(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2020년07월28일
(11) 등록번호 10-2138697
(24) 등록일자 2020년07월22일

(51) 국제특허분류(Int. Cl.)
G06F 12/0855 (2016.01) *G06F 12/0864* (2016.01)
G06F 12/0875 (2016.01) *G06F 12/0895* (2016.01)
G06F 12/128 (2016.01)
(52) CPC특허분류
G06F 12/0855 (2013.01)
G06F 12/0864 (2013.01)
(21) 출원번호 10-2017-7033433
(22) 출원일자(국제) 2016년04월08일
심사청구일자 2020년02월20일
(85) 번역문제출일자 2017년11월17일
(65) 공개번호 10-2018-0008507
(43) 공개일자 2018년01월24일
(86) 국제출원번호 PCT/US2016/026664
(87) 국제공개번호 WO 2016/186747
국제공개일자 2016년11월24일
(30) 우선권주장
14/716,947 2015년05월20일 미국(US)

(56) 선행기술조사문헌
US06122709 A*
岩田 英次, “可變構造型竝列計算機のキャッシュ・アーキテクチャ”, 情報處理學會論文誌, 제32권, 제6호, pp. 777-788, 1991년 6월.*
JP2012003314 A*
US20130007358 A1
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
퀄컴 인코포레이티드
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
(72) 발명자
펠리린, 3세, 헨리 아더
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
스파이어, 토마스 필립
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
(뒷면에 계속)
(74) 대리인
특허법인 남앤남

전체 청구항 수 : 총 15 항 심사관 : 남윤권

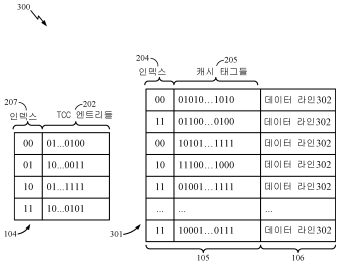
(54) 발명의 명칭 **캐시 태그 압축을 위한 방법 및 장치**

(57) 요 약

메모리 구조는 인덱싱된 태그 압축 구조를 사용하여 메모리 태그의 일부를 압축한다. 메모리 태그의 더 높은 차수 비트들의 세트는 인덱싱된 태그 압축 구조에 저장될 수 있으며, 여기서 더 높은 차수 비트들의 세트는 인덱스 값에 의해 식별된다. 태그 어레이는 메모리 태그의 더 낮은 차수 비트들의 세트 및 메모리 태그의 더 높은 차수

(뒷면에 계속)

대 표 도 - 도3



비트들의 세트를 저장하는 태그 압축 구조에서 엔트리를 식별하는 인덱스 값을 저장한다. 메모리 태그는 데이터 어레이에 저장된 데이터 엘리먼트의 메모리 어드레스의 적어도 일부를 포함할 수 있다.

(52) CPC특허분류
G06F 12/0875 (2013.01)
G06F 12/0895 (2013.01)
G06F 12/128 (2013.01)
G06F 2212/1016 (2013.01)
G06F 2212/401 (2013.01)
G06F 2212/6032 (2013.01)
G06F 2212/621 (2013.01)

(72) 발명자
사토리어스, 토마스 앤드류
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
모로, 미카엘 윌리엄
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
디펜더퍼, 제임스 노리스
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
닥서, 케네스 알란
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
매킬베인, 미카엘 스콧
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775

명 세 서

청구범위

청구항 1

캐시 메모리로서,

복수의 태그 압축 엔트리들을 저장하는 태그 압축 구조 - 상기 복수의 태그 압축 엔트리들 중 적어도 하나는 적어도 2 개의 메모리 태그들에 의해 공유되는 미리결정된 길이의 더 높은 차수 비트들의 고유 세트를 포함하고 각각의 인덱스 값에 의해 식별됨 - ; 및

복수의 태그 어레이 엔트리들을 저장하는 태그 어레이를 포함하고,

상기 복수의 태그 어레이 엔트리들 각각은 상기 메모리 태그들 중 하나의 미리결정된 길이의 더 낮은 차수 비트들의 각각의 세트 및 인덱스 값을 포함하고, 각각의 태그 어레이 엔트리의 인덱스 값은 상기 각각의 태그 어레이 엔트리의 메모리 태그의 더 높은 차수 비트들을 포함하는 태그 압축 엔트리의 인덱스 값에 대응하고, 복수의 메모리 태그들 각각은 데이터 어레이에 저장된 데이터 엘리먼트의 각각의 메모리 어드레스의 적어도 일부를 포함하는,

캐시 메모리.

청구항 2

제 1 항에 있어서,

제 1 태그 압축 엔트리의 메모리 태그의 더 높은 차수 비트들의 세트가 입력 메모리 어드레스의 더 높은 차수 비트들의 세트와 매칭하는 것으로 결정 시에, 상기 태그 압축 구조로부터 상기 제 1 태그 압축 엔트리에 대응하는 인덱스 값을 리턴하고;

상기 입력 메모리 어드레스의 더 낮은 차수 비트들의 세트가 제 1 태그 어레이 엔트리의 메모리 태그의 더 낮은 차수 비트들의 세트와 매칭하는 것으로 결정 시에, 상기 제 1 태그 어레이 엔트리에 저장된 메모리 태그의 더 낮은 차수 비트들의 세트 및 상기 인덱스 값을 리턴하고; 그리고

상기 제 1 태그 압축 엔트리의 리턴된 인덱스 값이 상기 제 1 태그 어레이 엔트리의 리턴된 인덱스 값과 매칭하는 것으로 결정함으로써,

상기 입력 메모리 어드레스에 대한 태그 어레이의 히트(hit)의 표시를 리턴하도록 구성된 로직을 더 포함하는,

캐시 메모리.

청구항 3

제 1 항에 있어서,

입력 메모리 어드레스의 더 높은 차수 비트들의 세트가 상기 복수의 태그 압축 엔트리들의 더 높은 차수 비트들의 세트들 중 어느 것과도 매칭하지 않는다는 것;

상기 입력 메모리 어드레스의 더 낮은 차수 비트들의 세트가 상기 복수의 태그 어레이 엔트리들에 저장된 더 낮은 차수 비트들의 세트들 중 어느 것과도 매칭하지 않는다는 것; 그리고

상기 입력 메모리 어드레스의 더 낮은 차수 비트들과 매칭하는 더 낮은 차수 비트들의 세트를 저장하는 태그 어레이 엔트리로부터 리턴된 인덱스 값이 상기 입력 메모리 어드레스의 더 높은 차수 비트들과 매칭하는 더 높은 차수 비트들의 세트를 저장하는 태그 압축 엔트리의 인덱스 값과 매칭하지 않는다는 것

중 적어도 하나의 결정 시에, 상기 입력 메모리 어드레스에 대한 태그 어레이의 미스(miss)의 표시를 리턴하도록 구성된 로직을 더 포함하는,

캐시 메모리.

청구항 4

제 3 항에 있어서,

상기 태그 어레이에서 상기 미스의 표시를 리턴하는 것에 응답하여,

메모리 계층의 다음 레벨에 저장된 데이터를 리트리브하고;

상기 데이터 어레이의 제 1 엔트리에 상기 데이터를 저장하고;

상기 태그 압축 구조의 제 2 태그 압축 엔트리에 상기 입력 메모리 어드레스의 더 높은 차수 비트들의 세트를 저장하고; 그리고

상기 태그 어레이의 제 2 태그 어레이 엔트리에, (i) 상기 입력 메모리 어드레스의 더 낮은 차수 비트들의 세트, 및 (ii) 제 2 태그 압축 엔트리에 대응하는 인덱스 값을 저장하도록 구성된 로직을 더 포함하고,

상기 제 2 태그 어레이 엔트리는 상기 데이터 어레이의 제 1 엔트리에 대응하는,

캐시 메모리.

청구항 5

제 1 항에 있어서,

상기 태그 압축 구조의 엔트리들은 추가로, (i) 프로세스의 ASID(address space identifier), (ii) 보안 비트들의 세트, 및 (iii) 예외 레벨 비트들의 세트 중 하나 또는 그 초과를 저장하도록 구성되는,

캐시 메모리.

청구항 6

제 1 항에 있어서,

제 1 태그 어레이 엔트리 및 제 2 태그 어레이 엔트리는 제 1 태그 압축 엔트리에 대응하는 인덱스 값을 각각 특정하며, 상기 제 2 태그 어레이 엔트리의 더 낮은 차수 비트들의 세트는 상기 제 1 태그 어레이 엔트리의 더 낮은 차수 비트들의 세트와 매칭하지 않는,

캐시 메모리.

청구항 7

제 1 항에 있어서,

상기 태그 압축 구조는 완전 연관 콘텐츠 어드레싱 가능한 메모리(fully associative content addressable memory)를 포함하며, 상기 태그 어레이는 (i) 데이터 캐시, (ii) 변환 룩어사이드 버퍼(translation lookaside buffer), (iii) 라인 필 버퍼(line fill buffer) 및 (iv) 명령 캐시 중 적어도 하나의 부분을 포함하는,

캐시 메모리.

청구항 8

제 1 항에 있어서,

상기 캐시 메모리는 집적 회로에 배치되는,

캐시 메모리.

청구항 9

방법으로서,

태그 압축 구조의 복수의 태그 압축 엔트리들의 제 1 태그 압축 엔트리에, 제 1 메모리 태그 및 제 2 메모리 태그에 의해 공유되는 미리결정된 길이의 더 높은 차수 비트들의 세트를 저장하는 단계 – 상기 복수의 태그 압축 엔트리들 각각은 각각의 인덱스 값에 의해 식별됨 – ; 및

태그 어레이의 복수의 태그 어레이 엔트리들의 제 1 태그 어레이 엔트리에, 상기 제 1 메모리 태그의 미리결정된 길이의 더 낮은 차수 비트들의 세트 및 상기 제 1 태그 어레이 엔트리에 대한 인덱스 값을 저장하는 단계를 포함하고,

상기 제 1 태그 어레이 엔트리의 인덱스 값은 상기 제 1 태그 압축 엔트리의 인덱스 값에 대응하고, 상기 제 1 메모리 태그는 데이터 어레이에 저장된 데이터 엘리먼트의 메모리 어드레스의 적어도 일부를 포함하고, 상기 복수의 태그 어레이 엔트리들의 제 2 태그 어레이 엔트리는 상기 제 1 태그 압축 엔트리에 대응하는 인덱스 값을 특정하고 상기 제 2 메모리 태그의 더 낮은 차수 비트들의 세트를 저장하고, 상기 제 2 메모리 태그의 더 낮은 차수 비트들의 세트는 상기 제 1 메모리 태그의 더 낮은 차수 비트들의 세트와 매칭하지 않는,

방법.

청구항 10

제 9 항에 있어서,

제 1 태그 압축 엔트리의 메모리 태그의 더 높은 차수 비트들의 세트가 입력 메모리 어드레스의 더 높은 차수 비트들의 세트와 매칭하는 것으로 결정 시에, 상기 태그 압축 구조로부터 상기 제 1 태그 압축 엔트리에 대응하는 인덱스 값을 리턴하고;

상기 입력 메모리 어드레스의 더 낮은 차수 비트들의 세트가 상기 제 1 태그 어레이 엔트리의 메모리 태그의 더 낮은 차수 비트들의 세트와 매칭하는 것으로 결정 시에, 상기 제 1 태그 어레이 엔트리에 저장된 메모리 태그의 더 낮은 차수 비트들의 세트 및 상기 인덱스 값을 리턴하고; 그리고

상기 제 1 태그 압축 엔트리의 리턴된 인덱스 값이 상기 제 1 태그 어레이 엔트리의 리턴된 인덱스 값과 매칭하는 것으로 결정함으로써,

상기 입력 메모리 어드레스에 대한 태그 어레이의 히트의 표시를 리턴하는 단계를 더 포함하는,

방법.

청구항 11

제 9 항에 있어서,

상기 태그 압축 구조의 복수의 태그 압축 엔트리들 각각은 각각의 복수의 메모리 태그들에 의해 공유되는 더 높은 차수 비트들의 고유 세트를 저장하도록 구성되고, 상기 복수의 태그 어레이 엔트리들 각각은 각각의 메모리 태그의 더 낮은 차수 비트들의 세트 및 인덱스 값을 저장하도록 구성되고, 각각의 태그 어레이 엔트리의 인덱스 값은 상기 각각의 태그 어레이 엔트리의 메모리 태그의 더 높은 차수 비트들을 포함하는 태그 압축 엔트리의 인덱스 값에 대응하고,

상기 방법은,

입력 메모리 어드레스의 더 높은 차수 비트들의 세트가 상기 복수의 태그 압축 엔트리들의 더 높은 차수 비트들의 세트들 중 어느 것과도 매칭하지 않는다는 것;

상기 입력 메모리 어드레스의 더 낮은 차수 비트들의 세트가 상기 복수의 태그 어레이 엔트리들에 저장된 더 낮은 차수 비트들의 세트들 중 어느 것과도 매칭하지 않는다는 것; 및

상기 입력 메모리 어드레스의 더 낮은 차수 비트들과 매칭하는 더 낮은 차수 비트들의 세트를 저장하는 태그 어레이 엔트리로부터 리턴된 인덱스 값이 상기 입력 메모리 어드레스의 더 높은 차수 비트들과 매칭하는 더 높은 차수 비트들의 세트를 저장하는 태그 압축 엔트리의 인덱스 값과 매칭하지 않는다는 것

중 적어도 하나의 결정 시에, 상기 입력 메모리 어드레스에 대한 태그 어레이의 미스의 표시를 리턴하는 단계를 더 포함하는,

방법.

청구항 12

제 11 항에 있어서,

상기 태그 어레이에서 상기 미스의 표시를 리턴하는 것에 응답하여,

메모리 계층의 다음 레벨에 저장된 데이터를 리트리브하는 단계;

상기 데이터 어레이의 제 1 엔트리에 상기 데이터를 저장하는 단계;

상기 태그 압축 구조의 제 2 태그 압축 엔트리에 상기 입력 메모리 어드레스의 더 높은 차수 비트들의 세트를 저장하는 단계; 및

상기 태그 어레이의 제 2 태그 어레이 엔트리에, (i) 상기 입력 메모리 어드레스의 더 낮은 차수 비트들의 세트, 및 (ii) 제 2 태그 압축 엔트리에 대응하는 인덱스 값을 저장하는 단계를 더 포함하는,

상기 제 2 태그 어레이 엔트리는 상기 데이터 어레이의 제 1 엔트리에 대응하는,

방법.

청구항 13

제 9 항에 있어서,

상기 태그 압축 구조의 엔트리들은 추가로, (i) 프로세스의 ASID(address space identifier), (ii) 보안 비트들의 세트, 및 (iii) 예외 레벨 비트들의 세트 중 하나 또는 그 초과를 저장하도록 구성되는,

방법.

청구항 14

제 9 항에 있어서,

상기 태그 압축 구조는 완전 연관 콘텐츠 어드레싱 가능한 메모리를 포함하며, 상기 태그 어레이는 (i) 데이터 캐시, (ii) 변환 룩어사이드 버퍼, (iii) 라인 필 버퍼 및 (iv) 명령 캐시 중 적어도 하나의 부분을 포함하는,

방법.

청구항 15

프로세서에 의해 실행되는 경우, 상기 프로세서로 하여금 제 9 항 내지 제 14 항 중 어느 한 항의 방법 단계들을 수행하게끔 하는 명령들을 저장하는 비-일시적인 컴퓨터-판독 가능한 저장 매체.

청구항 16

삭제

청구항 17

삭제

청구항 18

삭제

청구항 19

삭제

청구항 20

삭제

청구항 21

삭제

청구항 22

삭제

청구항 23

삭제

발명의 설명

기 술 분 야

[0001] [0001] 본 출원은 2015년 5월 20일 출원된 미국 특허 출원 일련번호 제14/716,947호를 우선권으로 주장하며, 이 출원은 본원의 양수인에게 양도되었으며 그리하여 인용에 의해 본원에 명시적으로 포함된다.

[0002] [0002] 본원에서 개시되는 양상들은 컴퓨터 마이크로프로세서들(본원에서 프로세서들로서 또한 지칭됨)의 분야와 관련된다. 보다 구체적으로, 본원에서 개시되는 양상들은 캐시 태그 압축(cache tag compression)과 관련된다.

배 경 기 술

[0003] [0003] 현대 프로세서들은 통상적으로 프로세싱 성능을 개선하기 위해 캐시들에 의존한다. 캐시는 워크로드(workload)의 명령 스트림들 및 데이터 스트림들에서 시간적 및 공간적 지역성을 이용함으로써 작동한다. 캐시의 일부는 캐시 태그 어레이들을 저장하는데 전용된다. 캐시 태그들은 메인 메모리로부터 가져온 실제 데이터의 어드레스를 저장한다. 캐시에서 히트(hit) 또는 미스(miss)를 결정하는 속도는 태그 어레이의 폭에 의해 제한되며, 여기서 더 큰 태그 어레이들은 통상적으로 더 많은 타이밍 압력을 생성한다. 더 넓은(또는 더 큰) 태그는 더 많은 비트들을 갖는다. 캐시에서 히트가 존재하는지를 결정하기 위해, 태그의 모든 비트들이 프로브 어드레스와 비교되어야 한다. 따라서, 비교 시에 더 많은 비트들은 비교 결과를 리턴하는데 더 많은 지연을 요구할 수 있다.

발명의 내용

[0004] [0004] 본원에서 개시되는 양상들은 태그 압축 캐시에 더 높은 차수 태그 비트들의 고유 값들을 저장함으로써 태그 저장을 위해 필요한 영역을 감소시킨다.

[0005] [0005] 일 양상에서, 캐시 메모리는 태그 압축 구조 및 태그 어레이를 포함한다. 태그 압축 구조는 복수의 태그 압축 엔트리들을 저장한다. 복수의 태그 압축 엔트리들 각각은 메모리 태그의 더 높은 차수 비트들의 각각의 세트를 포함하고 각각의 인덱스 값에 의해 식별된다. 태그 어레이는 복수의 태그 어레이 엔트리들을 저장한다. 복수의 태그 어레이 엔트리들 각각은 메모리 태그의 더 낮은 차수 비트들의 각각의 세트 및 태그 압축 구조의 인덱스 값들 중 하나에 대응하는 인덱스 값을 포함한다. 제 1 메모리 태그는 데이터 어레이에 저장된 데이터 엘리먼트의 메모리 어드레스의 적어도 일부를 포함할 수 있다.

[0006] [0006] 일 양상에서, 방법은, 태그 압축 구조 내의 복수의 태그 압축 엔트리들의 제 1 태그 압축 엔트리에, 제 1 메모리 태그의 더 높은 차수 비트들의 세트를 저장하는 단계를 포함한다. 복수의 태그 압축 엔트리들 각각은 각각의 인덱스 값에 의해 식별된다. 방법은 태그 어레이의 복수의 태그 어레이 엔트리들의 제 1 태그 어레이 엔트리에, 메모리 태그의 더 낮은 차수 비트들의 세트 및 제 1 태그 어레이 엔트리에 대한 인덱스 값을 저장하는 단계를 더 포함한다. 제 1 태그 어레이 엔트리의 인덱스 값은 태그 압축 구조의 제 1 태그 압축 엔트리에 대응한다. 제 1 메모리 태그는 데이터 어레이에 저장된 데이터 엘리먼트의 메모리 어드레스의 적어도 일부를 포함할 수 있다.

[0007] [0007] 일 양상에서, 비-일시적인 컴퓨터-판독 가능한 매체는 프로세서에 의해 실행될 때, 프로세서로 하여금, 태그 압축 구조 내의 복수의 태그 압축 엔트리들의 제 1 태그 압축 엔트리에, 제 1 메모리 태그의 더 높은 차수 비트들의 세트를 저장하는 단계를 포함하는 동작을 수행하게 하는 명령들을 저장한다. 복수의 태그 압축 엔트리들 각각은 각각의 인덱스 값에 의해 식별된다. 동작은 태그 어레이의 복수의 태그 어레이 엔트리들의 제 1 태그 어레이 엔트리에, 메모리 태그의 더 낮은 차수 비트들의 세트 및 제 1 태그 어레이 엔트리에 대한 인덱스 값을 저장하는 단계를 더 포함한다. 제 1 태그 어레이 엔트리의 인덱스 값은 태그 압축 구조의 제 1 태그 압축 엔트리에 대응한다. 제 1 메모리 태그는 데이터 어레이에 저장된 데이터 엘리먼트의 메모리 어드레스의 적어도 일부를 포함할 수 있다.

[0008] [0008] 일 양상에서, 장치는, 복수의 태그 압축 엔트리들의 제 1 태그 압축 엔트리에, 제 1 메모리 태그의 더 높은 차수 비트들의 세트를 저장하기 위한 수단을 포함한다. 복수의 태그 압축 엔트리들 각각은 각각의 인덱스 값에 의해 식별된다. 장치는, 복수의 태그 어레이 엔트리들의 제 1 태그 어레이 엔트리에, 메모리 태그의 더 낮은 차수 비트들의 세트 및 제 1 태그 어레이 엔트리에 대한 인덱스 값을 저장하기 위한 수단을 더 포함한다. 제 1 태그 어레이 엔트리의 인덱스 값은 제 1 태그 압축 엔트리에 대응한다. 제 1 메모리 태그는 데이터 엘리먼트의 메모리 어드레스의 적어도 일부를 포함할 수 있다.

[0009] [0009] 태그 압축 캐시의 메모리 태그의 일부를 저장함으로써, 본원에서 개시되는 양상들은 태그 어레이에 중복(redundant) 비트의 저장을 감소시킨다. 이렇게 하면 메모리 태그들에 대해 더 적은 물리적인 영역을 요구하고, 더 빠른 비교 결과들을 제공하고, 비교를 수행하는데 더 적은 전력을 요구한다.

도면의 간단한 설명

[0010] [0010] 위에서 인용된 양상들이 달성되고 상세히 이해될 수 있는 방식으로, 위에서 간략하게 요약된 본 개시의 양상들에 대한 보다 구체적인 설명이 첨부 도면들을 참조하여 이루어질 수 있다.

[0011] [0011] 그러나 본 개시가 다른 양상들에 대해 허용될 수 있기 때문에, 첨부된 도면들은 본 개시의 양상들만을 예시하고, 따라서, 본 개시의 범위에 대한 제한으로 고려되어서는 안 된다는 것에 주의해야 한다.

[0012] [0012] 도 1은 일 양상에 따라, 캐시 태그 압축을 제공하는 프로세서를 예시한다.

[0013] [0013] 도 2는 일 양상에 따라, 캐시 태그 압축을 제공하도록 구성된 프로세서의 컴포넌트들의 로직도이다.

[0014] [0014] 도 3은 일 양상에 따라, 태그 압축 캐시, 태그 어레이 및 데이터 어레이에 저장된 예시적인 값들을 예시하는 개략도이다.

[0015] [0015] 도 4는 일 양상에 따라, 캐시 태그 압축을 제공하기 위한 방법을 예시하는 흐름도이다.

[0016] [0016] 도 5는 일 양상에 따라, 새로운 태그 압축 캐시 엔트리를 설정하기 위한 방법을 예시하는 흐름도이다.

[0017] [0017] 도 6은 일 양상에 따라, 캐시 태그 압축을 제공하도록 구성된 프로세서를 통합하는 컴퓨팅 디바이스를 예시하는 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0011] [0018] 본원에서 개시되는 양상들은 메모리 태그 저장을 위한 영역을 감소시키는 기술들을 제공한다. 예를 들어, 캐시들은 상위 어드레스 비트들의 임의의 세트로 태깅될 수 있다. 일반적으로, 캐시 태그들은 매우 많은 비트들에 대한 저장을 필요로 하며, 현대의 캐시 아키텍처들은 통상적으로 이전의 세대들보다 상당히 큰 태그들을 필요로 한다. 예를 들어, 64 비트 라인들을 가진 가상(hypothetical) 4-웨이 32 KB(kilobyte) 명령 캐시를 고려한다. 이러한 캐시의 라인들의 수는 $2^{15}/2^{9}$ 또는 512 라인들이다. 각각의 라인은 태그와 연관된다. 캐시의 세트들의 수는 $2^{15}/2^{6}/2^{2}$ 또는 128 세트들(7 비트 세트 디코드의 경우)과 동일하다. 적어도 하나의 양상에서, 태그 비트들은 7-비트 세트 디코드 또는 6-비트 바이트-오프셋을 포함하지 않고, 이는 태그 비트들이 A[7+6] 또는 A[13]에서 시작한다는 것을 의미하며, 여기서 "A"는 태그 검색에 대한 어드레스 입력을 표시한다. 따라서, 64 비트의 예시적인 어드레스에 대해, 태그는 A[63:13]와 비교되고 태그 저장은 라인 당 51 비트이다. 따라서, 캐시에 대한 총 태그 저장은 51 비트/라인 * 512 라인 또는 26,112 비트이며, 이는 캐시 태그들을 저장하기 위해 10 %의 저장 오버헤드를 초래한다.

[0012] [0019] 태그 값들은 임의적이고 코드-의존적이지만, 태그 값들 각각의 비트들의 서브세트는 거의 변동성을 갖지 않을 수 있어서, 비트들의 서브세트는 다수의 캐시 태그들에 걸쳐 동일한 값들을 갖는다. 본원에서 개시되는 양상들은 다수의 캐시 태그들에 걸쳐 동일한 값의 다수의 카피들의 중복 저장을 제거함으로써 이러한 경향을 이용한다. 특히, 본원에서 개시되는 양상들은 고유 태그 값들을 저장하도록 구성된 태그 압축 캐시를 도입한다. 이렇게 하면 캐시 태그로부터 다수의 비트들이 제거된다. 캐시 태그로부터 제거된 비트들은 태그 압축 캐시 엔트리를 가리키는 보다 적은 수의 비트들로 대체될 수 있다. 태그 압축 캐시는 상위(또는 더 높은) 어드레스 비트, ASID(address space identifier)들, 보안 비트들, 예외 레벨 비트들 등을 (제한 없이) 포함하는 캐시 태그로부터의 비트들의 임의의 세트를 보유할 수 있다.

[0013] [0020] 태그 압축 캐시는 오리지날 캐시 태그의 일부를 보유하기 때문에, 액세스가 캐시에서 "히트"하기 위해

태그 압축 캐시에서 "히트"해야 한다. 유사하게, 태그 압축 캐시에서 "미스"하는 액세스는 캐시 그 자체에서 또한 "미스"해야 한다. 태그 압축 캐시 엔트리를 축출(evicting)하는 것은 모든 대응하는 캐시 라인들(즉, 이 라인들은 축출되는 태그 압축 캐시의 인덱스와 매칭하는 태그 압축 캐시 인덱스 값을 가짐)의 무효화를 필요로 한다. 그러나 캐시 라인을 무효화하는 것은 태그 압축 캐시에 대한 어떠한 업데이트도 필요로 하지 않는다.

[0014] [0021] 본원에서 개시되는 양상들은 태그 압축 캐시 및 캐시를 병렬로 액세스하여 입력 메모리 어드레스가 캐시 "히트" 또는 "미스"를 초래하는지를 결정한다. 특히, 본원에서 개시되는 양상들은 입력 메모리 어드레스의 더 높은 차수 비트들(higher order bits)의 세트를 사용하여 태그 압축 캐시에 액세스한다. 태그 압축 캐시에 히트가 존재하는 경우(즉, 태그 압축 캐시의 엔트리가 입력 메모리 어드레스의 더 높은 차수 비트들과 매칭함), 태그 압축 캐시는 매칭하는 엔트리의 인덱스 값을 리턴하도록 구성된다. 동시에, 본원에서 개시되는 양상들은 입력 메모리 어드레스의 더 낮은 차수 비트들(lower order bits)의 세트를 사용하여 캐시를 검색한다. 캐시에서 히트가 존재하는 경우(즉, 캐시 태그 어레이의 태그 엔트리가 입력 메모리 어드레스의 더 낮은 차수 비트들과 매칭함), 캐시는 매칭하는 엔트리의 인덱스 값 및 더 낮은 차수 비트들을 리턴하도록 구성된다. 그 후, 본원에서 개시되는 양상들은 캐시 및 태그 압축 캐시로부터 리턴된 인덱스 값들을 비교할 수 있다. 인덱스 값이 매칭하지 않는 경우, 본원에서 개시되는 양상들은 캐시 미스의 표시를 리턴한다. 본원에서 개시되는 양상들은 추가로, 캐시 태그 엔트리로부터 리턴된 더 낮은 비트들을 입력 메모리 어드레스의 더 낮은 비트들과 비교할 수 있다. 더 낮은 비트들이 매칭하지 않는 경우, 본원에서 개시되는 양상들은 캐시 미스의 표시를 리턴한다. 그러나 인덱스 값들이 매칭하고 더 낮은 비트들이 매칭하는 경우, 본원에서 개시되는 양상들은 캐시 히트의 표시를 리턴한다(즉, 요청된 데이터가 캐시에 존재함).

[0015] [0022] 캐시는 본 개시의 양상들의 설명을 용이하게 하기 위해 참조 예로서 본원에서 사용된다. 그러나 본원에서 설명되는 기술들은 프로세서 캐시들, 데이터 캐시들, 명령 캐시들, 라인 필 버퍼들, TLB(Translation Lookaside Buffer)들 등과 같이 태그 필드들을 갖는 다른 하드웨어 구조에 적용되므로, 캐시의 사용이 본 개시를 제한하는 것으로 간주해서는 안 된다. 특정 하드웨어 구조에 대한 임의의 참조는 본 개시를 제한하는 것으로 간주되어서는 안 된다.

[0016] [0023] 도 1은 일 양상에 따라, 캐시 태그 압축을 제공하는 프로세서(101)를 예시한다. 도시된 바와 같이, 프로세서(101)는 명령들을 실행하는 명령 실행 파이프라인(112)을 포함한다. 파이프라인(112)은 각각이 다양한 비-구조화된 레지스터들(도시되지 않음) 및 하나 또는 그 초과의 산술 로직 유닛들(또한 도시되지 않음)을 포함하는 다수의 병렬 파이프라인들을 갖는 수퍼스칼라 설계일 수 있다. 도시된 바와 같이, 프로세서(101)는 또한 하나 또는 그 초과의 상위 레벨들의 메모리(108)로부터의 데이터의 라인들을 저장하는 캐시(102)(캐시 메모리(102)로서 또한 지칭됨)를 포함한다. 상위 레벨들의 메모리(108)는 상위 레벨들의 캐시들 및/또는 메인(시스템) 메모리를 (제한 없이) 포함할 수 있다. 적어도 하나의 양상에서, 캐시(102)는 레벨 1(L1) 데이터 캐시이다. 일반적으로, CPU(101)는 다수의 변동들을 포함할 수 있고, 도 2에 도시된 CPU(101)는 예시 목적을 위한 것이며, 본 개시의 제한으로 간주되어서는 안 된다. 예를 들어, CPU(101)는 그래픽 프로세싱 유닛(GPU)일 수 있다.

[0017] [0024] 도시된 바와 같이, 캐시(102)는 캐시 로직(103), 태그 압축 캐시(104), 태그 어레이(105) 및 데이터 어레이(106)를 포함한다. 캐시 로직(103)은 일반적으로, 캐시 히트들 또는 미스들이 특정 동작에서 발생하는지를 결정하는 것과 같은 캐시(102)의 동작을 제어한다. 태그 압축 캐시(TCC)(104)는 메모리 태그들의 상위(또는 더 높은) 차수 비트들의 세트를 보유하도록 구성된 하드웨어 구조이다. 예를 들어, TCC (104)는 메모리 태그의 상위 어드레스 비트들, ASID 비트들, 보안 비트들, 예외 레벨 비트들 등 중 하나 또는 그 초과를 보유할 수 있다. TCC(104)는 일반적으로, 태그 어레이(105)에 중복적으로 달리 저장되었을 중복 값들을 저장하도록 구성된다. 태그 어레이(105)는 일반적으로 캐시(102)에 저장된 데이터의 어드레스를 저장하도록 구성된다. 보다 구체적으로, 태그 어레이(105)의 각각의 엔트리는 TCC(104)의 엔트리의 인덱스 값에 대응하는 인덱스 값 및 메모리 태그들의 더 낮은 차수 비트들의 세트를 저장하도록 구성된다. 데이터 어레이(106)는 캐시 라인들의 데이터를 저장한다. 적어도 하나의 양상에서, TCC(104)는 CAM(content addressable memory) 구조로서 구현된다. 유사하게, 태그 어레이(105) 및 데이터 어레이(106)는 CAM으로서 구현될 수 있다. 또한, 적어도 하나의 양상에서, TCC(104)는 완전-연관(fully-associative)의 LRU(least recently used) 대체 구조이다. TCC(104)는 2개, 4개 또는 8개와 같은 임의의 수의 엔트리들을 가질 수 있다. TCC(104), 태그 어레이(105) 및 데이터 어레이(106)에 데이터를 저장하기 위한 수단은 하나 또는 그 초과의 메모리 셀들을 포함할 수 있다.

[0018] [0025] 동작 시에, 프로세서(101)는 상위 레벨들의 메모리(108) 중 하나에 로케이팅된 데이터가 캐시(102) 내에 존재하는지를 결정하도록 시도할 수 있다. 프로세서(101)가 캐시(102)에 입력 메모리 어드레스(가상 어드레

스일 수 있음)를 제공하면, 캐시 로직(103)은 입력 메모리 어드레스의 데이터가 캐시(102)에 있는지를 결정할 수 있다. 이렇게 하기 위해, 캐시 로직(103)은 입력 메모리 어드레스의 더 높은 차수 비트들을 이용하여 TCC(104)의 CAM 검색을 개시할 수 있다. TCC(104)의 엔트리가 입력 메모리 어드레스의 더 높은 차수 비트들과 매칭하는 경우, TCC(104)는 TCC(104)에서 매칭하는 엔트리의 인덱스 값을 리턴하도록 구성된다. 캐시 로직(103)은 TCC(104)의 CAM 검색과 병렬로 태그 어레이(105)의 CAM 검색을 개시할 수 있다. 캐시 로직(103)은 입력 메모리 어드레스의 더 낮은 차수 비트들을 이용하여 태그 어레이(105)를 검색할 수 있다. 태그 어레이(105)의 엔트리의 더 낮은 차수 비트들이 입력 메모리 어드레스의 더 낮은 차수 비트들과 매칭하는 경우, 태그 어레이(105)는 매칭하는 엔트리의 인덱스 값 및 더 낮은 차수 비트들을 리턴하도록 구성된다. 그 후, 캐시 로직(103)은 TCC(104) 및 태그 어레이(105)에 의해 리턴된 인덱스 값을 비교할 수 있다. 인덱스 값이 매칭하지 않는 경우, 캐시 로직(103)은 캐시 미스의 표시를 리턴할 수 있다. 캐시 로직(103)은 또한 태그 어레이(105)에 의해 리턴된 더 낮은 차수 비트들을 입력 메모리 어드레스의 더 낮은 차수 비트들과 비교할 수 있다. 더 낮은 차수 비트들이 매칭하고 인덱스 값들이 매칭하는 경우, 캐시 로직(103)은 캐시 히트의 표시를 리턴하도록 구성된다.

[0019] [0026] TCC(104)의 검색이 미스를 초래하는 경우, 캐시 로직(103)은 TCC(104)에 입력 메모리 어드레스에 대한 엔트리를 생성하도록 구성될 수 있다. TCC(104)에 자유(free)(또는 그렇지 않으면, 이용 가능한) 엔트리가 존재하는 경우, 캐시 로직(103)은 입력 메모리 어드레스의 더 높은 차수 비트들의 세트를 자유 엔트리에 저장할 수 있다. TCC(104)에서 어떠한 엔트리들도 이용 가능하지 않은 경우, 캐시 로직(103)은 TCC(104)의 기존 엔트리를 축출하고, 축출된 엔트리의 인덱스 값을 특정하는 태그 어레이(105)(및 데이터 어레이(106))의 임의의 대응하는 엔트리들을 무효화할 수 있다. 캐시 로직(103)은, (TCC(104)의 각각의 엔트리에 대한 카운터에 의해 결정될 수 있는) 최소 캐시 라인들과 연관된 TCC(104) 엔트리를 대체하거나, 또는 LRU와 같은 TCC(104)에 대한 임의의 적절한 대체 정책을 적용할 수 있다. 캐시 로직(103)은 임의의 실행 가능한 방식으로, 예컨대, 플래시 무효화 또는 라인별 무효화에 의해 엔트리들을 무효화할 수 있다. 입력 메모리 어드레스의 더 높은 차수 비트들을 특정하는 엔트리를 TCC(104)에 추가하는 것 외에도, 캐시 로직(103)은 또한, 입력 메모리 어드레스의 더 높은 차수 비트들을 저장하는 TCC(104) 엔트리의 인덱스 값 및 입력 메모리 어드레스의 더 낮은 차수 비트들을 특정하는 엔트리를 태그 어레이(105)에 생성할 수 있다.

[0020] [0027] 적어도 하나의 양상에서, 캐시 로직(103)은 매 클록 사이클마다 TCC(104)를 조사(probe)할 수 있다. 그러나 이는 많은 양의 전력을 소모(draw)할 수 있으며 임계 타이밍 경로에 있을 수 있다. TCC(104)에 내장되는 엔트리들이 많을수록, TCC(104)를 조사하는데 더 많은 전력 및 시간이 요구될 수 있다. 따라서, 일부 양상들에서, 캐시 로직(103)은 대부분의 사이클에서 TCC(104)를 검색할 필요성을 방지하는 로직을 포함할 수 있다. 예를 들어, 레지스터들의 세트는 TCC(104)에 대한 이전 액세스에 의해 리턴된 비트들(그리고 대응하는 TCC(104) 엔트리가 유효하게 유지되는지 여부)을 저장하도록 구성될 수 있다. 일반적으로, 캐시 로직(103)은 TCC(104)에 보유된 것들에 대응하는 입력 메모리 어드레스 비트들이 TCC(104)에 대한 이전 액세스에서와 동일한지를 결정할 수 있다. 그 후, 캐시 로직(103)은 연관된 TCC(104) 엔트리가 마지막 액세스 이후부터 무효화되지 않았는지를 결정할 수 있다. 양자의 결정들이 참이면, 캐시 로직(103)은 압축 캐시 태그가 TCC(104)에 있을 것이고, 이전 사이클의 인덱스를 재사용하는 것이 안전하여 현재 사이클에서 TCC(104)를 검색하는 것이 불필요하다는 것을 결정할 수 있다. 어느 하나의 결정도 참이 아니라면, 캐시 로직(103)은 이전 사이클의 TCC(104) 인덱스를 재사용할 수 없고, 제어 로직(103)은 TCC(104)를 검색해야 한다. TCC(104), 태그 어레이(105) 및 데이터 어레이(106)를 검색하기 위한 수단은 하드웨어 및/또는 소프트웨어로서 구현된 로직을 포함한다. 유사하게, 하드웨어 및/또는 소프트웨어로서 구현된 로직은 값들을 판독 및/또는 기록하고, 히트들 및/또는 미스들의 표시들을 리턴하고, TCC(104), 태그 어레이(105) 및 데이터 어레이(106)로부터 값들을 리턴하기 위한 수단으로서 역할을 한다. 이러한 수단 로직의 예는 메모리 제어기들, 캐시 제어기들 및 데이터 제어기들을 포함한다.

[0021] [0028] 이전에 표시된 바와 같이, 도 1에 도시된 압축 방식은 다른 PC-태깅 및/또는 ASID-태깅 마이크로아키텍처 구조들에 적용될 수 있다. 예를 들어, 가상 어드레스는 {태그 압축 인덱스 값, 부분 가상 어드레스} 형태로 변형될 수 있으며, 이는 블록 전체에서 사용될 수 있다. 이렇게 하면 TLB(Translation Lookaside Buffer), 브랜치 예측기들, 중단 지점들, 캐시들 등을 비롯해서, 모든 데이터-경로 엘리먼트들이 이 단축된 형태로 동작하도록 허용할 것이다. 필요한 경우, 변형된 어드레스는 캐시-콘텐츠 디버그 설비들 등에 대해 필요할 때마다 예컨대, 블록 경계들에서 오리지날 어드레스로 변환될 수 있다. 유사하게, 태그 압축 기술은 명령 어드레스들, 로드/저장 어드레스들 및 물리적으로 태깅된 구조들에 독립적으로 적용될 수 있다.

[0022] [0029] 일 양상에서, 명령 실행 파이프라인(112) 및 캐시(102)를 포함하는 프로세서(101)는 집적 회로 상에 배

치된다. 다른 양상에서, 캐시(102)는 프로세서(101)를 포함하는 집적 회로와 별개의 집적 회로 상에 로케이팅될 수 있다.

[0023] [0030] 도 2는 일 양상에 따라, 캐시 태그 압축을 제공하도록 구성된 프로세서(101)의 캐시 로직(103)의 일부의 로직도이다. 도시된 바와 같이, 프로세서(101)는 태그 압축 캐시(TCC)(104) 및 캐시 태그 어레이(105)를 포함한다. 이전에 설명된 바와 같이, TCC(104)는 복수의 TCC 엔트리들(202)을 저장하는 인덱싱된 하드웨어 구조이다. TCC 엔트리들(202)은 메모리 태그의 더 높은 차수 어드레스 비트들의 세트를 포함할 수 있다. 적어도 일부 양상들에서, TCC 엔트리들(202)은 ASID들, 보안 레벨들, 예외 레벨들 등과 관련된 비트들을 더 포함할 수 있다. 도시된 바와 같이, 캐시 태그 어레이(105)는 복수의 엔트리들을 포함하며, 엔트리들 각각은 태그 압축 캐시(TCC) 인덱스 값(204) 및 캐시 태그(205)를 갖는다. TCC 인덱스 값(204)은 TCC(104)의 엔트리(202)의 인덱스 값에 대응할 수 있다. 캐시 태그들(205)은 메모리 태그의 더 낮은 차수 비트들을 포함할 수 있다. 도시된 바와 같이, 캐시 태그들(205)은 메모리 태그의 더 낮은 차수 비트들, 즉, N-비트 메모리 태그의 비트 0 내지 M을 저장한다. 유사하게, TCC(104)는 N-비트 메모리 태그의 더 높은 차수 비트들, 즉, 메모리 태그의 비트들 M+1 내지 N을 저장한다. 따라서, 완전한 메모리 태그(비트들 0 내지 N)는 TCC(104)에 저장된 더 높은 차수 비트들(비트들 M+1 내지 N)을 캐시 태그(205)의 더 낮은 차수 비트들(비트들 0 내지 M)과 결합함으로써 생성될 수 있다.

[0024] [0031] 도시된 바와 같이, 프로세서(101)는 메모리 어드레스(201)에 저장된 데이터를 요청할 수 있다. 캐시 로직(103)은 메모리 어드레스(201)에 저장된 데이터가 캐시(102)에 로케이팅되는지를 결정하도록 구성될 수 있다. 적어도 하나의 양상에서, 메모리 어드레스(201)는 N 비트들의 길이를 갖는 가상 메모리 어드레스일 수 있다. 메모리 어드레스(201)에 저장된 데이터가 캐시(102)에 있는지를 결정하기 위해, 캐시 로직(103)은 TCC(104) 및 캐시 태그 어레이(105)를 병렬로 조사할 수 있다. 보다 구체적으로, 도시된 바와 같이, 캐시 로직(103)은 캐시 태그 어레이(105)의 CAM 검색을 수행할 수 있다. 캐시 태그들(205) 중 하나가 메모리 어드레스(201)의 비트들 0 내지 M과 매칭하는 엔트리를 포함하는 경우, 캐시 태그 어레이(105)는 TCC 인덱스 값(208) 및 캐시 태그(209)를 리턴할 수 있다. 또한, 도시된 바와 같이, 분할기(213)는 메모리 어드레스(201)를 분할할 수 있으며, 이는 메모리 어드레스(201)의 비트들(M+1 내지 N)을 TCC(104)에 제공한다. TCC(104) 상의 CAM 검색은 메모리 어드레스(201)의 비트들 M+1 내지 N을 사용하여 수행될 수 있다. TCC(104)의 엔트리(202)가 메모리 어드레스(201)의 비트들 M+1 내지 N의 값과 매칭하는 값을 포함하는 경우, TCC(104)는 TCC 엔트리(206) 및 TCC 인덱스 값(207)을 리턴하도록 구성되며, 여기서 인덱스 값(207)은 매칭하는 TCC 엔트리(206)를 식별한다. TCC 인덱스 값(207)은 TCC(104)의 각각의 엔트리(202)가 각각의 TCC 인덱스 값(207)에 의해 고유하게 식별되도록 허용하기에 적합한 길이의 임의의 수의 비트들일 수 있다. 따라서, 예를 들어, TCC(104)가 4 개의 하드웨어 엔트리들(예를 들어, 4 개의 TCC 엔트리들(202))을 가진 경우, 각각의 엔트리에 대응하는 TCC 인덱스 값(207)은 길이가 2 비트일 수 있다. 그러한 예에서, 캐시 태그 어레이(105)에 저장된 각각의 TCC 인덱스(204)는 또한 길이가 2 비트일 것이다. 적어도 하나의 양상에서, TCC 인덱스 값들(207)은 주어진 TCC 엔트리(202)의 위치(또는 어드레스)에 기초하여 암시적으로 식별된다.

[0025] [0032] 도시된 바와 같이, 비교기(210)는 TCC(104)로부터 리턴된 TCC 인덱스 값(207)을 캐시 태그 어레이(105)에 의해 리턴된 TCC 인덱스 값(208)과 비교한다. 또한, 비교기(211)는 메모리 어드레스(201)의 더 낮은 차수 비트들 0 : M을, 캐시 태그 어레이(105)로부터 리턴된 캐시 태그 엔트리(209)에 저장된 더 낮은 차수 비트들 0 : M과 비교한다. 도시된 바와 같이, AND 로직(212)은 비교기들(210, 211)의 출력들 상에서 로직 AND 연산을 수행하도록 구성된다. 양자의 비교기들(210, 211)이 매칭을 표시하는 경우, AND 로직(212)의 출력은 캐시 히트의 표시를 리턴하도록 구성된다. 다르게 말하면, 비교기(210)에 의해 비교된 인덱스 값들이 매칭하고, 비교기(211)에 의해 비교된 더 낮은 차수 비트들 0 : M이 매칭하는 경우, 요청된 메모리 어드레스(201)의 콘텐츠들이 캐시(102)에 저장되고, 캐시 로직(103)은 캐시 히트의 표시를 리턴할 수 있다. 그러나 비교된 인덱스 값들이 매칭하지 않음을 비교기(210)가 표시하거나, 또는 더 낮은 차수 비트들 0 : M이 매칭하지 않음을 비교기(211)가 표시하는 경우, 요청된 데이터는 캐시(102)에 있지 않고, 캐시 로직(103)은 캐시 미스의 표시를 리턴할 수 있다. 따라서, 도시된 바와 같이, 캐시 로직(103)은 비트들 0 : M 및 인덱스 값들의 비교를 야기한다. 그렇게 하면 캐시 로직(103)이 종래의 (비트들 0 : N의) 태그 비교 보다 적은 비트들을 비교하도록 허용하며, 이는 종래의 비교보다 빠른 비교를 초래한다.

[0026] [0033] 캐시 미스의 이벤트 시에, 캐시 로직(103)은 위에서 설명된 바와 같이, 요청된 데이터를 더 높은 차수 메모리로부터 가져와서 TCC(104) 및 캐시 태그 어레이(105)에 파퓰레이팅(populate)하도록 구성될 수 있다. 유사하게, TCC(104) 및 캐시 태그 어레이(105)의 초기 CAM 검색들 중 하나가 미스하면, 캐시 로직(103)은 위에서

설명된 바와 같이, 요청된 데이터를 더 높은 차수 메모리로부터 가져와서 TCC(104) 및 캐시 태그 어레이(105)에 파퓰레이팅하도록 구성될 수 있다.

[0027] [0034] 도 3은 일 양상에 따라, 태그 압축 캐시(TCC)(104), 캐시 태그 어레이(105) 및 데이터 어레이(106)에 저장된 예시적인 값들을 예시하는 개략도(300)이다. 도시된 바와 같이, TCC(104) 내의 각각의 TCC 엔트리(202)는 메모리 태그의 일부를 특정하고 각각의 TCC 인덱스 값(207)에 의해 식별된다. 각각의 인덱스 값(207)은 각각의 TCC 엔트리(202)를 고유하게 식별한다. 적어도 하나의 양상에서, 인덱스 값(207)은 TCC(104)의 각각의 엔트리의 위치(또는 어드레스)에 의해 암시적으로 제공된다. 따라서, 이러한 양상들에서, 인덱스 값(207)은 TCC(104)에서 명시적으로 정의되지 않고, 인덱스 값들(207)은 이러한 양상들의 설명을 용이하게 하기 위해 도 3에 도시된다. 각각의 TCC 엔트리(202)의 태그 부분은 메모리 태그의 더 높은 차수 어드레스 비트들일 수 있고, ASID 비트들, 보안 비트들, 예외 레벨 비트들 등과 같은 부가적인 정보를 포함할 수 있다.

[0028] [0035] 도시된 바와 같이, 표(301)는 캐시 태그 어레이(105) 및 데이터 어레이(106)의 예시적인 값들을 포함한다. 데이터 어레이(106)는 캐시(102)의 각각의 엔트리에 대한 데이터 라인(302)을 저장한다. 캐시 태그 어레이(105)는 앞서 설명된 바와 같이, 인덱스 값들(204) 및 캐시 태그들(205)을 포함한다. 캐시 태그 어레이(105)의 인덱스 값들(204)은 TCC(104)의 주어진 TCC 엔트리(202)에 대응하는 인덱스 값(207)에 대응하는 반면에, 각각의 대응하는 캐시 태그(205)는 메모리 태그의 더 낮은 차수 비트들을 저장한다. 따라서, 도시된 바와 같이, 표(301) 내의 다수의 엔트리들은 동일한 인덱스 값들(204)을 저장할 수 있다. 유리하게는, TCC(104)는 메모리 태그의 더 높은 차수 비트들의 전체 비트 패턴의 단일 카피를 저장하여, 캐시 태그 어레이(105)에 저장되는 중복 정보(즉, 메모리 태그의 너무 긴 더 높은 차수 비트들이 아니라 상이한 인덱스 값들(204))의 양을 감소시킨다. 따라서, 메모리 태그는 TCC(104) 및 캐시 태그 어레이(105)에 걸쳐 저장되며, 더 높은 차수 비트들은 TCC(104)에 저장되고, 더 낮은 차수 비트들은 캐시 태그 어레이(105)에 저장된다. 개념적으로 말하면, TCC(104)의 TCC 엔트리(202)의 태그 부분은 캐시 태그 어레이(105)의 인덱스 값(204)을 대체하여 완전한 메모리 태그를 생성할 수 있다. 예를 들어, 표(301)의 제 1 행에서, 인덱스(204)의 비트들 "00"은 TCC(104)의 "00"의 인덱스 값(207)에 대응하는 TCC 엔트리(202)의 태그 비트들 "01 ... 0100"에 의해 대체될 것이다.

[0029] [0036] 일 예시적인 양상에서, 도 3에 도시된 구성은 가상 명령 캐시를 표 1에 도시된 예시적인 구성으로 대체할 수 있다 :

표 1

[0030]

라인 크기	64 바이트
연관	4-웨이
세트들의 수	128
라인들의 총수	512
공칭 용량	32 KB
어드레스 폭	49 비트
세트 인덱스	가상 어드레스(7비트)
태그 비트들의 수(라인 당)	{가상 어드레스, ASID}(36+16 = 52 비트)
태그 비트들의 수(총)	(라인당 52 비트 * 512 라인) = 26,624 비트

[0031] [0037] 따라서, 표 1에 도시된 바와 같이, 가상 명령 캐시는 32KB(또는 32,768 바이트 또는 262,144 비트)의 총 캐시 저장(total cache storage) 중에서, 태그 저장을 위해 26,624 비트의 저장 오버헤드를 필요로 한다. 이 예시적인 구성에서 필요한 명령 캐시 태그는 {PC[48:13], ASID}일 것이다.

[0032] [0038] 그러나 이러한 비트들의 서브세트(예컨대, {PC[48:28], ASID})은 실-세계 워크로드들에서 거의 가변성을 보이지 않는 경향이 있다. 이 예시적인 명령 캐시에 대해 4-엔트리 TCC(104)를 생성함으로써, 각각의 TCC 엔트리(202)는 총 37 비트들에 대해 21 PC [48:28] 비트들 및 ASID의 16 비트들을 저장할 것이다. 따라서, TCC(104)에 대해 요구되는 총 저장량은 "4개의 엔트리들 x 37 비트들" 또는 148 비트일 것이다. TCC 엔트리(202)는 TCC(104)에서 각각의 개별 TCC 엔트리(202)의 위치(또는 어드레스)에 의해 암시적으로 인덱싱될 수 있다(예를 들어, 4-엔트리 TCC(104)에서, 위치들 0, 1, 2 및 3은 00, 01, 10 및 11의 이진 인덱스 값들에 각각 대응함). 한편, 캐시 태그 어레이(105)의 인덱스들(204)은 2-비트 인덱스 값들을 저장할 것인 반면에, 캐시 태그들(205)은 총 17 비트들에 대해 태그 비트들 PC [27:13] 또는 15 비트들을 저장할 것이다. 명령 캐시의 512 라인들에 걸쳐서, 캐시 태그 어레이(105)의 총 저장은 총 8,704 비트들에 대해 "512 개의 엔트리들 x 17 비트들"일 것이다. TCC(104) 및 캐시 태그 어레이(105)에 걸쳐 필요한 총 저장은 8,852 비트들(148+8,704)일

것이며, 이는 표 1에 설명된 명령 캐시에 의해 요구되는 26,624 비트보다 상당히 적다.

[0033] [0039] 도 4는 일 양상에 따라, 캐시 태그 압축을 제공하기 위한 방법(400)을 예시하는 흐름도이다. 방법(400)은 캐시 로직(103)이 N 비트 길이의 메모리 어드레스를 수신할 수 있는 단계(410)에서 시작한다. 메모리 어드레스는 메모리 어드레스에 저장된 데이터가 캐시(102)에 로케이팅되는지를 결정하기 위해 요청의 일부로서 수신될 수 있다. 그 후, 캐시 로직(103)은 TCC(104) 및 캐시 태그 어레이(105)를 병렬로 조사할 수 있다. 보다 구체적으로, 단계(415)에서, 캐시 로직(103)은 단계(410)에서 수신된 메모리 어드레스의 하위 M개의 비트들을 이용하여 캐시 태그 어레이(105) 상에서 CAM 검색을 수행할 수 있다. 캐시 태그 어레이(105)의 엔트리가 메모리 어드레스의 하위 M개의 비트들과 매칭하는 값을 저장하는 경우, 캐시 태그 어레이(105)에 히트가 존재하고, 방법은 단계(460)로 진행된다. 또한, 캐시 태그 어레이(105)는 매칭하는 엔트리에 저장된 메모리 어드레스의 하위 M개의 비트들 및 인덱스 값을 리턴할 수 있다. 메모리 어드레스의 하위 M개의 비트들과 매칭하는 값을 저장한 캐시 태그 어레이(105)의 엔트리가 없는 경우, 캐시 태그 어레이(105)에 미스가 존재하고, 방법은 단계(480)로 진행된다.

[0034] [0040] 단계(420)에서, 캐시 로직(103)은 요청된 메모리 어드레스의 더 높은 차수 비트들(M+1 내지 N)을 이용하여 TCC(104)를 조사할 수 있다. TCC(104)의 엔트리가 요청된 메모리 어드레스의 더 높은 차수 비트들과 매칭하는 값을 저장한 경우, 방법은 단계(460)로 진행된다. TCC(104)에서 히트의 이벤트 시에, TCC(104)는 메모리 어드레스의 더 높은 차수 비트들과 매칭하는 TCC(104)의 엔트리의 인덱스 값을 리턴할 수 있다. TCC(104)에 미스가 존재하는 경우, 방법은, 이용 가능한 엔트리들이 TCC(104)에 존재하는지를 캐시 로직(103)이 결정할 수 있는 단계(430)로 진행된다. 엔트리가 이용 가능하지 않은 경우, 방법은, 캐시 로직(103)이 TCC(104)의 엔트리를 축출하고 캐시 태그 어레이(105)의 임의의 대응하는 엔트리들을 무효화할 수 있는 단계(440)로 진행된다. 캐시 로직(103)은 최소 최근 사용 엔트리(least recently used entry)를 축출하는 것, 또는 캐시 태그 어레이(105)의 가장 적은 엔트리들과 관련된 TCC(104)의 엔트리를 축출하는 것과 같은 임의의 적합한 축출 정책을 사용할 수 있다. 캐시 로직(103)은 임의의 실행 가능한 방식으로, 예컨대, 플래시 무효화 또는 라인별 무효화에 의해 캐시 태그 어레이(105)의 엔트리들을 무효화할 수 있다. 도 5를 참조하여 보다 상세히 설명되는 단계(450)에서, 캐시 로직(103)은 TCC(104)에 새로운 엔트리를 설정할 수 있다. 일반적으로, 캐시 로직(103)은 메모리 어드레스를 하위 세트의 비트들 0 : M 및 상위 세트의 비트들 M+1 : N로 분할할 수 있다. 그 후, 캐시 로직(103)은 캐시 태그 어레이(105)에 하위 세트의 비트들을 저장하고, TCC(104)에 상위 세트의 비트들을 저장할 수 있다.

[0035] [0041] 단계(460)에서, 캐시 로직(103)은 TCC(104) 및 캐시 태그 어레이(105)에 의해 리턴된 인덱스 값들을 비교할 수 있다. 캐시 로직(103)은 추가로, 캐시 태그 어레이(105)에 의해 리턴된 더 낮은 차수 비트들을 요청된 메모리 어드레스의 더 낮은 차수 비트들과 비교할 수 있다. 이 둘다의 비교들에 매칭이 발생하는 경우, 방법은 캐시 로직이 캐시 히트의 표시를 리턴할 수 있는 단계(470)로 진행된다. 단계(460)에서 수행된 비교들 중 하나(또는 둘 다)에 매칭이 발생하지 않는 경우, 방법은 캐시 로직(103)이 캐시 미스의 표시를 리턴하는 단계(480)로 진행된다.

[0036] [0042] 도 5는 일 양상에 따라, 태그 압축 캐시(104)에 새로운 엔트리를 설정하기 위한 단계(450)에 대응하는 방법(500)을 예시하는 흐름도이다. 도시된 바와 같이, 방법(500)은, 프로세서(101)가 단계(410)에서 제공된 메모리 어드레스와 연관된 데이터를 상위 레벨 메모리(예컨대, 상위 레벨 캐시 또는 메인 메모리)로부터 리트리브할 수 있는 단계(510)에서 시작한다. 단계(520)에서, 캐시 로직(103)은 TCC(104)의 제 1 엔트리에 메모리 어드레스의 더 높은 차수 비트들(예컨대, 비트들 M+1 : N)을 저장할 수 있다. 단계(530)에서, 캐시 로직(103)은 TCC(104)의 제 1 엔트리의 인덱스 값 및 메모리 어드레스의 더 낮은 차수 비트들(예컨대, 비트들 0 : M)을 태그 어레이(105)의 제 1 엔트리에 저장할 수 있다. 단계(540)에서, 캐시 로직(103)은 단계(510)에서 리트리브된 데이터를 태그 어레이(105)의 제 1 엔트리에 대응하는 데이터 어레이(106)의 엔트리에 저장할 수 있다.

[0037] [0043] 도 6은 일 양상에 따라, 캐시 태그 압축을 제공하도록 구성된 프로세서(101)를 통합하는 컴퓨팅 디바이스(601)를 예시하는 블록도이다. 도 1 내지 도 5에 도시된 모든 장치들 및 방법들은 컴퓨팅 디바이스(601)에 포함되거나 컴퓨팅 디바이스(601)에 의해 수행될 수 있다. 컴퓨팅 디바이스(601)는 또한 네트워크(630)를 통해 다른 컴퓨팅 디바이스들에 연결될 수 있다. 일반적으로, 네트워크(630)는 원격통신 네트워크 및/또는 광역 네트워크(WAN)일 수 있다. 특정 양상에서, 네트워크(630)는 인터넷이다. 일반적으로, 컴퓨팅 디바이스(601)는 데스크톱 컴퓨터, 랩톱 컴퓨터, 태블릿 컴퓨터 및 스마트폰을 (제한 없이) 포함하여, 캐시 태그 압축을 구현하도록 구성된 프로세서를 포함하는 임의의 디바이스일 수 있다.

[0038] [0044] 컴퓨팅 디바이스(601)는 일반적으로 버스(620)를 통해 메모리(608), 네트워크 인터페이스 디바이스(618), 저장소(609), 입력 디바이스(622) 및 출력 디바이스(624)에 연결된 프로세서(101)를 포함한다. 컴퓨팅 디바이스(601)는 일반적으로 운영 체제(도시되지 않음)의 제어 하에 있다. 본원에서 개시되는 기능들을 지원하는 임의의 운영 체제가 사용될 수 있다. 프로세서(101)는 단일 CPU, 다수의 CPU들, 다수의 프로세싱 코어들을 갖는 단일 CPU 등을 나타내도록 포함된다. 네트워크 인터페이스 디바이스(618)는 컴퓨팅 디바이스(601)가 네트워크(630)를 통해 다른 컴퓨팅 디바이스들과 통신하도록 허용하는 임의의 타입의 네트워크 통신 디바이스일 수 있다.

[0039] [0045] 저장소(609)는 영구적인 저장 디바이스일 수 있다. 저장소(609)가 단일 유닛으로 도시되지만, 저장소(609)는 고정식 및/또는 제거 가능한 저장 디바이스들, 예컨대, 고정식 디스크 드라이브들, 고상 드라이브들, SAN 저장소, NAS 저장소, 제거 가능한 메모리 카드들 또는 광학 저장소의 조합일 수 있다. 메모리(608) 및 저장소(609)는 다수의 1 차 및 2 차 저장 디바이스들에 걸쳐 있는 하나의 가상 어드레스 공간의 일부일 수 있다.

[0040] [0046] 입력 디바이스(622)는 컴퓨팅 디바이스(601)에 입력을 제공하기 위한 임의의 디바이스일 수 있다. 예를 들어, 키보드 및/또는 마우스가 사용될 수 있다. 출력 디바이스(624)는 컴퓨팅 디바이스(601)의 사용자에게 출력을 제공하기 위한 임의의 디바이스일 수 있다. 예를 들어, 출력 디바이스(624)는 임의의 종래의 디스플레이 스크린 또는 스피커들의 세트일 수 있다. 입력 디바이스(622)와 별개로 도시되었지만, 출력 디바이스(624) 및 입력 디바이스(622)는 결합될 수 있다. 예를 들어, 통합된 터치 스크린을 갖는 디스플레이 스크린이 사용될 수 있다.

[0041] [0047] 유리하게는, 본원에서 개시되는 양상들은 메모리 태그 값들을 저장하는데 필요한 공간의 양을 감소시킨다. 더 작은 태그 압축 캐시에 메모리 태그의 더 높은 차수 비트들의 중복 값들을 저장함으로써, 보다 작은 태그 어레이 구조가 메모리 태그의 더 낮은 차수 비트들을 저장하는데 사용될 수 있다. 이렇게 하면 이러한 구조들이 더 작은 전체 실리콘 다이를 사용하여 생성되도록 허용할 수 있으며, 이는 다른 하드웨어 구조들이 서로 더 근접하게 배치되도록 허용함으로써 제조 비용들을 낮추고 누설 전류를 감소시키고 회로 타이밍을 개선한다. 부가적으로 또는 대안적으로, 절감된 공간은 다른 마이크로아키텍처 구조들에 쓰일 수 있다.

[0042] [0048] 다수의 양상들이 설명되었다. 그러나 이들 양상들에 대한 다양한 수정들이 가능하며, 본원에서 제시된 원리들은 다른 양상들에도 마찬가지로 적용될 수 있다. 그러한 방법들의 다양한 과제들은 마이크로프로세서들, 임베딩된 제어기들 또는 IP 코어들과 같은, 로직 엘리먼트들의 하나 또는 그 초과의 어레이들에 의해 실행 가능한 명령들의 세트로서 구현될 수 있다.

[0043] [0049] 위에서 설명된 방법들의 다양한 동작들은 프로세서, 펌웨어, ASIC(application specific integrated circuit), 게이트 로직/레지스터들, 메모리 제어기 또는 캐시 제어기와 같이 동작들을 수행할 수 있는 임의의 적절한 수단에 의해 수행될 수 있다. 일반적으로, 도면들에서 예시되는 임의의 동작들은, 그 동작들을 수행할 수 있는 대응하는 기능적 수단에 의해 수행될 수 있다.

[0044] [0050] 앞서 개시된 디바이스들 및 기능성들은 컴퓨터 판독 가능한 매체들 상에 저장된 컴퓨터 파일들(예를 들어, RTL, GDSII, GERBER 등)로 설계 및 구성될 수 있다. 이러한 파일들 중 일부 또는 전부는 이러한 파일들에 기초하여 디바이스들을 제조하는 제조 취급자에 제공될 수 있다. 결과적인 제품들은 반도체 웨이퍼들을 포함하며, 이 반도체 웨이퍼들은 그 후 반도체 다이로 절단되고 반도체 칩으로 패키징된다. 이러한 파일들 중 일부 또는 전부는 본원에서 설명되는 디바이스들을 제조하기 위해 설계 데이터를 사용하여 제조 장비를 구성하는 제조 취급자들에 제공될 수 있다. 컴퓨터 파일들로부터 형성된 결과적인 제품들은 추후에 반도체 다이(예를 들어, 프로세서(101))로 절단되어 패키징되는 반도체 웨이퍼들을 포함하며, 모바일 전화들, 스마트폰들, 랩톱들, 넷북들, 태블릿들, 울트라북들, 데스크톱 컴퓨터들, 디지털 비디오 레코더들, 셋-톱 박스들 및 집적 회로들이 사용되는 임의의 다른 디바이스들을 포함(그러나 이것으로 제한되지 않음)하는 제품들에 추가로 통합될 수 있다.

[0045] [0051] 일 양상에서, 컴퓨터 파일들은 물리적 설계 레이아웃들, 스키메틱(schematics), 하드웨어-기술 언어(예를 들어, Verilog, VHDL 등)의 형태로 도면들에 도시되고 위에서 설명된 회로들을 포함하는 설계 구조를 형성한다. 예를 들어, 설계 구조는 도면들에서 도시되고 위에서 설명된 바와 같은 회로의 그래픽 표현 또는 텍스트 파일일 수 있다. 설계 프로세스는 바람직하게는 아래에 설명된 회로들을 넷리스트(netlist)로 합성(또는 변환)되며, 여기서 넷리스트는 예를 들어, 머신 판독 가능한 매체들 중 적어도 하나에 레코딩되며 집적 회로 설계의 다른 엘리먼트들 및 회로들에 대한 연결들을 설명하는 와이어들, 트랜지스터들, 로직 게이트들, 제어 회로들, I/O, 모델 등의 리스트이다. 예를 들어, 매체는 CD, 콤팩트 플래시, 다른 플래시 메모리 또는 하드-디

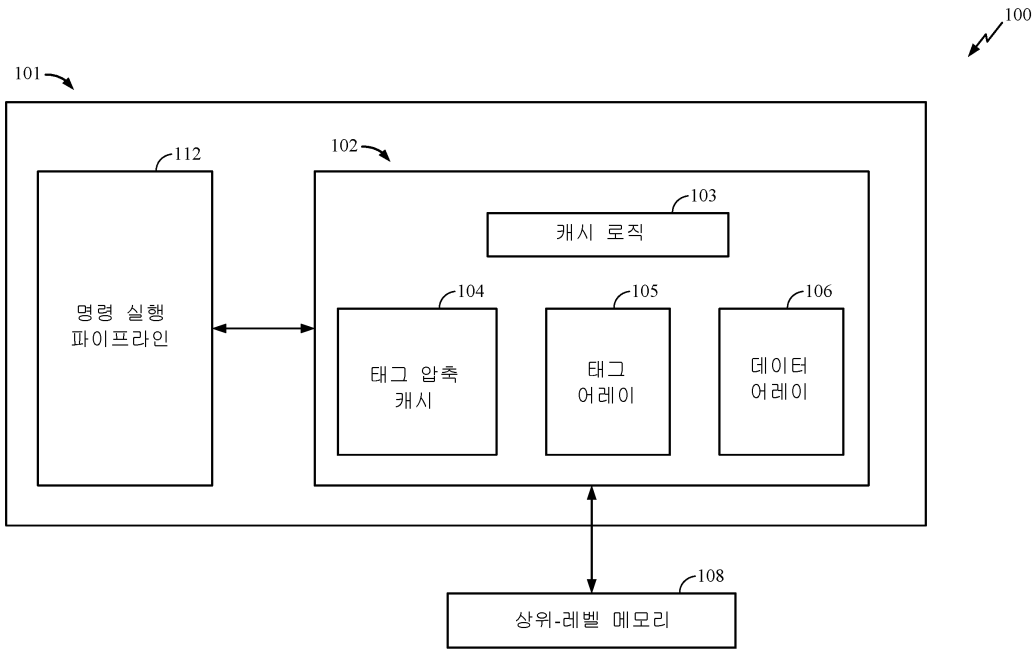
스크 드라이브와 같은 저장 매체일 수 있다. 다른 양상에서, 본원에서 설명되는 하드웨어, 회로 및 방법은 프로세서에 의해 실행될 때 도면들에 도시되고 위에서 설명된 회로들의 기능을 시뮬레이팅하는 컴퓨터 파일들로 구성될 수 있다. 이러한 컴퓨터 파일들은 회로 시뮬레이션 툴들, 스키메틱 편집기들 또는 다른 소프트웨어 애플리케이션들에서 사용될 수 있다.

[0046] [0052] 본원에서 개시되는 양상들의 구현들은 또한, 로직 엘리먼트들(예를 들어, 프로세서, 마이크로프로세서, 마이크로제어기 또는 다른 유한 상태 머신)의 어레이를 포함하는 머신에 의해 실행 가능한 하나 또는 그 초과의 명령들의 세트들로서 유형으로(tangibly) (예를 들어, 본원에서 나열된 하나 또는 그 초과의 컴퓨터-판독 가능한 저장 매체들의 유형의 컴퓨터-판독 가능한 특징들에서) 구현될 수 있다. "컴퓨터-판독 가능한 매체"라는 용어는 휘발성, 비휘발성, 제거 가능한 및 제거 불가능한 저장 매체를 포함하여, 정보를 저장하거나 전달할 수 있는 임의의 매체를 포함할 수 있다. 컴퓨터-판독 가능한 매체의 예들은, 전자 회로, 반도체 메모리 디바이스, ROM, 플래시 메모리, EROM(erasable ROM), 플로피 디스켓 또는 다른 자기 저장소, CD-ROM/DVD 또는 다른 광학 저장소, 하드 디스크 또는 원하는 정보를 저장하는데 사용될 수 있는 임의의 다른 매체, 광섬유 매체, 라디오 주파수(RF) 링크, 또는 원하는 정보를 전달하는데 사용될 수 있고 액세스될 수 있는 임의의 다른 매체를 포함한다. 컴퓨터 데이터 신호는 전자 네트워크 채널들, 광섬유들, 에어(air), 전자기, RF 링크들 등과 같은 송신 매체를 통해 전파될 수 있는 임의의 신호를 포함할 수 있다. 코드 세그먼트들은 인터넷 또는 인트라넷과 같은 컴퓨터 네트워크들을 통해 다운로드될 수 있다. 어떠한 경우에도, 본 개시의 범위는 이러한 양상들에 의해 제한되는 것으로 해석되어서는 안 된다.

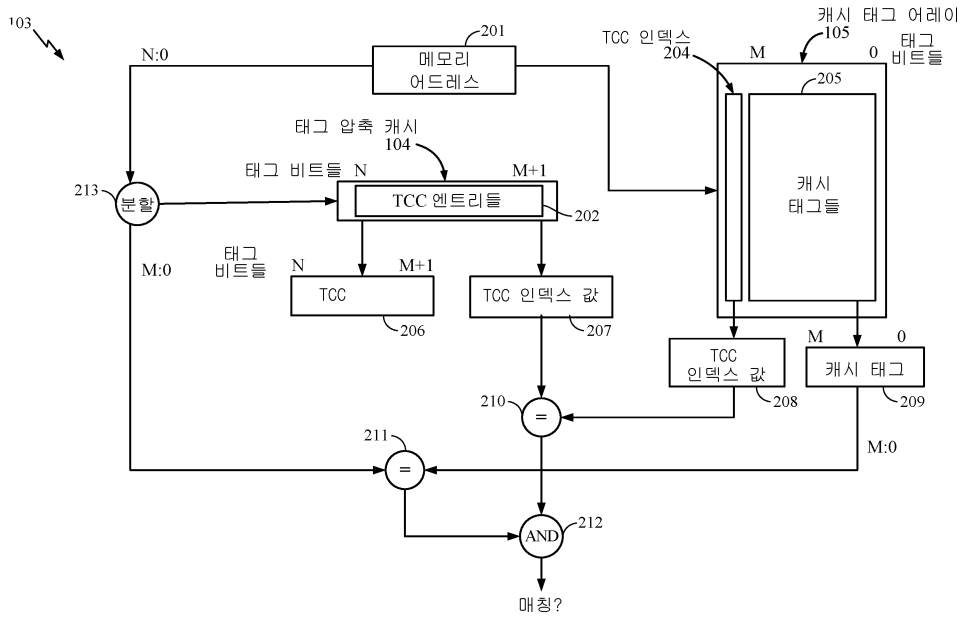
[0047] [0053] 개시된 양상들의 이전의 설명은 당업자가 개시된 양상들을 실시하거나 이용하는 것을 가능케 하도록 제공된다. 이들 양상들에 대한 다양한 변형들이 당업자들에게 쉽게 자명하게 될 것이며, 본원에서 정의되는 원리들은 본 개시의 범위로부터 벗어남 없이 다른 양상들에 적용될 수 있다. 따라서, 본 개시는 본원에서 도시된 양상들로 제한되도록 의도되는 것이 아니라, 다음의 청구항들에 의해 정의된 바와 같은 원리들 및 신규한 특징들에 부합하는 최광의의 가능한 범위로 허여될 것이다.

도면

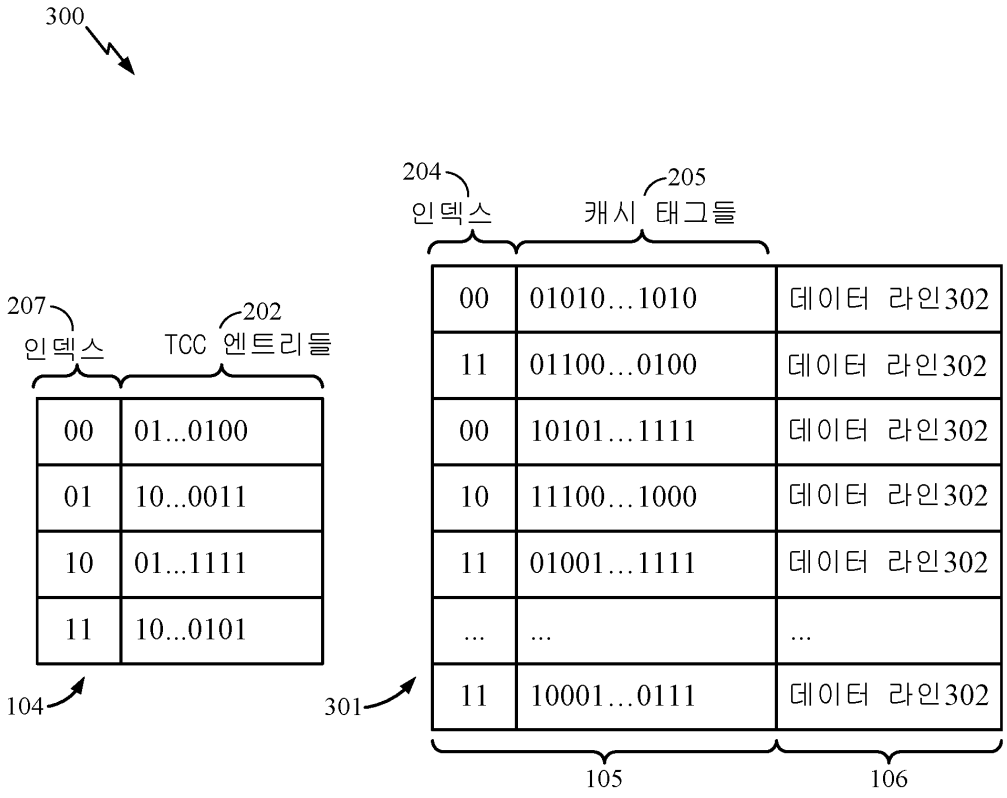
도면1



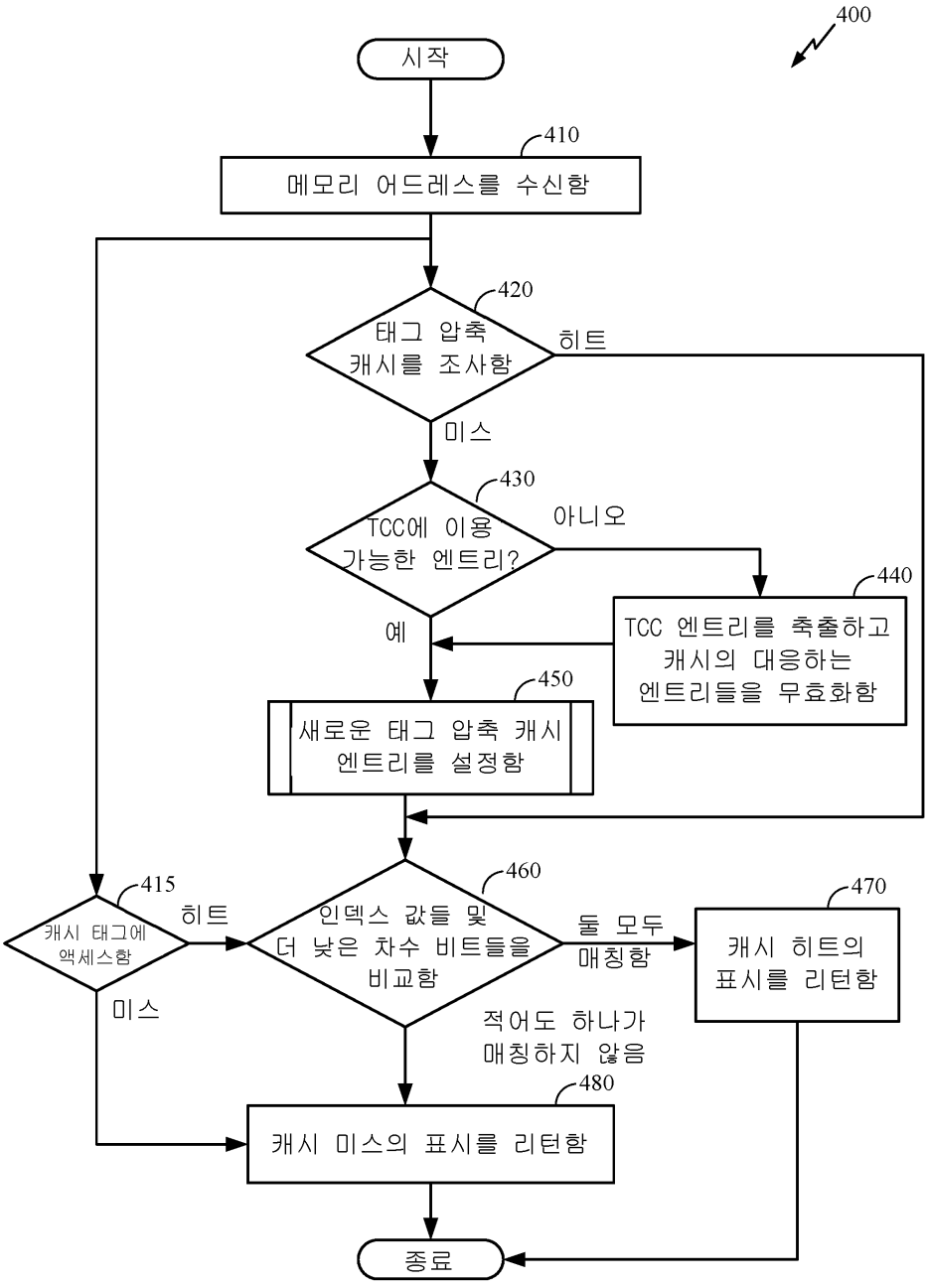
도면2


103
N:0
201
메모리
어드레스
TCC 인덱스
204
캐시 태그 어레이
105
M
0
태그
비트들
205
캐시
태그들
태그 압축 캐시
104
태그 비트들 N
M+1
213
분할
TCC 엔트리들
202
M:0
태그
비트들
N
M+1
TCC
206
TCC 인덱스 값
207
TCC
인덱스 값
208
캐시 태그
209
210
=
211
=
M:0
AND
212
매칭?

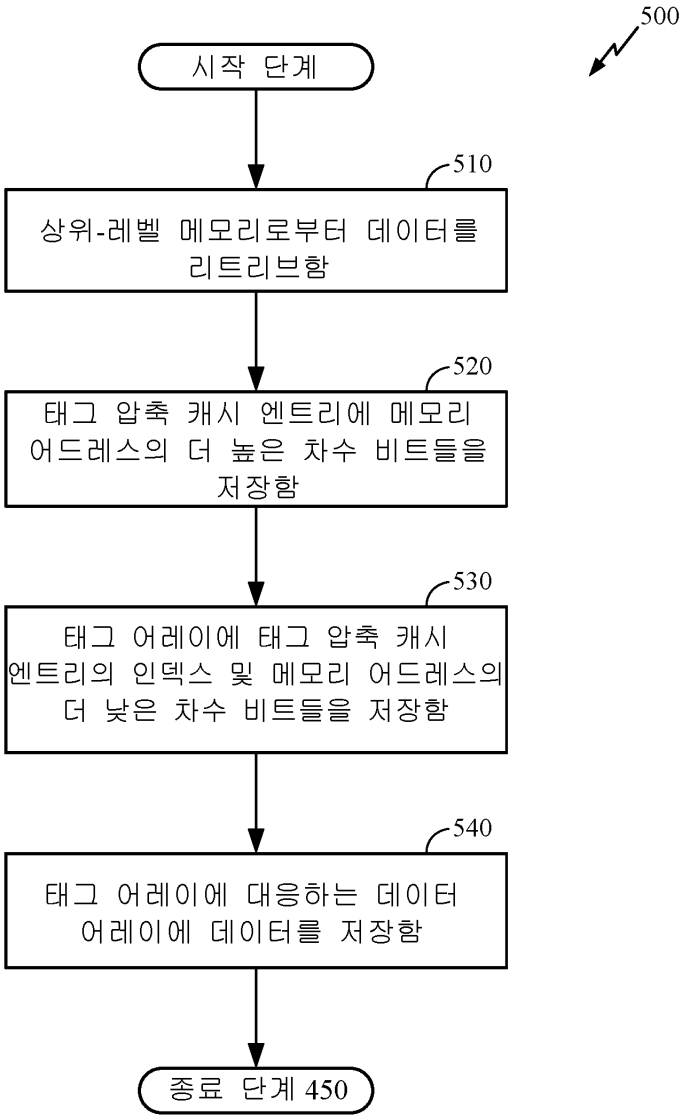
도면3


300
204
인덱스
205
캐시 태그들
207
인덱스
202
TCC 엔트리들
00
01...0100
01
10...0011
10
01...1111
11
10...0101
104
301
00
01010…1010
데이터 라인302
11
01100…0100
데이터 라인302
00
10101…1111
데이터 라인302
10
11100…1000
데이터 라인302
11
01001…1111
데이터 라인302
...
...
...
11
10001…0111
데이터 라인302
105
106

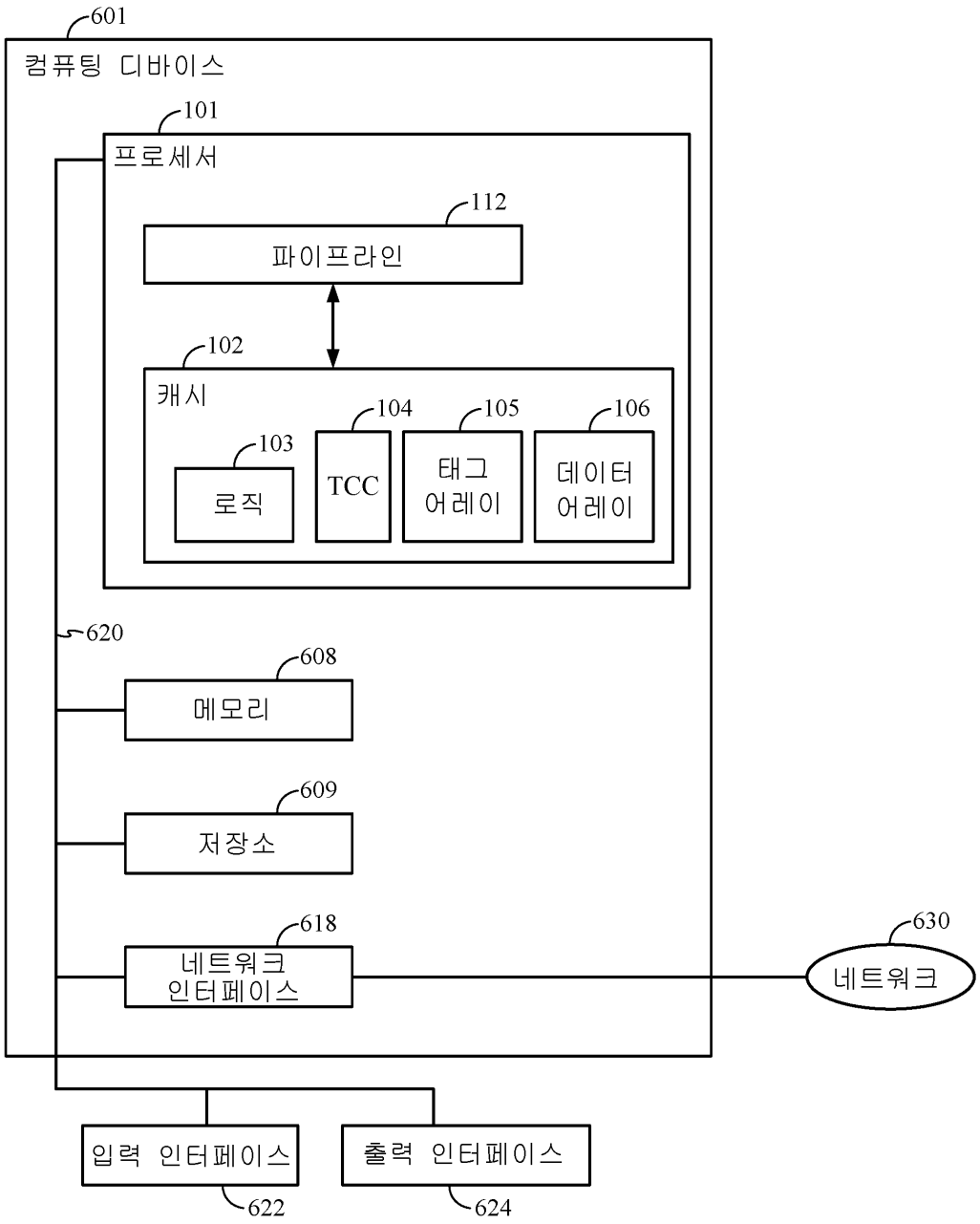
도면4


400
시작
410
메모리 어드레스를 수신함
420
태그 압축 캐시를 조사함
히트
미스
430
TCC에 이용 가능한 엔트리?
아니오
440
TCC 엔트리를 축출하고 캐시의 대응하는 엔트리들을 무효화함
예
450
새로운 태그 압축 캐시 엔트리를 설정함
415
캐시 태그에 액세스함
히트
미스
460
인덱스 값들 및 더 낮은 차수 비트들을 비교함
둘 모두 매칭함
적어도 하나가 매칭하지 않음
470
캐시 히트의 표시를 리턴함
480
캐시 미스의 표시를 리턴함
종료

도면5


500
시작 단계
510
상위-레벨 메모리로부터 데이터를 리트리브함
520
태그 압축 캐시 엔트리에 메모리 어드레스의 더 높은 차수 비트들을 저장함
530
태그 어레이에 태그 압축 캐시 엔트리의 인덱스 및 메모리 어드레스의 더 낮은 차수 비트들을 저장함
540
태그 어레이에 대응하는 데이터 어레이에 데이터를 저장함
종료 단계 450

도면6


601
컴퓨팅 디바이스
101
프로세서
112
파이프라인
102
캐시
104
105
106
103
로직
TCC
태그 어레이
데이터 어레이
620
608
메모리
609
저장소
618
네트워크 인터페이스
630
네트워크
입력 인터페이스
622
출력 인터페이스
624