(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.[6]
G06F 13/12

(45) 공고일자 2005년06월22일
(11) 등록번호 10-0466903
(24) 등록일자 2005년01월10일

(21) 출원번호 10-1998-0010672
(22) 출원일자 1998년03월27일
(65) 공개번호 10-1998-0080759
(43) 공개일자 1998년11월25일

(30) 우선권주장 97-074931 1997년03월27일 일본(JP)

(73) 특허권자 소니 컴퓨터 엔터테인먼트 인코포레이티드
일본국 도쿄도 미나도구 미나미아오야마 2쵸메 6반 21고

(72) 발명자 스즈오키 마사카즈
일본 도쿄도 시나가와쿠 기다시나가와 6-7-35 소니(주)내

(74) 대리인 정상구
이병호
신현문
이범래

심사관 : 임영희

(54) 정보처리장치및정보처리방법

요약

정보 처리 장치에 있어서, 우선 순위들이 복수의 중앙 처리 장치(CPU)에 할당되며 CPU는 드로잉 명령의 그 각각의 디스플레이 리스트를 우선 순위에 따라 드로잉 장치에 전송한다. 그러한 체계에 있어서, 매스터 CPU(기하 서브시스템(0))가 디스플레이 리스트(리스트 #0-1)를 생성하는 중이고 드로잉 장치(렌더링 시스템)가 아이들 상태에 있을 때, 드로잉 장치를 액세스하는 권리는 슬레이브 CPU(기하 서브시스템(1))으로 넘겨지고, 슬레이브 CPU로 하여금 그에 의해 발생된 디스플레이 리스트(리스트 1-1)를 만약 있다면 드로잉 장치에 공급할 수 있게 한다. 디스플레이 리스트(리스트 #1-1)를 수신하여, 드로잉 장치는 리스트(#1-1)에 따라서 드로잉 처리를 시작한다. 매스터 CPU가 디스플레이 리스트(리스트 #0-1)의 생성을 완료함에 따라, 슬레이브 CPU는 드로잉 장치에 액세스하는 권리를 매스터 CPU에 복귀시키며, 매스터 CPU로 하여금 리스트(#0-1)를 드로잉 장치에 공급할 수 있게 한다. 디스플레이 리스트(리스트 #0-1)를 수신하여, 드로잉 장치는 리스트(#0-1)에 따라서 드로잉 처리를 시작한다. 그후, 매스터 CPU가 디스플레이 리스트를 생성하는 중이고 드로잉 장치가 아이들 상태에 있을 때, 드로잉 장치를 액세스하는 권리는 슬레이브 CPU(기하 서브시스템(1))에 넘겨지고, 슬레이브 CPU로 하여금 그에 의해 발생된 또다른 디스플레이 리스트를 만일 있다면 동일한 방법으로 드로잉 장치에 공급할 수 있게 한다.

대표도

도 7a

색인어

실행 수단, 전송 수단, 저장 수단, 타입 처리. 넌타입 처리

명세서

도면의 간단한 설명

도 1은 본 발명에 의해서 제공된 정보 처리 장치가 적용되는 전형적인 가정용 오락 시스템을 도시하는 평면도

도 2는 도 1에 도시된 가정용 오락 시스템(1)의 정면도를 도시한 도면

도 3은 도 1에 도시된 가정용 오락 시스템(1)의 측면도를 도시한 도면

도 4는 도 1에 도시된 가정용 오락 시스템(1)에서 정보가 실행되는 전형적인 CD-ROM을 도시하는 평면도

도 5는 도 1에 도시된 가정용 오락 시스템(1)의 전형적인 내부 전기 배치 구성을 도시하는 블록도

도 6은 도 5에 도시된 메인 DMAC(46), 메인 CPU(44), 메인 메모리(45), 제 2 벡터 처리 엔진(VPE1)(48) 및 GPU(49)의 상세한 구성을 도시한 블록도

도 7a 및 도 7b는 복수의 처리기에 의해서 발생된 디스플레이 리스트를 처리하기 위한 절차를 도시하는 도면

도 8은 3개의 처리기가 GPU(49)를 제어하는 가정용 오락 시스템(1)의 또다른 전형적인 구성을 도시하는 블록도

도 9는 메타 명령의 전형적인 포맷을 도시하는 도면

도 10은 메타 명령에 따라서 데이터를 전송하기 위한 절차를 설명하기 위해 사용되는 도면

도 11은 메타 명령에 따라서 데이터를 전송하기 위한 또다른 절차를 설명하기 위해 사용되는 도면

도 12는 메타 명령에 따라서 데이터를 전송하기 위한 또다른 절차를 설명하기 위해 사용된 도면

도 13은 디스플레이 리스트에 따라서 데이터를 전송하기 위한 절차를 설명하기 위해서 사용된 도면

도 14는 스톨 제어를 설명하기 위해 사용된 도면

* 도면의 주요 부분에 대한 부호의 설명 *

2: 오락 시스템 메인 장치 3: 디스크 장착 서브장치

4: 리셋 스위치 5: 전원 스위치

8: 기록 삽입부 9: 셔터

38: 기록 장치

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

일반적으로, 본 발명은 정보 처리 장치 및 정보 처리 방법에 관한 것이다. 보다 상세하게는, 본 발명은 정보 처리 장치 및 정보 처리 방법에 관한 것으로서, 이러한 장치 및 방법에서는 전형적으로 복수의 CPU가 그들 각각의 명령을 동시에 발생시키고 그 발생된 명령을 CPU에 할당된 우선 순위에 따라 드로잉 장치에 전송하여 드로잉(drawing) 처리가 고효율적으로 드로잉 장치에 의해 실행될 수 있게 한다.

근년의 LSI의 집적도의 향상에 의해, 복수의 동일한 회로들이 하나의 LSI에서 병렬로 레이아웃될 수 있으며, 이에 의해 실행되는 처리 속도가 증가될 수 있게한다. 그러나, 근년에 제조된 비디오 게임 시스템에는 중앙 처리 장치(CPU) 및 드로잉 장치가 포함된다. 제어기로부터 디스플레이되는 폴리곤을 나타내는 데이터를 수신하여, CPU는 데이터에 대한 2 차원 또는 3 차원 좌표 변화를 실행하고 2 차원 프레임 버퍼 상에서 폴리곤을 드로잉하기 위해서 사용된 드로잉 명령의 리스트를 생성한다. 그 다음, CPU는 이하 디스플레이 리스트라 칭하여지는 리스트를 디스플레이 리스트에 따라서 폴리곤을 드로잉하기 위한 드로잉 장치에 전송한다.

그러나, 병렬 처리 기술을 사용함으로써 처리 시스템의 처리 능력을 개선하는 효과는, 다수의 단순한 조작이 드로잉 장치의 경우와 같이 반복적으로 실행되는 시스템에 대해서 더 효율적으로 얻을 수 있다. 따라서, CPU에 의해서 실행된 처리 속도는 근년에 제조된 병렬 처리 드로잉 장치와 비교할 때 비교적 낮다. 또한, 3 차원 모델 데이터의 량을 감소시키기 위해서, 드로잉 동작이 실행될 때마다 몇 개의 정점 데이터(제어 점)로부터 굴곡된 표면과 같은 복잡한 그래픽을 발생시키는 것이 필요하다. 그러한 복잡한 그래픽을 발생하기 위해서 CPU는 처리의 부하를 받는다. 그 결과, 드로잉 처리가 고효율적으로 실행될 수 없다는 문제가 야기된다.

발명이 이루고자 하는 기술적 과제

전술한 문제를 해소하기 위하여, 본 발명의 목적은 복수의 CPU에 우선 순위를 할당하고, CPU가 동시에 동작하게 하며, 가장 높은 우선순위를 갖는 CPU에 의해서 발생된 디스플레이 리스트로 시작하는 우선 순위에 기초하여 CPU에 의해서 발생된 디스플레이 리스트를 처리함으로써 고효율적으로 데이터를 처리할 수 있게 하는정보 처리 장치 및 정보 처리 방법을 제공하는데 있다.

청구항 제1항에 따른 정보 처리 장치는,

제 1 명령을 발생하기 위한 처리를 실행하는 제 1 발생 수단;

제 2 명령을 발생하기 위한 처리를 실행하는 제 2 발생 수단;

소정의 타이밍으로 상기 제 1 명령으로부터 상기 제 2 명령으로 그리고 그 역으로 스위칭함으로써 상기 제 1 및 제 2 명령들을 실행하는 실행 수단;

상기 제 1 명령을 상기 실행 수단에 전송하는 제 1 전송 수단;

상기 제 2 명령을 상기 실행 수단에 전송하는 제 2 전송 수단을 포함하며;

상기 제 1 발생 수단이 처리중에 있고 상기 실행 수단이 대기 상태에 있을 때, 상기 제 2 전송 수단은 만약 있다면 상기 제 2 명령을 상기 실행 수단에 공급하고, 상기 실행 수단은 상기 제 2 명령을 수신하여 상기 제 2 명령을 실행하는 것을 특징으로 한다.

청구항 제 5 항에 따른 정보 처리 방법은 다음과 같은 특징이 있다. 즉, 제 1 발생 수단이 처리중에 있으며 실행 수단이 대기 상태에 있을 때, 제 2 전송 수단은 제 2 명령을 상기 실행 수단에 공급하며, 상기 제 2 명령을 수신하여, 상기 실행 수단은 상기 제 2 명령을 실행한다.

청구항 제 1 항에 따른 정보 처리 장치에서,

제 1 발생 수단은 제 1 명령을 발생하기 위한 처리를 실행하며;

제 2 발생 수단은 제 2 명령을 발생하기 위한 처리를 실행하며;

정해진 시간동안에 하나의 실행 수단이 상기 제 1 명령으로부터 상기 제 2 명령으로 그리고 그 역으로 스위칭함으로써 상기 제 1 및 제 2 명령을 실행하며;

제 1 전송 수단은 상기 제 1 명령을 상기 실행 수단으로 전송하며;

제 2 전송 수단은 상기 제 2 명령을 상기 실행 수단에 전송하며;

상기 제 1 발생 수단이 처리중에 있으며 상기 실행 수단이 대기 상태에 있을 때, 상기 제 2 전송 수단은 상기 제 2 명령을 상기 실행 수단에 공급하며, 상기 제 2 명령을 수신하여, 상기 실행 수단은 상기 제 2 명령을 실행한다.

청구항 제 5 항에 따른 정보 처리 방법은, 제 1 발생 수단이 처리중에 있고 실행 수단이 대기 상태에 있을 때, 제 2 전송 수단은 만약 있다면 제 2 명령을 상기 실행 수단에 공급하고, 상기 실행 수단은 상기 제 2 명령을 수신하여 상기 제 2 명령을 실행하는 것을 특징으로 한다.

본 발명의 상기 및 기타 다른 목적들, 특징들 및 이점들은 단지 예들로서 기술된 본 발명의 양호한 실시예를 예시하는 첨부 도면을 참조할 때 다음의 상세한 설명으로부터 보다 더 명백해 질 것이다.

발명의 구성 및 작용

본 발명의 양호한 실시에에 대해서 첨부 도면을 참조하여 설명한다.

도 1 내지 도 3은 각각 본 발명에 의해서 제공된 정보 처리 장치가 적용되는 전형적인 가정용 오락 시스템을 도시하는 각각의 도면이다. 도면에 도시된 바와 같이, 가정용 오락 시스템은 오락 시스템 메인 장치(2)를 포함하는데 이는 조작 장치(17)와 오락 시스템 메인 장치(2)에 접속되는 기록 장치(38)에 추가해서 포함한다.

도 1 내지 도 3에 도시된 바와 같이, 오락 시스템 메인 장치(2)는 단지 사각형 형태이다. 오락 시스템 메인 장치(2)는, 그 중심에 배치된 CD-ROM(컴팩트 디스크 판독 전용 메모리)(40)를 장치하기 위한 디스크 장착 서브 장치(3), 사용자에 의해서 실행 애플리케이션을 임의로 리셋팅하기 위해서 오락 시스템 메인 장치(2)상에서 적당한 위치에 배치된 리셋트 스위치(4), 전원을 턴 온 및 오프시키기 위해서 사용자에 의해서 사용되는 전원 스위치(5), 디스크 장착 서브장치(3)상에 디스크를 장착하기 위해서 사용자에 의해서 사용되는 디스크 동작 스위치(6), 및 애플리케이션이 진행되는 동안에 동작을 실행하기 위해 사용되는 조작 장치(17)와 실행 애플리케이션과 같은 것의 정보의 여러 종류를 설정하는 것을 기록하기 위한 기록

장치(38)에 오락 시스템 메인 장치(2)를 접속시키기 위해서 사용자에 의해서 사용되는 우측 및 좌측의 접속기(7A,7B)를 포함한다. CD-ROM이 도 4에 도시된 것과 같은 광 디스크의 종류임을 알 수 있다. CD-ROM은 실행 애플리케이션 매체를 기록에 사용되는 디스크이다.

도 2 및 3에 도시된 바와 같이, 접속기(7A,7B)는 두 개의 레벨로 각각 설계된다. 각 접속기(7A,7B)의 상부 레벨에서, 오락 시스템 메인 장치(2)를 기록 장치(38)로 접속하기 위한 기록 삽입부(8)가 제공된다. 한편, 각각의 접속기(7A,7B)의 하부 레벨에서, 오락 시스템 메인 장치(2)를 조작 장치(17)에 접속시키기 위한 접속기 핀 삽입부(12)가 제공된다.

기록 삽입부(8)는 수평의 긴 사각형 삽입 홀 및 기록 장치(38)가 삽입되는 메모리 단자를 포함한다. 메모리 단자는 홀의 내부에 배치되며 도면에는 도시되지 않는다. 오락 시스템 메인 장치(2)에 접속되지 않은 기록 장치(38)에서, 기록 삽입부(8)는 도 2에 도시된바와 같이 먼지와 같은 것을 방지하기 위해서 메모리 단자를 보호하기 위한 셔터(9)에 의해서 커버된다. 기록 장치(38)는 메인 CPU(44)가 응용 소프트웨어의 데이터를 기록하는 전기적으로 프로그램가능한 ROM을 갖는다.

오락 시스템 메인 장치(2)상에 기록 장치(38)를 장착시킬 때, 사용자는 기록 장치(38)의 끝단부를 사용하여 기록 삽입부(8)의 내부로 셔터(9)를 밀며, 또한 기록 장치(38)가 메모리 단자에 접속될 때까지 기록 장치(38)를 삽입 홀로 삽입한다.

도 2에 도시된 바와 같이, 접속기 핀 삽입부(12)는 접속기 삽입부를 조작 장치(17)의 접속 단자부(26)로 접속시키기 위한 수평의 긴 사각형 삽입 홀 및 접속 단자(12A)를 갖는다.

도 1에 도시된 바와 같이, 조작 장치(17)는 사용자의 양손으로 집을 수 있으며 샌드위치되는 구조이며 각각의 손의 다섯 손가락의 자유로운 이동에 의해서 동작될 수 있다. 조작 장치(17)는 우측 및 좌측 측면에 대칭으로 배치된 조작 서브 장치(18, 19), 조작 서브장치(18, 19)사이에 배치된 선택 스위치(22) 및 스타트 스위치(23), 조작 서브장치(18, 19)의 정면에 배치된 조작 서브장치(24, 25), 및 조작 장치를 오락 시스템 메인 장치(2)에 접속시키기 위한 접속기(26, 27)를 포함한다.

도 5는 오락 시스템 메인 장치(2)의 전형적인 내부 전기배치 구성을 도시하는 도면이다. 도면에서 볼 수 있듯이, 오락 시스템 메인 장치(2)는 서브버스 인터페이스(SBUSIF)(43)에 의해서 서로 접속되는 메인 버스(41) 및 서브버스(42)를 갖는다.

메인 버스(41)에는 마이크로프로세서 및 VPE0(제 1 벡터 처리 엔진)(71)과 같은 성분에 의해서 실행되는 메인 CPU(중앙 처리 장치)(44)(리스트 발생 수단), RAM(랜덤 액세스 메모리)에 의해서 실행되는 메인 메모리(45), 메인 DMAC(메인 직접 메모리 액세스 제어기)(46)(데이터 전송 수단), MDEC(MPEG)(동화상 전문 그룹)디코더)(47), VPE1(제 2 벡터 처리 엔진)(48)에 접속된다. 이에 더해서, GPU(그래픽 처리 장치)(49)는 GPUIF(그래픽 처리 장치 인터페이스)(72)를 통해서 메인 버스(41)에 접속된다. CRTC(CRT 제어기)(84)는 GPU(49)상에 제공된다. 이에 더해서, 프레임 메모리(58)가 GPU(49)에 접속된다.

한편, 서브버스(42)는, 마이크로프로세서, RAM에 의해서 실행된 서브메모리(51), 서브 DMAC(52), 운영 체제와 같은 프로그램을 저장하기 위한 ROM(53), SPU (사운드 처리 장치)(54), 통신 제어 장치(ATM)(55), 전에 언급된 디스크 장착 서브장치(3)로서 사용되는 CD-ROM 드라이브(56) 및 입력 장치(57)가 접속된다. 입력 장치(57)의 접속 단자(12A)는 전술된 바와 같은 조작 장치(17)의 접속 단자부(26)에 접속된다.

메인 버스(41) 및 서브버스(42)에 접속되어, SBUSIF(43)는 메인 버스(41)로부터 서브버스(42)로 들어오는 데이터를 통과시키며, 한편, 서브버스(42)로부터 메인 버스(41)로 들어오는 데이터를 전송한다.

오락 시스템 메인 장치(2)가 실행될 때, 메인 CPU(44)는 SBUSIF(43)를 경유하여 서브버스(42)에 접속된 ROM(53)으로부터 활성 프로그램의 명령을 페치하며, 운영 체제를 활성화하기 위해서 활성 프로그램의 명령을 실행한다.

이에 더해서, 메인 CPU(44)는 CD-ROM 드라이브(56)상에 장착된 CD-ROM(40)으로부터 데이터 및 응용 프로그램을 획득하기 위해서 CD-ROM 드라이브(56)에 데이터를 판독하기 위한 요청을 발생하며, 응용 프로그램을 메인 메모리(45)로 로딩한다.

더욱이, 제 1 벡터 처리 엔진(VPE0)과 관련하여, 메인 CPU(44)는 넌타입 처리를 위한 데이터를 발생하며, 즉, 폴리곤 정의 정보를 CD-ROM(40)으로부터 판독된 폴리곤과 같은 다수의 기본 도면을 포함하는 3 차원 오브젝트의 데이터로부터 발생한다. 3 차원 오브젝트의 데이터의 예는 폴리곤의 수직 좌표 값 또는 대표점들이다. VPE0(제 1 벡터 처리 엔진)(71)은 부동 소수점 실수를 처리하기 위한 다수의 처리 요소를 갖으며 따라서 병렬로 부동 소수점의 처리의 일부를 실행할 수 있다.

그것을 상세히 입력하기 위해서, 메인 CPU(44) 및 VPE0(제 1 벡터 처리 엔진)(71)은 폴리곤 장치내의 상세한 동작을 야기하는 기하학적 처리를 실행한다. 그러한 처리의 예는 바람에 의해서 떨어진 나무의 낙엽의 흔들리는 상태 또는 자동차의 정면 유리에 부딪치는 빗물 방울의 상태를 표시하는 폴리곤의 데이터 발생이다. 따라서, 처리로부터 발견된 정점 정보 및 세이딩 모드 정보와 같은 폴리곤 정의 정보는 메인 버스(41)를 경유해서 패킷으로서 메인 메모리(45)로 공급된다.

폴리곤 정의 정보는 드로잉 영역 세팅 정보 및 폴리곤 정보를 포함한다. 드로잉 영역 세팅 정보는 드로잉 영역의 프레임 메모리(58)내의 옵셋 좌표 즉, 드로잉 영역의 프레임 메모리 어드레스와, 드로잉 영역의 외부에 존재하는 좌표를 갖는 폴리곤에 의해서 표시되는 드로잉 지역을 그리기 위해서 동작을 소거하기 위한 드로잉 클리핑의 좌표를 포함한다. 한편, 폴리곤 정보는 폴리곤 속성 정보 및 정점 정보를 포함한다. 여기서, 폴리곤 속성 정보는 세이딩 모드, ALPHA 블렌딩 모드 및 구조 맵핑 모드를 규정하기 위해 사용된 정보이다. 한편, 정점 정보는 몇 개만 언급하면, 정점 드로잉 영역내의 좌표, 정점 구조 영역내의 좌표 및 정점의 칼라에 대한 정보이다.

제 1 벡터 처리 엔진(VPE0)(71)과 매우 유사하게, 제 2 벡터 처리 엔진(VPE1)(48)은 부동 소숫점 실수를 처리하기 위한 다수의 처리 요소를 가지며 따라서 병렬로 부동 소수점 처리의 일부를 실행할 수 있다. VPE1(48)은 조작 장치(17) 및 매트릭스 동작을 사용함으로써 실행되는 동작에 따라서 화상의 발생이 가능하다. 말하자면, 제 2 벡터 처리 엔진(VPE1)(48)은 타입 처리 즉, VPE1(48)상의 프로그램 실행에 의해서 실행되기에 충분하도록 비교적 간단한 처리를 위해서 데이터(폴리곤 정의 정보)를 발생한다. 제 2 벡터 처리 엔진(VPE1)(48)에 의해서 실행되는 처리의 예는 빌딩이나 차량과 같은 간단한 형태를 갖는 오브젝트를 위한 방사선 투시 카피 변환, 평행 광 소스 계산 및 2 차원 굴곡된 표면의 발생이다. 다음으로, VPE1(48)에 의해서 발생된 폴리곤 정의 정보는 GPUIF(72)에 공급된다.

메인 CPU(44)에 의해서 제어되는 GPUIF(72)는 메인 버스(41)를 통해서 메인 메모리(45)로부터 폴리곤 정의 정보를 수신한다. 이러한 이유로, GPUIF(72)는 제 2 벡터 처리 엔진(48)에 의해서 공급된 폴리곤 정의 정보와 충돌하는 것으로부터 메인 CPU(44)에 의해서 발생된 폴리곤 정의 정보를 방지하기 위해서 처리 타이밍을 조절하며, 따라서 그것들을 GPU(49)로 전송한다.

GPU(49)는 프레임 메모리(58)상에서 GPUIF(72)를 경유하여 공급된 폴리곤 정의 정보에 기초한 폴리곤을 이용하여 3 차원 오브젝트를 표시하는 화상을 그린다. 3 차원 오브젝트를 표시하기 위해 폴리곤을 이용하여 그려진 화상은 이하 폴리곤 화상으로 일컬어진다. GPU(49)가 정점 메모리로서 프레임 메모리(58)를 이용할 수 있으므로, GPU(49)는 정점으로서 폴리곤상에 프레임 메모리(58)내의 화소 화상을 고정시키기 위해서 정점 맵핑 처리를 실행할 수 있다.

메인 DMAC(46)는 다른 동작중에서, DMA 전송을 메인 버스(41)로 접속된 다양한 회로로부터 또는 그 회로로 제어하기 위해서 사용된다. 이에 더해서, SBUSIF(43)의 상태에 따라서, 메인 DMAC(46)가 다른 동작중에서, 서브버스(42)로 접속된 다양한 회로로부터 또는 그 회로로 DMA 전송을 제어할 수 있다. MDEC(47)는 메인 CPU(44)와 동시에 동작하며, MPEG(동화상 전문 그룹) 시스템 또는 JPEG(조인트 사진 전문 그룹)에 따라서 압축된 데이터를 압축 해제한다.

서브 CPU(50)는 ROM(53)내에 저장된 프로그램을 실행함으로써 다양한 종류의 처리를 실행한다. 서브 DMAC(52)는 다른 동작중에서, SBUSIF(43)가 메인 버스(41) 및 서브버스(42)에 단선될 때만 서브버스(42)에 접속된 다양한 회로로부터 그리고 그 회로로 DMC 전송을 제어한다.

SPU(54)는 사운드 메모리(59)로부터 사운드 데이터를 판독하며, 서브CPU(50) 또는 서브 DMAC(52)로부터 수신된 사운드 명령에 따라서 오디오 신호로서 사운드 데이터를 출력한다. 출력 오디오 신호는 증폭기 회로(201)를 경유하여 스피커(202)로 공급되며 이는 최종적으로 사운드로서 스피커(202)에 의해서 출력된다.

통신 명령 장치(ATM)(55)는 공중 통신선과 같은 것에 접속되며 라인을 통해서 전송 및 수신하기 위해서 사용된다.

입력 장치(57)는 조작 장치(17)를 오락 시스템 메인 장치(2)에 접속시키기 위한 접속 단자(12A), 도면에 도시되지 않은 다른 장치로부터 나오는 비디오 데이터를 오락 시스템 메인 장치(2)로 공급하기 위한 비디오 입력 회로(82) 및 다른 장치로부터 나오는 오디오 데이터를 오락 시스템 메인 장치(2)로 공급하기 위한 오디오 입력 회로(83)를 포함한다.

도 6은 도 5에 도시된 메인 DMAC(46), 메인 CPU(44), 메인 메모리(45), 제 2 벡터 처리 엔진(VPE1)(48) 및 GPU(49)의 세부 구성을 도시하는 블록도이다.

도 6에 도시된 바와 같이, 메인 CPU(44)는 CPU 코어(CORE)(94), 명령 캐시(I$)(95), 스크래치 패드 RAM(SPR)(96), 데이터 캐시(D$)(97) 및 제 1 벡터 처리 엔진(VPE0)(71)을 포함한다. CPU 코어(94)는 선정된 명령을 실행한다. 명령 캐시(95)는 CPU(94)에 공급되는 명령을 일시적으로 저장하기 위해서 사용된다. 스크래치 패드 RAM(96)은 CPU 코어(94)에 의해서 실행된 처리 결과를 저장하기 위해서 사용된다. 마지막으로, 데이터 캐시(97)는 CPU 코어(94)에 의해서 처리의 실행에서 사용되는 데이터를 일시적으로 저장하기 위해서 사용된다.

제 1 벡터 처리 엔진(VPE0)(71)은 마이크로메모리(microMEM)(98), FMAC(부동 복수 가산기 계산) 장치(99), 분할기(DIV)(100), VU-MEM 및 패킷 확장기(PKE)(102)로 불리는 기능 장치(101)를 포함한다. VU-MEM(101)은 부동 소수점 벡터 처리 장치(VU) 및 내장 메모리(MEM)를 포함한다. 부동 소수점 벡터 처리기 장치는 VU 및 내장 메모리의 내부 레지스터 내에 저장된 데이터를 처리하기 위해서 더 나중에 설명되는 마이크로메모리(98)내에 저장된 마이크로프로그램의 64 비트 마이크로 명령을 실행한다.

PKE(102)는 마이크로메모리(98)내의 마이크로프로그램으로서 저장되며 VU에 의해서 실행되는 마이크로명령으로 후에 설명되는 메인 DMAC(109)에 의해서 실행된 제어에 따라서 공급된 마이크로코드를 확장하며, 메인 DMAC(109)에 의해서 실행된 제어에 따라서 공급된 팩 데이터의 패킷을 확장하여, VU-MEM(101)내에 사용된 내장 메모리(MEM)로 확장된 패킷을 저장한다. FMAC(부동 다중 가산기 계산) 장치(99)는 분할기(DIV)(100)가 분할을 실행하는 부동 소수점 처리를 실행한다. 전술된 바와 같이, 제 1 벡터 처리 엔진(VPE0)(71)은 VPE0(71)와 관련하여 넌타입 처리를 실행하는 메인 CPU(44)내에 내장된다.

제 1 벡터 처리 엔진(VPE0)(71)과 매우 유사하게, 제 2 벡터 처리 엔진(VPE0)(48)은 마이크로메모리(microMEM)(103), FMAC(부동 다중 가산기 계산) 장치(107), 분할기(DIV)(106), VU-MEM으로 불리는 기능 장치(104) 및 패킷 확장기(PKE)(108)를 포함한다. VU-MEM(107)은 부동 소수점 벡터 처리기 장치(VU) 및 내장 메모리(MEM)를 포함한다. 부동 소수점 벡터 처리기 장치는 VU 및 내장 메모리의 내부 레지스터에 저장된 데이터를 처리하기 위해서 후술되는 마이크로메모리(microMEM)(103)내에 저장된 마이크로프로그램의 64비트 마이크로 명령을 실행한다.

PKE(108)는 마이크로메모리(103)내의 마이크로프로그램으로서 저장되며 VU에 의해서 실행되는 마이크로명령으로 메인 DMAC(46)에 의해서 실행된 제어에 따라서 공급된 마이크로코드를 확장하며, 메인 DMAC(46)에 의해서 실행된 제어에 따라서 공급된 팩 데이터의 패킷을 확장하며, 따라서 VU-MEM(107)로 내장된 내장 메모리(MEM)로 확장된 패킷을 저

장한다. FMAC(부동 다중 가산기 계산) 장치(104)는 분할기(DIV)(106)가 분할을 실행하는 처리를 부동 소수점 처리를 실행한다. 제 2 벡터 처리 엔진(48)은 메인 메모리(45)로부터 공급된 데이터 상에서 타입 처리를 실행하며 GPU(49)를 경유하여 GPUIF(72)에 처리 결과를 공급한다.

메인 메모리(45)는 3 차원 오브젝트의 데이터를 저장하기 위해서 사용되며, 필요할 때, 제 1 벡터 처리 엔진(71) 및 제 2 벡터 처리 엔진(48)에 데이터를 공급한다. 메인 CPU(44) 및 제 1 벡터 처리 엔진(VPE0)(71)에 의해서 함께 발생된 디스플레이 리스트는 메인 버스(41)를 경유하여 GPUIF(72)에 공급되기 전에 메인 메모리(45)내에 내장된 메모리 FIFO(MFIFO)내에 일시적으로 저장된다. 디스플레이 리스트가 메모리 FIFO내에 일시적으로 저장되는 이유는 메인 CPU(44) 및 제 1 벡터 처리 엔진(71)이 각각 제 2 벡터 처리 엔진(48)의 순위보다 낮은 처리 순위를 갖으며, 이는 제 2 벡터 처리 엔진(48)이 아이들 상태로 들어갈 때까지 메모리 FIFO내에 디스플레이 리스트를 유지하는 것이 필요하게 한다.

이에 더해서, 메인 CPU(44) 및 제 1 벡터 처리 엔진(VPE0)(71)은 제 2 벡터 처리 엔진(48)에 의해서 처리되는 매트릭스를 함께 발생하여, 메인 메모리(45)내에 매트릭스를 저장한다. 그후, 제 2 벡터 처리 엔진(48)은 매트릭스를 이용함으로써 디스플레이 리스트를 만든다.

GPUIF(72)를 경유하여 제 1 벡터 처리 엔진(71)으로부터 공급된 넌타입 처리를 위한 디스플레이 리스트와 제 2 벡터 처리 엔진(48)으로부터 공급된 타입 처리를 위한 디스플레이 리스트를 처리하기 위해서, GPU(49)는 다른 것들 중에서, 드로잉 옵셋의 그래픽 내용(즉, 드로잉 설정 상태) 및 각각의 디스플레이 리스트에 대한 드로잉 시간으로 일컬어지는 클럽 범위를 유지한다. 다음 설명에서 사용된 기호 CG0은 넌타입 처리를 위한 그래픽 내용을 표시하며 한편 기호 CG1은 후에 시술되듯이 타입 처리를 위한 그래픽 내용을 표시한다.

예를 들어, 전술된 바와 같이, PKE(102)는 DMAC(109)에 의해서 실행되는 제어에 따라서 메인 버스(41)를 경유하여 메인 메모리(45)로부터 제 1 벡터 처리 엔진(71)으로 공급된 마이크로코드를 마이크로메모리(98)내에서 마이크로프로그램으로서 저장되며 VU에 의해서 실행되는 마이크로명령으로 확장하며, DMAC(109)에 의해서 실행된 제어에 따라서 메인 버스(41)를 경유하여 메인 메모리(45)로부터 공급된 3 차원 오브젝트의 데이터와 같은 팩 데이터의 패킷을 확장하며, 따라서 확장된 패킷을 VU-MEM(101)내에서 사용되는 내장 메모리(MEM)로 확장된 패킷을 저장한다. 그후, FMAC(99) 및 DIV(100)는 매트릭스 처리, 좌표 변환 및 3 차원 오브젝트의 데이터 상에서 방사선 카피 변환과 같은 처리의 일부를 실행한다. 그때, 복잡한 처리는 CPU 코어(94)와 관련하여 실행된다. 그 처리는 전형적으로 바람에 의해 불어진 나무 잎의 흔들리는 상태 또는 차량의 정면 유리에 부딪치는 빗방울의 상태를 그리기 위한 디스플레이 리스트를 야기한다.

스크린 상에서 이러한 방법으로 발생된 2 차원 오브젝트를 그리기 위한 디스플레이 리스트(복잡한 스트림)는 GPUIF(72)에 최종적으로 공급되기 전에 메인 버스(41)를 통해서 메인 메모리(45)의 MFIFO내에 일시적으로 저장된다.

한편, 전술된 바와 같이, PKE(108)는, 메인 DMAC(46)에 의해서 실행된 제어에 따라서 메인 버스(41)를 경유하여 메인 메모리(45)로부터 제 2 벡터 처리 엔진(48)으로 공급된 마이크로코드를 마이크로메모리(13)내에서 마이크로프로그램으로서 저장되며 VU에 의해서 실행되는 마이크로명령으로 확장하며, 메인 버스(46)에 의해서 실행된 제어에 따라서 메인 버스(41)를 경유하여 메인 메모리(45)로부터 공급된 3 차원 오브젝트의 데이터와 같은 팩 데이터의 패킷을 확장시키며, 따라서 확장된 패킷을 VU-MEM(107)내에서 사용된 내장 메모리(MEM)로 확장된 패킷을 저장한다. 그후, FMAC(104) 및 DIV(106)는 매트릭스 처리, 좌표 변환 및 3 차원 오브젝트의 데이터 상에서 방사선 카피 변환과 같은 처리의 일부를 실행한다. 메인 CPU(44) 및 제 1 벡터 처리 엔진(71)에 의해서 함께 발생된 매트릭스 및 그래픽 내용에 기초해서 그리고 메인 버스(41)를 경유하여 메인 메모리(45)로부터 제 2 벡터 처리 엔진(48)에 공급되어, 처리는 비교적 간단한 형태의 처리이다.

스크린 상에서 이러한 방법으로 발생된 2 차원 오브젝트를 그리기 위한 디스플레이 리스트(간단한 스트림)는 메인 버스(41)를 경유하여 GPUIF(72)로 최종적으로 공급된다. 두 개의 스트림, 즉, 복잡하고 간단한 스트림은 중재에 의해서 시분할에 기초하여 GPU(49)로 전송된다.

GPU(49)는 GPUIF(72)에 의해서 공급된 디스플레이 리스트에 기초한 드로잉 처리를 실행하며, 프레임 메모리(58)상에서 폴리곤을 그린다. 디스플레이 리스트가 메인 CPU(44) 및 메인 메모리 장치(45)상의 제 1 벡터 처리 엔진(71)에 의해서 함께 발생된 디스플레이 리스트이며, 메인 버스(41)를 경유하여 GPU(49)에 공급되면, GPU(49)는 전술된 그래픽 내용 GC0를 이용함으로써 드로잉 처리를 실행한다. 한편, 디스플레이 리스트가 제 2 벡터 처리 엔진(48)에 의해서 발생된 디스플레이 리스트이면, GPU(49)는 전술된 그래픽 내용 GC1을 이용함으로써 드로잉 처리를 실행한다.

프레임 메모리(58)상에 그려진 폴리곤은 CRTC(84)에 의해서 실행된 제어에 따라서 폴리곤을 위한 출력 비디오 신호로 변환된다.

도 7a 및 도 7b는 두 개의 디스플레이 리스트가 처리되는 타이밍을 도시하는 도면이다. 도 7a 및 도 7b에 도시된 기하 서브시스템(0)은 도 6에 도시된 제 2 벡터 처리 엔진(48)에 대응하며 여기서 기하 서브시스템(1)은 메인 CPU(44) 및 제 1 벡터 처리 엔진(71)에 대응한다. 렌더링 서브시스템은 GPU(49)에 대응한다. 도면에 도시된 해칭 부분은 작업 이름에 의해서 표시된 작업이 아이들 상태임을 표시한다.

도 7a는 하나의 처리기, 즉 기하 서브시스템(0)이 존재하는 경우에 처리 절차를 도시하는 도면이다. 이 경우에, 기하 서브시스템(0)은 디스플레이 리스트(리스트 #0-1)를 만들며, 그 리스트를 렌더링 시스템에 공급한다. 그후, 기하 서브시스템(0)은 리스트 #0-1를 따르는 디스플레이 리스트 즉, 리스트 #0-2 및 후속되는 디스플레이 리스트를 계속해서 만든다. 렌더링 시스템은 기하 서브시스템(0)으로부터 공급된 디스플레이 리스트(리스트 #0-1)에 따라서 드로잉 처리를 실행한다. 기하 서브시스템(0)이 리스트 #0-1에 따라서 렌더링 서브시스템이 드로잉 처리를 완료하는 시점에 다음 디스플레이 리스트(#0-2)를 만드는 중이면, 렌더링 서브시스템은 아이들 상태로 들어가며, 다음 디스플레이 리스트(리스트 #0-2)의 발생을 완성하며 리스트를 렌더링 서브시스템으로 공급하기 위해서 기하 서브시스템(0)을 대기한다.

그후, 전술된 바와 매우 유사하게, 기하 서브시스템(0)이 현재의 디스플레이 리스트에 따른 드로잉 처리를 렌더링 서브시스템이 완성하는 시점에서 다음 디스플레이 리스트를 만드는 처리를 완성하지 않았으면, 렌더링 서브시스템은 아이들 상태로 들어가며, 다음 리스트를 렌더링 서브시스템으로 공급하기 위해서 기하 서브시스템(0)을 대기한다.

도 7b는 두 개의 처리기, 즉 기하 서브시스템(0) 및 기하 서브시스템(1)이 존재하는 경우에 대한 처리 절차를 도시하는 도면이다. 이 경우에, 기하 서브시스템(0)이 디스플레이 리스트(리스트 #0-1)을 만드는 동안에, 렌더링 시스템(0)은 아이들 상태에 놓이게 된다. 이러한 이유로, 기하 서브시스템(1)에 의해서 이미 발생되었으며 메인 메모리(45)에 저장된 디스플레이 리스트(리스트 #1-1)와 결합된 데이터는 렌더링 서브시스템에 공급된다. 기하 서브시스템(1)에 의해서 발생된 제 1 디스플레이 리스트(리스트 #1-1)를 수신하여, 렌더링 서브시스템은 기하 서브시스템(1)에 의해서 렌더링 시스템에 공급된 제 1 디스플레이 리스트(리스트 #1-1)에 첨부된 기하 서브시스템(1)에 대한 그래픽 내용에 기초한 드로잉 처리를 실행한다.

기하 서브시스템(0)이 제 1 디스플레이 리스트(리스트 #0-1)를 발생하기 위한 처리를 완료할 때, 기하 서브시스템(1)은 다음 디스플레이 리스트(리스트 #1-2)를 렌더링 시스템에 공급하고 있다. 이 시점에서, 기하 서브시스템(1)은 렌더링 서브시스템에 다음 디스플레이 리스트(리스트 #1-2)를 공급하기 위한 동작을 중지시키도록 강요된다. 따라서, 기하 서브시스템(0)은 완성된 제 1 디스플레이 리스트(리스트 #0-1)를 렌더링 서브시스템에 공급할 수 있으며 다음 디스플레이 리스트(리스트 #0-2)로 만들기 시작한다. 기하 서브시스템(0)으로부터 제 1 디스플레이 리스트(리스트 #0-1)를 수신하여, 렌더링 서스시스템은 제 1 디스플레이 리스트(리스트 #0-1)에 기초한 드로잉 처리를 실행한다.

렌더링 서브시스템이 제 1 디스플레이 리스트(리스트 #0-1)에 기초한 드로잉 처리를 완료할 때, 기하 서브시스템(0)은 아직도 다음 디스플레이 리스트(리스트#0-2)를 만드는 중이다. 이러한 이유로, 기하 서브시스템(1)은 다음 디스플레이 리스트(리스트 #1-2)를 렌더링 서브시스템에 공급하기 위해서 지연된 동작을 개시한다. 한편, 렌더링 서브시스템은 아이들 상태로 들어간다. 기하 서브시스템(1)에 의해서 발생된 다음 디스플레이 리스트(리스트 #1-2)를 수신하여, 렌더링 서브시스템은 다음 디스플레이 리스트(리스트 #1-2)에 기초한 드로잉 처리의 실행을 시작한다.

그후에, 전술된 것과 매우 유사하게, 기하 서브시스템(1)은 기하 서브시스템(0)이 아직도 디스플레이 리스트를 만들고 있을 때만 발생된 디스플레이 리스트를 공급하며, 아이들 상태에 렌더링 서브시스템을 놓게 된다. 그 결과로서, 복수의 처리기에 의해서 실행된 디스플레이 리스트는 렌더링 서브시스템에 의해서 효율적으로 처리될 수 있다.

예를 들어, 좌표 변환 처리를 더 빨리 실행하기 위해서, 서브프로세서 또는 좌표 변환 코프로세서는 공통 드로잉 장치, 즉 GPU(49)를 공유하는 복수의 벡터 처리 엔진 각각에서 CPU(전술된 VU)로부터 분리되어 공급될 수 있으며, 각각의 출력 디스플레이는 GPU(49)로 리스트된다. 그러한 서브프로세서 또는 코프로세서를 사용함으로써, 각각의 벡터 처리 엔진내의 CPU(프로세서)는 현재 공유된 GPU(49)에 디스플레이 리스트를 더 자주 공급할 수 있다. 따라서, GPU(49)는 짧은 시간 간격동안 프로세서간에 스위치되어야 한다. 그렇지 않으면, 각각의 처리기에 대해서 제공된 로컬 메모리 내에서 오버플로우가 발생된다. 이러한 이유로, 우선 레벨이 각각의 프로세서에 지정되며, 즉, 도 7b에 도시된 바와 같이, 기하 서브시스템(0) 및 기하 서브시스템(1)에 지정된다. 마스터 처리기로부터 GPU(49)에 전송되는 디스플레이 리스트가 더 없으면, 즉, 도 6에 도시된 정보 처리 장치 또는 도 7a 및 도 7b에 도시된 기하 서브시스템(0)의 경우에 제 2 벡터 처리 엔진(48)내에서 사용된 CPU 또는 가장 높은 우선 순위를 갖는 CPU에서, GPU(49)를 액세스하기 위한 권리는 슬레이브 프로세서에 전송되는데, 즉, 도 6에 도시된 정보 처리 장치 또는 도7a 및 도 7b에 도시된 기하 서브시스템(1)의 경우에 제 1 벡터 처리 엔진(71)에서 사용된 CPU 또는 매스터 CPU(44)에 대해서 두 번째 우선 순위를 갖는 CPU이다.

매스터 프로세서가 디스플레이 리스트를 만드는 처리를 완료하며 GPU(49)에 디스플레이 리스트를 전송하기 위해서 준비되자마자, 슬레이브 프로세서는 디스플레이 리스트가 완료되도록 남겨지고 슬레이브 프로세서에 의해서 GPU(49)에 의해서 전송된다 할지라도 매스터 프로세서에 GPU(49)를 액세스하기 위한 권리를 리턴시키도록 강요된다.

일반적으로, 매스터 프로세서는 높은 속도로 처리할 수 있지만 비교적 작은 저장 용량을 갖는 로컬 메모리를 갖는다. 한편, 슬레이브 프로세서는 비교적 낮은 속도로 처리를 실행하지만 비교적 큰 저장 용량을 갖는 로컬 메모리를 갖는다.

슬레이브 처리기(1)에 대한 슬레이브로서 사용되는 처리기(2)가 도 8에 도시된 바와 같이 더 접속되는 정보 처리 장치가 있다. 그러한 정보 처리 장치에서, 특별히 낮은 우선 순위를 갖는 처리기는 더 많은 디스플레이 리스트를 저장하기 위해서 훨씬 더 큰 저장 용량을 갖는 로컬 메모리를 필요로 한다. 이러한 이유로, 낮은 우선 순위가 보통은 메인 메모리가 제공되는 메인 프로세서로 지정된다. 이러한 방식으로, 메인 프로세서는 슬레이브 프로세서로서 사용된다.

GPU(49)에 의해서 실행된 드로잉 처리엣, 디스플레이 리스트에 기술된 정점 정보에 더해서, 드로잉 시간에 드로잉 옵셋 및 클립 범위와 같은 그래픽 내용으로서 일컬어지는 환경 파라미터 또는 드로잉 세팅 조건이 전술된 바와 같이 요청된다. 렌더링 서브시스템(즉, GPU(49)은 기하 서브시스템에 대한 그래픽 내용에 따른 각각의 기하 서브시스템(즉, CPU)에 의해서 공급된 디스플레이 리스트에 기초한 드로잉 처리를 실행한다. 그러나, 디스플레이 리스트 공급기가 하나의 기하 서브시스템으로부터 다른 서브시스템으로 스위치될 때, 그래픽 내용을 새롭게 설정하기 위한 작업량이 실행될 필요가 있다. 이러한 문제를 해결하기 위해서, 렌더링 서브시스템은 기하 서브시스템만큼 많은 그래픽 내용을 유지한다.

그래픽 내용이 도시되는 GPU(49)에 공급된 각각의 오브젝트에 대해서 도 7a 및 도 7b에 도시된 바와 같이 디스플레이 리스트에 가산된다. 그 결과, GPU(49)는 오브젝트와 결합된 그래픽 내용을 기초로 하여 각각의 오브젝트에 대한 드로잉 처리를 실행하는 것이 가능하다.

기하 서브시스템 및 렌더링 서브시스템은 데이터 버스 및 어드레스 버스를 포함하는 메인 버스(41)를 공유한다. 렌더링 서브시스템을 액세스하는 기하 서브시스템은 기하 서브시스템의 ID와 기하 서브시스템에 의해서 발생된 디스플레이 리스트를 버스 및 데이터 버스를 통해서 렌더링 서브시스템으로 전송한다. ID 및 디스플레이 리스트를 수신하여, 렌더링 서브시스템은 ID에 대응하는 그래픽 내용을 선택하며 그래픽 내용을 기초로 하여 디스플레이 리스트를 변환한다. 렌더링 서브시스템은 프레임 버퍼 상에서 화상을 그린다.

복수의 처리기(벡터 처리 엔진 또는 기하 서브시스템)가 전술된 바와 같이 우선 순위에 기초하여 GPU(49)를 제어하게 함으로써, 프로세서 내에서 만들어진 디스플레이 리스트를 일시적으로 저장하기 위한 각각의 프로세서 내에 제공된 로컬 메모리의 저장 용량이 최소로 감소될 수 있다. 그 결과, 로컬 메모리의 비용을 증가시키지 않고도 처리기에서 디스플레이 리스트를 병렬로 만들기 위한 처리를 실행하는 것이 가능하다. 이에 더해서, 각각의 프로세서에 대한 그래픽 내용을 GPU(49)(렌더링 서브시스템)내로 유지시키므로서, 복사된 데이터 전송 수는 즉, 내용 스위칭 중에 실행되는 오버헤드 작업량은 감소될 수 있다.

엄격히 말해서, 프로세서는 데이터 버스를 공유하며, 시간 공유상에서 메인 메모리를 공유한다. 다음은 GPU(49)에 데이터를 전송하는 동안에 데이터 자체에 내장된 메타 명령에 따라서 데이터를 제어하는 기술을 설명한다.

도 9는 메타 명령의 전형적인 포맷을 도시한 도면이다. 메타 명령은 전송되는 데이터의 정면에 부가되는 명령이다. 메타 명령은 전송된 데이터, 데이터 전송 목적지 및 메타 명령의 동작 코드의 길이를 규정한다. 메타 명령은 128 비트를 포함하며, 도면에 도시된 것들중 단지 64 비트만이 유효하다. 전송되는 데이터의 크기는 제 1 16 비트 필드 QWC내에서 규정된다. 이러한 메타 명령의 동작 코드는 24 번째 비트로부터 31 번째 비트의 필드를 점유한다. 32번째 비트로부터 63 번째 비트의 필드는 전송되는 이러한 데이터가 저장되거나 메타 명령이 다음에 판독되는 어드레스를 규정하기 위해서 사용된다.

데이터의 전송은 다음과 같이 데이터에 내장된 메타 명령의 동작 코드에 따라서 제어된다.

동작 코드가 "cnt"이면, 이러한 메타 명령에 이은 QWC 필드에 의해서 규정된 데이터의 워드가 전송된 후에, 이러한 패킷(즉, 메타 명령 및 데이터)에 이은 어드레스에서 저장된 메타 명령은 프로세서에 의해서 실행된다. 동작 코드가 "cnt"이면, 이러한 메타 명령에 이은 QWC 필드 에 의해서 규정된바와 같은 데이터 워드가 스톨 제어를 실행함으로써 전송된 후에, 이러한 패킷에 이은 어드레스에 저장된 메타 명령이 프로세서에 의해서 실행된다. 동작 코드가 "next"이면, 이러한 메타 명령에 이은 QWC 필드에 의해서 규정된 데이터 워드가 전송된 후에, 어드레스 필드 내에서 규정된 어드레스에서 저장된 메타 명령이 프로세서에 의해서 실행된다.

스톨 제어는 프로세서에 의해서 만들어진 액세스를, 또다른 프로세서에 의해서 만들어진 메인 메모리)45)에 대한 액세스가 완료될 때까지 대기 상태로 메인 메모리(45)로 놓기 위해서 프로세서에 의해서 실행된 타이밍 제어이다.

동작 코드가 "REF"인 경우, 어드레스 필드에 지정된 어드레스 ADDR에 저장된 QWC 필드에 의해 지정된 수만큼의 데이터 워드가 전송된 후에, 메타 명령에 뒤이은 어드레스에 저장된 메타 명령이 프로세서에 의해 실행된다. 동작 코드가 "refs" 일 때, 어드레스 필드 내에 규정된 어드레스 ADDR에 저장된 QWC 필드에 의해 규정된 데이터 워드가 스톨 제어를 실행함으로써 전송된 후에, 이러한 메타 명령에 이어서 어드레스에 저장된 메타 명령은 프로세스에 의해서 실행된다. 전술된 바와 같이, "REF" 메타 명령은 어드레스 필드 내에서 규정된 어드레스로부터 QWC 필드 내에서 규정된 길이를 갖는 데이터를 전송하기 위해서 사용된다.

동작 코드가 "call"이면, 이러한 메타 명령에 이은 QWC 필드에 의해서 규정된 바와 같이 데이터 워드가 전송된 후에, 이러한 패킷에 이은 어드레스는 리턴 어드레스로서 레지스터로 밀리며(또는 로딩되며) 메타 명령의 어드레스 필드 내에서 규정된 어드레스에 저장된 메타 명령은 프로세서에 의해서 실행된다. 동작 코드가 "ret"이면, 이러한 메타 명령에 이은 QWC 필드에 의해서 규정된 데이터의 워드가 전송된 후에, 레지스터로부터 폽 백(또는 판독)된 어드레스에 저장된 메타 명령은 프로세서에 의해서 실행된다. 이러한 "RET" 메터 명령과 연결된 "CALL" 동작 코드를 갖는 메타 명령의 실행 동안에 어드레스가 리턴 어드레스로서 레지스터에 밀리게 됨을 알 수 있다. 동작 코드가 "end" 이면, 이러한 메타 명령에 이은 QWC 필드에 의해서 규정된 데이터의 워드가 전송된 후에, 처리는 끝난다.

도 10은 메타 명령의 동작 코드가 이러한 메타 명령에 이은 다음 데이터가 전송되는 것을 의미하는 "next"일 때 프로세서에 의해서 실행되는 동작을 설명하기 위해 사용된 도면이다. 무엇보다도, 메인 DMAC(46)는 Tag 어드레스 레지스터 Dn_TADR내에 저장된 어드레스 ADDR0로부터 메타 명령 워드로서 1 워드를 판독한다. 메타 명령이 "NEZT,ADDR =ADDR2, LEN =8" 로서 동작 코드가 "next"를 의미할 때, QWC 필드는 전송되는 데이터의 길이가 8 쿼드(16 진 워드)로서 여기서 1 쿼드는 128 비트이며 ADDR2는 어드레스 필드 내에서 규정된 어드레스이다. 따라서, 메타 명령 "NEXT, ADDR = ADDR2, LEN =8"의 실행에서, 8 쿼드 데이터가 전송된다. 그후, 어드레스 ADDR2에 저장된 메타 명령 "NEXT,ADDR =ADDR1, LEN =2"가 실행된다.

동일한 토큰에 의해서, 메타 명령 "NEXT, ADDR = ADDR1, LEN =2"를 실행하는데 있어, 2 쿼드 데이터가 메인 DMAC(46)에 의해서 실행된 제어하에서 전송된다. 그후, 어드레스 ADDR1에서 저장된 메타 명령 "END,ADDR = -, LEN = 8"이 실행된다. 메타 명령 "END,ADDR = -, LEN =8"을 실행하는데, 8 쿼드 데이터가 전달된다. 그후, 처리는 끝난다.

도 11은 메타 명령의 동작 코드가 "REF"일 때 처리기에 의해서 실행되는 동작을 설명하기 위해서 사용되는 도면이다. 먼저, 메인 DMAC(46)는 태그 어드레스 레지스터 Dn_TADR에 저장된 어드레스 ADDR0로부터 메타 명령 워드로서 1워드를 판독한다. 메타 명령이 "REF, ADDR = ADDR1, LEN =2"라고 가정한다. 메타 명령 "REF, ADDR = ADDR2, LEN =2"의 실행에서, 어드레스 ADDR2에 저장된 2 쿼드 데이터가 전송된다. 그후, 이러한 메타 명령에 이어지는 메타 명령 "REF, ADDR =ADDR1, LEN =8"이 실행된다.

메타 명령 "REF, ADDR =ADDR1, LEN =8"의 실행에 있어서, 어드레스 ADDR1에 저장된 8 쿼드 데이터가 전송된다. 그후, 이러한 메타 명령에 이어지는 메타 명령 "END, ADDR = -, LEN =8"이 실행된다. 메타 명령 "END, ADDR = -, LEN = 8"을 실행하여, 8 쿼드 데이터가 전송된다. 그후, 처리가 끝난다.

도 12는 메타 명령의 동작 코드가 "CALL" 및 "RET"일 때 프로세서에 의해서 실행된 동작을 설명하기 위해 사용된 도면이다. 먼저, 메인 DMAC(46)는 태그 어드레스 레지스터 Dn_TADR내에 저장된 어드레스 ADDR0로부터 메타 명령 워드로서 1 워드를 판독한다. 메타 명령이 "CALL, ADDR = ADDR1, LEN = 0"라고 가정한다. (메타 명령 "CALL, ADDR

=ADDR1, LEN=0"를 실행하면, 이러한 메타 명령에 이어지는 데이터가 전송되지 않는다. 왜냐하면 LEN =0 이기 때문이다. 메타 명령 "CALL, ADDR = ADDR2, LEN =8"에 이어지는 메타 명령의 어드레스는 리턴 어드레스로서 제 1 레지스터에 밀린다.) 메타 명령 "CALL, ADDR = ADDR1, LEN =0"의 실행 후에, 도12의 우측면에 도시되며 어드레스 ADDR1에 저장된 메타 명령 "CALL, ADDR = ADDR2, LEN =8"이 실행된다. 메타 명령 "CALL, ADDR =ADDR2, LEN =8"의 실행에서, 이러한 메타 명령에 이어지는 8 쿼드 데이터가 전송된다(, 이러한 패킷에 이어지는 메터 명령 "RET, ADDR = -, LEN =0"의 어드레스를 리턴 어드레스로서 제 2 레지스터로 민다.). 그후, 어드레스 ADDR2에 저장된 메타 명령 "RET, ADDR = -, LEN = 8"이 실행된다.

메타 명령의 "RET, ADDR = -, LEN = 8"의 실행에 있어서, 이러한 메타 명령에 이어지는 8 쿼드 데이터가 전송된다. 그후, 메타 명령 "RET, ADDR = -, LEN = 0"로서, 우측에서 도시된 메타 명령 "CALL, ADDR = ADDR2, LEN = 8"의 실행에서 제2 레지스터에서 푸시된 어드레스가 실행된다. 메타 명령 "REM, ADDR = -, LEN = 0"의 실행에서, 이러한 명령에 이어지는 데이터가 전송되지 않는다. 왜냐하면 LEN=0 이기 때문이다. 그후, 좌측면에서 메타 명령 "CALL, ADDR = ADDR2, LEN =8"에서, 그 어드레스가 메타 명령 "CALL, ADDR =ADDR1, LEN =0"의 실행에서 레지스터가 먼저 푸시된 것으로서, 실행된다. 좌측면의 메타 명령 "CALL, ADDR = ADDR2, LEN = 8"의 실행에서, 이러한 메타 명령을 따르는 8 쿼드 데이터가 전달된다(, 리턴 어드레스로서 제 1 레지스터로 이러한 패킷을 따르는 메타 명령 "END, ADDR = -, LEN =0"의 드레스를 푸시한다.). 그후, ADDR2에 저장된 메타 명령 "RET, ADDR = -, LEN = 8"이 실행된다.

메타 명령 "RET, ADDR= -, LEN=8" 의 실행에서, 이러한 메타 명령에 이어지는 8쿼드 데이터가 전송된다. 그후, 메타 명령 "END, ADDR= -, LEN=0"으로서, 좌측면에서 그 어드레스가 메타 명령 "CALL, ADDR =ADDR2, LEN =8"의 실행에서 제 1 레지스터 내에 푸시된, 명령이 실행된다. 메타 명령 "END,M ADDR = -, LEN = 0"의 실행에서, 데이터가 전송되지 않는다. 왜냐하면 LEN = 0이기 때문이다. 그후, 처리는 끝난다.

전술된 바와 같이, 데이터 전송은 데이터에 내장된 메타 명령에 따라서 제어된다.

도 13은 데이터의 전송이 데이터에 내장된 메타 명령에 따라서 제어되는 상태를 도시하는 도면이다. 메인 CPU(44)가 디스플레이 리스트(디스플레이 리스트 #0)를 만들고 있을 때, 1 프레임만큼 디스플레이 리스트 #0에 앞서는 디스플레이 리스트(디스플레이 리스트 #1)와 결합된 데이터는 제 2 벡터 처리 엔진(VPE1)(48)에 전송된다.

무엇보다도, 제 1 벡터 처리 엔진(71)과 관련된 메인 CPU(44)는, "NEXT" 동작 코드를 갖는 메타 명령, 내용, "REF" 동작 코드를 갖는 메타 명령, "REF" 동작 코드를 갖는 메타 명령, 매트릭스, "REF" 동작 코드를 갖는 메타 명령, 매트릭스, "REF" 명령 코드를 갖는 메타 명령, "REF" 동작 코드를 갖는 메타 명령, "REF" 명령 코드를 갖는 메타 명령, "REF" 명령 코드를 갖는 메타 명령, 매트릭스 및 "REF" 명령 코드를 갖는 메타 명령을 포함하는 디스플레이 리스트(디스플레이 리스트 #0)를 만든다.

전술된 바와 같이, 메인 CPU(44)가 제 1 벡터 처리 엔진(71)과 관련된 디스플레이 리스트(디스플레이 리스트 #0)를 만드는 동안에, 1 프레임만큼 디스플레이 리스트 #0 앞서는 디스플레이 리스트(디스플레이 리스트 #1)와 관련된 데이터가 다음과 같은 제 2 벡터 처리 엔진(VPE1)(48)에 전송된다. 먼저, 디스플레이 리스트 #1의 헤드에서 "NEXT" 동작 코드를 갖는 메타 명령은 연속되는 내용을 제 2 벡터 처리 엔진(48)으로 전송하기 위해서 실행된다. 그후, 전송된 내용에 이어지는 "REF" 동작 코드를 갖는 제 1 메타 명령은 메인 메모리내의 오브젝트 데이터 베이스에 저장된 프로그램(0)을 판독하기 위해서 실행되며 (프로그램을 제 2 벡터 처리 엔진(48)에 전송한다). 예를 들면, 정점 데이터 세트, 즉, 도 13에 도시된 오브젝트의 내용은 어딘가에 저장될 수 있으며 디스플레이 리스트중 매트릭스만이 업데이트된다. 따라서 관찰자의 눈에 기초해서 화상을 발생하는 것이 가능하다. 이러한 방법으로, 그 내용을 갖는 데이터는 (디스플레이 리스트 상에서 데이터를 포함할 필요없이 제 2 벡터 처리 엔진(48)에 데이터를 전송하기 위해서) 메타 명령을 이용함으로써 프로그램이 디스플레이 리스트로부터 판독되듯이 프레임간에 변동되지 않는다. 문자들과 그것들 사이의 내용 데이터와 같은 고정된 데이터는 다양한 디스플레이 리스트내의 메타 명령에 의해서 공유될 수 있다. 그 결과, 디스플레이 리스트는 쉽게 디스플레이 리스트상에 포함된 위치 데이터(즉, 매트릭스)만을 업데이트함으로써 만들어지며, 그 내용이 프레임 간에 변동하며, 이미 만들어진 현재의 디스플레이 리스트의 위치 데이터를 업데이트함으로써 만들어진다.

오브젝트 데이터 베이스가 삼차원 오브젝트(오브젝트의 정점으로서 일컬어짐)를 묘사하기 위한 3 차원 데이터를 포함하며 오브젝트 데이터를 변환하기 위한 프로그램을 포함한다. 이에 더해서, 구조 맵핑이 오브젝트의 장식(ornament)상에서 실행되며, 정점(정점 화상)으로 사용된 화상 데이터는 오브젝트 데이터 베이스내에 저장된다.

이어서, 전술된 "REF" 메타 명령에 이어지는 "REF" 동작 코드를 갖는 제 2 메타 명령은 오브젝트(0)의 3 차원 좌표 데이터 정점을 판독하기 위해서 실행되며, 즉, 오브젝트(0)의 정점 좌표의 판독을 실행하며(오브젝트(0)의 정점 좌표를 제 2 벡터 처리 엔진(48)으로 전송한다). 그후, 제 1 메트릭스는 제 2 벡터 처리 엔진(48)에 전송된다(이는 제 1 메트릭스의 헤드에서 제 2 "REF" 메타 명령에 이어지는 NEXT 동작 코드를 갖는 메타 명령을 실행함으로써 전송된다). 이어서, 전송된 제 1 매트릭스에 이어서 "REF" 동작 코드를 갖는 제 3 메타 명령이 오브젝트(1)의 3 차원 좌표 데이터 정점을 판독하기 위해서 실행되며, 즉, 오브젝트(1)의 정점 좌표를 판독하기해서 실행되며 (그후 오브젝트(1)의 정점 좌표를 제 2 벡터 처리 엔진(48)으로 전송한다.)

그후, 제 3 "REF" 메타 명령에 이어지는 제 2 매트릭스는 제 2 매트릭스의 헤드에서 "REF" 메타 명령에 이어지는 NEXT 동작 코드를 갖는 메타 명령의 실행에 의해서 제 2 벡터 처리 엔진(48)에 전송된다. 계속해서, 전송된 제 2 매트릭스에 이어지는 "REF" 동작 코드를 갖는 제 4 메타 명령은 오브젝트(1)의 3 차원 좌표 데이터 정점을 다시 판독하기 위해서 실행되며, 즉, 오브젝트(1)의 정점 좌표를 판독하기 위해서 실행되며 (오브젝트(1)의 정점 좌표를 제 2 벡터 처리 엔진(48)에 전송한다). 그후, 제 3 매트릭스는 제 3 매트릭스의 헤드에서 제 4 "REF" 메타 명령에 이어서 NEXT 동작 코드를 갖는 메타 명령의 실행에 의해서 제 2 벡터 처리 엔진(48)에 전송된다. 이어서, 전송된 제 3 매트릭스에 이어서 "REF" 동작 코드를 갖는 제 5 메타 명령은 프로그램(3)을 판독하기 위해서 실행되며 (프로그램을 제 2 벡터 처리 엔진(48)으로 전송한다). 그후, 전술된 제 5 "REF" 명령에 이어지는 "REF" 동작 코드를 갖는 제 6 메타 명령이 프레임 메모리(58)로부터 정점 화상 데이터를 판독하며 데이터를 제 2 벡터 처리 엔진(48)으로 전송하기 위해서 실행된다.

구조 화상 데이터가 프레임 메모리(58)내에 저장되지 않으면, 구조 화상 데이터는 오브젝트(4)의 오브젝트 데이터 정점이 "REF"를 갖는 이어지는 제 7 메타 명령에 의해서 전송되기 전에 프레임 메모리(58)에 전송된다. 정점 화상 데이터가 MDEC(47)로부터 나오는 드로잉 데이터이거나 또는 서브버스(42)로부터 나오는 전송된 데이터이면, 구조 화상 데이터는 프레임들간에 변동된다. 이 경우에, 스톨 기능은 후술되듯이 데이터 전송의 동기화를 설정하기 위해서 사용된다.

화상 데이터가 제 2 벡터 처리 엔진(48)에 전송되는 동안에, 제 2 벡터 처리 엔진(48)에 의해서 실행된 처리는 일시적으로 지연된다. 이러한 기간동안에 다른 DMA 채널에 의해서 실행된 활동을 정지시키므로서 화상 데이터의 전송 기간을 최소화할 필요가 있다. 다른 DMA 채널에 의해서 실행된 활동을 정지시키는 것은 화상 데이터를 전송하는데 사용되는 메타 명령에서 선정된 제어 비트에 의해서 규정될 수 있다. 예를 들어, 도 9에 도시된 메타 명령의 24번째 및 25번째 비트가 그러한 제어 비트로서 사용된다.

그후, "REF" 동작 코드를 갖는 마지막 (제 7) 메타 명령은 오브젝트(4)의 3 차원 좌표 데이터 정점을 판독하기 위해서 실행되며 (데이터를 제 2 벡터 처리 엔진(48)으로 전송한다.) 그후, 마지막 (제 4) 매트릭스는 제 2 벡터 처리 엔진(48)에 전송되는데 (제 4 매트릭스의 헤드에서 제 7 "REF" 메타 명령에 이어지는 NEXT 동작 코드를 갖는 메타 명령의 실행에 의해서 전송된다.) 마지막으로, "REF" 동작 코드를 갖는 메타 명령은 전송 처리를 끝내기 위해서 실행된다.

도 14는 전술된 스톨 제어를 설명하기 위해서 사용된 도면이다. 데이터가 장치(0)로부터 메인 메모리로 전송되며 메인 메모리로부터 장치(1)로 전송되는데 메인 메모리 내에서 데이터 저장 어드레스의 증가 순서로 전송된다. 이 경우에, 데이터가 장치(1)로 전송되는 메인 메모리내의 어드레스는 메인 메모리내의 어드레스를 한번에 초과할 수 있으며 이 경우 데이터는 어떤 이유로 가장 최근에 장치(0)로부터 전송되었다. 그러한 시간 동안에, 메인 메모리로부터 장치(1)로 데이터가 전송하는 것은 스톨된 상태에서 이루어진다.

도 13에 도시된 데이터 전송의 예에서, 구조 화상 데이터는 MDEC(47)내의 저장 메모리로부터 메인 메모리(45)로 전송되며 메인 메모리로부터 제 2 벡터 처리 엔진(48)으로 전송되는데 메인 메모리(45)내의 데이터 저장 어드레스의 증가하는 순서로 전송된다. 이 경우에, 데이터가 제 2 벡터 처리 엔진(48)으로 전송되는 메인 메모리(45)내의 어드레스는, 데이터가 어떤 이유로 가장 최근에 VDEC(47)내에 저장 메모리로부터 전송된 메인 메모리(45)내에서 어드레스를 한반에 초과할 수 있다. 그러한 시간동안에, 메인 메모리(45)로부터 제 2 벡터 처리 엔진(48)으로 구조 화상 데이터를 전송하는 것은 전송 동기화를 설정하기 위해서 스톨된 상태에서 설정된다.

전술된 바와 같이, 메인 DMAC(46)는 디스플레이 리스트로부터 메타 명령을 페치하며 데이터를 프로세서에 분배하기 위해서 메타 명령을 실행한다. 그 결과로서, 디스플레이 리스트가 미리 처리기에 의해서 발생되는 시간에 데이터내의 순서 및 형태 또는 데이터 전송 순위를 프로그램함으로써, 데이터는 데이터의 특징에 따라서 최적의 방법으로 전송될 수 있다. 이에 더해서, 미리 디스플레이 리스트와 같은 리스트의 형태로 데이터를 전송하는 순서를 프로세서가 규정하게 함으로써, 메모리내의 전송 작업을 위한 낭비되는 카피 데이터를 유지하게 하는 것이 필요없다. 그 결과, 메모리에 대한 낭비되는 액세스 및 디스플레이 리스트의 크기에 대한 숫자는 감소된다.

더욱이, 개별 디스플레이 리스트에 대한 2 위치에서 분리되어 프레임간에 변동하며 전송되는 각각의 데이터부를 저장하는 것이 필요하다. 프레임간에 변동되지 않는 디스플레이 리스트의 일부가 모든 디스플레이 리스트에 공통인 메모리 영역내에 저장될 수 있다. 따라서, 디스플레이 리스트를 저장하는데 요구되는 메모리의 크기는 감소될 수 있다. 말하자면, 복수의 디스플레이 리스트가 작은 저장 용량을 갖는 메모리 내에서 저장될 수 있다.

그 상부에, 데이터가 데이터 내에 내장된 메타 명령에 따라서 전송되므로, 데이터를 판독 및 기록하기 위한 동작의 동기화는 쉽게 복수의 프로세서 가운데서 설정될 수 있다. 그 결과, 복수의 프로세서는 메모리 내에서 더블 버퍼를 제공할 필요없이 메모리를 공유하는 것이 허용된다.

전술된 실시예의 경우에, 데이터는 CD-ROM에 저장된다. 그러나, 다른 기록 매체도 사용될 수 있음을 알 수 있다.

청구 범위 제 1 항에 따른 정보 처리 장치 및 청구 범위 제 5항에 따른 정보처리 방법에서, 제 1 발생 수단이 처리의 경로에 있으며 실행 수단이 대기 상태에 있을 때, 제 2 전송 수단은 제 2 명령을 실행 수단에 공급하며, 제 2 명령을 수신하여, 실행 수단은 제 2 명령을 실행한다.

발명의 효과

본 발명을 실시함으로써 복수의 CPU에 우선 순위를 지정하고, CPU가 동시에 동작하게 하며, 가장 높은 순위를 갖는 CPU에 의해서 발생된 디스플레이 장치로 시작하는 우선 순위에 기초하여 CPU에 의해서 발생된 디스플레이 리스트를 처리함으로써 높은 등급의 효율로 데이터가 처리되게 하는 정보 처리 장치 및 정보 처리 방법을 제공할 수 있게 된다.

(57) 청구의 범위

청구항 1.

정보 처리 장치에 있어서,

제 1 명령을 발생하기 위한 처리를 실행하는 제 1 발생 수단;

제 2 명령을 발생하기 위한 처리를 실행하는 제 2 발생 수단;

소정의 타이밍으로 상기 제 1 명령으로부터 상기 제 2 명령으로 그리고 그 역으로 스위칭함으로써 상기 제 1 및 제 2 명령들을 실행하는 실행 수단;

상기 제 1 명령을 상기 실행 수단에 전송하는 제 1 전송 수단;

상기 제 2 명령을 상기 실행 수단에 전송하는 제 2 전송 수단을 포함하며;

상기 제 1 발생 수단이 처리중에 있고 상기 실행 수단이 대기 상태에 있을 때, 상기 제 2 전송 수단은 만약 있다면 상기 제 2 명령을 상기 실행 수단에 공급하고, 상기 실행 수단은 상기 제 2 명령을 수신하여 상기 제 2 명령을 실행하는, 정보 처리 장치.

청구항 2.

제 1 항에 있어서,

상기 실행 수단은 상기 제 1 및 제 2 발생 수단만큼 많은 상기 제 1 및 제 2 명령의 실행을 위한 파라미터를 유지하고;

상기 제 1 명령이 상기 제 1 발생 수단중 임의의 하나로부터 수신될 때, 상기 실행 수단은 상기 제 1 명령을 공급하는 상기 제 1 발생 수단과 관련된 상기 파라미터들중 하나를 이용함으로써 상기 제 1 명령을 실행하고, 상기 제 2 명령이 상기 제 2 발생 수단중 임의의 하나로부터 수신될 때, 상기 실행 수단은 상기 제 2 명령을 공급하는 상기 제 2 발생 수단과 관련된 상기 파라미터들중 하나를 이용함으로써 상기 제 2 명령을 실행하는, 정보 처리 장치.

청구항 3.

제 1 항에 있어서,

상기 장치에는 상기 제 2 발생 수단에 의해서 발생된 상기 제 2 명령을 저장하는 저장 수단이 더 구비되며;

상기 제 2 전송 수단은 상기 저장 수단에 저장된 상기 제 2 명령을 상기 실행 수단에 전송하는, 정보 처리 장치.

청구항 4.

제 1 항에 있어서,

상기 제 1 명령은 타입(type) 처리를 위해서 발생되며 상기 제 2 명령은 넌타입(non-type) 처리를 위해서 발생되는, 정보 처리 장치.

청구항 5.

제 1 명령을 발생하기 위한 처리를 실행하는 제 1 발생 수단;

제 2 명령을 발생하기 위한 처리를 실행하는 제 2 발생 수단;

소정의 타이밍으로 상기 제 1 명령으로부터 상기 제 2 명령으로 그리고 그 역으로 스위칭함으로써 상기 제 1 및 제 2 명령들을 실행하는 실행 수단;

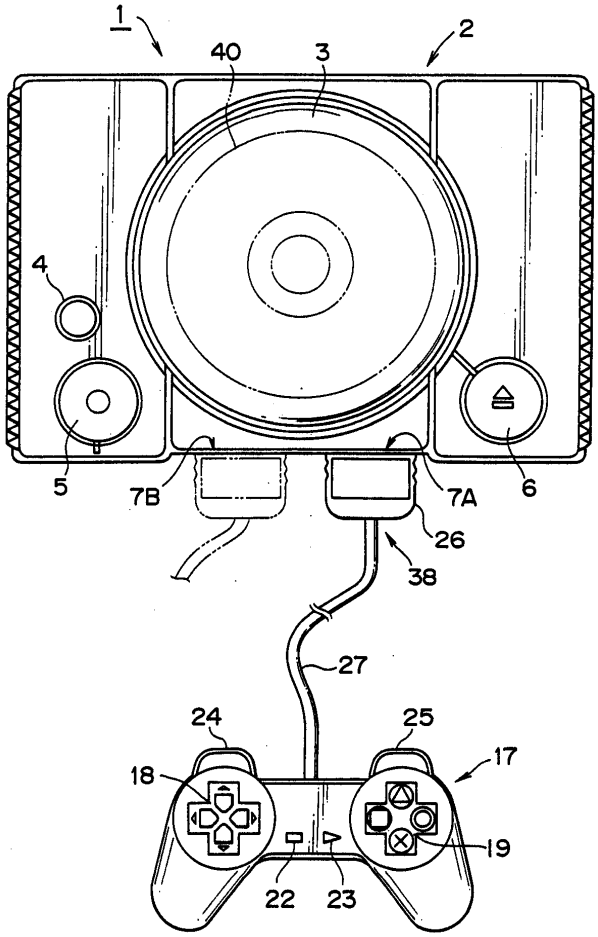
상기 제 1 명령을 상기 실행 수단에 전송하는 제 1 전송 수단;

상기 제 2 명령을 상기 실행 수단에 전송하는 제 2 전송 수단을 포함하는 정보 처리 장치에 적합한 정보 처리 방법에 있어서,

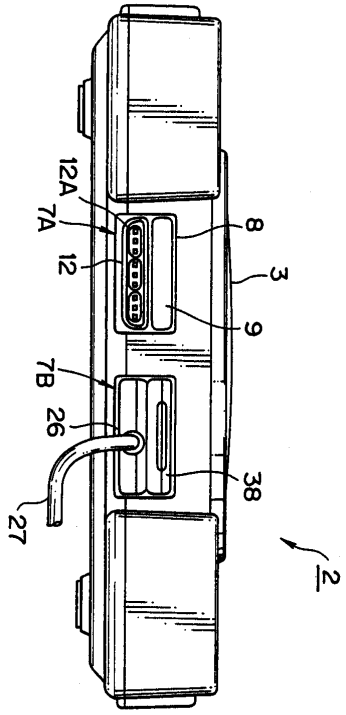
상기 제 1 발생 수단이 처리중에 있고 상기 실행 수단이 대기 상태에 있을 때, 상기 제 2 전송 수단은 만약 있다면 상기 제 2 명령을 상기 실행 수단에 공급하고, 상기 실행 수단은 상기 제 2 명령을 수신하여 상기 제 2 명령을 실행하는, 정보 처리 방법.

도면

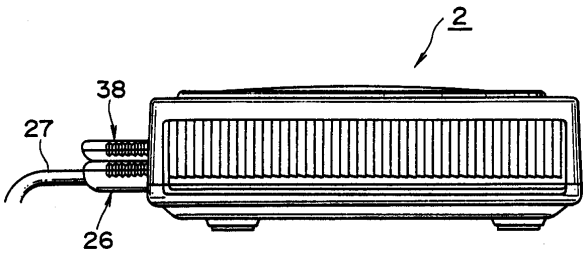
도면1


1
2
40
3
4
5
7B
7A
6
26
38
27
24
25
17
18
19
22
23

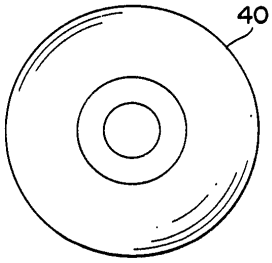
도면2


12A
7A
8
12
3
9
7B
26
38
27
2

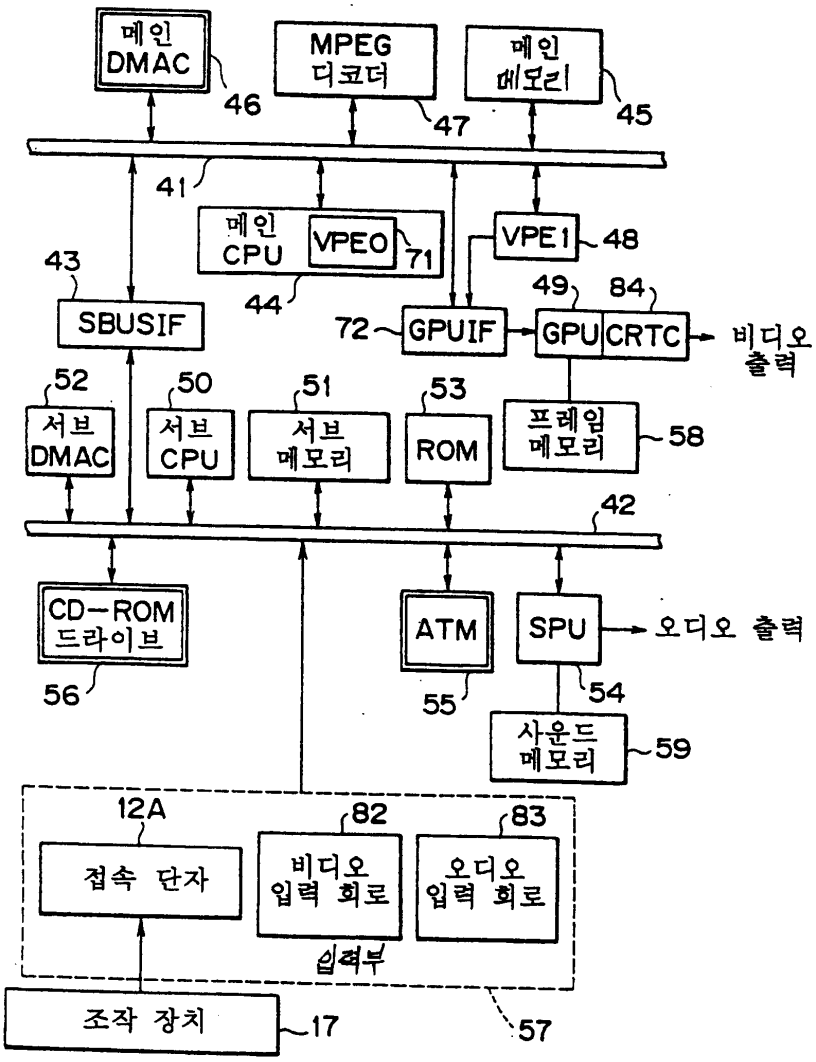
도면3



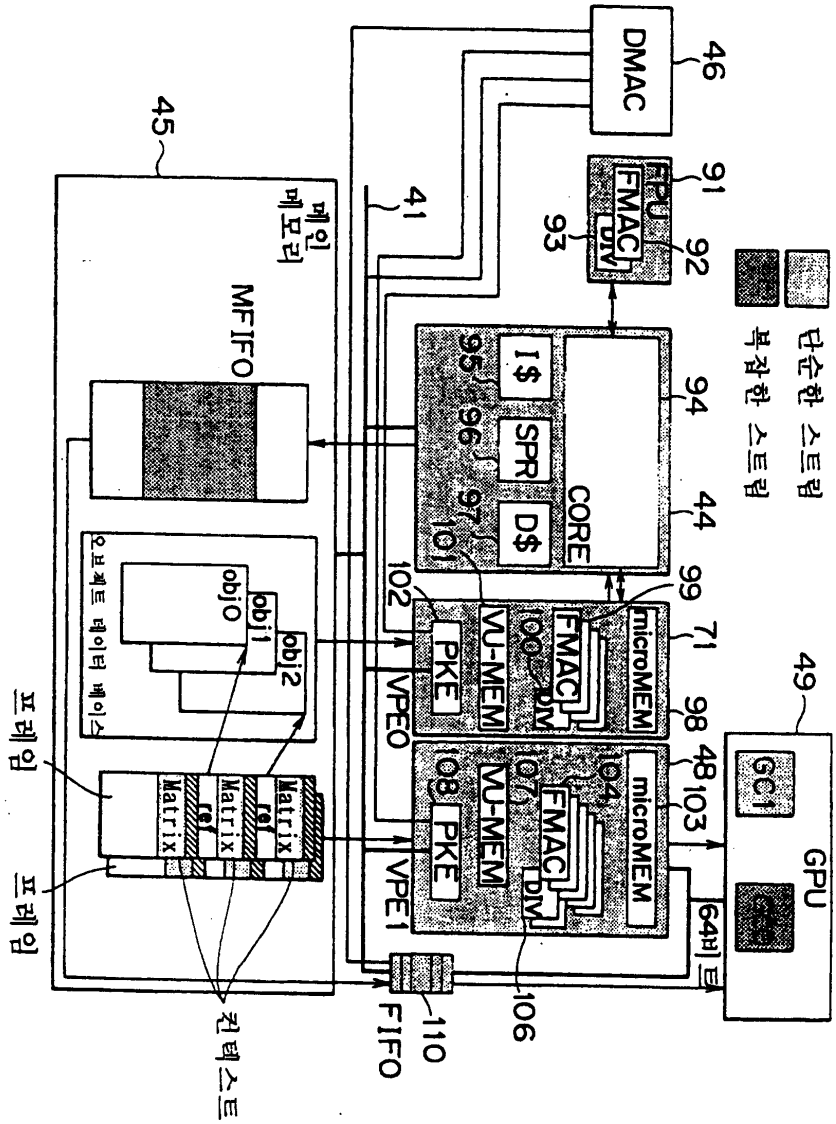
도면4



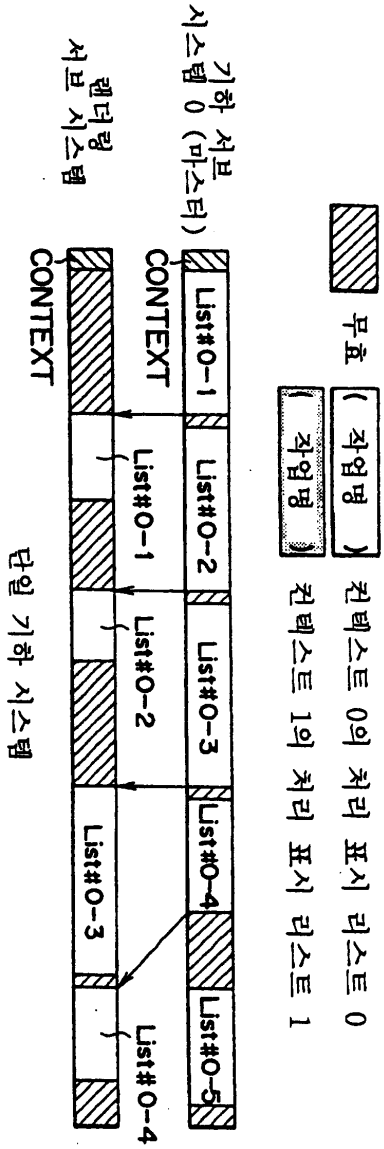
도면5


메인 DMAC
46
MPEG 디코더
47
메인 메모리
45
41
메인 CPU
VPE0
71
VPE1
48
43
44
49
84
SBUSIF
72
GPUIF
GPU
CRTC
비디오 출력
52
50
51
53
서브 DMAC
서브 CPU
서브 메모리
ROM
프레임 메모리
58
42
CD-ROM 드라이브
56
ATM
55
SPU
오디오 출력
54
사운드 메모리
59
12A
82
83
접속 단자
비디오 입력 회로
오디오 입력 회로
입력부
조작 장치
17
57

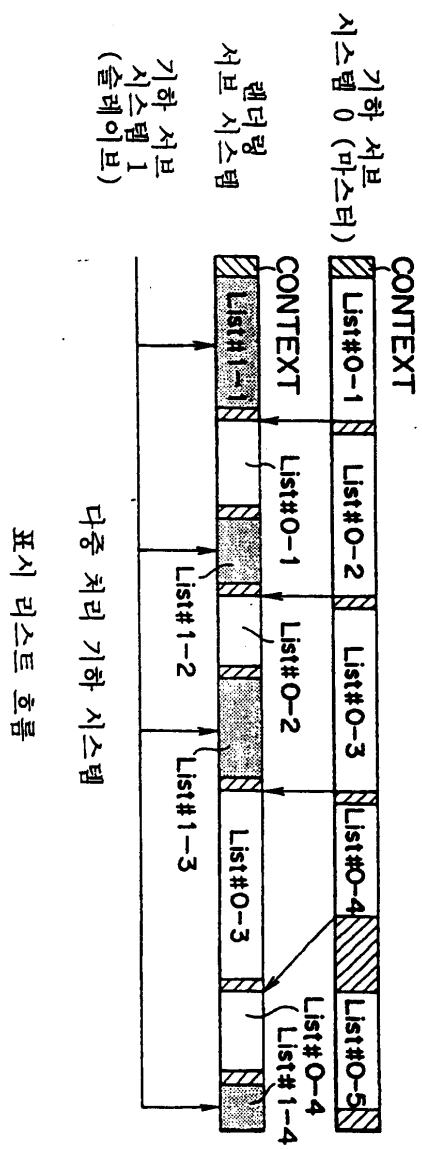
도면6


DMAC
46
45
41
FPU
FMAC
DIV
91
92
93
단순한 스트림
복잡한 스트림
I$
SPR
D$
CORE
94
95
96
97
101
44
99
71
98
microMEM
FMAC
DIV
VU-MEM
PKE
100
102
VPE0
48
103
104
107
108
VPE1
106
110
FIFO
49
GC1
GPU
64비트
메인 메모리
MFIFO
obj0
obj1
obj2
오브젝트 데이터 베이스
Matrix
ref
프레임
프레임
컨텍스트

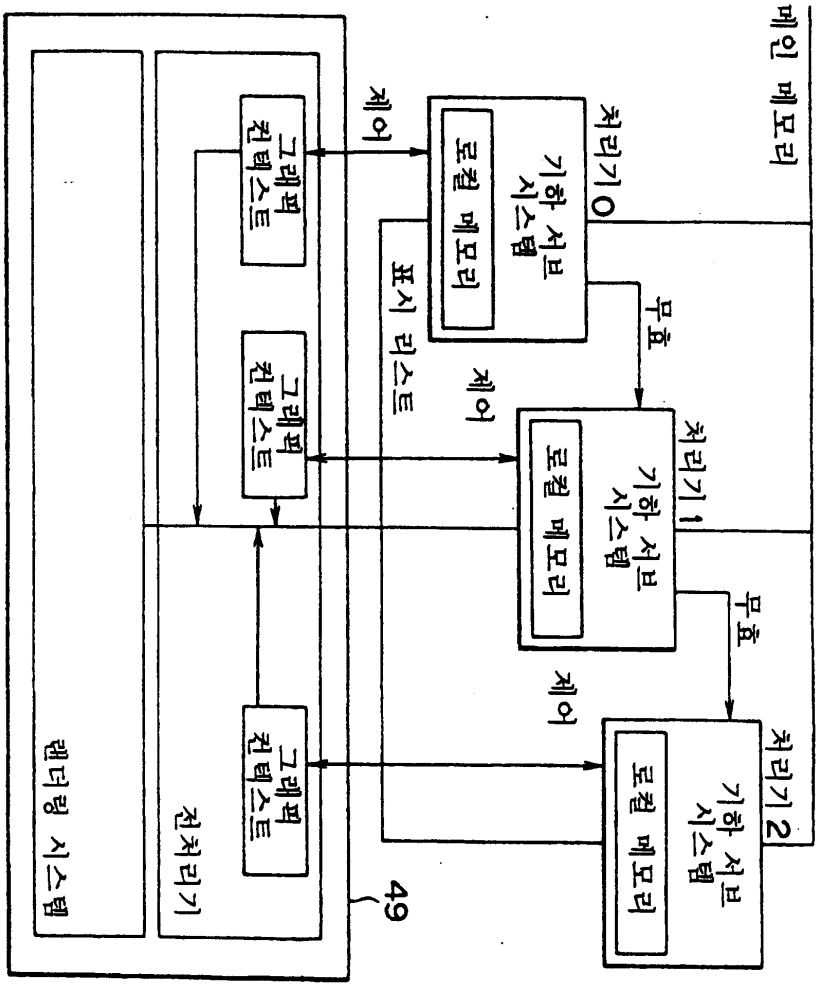
도면7a


무효
작업명
작업명
컨텍스트 0의 처리 표시 리스트 0
컨텍스트 1의 처리 표시 리스트 1
기하 서브
시스템 0 (마스터)
래더링
서브 시스템
CONTEXT
CONTEXT
List#0-1
List#0-2
List#0-3
List#0-4
List#0-5
List#0-1
List#0-2
List#0-3
List#0-4
단일 기하 시스템

도면7b


기하 서브
시스템 0 (마스터)
렌더링
서브 시스템
기하 서브
시스템 1
(슬레이브)
CONTEXT
List#0-1
List#0-2
List#0-3
List#0-4
List#0-5
CONTEXT
List#1-1
List#0-1
List#0-2
List#0-3
List#0-4
List#1-4
List#1-2
List#1-3
다중 처리 기하 시스템
표시 리스트 흐름

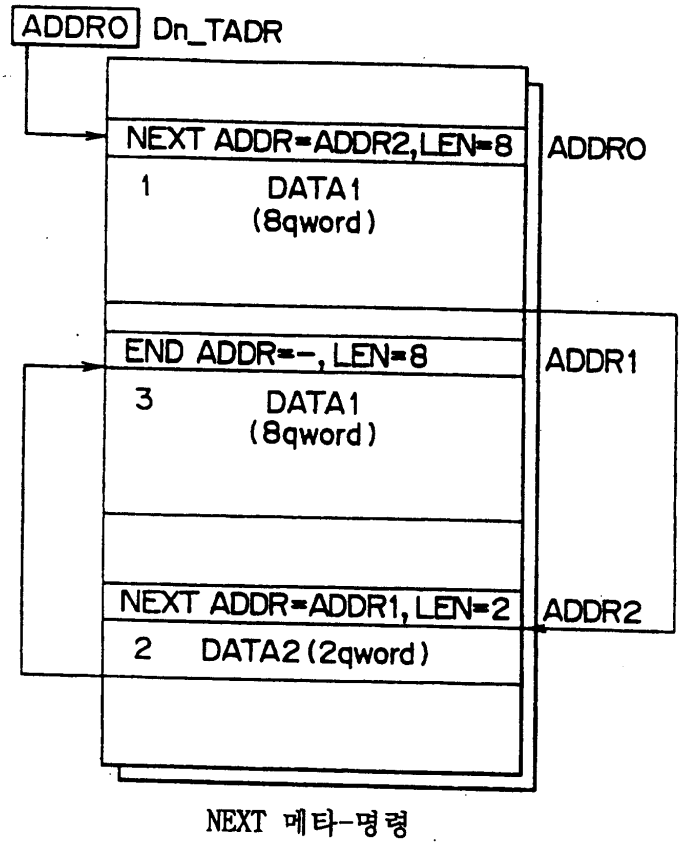
도면8


메인 메모리
처리기0
기하 서브 시스템
로컬 메모리
표시 리스트
제어
그래픽 컨텍스트
무효
처리기1
기하 서브 시스템
로컬 메모리
제어
그래픽 컨텍스트
무효
처리기2
기하 서브 시스템
로컬 메모리
제어
그래픽 컨텍스트
전처리기
렌더링 시스템
49

도면9

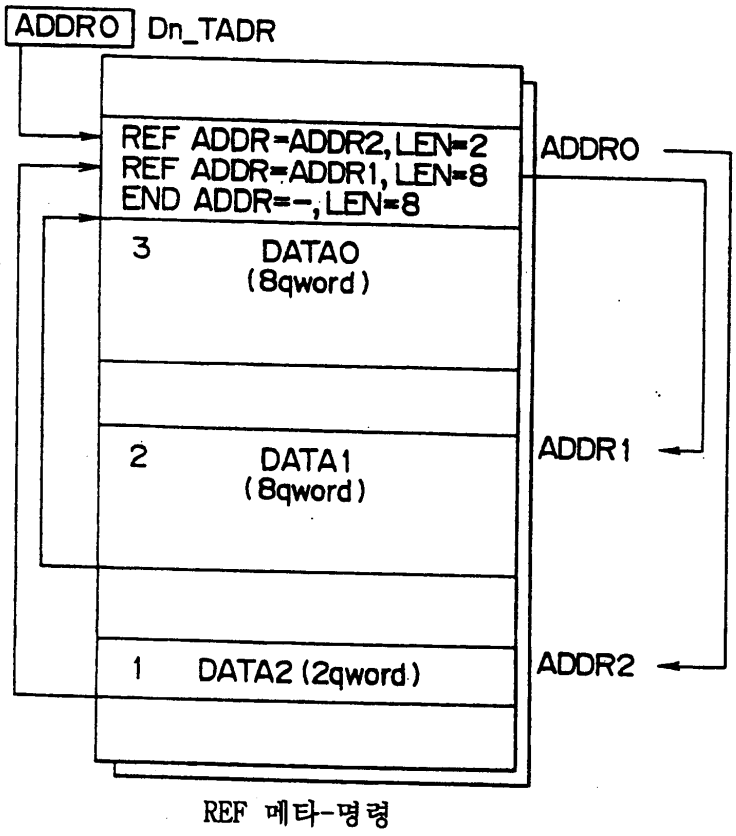


도면10



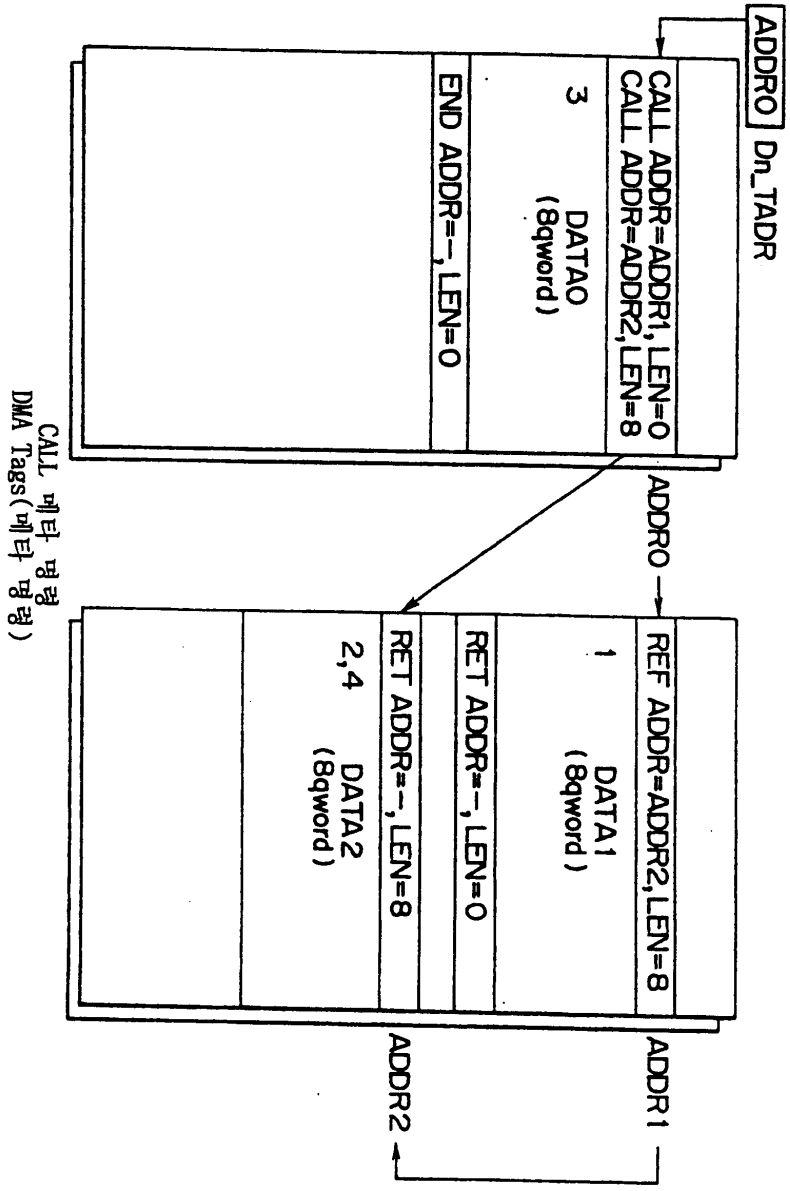
NEXT 메타-명령

도면11

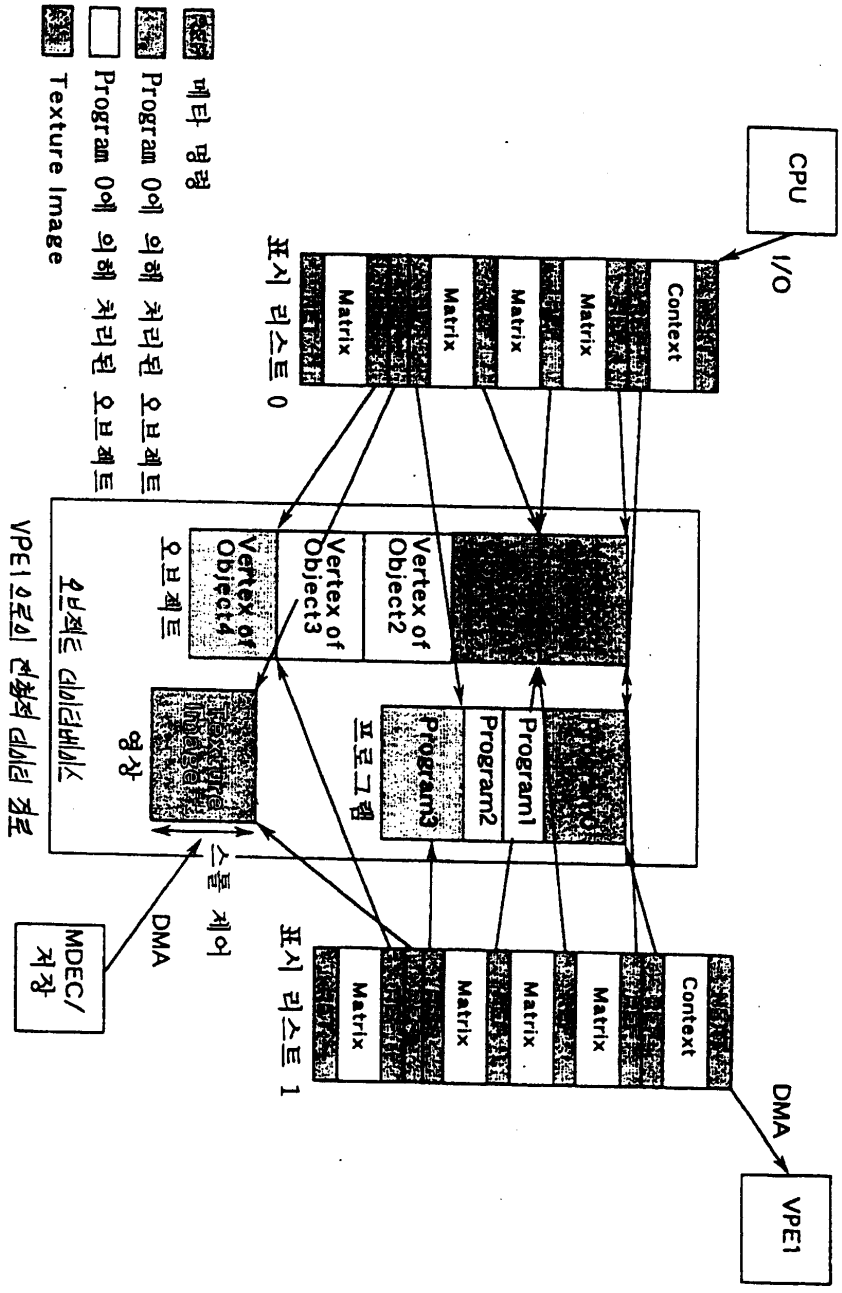


REF 메타-명령

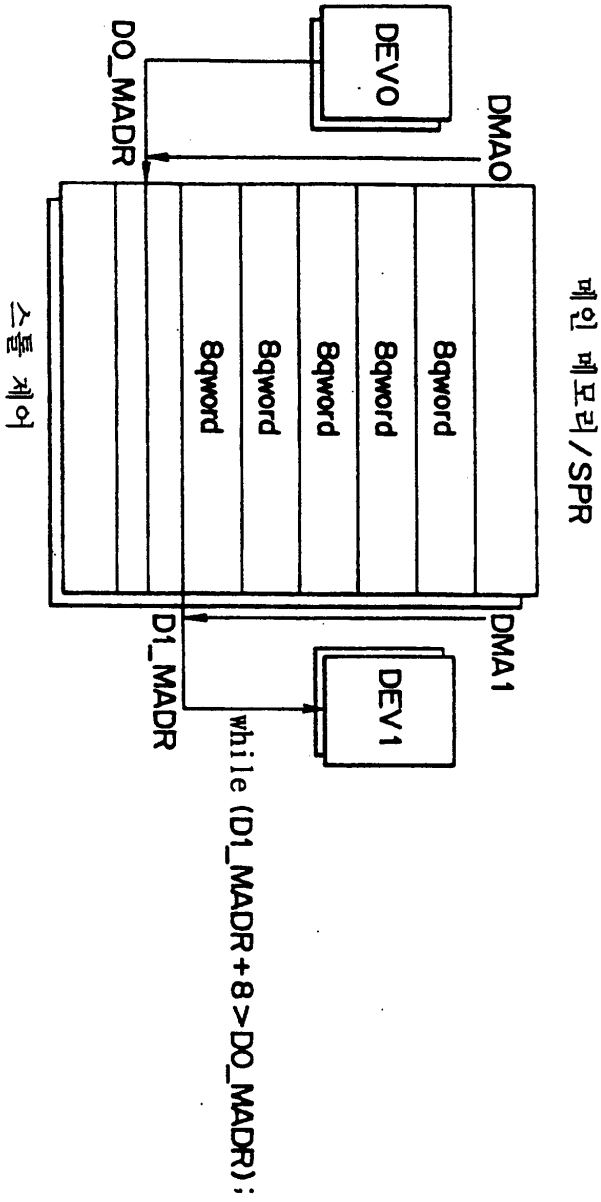
도면12


ADDR0
Dn_TADR
CALL ADDR=ADDR1, LEN=0
CALL ADDR=ADDR2, LEN=8
3
DATA0
(8qword)
END ADDR=-, LEN=0
CALL 메타 명령
DMA Tags(메타 명령)
ADDR0
REF ADDR=ADDR2, LEN=8
1
DATA1
(8qword)
RET ADDR=-, LEN=0
RET ADDR=-, LEN=8
2,4
DATA2
(8qword)
ADDR1
ADDR2

도면13


메타 명령
Program 0에 의해 처리된 오브젝트
Program 0에 의해 처리된 오브젝트
Texture Image
CPU
I/O
표시 리스트 0
Context
Matrix
Matrix
Matrix
Matrix
Matrix
Vertex of Object4
Vertex of Object3
Vertex of Object2
오브젝트
Program3
Program2
Program1
프로그램
Texture Image
영상
오브젝트 데이터베이스
VPE1 으로의 변환적 데이터 경로
스톨 제어
DMA
MDEC/ 저장
표시 리스트 1
Context
Matrix
Matrix
Matrix
Matrix
Matrix
DMA
VPE1

도면14


DEV0
DMA0
DO_MADR
8qword
8qword
8qword
8qword
8qword
8qword
메인 메모리/SPR
스톨 제어
DMA1
D1_MADR
DEV1
while (D1_MADR+8>DO_MADR);