(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.[6]
G06F 13/14

(45) 공고일자 2005년05월24일
(11) 등록번호 10-0491435
(24) 등록일자 2005년05월17일

(21) 출원번호	10-1999-7002186	(65) 공개번호	10-2000-0036144
(22) 출원일자	1999년03월15일	(43) 공개일자	2000년06월26일
번역문 제출일자	1999년03월15일		
(86) 국제출원번호	PCT/US1997/016207	(87) 국제공개번호	WO 1998/11488
국제출원일자	1997년09월16일	국제공개일자	1998년03월19일

(81) 지정국

국내특허 : 알바니아, 아르메니아, 오스트리아, 오스트레일리아, 아제르바이잔, 보스니아 헤르체고비나, 바르바도스, 불가리아, 브라질, 벨라루스, 캐나다, 스위스, 중국, 쿠바, 체코, 독일, 덴마크, 에스토니아, 스페인, 핀란드, 영국, 그루지야, 헝가리, 이스라엘, 아이슬랜드, 일본, 케냐, 키르키즈스탐, 북한, 대한민국, 카자흐스탄, 세인트루시아, 스리랑카, 리베이라, 레소토, 리투아니아, 룩셈부르크, 라트비아, 몰도바, 마다가스카르, 마케도니아공화국, 몽고, 말라위, 멕시코, 노르웨이, 뉴질랜드, 슬로베니아, 슬로바키아, 타지키스탄, 투르크맨, 터어키, 트리니아드토바고, 우크라이나, 우간다, 우즈베키스탄, 베트남, 폴란드, 포르투칼, 루마니아, 러시아, 수단, 스웨덴, 싱가포르, 시에라리온, 세르비아 앤 몬테네그로, 가나, 짐바브웨,

AP ARIPO특허 : 케냐, 레소토, 말라위, 수단, 스와질랜드, 우간다, 가나, 짐바브웨,

EA 유라시아특허 : 아르메니아, 아제르바이잔, 벨라루스, 키르키즈스탐, 카자흐스탄, 몰도바, 러시아, 타지키스탄, 투르크맨,

EP 유럽특허 : 오스트리아, 벨기에, 스위스, 독일, 덴마크, 스페인, 프랑스, 영국, 그리스, 아일랜드, 이탈리아, 룩셈부르크, 모나코, 네덜란드, 포르투칼, 스웨덴, 핀란드,

OA OAPI특허 : 부르키나파소, 베닌, 중앙아프리카, 콩고, 코트디브와르, 카메룬, 가봉, 기니, 말리, 모리타니, 니제르, 세네갈, 차드, 토고,

(30) 우선권주장 08/714,750 1996년09월16일 미국(US)

(73) 특허권자
인텔 코오퍼레이션
미합중국 캘리포니아 산타클라라 미션 칼리지 블러바드 2200

(72) 발명자
장 스티븐 에스.
미합중국 캘리포니아주 91740 글렌도라 이스트 머틀 528

화이트 조지 피이.
미합중국 캘리포니아주 90814 롱 비치 하바나 애비뉴 533

보흐트 피트 디.
미합중국 콜로라도주 80303 볼더 센티니얼 트레일 5395

(74) 대리인
유미특허법인
송만호

심사관 : 손영태

(54) 다수의 시스템 버스를 가지는 컴퓨터 시스템 내의 메모리 일관성을 유지하기 위한 시스템 및 방법

요약

본 발명은 다중 시스템 버스(1, 2) 및 I/O 버스(30)를 공유된 주 메모리(132)에 상호연결하고, 시스템 내의 지연시간 및 대역폭에 의한 영향을 최소화하면서 캐시 일관성을 효율적으로 유지하기 위한 캐시 일관성, 다중버스, 다중처리 시스템 및 방법에 관한 것이다. 본 시스템은 캐시 메모리 일관성을 작은 양의 크로스 버스 트래픽으로 유지할 수 있도록 버스간 통신을 조정하는 일관성 필터(200)를 제공한다. 또한 본 시스템은 다중버스와 상호접속하는 다중포트화된 메모리 셀 영역을 제공한다.

대표도

도 1

색인어

캐시 일관성, 다중버스, 다중처리, 일관성 필터, 다중포트화

명세서

기술분야

본 발명은 다수의 시스템 버스를 가지는 컴퓨터 시스템 내의 메모리 일관성(memory coherency)을 유지하는 시스템 및 방법에 관한 것이며, 구체적으로 말하면 다중 시스템 버스 및 I/O 버스를 공유된 주 메모리에 상호연결하고, 시스템 내의 지연시간 및 대역폭에 의한 영향을 최소화하면서 캐시 일관성을 효율적으로 유지하기 위한 캐시 일관성, 다중버스, 다중처리 시스템 및 방법에 관한 것이다.

배경기술

단일 중앙 처리 장치(central processing unit; CPU)를 가지는 컴퓨터의 성능을 개선하기 위하여, 컴퓨터 설계자들은 다수의 CPU를 가지는 컴퓨터를 개발해왔다. 종종 이러한 다중처리(multiprocessing) 컴퓨터의 CPU들은 하나의 공통 버스를 통해 서로 연결되며, 또한 시스템의 주 메모리와도 연결된다. 그러나 최근에는 CPU 성능이 버스 성능 기술보다 빠른 속도로 증가하고 있으며, 이러한 고속의 내부 CPU 성능은 보다 넓은 외부 대역폭을 요구한다. 다시 말해, 증가된 CPU 성능을 지원하기 위해서는 공통 버스 상에서 전송되는 데이터의 양을 증가시켜야 한다. 따라서 공통 버스와 연결될 수 있는 CPU의 개수는 CPU를 지원하기 위해 필요한 대역폭과 공통 버스의 전체 대역폭에 의해 제한된다.

다중처리 시스템에서 각 프로세서에 의해 요구되는 버스 대역폭을 감소시키기 위한 하나의 방법은 각 프로세서와 공통 버스 사이에 캐시 유닛(cache unit)을 두는 것이다. 프로세서의 연관된 캐시 유닛에 데이터가 로드되면, 이 프로세서는 공통 버스를 사용하지 않고서도 캐시 유닛 내의 데이터에 액세스할 수 있다. 일반적으로 프로세서가 캐시 유닛으로부터 데이터를 얻는 경우, 제한된 대역폭의 공통 버스를 통해 보다 적은 양의 데이터가 전송된다.

많은 경우에, 프로세서는 특정 데이터 값을 수회 수정하며, 따라서 데이터 값이 수정될 때마다 이 데이터 값을 주 메모리에 다시 재기록(rewrite)할 필요가 있다. 그러나 수정된 데이터 값을 주 메모리에 다시 재기록하는 것은 프로세서를 지원하기 위해 필요한 버스 대역폭의 양을 증가시킨다. 그러므로 기록 동작의 개수를 감소시킬 수 있으면, 프로세서를 지원하기 위해 필요한 버스 대역폭을 감소시킬 수 있다.

기록 동작의 개수를 감소시키는 캐시 유닛의 한 형태를 "재기록(write-back) 캐시라 한다. 재기록 캐시(write-back cache)는 수정된 데이터 값을 임시로 저장하며, 따라서 데이터 값을 주 메모리에 재기록하는데 필요한 버스 트랜잭션(bus transaction)의 수를 감소시킨다. 예를 들어, 프로세서는 주 메모리에 데이터를 재기록하지 않고서도 재기록 캐시 내의 데이터 값을 수회 수정할 수 있다. 재기록 캐시는 수정된 데이터가 결과적으로 주 메모리에 재기록되었음을 보장한다.

재기록 캐시는 다중처리 시스템에 의해 요구되는 전체 버스 대역폭을 감소시키는 데 있어서는 매우 효율적일 수 있지만, 재기록 캐시는 불행하게도 메모리 일관성 문제점을 발생시킨다. 예를 들어, 각 재기록 캐시는 데이터 값에 대한 그 자신의 복사본을 포함한다. 이러한 경우 하나 이상의 프로세서가 데이터 값을 독립적으로 수정하면, 동일한 데이터 값에 대하여 서로 다른 버전이 하나 이상의 재기록 캐시 내에 존재할 수 있다. 이것은 잘못된 동작(erroneous operation)을 발생시키며, 따라서 일부 메커니즘에서는 모든 프로세서가 항상 모든 데이터 값에 대하여 일관성(consistency)을 유지하고 있음을 확인해야 한다.

예를 들어, 프로세서가 어느 데이터 값을 수정하면, 수정된 데이터 값은 주 메모리에 재기록되기 전에 재기록 캐시에 존재한다. 이러한 예에서, 재기록 캐시가 수정된 데이터 값을 주 메모리에 재기록할 때까지, 주 메모리 및 다른 캐시 유닛은 이 데이터 값에 대한 쓸모 없는 복사본(stale copy)을 가지게 된다. 그러나 데이터 완전성(data integrity)을 유지하기 위해, 데이터 값을 요청하는 다른 프로세서는, 쓸모 없는 데이터 값이 아닌 최신 버전(up-to-date version)의 데이터 값을 얻어야 한다.

모든 프로세서가 모든 데이터 값에 대하여 일관성을 가지고 있는 지를 확인하는 절차를 캐시 일관성(cache coherency)이라 한다. 캐시 일관성을 구현하기 위한 대중적이고 성공적인 하나의 방법은 "스누핑 동작(snooping operation)"이라 불리는 동작에 의존한다. 다양한 형태의 스누핑 동작이 존재하지만, 기본적으로 캐시 유닛 내의 스누핑 동작은 공통 버스 상

의 버스 트랜잭션을 모니터한다. 스누핑 동작은 어느 트랜잭션이 캐시 유닛 내의 내용에 영향을 미치는 지를 식별하거나, 어느 트랜잭션이 캐시 유닛 내에 존재하는 수정된 데이터 값과 연관되어 있는 지를 식별한다. 일반적으로 스누핑 동작은 모든 프로세서 및 프로세서에 연관된 캐시 유닛이 공통 버스를 공유할 것을 요구한다. 공통 버스를 공유함으로써 캐시 유닛은 버스 트랜잭션을 모니터할 수 있으며, 특정 캐시 유닛이 수정된 데이터 값을 가지고 있는 경우 버스 트랜잭션과 인터페이스할 가능성이 있다.

또한 캐시 일관성 방법은 일반적으로 캐시 유닛 내의 특정 데이터 값이 무효화(invalid) 상태인지, 수정되었는지, 공유되었는지, 또는 배타적으로 점유(exclusively owned)인지 등의 여부를 표시하는 일관성 상태 정보(coherency status information)를 사용한다. 다수의 캐시 일관성 방법이 존재하지만, 2개의 대중적인 방법에는 MESI 캐시 일관성 프로토콜과 MOESI 캐시 일관성 프로토콜이 있다. MESI는 수정(Modified), 배타적(Exclusive), 공유(Shared), 및 무효화(Invalid) 상태의 두문자이며, MOESI는 수정(Modified), 점유(Owned), 배타적(Exclusive), 공유(Shared), 및 무효화(Invalid) 상태의 두문자이다.

이들 상태의 의미는 각 실행마다 다르다. 광범위하게 말하면, 수정 상태는 일반적으로 특정 캐시 유닛이 수정된 특정 데이터 값을 가지고 있음을 의미한다. 배타적 상태 및 점유 상태는 일반적으로 특정 캐시 유닛 데이터 값에 대한 복사본을 수정할 수 있음을 의미한다. 공유 상태는 일반적으로 데이터 값의 복사본이 서로 다른 캐시 유닛 내에 존재할 수 있음을 의미하며, 무효 상태는 캐시 유닛 내의 데이터 값이 무효 상태임을 의미한다.

동작 시에, 캐시 유닛은 버스 동작을 스누핑하고, 일관성 상태 정보를 사용하여 캐시 일관성을 확인한다. 예를 들어, 제1 캐시 유닛을 구비하는 제1 프로세서가 특정 데이터 값을 얻기를 원한다고 가정한다. 또한 제2 캐시 유닛을 구비하는 제2 프로세서가 수정된 버전의 데이터 값을 포함하고 있다고 가정한다. 일관성 상태 정보는 제2 캐시 유닛 내의 데이터 값이 수정된 상태임을 표시한다.

이 예에서, 제1 프로세서는 데이터 값을 얻기 위해 판독 버스 요청(read bus request)을 개시한다. 제2 캐시 유닛은 이러한 판독 버스 요청을 스누핑하고, 자신이 수정된 버전의 데이터 값을 포함하고 있는 지의 여부를 결정한다. 그 후 제2 캐시 유닛이 개입하여, 공통 버스를 통해 수정된 데이터 값을 제1 프로세서로 전달한다. 시스템에 따라, 수정된 데이터 값은 이와 동시에 주 메모리에 기록될 수도 있으며, 기록되지 않을 수도 있다.

다른 예로서, 제1 프로세서가 특정 데이터 값을 배타적으로 소유하기를 원한다고 가정한다. 또한 제2 캐시 유닛이 수정되지 않았으며, 공유된 복사본의 데이터 값을 포함하고 있다고 가정한다. 일관성 상태 정보는 제2 캐시 유닛 내의 데이터 값이 공유 상태임을 표시한다. 이 예에서 제1 프로세서는 배타적 사용을 위한 데이터를 요청하는 판독 버스 요청을 개시한다.

제2 캐시 유닛은 판독 버스 요청을 스누핑하고 자신이 데이터 값의 공유된 복사본을 포함하고 있는 지를 결정한다. 그 후 제2 캐시 유닛은 데이터 값의 일관성 상태 정보를 무효 상태로 변경함으로써 공유된 데이터 값을 무효화한다. 데이터 값의 일관성 상태를 무효 상태로 변경함으로써 제2 캐시 유닛 내의 데이터 값을 무효화한다. 그 후 제1 프로세서는 판독 버스 요청을 완료하고 배타적 사용을 위해 주 메모리로부터 데이터 값의 복사본을 얻는다.

단일 공통 버스를 구비하는 다중처리 시스템에서는 스누핑 동작을 통하여 캐시 일관성을 유지하지만, 보다 강력한 컴퓨터는 다중 프로세서와 주 메모리를 상호연결하는 하나 이상의 버스를 포함한다. 즉 공통 버스가 지원할 수 있는 프로세서의 수를 늘리는 것에는 제한이 있기 때문에, 원하는 레벨의 성능을 구현하기 위해서는 다중버스 시스템이 필요할 수 있다. 그러나 이와 같은 다중버스 시스템은 하나의 버스 상의 프로세서가 다른 버스 상의 프로세서에 의해 개시된 트랜잭션을 모니터할 수 없다는 문제점이 있다. 따라서 스누핑 동작은 다중버스 컴퓨터 내에서는 메모리 일관성을 유지할 수 없다.

다중버스 시스템 내의 캐시 일관성을 유지하기 위한 하나의 방법은 각 버스 상에서 개시된 버스 트랜잭션을 모든 다른 버스로 통보(broadcasting)하는 것이다. 불행하게도, 이러한 방법은 각 버스로 전송된 모든 버스의 버스 대역폭 로드(bus bandwidth load)를 결합시키는 결과를 가져온다. 예상할 수 있듯이, 이것은 시스템 성능을 상당히 감소시키며, 다중버스의 이점을 감소시킨다.

다중버스 시스템 내의 캐시 일관성을 유지하기 위한 다른 방법은 소위 디렉토리-기초(directory-based) 캐시 일관성 방법이라고 불리는 방법이다. IEEE 기준화 일관성 상호접속(Scaleable Coherent Interconnect; SCI)은 다중버스, 디렉토리-기초 캐시 일관성 시스템의 일례이다. 디렉토리 방식에서, 프로세서는 버스 트랜잭션을 스누핑하지 않는다. 대신, 주 메모리 서브시스템은 실제 데이터를 가지는 추가 정보를 저장함으로써 메모리 일관성을 유지한다.

주 메모리 서브시스템 내의 추가 정보는 일반적으로 1) 어느 프로세서 또는 프로세서들이 데이터 값에 대한 복사본을 얻었는 지에 대한 정보 및 2) 데이터 값에 대한 일관성 상태를 표시한다. 예를 들어 추가 정보는 하나 이상의 프로세서가 동일한 데이터 값을 공유하고 있음을 표시할 수 있다. 또 다른 예로, 추가 정보는 단지 하나의 프로세서만이 특정 데이터 값을 수정할 권리가 있음을 표시할 수 있다.

프로세서가 데이터 값을 요청하면, 주 메모리 서브시스템은 이 프로세서가 최신 버전의 데이터 값을 가지고 있는 지를 결정한다. 이 프로세서가 최신 버전의 데이터 값을 가지고 있지 않다면, 주 메모리 서브시스템은 최신 데이터 값을 가지는 프로세서로부터 데이터 값을 요청한 프로세서(이를 요청 프로세서(requesting processor)라 함)로 최신 데이터 값을 전송한다. 다른 예로서, 주 메모리가 다른 프로세서 중 어느 프로세서가 최신 값을 가지고 있는 지를 요청 프로세서에 표시할 수 있다.

주 메모리 서브시스템이 각 데이터 값에 대한 최신 버전이 어디에 위치하고 있는 가에 대한 정보를 보존하고 있기 때문에, 프로세서는 버스 트랜잭션을 "스누핑"할 필요가 없다. 그러나 이러한 디렉토리를 보존하는 것은 시스템에 대한 비용을 상당히 증가시키는데, 이는 각 데이터 값에 대하여 주 메모리 내에서 유지해야 하는 추가 정보 때문이다. 또한 각 데이터 값에 대한 디렉토리를 주 메모리 내에서 유지하는 것은 요청된 데이터를 요청 프로세서에 위치시키거나 전송시키는 데 필요한 시간 때문에 시스템 성능을 열화시킨다.

디렉토리-기초 시스템의 대안으로서, 실제적으로 캐시 유닛 내에 실제 저장된 데이터 값과 연관된 일관성 상태 정보를 저장하는 버스 상호접속(bus interconnect)을 들 수 있다. 이에 따라, (디렉토리-기초 방식에서와 같이) 주 메모리가 증가함에 따라 저장이 이에 비례하여 증가하는 것 대신에, 주 메모리보다 훨씬 작은 크기의 결합 캐시 유닛과 연관되어 있다. 그러나 이러한 방법은 다중버스 시스템이 각 캐시 유닛 내의 모든 데이터 값과 연관된 일관성 상태 정보의 이중화 복사본(duplicate copy)을 저장하기를 요구한다.

예를 들어, Sun Microsystem사의 UltraSpare 시스템은 버스 스위치를 사용하여 다중버스를 상호연결하며, 여기서 각 버스는 내부 캐시 유닛을 구비하는 프로세서와 통신한다. 버스 스위치는 캐시 유닛 내의 모든 데이터 값과 연관된 일관성 상태 정보의 이중화 복사본을 유지한다. UltraSpare 시스템에서, 버스 스위치는 일관성 상태 정보의 이중화 복사본을 유지할 수 있는데, 이는 UltraSpare 시스템 의 프로세서가 외부 캐시 태그(tag)를 유지하면서 어느 데이터 값이 대체되는 지에 대한 정확한 정보를 제공하도록 구성되어 있기 때문이다.

그러나 이러한 버스 스위치는 정확한 캐시 데이터 대체 정보를 출력하지 못하기 때문에, 다수의 오프 더 셸프(off-the-shelf) 프로세서에 있어서 실용적이지 못하다. 예를 들어, 다수의 종래 프로세서는 그들의 내부 캐시 유닛 내에서만 정확한 일관성 상태 정보를 유지할 수 있다. 그리하여, 다른 장치는 언제 데이터 값이 내부 캐시 유닛으로부터 제거(remove)되는지를 결정할 수 없다. 내부 캐시 유닛 내의 일관성 상태 정보에 대한 정확한 정보가 없으면, 버스 스위치는 일관성 상태 정보의 이중화 복사본을 유지할 수 없다.

발명의 상세한 설명

본 발명은 단일 버스 시스템의 전체 프로세서 성능 제한을 효과적으로 증가시키는 캐시-일관성 다중버스 시스템을 제공한다. 본 발명은 다중버스, 다중처리 시스템이 1) 다수의 시스템 버스 및 다수의 I/O 장치와 상호접속하여 주 메모리와 공유하는 시스템, 및 2) 시스템 내의 지연시간 및 전체 대역폭에 대한 영향을 최소화하면서 캐시 일관성을 효율적으로 유지하는 시스템과 같은 지연시간이 짧으며(low latency), 고대역의 시스템이 필요로 함을 인식한다. 본 발명은 이들 문제점을 "일관성 필터"를 사용하여 버스간 통신을 조정하여, 크로스 버스 트래픽의 오버헤드(overhead)를 감소시키면서 캐시 메모리 일관성을 유지한다.

본 발명의 실시예에서, 시스템 버스, I/O 버스 및 메모리 유닛은 다중포트화된 버스 스위치를 통해 연결된다. 버스 스위치는 임의의 시스템 버스 또는 I/O 버스를 임의의 메모리 유닛과 연결시킬 뿐만이 아니라 크로스 버스 트래픽을 다룬다. 또한 본 발명의 실시예에 의한 버스 스위치는 버스 트랜잭션에 응답하기 위해 필요한 동작을 결정하는 버스 인터페이스 로직(logic)을 포함한다. 그러나 본 발명이 이러한 다중포트화 버스 스위치에 제한되는 것은 아니며, 서로 다른 데이터 경로에 대하여 개별적인 버스 브리지가 존재하는 경우와 같은 다양한 형태의 다른 버스 상호접속에 적용될 수 있다.

다중버스, 다중처리 시스템 내의 캐시 일관성을 보장하기 위해, 캐시를 지원하는 각 버스는 할당된 일관성 필터를 구비한다. 각 일관성 필터는 태그 제어장치, 사이클 인코더(cycle encoder) 및 규칙표(rules table)를 포함한다. 또한 각 일관성 필터는 태그 메모리와 연결된다. 일반적으로 말해, 각 태그 제어장치는 모든 태그 메모리와 인터페이스한다. 각 사이클 인코더는 사이클 인코더의 할당된 버스 상에서 발생하는 버스 트랜잭션의 종류를 결정하고, 각 규칙표는 캐시 일관성을 유지하기 위해 필요한 버스 트랜잭션을 결정한다.

태그 메모리에 초점을 맞춰 설명하면, 각 태그 메모리는 1) 태그 메모리의 할당된 버스에 연결된 캐시 유닛에 위치하는 데이터 값의 어드레스 및 2) 데이터 값과 연관된 캐시 일관성 상태에 대한 레코드를 유지한다. 공지되어 있는 바와 같이, 주 메모리 내의 각 데이터 값은 해당 메모리 어드레스에 의해 식별된다. 본 발명의 실시예에서, 태그 메모리는 실제 데이터 값이 아니라, 데이터 값을 식별하는 데이터 값 어드레스를 저장한다. 본 발명의 실시예에 의한 태그 메모리는 데이터 값 어드레스 저장하는 것 이외에도, 데이터 값 어드레스와 연관된 일관성 상태 정보를 저장한다.

예를 들어, 제1 일관성 필터 및 제1 태그 메모리가 제1 버스에 할당되어 있다고 가정한다. 또한 제1 버스 상의 제1 프로세서가 주 메모리로부터 데이터 값을 요구한다고 가정한다. 제1 일관성 필터는 제1 태그 메모리 내의 메모리 어드레스의 레코드를 유지하며, 또한 제1 태그 메모리 내의 메모리 어드레스와 연관된 일관성 상태 정보를 저장한다.

메모리 트랜잭션 내에서 액세스된 데이터의 양은 시스템마다 다르다. 종래의 시스템에서, 프로세서가 메모리 판독 트랜잭션을 수행하면, 프로세서는 내부 캐시 메모리의 일부를 채우기 위해 메모리를 충분히 액세스한다. 일반적으로 내부 캐시 메모리는 다중 데이터 값을 캐시 라인에 저장한다.

공지되어 있는 바와 같이, 종래 컴퓨터 처리 시스템은 8 비트(바이트) 수량, 16 비트(워드) 수량, 32 비트(더블 워드) 수량으로 분할된다. 현재의 다수의 32 비트 프로세서에 있어서, 주 메모리는 더블 워드(32 비트) 경계(boundary)로 구성된다. 대부분의 32 비트 프로세서에 있어서, 각 캐시 라인은 다수의 더블 워드를 유지할 수 있다.

일반적으로, 프로세서가 일정한 데이터 값을 요구하면, 프로세서는 전체 캐시 라인을 채우기에 충분한 데이터를 구한다. 예를 들어, Pentium Pro 프로세서에 있어서, 각 내부 데이터 값의 크기는 다양하지만, 64 비트보다는 크지 않다. 그러나 Pentium Pro의 캐시 라인은 32 바이트의 데이터(256 비트)를 유지한다. Pentium Pro 프로세서가 주 메모리로부터 데이터 값을 구하면, 일반적으로 하나의 캐시 라인을 채우는데 필요한 8 데이터 값(256 비트)을 구한다.

종래의 시스템에 있어서, 각 캐시 라인은 캐시 라인 어드레스에 의해 식별된다. 예를 들어, Pentium Pro 시스템에서, 캐시 라인은 캐시 라인 내의 하위 데이터 값의 메모리 어드레스와 동일한 캐시 라인 어드레스를 가진다. 그러나 각 캐시 라인이 32 바이트의 데이터를 포함하기 때문에, 각 캐시 라인의 캐시 라인 어드레스는 길이가 더 짧으며, 5개의 하위 어드레스 비트를 포함하지 않는다. 본 발명의 실시예에서, 특정 버스에 할당된 각 태그 메모리는 캐시 라인 어드레스를 저장한다.

각 태그 메모리는 캐시 라인 어드레스를 저장하는 것 이외에도, 캐시 라인 어드레스와 연관된 일관성 상태를 저장한다. 일관성 상태는 캐시 유닛 내의 캐시 라인의 상태와 연관되어 있다. 본 발명의 실시예에서, 일관성 상태는 3개의 서로 다른 일관성 상태—무효 상태, 공유 상태 또는 점유 상태—를 포함한다.

무효 상태는 캐시 라인이 무효 상태이며, 캐시 라인을 저장하는 캐시 엔트리가 비어 있으며 새로운 캐시 라인을 저장할 수 있음을 의미한다. 공유된 상태는 프로세서가 캐시 라인의 복사본을 가지고 있으며, 수정 권한을 가지고 있지 않음을 의미한다. 예를 들어, 공유된 캐시 라인은 종종 수정되지 않은 프로그램 인스트럭션이거나, 가장 최근에 판독된 데이터 항목이다.

그러나 당업자는 캐시 라인의 일관성 상태가 무효, 공유 및 점유 프로토콜로 제한되지 않는다는 것을 이해하게 될 것이다. 사실 당업자는 일관성 상태가 수정, 배타적, 공유, 및 무효(MESI) 프로토콜, 수정, 점유, 배타적, 공유 및 무효(MOESI) 프로토콜, 수정, 공유, 및 무효(MSI) 프로토콜, 무효 및 점유의 2상태 프로토콜, 버클리 프로토콜, the University of Illinois 일관성 프로토콜, Digital Equipment의 Firefly 프로토콜, Xerox Dragon 프로토콜 등과 같은 다양한 범위의 일관성 프로토콜을 사용하여 구현될 수 있다. 본 발명의 실시예에서는 무효, 공유 및 점유 상태를 사용하며, 이는 이들 상태가 MESI 프로토콜을 사용하는 Pentium Pro 프로세서와 효율적으로 인터페이스할 수 있기 때문이다.

다수의 종래의 프로세서에 있어서, 프로세서는 내부 캐시 유닛 내에 저장된 캐시 라인에 대하여 정확한 일관성 상태 정보를 출력하지 않는 내부 캐시 유닛을 구비한다. 예를 들어, 내부 캐시 유닛은 캐시 라인이 폐기되었음을 신호하지 않고, 수정되지 않은 캐시 라인을 폐기할 수 있다. 다른 예에서, 내부 캐시 유닛은 캐시 유닛이 수정하지 않는 수정 특권(modification privilege)을 가지는 캐시 라인을 구할 수 있다. 이 예에서, 캐시 유닛은 캐시 라인이 폐기되었음을 신호하지 않고, 캐시 라인을 폐기할 수 있다. 따라서 캐시 유닛을 모니터하는 장치는 캐시 라인이 사실상 폐기된 캐시 라인을 가지는 경우 캐시 유닛이 캐시 라인의 수정된 복사본을 가진다는 사실을 믿을 수 있다. 그러나 본 발명의 실시예에서, 각 태그 메모리는 독창적으로 현재의 일관성 상태 정보를 출력하지 않는 내부 캐시 유닛에 대한 캐시 일관성을 보장하기에 적합하다.

본 발명의 중요한 특징은 각 태그 메모리가 현재 내부 캐시 유닛 내에 유지될 가능성이 있는 캐시 라인 어드레스의 슈퍼세트를 유지함으로써 캐시 일관성을 보장한다는 것이다. 따라서 태그 메모리 내의 캐시 라인 어드레스의 슈퍼세트는 캐시 유닛이 사실상 폐기된 캐시 라인을 가지고 있는 경우, 캐시 유닛 내의 특정 캐시 라인이 공유 상태임을 표시할 수 있다. 다른 경우, 태그 메모리 내의 캐시 라인 어드레스의 슈퍼세트는 캐시 유닛이 사실상 캐시 라인을 주 메모리에 재기록하는 경우 캐시 유닛 내의 특정 캐시 라인이 수정 상태임을 표시할 수 있다.

캐시 라인 어드레스의 슈퍼세트를 유지하기 위해, 본 발명의 실시예에 의한 일관성 필터는 포함 규칙(inclusion rule)이라 불리는 규칙을 사용한다. 포함 규칙은 특정 버스와 연결된 캐시 유닛 내에 저장된 캐시 라인 어드레스가 항상 버스에 할당된 태그 메모리 내의 캐시 라인 어드레스의 서브세트임을 보장한다. 각 일관성 필터가 연관된 버스에 의해 액세스된 모든 캐시 라인을 모니터하기 때문에, 각 액세스된 캐시 라인과 연관된 어드레스는 버스에 할당된 태그 메모리 내에서 유지된다. 캐시 라인 어드레스가 태그 메모리 중의 하나로부터 삭제되면, 포함 규칙은 연관된 캐시 유닛에게 그들의 캐시 메모리 내의 캐시 라인을 삭제하도록 지시한다.

예를 들어, 태그 메모리가 새로운 캐시 라인 어드레스를 유지하기 위한 메모리 용량을 가지고 있지 않으면, 태그 메모리에 존재하는 하나의 캐시 라인 어드레스(구 캐시 라인 어드레스)를 배출시킴으로써 새로운 캐시 라인 어드레스를 위한 공간을 태그 메모리 내에 형성해야 한다. 구 캐시 라인 어드레스가 무효 상태이면, 버스와 연결된 캐시 유닛은 더 이상 구 캐시 어드레스와 연관된 캐시 라인을 사용하지 않으며, 태그 메모리에 할당된 일관성 필터는 단순히 구 캐시 라인 어드레스를 새로운 캐시 라인 어드레스로 교체한다.

그러나 구 캐시 라인 어드레스가 공유 또는 점유 상태이면, 캐시 유닛이 구 캐시 라인 어드레스를 무효화할 때까지, 일관성 필터는 태그 메모리로부터 구 캐시 라인 어드레스를 배출할 수 없다. 전술한 바와 같이, 본 발명의 실시예에 의한 태그 메모리는 캐시 라인 어드레스의 슈퍼세트를 유지해야 하며, 따라서 구 캐시 라인 어드레스를 새로운 캐시 라인 어드레스로 교체하기 전에 캐시 유닛 내의 구 캐시 라인 어드레스가 먼저 무효화되어야 한다.

일관성 필터는 무효화 버스 트랜잭션을 수행함으로써 캐시 유닛 내의 구 캐시 라인 어드레스를 먼저 무효화시킨다. 무효화 버스 트랜잭션은 버스와 연결된 캐시 유닛에게 구 캐시 라인 어드레스 및 할당된 캐시 라인을 내부적으로 무효화하도록 지시한다.

예를 들어, 제1 캐시 유닛을 구비하는 제1 프로세서 및 제2 캐시 유닛을 구비하는 제2 프로세서가 할당된 일관성 필터 및 태그 메모리를 가지는 제1 버스와 연결되어 있다고 가정한다. 또한 제1 캐시 유닛이 공유 상태의 제1 캐시 라인을 포함한다고 가정한다. 이 실시예에서, 태그 메모리는 제1 캐시 라인 어드레스 및 공유된 상태의 정보를 포함한다. 또한 제2 프로세서가 제2 캐시 라인을 요구하는 판독 버스 트랜잭션을 개시한다고 가정한다. 최종적으로 태그 메모리가 제2 캐시 라인 어드레스에 대한 메모리 용량을 가지고 있지 않다고 가정한다.

이 실시예에서, 일관성 필터는 제2 캐시 라인 어드레스에 대한 공간을 형성하기 위해 제1 캐시 라인 어드레스를 배출시켜야 한다. 그러나 일관성 필터가 제1 캐시 라인 어드레스를 배출하기 전에, 일관성 필터는 제1 캐시 유닛 내의 제1 캐시 라인 어드레스를 개시하는 버스 트랜잭션을 수행해야 한다. 제1 캐시 라인 어드레스를 무효화하기 위해, 일관성 필터는 무효화 버스 트랜잭션을 수행하는데, 여기서 무효화 버스 트랜잭션은 제1 캐시 유닛에게 제1 캐시 라인 어드레스와 연관된 캐시 라인을 무효화하도록 지시한다.

무효화 버스 트랜잭션을 수행하는 동안, 일관성 필터는 제2 캐시 라인 어드레스에 대한 판독 버스 트랜잭션을 연기한다. 제1 캐시 라인 어드레스(제1 캐시 라인은 수정되지 않음)가 공유 상태이기 때문에, 제1 캐시 유닛은 무효화 버스 트랜잭션에 응답하고 제1 캐시 라인을 무효화한다. 무효화 버스 트랜잭션이 완료된 후에, 일관성 필터는 태그 메모리 내의 제1 캐시 라인 어드레스를 제2 캐시 라인 어드레스로 대체한다.

때때로, 제1 캐시 유닛은 수정된 제1 캐시 라인(다시 말해, 제1 캐시 라인은 점유 상태임)이다. 제1 캐시 라인이 점유 상태이면, 제1 일관성 필터는 제1 캐시 라인을 무효화하는 무효 버스 트랜잭션을 다시 수행한다. 그러나 제1 캐시 유닛이 제1 캐시 유닛을 수정하면, 제1 캐시 유닛은 수정된 제1 캐시 라인을 주 메모리로 재기록하는 기록 버스 트랜잭션을 수행함으로써 무효화 버스 트랜잭션에 응답한다.

수정된 제1 캐시 라인을 주 메모리에 기록한 후에, 제1 캐시 유닛은 제1 캐시 라인을 무효화한다. 그후 일관성 필터는 태그 메모리 내의 제1 캐시 라인 어드레스를 제2 캐시 라인 어드레스로 대체한다. 그리하여 때로는 태그 메모리 내의 캐시 라인 어드레스의 슈퍼세트를 유지하는 것은 캐시 유닛에게 태그 메모리 내의 캐시 라인을 무효화하기 전에, 수정된 데이터를 주 메모리에 재기록할 것을 요구한다.

본 발명의 실시예에서, 일관성 필터는 직접 매핑 기술(direct mapping techniques)을 사용하여 태그 메모리 내에 캐시 라인 어드레스를 저장한다. 직접 매핑 기술은 각 캐시 라인 어드레스가 태그 메모리 내의 하나의 특정 태그 엔트리에 매핑하도록 지정한다. 본 발명의 실시예에서는 직접 매핑 기술을 사용하지만, 당업자는 다수의 다른 기술을 사용하여 태그 메모리 내의 캐시 라인 어드레스를 구성할 수 있다는 것을 이해하게 될 것이다. 완전한 어소시어티브 시스템에서는, 임의의 태그 엔트리 내에 임의의 캐시 라인 어드레스가 존재할 수 있다. 다른 실시예에서 각 캐시 라인 어드레스는 2개의 서로 다른 태그 엔트리(2중 세트 어소시어티브) 중에서 단지 하나의 태그 엔트리에만 저장되거나 4개의 서로 다른 태그 엔트리(4중 세트 어소시어티브) 중에서 단지 하나의 태그 엔트리에 저장될 수 있다.

본 발명의 실시예의 직접 매핑 기술에 초점을 맞춰, 각 캐시 라인 어드레스는 특정 태그 엔트리를 식별하는 인덱스로서 사용된다. 본 발명의 실시예에서, 태그 메모리 내의 엔트리의 개수는 태그 페이지의 크기를 결정한다. 바람직하게 각 시스템에 연결된 태그 메모리는 동일한 태그 페이지 크기를 가진다. 태그 페이지 크기는 프로세서의 캐시 내의 전체 캐시 메모리의 양과 연관되어 있다. 또한 이들 장치에 비해 I/O 버스와 연결된 태그 메모리는 그 크기가 작으며, 이는 일반적으로 I/O 브리지와 연결된 작은 크기의 캐시 유닛 때문이다.

태그 페이지는 주 메모리의 페이지와 구분되어서는 안된다. 공지되어 있는 바와 같이, 컴퓨터의 물리적 메모리 어드레스 공간은 개념적으로 메모리 페이지라 불리는 다중 섹션으로 구성될 수 있으며, 각 메모리 페이지는 다중 캐시 라인을 포함한다. 메모리 페이지는 처리 시스템에 의해 구성되며, 태그 페이지와 독립적이다.

본 발명의 실시예에서, 캐시 라인 어드레스는 1)캐시 라인 어드레스를 포함하는 태그 페이지 2) 태그 페이지 내의 캐시 라인 어드레스의 위치를 식별한다. 특히 캐시 라인 어드레스 내의 상위 비트가 태그 페이지를 식별하고, 하위 비트는 태그 페이지 내의 캐시 라인 어드레스의 위치를 식별한다.

일반적으로 하위 비트는 태그 페이지 내의 캐시 라인 어드레스의 위치를 식별하기 때문에, 인덱스(index)라 불린다. 예를 들어, 제1 태그 페이지 내의 제1 캐시 라인 어드레스에 있어서, 상위 어드레스 비트는 제1 태그 페이지를 식별하고, 하위 어드레스 비트는 제1 태그 페이지 내의 캐시 어드레스 라인의 위치를 식별한다.

본 발명의 실시예에서, 일관성 필터 내의 태그 제어장치는 캐시 라인 어드레스를 태그 메모리로 직접 매핑한다. 예를 들어, 제1 버스와 연결된 프로세서가 특정 캐시 라인 어드레스를 요청하는 버스 트랜잭션을 개시하면, 제1 태그 제어장치는 캐시 라인 어드레스를 평가한다. 제1 태그 제어장치는 하위 어드레스 비트를 인덱스로 사용하여, 제1 태그 메모리 내의 특정 태그 엔트리를 식별한다. 그후 제1 태그 제어장치는 상위 비트(태그 페이지)를 식별된 태그 엔트리에 저장한다.

본 발명의 실시예에서, 태그 메모리는 정적(static) 메모리로서 구현된다. 정적 메모리는 버스 트랜잭션 동안에 각 태그 제어장치가 각 태그 메모리를 신속하게 액세스하는 것을 허용한다. 본 발명이 정적 메모리에서 구현되지만, 당업자는 서로 형태의 저장 메커니즘을 사용하여 태그 메모리를 구현할 수 있다는 것을 인식하게 될 것이다. 바람직하게 서로 다른 형태의 저장 메커니즘은 버스 클록 속도와 비례하는 메모리 액세스 속도를 제공하여, 성능을 최적화시킨다.

2개의 캐시 라인 어드레스가 동일한 태그 엔트리로 매핑되면, 태그 제어장치는 이전의 캐시 라인 어드레스를 배출시켜, 새로운 캐시 라인 어드레스를 위한 공간을 형성한다. 전술한 바와 같이, 이러한 프로세스는 구 캐시 라인 어드레스가 무효화될 때까지 새로운 캐시 라인 어드레스와 연관된 버스 트랜잭션을 연기시킬 수 있다. 또한 구 캐시 라인 어드레스를 무효화하는 것은 태그 메모리가 캐시 유닛 내에 존재하는 캐시 라인 어드레스의 슈퍼세트를 유지한다는 것을 보장하기 위한 추가 버스 트랜잭션을 요구할 수 있다.

본 발명의 실시예에서, 각 일관성 필터는 새로운 캐시 라인 어드레스와 연관된 버스 트랜잭션을 연기하지 않고 구 캐시 라인 어드레스와 새로운 캐시 라인 어드레스를 유지하는 무효화 큐를 추가로 포함한다. 이것은 시스템 성능을 증가시키며, 이는 이후에 구 캐시 라인을 무효화하는 무효화 버스 트랜잭션이 발생할 수 있기 때문이다.

다중버스 내의 캐시 일관성을 유지하기 위한 방법에 초점을 맞춰 설명하면, 본 발명의 실시예에 의한 일관성 필터는 할당된 버스 상에서 버스 트랜잭션을 모니터함으로써 언제 크로스 버스 트랜잭션이 요구되는 지를 결정한다. 특히 각 일관성 필터 내의 사이클 인코더는 일관성 필터의 할당된 버스 상에서 발생하는 각 버스 트랜잭션을 모니터한다. 본 발명의 실시예에서, 사이클 인코더는 버스 제어 라인을 모니터하는 공지된 버스 모니터링 로직을 사용한다. 그후 사이클 인코더는 1) 버스 트랜잭션의 형태 및 2) 버스 트랜잭션과 연관된 태그 메모리 내의 캐시 상태 정보를 일관성 규칙표로 전송한다.

규칙표에 초점을 맞춰 설명하면, 규칙표는 캐시 일관성 보장하기 위해 크로스 버스 트랜잭션을 수행해야 하는 시간을 결정한다. 본 발명의 실시예에서, 규칙표는 태그 메모리 내의 일관성 상태 정보에 부분적으로 기초하여 크로스 버스 트랜잭션을 수행해야 하는 지의 여부를 결정한다. 예를 들어, 제1 버스 상의 버스 판독 트랜잭션이 특정 캐시 라인 어드레스를 식별하면, 제1 버스에 할당된 규칙표는 다른 버스에 할당된 태그 메모리(리모트 태그 메모리) 내의 캐시 라인 어드레스의 일관성 상태를 평가한다.

리모트 태그 메모리로부터의 일관성 상태를 사용하여, 규칙표는 캐시 일관성을 보장하기 위해 리모트 버스 트랜잭션이 필요한 지의 여부를 결정한다. 아래에서 더 상세하게 기술되어 있듯이, 특정 일관성 필터에서, 태그 제어장치는 리모트 태그 메모리를 액세스하고, 캐시 상태를 규칙표에 입력한다. 또한 사이클 인코더는 버스 트랜잭션의 형태를 결정하고, 버스 트랜잭션 정보를 규칙표에 입력한다.

본 발명의 실시예에서, 규칙표는 큰 진리표의 기능을 수행한다. 버스 트랜잭션 정보 및 리모트 태그 메모리 정보를 사용하여, 규칙표는 캐시 일관성을 유지하기 위해 필요한 크로스 버스 트랜잭션 또는 버스 트랜잭션의 세트를 결정한다.

예를 들어, 프로세서가 제1 버스 상에서 판독 버스 트랜잭션을 시작한다고 가정한다. 이 예에서, 버스 트랜잭션을 시작하는 제1 버스를 로컬 버스라 하며, 다중버스 시스템 내의 다른 버스를 리모트 버스라 한다. 판독 버스 트랜잭션은 원하는 캐시 라인 어드레스를 로컬 버스에 할당된 일관성 필터(로컬 일관성 필터)로 전송한다. 그후 로컬 일관성 필터는 캐시 라인 어드레스가 리모트 버스에 할당된 태그 메모리(리모트 태그 메모리) 내에 존재하는 지의 여부를 평가한다.

이 예에서의 리모트 태그 메모리는 원하는 캐시 라인 어드레스의 일관성 상태가 무효 상태임을 표시한다. 이러한 경우 캐시 일관성을 유지하기 위해 크로스 버스 트랜잭션을 수행할 필요가 없으며, 이는 리모트 버스 내의 캐시 라인 어드레스가 무효 상태이기 때문에 가능하다. 그러므로 로컬 일관성 규칙표는 리모트 버스 상에서 트랜잭션을 발생시키지 않으며, 버스 트랜잭션을 로컬 버스 및 주 메모리로 제한한다. 버스 트랜잭션을 로컬 버스로 제한하는 것은 크로스 버스 트래픽을 감소시킨다.

그러나 리모트 태그 메모리는 크로스 버스 트랜잭션이 필요함을 표시하며, 규칙표는 캐시 일관성을 보장하기 위해 필요한 적당한 크로스 버스 트랜잭션 또는 트랜잭션 세트를 결정한다. 예를 들어, 리모트 태그 메모리 중의 하나는 캐시 라인 어드레스가 점유 상태임을 표시할 수 있으며, 따라서 리모트 버스와 연결된 캐시 유닛은 캐시 라인의 수정된 버전을 소유할 가능성이 있다. 리모트 태그 메모리가 캐시 라인 어드레스가 점유 상태임을 표시하면, 로컬 진리표는 캐시 일관성을 보장하기 위해 리모트 버스와 연결된 버스 마스터 로직이 리모트 버스 상에서 버스 판독 커맨드를 수행할 필요가 있음을 표시한다.

버스 판독 커맨드가 리모트 상에서 실행되면, 리모트 캐시 유닛이 버스 판독 커맨드를 스누핑하고, 이들이 원하는 캐시 라인의 수정된 버전을 가지고 있는 지의 여부를 결정한다. 리모트 버스 상의 리모트 캐시 유닛 중의 하나가 캐시 라인의 수정된 버전을 되돌리면, 규칙표는 캐시 라인을 로컬 버스 상의 요청 프로세서로 전송한다.

그러나 리모트 버스 상의 캐시 라인 중의 어느 하나라도 수정된 캐시 라인을 수정하지 않으면, 캐시 유닛은 버스 판독 커맨드에 응답하지 않는다. 그후 규칙표는 최신 캐시 라인이 주 메모리 내에 존재하는 지의 여부를 결정한다. 따라서 본 발명의 실시예에서는 캐시 일관성을 유지하기 위해 크로스 버스 트랜잭션이 필요한 경우에는 태그 메모리 내의 캐시 라인 어드레스의 슈퍼세트를 사용한다.

본 발명의 다른 특징은 입/출력 장치와 인터페이스하기 위한 전용 제3 버스를 사용한다는 것이다. 본 발명의 실시예에 의한 I/O 버스는 다른 프로세서 버스와 동일한 형태를 가진다. 그러나 당업자는 I/O 버스 및 각 다른 프로세서 버스가 서로 다른 버스 프로토콜을 사용할 수 있다는 것을 이해하게 될 것이다.

본 발명의 실시예에 의한 I/O 버스는 다른 프로세서와 유사한 방식으로 동작한다. 대부분의 고성능 컴퓨터의 I/O 버스는 직접 메모리 액세스(direct memory access; DMA) 전송을 수행한다. DMA 전송은 일반적으로 I/O 장치에 의해 시작되며, I/O 장치는 중앙 프로세서와는 직접 연관되지 않으며 주 메모리와 I/O 장치 사이에서 데이터를 직접 이동시킨다. I/O 버스 상에서 발생하는 I/O 트랜잭션에 대하여 메모리 일관성을 유지하는 것은 각 DMA 전송이 이루어지기 전 및 후에 캐시 유닛 내의 캐시 라인의 플러싱(flushing)을 방지한다.

다른 형태의 I/O 전송은 프로세서에 의한 I/O 데이터의 직접 프로그래밍 액세스와 관련되어 있다. 본 발명의 실시예에서, 버스 스위치는 직접 I/O 전송을 I/O 버스로 전송하고, 메모리-매핑된 I/O 전송으로서 주 메모리 어드레스 공간에 대한 액세스 이외의 모든 메모리 액세스를 I/O 버스로 전송한다. 아래에서 보다 상세하게 기술하고 있듯이, 이러한 I/O 전송은 캐시 일관성과 관련되어 있지 않지만, 독창적인 방법을 사용하여 하나의 버스로부터 다른 버스로 전송된다.

본 발명의 실시예에 의한 I/O 버스는 I/O 일관성 필터 및 I/O 버스 인터페이스를 포함하며, 이는 다중 시스템 버스 양단의 I/O 매핑을 개선시키며, I/O 데이터 처리를 개선시키며, 시스템 버스의 구조를 간단하게 한다. 버스 상에서 발생하는 I/O 데이터 트랜잭션은 I/O 버스로 자동적으로 전송된다. 또한 I/O 버스 상에서 시작되는 트랜잭션은 목적지 버스로 전송되며 버스 트랜잭션을 다른 버스로 전송하지 않는다.

본 발명의 다른 특징은 다중버스간의 통신을 최적화하는 것이다. 예를 들어, 종래의 버스 스위치는 서로 다른 버스를 독립적인 접속 경로와 상호접속시킨다. 그리하여 종래의 다중버스 시스템에서, 제1 버스 및 제2 버스는 항상 독립적인 제1 접속 경로와 상호접속되어 있으며, 제1 버스 및 제3 버스는 독립적인 제2 접속 경로와 상호접속되어 있으며, 제2 버스 및 제3 버스는 독립적인 제3 접속 경로와 상호접속되어 있다. 예상되는 바와 같이, 이러한 독립적인 접속 경로는 버스 스위치 구현을 더 복잡하게 한다.

예를 들어, 제1 버스가 제1 버스 트랜잭션이 제2 버스를 향하는 것을 원한다면, 제1 버스는 제1 버스 트랜잭션을 제1 큐 내에 위치시켜, 제1 버스를 제2 버스와 링크한다. 이와 유사하게, 제2 버스가 제2 버스 트랜잭션이 제1 버스를 향하는 것을 원한다면, 제2 버스는 제2 버스 트랜잭션을 제2 큐 내에 위치시켜, 제2 버스를 제1 버스와 링크한다. 그후 제1 버스는 제2 큐의 출력으로부터 제2 버스 트랜잭션을 구한다.

따라서 2개의 버스는 2개의 큐를 요구한다. 추가 버스가 상호접속되면, 더 많은 수의 큐가 필요하다. 예를 들어, 3중 버스 시스템에서, 각 버스간 접속은 2개의 큐를 요구한다. 따라서 3중 버스 시스템은 6개의 큐를 요구한다.

그러나 본 발명의 실시예에 의한 새로운 버스 스위치 구현 방법은 모든 버스에 의해 액세스 가능한 다중포트화된 메모리 셀 영역을 사용하여 이러한 시스템의 구성을 더 간단하게 한다. 새로운 버스 스위치를 사용함으로써, 데이터를 임의의 버스로부터 다른 임의의 버스로 전송할 수 있으며, 동일한 시간에 발생할 수 있는 다른 데이터 전송과 충돌하지 않는다. 아래에서 보다 상세하게 기술되어 있는 바와 같이, 각 버스로부터의 버스 전송은 공통 메모리 셀 영역으로 입력된다. 그후 공통 메모리 셀 영역의 버스 트랜잭션은 목적지 버스로 향하게 된다. 임의의 버스는 독립적인 접속 경로를 사용하지 않고서도 임의의 다른 버스로부터 데이터를 판독하거나 기록할 수 있다.

본 발명의 실시예에서, 각 버스 트랜잭션과 연관된 정보는 3개의 서로 다른 메모리 셀—데이터 셀, 요청 셀 및 어드레스 셀—에 저장된다. 데이터 셀은 버스 트랜잭션과 연관된 데이터를 저장한다. 요청 셀은 목적지 버스로 전송된 버스 트랜잭션의 형태를 정의하는 버스 트랜잭션 정보를 포함한다. 최종적으로 어드레스 셀은 버스 트랜잭션과 관련된 어드레스 정보 및 일관성 상태 정보를 포함한다.

본 발명의 실시예에서, 각 데이터 셀, 각 요청 셀, 각 어드레스 셀간에는 1대1 대응이 존재한다. 따라서 각 데이터 셀, 요청 셀, 또는 어드레스 셀, 또는 이들 셀의 임의의 조합은 특정 버스 트랜잭션에 대한 정보를 포함할 수 있다. 본 발명의 실시예에서 메모리 셀을 사용하여 버스 트랜잭션 정보를 유지하지만, 버스 트랜잭션 정보는 3개 이하의 메모리 셀 내에 존재할 수 있다.

개념적으로, 데이터 셀, 요청 셀 및 어드레스 셀을 다중포트화된 단일 메모리 영역으로 간주할 수 있다. 본 발명의 실시예에서는 데이터 셀, 요청 셀, 어드레스 셀이 서로 다른 구성요소에 위치하지만, 이들은 계속하여 1대1 대응을 유지한다. 본 발명의 실시예에서, 데이터 인터페이스 버퍼는 데이터 셀을 포함하고, 시스템 액세스 제어장치는 어드레스 셀 및 요청 셀을 포함한다.

본 발명의 실시예에 의한 데이터 인터페이스 버퍼에 초점을 맞춰 설명하면, 데이터 인터페이스 버퍼 내의 각 데이터 셀은 다중포트화되어 모든 버스에 의해 액세스 가능하다. 각 데이터 셀은 특정 버스 트랜잭션과 연관된 데이터를 포함한다. 데이터 인터페이스 버퍼 내의 데이터 셀 영역은 버스 데이터 경로와 상호접속하는 것이 바람직하다.

본 발명의 실시예에 의한 시스템 액세스 제어장치에 초점을 맞춰 설명하면, 시스템 제어장치는 중앙 요청 리스트, 버퍼 관리자, 복수의 버스 마스터 및 복수의 버스 슬레이브를 포함한다. 공지되어 있는 바와 같이, 각 버스 마스터는 하나의 버스 상에서 버스 트랜잭션을 시작하고, 각 버스 슬레이브는 상기 하나의 버스와 연결된 다른 장치에 의해 시작된 버스 트랜잭션을 수신한다. 중앙 요청 리스트는 요청 셀 영역을 관리하고, 버퍼 관리자는 어드레스 셀 영역을 관리한다.

중앙 요청 셀 내의 각 요청 셀은 다중포트화되며, 모든 버스에 의해 액세스 가능하다. 각 요청 셀은 타겟 버스 식별자, 버스 트랜잭션 코드로 불리는 동작 코드 및 오너 버스 식별자를 포함한다. 타겟 버스 식별자는 특정 목적지 버스를 식별하며, 버스 트랜잭션 코드는 특정 버스 트랜잭션을 식별하고, 오너 버스 식별자는 발신(originating) 버스를 식별한다.

버퍼 관리자 내의 어드레스 셀 영역에 초점을 맞춰 설명하면, 각 어드레스 셀은 다중포트화되며 "사용중" 정보, 메모리 어드레스 및 데이터 셀 상태 정보를 포함한다. 어드레스 셀 내의 사용중 정보는 어드레스 셀이 사용가능한 지의 여부를 표시한다. 본 발명의 실시예에서, 사용중 정보는 어드레스 셀이 사용중인지 또는 미사용 상태인지를 표시하도록 설정된 사용중 비트를 포함한다. 사용중 비트가 미사용 상태로 설정되면, 유효 데이터가 데이터 셀 내에 존재할 수 있다. 이것은 미사용 데이터 셀 내의 유효 데이터를 최적으로 재사용할 수 있도록 허용한다.

한편 메모리 어드레스는 버스 트랜잭션과 연관된 메모리 어드레스를 포함하며, 데이터 셀 상태는 데이터 셀 내의 데이터의 상태를 표시한다. 버퍼 관리자는 어드레스 셀 영역 이외에도 어드레스 셀 우선순위 인코더, 다중 FIFO(first-in first-out) 메모리 및 다중 어드레스 비교기를 포함한다. 어드레스 셀 우선순위 인코더는 사용중인 셀, 및 새로운 버스 트랜잭션 정보를 수신할 수 있는 미사용 상태의 셀을 결정한다. 본 발명의 실시예에서, 어드레스 셀 우선순위 인코더는 각 어드레스 셀 내의 사용중 정보를 평가하여, 미사용 상태의 셀을 결정한다.

어드레스 셀 우선순위 인코더는 미사용 상태의 셀을 결정할 뿐만 아니라, 미사용 상태의 어드레스 셀을 서로 다른 버스에 할당한다. 바람직하게, 어드레스 셀 우선순위 인코더는 미사용 상태의 어드레스 셀을 서로 다른 버스에 할당한다. 미사용 상태의 어드레스 셀을 버스에 할당한 후에, 어드레스 우선순위 인코더는 사용중 비트를 어드레스 셀이 미사용 상태가 아님을 표시하도록 설정한다. 본 발명의 실시예에 의한 우선순위 인코더는 제1 미사용 상태의 어드레스 셀을 제1 버스에 할당하고, 제2 미사용 상태의 어드레스 셀을 제2 버스에 할당하고, 제3 미사용 상태의 어드레스 셀을 제3 버스에 할당한다.

제4 어드레스 셀이 미사용 상태가 되면, 어드레스 셀 우선순위 인코더는 제1 버스를 역회전하고, 제4 어드레스 셀을 제1 버스에 할당한다. 본 발명의 실시예에 의한 어드레스 셀 우선순위 인코더는 이러한 기술을 사용하여 미사용 상태의 어드레스 셀을 서로 다른 버스에 할당하지만, 당업자는 어드레스 셀 우선순위 인코더가 광범위한 범위의 할당 방식을 사용하여 미사용 상태의 어드레스 셀을 서로 다른 버스에 할당할 수 있다는 것을 이해하게 될 것이다.

버퍼 관리자의 FIFO 메모리에 초점을 맞춰 설명하면, FIFO 메모리는 할당된 어드레스 셀이 버스에 의해 요구될 때까지 이들을 저장한다. 본 발명의 실시예에서, FIFO 메모리는 할당된 어드레스 셀을 식별하는 어드레스 셀 식별자를 저장한다. 어드레스 셀 식별자는 할당된 어드레스 셀의 메모리 위치를 포함하는 가변 데이터이다. 버스는 어드레스 셀 식별자를 사용하여, 어드레스 셀 식별자에 의해 식별된 어드레스 셀 메모리 위치를 액세스한다.

본 발명의 실시예에서, 각 FIFO 메모리는 특정 메모리로 할당된다. 또한 각 FIFO 메모리는 버스 슬레이브 중 하나 및 각 FIFO 메모리와 동일한 버스에 할당된 일관성 필터 중의 하나와 연결된다. 하나의 버스 슬레이브 또는 하나의 일관성 필터가 버스 트랜잭션을 다른 버스로 전송하기를 원한다면, 이들 장치는 할당된 FIFO 메모리로부터 어드레스 셀 식별자 중의 하나를 구한다.

예를 들어, 제1 버스 상의 제1 프로세서가 데이터 값을 제2 버스 상의 제2 I/O 장치로 전송하기를 원한다고 가정한다. 이 예에서, 제1 버스 슬레이브는 제1 버스와 연결된다. 제1 프로세서가 제2 I/O 장치로 데이터 값을 전송하는 버스 트랜잭션을 시작하면, 제1 버스 슬레이브는 이러한 버스 트랜잭션을 수신한다. 그후 제1 버스 슬레이브는 제2 버스로 전송될 필요가 있는 버스 트랜잭션을 결정한다.

따라서 제1 버스 슬레이브는 버퍼 관리자 내의 제1 FIFO 메모리를 액세스하고, 어드레스 셀 식별자를 구한다. 어드레스 셀 식별자를 사용하여, 제1 버스 슬레이브는 식별된 어드레스 셀을 액세스하고, 데이터 값 어드레스를 저장하며, 필요하다면 어드레스 셀 내의 데이터 값의 일관성 상태를 저장한다. 해당 요청 셀에 있어서, 제1 버스 슬레이브는 제2 버스를 타겟 버스 식별자에 지정하고, 버스 트랜잭션 코드를 동작 코드(버스 트랜잭션 코드라고도 함)에 지정하고, 제1 버스를 오너 버스 식별자에 지정한다. 또한 제1 버스 슬레이브는 버스 트랜잭션과 연관된 데이터 값을 해당 데이터 셀 내에 저장한다.

다른 예에서는, 제1 버스에 할당된 제1 캐시 일관성 필터에 의해 캐시 라인 액세스가 제2 버스 상에서 캐시 일관성을 보장하기 위한 버스 트랜잭션을 요구한다 것이 결정되었다고 가정한다. 이 예에서, 제1 캐시 일관성 필터는 버퍼 관리자 내의 FIFO 메모리를 액세스하고, 어드레스 셀 식별자를 구한다.

제1 일관성 필터는 어드레스 셀 식별자를 사용하여 식별된 어드레스 셀을 액세스한다. 그후 제1 일관성 필터는 캐시 라인 어드레스 및 어드레스 셀 내의 일관성 상태 정보를 포함한다. 또한 요청 셀에서, 제1 일관성 필터는 타겟 버스 식별자 내의 제2 버스, 적당한 버스 트랜잭션 코드 및 버스 식별자 내의 제1 버스를 지정한다. 그러나 이 예에서, 해당 데이터 셀은 빈 상태로 남으며, 이는 캐시 일관성을 보장하기 위해 캐시 라인 데이터가 필요하지 않기 때문에 가능하다. 버스 트랜잭션 정보가 셀에 더해지면, 적당한 버스는 버스 트랜잭션 정보를 구하고 원하는 버스 트랜잭션을 실행시킨다.

본 발명의 실시예에서, 중앙 요청 리스트 내의 복수의 버스 우선순위 인코더는 요청 셀로 연결된다. 전술한 바와 같이, 요청 셀 내의 타겟 버스 식별자는 목적지 버스를 식별한다. 일반적으로 말해, 버스 우선순위 인코더는 요청 셀 내의 타겟 버스 식별자를 평가하여, 버스 트랜잭션을 수행해야할 버스를 결정한다.

예를 들어, 요청 셀 내의 타겟 버스 식별자가 제1 버스와 제2 버스를 지정한다고 가정한다. 이 예에서, 제1 버스 우선순위 인코더는 요청 셀 내의 타겟 버스 식별자를 평가하여, 제1 버스를 위한 요청 셀을 식별하고, 제2 버스 우선순위 인코더는 타겟 버스 식별자를 평가하여 제2 버스를 위한 요청 셀을 식별한다.

목적지 버스를 식별하는 것 이외에도, 각 버스 우선순위 인코더는 가장 높은 우선순위를 가지는 특정 버스와 연관된 버스 요청 셀을 결정한다. 본 발명의 실시예에서, 각 버스 우선순위 인코더는 라운드 로빈 기술을 사용하여 가장 높은 우선순위를 가지는 버스 요청 셀을 결정한다. 라운드 로빈 기술은 각 버스 우선순위 인코더가 가장 높은 우선순위를 버스 요청 셀에 순차적으로 할당하도록 한다.

각 버스 우선순위 인코더는 가장 높은 우선순위를 가지는 버스 요청 셀을 하나의 버스 마스터로 전송한다. 전술한 바와 같이, 버스 식별자를 가지는 것 이외에도, 요청 셀은 버스 트랜잭션 코드를 포함한다. 그후 버스 마스터는 요청 셀 내에 식별된 버스 트랜잭션을 수행한다. 아래에서 보다 상세하게 기술하고 있는 바와 같이, 트랜잭션을 수행하는 버스는 버스 트랜잭션을 시작한 버스로 데이터를 재기록할 필요가 있다. 이러한 경우에 버스 마스터는 데이터 셀을 사용하여 재기록 데이터를 저장하고, 발신 버스와 통신하기 위해 요청 셀을 재사용한다. 전술한 바와 같이, 발신 버스는 요청 셀 내에 존재하는 오너 버스 식별자에 의해 식별된다. 그러나 버스 트랜잭션이 완료되는 즉시, 버스 마스터는 어드레스 셀(500), 요청 셀(600) 및 데이터 셀(700)을 미사용 상태로 설정한다.

버스 마스터가 버스 트랜잭션을 수행하면, 버스 우선순위 인코더는 버스에 할당된 우선순위가 다음으로 가장 높은 요청 셀을 식별하고, 상기 요청 셀을 버스 마스터로 전송한다. 버스 우선순위 인코더가 버스에 할당된 최종 버스 요청 셀에 도달하면, 버스 우선순위 인코더는 버스에 할당된 제1 버스 요청 셀을 역회전시킨다. 라운드 로빈 방식에 기초하여 각 버스 요청 셀에 가장 높은 우선순위를 부여함으로써 결과적으로 모든 버스 요청 셀을 버스로 전송한다. 새로운 요청 셀이 중앙 요청 리스트로 더해지면, 각 버스 우선순위 인코더는 새로운 요청 셀을 즉각적으로 액세스하여 가장 높은 우선순위를 할당한다.

본 발명의 다른 특징에서, 버퍼 관리자는 어드레스 충돌을 식별하는 복수의 어드레스 비교기를 포함한다. 일반적으로 2개의 서로 다른 버스 트랜잭션이 동일한 데이터 값과 연관되어 있으면, 거의 동시에 어드레스 충돌이 발생한다. 이러한 경우에, 동일한 데이터에 대한 2개의 버스 트랜잭션이 어드레스 셀, 요청 셀, 및 데이터 셀 내에 동시에 존재할 수 있다. 예상할 수 있는 바와 같이, 이러한 어드레스 충돌은 적당하지 않는 결과를 초래한다.

본 발명의 실시예에서, 하나의 어드레스 비교기 세트는 각 버스로 할당된다. 각 어드레스 비교기 세트는 버퍼 관리자 내의 하나의 일관성 필터, 하나의 버스 슬레이브, 및 모든 어드레스 셀과 연결된다. 각 버스 트랜잭션에 있어서, 상기 버스에 할당된 어드레스 비교기 세트는 버스 트랜잭션 어드레스를 어드레스 셀 내의 모든 어드레스와 비교한다. 어드레스 충돌이 검출되면, 적당한 동작을 보장하기 위해, 아래에서 상세하게 기술하고 있는 바와 같이 적당한 동작이 수행되어야 한다.

도면의 간단한 설명

본 발명의 이들 및 다른 특징, 이점 및 신규한 특징은 다음의 도면을 참조하여 다음의 상세한 설명을 읽는다면 명확해질 것이다.

도 1은 본 발명의 실시예에 따른 다중처리, 다중버스 시스템의 블록도이며,

도 2는 본 발명의 실시예에 따른 시스템 액세스 제어장치(system access controller)의 블록도이며,

도 3은 본 발명의 실시예에 따른 일관성 필터(coherency filter) 및 태그 메모리(tag memory)의 블록도이며,

도 4는 본 발명의 실시예에 따른 캐시 라인 어드레스 및 태그 메모리의 블록도이며,

도 5a 및 도 5b는 본 발명의 실시예에 따른 버퍼 관리자(buffer manager)의 블록도이며,

도 6은 중앙 요청 리스트(central request list)의 블록도이며,

도 7은 데이터 인터페이스 버퍼(data interface buffer)의 블록도이며,

도 8은 본 발명의 실시예에 따른 버스 판독 커맨드(Bus Read command) 동안에 메모리 일관성을 유지하기 위한 방법을 나타내는 순서도이며,

도 9는 본 발명의 실시예에 따른 버스 무효화 커맨드(Bus Invalidate command) 동안에 메모리 일관성을 유지하는 방법을 나타내는 순서도이며,

도 10은 본 발명의 다른 실시예에 따른 무효화 큐(invalidation queue)의 블록도이다.

도면에서, 세자리 수의 첫 번째 숫자는 구성요소가 처음으로 도시되는 도면의 번호를 표시한다. 예를 들어, 도면부호 402를 가지는 구성요소는 도 4에 처음 도시된다. 또한 구성요소간의 일치성을 표시하기 위해 도면 전체에 걸쳐 동일한 도면부호가 동일한 구성요소를 가리키도록 한다.

실시예

본 발명의 실시예는 1) 다중 시스템 버스 및 I/O 버스를 공유된 주 메모리와 상호연결하며, 2) 시스템 내의 지연시간 및 전체 대역폭에 대한 영향을 최소화하면서 캐시 일관성을 효율적으로 유지하는, 캐시-일관성, 다중버스 시스템을 제공한다. 특히 본 발명은 버스간(bus-to-bus) 통신을 조정하여, 캐시 메모리 일관성을 작은 양의 크로스 버스 트래픽(cross-bus traffic)으로 유지하는 일관성 필터를 포함한다.

이하에서는 본 발명의 실시예에 따른 다중버스 시스템에 대하여 기술하고 있지만, 본 발명이 이러한 다중버스 시스템에 한정되는 것은 아니며, 다른 형태의 다중버스에서도 사용될 수 있다. 본 발명의 완전한 이해를 돕기 위해, 나머지 상세한 설명은 다음의 항과 하부 항으로 구성되어 있다.

Ⅰ. 용어 및 두문자의 용어해설

Ⅱ. 본 발명의 실시예에 따른 다중버스 시스템의 개요

Ⅲ. 시스템 액세스 제어장치

A. 일관성 필터

1. 태그 메모리

2. 태그 제어장치

3. 사이클 인코더(cycle encoder)

4. 규칙표(rules table)

B. 버퍼 관리자

C. 중앙 요청 리스트

Ⅳ. 데이터 인터페이스 버퍼

Ⅴ. 캐시 일관성 유지

A. 버스 판독 라인 커맨드 처리

B. 버스 판독 무효화 라인 커맨드 처리

Ⅵ. 다른 실시예

Ⅶ. 결론

Ⅰ. 용어 및 두문자의 용어해설

ASIC: 응용 주문용 집적 회로(Application-Specific Integrated Circuit)

BRL 커맨드: 버스 판독 라인 커맨드(Bus Read Line Command). 버스 판독 라인 커맨드는 버스 상의 캐시 라인을 판독한다.

BRIL 커맨드: 버스 판독 무효화 라인 커맨드(Bus Read Invalidate Line Command). 버스 판독 무효화 라인 커맨드는 버스 상에서 캐시 라인을 판독하고 이를 무효화한다.

버스 마스터(bus master): 특정 버스 트랜잭션을 제어하는 제어 로직. 하나 이상의 장치가 공통 버스를 공유하는 몇몇의 시스템에서, 각 장치는 버스 마스터가 되기 위한 내부 로직을 가진다. 버스 마스터가 버스 트랜잭션을 수행한 후에, 버스 마스터는 다른 장치가 버스 마스터가 되도록 하기 위해 이 버스에 대한 제어권을 폐기(relinquish)한다. 이러한 장치에는 프로세서, I/O 장치, 메모리 제어장치 등이 있다.

버스 슬레이브(bus slave): 버스 마스터로부터 버스 트랜잭션을 수신하는 제어 로직.

버스 스누핑: 다른 프로세서에 의해 수행된 메모리 트랜잭션을 모니터하기 위해 캐시 메모리를 구비하는 프로세서에 의해 사용되는 기술.

DIB: 데이터 인터페이스 버퍼(data interface buffer)

DEFER# 신호: DEFER# 신호는 버스 트랜잭션을 지연시킨다. Pentium Pro 버스의 예에서 DEFER# 신호는 DEFER# 신호의 의미를 표시하는 제어 신호를 부호화하는 유일한 신호는 아니지만 독창적인(unique) 신호이다.

DEN# 신호: DEN# 신호는 언제 버스 트랜잭션이 지연될 수 있는 지를 표시한다. 지연된 버스 트랜잭션은 무순서(out of order)로 실행된다. 다시 말해, 제1 버스 트랜잭션이 지연되면, 지연된 제1 버스 트랜잭션 이전에 제2 버스 트랜잭션이 완료된다. 버스 트랜잭션을 발행(issue)하는 프로세서는 DEN# 신호를 인가함으로써 버스 트랜잭션이 지연가능하는 지의 여부를 표시한다.

FIFO: 선입선출(first-in first-out) 메모리. FIFO는 항목이 더해지는 순서와 동일한 순서로 항목을 출력하는 메모리 큐이다. 다시 말해 첫 번째 항목이 제일 먼저 출력된다.

HIT# 신호: 본 발명의 실시예에서, 각 프로세서는 버스 상에서 함께 논리합(OR) 연산된 HIT# 신호를 가진다. 프로세서는 HIT# 신호를 발생시켜, 데이터 값이 공유되고 있음을 표시한다. 예를 들어, 제1 프로세서가 데이터 값을 요청하면, 제2 프로세서는 버스 트랜잭션을 모니터하여, 제2 프로세서가 요청된 데이터 값의 복사본을 포함하는 경우 HIT# 신호를 발생시킨다.

HITM# 신호: 본 발명의 실시예에서, 각 프로세서는 HITM# 신호를 가진다. 수정된 데이터 값이 프로세서의 캐시 메모리 내에 존재한다는 사실이 버스 스누핑 동작에 의해 표시되면, 프로세서는 HITM# 신호를 인가한다.

I/O: 입/출력. 일반적으로 입/출력 장치와의 입/출력 트랜잭션을 가리킨다.

PCI 버스: 주변 장치 접속 버스(Peripheral Component Interconnect Bus)

프로세서: 본 발명의 상세한 설명에서 프로세서는 계산형 유닛 또는 제어 유닛을 의미한다. 프로세서는 버스 트랜잭션을 통해 주 메모리와 통신하며, CPU, 마이크로프로세서, 스마트 입/출력 장치 및 정보를 저장하고 처리하거나 전송하기 위한 다른 장치를 포함한다.

RAM: 임의 액세스 메모리(Random Access Memory)

SRAM: 정적 임의 액세스 메모리(Static Random Access Memory)

SDRAM: 동기식 동적 임의 액세스 메모리(Synchronous Dynamic Random Access Memory)

스누핑 히트: 프로세서에 의해 캐시 메모리가 특정 메모리 트랜잭션에서 요청되는 데이터를 포함하고 있다는 사실이 검출되면, 스누핑 히트가 발생한다.

Ⅱ. 본 발명의 실시예에 따른 다중버스 시스템의 개요

도 1에 도시되어 있는 바와 같이, 본 발명의 실시예에 따른 다중버스 시스템(100)은 3개의 시스템 버스—제1 시스템 버스(102), 제2 시스템 버스(104), 및 제3 시스템 버스(106)—를 포함한다. 각 시스템 버스(102, 104 및 106)는 어드레스 및 제

어 라인 세트(108a, 108b 및 108c) 및 데이터 라인 세트(110a, 110b, 및 110c)를 포함한다. 이들 어드레스 및 제어 라인은 총괄하여 어드레스 및 제어 라인(108)으로 언급된다. 또한 이들 데이터 라인(110a, 110b, 및 110c)은 총괄하여 데이터 라인(110)으로 언급된다. 이하의 설명에서는, 제1 시스템 버스(102)를 좌측 버스(102)라 지칭하며, 제2 시스템 버스(104)를 우측 버스(104)라 지칭하며, 제3 시스템 버스(106)를 I/O 버스(106)라 지칭한다.

본 발명의 실시예에서, 각 시스템 버스(102, 104 및 106)는 Intel사에서 규정한 Pentium Pro 시스템이다. Pentium Pro 시스템 버스는 36 비트의 어드레스, 64 비트의 데이터 및 다양한 형태의 제어 및 오류 검출 신호(erroneous detection signal)를 제공한다. 본 발명의 실시예에서는 Pentium Pro 버스를 사용하고 있지만, 당업자는 본 발명을 캐시 일관성 프로토콜 스누핑을 구현하는 광범위한 시스템 버스에 적용할 수 있음을 인식하게 될 것이다. 또한 버스의 포맷(format)은 버스마다 서로 다를 수 있다.

좌측 버스(102)에 복수의 프로세서(112a, 및 112b 등)가 연결되어 있으며, 우측 버스(104)에 복수의 프로세서(112c, 및 112d 등)가 연결되어 있다. 프로세서(112a, 112b, 112c, 및 112d)는 총괄하여 프로세서(112)로 언급된다. 본 발명의 실시예에서, 각 프로세서(112)는 내부 캐시 유닛(114)을 구비한다. 도 1에는 4개의 프로세서(112)가 도시되어 있지만, 각 버스(102, 및 104)는 추가 프로세서(112)에 연결될 수도 있다. 본 발명의 실시예에서 사용되는 프로세서(112)는 Intel사에서 제조되는 Pentium Pro 프로세서이다. 본 발명의 실시예에서는 Pentium Pro 프로세서를 사용하고 있지만, 당업자는 본 발명을 특정 시스템 버스와 호환가능한(compatible) 광범위한 프로세서(112)에 적용할 수 있다는 것을 인식하게 될 것이다.

프로세서(112) 내의 캐시 메모리(114)는 데이터를 국부적(locally)으로 저장함으로써 처리 성능을 개선시킨다. 일반적으로, 캐시 메모리(114)는 작은 버스 대역폭을 사용하여, 프로세서(112)가 데이터 값을 액세스하도록 허용한다. 본 발명의 실시예에 의한 캐시 메모리(114)는 데이터 값을 캐시 라인으로 편성하는데, 여기서 상기 캐시 라인은 32 바이트의 데이터를 포함한다.

본 발명의 실시예에서의 제3 시스템 버스(106)는 복수의 I/O 브리지(120)와 주 메모리(132) 사이에서 입/출력 트랜잭션을 전송하며, 이에 따라 I/O 버스(106)로 언급된다. 본 발명의 실시예의 I/O 브리지(120)는 Intel사에서 제조되는 82450 GX PCIset Orion PCI 브리지이다. 본 발명의 실시예에서, I/O 브리지(120)는 PCI 버스를 사용하여 I/O 버스(106)로부터 I/O 트랜잭션을 복수의 I/O 장치(122)로 전송한다. 그러나 당업자는 I/O 브리지(120)를 다양한 I/O 장치(122)를 액세스하는 광범위한 장치로 구현할 수 있다는 것을 인식하게 될 것이다. 또한 I/O 브리지(120)는 선택적(optional)이며, 호환성 I/O 장치(122)가 I/O 버스(106)에 직접 부착될 수 있다.

또한 본 발명의 실시예에 의한 다중버스 시스템(100)은 홀수 주 메모리 모듈(odd main memory module; 132a), 짝수 주 메모리 모듈(even main memory module; 132b) 및 데이터 인터페이스 버퍼(134)를 포함한다. 본 발명의 실시예에서, 홀수 주 메모리 모듈(132a) 및 짝수 주 메모리 모듈(132b)은 총괄하여 주 메모리(132)로 언급된다. 본 발명의 실시예에 따른 홀수 메모리 모듈(132a) 및 짝수 메모리 모듈(132b)은 32 메가바이트 내지 32 기가바이트의 크기를 가지는 SDRAM을 포함한다. SDRAM은 고속 데이터 버스팅(high-speed bursting of data)을 제공하는 동기식 파이프라인 인터페이스를 사용한다.

또한 본 발명의 실시예의 다중버스 시스템은 시스템 액세스 제어장치(130) 및 데이터 인터페이스 버퍼(134)를 포함한다. 일반적으로 말하면, 시스템 액세스 제어장치는 다중버스 시스템의 동작을 제어한다. 시스템 액세스 제어장치는 각 버스(102, 104 및 106)의 어드레스 및 제어 라인(108a, 108b, 및 108c)과 연결되어 있다. 또한 시스템 액세스 제어장치(130)는 홀수 메모리 어드레스 및 제어 라인 세트(138a), 짝수 메모리 어드레스 및 제어 라인 세트(138b), 및 데이터 인터페이스 버퍼(DIB) 제어 라인 세트(140)와 연결되어 있다.

한편, 데이터 인터페이스 버퍼(134)는 서로 다른 버스(102, 104, 및 106)들 및 주 메모리 사이에 데이터 경로(data path)를 제공한다. 데이터 인터페이스 버퍼(134)는 각 버스(102, 104, 및 106)의 데이터 라인(110a, 110b, 및 110c)과 연결된다. 또한 데이터 인터페이스 버퍼(134)는 주 메모리 데이터 라인(142)과 연결된다.

프로세서(112)는 주 메모리(132)로부터 데이터를 판독함으로써 자신들의 캐시 메모리(114)를 채운다. 캐시 메모리(114) 내의 최신 데이터를 유지하기 위해, 특정 버스 상의 프로세서(112) 내의 캐시 메모리(114)는 자신에 할당된 버스(102, 104 또는 106) 상에서 발생하는 주 메모리 버스 트랜잭션을 스누핑한다. 이것을 버스 스누핑이라 한다.

캐시 메모리(114)가 버스 트랜잭션에서 식별된 캐시 라인과 동일한 캐시 라인을 포함하면, 스누핑 히트가 발생한다. 스누핑 히트가 발생하면, 이 데이터 값에 대한 공유된 복사본을 가지고 있는 프로세서(112) 또는 프로세서들(112)이 HIT# 신호를 인가한다. HIT# 신호는 어드레스 및 제어 라인(108) 중의 하나에 인가된다. HIT# 신호는 다른 프로세서(112)에게 메모리 트랜잭션과 연관된 데이터를 공유하고 있음을 통지한다. 본 발명의 실시예에서, 다른 프로세서(112) 중 어느 프로세서도 HIT# 신호를 인가하지 않으면, 요청 프로세서는 원하는 데이터 값의 일관성 상태를 그 자신의 상태로 설정한다.

다른 경우에, 프로세서(112)는 자신의 캐시 메모리(114) 내의 캐시 라인을 수정할 수 있다. 그러나 버스 트래픽을 감소시키기 위해, 캐시 메모리(114)는 다른 프로세서(112)가 주 메모리(132)로부터 동일한 캐시 라인을 요청할 때까지, 수정된 캐시 라인을 주 메모리(132)로 전송하지 않을 수 있다. 예를 들어, 어떤 프로세서(112)가 캐시 라인을 수정하면, 다른 프로세서(112)는 이 캐시 라인의 복사본을 가지고 있지 않거나, 동일한 캐시 라인을 얻고자 할 수도 있다. 이러한 경우에, 최신 캐시 라인을 가지는 프로세서(112)는 최신 캐시 라인을 주 메모리(132)에 기록하지 않고, 그 대신에, 프로세서(112)는 캐시 메모리(114) 내의 최신 캐시 라인을 유지한다.

본 발명의 실시예에서, 수정된 캐시 라인이 프로세서의 캐시 메모리(114) 내에 존재한다는 사실이 버스 스누핑 동작에 의해 표시되면, 프로세서들(112) 중 하나의 프로세서가 HITM# 신호를 인가한다. HITM# 신호는 어드레스 및 제어 라인(108) 중의 하나에 인가되며, 최신 캐시 라인을 가지는 프로세서(112)가 캐시 라인을 주 메모리(132)에 재기록할 필요가 있음을 표시한다. 예를 들어, 본 발명의 실시예에서, 좌측 버스(102) 상의 제1 프로세서(112a)가 캐시 메모리(114) 내의 수정된 캐시 라인을 포함하고 있다고 가정한다. 좌측 버스(102) 상의 제2 프로세서(112b)가 동일한 캐시 라인의 복사본을

얻고자 하는 경우, 제2 프로세서(112b)는 좌측 버스(102)에 대한 제어권을 획득하고, 원하는 캐시 라인을 식별하는 주 메모리(132)로의 메모리 트랜잭션을 실행한다. 그 후 제1 프로세서(112a)는 이 메모리 트랜잭션을 스누핑하고, 제1 프로세서(112a)가 원하는 캐시 라인의 수정된 버전을 포함하고 있는 지를 결정한다.

이에 응답하여, 제1 프로세서(112a)는 자신이 수정된 버전의 캐시 라인을 포함하고 있음을 표시하는 HITM# 신호를 인가한다. 제1 프로세서(112a)는 좌측 버스(102)의 커맨드를 수신하여, 수정된 캐시 라인을 주 메모리(132)에 재기록한다. 제1 프로세서(112a)가 수정된 캐시 라인을 주 메모리(132)에 재기록한 후에, 제2 프로세서(112b)는 주 메모리 트랜잭션을 다시 개시하여, 최신 캐시 라인을 획득한다.

본 발명의 실시예의 프로세서(112)는 버스 트랜잭션이 지연될 수 있는 지의 여부를 표시한다. 버스 트랜잭션을 발행하는 프로세서(112)는 DEN# 신호를 사용하여 버스 트랜잭션을 지연할 수 있는 지의 여부를 표시한다. HITM# 신호를 인가하는 프로세서는 트랜잭션 응답을 제어한다. 프로세서가 HITM# 신호를 인가하는 경우, 프로세서는 메모리가 행하듯이, 버스 트랜잭션을 순서대로 실행한다. 또는 트랜잭션이 지연가능한 경우에는, 시스템 액세스 제어장치(130)가 버스 트랜잭션을 지연시키는 DEFER# 신호를 인가한다. DEFER# 신호는 특정 제어 라인(108)의 부호화(encoding) 신호이다.

Ⅲ. 시스템 액세스 제어장치

도 2에 도시된 시스템 액세스 제어장치(130)에 초점을 맞춰 설명하면, 본 발명의 실시예에 따른 시스템 액세스 제어장치(130)는 응용 주문용 집적 회로(Application-Specific Integrated Circuit; ASIC)로 구현된다. 일반적으로 말하면, 시스템 액세스 제어장치(130)는 다중버스 시스템(100) 내의 캐시 일관성을 유지하면서, 3개의 버스(102, 104, 및 106) 및 주 메모리(132)를 제어한다.

시스템 액세스 제어장치(130)는 좌측 일관성 필터(200a), 우측 일관성 필터(200b), 및 I/O 일관성 필터(200c)를 포함한다. 이하의 설명에서는 좌측 일관성 필터(200a), 우측 일관성 필터(200b), 및 I/O 일관성 필터(200c)를 일관성 필터(200)라 지칭한다. 또한 시스템 액세스 제어장치(130)는 좌측 버스 마스터(202a), 우측 버스 마스터(202b) 및 I/O 버스 마스터(202c)를 포함한다. 이하의 설명에서는 좌측 버스 마스터(202a), 우측 버스 마스터(202b) 및 I/O 버스 마스터(202c)를 총괄하여 버스 마스터(202)라 지칭한다.

또한 시스템 액세스 제어장치(130)는 좌측 버스 슬레이브(204a), 우측 버스 슬레이브(204b), 및 I/O 버스 슬레이브(204c)를 포함한다. 이하의 설명에서는, 좌측 버스 슬레이브(204a), 우측 버스 슬레이브(204b), 및 I/O 버스 슬레이브(204c)를 총괄하여 버스 슬레이브(204)라 지칭한다. 또한 시스템 액세스 제어장치(130)는 홀수 메모리 제어장치(206a) 및 짝수 메모리 제어장치(206b)를 포함한다. 이하의 설명에서는, 홀수 메모리 제어장치(206a) 및 짝수 메모리 제어장치(206b)를 총괄하여 메모리 제어장치(206)라 지칭한다.

또한 시스템 액세스 제어장치(130)는 중앙 요청 리스트(208) 및 버퍼 관리자(210)를 포함한다. 최종적으로 시스템 액세스 제어장치(130)는 외부 좌측 태그 메모리(212a), 외부 우측 태그 메모리(212b), 및 내부 I/O 태그 메모리(212c)와 연결된다. 이하의 설명에서는, 좌측 태그 메모리(212a), 우측 태그 메모리(212b), 및 내부 I/O 태그 메모리(212c)를 총괄하여 태그 메모리(212)라 지칭한다.

본 발명의 실시예에서, 좌측 태그 메모리(212a), 좌측 일관성 필터(200a), 좌측 버스 마스터(202a) 및 좌측 버스 슬레이브(204a)는 좌측 버스(102)로 할당된다. 우측 태그 메모리(212b), 우측 일관성 필터(200b), 우측 버스 마스터(202b) 및 우측 버스 슬레이브(204b)는 우측 버스(102)에 할당된다. I/O 태그 메모리(212c), I/O 일관성 필터(200c), I/O 버스 마스터(202c), 및 I/O 버스 슬레이브(204c)는 I/O 버스(106)에 할당된다.

시스템 액세스 제어장치(130)의 상호접속에 초점을 맞춰 설명하면, 좌측 일관성 필터(200a)는 좌측 버스 어드레스 및 제어 라인(108a), 좌측 태그 메모리(212a), 우측 태그 메모리(212b), I/O 태그 메모리(212c), 버퍼 관리자(210) 및 중앙 요청 리스트(208)와 통신한다. 우측 일관성 필터(200b)는 우측 버스 어드레스 및 제어 라인(108b), 좌측 태그 메모리(212a), 우측 태그 메모리(212b), I/O 태그 메모리(212c), 버퍼 관리자(210) 및 중앙 요청 리스트(208)와 통신한다. I/O 일관성 필터(200c)는 I/O 버스 어드레스 및 제어 라인(108c), 좌측 태그 메모리(212a), 우측 태그 메모리(212b), I/O 태그 메모리(212c), 버퍼 관리자(210) 및 중앙 요청 리스트(208)와 통신한다.

좌측 버스 마스터(202a)는 좌측 버스 어드레스 및 제어 라인(108a) 및 중앙 요청 리스트(208)와 통신한다. 우측 버스 마스터(202b)는 우측 버스 어드레스 및 제어 라인(108b) 및 중앙 요청 리스트(208)와 통신한다. I/O 버스 마스터(202c)는 I/O 버스 어드레스 및 제어 라인(108c) 및 중앙 요청 리스트(208)와 통신한다.

좌측 버스 슬레이브(204a)는 좌측 버스 어드레스 및 제어 라인(108a), 홀수 메모리 제어장치(206a), 짝수 메모리 제어장치(206b) 및 버퍼 관리자(210)와 통신한다. 우측 버스 슬레이브(204b)는 우측 버스 어드레스 및 제어 라인(108b), 홀수 메모리 제어장치(206a), 짝수 메모리 제어장치(206b) 및 버퍼 관리자(210)와 통신한다. I/O 버스 슬레이브(204c)는 I/O 버스 어드레스 및 제어 라인(108c), 홀수 메모리 제어장치(206a), 짝수 메모리 제어장치(206b) 및 버퍼 관리자(210)와 통신한다.

따라서 중앙 요청 리스트는 좌측 일관성 필터(200a), 우측 일관성 필터(200b), I/O 일관성 필터(200c), 좌측 버스 마스터(202a), 우측 버스 마스터(202b), 및 I/O 버스 마스터(202c)와 통신한다. 버퍼 관리자(210)는 좌측 일관성 필터(200a), 우측 일관성 필터(200b), I/O 일관성 필터(200c), 좌측 버스 마스터(202a), 우측 버스 마스터(202b), 및 I/O 버스 마스터(202c), 좌측 버스 슬레이브(204a), 우측 버스 슬레이브(204b), I/O 버스 슬레이브(204c), 홀수 메모리 제어장치(206a) 및 짝수 메모리 제어장치(206b)와 통신한다.

버스 마스터(202)에 초점을 맞춰 설명하면, 버스 마스터(202)는 자신에 할당된 버스(102, 102 및 106) 상의 버스 트랜잭션을 제어한다. 예를 들어, 좌측 버스 마스터(202a)는 좌측 버스(102) 상의 버스 트랜잭션을 개시한다. 버스 마스터(202)가 하나 이상의 버스 트랜잭션을 수행한 후에는, 버스 마스터(202)가 이 버스를 폐기하여, 다른 장치가 버스 마스터가 되도록 한다. 버스 마스터(202)를 구현하기 위한 제어 로직은 당업자에게 공지되어 있다.

버스 슬레이브(204)에 초점을 맞춰 설명하면, 버스 슬레이브(204)는 자신에 할당된 버스(102, 104, 및 106) 상의 프로세서들(112) 또는 I/O 브리지(120) 중 하나에 의해 개시된 버스 트랜잭션을 수신한다. 예를 들어, 프로세서들(112) 중 하나의 프로세서는 주 메모리(132) 내의 특정 데이터 값에 대한 판독 버스 트랜잭션을 개시할 수 있다. 버스 슬레이브(204)는 버스 트랜잭션을 수신하고, 주 메모리(132)로부터 요청된 데이터 값을 구한다. 버스 슬레이브(204)를 구현하기 위한 제어 로직은 당업자에게 공지되어 있다.

홀수 메모리 제어장치(206a) 및 짝수 메모리 제어장치(206b)는 각각 홀수 주 메모리 모듈(132a) 및 짝수 주 메모리 모듈(132b)에 대한 액세스를 제어한다. 홀수 메모리 제어장치(206a) 및 짝수 메모리 제어장치(206b)는 당업자에게 공지된 메모리 제어 기술을 사용하여 홀수 주 메모리 모듈(132a) 및 짝수 주 메모리 모듈(132b)을 제어한다.

A. 일관성 필터

본 발명의 실시예의 일관성 필터(200)에 초점을 맞춰 설명하면, 각 일관성 필터(200)는 어느 버스 트랜잭션 또는 어느 세트의 버스 트랜잭션이 캐시 일관성을 유지하기 위해 필요한 지를 결정한다. 본 발명의 실시예에서, 각 일관성 필터(200)는 캐시 일관성을 유지하기 위해 크로스 버스 트랜잭션이 반드시 필요할 것은 아닌 경우에 크로스 버스 트래픽을 제한함으로써, 다중버스 시스템(200)의 성능을 개선한다.

전술한 바와 같이, 좌측 일관성 필터(200a)는 좌측 버스 어드레스 및 제어 라인(108a)을 모니터하고, 우측 일관성 필터(200b)는 우측 버스 어드레스 및 제어 라인(108b)을 모니터하고, I/O 일관성 필터(200c)는 I/O 어드레스 및 제어 라인(108c)을 모니터한다. 도 3에 도시되어 있는 바와 같이, 각 일관성 필터(200)는 태그 제어장치(300), 사이클 인코더(302) 및 규칙표(304)를 포함한다. 또한 각 일관성 필터(200)는 태그 메모리(212)중의 하나에 할당된다.

1. 태그 메모리

본 발명의 실시예에서, 각 태그 메모리(212)는 시스템 액세스 제어장치(130)의 외부에 위치한 SRAM에 저장된다. 좌측 태그 메모리(212a) 및 우측 태그 메모리(212b)는 동일한 크기를 가지는 것이 바람직하며, 반면 I/O 태그 메모리의 크기는 I/O 브리지(120) 등과 연결된 작은 크기의 캐시 때문에 일반적으로 이들 장치에 비해 작다. 특정 버스에 할당된 태그 메모리(212)는 1) 할당된 버스와 연결된 캐시 메모리(114)에 위치하는 캐시 라인 및 2) 캐시 라인의 캐시 상태에 대한 슈퍼셋(superset) 레코드를 유지한다. 본 발명의 실시예에서, 캐시 상태는 캐시 메모리(114) 내의 캐시 일관성 상태보다 적은 수의 상태를 포함하는 캐시 일관성 상태 세트에 의해 표현된다.

공지되어 있는 바와 같이, 주 메모리 내의 각 캐시 라인은 캐시 라인 어드레스로 식별된다. 캐시 라인의 크기는 다중버스 시스템(100)에서부터 다른 다중버스 시스템(100)에 이르기까지 다양하다. 본 발명의 실시예에서, 캐시 라인은 32개의 8비트 데이터 값(256 비트)을 포함한다. 각 태그 메모리(212)는 실제 캐시 라인을 저장하는 것이 아니라, 캐시 라인을 식별하기 위한 캐시 라인 어드레스를 저장한다. 캐시 라인 어드레스를 저장하는 것 이외에도, 또한 각 태그 메모리(212)는 캐시 라인 어드레스와 연관된 일관성 상태 정보를 저장한다. 그리하여 각 태그 메모리(212)는 태그 메모리의 할당된 버스와 연결된 캐시 메모리(114)에 저장된 데이터의 상태 및 가능성 있는 내용(probable content)에 관한 정보를 저장한다.

예를 들어 좌측 버스(102) 상의 프로세서(112a)가 주 메모리(132) 내의 캐시 라인을 액세스하는 판독 버스 트랜잭션을 발생시키는 경우, 이 프로세서(112a)는 원하는 캐시 라인 어드레스를 좌측 버스 어드레스 및 제어 라인(108a) 상에 위치시킨다. 아래에서 보다 상세하게 기술하듯이, 좌측 일관성 필터(200a)는 원하는 캐시 라인 어드레스를 수신하고 이 캐시 라인 어드레스를 좌측 태그 메모리(212a)에 저장한다. 또한 좌측 일관성 필터(200a)는 이 캐시 라인 어드레스와 연관된 일관성 상태를 좌측 태그 메모리(212a)에 저장한다.

각 캐시 라인 어드레스와 연관된 일관성 상태는 캐시 메모리(114) 내의 캐시 라인의 상태와 관련되어 있다. 본 발명의 실시예에서, 일관성 상태는 3개의 서로 다른 일관성 상태—무효 상태, 공유 상태 또는 점유 상태—를 포함한다. 무효 상태는 캐시 라인이 무효 상태이며 프로세서(112)가 이 캐시 라인을 사용해서는 안된다는 것을 의미한다. 공유 상태는 프로세서(112)가 캐시 라인을 수정할 수 없음을 의미한다. 예를 들어, 공유된 캐시 라인은 종종 수정되지 않은 프로그램 인스트럭션이다. 점유 상태는 캐시 라인이 자신을 점유하는 프로세서(112)에 의해 수정될 수 있음을 의미한다. 본 발명의 실시예에서는 무효, 공유, 및 점유 프로토콜을 사용하는데, 이는 이들 프로토콜이 Pentium Pro 프로세서(112)에 의해 사용된 MESI 프로토콜을 포함하는 광범위한 캐시 일관성 프로토콜에 적용가능하기 때문이다.

그러나 당업자는 캐시 라인의 일관성 상태가 무효, 공유 및 점유 프로토콜에 한정되지 않는다는 것을 이해할 것이다. 사실 당업자는 수정(M), 배타적(E), 공유(S), 무효(I)의 MESI 프로토콜, 수정(M), 점유(0), 배타적(E), 공유(S) 및 무효(I)의 MOESI 프로토콜, 수정(M), 공유(S), 무효(I)의 MSI 프로토콜, 및 버클리(Berkeley) 프로토콜, University of Illinois 일관성 프로토콜과 같은 광범위한 일관성 프로토콜을 사용하여 일관성 상태를 구현할 수 있다는 것을 인식하게 될 것이다.

일관성 상태는 2개의 무효 및 점유 프로토콜과 같은 2상태 일관성 프로토콜에 의해서도 구현될 수 있다. 이중 버스(dual-bus) 시스템에서, 임의의 버스의 2상태 일관성 프로토콜은 임의의 캐시 라인의 공유된 복사본을 가지는 것으로 간주된다. 이는 종종 일관성 메모리의 크기를 감소시키는데, 이는 모든 캐시 라인이 공유된 것으로 처리되는 경우, 일관성 메모리는 단지 무효 및 점유 상태의 데이터 값과 연관된 레코드만을 유지하면 되기 때문이다. 버스가 공유된 데이터에 대하여 표준 판독 트랜잭션을 수행하는 경우, 공유된 상태 정보는 일관성 메모리에 저장되지 않는다. 이것은 일관성 메모리 내의 일관성 상태 정보의 슈퍼세트를 유지하는데 있어서 발생하는 문제점을 감소시킨다.

그러나 이러한 2상태 시스템에서, 모든 배타적 판독 트랜잭션 또는 무효화 트랜잭션은 캐시 일관성을 보장하기 위해 하나 이상의 크로스 버스 트랜잭션을 요구한다. 이러한 크로스 버스 트랜잭션은 다른 버스가 실제로 무효화 커맨드의 배타적 판독 트랜잭션과 연관된 데이터의 공유된 버전을 포함하는 지를 살펴볼 필요가 있다. 이중 버스 시스템에서, 2상태 프로토콜은 전체적 성능을 향상시키는데, 2개 이상의 버스를 가지는 경우 2상태 프로토콜의 이점은 작업 로드(work load)의 특성에 따라 많이 달라진다.

본 발명의 Pentium Pro 프로세서(112)는 자신의 내부 캐시 메모리(114)에 저장된 캐시 라인에 대한 정확한 일관성 상태 정보를 출력하지 않는다. 예를 들어, 내부 캐시 메모리(114)는 캐시 라인이 폐기되었다는 신호를 발생하지 않고 수정되지 않은 캐시 라인을 폐기한다. 본 발명의 실시예에서, 각 태그 메모리(212)는 현재의 일관성 상태 정보를 출력하지 않는 내부 캐시 메모리(114) 내의 캐시 일관성을 보장하는데 독창적으로 적용된다.

각 태그 메모리(212)는 특정 버스와 연결된 내부 캐시 메모리(114)에서 현재 보유하고 있을 가능성이 있는 캐시 라인 어드레스의 슈퍼세트를 유지함으로써 캐시 일관성을 보장한다. 예를 들어, 좌측 태그 메모리(212)는 프로세서(112a) 및 프로세서(112b)의 내부 캐시 메모리(114)에서 보유하고 있을 가능성이 있는 캐시 라인 어드레스의 슈퍼세트를 유지한다. 캐시 라인 어드레스의 슈퍼세트가 반드시 정확한 태그 상태 정보를 포함해야 하는 필요가 있는 것은 아니기 때문에, 캐시 메모리(114)가 사실상 캐시 라인을 폐기하는 경우, 태그 메모리(212)는 캐시 메모리(114) 내의 특정 캐시 라인이 공유 상태임을 표시할 수 있다. 다른 경우, 캐시 메모리(114)가 사실상 캐시 라인을 주 메모리(132)에 재기록하는 경우, 태그 메모리(212) 내의 캐시 라인 어드레스 슈퍼세트는 캐시 메모리(114) 내의 특정 캐시 라인이 수정된 상태임을 표시할 수 있다.

캐시 라인 어드레스의 슈퍼세트를 유지하기 위해, 본 발명의 실시예의 태그 메모리(212)는 포함 규칙(inclusion rule)이라 불리는 규칙을 사용한다. 포함 규칙은 특정 버스와 연결된 캐시 메모리(114)에 저장된 캐시 라인 어드레스가 항상 버스에 할당된 태그 메모리(212) 내의 캐시 라인 어드레스의 서브세트임을 보장한다. 캐시 라인 어드레스가 태그 메모리(212) 중의 하나로부터 삭제되면, 포함 규칙은 연관된 캐시 메모리(114)에게 그들의 캐시 메모리 내의 캐시 라인을 무효화하도록 지시한다.

예를 들어, 좌측 태그 메모리(212a)가 새로운 캐시 라인 어드레스를 유지하기 위한 메모리 용량 또는 연관성(associativity)을 가지고 있지 않는 경우에는, 좌측 태그 메모리(212)로부터 존재하는 하나의 캐시 라인 어드레스(구 캐시 라인 어드레스)를 배출시킴으로써 새로운 캐시 라인 어드레스를 위한 공간을 좌측 태그 메모리(212a) 내에 형성해야 한다. 구 캐시 라인 어드레스가 무효 상태인 경우, 버스와 연결된 캐시 메모리(114)는 더 이상 구 캐시 어드레스와 연관된 캐시 라인을 사용하지 않으며, 좌측 일관성 필터(200a)는 단순히 구 캐시 라인 어드레스를 새로운 캐시 라인 어드레스로 교체한다.

그러나 구 캐시 라인 어드레스가 공유 또는 점유 상태인 경우에는, 캐시 메모리(114)가 구 캐시 라인 어드레스를 무효화할 때까지, 좌측 일관성 필터(200a)는 좌측 태그 메모리(212a)로부터 구 캐시 라인 어드레스를 배출할 수 없다. 전술한 바와 같이, 좌측 태그 메모리(212a)는 좌측 버스(102)와 연결된 캐시 메모리(114) 내의 캐시 라인 어드레스의 슈퍼세트를 유지해야 하며, 따라서 좌측 태그 메모리(212a)가 구 캐시 라인 어드레스를 새로운 캐시 라인 어드레스로 교체하기 전에 캐시 메모리(114)내의 구 캐시 라인 어드레스가 먼저 무효화되어야 한다.

아래에서 보다 상세하게 기술하듯이, 좌측 일관성 필터(200a)는 무효화 버스 트랜잭션을 수행함으로써 좌측 버스(102)와 연결된 캐시 메모리(114) 내의 구 캐시 라인 어드레스를 무효화시킨다. 캐시 메모리(114)에게 구 캐시 라인 어드레스를 내부적으로 무효화하도록 지시하는 무효화 버스 트랜잭션이 좌측 버스 상에서 발생한다.

그러나 때로는 구 캐시 라인이 점유 상태가 될 수 있으며, 캐시 메모리(114) 중의 하나가 구 캐시 라인을 수정했을 수도 있다. 어떤 캐시 메모리(114)가 구 캐시 라인을 수정하면, 버스 판독 무효화 라인(BRIL) 트랜잭션이 수행된다. 캐시 메모리(114)가 구 캐시 라인을 수정하면, 버스 판독 무효화 라인 트랜잭션 동안, 캐시 메모리(114)는 버스 트랜잭션의 커맨드를 수신하고, 구 캐시 라인 어드레스에 해당하는 수정된 캐시 라인을 주 메모리(132)에 재기록한다. 수정된 캐시 라인을 주 메모리에 재기록한 후에, 좌측 일관성 필터(200a)는 좌측 태그 메모리(212a) 내의 구 캐시 라인 어드레스를 새로운 캐시 라인 어드레스로 대체한다. 따라서 때로는 태그 메모리(212) 내의 캐시 라인 어드레스의 슈퍼세트를 유지하기 위해, 캐시 메모리(114)는 구 캐시 라인이 대체되기 전에 수정된 데이터를 주 메모리(132)에 재기록해야 한다.

캐시 라인 어드레스(400) 및 각 태그 메모리(212)의 포맷이 도 4에 도시되어 있다. 캐시 라인 어드레스(400)는 개념적으로 2개의 부분으로 분할된다. 제1 부분은 태그 페이지 어드레스(402)를 포함하며, 제2 부분은 오프셋 어드레스(404)를 포함한다. 본 발명의 실시예에서, 태그 메모리(212) 내의 태그 엔트리(410)의 개수는 태그 페이지의 크기를 정의한다. 바람직하게, 각 태그 메모리(212)는 일정한 개수의 태그 엔트리(410)를 가지며, 따라서 동일한 크기의 태그 페이지를 가진다.

본 발명의 실시예에서, 태그 페이지 어드레스(402)는 캐시 라인 어드레스(400)를 포함하는 태그 페이지를 식별하며, 오프셋 어드레스(404)는 태그 페이지 내의 캐시 라인 어드레스(400)의 위치를 식별한다. 특히 캐시 라인 어드레스(400) 내의 상위 비트(high-order bit)는 태그 페이지 어드레스를 식별하고, 하위 비트(lower-order bit)는 오프셋 어드레스(404)를 식별한다.

일반적으로, 하위 비트는 태그 페이지 내의 캐시 라인 어드레스의 위치를 식별하기 때문에, 이 하위 비트는 인덱스(index)라 지칭된다. 예를 들어, 제1 캐시 라인 어드레스(400)에 있어서, 상위 어드레스 비트는 태그 페이지 어드레스(402)를 식별하고, 하위 어드레스 비트는 태그 페이지 내의 캐시 어드레스 라인(400)의 위치를 식별한다.

태그 페이지의 크기가 변할 수 있기 때문에, 태그 페이지 어드레스(402) 및 오프셋 어드레스(404)에 할당된 비트의 개수 또한 변할 수 있다. 본 발명의 실시예에서, 좌측 태그 메모리(212a) 및 우측 태그 메모리(212b)는 2^{16}개의 엔트리를 포함

하며, 오프셋 어드레스(404)는 하위 16 비트의 캐시 라인 어드레스(400)를 포함하며, 태그 페이지 어드레스(402)는 상위 15 비트의 캐시 라인 어드레스(400)를 포함한다. 전술한 바와 같이, I/O 태그 메모리(212c)는 크기가 더 작으며, 따라서 적은 수의 엔트리를 포함한다. 본 발명의 실시예에서, I/O 태그 메모리(212)는 32개의 엔트리를 포함한다.

다음의 표 1은 태그 메모리(212)의 서로 다른 크기 및, 이에 해당하는 캐시 라인 어드레스(400) 내의 태그 페이지 어드레스(402) 및 오프셋 어드레스(404)의 크기를 예시한다.

[표 1]

태그 메모리의 크기	태그 페이지 어드레스의 폭	오프셋의 폭
2^{16} 엔트리	15 비트	16 비트
2^{17} 엔트리	14 비트	17 비트
2^{18} 엔트리	13 비트	18 비트
2^{19} 엔트리	12 비트	19 비트

본 발명의 실시예에서, 일관성 필터(200)는 직접 매핑 기술(direct mapping techniques)을 사용하여 태그 메모리(200) 내에 캐시 라인 어드레스(400)를 저장한다. 직접 매핑 기술은 각 캐시 라인 어드레스(400)가 하나의 특정 태그 엔트리(410)에 매핑됨을 의미한다. 도 4는 2^{16}개의 태그 엔트리(410)를 포함하는 태그 메모리(212)의 예를 도시한다. 각 태그 엔트리(410)는 캐시 라인 어드레스(400)의 태그 페이지 어드레스(402)를 포함한다. 또한 각 태그 엔트리(410)는 일관성 상태 비트(412) 및 패리티 비트(414)를 포함한다. 패리티 비트(414)는 잘 알려진 패리티 오류 검출을 제공한다.

프로세서(112)가 캐시 라인 어드레스(400)를 액세스하면, 일관성 필터(200)는 오프셋 어드레스(404)를 사용하여 특정 태그 엔트리(410)를 식별한다. 그 후 일관성 필터(200)는 태그 페이지 어드레스(402)를 식별된 태그 엔트리(410)에 저장한다. 예를 들어, 좌측 버스(102) 상의 제1 프로세서(112a)가 제1 태그 페이지 내의 제1 캐시 라인 어드레스(400)를 액세스하면, 좌측 일관성 필터(200a)는 캐시 라인 오프셋 어드레스(404)를 사용하여, 좌측 태그 메모리(212a) 내에 제1 태그 엔트리(410)를 위치시킨다. 그 후 좌측 일관성 필터(212a)는 좌측 태그 메모리(212a) 내의 제1 태그 엔트리(410)에 태그 페이지 어드레스(402)를 저장한다.

태그 엔트리(410)는 캐시 라인 어드레스(400)의 태그 페이지 어드레스(402)를 저장하는 것 이외에도, 또한 캐시 라인 어드레스(400)에 대한 하나의 일관성 상태 비트 세트(412)를 저장한다. 바람직하게, 태그 엔트리(410) 내의 일관성 상태 비트(412)는 3개의 일관성 상태—무효 상태, 공유 상태 또는 점유 상태—를 포함하는 것이 바람직하다. 이들 3개의 일관성 상태는 2개의 일관성 상태 비트(412)로 해 표현된다. 다음의 표 2는 일관성 상태 비트(412)로 할당된 일관성 상태를 정의한다.

[표 2]

일관성 상태 비트	일관성 상태
00	무효화
01	공유
10	점유
11	(보류)

2. 태그 제어장치

캐시 라인 어드레스(400)를 태그 메모리(212)에 직접 매핑하기 위한 제어 로직이 태그 제어장치(300)에 위치한다. 좌측 버스(102)로 할당된 태그 제어장치(300)를 좌측 태그 제어장치(300a)라 지칭하고, 우측 버스(104)로 할당된 태그 제어장치(300)를 우측 태그 제어장치(300b)라 지칭하고, I/O 버스(106)로 할당된 태그 제어장치(300)를 I/O 태그 제어장치(300c)라 지칭한다. 예를 들어, 좌측 태그 제어장치(300)는 캐시 라인 어드레스(400)를 좌측 태그 메모리(212a)에 직접 매핑하는 직접 매핑 로직을 포함한다.

태그 제어장치(300) 내의 제어 로직은 캐시 라인을 캐시 메모리(114)에 직접 매핑하기 위해 사용되는 제어 로직과 유사하다. 따라서 태그 제어장치(300)는 당업자에게 널리 알려진 직접 매핑 로직을 사용한다. 그러나 이는 4중 세트 어소시어티브 매핑(four-way set associative mapping) 기술을 사용하는 Pentium Pro 프로세서(112) 내의 캐시 제어 로직과는 다르다.

아래에서 상세하게 기술하듯이, 일관성 필터(200)는 부분적으로 태그 메모리(212) 내의 일관성 상태 정보에 기초하여 크로스 버스 트랜잭션을 수행할 것인 지의 여부를 결정한다. 본 발명의 실시예에서, 또한 각 태그 제어장치(300)는 그 자신의 태그 메모리(212) 및 다른 버스에 할당된 태그 메모리(212)와 연결된다. 각 태그 제어장치(300)는 태그 메모리(212) 내의

태그 엔트리(410)를 액세스할 수 있으며, 각 태그 제어장치(300)는 할당된 태그 메모리(212) 내의 태그 엔트리만을 수정할 수 있다. 예를 들어, 좌측 태그 제어장치(300a)는 좌측 태그 메모리(212a), 우측 태그 메모리(212b) 및 I/O 태그 메모리(212c) 내의 태그 엔트리(410)를 액세스할 수 있으며, 좌측 태그 제어장치(300)는 좌측 태그 메모리(212a) 내의 태그 엔트리(410)만을 수정할 수 있다.

예를 들어, 좌측 버스(102)가 특정 캐시 라인 어드레스(400)에 대한 버스 트랜잭션을 전송하면, 좌측 태그 제어장치(300a)는 캐시 라인 어드레스(400)를 사용하여 우측 태그 메모리(212a) 및 I/O 태그 메모리(212c)로부터 대응하는 일관성 상태 비트(412)를 얻는다. 이 예에서는, 캐시 라인 어드레스(400)를 우측 태그 메모리(212b)에 직접 매핑하여, 우측 일관성 상태 비트를 구한다. 또한 캐시 라인 어드레스를 I/O 태그 메모리(212c)에 직접 매핑하여, I/O 일관성 상태 비트(412)를 구한다. 이하의 설명에서는, 다른 버스에 할당된 다른 태그 메모리(212)로부터 구해진 일관성 상태 비트(412)를 리모트(remote) 상태 비트(412)라 지칭한다. 좌측 태그 제어장치(300a)가 리모트 일관성 상태 비트(412)를 얻으면, 좌측 태그 제어장치(300a)는 이 리모트 일관성 상태 비트(412)를 사이클 인코더(302)로 전송한다.

3. 사이클 인코더

각 사이클 인코더(302)는 어떤 종류의 버스 트랜잭션이 버스(102, 104, 또는 106) 중의 하나의 버스 상에서 발생하는 지를 결정한다. 좌측 버스(102)를 모니터하기 위해 할당된 사이클 인코더(302)를 좌측 사이클 인코더(302a)라 지칭한다. 우측 버스(104)를 모니터하기 위해 할당된 사이클 인코더(302)를 우측 사이클 인코더(302b)라 지칭하며, I/O 버스(102)를 모니터하기 위해 할당된 사이클 인코더(302)를 I/O 사이클 인코더(302c)라 지칭한다. 그리하여 각 사이클 인코더(302)는 할당된 버스(102, 104, 또는 106)와 연관된 어드레스 및 제어 라인(108)을 모니터한다.

바람직하게 사이클 인코더(302)는 버스(102, 104, 및 106) 상에서 구현된 특정 버스 프로토콜과 관련된 공지된 기술을 사용하여, 버스 트랜잭션이 주 메모리(132)로부터 데이터를 판독하는지, 또는 주 메모리(132)에 데이터를 기록하는 지를 결정하기 위해 버스 제어 라인을 모니터한다. 본 발명의 실시예에서, 사이클 인코더(302)는 공지된 기술을 사용하여 Pentium Pro 버스 트랜잭션이 주 메모리(132)로부터 데이터를 판독하고 주 메모리(132)에 기록하는 지를 결정한다. 사이클 인코더(302)가 자신에 할당된 버스 상에서 발생하는 버스 트랜잭션의 형태를 결정하면, 사이클 인코더(302)는 버스 사이클의 형태를 규칙표(304)로 전송한다.

4. 규칙표

도 3에 도시된 규칙표(304)에 초점을 맞춰 설명하면, 규칙표는 캐시 일관성을 보장하기 위해 언제 크로스 버스 트랜잭션을 수행하는 지를 결정한다. 본 발명의 실시예에서, 좌측 버스(102)로 할당된 규칙표(304)를 좌측 규칙표(304a)라 지칭하고, 우측 버스(102)로 할당된 규칙표(304)를 우측 규칙표(304b)라 지칭하며, I/O 버스로 할당된 규칙표(304)를 I/O 규칙표(304c)라 지칭한다.

각 규칙표(304)는 사이클 인코더(302)로부터 구해진 리모트 일관성 상태 비트(412) 및 버스 트랜잭션 형태 정보를 평가한다. 본 발명의 실시예에서, 각 규칙표(304)는 SRAM에 위치한 큰 규모의 진리표(truth table)이다. 캐시 일관성을 보장하기 위해 실행될 필요가 있는 크로스 버스 트랜잭션 및 로컬 버스 트랜잭션의 형태는 각 규칙표 메모리 위치에 저장되어 있다.

또한 규칙표(304)는 스누핑 동작을 수행한다. 본 발명의 실시예에서, 리모트 태그 메모리가 캐시 라인의 공유된 복사본을 가지고 있음을 표시하면 규칙표(304)는 HIT# 신호를 인가할 수 있다. 인가된 HIT# 신호는 리모트 버스 상의 하나 이상의 캐시 메모리가 캐시 라인의 공유된 복사본을 가지고 있음을 표시한다. 규칙표(304)는 HIT# 신호를 인가하여 코드 판독 트랜잭션을 공유 상태로 설정할 수 있으며, 그 결과 다른 버스 상에서 앞으로 실행될 코드 판독 트랜잭션은 크로스 버스 트래픽을 발생시키지 않는다.

캐시 일관성을 보장할 필요가 있는 특정 크로스 버스 트랜잭션 코드는 시스템 액세스 제어장치(130)의 제어 하의 규칙표(304)에 로드된다. 시스템이 개시되는 동안, 본 발명의 실시예의 다중버스 시스템(100)은 크로스 버스 트랜잭션 코드를 규칙표(304)에 로드한다. 크로스 버스 트랜잭션 코드를 수정할 수 있기 때문에, 크로스 버스 트랜잭션은 서로 다른 다중버스 시스템(100)에 있어서 융통성있게 튜닝될 수 있다.

B. 버퍼 관리자

도 1 및 도 2에 도시된 버퍼 관리자(210), 중앙 요청 리스트(208), 및 데이터 인터페이스 버퍼(134)는 다중버스간의 통신을 최적화한다. 본 발명의 실시예에서, 중앙 요청 리스트(208), 버퍼 관리자(210) 및 데이터 인터페이스 버퍼(134)는 모든 버스(102, 104, 및 106)에 의해 액세스 가능한 (도시되지 않은) 메모리 셀의 다중포트화 영역(multiported pool)을 포함한다. 바람직하게, 임의의 버스(102, 104, 또는 106)는 독립적인 접속 경로(connection path)를 사용하지 않고서도 다른 모든 버스(102, 104, 또는 106)로부터 데이터를 판독하거나 데이터를 기록할 수 있다.

본 발명의 실시예에서, 각 버스 트랜잭션과 연관된 정보는 3개의 부분으로 분할되어, 3개의 서로 다른 메모리 셀—어드레스 셀, 요청 셀 및 데이터 셀—에 저장된다. 따라서 각 데이터 셀, 각 요청 셀, 각 어드레스 셀간에는 1대1 대응이 존재한다. 아래에서 상세하게 기술하듯이, 어드레스 셀은 버퍼 관리자(210)에 위치하며, 요청 셀은 중앙 요청 리스트(208)에 위치하며, 데이터 셀은 데이터 인터페이스 버퍼(134)에 위치한다.

본 발명의 실시예에 따른 버퍼 관리자(210)가 도 5a 및 도 5b에 도시되어 있다. 도 5a에 초점을 맞춰 설명하면, 버퍼 관리자(210)는 어드레스 셀 영역(address cell pool; 500), 어드레스 셀 우선순위 인코더(address cell priority encoder; 504), 좌측 FIFO 메모리(506a), I/O FIFO 메모리(506b) 및 우측 FIFO메모리(506c)를 포함한다. 각 어드레스 셀(500)은 다중포트화되어 있으며, 어드레스 셀 우선순위 인코더(504), 버스 마스터(202), 버스 슬레이브(204), 메모리 제어장치(206)와 통신하며, 도 5b에 도시되어 있듯이 비교기(510)와도 통신한다.

본 발명의 실시예에 따른 버퍼 관리자(210)는 64개의 어드레스 셀(500)을 포함한다. 도 5a에 도시되어 있듯이, 각 어드레스 셀(500)은 "사용중(in-use)" 비트(502), 메모리 어드레스(504) 및 데이터 셀 상태 비트(505) 세트를 포함한다. 사용중 비트(502)는 특정 어드레스 셀(500)이 사용가능한 지의 여부를 표시한다. 본 발명의 실시예에서, 사용중 비트(502)는 어드레스 셀(500)이 사용 중인지 아니면, 미사용 상태인지를 표시하도록 설정되어 있다. 메모리 어드레스(504)는 메모리 어드레스(504)를 포함하며, 데이터 셀 상태 비트(505)는 버스 트랜잭션의 형태를 표시한다. 특히 데이터 셀 상태 비트(505)는 무효화 버스 트랜잭션이 수행되는 지의 여부를 표시한다.

어드레스 셀 우선순위 인코더(504)에 초점을 맞춰 설명하면, 어드레스 셀 우선순위 인코더(504)는 어느 어드레스 셀(500)이 사용중이며, 어는 어드레스 셀(500)이 새로운 버스 트랜잭션을 수신할 수 있는 미사용 상태의 어드레스(500)인지를 결정한다. 본 발명의 실시예에서, 어드레스 셀 우선순위 디코더(504) 내의 로직은 각 어드레스 셀(500) 내의 사용중 비트(502)를 평가함으로써 미사용 상태의 어드레스 셀(500)을 결정한다. 사용중 비트(502)가 설정되면, 어드레스 셀(500)은 사용 중 상태가 된다. 사용중 비트(502)가 설정되지 않으면, 어드레스 셀은 미사용 상태가 된다. 미사용 상태의 셀은 라운드 로빈 방식(round robin fashion)으로 선택된다.

어드레스 셀 우선순위 인코더(504)는 미사용 상태의 어드레스 셀(500)을 결정할 뿐만이 아니라, 미사용 상태의 어드레스 셀(500)을 서로 다른 버스(102, 104, 및 106)로 할당한다. 바람직하게, 어드레스 셀 우선순위 인코더(504)는 라운드 로빈 방식을 사용하여 미사용 상태의 어드레스 셀(500)을 서로 다른 버스(102, 104, 및 106)로 할당한다. 아래에서 보다 상세하게 기술하고 있듯이, 버스 트랜잭션 정보가 미사용 상태의 어드레스 셀(500)에 저장되면, 사용 중 비트(502)는 이 어드레스 셀(500)이 미사용 상태가 아님을 표시하도록 설정된다.

예를 들어, 제1, 제2, 및 제3 어드레스 셀(500)을 미사용 상태로 가정한다. 우선순위 인코더(504)는 제1, 제2, 및 제3 어드레스 셀(500)이 미사용 상태임을 결정하고, 제1 어드레스 셀(500)을 좌측 버스(102)로 할당하고, 제2 어드레스 셀(500)을 우측 버스(104)로 할당하고, 제3 어드레스 셀(500)을 I/O 버스(106)로 할당한다. 제4 어드레스 셀(500)이 미사용 상태가 되면, 어드레스 셀 우선순위 인코더(504)는 좌측 버스(102)로 역순환(cycles back)시키고, 제4 어드레스 셀(500)을 좌측 버스(500)로 할당한다. 본 발명의 실시예에서는 어드레스 셀 우선순위 인코더(504)가 미사용 상태의 어드레스 셀을 서로 다른 버스로 할당하지만, 당업자는 어드레스 셀 우선순위 인코더(504)가 광범위한 할당 방식을 사용하여 미사용 상태의 어드레스 셀(500)을 서로 다른 버스(102, 104, 및 106)로 할당할 수 있다는 것을 이해하게 될 것이다.

버퍼 관리자의 FIFO 메모리(506)에 초점을 맞춰 설명하면, FIFO 메모리(506)는 할당된 어드레스 셀(500)을 식별하는 어드레스 셀 식별자(identifier)를 임시로 저장한다. 어드레스 셀 식별자는 할당된 어드레스 셀(500)의 메모리 위치를 포함하는 데이터 변수이다. 본 발명의 실시예에서, 어드레스 셀 식별자는 64개의 어드레스 셀(504)을 식별한다. 아래에서 보다 상세하게 기술하고 있듯이, 버스 마스터(202)는 FIFO 메모리(506) 내의 어드레스 셀 식별자를 사용하여, 어드레스 셀 식별자에 의해 식별되는 어드레스 셀 메모리 위치를 액세스한다.

본 발명의 실시예에서, 각 FIFO 메모리(508)는 버스 슬레이브(204) 및 일관성 필터(200)로 출력한다. 즉 좌측 FIFO 메모리(506a)는 좌측 버스 슬레이브(204a) 및 좌측 일관성 필터(200a)로 출력한다. 우측 FIFO 메모리(506b)는 우측 버스 슬레이브(204b) 및 우측 일관성 필터(200b)로 출력하고, I/O FIFO 메모리(506c)는 I/O 버스 슬레이브(204c) 및 I/O 일관성 필터(200c)로 출력하고, 버스 슬레이브(204) 중의 하나의 슬레이브 또는 일관성 필터(200) 중의 하나의 필터가 다른 버스에 버스 트랜잭션을 전송하기를 희망하면, 이들 장치는 이들의 할당된 FIFO 메모리(506)로부터 어드레스 셀 식별자를 구한다.

그리하여 본 발명의 실시예에 의한 버퍼 관리자(210)는 모든 버스(102, 104, 및 106)와 상호접속하는 어드레스 셀(500) 영역을 제공한다. 개별적인 버스 경로 대신에 이러한 하나의 어드레스 셀 영역(500)을 제공하는 것은 결과적으로 시스템 구성을 간단하게 한다. 또한 어드레스 셀 우선순위 인코더(504) 및 FIFO 메모리(506)는 미사용 상태의 어드레스 셀(500)이 버스(102, 104, 및 106) 사이에 균등하게 분배되도록 한다.

도 5b에 도시된 본 발명의 다른 태양에서, 본 발명의 실시예에 의한 버퍼 관리자(210)는 어드레스 충돌(conflict)을 식별하는 복수의 어드레스 비교기(510)를 포함한다. 2개의 서로 다른 버스 트랜잭션이 동일한 데이터 값과 연관되어 있으면, 일반적으로 거의 동시에 어드레스 충돌이 발생한다. 이러한 상황에서, 2개의 어드레스 셀(500)은 어드레스 셀(500) 내의 2개의 서로 다른 버스 트랜잭션에 대하여, 동일한 메모리 어드레스(504)를 포함할 수 있다. 이러한 상황에서는 부적절한 버스 트랜잭션이 발생할 수 있다.

하나의 버스 슬레이브(204)가 버스 트랜잭션을 수신하면, 버스 슬레이브(204)는 상기 버스 트랜잭션과 연관된 어드레스를 버스 슬레이브(204)와 동일한 버스에 할당된 어드레스 비교기(510)에 전송한다. 어드레스 비교기(510)는 새로운 메모리 어드레스(504)를 사용중인 어드레스 셀(500)에 존재하는 모든 메모리 어드레스(504)와 비교한다. 사용중인 어드레스 셀(500) 내에서 동일한 메모리 어드레스가 검출되면, 어드레스 비교기(510)는 어드레스 충돌이 발생했다는 것을 버스 슬레이브(204)에 통지하기 위해 출력을 발생시킨다. 그 후 버스 슬레이브(204)는 프로세서(112)에 게 어드레스 충돌을 발생시킨 버스 트랜잭션을 개시하는 재시도(retry) 신호를 전송한다. 이후에 프로세서(112)는 버스 트랜잭션을 개시다.

본 발명의 실시예에서, 어드레스 비교기(510) 세트가 각 버스에 할당된다. 즉 좌측 어드레스 비교기 세트(510a)가 좌측 버스(102)에 할당되고, 우측 어드레스 비교기 세트(510c)는 우측 버스(104)에 할당되며, I/O 어드레스 비교기 세트(510c)는 I/O 버스(106)에 할당된다. 버스 슬레이브(204)가 새로운 메모리 어드레스(504)를 수신하면, 버스 슬레이브(204)는 이 새로운 메모리 어드레스(504)를 어드레스 비교기(510)에 전송한다. 그후 어드레스 비교기(501)는 사용중인 어드레스 셀(500) 내에 존재하는 메모리 어드레스(504)를 평가하여 어드레스 충돌이 존재하는 지의 여부를 결정한다.

예를 들어, 우측 버스 슬레이브(202b)는 I/O 버스(106) 상에서 하나의 I/O 브리지(120) 중 하나에 입력되는 I/O 트랜잭션을 수신할 수 있다. 우측 버스 슬레이브(202b)는 I/O 트랜잭션과 연관된 메모리 어드레스를 수신하는 즉시, 메모리 어드레스를 우측 어드레스 비교기(510c)로 전송한다. 우측 어드레스 비교기(510c)는 이 메모리 어드레스를 사용중인 어드레스 셀(500) 내에 존재하는 메모리 어드레스(504)와 비교한다. 이 메모리 어드레스의 복사본이 사용중인 어드레스 셀(500)

내에 존재하면, 어드레스 비교기는 우측 버스 슬레이브(202b)에게 어드레스 충돌이 발생했다는 것을 통지하기 위해 신호를 재전송한다. 이 실시예에서, 우측 버스 슬레이브(202b)는 프로세서(112c 또는 112d)에게 이후에 I/O 트랜잭션을 재시도하도록 지시한다.

태그 메모리(212) 중의 하나의 메모리 내의 엔트리를 무효화하기 위한 무효화 버스 트랜잭션을 수행할 필요가 있는 경우, 그 버스가 사용중이 아닌 경우가 될 때까지 무효화 동작을 지연하는 것이 바람직하다. 이에 대해 하나의 방법은 버퍼 관리자 내에 구 캐시 라인 어드레스(무효화될 캐시 라인 어드레스)를 저장하는 것이다. 본 발명의 실시예에서, 버퍼 관리자(210) 내의 어드레스 셀(500)은 캐시 라인의 전체 메모리 어드레스를 포함하며, 여기서 전체 메모리 어드레스는 캐시 라인과 연관된 태그 페이지 어드레스(402) 및 오프셋 어드레스(404)를 포함한다. 구 캐시 라인 어드레스를 어드레스 셀(500) 중의 하나의 어드레스 셀에 더하는 것은 구 캐시 라인 어드레스가 실제로 무효화되지 않았음에도 불구하고 다중버스 시스템(100)이 새로운 버스 트랜잭션을 처리할 수 있도록 한다.

다른 버스 트랜잭션이 버퍼 관리자(210) 내에 존재하는 구 캐시 라인 어드레스(400)와 연관되어 있다고 가정하면, 버스 트랜잭션이 버스 슬레이브(204) 중의 하나의 슬레이브에 의해 수신되는 경우, 버스 슬레이브(204)는 이러한 버스 트랜잭션과 연관된 어드레스를 어드레스 비교기(510)에 전송한다. 어드레스 비교기(510)는 버스 트랜잭션과 연관된 오프셋 어드레스(404)를 구 캐시 라인 어드레스의 오프셋 어드레스(404)와 비교한다. 어드레스 충돌이 존재하면, 비교기는 데이터 셀 상태 비트(505)를 평가한다. 데이터 셀 상태 비트(505)가 구 캐시 라인 어드레스를 무효화할 필요가 있음을 표시하면, 구 캐시 라인 어드레스(400)를 무효화할 수 있을 때까지 새로운 버스 트랜잭션을 재시도한다. 이러한 접근방법 이외의 다른 방법에 대해서는 "다른 실시예"라는 이름 하의 장에서 상세하게 기술한다.

C. 중앙 요청 리스트

도 6에 도시된 본 발명의 실시예에 의한 중앙 요청 리스트(208)에 초점을 맞춰 설명하면, 중앙 요청 리스트(208)는 요청 셀 영역(600), 좌측 버스 우선순위 인코더(602a), 우측 버스 우선순위 인코더(602b), 및 I/O 버스 우선순위 인코더(602c)를 포함한다. 각 요청 셀(600)은 다중포트화되며, 각 어드레스 셀 우선순위 인코더(602)와 통신한다. 본 발명의 실시예에 의한 중앙 요청 리스트(208)에서는, 64개의 요청 셀(600)이 존재한다. 또한 64개의 어드레스 셀(500)과 64개의 요청 셀(600) 사이에는 1대1 대응이 존재한다.

각 요청 셀(600)은 타겟 버스 식별자(604), 버스 트랜잭션 코드(606), 및 오너(owner) 버스 식별자(608)를 포함한다. 타겟 버스 식별자(604)는 목적지 버스(102, 104, 106)를 식별한다. 아래에서 보다 상세하게 기술하고 있듯이, 버스 식별자(604)가 미리 정의되어 있는 것은 아니며, 오히려 본 발명의 실시예에서는 버스 식별자(604)를 우측 버스(102), 좌측 버스(104) 또는 I/O 버스(106) 중의 하나를 식별할 필요가 있는 것으로 설정한다. 예를 들어, 타겟 버스 식별자(604)는 버스 트랜잭션 코드(606)가 우측 버스(104)에 대한 것인가를 지정할 수 있다. 한편 버스 트랜잭션 코드는 목적지 버스 상에서 수행될 버스 트랜잭션의 형태를 식별한다. 본 발명의 실시예에서, 버스 마스터(202)는 버스 트랜잭션 코드(606)에 의해 식별된 버스 트랜잭션을 수행한다. 오너 버스 식별자(owner bus identifier; 608)는 발신(originating) 버스를 식별한다. 예를 들어, 오너 버스 식별자(608)는 좌측 버스(102) 상에서 발신된 버스 트랜잭션 코드(606)를 표시할 수 있다.

버스 우선순위 인코더는 각 요청 셀(600) 내의 타겟 버스 식별자(604)를 평가하여, 어느 요청 셀이 서로 다른 버스를 지정하는 지를 결정한다. 그후, 버스(102, 104, 및 106) 중의 하나의 버스와 연관된 버스 마스터(202)는 특정 요청 셀(600) 내에서 식별된 버스 트랜잭션을 수행한다. 예를 들어, 좌측 버스 우선순위 인코더(602)는 모든 타겟 버스 식별자(604)를 평가하여, 어느 요청 셀(600)이 좌측 버스(102)에 대한 것인지를 식별한다. 우측 버스 우선순위 인코더(602b)는 모든 타겟 버스 식별자(604)를 평가하여, 요청 셀(600)이 우측 버스(104)에 대한 것인지를 식별한다. I/O 버스 우선순위 인코더(602c)는 모든 타겟 버스 식별자(604)를 평가하여, 어느 요청 셀(600)이 I/O 버스(106)에 대한 것인지를 식별한다.

목적지 버스(102, 104, 및 106)를 식별하는 것 이외에, 또한 각 버스 우선순위 인코더(602)는 특정 버스(102, 104, 또는 106)와 연관된 요청 셀(500) 중 가장 높은 우선순위를 가지는 요청 셀을 결정한다. 본 발명의 실시예에서, 각 버스 우선순위 인코더(602)는 라운드 로빈 로직을 사용하여 우선순위가 가장 높은 요청 셀(600)을 결정한다. 각 버스 우선순위 인코더(602)가 우선순위가 가장 높은 요청 셀(600)을 할당된 버스에 전송하면, 각 버스 우선순위 인코더(602)는 각 요청 셀(600)에 대하여 우선순위를 순환식으로 할당한다. 라운드 로빈 로직은 각 버스 우선순위 인코더(602)가 가장 높은 우선순위를 순차적으로 모든 버스 요청 셀(600)에 할당한다. 이러한 라운드 로빈 회로는 당업자에게 공지되어 있다.

예를 들어, 우측 버스 우선순위 인코더(602b)가 우측 버스(104)에 대하여 2개의 요청 셀(600)을 식별했다고 가정한다. 우측 버스 우선순위 인코더(602b)는 제1 요청 셀(600)에 가장 높은 우선순위를 할당하고, 제1 요청 셀 식별자를 우측 버스 마스터(202b)로 전송한다. 그리고 나서 우측 버스 우선순위 인코더(602b)는 제2 요청 셀(600)에 가장 높은 우선순위를 할당하고, 제2 요청 셀 식별자를 우측 버스 마스터(202b)로 전송한다. 우측 버스 우선순위 인코더(602b)가 우측 버스(104)에 할당된 요청 셀(600)의 끝에 도달하면, 우측 버스 우선순위 인코더는 요청 셀(500) 영역의 처음으로 되돌아간다.

그리하여 중앙 요청 리스트(208)는 모든 버스(102, 104, 및 106)를 상호연결시키는 요청 셀(600) 영역을 제공한다. 개별적인 버스 연결 경로와 비교하여, 이러한 요청 셀(600) 영역은 결과적으로 시스템 구성을 간단하게 한다. 또한 중앙 요청 리스트(208) 내의 버스 우선순위 인코더(602)는 요청 셀 내의 각 버스 트랜잭션이 버스(102, 104 및 106)에 의해 수행됨을 보장한다.

Ⅳ. 데이터 인터페이스 버퍼

도 7에 도시된 본 발명의 실시예에 의한 데이터 인터페이스 버퍼(134)에 초점을 맞춰 설명하면, 데이터 인터페이스 버퍼(134)는 데이터 셀 영역(700)과, 제어 및 인덱싱 메커니즘(indexing mechanism; 702)을 포함한다. 각 요청 셀(600)은 다중포트화되며, 홀수 주 메모리 모듈(132a) 및 짝수 주 메모리 모듈(132b)과 연결된 데이터 라인(110a, 110b, 및 110c) 및 데이터 라인(142a 및 142b)과 통신한다. 본 발명의 실시예에 의한 중앙 요청 리스트(208)에는 64개의 데이터 셀(700)이 존재한다. 또한 64개의 데이터 셀(700), 64개의 요청 셀(600) 및 64개의 어드레스 셀(500) 사이에는 1대1 대응이 존재한다. 각 데이터 셀(700)은 버스 트랜잭션과 연관된 데이터 값을 포함한다.

제어 및 인덱싱 메커니즘(702)은 버스 슬레이브(204)로부터 DIB 제어 라인(140)을 수신한다. DIB 제어 라인(140)은 특정 데이터 셀(700) 및 버스(102, 104, 106) 중 하나의 버스를 식별한다. 예를 들어, I/O 버스 슬레이브(204)가 새로운 버스 트랜잭션을 어드레스 셀(500) 중 하나의 어드레스 셀과 요청 셀(600) 중 하나의 요청 셀에 더하면, I/O 버스 슬레이브(204)는 또한 새로운 버스 트랜잭션과 연관된 데이터를 해당 데이터 셀(700)에 더한다. 이 예에서, I/O 버스 슬레이브(204)는 적당한 데이터 셀(700)을 식별하기 위해 DIB 제어 라인을 인가한다. 그후, 제어 및 인덱싱 메커니즘(702)은 I/O 데이터 라인(110c)으로부터 데이터를 수신하기 위해 적당한 데이터 셀(700)을 인에이블한다.

그리하여 데이터 인터페이스 버퍼는 임의의 버스(102, 104, 106)에 의해 액세스 가능한 데이터 셀(700) 영역을 제공한다. 개별적인 버스 상호연결 대신에 데이터 셀 영역(700)을 제공하는 것은 결과적으로 시스템 구성을 간단하게 하며, 성능을 증가시킨다.

Ⅴ. 캐시 일관성 유지

일관성 필터(200)에 의해 모니터된 버스 트랜잭션은 버스 판독 라인(Bus Read Line; BRL) 커맨드 및 버스 판독 무효화 라인(Bus Read Invalidate; BRIL) 커맨드를 포함한다. 버스 판독 라인(BRL) 커맨드는 인스트럭션 코드의 캐시 라인 또는 주 메모리(132)로부터의 데이터를 무효화한다. 버스 판독 무효화 라인(BRIL) 커맨드는 캐시 라인을 무효화한다. 본 발명의 실시예에 의한 규칙표(304)가 이들 버스 커맨드에 적용되지만, 당업자는 규칙표(304)가 이와 다른 버스 커맨드에 적용될 수 있으며, 따라서 본 발명의 실시예의 버스 커맨드에 한정되지 않는다는 것을 인식하게 될 것이다.

A. 버스 판독 라인 커맨드 처리

도 8은 버스(102, 104 또는 106) 중의 하나의 버스가 버스 판독 라인(BRL) 커맨드를 실행하는 경우, 캐시 일관성을 유지하기 위한 순서도를 도시한다. BRL 커맨드가 실행되는 동안, 프로세서들(112) 중의 하나는 원하는 캐시 라인의 캐시 라인 어드레스(400)를 할당된 일관성 필터(200) 및 버스 슬레이브(204)로 전송한다.

버스 슬레이브(204)가 캐시 라인 어드레스를 수신하면, 버스 슬레이브(204)는 캐시 라인 어드레스를 동일한 버스에 할당된 어드레스 비교기(510)로 전송한다. 어드레스 비교기(510)는 오프셋 어드레스(404)를 사용중인 어드레스 셀(500) 내에 존재하는 메모리 어드레스(504)의 오프셋 부분과 비교하여, 어드레스 충돌이 존재하는 지의 여부를 결정한다. 또한 어드레스 비교기(510)는 데이터 셀 상태 비트(505)를 평가하여, 어드레스 충돌을 발생시키는 어드레스 셀(500)이 무효화 트랜잭션과 연관되어 있는 지를 관찰한다. 만약 연관되어 있다면, 어드레스 비교기(510)는 어드레스 충돌 신호를 버스 슬레이브(204)로 전송한다. 그후 버스 슬레이브(204)는 요청 프로세서(112a)에게 이후에 BRL 커맨드를 재시도할 것을 지시한다.

본 발명의 실시예의 단계(802)에서, 어드레스 비교기가 어드레스 충돌이 존재하지 않음을 표시하면, 사이클 인코더(302)는 버스(102, 104, 또는 106)가 BRL 커맨드를 수행하는 지의 여부를 결정한다. 또한 버스(102, 104, 또는 106)에 할당된 태그 제어장치(300)는 BRL 커맨드에서 식별된 캐시 라인 어드레스(400)를 사용하여, 다른 버스(102, 104, 또는 106)로 할당된 태그 메모리(212)(또는 리모트 태그 메모리(212)) 내의 캐시 라인 어드레스(400)를 액세스한다. 이 리모트 태그 메모리(212)는 리모트 캐시 라인 어드레스(400)가 무효 상태인지, 공유 상태인지, 또는 점유 상태인지를 표시하는 리모트 일관성 상태 비트(412)를 되돌린다.

단지 하나의 리모트 태그 메모리(212)만이 점유 상태의 특정 캐시 라인 어드레스를 포함할 수 있다. 그러나 하나 이상의 리모트 태그 메모리(212)는 공유 또는 무효 상태의 특정 캐시 라인 어드레스를 포함할 수 있다. 하나의 리모트 태그 메모리(212)가 공유 상태의 캐시 라인 어드레스의 복사본을 포함하고 다른 리모트 태그 메모리(300)가 무효 상태의 동일한 캐시 라인 어드레스의 복사본을 포함하면, 공유 상태의 일관성 상태 비트(412)가 우선순위를 가진다. 아래에서 보다 상세하게 기술하듯이, 하나 이상의 리모트 태그 메모리(300)가 공유 상태의 특정 캐시 라인 어드레스의 복사본을 포함하고 있으면, 일관성 규칙(304)은 하나 이상의 리모트 버스(102, 104, 또는 106) 상에서 리모트 버스 트랜잭션을 수행하여 캐시 일관성을 보장할 수 있다. 또한 리모트 태그 메모리(300) 중 어느 것도 특정 캐시 라인 어드레스(400)를 포함하고 있지 않으면, 캐시 라인 어드레스(400)와 연관된 리모트 일관성 상태 비트(412)는 무효 상태로 처리된다.

예를 들어, 단계 802에서 좌측 버스(102) 상의 제1 프로세서(112a)가 특정 캐시 라인 어드레스(400)에 대하여 BRL 커맨드를 발생시키면, 좌측 사이클 인코더(302a)는 좌측 버스가 BRL 커맨드를 수행하고 있는 지를 결정한다. 또한 좌측 태그 제어장치(300)는 이 캐시 라인 어드레스(400)를 사용하여, 리모트 우측 태그 메모리(212b) 및 리모트 I/O 태그 메모리(212c)를 액세스한다. 태그 제어장치(300)는 우측 태그 메모리(212b) 및 I/O 태그 메모리(212c)로부터 리모트 일관성 상태 비트(412)를 구하는데, 이 리모트 일관성 상태 비트는 우측 태그 메모리(212b) 및 I/O 태그 메모리(212c) 내의 캐시 라인 어드레스(400)가 무효 상태인지, 공유 상태인지, 또는 점유 상태인지를 표시한다. 우측 태그 메모리(212b) 및 I/O 태그 메모리(212c) 중의 어느 하나도 캐시 라인 어드레스(400)를 포함하고 있지 않으면, 좌측 태그 제어장치(300a)는 리모트 일관성 상태 비트(412)를 무효 상태로 처리한다.

본 발명의 실시예에 의한 규칙표(304)의 동작에 초점을 맞춰 설명하면, 로컬 또는 리모트 목적지를 사용하여, 로컬 버스(102, 104, 또는 106)와 연관된 사이클 인코더(302)는 로컬 규칙표(304)에 버스 트랜잭션의 형태(BRL 커맨드) 및 리모트 캐시 상태 비트(412)의 상태를 전송한다. 리모트 일관성 상태 비트(412)가 리모트 캐시 라인 어드레스(400)가 무효 상태임을 표시하면, 로컬 규칙표(304)는 단계(804)로 진행된다. 본 발명의 실시예에서, 리모트 일관성 상태 비트(412)가 무효 상태이면, 로컬 규칙표(304)는 어떠한 리모트 버스(102, 104, 또는 106) 상에서도 BRL 크로스 버스 트랜잭션을 발생시키지 않는다. 오히려 로컬 규칙표(304)는 로컬 태그 메모리(300) 내의 캐시 라인 어드레스의 슈퍼세트(400)를 유지한다.

단계(804)에서, 로컬 프로세서(112)는 임의의 로컬 프로세서(112)가 원하는 캐시 라인의 복사본을 포함하고 있는 지를 결정하기 위해, 내부 캐시 메모리(114)를 질의(interrogate)한다. 본 발명의 실시예에서, 또한 버스 스누핑 동작이 수정된

캐시 라인이 프로세서의 캐시 메모리(114) 내에 존재한다는 것을 표시하면, 프로세서(112) 중 하나가 HITM# 신호를 인가한다. HITM# 신호는 어드레스 및 제어 라인(108) 중의 하나이며, 최신의 캐시 라인을 가지는 프로세서(112)가 주 메모리(132)에 캐시 라인을 재기록할 필요가 있다는 것을 표시한다.

단계(806)에서, 수정된 캐시 라인을 가지는 로컬 프로세서는 수정된 캐시 라인을 주 메모리(132)에 재기록함으로써, 버스 트랜잭션에 응답하고 이와 동시에 주 메모리(132)를 갱신한다. 단계(808)에서, 로컬 규칙표(304)는 로컬 태그 제어장치(300)에게 로컬 태그 메모리(212) 내의 일관성 상태 비트(412)를 공유 상태로 설정하도록 지시한다. 그 후에 로컬 규칙표(304)는 마지막 단계인 단계(810)로 진행한다.

단계(804)에서, 로컬 프로세서(112)가 캐시 라인의 수정된 복사본을 포함하지 않으면, 규칙표(304)는 단계(812)로 진행된다. 단계(812)에서, 로컬 규칙표(304)는 로컬 버스(102, 104, 106) 상의 HIT# 신호를 평가한다. 전술한 바와 같이, 로컬 프로세서(112)가 자신의 캐시 메모리(114) 내에 캐시 라인의 수정되지 않은 복사본을 포함하는 경우, 로컬 프로세서(112)는 HIT# 신호를 발생한다. 로컬 프로세서(112)가 캐시 라인의 수정되지 않은 복사본을 포함하고 있는 경우, 로컬 규칙표(304)는 단계(814)로 진행된다.

단계(814)에서, 로컬 버스 슬레이브(204)는 주 메모리(132)로부터 캐시 라인을 구한다. 이 캐시 라인 어드레스가 홀수이면, 로컬 버스 슬레이브(204)는 홀수 주 메모리(132a)로부터 캐시 라인을 구하여, 캐시 라인 어드레스가 짝수이면, 로컬 버스 슬레이브(204)는 짝수 주 메모리(132b)로부터 캐시 라인을 구한다. 그후, 로컬 버스 슬레이브(204)는 캐시 라인을 요청 프로세서(112)로 전송한다.

단계(808)에서, 로컬 규칙표(304)는 로컬 태그 제어장치(300)에게 로컬 태그 메모리(300) 내의 일관성 상태 비트(412)를 공유 상태로 설정할 것을 지시한다. 그후 로컬 규칙표(304)는 마지막 단계인 단계(810)로 진행된다.

단계(812)에서, 로컬 프로세서(112)가 HIT# 신호 또는 HITM# 신호를 인가하지 않으면, 단계(816)에 나타낸 바와 같이, 캐시 라인은 더 이상 로컬 프로세서(112)에 존재하지 않는다. 단계(818)에서, 로컬 태그 메모리(212)가 캐시 라인 어드레스(400)의 복사본을 가지고 있지 않으면, 규칙표(304)는 캐시 라인 어드레스(400)를 로컬 태그 메모리(212)에 더한다. 전술한 바와 같이, 로컬 태그 메모리는 로컬 태그 메모리에 존재하는 캐시 라인 어드레스를 배출시켜 새로운 캐시 라인 어드레스를 위한 공간을 형성할 필요가 있을 수 있다. 새로운 캐시 라인 어드레스를 위한 공간을 형성하였다면, 새로운 캐시 라인을 로컬 태그 메모리에 더하기 전에 구 캐시 라인 어드레스를 먼저 무효화시킨다. 그후 좌측 버스 슬레이브는 주 메모리(132)로부터 원하는 캐시 라인을 구하고, 이를 요청 프로세서(112)에 전송한다.

단계(822)에서 로컬 규칙표(304)는 로컬 태그 제어장치(300)에게 로컬 태그 메모리(212) 내의 캐시 라인의 일관성 상태 비트(412)를 점유 상태로 설정하도록 지시한다. 그러나 로컬 버스 상의 프로세서(112) 중 하나가 HIT# 신호를 인가하면, 로컬 태그 메모리(212)는 공유 상태로 설정된다. 또한 BRL 커맨드가 코드용이고, 리모트 상태가 점유되지 않으면, 규칙표(304)는 HIT# 신호를 인가한다. 어떠한 HIT# 신호도 검출되지 않으면, BRL 커맨드를 개시한 프로세서는 자신의 내부 캐시 상태를 공유 상태로 표시한다. HITM# 신호가 인가되면, 숨겨진(implicit) 재기록이 수행되고 로컬 태그 메모리(212)가 공유상태로 설정된다. 로컬 규칙표(304)는 마지막 단계인 단계(810)로 진행된다.

단계(802)에서, 리모트 태그 메모리(300)가 리모트 캐시 라인 어드레스(400)가 공유 상태임을 표시하는 경우에 발생하는 것에 초점을 맞춰 설명한다. 리모트 일관성 상태 비트(412)가 캐시 라인 어드레스(400)가 공유 상태임을 표시하면, 로컬 규칙표(304)는 리모트 버스(102, 104, 또는 106) 상에서 BRL 크로스 버스 트랜잭션을 발생시키지 않는다. 오히려 로컬 규칙표(304)는 주 메모리(132)로부터 원하는 캐시 라인을 구하고, 로컬 태그 메모리(300)를 수정한다.

단계(830)에서, 코드 판독에 대하여, 로컬 규칙표(304)는 로컬 버스(102, 104, 106) 상에서 HIT# 신호를 인가한다. 그후 로컬 규칙표(304)는 단계(832)로 진행된다. 단계(832)에서, 로컬 규칙표(304)는 주 메모리(132)로부터 원하는 캐시 라인을 구한다. 원하는 캐시 라인 어드레스가 로컬 태그 메모리(212) 내에 존재하지 않으면, 태그 제어장치는 새로운 캐시 라인 어드레스(400)를 로컬 태그 메모리(212)에 더한다. 전술한 바와 같이, 태그 제어장치(300)는 구 캐시 라인 어드레스(400) 중의 하나를 배출시켜, 새로운 캐시 라인 어드레스(400)에 대한 새로운 공간을 형성한다.

단계(808)에서, 로컬 규칙표(304)는 캐시 라인의 일관성 상태 비트(412)를 공유 상태로 설정하고, 마지막 단계인 단계(810)로 진행된다.

단계(802)에서, 리모트 태그 메모리(212) 중의 하나가 리모트 캐시 라인 어드레스(400) 중의 하나가 점유 상태임을 표시하는 경우에 발생하는 것에 초점을 맞춰 설명한다. 리모트 일관성 상태 비트(412)가 리모트 캐시 라인 어드레스(400) 중의 하나가 점유 상태임을 표시하면, 리모트 버스는 원하는 최신 버전의 캐시 라인을 가지게 되며, 따라서 리모트 버스로부터 최신 캐시 라인을 구해야 한다. 단계(802)에서, 규칙표(304)는 캐시 일관성을 보장하기 위해 필요한 BRL 크로스 버스 트랜잭션을 발생시켜야 한다. 본 발명의 실시예에 의한 무효화, 공유 및 점유 프로토콜에서는, 주어진 시각에 단지 하나의 버스만이 그 자신의 캐시 라인을 소유할 수 있다. 따라서 리모트 캐시 라인은 서로 다른 일관성 상태로 변해야 한다.

단계(840)에서, 로컬 규칙표(304)는 로컬 BRL 커맨드가 지연될 수 있는 지의 여부를 결정한다. 로컬 BRL 커맨드를 지연하는 것은 로컬 버스가 계속하여 버스 트랜잭션을 전송할 수 있도록 한다. 전술한 바와 같이, 버스 트랜잭션을 발행하는 프로세서(104)는 DEN# 신호를 인가함으로써 상기 버스 트랜잭션을 지연할 수 있는 지의 여부를 표시한다. DEN# 신호가 인가되면, 로컬 규칙표(304)는 단계(842)로 진행되며, 로컬 버스 슬레이브(204)에게 로컬 BRL 커맨드를 지연하도록 지시한다. 그후 단계(844)에서, 로컬 규칙표(304)는 리모트 버스(102, 104, 또는 106) 중의 하나의 버스 상에서 BRL 커맨드를 수행하도록 지시한다.

단계(840)에서, 버스 트랜잭션을 지연할 수 없으면, 로컬 규칙표(304)는 단계(846)로 진행되며, 로컬 버스 슬레이브(204)에게 로컬 BRL 커맨드를 기능정지(stall)시키도록 지시한다. 그후 단계(844)에서, 구 로컬 규칙표(304)는 리모트 버스(102, 104, 또는 106) 중의 하나의 버스 상에서 BRL 커맨드를 수행하도록 지시한다.

단계(844)에서, 버퍼 관리자(210) 및 중앙 요청 리스트(208)는 BRL 커맨드를 원하는 버스로 전송한다. 예를 들어, 좌측 버스(102) 내의 제1 프로세서(112a)가 특정 캐시 라인 어드레스에 대하여 BRL 커맨드를 실행한다고 가정한다. 또한 캐시 라인 어드레스(400)에 대한 우측 버스 일관성 상태 비트(412)가 캐시 라인 어드레스(400)가 점유 상태임을 표시한다고 가정한다.

이 예에서, 좌측 규칙표(304a)는 캐시 일관성을 보장하기 위해 우측 버스(104) 상의 BRL 커맨드가 필요한지의 여부를 결정한다. 그후 좌측 규칙표(304a)는 좌측 FIFO(506a)로부터 미사용 상태의 어드레스 셀 식별자를 구한다. 좌측 규칙표(304a)는 어드레스 셀 식별자에 의해 식별된 어드레스 셀(506a)을 액세스하고, 캐시 라인 어드레스(400)를 메모리 어드레스(504)에 입력한다. 또한 좌측 규칙표(304a)는 해당 요청 셀(600)을 액세스하고, 타겟 버스 식별자(604) 내의 우측 버스 식별자, 버스 트랜잭션 코드(606) 내의 BRL 커맨드, 오너 버스 식별자(608) 내의 좌측 버스 식별자를 저장한다.

우측 버스 우선순위 인코더(602b)가 가장 높은 우선순위를 요청 셀(600)에 할당하면, 우측 버스 마스터(202b)는 우측 버스(104) 상에서 BRL 커맨드를 수행한다. 단계(848)에서, 우측 프로세서는 우측 버스(104) 상에서 BRL 커맨드에 응답한다. 단계(848)에서, 우측 프로세서는 자신의 캐시 메모리(114) 내의 캐시 라인 어드레스의 상태를 평가한다. 우측 프로세서가 더 이상 원하는 캐시 라인 어드레스의 복사본을 가지고 있지 않다면, 우측 프로세서는 HIT# 신호 및 HITM# 신호를 인가하지 않는다.

단계(850)에서, 우측 태그 제어장치(300b)는 우측 태그 메모리(212b) 내의 일관성 상태 비트(412)를 무효 상태로 설정한다. 로컬 버스 트랜잭션이 지연되면, 좌측 버스 슬레이브(202a)는 좌측 버스(102) 상에서 지연된 응답 트랜잭션을 발행한다. 단계(818)에서, 좌측 버스 슬레이브(202a)는 주 메모리(132)로부터 원하는 캐시 라인을 구하고, 이를 좌측 버스(102) 상의 요청 프로세서(112)로 전송한다. 그후 좌측 규칙표(304a)는 단계(822) 또는 단계(824)를 통해, 전술한 마지막 단계(810)에 도달한다.

단계(848)로 돌아가 설명하면, 우측 프로세서(112c 또는 112d)가 캐시 라인의 수정되지 않은 복사본을 포함하고 있으면, 이들 장치는 우측 HIT# 신호를 인가하고, 본 실시예는 단계(852)로 진행된다. 단계(852)에서, 우측 태그 제어장치(300b)는 우측 태그 메모리(212b) 내의 일관성 상태 비트(412)를 공유 상태로 설정한다.

단계(854)에서, 좌측 태그 메모리(212b)가 캐시 라인 어드레스(400)의 복사본을 가지고 있지 않으면, 좌측 규칙표(304)는 좌측 태그 제어장치(300A)에게 캐시 라인 어드레스(400)를 좌측 태그 메모리(212a)에 더하도록 지시한다. 전술한 바와 같이, 로컬 태그 메모리는 좌측 태그 메모리 내에 존재하는 캐시 라인 어드레스(400)를 배출시켜 새로운 캐시 라인 어드레스를 위한 공간을 형성할 필요가 있다. 만약 새로운 캐시 라인 어드레스를 위한 공간이 형성하였다면, 구 캐시 라인 어드레스를 먼저 무효화한 다음, 새로운 캐시 라인 어드레스를 좌측 태그 메모리에 더한다.

단계(852)에서, 좌측 버스 슬레이브는 주 메모리(132)로부터 원하는 캐시 라인을 구하고, 이를 좌측 버스(102) 상의 요청 프로세서(112a)로 전송한다. 단계(808)로 진행하여, 좌측 규칙표(304a)는 좌측 태그 제어장치(300a)에게 좌측 태그 메모리(212a) 내의 일관성 상태 비트(412)를 공유 상태로 설정하고, 마지막 단계인 단계(810)로 진행된다.

단계(848)로 돌아가 설명하면, 우측 프로세서(112c 또는 112d) 중의 하나의 프로세서가 자신의 캐시 메모리(114) 내의 캐시 라인의 수정된 복사본을 가지고 있으면, 우측 프로세서(112c 또는 112d)는 우측 HITM# 신호를 인가한다. 단계(858)에서, 수정된 캐시 라인은 좌측 버스(102)에 제공된다. 특히, 수정된 캐시 라인은 데이터 인터페이스 버퍼(134) 내의 해당 데이터 셀(700)에 로드되어, 그후 오너 버스 식별자(608) 내에서 식별된 좌측 버스(102)로 전송된다. 그후 데이터 셀(700)은 수정된 캐시 라인을 좌측 버스(102)로 전송하는 좌측 버스 슬레이브(204a)에 의해 액세스되며, 여기서 캐시 라인은 원 버스 트랜잭션이 지연가능한 경우 지연된 응답을 가진다. 또한 우측 태그 제어장치(300b)는 우측 태그 메모리 내의 일관성 상태 비트를 공유 상태로 설정한다.

단계(858)에서, 응답의 일부분으로서, 수정된 캐시 라인은 주 메모리(132)에 재기록된다. 그후 본 발명은 전술한 바와 같이 단계(852, 854), 단계(808), 및 마지막 단계인 단계(810)로 진행된다. 그리하여 본 발명은 캐시 일관성을 유지하기 위해 필요한 경우에는 BRL 크로스 버스 트랜잭션을 수행한다. 상기한 예에서는 좌측 버스(102)가 BRL 커맨드를 개시하는 경우의 절차에 대하여 기술하였지만, 이와 유사한 절차가 우측 버스(104) 또는 I/O 버스(106)가 BRL 커맨드를 개시하는 경우에 발생할 수 있다.

B. 버스 판독 무효화 라인 커맨드 절차

도 9는 버스(102, 104, 또는 106) 중의 하나의 버스가 버스 판독 무효화 라인(Bus Read Invalidate Line(BRIL) 커맨드)을 실행하는 경우에 캐시 일관성을 유지하기 위한 순서도를 도시한다. BRIL 커맨드를 실행하는 동안, 프로세서(112) 중의 하나가 원하는 캐시 라인 어드레스(400)를 할당된 일관성 필터(200)로 전송한다.

단계(902)에서, 사이클 인코더(302)는 버스(102, 104, 또는 106)가 BRIL 커맨드를 수행하는 지의 여부를 결정한다. 또한 할당된 태그 제어장치(300)는 BRIL 커맨드에서 식별된 캐시 라인 어드레스(400)를 사용하여, 리모트 태그 메모리(212) 내의 캐시 라인 어드레스(400)를 액세스한다. 리모트 태그 메모리(212)가 캐시 라인 어드레스(400)를 포함하고 있으면, 리모트 태그 메모리는 리모트 캐시 라인 어드레스(400)가 무효 상태임을 표시하는 리모트 일관성 상태 비트(412)를 공유 상태 또는 점유 상태로 되돌린다.

전술한 바와 같이, 단지 하나의 리모트 태그 메모리(212)만이 점유 상태의 특정 캐시 라인 어드레스를 포함할 수 있다. 그러나 하나 이상의 리모트 태그 메모리(212)는 공유 상태 또는 무효 상태의 특정 캐시 라인 어드레스를 포함할 수 있다. 하나의 리모트 태그 메모리(212)가 공유 상태의 캐시 라인 어드레스의 복사본을 포함하고, 다른 리모트 태그 메모리(300)가 무효 상태의 동일한 캐시 라인 어드레스의 복사본을 포함하는 경우, 공유 상태의 일관성 상태 비트(412)가 우선순위를 가진다. 아래에서 보다 상세하게 기술되어 있듯이, 하나 이상의 리모트 태그 메모리(300)가 공유 상태의 특정 캐시 라인 어드

레스의 복사본을 포함하면, 일관성 규칙(304)은 하나 이상의 리모트 버스(102, 104, 또는 106) 상에서 리모트 버스 트랜잭션을 수행하여 캐시 일관성을 보장할 수 있다. 또한 모든 리모트 태그 메모리(300)가 특정 캐시 라인 어드레스(400)를 포함하고 있지 않다면, 캐시 라인 어드레스(400)와 연관된 리모트 일관성 상태 비트(412)는 무효 상태로 처리된다.

예를 들어, 단계(902)에서 좌측 버스(102) 상의 프로세서(112a)가 특정 캐시 라인 어드레스(400)에 대하여 BRIL 커맨드를 발생하면, 좌측 사이클 인코더(302a)는 좌측 버스(102)가 BRIL 커맨드를 수행하는 지의 여부를 결정한다. 또한 좌측 태그 제어장치(300a)는 캐시 라인 어드레스(400)를 사용하여 우측 태그 메모리(212b) 및 I/O 태그 메모리(212c)에 액세스한다. 태그 제어장치(300)는 우측 태그 메모리(212b) 및 I/O 태그 메모리(212c)로부터 리모트 일관성 상태 비트(412)를 구하는데, 여기서 리모트 일관성 상태 비트(412)는 우측 태그 메모리(212b) 및 I/O 태그 메모리(212c) 내의 캐시 라인 어드레스가 무효 상태인지, 공유 상태인지, 또는 점유 상태인지를 표시한다. 우측 태그 메모리(212b) 또는 I/O 태그 메모리(212c) 중의 어느 하나도 캐시 라인 어드레스(400)를 포함하고 있지 않으면, 좌측 태그 제어장치(300a)는 리모트 일관성 상태 비트(412)를 무효 상태로 처리한다.

본 발명의 실시예에 의한 규칙표(304)의 동작과, 로컬 및 리모트 목적지를 사용하는 것에 대하여 초점을 맞춰 설명하면, 로컬 버스(102, 104, 또는 106)와 연관된 사이클 인코더(302)는 리모트 일관성 상태 비트(412)의 상태 및 버스 트랜잭션의 형태(BRIL 커맨드)를 로컬 규칙표(304)로 전송한다. 리모트 일관성 상태 비트(412)가 리모트 캐시 라인 어드레스(400)가 무효 상태임을 표시하면, 로컬 규칙표(304)는 단계(904)로 진행된다. 본 발명의 실시예에서, 리모트 일관성 상태 비트(412)가 무효 상태이면, 로컬 규칙표(304)는 어떠한 리모트 버스(102, 104, 또는 106) 상에서도 BRIL 크로스 버스 트랜잭션을 발생시키지 않는다. 오히려 로컬 규칙표(304)는 로컬 태그 메모리(300) 내의 캐시 라인 어드레스(400) 슈퍼세트를 유지한다.

단계(904)에서, 로컬 프로세서(112)는 임의의 로컬 프로세서(112)가 원하는 캐시 라인의 복사본을 포함하고 있는 지를 결정하기 위해, 그들의 내부 캐시 메모리(114)를 질의한다. 본 발명의 실시예에서, 버스 스누핑 동작이 프로세서의 캐시 메모리(114) 내에 수정된 캐시 라인이 존재한다는 것을 표시하면, 프로세서(112) 중의 하나가 또한 HITM# 신호를 인가한다. HITM# 신호는 어드레스 및 제어 라인(108) 중의 하나이며, 최신 캐시 라인을 가지는 프로세서(112)가 주 메모리(132)에 캐시 라인을 재기록할 필요가 있다는 것을 표시한다.

단계(906)에서, 로컬 프로세서(112)는 수정된 캐시 라인을 주 메모리(132)에 재기록함으로써, 버스 트랜잭션에 응답하고 이와 동시에 주 메모리(132)를 갱신한다. 단계(908)에서, 로컬 규칙표(304)는 로컬 태그 제어장치(300)에게 로컬 태그 메모리(212) 내의 일관성 상태 비트(412)를 점유 상태로 설정하도록 지시한다. 그후 로컬 규칙표(304)는 마지막 단계인 단계(910)로 진행된다.

단계(904)에서, 로컬 프로세서(112)가 캐시 라인의 수정된 복사본을 포함하고 있지 않으면, 로컬 규칙표(304)는 단계(912)로 진행된다. 전술한 바와 같이, 로컬 태그 메모리(212)가 캐시 라인 어드레스(400)의 복사본을 가지고 있지 않으면, 로컬 규칙표(304)는 캐시 라인 어드레스(400)를 로컬 태그 메모리(304)에 더한다. 그리하여 로컬 태그 메모리는 로컬 태그 메모리에 존재하는 캐시 라인 어드레스를 배출시켜 새로운 캐시 라인 어드레스를 위한 공간을 형성할 필요가 있다. 만약 새로운 캐시 라인 어드레스를 위한 공간을 형성하였다면, 구 캐시 라인 어드레스를 먼저 무효화한 다음, 새로운 캐시 라인 어드레스를 로컬 태그 메모리에 더한다. 단계(912)에서, 좌측 버스 슬레이브는 주 메모리(132)로부터 원하는 캐시 라인을 구하고, 이를 요청 프로세서(112)에 전송한다.

단계(902)로 돌아가, 리모트 태그 메모리(300)가 리모트 캐시 라인 어드레스(400)가 공유 또는 점유 상태임을 표시하는 경우에 발생하는 것에 초점을 맞춰 설명한다. 리모트 일관성 상태 비트(412)가 리모트 캐시 라인 어드레스(400)가 공유 또는 점유 상태임을 표시하면, 로컬 규칙표(304)는 캐시 일관성을 유지하는데 필요한 BRIL 크로스 버스 트랜잭션을 발생시킨다.

단계(920)에서, 로컬 규칙표(304)는 로컬 BRIL 커맨드가 지연될 수 있는지를 결정한다. 전술한 바와 같이, 버스 트랜잭션을 발행하는 프로세서(104)는 DEN# 신호를 인가함으로써 버스 트랜잭션이 지연가능한 지를 표시한다. DEN# 신호가 인가되면, 로컬 규칙표(304)는 단계(922)로 진행하고, 로컬 버스 슬레이브(204)에게 로컬 BRIL 커맨드를 지연하도록 지시한다. 그후 단계(924)에서 로컬 규칙표(304)는 리모트 버스(102, 104, 또는 106) 상에서 BRIL 커맨드를 수행하도록 지시한다.

단계(920)로 돌아가 설명하면, 버스 트랜잭션을 지연할 수 없는 경우에는, 로컬 규칙표(304)는 단계(926)로 진행되고, 로컬 버스 슬레이브(204)에게 로컬 BRIL 커맨드를 기능정지하도록 지시한다. 단계(924)에서, 로컬 규칙표(304)는 리모트 버스(102, 104, 또는 106) 상에서 리모트 BRIL 커맨드를 수행하도록 지시한다.

단계(924)에서, 단계(844)와 관련하여 전술한 바와 같이, 버퍼 관리자(210) 및 중앙 요청 리스트(208)는 BRIL 커맨드를 원하는 리모트 버스(102, 104, 또는 106)로 전송한다. 단계(928)에서, BRIL 커맨드가 리모트 버스(102, 104, 및 106) 상에서 실행된다. 단계(928)에서, 리모트 프로세서는 자신들의 캐시 메모리(114) 내의 캐시 라인 어드레스의 상태를 평가한다. 리모트 프로세서가 더 이상 원하는 캐시 라인 어드레스(400)의 복사본을 가지고 있지 않으면, 우측 프로세서는 HIT# 신호 또는 HITM# 신호를 인가하지 않는다.

단계(930)에서, 리모트 태그 제어장치(300)는 리모트 태그 메모리(212) 내의 일관성 상태 비트(412)를 무효 상태로 설정한다. 단계(932)에서, 좌측 버스 슬레이브는 주 메모리(132)로부터 원하는 캐시 라인을 구하고, 이를 좌측 버스(102) 상의 요청 프로세서(112)로 전송한다. 또한 캐시 라인 어드레스(400)가 로컬 태그 메모리(212) 내에 존재하지 않으면, 로컬 규칙표(304)는 로컬 태그 제어장치(300)에게 새로운 캐시 라인 어드레스(400)를 로컬 태그 메모리에 더하도록 지시한다. 전술한 바와 같이, 이것은 현존하는 캐시 라인 어드레스의 무효화를 요구할 수 있다. 단계(908)에서, 로컬 태그 제어장치(300)는 로컬 태그 메모리(212) 내의 일관성 상태 비트를 점유 상태로 하고, 그후에 마지막 단계인 단계(910)로 진행된다.

단계(928)로 돌아가 설명하면, 리모트 프로세서(112)가 캐시 라인의 수정되지 않은 복사본을 포함하는 경우, 이들 장치는 리모트 HIT# 신호를 인가하고, 본 발명의 실시예는 전술한 바와 같이, 단계(930), 단계(932), 단계(908)로 진행된 후에

마지막 단계인 단계(910)로 진행된다. 단계(928)로 다시 돌아가 설명하면, 리모트 프로세서(112) 중의 하나가 자신의 캐시 메모리(114) 내의 캐시 라인의 수정된 복사본을 가지고 있으면, 리모트 프로세서(112)는 리모트 HITM# 신호를 인가한다.

단계(934)에서, 수정된 캐시 라인은 로컬 버스(102, 104, 또는 106)로 제공된다. 특히 수정된 캐시 라인은 데이터 인터페이스 버퍼(134) 내의 적당한 데이터 셀(700)에 로드되어, 오너 버스 식별자(608) 내에서 식별된 버스로 재전송된다. 그후 로컬 버스 슬레이브(204)는 로컬 좌측 버스(102) 상에서 (원(original) 버스 트랜잭션이 지연가능하다면, 지연된 응답을 가지는) 수정된 캐시 라인을 전송한다. 또한 리모트 태그 제어장치(300)는 리모트 태그 메모리 내의 리모트 일관성 상태 비트를 무효 상태로 설정한다. 응답의 일부로서, 수정된 캐시 라인은 주 메모리(132)로 재기록된다.

전술한 바와 같이, 그후에 본 발명의 실시예는 단계(908)로 진행된 다음, 마지막 단계인 단계(910)로 진행된다. 그리하여 본 발명의 실시예는 캐시 일관성을 유지하기 위해 필요한 경우에는 BRIL 크로스 버스 트랜잭션을 수행한다.

Ⅵ. 다른 실시예

본 발명의 다른 실시예는 무효화 큐(invalidation queue)를 포함하여 구현된다. 도 10에 도시된 무효화 큐(1000)에 초점을 맞춰 설명하면, 각 버스는 할당된 무효화 큐(1000)를 가진다. 무효화 큐(1000)는 비교적 적은 개수의 큐 엔트리(1002)를 가진다. 각 큐 엔트리(1002)는 하나의 캐시 라인 어드레스(400)에 대한 태그 페이지 어드레스(402), 오프셋 어드레스(404), 일관성 상태 비트(412) 및 패리티 비트(414)를 저장한다.

본 발명의 실시예에서, 본 발명의 실시예에 의한 태그 메모리(212)의 크기는 주 메모리(132)에 비해 작다. 그리하여, 하나 이상의 캐시 라인 어드레스(400)가 동시에 동일한 태그 엔트리(410)로 매핑된다. 캐시 라인이 점유된(occupied) 태그 엔트리(410)로 매핑되면, 태그 제어장치(300)는 태그 메모리(212)로부터 점유된 태그 엔트리(410)를 배출시켜야 한다. 다시 말해, 태그 제어장치(300)가 새로운 캐시 라인 어드레스(400)를 태그 엔트리(410)에 저장하도록 하기 위해, 점유된 태그 엔트리(410) 내에 존재하는 캐시 라인을 무효화시켜야 한다.

또한 2개의 프로세서(112)가 동일한 태그 엔트리(410)와 매핑되는 2개의 캐시 라인을 액세스하려고 시도하면, 일관성 필터(200)는 제1 캐시 라인 어드레스(400)를 무효화하고, 이를 제2 캐시 라인 어드레스(400)로 대체한다. 그후 일관성 필터(200)는 제2 캐시 라인 어드레스(400)를 무효화한 다음, 이를 제1 캐시 라인 어드레스(400)로 대체한다. 이러한 핑퐁 효과(ping-pong effect)를 태그 메모리 스래싱(tag memory thrashing)이라 한다. 보다 큰 태그 메모리(212)를 사용하여 태그 메모리 스래싱을 최소화하는 것이 가능하지만, 이러한 접근방법은 태그 메모리(212)의 비용을 상당히 증가시킨다. 이에 비해 무효화 큐(1000)는 추가되는 메모리 비용을 최소화시키면서 태그 메모리 스래싱을 상당히 감소시킨다.

또한 구 캐시 라인 어드레스를 무효화해야 되는 경우, 구 캐시 라인의 무효화 동작이 완료될 때까지 새로운 캐시 라인 어드레스와 연관된 버스 트랜잭션을 지연시켜야 한다. 본 발명의 실시예에 의한 무효화 큐(1000)는 새로운 버스 트랜잭션이 처리되도록 한다. 버스가 미사용 상태가 되면, 이후에 구 캐시 라인 어드레스를 무효화시킬 수 있다.

일관성 필터(200) 중 하나가 태그 메모리(212)로부터 태그 엔트리(410)를 배출시킬 필요가 있는 경우, 태그 엔트리(410)는 일관성 필터의 무효화 큐(1002)에 저장된다. 본 발명의 실시예에 의한 무효화 큐(1000)는 최대 8개의 배출된 태그 엔트리(410)를 저장할 수 있지만, 무효화 큐(1002)는 이보다 더 많은 개수의 배출된 태그 엔트리(410)를 포함할 수 있다. 일관성 필터(200) 중의 하나의 필터가 8개 이상의 태그 엔트리(410)를 배출하면, 무효화 큐(1000)는 가장 최근에 액세스된 태그 엔트리(410)를 유지하고, 플러시(flush)된 태그 엔트리(410)를 무효화한다.

무효화 큐(1000)는 큐 엔트리(1002)를 태그 메모리(212) 내의 태그 엔트리(410)와는 다른 포맷으로 저장한다. 특히 본 발명의 실시예에 의한 큐 엔트리(1002)는 태그 메모리(212)로부터 배출된 태그 엔트리(410)의 태그 페이지 어드레스(402), 오프셋 어드레스(404), 일관성 상태 비트(412) 및 패리티 비트(414)를 포함한다. 임의의 캐시 라인 어드레스(400)가 임의의 큐 엔트리(1002) 내에 존재할 수 있기 때문에, 각 무효화 큐는 완전히 연관되어 있다고 표현된다.

각 무효화 큐(1000)가 비교적 적은 수의 무효화 큐 엔트리(1002)를 포함하고 있기 때문에, 각 무효화 큐(1000)는 태그 메모리(212)의 크기를 증가시키는 것에 대한 낮은 비용의 대안을 제공한다. 본 발명의 실시예에서, 무효화 큐(1000)는 시스템 액세스 제어장치(130) 내에 존재하는 SRAM을 사용하여 구현된다.

Ⅶ. 결론

본 발명의 실시예에 대하여 기술하였지만, 이들 실시예는 단지 예시를 목적으로 기술되었으며, 이들 실시예가 본 발명의 범위를 제한하는 것은 아니다. 예를 들어, 본 명세서에 기술된 본 발명의 실시예에서는 3중 버스(three-bus) 시스템(100)을 사용하고 있지만, 그 이상의 버스를 사용할 수도 있다. 또한 본 발명은 다양한 형태의 다중처리 시스템 상에서 구현될 수 있다. 따라서 본 발명의 원리 및 범위는 다음의 청구의 범위에 의해 정의된다.

산업상 이용 가능성

상기 내용 참고

(57) 청구의 범위

청구항 1.

시스템 메모리,

상기 시스템 메모리와 통신하는 제1 버스로서, 상기 시스템 메모리로부터 구해진 복수의 데이터 값을 저장하도록 구성되고, 상기 데이터 값에 대한 캐시 일관성 정보(cache coherency information)를 제1 일관성 상태 세트(a first set of coherency states)와 함께 유지하도록 구성된 적어도 하나의 캐시 메모리가 연결되어 있는 제1 버스,

상기 제1 버스와 통신하며, 상기 데이터 값과 연관된 일관성 상태 레코드(coherency status record)를 제2 일관성 상태 세트(a second set of coherency states)와 함께 유지하도록 구성된 일관성 메모리(coherency memory),

상기 시스템 메모리와 통신하며, 버스 트랜잭션을 전송할 수 있는 제2 버스, 및

상기 제2 버스 및 상기 일관성 상태 레코드와 통신하지만 상기 제1 버스에는 연결되어 있지 않으며, 상기 제2 버스 상의 상기 버스 트랜잭션을 모니터하고 상기 일관성 상태 레코드에 기초하여 상기 제2 버스로부터 상기 제1 버스로의 크로스 버스 트랜잭션을 금지(inhibit)하도록 구성된 일관성 필터

를 포함하는 다중버스, 다중처리 시스템.

청구항 2.

제1항에 있어서,

상기 제1 버스에 연결되어 있으며, 캐시 일관성 정보(cache coherency information)를 상기 제1 일관성 상태 세트와 함께 유지하는 복수의 제1 캐시 메모리를 추가로 포함하는 다중버스, 다중처리 시스템.

청구항 3.

제1항에 있어서,

상기 버스 트랜잭션은 상기 데이터 값 중 하나의 데이터 값과 연관된 메모리 어드레스를 식별하는, 다중버스, 다중처리 시스템.

청구항 4.

제3항에 있어서,

상기 일관성 상태 레코드는 복수의 엔트리를 포함하고, 상기 각 엔트리는 상기 캐시 메모리에 기억된 데이터 값과 연관된 메모리 어드레스를 저장하도록 구성된, 다중버스, 다중처리 시스템.

청구항 5.

제4항에 있어서,

상기 각 엔트리는 상기 제1 일관성 상태 세트보다 적은 수의 일관성 상태(coherency states)를 포함하는 제2 일관성 상태 세트를 저장하도록 구성된, 다중버스, 다중처리 시스템.

청구항 6.

제5항에 있어서,

상기 각 엔트리는 상기 제2 일관성 상태 세트 중의 적어도 하나를 기억하도록 구성된, 다중버스, 다중처리 시스템.

청구항 7.

제6항에 있어서,

상기 각각의 메모리 어드레스는 상기 엔트리 중 하나에 직접 매핑되는, 다중버스, 다중처리 시스템.

청구항 8.

다중버스 시스템 내에서 캐시 일관성(cache coherency)을 유지하는 방법에 있어서,

제1 버스와 연결된 캐시 유닛 내에 존재하는 복수의 제1 데이터 값과 연관된 캐시 상태 정보(cache status information)를 유지하는 단계,

상기 복수의 제1 데이터 값과 연관된 제1 버스 일관성 상태 레코드(a first bus-coherency status record)를 유지하는 단계,

제1 일관성 필터(a first coherency filter)로 상기 제1 버스상의 버스 트랜잭션을 모니터링하는 단계,

제2 일관성 필터(a second coherency filter)로 제2 버스상의 버스 트랜잭션을 모니터링하는 단계,

상기 제2 버스 상의 버스 트랜잭션을 전송하는 단계, 그리고

상기 제1 버스 일관성 상태 레코드가, 크로스 버스 트랜잭션을 금지하여도 메모리 일관성 오류(incoherency)가 발생하지 않을 것임을 나타내는 경우, 상기 제2 버스로부터 상기 제1 버스로의 크로스 버스 트랜잭션을 상기 제2 일관성 필터로 금지하는 단계

를 포함하는 캐시 일관성 유지 방법.

청구항 9.

제8항에 있어서,

상기 제1 버스 일관성 상태 레코드는 상기 각 데이터 값에 대한 적어도 하나의 엔트리를 유지하도록 구성되는 캐시 일관성 유지 방법.

청구항 10.

제9항에 있어서,

상기 금지하는 단계는 상기 엔트리가 무효 상태인지 여부를 결정하기 위해 상기 버스 트랜잭션과 연관된 상기 제1 버스 일관성 상태 레코드 내의 상기 엔트리를 참조(consulting)하는 단계를 추가로 포함하는 캐시 일관성 유지 방법.

청구항 11.

제9항에 있어서,

상기 금지하는 단계는 상기 엔트리가 공유 상태인지 여부를 결정하기 위해 상기 버스 트랜잭션과 연관된 상기 제1 버스 일관성 상태 레코드 내의 상기 엔트리를 참조하는 단계를 추가로 포함하는 캐시 일관성 유지 방법.

청구항 12.

제11항에 있어서,

상기 엔트리가 점유 상태이면 상기 버스 트랜잭션과 연관된 상기 엔트리를 수정하는 단계를 추가로 포함하는 캐시 일관성 유지 방법.

청구항 13.

제11항에 있어서,

상기 버스 트랜잭션과 연관된 상기 엔트리를 점유 상태에서 공유 상태로 수정하는 단계를 추가로 포함하는 캐시 일관성 유지 방법.

청구항 14.

제11항에 있어서,

상기 버스 트랜잭션과 연관된 상기 엔트리를 점유 상태에서 무효 상태로 수정하는 단계를 추가로 포함하는 캐시 일관성 유지 방법.

청구항 15.

시스템 메모리,

상기 시스템 메모리와 통신하며, 상기 시스템 메모리로부터 얻은 복수의 데이터 값을 저장하도록 구성된 적어도 하나의 캐시 메모리에 연결되고, 다중 버스 트랜잭션을 전송할 수 있는 시스템 버스,

상기 시스템 버스와 통신하며, 상기 데이터 값과 연관된 시스템 버스 일관성 상태 레코드(system bus coherency status record)를 유지하는 일관성 메모리(coherency memory),

상기 시스템 메모리와 통신하며, 다중 버스 트랜잭션을 전송 가능한 입/출력 버스, 그리고

상기 입/출력 버스 및 상기 일관성 메모리와 통신하며, 상기 입/출력 버스 상의 상기 버스 트랜잭션을 모니터링하지만 상기 시스템 버스상의 버스 트랜잭션은 모니터링하지 않으며, 상기 시스템 버스 일관성 상태 레코드에 기초하여 상기 입/출력 버스로부터 상기 시스템 버스로의 크로스 버스 트랜잭션을 금지하도록 구성된 입/출력 일관성 필터(I/O coherency filter)

를 포함하는 다중버스, 다중처리 시스템.

청구항 16.

제15항에 있어서,

상기 입/출력 일관성 필터는 상기 입/출력 버스 상의 직접 메모리 액세스(direct memory access; DMA)를 모니터링하고, 상기 캐시 메모리 내의 캐시 라인의 플러싱(flushing)을 방지하는 데 필요한 크로스 버스 트랜잭션을 수행하는, 다중버스, 다중처리 시스템.

청구항 17.

제15항에 있어서,

상기 입/출력 버스는 내부 I/O 캐시 메모리를 포함하는 복수의 I/O 브리지를 추가로 포함하며,

상기 I/O 캐시 메모리는 복수의 제2 데이터 값을 기억하도록 구성되며, 상기 복수의 제2 데이터 값에 대한 캐시 일관성 정보를 유지하도록 구성된, 다중버스, 다중처리 시스템.

청구항 18.

제17항에 있어서,

상기 입/출력 버스와 통신하는 I/O 일관성 메모리(I/O coherency memory)를 추가로 포함하며,

상기 I/O 일관성 메모리는 상기 복수의 제2 데이터 값과 연관된 I/O 일관성 상태 레코드(I/O coherency status record)를 유지하도록 구성된, 다중버스, 다중처리 시스템.

청구항 19.

제18항에 있어서,

상기 시스템 버스 및 상기 I/O 일관성 메모리와 통신하는 시스템 버스 일관성 필터(system bus coherency filter)를 추가로 포함하며,

상기 시스템 버스 일관성 필터는 상기 시스템 버스 상의 상기 버스 트랜잭션을 모니터링하고 상기 I/O 일관성 상태 레코드에 기초하여 상기 시스템 버스로부터 상기 입/출력 버스로의 크로스 버스 트랜잭션을 금지하도록 구성되는 다중버스, 다중처리 시스템.

청구항 20.

다중버스 시스템 내에서 캐시 일관성(cache coherency)을 유지하는 방법에 있어서,

시스템 버스와 연결된 캐시 유닛 내에 존재하는 복수의 제1 데이터 값과 연관되고, 제1 일관성 상태 세트(a first set of coherency states)에 기초하는 캐시 상태 정보를 유지하는 단계,

상기 복수의 제1 데이터 값과 연관된 일관성 상태 레코드(coherency status record)를 유지하는 단계,

제1 일관성 필터(a first coherency filter)로 상기 시스템 버스상의 버스 트랜잭션을 모니터링하는 단계,

제2 일관성 필터(a second coherency filter)로 입/출력 버스상의 버스 트랜잭션을 모니터링하는 단계,

상기 입/출력 버스 상의 버스 트랜잭션을 전송하는 단계, 그리고

상기 일관성 상태 레코드가, 크로스 버스 트랜잭션을 금지하여도 메모리 일관성 오류가 발생하지 않을 것임을 나타내는 경우, 상기 입/출력 버스로부터 상기 시스템 버스로의 크로스 버스 트랜잭션을 상기 제2 일관성 필터로 금지하는 단계

를 포함하는 캐시 일관성 유지 방법.

청구항 21.

제20항에 있어서,

상기 입/출력 버스 상의 버스 트랜잭션을 전송하는 단계는 직접 메모리 액세스인, 캐시 일관성 유지 방법.

청구항 22.

제20항에 있어서,

상기 입/출력 버스 상에서 전송된 복수의 I/O 데이터 값을 캐시에 저장하는 단계를 추가로 포함하는 캐시 일관성 유지 방법.

청구항 23.

제22항에 있어서,

상기 I/O 데이터 값과 연관된 캐시 상태 정보를 유지하는 단계를 추가로 포함하는 캐시 일관성 유지 방법.

청구항 24.

제23항에 있어서,

상기 I/O 데이터 값과 연관된 I/O 일관성 상태 레코드를 유지하는 단계를 추가로 포함하는 캐시 일관성 유지 방법.

청구항 25.

제24항에 있어서,

상기 시스템 버스 상의 버스 트랜잭션을 전송하는 단계, 그리고

상기 I/O 일관성 상태 레코드가, 상기 시스템 버스로부터 상기 I/O 버스로의 크로스 버스 트랜잭션을 금지하여도 메모리 일관성 오류가 발생하지 않을 것임을 나타내는 경우, 상기 시스템 버스로부터 상기 입/출력 버스로의 크로스 버스 트랜잭션을 상기 제2 일관성 필터로 금지하는 단계

를 추가로 포함하는 캐시 일관성 유지 방법.

청구항 26.

시스템 메모리,

상기 시스템 메모리와 통신하며, 상기 시스템 메모리로부터 구해진 복수의 데이터 값을 저장하도록 구성된 적어도 하나의 캐시 메모리에 연결되어 있는 제1 시스템 버스 및 제2 시스템 버스;

상기 제1 시스템 버스와 통신하며, 상기 제1 시스템 버스에 연결된 캐시 메모리 내의 데이터 값과 연관된 제1 일관성 상태 레코드(a first coherency status record)를 유지하도록 구성되는 제1 일관성 메모리(a first coherency memory),

상기 제2 시스템 버스와 통신하며, 상기 제2 시스템 버스에 연결된 캐시 메모리 내의 데이터 값과 연관된 제2 일관성 상태 레코드(a second coherency status record)를 유지하도록 구성되는 제2 일관성 메모리(a first coherency memory),

상기 제1 시스템 버스 및 상기 제2 일관성 메모리와 통신하며, 상기 제1 시스템 버스상의 버스 트랜잭션을 모니터링하고 상기 제2 일관성 상태 레코드에 기초하여 상기 제1 시스템 버스로부터 상기 제2 시스템 버스로의 크로스 버스 트랜잭션을 금지하도록 구성된 제1 일관성 필터(a first coherency filter), 그리고

상기 제2 시스템 버스 및 상기 제1 일관성 메모리와 통신하며, 상기 제2 시스템 버스상의 버스 트랜잭션을 모니터링하고 상기 제1 일관성 상태 레코드에 기초하여 상기 제2 시스템 버스로부터 상기 제1 시스템 버스로의 크로스 버스 트랜잭션을 금지하도록 구성된 제2 일관성 필터(a second coherency filter)

를 포함하는 다중버스, 다중처리 시스템.

청구항 27.

제26항에 있어서,

적어도 하나의 I/O 캐시 메모리 내에 I/O 데이터 값을 저장하는 복수의 I/O 장치 및 상기 시스템 메모리와 통신하는 I/O 버스, 그리고

상기 I/O 버스와 통신하며, 상기 I/O 데이터 값과 연관된 I/O 일관성 상태 레코드(I/O coherency status record)를 유지하도록 구성된 I/O 일관성 메모리(I/O coherency memory)

를 추가로 포함하는 다중버스, 다중처리 시스템.

청구항 28.

제27항에 있어서,

상기 제1 일관성 필터는 상기 I/O 일관성 메모리와 통신하며, 상기 제1 시스템 버스 상의 버스 트랜잭션을 모니터링하고 상기 I/O 일관성 상태 레코드에 기초하여 상기 제1 시스템 버스로부터 상기 I/O 버스로의 크로스 버스 트랜잭션을 금지하도록 구성된, 다중버스, 다중처리 시스템.

청구항 29.

제27항에 있어서,

상기 제2 일관성 필터는 상기 I/O 일관성 메모리와 통신하며, 상기 제2 시스템 버스 상의 버스 트랜잭션을 모니터링하고 상기 I/O 일관성 상태 레코드에 기초하여 상기 제2 시스템 버스로부터 상기 I/O 버스로의 크로스 버스 트랜잭션을 금지하도록 구성된, 다중버스, 다중처리 시스템.

청구항 30.

제27항에 있어서,

상기 제1 일관성 메모리 및 상기 제2 일관성 메모리와 통신하며, 상기 I/O 버스 상의 버스 트랜잭션을 모니터링하고, 상기 제1 일관성 상태 레코드에 기초하여 상기 I/O 버스로부터 상기 제1 시스템 버스로의 크로스 버스 트랜잭션을 금지하고, 상기 제2 일관성 상태 레코드에 기초하여 상기 I/O 버스로부터 상기 제2 시스템 버스로의 크로스 버스 트랜잭션을 금지하는 I/O 일관성 필터(I/O coherency filter)를 추가로 포함하는 다중버스, 다중처리 시스템.

청구항 31.

시스템 메모리,

상기 시스템 메모리와 통신하며, 상기 시스템 메모리로부터 얻은 복수의 데이터 값을 저장하도록 구성된 적어도 하나의 캐시 메모리에 연결되고, 상기 데이터 값에 대한 캐시 일관성 정보(cach coherency information)를 유지하도록 구성된 제1 버스 및 제2 버스,

상기 제1 버스와 연관된 데이터 값의 제1 일관성 상태 레코드(a first coherency status record)를 유지하는 제1 일관성 메모리 수단(a first coherency memory means),

상기 제2 버스와 연관된 데이터 값의 제2 일관성 상태 레코드(a second coherency status record)를 유지하는 제2 일관성 메모리 수단(a second coherency memory means),

상기 제1 버스 상의 버스 트랜잭션을 모니터링하고 상기 제2 일관성 상태 레코드에 기초하여 상기 제1 버스로부터 상기 제2 버스로의 크로스 버스 트랜잭션을 금지하는 제1 일관성 필터 수단(a first coherency filter means), 그리고

상기 제2 버스 상의 버스 트랜잭션을 모니터링하고 상기 제1 일관성 상태 레코드에 기초하여 상기 제1 버스로부터 상기 제2 버스로의 크로스 버스 트랜잭션을 금지하는 제2 일관성 필터 수단(a second coherency filter means)

을 포함하는 다중버스, 다중처리 시스템.

청구항 32.

시스템 메모리,

상기 시스템 메모리와 통신하며, 상기 시스템 메모리로부터 얻은 복수의 데이터 값을 저장하고 상기 데이터 값에 대한 일관성 상태 정보(coherency status information)를 유지하도록 구성된 적어도 하나의 캐시 메모리에 연결되어 있는 제1 버스,

상기 제1 버스와 통신하며, 상기 데이터 값과 연관된 일관성 상태 레코드(coherency status record)를, 상기 제1 일관성 상태 세트(set of coherency states)와는 다른 제2 일관성 상태 세트로 유지하는 일관성 메모리(cohrerency memory),

상기 시스템 메모리와 통신하며, 버스 트랜잭션을 전송할 수 있는 제2 버스, 그리고

상기 제2 버스 및 상기 일관성 메모리와 통신하며, 상기 제2 버스 상의 버스 트랜잭션을 모니터링하고 상기 일관성 상태 레코드에 기초하여 상기 제2 버스로부터 상기 제1 버스로의 크로스 버스 트랜잭션을 금지하기 위한 일관성 필터(coherency filter)

를 포함하는 다중버스, 다중처리 시스템.

청구항 33.

제32항에 있어서,

상기 일관성 필터는 제1 일관성 메모리 제어장치를 추가로 포함하며,

상기 제1 일관성 메모리 제어장치는 버스 트랜잭션 어드레스를 수신하고 상기 버스 트랜잭션 어드레스에 해당하는 상기 일관성 상태 레코드 내의 엔트리를 액세스하도록 구성된, 다중버스, 다중처리 시스템.

청구항 34.

제33항에 있어서,

상기 제1 일관성 메모리 제어장치는 상기 엔트리 내에 존재하는 상기 일관성 상태를 획득하는, 다중버스, 다중처리 시스템.

청구항 35.

제34항에 있어서,

상기 일관성 필터는 상기 버스 트랜잭션을 식별하는 버스 트랜잭션 코드를 식별하는 사이클 인코더를 추가로 포함하는, 다중버스, 다중처리 시스템.

청구항 36.

제35항에 있어서,

상기 일관성 필터는 상기 버스 트랜잭션을 식별하는 버스 트랜잭션 코드 및 상기 일관성 상태를 수신하는 규칙표를 추가로 포함하는, 다중버스, 다중처리 시스템.

청구항 37.

다중버스 시스템 내에서 캐시 일관성(cache coherency)을 유지하는 방법에 있어서,

제1 버스와 연결된 캐시 유닛 내에 존재하는 복수의 제1 데이터 값과 연관되고, 제1 일관성 상태 세트(a first set of coherency states)에 기초하는 캐시 상태 정보를 유지하는 단계,

상기 데이터 값과 연관되어 있으며, 상기 제1 일관성 상태 세트와는 다른 제2 일관성 상태 세트(a second set of coherency states)에 기초한 일관성 상태 레코드(coherency status record)를 유지하는 단계,

제1 일관성 필터(a first coherency filter)로 상기 제1 버스상의 버스 트랜잭션을 모니터링하는 단계,

제2 일관성 필터(a second coherency filter)로 제2 버스상의 버스 트랜잭션을 모니터링하는 단계,

상기 제2 버스 상의 버스 트랜잭션을 전송하는 단계, 그리고

상기 일관성 상태 레코드가, 크로스 버스 트랜잭션을 금지하여도 메모리 일관성 오류가 발생하지 않을 것임을 나타내는 경우, 상기 제2 버스로부터 상기 제1 버스로의 크로스 버스 트랜잭션을 금지하는 단계

를 포함하는 캐시 일관성 유지 방법.

청구항 38.

제37항에 있어서,

버스 트랜잭션 어드레스에 해당하는 상기 일관성 상태 레코드 내의 엔트리를 액세스하는 단계를 추가로 포함하는 캐시 일관성 유지 방법.

청구항 39.

제38항에 있어서,

상기 엔트리 내에 존재하는 상기 일관성 상태를 얻는 단계를 추가로 포함하는 캐시 일관성 유지 방법.

청구항 40.

제39항에 있어서,

상기 버스 트랜잭션을 식별하는 버스 트랜잭션 코드를 생성하기 위해, 상기 버스 트랜잭션 코드를 부호화하는 단계를 추가로 포함하는 캐시 일관성 유지 방법.

청구항 41.

제40항에 있어서,

캐시 일관성을 유지하기 위해 크로스 버스 트랜잭션이 필요한지의 여부를 결정하기 위해 상기 버스 트랜잭션 코드 및 상기 일관성 상태를 평가하는 단계를 추가로 포함하는 캐시 일관성 유지 방법.

청구항 42.

시스템 메모리,

상기 시스템 메모리와 통신하며, 스누핑 동작을 수행하고 상기 시스템 메모리로부터 구해진 복수의 데이터 값에 대한 캐시 상태 정보를 유지하고 내부적 수정을 외부로 출력하지 않고 상기 데이터 값 중의 일부와 연관된 상기 캐시 상태 정보를 내부적으로 수정하도록 구성된 복수의 캐시 메모리에 연결되어 있는 제1 버스 및 제2 버스,

상기 제1 버스와 통신하며, 상기 제1 버스 상의 캐시 메모리에 의해 구해진 상기 데이터 값과 연관된 일관성 상태 레코드(coherency status record)를 유지하도록 구성된 일관성 메모리(coherency memory), 그리고

상기 제2 버스 및 상기 일관성 메모리와 통신하며, 상기 제2 버스 상의 버스 트랜잭션을 모니터링하지만 상기 제1 버스상의 버스 트랜잭션은 모니터링하지 않으며, 상기 일관성 상태 레코드에 기초하여 상기 제2 버스로부터 상기 제1 버스로의 크로스 버스 트랜잭션을 금지하도록 구성된 일관성 필터(coherency filter)

를 포함하는 다중버스, 다중처리 시스템.

청구항 43.

다중버스 시스템 내에서 캐시 일관성(cache coherency)을 유지하는 방법에 있어서,

제1 시스템 버스와 연결된 캐시 유닛 내에 존재하는 복수의 제1 데이터 값과 연관된 캐시 상태 정보를 유지하는 단계,

상기 캐시 유닛 내의 상기 캐시 상태 정보를 내부적으로 수정하고, 상기 내부적 수정을 출력하지 않는 단계,

상기 데이터 값과 연관된 일관성 상태 레코드(coherency status record)를 유지하는 단계,

일관성 필터(coherency filter)로 제2 시스템 버스상의 버스 트랜잭션만 모니터링하고 상기 제1 시스템 버스상의 버스 트랜잭션은 모니터링하지 않는 단계,

상기 제2 시스템 버스 상의 버스 트랜잭션을 전송하는 단계, 그리고

상기 일관성 상태 레코드가, 크로스 버스 트랜잭션을 금지하여도 메모리 일관성 오류가 발생하지 않을 것임을 나타내는 경우, 상기 제2 시스템 버스로부터 상기 제1 시스템 버스로의 크로스 버스 트랜잭션을 금지하는 단계

를 포함하는 캐시 일관성 유지 방법.

청구항 44.
삭제

청구항 45.
삭제

청구항 46.
삭제

청구항 47.
삭제

청구항 48.
삭제

청구항 49.
삭제

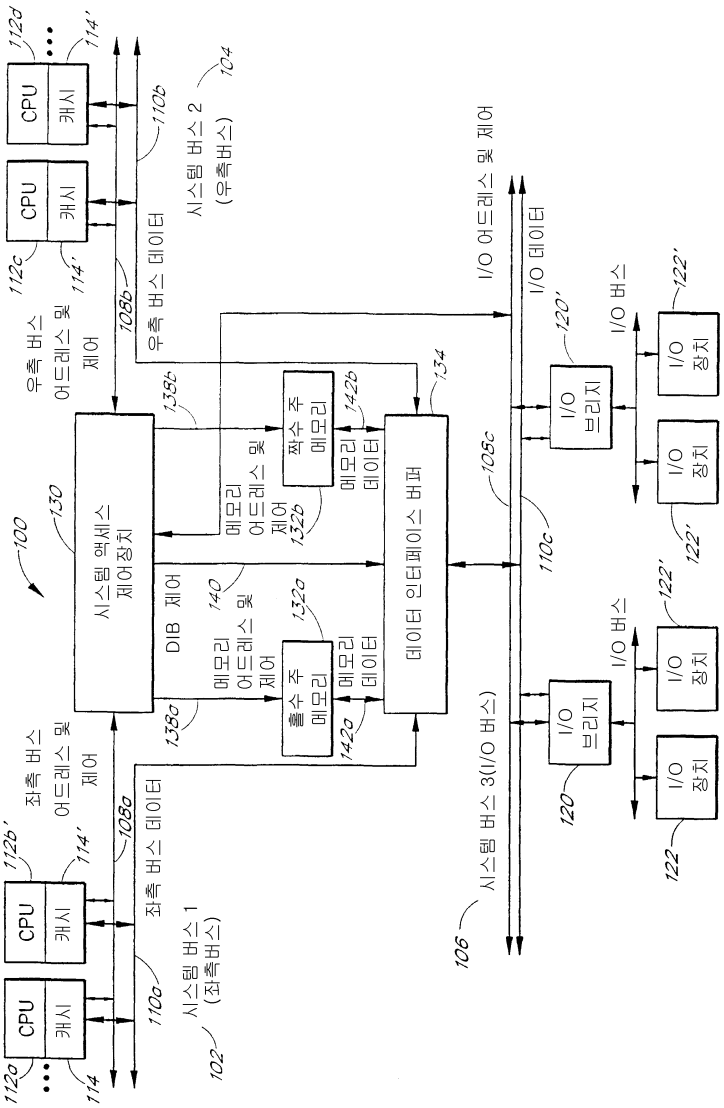
청구항 50.
삭제

청구항 51.
삭제

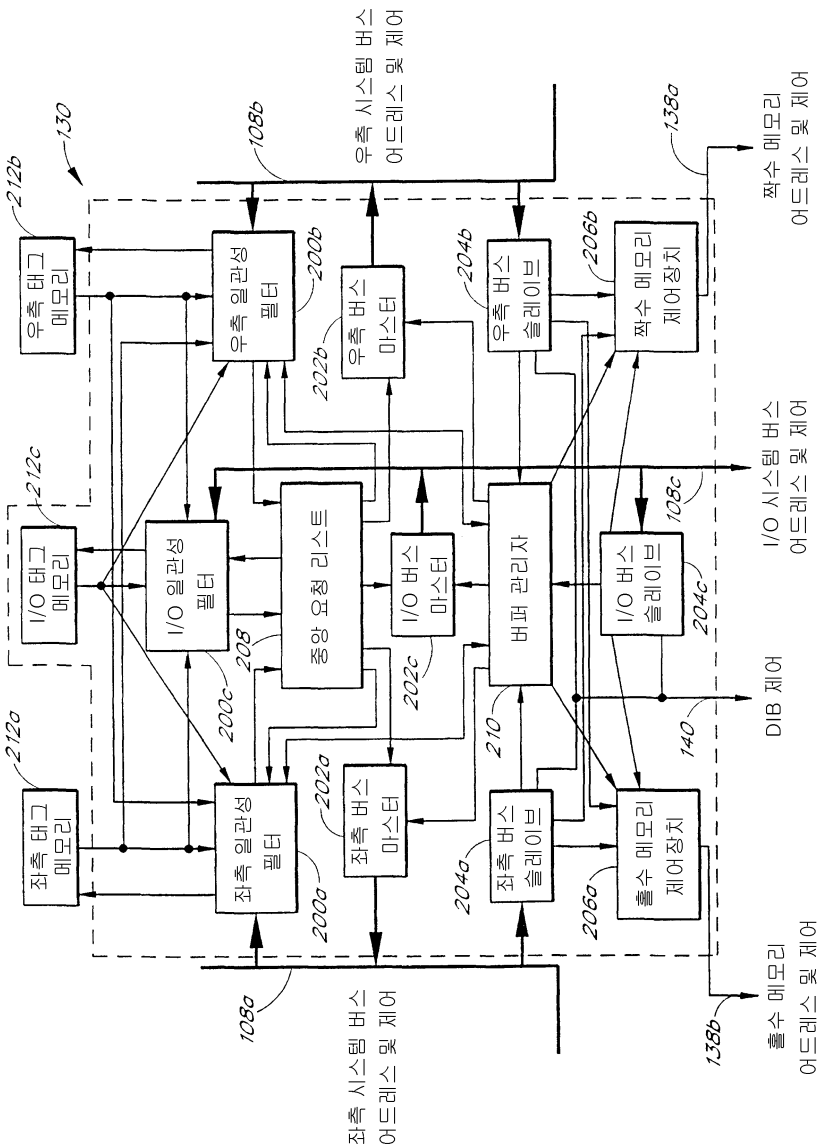
청구항 52.
삭제

도면

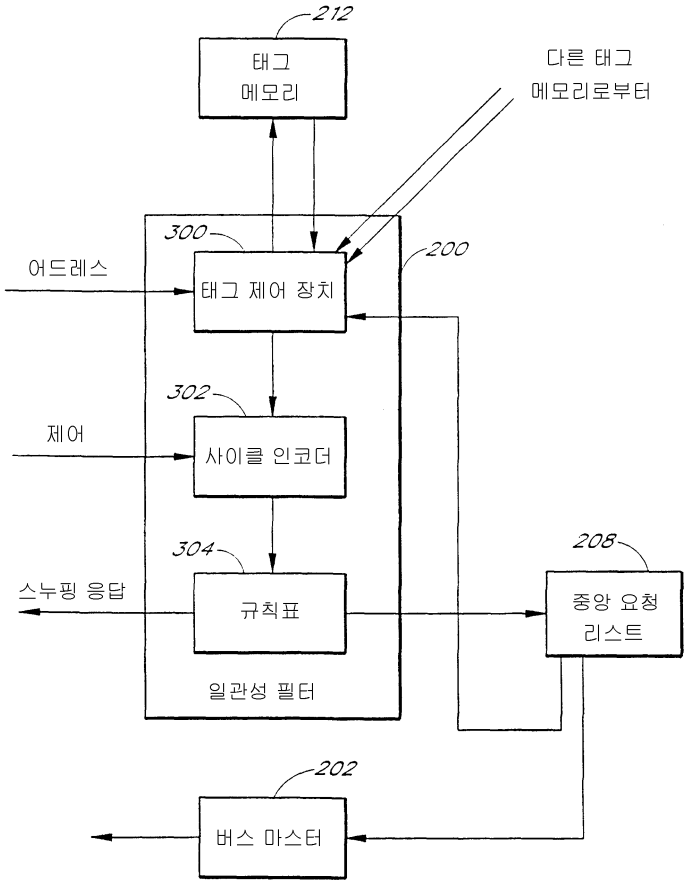
도면1


CPU
캐시
시스템 액세스 제어 장치
DIB 제어
메모리 어드레스 및 제어
짝수 주 메모리
홀수 주 메모리
메모리 데이터
데이터 인터페이스 버퍼
우측 버스 어드레스 및 제어
우측 버스 데이터
시스템 버스 2 (우측버스)
좌측 버스 어드레스 및 제어
좌측 버스 데이터
시스템 버스 1 (좌측버스)
I/O 어드레스 및 제어
I/O 데이터
시스템 버스 3(I/O 버스)
I/O 브리지
I/O 버스
I/O 장치
100
102
104
106
110a
110b
110c
112a
112b
112c
114
114'
120
120'
122
122'
130
134
140
1320
1322
1326
1330
1332
1334
1336
1420
1422
1426

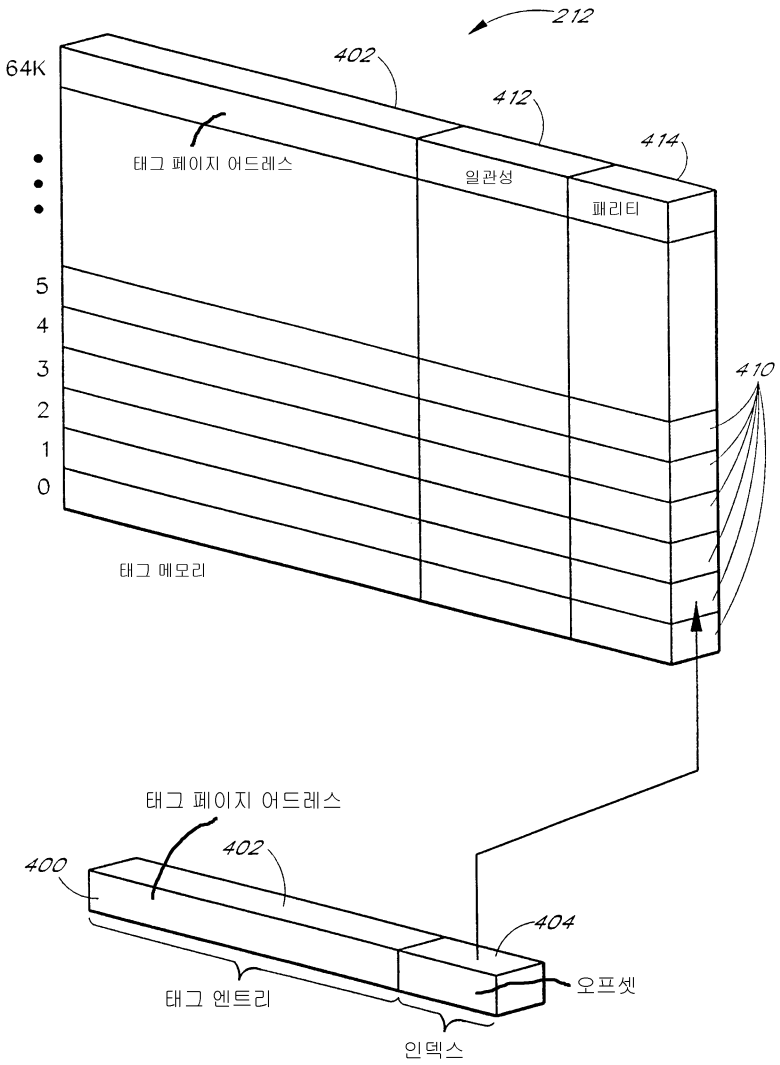
도면2


우측 시스템 버스
어드레스 및 제어
짝수 메모리
어드레스 및 제어
I/O 시스템 버스
어드레스 및 제어
DIB 제어
홀수 메모리
어드레스 및 제어
좌측 시스템 버스
어드레스 및 제어
130
212b
212c
212a
108b
108c
108a
138a
138b
140
210
208
200b
200c
200a
202b
202c
202a
204b
204c
204a
206b
206a
우측 태그 메모리
우측 일관성 필터
우측 버스 마스터
우측 버스 슬레이브
짝수 메모리 제어장치
I/O 태그 메모리
I/O 일관성 필터
중앙 요청 리스트
I/O 버스 마스터
버퍼 관리자
I/O 버스 슬레이브
좌측 태그 메모리
좌측 일관성 필터
좌측 버스 마스터
좌측 버스 슬레이브
홀수 메모리 제어장치

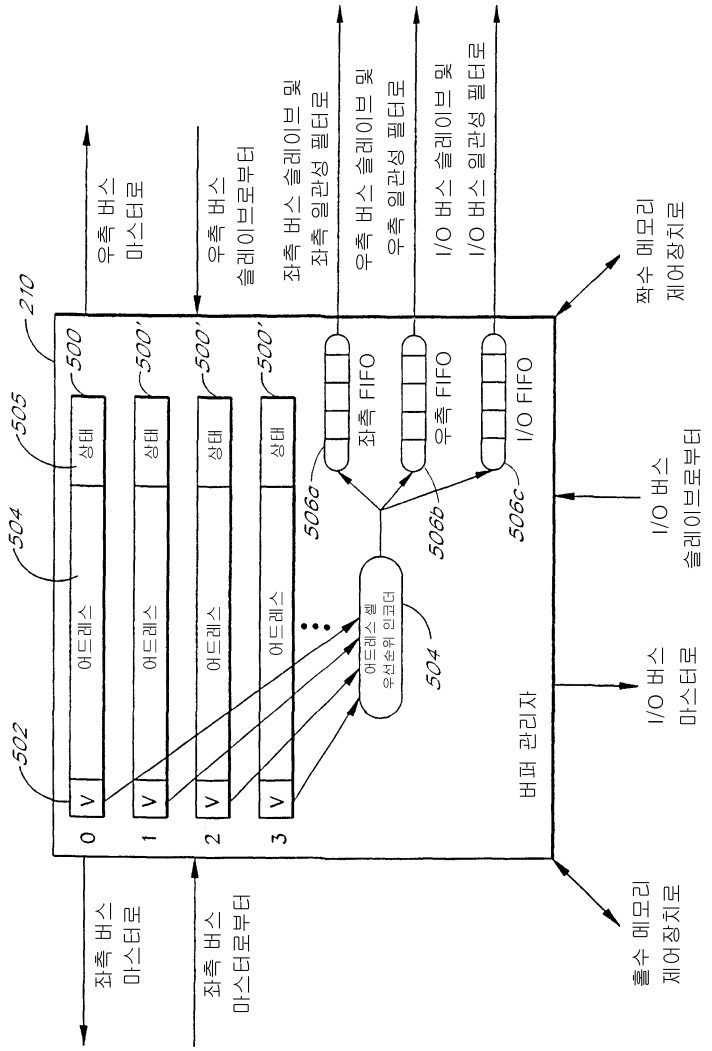
도면3


212
태그
메모리
다른 태그
메모리로부터
300
200
어드레스
태그 제어 장치
302
제어
사이클 인코더
304
208
스누핑 응답
규칙표
중앙 요청
리스트
일관성 필터
202
버스 마스터

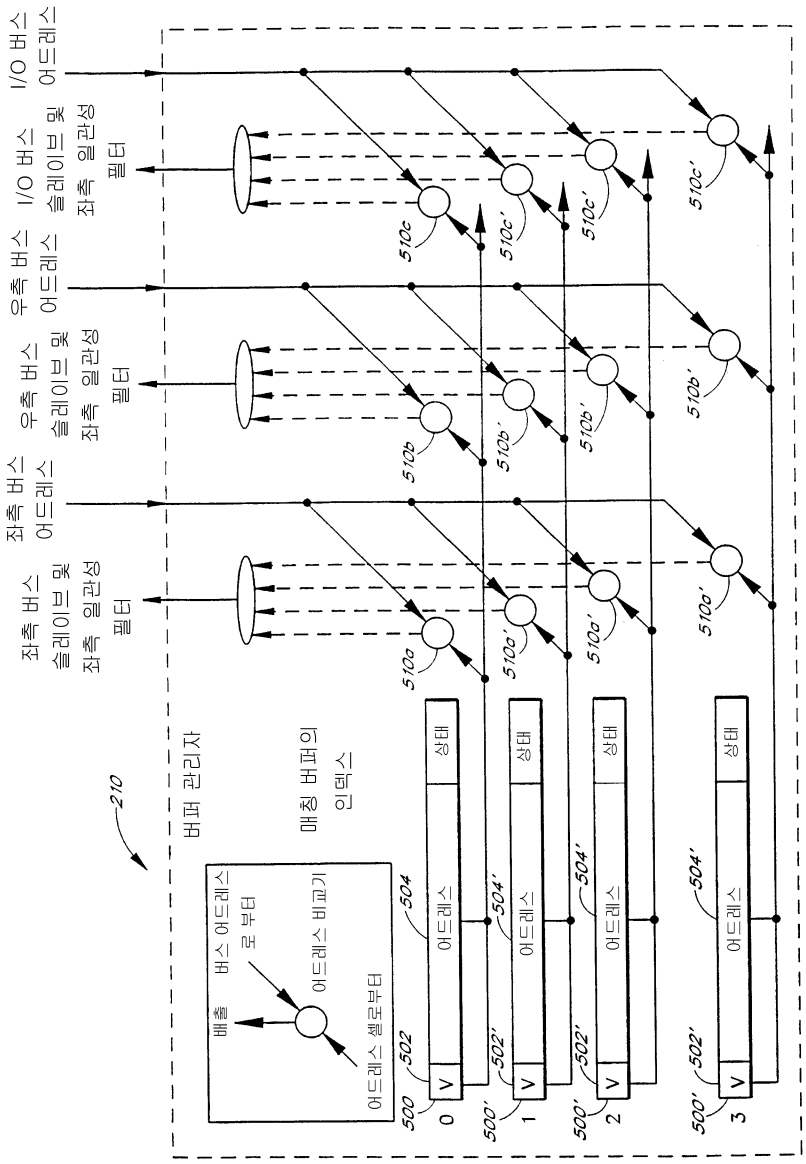
도면4


212
64K
402
412
414
태그 페이지 어드레스
일관성
패리티
5
4
3
2
1
0
410
태그 메모리
태그 페이지 어드레스
400
402
404
오프셋
태그 엔트리
인덱스

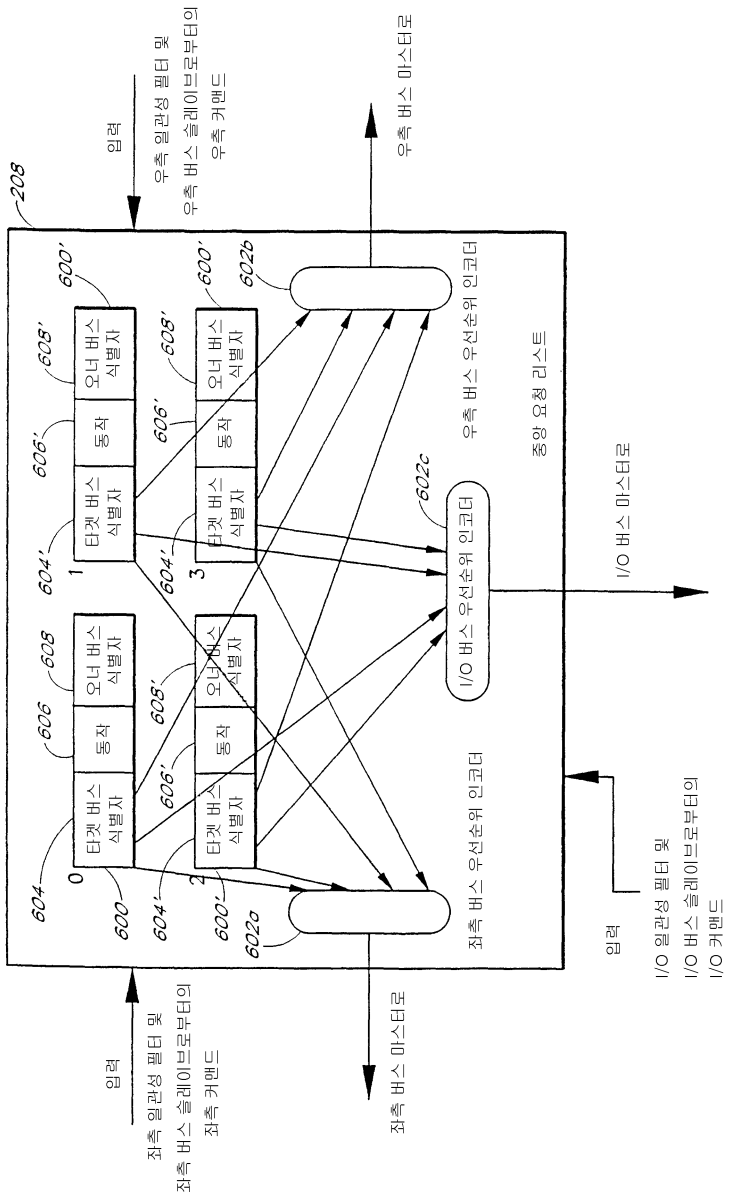
도면5a


210
500
500'
505
504
502
상태
어드레스
V
0
1
2
3
우측 버스
마스터로
우측 버스
슬레이브로부터
좌측 버스 슬레이브 및
좌측 일관성 필터로
우측 버스 슬레이브 및
우측 일관성 필터로
I/O 버스 슬레이브 및
I/O 버스 일관성 필터로
506a
506b
506c
좌측 FIFO
우측 FIFO
I/O FIFO
어드레스 셀
우선순위 인코더
504
버퍼 관리자
짝수 메모리
제어장치로
I/O 버스
슬레이브로부터
I/O 버스
마스터로
홀수 메모리
제어장치로
좌측 버스
마스터로
좌측 버스
마스터로부터

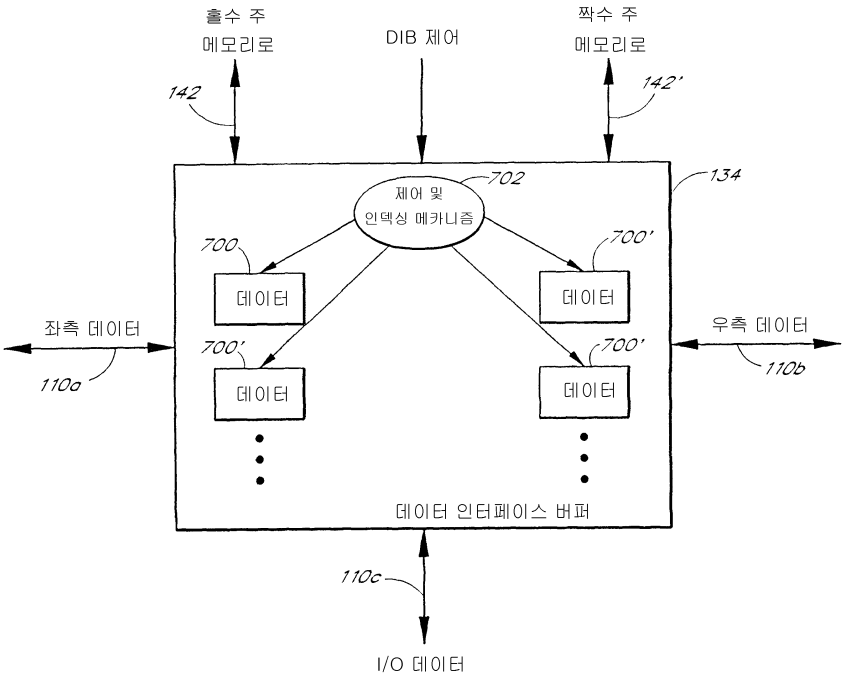
도면5b


I/O 버스 어드레스
I/O 버스 슬레이브 및 좌측 일관성 필터
우측 버스 어드레스
우측 버스 슬레이브 및 좌측 일관성 필터
좌측 버스 어드레스
좌측 버스 슬레이브 및 좌측 일관성 필터
210
버퍼 관리자
매칭 버퍼의 인덱스
배출
버스 어드레스 로 부터
어드레스 비교기
어드레스 셀로부터
상태
어드레스
V
0
1
2
3

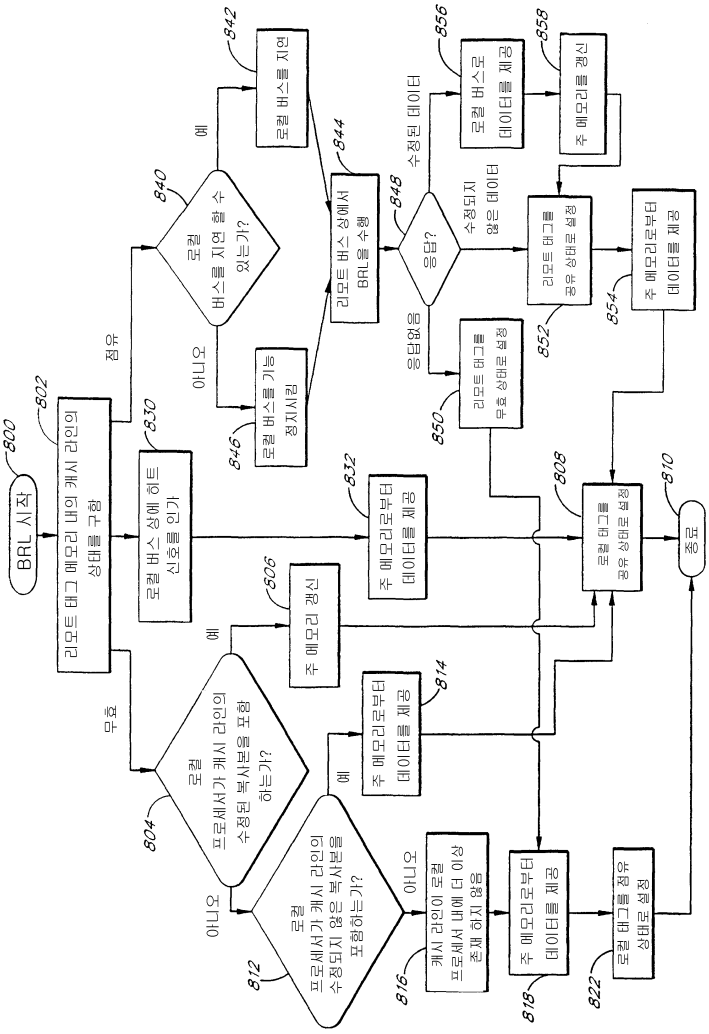
도면6


208
입력
우측 일관성 필터 및
우측 버스 슬레이브로부터의
우측 커맨드
우측 버스 마스터로
600'
608'
606'
604'
타겟 버스 식별자
동작
오너 버스 식별자
1
3
602b
602c
I/O 버스 우선순위 인코더
우측 버스 우선순위 인코더
중앙 요청 리스트
I/O 버스 마스터로
0
2
602a
좌측 버스 우선순위 인코더
입력
I/O 일관성 필터 및
I/O 버스 슬레이브로부터의
I/O 커맨드
입력
좌측 일관성 필터 및
좌측 버스 슬레이브로부터의
좌측 커맨드
좌측 버스 마스터로

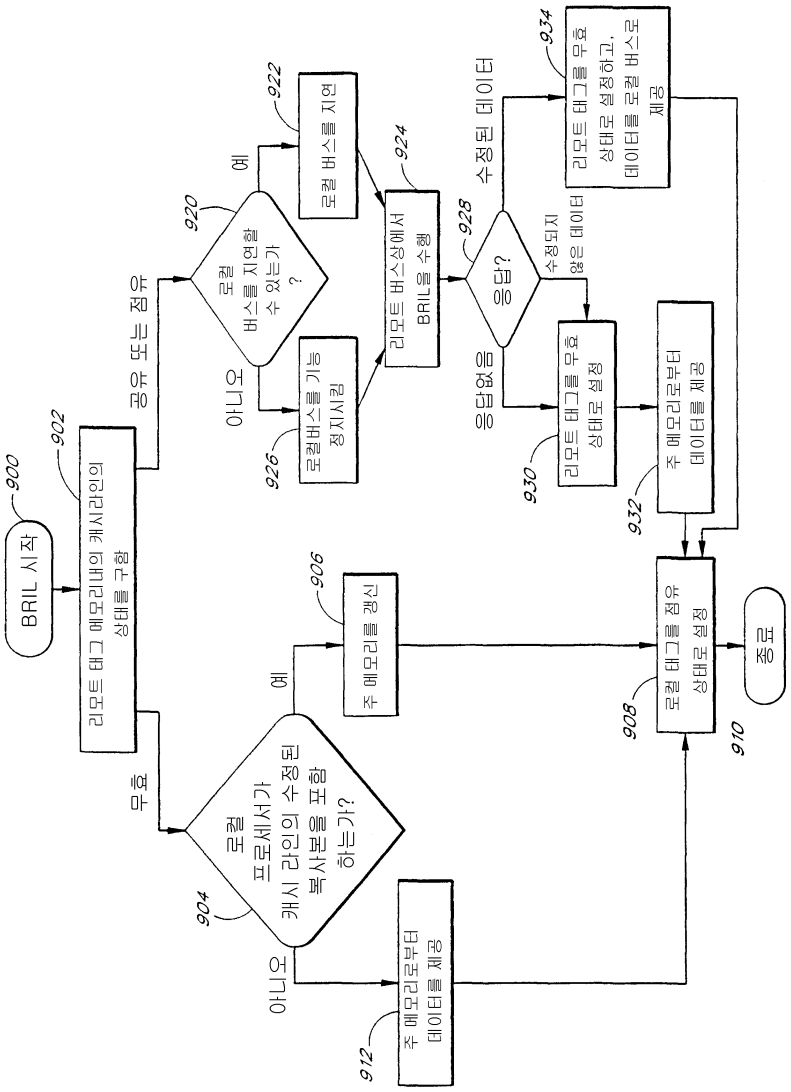
도면7


홀수 주
메모리로
DIB 제어
짝수 주
메모리로
142
142'
134
제어 및
인덱싱 메카니즘
702
700
700'
데이터
데이터
좌측 데이터
우측 데이터
700'
700'
데이터
데이터
110a
110b
데이터 인터페이스 버퍼
110c
I/O 데이터

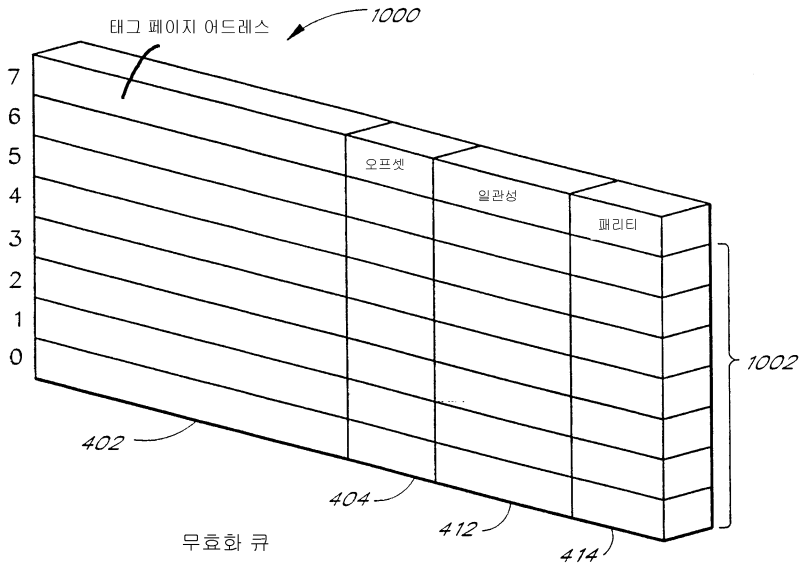
도면8


800
BRL 시작
802
리모드 태그 메모리 내의 캐시 라인의 상태를 구함
점유
무효
830
로컬 버스 상에 히트 신호를 인가
832
주 메모리로부터 데이터를 제공
808
로컬 태그를 공유 상태로 설정
810
종료
806
주 메모리 갱신
804
로컬 프로세서가 캐시 라인의 수정된 복사본을 포함하는가?
예
814
주 메모리로부터 데이터를 제공
아니오
812
로컬 프로세서가 캐시 라인의 수정되지 않은 복사본을 포함하는가?
예
816
캐시 라인이 로컬 프로세서 내에 더 이상 존재 하지 않음
아니오
818
주 메모리로부터 데이터를 제공
822
로컬 태그를 점유 상태로 설정
840
로컬 버스를 지연 할 수 있는가?
예
842
로컬 버스를 지연
아니오
846
로컬 버스를 기능 정지시킴
844
리모드 버스 상에서 BRL을 수행
848
응답?
수정된 데이터
856
로컬 버스로 데이터를 제공
858
주 메모리를 갱신
수정되지 않은 데이터
852
리모드 태그를 공유 상태로 설정
854
주 메모리로부터 데이터를 제공
응답없음
850
리모드 태그를 무효 상태로 설정

도면9


BRIL 시작
900
리모트 태그 메모리내의 캐시라인의 상태를 구함
902
공유 또는 점유
무효
로컬 프로세서가 캐시 라인의 수정된 복사본을 포함하는가?
904
예
주 메모리를 갱신
906
아니오
주 메모리로부터 데이터를 제공
912
로컬 태그를 점유 상태로 설정
908
종료
910
로컬 버스를 지연할 수 있는가?
920
예
로컬 버스를 지연
922
아니오
로컬버스를 기능 정지시킴
926
리모트 버스상에서 BRIL을 수행
924
응답?
928
수정된 데이터
리모트 태그를 무효 상태로 설정하고, 데이터를 로컬 버스로 제공
934
응답없음
수정되지 않은 데이터
리모트 태그를 무효 상태로 설정
930
주 메모리로부터 데이터를 제공
932

도면10


태그 페이지 어드레스
1000
7
6
5
4
3
2
1
0
오프셋
일관성
패리티
1002
402
404
412
414
무효화 큐