52) 는 컴파일 시간에 명령들 (54A) 을 수정할 수도 있다. 공유된 변수들이 바인드 시간에 알려지면, GPU 드라이버 (51) 는 바인드 시간에 명령들 (54A) 을 수정할 수도 있다. 어느 경우에나, 수정된 명령들은 명령들 (54B) 을 포함한다. 또한, 일부 예들에서, GPU 컴파일러 (52) 또는 GPU 드라이버 (51) 는 그 명령들을 커널의 명령들 속으로 삽입하지 않을 수도 있고, 그보다는, 컴파일러에 의해 원래 삽입된 NOP 명령들을 위에서 설명된 바와 같은 동기 동작들로 플레이스홀더 (placeholder) 들로서 교체할 수도 있다. 일단 GPU 드라이버 (51) 또는 GPU 컴파일러 (52) 가 명령들 (54A) 을 수정하면, GPU (20) 는, GPU (20) 및 멀티-코어 시스템 (2) 의 다른 프로세서들 간에 캐시 코히어런시를 보장하는 그런 방법으로 수정된 명령들 (54B) 을 실행할 수도 있다.

[0065] 도 4a는 오래된 (stale) 공유된 변수를 포함하는 캐시를 예시하는 개념도이다. 도 4a는 도 1의 시스템 메모리 (14), 및 GPU (20) 의 캐시 (22, 도 1) 를 예시한다. 비록 GPU (20) 와 GPU 캐시 (22) 에 관해 예시되었지만, 임의의 코어 또는 프로세서의 임의의 캐시가 도 4a 및 도 4b에서 예시된 기법들을 수행할 수도 있다.

[0066] 캐시 (22) 는 5 개의 엔트리들을 포함하는데, 그것들의 각각은 수직 직사각형들로서 표현된다. 캐시 (22) 에서의 엔트리들의 수는 예의 목적을 위해 크게 감소되었다. 캐시 (22) 는 (대각선 해싱을 가지는 캐시 엔트리들에 의해 나타내어진) 오래된 데이터를 가지는 2 개의 엔트리들을 포함한다. 오래된 데이터는 멀티-코어 시스템 (2) 에서 다른 프로세스에 의해 변할 수도 있는 공유된 변수 값들이다. 도 4a 및 도 4b에서의 화살표들은, GPU (20) 가 화살표들 중 임의의 것의 꼬리에 의해 나타내어진 시스템 메모리 (14) 의 로케이션으로부터 공유된 데이터를 읽는다면, 그 값이 화살표의 머리에 의해 나타내어진 캐시 (22) 의 로케이션에 저장되도록 시스템 메모리 (14) 의 로케이션과 캐시 (22) 에서의 엔트리 사이의 연관을 나타낸다.

[0067] 오래된 데이터를 포함하는 캐시 (22) 의 라인으로부터 읽은 후, GPU (20) 는 획득과-함께-로드 동작을 실행할 수도 있다. 획득과-함께-로드 동작은 CPU (16) 가 캐시 (18) 의 하나 이상의 라인들을 무효화하는 것을 유발하며, 캐시 (18) 의 캐시 필을 유발하고, 획득과-함께-로드 동작의 실행 전의 실행을 위해 읽기들이 재순서화되지 않는 것을 보장한다.

[0068] 위에서 지적했듯이, 캐시 라인 (62) 이 획득과-함께-로드 동작으로 인해 무효로서 마킹되기 때문에, GPU (20) 는 시스템 메모리 (14) 의 주소 (64) 로부터 공유 변수 데이터 (28) 의 새로운 복사본을 페치한다. 일부 예들에서, GPU (20) 는 시스템 메모리로부터 새로운 복사본을 페치하는 것을 필요로 하지 않을 수도 있고, 그보다는, CPU (16) 및 GPU (20) 사이에서 공유되는 상위 레벨 캐시, 예컨대, 레벨 2 또는 레벨 3 캐시로부터 새로운 복사본을 페치하는 것을 필요로 할 수도 있다. 다른 예에서, CPU (16) 는 캐시 라인 (62) 에 저장된 공유 변수 데이터 (28) 의 값을 수정 또는 "스누프" 및 업데이트할 수도 있다.

[0069] 도 4b는, 예컨대, 획득과-함께-로드 동작을 실행한 결과로서, 무효로서 마킹된 캐시 라인으로부터 읽는 것으로 인해 메인 메모리로부터 캐시를 업데이트하는 것을 예시하는 개념도이다. 도 4b는 무효로서 마킹되는 캐시

라인 (62) 으로부터 읽고 시스템 메모리 (14) 의 로케이션 (62) 에 저장된 공유 변수 데이터 (28) 의 최신 복사본을 페치한 GPU (20) 의 결과들을 예시한다. 비록 시스템 메모리 (14) 로부터 읽는 것으로서 예시되지만, GPU (20) 는 또한, 다른 캐시, 이를테면 CPU (16) 의 캐시 (18) 로부터 공유 변수 데이터 (28) 의 새로운 값을 읽을 수도 있다. CPU (16) 가 캐시 라인 (62) 을 무효로서 마킹한 후, 도 4a에 관해 위에서 설명된 바와 같이, GPU (20) 는 시스템 메모리 로케이션 (64) 으로부터 공유 변수 데이터 (28) 의 값을 읽을 수도 있고, 시스템 메모리 로케이션 (64) 으로부터의 공유 변수 데이터 (28) 를 캐시 로케이션 (62) 에 저장한다. 비록 단일 캐시 엔트리만을 무효화하는 것으로서 예시되지만, 캐시 무효화 명령은 또한, 캐시 (22) 의 모든 또는 다수의 엔트리들, 이를테면 공유된 것으로 마킹되는 모든 캐시 엔트리들을 무효화할 수도 있다. 공유된 캐시 엔트리들은 아래에서 더 상세히 논의된다.

[0070] 도 5a는 오래된 공유된 변수 값을 포함하는 시스템 메모리를 예시하는 개념도이다. 도 5a에서, 캐시 (22) 의 라인 (80) 에 저장된 공유 변수 데이터 (28) 의 값은 최신이지만, 시스템 메모리 (14) 의 로케이션 (82) 에 되 쓰여지지 않는다. 이 상황은, 예를 들어, GPU (20) 가 공유 변수 데이터 (28) 에 대한 새로운 값을 계산하고 공유 변수 데이터 (28) 의 그 값을 캐시 (22) 에는 쓰지만 시스템 메모리 (14) 에는 쓰지 않는 경우에 발생한다.

[0071] 시스템 메모리 (14) 의 로케이션 (82) 이 공유된 변수의 현재 값을 가지는 것을 보장하기 위해, GPU (20) 는 도 4a에 관해 설명된 프로세스와 유사한 프로세스를 수행한다. 도 4a에 관해 설명된 기법과 유사하게, GPU (20) 는 동기 동작, 이를테면 해제와-함께-저장 동작의 실행을 수행한다. 해제와-함께-저장 동작의 실행 전에, GPU (20) 는 하나 이상의 쓰기들을 수행하여, 캐시 라인 (80) 의 업데이트된 값이 생겨나게 할 수도 있는데, 이는 시스템 메모리 (14) 의 메모리 주소 (82) 로 전파되지 않는다. 캐시 라인 (80) 에 대한 쓰기들을 수행한 후, GPU (20) 는 해제와-함께-저장 동작을 실행하는데, 이는 GPU (20) 가 캐시 라인 (80) 에 저장된 공유 변수 데이터 (28) 의 값을 시스템 메모리 (14) 의 주소 (82) 로 플러시하는 것을 유발할 수도 있다. 플러시하는 것은, 일부 예들에서, 해제와-함께-저장 동작은 또한, GPU (20) 또는 다른 프로세서가 오래된 공유 변수 데이터 (28) 의 값들을, 예컨대, CPU (16) 의 캐시 (18) 또는 DSP (24) 의 캐시 (26) 에 저장하는 하나 이상의 캐시 라인들을 무효화하는 것을 유발할 수도 있다. 이 메모리 상태는 도 5a에 예시되어 있다.

[0072] 해제와-함께-저장 동작은 추가로, 공유 변수 데이터 (28) 에 대한 쓰기가 획득과-함께-로드 동작의 실행 후의 실행을 위해 재순서화되지 않는 것을 보장한다. 해제와-함께-저장 명령은 해제와-함께-저장 동작이 실행을 완료한 후까지 후속 명령이 발행되지 않도록 특수 비트를 사용하여 명령 스트림에서 하나 이상의 후속 명령들을 태깅함으로써 이 실행 순서화를 보장할 수도 있다.

[0073] 해제와-함께-저장 명령은 또한, 프로그램 코드 시퀀스에서 해제와-함께-저장 명령 전에 발생하는 임의의 쓰기들이, 해제와-함께-저장 명령이 실행되기 전에 완료하는 것을 보장한다. 임의의 이전의 쓰기들이 해제와-함께-저장 명령 전에 완료하는 것을 보장하기 위해, 프로세서, 이를테면 GPU (12) 는 GPU (12) 의 캐시의 전부 또는 일부를 플러시하는 마이크로시퀀스의 명령들을 실행할 수도 있다. GPU (12) 는 마이크로시퀀스 캐시 플러시의 완료 시에만 해제와-함께-저장 명령의 저장된 값을 쓴다.

[0074] 도 5b는 캐시 플러시를 수행한 후의 시스템 메모리의 상태를 예시하는 개념도이다. 도 5b에서, GPU (20) 는 캐시 (22) 의 캐시 라인 (80) 에 저장된 공유 변수 데이터 (28) 의 값을 시스템 메모리 (14) 의 로케이션 (82) 에 되 쓰기를 하였다. GPU (20) 는 공유 변수 데이터 (28) 의 값을 해제와-함께-저장 동작의 일부로서 썼을 수도 있는데, 이 동작은, 일부 경우들에서, GPU (20) 가 공유 변수 데이터 (28) 의 값에 대응하는 캐시 라인들, 예컨대, 캐시 라인 (80) 상에서 캐시 플러시를 수행하는 것을 유발할 수도 있다. 비록 캐시 플러시가 로케이션 (80) 의 값만을 시스템 메모리 로케이션 (82) 으로 되 플러시하는 것으로서 도 5b에서 예시되고 있지만, 플러시는 캐시 (80) 의 모든 또는 다수의 값들을 시스템 메모리 (14) 로 되 플러시할 수도 있다.

[0075] 도 6은 본 개시물의 기법들에 따른 GPU 및 CPU의 캐시들 및 메모리 관리 유닛들을 예시하는 개념도이다. 도 6은 메모리 관리 유닛 (MMU) (102), 및 물리적 캐시 (104) 를 추가로 구비하는 CPU (16) 를 예시한다. 도 6은 또한 가상 캐시 (106), 및 시스템 메모리 관리 유닛 (SMMU) (108) 을 구비하는 GPU (20) 를 예시한다. CPU (16) 와 GPU (20) 는 시스템 메모리 (14) 에 커플링되는데, 그 시스템 메모리는 공유 변수 데이터 (28) 를 포함할 수도 있다. 비록 GPU (20) 가 가상 캐시를 가지는 것으로서 예시되고 CPU (16) 가 물리적 캐시를 가지는 것으로 예시되지만, 가상 및/또는 물리적 캐시들의 임의의 조합이 가능할 수도 있다.

[0076] 물리적 캐시 (104) 는 다양한 예들에서 L1, L2, 또는 L3 캐시 중 하나 이상을 포함할 수도 있다. 물리적 캐시 (104) 는 또한, MESI 또는 MOESI 캐시 코히어런스 프로토콜들을 구현하는 I/O 코히어런트 캐시를 포함한다.

예시의 목적으로, 물리적 캐시 (104) 는 복수의 캐시 라인들을 포함한다. 물리적 캐시 (104) 의 캐시 라인들의 각각은 인덱스라고 지칭되는 물리적 주소의 부분 (보통 가상 주소의 다수의 최소 유효 비트들) 에 기초하여 인덱싱된다. 캐시 엔트리가 또한 "태그"를 포함하는데, 그 태그는 캐시에 저장된 데이터에 연관된 주소의 최대 유효 비트들을 포함한다. 물리적 캐시, 이를테면 물리적 캐시 (104) 의 경우, 태그는 캐싱된 데이터에 연관된 물리적 주소의 최대 유효 비트들을 포함한다. 특정 주소가 캐싱되는지의 여부를 결정하기 위해, MMU (102) 는 번역 색인 버퍼 (translation lookaside buffer; TLB) 에 저장될 수도 있는 페이지 테이블로부터 가상 주소를 번역하고, 번역된 물리적 주소의 최대 유효 비트들과 물리적 캐시의 태그를 그 2 개가 일치하는지의 여부를 결정하기 위해 비교한다. 2 개가 일치하면, 주소에 연관된 데이터는 캐시에 저장되는데, 이는 캐시 "히트"라고 지칭된다. 그렇지 않으면, 그 데이터는 캐시에 저장되지 않는데, 이는 캐시 "미스"라고 지칭된다.

[0077] 각각의 캐시 라인은 또한, 오프셋을 포함할 수도 있다. 페이지 테이블 엔트리 및 캐시 테이블 엔트리의 사이즈가 워드 사이즈 (즉, 프로세서의 최대 주소가능 워드) 보다 (예컨대, 킬로바이트 또는 메가바이트) 더 클 수도 있기 때문에, CPU (16) 는 캐시 엔트리 내의 특정 워드를 특정하는 오프셋의 값에 기초하여 캐시 라인 데이터의 특정 워드에 인덱싱할 수도 있다.

[0078] 물리적 캐시 (104) 및/또는 가상 캐시 (106) 의 각각의 캐시 라인은 가상 캐시 (106) 의 개개의 캐시 라인들에 연관된 상태, 예컨대 하나 이상의 상태 비트들을 포함할 수도 있다. 그 상태는 가상 캐시 (106) 의 특정 캐시 라인이 더티인지, 유효인지, 또는 무효인지를 나타낼 수도 있다. 덧붙여, 그 상태는 캐시 라인의 개개의 바이트들에 적용할 수도 있다. 물리적 캐시 (104) 및/또는 가상 캐시 (106) 가 유지하는 상태는 도 7에 관해 아래에서 더 상세히 설명된다.

[0079] 도 6은 또한, GPU (20) 를 예시하는데, 그 GPU는 가상 캐시 (106) 및 SMMU (108) 를 더 포함한다. 가상 캐시 (106) 는, 물리적 캐시 (104) 와는 달리, 물리적 주소들보다는 가상 주소에 기초하여 캐싱된 데이터를 태깅할 수도 있다. 물리적 캐시 (104) 가 가상 주소를 물리적 주소로 번역하고 다수의 최대 유효 비트들을 캐시 라인의 태그로서 저장할 수도 있는 반면, 가상 캐시 (106) 는 가상 주소의 다수의 비트들을 캐시 라인 태그로서 저장할 수도 있다. 이러한 캐시는 "가상적으로 태깅된다"고 지칭될 수도 있다. 가상적으로 태깅된 캐시를 이용함으로써, SMMU (108) 는 가상 캐시 (106) 를 액세스하는 경우에 가상 주소로부터 물리적 주소로 번역하는 것을 필요로 하지 않고, 대신, 캐시에 액세스한 후에 가상 주소 태그를 물리적 주소로 번역한다. 반면에, MMU (102) 는 물리적 캐시 (104) 에 액세스하기 위해 물리적 주소로부터 가상 주소로 번역한다.

[0080] 메모리의 각각의 청크 (chunk) 또는 페이지는 "물리적 주소"라고 지칭되는 주소에 의해 액세스가능하다. 그러나, 애플리케이션들이 연속하는 주소 공간을 예상할 수도 있는 반면, 이용가능한 물리적 주소 범위들은 불연속적일 수도 있다. 각각의 애플리케이션에 대해 연속하는 메모리 주소 공간을 제공하기 위해, CPU (16) 와 GPU (20) 는 가상 어드레싱을 구현할 수도 있다. 가상 어드레싱에서, 애플리케이션들이 가상 주소 공간들에 할당되는데, 그 가상 주소 공간들은 물리적 주소들에 매핑된다. 가상 주소와 물리적 주소 사이의 매핑들은 페이지 테이블 엔트리들에 저장된다. 페이지 테이블 엔트리들은 하나 이상의 페이지 테이블들에 저장될 수도 있다. 다양한 예들에서, CPU (16) 와 GPU (20) 는 별개의 페이지 테이블들 또는 단일, 공통 페이지 테이블을 각각 가질 수도 있다.

[0081] 각각의 페이지 테이블 엔트리는 특정 가상 주소에 연관된 물리적 주소의 다수의 최대 유효 비트들, 및 속성 데이터를 포함하는 다수의 비트들을 포함한다. 그 물리적 주소에 기초하여, MMU (102) 와 SMMU는 물리적 주소에 연관된 데이터에 액세스할 수도 있는데, 그 데이터는 시스템 메모리 (14) 의, 또는 다른 저장 디바이스, 이를테면 하드 디스크 드라이브, 또는 고체 상태 드라이브 상의 특정 주소에 저장될 수도 있다.

[0082] 가상 주소와 물리적 주소 간을 번역하는 프로세스는 원치 않는 양들의 레이턴시를 초래할 수도 있다. 가상 대 물리적 주소 번역에 연관된 레이턴시를 줄이기 위해, MMU (102) 와 SMMU (108) 는 번역 색인 버퍼 (TLB), 이를테면 TLB (106) 및 TLB (110) 를 각각 포함할 수도 있다. TLB (106) 와 TLB (110) 각각은 하나 이상의 빈번하게 액세스되는 물리적 주소들에 연관된 빈번하게 사용되는 페이지 테이블 엔트리들을 캐시한다. MMU (102) 와 SMMU (108) 의 각각의 엔트리는 페이지 테이블 엔트리 (page table entry; PTE) 이다.

[0083] MMU (102), 물리적 캐시 (104), 가상 캐시 (106), 및 SMMU (108) 의 각각은 또한, 시스템 메모리 (14) 에 저장된 페이지 테이블에 쓸 수도 있고 그 테이블로부터 읽을 수도 있다. 예를 들어, 특정 페이지 테이블 엔트리가 MMU (102) 또는 SMMU (108) 의 TLB로부터 퇴거 (eviction) 되면, MMU는 퇴거된 PTE를 시스템 메모리 (14) 에 쓸 수도 있다. TLB가 요청된 PTE의 값을 포함하지 않으면 (TLB 미스라고 지칭됨), MMU (102) 또는 SMMU

(108) 는 요청된 PTE를 시스템 메모리 (14) 로부터 취출할 수도 있다.

[0084] 일부 예들에서, MMU (102) 와 SMMU (108) 는 캐시 필들 및 퇴거들을 또한 핸들링할 수도 있다. 일부 예들에서, MMU (102) 또는 SMMU (108) 는 시스템 메모리 (14) 로부터 물리적 캐시 (104) 에 있지 않은 값을 취출하고 그 값을 물리적 캐시 (104) 또는 가상 캐시 (106) 의 라인에 저장함으로써 캐시 필을 수행할 수도 있다. 캐시 라인이 퇴거되면, 즉, 캐시 엔트리를 위한 방 (room) 이 없으면, MMU (102) 또는 SMMU (108) 는 물리적 캐시 (104) 또는 가상 캐시 (106) 의 퇴거된 캐시 라인을 시스템 메모리 (14) 에 쓸 수도 있다.

[0085] MMU (102) 와 SMMU (108) 는 또한, "스누핑 (snooping)" 동작들을 수행할 수도 있다. 스누프 동안, MMU, 이를테면 SMMU (108) 는 다른 프로세서의 캐시 라인들, 예컨대, CPU (16) 의 물리적 캐시 (104) 를 검사할 수도 있다. 일부 경우들에서, SMMU (108) 는 물리적 캐시 (104) 의 하나 이상의 캐시 라인들의 값들을 업데이트할 수도 있다. 예를 들면, GPU (20) 는 해제와-함께-저장 동작을 실행함으로써 가상 캐시 (106) 에 저장된 공유 변수 데이터 (28) 의 값을 업데이트할 수도 있다. 해제와-함께-저장 동작 동안, SMMU (108) 는 물리적 캐시 (104) 를 스누프하고 물리적 캐시 (104) 의 대응하는 캐시 라인을 업데이트할 수도 있다. CPU (16) 가 공유 변수 데이터 (28) 의 값에 액세스할 것을 시도하면, 공유 변수 데이터 (28) 의 가장 최신 값은 물리적 캐시 (104) 에 이미 저장되어 있다. 따라서, CPU (16) 는 물리적 캐시 (104) 가 가장 최신 값을 가짐을 검출할 수도 있고, 예컨대 CPU (16) 가 획득과-함께-로드 동작을 실행하는 경우, 물리적 캐시 (104) 에 저장된 캐싱된 값을 단순히 읽는다.

[0086] 이 예에서, CPU (16) 는 공유 변수 데이터 (28) 의 값에 연관된 캐시 라인들을 무효화하지 않거나 또는 CPU (16) 는 시스템 메모리 (14) 로부터 공유 변수 데이터 (28) 의 새로운 값을 페치하지 않는다. 이런 방식으로, 본 개시물에 따른 스누핑 기법들은 공유 변수 데이터에 쓰는 경우 다중프로세서 시스템의 성능을 개선할 수도 있다.

[0087] 일부 예들에서, GPU (20) 는 가상 캐시 (106) 가 아니라, 온-디맨드 코히어런스를 지원하는 물리적 캐시를 구현할 수도 있다. GPU (20) 가 물리적 캐시, 이를테면 물리적 캐시 (104) 에 유사한 캐시를 구현하면, GPU (20) 는 가상 주소들이 아니라, 그것들의 연관된 물리적 주소들에 기초하여 물리적 캐시에서의 캐시 라인들 엔트리들에 태깅한다. GPU (20) 가 물리적 캐시를 포함하는 예에서, TLB (110) 는 물리적 캐시에 쓰기 전에 가상 주소를 물리적 주소로 번역한다.

[0088] 도 7은 본 개시물의 기법들에 따른 속성 비트들을 구현하는 캐시 라인을 예시하는 개념도이다. 위에서 설명된 바와 같이, 가상 캐시 (106) 는 복수의 캐시 라인들을 포함한다. 도 7은, 데이터 부분 (122), 가상 주소 태그 (124) 를 포함하고, 옵션의 공유가능 비트 (126), CleanValid bit (128), DirtyValid bit (130), 및 ByteEnables 비트들 (132) 을 포함할 수도 있는 단일 캐시 라인 (120) 을 예시한다.

[0089] 공유된 비트 (126) 가, 캐시 라인에 포함될 수도 있거나 또는 포함되지 않을 수도 있는 옵션의 비트이다. 공유된 비트 (126) 의 값은 데이터의 임의의 바이트들이 당해 특정 캐시 라인에서 공유되는지의 여부를 나타내고, 그러므로 다른 프로세서들, 예컨대 CPU (16), 및/또는 DSP (24) 에 가시적이다. 캐시 라인이 다른 프로세서들에 가시적이면, 데이터의 공유된 바이트들의 값에 대한 변경들은 시스템 메모리 (14) 로 플러시되는 것을 필요로 할 수도 있는데, 다른 프로세서들이 공유된 캐시 라인들의 현재 값을 가져야만 하기 때문이다. 비공유 (unshared) 값들이 시스템 메모리 (14) 에 대해 플러시되는 것은 필요하지 않다. 다양한 예들에서, 운영 체제 (OS) 가 공유된 비트의 디폴트 값을 설정할 수도 있거나, 또는 사용자 또는 프로그램이 공유된 비트 (126) 의 디폴트 값을 설정하기 위해 API (Application Programming Interface) 호출을 할 수도 있다. 프로세서에 의한 후속하는 읽기들이, 일부 예들에서, 공유된 비트 (126) 의 값을 변경할 수도 있다. 캐시 미스 동안, 캐시 라인이 가상 캐시 (106) 속으로 이동되는 것이 필요한 경우, 공유된 비트 (126) 의 값은 디폴트로 공유 상태로 설정될 수도 있고, 동일한 캐시 라인에 대한 후속하는 캐시 액세스들 (즉, 캐시 히트들) 은 공유된 비트 (126) 의 상태를 변경시키지 않을 수도 있다.

[0090] ByteEnables 비트들 (132) 은 캐시 데이터 (122) 의 각각의 바이트를 위한 비트를 포함한다. ByteEnables 비트들 (132) 의 각각의 비트는 당해 특정 캐시 바이트가 더티인지의 여부를 나타내고, 다른 캐시에 또는 시스템 메모리 (14) 에 쓰이는 것을 필요로 할 수도 있다. DirtyValid 비트 (130) 는 ByteEnables 비트들 (132) 의 각각의 논리적 OR와 동일하다. DirtyValid 비트 (130) 가 1과 동일하고 공유된 비트 (126) 가 1과 동일하면, SMMU (108) 는 캐시 플러시 동작 동안 임의의 더티 바이트들의 값들을 시스템 메모리 (14) 에 쓸 수도 있다.

[0091] CleanValid 비트 (128) 는 캐시 라인 내에 임의의 비-더티 데이터 바이트들이 있는지의 여부를 나타낸다. 캐시 라인이 공유된다는 것을 공유된 비트 (126) 가 나타내고 CleanValid 비트 (128) 가 1과 동일하면, SMMU (108) 는, 캐시 무효화 동작을 수행하는 경우, 캐시 라인, 예컨대 캐시 라인 (120) 을 무효화하는데, 최신이 아닌 데이터 (122) 의 적어도 하나의 바이트가 있을 수도 있기 때문이다.

[0092] 도 8은 본 개시물의 기법들에 따른 GPU 캐시를 예시하는 개념도이다. 도 8의 예에서, GPU (20) 는 가상 캐시 (106) 를 포함한다. 가상 캐시 (106) 는 복수의 캐시 라인들을 포함한다. 그 라인들의 각각은 도 7의 예에서 설명된 바와 같은 속성들, 이를테면 공유된 비트 (126), CleanValid 비트 (128), DirtyValid 비트 (130), 및 ByteEnables 비트들 (132) 이다. 각각의 캐시 라인의 상태들은 "클린 (clean)", "유효", "무효", 및 "더티"의 혼합체일 수도 있다. 본 개시물의 기법들은, 프로세서의 메모리 관리 유닛 (MMU), 이를테면 GPU (20) 의 SMMU (108) 가 가상 캐시 (106) 의 각각의 라인의 상태를 나타내는 캐시의 각각의 라인에 대한 상태 비트들을 유지하는 것을 가능하게 할 수도 있다.

[0093] SMMU (108) 는, 캐시 라인이 시스템 메모리 (14) 또는 다른 메모리에 쓰여지지 않은 하나 이상의 업데이트된 값들을 포함하면, 그 캐시 라인을 더티로서 마킹할 수도 있다. SMMU (108) 는, 캐시 라인의 콘텐츠들이 최신이면, 즉, 다른 프로세서가 당해 캐시 엔트리에 연관된 물리적 주소에 더 새로운 값을 쓰지 않았으면, 그 캐시 라인을 유효로서 마킹할 수도 있다. 반면에, SMMU (108) 는, 캐시 라인이 특정 메모리 주소의 가장 최신 복사본을 포함하지 않으면, 그 캐시 라인을 무효로서 마킹할 수도 있다. SMMU (108) 가 무효 라인에 액세스하는 것을 시도하는 경우, SMMU (108) 는 다른 캐시로부터, 또는 메인 메모리로부터 공유된 데이터의 새로운 복사본을 취출해야만 한다.

[0094] 도 8의 예에서, 어떠한 해싱도 없는 캐시 라인들, 이를테면 캐시 라인 (140) 이, 클린 및 유효 캐시 라인들을 나타낸다. 대각선 해싱을 갖는 캐시 라인들, 이를테면 캐시 라인 (144) 은 무효 캐시 라인들을 나타내고, 수직 해싱을 갖는 캐시 라인들은 더티 캐시 라인들을 나타낸다. 캐시 라인들은 또한, 블록들 또는 지역들, 예컨대, 상이한 캐시 상태들을 갖는 바이트 사이즈의 지역들로 나누어질 수도 있다. 일 예로서, 캐시 라인 (142) 은 무효 및 더티 양쪽 모두인 블록들을 포함하고, 캐시 라인 (146) 은 더티 및 클린 양쪽 모두인 (뿐만 아니라 유효한) 블록들을 포함한다. 정의에 의해, 클린이거나 또는 더티인 임의의 블록이 또한 유효하다.

그러나, 동일한 캐시 라인 내의 다른 블록들이 무효일 수도 있다. 본 개시물의 기법들에 따라 가상 캐시 (106) 에 저장된 각각의 블록의 상태를 유지함으로써, 캐시 퇴거 또는 캐시 필을 수행하는 경우에 불필요한 캐시 플러시들 또는 무효화들을 줄이는 것이 가능할 수도 있다.

[0095] 위에서 설명된 바와 같이, 공유 변수 데이터 (28) 를 포함하는 캐시 라인이 공유된 값, 이를테면 공유 변수 데이터 (28) 의 가장 최신 값을 가지지 않으면, GPU (20) 는 획득과-함께-로드 동작을 실행하는 일부로서 최신 값을 취출할 수도 있다. 다양한 예들에서, 최신 공유된 변수 값을 취출하는 것은, SMMU (108) 가, 예컨대 시스템 메모리 (14) 로부터 최신 값을 갖는 캐시 라인 또는 캐시 라인의 부분을 채우는 것 ("캐시 필"이라고 지칭됨) 을 포함할 수도 있다.

[0096] 캐시 필을 수행하는 경우에 전체 캐시 라인을 채우는 것은 일부 예들에서 바람직하지 않을 수도 있다. 따라서, 본 개시물의 기법들은, 위에서 설명된 바와 같이, 캐시 라인의 개개의 블록들의 캐시 상태들을 마킹한다.

각각의 블록의 상태에 기초하여, SMMU (108) 는 공유 변수 데이터 (28) 의 최신 값들을 포함하지 않는 블록들을 단지 무효화하고 업데이트하거나, 또는 플러시할 수도 있다. 일 예로서, 캐시 라인, 이를테면 캐시 라인 (144) 이 더티 및 무효 양쪽 모두의 블록들을 포함하면, SMMU (108) 은 캐시 필을 수행하는 경우 무효 블록들에 대응하는 캐시 데이터만을 취출할 수도 있다. 더티로서 마킹된 블록들은 캐시 필을 수행하는 경우에 업데이트되지 않는데, 더티 블록들이 공유 변수 데이터의 가장 최신 값들을 포함하기 때문이다. 비-더티 블록들만을 채우는 것은 캐시 필을 수행하는 것에 연관된 레이턴시 및/또는 소비되는 대역폭을 감소시킬 수도 있다.

[0097] 위에서 설명된 바와 같이, GPU (20) 는 해제와-함께-저장 동작을 실행하는 경우 캐시 플러시 동작을 수행할 수도 있다. 캐시 플러시는 시스템 메모리 (14) 에 공유 변수 데이터 (28) 의 값들을 쓸 수도 있다. 하나의 기법에서, MMU (108) 는 캐시 플러시 동안 전체 캐시 라인을 시스템 메모리 (14) 에 쓸 수도 있다. 그러나, 개개의 블록들이 아니라 전체 캐시 라인을 쓰는 것은, 일부 경우들에서 불필요할 수도 있다. 예를 들어, 캐시 라인이 더티 블록들 및 무효 블록들 (예컨대, 캐시 라인 (142)), 또는 더티 블록들 및 유효 블록들 (예컨대, 캐시 라인 (146)) 을 포함할 수도 있다. 캐시 플러시 동안, 정의에 의해, 공유된 데이터의 가장 새롭고 최신인 값들을 가지는 더티 블록들만이 시스템 메모리 (14) 에 쓰여지는 것이 필요하다. 따라서,

SMMU (108) 는 캐시 플러시 동작을 수행하는 경우 더티 블록들만을 시스템 메모리 (14) 에 쓸 수도 있다.

[0098] 더티 및 무효 양쪽 모두의 블록들을 포함하는 가상 캐시 (106) 의 캐시 라인의 경우, SMMU (108) 는 더티 블록들을 시스템 메모리 (14) 에 쓸 수도 있고, 그 쓰기의 완료 시, 가상 캐시 (106) 의 더티 블록들의 각각을 무효로서 마킹할 수도 있다. 가상 캐시 (106) 의 캐시 라인, 이를테면 캐시 라인 (146) 이 더티 및 유효 양쪽 모두의 블록들을 포함하는 경우, SMMU (108) 는 더티 블록들을 시스템 메모리 (14) 에 쓸 수도 있고, 그 쓰기의 완료 시, 그 더티 블록들을 유효로서 마킹할 수도 있다. 이런 방식으로, 캐시 필을 수행하는 SMMU (108) 에 연관된 데이터, 대역폭 및/또는 레이턴시는, 일부 경우들에서, 전체 캐시 라인을 쓰는 것에 비하여 감소될 수도 있다.

[0099] SMMU (108) 는 또한, 예컨대 캐시 압력으로 인해 교체되고 있는 캐시 라인을 SMMU (108) 가 교체하는 일부 경우들에서 라인 퇴거를 수행할 수도 있다. 캐시 퇴거 프로세스의 일부로서, SMMU (108) 는 퇴거될 라인을 시스템 메모리 (14) 에 쓴다. 라인 퇴거에서, SMMU (108) 는 우선, 더티로서 마킹된 임의의 바이트들 또는 워드들을 시스템 메모리 (14) 에 쓸 수도 있다. SMMU (108) 는 그 다음에 퇴거될 전체 캐시 라인을 무효화할 수도 있다. SMMU (108) 가 더티로서 마킹된 블록들만을 쓰게 함으로써, 본 개시물의 기법들은 캐시 퇴거의 수행에 연관된 데이터, 대역폭, 및/또는 시간을 감소시킬 수도 있다.

[0100] 도 9는 본 개시물에서 설명되는 하나 이상의 예들에 따른 멀티-코어 시스템에서 캐시 코히어런시를 유지하는 프로세스를 예시하는 흐름도이다. 예시만의 목적을 위해, 도 1의 GPU (20) 가 참조되지만, 멀티-코어 시스템 (2) 의 임의의 프로세서가 본 개시물의 기법들을 수행할 수도 있다.

[0101] 도 9에서, 제 1 프로세서, 예컨대 GPU (20) 는, 제 1 프로세서의 제 1 캐시의 캐시 라인들에 공유 변수 데이터 (28) 를 쓰도록 구성될 수도 있다 (300). GPU (20) 는 그 다음에 해제와-함께-저장 동작을 실행하거나 또는 펜스를 해제할 수도 있다 (302). 일부 예들에서, 해제와-함께-저장 동작을 실행하는 것은, GPU (20) 가 공유 변수 데이터 (28) 에 연관된 GPU (20) 의 캐시 (22) 의 캐시 라인들을 플러시하는 것을 유발할 수도 있다. 그 방법은 추가로, 제 2 프로세서, 예컨대 CPU (16) 가, 획득과-함께-로드 동작을 실행하는 것 또는 펜스를 획득하는 것을 포함할 수도 있다 (306). 일부 예들에서, 획득과-함께-로드 동작을 실행하는 것은 CPU (16) 가 공유 변수 데이터 (28) 에 연관된 캐시 (18) 의 캐시 라인들을 무효화하는 것을 유발할 수도 있다 (308). CPU (16) 는 그 다음에 캐시 (18) 로부터 공유 변수 데이터 (28) 의 값을 로드할 수도 있다 (310).

[0102] 일부 예들에서, 해제와-함께-저장 동작을 실행하는 것은 추가로, 제 1 프로그램가능 프로세서로, 공유 변수 데이터에 연관된 제 1 캐시의 캐시 라인들을 플러시하는 것과, 제 1 프로그램가능 프로세서로, 임의의 이전의 저장들이 완료하기를 기다리는 것을 더 포함할 수도 있다.

[0103] 일 예에서, 획득과-함께-로드 동작을 실행하는 것은 추가로, 제 2 프로그램가능 프로세서로, 공유 변수 데이터에 연관된 제 2 프로그램가능 프로세서의 캐시 라인들을 무효화하는 것, 제 2 프로그램가능 프로세서로, 상기 공유 변수 데이터에 연관된 제 2 캐시의 캐시 라인들을 채우는 (fill) 것, 및 제 2 프로그램가능 프로세서로, 획득과-함께-로드 동작이 실행을 완료하기까지 후속 명령들의 발행을 방지하는 것을 포함한다.

[0104] 일부 예들에서, 도 9의 방법은 추가로, 제 2 프로세서로, 제 1 프로그램가능 프로세서의 캐시를 스누프하는 것, 제 2 프로그램가능 프로세서로, 공유 변수 데이터의 업데이트된 값에 연관된 캐시 히트를 검출하는 것, 및 제 2 프로그램가능 프로세서로, 공유 변수 데이터에 연관된 캐시 히트의 검출에 응답하여 제 2 프로그램가능 프로세서의 캐시에 공유 변수 데이터의 업데이트된 값을 쓰는 것을 포함한다.

[0105] 일부 예들에서, 해제와-함께-저장 동작을 실행하는 것은 추가로, 제 1 캐시보다 높은 레벨을 갖는 상위 레벨 캐시, 시스템 메모리, 및 제 2 프로세서의 제 2 캐시 중 하나에 공유 변수 데이터를 쓰는 것을 포함한다.

[0106] 다른 예에서, 제 1 프로세서는 중앙 프로세싱 유닛, 예컨대, CPU (16) 를 포함하고, 제 2 프로세서는 그래픽 프로세싱 유닛 (GPU), 예컨대, GPU (20) 를 포함한다. 제 2 캐시의 캐시 라인들을 무효화하기 위해, GPU (20) 는, 컴파일러로, GPU에 의해 작동가능한 셰이더 코드 속으로 무효화 명령을 삽입할 수도 있다.

[0107] 다른 예에서, 제 1 프로세서의 캐시 라인들의 모두 및 제 2 프로세서의 캐시 라인들의 모두가 공유된다. 해제와-함께-저장 동작을 실행하는 것은 추가로, 제 1 프로세서의 캐시 라인들의 모두를 플러시하는 것을 포함할 수도 있다. 획득과-함께-로드 동작을 수행하는 것은 제 2 프로세서의 캐시 라인들의 모두를 무효화하는 것을 포함한다.

[0108] 어떤 경우들에서는, 제 1 프로세서의 캐시 라인들의 공유된 서브 세트의 적어도 하나의 공유된 라인이 제 1 공

유성 (hareability) 속성에 의해 나타내어지고, 제 2 프로세서의 캐시 라인들의 공유된 서브 세트의 각각의 공유된 라인이 제 2 공유성 속성에 의해 나타내어진다.

[0109] 다른 경우에, 도 9의 방법은 추가로, 캐시 필 동작을 수행하는 것과, 캐시 필 동작의 수행에 응답하여, 페이지 테이블로부터 제 1 공유성 속성 또는 제 2 공유성 속성 중 하나를 읽는 것을 포함한다.

[0110] 또 다른 경우에, 도 9의 방법은 추가로, 컴파일러에 의해, 적어도 하나의 로드 명령 또는 저장 명령을 발행하는 것을 포함하는데, 적어도 하나의 로드 명령 또는 저장 명령은 제 1 공유성 속성 또는 제 2 공유성 속성 중 적어도 하나를 나타낸다.

[0111] 다른 예에서, 로드 명령은 공유된 로드 명령을 포함하고, 저장 명령은 공유된 저장 명령을 포함한다.

[0112] 또 다른 예에서, 그 방법은 추가로, 제 1 공유성 속성 및 제 2 공유성 속성 중 적어도 하나를 나타내는 복수의 주소 포인터들의 하나 이상의 비트들을 읽는 것을 포함한다.

[0113] 또 다른 예에서, 제 1 공유성 속성 및 제 2 공유성 속성 중 적어도 하나는, 제 1 캐시 및 제 2 캐시 중 적어도 하나가 비공유 상태에 있다는 것을 나타내고, 그 방법은 추가로, 제 1 캐시 및 제 2 캐시 중 적어도 하나에 대해 캐시 동작을 수행하는 것과, 그 캐시 동작의 수행에 응답하여, 제 1 캐시 및 제 2 캐시 중 적어도 하나가 공유 상태에 있다는 것을 나타내기 위해 제 1 공유성 속성 및 제 2 공유성 속성 중 적어도 하나를 변경하는 것을 포함한다.

[0114] 또 다른 예에서, 제 1 공유성 속성 및 제 2 공유성 속성 중 적어도 하나는, 제 1 캐시 및 제 2 캐시 중 적어도 하나가 공유 상태에 있다는 것을 나타내고, 그 방법은 추가로, 제 1 캐시 및 제 2 캐시 중 적어도 하나에 대해 캐시 동작을 수행하는 것과, 그 캐시 동작의 수행에 응답하여, 제 1 캐시 및 제 2 캐시 중 적어도 하나가 비공유 상태에 있다는 것을 나타내기 위해 제 1 공유성 속성 및 제 2 공유성 속성 중 적어도 하나를 변경하는 것을 포함한다.

[0115] 또 다른 예에서, 그 방법은 추가로, 제 1 캐시 및 제 2 캐시 중 적어도 하나의 주소 지역을 결정하기 위해 하나 이상의 레지스터에 저장된 주소와 제 1 캐시 및 제 2 캐시 중 적어도 하나의 주소를 비교하는 것과, 그 주소 지역이 공유되는 것인지 또는 비공유되는 것인지를 결정하는 것을 포함한다.

[0116] 또 다른 예에서, 그 방법은 추가로, 제 1 캐시 및 제 2 캐시 중 적어도 하나의 더티 바이트들을 결정하는 것, 캐시 퇴거 및 캐시 플러시 중 적어도 하나를 수행하는 것, 및 캐시 퇴거 및 캐시 플러시 중 적어도 하나 동안 더티 바이트들만을 캐시 또는 시스템 메모리에 쓰는 것을 포함한다.

[0117] 또 다른 예에서, 해제와-함께-저장 동작을 실행하기 위해, 제 1 프로세서는 인터럽트, 레지스터 읽기, 레지스터 쓰기, 및 프로그램가능 지연 중 적어도 하나를 수신하는 것에 응답하여 제 1 프로세서의 캐시를 플러시한다.

[0118] 또 다른 예에서, 해제와-함께-저장 동작을 실행하기 위해, 제 1 프로세서는 캐시 압력, 타임아웃, 및 명시적 소프트웨어 캐시 유지보수 중 적어도 하나에 응답하여 제 1 캐시를 플러시한다.

[0119] 또 다른 예에서, 해제와-함께-저장 동작을 실행하기 위해, 제 1 프로세서는 캐시 압력, 타임아웃, 및 명시적 소프트웨어 캐시 유지보수 중 적어도 하나에 응답하여 제 1 캐시를 플러시한다.

[0120] 또 다른 예에서, 제 1 캐시 및 제 2 캐시 중 적어도 하나는 가상적으로 태깅된다. 일부 예들에서, 해제와-함께-저장 동작을 실행하는 것은, 제 1 캐시에 저장된 가상 주소를 물리적 주소로 번역하는 것과, 물리적 주소에 기초하여 가상 주소를 물리적 주소로 변환한 후에 제 1 캐시의 캐시 라인들을 플러시하는 것을 포함한다.

[0121] 또 다른 예에서, 획득과-함께-로드 동작을 실행하는 것은, 제 2 캐시에 저장된 가상 주소를 물리적 주소로 번역하는 것과, 물리적 주소에 기초하여 가상 주소를 물리적 주소로 변환한 후에 제 2 캐시의 캐시 라인들을 무효화하는 것을 포함한다.

[0122] 예에 의존하여, 상이한 시퀀스로 수행될 수도 있는 본원에서 설명된 기법들 중 임의의 것의 특정한 액트들 또는 이벤트들이 부가되거나, 병합되거나, 또는 다 함께 제외될 수도 있다 (예컨대, 모든 설명된 액트들 또는 이벤트들이 그 기법들의 실용화에 필요한 것은 아니다) 는 것이 이해되어야 한다. 더구나, 특정한 예들에서, 액트들 또는 이벤트들은 순차적으로라기 보다는, 예컨대, 다중 스레드식 프로세싱, 인터럽트 프로세싱, 또는 다수의 프로세서들을 통하여 동시에 수행될 수도 있다.

[0123] 하나 이상의 예들에서, 설명된 기능들은 하드웨어, 소프트웨어, 펌웨어, 또는 그것들의 임의의 조합으로 구현될

수도 있다. 소프트웨어로 구현된다면, 기능들은 하나 이상의 명령들 또는 코드로서 컴퓨터 판독가능 매체 상에 저장될 수도 있다. 컴퓨터 판독가능 매체들은 컴퓨터 데이터 저장 매체들을 포함할 수도 있다. 데이터 저장 매체들은 본 개시물에서 설명된 기법들의 구현을 위한 명령들, 코드 및/또는 데이터 구조들을 취출하기 위해 하나 이상의 컴퓨터들 또는 하나 이상의 프로세서들에 의해 액세스될 수 있는 임의의 이용가능 매체들일 수도 있다. 비제한적인 예로서, 그런 컴퓨터 판독가능 매체들은 랜덤 액세스 메모리 (RAM), 판독 전용 메모리 (ROM), EEPROM, CD-ROM 또는 다른 광 디스크 스토리지, 자기 디스크 스토리지, 또는 다른 자기적 저장 디바이스들, 또는 소망의 프로그램 코드를 명령들 또는 데이터 구조들의 형태로 저장하는데 사용될 수 있고 컴퓨터에 의해 액세스될 수 있는 임의의 다른 매체를 포함할 수 있다. 디스크 (disk 및 disc) 는 본원에서 사용되는 바와 같이, 콤팩트 디스크 (compact disc, CD), 레이저 디스크, 광 디스크, 디지털 다용도 디스크 (DVD), 플로피 디스크 (floppy disk) 및 블루레이 디스크를 포함하는데, disk들은 보통 데이터를 자기적으로 재생하지만, disc들은 레이저들로써 광적으로 데이터를 재생한다. 상기한 것들의 조합들은 또한 컴퓨터 판독가능 매체들의 범위 내에 포함되어야 한다.

[0124] 코드는 하나 이상의 프로세서들, 이를테면 하나 이상의 디지털 신호 프로세서들 (DSP들), 범용 마이크로프로세서들, 주문형 집적회로들 (ASIC들), 필드 프로그램가능 로직 어레이들 (FPGA들), 또는 다른 동등한 집적 또는 개별 로직 회로에 의해 실행될 수도 있다. 따라서, 본원에서 사용되는 바와 같은 용어 "프로세서"는 앞서의 구조 또는 본원에서 설명된 기법들의 구현에 적합한 임의의 다른 구조 중 임의의 것을 말할 수도 있다. 또한, 본 기법들은 하나 이상의 회로들 또는 로직 엘리먼트들 내에 완전히 구현될 수 있다.

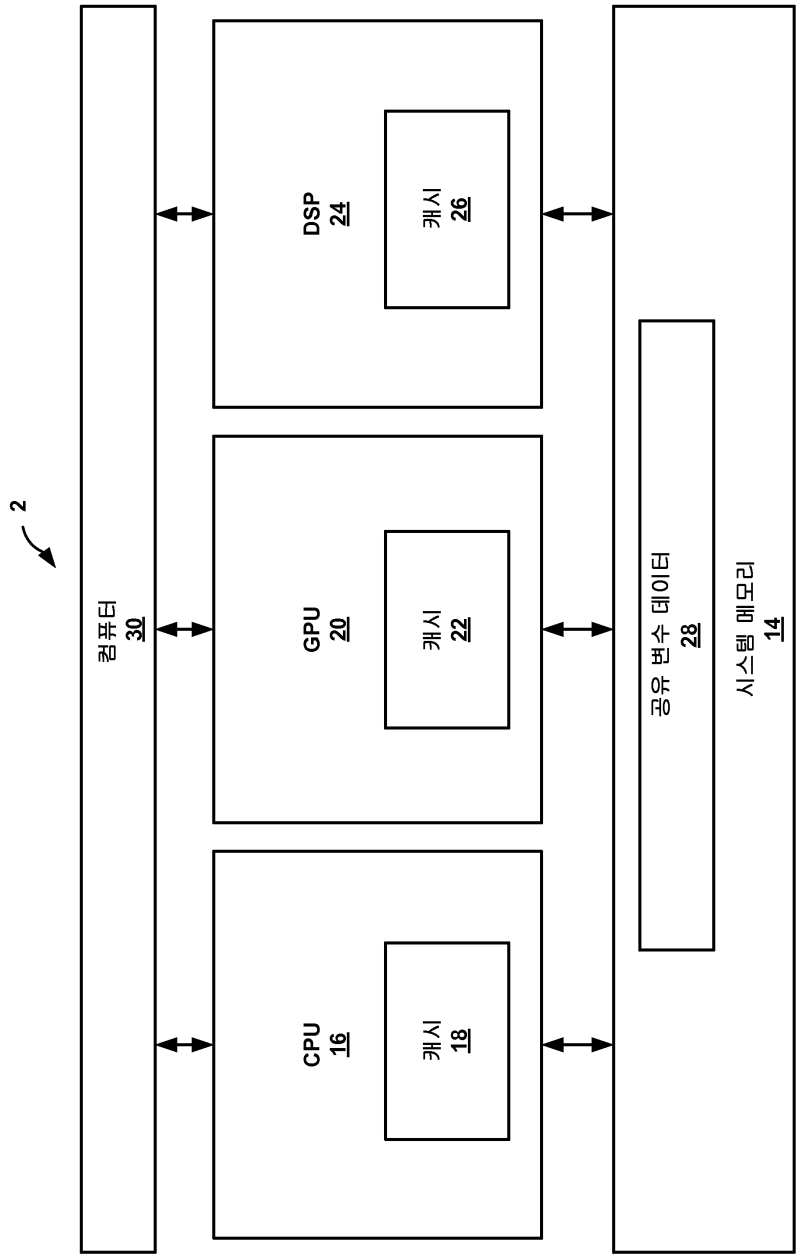
[0125] 본 개시물에 따른, 예컨대, 해제 일관성 메모리 순서화 모델을 사용하여 온-디맨드 캐시 코히어런시를 구현하는 기법들이 위에서 설명되어 있다. 부가물이 또한 본 개시물에 동반된다. 부가물에서 설명되는 기법들은 도 1에 예시된 유닛들 중 하나 이상, 예컨대, CPU (16), GPU (20), 및/또는 DSP (24) 상에서 구현될 수도 있다. 첨부의 부가물은 그것의 전체가 본 개시물에 통합되고, 본 개시물에 대한 부가물로서 간주될 수도 있다.

[0126] [0136] 본 개시물의 기법들은 무선 핸드셋, 집적 회로 (IC) 또는 IC들의 세트 (즉, 칩 셋) 를 포함하는 매우 다양한 디바이스들 또는 장치들로 구현될 수도 있다. 다양한 컴포넌트들, 모듈들, 또는 유닛들이 개시된 기법들을 수행하도록 구성된 디바이스들의 기능적 양태들을 강조하기 위해 본 개시물에서 설명되지만, 상이한 하드웨어 유닛들에 의한 실현을 반드시 요구하지는 않는다. 그보다는, 위에서 설명된 바와 같이, 다양한 유닛들이 하드웨어 유닛에 결합되거나 또는 적합한 소프트웨어 및/또는 펌웨어와 함께, 위에서 설명된 바와 같은 하나 이상의 프로세서들을 포함하는 상호운용적 하드웨어 유닛들의 컬렉션에 의해 제공될 수도 있다.

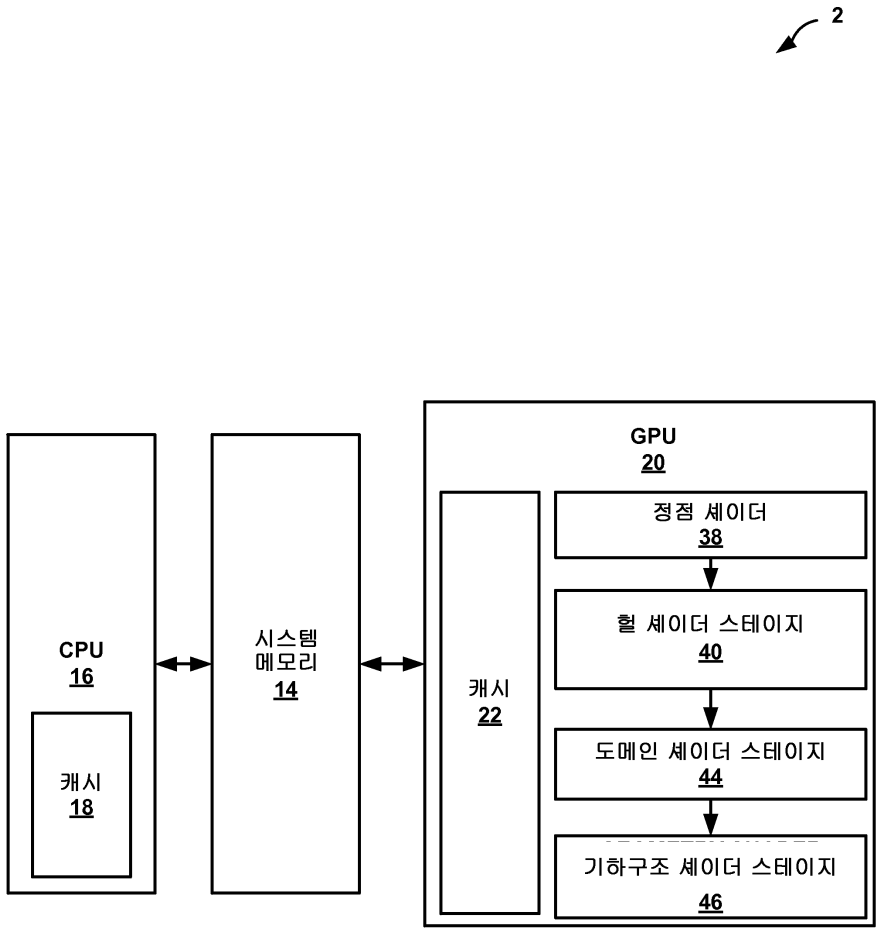
[0127] 다양한 예들이 설명되어 있다. 이들 및 다른 예들은 다음의 청구항들의 범위 내에 있다.

도면

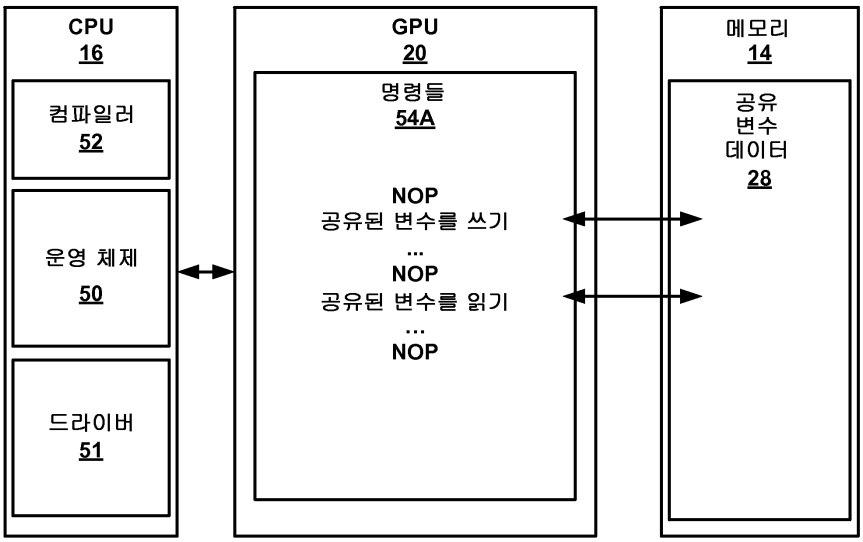
도면1


2
컴퓨터
30
CPU
16
캐시
18
GPU
20
캐시
22
DSP
24
캐시
26
공유 변수 데이터
28
시스템 메모리
14

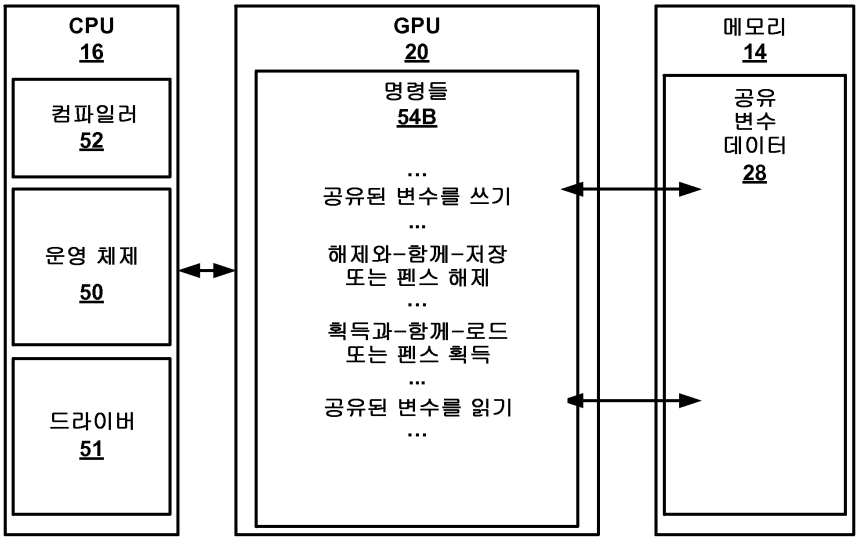
도면2


2
CPU
16
캐시
18
시스템
메모리
14
GPU
20
캐시
22
정점 셰이더
38
헐 셰이더 스테이지
40
도메인 셰이더 스테이지
44
기하구조 셰이더 스테이지
46

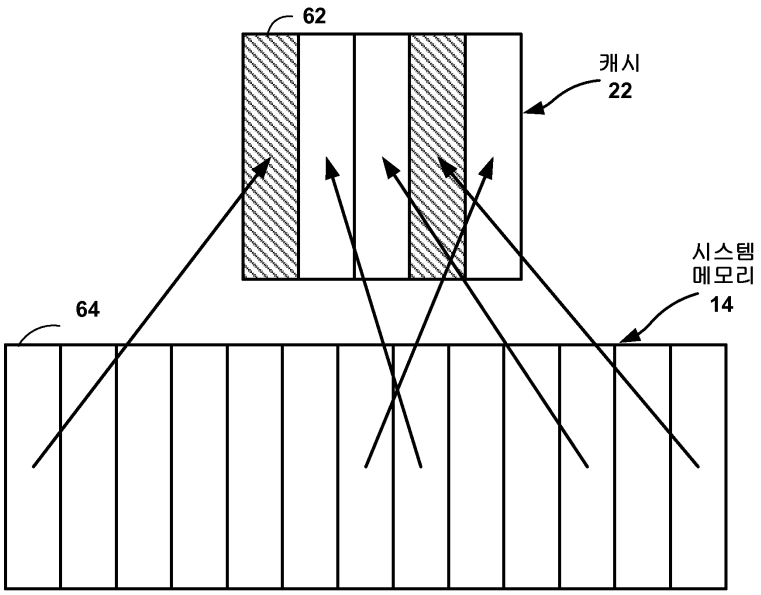
도면3a


CPU
16
컴파일러
52
운영 체제
50
드라이버
51
GPU
20
명령들
54A
NOP
공유된 변수를 쓰기
...
NOP
공유된 변수를 읽기
...
NOP
메모리
14
공유
변수
데이터
28

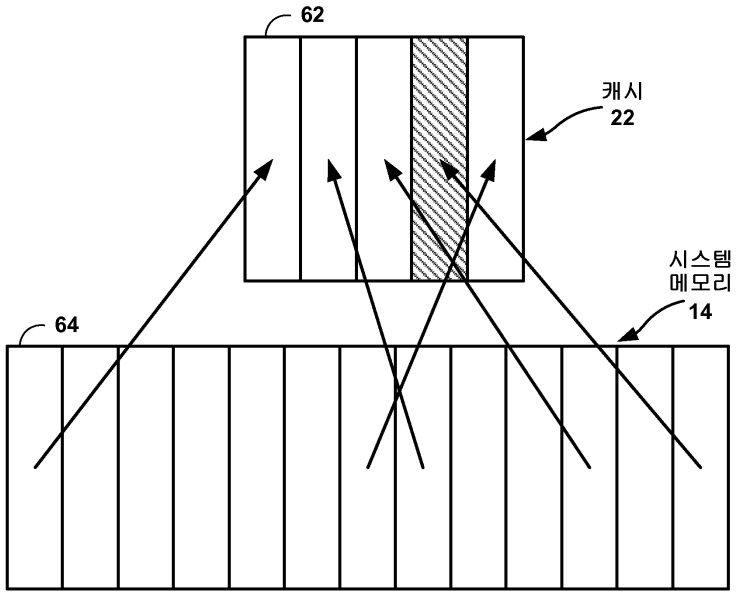
도면3b


CPU
16
컴파일러
52
운영 체제
50
드라이버
51
GPU
20
명령들
54B
...
공유된 변수를 쓰기
...
해제와-함께-저장
또는 펜스 해제
...
획득과-함께-로드
또는 펜스 획득
...
공유된 변수를 읽기
...
메모리
14
공유
변수
데이터
28

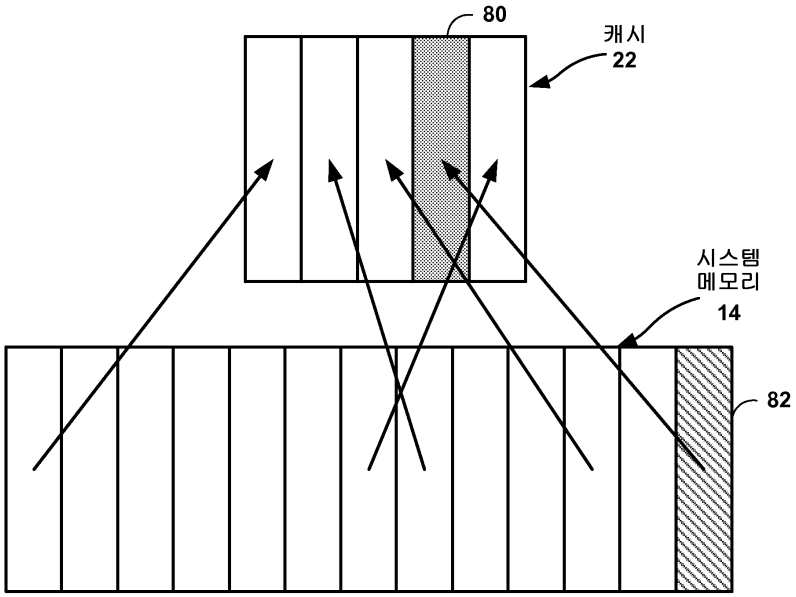
도면4a


62
캐시
22
시스템
메모리
14
64

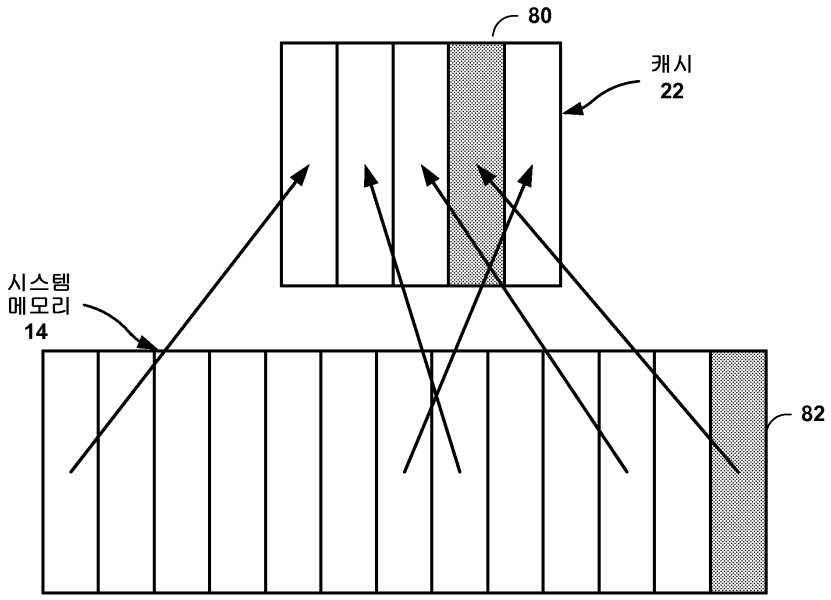
도면4b



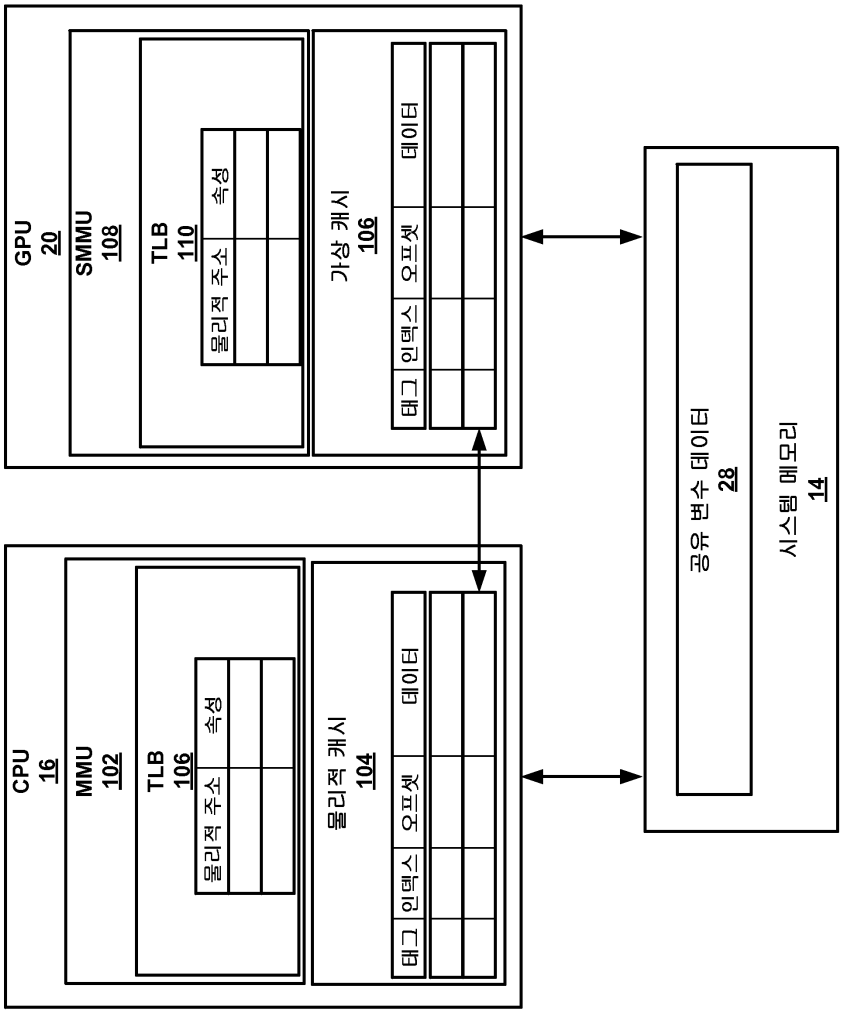
도면5a



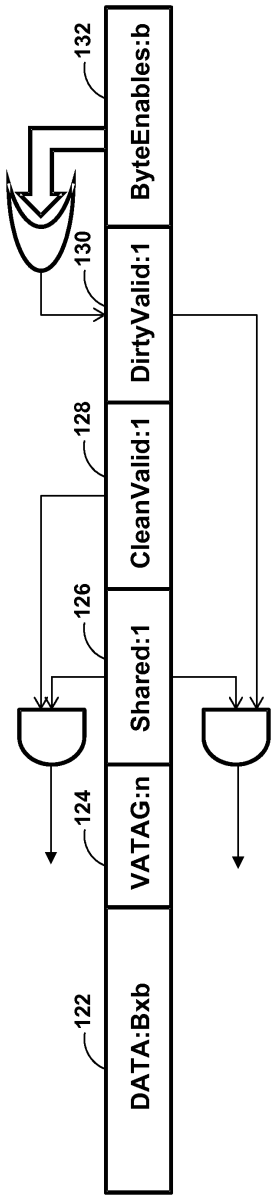
도면5b


80
캐시
22
시스템
메모리
14
82

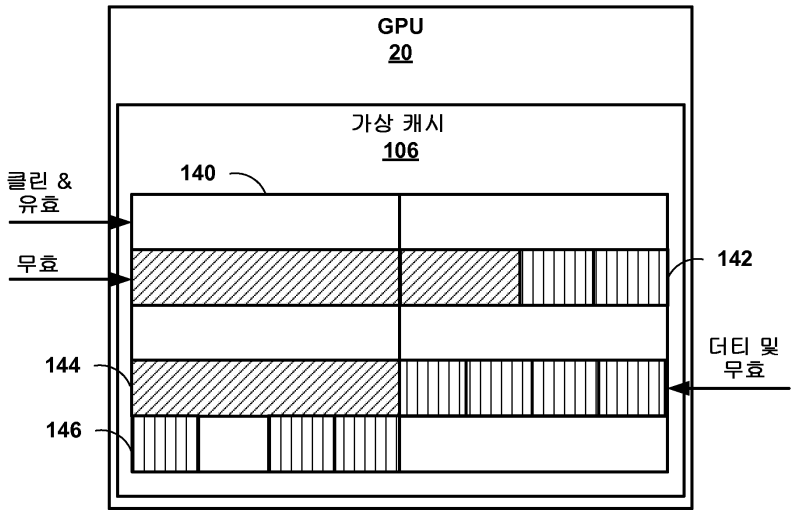
도면6


GPU
20
SMMU
108
TLB
110
물리적 주소
속성
가상 캐시
106
태그
인덱스
오프셋
데이터
CPU
16
MMU
102
TLB
106
물리적 주소
속성
물리적 캐시
104
태그
인덱스
오프셋
데이터
공유 변수 데이터
28
시스템 메모리
14

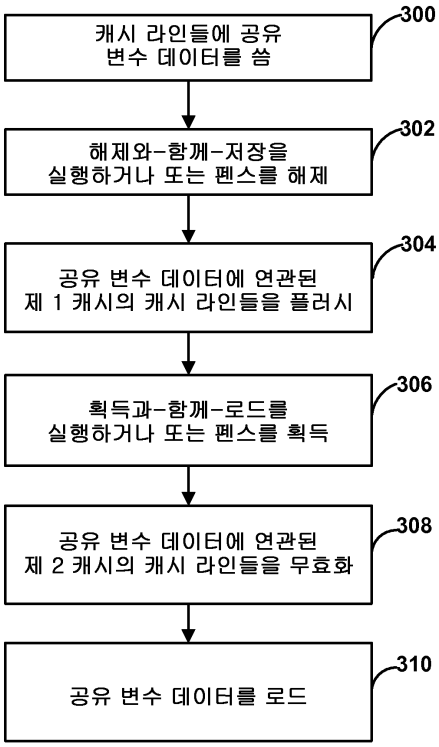
도면7


132
ByteEnables:b
130
DirtyValid:1
128
CleanValid:1
126
Shared:1
124
VATAG:n
122
DATA:Bxb

도면8



도면9



【심사관 직권보정사항】

【직권보정 1】

【보정항목】 발명(고안)의 설명

【보정세부항목】 식별번호 [0041]

【변경전】

소망의 프로그램 코드를 명령들 및/또는 데이터 구조들의 형태로 운반하거나 저장하는데 사용될 수 있고

【변경후】

소망의 프로그램 코드를 명령들 및/또는 데이터 구조들의 형태로 저장하는데 사용될 수 있고