(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) Int. Cl.[6]
G06F 12/08
G06F 12/00

(45) 공고일자 2000년12월01일
(11) 등록번호 10-0273039
(24) 등록일자 2000년09월01일

(21) 출원번호 10-1998-0000913
(22) 출원일자 1998년01월14일
(65) 공개번호 특1998-0079610
(43) 공개일자 1998년11월25일

(30) 우선권주장 8/839,543 1997년04월14일 미국(US)

(73) 특허권자 인터내셔널 비지네스 머신즈 코포레이션 포만 제프리 엘
미국 10504 뉴욕주 아몬크
(72) 발명자 아리밀리 라비 쿠마르
미국 78759 텍사스주 오스틴 스파이스브러쉬 드라이브9221
도드슨 존 스티븐
미국 78660 텍사스주 플루거빌 벨 록 써클 1205
(74) 대리인 김성기, 송병옥

심사관 : 김세영

(54) 멀티프로세서 데이터 처리 시스템의 캐쉬 일관성을 유지하기위한 캐쉬 일관성 프로토콜 제공 방법 및시스템

요약

본 발명은 멀티프로세서 데이터 처리 시스템의 캐쉬 일관성(cache coherency)을 유지하기 위한 캐쉬 일관성 프로토콜을 제공하는 방법 및 시스템을 개시한다. 본 발명의 방법과 시스템에 따르면, 각 프로세서는 적어도 제 1 레벨 캐쉬와 제 2 레벨 캐쉬의 캐쉬 계층 구조를 가지며, 제 1 레벨 캐쉬는 제 2 레벨 캐쉬의 상위 계층이다. 각각의 캐쉬들은 다중 캐쉬 라인을 포함하며, 각각의 캐쉬 라인들은 갱신 상태(modified state), 배타상태(exclusive state), 공유 상태(shared state), 무효 상태(invalid state), 최근 판독 상태(recently-read state) 및 상위 계층 미정 상태(upstream undefined state)를 포함하는 적어도 6개의 상이한 상태로된 캐쉬 라인을 식별하는데 사용되는 상태 비트 필드(state-bit field)와 연관되어 있다. 가장 최근에 액세스(access)된 정보의 사본(copy)을 포함하는 캐쉬 라인의 표시에 응답하여, 캐쉬 라인의 상태는 무효 상태로부터 최근 판독 상태로 변경된다. 라인 필(linefill) 동작을 수행함이 없이 제 1 레벨 캐쉬 내의 캐쉬 라인의 정보 갱신에 응답하여, 캐쉬 라인의 단계는 무효 상태로부터 상위 계층 미정 상태로 천이된다.

대표도

도1

명세서

도면의 간단한 설명

도 1은 본 발명이 적용될 수 있는 멀티프로세서 데이터 처리 시스템의 블록 다이어그램.

도 2는 종래 기술 멀티프로세서 데이터 처리 시스템의 캐쉬 일관성 프로토콜을 도시한 상태 다이어그램.

도 3은 본 발명의 바람직한 실시예에 따른 멀티프로세서 데이터 처리 시스템의 캐쉬 일관성 프로토콜을 도시한 상태 다이어그램.

* 도면의 주요부분에 대한 부호의 설명

10 : 멀티프로세서 데이터 처리 시스템 11a, 11B : 처리 유닛
12 : 프로세서 코어 13 : 인스트럭션 캐쉬
14 : 데이터 캐쉬 15 : 제 2 캐쉬
16 : 버스 17 : 펌 웨어
18 : I/O 장치 19 : 시스템 메모리

발명의 상세한 설명

발명의 목적

발명이 속하는 기술분야 및 그 분야의 종래기술

본 발명은 캐쉬 일관성을 유지하기 위한 방법과 시스템에 관한 것으로, 구체적으로는 데이터 처리 시스템 내의 캐쉬 일관성을 유지하기 위한 방법 및 시스템에 관한 것이다. 더욱 구체적으로 본 발명은 멀티프로세서 데이터 처리 시스템 내의 캐쉬 일관성을 유지하기 위한 캐쉬 일관성 프로토콜을 제공하는 방법 및 시스템에 관한 것이다.

대칭형 멀티프로세서(symmetric multiprocessor: SMP) 데이터 처리 시스템에서, 모든 처리 유닛들(processing units)은 일반적으로 동일하다. 즉, 모든 처리 유닛들은 작동을 위해서 인스트럭션(instruction) 및 프로토콜의 공통 세트(common set) 또는 서브세트(subset)를 사용하며 일반적으로 동일한 구조를 갖는다. 각 처리 유닛은 프로그램 인스트럭션들을 실행하기 위한 다중 레지스터들과 실행 유닛들을 갖는 프로세서 코어를 포함한다. 또한, 각 처리 유닛은 고속 메모리들을 사용하여 구현되는 인스트럭션 캐쉬 및/또는 데이터 캐쉬와 같은 하나 이상의 주 캐쉬들(primary cashes) (즉, 1 레벨 캐쉬 또는 L1 캐쉬)을 가질 수 있다. 또한, 각 처리 유닛들은 상술한 것과 같은 주 캐쉬를 지원하기 위해 통상적으로 부 캐쉬(secondary cache) 또는 레벨 2(L2) 캐쉬로 불리는 캐쉬들을 추가적으로 포함할 수 있다.

SMP 환경에서는 각 개별 메모리 위치에 대한 판독 또는 기록 동작이 모든 프로세서에 대해 소정의 순서(order)에 따라 직렬화(serialize)되도록 메모리 일관성 방법을 제공하는 것이 중요하다. 바꾸어 말하면, 모든 처리 유닛들은 소정의 순서에 따라 행해지는 소정의 메모리 위치에 대한 판독 또는 기록 동작을 준수한다.

캐쉬들에 관해서는, 캐쉬 일관성을 달성하기 위한 많은 프로토콜들 및 기법들이 당업자에게 공지되어 있다. 공통적으로, 이러한 모든 프로토콜들은 임의의 소정 시점에서 캐쉬 라인에 기록하기 위한 " 허가(permission)" 를 단지 하나의 처리 유닛에만 부여한다. 이러한 요구조건으로 인해, 처리 유닛이 캐쉬 라인에 기록을 시도할 때마다, 다른 모든 처리 유닛들에게 자신이 캐쉬 라인에 기록하기를 원한다는 것을 통지하고, 그 기록 동작을 실행하기 전에 다른 모든 처리 유닛으로부터의 허락을 수신하여야 한다.

여러 처리 유닛들 간에 통신하기 위해, 요구(requesting) 처리 유닛은 자신이 캐쉬 라인을 판독 또는 기록하기를 원한다는 것을 나타내는 메세지를 (버스와 같은)상호 연결부를 통해 전달하여야 한다. 요구가 상호 연결부 상에 출력되면, 다른 모든 처리 유닛들은 이 동작을 " 스누핑(snooping)" (또는 모니터(monitor)링)하여 자신들의 캐쉬 상태가 요구된 동작의 진행을 허가할 수 있는지와 어떤 조건하에서 허가할 수 있는지를 결정한다. 버스 트랜잭션(transaction)을 허가하고 캐쉬 일관성을 유지하기 위해 스누핑과 추종(follow-up) 동작을 요구하는 여러 가지의 버스 트랜잭션이 존재한다. 스누핑 동작은 특정 버스 신호의 확인에 의해 발생되는 자격을 갖춘(qualified) 스누핑 요구를 수신함으로써 트리거된다. 인스트럭션 처리는 스누프 적중(hit)이 발생할 때만 인터럽트되고 스누프 상태 머신은 손상된 캐쉬 라인의 일관성을 해결하기 위해 추가적인 캐쉬 스누프가 필요한지의 여부를 결정한다.

이런 종류의 통신은 가장 최근의 유효한 정보의 사본이 시스템 메모리로부터 시스템 내의 하나 이상의 캐쉬로 옮겨졌을 수 있기 때문에 필요하다. 실제로, 정보의 정확한 버전(version)이 시스템 메모리 또는 시스템내의 캐쉬들 중의 어느 하나에 존재하거나 또는 둘 다에 존재할 수도 있다. 따라서 정확한 버전이 시스템 내의 하나 이상의 다른 캐쉬들에 존재한다면 시스템 메모리가 아니라 캐쉬(들)로부터 정확한 값을 얻는 것이 중요하다.

시스템 내의 캐쉬 일관성을 달성하기 위하여 상태 비트 필드가 캐쉬 라인의 현재 " 상태(state)" 를 나타내는데 사용된다. 그 다음, 이 상태 정보는 일반화된 상호 연결과 캐쉬 상호 간 연결(inter-cache connections) 상의 메시지 트래픽(traffic)을 줄이기 위하여 캐쉬 일관성 프로토콜에서 소정 최적화를 허용하는데 사용된다. 이러한 메카니즘의 하나의 예로, 처리 유닛이 판독을 실행할 때 처리 유닛은 판독이 나중에 재시도되어야 하는지의 여부를 나타내는 메세지를 수신한다. 만약 판독 동작이 재시도되지 않는다면, 메세지는 통상적으로 처리 유닛으로 하여금 임의의 다른 처리 유닛이 또한 유효하고 활성화된 정보의 사본을 가지고 있는지를 결정하도록 하는 정보를 포함한다(이것은 다른 최저 레벨 캐쉬(lowest-level cache)가 재 시도하지 않은 임의의 판독에 대해 " 공유(shared)" 또는 " 비공유(not shared)" 표시를 가지도록 해줌으로써 달성된다). 따라서, 처리 유닛은 시스템 내의 임의의 다른 처리 유닛이 정보의 사본을 가지고 있는지의 여부를 결정할 수 있다. 다른 처리 유닛이 모두 활성화된 정보의 사본을 가지고 있지 않다면 판독 처리 유닛은 캐쉬 라인의 상태 비트 필드를 배타(Exclusive)로 표시한다. 캐쉬 라인이 배타(Exclusive)로 표시되어 있으면, 처리 유닛이 시스템 내의 다른 처리 유닛들과 먼저 통신할 필요 없이 나중에 캐쉬 라인에 기록하는 것을 허용할 수 있는데, 그 이유는 다른 처리 유닛이 동일한 정보의 사본을 가지고 있지 않기 때문이다. 따라서, 처리 유닛이 상기 상호 연결을 통해 이 의도(intention)를 먼저 통신하지 않고도 캐쉬 라인을 판독 또는 기록하는 것이 가능하지만, 이는 다른 처리 유닛이 모두 동일한 정보에 관심을 가지고 있지 않다는 것을 일관성 프로토콜이 보장했을 때에만 가능하다.

캐쉬 블록들을 액세스하는데 있어서 " 중재(intervention)" 라고 알려진 절차를 이용하면 추가적인 개선점이 달성될 수 있다. 중재 절차는 캐쉬로 하여금 메모리 블록에 대한 제어를 가지도록 하여 판독 형태의 동작에 대한 값을 요구하는 다른 캐쉬로 상기 메모리 블록 내의 데이터 또는 인스트럭션을 직접 제공한다. 바꾸어 말하면, 중재 절차는 우선 시스템 메모리에 데이터 또는 인스트럭션을 기록하기 위한 요구를 바이패스(bypass)하고, 그 다음 요구 처리 유닛이 시스템 메모리로부터 다시 그 데이터 및 인스트럭션을 판독하도록 한다. 중재는 캐쉬 라인이 갱신 또는 배타 상태인 값을 갖는 캐쉬에 의해서만 수행될 수 있다. 이러한 갱신 및 배타 상태의 모두에 있어서, 상기 값의 유효한 사본을 갖는 캐쉬 라인이 단 하나만 존재하기 때문에, 유효한 값을 시스템 메모리에 먼저 기록할 필요없이 버스를 통해 그 유효한 값을 제공하는(source) 것은 간단한 문제이다. 따라서, 중재 절차는 시스템 메모리에 기록하고 또한 시스템 메모리로부터 판독하는 (통상 3개의 버스 동작과 2개의 메모리 동작을 필요로 하는)긴 프로세스를 피함으로써 처리 속도를 높인다. 그러므로, 중재 절차는 대기 시간면에서 향상을 가져올 뿐만 아니라 버스 대역폭을 증가시킨다.

일반적으로 어느 캐쉬가 정보를 제공(source)할 것인지를 결정하는 것이 어렵기 때문에, 종래의 캐쉬 일

관성 프로토콜은 데이터 또는 인스트럭션이 2개 이상의 캐쉬에 의해 공유 상태(shared state)로 유지될 경우 중재를 제공하지 못한다. 시스템이 모든 공유 응답들(shared responses)을 취합하고, 그런 다음 어느 캐쉬가 정보를 제공해야 하는 지를 (예를들어, 임의로) 선택하는 경우 공유 캐쉬 상태를 가진 중재가 제공될 수 있다. 그러나, 이러한 접근 방법은 시스템 메모리에 먼저 기록한 다음 시스템 메모리로부터 판독하는 것보다 빠르지 않기 때문에 어떠한 이점도 제공하지 않는다. 또한, (데이터와는 달리) 인스트럭션들은 기록되는 일이 없기 때문에 유효한 인스트럭션을 포함하는 임의의 캐쉬 블록의 상태는 항상 공유(shared)이며, 따라서 인스트럭션은 중재 방법으로 제공될 수 없다.

결과적으로, 공유 상태(shared state)를 갖는 데이터의 효율적인 중재가 가능한 캐쉬 일관성을 유지하기 위한 개선된 캐쉬 일관성 프로토콜을 고안하는 것이 바람직하다. 또한, 그러한 개선된 캐쉬 일관성 프로토콜은 분할된(sectored) 캐쉬에 대한 불필요한 버스 동작을 피하기 위하여 캐쉬 라인이 할당되었다는 표시(indication)와 소정의 캐쉬 레벨에서 정의되지 않은 그 소정의 캐쉬 레벨의 유효한 상위 계층을 제공하는 것이 바람직하다.

발명이 이루고자 하는 기술적 과제

본 발명의 목적은 캐쉬 일관성을 유지하기 위한 개선된 방법과 시스템을 제공하는 것이다.

본 발명의 다른 목적은 데이터 처리 시스템 내의 캐쉬 일관성을 유지하기 위한 개선된 방법과 시스템을 제공하는 것이다.

본 발명의 또 다른 목적은 멀티프로세서 데이터 처리 시스템 내의 캐쉬 일관성을 유지하기 위한 캐쉬 일관성 프로토콜을 제공하는 개선된 방법과 시스템을 제공하는 것이다.

발명의 구성 및 작용

본 발명의 방법과 시스템에 따른 각 프로세서는 적어도 제 1 레벨 캐쉬와 제 2 레벨 캐쉬의 계층구조를 가지며, 제 1 레벨 캐쉬는 제 2 레벨 캐쉬의 상위 계층이다. 각각의 캐쉬들은 다중 캐쉬 라인을 포함하며, 다른 캐쉬 라인은 각각 갱신 상태, 배타 상태, 공유 상태, 무효 상태, 최근 판독 상태, 상위 계층 미정 상태를 포함하는 적어도 6개의 상이한 상태의 캐쉬 라인을 식별하는데 사용되는 상태 비트 필드와 연관되어 있다. 가장 최근에 액세스된 정보의 사본을 포함하는 캐쉬 라인의 표시에 응답하여, 캐쉬 라인의 상태는 무효 상태로부터 최근 판독 상태로 천이(transition)된다. 라인필 동작을 수행함이 없이 제 1 레벨 캐쉬 내의 캐쉬 라인의 정보 갱신에 응답하여, 캐쉬 라인의 상태는 무효 상태로부터 상위 계층 미정 상태로 천이한다. 본 발명의 목적, 특징 및 장점들은 모두 이하의 상세한 설명에서 명백해질 것이다.

본 발명 자체뿐만 아니라, 바람직한 사용 모드, 추가적인 목적 및 장점들은 예시적인 실시예의 이하의 상세한 설명과 첨부 도면을 참조하여 가장 잘 이해될 수 있다.

본 발명은 캐쉬 메모리를 가진 임의의 데이터 처리 시스템에서 구현될 수 있다. 또한, 본 발명의 특징은 주(primary) 캐쉬와 부(secondary) 캐쉬를 가진 다양한 데이터 처리 시스템에서 구현될 수 있다는 것을 이해할 수 있다.

이하 도면들을 참조하면, 도 1에는 본 발명이 적용될 수 있는 멀티프로세서 데이터 처리 시스템 (10)의 블록 다이어그램이 도시되어 있다. 멀티프로세서 데이터 처리 시스템 (10)은 여러 개의 처리 유닛을 가지고 있는데, 그 중 2개가 도시되어 있다. 각각의 처리 유닛 (11a) 및 (11b)는 프로세서 코어 (12), 온-칩 인스트럭션 캐쉬 (13), 온-칩 데이터 캐쉬 (14) 및 부 캐쉬 (15)를 포함할 수 있다. 또한, 각 처리 유닛들 (11a) 및 (11b)는 입력/출력 장치 (18), 시스템 메모리 (19) 및 초기 프로그램이 로드(load)되는 동안 하나의 주변 장치로부터 운영 체제를 탐색하고 로드하는 것이 주 목적인 펌웨어(firmware) (17)을 포함하는 다양한 주변 장치들에 연결되어 있다. 처리 유닛 (11a)와 (11b)는 버스 (16)을 포함하는 다양한 수단을 통해 상기 주변 장치들과 통신할 수 있다. 당업자는 시스템 (10)이 모뎀 또는 프린터와 같은 주변 장치 연결용 직렬 및 병렬 포트와 같은 도시되지 않은 다수의 추가적인 구성 요소들을 가질 수 있다는 것을 알 수 있을 것이다.

이하 도 2를 참조하면, 종래 기술의 멀티프로세서 데이터 처리 시스템에 대한 캐쉬 일관성 프로토콜을 기술하는 상태 다이어그램이 도시되어 있다. 이러한 종래의 기술 " MESI" 프로토콜에 있어서, 캐쉬 라인은 4가지 상태 즉 갱신 (M), 배타 (E), 공유 (S), 무효 (I) 중 하나의 상태에 있을 수 있다. 각 캐쉬 라인은 일반적으로 상기 4개의 상태 중 하나를 표시하기 위해 2개의 상태 비트를 가진다. 엔트리(entry)의 초기 상태와 요구 프로세서에 의해 탐색된 액세스 형태에 따라 상태가 변경될 수 있으며, 특정 상태가 요구 프로세서의 캐쉬 라인 내의 엔트리에 대하여 세트(set)된다. 예를 들어, 캐쉬 라인이 갱신 상태(Modified state)에 있을 때, 어드레스된(addressed) 라인은 데이터 처리 시스템 내에 갱신 캐쉬 라인을 갖는 캐쉬에서만 유효하며, 갱신된 값은 시스템 메모리에 다시 기록되지 않았다. 캐쉬 라인이 배타 상태(Exclusive state)에 있을 경우, 어드레스된 라인은 언급된 상기 갱신 캐쉬 라인 내에만 존재하며 그 값은 시스템 메모리 내의 값과 일치한다. 캐쉬 라인이 공유 상태(Shared state)에 있을 경우, 캐쉬 라인은 그 캐쉬 내 및 데이터 처리 시스템 내의 적어도 하나의 다른 캐쉬 내에서 유효하다. 공유 캐쉬 라인 내의 값은 시스템 메모리 내의 값과 일치한다. 마지막으로, 캐쉬 라인이 무효 상태에 있으면, 어드레스된 메모리 위치가 캐쉬 내에 존재(resident)하지 않는다는 것을 나타낸다. 캐쉬 라인이 갱신, 배타, 공유, 또는 무효 상태 중 어느 하나의 상태에 있으면, 캐쉬 라인은 도 2에 도시된 바와 같이 특정한 버스 트랜잭션에 따라 여러 상태들 사이에서 이동할 수 있다. 예를 들어, 배타 상태에 있는 캐쉬 라인이 다른 3개의 상태 중 어느 하나의 상태로 이동할 수 있는 반면, 배타 상태로 될 수 있는 캐쉬 라인은 무효 상태 또는 갱신 상태로부터 오는 경우만 가능하다.

도 3을 참조하면, 본 발명의 바람직한 실시예에 따른 멀티프로세서 데이터 처리 시스템에 대한 캐쉬 일관성 프로토콜을 기술하고 있는 상태 다이어그램이 도시되어 있다. 이 프로토콜은 동일한 4가지 상태－갱신, 배타, 공유, 무효－를 포함한다는 점에서 종래 기술 MESI 프로토콜과 유사하지만, 2개의 새

로운 상태－최근 판독(Recently Read) (R)상태와 상위 계층 미정(Upstream Undefined) (U) 상태－를 추가로 포함한다.

I. 최근 판독 상태(Recently-Read State)

최근 판독 상태는 그렇지 않았으면 공유 상태에 있을 수 있는 가장 최근에 참조된 블록 표시를 제공하는데 사용된다. 바꾸어 말하면 2개 이상의 캐쉬들이 각각 인스트럭션의 유효한 사본을 가지고 있을 때, 모든 캐쉬는 최근에 액세스되었던 캐쉬를 제외하고 공유 상태에 있다. 이러한 가장 최근에 액세스되었던 캐쉬는 최근 판독 상태(Recently-Read State)에 있게 된다.

종래의 기술 MESI 프로토콜에 있어서, 4개의 M-E-S-I 상태는 엔트리의 초기 상태와 요구 프로세서에 의해 탐색된 액세스 형태를 기초로 하여 변경될 수 있다. 이러한 4개의 상태가 변경되는 방법은 이하에서 기술될 예외를 제외하고는 일반적으로 종래의 기술 MESI 프로토콜과 동일하다.

[표 1]

	버스 동작	마스터 상태	일관성 응답
1	판독	I → R	Shrl, Shr, 또는 Mod
2	갱신 목적 판독	I → M	Shrl, Shr, Mod, 또는 Null
3	판독	I → E	Null

표 1은 판독 형식 동작에 대한 마스터 캐쉬 상태 천이(master cache state transition)를 나타낸다.

일관성 응답이 공유 중재(Shared-Intervention) (Shrl), 공유(Shared) (Shr), 갱신(Modified) (Mod)인 경우, ″판독 미스(read miss)″ 버스 동작(표 1의 1번째 행)으로 인하여 최근 판독 상태가 된다. 일관성 응답이 ″판독 미스″ 동작시 갱신되는 경우 갱신된 캐쉬는 인스트럭션을 시스템 메모리로 전송하므로 갱신된 캐쉬는 더 이상 갱신 상태가 아니다. 갱신 목적 판독(Read With Intent To Modify: RWITM) ″미스″가 발생하면, 일관성 응답이 Shrl, Mod, 또는 Null(표 1의 2번째 행)인 경우 갱신 상태로 된다. 인스트럭션은 통상 갱신되지 않기 때문에, RWITM의 경우는 데이터에만 적용되며, 인스트럭션에는 적용되지 않는다. ″판독 미스″ 동작(표 1의 3번째 행)시 일관성 응답이 존재하지 않는 경우(Null), 종래의 MESI 프로토콜에 있어서와 같이 배타 상태로 된다.

[표 2]

	버스 동작	스누퍼 상태	일관성 응답
1	판독-버스트(Burst)	R → S	Shrl
2	판독-버스트	E → S	Shrl
3	판독-버스트	M → S	Mod
4	갱신 목적 판독	E 또는 R → I	Shrl
5	갱신 목적 판독	M → I	Mod

표 2는 버스 트랜잭션이 스누퍼(snooper) 역할을 할 때, 판독 형태의 동작에 대해 캐쉬에 어떻게 영향을 끼치는지에 대한 예를 나타내고 있다.

시작 상태(beginning state)가 배타(Exclusive) 또는 최근 판독 상태중의 하나(표 2의 1번째 행, 2번째 행 및 4번째 행)에 대해 캐쉬는 공유-중재(Shared-Intervention) 일관성 응답을 전송하는데, 이는 중재 절차에 의해 인스트럭션의 사본을 제공한다는 것(즉, 시스템 메모리 개입없이 요구 프로세서에 직접 제공하는 것)을 의미한다. 인스트럭션이 제공될 때마다 다음 상태는 판독 ″적중″이 발생 (표 2의 1번째 행 및 2번째 행)하는 경우 공유 상태(Shared state)로 되거나 ″갱신 목적 판독 적중″ 동작 (표 2의 4번째 행)인 경우 무효 상태(Invalid state)로 된다. 시작 상태가 갱신 상태인 경우(표 2의 3번째 행 및 5번째 행)에는, 프로그램 인스트럭션들은 통상 갱신되지 않기 때문에, 인스트럭션에는 적용되지 않는다. 버스 동작이 ″갱신 목적 판독″인 경우, 일관성 응답이 갱신되기 때문에 중재가 발생된다. 바꾸어 말하면, 캐쉬 블록 내의 데이터는 표 2의 3번째 행에 예시된 경우에 대해서만 요구 프로세서에 의해 시스템 메모리에 기록되고 그 시스템 메모리로부터 판독된다.

상술한 바와 같이 표 1과 2에 예시되지 않은 상태들 및 동작들에 대해 천이와 일관성 응답이 하나의 자격(qualification)을 가진 종래 기술 MESI 프로토콜에 따라 수행된다. 즉, 캐쉬 엔트리는 기록 동작에 종속되는 최근 판독 상태를 가질 수 있으며, 또한 공유 엔트리가 기록 동작에 종속되는 경우 발생하는 것과 유사한 방법으로 갱신 상태로 상기 캐쉬 엔트리는 천이한다.

또한, 엔트리는 무효 상태로부터 공유 상태로는 결코 천이할 수 없으며 대신 최근 판독 상태로 천이한다. 엔트리는 배타 상태로부터 최근 판독 상태로는 결코 천이할 수 없으며 공유 상태로 천이한다. 엔트리는 공유 엔트리가 배타 상태로 천이할 수 없는 것과 마찬가지로 최근 판독 상태로부터 배타 상태로는 결코 천이할 수 없다. 마지막으로, 엔트리는 갱신 상태로부터 최근 판독 상태로는 결코 천이할 수 없다. 즉. 요구 프로세서의 캐쉬 내 엔트리가 최근 판독 상태로 천이하는 반면, 상기 엔트리는 공유 상태로 천이한다.

이러한 최근 판독 상태에 있어서, 인스트럭션을 판독하기 위하여 블록의 소유권(ownership)이 최종 캐쉬로 이동하는데, 이는 최저 사용 빈도(LRU) 캐쉬의 대체 방법이 채용되는 경우, 가장 최근에 사용된 상태에 머물게 됨으로써 할당을 해제하는 기회를 줄이는 추가적인 장점을 가진다. 또한, 최근 판독 상태는 캐쉬된(cached) I/O 상태 장소(location)를 가장 최근에 판독했던 프로세서/캐쉬를 인터럽트(interrupt)하

는 지능형(intelligent) 입력/출력 (I/O) 제어기와 같은 다른 응용 품에 대해서도 효과적으로 사용될 수 있는데, 그 이유는 상기 프로세서/캐쉬가 I/O 장치 구동 코드를 캐쉬(cached) 했을 가능성이 가장 크며, 따라서 그 코드를 자신의 캐쉬로 펫취(fetch)할 필요가 있는 다른 프로세서에서 보다 신속하게 코드를 실행할 수 있기 때문이다.

II. 상위 계층 미정 상태(Upstream-Undefined State)

L1과 같은 상위 레벨 캐쉬(higher-level cache)는 관련된 기존 데이터(old data)를 시스템 메모리로부터 먼저 펫취할 필요없이, OOI 되게 캐쉬 라인 내에 있는 데이터 갱신을 원할 수 있다. 이러한 동작은 메모리 영역을 새로운 프로세스에 재할당할 때 통상 수행된다. 응답시, 하위 레벨 캐쉬(low -level cache)는 대응하는 캐쉬 라인을 할당하고 그 캐쉬 라인을 0으로 할 필요가 있다. 이러한 절차를 구현하는 통상적인 방법은, 시스템 메모리로부터 캐쉬 라인을 판독한 다음 상위 레벨 캐쉬 내의 캐쉬 라인에 대응되는 부분이 OOI 되게 하는 것이다. 이러한 접근 방법이 반드시 재할당되어야 하는 시스템 메모리로부터 데이터를 판독하는 것을 피하는 동작의 궁극적 목적을 저해한다는 것은 명백하다. 또한, 프로세서는 매우 짧은 시간 동안에 하위 레벨 캐쉬에 있는 나머지 부분에 포함되는 추가적인 캐쉬 라인을 할당하여 0으로 할 가능성이 있다(그러나, 하위 레벨 캐쉬는 반드시 이러한 경우에 해당된다고 할 수는 없음). 따라서, 첫째 문제는 상위 레벨 캐쉬 내에서는 유효하지만 하위 레벨 캐쉬들 내에서는 유효하지 않은 섹터들을 추적하는 것이다.

예를 들어 2개의 섹터를 가진 분할된 캐쉬에서는, 다음과 같은 3개의 경우가 가능하기 때문에 3개의 상위 계층 미정 상태들이 존재한다: (1) 2개의 섹터 중 첫 번째 섹터(" 홀수" 섹터)가 갱신된다, (2) 2개의 섹터중 두 번째 섹터(" 짝수" 섹터)가 갱신된다: (3) 섹터가 모두 갱신되지 않는다(2 개의 캐쉬 가능한 라이트-스루 동작(cachable write-through operation)의 결과로서 2개의 섹터는 둘 다 공유 상태이다). 본 명세서에서, 이러한 상태들 중 첫 번째 상태는 " U_{IM}" , 두 번째 상태는 " U_{MI}" , 세 번째 상태는 " U_{SS} " 로 표기한다. 이 2개로 분할될 캐쉬의 예에서, 각 캐쉬 라인은 라인의 상태를 표시하는 3개의 비트를 가질 수 있다. 캐쉬 라인 내에 2개 이상의 섹터가 제공되는 경우, 추가적인 상태들에 대해 캐쉬 라인 내의 추가 비트들이 필요하게 된다.

[표 3]

	상위 레벨 (L1) 동작	하위 레벨 캐쉬 천이
1	DCBZ-짝수 섹터	I → U_{IM}
2	DCBZ-홀수 섹터	I → U_{MI}
3	DCBZ-짝수 섹터	U_{IM} → M
4	DCBZ-홀수 섹터	U_{MI} → M
5	판독/갱신 목적 판독	U_{MI}|U_{IM} → I
6	임의의 L1 " 히트"	U_{SS} → U_{SS}
7	캐쉬가능한 라이트-스루	I → U_{SS}
8	임의의 다른 동작	통상의 MESI

표 3은 상위 레벨 캐쉬 내의 동작들을 포함하는 캐쉬 천이(cache transition)를 나타낸다.

표 3의 1번째 행에서, 할당-및-제로(allocate-and-zero) 동작(DCBZ)의 경우, 기록 형태의 동작은 L1 캐쉬의 캐쉬 라인 내의 짝수 섹터(2번째 섹터) 상에서 수행되며, 무효 상태에 있는 임의의 대응하는 하위 레벨 캐쉬는 " U_{IM}" 상태로 천이한다. 즉, 2번째 섹터만이 갱신된 것으로 표시되어 있다. 표 3의 2번째 행에서, DCBZ 동작이 L1 캐쉬의 캐쉬 라인 내의 홀수 섹터(1번째 섹터)상에서 수행될 때, 무효 상태에 있는 임의의 대응하는 하위 레벨 캐쉬는 " U_{IM}" 상태로 천이한다. 즉, 1번째 섹터는 갱신된 것으로 나타나 있다.

동일한 캐쉬 라인의 홀수 번째 섹터가 미리 DCBZ 동작을 수행하였고, 대응하는 하위 레벨 캐쉬가 " U_{MI}" 상태에 있을 때(표 3의 4번째 행)는 DCBZ 동작이 짝수 번째 섹터 상에서 수행되거나, 또는 동일한 라인의 짝수 번째 섹터가 이미 DCBZ 동작을 수행하였고, 대응하는 하위 레벨 캐쉬가 " U_{IM}" 상태에 있을 때(표 3의 4번째 행), DCBZ 동작이 홀수 번째 섹터 상에서 수행되는 경우, 하위 레벨 캐쉬들은 전체 라인이 갱신된다는 것을 표시하기 위해 갱신 상태로 천이할 것이다. 그러나 오직 하나의 DCBZ 동작만이 주어진 라인에 대해 이미 발생하였으며, 하위 레벨 캐쉬가 " U_{IM}" 상태 또는 " U_{MI}" 상태에서 그 라인을 가지고 있으며, 다른 무효 라인(Invalid line)이 " 판독" 동작 또는 " 갱신 목적 판독" 의 대상이라면 하위 레벨 캐쉬 라인은 무효 상태로 천이하고 갱신된 섹터는 상위 레벨 캐쉬로부터 플러시(flush)된다.

표 3의 6번째 행에서는, 대상 블록에 대해 L1 " 히트" 이 발생되고 하위 레벨 캐쉬들이 " U_{SS}" 상태에 있으면, 이들 하위 레벨 캐쉬들은 그 상태를 유지한다. 즉, 유효한 것으로 취급되지만 캐쉬(cache)되지는 않는다. 만약 캐쉬 가능한(cachable)/라이트 스루 판독(write through read) 동작이 블록(표 3의 7번째 행) 상에서 수행되고, 하위 레벨 캐쉬들이 무효 상태(Invalid state)에 있는 대응하는 블록을 가지면, 하위 레벨 캐쉬들은 " U_{SS}" 상태로 천이한다. 마지막으로, 표 3의 8번째 행에 나타난 바와 같이, 상술에서 특정되지 않은 모든 다른 L1 동작들은 통상적으로 천이한다.

[표 4]

	버스 동작	스누퍼 상태	일관성 응답
1	임의의 스누프 " 히트"	U_{IM} → I	재시도
2	임의의 스누프 " 히트"	U_{MI} → I	재시도
3	비판독 스누프 " 히트"	U_{SS} → I	재시도
4	판독 스누프 " 히트	U_{SS} → U_{SS}	공유

표 4는 트랜잭션을 스누프한 시스템 버스가 상위 계층 미정 상태들 중 하나의 상태에 있는 캐쉬들에 어떠한 영향을 미치는지를 나타낸다.

캐쉬는 상위 계층 미정 상태들 중 하나의 상태에 있는 경우, 적절한 동작을 결정하기 위해 동작을 취해야 하며 스누프 상위 계층을 전송할 필요가 있다는 것을 알고 있다. 표 4는 상위 계층 미정 상태들 중 하나의 상태에 대해 스누프 히트이 발생할 가능성이 거의 없는 경우들만을 예시한다. 이러한 경우에, 하위 레벨 캐쉬는 일관성 응답이 공유된 " U_{SS}" 상태에 대해 판독 스누프 적중이 발생하는 경우를 제외하고는, 상위 계층 캐쉬의 내용을 플러시하고 무효 상태로 천이하여 " 재시도(Retry)" 응답을 발행한다.

상기 상위 계층 미정 상태들의 경우, 불필요한 버스 동작을 실행하지 않으며 또한 캐쉬할 수 있는 라이트-스루 동작을 효율적으로 지원하지 않는 상위 레벨 캐쉬에서 유효한 섹터들을 추적하는 것과 관련된 상술한 2 가지 문제점이 해결된다. 그 결과, 메모리 대역폭이 증가되고 바이트-기록 능력뿐만 아니라 어드레스 대역폭이 프리 업(free up)된다. 설명을 간단히 하기 위해 도 3에는 하나의 상위 계층 미정 상태만이 도시되어 있으며, 상위 계층 미정 상태의 수는 캐쉬 라인 내에 있는 섹터의 수에 좌우된다는 사실에 유의해야 한다.

발명의 효과

상술한 바와 같이, 본 발명은 멀티프로세서 데이터 처리 시스템 내의 캐쉬 일관성을 유지하기 위한 캐쉬 일관성 프로토콜을 제공하는 개선된 방법을 제공한다.

본 발명은 바람직한 실시예를 참조하여 특별히 개시되고 기술되었지만, 당업자들은 본 발명의 정신과 범위를 이탈함이 없이 형태 및 세부 항목에서 다양한 변경이 이루어질 수 있다는 것을 이해할 수 있을 것이다.

(57) 청구의 범위

청구항 1

각 프로세서가 최소한 제 1 레벨 캐쉬(first-level cache)와 제 2 레벨 캐쉬(second-level cache)－여기서 제 1 레벨 캐쉬는 제 2 레벨 캐쉬의 상위 계층이며, 각 캐쉬들은 복수개의 캐쉬 라인들을 포함함－로 이루어지는 캐쉬 계층 구조를 갖는 멀티프로세서 데이터 처리 시스템(multiprocessor data-processing system) 내의 캐쉬 일관성(cache coherency)을 유지하기 위한 캐쉬 일관성 프로토콜을 제공하는 방법에 있어서,

a) 갱신 상태(Modified state), 배타상태(Exclusive state), 공유 상태(Shared state), 무효 상태(Invalid state), 최근 판독 상태(Recently-Read state) 및 상위 계층 미정 상태(Upstream-undefined state)를 포함하는 복수개의 상태들을 식별하는데 사용되는 상태 비트 필드(state bit field)를 복수개의 캐쉬 라인들의 각각에 연관시키는 단계,

b) 가장 최근에 액세스된 정보의 사본을 포함하는 캐쉬 라인의 표시(indication)에 응답하여, 상기 무효 상태(Invalid state)로부터 상기 최근 판독 상태(Recently-Read state)로 천이하는 단계 및

c) 라인필 동작(linefill operation)을 수행함이 없이 상기 제 1 레벨 캐쉬 내의 캐쉬 라인의 정보 갱신(information modification)에 응답하여, 상기 무효 상태(Invalid state)로부터 상기 상위 계층 미정 상태(Upstream-undefined state)로 천이하는 단계를 포함하는 캐쉬 일관성 프로토콜 제공 방법.

청구항 2

제 1항에 있어서, 상기 최근 판독 상태(Recently-read state)로부터 무효 상태(Invalid state)로 천이하는 단계를 추가로 포함하는 캐쉬 일관성 프로토콜 제공 방법.

청구항 3

제 1 항에 있어서, 상기 최근 판독 상태로(Recently-read state)부터 갱신 상태(Modified state)로 천이하는 단계를 추가로 포함하는 캐쉬 일관성 프로토콜 제공 방법.

청구항 4

제 1 항에 있어서, 상기 최근 판독 상태(Recently-read state)로부터 공유 상태(Shared state)로 천이하는 단계를 추가로 포함하는 캐쉬 일관성 프로토콜 제공 방법.

청구항 5

제 1 항에 있어서, 상기 갱신 상태(Modified state)로부터 공유 상태(Shared state)로 천이하는 단계를

추가로 포함하는 캐쉬 일관성 프로토콜 제공 방법

청구항 6

각 프로세서가 최소한 제 1 레벨 캐쉬와 제 2 레벨 캐쉬－여기서 제 1 레벨 캐쉬는 제 2 레벨 캐쉬의 상위 계층임－로 이루어지는 캐쉬 계층 구조를 갖는 멀티프로세서 데이터 처리 시스템 내의 캐쉬 일관성을 유지하기 위한 캐쉬 일관성 프로토콜을 가진 캐쉬 메모리에 있어서,

a) 복수개의 캐쉬 라인,

b) 상기 복수개의 캐쉬 라인들 각각에 연관된 상태 비트 필드－여기서 상태 비트 필드는 갱신 상태, 배타 상태, 공유 상태, 무효 상태, 최근 판독 상태 및 상위 계층 미정 상태를 포함하는 복수의 상태들을 식별하는데 사용됨－,

c) 가장 최근에 액세스된 정보의 사본을 포함하는 캐쉬 라인의 표시에 응 답하여, 상기 무효 상태로부터 상기 최근 판독 상태로 천이하기 위한 수 단 및

d) 라인필 동작을 수행함이 없이 상기 제 1 레벨 캐쉬 내의 캐쉬 라인의 정보 갱신에 응답하여, 상기 무효 상태로부터 상기 상위 계층 미정 상태로 천이하기 위한 수단을 포함하는 캐쉬 일관성 프로토콜을 가진 캐쉬 메모리.

청구항 7

제 6항에 있어서, 캐쉬 메모리가 상기 최근 판독 상태로부터 무효 상태로 천이하기 위한 수단을 추가로 포함하는 캐쉬 일관성 프로토콜을 가진 캐쉬 메모리.

청구항 8

제 6항에 있어서, 캐쉬 메모리가 상기 최근 판독 상태로부터 갱신 상태로 천이하기 위한 수단을 추가로 포함하는 캐쉬 일관성 프로토콜을 가진 캐쉬 메모리.

청구항 9

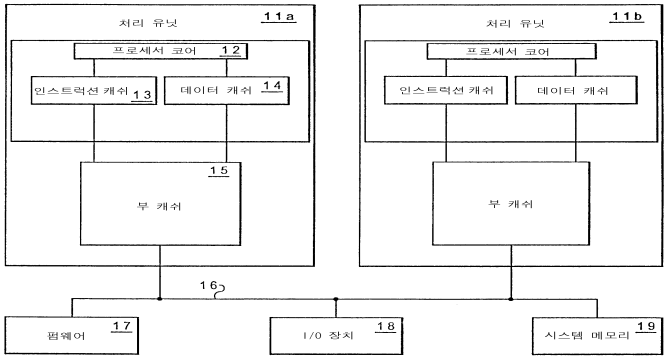
제 6항에 있어서, 캐쉬 메모리가 상기 최근 판독 상태로부터 공유상태로 천이하기 위한 수단을 추가로 포함하는 캐쉬 일관성 프로토콜을 가진 캐쉬 메모리.

청구항 10

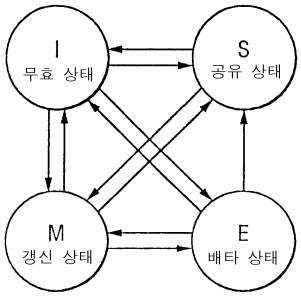
제 6항에 있어서, 캐쉬 메모리가 상기 갱신 상태로부터 공유 상태로 천이하기 위한 수단을 추가로 포함하는 캐쉬 일관성 프로토콜을 가진 캐쉬 메모리.

도면

도면1



도면2



종래 기술

도면3