(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) Int. Cl.[7]
G06F 13/42

(45) 공고일자 2000년08월01일
(11) 등록번호 10-0263633
(24) 등록일자 2000년05월18일

(21) 출원번호	10-1997-0030737	(65) 공개번호	특1998-0010806
(22) 출원일자	1997년07월03일	(43) 공개일자	1998년04월30일
(30) 우선권주장	8/675,723 1996년07월03일 미국(US)		

(73) 특허권자 인텔 코오퍼레이션 피터 엔. 데트킨
미합중국 캘리포니아 산타클라라 미션 칼리지 블러바드 2200
(72) 발명자 피슈 매튜 에이.
미국 오리건 97006 비버톤 넘버 비4 노쓰 웨스트 아일랜드 서클 315
제이콥슨 제임즈 이. 쥬니어
미국 오리건 97068 웨스트 린 폰더래이 드라이브 22485
로디하멜 마이클 더블유.
미국 오리건 97007 비버톤 사우쓰 웨스트 153 애비뉴 10165
(74) 대리인 장용식

심사관 : 오홍수

(54) 각종프로세서와버스프로토콜에적용가능한범용구조를제공하는컴퓨터시스템

요약

범용 구조를 제공하는 컴퓨터 시스템은 호스트 컴퓨터 시스템의 시스템 버스에 접속되어 있는 프로세서 카드를 포함하고 있다. 이 프로세서 카드는 상기 컴퓨터 시스템의 슬롯에 삽입되도록 채택되어 있으며, 그리고 프로세서와 버스 브리지 변환 장치를 수용하고 있다. 상기 프로세서는 상기 컴퓨터 시스템 버스의 신호 처리 프로토콜과는 다른 신호 처리 프로토콜에 따라 동작한다. 상기 버스 변환 장치는 상기 시스템 버스의 신호 처리 프로토콜을 상기 프로세서의 신호 처리 프로토콜로, 그리고 그 역으로 변환한다. 상기 버스 변환 장치는 버스 중재 변환, 버스 로크 변환, 및 캐시 코히어런시 제어를 위한 논리 회로를 포함하고 있다. 또한, 상기 카드가 상기 버스에 접속되어 있는 기타 다른 에이전트와 적절히 트랜잭션을 행할 수 있도록 입력 요구 및 출력 요구를 변환하는 논리 회로도 포함되어 있다.

대표도

도2

명세서

도면의 간단한 설명

도 1은 Pentium[R] 프로세서의 핀배열을 나타낸 도면.

도 2는 본 발명의 일실시예의 하이 레벨 블록도.

도 3은 고성능 버스를 통해 접속되어 있는 복수의 에이전트를 포함하고 있는 컴퓨터 시스템에서 본 발명의 일실시예를 보인 도면.

도 4는 도 2에 도시된 버스 변환 장치의 일실시예의 개념적인 블록도.

도 5는 도 4에 도시된 중재 변환기 회로의 일실시예를 나타낸 도면.

도 6은 도 4에 도시된 출중계 요구 변환기 회로의 일실시예의 블록도.

도 7은 도 4에 예시된 버스 로크 변환기의 일실시예를 보인 도면.

도 8은 도 4의 입중계 요구 변환기의 일실시예를 보인 도면.

도 9는 도 4에 도시된 캐시 코히어런시 제어 유닛의 일실시예의 블록도.

도 10은 본 발명의 일실시예에 사용된 버스 요구 프로토콜 변환 논리 유닛을 나타낸 도면.

발명의 상세한 설명

발명의 목적

발명이 속하는 기술분야 및 그 분야의 종래기술

본 발명은 전반적으로 고성능 컴퓨터 시스템 분야에 관한 것으로, 특히 진보된 마이크로프로세서용 버스와 관련된 신호 처리 프로토콜(signaling protocols)과 동작 요건에 관한 것이다.

매우 진보된 마이크로프로세서와 해당 버스 구조의 개발은 급속히 진행되어 왔다. 예컨대, 이전 세대의 Pentium® 프로세서에 대한 시장 수요가 아직도 확대되어가고 있는 때에, 미국, 캘리포니아, 산타 클라라 소재의 인텔 코포레이션에서는 최근에 최신의 프로세서, 즉 Pentium Pro™를 소개하였다. 상기 Pentium Pro™ 프로세서는 이전 세대의 Pentium® 프로세서보다 매우 향상된 성능을 제공하고 있고, 새로운 고성능의 파이프라인 시스템 버스 구조를 소개하고 있다.

최신의 버스 구조를 기초로 시스템을 설계하고자 하는 컴퓨터 시스템 개발자들에게는 문제가 야기되는데, 이들은 여전히 이전 세대의 프로세서, 또는 대체 버스 구조를 가지고 있는 프로세서를 기초로 프로세서 설계를 행하기를 원한다. 예컨대, 많은 시스템 설계자들은 인텔의 새로운 Pentium Pro™ 버스와 호환성이 있도록 최신의 특성을 이용하여 자신들의 컴퓨터 시스템을 설계하기를 원하지만, 이때 이들 설계자들은 Pentium® 프로세서 구조와의 호환성에 대한 현재의 시장 요구를 여전히 충족시켜 주어야 한다. 이는 일반적으로 상기 시스템 프로세서가 동일한 핀배열을 가지고 있어야 하고, 그리고 상기 시스템 프로세서는 Pentium® 프로세서용으로 설계된 버스 구조에 따라 동작해야 함을 의미한다.

발명이 이루고자 하는 기술적 과제

불행히도, 상기 Pentium Pro™ 프로세서의 버스 신호 처리 프로토콜은 상기 Pentium® 프로세서의 버스 신호 처리 프로토콜과 크게 다르다. 결과적으로, 시스템 개발자들은 최신 버스 구조와 관련된 이점은 별도로 주류의 프로세서 설계와 호환성을 유지할 것인지, 아니면 Pentium® 프로세서 버스의 시스템의 광범위한 고객 기반에 의한 시장의 수용성의 감소를 희생으로 하고 Pentium Pro™프로세서의 진보된 버스 구조의 신호 프로토콜에 따라 시스템을 설계할 것인지라고 하는 괴로운 선택을 강요받아 왔다. 그러므로, 각각 다른 버스 구조를 잠재적으로 이용하고 있는 다양한 종류의 프로세서용으로 채택되어 있는 컴퓨터 시스템 구조가 필요하다.

발명의 구성 및 작용

이하에서 알 수 있는 바와같이, 본 발명은 표준화된 컴퓨터 버스 인터페이스를 통해 호스트에 접속되어 있는 프로세서 서브시스템(또는 모듈)을 포함하고 있는 범용 컴퓨터 구조를 제공함으로써 종래기술에 내재하는 문제를 해결해 준다. 본 발명은 시스템 기본 구조를 수정할 필요가 없도록 해 주면서 다양한 프로세서 종류와의 호환성을 제공한다. 본 발명은 유리하게 이전 세대의 프로세서(예컨대, Pentium® 프로세서)가 심리스(seamless) 방식으로 고성능(즉, Pentium Pro™ 프로세서) 시스템 버스에 접속되어 있는 기타 다른 에이전트(agent)와 정보를 주고 받을 수 있도록 할 수 있다.

<u>발명의 개요</u>

본 발명은 표준화 시스템 버스를 지원하는 호스트 컴퓨터 시스템용의 프로세서와 버스 변환 장치를 포함하고 있는 프로세서 서브시스템를 구비하고 있는 범용 컴퓨터 구조를 포함하고 있다. 표준화 버스 인터페이스는 상기 시스템 버스측에 접속을 제공하고 그리고 상기 프로세서 서브시스템과 상기 호스트 컴퓨터 시스템간의 통신을 가능하게 해 준다. 상기 호스트 컴퓨터는 호스트 프로세서, 메인 메모리, 주변 장치 등과 같은 구성 요소를 포함한다. 상기 서브시스템내의 프로세서는 상기 호스트 시스템의 상기 표준화 버스 인터페이스와는 상이한 특정 버스 인터페이스에 따라 동작한다. 상기 버스 변환 장치는 상기 호스트 시스템의 표준화 버스 인터페이스와 상기 프로세서의 특정 버스 인터페이스간의 변환을 행한다.

일실시예에서, 상기 프로세서 서브시스템은 상기 호스트 컴퓨터 시스템의 슬롯에 플러그 접속 또는 삽입될 수 있는 카드에 수용되어 있다. 상기 카드는 상기 시스템 버스의 신호 처리 프로토콜과는 다른 특정 신호 처리 프로토콜에 따라 동작하는 프로세서를 포함하고 있다. 상기 버스 변환 장치는 상기 프로세서의 핀과 상기 카드의 표준화 버스 인터페이스에 접속되어 있고, 이 버스 인터페이스는 상기 시스템 버스에 접속되어 있다. 상기 버스 변환 장치는 상기 시스템 버스의 신호 처리 프로토콜을 상기 프로세서 서브시스템의 신호 처리 프로토콜로, 그리고 그 역으로 변환한다.

상기 프로세서 서브시스템과 인터페이스를 행하는 상기 호스트 컴퓨터 시스템은 다양한 자율 에이전트를 지원할 수 있다. 이들 자율 에이전트는 부가적인 프로세서, 직접 메모리 액세스(DMA) 디바이스와 같은 각종 잘 알려진 디바이스, 또는 기타 다른 서브시스템을 포함하고 있을 수 있다. 이들 에이전트는 실시형태의 일예에서 파이프라인 버스 구조를 통해 서로 통신한다. 상기 파이프라인 버스에 접속되어 있는 에이전트는 대칭 에이전트 또는 우선 순위 에이전트로 분류된다. 상기 대칭 에이전트는 순환형 우선 순위 알고리즘에 따라 분산 중재 기술을 지원한다.

상기 프로세서 서브시스템은 상기 파이프라인 버스상의 대칭 에이전트로 분류되고, 그리고 순환형 우선순위 알고리즘에 따라 상기 프로세서 대신에 상기 버스의 소유권을 요구한다. 특정 구현예에서, 라운드 로빈(round-robin) 알고리즘이 이용될 수 있다. 대부분의 경우에, 우선 순위 에이전트는 상기 버스상으로 우선 순위 요구 신호를 전송함으로써 임의의 대칭 에이전트보다 버스의 소유권을 바로 획득할 수 있다.

본 발명은 이하의 상세한 설명으로부터 그리고 첨부된 도면으로부터 보다 완전하게 이해될 수 있지만, 본 발명은 이하의 특정 실시예에 한정되는 것으로 간주되어서는 안되며, 이는 오히려 설명 및 이해를 돕기 위한 것이다.

<u>상세한 설명</u>

범용 컴퓨터 구조에 대해 설명한다. 본 발명에 따라, 프로세서 서브시스템은 관련 시스템 버스 구조를 가지고 있는 호스트 컴퓨터 시스템과 통신한다. 프로세서 서브시스템은 프로세서의 신호 처리 프로토콜을 시스템 버스의 신호 처리 프로토콜로 변환하는 버스 변환 장치와 더불어 플러그인 카드에 수용되어 있는 프로세서를 포함하고 있다. 이하의 설명에서, 본 발명의 철저한 이해를 제공하기 위해, 에이전트 종류, 논리 블록, 신호 접속 등과 같은 수많은 특정 내용에 대해 설명한다. 데이타 처리 기술 분야에서 통상의 기술을 가진 당업자는 다수의 이들 특정 내용없이도 본 발명이 실시될 수 있음을 이해할 수 있다. 기타 다른 예에서, 잘 알려진 신호 타이밍, 구성 요소, 및 회로에 대해서는 본 발명의 불명료를 피하기 위해 설명하지 않는다.

컴퓨터 기술 분야에 숙련된 자는 또한 후술되는 설명이 관련 분야내에서 특정의 정의적 의미를 가지고 있는 각종 용어를 포함하고 있음을 알 수 있다. 예컨대, 용어 "표준화 버스 인터페이스"는 에이전트와 버스간의 신호 전송이 시스템 전체에 걸쳐 적용되고, 일반적으로 사용되는 신호 처리 프로토콜을 따르고 있음을 의미함을 당업자는 알고 있다. 또한, 상기 표준화 버스 인터페이스는 각종 전문기관에 의해 추진된 업계 표준 타입을 반드시 참조하는 것은 아니다. 오히려, 상기 인터페이스는 다수의 기준중 어느 한 기준을 기초로 선택될 수 있다. 예컨대, 표준화 버스 인터페이스가 종래의 프로세서 세대와의 역방향 호환성을 제공하는 고성능 버스 구조를 기초로 하여 본 발명의 실시형태가 개시된다. 기타 다른 예에서, 비용을 줄이기 위해 또는 구성요소의 복잡성을 감소시켜 대규모 시장성을 지원하기 위해, 상기 규격화 버스 인터페이스는 비교적 간단한 버스 구조를 기초로 하고 있을 수 있다.

도 1에는 인텔의 Pentium® 마이크로프로세서의 신호 인터페이스가 예시되어 있다. 배경 설명으로서, 상기 Pentium® 마이크로 프로세서와 관련된 버스 신호 처리 프로토콜 뿐만아니라, 도 1에 도시된 신호의 동작과 기능에 대한 상세한 설명이 수많은 간행물에서 발견될 수 있다. 예컨대, 돈 앤더슨(Don Anderson)과 톰 샌리(Tom Shanley) 공저, 애디슨-웨슬리 출판사(1995)의 문헌 "펜티엄 프로세서 시스템 구조, 제2판"에는 그러한 설명이 포함되어 있다.

이하의 설명은 본 발명의 예시적 실시예의 Pentium® 마이크로프로세서를 대상으로 하고 있지만, 상기 Pentium® 마이크로프로세서와 핀 호환성을 유지하고 있는 다른 내부 구조를 이용하고 있는 기타 다른 마이크로프로세서가 현재 존재하고 있거나 장래에 존재할 수 있음은 물론이다. 이와 유사하게, 기타 다른 프로세서는 유사한 기능을 가지고 있는 신호를 제공하며, 그리고 상이한 핀배열로 설계되어 있지만, 상기 Pentium® 마이크로프로세서에 정의된 버스 구조와 호환성을 유지하고 있다.

그러므로, 본 출원의 Pentium® 마이크로프로세서에 대한 참조는 이들 등가물이 포함될 수 있도록 행해져야 한다. 또 다른 방법을 위해, 본 발명은 Pentium® 마이크로프로세서에 한정되어 있지 않으며, 오히려 본 발명은 각종 데이타 처리 디바이스중 어떤 디바이스로도 실시될 수 있다. 또한, 본 발명은 Pentium® 프로세서 구조를 기초로 한 호스트 시스템에 대해 설명되어 있지만, 당업자는 상이한 구조를 가지고 있는 프로세서가 본 발명에 의해 제공된 이점을 이용할 수 있도록 쉽게 채택될 수 있음을 이해해야 한다.

이제, 도 2를 참조하면, 본 발명의 일실시예의 하이 레벨의 블록도가 도시되어 있다. 도 2의 실시예에서, 복수의 집적회로 디바이스는 인쇄회로기판(PCB) 카드(또는 모듈)(200)에 장착 또는 수용되어 있다. 카드(200)는 라인(17)을 따라 하나 이상의 캐시 메모리(12)에 접속되어 있는 프로세서(11)를 포함하고 있다. 라인(18)을 통해 프로세서(11)와 접속되어 있는 버스 변환 장치(15)가 카드(200)에 포함되어 있다. 버스 변환 장치(15)는 시스템 버스(101)에 접속되어 있다. 설명된 실시예에서, 시스템 버스(101)는 표준화 버스 인터페이스를 가지고 있는 고성능 파이프라인 버스를 구비하고 있다.

데이타(19), 주소(13) 및 제어 라인(16)은 각각 프로세서(11)로부터 (도시된) 시스템 버스(101)측으로 집적 접속될 수 있거나, 대안으로 변환 장치(15)를 경유하여 접속될 수 있다.

이하에서 상세히 설명되는 바와같이, 버스 변환 장치(15)는 프로세서(11)와 관련된 버스 신호 처리 프로토콜 및 전기적 특성을 시스템 버스(101)와 관련된 신호 프로토콜 및 전기적 특성으로, 그리고 그 역으로 변환하는 기능을 한다. 이 방법으로, 정보가 상기 프로세서와 시스템 버스(101)에 접속되어 있는 기타 다른 에이전트 간에 전송될 수 있다. (카드(200)에 포함된 기타 다른 디바이스에 접속되어 있는 프로세서(11)로의 기타 다른 입/출력 라인이 존재함을 나타내기 위해 신호 라인(14)이 도시되어 있음에 주의하자. 예컨대, 라인(14)은 클럭 입력 신호, 동작 전위 공급 라인, 부가적인 집적회로에 대한 접속 등을 포함하고 있을 수 있다.)

이제, 도 3을 참조하면, 고성능 버스(101)에 접속되어 있는 다수의 에이전트를 포함하고 있는 컴퓨터 시스템에 대한 본 발명의 일실시예가 도시되어 있다. 도 3의 실시예에서, 버스(101)는 Pentium Pro™ 프로세서용으로 설계된 구조와 같은 구조를 가지고 있는 파이프라인을 구비하고 있다. 하나의 에지를 따라 설치되어 있는 복수의 인터페이스 단자를 가지고 있는 프로세서 카드(200)가 예시되어 있다. 이들 단자는 한 세트의 대응 단자 접속을 가진 슬롯을 가지고 있는 접속기(201)와 플러그 접속된다. 예컨대, 이 구조는 컴퓨터 시스템내에서 이용가능한 슬롯을 통해 마더 보드에 접속되어 있는 도터 카드(daughtercard)를 나타낼 수 있다. 이 구조에 의해, 고성능버스(101)는 상이한 각종 프로세서를 각각 포함하고 있을 수 있는 복수의 프로세서 카드와 인터페이스를 행할 수 있다.

도면으로부터 알 수 있는 바와같이, 프로세서 카드(200)는 캐시 메모리(12a-12c)와 접속되어 있는 프로세서(P_A)(11)를 구비하고 있고, 그리고 또한 버스 변환 장치(15)를 통해 버스(101)에 상호 접속되어 있다. 버스(101)에 접속되어 있는 기타 다른 디바이스로는 버스 마스터(205), 직접 메모리 액세스(DMA) 디바이스(202), I/O 디바이스(들)(207), 확장 디바이스(203), 및 클러스터(206)에 편성된 프로세서(P_B)의 그룹이 있다. 예컨대, 프로세서(P_B)는 파이프라인 버스(101)의 버스 구조와 호환성이 있는 Pentium Pro™ 프로세서를 구비하고 있을 수 있다. 이와같이, 마이크로프로세서(MP) 시스템은 도 3의 컴퓨터 시스템에 예시되어 있다.

이하를 설명할 목적으로, 고성능 버스(101)는 인텔 코포레이션의 Pentium Pro™ 프로세서 버스로 간주될 수 있다. 상기 Pentium Pro™ 프로세서 버스와 관련된 신호 처리 프로토콜에 대해서는 간행물, "Pentium Pro™ 프로세서 패밀리 개발자의 매뉴얼, 제1권: 사양서"(1996)에 설명되어 있으며, 이 간행물은 미국 1-800-879-4683번으로 전화를 함으로써 이용가능하고 이 간행물은 본 명세서에서 참조 문헌으로서 포함되어 있다. 하지만, 상기 Pentium Pro™ 프로세서 버스의 각종 속성은 본 발명에 관련되어 있으므로, 본 발명의 특정 측면을 보다 잘 이해할 수 있도록, 이들 속성에 대해 설명한다.

먼저, Pentium Pro™ 프로세서 버스의 버스 중재 프로토콜에 따라, 두 분류의 버스 에이전트, 즉 대칭 에이전트와 우선 순위 에이전트가 있다는 것이 이해된다. 대칭 에이전트는 라운드 로빈(순환형 우선 순위) 알고리즘을 이용하여 공정하고 분산된 중재를 지원한다. 각각의 대칭 에이전트는 RESET시에 할당된 특정 에이전트 식별(ID); 다음 중재 사건에 대한 최하위 우선 순위를 가지고 있는 상기 대칭 에이전트를 반영하고 있는 회전 ID 값; 및 "비지(busy)" 상태 또는 "아이들(idle)" 상태를 지시해 주는 대칭 소유권 상태 지시기를 가지고 있다. 도 3의 실시예에서, 예컨대, 프로세서 카드(200)는 버스(101)에 접속되어 있는 여러 대칭 에이전트중 한 에이전트로 간주된다.

도 3에 도시된 구성과 같은 일반적인 시스템 구성에서, 대칭 에이전트는 개별적으로 또는 클러스터(206)에 편성된 프로세서(P_B), 버스 마스터(205), 및 각종 확장 디바이스(203)와 같은 디바이스를 포함하고 있을 수 있으며, 이때 이들은 버스(101)에 접속되어 있다. 도 3의 컴퓨터 시스템에서, 직접 메모리 액세스(DMA) 디바이스(202)는 상위 우선순위 에이전트로서 동작할 수 있는데, 이는 이 디바이스가 통상적으로 메모리에 대해 직접적인 경로를 필요로 하기 때문이다. 기타 다른 구성에서, 상기 우선 순위 에이전트(들)는 도 3에 I/O 블록(207)으로 도시된 바와같이, 복수의 I/O 디바이스를 대신하여 중재를 행하는 상위 우선 순위 중재기를 포함하고 있을 수 있다.

중재 사건은 새로운 대칭 버스 소유자를 결정 및 변경하는 프로세스이다. 모든 중재 사건에서는, 최상위 우선 순위를 가지고 있는 대칭 에이전트가 대칭 소유자가 된다. 상기 대칭 소유자는 반드시 전체적인 버스 소유자일 필요는 없는데, 이는 어떤 대칭 소유자보다 상위의 우선 순위를 가지고 있는 우선 순위 에이전트가 한 대칭 에이전트로부터 상기 버스의 소유권을 가로챌 수 있기 때문이다. 일단, 상기 우선 순위 에이전트가 상기 버스를 중재해 주면, 새로운 트랜잭션이 진행중인 버스 로크 동작의 일부분이 아닌 경우에는 상기 에이전트는 상기 대칭 소유자가 새로운 요구 단계로 진입하는 것을 막는다. 버스 로크 동작은 인터럽트될 수 없는 단위 동작이다. 이러한 버스 동작의 일예로는 판독-수정-기록 사이클을 들 수 있다.

이제, 도 4를 참조하면, 이전에 도 2 및 도 3에 도시된 버스 변환 장치(15)의 일실시예의 개념적인 블록도가 도시되어 있다. 점선(10,100)은 각각 프로세서(11)와 버스(101)의 변환기(15)간의 버스 신호 인터페이스를 나타낸다. 각각의 인터페이스(10,100)는 각각의 버스 인터페이스와 관련된 전압과 논리레벨을 변환하는데 필요한 데이타 및 주소 래치, 전압 변환 회로, 신호 변환 논리 회로 등을 포함하고 있을 수 있음을 당업자는 알 수 있다.

파이프라인 버스(101)에 대한 인터페이스를 행하는 버스 인터페이스(100)의 경우에, 인터페이스 유닛 순차 큐(IOQ)(70)가 포함되어 있다. 이 IOQ(70)는 버스(101)의 현재 상태의 트랙을 유지해 주는 Pentium Pro™ 프로세서 버스와 관련된 표준 논리 블록이다. 모든 버스 에이전트는 상기 버스측으로 발생된 모든 트랜잭션을 트래킹하기 위해 동일한 IOQ 상태를 유지한다. 트랜잭션이 상기 버스측으로 발생되면, 이 트랜잭션은 또한 각각의 에이전트의 IOQ내로 진입한다.

IOQ(70)의 깊이는 얼마나 많은 순차 트랜잭션이 상기 버스상에 동시에 미처리로 존재할 수 있는지에 대한 한계치이다. 트랜잭션은 발생된 순서와 동일한 순서로 응답과 데이타를 수신하므로, IOQ(70)의 상부에서의 트랜잭션은 상기 응답 및 데이타 단계로 진입하기 위한 새로운 트랜잭션이다. 상기 응답 단계의 완료 이후에, 트랜잭션은 상기 IOQ로부터 제거된다. 상기 IOQ는 또한 기타 다른 버스 프로토콜 신호 뿐만아니라, HIT#/HITM#, DRDY#, DBSY#와 같은 신호를 전송하는 일을 한다. 상기 IOQ의 디폴트 깊이는 8이지만, 버스 에이전트는 RESET#의 핀(A7#)의 기동으로 깊이 1인 IOQ(70)를 구성할 수 있다.

상이한 버스 프로토콜과 관련된 중재기 신호를 변환하는 버스 중재 변환기(BAC)(60)가 버스 변환 장치(15)내에 포함되어 있다. 예컨대, BAC(60)에 의해, 프로세서(11)는 버스(101)와 관련된 버스 프로토콜로 적절히 변환된 중재기 신호를 가지고 있음으로써 파이프라인 버스(101)에 대한 제어 또는 소유권을 획득할 수 있다.

버스 중재기 변환기(60)의 상세 블록도가 도 5에 예시되어 있다. BAC(60)에는 에이전트 식별(ID) 결정 상태 머신(61), 대칭 소유자 결정 상태 머신(62), 정지 표명 논리 회로(63), 및 버스 요구 논리 회로(64)가 포함되어 있다. 시스템 버스(101)의 대칭 중재 기술에서, 상기 버스에 대한 각각의 에이전트 요구 액세스는 적절한 요구(BREQ#) 신호를 표명해야 함을 알 수 있다. 예컨대, 프로세서(11)는 이 프로세서(11)의 버스 프로토콜에 따라 상기 BREQ# 신호를 구동함으로써 버스(101)로의 요구를 전송하는 의도를 나타낼 수 있다.

상기 신호(BREQ, LOCK#, HLDA)는 버스 요구 논리 회로(64)로부터의 프로세서(11)의 입력이다. (표명될 때 논리 로우 상태를 가정하는 각각의 신호의 뒤에는 파운드 기호(#)가 있음에 주의하자. 예컨대, 상기 LOCK# 신호는 프로세서(11)가 단위 트랜잭션을 수행하고 있을 때 로우로 표명된다.)

상기 BREQ 신호는 버스 사이클을 수행하기 위해 버스(101)의 소유권을 획득할 필요가 있음을 버스 요구 논리 회로(64)측에 알려 주는, 프로세서(11)에 의해 표명되는 버스 요구 출력이다. 상기 HLDA 신호는 상기 버스를 더 이상 소유하지 않음을 다른 요구 디바이스측에 알려 주는, 상기 프로세서(11)에 의해 표명되는 홀드 확인 출력이다. 프로세서(11)와 관련된 버스 프로토콜에 따라, HLDA는 모든 미처리 버스 사이클(즉, 이전에 파이프라인처리된 사이클)이 완료될 때까지 표명되지 않는다. 상기 버스 요구 논리 회로(64)는 프로세서(11)에 의해 발생된 버스 요구 신호를 시스템 버스(101)의 프로토콜과 관련된 상기 버스 요구 신호와 호환성이 있는 상기 BREQ#[0]으로 변환한다.

일실시예에서, 버스 중재기 변환기(60)는 Pentium Pro™ 프로세서 버스상에서 행해지는 라운드 로빈 중재 기술에 따라 대칭 에이전트로서 버스(101)의 소유권을 획득한다. 버스(101)에 적절히 접속하기 위해, 각

각의 대칭 에이전트는 비대칭 에이전트 식별(ID)을 할당받아야 한다. 이는 에이전트 ID 결정 유한 상태 머신(61)의 목적이다. 파워 온 또는 리세트시에, 대칭 에이전트 ID 결정 상태 머신(61)은 프로세서 카드(200)의 에이전트 ID 할당을 결정하기 위해 상기 BREQ#[3:0] 신호 라인을 수신한다. 상기 에이전트 ID 정보는 대칭 소유자 결정 상태 머신(62)측에 제공된다.

시스템 버스(101)의 소유권을 획득하기 위해, 상기 대칭 에이전트 BREQ#[0] 신호가 표명되고, 그리고 상기 대칭 소유자 결정 상태 머신(62)은 소유권이 상기 Pentium Pro™ 프로세서 버스의 버스 프로토콜에 따라 대칭 에이전트로서 획득될 때를 알아 내기 위해 상기 버스의 상태를 조사한다. 시스템 버스(101)와 관련된 상기 버스 중재 신호는 BREQ#[3:0], BPRI#, LOCK#(도 7에 도시됨), 및 RESET#를 포함하고 있다. 상기 BREQ#[3:0] 버스 신호는 프로세서 카드(200)의 버스 변환 장치(15)를 포함해서, 대칭 에이전트에 회전식으로 접속되어 있다. 이 배열은 파워 온 또는 RESET 동안에 특정 에이전트 ID를 가지고 있는 모든 대칭 에이전트를 초기화한다. 버스 요구 신호 BREQ#[3:0]는 개별적인 대칭 에이전트들이 시스템 버스(101)의 소유권을 중재하기 위해 버스 소유권 요구를 전송 및 수신하는데 이용하는 메카니즘을 제공한다.

상기 BPRI# 신호는 대칭 에이전트가 상위 우선 순위 버스 에이전트로부터 버스 소유권 요구를 수신하는 우선 순위 요구 신호이다. 예컨대, BPRI#는 버스(101)의 직접적인 소유권을 중재할 수 있도록 우선 순위 에이전트에 의해 표명될 수 있다. 버스(101)에 접속되어 있는 상기 대칭 에이전트는 우선 순위 에이전트가 상기 버스의 소유권을 요구하고 있음을 알리는 지시로서 상기 BPRI# 신호를 수신한다.

버스(101)상에서의 중재 사건의 경우에, 프로세서 카드(200)는 상기 시스템에서 현재 최상위의 우선 순위를 가지고 있으면 상기 버스의 대칭 소유자가 될 수 있다. 따라서, 우선 순위 에이전트는 상기 버스를 요구할 수 있고 그리고 프로세서 카드(200)상으로 소유권을 획득할 수 있다. 예컨대, I/O 디바이스(207)(도 3 참조)는 상기 프로세서 카드(200)가 물론 단위 트랜잭션을 수행하고 있지 않고 그리고 상기 LOCK# 신호를 표명하지 않은 경우에는, BPRI#를 표명함으로써 도터 카드(200)로부터 상기 버스의 소유권을 획득할 수 있다. 물론, 프로세서 카드(200)는 다른 실시예에서 우선 순위 에이전트로 지정될 수 있다.

홀드 표명 논리 회로(63)는 버스(101)로부터 상기 BPRI# 신호를 수신하여 프로세서(11)측으로 HOLD 신호와 AHOLD 신호입력을 발생해 줄 수 있도록 접속되어 있다. 상기 BPRI# 신호가 상기 버스(101)상에 표명되면, 이는 상기 버스의 소유권을 획득하기 위해 상위의 우선 순위 에이전트가 개입하고 있음을 의미한다. 상기 BPRI# 신호에 응답하여, 홀드 표명 논리 회로(63)는 어떤 추가적인 요구를 하지 않도록 프로세서(11)측에 이를 통보한다.

상기 AHOLD(주소 홀드) 신호는 프로세서(11)가 그 주소 버스 구동을 중지하도록 해 주며, 이에따라 상기 프로세서는 버스 사이클을 제어할 수 없음에 주의하자. 그러므로, 상기 HOLD 신호와 상기 AHOLD 신호의 구동에 의해, 버스 변환 장치(15)는 버스(101)상의 다른 대칭 에이전트와 더불어 동작할 수 있다. 이에 따라, 프로세서(11)는 디폴트로써 상기 버스를 소유할 수 없게 되며, 반면에 Pentium® 프로세서의 버스 구조에서는 상기 프로세서가 디폴트로써 상기 버스를 소유하게 된다. 복수의 프로세서 또는 에이전트를 가지고 있는 시스템 구성에서는, 상기 HOLD 신호와 AHOLD 신호를 사용하여 둘 이상의 프로세서가 서로 백오프하는 가능성이 배제되는 "라이브 로크(live lock)" 상태에 도달한다.

버스 요구 논리 회로(64)는 또한 상위의 우선 순위 에이전트가 상기 BPRI# 신호를 사용하여 상기 버스의 소유권을 요구하는 경우에, 프로세서(11)가 상기 버스의 제어를 바로 해제할 수 있도록 상기 프로세서(11)측에 상기 BOFF# 신호 입력을 발생해 준다. 상기 BOFF# 신호가 제거된 후에, 프로세서(11)는 전체적인 버스 사이클을 재시작한다.

다른 실시예에서, 프로세서 카드(200)는 컴퓨터 시스템에서 대칭 에이전트 대신에 상위의 우선 순위 에이전트로서 동작한다.

이제, 도 4 및 도 6을 참조하면, 버스 변환 장치(15)는 시스템 버스(101)상에서의 발생을 위해 상기 프로세서(11)에 의해 발생된 요구를 변환하는 출중계 요구 변환기(ORC)(20)를 더 구비하고 있다. 예시적인 실시예에서, 일단, 버스(101)의 소유권이 Pentium® 프로세서에 의해 획득되었으면, 상기 출중계 요구 엔코딩은 Pentium® 프로세서의 프로토콜로부터 상기 Pentium Pro™ 프로세서 버스(101)의 신호 처리 프로토콜로 변환되어야 한다. 또한, 상기 Pentium Pro™ 버스는 두 클럭 요구 사이클을 수행하는 반면에, 상기 Pentium® 프로세서는 단지 하나의 클럭 사이클로 동작한다. 그러므로, ORC(20)는 Pentium®으로부터 상기 Pentium Pro™ 프로세서 버스측으로 상이한 요구형 핀을 변환할 뿐만아니라 프로세서(11)에 의해 발생된 요구를 적절히 배열하는 책임을 진다.

출중계 요구 변환기(20)는 요구 사이클링 유한 상태 머신(21), 요구 엔코더(22,23), 및 속성 엔코더(26)를 포함하고 있다. 상기 요구 사이클링 상태 머신(21)은 프로세서(11)가 버스(101)의 소유권을 획득하였음을 지시해 주는 BAC(60)로부터 라인(66)상의 신호를 수신한다. 이때, 프로세서(11)는 유효 주소와 버스 사이클 정의가 존재함을 지시해 주기 위해 그 ADS#(주소 상태) 출력 신호를 공급할 수 있다.

상태 머신(21)은, 단일 사이클 프로세서 요구가 시스템 버스(101)상의 두 별개의 클럭 사이클로서 적절히 배열될 수 있도록, 각종 요구 및 엔코딩 변화를 위해 멀티플렉서 회로(24,25)에 접속되어 있는 제어 신호를 발생하는데 상기 정보를 이용한다. 상기 두 요구 사이클에 대한 엔코딩은 블록(22,23)에 의해 수행되며, 이들 블록은 프로세서(11)로부터 기록/판독(W/R#) 신호, 메모리 또는 I/O 트랜잭션(M/IO#) 신호 및 버스 사이클 정의(CACHE#) 신호를 입력으로서 수신한다. 상기 CACHE# 신호는 캐시 재기록 사이클 동안에 상기 프로세서에 의해 표명되고, 또한 상기 프로세서가 버스트(burst) 버스 사이클을 이용하여 캐시 라인 필(line-fill)을 수행하고자 함을 외부 메모리측에 통보해 준다.

또한, 요구 엔코더(22)는 요구된 판독 동작의 종류를 지시해 주기 위해 상기 D/C#(데이타/코드) 신호를 수신한다.

블록(22,23)에 의해 생성된 상기 엔코딩된 출력 신호는 멀티플렉서(24)측에 입력된다. 멀티플렉서(24)로

부터 선택된 출력은 도 6에 도시된 바와같이, 상기 요구 사이클링 상태 머신(21)에 의해 제어된다. ADS# 신호가 요구 사이클링 상태 머신(21)에 의해 버스(101)측으로 발생되면, 프로세서(11)로부터의 각종 요구 종류가 엔코더(22)에 의해 제1사이클동안 엔코딩된다. 이때, 프로세서(11)에 의해 제공된 주소 신호는 멀티플렉서(25)를 통해 시스템 버스 주소 라인 A#[35:3]측으로의 출력을 위해 선택된다. 상기 요구 종류에 대한 기본 정보는 또한 제1사이클 동안에 상기 REQ#[4:0] 라인을 통해 상기 시스템 버스(101)측으로 배치된다. 요구 엔코더(22)는 본 발명의 일실시예에 따라 Pentium® 형 핀을 상기 Pentium Pro™ 프로세서 버스상의 적절한 요구형 신호로 변환하는 통상적인 조합 논리를 사용하여 구현될 수 있다.

상기 제2클럭 사이클 동안에, 멀티플렉서(24)는 상기 시스템 버스(101)측으로의 출력을 위해 엔코더(23)의 Reqb[4:0] 출력을 선택한다. 또한, 상기 제2클럭 사이클 동안에, 상기 주소 신호 라인은 상기 요구 종류에 관한 부가적인 정보를 상기 파이프라인 버스측으로 전송하는데 이용된다. 이 정보는 바이트 인에이블 신호 라인(BE#[7:0]); 로크된 전송 요구가 예측된 버스 사이클수의 2배라는 논리를 통보하는 분할 사이클(SCYC#) 신호; L2 캐시가 기록중인 라인에 대한 재기록 또는 기록 쓰루 폴리시(write-through policy)를 사용해야 하는지를 지정해 주기 위한 페이지 기록 쓰루(PWT) 신호; 및 CACHE# 신호와 같은 요구 종류의 각종 속성을 포함하고 있다. 이 정보는 상기 제2사이클 동안에 멀티플렉서(25)에 의해 시스템 버스(101)의 주소 라인측으로의 출력으로서 선택된다.

트랜잭션을 지연시킬 수 있는(즉, 트랜잭션으로 하여금 고장을 완료하도록(complete)해 주는) 응답 에이전트를 포함하고 있는 시스템 구성에서, ORC(20)는 버스(101)에 출력된 각 요구가 있는 DEN# 신호 라인을 표명 해제하는 논리를 포함해야 한다. 상기 DEN# 신호는 지연 인에이블 신호이고, 그리고 EXF1#/Ab4# 핀(즉, 제2사이클 주소 핀의 비트 4)상의 요구 단계의 제2클럭에서 버스(101)상에서 구동된다. 예컨대, 이 신호 처리 논리 회로는 상태 머신(21)의 통상적인 조합 논리 회로에 포함되어 있을 수 있다. 각각의 출중계 요구를 갖고 있는 DEN#를 표명 해제하면 임의의 응답 에이전트가 트랜잭션을 지연하는 것을 방지한다.

또한, 주소 패리티 신호와 요구 패리티 신호는 출중계 요구의 일부분으로서 버스(101)측으로 블록(27,28)에 의해 각각 발생된다.

버스 변환 장치(15)는 단위 트랜잭션 동안에, 즉 인터럽트될 수 없는 여러 클럭 사이클에서의 트랜잭션 동안에, 상기 버스가 로크될 수 있도록 해 주는 버스 로크 변환기(BLC) 회로(50)를 더 포함하고 있다. 상기 Pentium® 프로세서용 로크 프로토콜은 Pentium Pro™ 프로세서 버스상에서 구현된 로크 신호 처리 프로토콜과 유사하지만, 각각의 버스 프로토콜에 관련하여 로크가 표명되는 정확한 타이밍에서는 상이하다.

도 7에는 ORC(20)로부터의 입력 뿐만아니라, 프로세서(11)로부터 LOCK# 신호를 수신하는 로크 변환 유한 상태 머신(51)을 구비하고 있는 버스 로크 변환기가 예시되어 있다. 시스템 버스(101)에서, 상기 LOCK# 신호는 모든 에이전트들 사이를 결합하는 버스 상의 쌍방향 신호이다. Pentium Pro™ 프로세서의 버스 중재 프로토콜에 따라, 현재 버스 소유자는 개별적인 버스 로크 동작을 정의하기 위해 LOCK#를 표명할 수 있다.

프로세서(11)가 그 LOCK# 신호를 표명한 직후에, 시스템 버스(101)는 버스(101)의 신호 처리 프로토콜에 따라 로크 변환 상태 머신으로부터 상기 변환된 LOCK# 신호를 수신한다. 기본적으로, 상태 머신(51)은 버스(101)상에 대응 LOCK# 신호를 표명하기 전에 ORC(20)에 의해 발생되도록 ADS# 등과 같은 신호를 대기하면서 변동가능한 지연을 실행한다. 버스(101)상의 상기 LOCK# 신호는 일련의 단위 트랜잭션을 통해 지속되고, 다음에 상기 트랜잭션이 완료된 후에 표명해제된다. 다시, 표명 해제는 버스(101)의 적절한 신호 처리 프로토콜에 따라 수행된다.

도 4에서 알 수 있는 바와같이, 버스 변환 장치(15)는 또한 상기 시스템 버스상의 신호 요구를 프로세서(11)측으로 입력될 수 있는 신호로 변환하는 입중계 요구 변환기(IRC)(30)를 포함하고 있다. Pentium® 프로세서의 신호 처리 프로토콜에 따라, 외부 주소 스트로브(EADS#) 신호는 유효 주소가 그 국부 주소 버스상에 존재하고 이 주소가 스누핑될 수 있음을 상기 프로세서측에 알려주기 위해 표명된다. 상기 스누프가 액티브되면, 상기 프로세서는 상기 버스로부터의 메모리 주소를 캐시 디렉토리측으로 전송하며, 이에따라 룩업테이블이 발생된다. 하지만, Pentium Pro™ 프로세서 버스는 EADS# 신호, 또는 그 등가 신호를 포함하고 있지 않으므로, 본 발명은 다음의 기술을 구현한다.

파이프라인 버스(101)는 일반적으로 멀티 에이전트 시스템 버스이므로, 상기 버스상의 ADS# 신호는 상기 복수의 에이전트중 한 에이전트에 의한 요구를 지시해 준다. 상기 버스에 접속되어 있는 모든 에이전트는 이들 요구를 간단히 관측할 수 있으며, 그리고 상기 요구 종류에 따라 버스(101)의 스누프에 적절한지 그리고 정보로 무엇을 처리해야하는 지를 결정한다.

도 8에 도시된 바와같이, IRC(30)는 시스템 버스(101)로부터 요구 신호 REQ#[4: 0]를 수신하는 입중계 요구 변환 논리 블록(31)을 포함하고 있다. 논리 블록(31)은 또한 버스(101)로부터 ADS# 신호를 수신한다. 입중계 요구 변환 논리 블록(31)은 버스(101)상의 요구가 프로세서(11)에 의해 스누핑가능한 지를 결정하는 동작을 행한다. 시스템 버스(101)상에서의 현재 트랜잭션이 스누핑가능하면, 논리 블록(31)은 주소 정보와 더불어 상기 프로세서측으로 EADS#가 구동되도록 해 준다. IRC 논리 블록(31)은 EADS# 신호와 프로세서(11)의 주소 핀의 표명/표명 해제하기 위해 3상태 버퍼(33,34)를 제어한다.

입중계 요구 변환 논리 블록(31)은 또한 상기 캐시 라인을 유효 상태로 유지해 두거나 스누프 히트의 경우에는 무효 상태로 만들기 위해 직접 프로세서(11)측으로 무효(INV) 신호를 발생한다. 상기 캐시 라인이 유효 데이타를 포함하고 있는 경우에는, 상기 라인은 공유 캐시 코히어런트 상태로 된다. 논리 블록(31)은 주소 홀드(AHOLD) 신호가 표명되었는지를 확인해야 하며, 이에따라 프로세서(11)는 상기 주소 버스를 적절히 스누핑할 수 있음에 주의하자. 이에따라 캐시 일관성의 유지가 보장된다.

이때, 도 8에 도시된 IRC(31)의 구현은 시스템 버스(101)가 신호 변환을 위해 파이프라인 해제된 것으로 추정하고 있음은 물론이다. 본 발명의 다른 실시예에서는, 버스(101)를 파이프라인 해제하지 않는 것이 바람직할 수 있다. 이러한 구현예에서, 버스 변환 장치(15)는 프로세서(11)측으로 모든 스누프를 전송

하고 시스템 버스(101)를 주기적으로 정지시키는 큐잉(queuing) 메카니즘을 더 포함하고 있을 수 있다.

또한, IRC(31)는 시스템 버스(101)상의 36 비트 요구가 상기 프로세서 인터페이스측으로 전송되지 않도록 해 주는 스누프 제어 논리 회로를 포함하고 있다. 예컨대, Pentium® 프로세서는 32비트 요구에 한정되어 있다. 그러므로, IRC 논리 블록(31)은 32비트보다 큰 비트의 요구가 프로세서(11)에 의해 스누핑되지 않음을 보장해 준다.

도 9는 도 3의 버스 변환 장치(15)에 도시된 캐시 코히어런시 제어 유닛(CCC)의 상세도이다. 도 9의 실시예는 3 상태 버퍼(43)를 통해 데이타 버스를 구동하는 더티 데이타 버퍼(42)에 접속되어 있는 스누프 결과 변환 상태 머신(41)을 구비하고 있는 CCC(40)를 보여 준다. 일단, IRC(31)는 스누프가 프로세서(11)측으로 구동됨을 알리는 신호를 상태 머신(41)측으로 전송해 주면, 몇 클럭 후에, 상기 프로세서는 캐시 히트, 더티 라인에 대한 캐시 히트, 또는 캐시 미스가 발생되었음을 지시해 주는 신호를 CCC(40)측에 제공한다. 이 정보는 신호 라인(HIT/HITM,ADS)을 통해 공급된다.

또한, 스누프 결과 변환 상태 머신(41)은 버스(101)와 관련된 입력 순서 큐(IOQ)와 통신한다. 상기 IOQ는 상기 시스템 버스를 감시하고 적절한 신호를 상기 상태 머신과 3 상태 버퍼(43)측으로 전송하며, 이에 따라 데이타는 버스(101)의 신호 처리 프로토콜에 따라 버스(101)측으로 전송될 수 있다. 일실시예에서, IOQ는 통상적인 버스 상태 트래킹 논리 회로를 구비하고 있다. 이 버스 상태 트래킹 논리 회로는 시스템 버스(101)를 통해 정확한 시간에 상태 정보를 간단히 제공해 준다.

일실시예에서, IOQ는 복수의 엔트리를 포함하고 있고, 이때 각각의 엔트리는 상기 파이프라인 버스상의 현재 상태로 변환된다. 각각의 IOQ 엔트리는 또한 상기 트랜잭션의 상태를 트래킹한다. 이 방법으로, 모든 버스 에이전트는 상기 버스상의 파이프라인을 통해 흐름에 따라 각각의 트랜잭션의 트랙을 유지한다. 현재 설명하는 실시예의 경우에, 상기 IOQ는 "1"의 깊이를 가지고 있는데, 이는 상기 버스가 프로세서(11)로의 변환을 위해 파이프라인해제되기 때문이다.

또한, 상태 머신(41)은 스누프 사건으로부터 시스템 버스(101)의 버스 트래킹 논리 회로측으로 다시 일부 정보를 통신한다. 클린 라인으로의 히트, 또는 캐시 미스의 경우에, 상태 머신(41)은 적절한 시간에 이 상태를 시스템 버스(101)측으로 간단히 보고해 준다. 하지만, 히트가 더티 라인측으로 발생되면(즉, 프로세서가 HITM# 신호를 표명하면), 이 상황은 특정 처리를 필요로 하는데, 이는 프로세서(11)가 직접 출력 더티 데이타로 설계되기 때문이다.

ADS# 신호가 스누프 히트 후에 상태 머신(41)에 의해 검출되면, 이는 더티 데이타가 프로세서(11)에 의해 바로 전송되게 됨을 의미한다. 시스템 버스(101)의 신호 처리 프로토콜은 데이타가 이 방법으로 덤프(dump)되도록 허용하지 않기 때문에, 더티 데이타 버퍼(42)는 프로세서(11)로부터 출력될 때 상기 데이타를 캡쳐하며, 이에따라 상기 데이타는 적절한 시간에 버스(101)측으로 전송될 수 있다. 이 프로세스는 물론 스누프 결과 변환 상태 머신(41)의 제어하에 있다. 상기 IOQ의 버스 상태 트래킹 논리 회로는 또한 도 9의 구현예에서 더티 데이타 버퍼(42)에 접속되어 있는데, 이는 트랜잭션을 완료하기 위해 더티 데이타가 시스템 버스(101)상으로 전송될 수 있음을 지시해 주는 정보를 상기 논리 회로가 가지고 있기 때문이다.

또한, 스누프 결과 변환 상태 머신(41)은 프로세서(11)에 의해 출력중인 데이타를 조절하는데 상기 BRDY# 신호를 사용할 수 있다. 이 다른 구현예에서, 버퍼(42)는 제거될 수 있으며, 그리고 프로세서(11)로부터의 데이타 버스 라인은 시스템 버스(101)에 접속되기 전에 3상태 디바이스(43)에 의해 간단히 3 상태로 될 수 있다. 이 실시예에서, BRDY#는 현재 주소 지정된 디바이스가 기록에 응답하여 Pentium® 프로세서로부터 데이타를 수신하였음을 지시해 주는데 사용됨에 주의하자. 환언하면, 스누프 결과 변환 상태 머신(41)은 적절한 핸드세이킹 신호를 제공해 줌으로써 데이타가 시스템 버스(101)측으로 바로 전송될 수 있도록 동작을 행한다.

반복을 위해, 논리 블록(41)은 프로세서(11)로부터의 스누프 결과를 입력받아 이를 상기 IOQ측으로 전달하며, 이때 이 IOQ는 버스 인터페이스(101)에 포함되어 있다. HITM# 신호가 프로세서(11)에 의해 공급되는 경우에, 스누프 결과 변환 상태 머신(41)은 다음 ADS# 신호가 프로세서(11)에 의해 출력될 때까지 대기하는데, 이는 이로부터 더티 데이타 덤프의 시작을 알 수 있기 때문이다.

도 10에는 통상적인 상태 머신과 조합 논리 회로를 구비하고 있는 버스 요구 프로토콜 변환 논리 유닛(70)이 예시되어 있다. 도시된 바와같이, 논리 유닛(70)은 버스 프로토콜 변환 유한 상태 머신(72)과 순차 큐(71)를 포함하고 있다. 상태 머신(71)은 프로세서(11)와 시스템 버스(101)사이에서 필요한 프로토콜 신호 변환을 수행한다. 순차 큐(71)는 상기 인터페이스의 대향측에서, 두 버스의 상태의 트랙을 유지하는데 이용된다. IOQ(71)는 시스템 버스(101)의 프로토콜에 따라 특정된 바와같이, 버스 상태 정보를 홀딩하기 위한 통상적인 논리 회로와 레지스터 기억 장치를 포함하고 있다.

상기 버스 인터페이스의 양측에 보인 신호의 상태를 트래킹하는 외에, 논리 유닛(70)은 또한 필요한 핸드세이킹 신호, 예컨대 데이타 전송 핸드세이킹을 제공한다. (특정예에서, 상기 인터페이스의 대향측의 버스 신호는 밀접하게 관련된 기능을 가지고 있음에 주의하자. 예컨대, 프로세서(11)에 접속되어 있는 상기 ADS# 신호는 단일 방향 주소 스트로브 신호이다. 대응하는 변환은 실질적으로 쌍방향 버스인 시스템 버스(101)의 상기 ADS# 신호에 대해 행해진다.)

알 수 있는 바와같이, 버스 트래킹 논리 유닛(70)은 시스템 버스(101)와 프로세서(11) 사이에서 전송되는 각종 신호를 변환한다. 이 그룹내에는 본 발명의 일실시예에 따른 주소 패리티 오류 신호(AERR#)가 포함되어 있다. 패리티 오류를 검출하는 에이전트는 상기 트랜잭션의 오류 단계 동안에 상기 AERR# 신호를 공급한다. 모든 버스 에이전트는 상기 AERR#를 관측하고, 그리고 상기 순차 큐로부터 상기 트랜잭션을 제거하고 상기 트랜잭션과 관련된 모든 나머지 단계를 취소함으로써 다음 클럭상에서 불량 트랜잭션을 억제한다. 상기 인터페이스의 프로세서측의 상기 대응하는 주소 패리티 신호는 APCHK#이다. 주소 패리티 오류의 경우에, 프로세서(11)는 APCHK#를 액티브시킨다.

또한, 논리 유닛(70)의 인터페이스의 상기 시스템 버스 측에는 신호(HIT#/HITM#)가 포함되어 있다. HIT#

와 HITM#은 상기 스누핑 에이전트에서 라인이 유효 또는 무효인지, 상기 라인이 캐싱 에이전트에서 수정(더티) 상태인지, 또는 상기 스누프 단계가 실행될 필요가 있는지를 지시해 주는데 사용된다. 상기 HIT# 신호와 HITM# 신호는 시스템 레벨로 캐시 코히어런시를 유지하는데 사용된다. 이전에 설명한 바와같이, 스누핑 에이전트가 HITM#을 표명하면, 상기 에이전트는 상기 데이타 단계(암시적인 재기록) 동안에 상기 수정 라인을 재기록하게 됨을 추정한다. 상기 DEFER# 신호는 또한 상기 스누프 단계에서 구동된다. 상기 DEFER#은 상기 트랜잭션이 순차 완료 보장될 수 있음을 지시해 주기 위해 표명 해제된다. DEFER#을 표명하는 에이전트는 적절한 응답을 발생함으로써 상기 IOQ(71)로부터의 상기 트랜잭션의 적절한 제거를 보장해 준다. 트랜잭션의 응답 신호 그룹은 동일한 트랜잭션의 스누프 단계 후에 발생되며, 그리고 상기 필요한 스누프 결과를 설명하는 엔코딩을 제공하는 신호(RS#[2:0])를 포함하고 있다. 상기 응답 에이전트는 상기 IOQ(71)의 상부에서 상기 트랜잭션을 완료하는 에이전트이다. 기록 트랜잭션의 경우에, TRDY#는 기록 데이타 또는 재기록 데이타를 입력받을 준비가 되어 있음을 지시해 주기 위해 상기 응답 에이전트에 의해 표명된다.

상기 데이타 전송 신호 그룹은 데이타 단계에서 구동되는 신호들을 포함하고 있고, 그리고 DBSY#/DRDY#(데이타 버스 비지 및 데이타 준비)를 포함하고 있다. DRDY#는 유효 데이타가 상기 버스상에 존재하고 그리고 래치되어야 함을 지시해 준다. 상기 데이타 버스 소유자는 유효 데이타가 전송되어야 하는 각각의 클럭 동안에 DRDY#를 공급한다. DBSY#는 다중 클럭 데이타 전송을 위해 제1DRDY# 전에 그리고 DRDY# 공급들 사이에 상기 버스를 홀딩하는데 사용된다. 상기 스누프, 응답, 및 데이타 단계 신호는 모두 IOQ(71)를 통해 트래킹된다.

버스 프로토콜 변환 상태 머신(72)은 다음 주소 입력(NA#)을 포함하고 있는 프로세서(11)측에 출력을 제공한다. 상기 NA# 입력은 현재 사이클 종료 전에 상기 버스상으로 다음 버스 사이클이 전송되도록 이를 요구하는 디바이스에 의해 공급된다. 외부 기록 버퍼 엠프티(empty)(EWBE#) 신호는, 메모리 동작이 실행 순서(즉, 스트롱 기억 순서)로 발생되도록 이를 보장해 주기 위해 프로세서(11)측에 제공된다. 이에따라, 모든 버퍼 기록은 다음 명령을 실행함으로써 완료될 수 있다. 이전에 설명한 바와같이, 상기 BRDY# 신호는 현재 주소 지정된 디바이스가 판독에 응답하여 상기 데이타 버스 핀상에 유효 데이타를 제공하였거나, 상기 현재 주소 지정된 디바이스가 기록에 응답하여 상기 프로세서로부터 데이타를 입력받았음을 지시해 준다. 이와 유사하게, KEN#은 판독중인 위치가 캐시가능 주소 공간내에 존재하는지를 지시해 주는 캐시 제어 신호이다. 주소가 캐시가능하지 않으면, 상기 KEN# 라인은 캐시 라인 필이 수행되지 않도록 이를 상기 프로세서측에 알려 주기 위해 인액티브로 구동된다.

발명의 효과

본 발명은 표준화 컴퓨터 버스 인터페이스를 통해 호스트에 접속되어 있는 프로세서 서브시스템(또는 모듈)을 포함하고 있는 범용 컴퓨터 구조를 제공함으로써 종래기술에 내재한 문제를 해결해 준다. 본 발명은 시스템 기본 구조를 수정할 필요가 없도록 해 주면서 다양한 프로세서 종류와의 호환성을 제공한다.

(57) 청구의 범위

청구항 1

멀티프로세서 컴퓨터 시스템에 있어서,

호스트 프로세서;

상기 호스트 프로세서와 호환성이 있는 표준 신호 처리 프로토콜에 따라 동작하는 시스템 버스;

상기 시스템 버스에 접속되고, 시스템 버스상에서 대칭 에이전트(들) 또는 우선 순위 에이전트(들)로 분류되고 있는 하나 이상의 에이전트;

상기 시스템 버스에 접속되어 있고, 시스템 버스 상에서 대칭 에이전트로서 기능을 하는 카드; 를 포함하며,

상기 카드는

표준 신호 처리 프로토콜과는 상이한 제2신호 처리 프로토콜에 따라 동작하는 추가 프로세서,

정보가 상기 추가 프로세서와 상기 호스트 프로세서 사이에서 전송될 수 있도록, 상기 추가 프로세서에 그리고 상기 시스템 버스에 접속되어, 상기 시스템 버스의 표준 신호 처리 프로토콜을 상기 제2신호 처리 프로토콜로, 그리고 그 역으로 변환하며, 우선 순위 알고리즘에 따라 추가 프로세서를 대신하여 상기 시스템 버스의 소유권을 획득하는 중재 변환 논리 회로를 포함하는 버스 변환 장치, 를 포함하고 있는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 2

제 1 항에 있어서, 상기 카드는

상기 추가 프로세서에 접속되어 있는 캐시 메모리;

메모리 디바이스를 포함하는 하나 이상의 에이전트;를 더 포함하며,

여기에서 정보가 상기 버스 변환 장치를 통해 상기 추가 프로세서와 상기 메모리 디바이스 사이에서 전송되는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 3

제 2 항에 있어서, 상기 우선 순위 알고리즘은 순환형 우선 순위 알고리즘이고, 상기 대칭 에이전트들은 순환형 우선 순위 알고리즘에 따라 분산 중재를 지원하는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스

템.

청구항 4

제 3 항에 있어서, 상기 버스 변환 장치는 상기 추가 프로세서로부터의 각 요구를 상기 시스템 버스상의 제1 및 제2요구 사이클로 변환하는 출중계 요구 변환기를 더 포함하고 있는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 5

제 4 항에 있어서, 상기 버스 변환 장치는 상기 추가 프로세서에 의해 스누핑가능한 상기 시스템 버스상의 트랜잭션을 식별하는 입중계 요구 변환기를 더 포함하며, 상기 트랜잭션이 상기 입중계 요구 변환기에 의해 상기 추가 프로세서측으로 입력되는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 6

제 5 항에 있어서, 상기 버스 변환 장치는 캐시 히트, 수정 라인에 대한 캐시 히트, 또는 캐시 미스를 지시해 주는 신호를 상기 추가 프로세서로부터 수신하는 캐시 코히어런시 제어 유닛을 더 포함하며, 상기 캐시 코히어런시 제어유닛은 상기 추가 프로세서에 의해 상기 시스템 버스측으로 출력된 상기 수정 라인으로부터의 더티 데이타를 조절하는 준비 신호를 표명/표명 해제함으로써 상기 수정 라인에 대한 캐시 히트에 응답하는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 7

제 5 항에 있어서, 상기 버스 변환 장치는 캐시 히트, 수정 라인에 대한 캐시 히트, 또는 캐시 미스를 지시해 주는 신호를 상기 추가 프로세서로부터 수신하는 캐시 코히어런시 제어 유닛을 더 포함하며, 상기 캐시 코히어런시 제어유닛은 상기 추가 프로세서로부터 출력된 상기 수정 라인에 대한 캐시 히트에 응답하여 더티 데이타를 저장하는 버퍼를 포함하고, 다음에 상기 더티 데이타는 상기 표준 신호 처리 프로토콜에 따라 상기 캐시 코히어런시 제어 유닛에 의해 상기 버스측으로 전송되는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 8

제 1 항에 있어서, 상기 버스 변환 장치는 단위 트랜잭션과 관련하여 상기 추가 프로세서에 의해 발생된 제1로크 신호를 상기 표준 신호 처리 프로토콜에 따라 상기 시스템 버스상에 제공된 제2로크 신호로 변환하는 버스 로크 변환기를 더 포함하고 있는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 9

제 1 항에 있어서, 상기 시스템 버스는 파이프라인 버스를 구비하고 있는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 10

멀티프로세서 컴퓨터 시스템에 있어서,

제1신호 처리 프로토콜에 따라 동작하는 호스트 프로세서;

상기 제1신호 처리 프로토콜과 호환성이 있는 버스 구조를 가지고 있는 파이프라인 시스템 버스;

중재 알고리즘에 따라 시스템 버스의 소유권을 중재하는 복수의 에이전트;

복수의 단자 접속부를 가지고 있는 슬롯; 및

슬롯에 플러그 접속되고, 상기 시스템 버스상에서 대칭 에이전트로서 기능을 하며, 복수의 반도체 디바이스를 수용하고 있고 그리고 상기 단자 접속부에 대응하는 복수의 인터페이스 단자를 가지고 있는 카드; 를 포함하며,

상기 복수의 반도체 디바이스는,

상기 시스템 버스의 제1신호 처리 프로토콜과는 다른 제2신호 처리 프로토콜에 따라 동작하는, 복수의 핀을 가지고 있는 추가 프로세서, 및

상기 파이프라인 시스템 버스의 제1신호 처리 프로토콜을 상기 프로세서의 제2신호 처리 프로토콜로, 그리고 그 역으로 변환하는 버스 변환 장치로서, 상기 시스템 버스의 제1신호 처리 프로토콜과 호환성이 있는 인터페이스 단자에 접속되어 있는 제1인터페이스, 및 상기 프로세서의 제2신호 처리 프로토콜과 호환성이 있는 상기 추가 프로세서의 핀에 접속되어 있는 제2인터페이스를 가지며, 중재 알고리즘에 따라 추가 프로세서를 대신하여 시스템 버스상의 소유권을 획득하는 중재 변환 논리회로를 포함하는 버스 변환 장치, 를 포함하고 있는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 11

제 10 항에 있어서, 상기 중재 알고리즘은 라운드 로빈 알고리즘을 포함하고 있는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 12

제 10 항에 있어서, 상기 시스템 버스상에 우선 순위 요구 신호의 표명시에 상기 대칭 에이전트중 한 에이전트로부터 상기 시스템 버스의 소유권을 바로 획득할 수 있는 하나 이상의 우선 순위 에이전트를 더 포함하고 있는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 13

제 12 항에 있어서, 상기 카드는 상기 시스템 버스상에 우선 순위 요구 신호의 표명시에 상기 대칭 에이전트중 한 대칭 에이전트로부터 상기 시스템 버스의 소유권을 바로 획득할 수 있는 우선 순위 에이전트로서 기능하는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 14

제 10 항에 있어서, 각각의 대칭 에이전트는 특정 에이전트 식별(ID), 및 어느 대칭 에이전트가 다음 중재 사건에 대해 최하위의 우선 순위를 가지고 있는지를 반영하는 회전 ID 값을 가지고 있는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 15

제 14 항에 있어서, 상기 버스 변환 장치는 상기 제1신호 처리 프로토콜에 따라 상기 추가 프로세서의 버스 요구 신호를 상기 시스템 버스의 버스 요구 신호로 변환하는 버스 요구 논리 회로;

상기 버스 요구 논리 회로에 접속되어, 상기 시스템 버스상의 RESET에 응답하여 상기 버스 변환 장치의 에이전트 ID를 결정하는 에이전트 ID 상태 머신;

상기 에이전트 ID 상태 머신과 상기 버스 요구 논리 회로에 접속되어, 상기 시스템 버스의 현재 상태를 감시하고, 상기 추가 프로세서가 상기 시스템 버스의 소유권을 취득한 때를 결정하는 대칭 소유자 상태 머신;

상기 대칭 소유자 상태 머신에 접속되어, 홀드 신호 입력을 상기 추가 프로세서측에 발생해 주는 홀드 표명 논리 회로로서, 상기 홀드 신호가 상기 시스템 버스의 소유권을 포기할 것을 상기 추가 프로세서측에 요구하는 홀드 표명 논리 회로를 더 포함하고 있는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 16

제 15 항에 있어서, 상기 버스 변환 장치는 상기 추가 프로세서로부터의 요구를 상기 시스템 버스상의 제1 및 제2요구 사이클로 변환하는 출중계 요구 변환기; 및

상기 추가 프로세서에 의해 스누핑가능한 시스템 버스상의 트랜잭션을 식별하는 입중계 요구 변환기로서, 상기 트랜잭션이 상기 입력 요구 변환기에 의해 상기 추가 프로세서측으로 입력되는 입중계 요구 변환기를 더 구비하고 있는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 17

제 16 항에 있어서, 상기 버스 변환 장치는 수정 라인에 대한 캐시 적중을 지시해 주는 신호를 상기 추가 프로세서로부터 수신하는 캐시 코히어런시 제어 유닛을 더 포함하며, 상기 캐시 코히어런시 제어유닛은 상기 추가 프로세서에 의해 상기 시스템 버스측으로 출력된 상기 수정 라인으로부터의 더티 데이타를 조절함으로써 상기 신호에 응답하는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 18

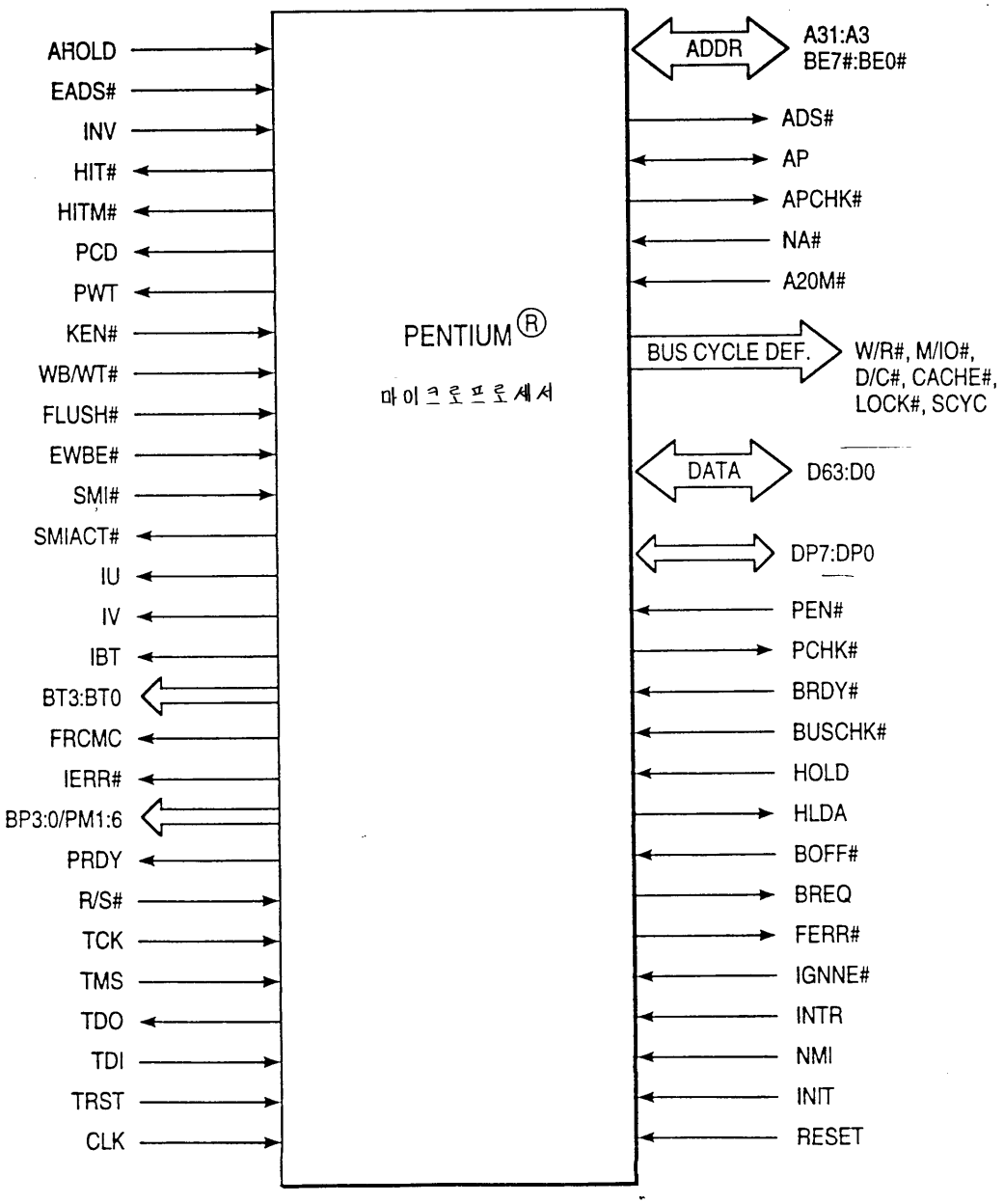
제 16 항에 있어서, 상기 버스 변환 장치는 수정 라인에 대한 캐시 히트를 지시해 주는 신호를 상기 추가 프로세서로부터 수신하는 캐시 코히어런시 제어 유닛을 더 포함하며, 상기 캐시 코히어런시 제어유닛은 더티 데이타를 저장하는 버퍼를 또한 포함하고 있으며, 상기 버퍼에 저장되어 있는 더티 데이타는 상기 추가 프로세서에 의해 상기 수정 라인으로부터 출력되어, 상기 제1신호 처리 프로토콜에 따라 상기 캐시 코히어런시 제어 유닛에 의해 상기 시스템 버스측으로 전송되는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

청구항 19

제 16 항에 있어서, 상기 버스 변환 장치는 단위 트랜잭션과 관련하여 상기 추가 프로세서에 의해 발생된 제1로크 신호를 상기 제1신호 처리 프로토콜에 따라 상기 시스템 버스상에 제공된 제2로크 신호로 변환하는 버스 로크 변환기를 더 구비하고 있는 것을 특징으로 하는 멀티프로세서 컴퓨터 시스템.

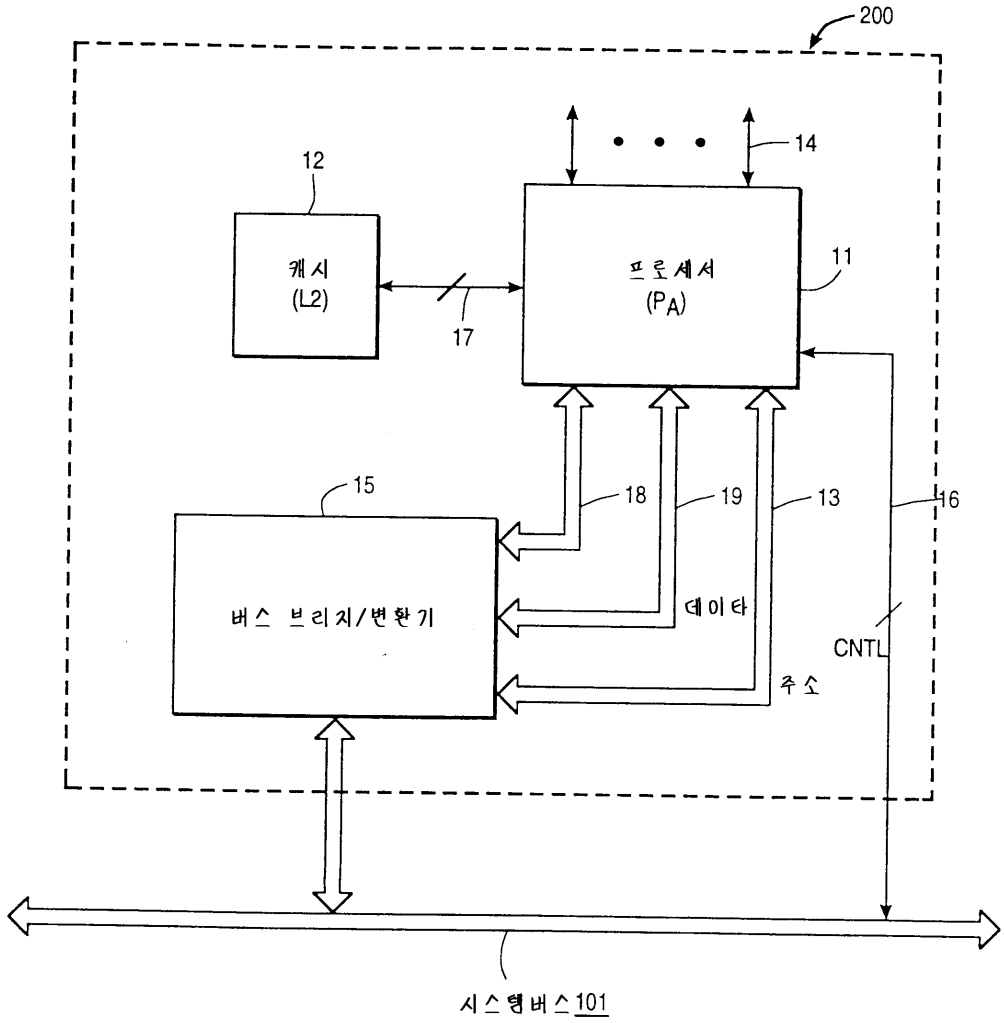
도면

도면1

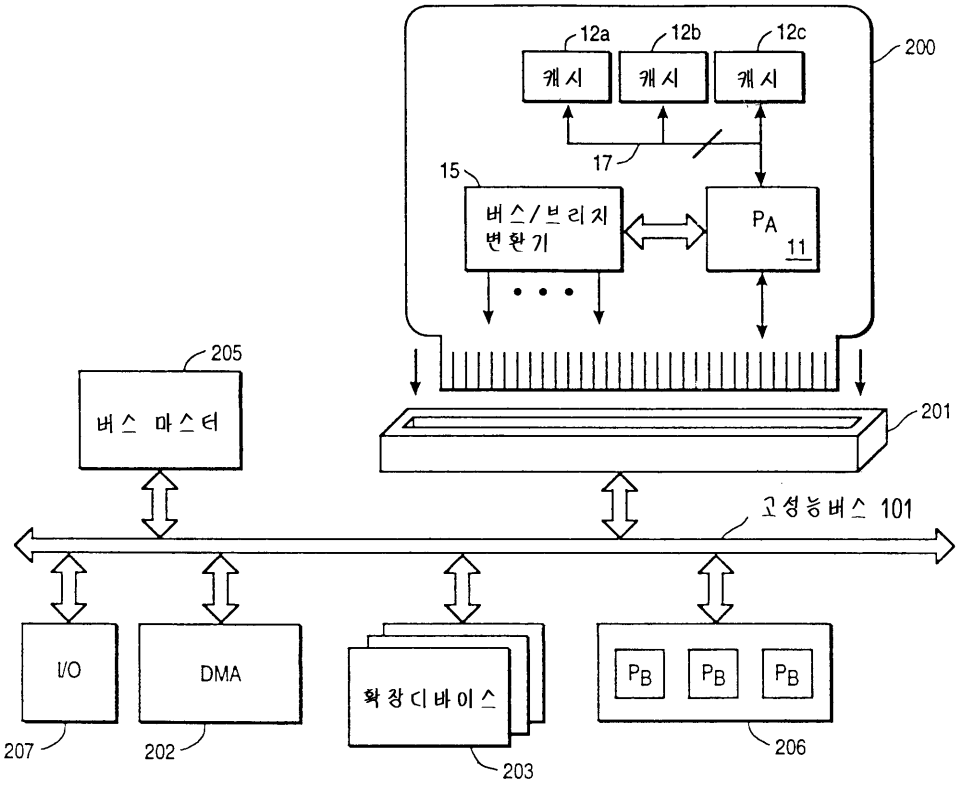

AHOLD
EADS#
INV
HIT#
HITM#
PCD
PWT
KEN#
WB/WT#
FLUSH#
EWBE#
SMI#
SMIACT#
IU
IV
IBT
BT3:BT0
FRCMC
IERR#
BP3:0/PM1:6
PRDY
R/S#
TCK
TMS
TDO
TDI
TRST
CLK
PENTIUM®
마이크로프로세서
ADDR
A31:A3
BE7#:BE0#
ADS#
AP
APCHK#
NA#
A20M#
BUS CYCLE DEF.
W/R#, M/IO#,
D/C#, CACHE#,
LOCK#, SCYC
DATA
D63:D0
DP7:DP0
PEN#
PCHK#
BRDY#
BUSCHK#
HOLD
HLDA
BOFF#
BREQ
FERR#
IGNNE#
INTR
NMI
INIT
RESET

(종래기술)

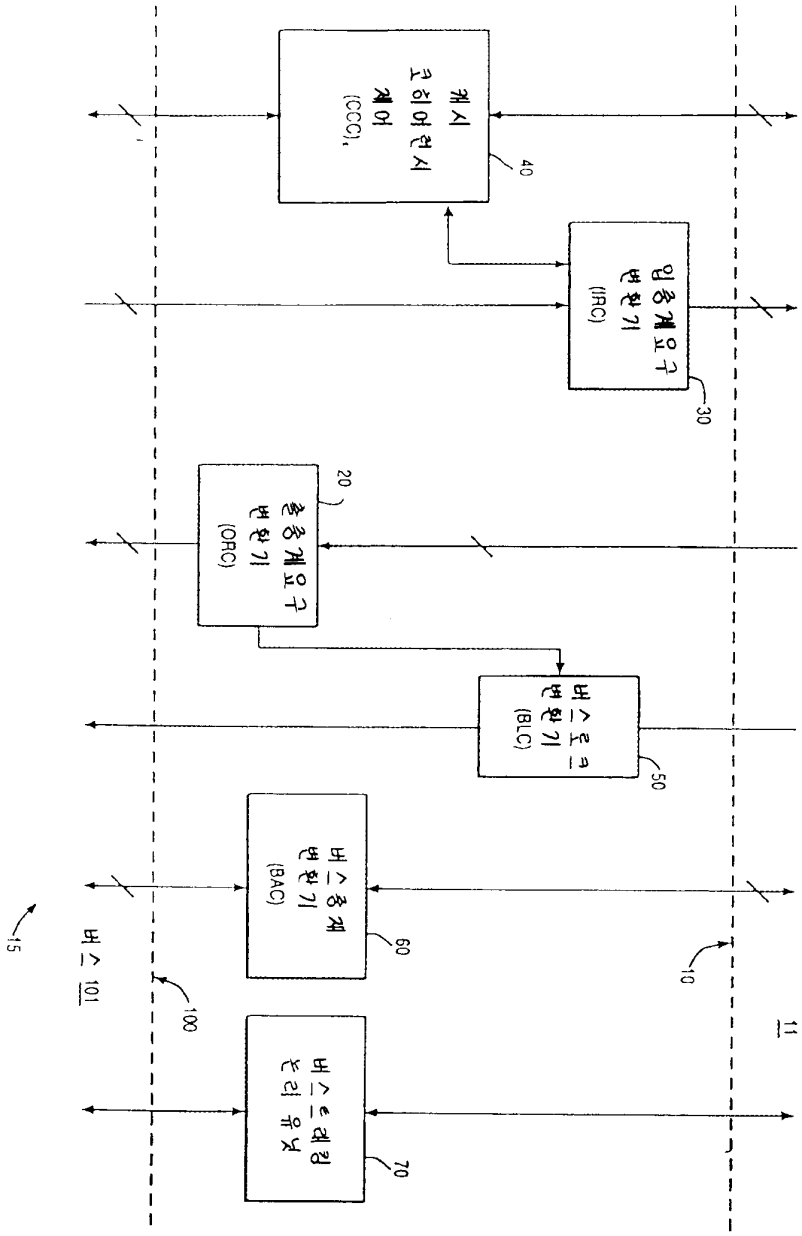
도면2


200
12
캐시
(L2)
17
14
프로세서
(PA)
11
15
18
19
13
16
버스 브리지/변환기
데이타
CNTL
주소
시스템버스 101

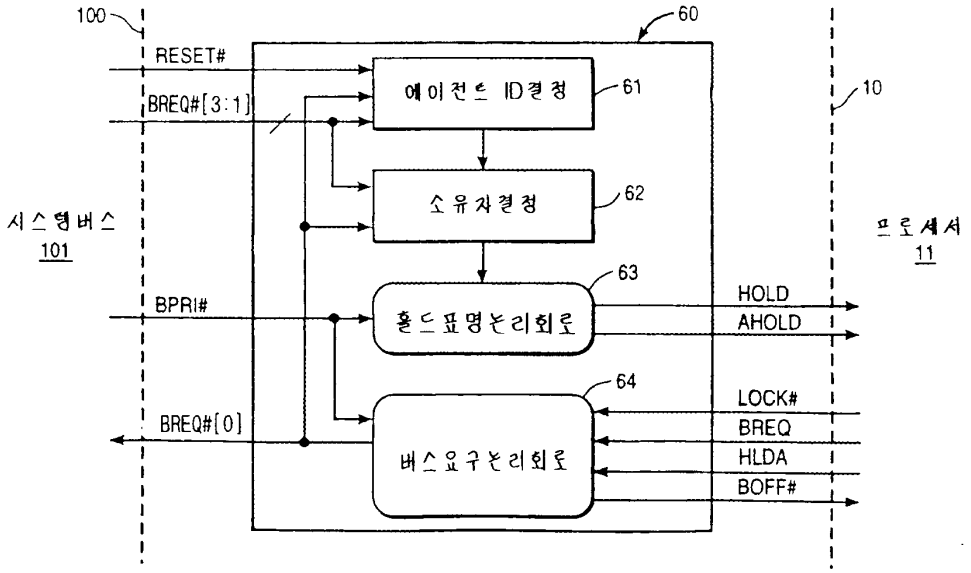
도면3


12a
12b
12c
200
캐시
캐시
캐시
17
15
버스/브리지
변환기
PA
11
205
버스 마스터
201
고성능버스 101
I/O
DMA
확장디바이스
PB
PB
PB
207
202
203
206

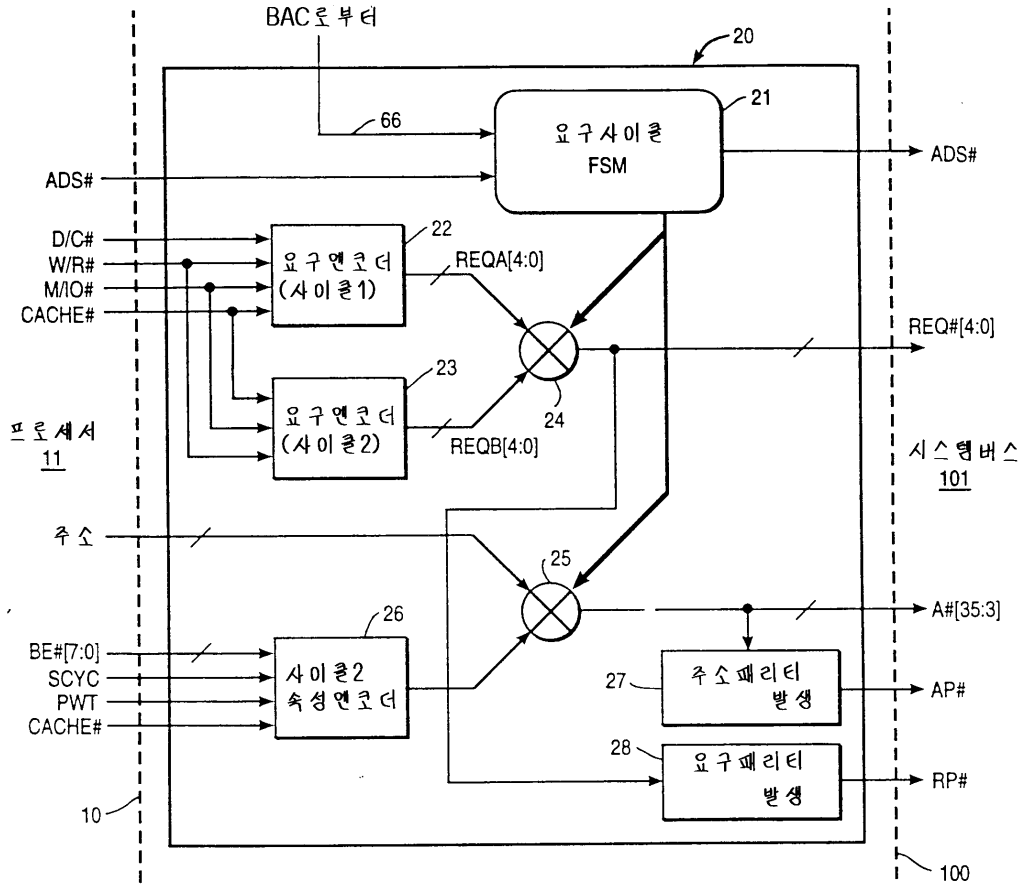
도면4


캐시 코히어런시 제어 (CCC)
40
입출계요구 변환기 (IRC)
30
출출계요구 변환기 (ORC)
20
버스 로크 변환기 (BLC)
50
버스 응계 변환기 (BAC)
60
버스 트래킹 논리 유닛
70
버스 101
100
10
11
15

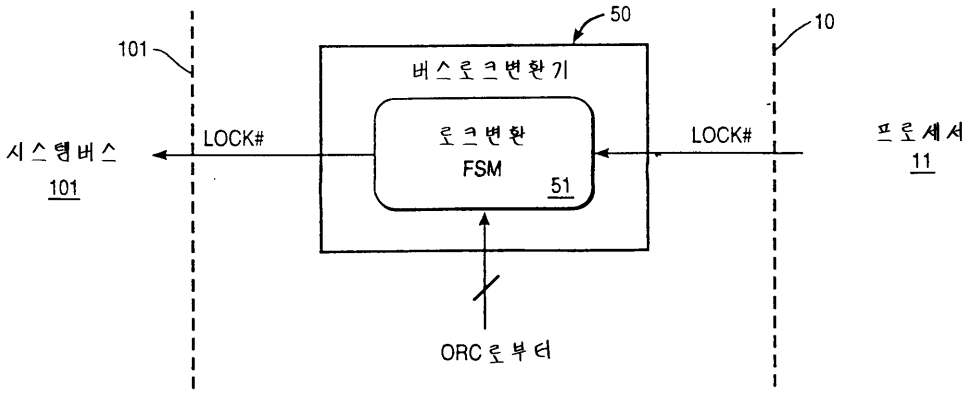
도면5


100
60
RESET#
BREQ#[3:1]
에이전트 ID결정
61
10
소유자결정
62
시스템버스
101
프로세서
11
63
BPRI#
홀드표명논리회로
HOLD
AHOLD
64
LOCK#
BREQ
BREQ#[0]
버스요구논리회로
HLDA
BOFF#

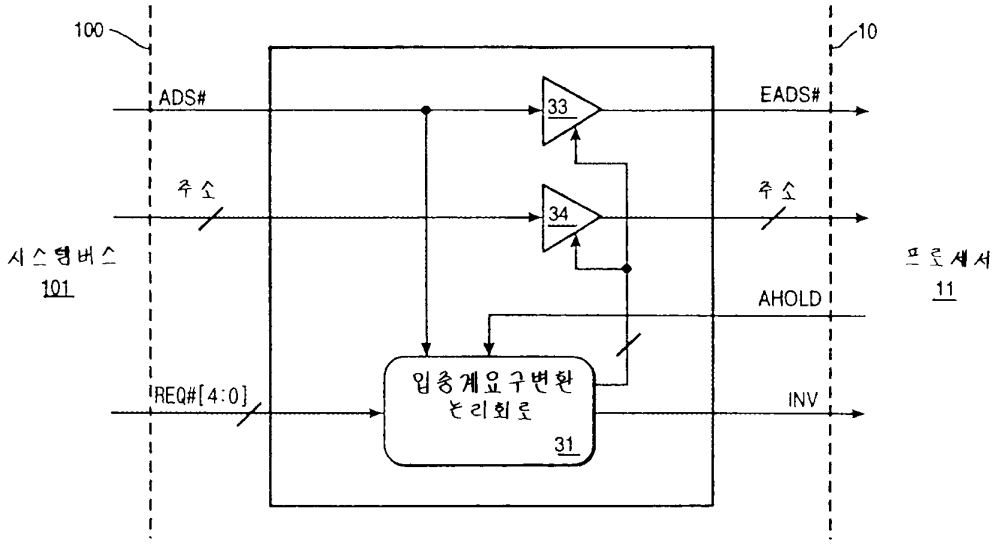
도면6


BAC로부터
20
66
요구사이클
FSM
21
ADS#
ADS#
D/C#
W/R#
M/IO#
CACHE#
22
요구엔코더
(사이클1)
REQA[4:0]
REQ#[4:0]
23
요구엔코더
(사이클2)
24
REQB[4:0]
프로세서
11
시스템버스
101
주소
25
A#[35:3]
26
BE#[7:0]
SCYC
PWT
CACHE#
사이클2
속성엔코더
27
주소패리티
발생
AP#
28
요구패리티
발생
RP#
10
100

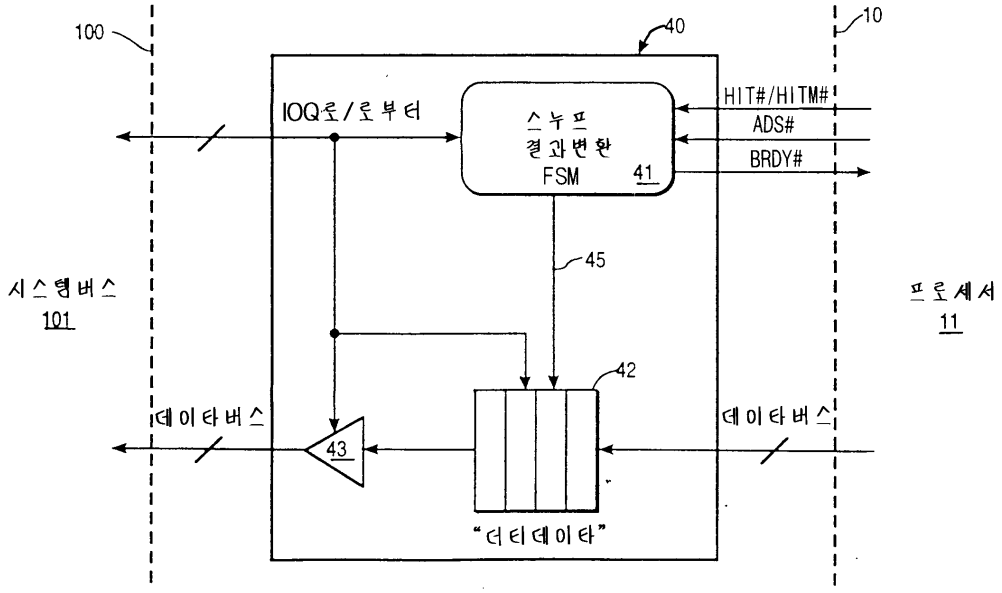
도면7


50
10
101
버스로크변환기
로크변환
FSM
51
LOCK#
LOCK#
시스템버스
101
프로세서
11
ORC로부터

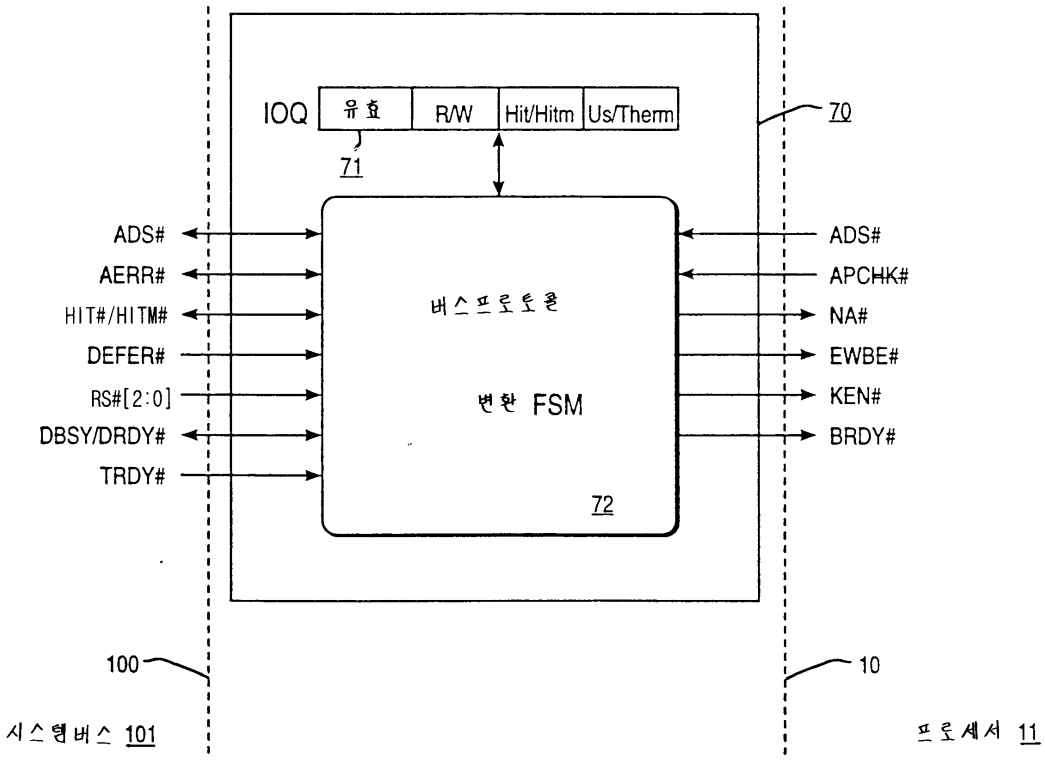
도면8


100
10
ADS#
EADS#
33
주소
34
주소
시스템버스
101
프로세서
11
AHOLD
입증계요구변환
논리회로
31
REQ#[4:0]
INV

도면9


100
40
10
IOQ로/로부터
스누프
결과변환
FSM
41
HIT#/HITM#
ADS#
BRDY#
45
시스템버스
101
프로세서
11
42
데이타버스
43
데이타버스
"더티데이타"

도면10


IOQ
유효
R/W
Hit/Hitm
Us/Therm
71
70
ADS#
AERR#
HIT#/HITM#
DEFER#
RS#[2:0]
DBSY/DRDY#
TRDY#
버스프로토콜
변환 FSM
72
ADS#
APCHK#
NA#
EWBE#
KEN#
BRDY#
100
10
시스템버스 101
프로세서 11