(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl. [6]
G06F 13/38

(45) 공고일자 2002년11월30일
(11) 등록번호 10-0347473
(24) 등록일자 2002년07월23일

(21) 출원번호 10-1995-0003486
(22) 출원일자 1995년02월23일

(65) 공개번호 특1995-0033878
(43) 공개일자 1995년12월26일

(30) 우선권주장 08/201,186 1994년02월24일 미국(US)

(73) 특허권자 휴렛-팩커드 컴퍼니(델라웨어주법인)
미합중국 캘리포니아주 (우편번호 94304) 팔로 알토 하노버 스트리트 3000

(72) 발명자 마이클엘.지글러
미합중국메사추세츠01588화이틴스빌아이비레인166
로버트제이.브룩스
미합중국캘리포니아95747로즈빌리버티레인2789
윌리암알.브리그
미합중국캘리포니아95070사라토가페레고웨이18630
케네스케이.찬
미합중국캘리포니아95133-1461산조세폭스리지웨이828
토마스알.하치키스
미합중국메사추세츠01450그로톤타운센드로드907
로버트이.나스
미합중국콜로라도80525포트콜린스비버크리크드라이브4313
로버트디.오디널
미합중국캘리포니아95661로즈빌타마락코트1001
브렌단에이.보지
미합중국캘리포니아95746그레니트베이이스트그레니트드라이브8290
제임스비.윌리암스
미합중국메사추세츠01852로웰할리루드애비뉴74
존엘.우드
미합중국뉴햄프셔03867로체스터필드스톤빌리지84

(74) 대리인 김창세
장성구

심사관 : 김재욱

(54) 버스시스템

요약

버스 시스템은 버스 중재 방안을 갖는다. 버스 시스템은 버스와 이 버스에 연결된 다수의 클라이언트 모듈을 포함한다. 클라이언트 모듈의 각각은 버스를 통하여 정보를 다른 클라이언트 모듈로 전송할 수 있고, 단지 하나의 클라이언트 모듈만이 소정시간에 버스를 통하여 정보를 전송할 수 있다. 버스를 통하여 정보를 전송할 수 있는 모듈은 사이클을 정의하는 최소 시간기간동안 버스를 제어한다. 버스를 사용할 수 있는 모듈을 결정하기 위하여, 각 클라이언트 모듈은 중재 신호에 응답하는 중재 신호 프로세서를 구비하여, 모듈이 상기 버스를 통하여 정보를 전송할 자격이 있는지의 여부를 판정한다. 또한, 시스템은 바람직하게, 소정 사이클에서 버스를 통하여 허용되는 트랜잭션의 유형을 클라이언트 모듈로 알려주는 호스트 모듈을 포함한다. 또한, 각 중재 신호 프로세서는 바람직하게, 초기 사이클동안 호스트 모듈에 의해 전송되는 클라이언트 선택사양 신호에 응답한다.

대표도

도 1

명세서

도면의 간단한 설명

제1도는 본 발명에 따라 중재 방안을 사용하는 컴퓨터 시스템이 하나의 실시예를 도시하는 블럭도.

도면의 주요 부분에 대한 부호의 설명

10 : 컴퓨터 시스템 14 : 주 메모리 제어기

16, 18 : I/O 모듈 20, 22, 24, 26 : 프로세서 모듈

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 버스(bus)에 관한 것으로서, 보다 구체적으로는 컴퓨터 시스템의 프로세서, 메모리 및 입력/출력 모듈간에 정보를 전송할 수 있는 공용 버스(shared buses)에 대한 중재 방안(arbitration schemes)에 관한 것이다.

컴퓨터 시스템은 통상적으로 프로세서, 메모리 및 입력/출력 장치, 그리고, 두가지 혹은 그이상의 구성요소들간의 정보를 전송하기 위한 공용 버스를 구비한다. 구성요소는 통상적으로, 각각이 하나 혹은 그이상의 프로세서, 메모리 및/ 혹은 입력/출력 장치를 포함할 수 있는 구성요소 모듈(component modules)의 형태로 버스에 연결된다. 정보는 버스 " 사이클" (bus " cycle")동안 구성요소 모듈간에 버스를 통하여 전송되고, 각 버스 사이클은 모듈이 버스의 제어를 가지고, 버스를 통하여 제한된 정보량을 전송 혹은 구동(drive)할 수 있는 시간주기이다.

공용 버스는 통상적으로, 특정 유형의 정보를 전문적으로 전송하기 위한 전기 경로 세트(예를 들면, 인쇄 배선회로상의 트레이스(traces))로 구성된다. 각 전기 경로 세트(혹은 하드웨어 라인)는 하드웨어 라인 세트상에 전송되는 정보를 제공 및 사용하는 선택된 구성요소 모듈에 연결된다. 버스는 통상적으로, 데이터를 전송하기 위한 하나의 하드웨어 라인 세트, 데이터를 식별하는 어드레스(addresses)를 전송하기 위한 세트 및 구성요소 모듈이 충돌(conflict)없이 버스

를 제어하도록 하는 제어 신호를 전송하기 위한 다른 하나의 세트를 포함한다. 동일한 하드웨어 라인 세트는 둘 이상의 목적에 사용될 수 있다. 예를 들면, 데이터 및 어드레스는 동일한 하드웨어 라인 세트를 통해 순차적으로 전송될 수 있다. 또한, 공용 버스내의 전문 하드웨어 라인 세트를 버스라 한다. 예를 들면, 고용 버스는 데이터 버스, 어드레스 버스 및 그외 다른 버스를 포함할 수 있다.

모듈은 " 판독" 및 " 기록" 트랜잭션(transactions)을 완료하기 위해 하나 이상의 사이클을 취하는 " 트랜잭션" 형태로 공용 버스를 통하여 서로 통신한다. 전형적인 판독 트랜잭션에 있어서, 모듈은, 하나 이상의 사이클 동안, 버스를 통하여 다른 모듈로부터 판독할 필요가 있는 데이터를 식별하며 식별된 데이터가 이 모듈로 전송되기를 요청하는 정보를 전송할 것이다. 그후, 응답 모듈(responding module)은 데이터 요청을 처리하고 데이터를 반환한다.

종래의 많은 버스 시스템에 있어서, 판독 트랜잭션의 개시와 요청된 데이터의 반환(return) 사이 중에는 버스가 다른 목적으로 사용될 수 없다. 메모리의 대기 시간(latency time)이 다수의 버스 사이클인 경우, 대기 기간동안에는 버스를 통하여 정보가 전송되지 않는다는 점에서 다수의 버스 사이클이 낭비된다. 이와 달리 " 분할 트랜잭션(split transaction): 버스에 있어서는 요청에 즉시 응답할 필요가 없다. 예를 들면, 모듈이 판독 트랜잭션을 개시한 후, 모듈은 버스의 제어를 포기하고, 응답 모듈이 트랜잭션을 처리하는 동안 버스는 다른 목적으로 사용될 수 있다. 응답 모듈이 요청된 데이터를 반환할 준비가 되었을 때, 그것은 버스의 제어를 획득하여 요청된 데이터를 요청 모듈로 전송한다. 따라서, 분할 트랜잭션 버스는 요청이 처리되는 동안 버스 사이클을 소비할 필요가 없다. 그러나, 분할 트랜잭션 버스는 모든 판독 트랜잭션마다 두 번 중재할 필요가 있다. 한 번은 판독 요청을 전송하기 위하여 버스의 제어를 획득하는 것이고, 다른 한 번은 요청된 데이터를 반환하기 위하여 버스의 제어를 획득하는 것이다.

기록 트랜잭션에서는 데이터가 이미 버스 속도로 판독(read-out)할 수 있는 고속 메모리내에 있으므로, 대기 시간 문제는 감소된다. 전형적인 기록 트랜잭션에 있어서, 모듈은 하나 혹은 그 이상의 사이클 동안 버스를 통하여 사전결정된 정보(predetermined information)를 전송함으로써 " 기록" 을 개시하고, 후속 사이클동안 즉시 데이터를 다른 모듈로 전송한다. 따라서, 전형적으로 기록 트랜잭션에서 낭비되는 사이클은 없으며, 기록 트랜잭션은 분할될 필요가 없다. 그러므로, 분할 트랜잭션 버스는 통상적으로, 두 번째 시간을 중재할 필요가 없도록 하기 위하여, 분할 판독 트랜잭션과 함께 종래의 기록 트랜잭션(즉, 분할되지 않음)을 수용한다.

전형적으로, 단지 하나의 모듈이 소정 사이클에서 공유 버스상에 정보를 전송 혹은 구동할 수 있다. 따라서, 공유 버스 시스템은 특정 사이클에서 버스를 통하여 정보를 구동할 수 있는지를 판정하기 위한 " 중재(arbitration)" 방안을 가져야만 한다. 다수의 종래 버스 중재 방안이 사용 가능하다. 대부분의 중재 방안에 있어서, 공유 버스 시스템의 각 모듈은 버스를 구동하기를 원하는 모듈과 통신하며, 하나 이상의 프로세서에 구현되는 중재 알고리즘은 어느 요청 모듈(requesting module)이 소정 사이클동안 버스를 구동할 자격이 있는지를 결정한다. 고성능 버스 중재를 위한 2가지 기본적인 해결방안은 집중형 중재(centralized arbitration) 및 분산형 중재(distributed arbitration)이다.

집중형 중재 방안에 있어서, 버스를 사용하려고 하는 각 모듈은 중재 신호(arbitration signal)를 중앙 중재기 회로(central arbiter circuit)로 전송한다. 중앙 중재기 회로는 중재 신호를 처리하여 다음 사용 가능한 사이클 동안 버스를 사용할 자격이 있는 모듈(즉, 다음 버스 소유자(bus owner)인지)을 결정한다. 그 다음, 중앙 중재기 회로는 버스르 사용할 자격이 있는지의 여부를 각 모듈에게 알리는 중재 응답 신호를 모듈에 되전송한다. 그 후, 중재를 " 획득(win)" 한 모듈은 버스를 통하여 정보를 구동한다. 각 중재동안 다수의 통신이 필요하므로, 집중형 중재는 전형적으로, 중재 신호가 전송되는 시간과 중재 획득자(arbitration winner)가 버스를 구동하는 시간 사이에 바람직하지 않는 긴 대기 기간(latency period)를 갖는다.

분산형 중재 방안에 있어서, 각 모듈은 그의 중재 신호를 시스템의 서로 다른 모듈로 전송한다. 각 모듈은 중재 알고리즘을 실행하는 논리 회로를 포함하여, 이들 중재 신호를 근거로 다음 버스 소유자를 결정한다. 중재 신호 수신시, 각 모듈은 다음 버스 소유자를 결정한다. 그 다음, 중재를 획득한 모듈은 버스를 통하여 그의 정보를 구동한다. 각 중재동안 단지 하나의 통신만이 필요하므로, 분산형 중재는 통상적으로 집중형 중재보다 짧은 대기 기간을 갖는다. 그러나, 분산형 중재는 집중형 중재보다 비싼데, 이는 각 모듈 내에 중재 알고리즘을 실행하기 위하여 개별적인 논리 회로를 필요로 하고, 각 모듈로부터 다른 모듈로 중재 신호를 통신하기 위한 가외(extra)의 하드웨어 라인을 필요로 하기 때문이다.

전술한 제약에 부가적으로, 대부분의 종래 및 분산형 중재 방안에서는 각종 비효율성으로 인하여 트랜잭션들간에 종종 버스 사이클이 낭비된다. 예를 들면, 버스 소유자가 더 이상 처리가 불가능한 모듈로 트랜잭션을 전송하는 경우, 트랜잭션은 하나의 사이클을 소비하면서 중지(abort)될 것이다. 또한, 중재와 버스 제어간에 대기 기간을 갖는 중재 방안에서는 버스를 구동할 수 있는 모듈이 없는 상황에서 중재의 대기 기간동안 사이클이 낭비된다. 이들 각각의 문제로 인하여 대역 폭(bandwidth)이 감소된다.

버스의 성능 특성을 개선하거나 혹은 변경하는데 다수의 해결방안이 사용되어 왔다. 예를 들면, 소정의 집중형 및 분산형 중재 버스에 있어서, 데이터가 버스를 통하여 전송될시에 주파수(frequency)는 종래의 파이프라이닝(pipelining)에 의해, 즉, 장래의 사이클(future cycle)동안 버스 소유권(bus ownership)에 대하여 하나 이상의 중재 단계를 가짐으로써 증가될 수 있다. 따라서 중재는 단지 적은 버스 시간만 소비할 것이며, 데이터는 보다 높은(rate)로 버스를 통하여 전송될 수 있다. 이것은 버스를 통한 데이터 전송 속도(rate of data transmission)를 개선시키는 반면에 분산형 버스의 하드웨어 효율 문제 혹은 집중형 버스의 대기 시간 한계를 제거하지는 못한다. 다른 유형의 버스에서, 중재 신호는 그들이 전송되었던 사이클과 동일한 사이클에서 처리되므로 중재는 바로 후속하는 버스 사이클동안 버스 소유권을 결정한다. 이러한 해결방안은 중재 획득자에 의한 중재와 버스 제어 사이의 사이클의 수를 감소시키지만, 각 사이클동안 필요한 최소 시간 기간을 증가시켜, 최대 버스 주파수를 감소시킨다. 높은 주파수는 통상적으로 더 높은 대역 폭을 허용한다는 점에서 바람직하다.

따라서, 현재의 버스가 하드웨어 효율, 대기 시간, 최대 주파수 및/혹은 대역 폭의 측면에서 개선될 필요가 있다.

따라서, 본 발명의 목적은 개선된 버스 중재 방안을 제공하는데 있다.

본 발명의 다른 목적은 하드웨어 측면에서 효율적인 버스 중재 방안을 제공하는 데 있다.

본 발명의 또다른 목적은 집중형 방안보다 더 짧은 대기 시간을 갖는 버스 중재 방안을 제공하는 데 있다.

본 발명의 또다른 목적은 높은 주파수와 대역 폭을 허용하는 버스 중재 방안을 제공하는 데 있다.

당업자라면 다음의 본 발명의 상세한 설명 및 바람직한 실시예, 첨부 도면, 그리고, 특허청구범위로부터 본 발명의 상기 및 기타 다른 목적들을 명백히 알 수 있을 것이다.

광범위하게 말하면, 본 발명의 파이프라인 버스 중재 방안을 갖는 버스 시스템에 관한 것이다. 버스 시스템은 버스와 이 버스에 연결되는 다수의 클라이언트 모듈(client modules)을 포함한다. 각 클라이언트 모듈은 버스를 통하여 다른 클라이언트 모듈로 정보를 전송할 수 있고, 단지 하나의 클라이언트 모듈만이 언제든지 버스를 통하여 정보를 전송할 자격이 있다. 버스를 통해 정보를 전송할 자격이 있는 모듈은 사이클을 정의하는 최소 시간 기간동안 버스를 제어한다.

어느 모듈이 버스를 사용할 자격이 있는지를 판정하기 위하여 각 클라이언트 모듈은 버스를 통하여 정보를 전송하려고 할 때 다른 모듈로 전송하는 중재 신호를 발생한다. 각 클라이언트 모듈은 모듈이 상기 버스를 통하여 정보를 전송할 자격이 있는지의 여부를 판정하기 위한 중재 신호에 응답하는 중재 신호 처리기를 구비한다.

또한, 시스템은 바람직하게 소정 사이클에서 버스에 허용되는 트랜잭션의 유형을 클라이언트 모듈에게 알려주는 호스트 모듈(host modules)을 포함한다. 각 중재 신호 처리기는 또한 바람직하게 초기 사이클(earlier cycle)동안 호스트 모듈에 의해 전송되는 클라이언트 선택사양 신호(client option signals)에 응답한다.

클라이언트 모듈은 바람직하게, 높은 우선수위 모듈과 낮은 우선순위 모듈로 분할된다. 높은 우선순위 모듈은 바람직하에 입력/출력 모듈이고, 낮은 우선순위 모듈은 바람직하게 프로세서 모듈(processor modules)이다. 중재 신호는 각 클라이언트 모듈로부터 그의 우선순위 그룹내의 다른 클라이언트 모듈로 전송된다. 또한 중재 신호는 각 입력/출력 모듈로부터 프로세서 모듈로 전송된다. 그러나, 입력/출력 모듈이 더 높은 우선순위를 가지므로, 중재 신호는 프로세서 모듈로부터 입력/츨력 모듈로 전송되지 않는다.

버스의 사용에 대한 우선순위은 바람직하게 내림차순의 우선순위로 현재 버스 소유자, 호스트 모듈, 입력/출력 모듈 및 프로세서 모듈의 우선순위 방안에 따라 결정된다.

본 발명은 고성능 공유 버스에 대한 중재 방안을 포함한다. 중재 방안은 공유 버스와 이 버스를 사용하여 상호 통신하는 구성요소 모듈 세트를 구비한 컴퓨터 시스템 환경하에 설명된다. 또한, 모듈은 버스의 제어에 대하여 상호 중재한다.

중재 방안은 분산형이다. 즉, 각 구성요소 모듈은 중재 신호를 다른 구성 요소 모듈로 전송한다. 각 모듈은 다음 버스 소유자인지를 판정하기 위한 프로세서를 구비한다. 소정 사이클동안 행해진 중재는 전형적으로, 장래(future) 두 버스 사이클의 소유권을 결정한다. 중재 방안은 파이프 라인형이다. 즉, 소정 버스 사이클, (t)동안, 중재 프로세서는 (t-1)에 수신한 중재 신호에 기초하여 사이클(t+1)동안의 버스 소유자를 계산한다. 또한, 중재 프로세서는 사이클(t+2)에서의 버스 소유권을 결정할 중재 신호를 전송 및 수신한다. 사이클(t)에서, 사이클(t-1)동안 결정된 버스 소유자는 버스를 구동한다.

본 발명의 바람직한 실시예에서, 중재를 위하여, 하나의 모듈이 " 호스트" 모듈로 작용하고, 나머지 모듈은 " 클라이언트" 모듈로 간주된다. 중재동안 클라이언트 모듈은, 그들이 버스의 제어를 원하는지의 여부를 표시하는 중재 신호를 다른 클라이언트 모듈로 전송한다. 호스트 모듈은 다음 사용가능한 버스 사이클동안 개시될 수 있는 트랜잭션의 유형을 표시하는 " 클라이언트 선택사양" 신호를 클라이언트 모듈로 전송한다. 클라이언트 선택사양 신호와 중재 신호에 기초하여, 각 모듈은 그것이 다음 버스 소유자인지의 여부를 판정한다.

특히, 중재 방안은 중재 상태, 평가 상태(evaluation state) 및 구동 상태(drive state)로서 지칭되는 3가지 단계 혹은 " 상태" 와 관련이 있다. 이들 3가지 상태는 바람직하게, 3개의 연속 버스 사이클동안 발생된다. 중재 상태동안, 호스트 모듈은 다음 사용가능한 사이클에 허용되는 트랜잭션 유형을 표시하는 클라이언트 선택사양 신호를 클라이언트 모듈로 전송하고, 클라이언트 모듈은 그들이 다음 사용가능한 사이클동안 버스 소유권을 가지려 하는지의 여부를 표시하는 중재신호를 다른 클라이언트 모듈로 전송한다. 평가 상태동안, 클라이언트 모듈과 호스트모듈의 각각은 그것이 다음 버스 소유자인지의 여부를 판정하는데 사용가능한 정보를 평가한다. 구동 상태동안, 중재를 획득한 모듈은 버스의 제어를 가지고, 클라이언트 선택사양 신호에 따라 버스를 사용할 수 있다. 중재 방안은 상세히 후술할 것이다.

제 1 도에는 본 발명에 따라 중재 방안을 사용하는 전형적인 컴퓨터 시스템이 도시되어 있다. 컴퓨터 시스템(10)은 버스(12)와 이 버스(12)에 연결된 다수의 구성요소를 포함하는 멀티프로세서 컴퓨터이다. 구성요소는 주 메모리 제어기(main memory controller)(14). 입렬/출력 모듈(16, 18) 및 프로세서 모듈(20, 22, 24 및 26)을 포함한다. 각 구성요소는, 정보가 전송되는 다수의 하드웨어라인 세트에 의해 버스(12)에 개별적으로 연결된다. 보다 상세히 후술하는

바와 같이, 주 메모리 제어기(14)는 " 호스트" 모듈이고, 나머지 구성요소는 버스의 " 클라이언트" 모듈이다.

프로세서(20, 22, 24 및 26)는 컴퓨터 시스템(10)에 대한 주 프로세서이고, 시스템에 대한 소프트웨어는 모든 프로세서상에서 동시에 실행될 수 있다. 각 프로세서 모듈은, 그것이 나머지 프로세서 모듈과 연결시키는 중재 신호 전송 라인(경우에 따라 본 명세서에서는 ARB 라인이라 함)을 제어한다. 프로세서(20, 22, 24 및 26)는 중재 신호 전송 라인(28, 30, 32 및 34)을 각각 제어한다. 프로세서가 버스(12)를 사용하기를 원할 때, 이 프로세서는 사전결정된 신호를 ARB 라인을 통하여 다른 프로세서 모듈로 전송한다. 전술한 바와 같이, ARB 라인은 통상적으로, 장래의 두 사이클의 버스(12) 제어에 대한 중재에 사용된다.

입력/출력 모듈(16, 18)은 컴퓨터 시스템(10)과 입력/출력 장치(도시되지 않음)간에 인터페이스(interface)로서 작용한다. 입력/출력 모듈(16, 18)은 각각 입력/출력 어댑터(input/output adaptor)를 포함한다. 각 입력/출력 모듈은, 그것을 나머지 입력/출력 모듈 및 각 프로세서 모듈과 연결시켜주는 중재 신호 전송라인 혹은 ARB 라인을 제어한다. 입력/출력 모듈(16, 18)은 중재 신호 전송라인(36, 38)을 각각 제어한다. 입력/출력 모듈이 버스(12)를 사용하기를 원할 경우, 사전 결정된 신호를 ARB 라인을 통하여 나머지 클라이언트 모듈로 전송한다.

주 메모리 제어기(14)는 종래의 방식으로, 주 메모리(도시되지 않음)로부터 정보를 판독하고, 주 메모리에 정보를 저장하는 역할을 한다. 주 메모리 제어기(14)는 직접 혹은 종래의 버스를 통하여, 메모리와 인터페이스한다. 전술한 바와 같이, 주 메모리 제어기(14)는 바람직하게, 버스 제어를 위한 호스트 모듈로서 작용하고, 보다 상세히 후술하는 바와 같이, 주 메모리 제어기(14)는 버스 액세스(access)에 대하여 최고 우선순위를 갖는다. 주 메모리 제어기(14)는 각 클라이언트 모듈에 직접 연결되는 CLIENT_OP 라인(40)을 제어한다. 주 메모리 제어기(14)는 다음 사용가능한 사이클동안 버스(12)상에 배치될 수 있는 트랜잭션의 유형을 표시하는 신호를 CLIENT_OP 라인(40)을 통하여 각 클라이언트 모듈로 전송한다. CLIENT_OP 신호는 통상적으로, 장래의 두 사이클(즉, 중재가 실행중인 동안의 사이클)의 버스(12) 상태와 관련이 있다.

제1도에 도시된 바와 같이, 주 메모리 제어기(14)는 클라이언트 모듈로부터 중재 신호를 수신하지 않고, 입력/출력 모듈은 프로세서 모듈로부터 중재 신호를 수신하지 않는다. 보다 상세히 후술하는 바와같이, 호스트(주 메모리 제어기)는 버스를 액세스하는 데에 있어 클라이언트 모듈보다 더 높은 우선순위를 가지고, 입력/출력 모듈은 프로세서 모듈보다 더 높은 우선순위를 갖는다. 본 발명에 따라 중재 방안을 사용하면, 각 모듈은 통상적으로, 낮은 우선순위 모듈로부터의 중재 신호를 고려하지 않고서도 버스를 사용할 자격이 있는지의 여부를 판정할 수 있다. 호스트는 CLIENT_OP 라인을 통하여 그의 우선순위를 단정한다(assert).

중재 처리를 상세히 기술하기 전에, 버스(12)를 간단히 설명할 것이다. 버스(12)는 고성능 프로세서-메모리-I/O 상호연결 버스이다. 버스(12)는 분할 트랜잭션 버스이다. 예를 들면, " 판독(READ)" 트랜잭션이 버스(12) 상에 발생된 후 " 판독" 을 발생했던 모듈은 버스를 포기하여, 다름 모듈이 다른 트랜잭션을 위해 버스를 사용할수 있도록 혀용한다. 요청된 데이터가 사용가능할 경우, " 판독" 에 대한 응답 모듈(responding module)은 버스에 대해 중재하고, 그 다음 데이터를 전송한다. " 기록(WRITE)" 트랜잭션은 분할되지 않으므로, 마스터(master)는 어드레스 사이클에 후속하여 즉시 " 기록" 데이타를 전송한다.

버스(12)는 바람직하게, 주로 데이터 전송과 관련된 적어도 3개의 버스, ADDR_DATA 버스, MASTER_ID 버스 및 TRANS_ID 버스를 포함한다. 또한 버스(12)는 버스(12)의 제어에 대한 중재와 관련된 LONG_TRANS 버스를 포함한다.

ADDR_DATA 버스는 어드레스 정보 및 데이터의 전송에 사용된다. ADDR_DATA 버스가 어드레스-관련 정보를 전송하는 사이클을 어드레스 사이클이라 하고, ADDR_DATA 버스가 데이터를 전송하는 사이클을 데이터 사이클이라 한다. 예를 들면, 기록 트랜잭션은 통상적으로, 하나 이상의 데이타 사이클이 곧바로 후속하는 하나의 어드레스 사이클을 갖는다. 버스 소유자는 데이터를 기록하기를 원하는 어드레스를 표시하는 기록 트랜잭션을 개시하고, 후속 사이클동안 데이타를 전송한다. 판독 트랜잭션은 통상적으로, 판독할 어드레스를 표시하기 위하여 버스 소유자에 의해 사용되는 하나의 어드레스 사이클을 갖는다. 이 어드레스 사이클에 이어서 소정 시간후에, 요청에 응답하는 모듈에 의해 데이터가 요청 모듈로 전송되는 하나 이상의 데이터 사이클이 후속된다. 또한, 어드레스-관련 정보 혹은 데이터가 전송되지 않는 유휴 사이클(idle cycles)이 발생될 수 있다.

MASTER_ID 및 TRANS_ID 버스는 함께 사용되므로, 분할 트랜잭션에 대한 반환 데이터가 원래의 트랜잭션과 유일하게(uniquely) 관련될 수 있다. 각 분할 트랜잭션은 MASTER_ID 버스상의 MASTER_ID 신호와 TRANS_ID 버스상의 TRANS_ID 신호에 의해 식별되는데, 이들 신호는 각각 트랜잭션을 발생하는 모듈을 식별하며, 그 트랜잭션과 그 모듈이 전송하는 다른 트랜잭션을 구별한다. 예를들면, 분할 트랜잭션의 판독은 MASTER_ID 신호와 TRANS_ID 신호의 고유 조합으로 전송된다. 그러면, MASTER_ID와 TRANS_ID는 요청된 데이터 반환을 수반하며, 그에 따라 요청 모듈은 이 반환 데이터를 수신하고, 반환 데이터는 적절한 트랜잭션과 상호관련된다.

트랜잭션 순서는 트랜잭션의 식별에 중요하지 않으므로, 이러한 메카니즘에서는 트랜잭션이 발생되었던 순서와 다른 순서로 다시 복귀될 수 있다. 고유 식별을 하기 위해서는, 소정 시간에 소정 트랜잭션 ID를 갖는 단지 하나의 트랜잭션만이 하나의 모듈로부터 돌출할 수 있다. 그러나, 트랜잭션이 MASTER_ID에 의해 구별될 수 있으므로, 두 개 이상의 개별적인 모듈이 동시에 동일한 트랜잭션 ID를 사용할 수도 있다.

긴 트랜잭션(long transaction)이 완료될 때까지, 버스(12)의 제어를 유지하기 위한 현재의 버스 소유자가 LONG_TRANS를 사용한다. 예를 들면, 일련의 사이클동안, 모듈은 대량의 데이터를 기록할 필요가 있다. 후술하는 바와 같이, LONG_TRANS가 단정(assert)되면, 다른 트랜잭션은 보다 높은 우선순위의 클라이언트 혹은 호스트에 의해 데이터의 중간에 삽입될 수 없다.

버스(12)에 대한 중재 프로세스를 보면, 중재 프로세스는 세가지 상태, 즉, 중재 상태, 평가 상태 및 구동 상태를 포함한다. 중재 상태동안, 모든 클라이언트 모듈은 그들의 ARB 라인을 활성적으로 구동하고, 호스트 모듈은 CLIENT_OP 버스를 구동하며, 현재 버스 소유자는 LONG_TRANS를 구동한다. 평가 상태동안, 각각의 클라이언트 및 호스트는 ARB, CLIENT_OP 및 LONG_TRANS 신호를 평가하여, 다음 이용가능한 버스 사이클을 구동할 것인지를 결정한다. 구동 상태동안, 중재가 효율적으로 완료된 후, 중재를 획득했던 모듈이 버스를 구동한다. 주어진 버스 사이클에서 버스(12)의 제어와 관련된 세가지 상태가 세 가지 연속적인 버스 사이클동안에 발생함이 바람직하다.

중재는 바람직하게 파이프라인된다. 주어진 사이클동안, 3가지 동작이 수행된다. 먼저 현재 버스 소유자는 버스(12)를 구동한다. 두 번째, 클라이언트는 이전 사이클(t-1)동안 전송된 중재-관련 신호를 평가하여, 다음 사이클(t+1)동안의 버스 소유자를 결정한다. 마지막으로, 장래의 두 사이클(t+2)동안 버스 소유자를 결정하기 위하여, 중재-관련 신호가 ARB 라인, CLIENT_OP 버스 및 LONG_TRANS 버스상에서 구동된다.

사이클(t+2)에서의 버스(12) 소유권은 최고 우선 순위부터 최저 우선순위까지(1) 현재 버스 소유자, (2) 주 메모리

제어기(14), (13) 입력/출력 모듈(16, 18) 및 (4) 프로세서 모듈(20, 22, 24 및 26)의 4단 우선순위 방안(four-tier priority scheme)에 따라 결정된다. 동일한 우선 순위를 갖는 두 개 이상의 모듈이 버스의 제어에 대하여 중재중이면, 임의의 종래의 중재 프로토콜(arbitration protocol)이 버스 소유자가 되는 하나의 모듈을 결정하는데 사용될 수 있다. 예를 들면, 종래의 라운드 로빈 프로토콜(round robin protocol)은 모듈이 버스의 액세스를 얻기 위하여 큐(queue)에서 효과적으로 대기하고, 가장 최근에 버스를 제어했던 모듈이 큐의 끝부분으로 이동되는데 이용될 수 있다.

동일한 우선순위를 갖는 모듈중에서 선택하기 위한 중재 프로토콜에 있어서, 중재를 획득한 모듈은, 다른 우선 순위의 모듈이 버스의 제어를 획득할 수 있기 전에, 적어도 두 개의 연속된 사이클동안 버스를 제어할 권리를 획득함이 바람직하다.

이것은 중재와 버스 소유권간의 두 사이클의 지연을 초래하여, 필요한 경우 다른 동일한 우선순위 모듈이 버스의 제어를 제공(grant)받기 전에 제어를 획득한 모듈이 LONG_TRANS를 단정하게 한다. 몇가지 실시예에 있어서, 단지 동일한 우선순위의 모듈들이 중재중일 때, 적어도 두 싸이클동안, 모듈이 자동적으로 제어를 획득한다. 다른 실시예에서, 하나의 모듈은 하나의 사이클동안 버스의 제어를 획득하고, 다음 사용가능한 사이클동안 다른 동일한 우선순위의 모듈보다 우선적으로 선취권을 제공받으며, 그에 따라 이 모듈이 그 사이클동안 중재하는 경우에 두 번째 사이클의 제어를 획득할 수 있게 된다.

현재 버스 소유자가 버스의 사용을 위한 최고 우선 순위를 가지므로써, 긴 트랜잭션의 효율적인 전송을 할 수 있게 된다. 즉 버스 소유자는 트랜잭션의 중간에 버스의 제어를 포기할 필요가 없으며, 차후 시간에 트랜잭션을 완료하기 위하여 버스의 제어를 중재한다. 중재 상태동안 현재 버스 소유자가 LONG_TRANS를 단정하는 경우, 이 현재 버스 소유자는 중재를 획득할 것이며, 구동 상태 단계에서 다시 버스를 구동할 자격을 가지게 된다. 다수의 사이클의 트랜잭션을 전송하는 버스 소유자는 다수 사이클의 트랜잭션중의 마지막 두 사이클을 제외한 모든 사이클동안 LONG_TRANS를 단정해야 한다. 중재 상태와 구동 상태간의 지연으로 인하여, 마지막 두 사이클동안 LONG_TRANS를 단정할 필요가 없다. 트랜잭션의 종단부에서, 모듈은 LONG_TRANS를 단정해제(de-assert)해야 하며, 그에 따라 다른 모듈이 성공적으로 중재를 할 수 있다. 호스트가 CLIENT_OP 버스의 제어를 가지므로, 버스(12)의 제어를 획득하기 위하여 LONG_TRANS를 사용할 필요가 없는데, 후술하는 바와 같이, 이는 클라이언트에 의한 새로운 트랜잭션을 허용하지 않는 CLIENT_OP 신호를 전송할 수 있기 때문이다. 따라서, 호스트는 버스(12)의 제어를 가질 때 LONG_TRANS를 단정해제시킨다.

호스트 모듈은 버스의 사용을 위한 제 2의 최고 우선순위를 갖는다. 호스트 모듈은 CLIENT_OP 버스를 사용하여, 다음 사용가능한 버스 사이클동안 허용되는 트랜잭션의 유형을 클라이언트 모듈에게 알려주고, 새로운 트랜잭션의 개시를 허용하지 않음으로써, 단독으로 버스를 제어한다.

바람직한 실시예에서, CLIENT_OP 버스는 표 1 에도시된 신호를 지원한다.

표 1

이름	값	의미
SHAR_RTN	000	t+2에서, 호스트는 공유 반환을 위하여 버스(12)를 제어
HOST_CONTROL	001	t+2에서, 호스트는 버스(12)를 제어
NONE_ALLOWED	010	t+2에서, 트랜잭션이 허용되지 않지만, 클라이언트는 여전히 버스(12)를 제어
ONE_CYCLE	011	t+2에서, 하나의 사이클 트랜잭션이 허용됨
RET_ONLY	100	t+2에서, 반환 혹은 응답 트랜잭션이 허용됨
NO_IO	101	t+2에서, I/O트랜잭션 이외의 트랜잭션이 허용됨
ATOMIC	110	t+2에서, "아토믹 소유자(atomic owner)"인 클라이언트가 소정의 트랜잭션을 발생할 수 있고, 다른 클라이언트는 단지 응답만을 발생할 수 있음
ANY_TRANS	111	t+2에서, 트랜잭션이 허용됨

버스 제어의 획득에 직접 관련되는 CLIENT_OP 신호는 후술되어 있으며, 본 발명의 중재 방안에서 중요하다. 표 1로부터의 다른 CLIENT_OP 신호는 컴퓨터 시스템(10)의 중재 방안에 있어서 중요하지 않지만, 예시를 위하여, CLIENT_OP 버스가 소정 사이클동안 허용되는 트랜잭션을 제한하는데 사용될 수 있는 방식을 보여기간 위하여 포함된다. 이들 다른 CLIENT_OP 신호를 간단히 기술할 것이다.

ANY_TRANS 및 HOST_CONTROL 클라이언트 선택사양 신호는 상대적으로 간단하다. ANY_TRANS의 CLIENT_OP는 사이클(t+2)동안 트랜잭션이 허용됨을 표시한다. HOST_CONTROL의 CLIENT_OP는 사이클(t+2)에서 호스트가 버스를 제어하려함을 표시한다.

ONE_CYCLE 클라이언트 선택사양 신호는 주로, 클라이언트로부터 호스트로의 소유권 변경을 위하여 사용된다. CLIENT_OP가 ONE_CYCLE인 경우, 사이클(t+2)에서 단지 하나의 사이클 트랜잭션만이 허용된다. 전형적인 경우에 있어서, 호스트가 버스(12)를 제어하려할 경우, 연속 사이클동안 ONE_CYCLE과 HOST_CONTROL을 발생한다. ONE_CYCLE 클라이언트 선택사양 신호는 호스트가 제어를 원하는 사이클 바로 앞의 사이클동안 클라이언트가 LONG_TRANS를 단정하지 못하게 한다.

전술한 바와 같이, 중재 상태와 본 발명의 중재 방안의 구동 상태간의 지연으로 인하여, ONE_CYCLE 신호가 필요하다.

사이클(t)동안 호스트 모듈은 어느 클라이언트가 사이클(t+1)동안 버스를 제어할지, 혹은 그 클라이언트가 LONG_TRANS를 단정할지를 알지 못한다. 클라이언트에게 사이클(t+1)동안 LONG_TRANS를 단정하는 것이 허용되었으면, 호스트는 적시에 이 정보를 평가할 수 없으므로, 사이클(t+2)에서 호스트와 LONG_TRANS를 단정하는 클라이언트간의 버스 소유권 주장으로 인한 분쟁을 피하게 된다. 따라서, 호스트는 사이클(t-1)동안 ONE_CYCLE 신호를 전송함으로써, 사이클(t+1)에서의 버스 소유자가 LONG_TRANS를 단정하지 못하게 한다. 물론, LONG_TRANS가 사이클(t)에서 단정될 수 있다.

그러나, LONG_TRANS가 사이클(t)에서 단정되는 경우, 호스트는 사이클(t+2)에서 버스를 구동할 수 없음을 알기 위하여, 적시에 정보를 처리할 수 있을 것이다.

현재 버스 소유자가 LONG_TRANS를 단정할 수 없고, 호스트 모듈은 클라이언트가 버스를 제어하도록 하는 클라이언트 선택사양 신호를 구동하며 버스를 중재하는 입력/출력 모듈이 없다고 가정하는 경우, 프로세서 모듈은 중재를 획득할 것이다. 획득자는 트랜잭션을 개시할 수 있거나 혹은 CLIENT_OP를 근거로 허용된 반환을 개시할 수 있다. 트랜잭션이 3개 이상의 사이클인 경우, 새로운 소유자는 트랜잭션동안 버스의 소유권을 유지하기 위해 LONG_TRANS를 단정해야 한다. 물론, 획득자는 새로운 트랜잭션의 개시 대신에 유휴 사이클을 구동하도록 허용된다. 이것은 예를 들면, 새로운 소유자가 긴 트랜잭션을 전송할 필요가 있고, CLIENT_OP가 ONE_CYCLE인 경우에 필요하다.

높은 우선순위 모듈이 낮은 우선순위 모듈로부터 중재 신호를 수신하지 못한다는 사실을 고려하여 중재를 획득하도록 높은 우선순위 모듈은 여분의 사이클동안 중재해야 한다. (LONG_TRANS가 단정되지 않고 클라이언트 제어가 허용된다고 가정하면) 우선순위 모듈이 중재중인 제 1 사이클에서, 프로세서 모듈은 여전히 중재를 획득할 것이다. 그 다음, 프로세서 모듈은 단지 하나의 사이클 트랜잭션일 수 있다는 부가된 제약과 함께 CLIENT_OP에 의해 허용되는 새로운 트랜잭션을 개시할 수 있다. 따라서, " 효과적인" CLIENT_OP는 프로세서 모듈이 관계되는 한 ONE_CYCLE이다. 이러한 전이 방법(transition method)은 사이클(t+1)동안 프로세서 모듈이 LONG_TRANS를 단정하지 못하게 한다.

높은 우선순위 모듈이 중재하는 제 2 사이클에서, 호스트 모듈은 혹은 현재 버스 소유자가 다음 사용가능한 사이클동안 버스를 제어할 자격이 없다고 가정시에, 이 높은 우선위 모듈은 중재를 획득할 것이다. 둘 이상의 입력/출력 모듈이 중재중이라면, 획득 모듈을 결정하는데 (라운드 로빈 프로토콜과 같은) 종래의 중재 프로토콜을 사용한다. 획득한 입력/출력 모듈은 CLIENT_OP 인코딩을 근거로 허용되는 반환 또는 임의의 트랜잭션을 개시할 수 있고, 프로세서 클라이언트에 대하여 전술한 방식으로 LONG_TRANS를 단정할 수 있다.

전술한 바와 같이, 단지 프로세서 모듈만이 중재중일 때, 프로세서 모듈은 중재를 획득하고 버스(12)를 구동할 것이다. 중재중인 클라이언트가 없고, 호스트가 버스(12)를 클라이언트에게 제공(grant)하는 경우에도, 프로세서 모듈은 전형적으로 유휴 사이클과 더불어 중재를 획득하고, 버스(12)를 구동시킬 것이다.

아래의 표 2는 전형적인 일련의 연속된 버스 사이클(10)동안의 중재 처리를 도시하고 있다. 표 2의 제 1 열은 사이클(1 내지 10)을 목록으로 만든 것이다. 표 2의 나머지 열에 있어서, 표 2는 각 사이클동안 CLIENT_OP 버스, LONG_TRANS 버스, 프로세서 모듈 ARB 라인 및 입력/출력 모듈 ARB 라인 상에 구동되는 신호를 보여준다. 표 2의 마지막 열은 10개의 사이클의 각각 동안의 현재 버스 소유자를 목록으로 만든 것이다. LONG_TRANS 혹은 ARB 라인 열에서의 " 1" 은 각각, LONG_TRANS가 단정되거나 혹은 표시된 ARB 라인을 제어하는 모듈이 다음 사용 가능한 버스 사이클 동안의 버스 소유권을 얻으려함을 나타낸다.

표 2

사이클	CLIENT_OP	LONG_TRANS	프로세서 모듈 ARB 라인 28/30/32/34	입력/출력 모듈 ARB 라인 36/38	소유자
1	ANY_TRANS	0	0/0/1/1	0/0	모듈(20)
2	ONE_CYCLE	0	0/0/1/1	0/0	모듈(20)
3	HOST_CONTROL	1	0/0/0/1	0/0	모듈(24)
4	HOST_CONTROL	1	0/0/0/1	0/0	모듈(24)
5	HOST_CONTROL	1	0/0/0/1	0/0	모듈(24)
6	HOST_CONTROL	1	0/0/0/1	0/0	모듈(24)
7	HOST_CONTROL	0	0/0/0/1	0/0	모듈(24)
8	HOST_CONTROL	0	0/0/0/1	0/0	모듈(24)
9	HOST_CONTROL	0	0/0/0/1	0/0	HOST
10	HOST_CONTROL	0	0/0/0/1	0/0	HOST

표 2에 도시된 바와 같이, 사이클(1)동안 프로세서 모듈(20)은 버스(12)를 소유하고, LONG_TRANS를 단정하지 않는다. 프로세서 모듈(24, 26)의 각각은 ARB 라인(32, 34)을 각각 단정하여, 버스(12)의 제어를 중재한다. HOST는 ANY_TRANS 클라이언트 선택사양 신호를 발생하고, 입력/출력 모듈은 버스(12)의 제어를 중재하지 않는다. 사이클(1)동안 전송된 신호는 사이클(2)동안 평가되고, 이들 신호를 근거로, 프로세서 모듈(24)은 다음 사용 가능한 버스 사이클(3)동안 버스 소유자가 된다.

사이클(2)동안, 호스트가 ONE_CYCLE 클라이언트 선택사양 신호를 발생하는 것을 제외하고는 거의 동일한 신호가 전송된다. 따라서, 프로세서 모듈(24)은 또한 사이클(4)동안 버스의 제어를 획득한다.

사이클(3 내지 6)동안 모듈(24)은 LONG_TRANS를 단정한다. 따라서, 프로세서 모듈(24)은 사이클(3) 내지 사이클(8)동안 버스(12)의 제어를 유지한다.

사이클(7)동안의 중재는 사이클(9)동안의 버스 소유권을 결정한다. 프로세서 모듈(24)이 사이클(7)동안 LONG_TRANS를 단정하지 않기 때문에 그리고, 호스트가 HOST_CONTROL을 단정하므로, 호스트는 중재를 획득하여 사이클(9)동안 버스(12)를 제어한다. 호스트는 사이클(8)동안 계속 HOST_CONTROL을 단정함으로써, 사이클(10)동안 버스(12)의 제어를 유지한다.

이제 표 1에 도시된 다른 예시적인 클라이언트 선택사양 신호를 간단히 기술할 것이다. 다수의 CLIENT_OP 신호는 입력/출력 모듈(16, 18)과 관련이 있다. 제1도에 도시된 바와 같이, 입력/출력 모듈(16, 18)은 모듈이 더 이상의 입력/출력 트랜잭션을 수용할 수 없음을 표시하는 신호를 메모리 제어기(14)로 전송하기 위해 STOP_IO 라인(58, 60)을 각각 제어함이 바람직하다. 또한 입력/출력 모듈(16, 18)은 바람직하게 신호를 메모리(14)로 전송하기 위한, 그리고 메모리 시스템을 효과적으로 제어하기 위하여 상호 전송하기 위해 STOP_MOST 라인(62, 64)을 각각 제어한다.

입력/출력 모듈이 새로운 입력/출력 트랜잭션에 응답할 수 없을 때, 각 입력/출력 모듈은 STOP_IO 라인을 사용하여 호스트에게 이를 알려준다. 입력/출력 모듈은 STOP_IO 라인을 사용하여 호스트에게 알려주어, 호스트는 새로운 트랜잭션을 허용하지 않는 CLIENT_OP들만을 순차로 발생한다.

따라서, 입력/출력 모듈이 사용중인 경우, 입력/출력 모듈은 그의 STOP_IO 라인을 단정한다. 그 다음, 호스트는 NO_IO 신호를 단정한다. CLIENT_OP가 NO_IO인 경우, I/O 트랜잭션을 제외한 모든 트랜잭션이 허용된다.

RET_ONLY 클라이언트 선택사양 신호는 단지, 이전에 보유한 전용-더티 캐시 라인(private-dirty cache lines)의 반환(라이트백(write-backs)), 혹은 이전의 트랜잭션에 대한 응답이 허용됨을 표시한다. 예를 들면, 프로세서(24)가 프로세서(20)의 캐시의 전용-더티인 캐시 라인의 코히어런드 판독(coherent read)을 발생하는 경우, 프로세서(20)는 캐시-캐시 복사(cache to cache copy)로 이 캐시 라인을 제공할 수 있다. 캐시-캐시 복사가 코히어런트 판독(coherent read)의 응답이기 때문에, 캐시-캐시 복사 트랜잭션은 RET_ONLY 클라이언트 선택사양 신호의 영향 하에 개시될 수 있다. 유사하게, I/O 모듈(16)은 RET-ONLY 클라이언트 선택 사양 신호의 영향 하에서, 초기 I/O 판독 트랜잭션으로부터 데이터를 반환할 수 있는데, 이는 데이터 반환이 I/O 판독 트랜잭션에 대한 응답이기 때문이다.

STOP_MOST 라인은 효과적으로 메모리 시스템을 제어하는 데 사용된다. 이것은, 예를 들어 다른 모듈이 적절한 메모리 어드레스로부터/로 데이터를 판독 혹은 기록하지 않고 하나의 모듈이 다수의 메모리 동작을 수행해야할 때 유용하다. 입력/출력 모듈이 STOP_MOST를 단정할 때, 그것은 " 아토믹 소유자" 가 되고, 단지 아토믹 소유자만이 새로운 트랜잭션을 발생하게 한다. 다른 모듈은 단지 반환 혹은 응답형 트랜잭션을 발생할 수 있다. 하나 이상의 입력/출력 모듈이 동시에 STOP_MOST를 단정하는 경우, 종래의 중재 알고리즘이 사용되어, 아토믹 소유자가 되는 입력/출력 모듈이 결정된다. 당업자라면, STOP_MOST 라인과 STOP_MOST 신호가 본 발명에 필요없다는 것을 알 수 있을 것이다.

흐름 제어가 정상적으로 ANY_TRANS를 허용한다고 가정시에 STOP_MOST를 단정하는 클라이언트에 직업 응답하여 ATOMIC CLIENT OP가 발생된다. ATOMIC CLIENT_OP는 STOP_MOST를 단정하는 클라이언트가 버스(12)상에서 다수의 연속된 트랜잭션을 수행하도록 한다. ATOMIC이 단정되는 사이클동안, 모든 다른 클라이언트가 버스를 얻는 경우, 그들은 단지 초기 전송된 트랜잭션에 응답하거나, 혹은 이전에 보유한 전용-더티 캐시 라인을 라이트백할 수 있게 된다. 또한, 호스트는 통상적으로, RET_ONLY 클라이언트 선택사양 신호를 사용하여 모든 클라이언트를 응답형 트랜잭션으로 제한할 수 있다. 따라서 아토믹 소유자가 있을 때, 이 아토믹 소유자에 대한 효과적인 클라이언트 선택사양 신호는 ANY_TRANS이며, 모든 다른 클라이언트에 대한 효과적인 클라이언트 선택사양 신호는 RET_ONLY이다.

NONE_ALLOWED 클라이언트 선택사양 신호는 사이클(t+2)에서 허용되는 트랜잭션이 없음을 나타내는데 사용된다. 또한 NONE_ALLOWED는 ONE_CYCLE 클라이언트 선택사양 신호의 사용과 유사한 방식으로, RET_ONLY후의 소유권 변경에 사용될 수 있다. 이러한 전형적인 경우에 있어서, 호스트는 후속 사이클에서 RET_ONLY, NONE_ALLOWED, 그다음, HOST_CONTROL을 발생시켜 버스(12)를 제어한다.

전술한 중재 방안에 필요한 것은 아니지만, 임의의 또는 모든 클라이언트 모듈(프로세서와 입력/출력)은 종래의 캐시 메모리를 구비할 수 있다. 본 기술분야에 잘 알려진 바와 같이, 컴퓨터 시스템은 빈번히 사용되는 데이터를 저장하기 위한 캐시 메모리와 모든 프로세서에 의해 사용되는 공유형 주 메모리를 구비한 다수의 프로세서 모듈을 가질 수 있다. 메모리는 전형적으로 빈번히 사용되는 데이터와 데이터 항목이 주 메모리에 저장되는 어드레스를 저장한다. 프로세서가 메모리의 어드레스로부터 데이터를 얻으려할 때, 데이터에 대한 어드레스를 사용하여 캐시 메모리로부터 그 데이터를 요청한다. 그의 캐시 메모리는 그 어드레스를 보유하고 있는지의 여부를 알기 위해 조사하고, 만약 보유하고 있으면, 데이터를 직접 프로세서로 반환한다. 캐시 메모리가 원하는 정보를 포함하지 않는 경우에는 정규 메모리 액세스가 발생된다. 캐시 메모리는 전형적으로 주 메모리(통상적으로 RAM)의 액세스가 마이크로프로세서 속도에 비하여 저속일 때 유용하다.

프로세서상에 실행하는 소프트웨어가 통상적으로 특정 어드레스에 저장된 데이터에 대한 가장 최근의 값을 사용하는 것이 필요하다. 각 프로세서가 캐시메모리를 구비한 컴퓨터 시스템(10)과 같은 공유 메모리 멀티-프로세서 컴퓨터에 있어서, 하나 혹은 그 이상의 캐시 메모리 혹은 주 메모리에 가장 최근의 데이터가 저장될 수 있다. 따라서, 프로세서가 다른 프로세서에 의해 사용되어 온 데이터를 탐색할 때마다, 프로세서에 사용가능한 데이터가 현재 데이터임을 확인하기 위하여 " 코히어런시 방안(coherency scheme)" 을 구현할 필요가 있다. 전형적인 코히어런시 방안에 있어서, 각 모듈은 그의 캐시 메모리의 코히어런시 검사를 수행하여, 요청된 어드레스와 관련된 데이터를 갖는지의 여부를 판정하고, 그 코히어런시 검사의 결과를 보고한다.(report). 각각의 모듈에 의해 수행된 코히어런시 검사의 결과는 분석되고, 가장 최근 데이터는 그 데이터를 요청한 모듈에 의해 사용된다.

컴퓨터 시스템(10)의 경우에 있어서, 캐시 메모리를 구비한 각 모듈은 메모리 제어기(14)로 직접 신호를 전송하기 위한 적어도 하나의 코히어런트 트랜잭션 신호 전송 라인(즉, COH 라인)을 제어하며, 이 때 메모리 제어기(14)로 전송되는 신호는 하나 이상의 캐시 메모리에 저장될 수 있는 데이터의 판독 혹은 기록과 관련된 트랜잭션을 조정하는 신호이다. 이러한 트랜잭션을 코히어런트 트랜잭션(coherent transactions)이라 한다. 코히어런트 트랜잭션은 멀티프로세서 컴퓨터에 대해 일반적으로 행해지고 있는 방식으로 조정될 수 있다. 프로세서 모듈(20, 22, 24 및 26)은 COH 라인들(42, 44, 46, 48)을 각각 제어한다. 입력/출력 모듈(16, 18)은 COH 라인(50, 54)을 각각 제어한다.

제1도에 도시된 바람직한 실시예에서, 입력/출력 모듈(16, 18)의 각각은 개별적인 캐시 메모리를 구비한 두 개의 개별적인 입력/출력 어댑터를 포함한다. 따라서 COH 라인(50, 54)의 제어에 부가적으로, 입력/출력 모듈(16, 18)은 또한 COH 라인(52, 56)을 제어하고, 그에 따라 각 입력/출력 어댑터는 개별적인 COH 라인을 제어한다.

CLIENT_OP 버스는 특정 트랜잭션과 관련된 코히어런시 상태를 데이터를 요청하는 모듈로 다시 전송하는데 사용될 수 있다. 호스트는 SHAR_RTN 클라이언트 선택사양 신호를 사용하여, 종래의 공유 코히어런시 상태를 갖는 데이터 반환에 대하여 중재한다. 반면에, 클라이언트 모듈은 바람직하게 판독 반환이 종래의 전용 상태를 갖는다고 가정한다.

당업자라면, 메모리 제어기(14)가 단지 편리성을 위하여 호스트로서 동작하며, 호스트 기능과 메모리 제어기 기능은 두 개 혹은 그 이상의 모듈로 분리될 수 있다는 것을 알 수 있을 것이다.

" 버스" 및 " 라인" 이라는 용어는 전술한 하나 혹은 그이상의 각종 전기 경로 세트를 표시하기 위하여 상세한 설명에서

사용되었다. 당업자라면, " 버스" 및 " 라인" 이라는 용어로 상호 배타적이거나 혹은 그들 자신을 제한하려는 것이 아님을 이해해야 할 것이다. 예를 들면, 항목 " LONG_TRANS 버스" 가 사용되는 동안, LONG_TRANS 버스는 종래의 공유 버스 즉, 신호가 하나 이상의 모듈로 전송될 수 있는 하나의 전기 경로로 구성될 수 있다. 유사하게, 항목 " CLIENT_OP 버스" 와 " CLIENT_OP 라인" 은 전술한 바와 같이 호스트에 의해 단지 구동되는 하드웨어 라인 세트를 표시하기 위하여 교체 가능하도록 사용된다.

당업자라면 전술한 설명 및 첨부 도면으로부터 본 발명의 각종 변형을 명백히 알 수 있을 것이다. 따라서, 본 발명은 단지 다음의 특허 청구 범위의 범주에 의해서만 제한된다.

(57) 청구의 범위

청구항 1.

버스 중재 방안(bus arbitration scheme)을 갖는 버스 시스템에 있어서,

버스와,

상기 버스에 연결된 다수의 모듈(modules) - 상기 모듈들은 다수의 우선 순위 클래스(priority classes)를 구비하고, 상기 클래스들중 적어도 하나는 상기 클래스에 속하는 적어도 2개의 모듈을 가지며, 상기 모듈의 각각은 상기 버스를 통하여 상기 모듈들중의 다른 모듈로 정보를 전송할 수 있고, 상기 모듈들중의 단지 하나의 모듈만이 소정의 주어진 시간에 상기 버스를 통하여 정보를 전송할 자격이 있으며, 상기 모듈이 상기 버스를 통하여 정보를 전송하기를 원할 때 상기 모듈의 각각은 그의 우선순위 클래스의 모든 모듈과 하위 우선순위 클래스의 모든 모듈로 전달되는 중재 신호(arbitration signal)를 발생하고, 상기 우선순위 클래스들은 순위가 매겨지며, 순위가 높은 우선순위 클래스에 속하는 모듈들이 순위가 낮은 클래스에 속하는 모듈보다 우선하여 상기 보스에 대한 액세스를 제공받음 - 을 포함하되,

소정의 주어진 시간에 상기 버스를 통하여 정보를 전송할 자격이 있는 모듈은 사이클(cycle)을 정의하는 최소 시간 기간동안 상기 버스를 제어하고,

상기 모듈의 각각은 적어도 하나의 중재 신호에 응답하는 중재 신호 프로세서를 구비하며, 상기 중재 신호 프로세서는 상기 모듈이 선택된 장래의 사이클(future cycle)동안 상기 버스를 통하여 정보를 전송할 자격이 있는지의 여부를 판정하는 버스 시스템.

청구항 2.

버스 중재 방안(bus arbitration scheme)을 갖는 버스 시스템에 있어서,

버스와,

상기 버스에 연결된 다수의 모듈(modules) - 상기 모듈들은 다수의 우선 순위 클래스(priority classes)를 구비하고, 상기 모듈의 각각은 상기 버스를 통하여 상기 모듈들중의 다른 모듈로 정보를 전송할 수 있고, 상기 모듈들중의 단지 하나의 모듈만이 소정의 주어진 시간에 상기 버스를 통하여 정보를 전송할 자격이 있으며, 상기 모듈이 상기 버스를 통하여 정보를 전송하기를 원할 때 상기 모듈의 각각은 그의 우선순위 클래스의 모든 모듈과 하위 우선순위 클래스의 모든 모듈로 전달되는 중재신호(arbitration signal)를 발생함 - 을 포함하되,

소정의 주어진 시간에 상기 버스를 통하여 정보를 전송할 자격이 있는 모듈은 사이클(cycle)을 정의하는 최소 시간 기간 동안 상기 버스를 제어하고,

상기 모듈의 각각은, 적어도 하나의 중재 신호에 응답하여, 상기 모듈이 선택된 장래의 사이클(future cycle)동안 상기 버스를 통하여 정보를 전송할 자격이 있는지의 여부를 판정하는 중재 신호 프로세서를 구비하며,

상기 우선 순위 클래스들 중 하나는 호스트 모듈(host module)을 포함하고, 상기 호스트 모듈은 최고 우선순위를 가지며, 클라이언트 선택사양 신호를 상기 각각의 모듈로 전송하는 클라이언트 선택사양 라인(client option line)을 제어하며, 상기 중재 신호 프로세서는 상기 클라이언트 선택사양 신호에 추가 응답하여, 다음 이용가능한 버스 사이클에서 상기버스에 배치될 수 있는 트랜잭션의 유형을 결정하는 버스 시스템.

청구항 3.

제 1 항에 있어서.

상기 모듈의 각각은, 단지, 그의 우선순위 클래스의 각기 다른 상기 모듈과, 그의 우선순위 클래스보다 낮은 우선순위를 갖는 우선순위 클래스의 각기 다른 상기 모듈로만 중재신호를 전송하는 버스 시스템.

청구항 4.

제 1 항에 있어서,

상기 버스를 통하여 정보를 전송할 자격이 있는 모듈은 긴 트랜잭션 신호를 상기 각각의 모듈로 전송하는 긴 트랜잭션 라인을 제어하고, 상기 중재 신호 프로세서는 상기 긴 트랜잭션 신호에 추가로 응답하는 버스 시스템.

청구항 5.

제 1 항에 있어서,

선택된 사이클 동안 상기 버스를 통해 정보가 전송되고,

제 1 선택된 장래 사이클동안에 모듈이 상기 버스를 통하여 정보를 전송할 자격이 있는지의 여부를, 각 중재 신호 프로세서가 상기 선택된 사이클 동안 결정하며,

제 2 선택된 장래 사이클동안에 상기 버스를 통하여 정보를 전송할 자격을 가진 모듈을 결정하는 중재 신호가, 상기 선택된 사이클 동안 발생되는 버스 시스템.

청구항 6.

제 2 항에 있어서,

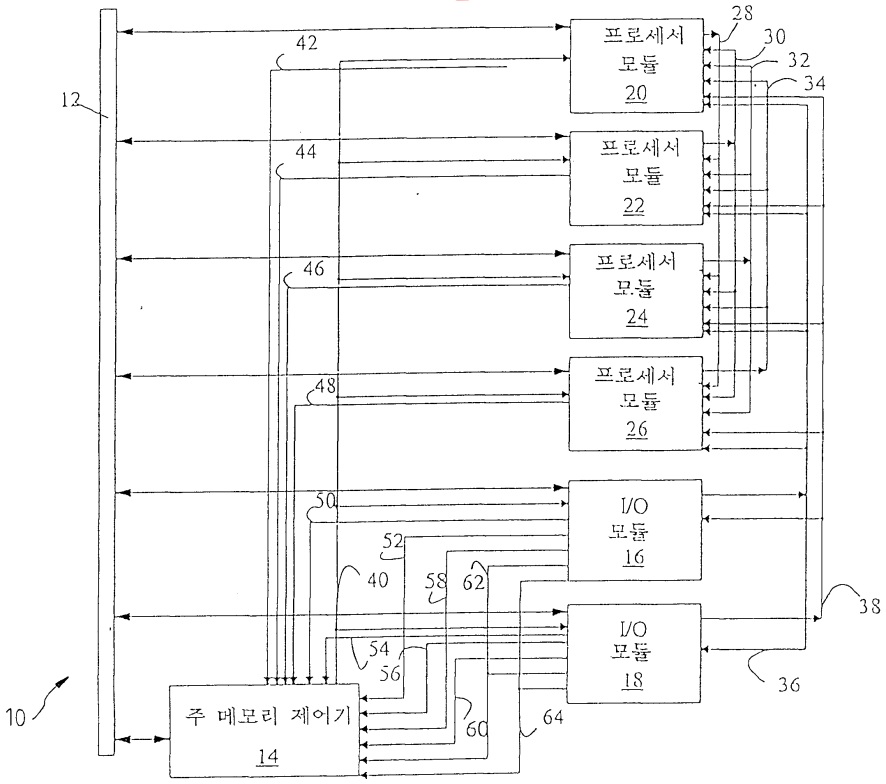
선택된 사이클 동안 산이 버스를 통해 정보가 전송되고,

제 1 선택된 장래 사이클동안에 모듈이 상기 버스를 통하여 정보를 전송할 자격이 있는지의 여부를, 각 중재 신호 프로세서가 상기 선택된 사이클 동안 결정하며,

제 2 선택된 장래 사이클동안에 상기 버스를 통하여 정보를 전송할 자격을 가진 모듈을 결정하는 중재 신호가, 상기 선택된 사이클 동안 발생되는 버스 시스템.

도면

도면 1


프로세서 모듈 20
프로세서 모듈 22
프로세서 모듈 24
프로세서 모듈 26
I/O 모듈 16
I/O 모듈 18
주 메모리 제어기 14
10
12
28
30
32
34
36
38
40
42
44
46
48
50
52
54
56
58
60
62
64