(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) Int. Cl.[7]
G06T 17/00

(45) 공고일자 2000년10월16일
(11) 등록번호 10-0270198
(24) 등록일자 2000년07월28일

(21) 출원번호 10-1992-0018674
(22) 출원일자 1992년10월10일
(65) 공개번호 특1993-0016869
(43) 공개일자 1993년08월30일

(30) 우선권주장 07/827,201 1992년01월30일 미국(US)
(73) 특허권자 에이/엔인코퍼레이티드
미합중국 워싱턴 레드몬드 사서함 957 엔.이.제 150 애버뉴 4820
(72) 발명자 제레미이.산
영국 런던 밀힐업 힐로드73
벤치이즈
영국 하트퍼드셔 멜버른 로이스톤 돌핀레인23
칼엔.그레이엄
영국런던엔.칭포드우드랜드로드15
피터알.워네스
영국 런던 이스트함 월폴로드12A
(74) 대리인 김명신

심사관 : 강갑연

(54) 비디오게임시스템등에 사용하기 위한 픽셀/문자변환 하드웨어를 갖는 프로그램가능 그래픽프로세서

요약

주 정보처리시스템과 연결된 이동식 외부메모리유닛에 내장되도록 설계되어 전적으로 프로그램가능한 그래픽프로세서가 공지된다.

한 예시적실시예에서, 주비디오게임시스템 및, 그래픽마이크로프로세서를 수용하고 있는 플러그접속가능한 비디오게임카트리지를 포함하는 비디오 게임시스템이 기술된다.

또한 상기 게임카트리지는 ROM과 RAM을 포함하고 있다.

그래픽코프로세서는 게임카트리지에 내장된 3개의 버스구조를 이용한다.

이 3버스구조를 사용하는 그래픽프로세서는 프로그램 ROM, 외부 RAM 또는 자기의 내부캐시 RAM중 하나에 기억되어 있는 프로그램을 실행한다.

전적으로 유저(사용자)에 의해 프로그램 가능한 그래픽코프로세서는 3차원 그래픽과 관련된 산술동작을 효과적으로 실행하도록 설계된 명령세트, 예를들면 주비디오게임시스템의 문자맵핑 디스플레이에 개별픽셀을 플로팅하기 위해 전용하드웨어에 의해 실행되는 특수명령을 포함하는 명령세트를 포함하고 있다.

그래픽코프로세서는 주코프로세서와 상호작용하기 때문에 그래픽코프로세서의 16개의 범용레지스터들은 항상 주코프로세서에 의해 액세스될 수 있다.

대표도

도1

명세서

[발명의 명칭]

비디오게임시스템등에 사용하기 위한 픽셀/문자변환 하드웨어를 갖는 프로그램가능 그래픽프로세서

[도면의 간단한 설명]

제1도는 본 발명의 예시적 실시예에 따른 외부메모리시스템의 블럭 다이어그램.

제2도는 본 발명의 예시적 실시예의 그래픽 코프로세서와 함께 사용하기 위한 주 처리시스템의 블럭다이어그램.

제3도는 그래픽 코프로세서를 수용한 게임카트리지의 구성과 주 프로세싱시스템을 수용하고 있는 베이스유닛의 예시적인 기계적 구성을 도시한 사시도.

제4a도 및 제4b도는 본 예시적 실시예에 따른 그래픽 코프로세서의 블럭 다이어그램.

제5도는 그래픽 코프로세서의 동작을 개시시키는 주 프로세싱시스템에 의해 수행될 동작 시퀀스를 설명하는 플로우챠트.

제6도는 제4a도의 산술논리 연산장치의 보다 상세한 블럭다이어그램.

제7도는 제4a도에 도시된 타입의 예시적인 픽셀 플롯회로의 보다 상세한 블럭다이어그램.

제8a도는 플롯제어기가 수신한 입력신호와 그 플롯제어기가 출력한 출력 신호를 도시한 블럭다이어그램.

제8b도는 픽셀 플롯회로의 칼라매트릭스 회로내에 포함된 구성요소들를 나타낸 도면.

제8c도는 픽셀 플롯회로와 관련된 타이밍신호, 제어신호 및 데이타신호를 타나낸 도면.

제9도는 제4a도의 RAM제어기의 보다 상세한 블럭다이어그램.

제9a도는 제9도의 RAM제어기와 관련된 타이밍신호, 제어신호 및 데이타 신호를 도시한 도면.

제10도는 제9도의 중재논리회로를 설명하는 회로도.

제11도는 본 발명의 그래픽 코프로세서의 예시적 실시예에서의 재동기 회로의 다이어그램.

제12도는 제11도의 재동기회로와 관련된 타이밍신호를 나타낸 도면.

제13도는 본 발명의 그래픽 코프로세서에서의 ROM제어기의 보다 상세한 블럭다이어그램.

제14도는 본 발명의 예시적실시예에 따른 그래픽 코프로세서에서의 캐시제어기의 블럭다이어그램.

제15a도는 본 발명에서의 그래픽 코프로세서의 명령디코딩 관련회로를 나타내는 블럭다이어그램.

제15b도는 제15a도의 록어헤드 논뢰회로의 동작을 설명하는 예시적인 타이밍신호를 나타내는 도면.

제16도 및 제17도는 본 발명의 예시적 실시예에 따른 그래픽 코프로세서의 레지스터 제어논리회로를 나타내는 블럭다이어그램.

제18도는 다각형 형성작업을 실행시 그래픽 코프로세서의 동작시퀀스를 설명하는 예시적인 플로우챠트.

제19도, 제20도 및 제21도는 본 발명의 예시적 실시예에 따른 스케일링 및 회전특성을 설명하기 위하여 발생된, 다각형 기초대상체의 예시적인 디스플레이를 나타낸 도면이다.

* 도면의 주요부분에 대한 부호의 설명

1, 18 : 커넥터	2 : 마리오칩
3 : 인증프로세서	4 : 외부서정진동자
6, 8 : RAM	10 : ROM
20 : 제어데크	22 : 주 CPU
24 : PPU	26 : APU
28, 32 : W-RAM	30 : V-RAM
34 : RF변조기	50 : ALU
52 : 플롯하드웨어	60 : 명령디코더
64 : 승산기	66 : 부분곱저항기
68 : 캐시제어기	72 : 캐시태그레지스터
74 : 직접데이타레치	76 : 레지스터블럭
78 : 레지스터 제어논리	94 : 캐시 RAM
98 : 게트 B레지스터	152 : 16비트 가감산기
164 : 156비트 6×1멀티플렉서	166 : CPU플래그회로
210 : 비트펜딩회로	212 : 3×8디멀티플렉서
206 : 8×8칼라매트릭스회로	208 : 비트플레인카운터
216 : 플롯어드레스레지스터	218 : 어드레스비교기
300 : 16비트 데이타레지스터	304 : MUX
306 : 16비트 MUX레지스터	314 : 데이타 뱅크레지스터
316 : 스크린 뱅크레지스터	

[발명의 상세한 설명]

본 발명은 프로그램가능프로세서가 내장된 특수한 외부메모리 유닛을 포함하고 있는 정보처리장치에 관한 것이다.

특히, 본 발명은 일부는 주처리시스템, 예를들면 비디오게임시스템에 의해 실행되고, 일부는 그 주처리시스템의 고속그래픽 처리능력을 확장시키도록 설계된 프로그램가능 마이크로프로세서에 의해 실행되는 프로그램을 저장하기 위한 프로그램메모리를 갖는 이동식 외부메모리유닛에 관한 것이다.

상기 프로그램가능프로세서는 픽셀기초형태를 문자기초형태로 변환시키는 하드웨어를 포함하고 있다.

본 출원은 명칭 "비디오게임 시스템등에 사용하기 위한 프로그램가능 그래픽프로세서를 갖는 외부메모리 시스템"로 산(San)등에 의해 동시에 출원된 건과, 명칭 "비디오게임시스템등에서 사용하기 위한 확자메모리 제어회로를 갖는 그래픽프로세서"로 샌등에 의해 출원된 건과 관련된다.

비디오게임 제어데크내에 내장되어 있는 8비트 마이크로프로세서 및 관련 디스플레이 처리서브시스템을 구비하고 있는 종래기술의 비디오게임기는 8×8비트 매트릭스형태로 게임카트리지내에 문자를 사전에 기억시킴으로써, 그리고 사전에 기억된 그 문자의 다양한 프로그램가능조합을 사용하여 스크린 디스플레이를 생성함으로써 그래픽을 형성해 왔다.

일반적으로, 그러한 종래기술의 비디오게임시스템에서 사용자에 의해 조정되는 다수의 "이동체"나 "스프라이트" 뿐만 아니라 전체적인 디스플레이 후면을 이동시킬 수 있었다.

그러나, 종래기술의 그러한 비디오게임시스템은 회전되어야하고 그리고 각 프레임에 대해 철회되어야만 하는 다각형의 조합으로 구성된 이동체를 포함하는 비디오게임을 실제로 수행할 수 있는 능력이 없다.

이 시스템에 포함된 종래 8비트 프로세서 및 관련 디스플레이 처리회로는 3차원이며 다각형 기초회전체를 회전시키거나 이 회전체를 3차원의 특수효과를 발생시키도록 스케일링하는데 필요한 계산을 수행할 수 없다.

본 발명자들은 정교한 그래픽은 픽셀바이픽셀에 기초한 스크린을 갱신하고 실시간에 기초하여 복잡한 수치계산을 수행하는 것이 요구됨을 인식하여 왔다.

문자에 기초를 둔 종래기술의 비디오게임기는 그러한 기능을 수행할 수 없다.

또한 종래의 8비트 비디오게임기는 픽셀바이픽셀에 기초한 스크린을 빠르게 갱신하는 것이 요구되는 다른 그래픽기술을 효과적으로 수행할 수 없다.

예를들면 그러한 시스템은 보다 정교한 칼라발생과 보다 좋은 그래픽 해상도를 제공한다.

그러한 16비트 비디오게임기는 문자에 기초를 둔 그래픽이나 스프라이프그래픽이 가능한 다양한 비디오게임을 제공하는 문자에 기초를 둔 시스템이다.

또한 이 16비트 비디오게임기는 다수의 칼라후면 플레인의 뒤 또는 앞에 배치된 이동체와 함께 고속으로 그 플레인들을 이동시키는 것이 가능하다.

그러나, 그러한 종래의 16비트 비디오게임기로는 각 프레임동안 변해야만 하는 다각형으로 구성된 정교한 대상체를 디스플레이하는 3차원 특수효과를 갖는 진보된 비디오게임을 실시할 수 없다.

예를들면, 프레임 바이 프레임 기초상에서 확대 및 축소되어야아만 하는 다수의 완전회전체나 스프라이트를 필요로하는 게임은 실제로 그러한 종래 기술의 문자기초 16비트 게임기에서는 실현할 수 없다.

본 발명자들은 완전히 회전하고 스케일링되는 다각형 기초대상체를 포함하는 그러한 게임을 효과적으로 실시할 수 있기 위해서는 다각형의 모서리를 그리는 것과 이러한 다각형 기초대상체에 픽셀바이픽셀 기초상에서 적절한 데이타를 채우는 것이 필요한을 인식하였다.

픽셀바이픽셀 기초상에서 행해져야하는 그러한 작업은 상당히 많은 처리시간을 소비한다.

종래기술에서, 이동게임카트리지는 주 마이크로프로세서와 관련된 어드레스라인의 갯수가 허용하는 것보다도 더 큰 프로그램 메모리어드레스 공간을 존재하는 프로세서가 어드레싱하게 함으로써 게임의 정교성을 향상시키도록 수정되어 왔다.

예를들면, 그러한 종래 8비트시스템은 메모리뱅크스위칭 및 기타 부가적인 기능을 수행하는 멀티메모리 제어기칩을 포함하는 게임카트리지를 사용해 왔다.

그러나, 상기 메모리뱅크스위칭 관련칩은 비디오게임시스템으로 하여금 상기 성질의 고속그래픽처리를 수행하게 할 수 없다.

여기서 기술된 그래픽 마이크로프로세서와 비디오게임시스템은 우수한 특징들을 많이 가지고 있는데 몇가지를 여기서 약술한다.

본 발명의 특수그래픽프로세서는 주 마이크로프로세서와 플러그 접속된다.

그 그래픽프로세서는 처리속도를 최대로하기 위해 상기 마이크로프로세서와 병렬로 동작한다.

한 예시적실시예에서는 그래픽코프로세서를 내장하고 있는 게임카트리지가 ROM과 RAM을 또한 포함하고 있다.

본 발명의 그래픽코프로세서는 자기와 주 마이크로프로세서간의 메모리 거래를 중재한다.

그리고 그 그래픽 코프로세서는 종래의 비디오게임시스템에서는 불가능했던 프로그램의 고속처리를 가능

하게 하기위해 주 마이크로프로세서와 동시에 프로그램을 실행할 수 있다.

또, 그 그래픽코프로세서는 게임카트리지에 포함되어 있는 3버스구조와 함께, 즉 3개의 버스에 연결되어 동작한다.

부언하면, 그 3버스구조는 주 마이크로프로세서와 그래픽코프로세서의 능력, 즉 RAM과 ROM을 효율적으로 사용하는 능력을 최적화함으로써 그 RAM과 ROM을 효율적으로 사용케한다.

사용자가 전적으로 프로그래밍하여 사용할 수 있는 본 발명의 상기 그래픽 코프로세서는 고속으로 처리되도록 설계된 특수한 명령세트를 포함하고 있다.

즉, 그 명령세트는 3차원 그래픽과 관련된 산술동작을 효율적으로 실행하도록 설계되었으며, 예를들면 그 명령세트는 주 비디오게임시스템의 문자맵핑 디스플레이에 개개 픽셀을 플롯시키는 기능을 가진 전용 하드웨어에 의해 실행되는 특수명령을 포함하고 있다.

또, 그 명령세트는 비록 주시스템이 문자에 기초하고 있더라도 개개 픽셀에 어드레스를 지정하는 것을 가능케 함으로써 프로그래머 관점에서 "가상"비트맵을 작성하는 특수픽셀 기초명령들을 포함하고 있다.

그래픽프로세서는 주문자기초 16비트머신이 전형적으로 이용하는 형태의 문자데이타로 픽셀데이타를 계속변경시킨다.

따라서, 예를들면 프로그래머가 픽셀을 플로팅하기 위하여 "PLOT"명령을 사용할지라도 관련데이타가 읽혀지면 그 데이타는 16비트 주머신이 사용할 수 있는 문자기초형태로 변경된다.

특수목적용 픽셀플로팅하드웨어는 고속 3차원그래픽이 효율적으로 실행되도록 하기 위하여 이 명령을 사용하다.

플롯하드웨어는 실시간으로 픽셀좌표어드레싱을 주시스템이 이용하는 성질의 문자맵어드레싱으로 변환하는데에 도움이 된다.

프로세서는 디스플레이상의 각 픽셀의 위치를 설정하는 X 및 Y좌표를 지정함으로써 편리하게 프로그래밍될 수 있다.

따라서, 그래픽동작은 픽셀을 지정하는 프로그래머에 기초하여 수행되며, 그리고 플롯하드웨어는 픽셀사양을 적절히 포햇된 문자데이타로 변환시킨다.

그리고나서, 문자데이타는 주프로세서의 비디오 RAM에서 디스플레이되기를 원하는 장소로 맵핑된다.

플롯하드웨어는 여러가지 플롯팅관련명령에 응답하여 디스플레이 스크린상의 X 및 Y좌표와 특정픽셀에 대한 소정의 색상을 프로그램밍을 이용하여 선택할 수 있음과 동시에, 대응픽셀을 플로팅하고 그 결과 X 및 Y좌표가 주프로세서의 비디오 RAM를 구동하는데 사용되는 형태의 문자설정에 대응하는 어드레스로 변환된다.

본 발명의 제반특성들은 첨부도면과 함께 고찰하면 더 잘 이해될 것이다.

본 예시적실시예에 따르면, 본 발명의 그래픽코프로세서는 닌텐도 오브 아메리카사가 판매하고 있는 모델명 "수퍼 닌텐도 오락시스템(Super NES)"인 16비트 비디오게임시스템과 대화한다.

그 수퍼 닌텐도 오락시스템은 명칭 "비디오처리장치"로 1991년 4월 10일자로 출원된 미국 출원 No. 07/651,265와 명칭 "직접메모리 액세스장치 및 이 장치에 사용되는 외부기억디바이스"로 1991년 8월 26일자로 출원된 미국출원 No. 07/749.530에 일부 기재되어 있다.

이들 출원건은 여기에서 특별히 참조한다.

여기서 주지해야할 사항은 본 발명은 수퍼 NES관련 출원에 한정되는 것이 아니고 다른 비디오게임시스템은 물론 비디오게임시스템이 아닌 정보처리장치에도 사용될 수 있다는 것이다.

참조하기 쉽도록 본 예시적실시예에 따른 그래픽프로세서를 이하 "마리오 칩(Mario chip)"이라 칭한다.

본 예시적실시예에서 그 마리오칩은 비디오게임카트리지내에 내장된다.

본 발명에서는 마리오칩이 프로그램메모리와 주처리장치에 접속되는 한 마리오칩은 프로그램메모리와 동일한 카트리지케이스내에 내장되지 않아도 된다.

제1도에 본 발명의 예시적 실시예에 따른 예시적 비디오게임카트리지와 외부메모리시스템이 도시되어 있다.

그 비디오게임카트리지는 제1도에 도시된 모든 부품이 부착될 PCB(도시 안됨)를 포함하고 있다.

그 비디오게임카트리지는 PCB의 입력단에 설치되어 수퍼 NES메인 콘트롤데크와 신호를 주고 받기위한 커넥터전극어레이(1)를 포함하고 있다.

커넥터전극어레이(1)는 수퍼 NES메인 콘트롤데크에 설치된 상대커넥터와 접속된다.

본 예시적 실시예에 따라서, 비디오게임카트리지에 내장된 마리오칩(그래픽 코프로세서)(2)은 100~128개의 핀을 갖는 IC칩이다.

그 마리오칩(2)은 주처리시스템(즉, Super NES)으로부터 많은 제어신호, 어드레스신호 및 데이타신호를 수신한다.

예를들면, 마리오칩(2)은 주처리시스템으로부터 핀(P112)을 통해 21MHZ의 클럭입력신호와 핀(P117)을 통

해 21MHZ(또는 다른 소정의 주파수)의 시스템클럭 입력신호를 수신한다.

이 시스템클럭 입력신호는 예를들면 마리오칩에 주 CPU메모리액세스에 대한 메모리타이밍정보를 제공하는데 사용되며 그리고 마리오칩내에서의 타이밍동작에 클럭신호를 제공하는데 사용된다.

마리오칩(2)은 또한 선택적인 외부클럭입력핀(P110)을 가지고 있는데 이 핀(P110)에는 예를 들면 주시스템으로부터 수신된 21MHZ보다 더 높은 주파수클럭레이트를 갖는 신호를 발생시켜 마리오 CPU를 구동시키기 위한 외부수정진동자(4)가 연결된다.

주처리시스템(즉, 수퍼 NES CPU/영상처리장치(PPU))의 주 CPU 어드레스입력핀(HA)은 핀(P37-P62)을 통해 마리오칩(2)에 연결된다.

유사하게, 주시스템의 데이타입력핀(HD)은 주 CPU데이타버스로부터 핀(P65-P72)을 경유하여 마리오칩(2)에 연결된다.

마리오칩(2)은 부가적으로 주 CPU로부터 핀(P119)을 통해 메모리 리프레쉬신호(RESH)를, 핀(P118)을 통해 리세트신호를, 그리고 핀(P104)(P105)을 통해 리드 및 라이트 제어신호를 각각 수신한다.

마리오칩은 인터럽트 요구신호(IRQ)를 발생시켜 핀(120)을 경유하여 수퍼 NES에 그 IRQ를 제공한다.

마리오칩은 그외에 다른 제어신호도 수신하는데 예를들면 주프로그램 ROM (10)을 액세스하는 동작을 개시시키기 ROMSEL신호를 수신한다.

게다가, 게임카트리지는 입력(I)라인, 출력(O)라인 및 리세트(R)라인 상에서 수퍼 NES인증(authentication)프로세서와 데이타를 교환하는 기능을 가진 인증프로세서(3)를 포함하고 있다.

게임카트리지를 인증하는데 사용되는 인증프로세서(3)와 보안시스템은 여기서 참조하고 있는 미합중국 특허 No. 4,799,635호에 개재된 타입의 것이다.

마리오칩의 RAM어드레스핀(P74-P91)은 RAM어드레스버스(RAM A)를 통해 RAM(6)(8)에 연결되고, 마리오칩의 RAM데이타핀(P93-P100)은 RAM데이타버스(RAM D)를 통해 RAM(6)(8)에 연결된다.

이들 RAM(6)(8)은 각각 마리오칩의 핀(P90)(P91)을 통해 제공된 행어드레스 스트로브신호와 열어드레스 스트로브신호(RAS)(CAS)를 사용하여 부분적으로 제어되는 다이나믹메모리 디바이스이다.

그리고 그 다이나믹 RAM대신에 하나이상의 스태틱 RAM이 사용될 수 있으며 이때에는 핀(P90)(P91)이 행어드레스 및 열어드레스 스트로브신호없이 어드레스 신호를 그들 각각의 스태틱 RAM에 제공하는데 사용될 수 있다.

쓰기가능 제어신호(WE)는 핀(P107)을 통해 RAM(6)(8)에 제공된다.

읽기제어신호 및 쓰기제어신호(R)(W)는 주 CPU에서 발생되어 핀(P104)(P105)을 통해 마리오칩에 제공된다.

마리오칩은 이들 읽기라인 및 쓰기라인을 감시함으로써 수퍼 NES CPU가 수행하려하는 메모리액세스하는 동작의 종류를 알아낼 수 있다.

비슷하게, 주시스템의 사실상 모든 어드레스라인 및 제어라인은 주 CPU가 수행하려하는 동작의 트랙을 보유하고 있는 마리오칩에 의해 감시된다.

마리오칩이 수신한 ROM어드레스신호와 RAM어드레스신호는 감시되고 난후에 적절한 메모리디바이스에 기억되게 된다.

이때 ROM어드레스는 ROM어드레스버스와 핀(P2-P26)을 통해 프로그램 ROM(10)에 기억되고, RAM어드레스는 핀(P74-P91)을 통해 RAM(6)(8)에 기억된다.

주 CPU의 ROM데이타입력과 RAM데이타입력은 각각 ROM데이터버스 및 핀(P28-P35)과, 핀(P93-P100)를 통해 ROM(10)에 기억된다.

한가지 주지해야할 사항은 마리오칩은 지금까지 기술한 ROM과 RAM뿐만 아니라 그의 다양한 메모리다바이스와도 연결된다는 것이다.

예를든다면, 마리오칩은 CD ROM을 사용하는 비디오게임시스템에 연결된다.

예를들면, 제1도에서 ROM(10)을 사용하는 대신에 CD ROM(도시안됨)이 문자데이타, 프로그램명령, 비디오데이타, 그래픽데이타 및 음성데이타를 저장하는데 사용될 수 있다.

종래의 CD플레이어(도시안됨)는 데이타버스(P28-P35)상으로 데이타 및 명령을 액세스할 목적으로 어드레스버스(P2-P26)상으로 메모리어드레스 신호를 수신하기 위하여 마리오칩(2)에 연결되었다.

CD플레이어나 CD ROM기억시스템의 특수한 구조나 동작에 대해서는 이분야에서 통상의 지식을 가진자에게 잘 알려져 있다.

CD ROM기억장치의 한가지 이점은 정보의 바이트당 필요한 기억장치의 비용이 상당히 감소된다는 것이다.

즉, 데이타는 반도체 ROM에서보다도 100% 내지 100%정도로 저렴하게 저장될 수 있다.

그러나, 불행하게도 CD ROM의 메모리액세스시간 및 메모리리드시간이 반도체 ROM보다 훨씬 느리다.

마리오칩은 적어도 3개의 버스상에 있는 정보가 병렬로 처리하는 것을 가능케하는 3버스구조를 이용하고 있다.

즉, 제1도에 도시된 게임카트리지에서 마리오칩(2)은 ROM데이타라인, ROM어드레스라인 및 ROM제어라인을 포함하는 ROM버스와 ; RAM데이타라인, RAM 어드레스라인 및 RAM제어라인을 포함하는 RAM버스와 ; 주데이타라인, 주어드레스라인 및 주제어라인을 포함하는 주프로세서라인과 연결되어 있다.

마리오칩구조는 파이프라인동작이 처리능력을 최적화하는 것을 가능케 한다.

즉, 마리오칩은 다른 데이타를 처리하면서 ROM으로부터 임의 데이타바이트를 읽어낼 수 있음과 동시에, 3차원 관련 그래픽을 매우 효과적으로 수행하기 위해 또다른 데이타를 RAM에 기록함으로써 처리능력의 최적화를 도모하고 있다.

후술하겠지만, 마리오칩(2)은 내부적으로 16비트구조를 사용하면서도 8비트 ROM(10)칩 및 8비트 RAM(6)(8)칩과 인터페이스 할수 있도록 설계되어 있다.

그리고, 마리오칩의 모든 내부데이타버스와 내부레지스터는 16비트이다.

또한, ROM(10)으로부터 데이타를 읽어내는 동작과 RAM(6)(8)에 데이타를 기록하는 동작은 버퍼링되어 있지만 이 버퍼링으로 인해 프로그램 실행속도는 늦어지지 않는다.

비슷하게 마리오칩(2)은 CD ROM으로부터 명령 및 그래픽데이타를 액세스하여 RAM(6)(8)에 기록하는데 이는 주프로세서, 가령 수퍼 NES영상처리장치(PPU)의 비디오 RAM으로 다음 DMA를 전송하기 위함이다.

이 분야에서 통상의 지식을 가진자라면 마리오칩(2)은 RAM기억동작 및 RAM액세스동작을 바이패스시켜 처리함으로써 CD ROM으로부터 PPU의 비디오 RAM으로 데이타를 직접 전송하는 것을 조정하도록 프로그램 되어질 수 있다.

CD ROM의 액세스시간이 길다는 사실에도 불구하고 CD ROM를 그래픽에 실제 응용할 수 있는 것을 마리오칩(2)의 처리속도가 초고속이기 때문이다.

비디오데이타 및 오디오데이타는 CD ROM에 저장되기전에 종래의 데이타 압축기술로서 압축된다.

데이타압축기술과 데이타비압축(decompression)기술은 이 분야에서 통상의 기술을 가진자에게 잘 알려져 있다.

마리오칩(2)은 압축된 데이타를 CD ROM으로부터 액세스한 후에 종래 그래픽 프로세서로 달성될 수 있는 시간보다 훨씬 더 짧은 시간내에 종래 데이타비압축알고리즘로써 데이타를 비압축한다.

마리오칩(2)은 21MHZ클럭으로 작동되기 때문에 데이타를 RAM(6)(8)으로 전송하는데 요구되는 소정의 시간내에 비압축을 완료한다.

따라서, 많은 양의 비디오데이타와 오디오데이타가 일반적인 CD ROM액세스 시간내에 압축된 형태로 액세스된다.

그러나, 이러한 비교적 긴 액세스시간으로 인해 생기는 영향은 최소화될 수 있는데 이것은 데이타 바이트당 실제 액세스시간이 마리오칩(2)에 의해 데이타가 비압축된후에 상당히 감소되었기 때문이다.

마리오칩(2)이 비압축을 수행하고 있더라도 주 그래픽프로세서, 즉 수퍼 NES PPU는 다른 처리작업을 자유로이 수행할 수 있다.

물론, 속도가 문제가 되지않는 특정한 응용에서는 마리오칩(2)은 압축되지 않은 형태로 데이타를 CD ROM으로부터 액세스할 수 있다.

또한 스태틱 RAM이 사용되면 게임카트리지는 백업배터리를 포함하게 된다.

백업배터리(12)는 데이타절감특성을 제공하는데 전력손실이 생긴 경우에 스태틱 RAM용 백업전압(RS RAM) 및 스택틱 RAM칩선선택신호(RAMCS)를 제공하기 위하여 저항기(R)를 통해 종래 백업배터리회로(14)에 연결된다.

부가적으로, RAM어드레스버스에 옵션설정저항기(16)가 연결된다.

보통 동작에서, 마리오칩 어드레스라인은 RAM(6)(8)에 출력을 제공한다.

그러나, 리세트동작이나 파워-온 동작시에 그 어드레스라인은 소정의 전압치인 VCC에 연결되느냐 아니면 그라운드(0V)에 연결되느냐에 따라 하이신호 또는 로신호를 발생시키는 입력라인으로서 사용된다.

이런식으로 "1" 또는 "0"이 내부마리오칩 레지스터로 읽혀지게 된다.

리세트후에, 마리오칩은 이들 저항기의 설정에 따라 가령 승수클럭비, 즉 마리오칩이 연결되는 RAM액세스시간을 프로그램 실행동안에 결정하며, 그 클럭비는 마리오칩등의 다른 동작과 더불어 사용된다.

이들 옵션설정 레지스터를 사용함으로써 가령, 마리오칩을 어떠한 설계상의 수정없이 다수의 다른 형태의 메모리디바이스와 사용가능하다.

예를들어 다이나믹 RAM설정이 검출되면 적절한 횟수의 리프레쉬신호가 가해진다.

게다가, 옵션설정은 가령 프로세서 곱셈회로의 동작속도를 제어하는데 사용될 수 있고, 그리고 다른 명령이 어떤 특정곱셈 명령실행보다 더 빠른 레이트로 그래픽프로세서에 의해 실행되도록하는 것을 가능하게 하는데 사용될 수 있다.

따라서 지연된 곱셈실행을 개시함으로써 잔여명령이 다른 가능한 클럭 레이트보다 더 빠른 클럭레이트로 실행될 수 있다(예를 들면, 프로세서는 30MHZ로 클럭되고, 반면에 옵션설정은 곱셈명령이 15MHZ 에서 효과적으로 실행되도록 한다).

제2도는 제1도에 나타낸 전형적인 게임카트리지가 연결되도록 설계된 전형적인 주 비디오게임시스템의 블럭다이어그램이다.

예를들면 제2도는 닌텐도 오브 아메리카사가 현재 판매하고 있는 수퍼 NES를 나타낸다.

그러나 본 발명은 제2도에 도시된 것과 같은 블럭다이어그램을 갖는 수퍼 NES관련 응용품이나 시스템에 제한되지 않는다.

수퍼 NES는 예를 들면 65816호환성 마이크로프로세서인 16비트 주 CPU(22)를 수퍼 NES의 제어테크(20)내에 가지고 있다.

주 CPU(22)는 예를들면 128K바이트의 작업 RAM(W-RAM)(32)에 연결되어 있다.

또, 주 CPU(22)는 예를들면 32K워드의 비디오 RAM(V-RAM)(30)에 연결된 영상처리장치(PPU)(24)와 연결된다.

주 CPU(22)는 수직 또는 수평블랙킹구간 동안 PPU(24)를 경유하여 비디오 RAM(30)에 접근하는 통로를 가지고 있다.

따라서, 주 CPU(22)는 PPU(24)가 비디오 RAM을 액세스하고 있을때 이외의 시간대에서 PPU(24)를 통해 비디오 RAM(30)을 액세스할 수 있을 뿐이다.

PPU(24)는 비디오 RAM(30)으로부터 사용자의 텔레비젼(36)상에 비디오 화상을 발생시킨다.

또한, CPU(22)는 작업 RAM(W-RAM)(28)에 연결된 음성처리장치(APU)(26)와 연결된다.

시판되고 있는 사운드칩을 구비하고 있는 APU(26)는 게임카트리지상의 ROM(10)에 저장된 비디오게임프로그램의 실행과 관련하여 음성을 발생시킨다.

CPU(22)는 단지 APU(26)을 통해 작업 RAM(28)을 액세스할 수 있다.

PPU(24)와 APU(26)는 RF변조기(34)를 통해 사용자의 가정텔레비젼(36)에 연결되어 있다.

수퍼 NES내에 있는 비디오 RAM(30)은 카트리지내의 프로그램 ROM(10)에 기억되어 있는 문자데이타로 적재되어야만 한다(프로그램 ROM(10)은 게임 프로그램뿐만 아니라 게임이 플레이되는 동안 사용될 문자데이타도 저장하고 있다).

어떤 이동체, 예를들면 스프라이트정보, 또는 디스플레이될 배경정보는 사용전에 비디오 RAM(30)에 기억되어 있어야 한다.

프로그램 ROM(10)은 제1도의 인쇄회로기판 모서리커넥터(1)와 접속되는 제2도의 결합커넥터(18)를 통해 주어드레스버스 및 주데이타버스 상에서 CPU(22)에 의해 액세스된다.

PPU(24)는 공유주 CPU데이타버스 및 공유주 CPU어드레스버스와 커넥터(23)를 거쳐 게임카트리지와 접속되는데 그 이유는 이 게임카트리지에 제공할 PPU데이타신호 및 제어신호를 전송하기 위한 통로를 제공하기 위함이다.

APU(26)는 공유주 CPU버스 및 오디오버스(27)를 거쳐 게임카트리지와 접속된다.

CPU(22)의 어드레스공간은 프로그램 ROM(10)의 위치가 위치 0에서부터 시작 되도록 맵핑되고, 전형적으로 32K바이트 세그먼트로 나뉘어진다.

그 프로그램 ROM(10)은 대략 CPU어드레스공간의 1/2을 사용한다.

일반적으로 CPU어드레스공간의 각 32K바이트 세그먼트에서 최고위치는 작업 RAM(32)과 여러레지스터의 어드레스지정을 위해 이동된다.

프로그램 ROM(10)의 기억용량은 대개 4메가바이트이다.

수퍼 NES에서 사용되는 CPU(22)는 프로그램 ROM(10)전체의 어드레스를 지정할 수 있다.

한편, 마리오칩(2)은 단지 16비트 프로그램카운터만을 가지고 있기 때문에 프로그램롬(10)내의 32K바이트뱅크와 32K바이트뱅크간을 선택하기 위한 뱅크레지스터를 포함하고 있다.

본 예시적인 실시예에서 마리오칩은 수퍼 NES메모리맵과 대응하는 전체 24비트 어드레스공간을 가지고 있다.

이것은 위치 $00 : 8000에서 시작하는 위치에 ROM(10)을 포함하며, 카트리지상의 RAM(6)(8)은 위치 $70 : 0000에서 시작한다.

카트리지상의 ROM(10)과 RAM(6)(8)은 별도의 버스를 이용하고 있으므로 마리오칩에 의해 병렬로 액세스될 수 있다.

또한 RAM(6)(8)은 ROM보다 더 빠른 속도로 액세스될 수 있는데 마리오 칩은 이 성능상의 장점을 활용하도록 설계된다.

마리오칩(2)은 수퍼 NES내에 있는 어떠한 메모리, 즉 작업 RAM(32)이나 비디오 RAM(30)에 접근할 수 있는 어떤 통로를 가지고 있지 않다.

마리오칩(2)이 데이타를 처리하기 위해서는 또는 비트맵을 작성하기 위해서는 데이타가 RAM(6)(8)내에 저장되어 있어야만 한다.

따라서, NES CPU프로그램과 마리오칩 프로그램이 공유하고 있는 어떤 변수들이 RAM(6)(8)내에 존재하여

야 한다.

마리오칩 프로그램이 필요로하는 사전에 기억된 어떤 데이타는 ROM(10)에 존재할 수 있고, 반면 어떤 변수는 RAM(6)(8)내에 존재할 것이다.

수퍼 NES프로그램에서만 필요한 전용변수는 RAM(6)(8)내에 존재할 필요가 없다.

사실상, 이 RAM(6)(8)은 메모리공간에 의해 지나치게 비싸므로 반드시 필요한 변수만이 저장되어야 할 것이다.

반드시 필요한 변수가 아닌 변수는 수퍼 NES의 내부에 RAM(32)에 기억 되어야만 한다.

마리오칩(2)이 작성한 비트맵은 마리오카트리지의 RAM(6)(8)내에 저장되고 그리고 각 비트프레임이 전부 완성되었을때 수퍼 NES의 제어하에서 비디오 RAM(30)을 DMA전송될 것이다.

수퍼 NES의 CPU(22)는 마치 마리오칩이 존재하지 않는 것처럼 수퍼 NES의 제어데크(20)내에 있는 모든 내부 RAM에 접근하는 통로를 가지고 있다.

마리오칩은 이들 RAM에 접근하는 통로를 가지고 있지 않으므로 마리오 ROM/RAM과 수퍼 NES RAM간에 전송될 모든 데이타는 CPU(22)자체에 의해 해결, 제공되어야 한다.

데이타는 DMA전송을 통해 프로그래밍되거나 블럭이동된 CPU(22)을 경유하여 전송될 수 있다.

마리오카트리지의 ROM(10)과 RAM(6)(8)은 모든 게임프로그램상에서 평소와 같이 맵핑된다.

CPU(22)는 카트리지의 ROM(10)과 RAM(6)(8)에 일시적으로 접근할 수 있는 통로제공하는 제어장치를 가지고 있다.

파워(전원)가 상승하거나 리세트시에 마리오칩은 턴오프되고, CPU(22)는 ROM(10)과 RAM(6)(8)으로의 통로를 점유하게 된다.

마리오칩이 프로그램을 실행하기 위해서는 CPU(22)의 프로그램이 ROM(10)이나 RAM(6)(8)에의 바람직하게는 이들 모두에의 접근을 포기하여 마리오칩이 주어진 작업을 완료할때까지 기다리거나, 또는 CPU(22)가 일부코드를 작업 RAM(32)에 카피하여 이곳(작업 RAM(32))에서 실행하는 것이다.

마리오칩을 수퍼 NES의 CPU(22)가 프로그래밍하고 읽어낼 수 있는 것이 가능한 많은 레지스터를 가지고 있다.

이들 레지스터들은 위치 $00 : 3000에서 시작하는 CPU(22)의 메모리맵으로 맵핑된다.

제2도에 도시된 바와같이 수퍼 NES는 다양한 제어신호를 송수신한다.

수퍼 NES의 CPU(22)는 ROM(10)을 액세스하고자할 때 제어신호(ROMSEL)를 발생시킨다.

수퍼 NES(22)는 메모리리프레쉬를 개시하고자할때 리프레쉬신호(RFSH)를 발생시킨다.

마리오칩은 동작을 완료하면 수퍼 NES의 CPU에 연결된 인터럽트 리퀘스트 라인상으로 인터럽트신호(IRQ)를 전송한다.

부가적으로 CPU(22)는 리드신호와 라이트신호를 발생시킨다.

시스템타이밍신호는 제어테트(20)내에 있는 타이밍체인회로(21)에서 발생된다.

파워-온/리세트신호도 제어테크(20)내에서 발생되어 게임카트리지에 제공된다.

또한, 수퍼 NES는 입력(I)도체, 출력(O)도체 및 리세트(R)도체상의 데이타를 상기 미합중국 특허 No.4,799,635호에 따른 게임카트리지상의 중재프로세서(3)와 교환하는 기능을 갖는 인증프로세서(25)를 포함하고 있다.

미합중국 특허 No.4,799,635호에 게재된 인증프로세서(25)는 인증이 설정될 때까지 CPU(22)를 리세트상태로 유지시킨다.

제2도에 블럭으로 나타낸 수퍼 NES비디오게임기는 여기에서 일반적으로만 기술되어 진다.

예를들면, PPU(24)를 포함하는 수퍼 NES에 관한 상세한 설명은 1991년 4월 10일자로 "비디오처리장치"의 명칭을 가지고 출원된 미합중국 출원 No.07/651,265호에 게재되어 있으며, 참고로 여기에서 인용한다.

가령, 수퍼 NES와 게임카트리지간에 정보가 어떻게 전송되는가등과 같은 더 자세한 설명은 "영상처리시스템의 직접메모리 액세스장치 및 그 장치에 사용된 외부기억장치"의 명칭으로 1991년 8월 26일에 출원된 미합중국 특허출원 No.07/749,530호와, 발명의 명칭"모자이크 영상디스플레이장치와 이에 사용된 외부기억장치"로 1991년 11월 19일에 출원된 미합중국 출원 No.07/793,735호에 게재되어 있고, 여기에서 이들을 참조한다.

본 발명자들은 일부응용에서 실제로 가능한 것보다 더 많은 정보가 그러한 주프로세서 DMA회로를 사용하는 수직블랭킹 동안에 전송되어야할 필요가 있음을 같이 인식하였다.

따라서, 수직블랭킹시간을 확장함으로써 영상사이즈가 축소되는 결과를 가져올지라도 그 수직블랭킹시간을 확장하는 것이 바람직하다.

이로써 처리속도와 영상갱신비에 있어서 상당한 이익이 실현된다.

제3도는 제1도에 도시된 마리오칩과 다른 카트리지 구성요소를 수용하기 위한 게임카트리지케이스(19)의

하나의 실시예를 나타낸 사시도이다.

또한, 제3도는 제2도에 도시된 수퍼 NES비디오 게임하드웨어를 수용하기 위한 비디오게임제어데크(20)에 대한 예시적인 외부하우징의 사시도이다.

이러한 비디오게임 제어테크(20)와 이동가능한 게임카트리지(19)의 기계적인 설계에 대하여는 여기에서 참조될 명칭 "TV게임기"로 1991년 8월 23일자로 출원된 미합중국 출원 No.07/748,938호의 제2도 내지 제9도에 도시되어 있다.

제4a도 및 제4b도는 제1도에 도시된 마리오칩(2)의 블럭다이어그램이다.

먼저, 제4a도 및 제4b도에 도시된 여러가지 버스에 대해 설명한다.

명령버스(INSTR)는 명령코드를 어떤 마리오칩 구성요소에 제공하는 8비트버스이다.

X버스, Y버스 및 Z버스는 16비트 데이타버스이다.

HA버스는 본 실시예에서 수퍼 NES어드레스버스에 연결되는 24비트 주 시스템어드레스버스이다.

HD버스는 수퍼 NES의 데이타버스에 연결되는 8비트 주데이타버스이다.

PC버스는 마이로칩의 프로그램카운터(즉, 일반적인 레지스터블럭(76)의 레지스터(R15))의 출력을 여러시스템 구성요소에 제공하는 16비트버스이다.

ROM A버스는 20비트 ROM어드레스버스이다.

ROM D버스는 8비트 ROM데이타버스이다.

RAM A버스는 1비트 RAM어드레스버스이다.

RAM D_IN버스는 8비트 RAM리드데이타버스이고, RAM D_OUT버스는 8비트 RAM라이트데이타버스이다.

마리오칩과 수퍼 NES는 마리오칩과 수퍼 NES간에 데이타를 전송하기 위한 메인메카니즘의 역할을 하는 카트리지의 RAM(6)(8)을 공유한다.

수퍼 NES는 어드레스버스(HA)와 데이타버스(HD)를 통해 마리오칩을 액세스한다.

수퍼 NES는 수퍼 NES어드레스버스(HA)를 통해 마리오칩의 레지스터(76)를 액세스한다.

수퍼 NES는 마리오칩(2)을 통해 카트리지프로그램 ROM(10)과 RAM(6)(8)을 액세스한다.

ROM제어기(104)와 RAM제어기(88)는 각각 ROM과 RAM의 액세스를 개시하기 위하여 수퍼 NES에 의해 발생된 메모리액세스 관련신호를 수신한다.

예로써, 마리오칩(2)은 수퍼 NES가 RAM의 어드레스지정을 시도하고 있음을 알리기 위하여 RAM선택신호(RAMCS)를 사용한다.

제4a도 및 제4b도에 도시된 X, Y, Z버스는 마리오칩의 내부데이타버스이다.

X버스와 Y버스는 소스데이타버스이고, Z데이타버스는 행선지버스이다.

이들 버스들은 16비트의 병령데이타를 전송한다.

마리오칩(2)은 명령어를 실행하면서 X버스와 Y버스상에 명령실행에 사용되는 데이타의 소스를, Z버스상에 행선지데이타를 각각 싣는다.

가령, 두 레지스터에 기억되어 있는 내용을 가산하여 그결과를 어떤 제3의 레지스터에 기억시키라는 명령을 실행할때에, 산술논리연산장치(ALU)(50)는 X버스와 Y버스를 통해 두 소스레지스터의 내용을 수신하여 연산한 후, 그 연산결과를 Z버스에 싣는다(차례로 이 Z버스는 레지스터(76)에 연결되어 있다).

마이로칩(2)내의 명령디코딩회로(60)로 명령의 옵코드를 디코딩하여 얻는 제어신호는 ADD연산을 개시시킬 목적으로 ALU(50)에 제공된다.

제1도에서 기술한 바와같이 마리오칩은 병렬로 신호를 통신할 수 있는 ROM버스, RAM버스 및 수퍼 NES주버스에 연결된다.

마리오칩(2)은 주시스템이 수행하고 있는 동작을 알아내기 위하여 주수퍼 NES버스를 통해 전송된 제어신호, 어드레시신호 및 데이타신호를 감시한다.

카트리지 ROM버스와 카트리지 RAM버스는 어떤 주어진 시간에 수행되고 있는 수퍼 NES동작에 따라서 병렬로 액세스된다.

종래의 주퍼 NES게임 카트리지에서, 주 CPU어드레스라인과 데이타 라인은 RAM과 ROM에 직접 연결되기 때문에 RAM과 ROM은 병렬로 액세스되지 않는다.

본 발명의 한측면에 따라서, 마리오칩(2)은 제1도에 도시된 ROM버스와 RAM버스를 수퍼 NES버스로부터 구조적으로 분리시킨다.

마리오칩(2)은 수퍼 NES버스상으로 전송된 신호를 감시하여, 시분할되지 않은 두 별도의 ROM버스와 RAM버스를 통해 어떤 신호가 ROM칩과 RAM칩에 연결되어야 하는지를 결정한다.

ROM버스와 RAM버스를 분리시킴으로써 마리오칩(2)은 ROM를 리드하는 것과 RAM에 라이트하는 것을 동시에 할 수 있다.

이로써, 마리오칩은 RAM를 액세스하기에 앞서 ROM의 액세스가 완료될때까지 기다릴 필요없이, RAM액세스 시간보다 상당히 느린 액세스시간을 갖는 값싼 ROM칩으로도 효과적으로 작동될 수 있다.

제4a도에서, 마리오칩(2)은 상기한 바와같이 전적으로 프로그램가능한 프로세서이며, ALU(50)를 포함하고 있다.

ALU(50)는 승산기(64)에 의한 곱셈연산과 플롯하드웨어(52)에 의한 특정픽셀플로핑연산을 제외하고 마리오칩내에서 수행되는 모든 산술연산을 실행한다.

ALU(50)는 명령어 디코더로부터 적절한 제어신호를 수신하여 가산연산, 감산연산, 배타적-OR연산 및 시프트연산등을 수행한다.

제4a도에서 도시된 바와같이, ALU(50)는 X버스와 Y버스로부터 처리될 정보를 수신하고, 명령디코더(60)의 제어신호에 의해 개시되는 연산을 수행하고, 그리고 이 연산결과를 Z버스에 제공한다.

이제 ALU가 제6도에 참조하면서 보다 상세하게 기술된다.

마리오칩(2)은 3차원의 특수한 효과 및 다른 그래픽동작이 효과적으로 수행될 수 있도록 하기위한 특수목적의 하드웨어를 또한 포함하고 있어, 이들 특징을 활용하는 비디오게임이 실제적으로 실현될 수 있다.

이때, 마리오칩(2)은 실시간에서 픽셀좌표 어드레싱을 문자맵어드레싱으로 변환시키는데 도움이 되는 플롯하드웨어(52)를 포함하고 있다.

마리오칩은 디스플레이 스크린상에 있는 각 픽셀의 위치를 설정하는 X좌표와 Y좌표를 지정함으로써 편리하게 프로그램되어진다.

따라서, 그래픽동작은 프로그래머가 지정한 픽셀에 기초하여 수행되며, 플롯하드웨어회로(52)는 픽셀사양을 정확히 포맷된 문자데이타로 변환시킨다.

그리고나서, 그 문자데이타는 디스플레이되기 위해 제2도의 수퍼 NES 비디오 RAN(30)의 원하는 위치로 맵핑된다.

이때, 마리오칩프로그래머는 문자맵핑시 수퍼 NES비디오 RAM(30)을 비트맵으로 간주할뿐이다.

플롯하드웨어(52)는 디스플레이 스크린상의 X및 Y좌표와 특정픽셀의 소정의 칼라을 프로그램가능하게 선택하는 것을 가능케하며, 대응픽셀을 플롯팅하여 X 및 Y좌표가 수퍼 NES비디오 RAM(30)를 구동시키는데 사용되는 형태의 문자 정의에 대응하는 주소로 변환시키는 여러 플로팅 관련명령에 응답한다.

플롯하드웨어(52)는 카트리지 RAM에 라이트하기에 앞서 가능한한 많은 픽셀데이타를 버퍼링시킴으로써 RAM데이타의 상호작용을 최소화시키는 기능을 갖는 데이타래치와 연결되어 있다.

문자정의 데이타는 X좌표데이타와 Y좌표데이타가 변환되어 플롯하드웨어(52)에 버퍼링되고난 후에 카트리지 RAM으로 전송된다.

플롯하드웨어(52)는 PLOT X레지스터(56)와 PLOT Y레지스터(58)를 통해 X좌표데이타와 Y좌표데이타를 각각 수신한다.

본 실시예에서, PLOT X레지스터와 PLOT Y레지스터는 제4a도에 도시된 바와같이 별개의 레지스터가 아니고 마리오칩의 범용레지스터(예를들면, 제4b도의 레지스터블럭(76)내의 레지스터(R1)(R2))이다.

또, 플롯하드웨어(52)는 칼라레지스터(54)를 통해 픽셀칼라정보를 수신한다.

후술되겠지만, 디스플레이될 각 픽셀의 칼라은 8×8레지스터 매트릭스에 저장되는데, 이때 각 픽셀칼라정보는 1컬럼(column)의 매트릭스를 차지한다.

플롯하드웨어(52)는 X, Y 및 칼라입력에 관련된 문자어드레스와 문자데이타를 처리하여 문자 RAM(6)(8)에 입력시킨다.

문자어드레스는 출력라인(53)을 통해 RAM제어기(88)와 RAM어드레스버스(RAM A)에 차례로 전송된다.

문자데이타는 출력라인(55), 멀티플렉서(93) 및 RAM데이타버스 RAM D-OUT를 통해 문자 RAM에 제공된다.

플롯하드웨어(52)는 수퍼 NES문자포맷과 호환성을 유지하면서 프로그래머에게 "가상"비트맵디스플레이 시스템을 제공하기 위하여 문자내의 픽셀이 개별적으로 주소지정되는 것을 가능케한다.

그 "가상"비트맵은 카트리지 RAM에 기억되어 있으며, 각 프레임의 디스플레이를 완료하기 위해 예를들면 전술한 출원 No.07/749,530의 DMA회로를 사용하여 수퍼 NES비디오 RAM(30)에 전송된다.

플롯하드웨어(52)는 개별픽셀을 고속으로 제어하는 것이 가능함으로 회전체와 스케일링체를 포함한 특정 3차원 그래픽효과가 실제로 실현될 수 있다.

픽셀을 문자형태로 변환시키는 것 때문에, 플롯하드웨어(52)는 카트리지 RAM (6)(8)으로부터 RAM_in 데이타래치(82)와 입력라인(83)을 통해 현재의 픽셀(X) (Y)근처에 있는 다른 픽셀과 관련된 정보를 또한 수신한다.

RAM에 라이트하는 횟수는 RAM(6)(8)으로부터 검색되어 RAM데이타래치에 일시적으로 저장된 이전의 픽셀 데이타를 사용함으로써 최소화될 수 있다.

또한, 제4a도에 도시된 RAM데이타래치(80, 84, 86)는 플롯하드웨어(52)에 그러한 데이타를 제공하기 위하여 카트리지 RAM내의 많은 비트플레이(bit plane)에 저장되어 있는 픽셀에 대해 수신된 칼라데이타를

버퍼링하는데 사용된다.

RAM데이타래치(80)는 수퍼 NES가 RAM데이타래치(80)의 내용을 읽어낼 수 있도록 수퍼 NES데이타버스에 연결되어 있다.

RAM데이타래치(80, 82, 84, 86)는 RAM제어기(88)로 제어된다.

RAM데이타래치(84)(86)는 RAM(6)(8)의 데이타를 수신하여 이 데이타를 레지스터블럭(76)내의 소정의 레지스터에 로딩시키기 위하여 행선지 Z버스에 제공한다.

부가적으로, RAM어드레스를 버퍼링시키는 래치(90)가 RAM제어기(88)에 연결된다.

RAM제어기(88)는 RAM A버스를 통해 RAM(6)(8)의 주소를 지정하기 위하여 래치(90)내에 저장되어 있는 어드레스를 사용한다.

또, 수퍼 NES는 어드레스버스(HA)를 통해 RAM제어기(88)를 액세스한다.

플롯하드웨어(52)는 레지스터(R1)의 내용에 의해 설정된 수평위치에 대한 픽셀칼라정보와 레지스터(R2)의 내용에 의해 설정된 수직위치에 대한 픽셀 픽셀정보를 리드하라는 리드픽셀(READ PIXEL) 명령에 응답하여, 그 결과를 행선지 Z버스와 출력라인(87)을 통해 레지스터블럭(76)내의 소정의 레지스터에 저장시킨다.

이 플롯하드웨어(52)는 뒤에 제7도, 제8a도 및 제8b도를 참조하면서 보다 상세히 기술된다.

파이프라인 버퍼레지스터(62)와 ALU제어기 명령디코더(60)는 명령버스 (INSTR)에 연결되어 명령버스에 실려있는 명령에 응답하여 동작을 개시시키기 위한 제어신호(CTL)(이 신호는 마이로칩 전체적으로 이용됨)를 발생시킨다.

마리오칩(2)은 현재의 명령을 실행하면서 다음에 실행될 명령을 페치하는 기능을 갖는 파이프라인된 마이크로프로세서이다.

파이프라인 레지스터(62)는 가능하면 1싸이클내에 명령이 실행될 수 있도록 다음에 실행될 명령을 저장하고 있다.

명령어버스상의 명령은 레지스터, 가열 제4b도의 레지스터블럭(76)내의 레지스터(R15)에 저장되어 있는 프로그램카운터의 내용에 의해 주소가 지정된다.

마리오칩(2)에 의해 실행될 명령은 제1도의 프로그램 ROM(10) 또는 마리오칩의 내부캐시 RAM(94) 또는 카트리지 RAM(6)(8)에서 얻는다.

프로그램명령이 ROM(10)에 저장되어 있다면 ROM제어기(104)(제4b도)는 이 ROM(10)에서 명령을 페치하여 마리오칩의 명령버스(INSTR)상에 싣는다.

프로그램명령이 캐시 RAM(94)에 저장되어 있다면, 그 명령은 캐시 RAM(94)으로부터 캐시 RAM출력버스(95)를 통해 명령버스상에 직접 실리게 된다.

주 CPU, 즉 수퍼 NES는 마리오칩 프로그램명령을 위해 프로그램 ROM(10)의 일부를 할애하도록 프로그램되어진다.

수퍼 NES프로그램은 마리오칩에게 소정의 기능을 수행하도록 지시하고나서, 마리오칩 프로그램코드를 액세스하기 위해 마리오칩에 ROM(10)에 저장되어 있는 어드레스를 제공한다.

파이프라인 레지스터(62)는 룩어헤드(lookahead)관련처리를 가능케하는 프로그램 실행중에 일어날 일을 디코더가 예측할 수 있게하는 명령관련 정보를 명령디코더(60)에 제공하기 위하여 실행될 명령앞의 명령 1바이트를 페치한다.

블럭(60)내의 디코딩 및 제어회로 ALU(50), 플롯하드웨어(52), 캐시제어(68)등에서 실행중인 명령코드에 의해 지시된 동작을 수행할 것을 명령하는 제어신호를 발생시킨다.

또, 마리오칩은 ALU(50)와는 별도로 고속형 병렬승산기(64)를 포함하고 있다.

그 승산기(64)는 소정의 명령에 응답하여 X소스버스와 Y소스버스로부터 각각 수신된 2개의 8비트수를 곱셈연산하여, 그 16비트 결과를 행선지 Z버스에 싣는다.

이 곱셈연산은 가능한한 1싸이클내에 수행되게 된다.

승산기(64)에로의 숫자입력은 부호가 붙거나 붙지 않는다.

또한, 승산기(64)는 긴 곱셈연산을 수행할 수 있으므로 2개의 16비트수가 곱해져 32비트의 수를 낳는다.

그리고 승산기(64)는 곱셈연산중에 발생된 부분곱을 저장하기 위하여 연결된 부분곱레지스터(66)를 포함하고 있다.

승산기(64)는 곱셈연산코드가 디코드될대 명령디코더(60)의 제어신호에 의해 인에이블 된다.

승산기(64)는 16비트워드의 곱셈을 포함하는 긴 곱셈명령을 최소한 4클럭싸이클내에 실행한다.

긴 곱셈명령은 다음의 형태를 가지고 있다 :

R4(로워드), DREG(하이워드) = Sreg * R6.

이 명령은 소스레지스터(Sreg)와 레지스터(R6)의 내용을 곱하여 그 32비트 결과를

레지스터(R4/DREG)(로/하이)에 기억시킴으로써 실행된다.

이 곱셈은 부호가 붙게되고, 32비트 결과치에 0 및 부호플래그가 설정된다.

이 연산은 다음의 6단계로 행해진다.

단계 1 : 부호없는 곱셈 R4 [0...15] = SREG [0...7] * R6 [0...7]

단계 2 : X에 부호가 붙음.

R4 [0...15] = R4 [0...15] + 256 * SREG [9...15] * R6 [0...7].

곱셈결과의 상위 8비트는 무시되지만 가산중의 캐리는 보존된다.

단계 3 : X에 부호가 붙음.

R5 [0...15] = (Y + (R6 [8...15] * SREG [0-7]) ÷ 256 ; 부호확장.

단계 4 : X에 부호가 붙고 Y에 부호가 붙지 않음.

R4 [0...15] = R4 [0...15] + 256 * SREG [0...7] * R6 [8...15].

곱셈치의 상위 8비트는 무시되지만 가산으로부터의 캐리는 보존된다.

단계 5 : Y에 부호가 붙음.

R5 [0...15] = R5 [0...15] + (Y + SREG [0...7] * R6 [8...15]) ÷ 256 ; 부호확장

단계 6 : X와 Y에 부호가 붙음.

R5 [0...15] = R5 [0...15] + RY [8...15] * R6 [8...15].

본 실시예의 승산기(64)는 가령 카바노(Cavanaugh)가 지은 맥그로우 힐(McGraw-Hill)출판사의 1984년판, 디지탈컴퓨터 아리스메틱(Digital Computer Arithmetic)에 게재되어 있는 타입의 승산기이다.

제4b도에서 캐시제어기(68)(제14도에서 상세히 설명됨)는 프로그래머가 고속으로 실행되기를 원하는 일부 프로그램을 캐시 RAM(94)에 로딩(loding)시키는 것을 효율적으로 개시할 수 있게 한다.

그러한 "캐싱(caching)"은 대개 그래픽처리에서 자주 일어나는 작은 프로그램루프를 실행할시에 이용된다.

마리오칩 명령세트에는 "캐시(cache)"명령이 포함되어 있다.

캐시명령의 바로뒤의 명령은 캐시 RAM이 꽉 찰때까지 캐시 RAM에 로드(load)된다.

개시 명령이 실행되면 현재의 프로그램카운터상태가 캐시베이스 레지스터 (70)에 로드된다.

따라서, 캐시베이스레지스터(70)의 내용은 캐싱이 시작되는 시작위치를 나타낸다.

대부분의 명령은 1싸이클내에서 실행된다.

RAM(10)이나 RAM(6)(8)과 같은 비교적 느린 외부메모리로부터의 명령은 실행되기전에 페치된다.

이것은 6싸이클정도를 더 필요로 한다.

프로그램 실행속도를 증가시키기 위해서는 마리오칩 자체내에 있는 캐시 RAM(94)이 사용되어야 한다.

캐시 RAM(94)은 512바이트 명령캐시 RAM인데 이것은 보통 프로그램의 사이즈보다 비교적 작은 사이즈이므로, 프로그래머는 이 캐시메모리(94)를 어떻게 잘 사용할지를 결정해야만 한다.

512바이트의 캐시사이즈에 맞는 프로그램루프를 전속력을 페치하고 실행하는데 1싸이클이면 된다.

그리고, 버스가 분할되었기 때문에 ROM 및 RAM은 내부캐시 RAM(94)에 기억되어 있는 코드가 실행되는 동안도 동시에 액세스될 수 있다.

캐시 RAM(94)은 캐시 RMA(94)에 기억되어 있는 루프를 시행시킴으로써 스프라이트를 회전시키는데 편리하게 사용될 수 있다.

그 루프는 회전 및 스케일링계산을 수행하면서 ROM(10)으로부터 각 픽셀의 칼라을 읽어내는 내용의 루프이며, 대조적으로 픽셀을 RAM(6)(8)에 라이트하고자 할때는 상기 루프로써 하는게 아니라 PLOT명령을 사용하여 라이트동작을 수행한다.

즉, 상기 모든 동작은 병렬로 처리되어, 가장 느린동작으로 인해 더디어진 처리능력을 매우 빠르게 한다.

보통 상기 가장느린 동작은 ROM에 기억되어 있는 데이타를 페칭하는 동작인데, 이유는 마리오칩이 ROM과 RAM에 접근하기 위한 버퍼된 통로를 사용하도록 설계되었기 때문이다.

비교적 느린 ROM(10)에서의 프로그램실행과 비교하여 볼때, 이 프로그램은 캐시 RAM(94)에서 약 6배나 더 빠르게 실행되지만, 먼저 그 프로그램이 ROM(10)으로부터 캐시 RAM(94)으로 로드되야만 한다.

이것은, 즉 ROM(10)에서 캐시 RAM(94)으로의 로딩은 명령을 캐시될 루프의 맨앞에 배치시킴으로써 행해진다.

따라서 이 루프의 캐시명령의 어드레스로 지정된 최초 512바이트만이 개시될 것이다.

그 프로그램은 루프의 최초반복을 위한 코드를 실행하는 동안 ROM(10)으로부터 페치되어 16바이트 청크(Chunk)로 캐시 RAM(94)에 메모리될 것이다.

루프의 계속된 모든 반복은 ROM(10)대신에 캐시 RAM(94)에서 수행된다.

캐시(CACHE)명령은 어떠한 반복적인 프로그램루프의 앞부분에서 자유롭게 사용될 수 있다.

루푸의 계속된 반복은 캐시 덕택이다.

만약 프로그램루프가 512바이트보다 더 커서 캐시 RAM(94)의 용량을 초과하더라도 프로그램루프는 정상 동작될 수 있지만, 최초 512바이트만이 개시 RAM(94)에서 실행되며 나머지 바이트는 평소와 같이 ROM(10)에서 실행된다.

이것은 부분적으로 속도를 상승시키지만 이상적인 것은 아니다.

본 실시예에서 캐시 콘트롤러(68)의 구성요소인 캐시 태그비트 레지스터(72)는 캐시 RAM(94)에 로드외어 있는 메모리위치를 식별한다.

캐시 테크비트는 프로그램명령이 프로그램 ROM(10)에서 보다 고속인 캐시 RAM에서 실행가능한지를 마리오칩이 빠르게 결정할 수 있게 한다.

캐시 RAM(94)은 수퍼 NES버스(HA)와 멀티플렉서(96)를 통해 캐시제어기(68)나 수퍼 NES에 의해 액세스된다.

캐시제어기(68)는 캐시베이스 레지스터(70)를 로드하기 위해 프로그램 카운터버스(PC)에 연결되며, 캐시 메모리 어드레스가 범위를 이탈했는지의 여부를 체크하는 동작을 수행한다.

ROM(10)을 병렬로 리드(read)하는 것과 유사하게, 마리오칩은 RAM(6)(8)에 병렬로 라이트(Write)하는 한 가지 방법도 제공한다.

마리오레지스터가 RAM(6)(8)에 라이트될때는 언제나, 메모리 트랜잭션은 예를들면 RAM제어기(88)내의 별도 RAM라이트회로를 개시시킬 것이다.

이것은 전형적으로 6싸이클이 소요되지만, 프로그래머가 그때에 또다른 RAM트랜잭션을 수행하지 않는다면 처리기를 지연시키지 않을 것이다.

예를 들면 각 기억명령 사이에서 다른 처리를 실시하는 것이 더 빠르다.

이 방식으로 RAM라이트회로는 작업을 할 시간을 갖게 된다.

예를들면(아래 명령세트의 명령을 사용한다) :

```
FROM    R8      ;  R8을 (R13)에 기억시킴
SM     (R13)
SM     (R14)    ;  R0을 (R14)에 기억시킴
TO      R1
FROM    R2
ADD     R3      ;  r1 = r2 + r3를 수행
TO      R4
FROM    R4
ADD     R6      ;  r4 = r5 + r6를 수행
```

2개의 기억명령은 서로 너무 가깝게 배치되어 있음에 주의하자.

제 2기억 명령은 RAM버스가 제 1기억명령을 완료하는데 분주하기 때문에 6싸이클이상을 필요로 한다.

더 빠르게 실행될 코드를 라이트하는 더 좋은 방법은 2개의 기억명령을 다른 유용한 코드와 간격을 두어 배치하는 것이다.

예를들면 :

```
FROM    R8      ;  R8을 (R13)에 기억시킴
SM     (R13)
TO      R1
FROM    R2
ADD     R3      ;  r2 = r2 + r3을 수행
TO      R4
FROM    R5
ADD     R6      ;  r4 = r5 + r6을 수행
SM     (R14)    ;  R0을 (R14)에 기억시킴
```

이 방법으로, 제 1기억명령이 RAM에 라이트하는 것과 동시에 명령이 몇개 더 병렬로 실행될 수 있다.

후술된 명령어세트는 레지스터를 최종적으로 사용된 RAM어드레스에 다시 기록시킬때 사용하는 고속명령

을 포함하고 있다.

이 고속명령은 RAM에 기억되어 있는 값을 로딩하고 어떤 처리를 실시하고 나서 다시 고속으로 그 값을 기억시킴으로써 데이타의 벌크처리를 가능케한다.

제4b도에서 직접데이타래치(74)는 명령버스(INSTR)에 연결된다.

이 직접데이타래치(74)는 명령자체가 데이타소스를 제공하는 것을 가능케하며 따라서 어떤 소스레지스터 명령에 의해 지정될 필요가 없다.

직접 데이타래치(74)의 출력은 행선지 Z버스에 연결되어 있고, 차례로 레지스터블럭(76)의 레지스터중 소정의 레지스터에 연결된다.

명령어 디코딩회로(60)는 "직접"데이타명령을 디코딩하여 레지스터동작에 대한 적절한 전달동작을 개시시킨다.

제4b도에 도시된 게트(GET)B레지스터(98)는 전술한 지연, 버퍼된 리드 동작에 사용된다.

이때, 광범위하게 사용되는 비교적 느린 액세스타임을 갖는 ROM들을 사용하면, 대개 종래기술의 프로세서는 ROM을 실행시킬 때마다 페치가 완료 될때까지 기다려야만 한다.

후술될 지연, 버퍼된 페치메카니즘을 사용함으로써 데이타을 페치하고 있는 중에도 다른 동작이 수행될 수 있다.

이 메카니즘에 따라서, 레지스터블럭(76)내의 레지스터(R14)가 어떤 방법으로든 액세스되거나 수정되면, ROM 또는 RAM의 페치가 R14의 내용으로 식별된 어드레스에서 자동적으로 개시된다.

제4b도에 나타낸 바와같이, 레지스터(R14)는 ROM제어기(104)에 연결된다.

레지스터(R14)의 내용이 어떤 방법으로든 수정되면, ROM제어기(104)는 ROM액세스를 개시하는 작동을 한다.

ROM을 액세스한 결과는 ROM데이타버스(ROMD)에 연결된 멀티플렉서(102)를 통해 게트 B레지스터(98)로 로드된다.

아래에 식별된 명령은 게트 B레지스터(98)에서 버퍼된 정보를 액세스하는 것을 가능케 한다.

이 정보는 멀티플렉서를 통해 행선지 Z버스상에 로드되고 나서 레지스터블럭 (76)내의 하나의 레지스터로 로드된다.

이 방법으로, ROM에서 데이타를 페치하는데는 소정의 처리싸이클의 걸리더라도 페치동작을 개시될 수 있으며, 그리고 마리오칩은 다른 동작을 수행하지 않으면서 기다리는게 아니고 상기 데이타페치동작이 개시된 후에 예를들면 비관련코드를 시행할 수 있다.

또, 게트 B레지스터(98)는 제4b도의 멀테플렉서(102)를 경유하여 R(6)(8)으로부터 검색된 정보를 저장하는데 이용된다.

레지스터블럭(76)내에는 16개의 16비트 레지스터(R0-R15)가 내장되어 있다.

레지스터(R0-R13)는 이들 일부레지스터가 후술된 특수목적으로 자주 사용되더라도 특수목적레지스터가 아니고 범용레지스터이다.

상술한 바와같이, 레지스터(R14)는 메모리를 판독하기 위한 포인터로서사용되며, 그 레지스터(R14)가 변경되면 ROM(또는 RAM)에 대한 리드싸이클이 개시된다.

레지스터(R14)로 판독된(리드된)바이트는 나중에 게트 L명령이나 게트 H명령에 의해 액세스될 수 있도록 임시버퍼(게트 B레지스터(98))내에 저장된다.

레지스터(R15)는 범용레지스터이며 각 명령의 실행이 개시될때 페치될 다음명령을 가리킨다. 전형적으로 누산기의 기능을 갖는다.

또한, 그 레지스터(R0)는 대부분의 1싸이클명령을 위한 디폴트소스 및 행선지레지스터이다.

예를들어, 레지스터(R0)의 내용과 레지스터(R4)의 내용을 함께 가산하려면 레지스터(R4)을 분명하게 지정하기만 하면 된다.

레지스터(R11, R12, R13)는 루프명령실행시 특별시 사용되는 레지스터들이다.

레지스터(R13)는 루프의 맨위에서 실행될 명령의 어드레스를 저장하며, 레지스터(R12)는 루프가 반복실행될 횟수를 저장한다.

레지스터(R12)의 내용이 0이 아니면 R13의 내용에 의해 지정된 어드레스에 있는 명령은 프로그램카운터(R15)에 로드되어 실행된다.

레지스터(R11)는 루프가 완료되고 난후 복귀할 주소를 저장한다.

레지스터 제어논로회로(78)는 레지스터블럭(76)에 연결되어 범용레지스터 (R0-R15)에 대한 액세스를 제어한다.

명령어 디코드논리회로(60)는 실행될 특정명령의 형식에 따라서 하나이상의 레지스터(R0-R15)를 지정한다.

레지스터 제어논리회로(78)는 실행될 다음명령이 이용하기를 원하는 레지스터를 지정한다.

레지스터 제어논리회로(78)는 적절한 레지스터의 출력을 X버스와 Y버스에 제공한다.

제4b도에 나타낸 바와같이 적절한 레지스터(R0-R15)는 레지스터 제어논리 회로(78)의 제어하에서 Z버스로부터 정보를 수신한다.

ROM제어기(104)는 수퍼 NES어드레스버스(HA) 또는 마리오칩으로부터 어드레스를 받는 즉시 그 어드레스를 액세스한다.

ROM제어기(104)가 제13도에 상세하게 도시되어 있다.

ROM(10)으로부터 액세스된 정보를 고속명령실행을 위해 캐시 RAM(94)으로 로드된다.

ROM제어기(104)와 RAM제어기(108)는 수퍼 NES액세스시도와 마리오칩 액세스 시도간을 중재하는 버스중재유닛을 가지고 있다.

후술되겠지만, 마리오칩은 수퍼 NES CPU에 의해 액세스가능한 상태레지스터를 예를들면 레지스터블럭(76) 또는 RAM(6)(8)내에 또한 가지고 있다.

이 상태 레지스터는 0플래그, 캐리플래그, 부호플래그, 오버플로우플래그, "고(GO)"플래그와 같은 상태조건을 식별하기 위한 플래그(여기서, 1은 마리오칩이 동작하고 있음을 나타내고, 0은 마리오칩이 정지하고 있음을 나타낸다)와 ; 래지스터(R14)가 액세스되고 있음을 나타내는 ROM바이트페치인 진행플래그와 ; ALT 1플래그, ALT 2플래그, 직접바이트로 플래그 및 직접바이트 하이플래그와, 소스 및 행선지레지스터가 "워드(WITH)"가 프리픽스명령에 의해 세트되었음을 알리는 플래그와, 인터럽트플래그를 포함하는 여러가지 모드지시플래그를 저장하고 있다.

초당 여러번의 작업을 수행하기 위하여 마리오칩은 온/오프를 전환시키는 수퍼 NES는 제 4a/4b도에 블럭다이그램으로 나타낸 마리오칩을 사용하고 있다.

초기에, 수퍼 NES가 턴온되면 ROM(10)에 저장된 게임프로그램이 부팅된다.

게임프로그램이 수퍼 NES 및 마리오칩프로세서로 실행되기전에 먼저 게임 카트리지가 인증(authentication)되어야 한다는 것에 주의하자.

예로써, 상기 인증은 처음에 수퍼 NES CPU를 리세트시킨 상태에서 미합중국 특허 No. 4,799,635호에 게재된 기술에 따른 게임카트리지 및 수퍼 NES의 메인 제어테크에 연결된 인증프로세서로 인증프로그램을 실행함으로써 이루어진다.

마리오칩은 초기에 스위치오프된 상태에 있게 된다.

이때, 수퍼 NES는 게임카트리지 프로그램 ROM과 게임카트리지 RAM을 자유로이 액세스할 수 있다.

수퍼 NES는 그래픽동작이나 수학적계산을 수행하기 위해 마리오칩 처리능력을 이용할 필요가 있을때 마리오칩이 처리기를 원하는 적절한 데이타를 카트리지 RAM(또는 소정의 마리오레지스터)에 저장하고, 마리오칩 프로그램카운터에 실행될 마리오프로그램의 어드레스를 로드시킨다.

마리오칩으로 처리될 데이타는 회전되고 확대 또는 축속되어야 할 대상물의 소정의 X, Y좌표 데이타이다.

마리오칩은 다수의 스프라이트나 이동체 후면 및 전면을 조작하는 내용의 알고리즘을 수행하는 프로그램을 실행할 수 있다.

마리오칩의 속도증가 하드웨어 및 소프트웨어를 사용하면 그러한 동작을 고속으로 수행할 수 있다.

스프라이트를 처리하는데 마리오칩을 사용하면 전반적인 비디오게임시스템의 능력을 크게 증대시킬 수 있다.

예를들면, 수퍼 NES는 프레임당 128개의 스프라이트를 디스플레이하도록 제한되어 있다.

그러나, 수퍼마리오칩의 사용으로 사실상 수백개의 스프라이트가 디스플레이 되며 예를들면 회전된다.

마리오칩이 수퍼 NES가 요구하는 기능을 완료하고나면 스톱(STOP) 명령이 실행되며, 지금 마리오칩이 동작을 완료하고 다음 작업을 수행할 준비가 되었음을 지지시하는 인터럽트신호가 발생되어 수퍼 NES에 전송된다.

마리오칩은 고속곱셈과 같은 간단한 작업을 수행하는데 사용되거나 스프라이트로 가득찬 스크린을 그리는데 사용될 수 있다.

어떤 경우에서는 수퍼 NES는 마리오칩이 RAM버스 또는 ROM버스를 사용하고 있는 중에 이들 버스를 사용치 않는다면 마리오칩과 병렬로 처리동작을 자유로이 수행할 수 있다.

수퍼 NES가 게임카트리지상에서 RAM버스와 ROM버스의 제어신호를 마리오칩에 제공할지라도 수퍼 NES는 제2도의 작업 RAM(32)에서 프로그램을 실행시킬 수 있음에 주의하자.

따라서 전반적인 시스템의 처리능력은 마리오칩이 동시에 어떤 프로그램을 처리하고 있는 동안, 실행될 수퍼 NES프로그램을 프로그램 ROM으로부터 작업 ROM으로 카피함으로써 증대된다.

제5도에 마리오칩으로 하여금 원하는 어드레스에 있는 ROM의 코드를 페치, 실행시키게 하는, 주 CPU(예를 들면, 수퍼 NES CPU)로 실행될 "런 마리오(RUN MARIO)"프로그램에 의해 수행되는 동작의 시퀀스를 나타낸 플로우챠트가 도시되어 있다.

제5도에 나타낸 루틴은 프로그램 ROM(10)으로부터 제2도에 도시된 작업 RAM(32)으로 카피된 후에 수퍼 NES CPU에 의해 실행된다.

이 루틴은 마리오칩이 어떤 동작을 수행하고자 할때마다 주 CPU에 의해 실행된다.

블럭 125에 도시된 바와같이 런 마리오(RUN MARIO)주 CPU루틴이 처음 실행될 때 수퍼 NES레지스터를 보호하는등의 초기화동작이 수행된다.

초기화단계 동안에 이 런 마리오 주 CPU루틴은 프로그램 ROM(10)으로 부터 주 CPU의 작업 RAM(32)으로 카피된다.

블럭 127에 나타낸 바와같이 실행될 마리오 프로그램코드를 저장하는 ROM (10)코드뱅크가 마리오칩 레지스터로 로드된다.

게다가, 코드뱅크내의 실제주소는 블럭 129에 나타낸 바와같이 마리오칩 스크린베이스 레지스터내에 저장된다.

그후 블럭 131에 나타낸 바와같이 I/O 입력/출력모드는 4나 16 또는 256칼라 모드중 어느 칼라모드가 사용될 것인가를 식별함으로써 마리오칩에 설정된다.

이들 모드는 주 CPU가 동작하는 칼라모드와 일치한다.

부가적으로 디스플레이될 문자의 갯수에 의해 스크린의 높이를 설정하는 모드가 세트된다.

또, 마리오칩에 ROM버스 및 RAM버스의 제어를 제공하는 모드비트가 세트된다.

ROM버스 및 RAM버스의 제어는 마리오칩이 ROM버스 또는 RAM버스 또는 ROM버스와 ROM버스모두로의 통로를 가지고 있는 모드에 세트되도록 별도로 선택가능하다.

따라서, "마리오 오너(Mario owner)"모드가 ROM 및 RAM에 대해 설정된다면 주 CPU는 ROM 또는 RAM에 대해 리드/라이트할 수 없다.

마리오칩이 프로그램 ROM버스를 사용하고 있는데도 주 CPU가 프로그램 ROM을 액세스하려 한다면 마리오칩이 모조어드레스(dummy address)를 수퍼 NES에게 되돌려주는 메카니즘이 제공된다.

그러한 어드레스로 분기하는 것은 마리오칩이 더이상 카트리지 ROM버스에의 액세스를 요구하지 않을때까지 수퍼 NES를 점유된채로 유지되게 한다.

블럭 133에 나타낸 바와같이 마리오칩은 마리오칩 프로그램카운터가 마리오루틴이 실행해야만 하는 최초 명령을 저장한 어드레스로 로드된 후에야 동작을 개시한다.

그리고나서, 주 CPU는 마리오칩으로부터 인터럽트신호가 오기를 기다린다(블럭 135).

수퍼 NES는 인터럽트신호가 수신되면 현재 마리오칩이 동작을 완료하여 정지상태에 있음을 알게된다(블럭 137).

그러나 주 CPU가 그러한 어떤 인터럽트신호도 수신받지 못하면 계속 인터럽트신호가 오기를 기다린다(블럭 135).

이 시간기간동안에 수퍼 NES는 제2도의 작업 RAM(32)에서 프로그램코드를 마리오칩동작과 병렬로 실행한다.

그리고나서, 수퍼 NES는 마리오칩이 동작중에 있음을 지시하는 마리오칩의 "고(GO)"플래그가 세트되었는지 결정하기 위해 상태레지스터(가령, 마리오칩 레지스터블럭(76)내에 있음)을 체크한다(블럭 137).

부가적으로 인터럽트플래그가 마리오칩이 주 CPU가 수신한 인터럽트 신호의 소스임을 지시하기 위하여 마리오칩의 상태레지스터에 세트된다.

따라서, 주 CPU(135)가 인터럽트신호를 수신한 후에 마리오칩에 그 인터럽트의 소스인지를 결정하기 위해 적절한 마리오상태레지스터가 테스트 된다.

RAM 및 ROM에 대한 마리오 오너 모드비트는 마리오칩이 정지하면 클리어 되므로 수퍼 NES가 ROM과 RAM에 대한 액세스를 독점하게 된다.

수퍼 NES는 루틴 141를 빠져나와 런 마리오루틴에 들어가기전에 실행하고 있었던 프로그램으로 복귀한다.

CPU(22)의 게임프로그램은 마리오칩을 ROM마리오 오너모드로 설정할때 자진해서 ROM을 액세스하던 동작을 정지하여야 한다.

CPU(22)가 어떤 이유로해서 ROM을 액세스하고자 할때는 ROM마리오 오너모드를 간단히 던 오프하면 된다.

마리오칩은 다음에 ROM을 액세스할 필요가 있을때 ROM마리오 오너모드가 다시 주어질때까지 자동적으로 지속될 것이다.

마리오칩이 ROM에 대해 마리오 오너모드에 설정되어 있으면 CPU(22)게임 프로그램은 그 ROM으로부터 아무것도 리드(read)할려고 해서는 안된다.

예를 들면 수직블랭킹으로 인해 인터럽트가 발생되면, 즉 인터럽트가 NMI를 야기시켜면 CPU(22)는 자동적으로 ROM으로부터 인터럽트벡터를 페치하려할 것이다.

그러나 이것은 CPU(22)가 마리오칩에게 ROM을 액세스하지 않겠다고 확실히 지시하고서도 이를 어기고 인

터럽트를 발생시켜 ROM을 액세스하려 하기 때문에 바람직하지 않다.

이 상황에서, 마리오 오너모드가 설정되어 있음에도 불구하고 CPU(22)가 ROM을 액세스하려하는 것은 마리오칩으로 하여금 그것이 인터럽트 벡터요구라고 오해를 불러 일으킬 것이다.

마리오칩은 ROM마리오 오너 모드상태에서 인터럽트벡터를 페치하는 중에 그 인터럽트벡터를 수퍼 NES의 내부에 있는 작업 RAM(32)의 스택영역의 바닥에 재배치시킬 것이다.

예를들면, 인터럽트벡터가 $00 : FFFC라면 그 인터럽트벡터는 위치 $00 : 010C로의 점프를 일으킬 것이다.

유사하게 $00 : ffex로부터의 모든 인터럽트벡터는 CPU(22)를 $00 : 010X에서 그들의 대응하는 위치로 점프시킬 것이다.

이 기술은 CPU(22)로하여금 ROM(10)을 액세스하지 말고 대신에 수퍼 NES의 RAM(32)을 액세스하게 한다.

한가지 주지해야할 사항은 RAM기초 인터럽트벡터는 인터럽트처리기로의 점프나 분기를 포함해야만 하는데, 즉 실제코드는 간단한 벡터어드레스가 아닌곳에 있어야 한다는 것이다.

마리오칩이 ROM마리오 오너모드상태에 있지 않으면 보통의 ROM인터럽트벡터가 사용되며, 그래서 RAM기초 인터럽트벡터와 동일장소로 가게하기 위하여 이들 위치에서 지적된 동일 어드레스를 유지하는 것이 바람직하다.

이제 명령세트에 대해서 기술한다.

마리오칩명령은 고속그래픽스 및 다른 처리알고리즘을 프로그래밍하는데 효율적인 수단을 제공한다.

일부 명령들에 대한 간단한 설명이 여러명령에 의해 사용되는 일부레지스터에 대한 설명과 함께 설명된다.

또한 명령세트의 명령에 대한 상세한 리스트도 포함하고 있다.

명령들은 8비트로서 대개 1클럭싸이클내에 실행된다.

그러나, 그 명령들은 8비트 프리픽스(prefix)명령들에 의해 수정될 수 있다.

마리오칩 명령세트는 프로그래머가 어떤 명령앞에 행선지 및 두 소스레지스터를 지정하는 것을 가능케하는 특정레지스터 오우버라이드 시스템(override system)을 포함하고 있다.

그러한 "프리픽스" 오우버라이드가 없으면 명령은 누산기상에서만 동작한다.

따라서, 명령세트는 무수한 조합을 갖는 다양한 길이의 명령세트이다.

1싸이클내에서 수행되는 1바이트길이의 일부 기본명령들이 있다.

프로그래머는 프리픽스명령으로 인해 명령의 기능을 확장할 수 있다.

명령은 프로그래머의 기호에 따라 8, 16 또는 24비트가 될 수 있다.

마리오프로세서는 고속인 내장캐시 ROM에 기억되어 있는 프로그램을 개시하는데 그리고 메모리에 지연되고 버퍼링된 입출력을 개시하는데 명령을 사용한다.

그래픽처리는 상기 픽셀플롯 하드웨어를 사용하여 동작을 개시시키는 1싸이클 픽셀플롯명령을 사용함으로써 효율적으로 처리될 수 있다.

마리오명령세트를 설명하기전에 명령실행시 프로세서에 의해 세트되거나 액세스되는 여러메모리 맵핑레지스터가 아래에 기술된다.

처음에 상태플래그 레지스터가 설명된다.

상태레지스터는 16비트레지스터이며 레지스터내에 각 16비트와 관련된 플래그가 아래에 설명된다.

상태 플래그 레지스터(16비트)

비트	플래그	
0	-	-
1	z	제로플래그
2	c	캐리플래그
3	s	부호플래그
4	v	오버플로우플래그([비트 14를 15로] XOR[15를 캐리로])
5	g	고(go)플래그 : 1 마리오칩작동, 0 정지
6	r	(R14)진행중인 ROM바이트페치
7	-	-

"고(GO)"플래그(비트 5)는 마리오칩이 동작중일때는 "1"상태로, 정지중일때는 "0"상태로 세트된다.

상기 마리오칩의 정지는 수퍼 NES에 제공될 인터럽트신호를 발생시킨다.

이 상태플래그의 상태는 수퍼 NES프로세서가 체크한다.

비트 6은 ROM바이트페치가 현재 진행중임을 나타낸다.

아래에 리스트된 게트(GET)바이트 명령은 이 플래그가 클리어될때까지 실행될 수 없다.

이 클리어는 데이타의 페치가 완료되었음을 나타낸다.

상태레지스터의 이들 최하위비트(LSB)는 마리오칩 프로세서 또는 주 CPU에 의해 잔여 8비트와는 독립적으로 또는 함께 리드(read)된다.

상태플래그 레지스터의 최상위비트(MSB)는 소정의 프로픽스명령에 의해 세트되며 그리고 명령해독의 여러모드를 설정한다.

비트	모드	
8	alt 1	변경 (ADD→ADC, SUB→SBC, · · ·)
9	alt 2	변경 (ADD→ADD#, SUB→SUB#, · · ·)
10	il	직접바이트 로 (ih전에 실행)
11	ih	직접바이트 하이 (hi가 준비될때까지 로바이트가 버퍼됨)
12	b	SReg & DReg, WITH에 의해 세트됨
13	-	-
14	-	-
15	irg	인터럽트플래그

상기 ALT 1모드에서, ADD명령은 애드 위드 캐리(ADD WITH CARRY)로 해독 되며, SUBTRACT명령은 서브트랙트 위드 캐리(SUBTRACT WITH CARRY)로 해독된다.

명령 ALT 1은 이 모드에서 개시된다.

ALT 2명령은 ADD명령의 해독을 애드 위드 이미디어트 데이타(ADD WITH IMMEDIATE DATA)로, SUBTRACT명령의 해독을 서브트랙트 위드 이미디어트 데이타 (SUBTRACT WITH IMMEDIATE DATA)로 변경한다.

"이미디어트(직접)"데이타는 명령뒤를 바로 따르는 바이트를 말한다.

명령 ALT 3은 비트 8와 비트 9를 논리 "1"로 세트시킴에 주의하자.

비트 10과 비트 11은 직접데이타가 직접하이바이트냐 직접로바이트냐에 따라 세트된다.

상태레지스터의 비트 12는 "b"모드를 정의하는 것이며, 여기서 소스 레지스터와 행선지레지스터가 프리픽스명령 "WITH"를 사용함으로써 세트된다.

상태레지스터의 비트 15는 마리오칩이 동작을 정지한 후에 설정되는 마리오 인터럽트신호를 저장한다.

마리오칩은 상기 상태레지스터이외에도 다수의 레지스터를 포함하고 있다.

상기한 바와같이, 마리오칩은 제4a도 및 제4b도의 레지스터블럭(76)을 논의 할때 설명한 16비트인 16개의 레지스터를 포함하고 있다.

대부분의 이러한 레지스터들은 범용레지스터로서, 데이타 및 어드레스를 저장하는데 사용된다.

그러나 상기한 바와같이 레지스터(R15)는 항상 프로그램카운터로서 사용된다.

일반적으로 레지스터들은 2가지 목적으로 사용되는데 즉, 주 CPU와 통신하고, 실행중인 프로그램을 제어하는데 사용된다.

부가적으로 다른 레지스터들은 마리오칩내에서 사용되며 이들의 기능이 다음표에 설명되어 있다.

레지스터	특수기능
r0	디폴트 DReg 및 SReg
r1	PLOT명령의 X좌표
r2	PLOT명령의 Y좌표
r3	없다
r4	LMULT명령결과의 하위워드
r5	없다
r6	FRMULT 및 LMULT명령의 워드승수
r7	MERGE명령의 소스1
r8	MERGE명령의 소스2
r9	없다
r10	없다
r11	서브루틴 호출용 링크레지스터
r12	LOOP명령의 계수
r13	루프명령이 분기되는 어드레스
r14	ROM어드레스
r15	프로그램카운터

기타 레지스터

8비트 PCBANK	프로그램코드 뱅크레지스터
8비트 ROMBANK	프로그램데이타 ROM 뱅크레지스터(64K뱅크)
8비트 RAMBANK	프로그램데이타 ROM 뱅크레지스터(64K뱅크)
16비트 SCB	스크린베이스
8비트 NBP	비트플레인의 수
8비트 SCS	스크린 칼럼사이즈 선택 : 256,320,512,640, 1024,1280(2,4 & 8비트 플레인에서 스크린 16 & 20문자 하이)

마리오칩은 또한 칼라모드(CMODE)레지스터를 포함하고 있다.

이 레지스터의 비트들중 4비트는 후술될 특수효과를 낳는 예시적실시예에서 사용된다.

CMODE레지스터 비트를 세팅함으로써 낳은 효과는 16 또는 256칼라해상도 모드가 아래예의 설명대로 세트되었는지에 따라 변하게 된다.

CMODE레지스터들은 다음과 같다.

CMODE 비트 0

· 플롯칼라 0비트(NOT 투명비트)

· 16칼라모드에서 : 비트 0=1이고 선택된 칼라니블 = 0이면 플로팅 하지 마라.

· 256칼라모드 및 비트 3=0에서 : 비트 0=1이고 칼라바이트=0이면 플로팅하지 마라.

· 256칼라모드 및 비트3=1에서 : 비트0=1이고 칼라 하위니블=0이면 플로팅 하지 마라.

· N.B.투명성 ON=0 투명성 OFF=1

· 투명성 OFF에 대한 사용만이 한 영역을 0, 즉 스크린을 클리어하는데 사용되는 0으로 채운다.

CMODE 비트 1

· 혼합비트

· 16칼라모드에서의 혼합(상위 니블 및 하위 니블은 두가지 칼라을 제공한다)

X위치 XOR Y위치 AND 1 = 0이면 하위 니블이 선택됨,

X위치 XOR Y위치 AND 1 = 1이면 상위 니블이 선택됨.

· 투명성이 유지되어 선택된 칼라 니블이 0이면 플로팅 하지마라.

· 256칼라모드에서의 혼합은 무효이다.

CMODE 비트 2

· 상위 니블 칼라비트

· CMODE비트 3이 세트된 16칼라모드나 256칼라모드에서 : 이 비트(비트 2)가 세트되면 COLOUR명령은 칼

라레지스터의 하위니블을 소스바이트의 상위니블에 세트시킨다(이 상위니블은 또 다른 스프라이트의 하이니블로서 저장된 18칼라 스프라이트를 언팩(unpact)하는데 사용된다).

· 칼라레지스터의 하위니블이 0이면 투명성이 0일리지라도 플로팅 하지마라.

CMODE 비트 3

· 복잡한 비트

· 단지 258칼라모드에서, 이 비트 3가 세트되면 칼라의 상위니블은 함부로
  사용치 못하도록 제한되고 COLOUR명령은 하위니블을 변경시킨다.
  투명성은 하위니블만으로 계산된다.

· 보통의 256칼라모드에서 투명도는 모든 비트를 사용하여 계산된다.

16칼라코드의 예

```
ibt    r0,$C0
colour              ;   colour $C0을 세팅
ibt    r0,%0000     ;   0으로 세트
cmode
ibt    r0,$97
colour
plot                ;   colour $7을 플로팅
ibt    r0,$30
colour
plot                ;   colour가 $0이므로 플로팅하지 않음
                    ;   (투명성 은 및 하위니블=0)
ibt    r0,%C001     ;   비트 1을 세팅
cmode
ibt    r0,$40
colour

plot                ;   colour $0을 플로팅
                    ;   (투명성 오프)
stop

16칼라모드, 비트 2세트의 예

ibt    r0,$Co
Colour              ;   coulour $Co를 세팅

256칼라모드, 비트 3세트의 예

ibt    r0,$Co
colour              ;   colour $Co를 세팅
ibt    r0,%1000     ;   비트 3를 세팅
cmode
ibt    r0,$47
colour
plot                ;   colour $C7을 플로팅
ibt   r0,$50
colour
plot                ;   colour가 $Co이므로 플로팅하지 않음
                    ;   (투명성 은 및 하위니블=0)
ibt    r0,%1001     ;   비트 3 및 비트 1을 세팅
cmode
ibt    r0,$60
colour
plot                ;   colour $Co를 플로팅
                    ;   (투명성 오프)
stop
```