(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2020-0052528
(43) 공개일자 2020년05월15일

(51) 국제특허분류(Int. Cl.)
G06F 12/0871 (2016.01) *G06F 12/0893* (2016.01)
(52) CPC특허분류
G06F 12/0871 (2013.01)
G06F 12/0893 (2013.01)
(21) 출원번호 10-2018-0135546
(22) 출원일자 2018년11월07일
심사청구일자 없음

(71) 출원인
에스케이하이닉스 주식회사
경기도 이천시 부발읍 경충대로 2091
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
김선욱
경기도 남양주시 도농로 34, 214동 2003호(도농동, 부영아파트)
이원준
서울특별시 동대문구 무학로49길 14, 103호(용두동)
(뒷면에 계속)
(74) 대리인
특허법인 이노

전체 청구항 수 : 총 18 항

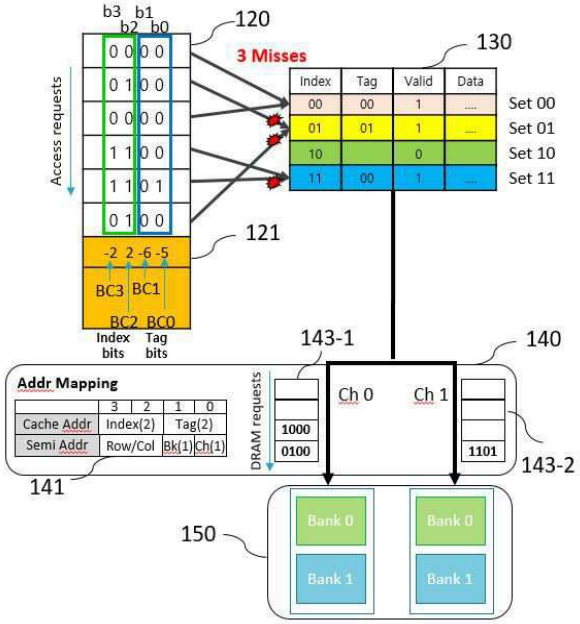
(54) 발명의 명칭 **비트 카운터를 이용하는 컴퓨팅 시스템 및 방법**

(57) 요 약

본 발명은 비트 카운터를 기반으로 address를 hashing하여 index bit를 결정함으로써 cache에서 일부 set로 집중된 접근으로 인한 cache conflict miss를 방지하여 반도체 장치으로의 접근 횟수를 줄이며 동시에 반도체 장치의 동일 bank로의 연속적인 접근을 방지하여 반도체 장치의 bank conflict를 감소시킬 수 있는 비트 카운터를 이

(뒷면에 계속)

대 표 도 - 도3b



용하는 컴퓨팅 시스템을 제공할 수 있다.

본원의 제1 발명에 따른 비트 카운터를 이용하는 컴퓨팅 시스템은, 호스트 장치; 상기 호스트 장치의 데이터를 임시 저장하고, 복수의 세트를 포함하는 Cache; 상기 호스트 장치로부터 복수 비트의 캐시 어드레스를 수신하고, 복수의 비트 카운터 유닛을 이용하여 상기 캐시 어드레스를 계수하고, 상기 Cache의 해시 함수를 결정하는 캐시 콘트롤러; 반도체 장치; 및 상기 캐시 콘트롤러로부터 상기 캐시 어드레스를 수신하고, 상기 캐시 어드레스와 반도체 장치 어드레스를 매핑하는 메모리 콘트롤러를 포함할 수 있다.

(72) 발명자

백윤아

서울특별시 성북구 안암로9가길 68, 304호(안암동5가)

전재영

서울특별시 성북구 인촌로12길 48, 2층(안암동3가)

명 세 서

청구범위

청구항 1

호스트 장치;

상기 호스트 장치의 데이터를 임시 저장하고, 복수의 세트를 포함하는 Cache;

상기 호스트 장치로부터 복수 비트의 캐시 어드레스를 수신하고, 복수의 비트 카운터 유닛을 이용하여 상기 캐시 어드레스를 계수하고, 상기 Cache의 해시 함수를 결정하는 캐시 콘트롤러;

반도체 장치; 및

상기 캐시 콘트롤러로부터 상기 캐시 어드레스를 수신하고, 상기 캐시 어드레스와 반도체 장치 어드레스를 매핑하는 메모리 콘트롤러

를 포함하는 비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 2

제1항에 있어서,

상기 캐시 콘트롤러는, 소정 횟수 동안 상기 복수의 비트 카운터 유닛을 이용하여 상기 캐시 어드레스를 비트별로 계수하여 비트별 계수값을 생성하고, 상기 비트별 계수값에 절대치를 취한 비트별 절대값을 생성하고, 상기 비트별 절대값 중 가장 작은 값의 비트 조합에 대응하는 어드레스를 상기 Cache의 index bit로 사용하도록 상기 Cache의 해시 함수를 결정하고 상기 해시 함수에 의해 상기 복수의 세트 중 어느 하나를 선택하는 것을 특징으로 하는 비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 3

제2항에 있어서,

상기 Cache 내 세트 각각은 상호 식별 가능한 태그를 가진 복수의 블록을 포함하는 비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 4

제2항에 있어서,

상기 캐시 콘트롤러는 상기 해시 함수를 이용하여 상기 Cache 내 데이터를 상기 반도체 장치로 플러시 하는 것을 특징으로 하는 비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 5

제2항에 있어서,

상기 비트 카운터는 상기 cache address 내 개별 비트가 "1"이면 양의 값으로, 개별 비트가 "0"이면 음의 값으로 계수하는 것을 특징으로 하는 비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 6

제1항에 있어서,

상기 Cache, 상기 캐시 콘트롤러, 상기 반도체 장치, 및 상기 메모리 콘트롤러는 하나의 모듈로 구현되는 비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 7

제1항에 있어서,

상기 캐시 콘트롤러는 상기 호스트 장치와 상기 Cache 사이에 배치되는 비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 8

제1항에 있어서,

상기 메모리 콘트롤러는 상기 캐시 콘트롤러와 상기 반도체 장치 사이에 배치되는 비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 9

호스트 장치;

상기 호스트 장치의 데이터를 임시 저장하고, 복수의 세트를 포함하는 Cache;

반도체 장치: 및

상기 호스트 장치로부터 복수 비트의 캐시 어드레스를 수신하고, 복수의 비트 카운터 유닛을 이용하여 상기 캐시 어드레스를 계수함으로써 상기 Cache의 해시 함수를 결정하고, 상기 캐시 어드레스와 반도체 장치 어드레스를 매핑하는 캐시 콘트롤러

를 포함하는 비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 10

제9항에 있어서,

상기 캐시 콘트롤러는, 소정 횟수 동안 상기 복수의 비트 카운터 유닛을 이용하여 상기 캐시 어드레스를 비트별로 계수하여 비트별 계수값을 생성하고, 상기 비트별 계수값에 절대치를 취한 비트별 절대값을 생성하고, 상기 비트별 절대값 중 가장 작은 값의 비트 조합에 대응하는 어드레스를 상기 Cache의 index bit로 사용하도록 해시 함수를 결정하고 상기 해시 함수에 의해 상기 복수의 세트 중 어느 하나를 선택하는 것을 특징으로 하는 비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 11

제10항에 있어서,

상기 Cache 내 세트 각각은 상호 식별 가능한 태그를 가진 복수의 블록을 포함하는 비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 12

제10항에 있어서,

상기 캐시 콘트롤러는 상기 해시 함수를 이용하여 상기 Cache 내 데이터를 상기 반도체 장치로 플러시 하는 것을 특징으로 하는 비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 13

제10항에 있어서,

상기 비트 카운터는 상기 cache address 내 개별 비트가 "1"이면 양의 값으로, 개별 비트가 "0"이면 음의 값으로 계수하는 것을 특징으로 하는 비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 14

제9항에 있어서,

상기 Cache, 상기 캐시 콘트롤러, 및 상기 반도체 장치는 하나의 모듈로 구현되는 비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 15

제1항에 있어서,

상기 호스트 장치는, 호스트, 서버, 저장 영역 네트워크의 저장 콘트롤러, 워크스테이션, 개인용 컴퓨터, 핸드헬드 컴퓨터, 수퍼컴퓨터, 컴퓨터 클러스터, 네트워크 스위치, 라우터, 데이터베이스, 데이터 획득 시스템, 진단 시스템, 테스트 시스템, 로봇, 포터블 전자 기기, 무선 기기 중 어느 하나인 것을 특징으로 하는

비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 16

제9항에 있어서,

상기 호스트 장치는, 호스트, 서버, 저장 영역 네트워크의 저장 콘트롤러, 워크스테이션, 개인용 컴퓨터, 핸드헬드 컴퓨터, 수퍼컴퓨터, 컴퓨터 클러스터, 네트워크 스위치, 라우터, 데이터베이스, 데이터 획득 시스템, 진단 시스템, 테스트 시스템, 로봇, 포터블 전자 기기, 무선 기기 중 어느 하나인 것을 특징으로 하는

비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 17

제9항에 있어서,

상기 캐시 콘트롤러는 상기 호스트 장치와 상기 Cache 사이에 배치되는 비트 카운터를 이용하는 컴퓨팅 시스템.

청구항 18

캐시 콘트롤러가 비트 카운터 유닛을 리셋시키는 단계;

상기 캐시 콘트롤러가 입력되는 어드레스 비트를 상기 비트 카운터 유닛 내 카운트값과 가산하는 단계;

상기 캐시 콘트롤러가 상기 비트 카운터 유닛 내 가산된 카운트값을 저장하는 단계;

상기 가산하는 단계와 상기 저장하는 단계를 소정 횟수 동안 반복하였는지 판단하는 단계; 및

상기 판단하는 단계 후, 다음 구간에 적용할 해시 함수를 결정하고, 캐시 내 데이터를 반도체 장치로 플러시 하

는 단계

를 포함하는 비트 카운터를 이용하는 컴퓨팅 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 비트 카운터를 이용하는 컴퓨팅 시스템 및 방법에 관한 것으로, 더욱 상세하게는 Cache에 비트 카운터 기법을 적용하여 주소를 해싱하여 인덱스 비트를 결정함으로써 일부 set로 집중된 접근으로 인한 cache conflict miss를 방지하여 반도체 장치로의 접근 횟수를 줄이며 동시에 반도체 장치의 동일 bank로의 연속적인 접근을 방지하여 반도체 장치의 bank conflict를 감소시킬 수 있는 비트 카운터를 이용하는 컴퓨팅 시스템 및 방법에 관한 것이다.

배 경 기 술

[0003] 일반적으로 컴퓨팅 시스템에서 CPU와 반도체 장치 사이에는 cache가 배치되고, CPU는 cache를 통해 반도체 장치에 Read/Write 요청을 수행한다. 그런데 cache에는 conflict miss에 의한 반도체 장치 request가 자주 발생된다.

[0004] 한편, cache는 식별 가능한 복수의 세트를 포함하고, 개별 세트는 식별 가능한 복수의 블록을 포함하며, 개별 블록은 식별 가능하도록 각각의 tag를 포함하도록 구성된다.

[0005] 이때, conflict miss라 함은 cache로의 접근이 cache 내 일부 세트에 집중되어 해당 세트에 대한 cache miss가 증가하며 동일한 index bit를 가지는 데이터의 eviction 되는 현상을 말한다.

[0006] 이러한 경우, cache miss의 증가로 인해 Read/Write 처리 시간의 지연뿐만 아니라 반도체 장치의 동일한 bank로의 request 발생으로 인하여 메모리 지연 또한 크게 증가하여 전체 시스템 성능에 악영향을 끼친다.

선행기술문헌

특허문헌

[0008] (특허문헌 0001) US 9092337 Apparatus, system, and method for managing eviction of data

(특허문헌 0002) US 9529730 Methods for cache line eviction

(특허문헌 0003) US 9886395 Evicting cached stores

발명의 내용

해결하려는 과제

[0009] 본 발명은 비트 카운터를 기반으로 address를 hashing하여 index bit를 결정함으로써 cache에서 일부 set로 집중된 접근으로 인한 cache conflict miss를 방지하여 반도체 장치로의 접근 횟수를 줄이며 동시에 반도체 장치의 동일 bank로의 연속적인 접근을 방지하여 반도체 장치의 bank conflict를 감소시킬 수 있는 비트 카운터를 이용하는 컴퓨팅 시스템 및 방법을 제공한다.

과제의 해결 수단

[0011] 본원의 제1 발명에 따른 비트 카운터를 이용하는 컴퓨팅 시스템은, 호스트 장치; 상기 호스트 장치의 데이터를 임시 저장하고, 복수의 세트를 포함하는 Cache; 상기 호스트 장치로부터 복수 비트의 캐시 어드레스를

수신하고, 복수의 비트 카운터 유닛을 이용하여 상기 캐시 어드레스를 계수함으로써 상기 Cache의 해시 함수를 결정하는 캐시 콘트롤러; 반도체 장치; 및 상기 캐시 콘트롤러로부터 상기 캐시 어드레스를 수신하여 반도체 장치 어드레스와 매핑하는 메모리 콘트롤러를 포함할 수 있다.

[0012] 또한, 본원의 제2 발명에 따른 비트 카운터를 이용하는 컴퓨팅 시스템은, 호스트 장치; 상기 호스트 장치의 데이터를 임시 저장하고, 복수의 세트를 포함하는 Cache; 반도체 장치: 및 상기 호스트 장치로부터 복수 비트의 캐시 어드레스를 수신하고, 복수의 비트 카운터 유닛을 이용하여 상기 캐시 어드레스를 계수함으로써 상기 Cache의 해시 함수를 결정하고, 상기 캐시 어드레스와 반도체 장치 어드레스를 매핑하는 캐시 콘트롤러를 포함할 수 있다.

[0013] 또한, 본원의 제3 발명에 따른 비트 카운터를 이용하는 컴퓨팅 방법은, 캐시 콘트롤러가 비트 카운터 유닛을 리셋시키는 단계; 상기 캐시 콘트롤러가 입력되는 어드레스 비트를 상기 비트 카운터 유닛 내 카운트값과 가산하는 단계; 상기 캐시 콘트롤러가 상기 비트 카운터 유닛 내 가산된 카운트값을 저장하는 단계; 상기 가산하는 단계와 상기 저장하는 단계를 소정 횟수 동안 반복하였는지 판단하는 단계; 및 상기 판단하는 단계 후, 다음 구간에 적용할 해시 함수를 결정하고, 캐시 내 데이터를 반도체 장치로 플러시 하는 단계를 포함할 수 있다.

발명의 효과

[0015] 본 발명에 따르면, 비트 카운터를 기반으로 address를 hashing하여 index bit를 결정함으로써 cache에서 일부 set로 집중된 접근으로 인한 cache conflict miss를 방지하여 반도체 장치로의 접근 횟수를 줄이며 동시에 반도체 장치의 동일 bank로의 연속적인 접근을 방지하여 반도체 장치의 bank conflict를 감소시킬 수 있는 효과가 있다.

도면의 간단한 설명

[0017] 도 1은 본 발명의 일실시예에 따른 비트 카운터를 이용하는 컴퓨팅 시스템의 전체 블록도,

도 2는 본 발명의 다른 실시예에 따른 비트 카운터를 이용하는 컴퓨팅 시스템의 전체 블록도,

도 3은 본 발명의 일실시예에 따라 Cache에 비트 카운터를 적용하기 전과 후의 request miss 분포를 나타내는 도면,

도 4는 본 발명의 일실시예에 따른 Cache 내 address bits에 대한 access request 처리시 비트 카운터 값 변화를 설명하기 위한 도면,

도 5는 본 발명의 일실시예에 따라 비트 카운터에서 해시 함수의 재설정을 설명하는 도면, 및

도 6은 본 발명의 일실시예에 따른 해시 함수 재설정 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0018] 본 발명의 이점 및 특징, 그리고 그것들을 달성하는 방법은 첨부되는 도면과 함께 상세하게 후술되어 있는 실시예를 참조하면 명확해질 것이다. 그러나 본 발명은 이하에서 개시되는 실시 예에 한정되는 것이 아니라 서로 다른 다양한 형태로 구현될 수 있으며, 단지 본 실시 예는 본 발명의 개시가 완전하도록 하고, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 발명의 범주를 완전하게 알려주기 위해 제공되는 것이며, 본 발명은 청구항의 범주에 의해 정의될 뿐이다.

[0019] 본 명세서에서 사용된 용어는 실시 예들을 설명하기 위한 것이며 본 발명을 제한하고자 하는 것은 아니다. 본 명세서에서, 단수형은 문구에서 특별히 언급하지 않는 한 복수형도 포함한다. 명세서에서 사용되는 '포함한다(comprises)' 및/또는 '포함하는(comprising)'은 언급된 구성요소, 단계, 동작 및/또는 소자는 하나 이상의 다른 구성요소, 단계, 동작 및/또는 소자의 존재 또는 추가를 배제하지 않는다.

[0020] 하나의 소자(elements)가 다른 소자와 '접속된(connected to)' 또는 '커플링된(coupled to)' 이라고 지칭되는 것은, 다른 소자와 직접 연결 또는 커플링 된 경우 또는 중간에 다른 소자를 개재한 경우를 모두 포함한다. 반면, 하나의 소자가 다른 소자와 '직접 접속된(directly connected to)' 또는 '직접 커플링된(directly coupled to)'으로 지칭되는 것은 중간에 다른 소자를 개재하지 않은 것을 나타낸다. '및/또는'은 언급된 아

이템들의 각각 및 하나 이상의 모든 조합을 포함한다.

[0021] 명세서 전문에 걸쳐 동일한 참조 부호는 동일한 구성 요소를 지칭한다. 따라서, 동일한 참조 부호 또는 유사한 참조 부호들은 해당 도면에서 언급 또는 설명되지 않았더라도, 다른 도면을 참조하여 설명될 수 있다. 또한, 참조 부호가 표시되지 않았더라도, 다른 도면들을 참조하여 설명될 수 있다.

[0023] 본 발명에서 캐시 축출(cache eviction)이라 함은 Cache의 특정 set에서 더 이상 저장할 공간이 없을 경우, 기존에 저장된 block의 데이터를 반도체 장치로 축출하는 동작을 의미한다.

[0024] 또한, 본 발명에서 플러싱(flush, flushing)이라 함은 Cache의 데이터를 반도체 장치에 덮어 쓰게 하고, Cache를 초기화하는 동작을 의미한다.

[0026] 도 1은 본 발명의 일실시예에 따른 비트 카운터를 이용하는 컴퓨팅 시스템의 전체 블록도이다.

[0027] 본 발명의 일실시예에 따른 비트 카운터를 이용하는 컴퓨팅 시스템은 호스트 장치(110), 캐시 콘트롤러(120), 캐시(130), 메모리 콘트롤러(140), 및 반도체 장치(150)를 포함한다.

[0028] 호스트 장치(110)는 호스트, 서버, 저장 영역 네트워크(SAN: storage area network)의 저장 콘트롤러, 워크스테이션, 개인용 컴퓨터, 랩터 컴퓨터, 핸드헬드 컴퓨터, 수퍼컴퓨터, 컴퓨터 클러스터, 네트워크 스위치, 라우터, 또는 응용기기, 데이터베이스 또는 저장 응용기기, 데이터 획득 또는 데이터 캡처 시스템, 진단 시스템, 테스트 시스템, 로봇, 포터블 전자 기기, 무선 기기 등일 수 있다.

[0029] 본 발명의 일실시예에 따르면, 호스트 장치(110)는 캐시 콘트롤러(120)와 통신할 수 있다. 캐시 콘트롤러(120)는 호스트 장치(110)로부터 read/write request와 read/write request를 위한 cache address(Cache Addr)를 수신하고, Cache Addr를 비트 카운터 유닛(121, BC: Bit Counter Unit)를 이용하여 변경된 cache index를 결정하고, 데이터를 Cache(130)에 캐싱하고, 메모리 콘트롤러(140)와 통신하여 캐싱된 데이터를 Cache(130)에서 반도체 장치(150)로 축출(evict) 할 수 있다. 여기서, Cache(130)에서 반도체 장치(150)로 축출한다는 것은 Cache(130)의 소정 Cache Addr에 저장된 데이터를 해당 Cache Addr에 매핑되는 반도체 장치(150) 내 반도체 장치 address(Semi Addr)에 저장하고, Cache Addr에 저장된 데이터를 삭제하는 것을 의미한다. 한편, 일반적으로 Cache의 저장 용량은 반도체 장치의 저장 용량에 비해 현저히 작고, 반도체 장치의 일부 영역에는 Cache 내 다수의 address와 동일한 address들이 존재한다.

[0030] 본 발명의 일실시예에 따르면, 캐시 콘트롤러(120), Cache(130), 메모리 콘트롤러(140) 및 반도체 장치(150)가 하나의 모듈 형태로 구현될 수 있다. 여기서, 반도체 장치(150)는 DRAM일 수 있다.

[0031]

[0032] 도 2는 본 발명의 다른 실시예에 따른 비트 카운터를 이용하는 컴퓨팅 시스템의 전체 블록도이다.

[0033] 본 발명의 다른 실시예에 따르면, 캐시 콘트롤러(220) 내에 비트 카운터 유닛(221)이 구현될 수 있다.

[0034] 본 발명의 일실시예에 따르면, 호스트 장치(110)는 캐시 콘트롤러(220)와 통신할 수 있다. 캐시 콘트롤러(220)는 호스트 장치(110)로부터 read/write request와 cache address(Cache Addr)를 수신하고, 호스트 장치(110)로부터 수신되는 cache address(Cache Addr)를 비트 카운터 유닛(221)을 이용하여 변경된 cache index를 결정하고, 데이터를 Cache(130)에 캐싱하고, 캐시 처리된 데이터를 Cache(130)에서 반도체 장치(150)로 축출할 수 있다.

[0035] 본 발명의 일실시예에 따르면, Cache(130) 및 반도체 장치(150)가 하나의 모듈 형태로 구현될 수 있다.

[0036] 본 발명의 다른 실시예에 따르면, Cache(130), 캐시 콘트롤러(220) 및 반도체 장치(150)가 하나의 모듈 형태로 구현될 수 있다. 여기서, 반도체 장치(150)는 DRAM일 수 있다.

[0038] 도 3a는 Cache에 비트 카운터를 사용하기 전의 request miss 분포를 나타내는 도면이고, 도 3b는 본 발명의 일실시예에 따라 Cache에 비트 카운터를 사용한 후의 cache miss 분포를 나타내는 도면이다.

[0039] 예컨대, 도 3a에 도시된 Cache(135)에 비트 카운터를 사용하기 전의 request miss 분포를 보면, index bits를,

하위 2비트(b1, b0)로 고정하여 access request를 처리하는 경우, 특정 set로의 접근이 집중되는 것을 볼 수 있다. 즉, 전체 6회의 접근 중 5회 동안 set 00에 접근하게 된다. 이로 인해 4회의 request miss가 발생하여 반도체 장치에 4회의 requests가 발생한다. 이때, 반도체 장치 request(Semi request)를 발생시킨 cache request의 index는 모두 같고 cache request의 index bit는 Semi request의 채널 및 bank에 매핑되므로, 4회의 Semi request는 모두 동일 채널의 동일 bank에 발생한다.

[0040] 여기서, 메모리 콘트롤러(145)는 Cache Addr와 Semi Addr를 매핑하여 반도체 장치 내 소정 뱅크에 데이터를 쓰거나 소정 뱅크로부터 데이터를 읽어들인다. 즉, Cache Addr의 제1 비트(b0)를 반도체 장치의 채널에 매핑하고, Cache Addr의 제2 비트(b1)를 반도체 장치의 뱅크에 매핑한다. 예컨대, Cache Addr 내 index bits의 제1 비트(b0)가 "0"이므로 반도체 장치 내 제1 채널(Ch0), 그리고 Cache Addr 내 index bits의 제2 비트(b1)가 "0"이므로 제1 채널(Ch0) 내 제1 뱅크(Bank 0)에 매핑하고, 제3 및 제4 비트(b2, b3)의 tag bits를 제1 채널(Ch0) 내 제1 뱅크(Bank 0)의 해당 Row/Col에 매핑한다.

[0042] 하지만, 도 3b에 도시된 바와 같이, 본 발명의 일실시예에 따라 Cache에 비트 카운터를 사용하면, access requests로 동일한 Cache Addr가 입력되는 경우에도, set별 접근 집중도를 완화시킬 수 있다.

[0043] 본 발명에 따르면, Cache(130)는 4개의 sets(set 00, set 01, set 10, set 11)를 포함하고, 개별 set는 복수의 블록(block0 ~ 3)을 포함하고, 개별 블록은 각각의 태그(Tag 0, Tag 1, ...)를 포함하고, 개별 태그마다 소정 크기의 데이터(Data0, Data1, ...)를 저장할 수 있다. 도 3b에서는 제2 블록 내지 제4 블록(block1, block 2, block3)을 생략하였다.

[0044] 캐시 콘트롤러(120)가 호스트 장치(110)로부터 read/write request와 read/write request를 위한 Cache Addr(b3, b2, b1, b0)를 수신하면, 캐시 콘트롤러(120) 내 4개의 비트 카운터 유닛(121: BC0, BC1, BC2, BC3)은 각각 순차적으로 수신되는 Cache Addr를 소정 횟수 동안 계수한다. 여기서 4 bits의 Cache Addr 중 임의의 2 bits를 index로 사용하고, 나머지 2 bits를 tag로 사용할 수 있다.

[0045] 예컨대, 수신되는 Cache Addr가 4 bits이므로, 4개의 비트 카운터 유닛(121)이 필요하고, 비트 카운터 유닛(121)이 2^{10} 횟수 동안 계수한다면, 개별 비트 카운터 유닛은 10개의 비트 카운터로 구현될 수 있다. 여기서 10개의 비트 카운터 내 중 1개는 부호 표시용이고, 나머지 9개는 비트 카운트용이다.

[0046] 그리고, 캐시 콘트롤러(120)는 개별 비트 카운터 유닛(121)이 계수한 4비트에 대하여 각각의 절대치를 생성하고, 가장 작은 절대치 2개를 묶어 조합을 생성하고, 이를 Cache의 index bit로 사용하도록 해시 함수를 생성한다.

[0047] 여기서, 해시 함수(hash function)라 함은 캐시 콘트롤러(120)가 호스트 장치(110)로부터 수신되는 read/write request와 Cache Addr를 이용하여 index bits를 찾아내기 위한 인자(factor)를 의미한다. 한편, 절대치를 생성하는 이유는 도 4에서 설명하기로 한다.

[0048] 이와 같이 소정 횟수 동안 Cache Addr를 계수하고, 계수한 절대치 중 가장 작은 값의 조합이 index로 사용되도록 해시 함수를 선택하게 되면, 소정 횟수 동안 해당 어드레스 비트의 "0"과 "1"의 분포가 상대적으로 가장 다양하였음을 의미한다. 이에 따라 Cache 내 set를 고르게 사용할 수 있는 효과가 있다.

[0049] 예컨대, 각 비트에 대응하는 제1, 2, 3, 4 비트 카운터(BC1, BC2, BC3, BC4)에 각각 계수된 값 "-2", "2", "-6", "-5"에 절대치를 취하고, 절대치가 가장 작은 비트 2개는 제 3 비트(b2), 제 4 비트(b3)이다.

[0050] 따라서 비트 조합(b2, b3)을 해시 함수로 선택한다. 이때, 캐시 콘트롤러(120)는 제3 및 제4 비트(b2, b3)를 index bits로, 제1 및 제2 비트(b0, b1)를 tag bits로 이용한다.

[0051] 캐시 콘트롤러(120)에 제1 access request의 Cache Addr "0000"이 입력되면, 캐시(130)에 set 00의 index bit(00)과 tag bit(00)이 일치하므로 해당 request를 수행한다.

[0052] 그런데 캐시 콘트롤러(120)에 제2 access request의 Cache Addr "0100"이 입력되면, 캐시(130)에 set 01의 index bit(01)은 일치하는 반면, tag bit(01)로 불일치하므로 해당 request를 수행하지 못한다. 이때, 캐시 콘트롤러(120)는 메모리 콘트롤러(140)에 제2 access request를 Semi request로 출력하고, 메모리 콘트롤러(140)는 매핑 스케쥴러(141)를 이용하여 Cache Addr를 Semi Addr와 매핑하고 Transaction queue(143-1, 143-2)로 입력되는 제1 Semi request를 수행한다. 즉, Transaction queue(143-1, 143-2)에 제1 Semi request의 Cache

Addr "0100"이 수신되면, 메모리 콘트롤러(140)는 매핑 스케쥴러(141)를 이용하여 제1 비트(b0) "0"을 제1 채널(Ch0)에, 제2 비트(b1) "0"을 제1 채널 내 제1 뱅크(Bank 0)에 각각 매핑하고, 제3 및 제4 비트(b2, b3)를 제1 채널 내 제1 뱅크(Bank 0)의 해당하는 Row/Col에 매핑하여 제1 Semi request를 수행한다.

[0053] 캐시 콘트롤러(120)에 입력되는 제3 내지 제6 access request도 동일한 방식으로 수행되는바, 구체적인 설명은 생략하기로 한다.

[0055] 이와 같은 방식으로 수행되면, 제1 내지 제6 access requests가 set 00으로 2회, set 01로 2회, set 11로 2회가 각각 발생하여 세트별 접근이 분산됨을 알 수 있다. 이로 인해 3회의 conflict miss가 발생하므로 반도체 장치로의 request도 3회로 줄어드는 것을 알 수 있다. 이때, Semi request를 발생시킨 address의 cache index가 같더라도 index bit이 Semi request의 채널 및 bank에 매핑되지 않으므로, 동일한 set에서 발생한 Semi request가 서로 다른 채널 및 bank로 매핑될 수 있다.

[0057] 도 4는 본 발명의 일실시예에 따른 Cache 내 address에 대한 access request 처리시 세트용 비트 카운터 값 변화를 설명하기 위한 도면이다.

[0058] 본 발명에 따르면, 각각의 Cache Addr를 계수하는 방법은 Cache Addr가 "1"이면 (+)1로 계수하고, Cache Addr가 "0"이면 (-)1로 계수한다.

[0059] 초기에는 제4 내지 제1 비트 카운터(BC3 ~ BC0)에 "0 0 0 0" 값이 저장되어 있다가, 제1 address(Addr0) "0 1 1 0"이 입력되면, 위와 같은 본 발명에 따른 Cache Addr 계수 방법에 따라 제4 내지 제1 비트 카운터(BC3 ~ BC0)는 "-1 1 1 -1"로 계수한다.

[0060] 이후 제2 address(Addr1) "0 1 0 1"이 입력되면, 제4 내지 제1 비트 카운터(BC3 ~ BC0)는 "-2 2 0 0" 로 계수한다.

[0061] 이러한 방식으로, 제m address(Addr(m-1)) "0 0 0 1"이 입력되면, 제4 내지 제1 비트 카운터(BC3 ~ BC0)는 "9 6 0 4"로 계수한다.

[0062] 캐시 콘트롤러(120)는 계수한 4비트에 대하여 각각의 절대치를 취하고, 4개의 절대치를 이용하여 6개의 비트 조합을 생성하며, 6개의 비트 조합 각각에 대하여 가산하고, 이들 가운데 절대치가 가장 작은 비트 조합(b0, b1)을 선택한다. 결국, 비트 조합(b1, b0)을 Cache의 index bit으로 사용하도록 해시 함수를 선택한다.

[0063] 그리고 4 bits의 Cache Addr 중 나머지 2 bits(b3, b2)를 tag로 사용한다.

[0064] 이와 같이 캐시 콘트롤러(120)는 소정 횟수 동안 입력되는 Cache Addr를 계수하여 절대치가 가장 작은 비트의 조합을 index bit로 사용하도록 해시 함수를 선택함으로써 Cache 내 set를 고르게 사용할 수 있는 효과가 있다.

[0066] 도 5는 본 발명의 일실시예에 따라 비트 카운터에서 해시 함수의 재설정을 설명하는 도면이다.

[0067] 본 발명에 따르면, 캐시 콘트롤러(120)는 4개의 비트 카운터(BC3 ~ BC0)를 이용하여 해시 함수를 소정 횟수마다 재설정하고, 특정 address를 새로운 index bit로 사용하도록 해시 함수를 결정하는 순간 Cache(130) 내 데이터들을 반도체 장치(150)로 플러싱(flush) 하여 Cache(130)를 삭제한다.

[0068] 즉, 캐시 콘트롤러(120)는 이전 구간 Previous Interval(i-1) 동안 비트 카운터 유닛에 카운트 값이 누적되면 현재 구간 Current Interval(i)에서 적용할 해시 함수를 재설정하고, Cache(130) 내 데이터를 반도체 장치(150)로 플러시(flush) 한다.

[0069] 이후 동일한 방식으로 캐시 콘트롤러(120)는 현재 구간 Current Interval(i) 동안 비트 카운터에 카운트 값이 누적되면 다음 구간 Next Interval(i+1)에서 적용할 해시 함수를 재설정하고, Cache(130) 내 데이터를 반도체 장치(150)로 플러시(flush) 한다.

[0070] 여기서, 소정 횟수마다 Cache의 데이터를 반도체 장치로 플러시 하는 이유는 다음과 같다.

[0071] 이전 구간 Previous Interval(i-1)에 적용되는 해시 함수와 현재 구간 Current Interval(i)에 적용되는 해시 함수가 다르거나 현재 구간 Current Interval(i)에 적용되는 해시 함수와 다음 구간 Next Interval(i+1)에 적

용되는 해시 함수가 다르게 되면 소정 데이터를 저장하는 Cache Addr와 Semi Addr가 불일치하게 된다.

[0072] 이러한 문제점을 해소하기 위하여 개별 구간마다 Cache Addr와 Semi Addr를 일치시키기 위해 매 구간이 시작될 때마다 Cache(130) 내 데이터를 반도체 장치(150)로 플러시(flush) 할 필요가 있다.

[0074] 도 6은 본 발명의 일실시예에 따른 해시 함수 재설정 흐름도이다.

[0075] 본 발명의 일실시예에 따르면, 캐시 콘트롤러(120)는 제1 내지 제4 비트 카운터(BC0 ~ BC3)를 리셋시킨다(S610).

[0076] 입력되는 어드레스 비트(Addr. bits)를 제1 내지 제4 비트 카운터(BC0 ~ BC3)내 카운트값과 가산하고(S620), 가산된 제1 내지 제4 비트 카운터(BC0 ~ BC3)내 카운트값을 저장한다(S630).

[0077] 단계 S620과 단계 S630를 소정 횟수(m) 동안 반복한다(S640).

[0078] 단계 S620과 단계 S630를 소정 횟수(m) 동안 반복한 것으로 판단되면(S640), 다음 구간에 적용할 해시 함수를 결정하고, Cache(130) 내 데이터를 반도체 장치(150)로 플러시(flush)한다.

[0080] 이상, 첨부된 도면을 참조하여 본 발명의 실시 예를 설명하였지만, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자는 본 발명이 그 기술적 사상이나 필수적인 특징을 변경하지 않고서 다른 구체적인 형태로 실시될 수 있다는 것을 이해할 수 있을 것이다. 그러므로 이상에서 기술한 실시 예에는 모든 면에서 예시적인 것이며 한정적이 아닌 것으로 이해해야만 한다.

부호의 설명

[0082] 110: 호스트 장치

120, 220: 캐시 콘트롤러

121, 221: 비트 카운터 유닛

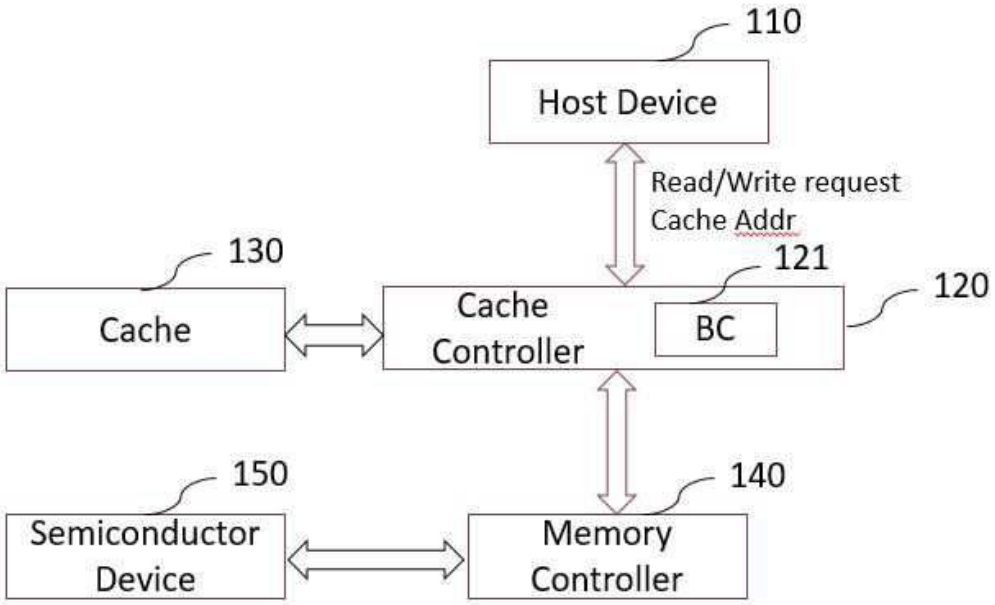
130: 캐시

140: 메모리 콘트롤러

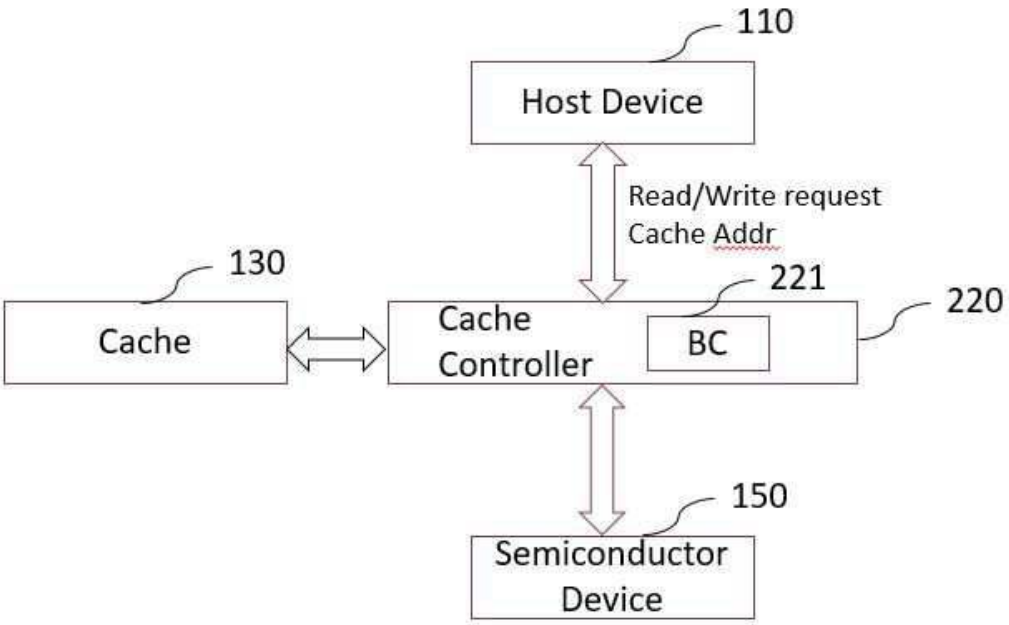
150: 반도체 장치

도면

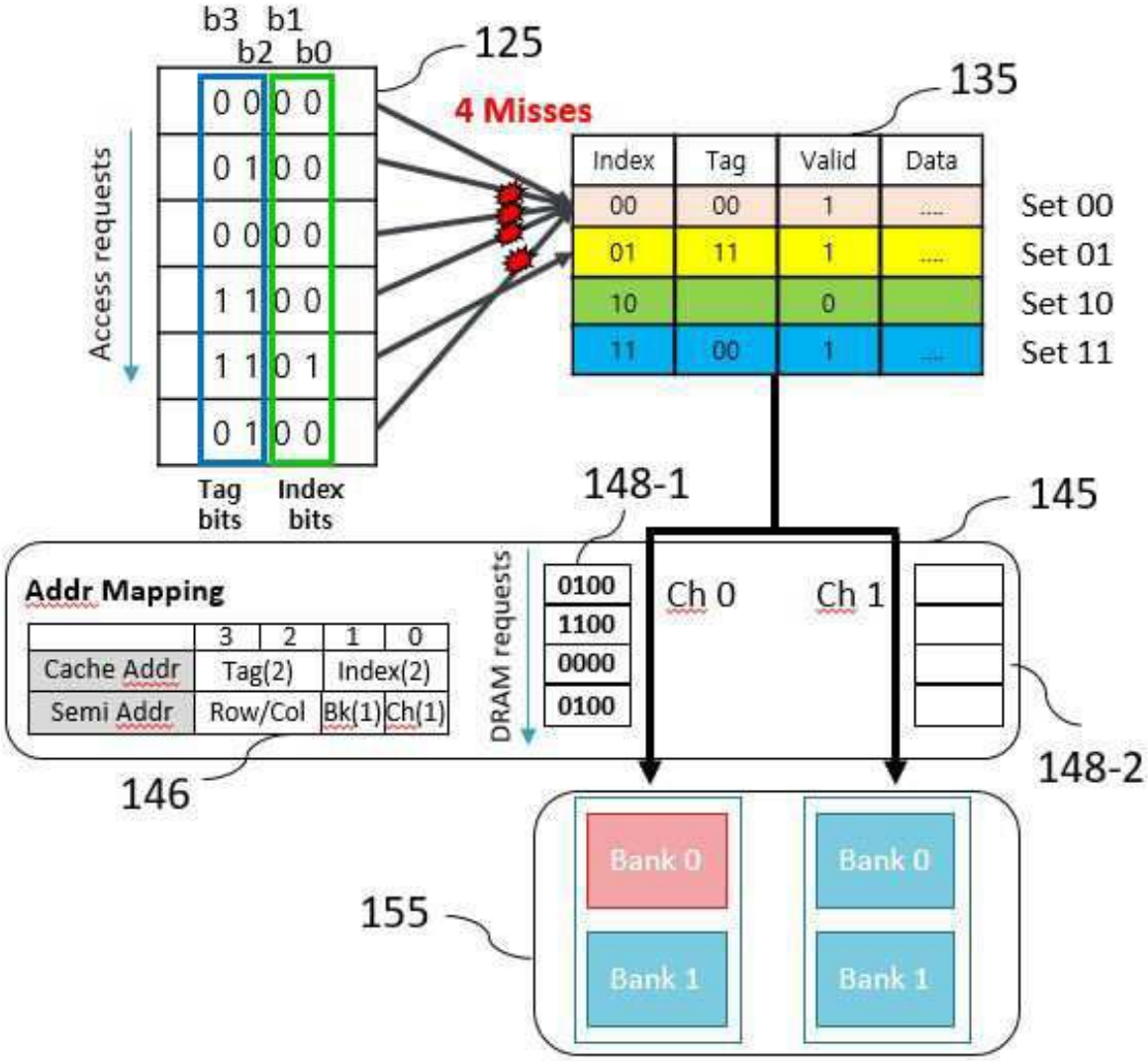
도면1


110
Host Device
Read/Write request
Cache Addr
130
Cache
121
Cache
Controller
BC
120
150
Semiconductor
Device
140
Memory
Controller

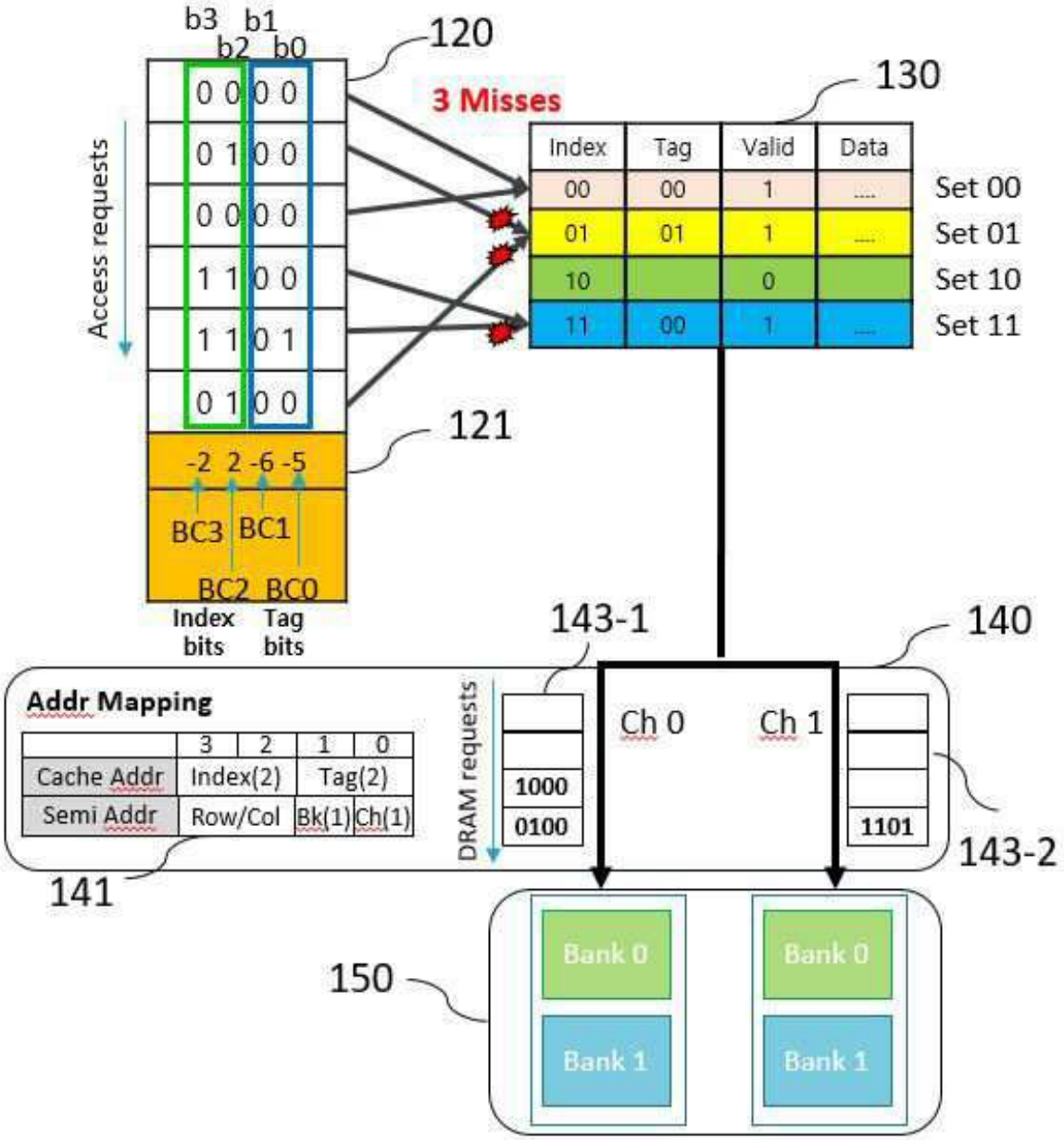
도면2


110
Host Device
Read/Write request
Cache Addr
130
Cache
221
Cache
Controller
BC
220
150
Semiconductor
Device

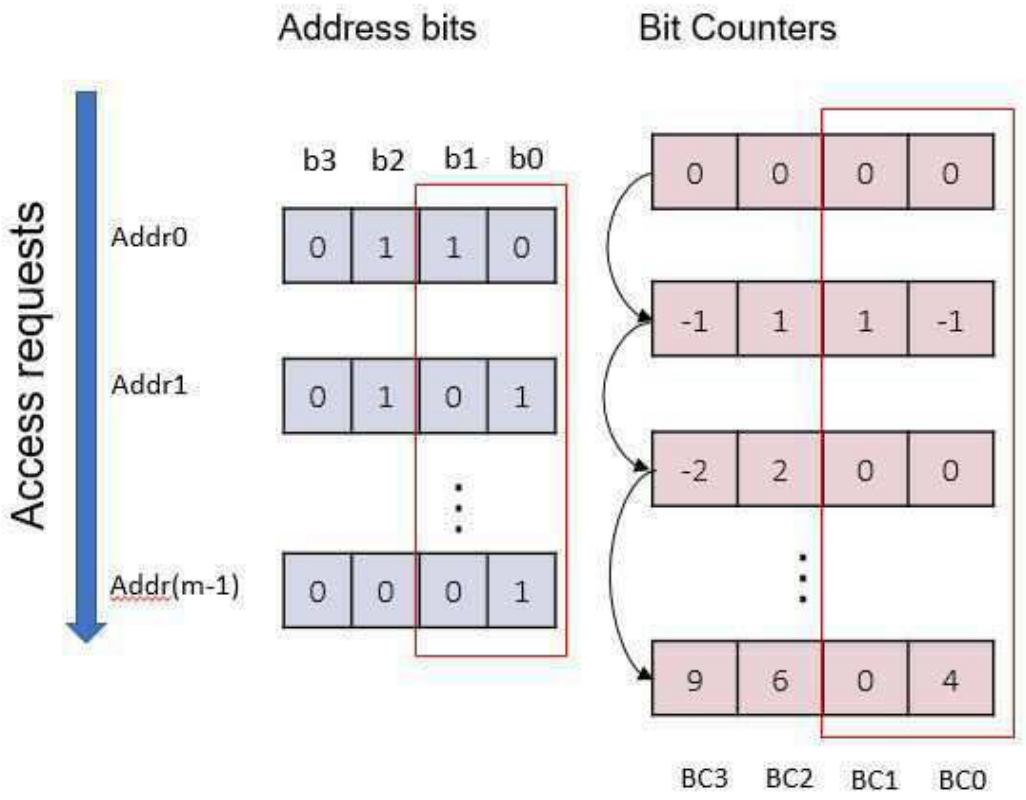
도면3a


b3 b1
b2 b0
125
4 Misses
135
Access requests
0 0 0 0
0 1 0 0
0 0 0 0
1 1 0 0
1 1 0 1
0 1 0 0
Tag bits
Index bits
Index
Tag
Valid
Data
00
00
1
01
11
1
10
0
11
00
1
Set 00
Set 01
Set 10
Set 11
148-1
145
Addr Mapping
3
2
1
0
Cache Addr
Tag(2)
Index(2)
Semi Addr
Row/Col
Bk(1)
Ch(1)
DRAM requests
0100
1100
0000
0100
Ch 0
Ch 1
148-2
146
155
Bank 0
Bank 1
Bank 0
Bank 1

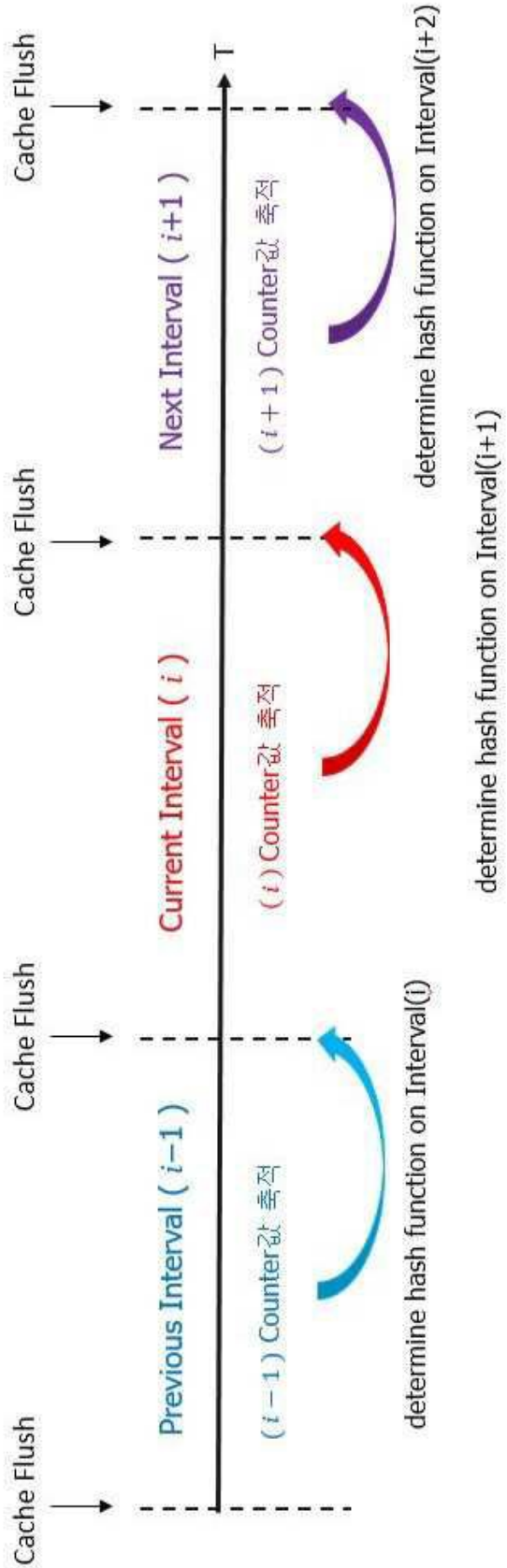
도면3b


b3 b1
b2 b0
120
3 Misses
130
Access requests
0 0 0 0
0 1 0 0
0 0 0 0
1 1 0 0
1 1 0 1
0 1 0 0
Index
Tag
Valid
Data
00
00
1
....
Set 00
01
01
1
....
Set 01
10
0
Set 10
11
00
1
....
Set 11
121
-2 2 -6 -5
BC3 BC1
BC2 BC0
Index bits
Tag bits
143-1
140
Addr Mapping
3 2 1 0
Cache Addr
Index(2)
Tag(2)
Semi Addr
Row/Col
Bk(1)
Ch(1)
DRAM requests
1000
0100
Ch 0
Ch 1
1101
143-2
141
150
Bank 0
Bank 1
Bank 0
Bank 1

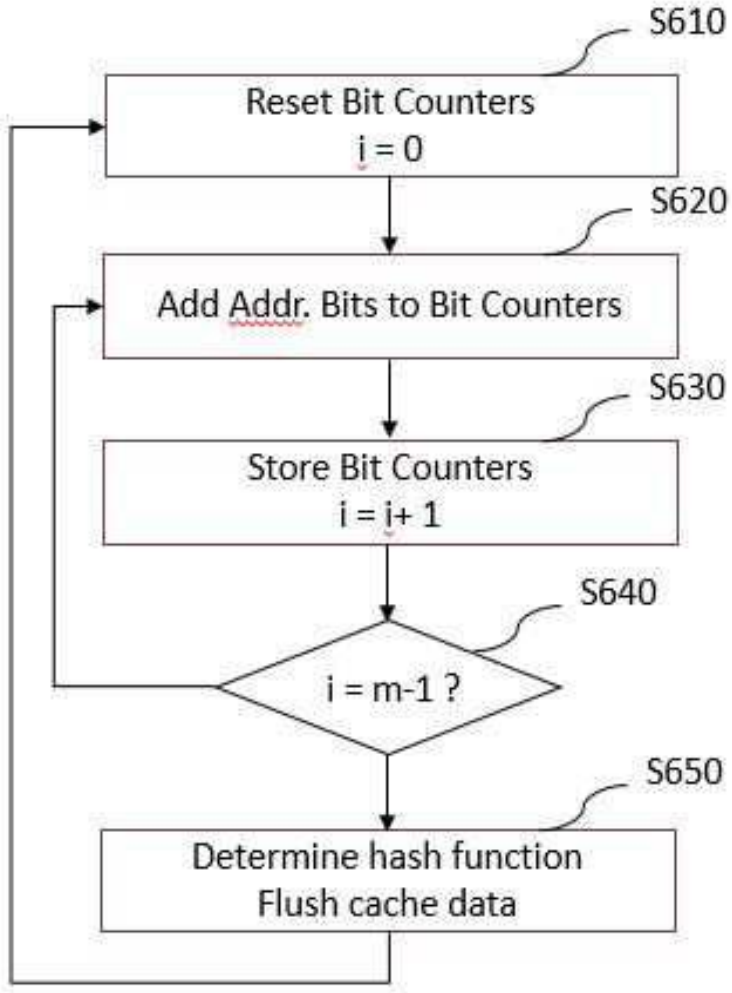
도면4


Address bits
Bit Counters
Access requests
b3 b2 b1 b0
Addr0
Addr1
Addr(m-1)
0 1 1 0
0 1 0 1
0 0 0 1
0 0 0 0
-1 1 1 -1
-2 2 0 0
9 6 0 4
BC3 BC2 BC1 BC0

도면5


Cache Flush
Cache Flush
Cache Flush
Cache Flush
T
Previous Interval (i−1)
Current Interval (i)
Next Interval (i+1)
(i − 1) Counter값 축적
(i) Counter값 축적
(i + 1) Counter값 축적
determine hash function on Interval(i)
determine hash function on Interval(i+1)
determine hash function on Interval(i+2)

도면6


S610
Reset Bit Counters
i = 0
S620
Add Addr. Bits to Bit Counters
S630
Store Bit Counters
i = i+ 1
S640
i = m-1 ?
S650
Determine hash function
Flush cache data