(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.[6]
G06F 13/12

(45) 공고일자 2005년07월12일
(11) 등록번호 10-0474658
(24) 등록일자 2005년02월23일

(21) 출원번호	10-1998-0709203	(65) 공개번호	10-2000-0011052
(22) 출원일자	1998년11월14일	(43) 공개일자	2000년02월25일
번역문 제출일자	1998년11월14일		
(86) 국제출원번호	PCT/US1997/001634	(87) 국제공개번호	WO 1997/46944
국제출원일자	1997년02월04일	국제공개일자	1997년12월11일

(81) 지정국

국내특허 : 아일랜드, 일본,

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

(30) 우선권주장 08/659,728 1996년06월06일 미국(US)

(73) 특허권자
어드밴스드 마이크로 디바이시즈, 인코포레이티드
미국 캘리포니아 94088-3453 서니베일 원 에이엠디 플레이스 메일 스톱68

(72) 발명자
싱 앨록
미국 캘리포니아 94555 프레몬트 노로코 서클 4504

로이 래자트
미국 캘리포니아 94087 서니베일 이턴 웨이 927

쿠오 제리
미국 캘리포니아 95129 산조세 페탈 웨이 1631

(74) 대리인
박장원
박장원

심사관 : 강철수

(54) 다수의전송패킷들에대처하기위한어드레스생성및SRAM으로/으로부터의데이터경로조정

요약

스테이션과 이서넷 간의 데이터의 전송을 제어하는 이서넷 제어기가 개시되는바, 이 이서넷 제어기는 스테이션 CPU, 메모리 버퍼 및 이서넷 간의 데이터의 전송을 관리하는 4개의 FIFO들을 갖는다. 상기 4개의 FIFO들 각각은 제어기의 성능을 최대화하도록 선택된 사이즈를 갖는다. 이 제어기는 FIFO들 각각으로부터의 어떤 미결 요구가 우선 순위를 가질 것인지를 조정하는 조정기를 포함한다. 이 제어기는 각 FIFO에 의한 데이터의 전송을 허가 마다 32 바이트들로 제한한다. 각 FIFO는 데이터를 제 1 비트 사이즈 포맷에서 제 2 비트 사이즈 포맷으로 변환하는 논리를 포함한다. 이 제어기는 또한 16 비트 어드레스를 8 비트 어드레스 버스를 통해 전송하기 위해 2개의 8 비트 부분들로 변환하는 논리, 및 상기 2개의 8 비트 부분들을16 비트 어드레스로 재포맷하는 논리를 포함한다.

대표도

도 3

명세서

기술분야

본 발명은 일반적으로 이서넷(ethernet)과 스테이션 간에 이루어지는 정보의 전송을 제어하는 방법 및 장치에 관한 것으로서, 특히 정보의 전송 성능을 증가시키는 방법 및 장치에 관한 것이다. 보다 구체적으로, 본 발명은 데이터를 제 1 비트 폭으로부터 제 2 비트 폭으로 변환시키고, 이 변환된 데이터에 대한 어드레스들을 생성하며, 그리고 SRAM 버퍼로/로부터의 기록 또는 판독 액세스를 위한 데이터 경로에 대한 액세스를 조정하는 방법 및 장치에 관한 것이다.

배경기술

국부 영역 네트워크("LAN")는 직장, 빌딩, 또는 이러한 빌딩들의 군(cluster)과 같은 제한된 지리적 영역 내에서, 퍼스널 컴퓨터, 워크스테이션, 파일 서버, 중계기, 데이터 단말 장치("DTE") 및 다른 정보 처리 장치가 서로 정보를 전자적으로 전송할 수 있게 하는 통신 시스템이다. LAN 내의 각 정보 처리 장치는 네트워크의 동작을 정의하는 고정 프로토콜(또는 표준 규격)에 따라 LAN 내의 다른 정보 처리 장치와 통신한다.

ISO의 개방형 시스템들의 상호 연결 기본 참조 모델은 LAN에서의 데이터 통신에 대해 7 계층 모델을 정의한다. 모델의 최하위 계층은 물리 계층이다. 이 물리 계층은 (a) 데이터가 전자적으로 전송되는 경로로서, 네트워크 노드들을 서로 연결하는 물리적인 매체, (b) 네트워크 노드들이 물리적인 전송 매체와 인터페이스하는 방식, (c) 물리적인 전송 매체를 통해 데이터를 전송하는 프로세스, 및 (d) 데이터 스트림의 프로토콜을 특징하는 모듈들로 이루어진다.

IEEE 표준 802.3, 캐리어 검출 다중 엑서스 ·충돌 검출(CSMA/CD) 엑세스 방법 및 물리 계층 사양은 물리 계층에 대해 가장 널리 이용되는 표준들중 하나이다. 일반적으로 이서넷이라 불리는 IEEE 표준 802.3은, 꼬임쌍 케이블들 또는 이 꼬임쌍 케이블들 보다 비싼 동축 케이블들을 통한 데이터의 전송 다룬다. IEEE 표준 802.3의 10Base-T 프로토콜은 꼬임쌍 케이블들을 통한 데이터의 전송 속도를 10 Mbps로 규정한다.

도면을 참조하여, 도 1은 CPU(12)로 표시되는 워크스테이션, 퍼스널 컴퓨터, 파일 서버, 데이터 단말 장치 또는 다른 정보 처리 장치를 갖는 종래 기술의 시스템(10)이 어떻게 이서넷(22), 또는 매체 독립 인터페이스(24)로 표현되는 다른 타입의 데이터 통신 장치들에 연결되는 지를 도시한다. 도 1에서, 일반적으로 네트워크 인터페이스 제어기라고도 알려져있는 이서넷 제어기(14)는 CPU(12)와 이서넷(22)의 일력(및 출력) 라인들 사이에 위치한다. 전형적으로, 이서넷(22)의 연결은 2쌍의 꼬임쌍 구리 케이블들, 즉 10R로서 언급되는 입력쌍 및 10T로서 언급되는 출 력쌍으로 이루어진다.

이서넷 제어기(14)는 출력쌍 또는 출력 케이블로의 출력 데이터의 전송 및 입력쌍 또는 입력 케이블로부터의 입력 데이터의 수신을 제어한다. 예를 들어, 출 력 데이터는, 출력쌍 또는 출력 케이블로 제공되기 전에, 전자기 간섭을 줄이기 위해 맨체스터 엔코딩(Manchester encoding)된다. 이 맨체스터 엔코딩에 의해, 데이터 스트림의 일부는 10 MHz로 펄스화되고, 데이터 스트림의 다른 부분들은 5 MHz로 펄스화된다.

보다 많은 데이터를 보다 빨리 전송하고자 하는 끊임없는 요구 및 데이터 처리 성능을 증가시키고자 하는 요구는, 10Base-T 프로토콜에 의해 규정되는 10 Mbps의 속도 보다 상당히 더 높은 데이터 전송 속도로의 확장을 필요로 한다. 그 결과로서, 현재의 기존 시스템들의 꼬임쌍 케이블들을 통해 100 Mbps의 유효 전송 속도로 이동하는 데이터에 대처하기 위해 IEEE 표준 802.3을 확장하는 100Base-TX 프로토콜이 존재한다. 물리적 전송 매체가 100Base-TX 속도 및 이보다 늦은 10Base-T 속도 모두에서 꼬임쌍 케이블들을 통해 전송되는 데이터를 처리할 수 있는 것이 바람직한 경우가 있다. 현재, 9.6 마이크로초의 패킷간 간격(interpacket gap)을 갖는 전이중 모드 동작을 지원하기 위해서는, 내부 PCI 버스 상에서 33 MHz의 PCI 속도를 지원하고, 100 Mbps 동작을 위해 최대 25 MHz의 이서넷 속도를 지원할 필요가 있다.

이서넷 또는 매체 독립 인터페이스를 통해 서로 다른 속도로 데이터를 전송하는 것과 관련된 문제들에 부가적으로, 퍼스널 컴퓨터, 워크스테이션, 파일 서버,중계기, 데이터 단말 장치 및 이러한 다른 정보 처리 장치의 데이터 처리 성능이 변하는 것과 관련하여 문제들이 발생한다. 예를 들어, 퍼스널 컴퓨터 시스템에서, CPU(12)는 이서넷(22)을 통해 데이터를 수신 또는 전송하는 것 외에, 다른 장치 또는 임무(duty)를 처리해야 한다.

이서넷 제어기(14)는 CPU(12)에서 이서넷(22)으로의 데이터의 전송을 제어한다. 이서넷 제어기(14)가 직면하는 주요 문제들중 하나는 다양한 구성 요소들의 서로 다른 메모리 디바이스들이 서로 다른 사이즈를 갖는 다는 것이다. 예를 들어, 반도체 디바이스들은 가능한한 작게 유지할 필요가 있다. 이에 따라, 디바이스의 성능을 저하시키지 않으면서, 버스의 사이즈를 가능한한 작게 하는 것이 유리하다. 이해할 수 있는 바와 같이, 16 비트 버스는 32 비트 버스 사이즈의 절반이다. 이러한 16 비트 버스로 32 비트 버스와 같은 성능을 제공할 수 있다면, 그 부분은 16 비트 버스로 설계하는 것이 바람직하다. 또한, 버스 사이즈가 작을수록, 버스에 결함이 덜 생긴다.

도 2는 서로 다른 사이즈의 구성 요소들을 도시한다. SRAM(16)은 16 비트의 메모리 디바이스이고, 데이터 버스(20)는 16 비트의 데이터 버스이며, 그리고 PCI 버스(18)는 전형적으로 32 비트의 버스이다. 현재에도 64 비트의 PCI 버스가 존재하기는 하지만, 미래의 컴퓨터 시스템들은 모두 64 비트 PCI 버스를 표준 버스 사이즈로서 가질 것이다. SRAM(16)은 이서넷 제어기(14)에 의해 버퍼로서 이용되어, 이서넷(22)으로부터 CPU(12)로, 또는 CPU(12)로부터 이서넷(22)으로 데이터 전송시 지연이 발생하는 것을 방지한다. 이러한 지연은, 예를 들어 CPU(12)에서의 긴 대기 시간에 의해, 또는 이서넷(22) 상에서의 충돌(이에 의해, 송신국은 방금 전송한 정보를 재전송해야 한다)에 의해 야기될 수 있다. BX FIFO(26), MX FIFO(28), BR FIFO(30) 및 MR FIFO(32)와 같은 다양한 FIFO들은 다양한 구성 요소들 간의 데이터전송을 제어한다. 예를 들어, BX FIFO(26)은 PCI 버스(18)를 통해 CPU(12)로부터 데이터를 수신한 다음, 이 데이터의 포맷을 16 비트 데이터 버스(20)를 통해 SRAM(16)으로 전송될 수 있도록 32 비트에서 16 비트로 바꾼다. 또한, 정보가 SRAM(16) 내에 배치되고 MX FIFO(28) 및 BR FIFO(30) 각각에 의해 SRAM(16)으로부터 효율적으로 검색되기 위해서는, BX FIFO(26) 및 MR FIFO(32)가 어드레스들을 생성해야 한다. 상기 MX FIFO(28) 및 BR FIFO(30)는 SRAM(16)으로부터 수신되는 16 비트 포맷을 32 비트 포맷으로 바꾼다.

또한, 데이터를 수신할 필요가 있는 동시에 이서넷(22)을 통해 데이터를 전송할 필요가 있을 수 있고, 이서넷 제어기(14)가 CPU(12)로부터 데이터를 수신해야할 필요가 있는 동시에 CPU(12)로 데이터를 전송할 필요가 있을 수 있기 때문에, 이서넷 제어기(14)는 어떤 데이터가 먼저 전송 또는 수신될 것인지, 그리고 어떤 순서로 이후의 데이터가 전송 또는 수신될 것인지를 지능적으로 선택해야 할 필요가 있다.

메모리 디바이스에 저장하기 위해 데이터의 사이즈를 32 비트에서 16 비트로 효율적으로 바꾸는 방법, SRAM에 저장될 16 비트 데이터에 대한 어드레스들을 생성하는 방법, SRAM으로부터 검색하기 위해 데이터의 사이즈를 16 비트에서 32 비트로 효율적으로 바꾸고, 검색된 32 비트 데이터에 대한 어드레스들을 생성하는 방법, 및 이서넷 제어기에 의해 제어되는 다양한 기록 및 판독 기능들 사이에서 조정을 수행하는 방법이 필요하다.

EA-A-150 084호는 네트워크와 CPU 간의 데이터 전송을 제어하는 네트워크 제어기를 개시하는바, 이는 메모리, 데이터의 전송을 관리하는 FIFO, 및 이 FIFO를 제어하는 조정기(arbiter)를 포함한다. 하지만, 이는 이서넷 네트워크에 대해서는 개시하지 않는다.

발명의 상세한 설명

본 발명은 CPU를 갖는 스테이션과 네트워크 간의 데이터의 전송을 제어하는 네트워크 제어기-이 네트워크 제어기는 메모리, 데이터 전송을 관리하는 적어도 1개의 FIFO 및 상기 각 FIFO를 제어하는 조정기(42)(본 발명의 일 실시예에서는 SRAM 제어기이다)를 포함한다-를 제공하는바,

상기 네트워크는 이서넷 네트워크이고;

상기 적어도 1개의 FIFO는:

상기 CPU로부터 상기 메모리로의 CPU 데이터의 전송을 관리하는 제 1 FIFO(26), 및 상기 제 1 FIFO에 결합되어 상기 CPU 데이터를 제 1 비트 사이즈에서 제 2 비트 사이즈로 변환하고 상기 제 2 비트 사이즈의 상기 CPU 데이터를 상기 메모리에 기록하기 위한 어드레스들을 생성하는 제 1 논리(34)와;

상기 메모리로부터 상기 이서넷으로의 상기 CPU 데이터의 전송을 관리하는 제 2 FIFO(28), 및 상기 제 2 FIFO에 결합되어 상기 메모리 내의 상기 CPU 데이터를 판독하기 위한 어드레스들을 생성하고 상기 데이터를 상기 제 2 비트 사이즈에서 제 1 비트 사이즈로 변환하는 제 2 논리를 포함하며;

상기 조정기는 상기 각 FIFO를 제어하고, 상기 각 FIFO를 통한 전송을 선택된 수의 바이트들로 제한하며, 상기 선택된 수의 바이트들이 전송된 후 상기 각FIFO를 폴링하여 전송 요구가 미결인지의 여부를 결정하는 것을 특징으로 한다.

이서넷 제어기는 스테이션 CPU에서 버퍼 메모리로, 버퍼 메모리에서 이서넷으로, 이서넷에서 버퍼 메모리로, 그리고 버퍼 메모리에서 스테이션 CPU로의 데이터 전송을 관리하는 관련된 논리 형태를 갖는 4개의 FIFO들을 포함한다. 이서넷 제어기는 어떤 FIFO가 데이터 전송에 대해 우선 순위를 갖게 될 것인 지를 조정하는 조정기를 포함한다.

각 FIFO와 관련된 논리는 데이터의 비트 사이즈를 제 1 비트 사이즈로부터 제 2 비트 사이즈로 변환하고, 버퍼 메모리로의 기록 또는 이 버퍼 메모리로부터의 판독을 위한 어드레스들을 생성한다. 이 논리는 또한 16 비트 어드레스를 8 비트 어드레스 버스를 통해 전송될 수 있도록 2개의 8 비트 부분들로 변환하며, 또한 이2개의 8 비트 부분들을 원래의 16 비트 어드레스로 변환한다.

조정기는 각 FIFO를 요구 마다 32 바이트 전송으로 제한하고, 각 32 바이트전송 후에 다른 FIFO들을 폴링하여 미결의 전송 요구가 있는 지의 여부를 결정한다. 이 조정기는 조정 알고리즘에 의해 요구들을 허가한다.

각 FIFO는 이서넷 제어기의 성능을 최대화할 수 있도록 선택된 사이즈를 갖는다.

본 발명은 첨부 도면을 참조하여 하기 설명되는 상세한 설명으로부터 명확해질 것이다. 하기의 설명으로부터 당업자에게 자명한 바와 같이, 본 발명을 수행하기 위한 최상의 방법으로서 고려되는 바람직한 실시예가 예시적으로 도시되어 설명된다. 알 수 있는 바와 같이, 본 발명의 범위를 벗어나지 않으면서, 다른 실시예들이 가능하며, 다양하고 명백한 양상들로 변형이 이루어질 수 있다. 따라서, 도면들 및 상세한 설명은 제한적인 것이 아닌 예시적인 것으로서 간주된다.

도면의 간단한 설명

명세서에 포함되어 본 발명의 일부를 형성하는 첨부 도면들은 상세한 설명과 함께 본 발명의 원리들을 설명한다.

도 1은 이서넷 및 매체 독립 인터페이스에 연결된 이서넷 제어기, 및 CPU를 갖는 종래 기술에 의한 시스템의 개관도이다.

도 2는 본 발명에 의해 교시되는 시스템을 도시한다.

도 3은 이서넷 제어기의 상세도이다.

도 4는 판독 및 기록 액세스들의 순서를 조정하기 위해 이서넷 제어기에 의해 이용되는 조정 알고리즘을 도시한다.

도 5는 서로 다른 비트 사이즈의 데이터가 어떻게 서로 다른 사이즈의 메모리 위치들에 위치되는 지를 도시한다.

도 6은 SRAM 버퍼 메모리에 데이터를 기록하기 위한 어드레스를 생성하는 방법을 도시한다.

도 7은 SRAM 버퍼 메모리로부터 데이터를 검색하기 위한 어드레스를 생성하는 방법을 도시한다.

실시예

도 1은 이서넷(22) 및 매체 독립 인터페이스(24)에 연결된 이서넷 제어기(14), 및 CPU(12)를 갖는 종래 기술에 의한 시스템(10)의 개관도이다. CPU(12)는 버스(17)에 의해 이서넷 제어기(14)에 연결된다.

도 2는 본 발명에 따른 시스템(11)을 도시한다. 도면들에서, 동일한 부호는 동일한 구성 요소를 나타낸다. 이서넷 제어기(14)의 일부가 도 2에 도시된다. 이해될 사항으로서, 이서넷 제어기(14)는 본 발명에 의해 교시되는 기능 이외의 다른 많은 기능들을 갖지만, 본원에서는 본 발명과 관련된 기능들에 대해서만 설명한다.

이서넷 제어기(14)의 도시된 부분의 기능은 이서넷(22) 그리고/또는 매체 독립 인터페이스(24)로/로부터의 데이터의 전송을 관리하는 것이다. 이서넷 제어기(14)는 CPU(12) 또는 이서넷(22) 또는 매체 독립 인터페이스(24)로/로부터의 데이터의 전송이 느려지는 것을 막기 위해, SRAM(16)을 버퍼로서 이용하여 데이터의 전송을 관리한다. 데이터의 전송이 느려지는 데에는 여러 가지 이유가 있는바, 예를들어 CPU(12)의 대기 시간이 길 수 있고, CPU(12)가 다른 인터럽트들로부터 자유(free)롭게 될 때 까지 이서넷(22)으로부터의 데이터의 전송이 중지될 수 있기 때문에 데이터의 전송이 느려진다. 역으로, CPU(12)가 이서넷(22)를 통해 데이터를 전송하고자 시도할 때 이 이서넷(22)은 사용중일 수 있기 때문에, CPU(12)로부터의 데이터는 이서넷(22)의 사용이 끝날 때까지 정지 또는 홀드된다.

정보의 전송 또는 수신을 완료할 수 없음으로써 야기되는 문제들을 해결하기위해, 이서넷 제어기(14)는 4개의 FIFO들을 갖는바, 각 FIFO는 이서넷 제어기(14)의 성능을 최대화할 수 있도록 선택된 사이즈를 갖는다. BX FIFO(26)는 180 바이트의 FIFO이고, MX FIFO(28)는 112 바이트의 FIFO이며, BR FIFO(30)는 160 바이트의FIFO이고, MR FIFO(32)는 108 바이트의 FIFO이다. BX FIFO(26) 및 BR FIFO(30)는 이서넷 제어기(14)의 BUS측에 있고, MX FIFO(28) 및 MR FIFO(32)는 이서넷 제어기(14)의 MAC(매체 액세스 제어)측에 있다. FIFO들 각각은 입력 기능 또는 출력 기능을 관리한다. BX FIFO(26)는 CPU(12)로부터 SRAM(16)으로의 데이터의 전송을 관리한다. MX FIFO(28)는 SRAM(16)으로부터 이서넷(22) 또는 매체 독립 인터페이스(24)로의 데이터의 전송을 관리한다. 유사하게, MR FIFO(32)는 이서넷(22) 또는 매체 독립 인터페이스(24)로부터의 데이터의 전송을 관리한다.

도 3은 이서넷 제어기(14)를 보다 상세히 도시한다. 도 2와 관련하여 설명된4개의 FIFO들 각각은 논리 블록을 포함하거나, 또는 이 논리 블록에 결합된다. BX FIFO(26)는 BX 논리(34)에 결합되는바, 이 BX 논리(34)는 SRAM(16)에 대한 어드레스들을 생성하고, CPU(12)로부터 16 비트 데이터 경로(20)를 통해 SRAM(16)으로 전송될 32 비트 데이터를 준비한다. MX FIFO(28)는 MX 논리(36)에 결합되는바, 이 MX논리(36)는 SRAM(16)으로부터 검색될 16 비트 데이터에 대한 어드레스들을 생성하고, SRAM(16) 내의 16 비트 데이터를 판독하고, SRAM(16)로부터 검색된 16 비트 데이터를 32 비트로 변환한 다음, 이 32 비트 데이터를 MX FIFO(28)에 기록한다. MR FIFO(32)는 MR 논리(38)에 결합되는바, 이 MR 논리(38)는 이서넷(22)으로부터 MR FIFO(32)에 의해 수신된 데이터를 16 비트 사이즈로 변환하고, SRAM(16)에 기록될16 비트 데이터에 대한 어드레스들을 생성한다. BR FIFO(30)는 BR 논리(40)에 결합되는바, 이 BR 논리(40)는 이서넷(22)으로부터 수신된 SRAM(16) 내의 데이터를 판독하여, 이 데이터를 16 비트 사이즈에서 32 비트 사이즈로 바꾼 다음, 이 32 비트 사이즈의 데이터를 BR FIFO(30)에 기록한다. 이 32 비트 사이즈의 데이터는 BR FIFO(30)로부터 CPU(12)로 전송된다.

SRAM 제어기(42)는 4개의 FIFO들을 제어하는바, 충돌이 발생하는 경우, 즉 1개 이상의 FIFO가 데이터를 전송하기 위해 버스에 대한 액세스를 요구하면, 이 SRAM 제어기(42)는 어떤 순서로 어떤 FIFO가 우선 순위를 가질 것인 지를 결정하는 소정의 알고리즘에 따라 이러한 상황을 조정한다. 이 알고리즘에 대해서는 하기에서 설명한다.

도 3에 도시된 시스템의 전체적인 동작은 다음과 같다. 이서넷 제어기(14)는CPU(12)로부터 이서넷(22)으로의 데이터의 전송에 대한 전체 동작 및 이서넷(22)으로부터 CPU(12)로의 데이터의 전송을 관리한다. 상기 설명된 바와 같이, 이서넷(22)으로/으로부터의 데이터 전송에 있어서의 주요 문제들중 하나는 이러한 임의 데이터의 전송에 영향을 줄 수 있는 여러 가지 요인들이 있다는 것이다. 한 요인은 서로 다른 속도들에 대처해야한다는 것이다. PCI 버스(18)는 33 MHz의 속도로 동작하고, 이서넷(22)은 9.6 마이크로초의 패킷간 간격을 갖는 전이중 동작을 지원하는100 Mbps의 동작에 대해 최대 25 MHz로 동작한다. 이서넷 제어기(14)의 주요 기능 들중 하나는 이러한 데이터의 수신 또는 전송시의 모든 지연을 최소화하는 방식으로 다양한 구성 요소들 간의 데이터의 전송을 관리하는 것이다. 예를 들어, CPU(12)(미도시)가 이서넷(22)으로 데이터를 전송하기를 원하면, 이 CPU(12)는 BX FIFO(26)와 통신하여 이 BX FIFO(26)에 데이터를 전송한다. BX FIFO(26)는 BX 요구라인(44)을 통해 SRAM 제어기(42)로부터 버스에 대한 액세스를 요구한다. BX 논리(34)가 BX 제어라인(46)을 통해 SRAM 제어기(42)로부터 허가(grant)를 수신하면, 이 BX 논리(34)는 CPU(12)로부터 수신된 32 비트 사이즈의 데이터를 16 비트 사이즈의 데이터로 변경한 다음, 이 16 비트 포맷의 데이터를 저장하기 위해 SRAM(16)에 대한 어드레스들을 생성한다. 변환된 데이터는 16 비트 버스(20)를 통해SRAM(16)으로 전송되어 이 내에 기록된다. BX 논리(34)는 BX FIFO(26)로부터SRAM(16)으로의 데이터의 흐름을 제어하는 상태 머신(state machine)으로서, SRAM제어기(42)로부터 허가 신호를 얻으면 동작한다. BX FIFO(26)는 32 비트 폭을 갖고 데이터 버스(20)는 16 비트 폭을 갖는다. 이에 따라, 각 사이를 마다 16 비트 사이즈의 데이터 버스 상에서 단지 16 비트 만을 기록할 수 있기 때문에, BX 논리(34)가 16 비트 사이즈의 데이터를 SRAM(16)에 기록하기 위해서는 2개의 사이클이 필요하다. BX FIFO(26)가 180 바이트의 사이즈를 가질 때, 이 BX FIFO(26)를 통한 데이터 전송이 최대화되는 것으로, 즉 BX FIFO(26)를 통한 데이터 전송시의 모든 지연이 최소화되는 것으로 밝혀졌다. 하기 설명되는 바와 같이, SRAM 제어기(42)는 시스템(11)의 성능을 최대화하는 조정 알고리즘에 따라, BX FIFO(26)를 포함하는 모든 FIFO들을 통한 데이터의 수신 및 전송을 관리한다.

SRAM(16)의 사이즈는 선택가능한바, 이 실시예에서는 64 킬로 바이트로 선택되었지만, 적어도 128 킬로 바이트까지 확장할 수 있다. 따라서, 각 메모리 위치를 액세스하기 위해서는 16 비트의 어드레스를 가질 필요가 있다. 그러나, 이서넷 제어기(14)는 공간을 절약하고 보다 비용 효율적으로 하기 위해 단지 8비트의 어드레스 포트들 만을 갖는다. 이러한 이유로, 16 비트 어드레스는 상위 8 비트 부분 및 하위 8 비트 부분으로 분할되어 8 비트 어드레스 버스를 통해 전송된 다음, 이서넷제어기(14)의 외부에서 16 비트 어드레스로 재조립되어야 한다. 어드레스 생성 및 전송에 대한 상세한 사항은 도 6 및 7을 참조하여 하기에서 설명된다.

이서넷 제어기(14)는 MX FIFO(28)를 통해 이서넷 네트워크와 항상 통신하고 있어, 이용되는 프로토콜의 형태에 따라, 시스템은 언제 이서넷(22)을 통해 데이터가 전송될 수 있는 지를 결정한다. 이서넷(22)을 통해 데이터가 전송될 수 있게 되면, MX FIFO(28)는 MX 요구 라인(47)을 통해 SRAM 제어기(42)로부터 데이터 버스(20)에 대한 액세스를 요구한다. 액세스가 허가되어 MX 제어 라인(48)을 통해 MX 논리(36)로 전송되면, SRAM(16)에 임시적으로 기록되었거나 또는 MX FIFO(28)에 있는 데이터는 CPU(12)로부터 BX FIFO(26)에 의해 수신된 원래의 32 비트 형태로 변환되어, MX FIFO(28)에 의해 이서넷(22)을 통해 전송된다. MX FIFO(28)는 SRAM(16)으로부터 MX FIFO(28)로의 데이터의 흐름을 제어하는 상태 머신으로서, SRAM 제어기(42)로부터 허가 신호를 얻으면 동작한다. MX FIFO(28)는 32 비트 폭을 갖고 데이터 버스(20)는 16 비트 폭을 갖는다. 따라서, MX 논리(36)가 16 비트 데이트를 2번 판독한 다음 논리를 이용하여 이들을 이중 워드(32 비트)로 조립한 후, 기록 신호가 MX FIFO(28)에 제공된다. 이는 MX FIFO(28)에 대한 기록은 1 사이클 걸러서 이루어지고, SRAM(16)으로부터의 판독은 매 사이클 마다 이루어진다는 것을 의미한다. MX FIFO(28)는 SRAM(16)으로부터의 판독 및 MX FIFO(28)에 대한 기록을 제어한다.

MR FIFO(32)가 이서넷으로부터 수신될 데이터가 존재하는 지를 감지하면, 이MR FIFO(32)는 MR 요구 라인(50)을 통해 SRAM 제어기(42)로부터 데이터 버스(20)에대한 액세스를 요구한다. SRAM 제어기(42)가 MR 제어 라인(52)을 통해 액세스를 허가하면, MR FIFO(32)에 의해 수신되는 데이터는 MR 논리(38)에 의해 16 비트 포맷으로 변환된 다음 16 비트 데이터 버스(20)를 통해 SRAM(16)으로 전송된다. MR 논리(38)의 동작은 상기 설명된 BX 논리(34)의 동작과 유사하다.

CPU가 이서넷으로부터 수신된 데이터를 수신할 준비가 되면, BR FIFO(30)는BR 요구 라인(54)을 통해 SRAM 제어기(42)로부터 데이터 버스(20)에 대한 액세스를 요구한다. 액세스가 허가되면, SRAM 제어기(42)는 BR 논리(40)와 통신하는바, 이 BR 논리(40)는 SRAM(16)으로부터 데이터를 판독하여, 16 비트 데이터 포맷으로 저장된 데이터를 32 비트 데이터 포맷으로 변환한 다음 BR FIFO(30)로 전송한다. BR FIFO(30)는 이 데이터를 CPU로 전송한다. BR 논리(40)의 동작은 상기 설명한 MX 논리(36)의 동작과 유사하다.

도 4는, 동시에 다수의 요구들이 있는 경우, SRAM 제어기(42)(도 4)가 어떤 순서로 버스에 대한 요구들을 처리하는 지를 나타내는 조정 알고리즘(56)을 도시한다. SRAM 제어기(42)는 도 4에 보인 알고리즘 방식을 이용하여 4개의 모든 논리 블록들로부터의 요구들을 처리한다.

또한, 본 발명은 이서넷 제어기(42)의 구성 요소들 간에 전송되어야 하는 최적수의 바이트들이 존재한다는 것을 교시한다. 바이트들의 최적수는 약 32개인 것으로 결정되었다. 이는, 예를 들어 BX FIFO(26)에 의해 버스에 대한 요구가 이루어지기 전에, BX FIFO(26) 내에서 32 바이트의 데이터가 조립되어야 함을 의미한다(유일한 예외는, 전송될 마지막 바이트가 BX FIFO(26) 내에서 검출되는 때이다). 또한, 한 구성 요소에서 다음 구성 요소로 전송되는 바이트들의 최적의 수는 32개의 바이트들인 것으로 결정되었다. 한계(이 경우 32 바이트)에 도달하면, 시스템은 전송 또는 수신을 중단한 다음, 다른 기능들을 폴링하여 전송 또는 수신될 필요가 있는 다른 정보가 있는 지를 결정한다.

도 4를 다시 참조하여, 다수의 요구들이 동시에 존재하고 시스템이 아이들(idle) 상태(58)에 있는 제 1 경우, 이 시스템은 먼저 수신측(60)을 주목한 다음, MAC측(62)의 수신에 대한 액세스를 처음으로 허가한다. 62에서 어떠한 미결 요구도 없으면 시스템은 전송측(64)으로 가고, 버스 전송(66)에서 미결 요구가 있으면 버스 전송에 대한 액세스를 허가한다. 버스 전송에서 어떠한 미결 요구도 없으면, 시스템은 수신측(60)으로 가서, 버스 수신(68)에 대한 액세스를 허가한다. 68에서 어떠한 미결 요구도 없으면, 시스템은 전송측(64)을 주목한 다음, MAC 전송(70)을 주목한다.

다른 기능들을 폴링하여 데이터 버스에 대한 미결 요구들이 있는 지의 여부를 결정하기 위해 시스템이 정지하는 경우, 이 시스템은 상기 설명한 것과 같은 순서로 미결 요구를 허가한다. 도 5는 바이트 사이즈의 데이터가 시스템의 다양한 구성 요소들에 어떻게 저장되는 지를 도시한다. 72 및 74로 표시된 위치들에는, FIFO에 저장된 5개의 데이터 바이트들(D0 내지 D4)을 갖는 BX FIFO(26)의 작은 부분이 도시되어 있다. 또한, 76으로 표시된 위치에는, 8 비트 바이트들로 저장된 32 비트의 STATUS 데이터(S0 내지 S3)가 도시되어 있다. 78로 표시된 메모리 위치에는, 8비트 바이트들로 저장된 32 비트의 DESCRIPTOR 데이터(DA0 내지 DA3)가 도시되어 있다. BX 논리(34)는 BX FIFO(26)로부터 32 비트 데이터를 판독하여, 80에서 16 비트 사이즈로 변환한 다음, 이 16 비트 데이터를 16 비트 데이터 버스(20)로 전송한다. 이 16 비트 데이터는 도시된 바와 같이 SRAM(16) 내에 기록된다. 제 1 메모리위치(82)는 시스템에 의해 예약되고, 2개의 8 비트 바이트들(EOP0 및 EOP1)을 포함하는바, 이들은 어디에 패킷의 끝이 위치하는지, 즉 어느 메모리 위치에 데이터 바이트(D4)가 위치하는 지에 대한 정보를 나타낸다. 제 2 메모리 위치(84) 또한 시스템에 의해 예약되고, 2개의 8비트 바이트들을 포함하는바, 이 예에서 이들은 상위바이트 위치에서는 HEX10이고 하위 바이트 위치에서는 HEX00이다. 메모리 위치들(86)에는 데이터(D0 내지 D4)가 있고, 계속하여 메모리 위치들(88 및 90)에는 2개의 16 비트 데이터가 뒤를 따르는바, 이 메모리 위치들(88 및 90)에는 4개의 8 비트 바이트의 STATUS 데이터들(S0 내지 S3)이 위치한다. 이러한 STAWS 데이터 정보 메모리 위치들에 이어서, 메모리 위치들(92 및 94)에 위치하는 4개의 8비트 바이트의 DESCRIPTOR 데이터(DA0 내지 DA3)가 뒤를 따른다.

MX FIFO(28)가 상기 설명한 바와 같이 데이터 버스(20)에 대한 액세스를 허가받으면, SRAM(16)으로부터 16 비트 데이터가 판독되어, 96에서 32 비트 데이터로 변환된 다음, MX FIFO(28)로 전송된다.

상기에서는, 이서넷 제어기(14)의 버스측으로부터 MAC측으로, 즉 CPU(12)(미도시)로부터 BX FIFO(26), SRAM(16), MX FIFO(28) 및 이서넷(22)으로의 데이터의 전송에 대해 설명했다. 이해될 사항으로서, MR FIFO(32)에서 데이터가 수신되어, SRAM(16)으로 전송되고, BR FIFO(30)에 의해 판독된 다음, CPU(12)로 전송될 때에도 동일한 설명이 적용될 수 있다.

도 6은 데이터가 SRAM(16) 내에 기록되고 있을 때 어드레스를 생성하는 방법을 도시한다. BX 논리(34)의 어드레스 생성부는 점선(98)에 있고, MR 논리(38)(도3)의 어드레스 생성부 또한 이 내에 있다. SRAM(16)에 대한 어드레스들을 생성하는 방법은, BX 기록 포인터(100)가 정보의 다음 패킷이 기록될 메모리 위치를 지시하는 것으로부터 시작된다. 패킷이 메모리 내에 기록되기 시작할 때, 개시 메모리 위치가 개시 패킷 포인터(102)에 저장된다. 따라서, 개시 패킷 포인터(102)는 BX 논리(34)가 패킷을 기록하기 시작하는 SRAM(16) 내의 메모리 위치의 어드레스를 홀드한다. 개시 메모리의 위치는 패킷의 끝의 메모리 위치와 함께 기록된다. 16 비트 어드레스는 8 비트 어드레스 버스를 통해 전송되어야하기 때문에, 이 16 비트 어드레스는 2개의 8 비트 부분들로 분할되어야 한다. 데이터가 기록될 때 어드레스들을 생성하기 위해서는, 각 바이트가 기록될 때 마다 어드레스의 하위 8 비트 부분을 증분(increment)시킬 필요가 있다. 상위 8 비트 부분은, 페이지가 기록될 때 마다, 즉 하위 8 비트가 HEXFF일 때 증분된다. 따라서, BX 기록 포인터(100)는 바이트가 SRAM(16) 내에 기록될 때 마다 BX 기록 포인터(100)를 증분시키는 하위 증분 입력(increment lower input)(104) 및 새로운 페이지가 시작될 때 마다, 즉 하위 어드레스 바이트가 HEXFF 값에 이른 후 BX 기록 포인터(100)를 증분시키는 상위 증분 입력(increment upper input)(106)을 갖는다. BX 기록 포인터(100)는 바이트가 방금 기록되었던 메모리 위치의 16 비트 어드레스를 포함한다. 16 비트 어드레스는 8비트 어드레스 버스를 통해 전송되기 때문에, MUX(108)는 16 비트 어드레스를 110으로 표현된 8 비트 버스를 통해 SRAM(16)으로 전송하기 위해 상위 8 비트 부분 및 하위 8 비트 부분으로 재포맷하는바, 도 6에는 SRAM(16)의 일부가 점선(112)으로 도시된다. 114에서 SRAM(16)은 2개의 8 비트 어드레스 부분들을 116으로 나타낸 16비트 어드레스(116)로 변환시킨다. 패킷의 끝이 SRAM(16) 내에 기록되면, 이 패킷 의 끝이 기록된 메모리 위치는, 개시 패킷 포인터(102)에 기록된 개시 메모리 위치내에 기록된다.

도 7은 SRAM(16)으로부터 데이터를 판독하기 위한 판독 어드레스를 생성하는MX 논리(36)의 점선(124) 내에 표시된 판독 어드레스 생성부를 도시한다. (점선(127) 내에 도시된) SRAM(16)의 일부의 판독 포인터(126)는 방금 판독된 바이트가 위치되는 16 비트 메모리 위치를 지시한다. 이 16 비트 메모리 위치는, MUX(128)에의해, 130으로 나타낸 8 비트 어드레스 버스 상에 배치되는 2개의 8 비트 부분들로 변환된다. 이러한 2개의 8 비트 어드레스 부분들은 MX 논리(36)에 의해 수신되어,플립 플롭(132)에 의해 16비트 어드레스(134)로 재조립된다.

본 발명의 바람직한 실시예에 대한 상기의 설명은 단지 예시의 목적으로 제시된 것으로서, 본 발명을 개시된 형태로 한정하고자 하는 것은 아니다. 상기 설명에 비추어 명백한 수정들 및 변형들이 이루어질 수 있다. 상기 설명된 실시예는 본발명의 원리들 및 실제 응용에 대한 최상의 예를 제공함으로써, 당업자로 하여금 고려되는 특정 용도에 따라 적절히 변형을 수행하여 본 발명을 다양한 실시예들에 이용할 수 있게 한다. 이러한 모든 변형 및 수정은 첨부된 청구항들에 의해 정의되는 본 발명의 범위 내에 포함된다.

(57) 청구의 범위

청구항 1.

CPU를 갖는 스테이션과 네트워크 간의 데이터의 전송을 제어하는 네트워크 제어기-이 네트워크 제어기는 메모리, 데이터 전송을 관리하는 적어도 1개의 FIFO및 상기 각 FIFO를 제어하는 조정기(42)를 포함한다-에 있어서,

상기 네트워크는 이서넷 네트워크이고;

상기 적어도 1개의 FIFO는:

상기 CPU로부터 상기 메모리로의 CPU 데이터의 전송을 관리하는 제 1 FIFO(26), 및 상기 제 1 FIFO에 결합되어 상기 CPU 데이터를 제 1 비트 사이즈에서 제 2 비트 사이즈로 변환하고 상기 제 2 비트 사이즈의 상기 CPU 데이터를 상기 메모리에 기록하기 위한 어드레스들을 생성하는 제 1 논리(34)와;

상기 메모리로부터 상기 이서넷으로의 상기 CPU 데이터의 전송을 관리하는 제 2 FIFO(28), 및 상기 제 2 FIFO에 결합되어 상기 메모리 내의 상기 CPU 데이터를 판독하기 위한 어드레스들을 생성하고 상기 데이터를 상기 제 2 비트 사이즈에서 제 1 비트 사이즈로 변환하는 제 2 논리를 포함하며;

상기 조정기는 상기 각 FIFO를 제어하고, 상기 각 FIFO를 통한 전송을 선택된 수의 바이트들로 제한하며, 상기 선택된 수의 바이트들이 전송된 후 상기 각FIFO를 폴링하여 전송 요구가 미결인지의 여부를 결정하는 것을 특징으로 하는 네트워크 제어기.

청구항 2.

제 1 항에 있어서,

상기 이서넷으로부터 상기 메모리로의 이서넷 데이터의 전송을 관리하는 제 3 FIFO(32)와; 그리고

상기 메모리로부터 상기 CPU로의 상기 이서넷 데이터의 전송을 관리하는 제 4 FIFO(30)를 더 포함하며;

상기 조정기가 상기 제 3 FIFO 및 상기 제 4 FIFO를 제어하는 것을 특징으로 하는 네트워크 제어기.

청구항 3.

제 2 항에 있어서,

상기 제 3 FIFO에 결합되어, 상기 이서넷 데이터를 상기 제 1 비트 사이즈로부터 상기 제 2 비트 사이즈로 변환하고, 상기 메모리에 상기 제 2 비트 사이즈의 상기 이서넷 데이터를 기록하기 위한 어드레스들을 생성하는 제 3 논리(38)와; 그리고

상기 제 4 FIFO에 결합되어, 상기 메모리 내의 상기 이서넷 데이터를 판독하기 위한 어드레스들을 생성하고 상기 이서넷 데이터를 상기 제 2 비트 사이즈로부터 상기 제 1 비트 사이즈로 변환하는 제 4 논리(40)를 더 포함하는 것을 특징으로 하는 네트워크 제어기.

청구항 4.

제 1 항 내지 제 3 항 중의 어느 항에 있어서,

상기 선택된 수의 바이트들은 32개의 바이트들인 것을 특징으로 하는 네트워크 제어기.

청구항 5.

제 1 항에 있어서,

상기 조정기는 조정 알고리즘에 따라 상기 FIFO들로부터의 요구들을 허가하는 것을 특징으로 하는 네트워크 제어기.

청구항 6.

제 5 항에 있어서,

상기 조정 알고리즘은 제 1 레벨에서 수신 FIFO의 액세스에 우선 순위를 주고, 수신 FIFO들 사이에서는 상기 이서넷 제어기의 매체 액세스측의 미결의 요구를 갖는 수신 FIFO에 우선 순위를 주며, 그리고 상기 제 1 레벨에서 전송 FIFO에 제 2의 우선 순위를 주고, 전송 FIFO들 사이에서는 상기 이서넷 제어기의 버스측의 미결의 요구를 갖는 전송 FIFO에 우선 순위를 주는 것을 특징으로 하는 네트워크 제어기.

청구항 7.

제 1 항에 있어서,

상기 제 1 FIFO는 제 1의 선택된 사이즈를 갖는 것을 특징으로 하는 네트워크 제어기.

청구항 8.

제 7 항에 있어서,

상기 제 1의 선택된 사이즈는 180 바이트인 것을 특징으로 하는 네트워크 제어기.

청구항 9.

제 1 항에 있어서,

상기 제 2 FIFO는 제 2의 선택된 사이즈를 갖는 것을 특징으로 하는 네트워크 제어기.

청구항 10.

제 9 항에 있어서,

상기 제 2의 선택된 사이즈는 112 바이트인 것을 특징으로 하는 네트워크 제어기.

청구항 11.

제 2 항에 있어서,

상기 제 3 FIFO는 제 3의 선택된 사이즈를 갖는 것을 특징으로 하는 네트워크 제어기.

청구항 12.

제 11 항에 있어서,

상기 제 3의 선택된 사이즈는 108 바이트인 것을 특징으로 하는 네트워크 제어기.

청구항 13.

제 2 항에 있어서,

상기 제 4 FIFO는 제 4의 선택된 사이즈를 갖는 것을 특징으로 하는 네트워크 제어기.

청구항 14.

제 13 항에 있어서,

상기 제 4의 선택된 사이즈는 160 바이트인 것을 특징으로 하는 네트워크 제어기.

청구항 15.

제 1 항에 있어서,

16 비트 어드레스를 8 비트 어드레스 버스를 통해 전송하기 위해 제 1의 8 비트 부분 및 제 2의 8 비트 부분으로 재포맷하는 제 1 수단을 더 포함하는 것을 특징으로 하는 네트워크 제어기.

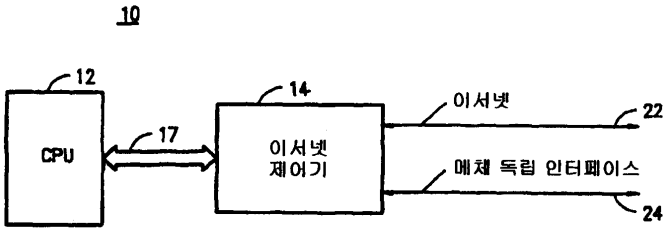
청구항 16.

제 15 항에 있어서,

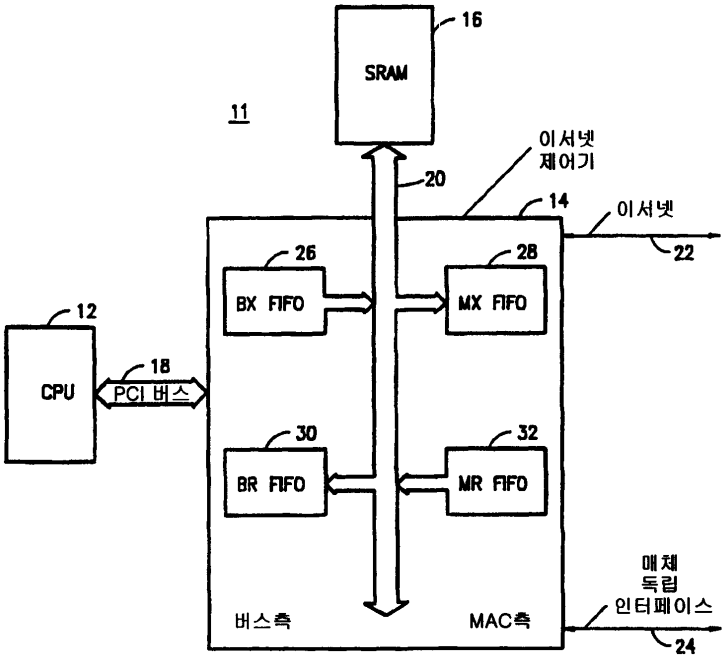
상기 8 비트 어드레스 버스를 통해 전송된 후 상기 제 1, 2의 8 비트 어드레스 부분들을 상기 16 비트 어드레스로 재포맷하는 제 2 수단을 더 포함하는 것을 특징으로 하는 네트워크 제어기.

도면

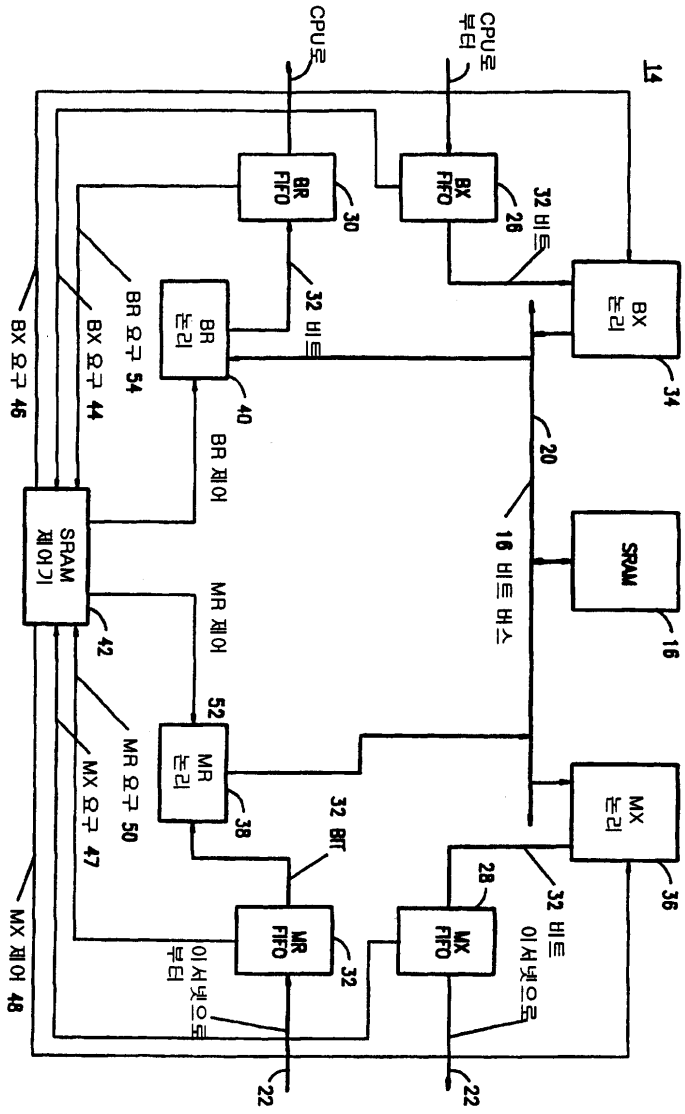
도면1


10
12
CPU
17
14
이서넷
제어기
이서넷
22
매체 독립 인터페이스
24

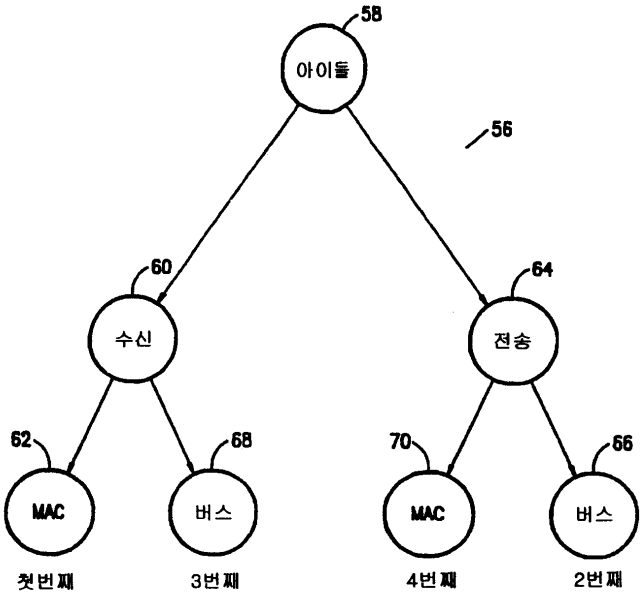
도면2


16
SRAM
11
이서넷
제어기
20
14
이서넷
22
26
BX FIFO
28
MX FIFO
12
CPU
18
PCI 버스
30
BR FIFO
32
MR FIFO
매체
독립
인터페이스
버스측
MAC측
24

도면3

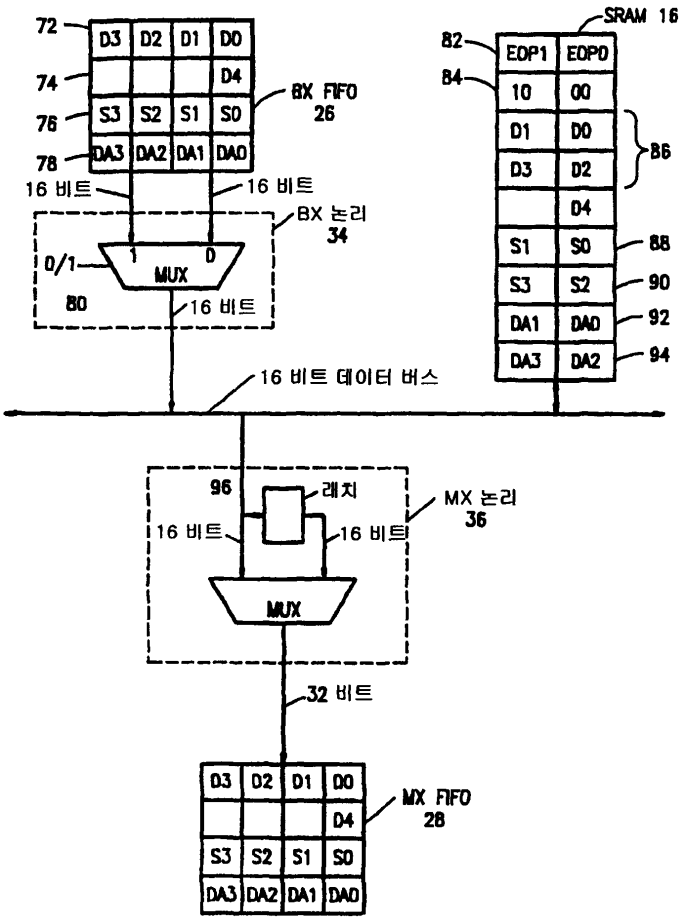

CPU로
CPU로부터
14
BR FIFO
30
BX FIFO
28
32 비트
BX 논리
34
BR 논리
32 비트
BR 요구 54
BX 요구 44
BX 요구 46
40
BR 제어
20
16 비트 버스
SRAM
16
SRAM 제어기
42
MR 제어
52
MR 논리
38
MR 요구 50
MX 요구 47
32 BIT
MX 논리
36
28
32 비트
이서넷으로
MR FIFO
32
MX FIFO
이서넷으로부터
MX 제어 48
22
22

도면4

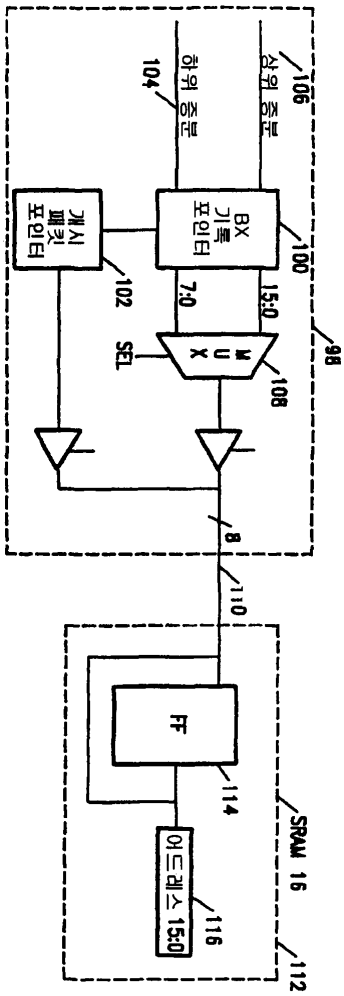


조정 알고리즘

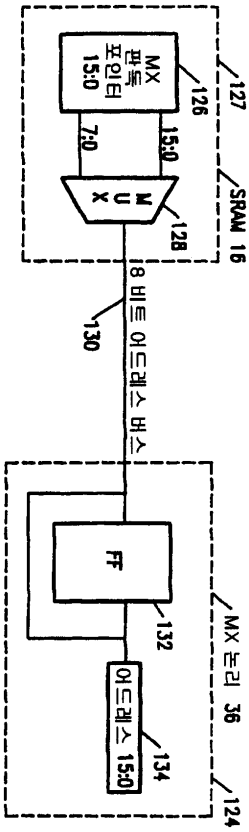
도면5



도면6


상위 출력
106
하위 출력
104
BX 기록 포인터
100
게시 패킷 포인터
102
7:0
15:0
SEL
MUX
108
98
8
110
FF
114
SRAM 16
어드레스 15:0
116
112

도면7


MX
포인터
15:0
126
127
15:0
7:0
M
U
X
128
SRAM 16
8 비트 어드레스 버스
130
FF
132
MX 논리 36
어드레스 15:0
134
124