(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) Int. Cl.[7]
H04N 7/24

(45) 공고일자 2001년03월02일
(11) 등록번호 10-0283161
(24) 등록일자 2000년12월06일

(21) 출원번호	10-1995-0704810	(65) 공개번호	특1996-0702249
(22) 출원일자	1995년10월27일	(43) 공개일자	1996년03월28일
번역문제출일자	1995년10월27일		
(86) 국제출원번호	PCT/US 94/04576	(87) 국제공개번호	WO 94/26069
(86) 국제출원일자	1994년04월26일	(87) 국제공개일자	1994년11월10일
(81) 지정국	AP ARIPO특허 : 말라위 수단		
	EA 유라시아특허 : 벨라루스 카자흐스탄 러시아		
	EP 유럽특허 : 오스트리아 스위스 리히텐슈타인 독일 덴마크 스페인 핀랜드 영국 룩셈부르크 네덜란드 포르투칼 스웨덴		
	OA OAPI특허 : 부르키나파소 베넹 중앙아프리카 콩고 코트디브와르 카메룬 가봉 기네 말리 모리타니 니제르 세네갈 차드 토고		
	국내특허 : 오스트레일리아 바베이도스 불가리아 브라질 캐나다 체코 헝가리 일본 북한 대한민국 스리랑카 마다가스카르 몽고 노르웨이 뉴질랜드 폴란드 루마니아 슬로바키아 우크라이나 베트남 중국 라트비아 슬로베니아 트리니다드토바고 우즈베키스탄		
(30) 우선권주장	8/055,711 1993년04월27일 미국(US)		

(73) 특허권자 삼성 일렉트로닉스 코포레이션, 리미티드 김광호
대한민국 서울특별시 중구 태평로 2가 250 삼성메인 빌딩어레이 마이크로시스템스, 인코포레이티드 이.프린트 시톤
미국 80906 콜로라도 콜로라도 스프링스 퀘일레이크 루프 1420
(72) 발명자 코펫,토마스,지.
미합중국 80906 콜로라도 콜로라도 스프링스로윅 드라이브 15
루이쿠오,게리,씨.
미합중국 80919 콜로라도 콜로라도 스프링스 오우크 밸리 드라이브 6905
류,스테판,디.
미합중국 80910 콜로라도 콜로라도 스프링스 #202 랄프리지로드 1828
(74) 대리인 남상선

심사관 : 정상태

(54) 모션 평가 코프로세서

요약

비디오 데이터 시스템에서 사용하기 위한 모션 계산 코프로세서(20) 모션 계산 코프로세서는 화소의 P행×Q행 이미지를 몇 개의 페이지로 분할한 비디오 메모리(175)와 함께 사용될 수 있다 페이지 구조는 비디오 데이터를 코프로세서로 효과적으로 적재할 수 있다. 모션 계산 코프로세서는 동시에 몇 개의 블록 매칭을 수행할 수 있다. 모션 계산 코프로세서는 철저한 블록 매칭을 수행할 수 있고 계층 검색을 사용할 수 있다.

대표도

도1

명세서

[발명의 명칭]

모션 평가 코프로세서

[발명의 상세한 설명]

본 발명은 특정 목적의 이미지 압축 프로세서, 특히 모션 평가 코프로세서에 관한 것이다.

화상 이미지를 디지털적으로 나타내기 위하여, 이미지 영역은 화소의 어레이로서 설명된다. 디지털 번호는 각 화소의 칼라, 휘도 및 색차를 설명한다. 화소 칼라 정보는 실제로 3개의 값: 적색을 위한 하나의 디지털 값, 녹색을 위한 하나의 디지털 값 및 청색을 위한 하나의 디지털 값으로 구성된다. 따라서, 하나의 신호 화소를 설명하기 위하여 필요한 순수한 데이터의 양은 완전한 화상 이미지의 디지털 표현이 예외

적으로 커다란 데이터 파일로 나타난다는 것을 의미한다.

완전한 모션 비디오에 있어서, 개별적인 화상 이미지를 설명하기 위하여 커다란 데이터의 블록이 필요할 뿐만 아니라, 새로운 이미지 또는 프레임은 모션의 착시를 생성하기 위하여 초당 거의 30개의 새로운 이미지를 시청자에게 나타내 주어야만 한다. 디지털 네트워크 또는 전화 라인을 통한 이들 커다란 비디오 데이타 스트림의 이동은 주어진 사용할 수 있는 대역폭에서 간단히 수행할 수 없게 하다.

데이터 압축은 주어진 메시지를 전송하기 위하여 요구되는 비트 수를 감소시키기 위한 기술이다. 데이터 압축은 반복되는 비트 문자열을 나타내도록 단일 속기 표시를 사용하거나 전송된 메시지로부터 데이터 비트를 생략하는 것이다. 최근의 압축 형식은 "손실적인(lossy)" 압축 및 생략된 데이터를 제공하기 위하여 사람들의 정신적인 능력을 이용한다. 기존의 비디오에 있어서, JPEG 표준은 데이터 압축에 사용되었고 기존의 이미지가 압축되는 방법을 결정한다. 모션 비디오에 있어서, 많은 화상 데이터가 프레임에서 프레임으로 가더라도 일정하게 남아 있다. 따라서, 비디오 데이터는 먼저 기준 프레임을 표현하고 기준 프레임으로부터 변한 부분의 다음 프레임을 표현함으로써 압축될 수 있다.

기준 프레임은 전방 예측, 후방 예측 및 보간의 세 가지 방법에 사용될 수 있다. 전방 또는 후방 예측은 단일 기준 프레임을 사용하고 개별적으로 기준 프레임과의 차이 부분의 다음 또는 이전 프레임을 표현한다. 보간은 전방 및 후방 기준 프레임 둘 모두를 사용한다. 전방 기준 프레임은 현재 프레임보다 시간적으로 앞선 지점의 데이터 스트림에 위치한다. 후방 기준 프레임은 기준 프레임보다 시간적으로 나중인 지점의 데이터 스트림에 위치된다. 현재 프레임은 제 1 기준 프레임과 제 2 기준 프레임 사이의 평균된 차이를 근거로 계산된다.

모션 압축을 수행하기 위한 몇 가지 특정 프로토콜들이 존재한다. 이들 프로토콜 중 몇 가지는 수용된 압축 표준 결여시 칩 제조에 의하여 하드웨어로 명확하게 되고 개발된다. 그러나 최근에 모션 비디오 압축을 위하여 수용된 두 개의 표준이 나타난다. CCITT(국제 전신 전화 자문 위원회) H.261 비디오 컨퍼런스 표준은 Px64라는 알고리듬을 사용한다. P는 범위 2 내지 30의 승수로 참조되고 64는 데이터를 전송하기 위한 단일 64Kbps ISDN채널로 참조된다. 그러나, ISDN 전화 라인을 통하여 이렇게 압축된 데이터 조차도 압축하는 것은 강력한 압축이 필요하다. 다행스럽게도, 통상적인 비디오 컨퍼런스는 프레임으로부터 프레임으로 많은 모션을 갖지 않고, Px64는 단일 프레임 시간에 대하여 전방 예측만을 사용한다.

전체 모션 비디오에 대한 더 높은 품질을 나타내기 위하여, MPEG(동화상 전문가 그룹)라 불리는 제 2 표준이 전개된다. MPEG 규정은 비디오를 압축하기 위한 정확한 절차를 정의하지 못한다. 오히려, 표준이 형식 및 압축된 출력의 데이터 율을 정의한다. MPEG에 의해 사용된 압축 세트는 인트라프레임 압축, 전방, 후방, 및 보간된 모션 압축, 및 오디오를 위한 부밴드 코딩을 위한 JPEG와 같은 방법을 포함한다.

더욱 특별히, MPEG 표준에 따른 동작은 비디오 시스템이 4개의 연속적인 이미지 프레임을 표현하는 다음 가정을 참조로 요약될 수 있다. 비디오 처리 시스템은 먼저 제 1 프레임을 수신한다. 이렇게 먼저 수신된 프레임은 기준 프레임으로 설명될 수 없고 인트라프레임(즉, 예측되지 않은)코딩만이 수행된다.

그리고 제 2 프레임이 수신된다. MPEG 압축 표준의 한가지 가능한 수행은 제 1프레임의 프레임, 또는 인트라프레임("1" 프레임) 및 제 1 전방 예측된("P") 프레임으로 제 2 프레임을 표현한다. 그러나, 이러한 제 1 P 프레임은 아직 정의되지 않았고 수신된 제 2 프레임의 압축은 처리 시스템에 의하여 제 1 P 프레임이 수신될 때까지 지연된다. 또한 제 3 프레임은 제 1 I 및 P 프레임으로 표현될 것이다.

이러한 가정된 실시예의 네 번째 프레임은 제 1 P 프레임을 형성하기 위하여 사용된다. P 프레임은 기준으로써 제 1 I 프레임을 사용하는 네번째 수신된 프레임을 예측함으로써 형성된다. 제 1 P 프레임의 계산에 근거하여, 모션 평가 프로세서는 이들 프레임의 블록과 제 1 I 및 P 프레임의 블록을 비교함으로써 양방향으로 예측된 "B" 프레임으로써 제 2 및 제 3 수신된 프레임을 처리할 수 있다. 이러한 처리과정을 수행하기 위하여, 모션 평가 프로세서는 먼저 기준으로써 제 1 I 프레임을 사용하여 처리된 수신된 프레임 블록의 전방 예측을 구한다. 모션 평가 프로세서는 그 후 기준으로써 제 1 P 프레임을 사용하여 동일한 블록의 후방 예측을 구한다. 두 개의 예측은 블록을 위한 최종 예측을 형성하기 위하여 평균화된다.

현재의 모션 평가 장치에 있어서, 철저한 완전 해상도 화소 단위 서치(search)가 I 또는 P 프레임의 각 블록에 대하여 수행된다. 이러한 방법은 비디오 데이터의 전송을 위하여 큰 대역폭 버스를 요구한다. 게다가, 데이터를 통하여 생산되도록 요구되는 처리 시간은 전체 시스템 속도를 느리게 한다.

본 발명의 모션 평가 코프로세서는 종래 기술의 장치에 대하여 개선된 성능을 제공한다. 예를들면, 본 발명의 모션 평가 코프로세서는 완전 또는 반의 화소 서치 해상도에 대하여 전체적이고, 철저한 블록 매칭 서치 또는 다단계 계층적 서치를 수행할 수 있다. 계층적 서치에 있어서, 최고의 매칭 블록은 최고 해상도 이미지 보다 더 적은 데이터 점을 포함하는 저 해상도 매크로블록을 사용하여 먼저 찾아진다. 최고 매칭 블록이 찾아진 이후에, 최고 매칭 블록상에서 최고 해상도 서치가 수행될 수 있다. 이러한 일련의 단계들은 모션 평가 처리 시스템에 의해 수행되어야만 하는 전체 계산 횟수를 감소한다.

본 발명의 다른 실시예에 따르면, 모션 평가 서치는 본 발명의 모션 평가 장치에 상주해 있는 알고리듬에 의해 결정된다. 사용자는 최고 또는 계층적 서치가 수행될지의 여부를 제어할 수 있다. 부가적으로 만약 계층적 서치가 수행된다면, 사용자는 계층적 서치의 폭을 결정할 수 있다. 이러한 방식에서, 전체 서치는 처리 속도 제한 및 시스템 대역폭이 제한 요소가 되지 않으면 더 정확하다. 사용자가 요구하는 더 빠른 속도 또는 시스템 대역폭의 최대 사용에 대한 바램은 계층적 서치를 수행하도록 선택할 수 있다.

본 발명의 다른 실시예에 따라, 모션 평가 장치 구조는 새로운 비디오 데이터, 미리 검색된 데이터의 블록 매칭 및 서치 결과의 출력을 동시에 패칭할 수 있다. 이러한 구조는 이들 작업의 병렬 수행을 통하여 종래 기술의 장치에 대하여 시스템 속도 및 성능을 개선시킨다.

본 발명의 다른 실시예에 따라, 모션 평가 프로세서에 의해 사용된 메모리는 다중 서치 윈도우의 입력을 입력할 수 있도록 분할할 수 있다. 추가로, 본 발명의 비디오 입력 버퍼도 또한 분할될 수 있고 전체 또는 4분 해상 화소를 동시에 패치하는 것을 제공한다. 이들 특성들은 동일한 화소의 재패치 필요성을 피함

으로써 필요한 시스템 대역 폭을 최소화한다.

본 발명의 또다른 실시예에 따라, 본 발명의 모듈형 블록 구조는 사용될 추가 블록 매칭 엔진을 사용할 수 있다. 추가 블록 매칭 엔진은 장치의 성능을 더욱 개선한다.

제1도는 본 발명을 사용하는 모션 이미지 압축을 포함하는 비디오 시스템의 블록도 이며;

제2a도는 본 발명의 모션 평가 기술을 사용하여 감소된 이미지의 계층적 서치의 실시예이고;

제2b도는 본 발명이 모션 평가 기술을 사용한 최고 해상도 이미지상의 계층적 서치의 실시예이며;

제3도는 본 발명의 실시예에 따른 MEC 의 블록도 이고;

제4도는 본 발명이 일실시예에 따른 버스 데이지 체인 우선 계획을 도시한 다이어그램 이고;

제5도는 본 발명의 실시예에 따른 MEC 비디오 버스 데이지 체인 신호를 위한 타이밍도 이며;

제6도는 본 발명의 실시예에 따른 메모리 이미지의 논리 매핑을 도시한 블록도 이고;

제7도는 본 발명의 실시예에 따른 비디오 데이터 버스 상으로의 메모리물리적인 매핑을 도시한 블록도 이고;

제8도는 본 발명의 실시예에 따른 8 × 8 화소 논리 이미지의 간략도이며;

제9도는 본 발명의 실시예에 따른 물리적인 좌표에 매핑된 제4-6도의 8 × 8 화소 이미지의 블록도 이고;

제10도는 본 발명의 실시예에 따른 오퍼랜드 기술자 필드를 사용하는 예시적인 그리드 레이아웃이며;

제11도는 본 발명의 실시예에 따른 MEC 비디오 기록 동작의 타이밍도 이고;

제12도는 본 발명의 실시예에 따른 MEC 비디오 판독 동작의 타이밍도 이고;

제13도는 본 발명의 실시예에 따른 RAS 리프레시 사이클 이전에 CAS의 타이밍도 이고;

제14도는 본 발명의 실시예에 따른 비디오 입력 버퍼 논리 페이지 구조의 블록도이고;

제15도는 본 발명의 실시예에 따른 비디오 버퍼 메모리에서 논리를 물리적으로 매핑되는 것을 도시한 블록도 이며;

제15a도는 본 발명의 실시예에 따른 최고 및 4분 해상도를 동시에 비디오 인터페이스 버퍼에 로드하는 블록도 이고;

제16도는 본 발명의 실시예에 따른 비디오 입력 버퍼 시퀀스를 위한 상태도 이고;

제17도는 본 발명의 실시예에 따른 서치 메모리의 블록도 이고;

제18도는 본 발명의 실시예에 따른 물리적 메모리 세그먼트로 논리적 메모리 세그먼트를 매핑하는 것을 도시한 블록도 이며;

제19도는 본 발명의 실시예에 따른 서치 메모리 어드레스를 계산하기 위한 하드웨어의 블록도 이고;

제20도는 본 발명의 실시예에 따른 절대 및 상대 서치 메모리 좌표의 블록도 이고;

제21도는 본 발명의 실시예에 따른 블록 매칭 프로세서 구조의 블록도 이며;

제21a도는 본 발명의 실시예에 따른 블록 매칭 프로세서 구조의 더 상세한 블록도 이고;

제22도는 본 발명의 실시예에 따른 블록 매칭 프로세서 구성의 도시이며;

제23도는 본 발명의 블록 매칭 프로세서에 의하여 수행된 실시예 서치 패턴이며;

제24도는 본 발명의 실시예에 따른 보간기의 블록도 이고;

제25도는 본 발명의 실시예에 따른 MEC CPU의 블록도 이고;

제26도는 본 발명의 실시예에 따른 디버그 제어기 상태 블록도 이고;

제27도는 본 발명의 실시예에 따른 디버그 사이클을 위한 타이밍도 이며;

제28도는 본 발명의 실시예에 따른 CPU 명령 형식이며;

제29도는 본 발명의 실시예에 따른 MEC 호스트 인터페이스 버스상의 논버스트 판독 사이클의 타이밍도 이고;

제30도는 본 발명의 실시예에 따른 MEC 호스트 인터페이스 버스상의 논버스트 기록 사이클의 타이밍도 이며;

제31도는 본 발명의 실시예에 따른 MEC 호스트 인터페이스 버스상의 4 워드 버스트 판독 사이클을 위한 타이밍도 이고;

제32도는 본 발명의 실시예에 따른 MEC 호스트 인터페이스 버스상의 모드 1 버스트 판독 사이클을 위한 타이밍도 이며;

제33도는 본 발명이 실시에에 따른 초기 종결 모드 1 버스트 판독 사이클을 위한 타이밍도 이며;

제34도는 본 발명의 실시예에 따른 MEC 호스트 인터페이스 버스상의 4워드 버스트 기록 사이클을 위한 타이밍도 이고;

제35도는 본 발명의 실시예에 따른 MEC 호스트 인터페이스 버스상의 모드 1 버스트 기록 사이클의 타이밍도 이며;

제36도는 본 발명의 실시예에 따른 MEC 프로세서 기록 사이클의 타이밍도 이며;

제37도는 본 발명의 실시예에 따른 MEC 프로세서 판독 사이클의 타이밍도 이며;

제38도는 본 발명의 실시예에 따른 IPREDICT 메모리 인터페이스의 블록도이며;

제39도는 본 발명의 실시예에 따른 OPREDICT 메모리 인터페이스의 블록도이며;

제40도는 본 발명의 실시예에 따른 WRMEM 메모리 인터페이스의 블록도이다.

[비디오 시스템 개괄]

제 1도는 본 발명의 모션 평가 코프로세서(20)(MEC 20)을 포함하는 시스템의 한 실시예를 도시하였다. 카메라(24)는 비디오 데이터 입력을 수신하고 디스플레이 유니트(26)은 비디오 데이터를 디스플레이 한다. 카메라(24) 및 디스플레이(26) 둘 모두에 연결된 디지털 비디오 제어 유니트(27)은 버스(30)를 통하여 카메라(24), 디스플레이(26) 및 비디오 전/후 프로세서(28), 및 호스트 프로세서와의 통신을 조정한다. 디지털 비디오 제어기(27)는 또한 그래픽 제어 프로세서(32)의 제어 하에서 동작하는 그래픽 프레임 저장 메모리(31)로부터 수신된 비디오 그래픽 데이터의 디스플레이를 조정한다.

제 1도의 비디오 시스템은 또한 오디오를 포함할 수 있다. 오디오는 오디오 변환 회로(35)에 연결된 마이크로폰(33) 및 스피커(34)를 추가될 수 있다. 오디오 데이터의 압축을 위한 오디오 압축 코프로세서(36)는 주장된 제어 프로세서(37)와 오디오 변환 유니트(35) 사이에 연결된다.

제 1도의 시스템은 호스트 프로세서(18)의 제어 하에서 동작하고, 그것은 버스(30)를 통하여 나머지 시스템 부재들에 접속된다. 바람직한 실시예에 있어서, 호스트 컴퓨터(38)는 캘리포니아 산타 클라라의 인텔사에 의해 제조된 인텔 1960 패밀리 프로세서와 같은 RISC 프로세서를 포함한다. 또한 호스트 프로세서(38) 프로그램 코드를 저장하기 위한 ROM(39) 및 DRAM(40)이 버스(30)에 연결되어 있다.

수신된 비디오 데이터가 압축되는 경우, 또는 압축되지 않은 데이터가 디스플레이 되는 경우는 직렬로 작업하는 한 그룹의 장치에 의해 처리되고 비디오 압축 엔진(41)으로 알려져 있다. 비디오 압축 엔진(41)의 일부분은 비디오 프레임 저장소(43) 및 이미지 압축 코프로세서(45)이다. 비디오 프레임 저장소(43)는 비디오 압축 회로에 데이터를 입력하고 비디오 압축 회로로부터 데이터를 출력하기 위한 버퍼 메모리로써 제공된다.

압축될 수신된 비디오 데이터, 또는 디시플레이될 비압축된 데이터는 직렬로 작동하는 한 그룹의 장치에 의하여 처리되고 비디오 압축(41) 엔진(41)으로 알려져 있다. 비디오 압축 엔진(41)의 일부분은 비디오 프레임 저장(43) 및 이미지 압축 코프로세서(45)를 포함한다. 비디오 프레임 저장(43)은 비디오 압축 회로에 데이터를 입력하고 비디오 압축 회로로부터 데이터를 출력하기 위한 버퍼 메모리로써 제공된다.

이미지 압축 코프로세서(45)는 모션 평가 허프만 인코딩(Huffman encoding) 및 디코딩, 및 비트 스트림 관리를 제외한 통상적인 시스템의 모든 비디오 압축 기능을 수행한다. 이미지 압축 코프로세서는 참조로 본 출원에 포함되어 있고 같은 날짜에 출원된 출원번호 제 08/054,684호 "비디오 인터페이스 유니트"에서 설명된다. 이미지 압축 코프로세서(45)는 선택적으로 주장된 제어 프로세서(37)를 통하여 버스(30)에 연결된다. 제어 프로세서(37)는 허프만 코딩과 같이 코프로세서(45)에 의하여 성취되지 않고 실제 압축해제 및 압축 기능을 호스트 프로세서(38)로부터 오프로드한다. 선택적으로, 제어 프로세서(37)에 의해 수행되는 작업은 호스트 프로세서(38)에 의해서도 수행될 수 있다.

모션 비디오에 대하여, MEC(20)은 압축된 모션 데이터를 처리한다. MEC(20)은 그것의 지역 메모리(54)와 비디오 예측 저장 메모리(55)와 관련하여 동작한다. 본 발명의 MEC의 동작 및 구조는 아래에서 더 자세히 설명된다.

블록 매칭은 MEC(20)에 의하여 수행되는 1차 동작이다. MEC(20)의 모션 평가 회로는 현재 매크로 블록의 위치에 집중된 제한된 서치 영역내에서 현재 프레임 매크로 블록과 과거 또는 미래의 매크로 블록(기준 프레임)을 비교한다. 블록 매칭은 현재 비디오 프레임을 나타내기 위한 기준으로 사용되는 기준 매크로 블록을 식별한다.

알맞은 기준 프레임을 식별하기 위하여, MEC(20)은 예를 들면 이미지 압축 코프로세서(45)로부터 휘도 데이터의 16 화소 × 16 화소 블록을 포함하는 토큰을 수신한다. 이미지 압축 코프로세서(45)로부터 수신된 16 화소 × 16 화소 블록은 현재 프레임을 나타낸다. MEC(20)은 MEC 프로그램 메모리에 위치한 서치 알고리듬을 사용하여 하나 또는 두 개의 온칩 서치 윈도우의 내용과 상기 블록을 비교한다.

불록 매칭은 현재 프레임과 서치 윈도우 사이의 평균 절대 차이(MAD)를 계산한다. MAD를 계산하기 위하여, 비교된 두 블록으로부터 대응하는 화소들이 감산되고, 이들 화소 차이의 절대값의 합이 계산된다. MAD가 작을수록, 드 불록들은 더 잘 "매칭"된다. MEC(20) 처리 시스템은 사용자 정의 모션 평가 알고리듬을 수행함으로써 최소 MAD를 사용하여 서치 메모리의 16 × 16 블록을 찾아낸다.

MEC(20) 처리 시스템은 서치 알고리듬 동안 계산된 최소 MAD의 트랙을 유지하고, 그에 따라 서치 윈도우에서 16 × 16 블록이 입력 토큰에 가장 잘 매칭되도록 결정한다. MEC(20) 처리 시스템은 현재 프레임과 가장 잘 매칭된 블록 사이의 차이를 나타내는 모션 벡터를 계산한다. MEC(20)은 모션 벡터와 블록을 출력 토큰 형태로 이미지 압축 코프로세서(45)에 전송한다.

상기 설명된 바와 같은 블록 매칭은 최고 해상도 서치 윈도우상에서 단일 단계로 또는 감소된 이미지 서치 윈도우를 사용하여 다단계로 수행될 수 있다. MEC의 내부 서치 메모리는 1 내지 4개의 윈도우로 논리적으로 구성할 수 있다. 4개의 윈도우 구조는 MPEG 양방향성 예측의 필요성을 조정한다. 다중 해상도 서치 윈도우를 사용하는 다단계 블록 매칭 처리는 계층적 서치으로 알려져 있다.

계층적 서치는 먼저 감소된 해상도 이미지를 사용하여 최상의 매칭 블록을 찾는다. 계층적 서치의 제 1 단계에서 찾아진 최상의 매칭 위치는 통상적으로 최고 해상도 이미지에 전체 서치를 이끌어내기 위하여 사용된다. 두 단계의 계층적 처리는 개별적인 화소의 비교가 감소된 해상도 이미지를 위한 수보다 적고 모션 벡터를 계산하기에 적당한 매크로 블록이 더 빨리 조사되기 때문에 계산 단계를 감소시킨다.

본 발명에 따른 계층적인 서치를 사용한 모션 평가 코프로세싱의 실시예가 제 2A 및 2B도에 도시되어 있다. 본 실시예에서, MEC(20)은 두 단계로 ±15 화소 계층적 서치를 수행한다. 제 2A도에 도시된 제 1단계는 감소된 이미지 상에서 ±5 화소 서치를 수행하기 위하여 8 × 8 블록 매칭을 사용한다. 본 실시예에서, 32 화소 사각 구성을 갖는 4분 해상도, 또는 4:1로 감소된 이미지가 사용된다. 서치 윈도우(100)는 제로 모션 벡터 위치(120) 주위를 이동하고 MAD가 계산된다. 제 2 단계에서, 제 2B도에 도시된 바와 같이, 최고 해상도 서치 윈도우(121)에서 ±5 화소 서치는 16 × 16 블록 매칭을 사용하여 제 2A도에서 찾아진 최상의 매칭 위치 주위에서 수행된다.

철저하게 수행된다면, 계층적 서치의 각 단계는 121 블록 매칭을 필요로 한다. 반대로, 46 화소 사각형 이미지상의 철저한 ±15 화소 최고 해상도 서치는 961 블록 매칭을 필요로 한다. 추가로, 계층적 블록 매칭의 반은 16 × 16 매칭의 계산의 1/4만의 필요로 하는 8 × 8 블록상에 있다.

제 3도는 본 발명의 MEC(20) 집적 회로의 어떤 실시예의 블록도를 포함한다. 본 발명의 바람직한 실시예에 있어서, MEC(20)은 모션 평가 블록 매칭 알고리듬의 실시간 수행을 위하여 특별히 최적화된 7.2BOPS 병렬 프로세서를 포함한다.

MEC(20)상의 모든 동작은 중심 프로세서 유니트(CPU)에 의하여 조정된다. CPU(122)는 호스트 프로세서 인터페이스(124)로부터 로드된 온칩 명령 RAM(123)에 저장된 프로그램을 실행한다. 본 발명의 한 실시예에서, RAM(123)은 512 워드 RAM을 포함한다. CPU(122) 명령 세트 및 구조는 프로그램될 일반적인 다양한 블록 매칭 알고리듬을 허용한다(예를 들면, 절대적이고, 보수적인 방향, 2D 대수 등등). 스크래치 패드 메모리(125)는 선택적으로 CPU(122)와 관련된다. 스크래치 패드 메모리(125)는 미리 정해진 기능을 갖지 않고 CPU 프로그램에 의해 요구되는 바와 같이 사용될 수 있다.

서치 메모리(160)는 비디오 입력 버퍼(175)로부터 수신된 화소를 저장한다. 비디오 입력 버퍼(175)는 예를 들면, 비디오 버스 상에 위치하는 DRAM 또는 VRAM(도시되지 않음)으로부터의 비디오 인터페이스 유니트(185)으로부터 로드된다. 본 출원과 동일자로 출원되었으며 본 출원에 참조로 포함되어 있는 본 출원의 공동출원인 출원번호 제 08/054,684호 "비디오 인터페이스 유니트"은 이러한 장치를 더 자세히 설명한다. 서치 메모리(160)로 부터의 화소 출력은 블록 보간 프로세서(190)를 통하여 전송된다. 블록 보간 프로세서(190)는 MPEG 표준에서 설명된 바와 같이 수평 및/또는 수직 반 화소 보간을 선택적으로 수행한다. 블록 보간 프로세서(190)로 부터의 화소 출력은 이미지 압축 코프로세서(45)(제 1도)에 역전송하기 위하여 블록 매칭 프로세서(190) 또는 토큰 출력 버퍼(265)로 전달한다.

블록 매칭 프로세서(250)는 블록 보간 프로세서(190)로주터 수신된 블록 및 토큰 입력 버퍼(270)로부터 수신된 블록에 대하여 MAD를 계산한다. 토큰은 이미지 압축 프로세서(45)로부터 수신된 압축되지 않은 이미지이다. 블록 매칭 프로세서(250)의 출력은 이미지 압축 코프로세서(45)와 통신하기 위한 토큰 출력 버퍼(265)에 대한 기준 프레임 및 모션 벡터 출력이다.

본 발명의 MEC(20)은 또한 온 보드 테스트 제어 유니트(280)를 포함할 수 있다. 테스트 제어 유니트(280)는 MEC(20)의 적당한 동작을 증명하기 위하여 IEEE1149.1 테스트 표준을 따르는 테스트 제어 신호의 오프칩 입력을 허용하게 한다.

동작시에, 제 3도의 실시예는 보조 인터페이스(281)를 통하여 이미지 압축 코프로세서(45)로부터 데이터 토큰을 수신하고 토큰 입력 버퍼(270)에 그것을 저장한다. 데이터 토큰은 계층적 서치를 가능하게 하도록 데시메이션(decimation) 필터링 및/또는 데시메이션하지 않은 형태 둘 모두로 저장된다.

토큰 입력 버퍼(270)가 새로 수신된 데이터 토큰을 로드하는 동안, 블록 매칭 프로세서(250)는 미리 수신된 토큰의 내용과 CPU(122)의 지시하에서의 서치 메모리(160)의 내용을 매칭한다. CPU(122)는 서치 메모리(160)에 저장된 서치 윈도우 세그먼트의 어드레스를 계산하기 위하여 RAM(123)에 저장된 서치 알고리듬을 수행한다. 제 3도의 실시예에 있어서, 블록 매칭 프로세서(250)의 하드웨어는 블록 매칭 계산 작업의 대부분을 수행한다. CPU(122) 및 블록 매칭 프로세서(250) 사이의 이러한 작업 할당은 CPU 작업 로드를 감소시키고, 계산 속도를 증가시키며, 추가 블록 매칭 전력을 위하여 블록 매칭 프로세서(250)와 직렬(tandem)로 작업하도록 추가 블록 매칭 프로세서를 인에이블할 수 있다. 추가로, 작업 로드의 이러한 할당은 CPU 계산량을 감소시키고 CPU(122)가 칩 제어기로써 우선적으로 기능하도록 한다.

CPU(122)는 현재 데이터 토큰용 최상의 매칭 블록의 트랙을 유지한다. 이러한 최상의 매칭 블록은 토큰 출력 버퍼(265)에 데이터 토큰으로써 저장되고 보조 인터페이스 유니트(281)을 통하여 이미지 압축 코프로세서에 출력된다.

동시에 블록 매칭 프로세서(250)는 상기된 블록 매칭 동작을 지휘하고, 비디오 인터페이스 유니트(185)는 서치 윈도우의 새로운 부분을 사용하여 비디오 입력 버퍼(175)를 로드한다. 통상적으로, 토크에 의해 설명되는 데이터 블록은 인접한 블록이고 및 서치는 래스터 주사 방식으로 처리된다. 서치 메모리, 비디오 메모리 및 비디오 입력 버퍼(175)는 미리 로드된 서치 윈도우로부터 변경된 화소의 "스트립"만이 로드될 필요가 있도록 구성된다. 따라서, 전체 서치 윈도우는 다시 로드될 필요가 없고 동작 효율은 개선된다.

제 3도의 MEC(20)은 병렬로 3개의 동작을 수행한다는 것에 주의하라. MEC(20)은 이미지 압축 코프로세서(45)로부터 데이터 토큰을 수신하거나 또는 코프로세서(45)에 데이터 토큰를 송신하고, 블록 매칭 동작을 수행하고, 및 비디오 인터페이스 유니트(185)을 통하여 추가 서치 윈도우를 검색한다. 이들 동작의 동시 수행은 MEC(20) 동작의 효율을 최대화한다. MEC(20)의 특정 구조는 서치 윈도우 및 병렬 동작의 가능한 분할이 아래의 섹션에서 자세하게 설명하도록 구성된다.

[MEC장치 부재의 자세한 설명]

제 3도와 관련하여 설명된 MEC 부재의 구조 및 동작이 아래에서 더 자세히 설명될 것이다 독자의 이해를 돕기 위하여, 명령, 명령 레지스터 및 다음절에서 설명될 I/O 핀을 규정한 테이블이 명세서의 끝부분에 제공된다.

1. 비디오 메모리 인터페이스

비디오 메모리 인터페이스는 비디오 인터페이스 유니트(185), 비디오 입력 버퍼(175) 및 비디오 버스를 포함한다. 비디오 인터페이스 유니트(185)은 오프칩 DRAM 및/또는 VRAM 메모리로부터 비디오 입력 버퍼(175)로 휘도 이미지 화소를 판독하고 기록한다. DRAM 및 VRAM 메모리들은 비디오 버스상에 위치한 것과 같은 개념이 될 수 있다. 비디오 입력 버퍼(175)가 로드되는 경우, CPU(122)는 버퍼(175)의 내용을 적당한 명령 레지스터(LDSM)에 기록함으로써 서치 메모리(160)에 복사한다.

a. MEC 비디오 버스

테이블 1에 도시한 바와 같이 본 발명의 어떤 실시예에 있어서, MEC 비디오 버스는 전체 66핀에 대해서 11 어드레스, 32 데이터, 22 제어, 및 1 클럭 신호 핀을 포함한다. 비디오 버스는 이미지 압축 코프로세서 비디오 버스와 기능적으로 및 전기적으로 호환할 수 있다. 10 제어 레지스터의 콜렉션은 신호 핀의 타이밍 특성을 결정한다. 이들 레지스터들은 출력 신호 또는 샘플 입력 신호를 주장하거나 주장하지 않는 경우를 결정하기 위하여 비디오 인터페이스 타이밍 발생기에 의하여 사용된다. 효과적으로, 이들 레지스터들은 MEC(20) 비디오 버스에 연결된 메모리에 대응하는 DRAM 타이밍 파라메터를 포함한다.

내부적으로, 지금까지 MEC 내에서의 세 가지 동작이 비디오 버스의 사용을 위하여 갖추어진다; 메모리 리프레시, 이미지 압축 코프로세서(45)로 부터 WRMEM 명령에 의해 초기화된 바와 같은 메모리 기록 전송; 및 로드 명령키를 사용하여 비디오 입력 버퍼(175)를 채우기 위하여 CPU에 의해 초기화된 바와 같은 메모리 판독 전송, MEC의 비디오 버스의 마스터쉽은 직렬 데이지 체인 조정 방법에 의하여 제어된다. 각 포텐셜 버스 마스터는 "버스 요청 입력", "버스 요청 출력", 버스 허가 입력 및 "버스 허가 출력" 핀을 갖는다. MEC(20)상에서, 이들 핀들은 개별적으로, VRQSTI, VRQSTO, VGRNTI, 및 VGRNTO 라벨링된다. 제 4도에 도시된 바와 같이, 버스 마스터는 이들 핀들을 사용하여 우선권 순위와 연결하고; 우선권은 우측에서 좌측으로 증가된다. VRQSTI 및 VRQSTO 핀의 데이지 체인은 최저 우선권 마스터(285)로부터 최고 우선권 마스터(286)에까지 버스 사용 요구를 전파한다. 이 방법을 따라, 낮은 우선권 마스터로부터의 버스 요청은 이웃한 마스터가 동시에 버스를 필요로 하거나 또는 그의 VRQSTI 핀상에 낮은 우선권 요청을 먼저 보게 되면 더 높은 우선권을 가진 이웃한 마스터에 의하여 차단된다.

최상의 우선권 마스터는, 만약 VRQSTI 핀이 주장되고 그것이 버스 그 자체를 필요로 하지 않는다면, 그의 VGRNTO 핀을 주장함으로써 다른 마스터에 버스를 허용한다. 이러한 "버스 허용"은 VGRNTI 및 VGRNTO 핀의 체인을 따라 버스를 필요로 하는 최고 우선권 마스터로 전파된다. 더 낮은 우선권 마스터에 대한 버스 허용은 버스를 요청한 최고 우선권 마스터에 의하여 차단된다. 따라서, 버스를 요청하고 그것의 VGRNTI 핀의 주장을 관찰하는 버스 마스터는 버스를 제어할 것이고 그의 VGRNTO 핀을 주장하지 않으며, 그에 따라서 더 이상의 버스 허용 신호의 전파가 멈춰진다. CPU(122)는 MEC(20)의 VARB 레지스터를 "1"로 세팅함으로써 비디오 버스상에 높은 우선권 마스터가 되도록 MEC(20)을 규정할 수 있다. 그러한 MEC(20)의 VRQSTO 핀은 임의의 다른 마스터에 연결되지 않는다. 비슷하게, MEC(20) VGRNTI 핀의 상태는 내부적으로 무시된다.

통상적인 동작에 있어서(VARB=1), 일반적인 판독/기록 전송은 메모리 리프레시에 대하여 우선권을 얻고 외부 버스는 VRQSTI에 의한 신호와 같이 요청된다. 그러나, 버스가 WRMEM 명령 또는 비디오 입력 버퍼(175)에 허용된 이후, 그 마스터는 종료될 때까지 버스를 사용한다. 판독/기록 전송은 한번에 하나의 행을 수행하지만, 버스는 메모리 리프레시를 허용하기 위한 행과 외부 버스 마스터 사이에 두 개의 VCLK 주기동안 순간적으로 한번에 하나의 열을 얻기 위한 기회가 포기된다. 순환하는 우선권 방법을 사용하는 이들 클럭 사이클동안 외부 버스가 요청되고 메모리 리프레시가 완료된다. 처음에, 외부 요청은 최고 우선권을 할당받는다. 최고 우선권 마스터가 버스를 요청할 때마다, 그것은 버스를 허용 받고, 그의 우선권은 최저로 감소되고, 최저 우선권 마스터가 최고로 증가된다. 만약 최저 우선권 마스터가 최고 우선권 마스터가 없는 경우 버스를 요청하면, 최저 우선권 마스터는 버스를 허용 받고, 우선권은 변하지 않고 유지된다.

MEC(20)은 그 자체의 버스 동작중 하나를 대신하여 VRQSTI 핀(VARB를 0으로 세트)의 주장에 응답하는 그의 VRQSTO핀을 주장함으로써 비디오 버스를 요청할 수 있다. MEC(20)은 만약 그것이 내부 요청과 동시에 발생되지 않는 경우에만 VRQSTI의 주장에 따라 VRQSTO로 통과하기를 개시할 것이다. MEC(20)이 외주에 따라 그의 VRQSTO 핀으로 통과하기를 개시한 이후에 발생한 내부 요청은 그의 VRQSTI 핀이 다시 주장되지 않을 때까지 기다린다.

순환하는 우선권 방법은 최고 우선권을 사용하여 일반적인 판독/기록 전송을 개시하는 내부 요청과 최저인 메모리 리프레시 사이의 충돌을 해결하기 위하여 사용된다. 만약 WRMEM 명령 및 비디오 인터페이스 버퍼가 동시에 버스를 요청하면, WRMEM 요청이 우선권을 획득한다. 우선권은 상기 설명된 바와 같이 VARB=1에 대해서와 동일한 방법을 사용하여 변경된다. 그 자체에 대치될 버스를 요청하는 VARB=0인 버스 마스터는 VGRNTI 핀의 주장을 통하여 버스를 허용 받고 버스를 사용하여 종결될 때까지 그의 VRQSTO 핀을 주장하기를 계속한다.

일반적인 판독/기록 전송을 위하여, VARB=0인 MEC(20)은 행 전송 사이 버스를 포기한다. 그러나, 만약 WRMEM 명령 또는 비디오 인터페이스 버퍼(175)가 열 전송의 말단에서 버스를 사용하는 것을 종결하지 않는다면, MEC(20)은 다음 계획된 버스 요청을 위하여 (만약 인에이블된다면) 리프레시되도록 내부적으로 경쟁한다. MEC(20)은 그의 VRQSTI핀이 주장되지 않을 때마다 그것의 VGRNTO 핀을 주장하지 않고, 그에 따라서 최저 우선권 마스터는 다시 버스를 요청하기 전에 최고 우선권 마스터에 대하여 기다려야 하는 시간을 감소시킨다.

제 5도는 내부 버스 요청을 갖지 않는 데이지 체인의 중심(즉, VARB=0)의 임의의 위치에서 MEC에 대한 비디오 버스 데이지 체인 신호의 타이밍을 도시한다. 모든 신호들은 VCLK(288)과 동기된다. 제 5도에 도시

된 바와 같이, MEC(20)의 VRQST1 핀은 버스를 요청하지 않은 보다 낮은 우선권 마스터(즉, 데이지 체인의 가장 우측에 있는)의 VRQST0 핀에 의해 주장된다. MEC(20)은 VRQSTI를 하나의 레지스터에 클럭하기 위하여 하나의 VCLK 사이클을 취하고 한 클럭 사이클 이후에(제 5도의 모서리 변화 1) VRQST0를 주장함으로써 요청을 통과한다. 이러한 MEC의 좌측에 다른 최고 우선권 마스터는 그의 VGRAT0 핀을 통하여 버스를 허용하게 될 VARB=1을 가지는 마스터를 따라 버스 요구를 계속하여 통과시킨다. 제 5도의 모서리 변화 2는 그것의 VGGNT0 핀(모서리 변화 3)으로 전송하기 위하여 단지 하나의 클럭 사이클만을 취하는 VGRNTI 핀을 통하여 MEC(20)으로 통과되는 상기 버스 통과 허용을 도시한다.

얼마 후에 버스 마스터는 버스를 사용하는 것을 종결하고 그의 VRQST0 핀을 대응 적으로 주장하지 않는다. 이러한 동작은 제 5도에 설명된 MEC(20)에 대하여 VRQSTI 핀을 주장하지 않을 때 반영된다. 다시, MEC(20)은 클럭인 VRQSTI에 대하여 한 사이클이 필요하고 VRQST0를 따라 통과시키기 위하여 다른 사이클이 필요하다(모서리 변화 4). MEC(20)은 또한 VRQST0과 동시에 VGRNT0를 주장하지 않고, 더 낮은 우선권과 이웃한 시간을 감소하는 것은 VGRNTI 핀상에서 버스 해제를 관찰하기 위하여 기다려야만 한다. 이러한 VGRNT0를 미리 주장하지 않는 것은 새로운 버스 요청를 요청할 때 더 높은 우선권 마스터와 더욱 동등하도록 더 낮은 우선권 마스터를 놓는다. 제 5도의 모서리 변화에서 VRQST0가 주장되지 않은 이후의 두 사이클 동안, MEC의 VRGNTI 핀은 또한 이웃이 MEC 또는 이미지 압축 코프로세서라고 가정하면, 이웃한 더 높은 우선권의 VGRNT0 핀에 의하여 주장되지 않을 것이다.

b. 비디오 버스 상에서 데이터를 전송하기 위한 논리적 및 물리적 이미지 좌표의 사용.

본 발명의 바람직한 실시예에 따라서, MEC(20)가 버스에 액세스하면, 4096 × 4096 화소까지 각각 구성된 8개 가지의 분리된 휘도 이미지는 16 또는 32 비트 페이지 모드 데이터 전송을 사용하여 비디오 메모리 인터페이스 유니트(185)에 의하여 액세스될 수 있다. MEC 비디오 메모리 버스의 데이터 폭은 CPU의 메모리 맵에서 VBUSMODE 레지스터에 의해 결정된 바와 같이 32 또는 16 비트로 구성될 수 있다. 만약 VBUSMODE가 "0"이라면, 버스 32 비트 폭으로 구성된다. 만약 VBUSMODE가 "1"이라면, 버스는 16 비트 폭으로 구성되고, 화소는 데이터 버스 0과1에 액세스된다. 논리적인 어드레스에서 물리적인 어드레스로의 변환은 32비트 및 16비트 버스 모드에 대하여 아래에서 개별적으로 설명된다.

32 비트 액세스 모드에 있어서, MEC의 4개의 8비트 비디오 메모리 데이터 버스는 매 메모리 사이클마다 동시에 네 개의 화소를 판독하거나 기록하는데 사용되고 4개의 데이터 버스에 연결된 메모리에 물리적으로 확산될 하나의 논리적 이미지에 대한 저장소를 요구한다. 이들 네 개의 화소는 논리적 이미지 공간에서 2 ×2 화소 사각형을 형성한다. 따라서 32 비트 액세스 모드는 논리적 이미지를 오버랩 되지 않은 2 × 2 화소형태의 타일로 분할하고, 각 (행, 열) MEC에 의한 메모리 어드레스 상의 출력은 이들 타일 중 하나를 선택한다.

32비트 모드에 있어서, 논리 좌표(제 1열, 제 1행)에서의 화소의 블록은 2 × 2 타일의 어레이로써 물리적으로 액세스된다. 실시예와 같이, 제 6도는 8 × 8 화소 논리적 이미지를 도시한다. 화소들은 개별적으로 행:열 위치로 표시되고; 타일 좌표 축(289, 290)은 제 6도의 상부 및 좌측을 따라 도시된다. 32 비트 액세스 모드는 이러한 64 화소 이미지를 타일의 4 × 4 그리드로 분할한다.

제 7도는 제 6도의 화소가 4개의 MEC 비디오 데이터 버스(290-293)상의 메모리에 물리적으로 매핑되는 방법을 도시한다. 제 6도에서의 짝수 번호의 열로부터의 화소는 데이터 버스 0 또는 1 상에 액세스되고 홀수 열로부터의 화소는 항상 데이터 버스(2 또는 3)에 액세스된다. 제 6도의 짝수행으로부터의 화소는 데이터 버스(1 또는 2)에 액세스되고, 반면에 홀수 행으로부터의 화소는 데이터 버스(1 또는 3)상에 액세스된다. MEC(20)은 비디오 어드레스 버스상 타일의 행 및 열 좌표의 오프 세트된 변형을 출력함으로써 타일을 만드는 4개의 화소에 동시에 액세스할 수 있다.

따라서 MEC(20)은 행 어드레스 first_tile_row 및 열 어드레스 firs_tile_col에서 시작하는 타일의 num_tile_rows ×num_tile_cols 어레이에 액세스한다. 타일은 좌에서 우로, 위에서 아래로 래스터 주사 형식으로 액세스된다. 행 어드레스들은 다음 어드레스로 타일의 한 행으로부터 일씩 증가되고; 열 어드레스는 행내에서 하나씩 증가한다. 각 행내의 타일은 MEC이 연속적인 타일 열 어드레스를 따르는 비디오 어드레스 버스상의 타일 행 번호를 출력하는 페이지 액세스 모드의 시리즈를 사용하여 액세스된다.

16 비트 액세스 모드에서, 비디오 메모리 데이터 버스(0 및 1)는 매메모리 사이클마다 두 개의 8비트 화소를 동시에 판독 또는 기록하기 위하여 사용되고, 두 개의 데이터 버스를 통하여 물리적으로 확산될 하나의 논리적 이미지의 저장을 필요로 한다. 이들 두 화소는 논리적 이미지 공간에서 두 개의 행 배열에 의해 하나의 화소를 형성한다. 결과적으로 16 비트 액세스 모드는 두 개의 행 타일에 의하여 논리적 이미지를 오버랩되지 않은 하나의 화소로 분할하고, MEC에 의한 각 (행, 열) 메모리 어드레스 쌍 출력은 이들 타일 중 하나를 선택한다.

제 8도는 8 × 8 화소 논리적 이미지(295)를 도시한다. 화소들은 그들의 행:열 위치를 사용하여 개별적으로 표시되고; 타일 좌표축은 도면의 상부 및 좌측을 따라 도시된다. 16 비트 액세스 모드는 이러한 64 화소 이미지를 4 행 × 8 열 타일 그리드로 분할한다.

제 9도는 제 8도의 화소가 제 9도의 참고 번호(300 및 301)에 의하여 식별된 비디오 데이터 버스(0 또는 1)상의 메모리에 물리적으로 매핑되는 방법을 도시한다. MEC(20)은 비디오 어드레스 버스상의 타일의 행 및 열의 오프 세트된 변형을 출력함으로써 타일을 만드는 두 개의 화소에 동시에 액세스한다.

제 9도에 도시한 바와 같이, 16비트 액세스 모드는 이미지 행 방식을 분할한다. 두 개의 비디오 버스상의 메모리들 사이 즉, 비디오 데이터 버스(0)상의 메모리는 짝수 논리 행으로부터의 모든 화소를 저장하고 다른 메모리는 홀수 행으로부터의 모든 화소를 저장한다. 따라서, 32비트 모드에서와 같이, MEC(20)은 행 어드레스 first_tile_row 및 열 어드레스 firs_tile_col에서 시작하는 파일의 num_tile_rows × num_tile_cols 어레이에 액세스한다. 타일들은 페이지 모드 메모리 액세스를 사용하여 32 비트 모드에서와 같이 동일한 형식으로 액세스된다.

c. 비디오 데이터를 비디오 메모리에 기록.

화소들은 보조 버스 인터페이스를 통하여 이미지 압축 코프로세서(45)로부터 수신된명령 (WRMEM)에 응답하는 비디오 인터페이스 유니트(185)에 의해 오프 칩 DRAM 및/또는 VRAM에 기록된다. MEC(20)은 오퍼랜드에 hpos 및 vpos 기술자 필드에 의하여 인덱스된 논리적 좌표 그리드를 사용하여 어드레스된 비디오 메모리에 명령의 4개의 블록 오퍼랜드 데이터 토큰에 포함된 휘도 데이터를 기록함으로써 WRMEM 명령을 수행한다. 한 실시예에서, 이러한 논리적인 좌표 그리드의 각각의 수평 및 수직 축의 해상도는 16 화소이고, 각각의 16×16 화소 그리드 엘리먼트는 2 × 2 블록 구조(즉 하나의 휘도 토큰에 대응하는)에 배열된 4개의 8 × 8 화소 블록에 대응한다. 제 10도는 hpos=4 및 vpos=2에 대응하는 그리드 엘리먼트(309)를 강조한 예시적인 그리드 레이아웃을 도시한다.

WRMEM 명령은 논리적인 좌표를 물리적인 메모리 어드레스로 변형하기 위하여 내부 레지스터 및 다양한 토큰 기술자 필드와 관련하여 사용된 3개의 파라메터를 가진다. 이들 파라메터들은 memsel, horgsel, 및 vorgsel이고, 이미지 압축 코프로세서(45)의 WRMEM 명령을 실행하기 위한 MEC(20)으로 전송할 때 프로세서 패킷 워드의 부분으로써 전송된다.

WRMEM 명령의 10:8 비트를 점유하는 memsel 파라메터는 분리된 이미지 메모리를 선택하기 위한 수단을 제공하고 0 내지 7 범위에 든다. 7:4 및 3:0 명령 비트를 점유하는 horgsel 및 vorgsel 파라메터는 메모리 memsel내에서 부-이미지를 선택하는데 사용된다. 부-이미지의 상부 좌측 모서리의 물리적인 (X, Y) 좌표는 X=128 * horgsel, 및 Y=128 * vorgsel에 의해 주어진다. 제 11도는 비디오 버스 기록 동작을 위한 MEC(20) 타이밍도를 도시한다. 저장된 비디오 데이터에 액세스하기 위하여, MEC(20)은 각행 액세스의 시작시 VA(10:0) 핀 상에 단일 행 어드레스를 출력하고, 동일한 핀 상에 열 어드레스의 벡터를 따르며; 행에서 32 비트 또는 16 비트 워드 각각을 위한 하나의 열 어드레스를 출력한다. 열 어드레스는 VRAS-(3:0) VCASO-(3:0) 핀을 통하여 열 어드레스로부터 식별된다.

d. 비디오 메모리로부터 데이터를 판독.

화소들은 2:1 수평 또는 수직 데시메이션을 사용하여 또는 사용하지 않는 비디오 인터페이스 유니트(185)에 의해 판독된다. 데시메이션 특성은 최고 해상도 이미지의 서치가 제 2a 및 제 2b도와 관련하여 논의된 바와 같이 공간적으로 동시 발생하는 사분 해상도 이미지의 서치에 의하여 선행되는 계층적 모션 평가에 사용된다. 본 발명의 바람직한 실시예에 있어서, 데시메이션 필터는 오버랩 되지 않은 2 × 2 화소 블록 상에서 동작하는 간단한 평균기이다. 만약 2 × 2 블록으로 구성된 네 개의 8 비트 무부호 화소 값이 a, b, c, 및 d로 주어지면, 필터 y의 출력은 다음과 같이 주어진다:

y=(a+b+c+d)//4

여기서 "//"는 반올림을 사용하여 가장 가까운 정수로 반올림되는 정수 나눗셈을 의미한다.

MEC(20)의 비디오 인터페이스 유니트(185)은 또한 서치 메모리를 위하여 새로운 이미지 데이터를 패칭할 수 있는 반면 그것의 현재 내용은 블록 비교를 수행하기 위하여 사용된다. 기본적으로, 이러한 특성은 인접한 매크로 블록을 둘러싼 서치 영역의 많은 수의 잔여 화소의 불필요한 패치를 피하고 이미지 압축 코프로세서(45)로부터 수신된 새로운 입력 토큰에 응답하는 서치 윈도우를 완벽하게 재로드하는 성능 패널티를 피하며, 서치 윈도우가 이미지에 대하여 효과적으로 "슬라이딩"되는 것을 허용한다.

판독 사이클 동안 중요한 DRAM 타이밍 파라메터와 만나도록 돕기 위하여, MEC은 CAS가 "하이"로 리턴된 이후에 수신된 데이터를 고정하도록 내부적인 투명한 래치를 갖는다. 이것은 MEC이 다음 사이클에서 데이터를 래치하도록 허용하고, 그에 따라 메모리 액세스 시간 요청이 감소된다. 이러한 래치는 일반적으로 DRAM CAS 핀에 연결된 입력 신호 VCASI-(3:0)에 의해 클럭된다. 제 12도는 비디오 버스 판독 동작을 위한 MEC(20) 타이밍도를 도시한다.

e. 비디오 메모리 리프레시

비디오 인터페이스 유니트(185)은 DRAM 및 VRAM 메모리를 리프레시하는데 사용될 수 있다. 내장(built-in)RAS-전-CAS 리프레시 제어기(도시되지 않음)는 비디오 버스상의 DRAM 바탕 메모리를 리프레시하기 위하여 선택적으로 사용될 수 있다. CPU(122)는 비디오 버스 리프레시 인에이블(VREN) 레지스터에 기록함으로써 제어기를 동작한다. "1"을 이 레지스터에 기록하는 것은 리프레시 제어기가 그것의 내부 16 비트 리프레시 카운터를 클리어하고 비디오 버스상의 하나의 RAS-전-CAS 신호를 수행하도록 한다. DRAM이 테스트 모드로 가도록 하는 리프레시 사이클을 방지하기 위하여, MEC(20) 리프레시 제어기는 주장될 수 있는 기록을 가진 모든 RAS-전-CAS 리프레시 사이클을 수행한다. VREN이 1인 동안, 리프레시 제어기는 VCLK의 각 상승 모서리 상 리프레시 카운터를 증가시키고 카운터가 다음 레지스터의 내용과 매칭될 때마다 비디오 버스 상의 다른 리프레시 사이클을 수행한다:

VRFPMS(7:0) - 비디오 버스 리프레시 주기, 최상부 반(Half)

VRFPLS(7:0) - 비디오 버스 리프레시 주기, 최하위 반

VRFPMS 및 VRFPLS는 CPU(122)에 의하여 초기화된다.

16비트 리프레시 카운터가 연속적인 VRFPMS 및 VRFPLS와 매칭될 때마다, 카운터는 자동적으로 리셋되고 다음 리프레시 사이클까지 카운팅하기 시작한다. 카운터는 리프레시 제어기가 비디오 버스상에서 리프레시 사이클을 수행하는 동안에까지도 계속해서 증가한다. 비디오 버스가 사용중일 동안 만약 리프레시 카운터가 리프레시를 나타내면, 제어기는 현재 버스 사용을 중단하는 것이 아니라, 버스를 사용할 수 있을 때까지 기다린다. 리프레시 카운터는 리프레시 제어기가 버스를 기다리는 동안 계속해서 증가한다.

통상적으로, 리프레시 주기는 버스상의 모든 DRAM에 대하여 계산된 바와 같이 (Tref/nrows)/Tvclk의 최소값으로 세트되고, 여기서: Tvclk는 VCLK의 주기이고; Tref는 비디오 버스상에 주어진 DRAM 또는 VRAM을 위한 리프레시 시간이고, nrows는 주어진 DRAM 또는 VRAM에서 행의 수이다. 이러한 형식은 사용중인 버스가 전적으로 중요하지 않기 때문에 많은 시간동안 리프레시 제어기가 리프레시를 수행하는 것으로부터 지연된다고 가정한다.

MEC 리셋동안 DRAM 및 VRAM 내용을 유지하는 것을 돕기 위하여, VREN 및 내부 리프레시 카운터 레지스터는 하드 리셋 또는 재부팅에 응답하여 클리어된다. MEC 소프트 리셋(호스트 프로세서(38)에 의해 야기되는)은 VREN, VRFPMS, VRFPLS, 또는 내부 리프레시 카운터의 상태에 영향을 주지 않는다.

MEC의 리프레시 제어기는 또한 대부분의 DRAMS 및 VRAMS의 특별한 요청을 수용하도록 설계된다. 다음과 같이 100 내지 200 u.s.의 초기 전력 공급 휴지 또는 그것을 리프레시하기 위한 임의의 시간 간격이 초과되면, DRAM 또는 VRAM은 통상적으로 장치가 적당하게 동작하기 전에 8개의 RAS 사이클 "웨이크-업"을 필요로 한다. MEC(20) 초기화의 부분으로서, CPU(122)는 VREN에 기록하는 것이 RAS-전-CAS 리프레시 사이클을 생성하기 때문에 하나의 행에서 VREN을 8번 "1"로 기록함으로써 이러한 웨이크-업을 성취한다(리프레시 사이클을 수행하기 위하여 요구된 시간에 의해 인접 기록을 분리). 택일적으로, VRFPMS 및 VRFPLS를 로드하고 VREN을 "1"로 세팅한 이후에, CPU(122)는 MEC(20) 프로그램 수행이 시작되기 전에 8개의 리프레시 사이클을 기다릴 수 있다.

제 13도는 MEC RAS-전-CAS 리프레시 사이클의 타이밍도를 도시한다. 모든 리프레시 사이클동안, 비디오 인터페이스 유니트(185)은 리프레시가 발생하였다고 신호하도록 모든 VMSEL(2:0)핀을 "로우"로 세트한다. 따라서, 만약 VREN=1이면, VMSEL(2:0)=0은 신호 리프레시 사이클로 예약되기 때문에, VMSEL(2:0) 핀 중 7개의 인코딩만이 일반적인 판독/기록 메모리 사이클동안 특정 메모리 뱅크를 선택하기 위하여 사용될 수 있다. 모든 VCASO-(3:0) 및 VRAS-(3:0) 핀은 VD3(7:0)를 통하여 VDO(7:0)상의 모든 메모리가 리프레시에 의해 영향을 받도록 제 13도에 도시한 바와 같이 사이클된다.

f. 비디오 입력버퍼 구조 및 동작 비디오 입력 버퍼(175)는 비디오 인터페이스 유니트(185)에 의해 패치된 최고 및/또는 4분 해상도 이미지 데이터를 저장한다. 비디오 입력 버퍼(175)는 하나 또는 넷 또는 그 이상의 분리된 버퍼 세그먼트로서 기능한다. 최고 및 4분 해상도 화소 둘 모두는 비디오 인터페이스 버퍼(175)의 다른 세그먼트에 저장된다. 이러한 로드 구조는 계층적 서치를 하는 동안 화소의 잔여 패치를 피한다. 비디오 입력 버퍼(175)의 화소는 CPU(122)의 지시하에서 서치 메모리의 선택된 세그먼트에 연속적으로 전송된다.

본 발명이 한 실시예에 있어서, 비디오 입력 버퍼(175)는 2k 바이트 메모리를 포함하고, 128 페이지로 분할되며 4개의 세그먼트로 나누어진다. 물리적으로, 비디오 입력 버퍼 저장은 짝수 뱅크와 홀수 뱅크로 구성된다. 짝수 뱅크는 외부 비디오 메모리로부터 짝수 차수의 행을 저장한다. 홀수 뱅크는 외부 비디오 메모리로부터 홀수 차수의 항을 저장한다. 본 발명의 실시예에서, 각 뱅크는 8개 화소 행을 형성하는 8개의 128 × 8 VIB_RAMs으로 구성된다. 제 14도의 실시예에 도시한 바와 같이, 각 페이지(399)는 2개의 행×8개의 열 화소 타일에 해당한다. 차례로 각 비디오 메모리 세그먼트는 페이지의 시각형 배열을 포함한다. CPU(122)는 테이블 2에 목록으로 잡힌 레지스터를 초기화함으로써 비디오 입력 세그먼트 k(k=0, 1, 2, 3에 대하여)를 형성한다.

CPU(122)는 로드 명령 레지스터(LDVIB(7:0))에 자동적으로 기록함으로써 하나 또는 그 이상의 비디오 입력 버퍼(175) 세그먼트를 로드하도록 비디오 인터페이스 유니트(185)에 명령할 수 있다. 로드 명령 레지스터에 기록한 CPU(122)에 응답하여 비디오 인터페이스 유니트(185)은 먼저 비디오 입력 버퍼(175)의 내용을 종결하기 위한 서치 메모리(160)에 전송하도록 임의의 미리 시작된 명령을 허용하고; 비디오 입력 버퍼(175)를 로드하기 전에 사용할 수 있도록 MEC의 비디오 버스를 기다린다. 로드 명령 레지스터의 비트 2k+1:2k는 세그먼트 k의 로드를 제어하고 테이블 3에 도시한 바와 같이 인코딩된다. 비디오 인터페이스 타이밍 제어 레지스터는테이블 4에 리스트되어 있다. 제 15도는 비디오 입력 버퍼(175)의 물리적인 저장 매핑에 대한 논리적인 세그먼트를 도시한다. 비디오 입력 버퍼(175)는 비디오 인터페이스 유니트(185)으로부터의 데이터가 로드된다. 최고 해상도 이미지를 비디오 입력 버퍼(175)에 로드하는 경우, 2개의 화소는 32비트 비디오 데이터 버스를 위한 시간 또는 16비트 데이터 버스를 위한 화소에서 버퍼 RAM의 각 뱅크에 기록된다. 4분 해상도 이미지를 비디오 입력 버퍼(175)에 로드하는 경우, 1 화소만이 한번에 비디오 입력 버퍼(175)로 기록된다. 비디오 입력 버퍼(175)로부터 서치 메모리(160)로 이미지를 복사하는 경우, 8개 화소의 전체 행은 각 비디오 입력 버퍼(175) 뱅크로부터 판독되고 서치 메모리(160)에 기록된다.

비디오 인터페이스 유니트(185)이 로드 명령 레지스터의 내용을 시험함으로써 세그먼트(0)부터 시작하기 위하여 비디오 입력 버퍼(175) 세그먼트를 서비스하기 시작한다. LDVIB(2k+1:2k)의 다양한 인코딩을 위한 비디오 인터페이스 유니트(185)에 의해 얻어지는 동작은 다음과 같고, 여기서 k=0, 1, 2, 3이다:

LDVIB(2k+1:2k) = 00

세그먼트 k 는 로드되지 않는다.

LDVIB(2k+1:2k) = 01:

VkPCNT 화소는 논리적인 이미지 좌표(VkX, VkY)에서 시작하는 각각의 2 * VkLENGTH 이미지 행으로부터 패치되고 행 0, 열 0에서 시작하는 세그먼트 k 에서만 데시메이트되지 않고 저장된다. 만약 VkPCNT가 8 * VkWIDTH 보다 작다면, 세그멘트 k의 2 * VkLENGTH 행의 각각의 제 1 VkPCNT만이 기록된다. VMSEL 핀은 VkMSEL로 세트된다.

LDVIB(2k+1:2k) = 10:

VkPCNT 화소는 논리적인 이미지 좌표(VkX, VkY)에서 시행하는 각각의 4 * VkLENGTH 이미지로부터 패치되고, 데시메이션 필터되며, 행 0, 열 0에서 시작하는 세그먼트 k에만 저장된다(즉, 세그먼트 k는 모든 VkPCNT * VkLENGTH 화소가 로드된다). 만약 VkPCNT가 16 * VkWIDTH 보다 작다면, 세그먼트 k의 2 * VkLENGTH 행의 각각의 제 1 VkPCNT/2 열만이 기록된다. VMSEL 핀은 VkMSEL로 세트된다.

LDVIB(2k+1:2k) = 11:

VkPCNT 화소는 논리적인 이미지 좌표(VkX, VkY)에서 시작하는 2 * VkLENGTH 이미지행의 각각으로부터 패치

되고, 행 0, 열 0에서 시작하는 세그먼트 k에서 데시메이션되지 않고 저장된다. 만약 VkPCNT가 8 * VkWIDTH 보다 작다면, 세그먼트 k의 2 * VkLENGTH 행의 각각의 제 1 VkPCNT 열이 기록된다. 논리적인 행 VnY에서 시작하여, 여기서 n=k+1 이고 VnY가 VkY보다 크거나 또는 같고, 다음 4 * VkLENGTH 이미지 각각으로 부터의 VkPCNT 화소는 또한 데시메이션 필터링되고 그 결과는 행 0, 열 0에서 시작하여 세그먼트 n에 저장되고; 제 15a도를 참조하면 상기 설명한 바와 같이, 최고 해상도 이미지(400) 및 4분 해상도 이미지(401)가 동시에 비디오 입력 버퍼(175)로 로드되는 방법의 실시예를 도시한다. VkPCNT는 16 * VkWIDTH 보다 작거나 같고, 4* VkLENGTH+VnY는 2 * VkLENGTH+VnY보다 작거나 또는 같아야만 한다. 만약 VkPCNT가 16* VkWIDTH보다 작다면, 세그먼트 n의 2 * VkLENGTH 행의 각각의 제 1 VkPCNT/2 열이 기록된다. VMSEL 핀은 VkMSEL로 세트된다.

제 16도는 비디오 입력 버퍼 시퀀서 상태 장치의 상태 천이도(459)를 도시한다. 리셋 이후에, 상태는 LDVIB 또는 LDSM 명령이 CPU(122)로부터 수신될 때까지 상태(0)에 남아있다. CPU 명령의 다른 형식은 상태 장치에 의해 즉시 수행될 수 있도록 얻어질 수 없기 때문에, 상태 장치가 그것을 처리할 때까지 래치된다. 본 발명이 한 실시예에 있어서, LDVIB 명령이 수행 가능하도록 되기 위하여, VIU(100)는 WRMEM 명령의 실행을 위한 기록 비디오 버퍼 유니트에 할당되지 않아야만 한다.

LDVIB 명령이 실행가능하다면, 상태는 상태(1)로 변한다. LDVIB 명령 래치(LDVIB_VLD_LCH)는 또한 리셋된다. 0-에서 시작한 현재 세그먼트를 위한 LDVIB 명령이 조사된다. no-op(0)이면, 세그먼트 포인터(SEG-CNT)는 다음 세그먼트를 지시하기 위하여 증가된다. no-op가 아니라면, 비디오 메모리 판독을 위한 시작 행 어드레스(RD_VM_RA)는 대응하는 Vk_Y 입력으로부터 형성된다. 상태 2로의 천이가 만들어진다. 페이지 모드 판독 사이클(VM_RD_START)을 위한 요청이 VIU로 전송된다. 상태 장치는 상태 3, 4, 5를 통하여 천이되고 페이지 모드 사이클이 종료될 때까지 상태 5에 남아 있다. 그리고나서 상태 6으로 천이된다. 상태 3 동안 증가된 페이지 모드 사이클(RASCNT)의 수는 현재 세그먼트(VLENGTH)의 길이와 비교된다. 만약 세그먼트가 아직 차지 않았다면, 상태 장치는 상태 2로 복귀하고 RD_VN_RA는 다음 페이지 모드 액세스를 위하여 증가된다. 만약 세그먼트가 완전히 찼다면, 상태는 상태 8로 천이한다. 만약 마지막이 아닌 세그먼트 및 현재 세그먼트의 LDVIB 명령이 최고 해상도 및 4분 해상도 이미지를 두 개의 인접한 세그먼트("2중-로드")에 로드 하지 않으면, 상태 장치는 상태 1로 복귀한다. 상태 8에서, RASCNT는 리셋되고, 세그먼트 포인터(SEG-CNT)는 증가된다. 만약 "이중-로드"가 실행되었다면, 세그먼트 포인터는 다음 세그먼트가 스킵되도록 증가되고 상태는 상태 9로 천이한다. 상태 9에서, 세그먼트 포인터는 다시 증가된다. 만약 마지막 세그먼트가 아니라면, 상태는 다시 상태 1로 복귀한다. 그렇지 않다면, 상태는 상태0으로 복귀하고 CPU에 대한 VIB_RDY 상태를 세트한다.

상태 0에서, 만약 LDSM 명령이 실행가능하다면, 상태 장치는 상태 10으로 스위치된다. 상태 10에서, 실행될 마지막 LDVIB 명령이 시험된다. 만약 현재 세그먼트(0에서 시작한)의 LDVIB 명령이 no-op 이라면, 상태는 직접 상태 15로 천이한다. 만약 no-op가 아니면, 상태는 상태 11로 천이하고 LDSM_START 신호는 한 클럭 사이클 동안 주장된다. 상태 장치는 상태 12로 천이한다. 상태 12에서, CPY_IP가 주장되고 상태는 상태 13으로 변화되고, 여기서 세그먼트가 서치 메모리에 복사될 때까지 남아있다. 상태 13 동안, 부행(subrow) 지시자는 매 클럭 사이클마다 증가하고, Vk_WIDTH-1에 의하여 "모듈로된다". RASCNT는 VIB에서 판독되는 논리 홀수/짝수 행쌍 카운트를 유지하는데 사용된다. RASCNT가 Vk_WIDTH-1에 도달하는 경우, 상태는 상태 15로 천이한다. 상태 15에서, 세그먼트 포인터 15는 증가한다. 만약 모든 세그먼트가 아직 처리되지 않았다면, 상캐 장치는 상태 10으로 복귀한다. 그렇지 않다면, 상태 0으로 복귀하고 SM_RDY 상태가 주장된다.

VIB 시퀀서는 또한 LDVIB 명령을 미리 처리하지 않은 경우 비디오 버퍼 유니트에 기록하도록 VIU를 할당 한다. LDVIB 및 기록 버퍼 유니트 요청(AIU_WR_CMD)과 동시에, VIU는 기록 비디오 버퍼 유니트(ALLOC_WRVM)에 할당된다. LDVIB 및 WRMEM의 실행은 일단 실행이 시작되면 다른 것에 의하여 인터럽트되지 않는다.

2. 서치 메모리

서치 메모리(160)는 입력 토큰이 최상의 모션 벡터에 도달하기 위한 CPU 프로그램과 비교하여 이미지의 섹션을 포함한다. 본 발명의 바람직한 실시예에서, 서치 메모리(160)는 512 페이지로 나누어지고, 비디오 입력 버퍼(175)와 같이 4개의 세그먼트로 분할된다. 서치 메모리(160)의 각 페이지는 화소의 2행 × 8열 타일에 대응한다. 차례로, 각 세그먼트는 페이지의 사각형 어레이를 포함하며, 임의의 서치 메모리(60)에서 페이지 열의 최대수는 16이고, 페이지 행의 최대수는 64이다. CPU(122)는 레지스터를 초기화함으로써 서치 메모리(60) 세그먼트 k(k=0,1,2,3)를 구성한다.

제 17도는 서치 메모리(160)의 블록도를 도시한다. 서치 메모리(160)는 SM 저장(460), SM 서퀀서(461), SM 어드레스 발생기(462), 및 SM 데이터 파이프라인 레지스터(463)를 포함한다. SM 시퀀서(461)는 CPU 블록 매칭 명령 및 출력 토큰 버퍼 명령에 대한 복사를 수행한다. 추가로, SM 시퀀서(461)는 VIB 시퀀서를 사용하여 직렬로 작업한다. SM 어드레스 발생기(462)는 SM 저장(460)을 위한 물리적인 어드레스를 계산한다.

서치 메모리 저장(460)은 짝수 뱅크(480) 및 홀수 뱅크(461)를 사용하여 8k 바이트의 논리적인 서치 메모리를 수행한다. 각 뱅크는 SM 저장의 각 뱅크에 대하여 8화소 물리적인 행을 형성하기 위하여 8개의 512 × 8 RAMS(SM_RAMS)로 구성된다. 4개의 논리적인 세그먼트는 이들 CPU-로드된 파라메터에 의해 규정된다: Sk_ORIGIN 및 Sk_WIDTH. Sk_ORIGIN은 짝수/홀수 행 쌍으로 규정되고; Sk_WIDTH는 뱅크당 8화소의 배수로 규정된다. 서치 메모리(160)는 서치 메모리의 대응하는 세그먼트에 VIB 세그먼트를 복사하는 LDSM에 의하여 초기화된다.

제18도는 논리적인 서치 메모리(485)로부터 물리적인 서치 메모리(486)로의 매핑을 도시한다. SM 어드레스 발생기(462)는 서치 RAMS의 두 뱅크에 대한 물리적인 어드레스를 계산한다. 서치 메모리(160)의 논리적인 것으로부터 물리적인 것으로의 변화에 있어서, 블록 매칭동안 물리적인 개시 열 및 행은 다음과 같

이 계산된다:

sm_col_addr = (BMXORG + (SPX 〉〉 1)) % 8;

여기서 "%"는 "모듈 오퍼레이터이고 "〉"는 부호 확장을 갖는 우측 쉬프터이다;

sm_row-addr = SK_ORIGIN + ((BMYORG + (SPY 〉〉 1)) / 2 *

SK_WIDTH + ((BMYORG + (SPX 〉〉 1)) / 8 % SK_WIDTH;

동일한 공식이 블록 매칭 파라메터를 위하여 대치된 OTB 파라메터를 사용하여 LDOTB 명령 수행동안 서치 메모리(160)를 위한 물리적인 개시 열 및 행 어드레스를 계산하기 위하여 사용된다 개시 열 어드레스는 입력 토큰을 사용하여 서치 이미지를 재배열하도록 보간기(190)에 주로 사용된다. LDSM 명령 수행동안, 물리적인 서치 메모리 어드레스는 다음과 같이 계산된다:

idsm_lrow + SK_ORIGIN + Vk_ROW * SK_WIDTH

idsm_lrow = idsm_lrow + Vk_COL;

for (i=0; i 〈 Vk_LENGTH;; i++)

{(subrow_cnt = 0; subrow_cnt 〈 Vk_WIDTH; subrow_cnt ++)

ob_adra = idsm_lrow + subrow_cnt;

idsm_lrow = idsm_lrow + SK_WIDTH:}

SM 어드레스 발생기(462)는 또한 3중 서치에서 제 2 및 제 3 서치 지시의 상대적인 y 좌표를 계산한다. SM 어드레스 발생기(462)는 블록 배칭될 다음 3중 서치의 x 좌표를 계산한다. 제 19도는 서치 메모리(160)를 위한 어드레스를 계산하기 위하여 산술 리소스의 블록도를 도시한다. 491에서 492로 통하는 어드레스는 개시 어드레스 및 다음 서치의 좌표를 형성하기 위하여 사용된다. 가산기(495 및 496)는 서치 메모리의 실제 동작동안 서치 메모리(160)를 위한 기록 및 판독 어드레스를 형성하기 위하여 사용된다.

SM 시퀀서(461)는 블록 매칭 명령 및 OTB 명령에 대한 복사를 실행하는 것을 담당한다. SM 시퀀서(462)는 블록 매칭 프로세서 유니트(250), 토큰 버퍼 유니트(175) 및 보간기(190)에 의하여 요구되는 타이밍 제어 신호를 발생하는 것을 담당한다. 이들 제어 신호는 유니트에서 데이터 경로의 래턴시(latency)를 매칭하기 위하여 적당히 파이프라인화할 것이다.

SM 시퀀서(461)는 CPU에 대한 SPRDY, BMPIDLE, LDOTBRDY 상태 플레그를 갱신한다. SPRDY 상태는 SPY_VLD 입력이 주장된 이후에 1 사이클이 리셋된다. SPY_VLD는 CPU가 CIU의 SPY 레지스터에 사이클을 기록한 후에 한 사이클로 동작된다. SPRDY는 CPU-로드된 서치 파라메터가 SM 어드레스 발생기(462)의 지역 레지스터에 복사되는 사이클상에 세트된다. BMPLDLE 상태 플레그는 SPRDY가 세트되는 것과 동시에 리셋된다. 이것은 마지막 서치가 종료되는 경우 BMP 유니트에서 최소 MAD 검출기 논리를 통하여 리셋된다. LDOTBRDY 상태 플레그는 OTBY_VLD 입력이 주장된 이후에 1 사이클이 리셋된다. OTBY_VLD는 CPU가 CLU의 OTBY 레지스터에 사이클을 기록한 이후에 1 사이클 활성화된다. 이것은 복사의 종료시 출력 데이터 토큰 버퍼 명령으로 세트된다.

CPU(122)는 LDSM 명령 레지스터에 기록함으로써 비디오 입력 버퍼(175)의 현재 내용을 서치 메모리(160)에 복사한다. 상태 장치는 4개의 비디오 입력 버퍼(175) 세그먼트가 전송될 데이터를 포함하도록 결정하기 위하여 LDVIB 명령 레지스터의 내용을 시험한다. 만약 k= 0 또는 2인 동안 LDVIB(2k+1:2) ≠ 00이라면, 상태 장치는 비디오 입력 버퍼(175) 세그먼트 k의 각각의 VkLENGTH 페이지 행으로부터 페이지 행 VkROW, 페이지 열 VkCOL에서 시작하는 서치 메모리 세그먼트 k로 16 * VnWITDH 화소를 복사한다. 만약 n= 1 또는 3이고 k=n-1인 동안 LDVIB(2n+1:2n) ≠ 00 또는 LDVIB(2k+1:2k) = 11(이진수)이라면, 상태 장치는 또한 비디오 입력 버퍼(175) 세그먼트 n의 각 VnLENGTH 페이지 행으로부터 페이지 행 VnROW, 페이지 열 VnCOL에서 시작하는 서치 메모리 세그먼트 n에 16 * VnWITDH를 복사한다.

모든 세그먼트가 복사된 이후에, 상태 장치는 CPU(122)에 신호한다. CPU는 미리 발행된 LDSM 명령이 실행되는 동안, 또는 블록 매칭이 처리 중인 동안 LDSM에 기록되지 않는다. 그러나, LDSM에 기록된 이후에, CPU(122)는 LDVIB 명령이 아직 실행을 시작하지 않을 것이기 때문에 LDSM의 실행을 중단하지 않고 LDVIB에 기록할 수 있다. 제 30도에 도시한 바와 같이, 2차원 좌표 시스템(500 및 501)의 두가지 형식은 서치 메모리의 세그먼트를 어드레싱하는 경우 CPU에 의하여 사용될 수 있다. 절대 (X, Y) 좌표(500)는 세그먼트에 기록된 사각형 화소의 상부 좌측 모서리를 원점으로 가정하고, 좌표값은 항상 양의 정수이다. (X, Y) 좌표(501)는 CPU(122)에 의해 규정된 원점과 관련하여 규정되고 양 또는 음이 될 수 있다. CPU(122)는 비디오 입력 버퍼로부터 서치 메모리로 복사된 화소를 기록하도록 규정되는 경우 절대 좌표를 사용하고 블록 매칭하는 동안 또는 토큰 출력 버퍼(265)에 전송되는 동안 서치 메모리(160)로부터 판독될 화소의 블록을 규정하는 경우 상대적 좌표를 사용한다.

서치 메모리(160)는 매 클럭 사이클마다 16 화소를 출력하고 하나 내지 4개의 분리된 서치 윈도우를 동시에 포함한다. 두 레벨 계층적 서치동안, 서치 윈도우는 통상적으로 최고 또는 4분 해상도 화소를 고정한다. 따라서, Px64에 의해 요청되는 것과 같은 간단한 예측 서치는 계층적으로 수행되는 경우 두개의 서치 윈도우를 사용한다. 계층적으로 수행되는 경우 MPEG 양방향성 예측은 4개의 서치 윈도우: 전방 예측을 위하여 2개 및 후방 예측을 위하여 2개를 필요로 한다.

3, 블록 매칭 프로세서

블록 매칭 프로세서(250)는 토큰 화소 버퍼(270)로부터의 8×8 또는 16×16 블록과 서치 메모리(160)로부터 읽어들인 동일한 사이즈의 블록을 비교한다. 블록들은 CPU(122)에 의하여 규정된 절대 원점과 상대적인 서치 좌표(X, Y)를 사용하여 CPU(122)에 의해 규정된 서치 메모리 세그먼트로부터 선택된다. 16 ×16

블록을 비교하기 위하여, 서치 좌표는 반 화소 해상도를 가질 수 있고, 블록 보간 프로세서(190)는 블록 매칭 프로세서(250)로 전송하기 전에 필요한 만큼 서치 메모리(160)로부터 화소를 보간할 것이다. 블록 매칭 프로세서(250)는 토큰 화소 버퍼의 블록과 CPU(122)에 의해 전송된 각 서치 좌표 쌍에 응답하는 서치 메모리(160)로부터의 3개의 블록을 비교한다. 세 개의 블록은 최상부 블록의 상부 좌측 모서리의 위치를 규정하는 서치 좌표 쌍을 사용하여 서치 메모리(160)에 수직으로 배열된다. 그 블록들은 서치 메모리(160)의 연속적인 또는 교변적인 행에서 시작할 수 있다.

제 21 및 21a도는 블록 매칭 프로세서(250)를 포함하는 블록 매칭 구조의 블록도이다. 블록 매칭 프로세서(250)는 3배의 서치 지점의 MAD를 계산하는 절대 차이 프로세서(251-253)(AD_PROC)를 포함한다. 선-지연 부-유니트(LINE_DLY)은 토큰 데이터의 행 및 블록 매칭된 3중 형식에 근거하여 보간 유니트(190)으로부터 이미지의 행을 지연한다. 선-지연으로 부터의 출력은 AD_PROC 2 및 3을 공급한다. 블록 매칭 프로세서(250)는 또한 최소 MAD 검출기 부-유니트(MINMAD_DET)을 포함한다. 이러한 부-유니트는 3개의 프로세서로부터의 MAD와 현재 최소값을 비교하여 및 최소값을 갱신한다.

바람직한 실시예에 있어서, 블록 매칭 프로세서(250)는 매 50 MHz 클럭 사이클마다 16개 절대 차이의 합을 계산하고 축적할 수 있는 3개의 가산 트리를 포함한다. 제 1 가산단은 이미지 데이터와 토큰 입력 데이터 사이의 차이를 형성하고; 나머지 가산단은 절대 차이를 가산하고 그 합을 축적한다. 이들 가산단에서의 캐리는 제 1 가산단으로 부터의 모든 음의 차이의 절대값을 종결하는데 사용된다. 이러한 계산 능력은 매 120nsec의 최대 비율로 수행될 16 × 16 화소 블록상에; 또는 매 33.3 nsec의 최대 비율에서 8 × 8 블록상에 블록 매칭을 허용한다.

서치 개시에 앞서, CPU(122)는 구성 레지스터를 사용하여 블록 매칭 프로세서(250) 및 관련된 논리를 구성한다. 제 22도는 구성(bmpmode) 레지스터의 다양한 세팅에 응답하여 형성된 블록 매칭 패턴(505, 506, 507, 및 508)을 도시한다. 블록 매칭 프로세서(250)는 3개의 블록 비교 각각에 대하여 MAD를 계산하고 테이블 5-1에 목록된 레지스터를 사용하여 최소값을 추적한다. 16 × 16 블록 비교에 대하여, MAD 16 비트 길이까지 무부호 양수이고; 8 × 8 블록 비교에 대하여, MAD는 14비트 길이까지 가능하다. 최적으로, CPU(122)는 최소값이 계산될 때 3개의 MADs 중 하나 또는 그 이상이 무시될 수 있다.

MAD 데이터를 고정하는 레지스터는 CPU(122)에 의해서 임의의 시간에 판독될 수 있다. MADMS은 및 MADMS 레지스터 둘 모두는 무부호 양의 정수를 포함하는 것처럼 다루어지고 12비트 정수를 형성하기 위하여 0비트를 사용하여 부호 확장된다. MADX 및 MADY는 현재 최소 MAD의 위치를 지시하고 CPU(122)에 의하여 전송된 서치 좌표로부터 프로세서(250)를 블록 매칭함으로써 유도된다. MADX 및 MADY는 부호 비트가 되는 비트 7을 갖는 2의 보수 정수를 포함하고 판독시 12비트로 부호-확장된다. 블록 매칭 프로세서(250)는 CPU(122)가 INITMAD 명령 레지스터에 기록하는 경우 최소 MAD를 "무한대"로 초기화한다.

서치 좌표 (X, Y)는 BMXORG 및 BMYORG에 저장된 절대 원점과 관련되고 토큰 화소 버퍼(210)로부터 입력 블록과 비교될 서치 메모리(160) 내의 3개의 수직 할당 블록의 상부 좌표 모서리의 위치를 가리킨다. 각 X 또는 Y 좌표는 이진 지점이 비트 1과 0 사이에 있는 범위 {-64.0, 63.5]내의 8비트 2의 보수 분수 정수이다. 반 화소상의 서치 좌표는 블록 보간 프로세서(190)에 의하여 수평 및/또는 수직 화소 보간 요청을 증가시킨다. 8 × 8 블록 매칭을 위하여, 모든 서치 좌표들은 완전한 화소 증가될 것이라고 자정하고, 각 좌표의 최하위 비트(즉, 분수 부분)는 그의 실제 값을 고려하지 않고 0으로 다루어진다. 서치 좌표 3개당 PCLK 클럭 사이클의 수의 항에 블록 매칭 성능은 BMPMODE 및 보간의 다양한 조합을 위하여 테이블 5-2에 도시되어 있다. 보간 결과는 블록 매칭 프로세서(250)로 전송된다.

서치 메모리를 판독하는 경우, 블록 매칭 프로세서(250)는 다음과 같이 상대적인 서치 좌표(xrel, yrel)를 절대 좌표(xabs, yabs)로 번역한다:

xabs = ((xrdel 〉〉 1) + BMXORG) mod (8 * SkWIDTH)

yabs = ((yrel〉〉 1) + BMyORG) & 0x7f

여기서 "a〉〉b는 부호 확장된 b 비트에 의한 우측 시프트를 의미하고, "a&b"는 역=비트 b를 "앤드"하는 것을 의미하고, "0x7f"는, 8진수 "7f"이고, 모든 가산은 8비트 2의 보수 산술을 사용하여 행해진다. 블록 매칭 프로세서(250)는 메모리 폭에 의하여 서치 메모리 "모듈"을 수평으로 어드레스한다; 즉, 서치 좌표는 수평으로 "랩-어라운드"될 수 있고 서치 메모리의 윈도우가 원형 버퍼와 같이 동작하도록 허용하여 이미지에 대하여 수평으로 "슬라이드"된다. 그러나, 수직 어드레스 랩-어라운드가 제공되지 않고, 서치 좌표는 서치 메모리 세그먼트의 제 1 또는 마지막 행을 오버런하도록 행 액세스를 야기하고, 패치된 데이터는 부정확하게 될 것이고 하나 또는 그 이상의 에러가 있는 MAD 계산을 야기한다.

CPU(122)는 테이블 6에 도시된 레지스터에 기록함으로써 블록 매칭 프로세서(250)에 의하여 처리되기 위한 서치 좌표의 패턴을 구성한다. CPU(122)는 SPRDY 상태 비트가 "1"이 되는 경우 테이블 6의 레지스터에 기록된다. CPU(122)는 SPY에 기록된다. 블록 매칭 프로세서(250)는 테이블 6의 모든 레지스터가 새로운 서치 패턴에 대하여 유효한 데이터를 포함하는 지시로 인하여 SPY에 대한 기록을 해석한다. CPU(122)에 의하여 SPY에 대한 기록은 또한 자동적으로 SPRDY를 "0"으로 세트한다. 블록 매칭 프로세서(250)가 새로운 패턴을 처리하기 시작할 때, 블록 매칭 프로세서(250)는 블록 매칭동안 사용하기 위한 테이블 6의 레지스터를 작업중인 레지스터 자체의 세트에 복사하고, SPRDY를 "1"로 세트하여 CPU(122)가 현재 패턴이 실행을 종결하기 전에 다른 패턴을 설명하도록 한다.

블록 매칭 프로세서(250)는 SPRDY 상태 비트가 "1"이 되고 모든 블록 매칭 프로세서 파이프라인이 비는 경우(즉, 최종 서치 패턴으로부터 최종 MAD가 계산되는 경우) BMPIDLE 상태 비트를 "1"로 세트한다. 처리 과정에서 임의의 서치 방해를 피하기 위하여 CPU(122)는 BMPIDLE이 "1" 인 경우 INITMAD 또는 테이블 7에 목록된 블록 매칭 구조 레지스터 중 어느 것에 기록될 수 있다.

제 23도는 블록 매칭 프로세서(250)가 테이블 6의 레지스터의 의하여 설명된 서치 패턴을 수행하는 방법의 실시예(510)를 도시한다. 블록 매칭 프로세서(250)는 k=0,1,...,SPCNT에 대하여 상대적인 좌표가 (SPX+k * SPINC, SPY)에 의해 주어지는 전체 SPCNT+1의 3배에 대하여 서치 패턴을 매칭한다. SPMASK 레지

스터를 이진수 "111"과는 다르게 세팅함으로써, CPU는 블록 매칭 프로세서(250)의 최소 MAD 계산으로부터 제외될 "마스크 아웃" MAD가 dlf 수 있다.

4. 블록 보간 프로세서

X 또는 Y 서치 좌표의 최하위 비트가 "1"이고 16 × 16 블록 매칭이 수행되면, 블록 보간 프로세서(190)는 서치 메모리 데이터의 수평 및/또는 수직보간을 수행한다. 본 발명의 바람직한 실시예에서, 블록 보간 프로세서(190)의 출력율은 블록 매칭 프로세서(250)의 최대 입력율과 매칭한다. 테이블 8은 블록 보간기(190)에 의하여 서치 메모리(160)로부터 판독한 다양한 블록 사이즈를 요약하고 블록 매칭 프로세서(250)에 의하여 처리된 16 × 16 블록을 형성하도록 보간된다.

화소 보간은 국제 표준 위원회 규정에 의하여 성취된 규약을 사용하여 테이블 8에 도시된 블록 사이즈상에서 수행된다. 예시한 바와 같이, 반화소 공간을 사용하여 논리적 그리드상의 다음 화소의 3 × 3 배열을 고려하자:

P1 H P2

V B X

P3 X P4

화소 P1, P2, P3, 및 P4는 전체 화소 좌표에 위치되고; 화소 H, V, 및 B는 반 화소 좌표에 위치되며 P1, P2, P3, 및 P4로부터 보간될 필요가 있다. MEC(20)에 의해 사용된 이진 보간 공식은 다음과 같다.

H = (P1 + P2) // 2 (수평 보간만)

V = (P1 + P3) // 2 (수직 보간만)

B = (P1 + P2 + P3 + P4) // 4 (양방향 보간)

여기서 "//"은 정수 몫을 가장 가까운 정수로 반올림하는 반올림된 반-정수값을 갖는 것을 지시한다.

제 21도의 블록도는 크로스바 스위치(509)와 관련된 보간기(190) 및 교체 mux(507)를 도시한다. 크로스바 스위치 부-유니트(XBARS)은 서치 메모리(160) 데이터 출력의 각 뱅크에 대하여 8개의 8가지 방법의 화소 폭 스위치를 실행한다. 스위치의 개별 세트는 항상 각 뱅크의 영 0에서 시작하는 대신에 블록 매칭 또는 로드 출력 토큰 버퍼가 어떤 열로부터 개시하도록 한다. 서치 메모리 데이터의 두 개의 뱅크를 위한 크로스바 스위치는 동이한 방법으로 동작된다.

라인 교체 부-유니트(SWAP_MUX)은 만약 원한다며, 블록 매칭 프로세서(250) 출력 토큰 버퍼에 전송하기 전에 서치 메모리(160)로부터 짝수 및 홀수 행을 교체한다. 이러한 논리는 블록 매칭을 허용하거나 홀수 논리 행으로부터 시작하기 위하여 출력 토큰 버퍼에 복사한다.

보간 유니트(190)은 또한 반-화소 보간 브로세서(512)(INTERPOLATE)를 포함한다. 이것은 X 축 또는 Y축상의 이미지, 또는 둘 모두, 또는 둘 모두 아닌 이미지를 보간할 수 있다. 보간기(512)는 16 화소 논리행을 3개의 통로에 나누어 놓는다. 제 1 통로에 있어서, 6개의 보간된 화소는 논리 행당 출력이다. 제 2 통로에서, 다음 6개의 화소가 출력된다. 마지막으로, 제 3 통로에 있어서, 미지막 4화소가 출력된다. 제 24도는 보간기(512)의 구조를 도시한다. 이것은 어떤 종류의 보간에 대해서도 3 클럭 사이클의 래턴시를 갖는다. 보간기의 마지막 단 멀티플렉서는 보간이 목표되지 않을 때 측로 기능을 한다. 그러한 경우, 각 서치 메모리 뱅크로 부터의 전체 8 화소가 보간기(190)를 통하여 전달된다.

5 . 중앙 처리 장치

CPU(122)는 본 발명이 모션 평가 코프로세서 칩상의 모든 동작을 조정한다. 제 25도는 본 발명의 CPU(122)의 한 실시예를 도시한다. 제 25도의 블록도에서, CPU(122)는 12 비트 RISC 마이크로제어기 커어(700) 및 64 워드 스크래치패드 메모리(125)를 포함한다. CPU(122)는 또한 디버그 제어기(710)를 포함한다. 코어(700)는 32 워드, 4 포트 레지스터 파일(702), 12 비트 ALU(706), 및 12비트 양방향 원통형 쉬프터(708)를 포함한다. CPU(122)는 몇 개의 레지스터와 프로그램 실행과정에서 액세스될 수 있는 메모리를 포함한다. 대부분의 CPU(122) 레지스터는 CPU 메모리 맵내에 위치하고 로드 및 저장 명령을 사용하는 CPU 프로그램에 의하여 액세스될 수 있다. 특별한 레지스터들은 CPU(122)가 CPU-to-호스트 인터럽트를 통하여 외부 호스트 프로세서(38)(제 1도)와 통신하도록 제공될 수도 있다.

CPU(122)는 기능적인 또는 디버그 모드 중 하나에서 동작한다. 상태 장치 내장 디버그 제어기(710)는 호스트 프로세서(38)로부터 수신된 명령에 근거하여 CPU(122)의 동작 모드를 결정한다. 디버그 제어기(720)에 대한 상태도(720)가 제 26도에 도시된다. 상태 천이는 시스템 클럭의 각 모서리에서 발생하고 호스트 메모리 맵의 RUN, HALT, 및 STEP 위치에 기록하는 호스트 프로세서(38)에 의해 제어된다.

a. 디버그 모드

디버그 모드에서, CPU(122)는 중지되고 호스트 프로세서(38)의 단일 단계 제어하에서만 명령을 수행한다. 디버그 모드동안, 호스트 프로세서(38)는 CPU(122) 레지스터 파일 및 프로그램 메모리, CPU(122) 메모리 버스에 연결된 모든 레지스터 및 메모리를 조사 및/또는 변경할 수 있고, CPU 현재(PC) 및 다음 프로그램(nPC) 카운터를 판독한다. PC 레지스터는 디버그 제어기(710)내에 있고, 그의 입력은 제 25도의 nPC 레지스터의 출력에 연결된다. CPU(122)는 MEC이 리셋될 때 TM 상태를 통하여 또는 CPU(122)가 기능 모드에 있는 경우 호스트가 HALT로 기록되는 TM상태를 통하여 디버그 모드가 된다. 디버그 모드에 있는 동안, CPU(122)는 TM, TW, 및 TS 상태 중 하나이다.

디버그 모드에서, CPU 명령은 PCLK의 다중 주기로 구성된 "사이클"에서 수행된다. 제 27도에 도시된 바와 같이, 각 사이클은 TM 상태에서 시작하고 TS 상태에서 종료된다. 단일 클럭 TM 및 TS 상태는 디버그 제어기(710)가 TW 상태에 있고 호스트 프로세서가 레지스터 및 메모리의 내용을 조사 및/또는 변경하는 동안

규정되지 않은 수의 클럭 주기(722)이다. 하나의 명령은 정확히 디버그 사이클보다 수행된다.

제 27도의 T_0에서 시작하는 TM 상태동안, 디버그 제어기(710)는 앞선 디버그 (또는, 가능하게, 기능적인) 사이클 동안 명령 레지스터(IR)에 존재하는 어떤 로드 또는 저장 명령을 제공한다. 로드 명령에 의한 데이터 판독은 12비트 메모리 데이터 고정 레지스터 내부 디버그 제어기(710)로 패치되고 클럭된다. 레지스터 파일, DAR, nPC 및 명령 레지스터의 CPU(122) 프로그램 지시된 로드는 디스에이블되고, ALU(706) 및 레지스터 파일(702) B 포트는 CPU 메모리 어드레스 및 데이터 출력 버스를 구동하는 것이 방지된다. 디버그 제어기(710)는 또한 DBRDY 상태 비트를 "1"로 세트하고 TM 상태 동안 HALT 플립-플롭을 클리어한다.

TM 상태에서 하나의 PCLK 사이클을 소비한 이후에, 디버그 제어기(710)는 그것이 다시 HALT 플립-플롭을 클리어하고 호스트 프로세서가 nPC 또는 PC 레지스터를 판독하고, CPU 레지스터 파일의 어떤 어드레스를 판독 또는 기록하며, 또는 CPU 메모리 맵의 어떤 어드레스를 판독 또는 기록하도록 하는 동안 T_1에서 시작하는 TW 상태로 들어간다. 호스트는 그의 어드레스를 호스트 메모리 맵의 RADDR(12:0)에 기록함으로써 레지지스터 또는 메모리를 판독한다. 어드레스 할당이 테이블 9에 나타나 있다. RADDR(12:0)에 대한 호스트 기록은 DBRDY 비트를 "0"으로 세트한다. 디버그 제어기(710)는 RADDR(12:0)에 의해 선택된 위치를 DBDAT(11:0)로 판독하고 종결시 DBRAY를 "1"로 세트한다. 그러는 동안, 호스트 프로세서(38)는 DBRDY를 폴할 수 있고 DBRDY가 "1"인 경우 DBDAT(11:0)를 판독한다.

호스트 프로세서(38)는 데이터를 DBDAT(11:0)에 먼저 기록함으로써 데이터를 레지스터 또는 메모리에 기록하고, 레지스터 또는 메모리의 어드레스를 호스트 메모리 맵의 WADDR(12:0)에 기록한다. 호스트에 WADDR(12:0)에 대한 호스트 기록은 또 DBRDY 상태 비트를 "0"으로 세트한다. 디버 그 제어기(710)는 DBDAT(11:0)의 내용을 WADDR(12:0)의 어드레스에 기록하고 종결시 DBDAT를 "1"로 세트한다. 호스튼 다른 판독 또는 기록을 시도하기 전에 DBRDY가 "1"로 되도록 DBRDY를 폴한다.

디버그 제어기(110) DBC가 TW 상에 있는 동안 만약 STEP 또는 RUN 플립플롭이 세트된다면, 디버그 제어기(710)는 제 27도에 도시된 바와 같이 T_2에서 TS 상태로 천이한다. 호스트 프로세서(38)는 디버그 제어기(170)가 DBRDY 상태 비트를 "0"으로 세트되도록 STEP에 기록한다. STEP에 대한 각각의 기록은 정확히 하나의 CPU 명령을 실행하는 것을 보장하여, 호스트가 DBRDY가 DBRDY를 판독하거나 STEP에 기록하는 사이에 적어도 4개의 PCLK 사이클의 지연을 간단히 주장함으로써 "1"이 되는 것을 보장할 수 없으면 STEP에 기록하지 않는다.

TS상태동안, 디버그 제어기(710)는 STEP 및 HALT 플립 플롭을 클리어하고 새로운 레지스터 상태(PC, nPC, 및 명령 레지스터의 새로운 내용을 포함하여)가 T_3에서 클럭되도록 T_0에서 미리 디스에이블된 레지스터 로드 및 CPU(122) 버스 신호 모두를 재-인에이블한다. 이것은 현재 디버그 사이클 이전의 사이클로부터의 로드 명령위 "지연 슬롯"이라면 레지스터를 고정하는 디버그 제어기 메모리 데이터의 내용을 레지스터 파일의 적당한 어드레스에 클럭킹하는 것을 포함한다. TS는 DBRDY의 상태를 변경하지 않는다.

만약 RUN 플립-플롭이 TS 상태동안 세트되지 않는다면, 디버그 제어기(710)는 T3에서 다음 디버그 사이클을 시작하기 위하여 TM상태로 천이한다. 만약 RUN 플립-플롭이 세트되었다면, 디버그 제어기(710)는 TR로 천이하고, CPU(122)는 기능 모드로 들어간다.

b. 기능 모드

CPU(122)는 호스트 프로세서(38)가 RUN으로 기록하는 동안 디버그에서 기능 모드로 천이한다. 기능 모드에 있는 경우, CPU(122)는 메모리에 저장된 RISC 프로그램을 수행하고 내부 메모리-맵된 로드 및 저장 동안을 통하여 칩상의 다른 유니트를 제어한다. RISC 프로그램은 블록 매칭 프로세서에 의해 실행된 서치7 알고리듬을 지시하고, 비디오 입력 버퍼, 토큰 입력버퍼, 및 토큰 출력 버퍼를 제어한다. RISC 코어는 단일 클럭 사이클 동안 메모리 로드 및 프로그램 점프와 관련된 것을 제외한 그것의 22 명령 각각의 수행을 제공한다. 메모리 로드 및 프로그램 점프와 관련한 명령은 종결되기 위해서는 여분의 클럭 사이클을 요구한다.

제 28도는 CPU(122)에 대한 실시예 명령 형식을 도시한다. 각 명령은 32비트 길이이고 명령의 형식을 규정하기 위하여 5비트 OP코드 필드(718)를 포함한다. 5비트 각각의 3 포트 식별 필드(719)는 레지스터 포트 및 뒤따르는 12 비트 어드레스 필드 비트를 식별한다.

기능 모드 동안, 레지스터 파일(702)(제 25도)와 A, B, C를 위한 어드레스는 명령 레지스터(IR)의 Ra, Rb, 및 Rc 필드(719)에 의해 각각 제공된다. 로드 명령에 의해 패치된 데이터를 저장하기 위하여 사용되는 D-포트의 어드레스는 패치 레턴시를 보상하기 위하여 하나의 PCLK 사이클에 의하여 명령 레지스터의 Rc 필드를 지연하는 D 포트 어드레스 레지스터(DAR)로부터 입력된다. 두 개의 C 및 D 포트로부터 동시에 동일한 레지스터에 대한 기록은 정의되지 않은 결과를 가진다.

로드 및 저장 명령에 의해 생성된 메모리 어드레스는 CPU 메모리 맵(테이블 13)을 참고한다. 어드레스는 로드 및 저장 명령이 명령 레지스터에 있는 사이클동안 ALU(706)로부터 12비트 내부 어드레스 버스상으로 구동된다. 동일한 사이클 동안, 저장 명령에 의하여 메모리에 기록된 어떤 데이터는 또한 레지스터 파일(702)의 B포트로부터 12비트 출력 데이터 버스상으로 구동된다. 어드레스 및 데이터 둘 모두(만약 적용가능하다면)는 버스 목적지에서 레지스터로 클럭된다. 다음 사이클 동안, 저장 데이터는 어드레스된 위치에 기록되고 로드 데이터는 어드레스 위치로부터 패치되며 레지스터 파일(702)의 D 포트에 기록된다. 래턴시 때문에, 로드 명령을 사용하여 패치된 데이터는 로드 명령에 후속되는 제 2 명령까지 사용될 수 없다.

브랜치 명령(즉, "B"와 함께 시작하는 op코드 기계어)은 서브루틴을 호출하기 위하여 사용되는 "링크" 특성을 포함한다. 예를 들면, 레지스터Rc는 JALR 명령을 통하여 서브루틴의 종단으로 실제로 점프된 (nPC) + 1 ("nPC"=다음 프로그램 카운터")에 의해 주어진 서브루틴 복귀 어드레스를 고정한다. 링크 동작은 Rc=R0로 세팅함으로써 브랜치 또는 점프 명령에서 효과적으로 디스에이블될수 있다.

테이블 10의 SRA, SRL, 및 SLL op코드는 0과 15 사이의 무부호 수로 보간된 명령 const 필드의 최하위 4 비트로부터 시프트 카운트를 얻는다; const 비트 15:4는 무시된다. 12 또는 그 이상의 시프트 카운트를

사용하는 논리적인 좌측 또는 우측 시프트(각각 SLL 또는 SRL op코드를 사용하여)는 12비트 0 결과를 발생하는 반면, 11 또는 그 이상의 시프트 카운트를 사용하는 산술 우측 시프트(SRA를 사용하여)는 입력 최상부 비트의 12회 복사로 구성된다는 결과를 생성할 수 있다.

6. 호스트 프로세서 인터페이스

MEC(20)은 16비트 데이터를 제공하고 다양한 호스트 프로세서(예를 들면, 제 1도의 호스트 프로세서(38))와 다양한 접속을 방해한다. 본 발명의 한 실시예에서, 호스트 가시 MEC 레지스터 및 메모리는 2048 워드 × 16비트 어드레스 공간으로 맵된다. MEC(20)은 호스트 프로세서에 대한 메모리 슬래이브로 나타난다.

MEC(20)내에 호스트-판독 가능한 및/또는 기록 가능한 메모리 위치 및 레지스터의 어드레스 할당은 테이블 11로 규정된다. 메모리 맵의 많은 레지스터 및 메모리들은 정의 되어 있지 않거나 예약된 비트들을 포함한다. 호스트 프로세서는 이들 레지스터 및 메모리에 기록하는 경우 이들 비트의 위치에 0을 기록한다. 그러나, 이들 비트 위치로 부터의 데이터 판독은 항상 정의되지 않을 것이다. 메모리 맵에서 정의되어 있지 않은 어드레스에 액세스하는 것은 예측할 수 없는 결과를 초래할 수 있다. 또한, 판독만 가능한 어드레스에 호스트 프로세서 기록 또는 기록만 가능한 어드레스로부터 호스트 프로세서 판독은 질문의 어드레스에 있는 데이터에 영향을 주지 않고 임의의 데이터 판독이 정의되지 않을 것이다.

a. 호스트 인터페이스 버스 전송.

테이블 12에 규정된 바와 같은 MEC(20) 호스트 인터페이스 버스는 1 클럭, 11 어드레스, 16 데이터, 1 인터럽트, 및 6개의 제어 신호 비트를 포함한다. 모든 신호 레벨은 TTL과 호환된다. 버스는 슬레이브 온리 모드에서 동작하고 논-버스트 및 버스트 판독 및 기록 버스 사이클 둘 모두를 지지한다. 버스 전송은 16비트 폭이다. 두 데이터 전송 타이밍 모드, 모드 0 또는 모드 1 중 하나가 어떤 액세스를 위하여 선택될 수 있다.

논-버스트 판독 사이클 타이밍은 제 29도에 도시되었다. 모든 논-버스트 판독 액세스는 타이밍 모드 0을 사용한다. 판독 사이클은 HA(10:0)상으로 판독되는 어드레스를 주장하고 판독 전송이 수행되는 것을 지시하기 위하여 HADS-를 통하여 MEC(20)으로 그것을 크럭킹하는 호스트 프로세서와 함계 시작한다. HMODE는 타이밍 모드 0을 선택하기 위하여 로우도 세트된다. HBRST(800)은 HADS-가 사이클의 시점에서 주장되는 동안 MEC(20)에 의해 무시되고, HA(10:0), HWE-, 및 HMODE는 HADS-가 주장되지 않을 때마다 무시된다는 것에 주의하라.

MEC(20)은 HADS-, HA(10:0), HWE-, 및 HMODE의 표본화를 따르기 위하여 제 2 클럭 사이클과 함계 시작하는 데이터 출력 구동기상를 작동한다. 한 클럭 사이클 이후에, MEC은 버스상에 유효 데이터를 주장하고 그렇게 행해진 호스트에 신호하기 위하여 HRDY-를 주장한다. 제 29도에 도시된 바와 같이, 데이터 및 HRDY- 둘 모두는 단일 클럭 사이클에 대해서만 주장된다. MEC(20)은 뒤따르는 클럭 사이클중에 데이터 출력 구동기를 턴 오프한다.

논-버스트 판독(또는 기록)을 종결하기 위하여, HBRST(800)는 MEC(20)이 HRDY-를 주장하는 클럭 사이클동안 주장되지 않는다(로우로 세트). 만약 HBRST(800)이 주장되지 않는다면, MEC(20)은 버스트 사이클이 진행중이고 자동적으로 다음 연속적인 어드레스에 액세스하기 시작한다고 가정한다.

또한 제 29도에 도시한 바와 같이, HADS-는 이전 사이클 동안 HRDY-의 주장을 즉각적으로 따르는 클럭 사이클동안 다른 판독(또는 기록) 사이클을 시작하기 위하여 주장될 수 있다. 본 발명의 실시예에 있어서, 연속적인 액세스의 파이프라이닝은 제공되지 않고 HADS-는 HRDY-가 주장되는 클럭 사이클 동안 주장되지 않는다.

논-버스트 기록 사이클 타이밍은 타이밍 모드 0 및 1 둘 모두에 대하여 제 30도에 도시된다. 기록 사이클은 HA(10:0)상에 기록된 어드레스를 주장하고 HCLK의 다음 상승 모서리상으로 HADS- 신호를 통하여 MEC(20)에 그것을 클럭킹하는 호스트 프로세서를 사용하여 시작된다. HWE-는 기록 전송이 수행되는 것을 지시하기 위하여 HADS-를 사용하여 주장되고(로우), HMODE는 각각 타이밍 모드 0 또는 1을 선택하기 위하여 로우 또는 하이로 세트된다.

모드 0에 대하여, MEC(20)은 HADS-의 샘플링 다음 HCLK의 제 2 상승 모서리에서 데이터 버스를 표본화하고, 모드 1동안, MEC(20)은 제 1 상승 모서리에서 데이터 버스를 표본화한다. 그러나, 호스트 프로세서 임의의 바로 직전 판독 사이클의 종단에서 데이터 출력 구동기를 턴 오프하기 위하여 MEC(20)에 필요한 시간만큼의 어드레스 버스상에 어드레스를 주장하는 것과 동시에 데이터 버스상에 데이터를 주장하기 시작할 것이다.

MEC(20)은 그것이 데이터를 샘플하는 클럭 사이클동안 HRDY-를 주장한다. 논-버스트 판독 사이클로 인하여, HBRST(800)는 또한 액세스를 종료하기 위하여 이러한 클럭 사이클 동안 주장(로우)되지 않아야 한다.

버스트 사이클은 매우 효과적인 방법으로 호스트 프로세서 및 MEC(20) 사이 연속적인 어드레스에서 데이터 워드의 스트림을 전송하기 위하여 사용될 수 있다. 타이밍 모드 0을 사용하는 버스트 사이클은 매 두 클럭 사이클마다 하나의 데이터 워드를 효과적으로 전송할 수 있고, 반면에 모드 1 버스트는 매 클럭 사이클상에 하나의 데이터 워드를 효과적으로 전송할 수 있다. 버스트 사이클을 MEC의 호스트 메모리 맵(테이블 12)내의 어떤 어드레스에서 시작할 수 있다.

4 워드 버스트 판독 사이클을 위한 모드 0 타이밍이 제 31도에 도시되어 있다. 사이클은 HA(10:0)상에 판독된 제 1 워드의 어드레스를 주장하고 동시에 HADS-를 주장하는 호스트 프로세서를 사용하여 시작한다. HWE는 또한 판독 전송이 수행되는 것을 지시하기 위하여 HADS-와 함께 주장되지 않고(하이로 세트), HMODE는 모드 0을 선택하기 위하여 로우로 세트된다. MEC(20)은 HADS-의 샘플링 다음 제 2 클럭 사이클과 함께 시작하는 그의 데이터 버스 출력 구동기를 턴 온한다. 한 클럭 사이클 이후에, MEC(20)은 버스상에 유효 데이터를 주장하고 단일 클럭 사이클에 대하여 HRDY-를 주장한다.

다음 연속되는 어드레스에서 다른 액세스를 요청하기 위하여, HBRST(800)은 MEC(20)이 HRDY-를 주장하는

클럭 사이클동안 주장된다(하이로 세트). MEC(20)은 버스트동안 HA(10:0) 핀의 상태를 무시한다. 버스트 사이클의 길이는 실제로 한계가 없다; 증가될 때 2047 어드레스는 자동적으로 0 주위를 감싼다. MEC(20)이 HRDY-를 주장하는 동안 만약 HBRST(800)가 주장된다면, MEC(20)은 두 클럭 이후에 데이터 버스상에 다음 액세스로부터의 유효 데이터를 주장하고, 다시 HRDY-를 주장한다. HBRST(800) 및 HRDY-의 일반적인 주장에 앞서서, 이전의 액세스로부터의 데이터는 버스상에 주장된채로 남아 있다.

버스 시퀀스의 최종 액세스는 HDRY- 클럭 사이클에 대응하는 동안 HBRST의 비주장에 의하여 신호되고, MEC(20)은 다음 상승 클럭 모서리상에서 버스트 사이클을 종료하며 그의 데이터 버스 출력 구동기를 턴오프한다.

제 32도는 타이밍 모드 1을 사용하는 7개의 워드 버스트 판독 사이클을 나타낸다. 모드 1에서 버스트는 다음 8비트 레지스터에 미리 로드된 버스트의 길이를 판독한다: BLENGTH(7:0) - 버스트 길이. BLENGTH는 MEC(20) 호스트 메모리 맵의 일부분이고 논-버스트 기록 사이클을 사용하여 초기화될 수 있다; 그의 리셋 상태는 0이다. BLENGTH는 판독을 위한 전체 워드의 수보다 적은 두 개가 로드된다; 예를 들면, 제 26도에 도시한 실시예와 같이 BLENGTH = 5. 따라서, 모드 1 버스트 판독의 최소 길이는 2 워드이고 최대 길이는 255 + 2 = 257워드이다. BLENGTH는 모드 0 버스트 판독 사이클의 수행에 영향을 미치지 않을 뿐만 아니라, 어떠한 종류의 버스트 판독 사이클을 수행하는데도 영향을 미치지 않는다.

제 31도에 도시되어 있는 모드 1 및 모드 0 버스트 판독 사이클 사이의 차이는 모드 1 사이클 동안, HMODE가 초기에 (로우 대신) 하이로 세트되며 다음으로부터 하나의 버스트 액세스와 분리하여 (두 개의 클럭 대신) 단일 클럭 사이클만 존재한다는 것이다. 그러나 모드 0 버스트 판독의 경우와 같이, HBRST(800)는 제 1 워드 이후에 액세스될 매 추가 워드를 위한 버스트 동안 주장되고, MEC(20)은 데이터 버스상에 주장하는 매 단어에 대하여 HRDY-를 주장한다. 그러나, 모드 0 버스트 판독과는 달리, 모드 1 버스트 판독은 마지막 액세스동안 주장되지 않을 HBRST(800)를 요구하지 않는다. 또한, 만약 HBRST(800)가 모든 (BLENGTH + 2) 워드를 판독하기에 앞서 모드 1 버스트 판독 사이클동안 주장되지 않는다면, 버스트 사이클은 비주장을 따르도록 다음 클럭의 상승 모서리 상에서 종결된다. 내부 파이프 라이닝으로 인하여, MEC(20)은 이 워드가 데이터 버스상에 결코 주장되지 않을 지라도 종결 지점에서 하나의 추가 워드를 판독한다. "초기 종결"C"일반적인 종결"과는 반대로)과 같은 실시예는 제 33도에 나타나 있다. 제 33도에 있어서, HBRST(800)는 "데이터 0"에 대하여 HRDY- 사이클 동안 주장되지 않고, 그 지점에서 버스트가 종결된다. 그러나, 내부적으로 MEC(20)은 "데이터 1"을 판독할 것이다.

4 워드 모드 0 버스트 기록 사이클은 제 34도에 도시되어 있다. 모드 0 버스트 기록 사이클은 모드 0 논-버스트 기록 사이클과 같이 시작하고 MEC(20)에 의해 HRDY-의 매 주장동안 HBRST(800)의 주장에 의해서 버스트 기록 사이클로 효과적으로 변환된다. MEC(20)은 각 추가 버스트 액세스상에 기록될 어드레스를 내부적으로 증가한다. 데이터는 매 두 클럭 사이클 마다 하나의 워드 비율로 기록되고, MEC(20)에 의하여 샘플된 것보다 먼저 데이터를 셋업하기 위하여 호스트 프로세서 두 클럭 사이클을 제공한다. 버스트 사이클은 최종 버스트 기록 액세스동안 HBRST의 비주장에 의하여 종결된다.

7 워드 모드 1 버스트 기록 액세스는 제 35도에 나타나 있다. 다시, 그것은 모드 1 논-버스트 기록 액세스와 같이 시작하고 MEC에 이한 각 데이터 워드의 샘플링 동안 HBRST의 주장에 의하여 버스트 기록으로 변환된다. 데이터는 하나의 워드 클럭 사이클의 비율로 기록되고, MEC(20)에 의하여 샘플되기에 앞서 데이터를 셋업하는 단일 클럭 사이클만을 호스트 프로세서에 제공한다. 버스트 사이클은 최종 버스트 기록 액세스 동안 HBRST의 비주장에 의하여 종결된다.

b. 호스트 프로세서/ MEC 통신.

MEC CPU(122) 및 호스트 프로세서는 CPU(122) 프로그램의 수행과 관련하여 그들 사이로 데이터를 전달하기 위하여 두 개의 12비트 레지스터를 사용한다. HDIN(11:0) 레지스터는 CPU(122)에 의하여 연속적으로 판독하기 위하여 호스트 프로세서에 의하여 임의의 데이터로 기록될 수 있다. CPU(122)는 HDIN에 기록될 수 없고, 그것으로부터 호스트 프로세서를 판독할 수도 없다. MEC(20)은 호스트가 HDIN에 기록할 때마다 HDINRDY 상태 비트는 "1"로 및 HDINACK 상태 비트는 "0"으로 세트하고 CPU(122)가 HDIN을 판독할때마다 HDINRDY는 "0"으로 및 HDINRDY는 "1"로 세트한다. HDIN는 CPU(122) 및 메모리 맵 둘 모두에 나타난다. HDINACK 및 HDINRDY는 각각 호스트 메모리 맵 및 CPU(122)메모리 맵에만 나타난다.

반대로, HDOUT(11:0) 레지스터는 호스트 프로세서에 의하여 연속적으로 판독되기 위하여 CPU(122)에 의해 임의의 데이터로 기록된다. 호스트 프로세서는 HDOUT에 기록하지 않고, HDOUT로부터 CPU(122)를 판독하지 않는다. MEC(20)은 CPU(122)가 HDOUT에 기록할 때마다 HDOUTRDY 상태 비트를 "1"로 및 HDOUTACK 상태 비트를 "0"으로 세트하고; 호스트가 HDOUT를 판독할 때마다 HDOUTRDY를 "0"으로 및 HDOUTACK를 "1"로 세트한다. HDOUT은 각각 CPU(122) 및 호스트 메모리 맵 둘 모두에 나타난다. HDOUTRDY 및 HDOUTACK는 각각 호스트 메모리 맵 및 CPU(122) 메모리 맵에만 나타난다.

CPU(122)는 HINT 명령 레지스터에 기록함으로써 HDOUT에 새로운 데이터의 존재를 나타낼 수 있고, 그에 따라 호스트 인터럽트를 야기한다. MEC의 HINT- 출력 명령 에러를 검출하는 그것의 메모리 맵 또는 MEC의 HINT 명령 레지스터에 기록하는 CPU(122)에 응답하여 (로우)로 주장될 것이다. 그러나, 호스트 프로세서는 MEC의 호스트 인터페이스 메모리 맵의 HINTEN 레지스터에 "1"을 기록함으로써 HINT-를 먼저 주장하여야 한다. 반대로, HINT-의 주장은 HINTEN에 "0"을 기록함으로써 디스에이블된다. HINT-가 주장되면, HINT-는 호스트가 호스트 메모리 맵의 CLRHINT 레지스터에 어떤 값을 기록하기까지 주장된 채로 남아 있다.

호스트 프로세서는 MEC 호스트 메모리 맵의 RESET 레지스터에 임의의 값을 기록함으로써 전체 MEC(20)을 리셋할 수 있다. MEC(20)의 레지스터는 각각 테이블 11 및 13의 호스트 및 CPU 메모리 맵에 규정된 바와 같이 그들의 사전 결정된 초기상태를 가정한다. 호스트 초기화 리셋의 수신후, CPU(122)는 디버그 동작 모드로 들어가고 CPU(122) 프로그램 수행은 위치 0으로 점프한다.

7. 보조 인터페이스 버스 및 토큰 입력 버퍼

보조 인터페이스 버스(281)는 MEC(20)가 이미지 압축 코프로세서와 통신하는 것을 허용하고 이미지 압축 코프로세서(45)가 MEC(20)상의 명령을 수행하도록 하며 그 결과를 수신한다. MEC보조 인터페이스 버스는 테이블 14에 도시한 바와 같이 1 클럭, 24 다중화된 어드레스/데이터, 및 8 제어 신호 핀을 포함하고 이미지 압축 코프로세서 보조 버스와 핀-호환가능하다.

MEC(20)은 테이블(15)에 도시한 op코드를 사용하여 이미지 압축 코프로세서(45) 외부 명령을 처리한다. 본 발명의 바람직한 실시예에서, 3개의 다른 명령은 두 명령이 없는 op코드의 비트들이 동일한 동안은 MEC(20)상에서 동시에 수행될 수 이다.

MEC(20)상 명령의 수행은 "보조 프로세서 기록 사이클"을 MEC(20)에서 수행하는 이미지 압축 코프로세서(45)를 사용하여 시작하고 명령이 수행하기를 마치면 " 보조 프로세서 기록 사이클"과 함께 종료된다. 이미지 압축 코프로세서(45) 외부 명령은 MEC의 XPID(1:0) 핀의 상태와 명령 op코드의 최상부 두 비트를 매칭함으로써 MEC(20)상에 매핑된다. MEC(20)을 위해 정의된 외부 명령은 0 또는 1 입력 토큰을 가지고, 결과적으로 0 또는 1 출력으로 복귀될 것이다. 입력 및 출력 토큰은 데이터 또는 제어 토큰이고 데이터 토큰은 4 데이터 블록을 가지도록 가정된다.

보조 프로세서 기록 사이클 시간은 제 36 도에서 도시된다. 기록 사이클은 XAD(23:0)에 두 개의 24 비트 "프로세서 패킷" 워드의 첫 번째 주장과 관련하여 양쪽 XADS- 및 XWE - 로우를 주장하는 이미지 압축 코프로세서(45)를 사용하여 시간 T_0에서 시작한다. 프로세서 패킷 워드 0 및 1은 테이블 16에서 정의된다. 이미지 압축 코프로세서(45)는 믿을수 있는 데이터 전송을 보증하기 위하여 기록 사이클을 시작하기 전에 적어도 하나의 XCLK(850) 사이클을 주장한다. 프로세서 패킷 워드 0은 시간 T_1에서 MEC(20)(및 모든 다른 외부 프로세서)에 클럭되고, 이미지 압축 코프로세서(45)는 버스에 프로세서 패킷 워드 1을 주장하기 시작한다. 프로세서 패킷 워드 1은 프로세서 패킷 워드 0의 비트 1:0에 의해 선택된 MEC(20)가 XRDY에 인접하지 않고 - 시간 T_2에서 구동하고 시간 T_3보다 늦은 로우 하나의 클럭 사이클을 주장함으로써 응답할때까지 버스상에 주장된다. 이미지 압축 코프로세서(45)는 시간 T4에서 XCLK(850)의 다음 상승 모서리상의 신호인 주장된 XRDY 신호를 샘플한다.

프로세서 패킷 워드 0을 수신한후, 보조 인터페이스 유니트(281)는 포로세서 패킷 워드 1을 따르는 어떤 오퍼랜드 토큰을 수신하기 위해 준비하는 프로세서 패킷 워드 0의 비트16:14 및 비트 11을 검사한다. 만약 비트 11이 "0"이면, 오퍼랜드 토큰은 전송되지 않고, 기록 사이클을 보조 인터페이스 버스 및 MEC(20) 비주장 XRDY(제 36 도에 도시되지 않음)와 인접하는 이미지 압축 코프로세서(45)를 사용하여 시간 T_4에서 끝난다. 그럼에도 불구하고, 이미지 압축 코프로세서(45)는 4개의 24 비트 워드에서 버스상에 96 비트 토큰 기술자를 보냄으로써 데이터 토큰 또는 오퍼랜드 제어의 전송을 시작한다. 만약 프로세서 패킷 워드 0의 비트 16이 "0"이면, 오퍼랜드는 제어 토큰이고, 전송은 제 36 도의 시간 T5에서 끝난다. 그럼에도 불구하고, 토큰 기술자 데이터의 마지막 워드는 프로세서 패킷 워드 0의 비트 15:14에 의해 지시된 화소 데이터의 128 워드를 따르고, 전송은 타임 T6에서 끝난다. 오퍼랜드 토큰을 위한 버스 전송 형식은 테이블(17)에서 도시된다. 또한 시간 T6에서, MEC는 XRDY를 주장하지 않고, 시간 T7에서 하나의 클럭 사이클은 그것을 인접시킨다. 이미지 압축 코프로세서(45)는 마지막 워드가 MEC(20)에 클럭된후 적어도 하나의 XCLK(850) 기간까지 다른 프로세서 기록(또는 판독) 사이클을 시작하지 않는다.

명령의 수행을 완료하기 위하여, MEC(20)는 시간 T_0에서 XRQST 핀을 주장함으로써 이미지 압축 코프로세서(45)로부터 판독 사이클을 요구한다. 이미지 압축 코프로세서(45)는 동시에 XADS-로우, XWE-하이를 동시에 세트하고, 하나의 XCLK 850 사이클 동안 시간 T_1에서 CPU(122) ID를 XAD(1:0)에 주장함으로써 요구를 알린다. 시간 T_2에서, MEC(20)는 XADS-, XWE-, 및 XAD(23:0)를 샘플하고, 이미지 압축 코프로세서(45)는 이들 신호를 주장하지않고, XAD(23:0) 출력 버퍼를 3-상태로 하고 XOE-를 주장함으로써 XAD(23:0)에 MEC(20)를 인에이블한다.

이 시기에, 이미지 압축 코프로세서(45)는 XRDY를 주장하기 위하여 MEC(20)를 기다린다. 시간 T2에서 시작하여 MEC(20)는 XID(1:0) 핀상에 자신의 ID와 XAD(1:0)으로부터 판독된 ID를 비교하다. 매치를 가정하여, 이미지 압축 코프로세서(45)는 XRDY를 3-상태로 하지 않고, 시간 T3에서 하이구동하고 시간 T4에서 로우 구동한다 ; 이미지 압축 코프로세서(45)는 시간 T4에 XRQST를 주장하지 않는다. MEC(20)는 시간 T5에서 데이터를 전송하기 시작한다.

초기에, 시간 T5에서, MEC(20)는 프로세서 패킷 워드(0)을 버스에 주장하고 패킷 프로세서 워드(0)는 다음 상승 클럭 모서리상의 이미지 압축 코프로세서(45)에 클럭되고, 프로세서 패킷 워드 1은 주장된다. 비트 16:14의 가능한 예외를 제외하고, 프로세서 패킷 워드 0은 명령이 시작한 프로세서 기록 사이클 동안 보내진 프로세서 패킷 워드 0의 복사본이다. 만약 필요하다면 비트 16:14는 결과 토큰을 기술하기 위하여 변조되고, 이미지 압축 코프로세서(45)로 되돌려 보내진다. 프로세서 패킷 워드 1은 프로세서 기록 사이클동안 보내진 프로세서 패킷 워드로부터 복사한다.

만약 프로세서 패킷 워드 0의 비트 13:12가 양쪽다 "0"이면, 명령은 결과 토큰을 가지지 않고, 프로세서 판독 사이클은 XRDY를 주장하지 않는 MEC(20) 및 XOE(도시되지 않음)를 주장하지 않는 이미지 압축프로세서(45)를 사용하여 시간 T6에서 끝낸다.

만약 프로세서 패킷 워드 0의 비트 13:12의 어느것도 "0"이 아니면, 결과 토큰을 존재하고 ; 프로세서 패킷 워드 0의 비트 16은 결과 토큰의 형태를 결정하기 위하여 사용된다. 만약 비트 16이 "0"이면, 결과는 XCLK(850)의 다음 4개의 기간에 걸쳐 이미지 압축 코프로세서(45)에 전달되는 제어 토큰이고, 판독 사이클은 시간 T7에서 끝난다. 만약 비트 16이 "1"이면, 결과는 다음 128 XCLK(850) 기간에 걸쳐 이미지 압축 코프로세서(45)에 전달되는 4개의 블록 데이터 토큰이라고 가장되고, 시간 T8에서 끝난다. 기술자 및 데이터 워드는 테이블 17에서 도시된 버스상에 포맷된다. 또한 시간 T8에서, MEC(20)는 XRDY를 주장하지 않고 이미지 압축 코프로세서(45)는 XOE를 주장하지 않는다. 시간 T9에서 하나의 클럭 사이클후에, MEC(20)는 XRDY를 3-상태로 만든다.

MEC(20)는 하드웨어 및 소프트 웨어 양쪽 명령을 조절한다. "하드웨어 에러 조절은 제공된 논리를 사용하

여 행해지는 반면, "소프트 웨어" 에러 조절을 CPU에서 동작하는 프로그램에 의해 행해진다. MEC에서 영구 전자회로의 논리는 테이블 18에서 도시된 전체 에러 상태를 위한 이미지 압축 코프로세서(45)로부터 프로세서 패킷 워드 0을 검사한다. 만약 에러가 검출되면, MEC(20)는 상태 장치의 모두를 디스에이블링하고, 에러 레지스터를 "1"로 세팅함으로써 "멈춘다". 상기 레지스터는 호스트 판독 가능하고 호스트(38) 메모리 맵에서 나타낸다. 만약 호스트(38)가 HINTEN 레지스터를 경유하여 MEC(20) 인터럽트를 생성하면, 하드웨어 에러는 MEC의 HINT 핀상에 인터럽트 출력을 유발한다.

MPEG 또는 H.261 인코더 알고리듬을 수행하는 이미지 압축 코프로세서 프로그램은 모션 벡터를 포함하는 제어 토큰 또는 발광 화소의 4개의 8×8 블록을 포함하는 데이터 토큰을 MEC의 CPU(122)상에서 동작하는 모션 평가 알고리듬에 의해 사용하기 위한 MEC(20)에 보내기 위하여 IPREDICT 명령을 사용한다. IPREDICT는 그것의 끝냄을 신호하기 위하여 임의적으로 제어 토큰을 출력할 수 있다.

MEC(20)는 IPREDICT 명령에서 파라미터 필드를 미리 한정하지 않는다. 상기 필드는 MEC 프로그램에 의해 번역되어 남는다. 예를들어, 파라미터 및/또는 IPREDICT 명령에서 한정되지 않는 OPCODE 비트는 수행(순방향, 역방향, 또는 양방향)되는 예상 형태를 선택하기 위하여 사용될수 있다.

IPREDICT 명령을 통하여 이미지 압축 코프로세서(45)로부터 수신된 토큰 화소는 그것이 입력 토큰 버퍼(270)에 진입할 때 최고 해상도 화소를 필터링하는 4:1 데시메이션에 의해 형성된 입력 토큰 더하기 다른 64 4분 해상도 화소로부터 토큰 화소 입력 버퍼(270)의 최고 해상도 화소의 쌍에 저장된다. ITPB에서 4분 해상도 화소는 계층적 모션 평가 서치동안 8×8 블록 비교에 사용된다. 토큰 입력 버퍼(270)에 의해 형성된 이중 버퍼링 메모리 인터페이스는 이미 입력 토큰이 블록 매칭에 사용되는 동안 MEC(20)가 이미지 압축 코프로세서(45)로부터 새로운 IPREDICT 오퍼랜드 토큰을 수신하게한다.

제 38 도는 IPREDICT 메모리 인터페이스를 도시한다. 상기 인터페이스는 레지스터의 수집 및 토큰 화소, 토큰 기술자, 및 프로세서 패킷 워드를 저장하는 레지스터 및 메모리 콜렉션을 포함한다. 상태가 SWAP 명령 레지스터(899)에 의해 토글(toggle)된 플립 플롭(888)은 토큰 입력 버퍼(270)가 보조 인터페이스 유니트(281)로부터 회소를 수신하고 토큰 입력 버퍼(270)가 화소를 블록 매칭 프로세서(250)에 출력하는 것을 결정하기 위하여 데이터 스위치(900)을 제어한다. 96 비트 토큰 기술자 및 버스로부터 수시된 두 개의 24 비트 프로세서 패킷 워드는 IPREDICT 토큰 기술자 버퍼(905) 및 IPREDICT 프로세서 패킷 버퍼(910)로 불리는 두 세트의 레지스터에 각각 저장된다. 버퍼(905 및 910)은 CPU(122)에 의해 직접적으로 액세스될수 없다. 대신, CPU(122)으로부터 SWAP 명령은 버퍼(905 및 910)에서 132개의 144 비트 내용이 CPU(122)에 의해 판독될수 있는 "교환" 레지스터(920) 및 922의 두 세트에 복사되게한다. 상기 복사 동작은 하나의 PCLK 사이클에서 종료된다. 토큰 기술자 및 프로세서 패킷 교환 레지스터는 각각 테이블(19 및 20)에서 기술된다. CPU(122)는 이들 레지스터에 기록하지 않는다.

처음에, IPREDICT 명령의 프로세서 기록 사이클동안, MEC(20)는 버퍼(910)에서 프로세서 패킷 워드 0 및 프로세서 패킷 워드 1을 저장한다. 만약 프로세서 패킷 워드 0의 비트 11이 "1"이면, MEC(20)는 버퍼(905)에서 입력 오퍼랜드로부터 기술자를 저장한다. 마지막으로, 만약 프로세서 패킷 워드 0의 비트 16이 "1"이면, 오퍼랜드는 데이터 토큰이고, MEC(20)는 입력 데이터를 수신하기 위하여 토큰 입력 버퍼(270) 세트에 320 화소(보조 버스로부터 256 및 필터된 다른 64)를 저장한다. MEC(20)는 ITBRDY 상태 비트를 "1"로 지정한다. 에러 조건은 만약 프로세서 패킷 워드 0의 비트 11이 "0"(오퍼랜드 없음) 또는 만약 비트 16가 "1" 및 프로세서 패킷 워드 0의 비트 15:14 모두가 "1"이 아니면 존재한다(오퍼랜드는 4개의 블록보다 작은 데이터 토큰이다.)

일단 CPU(122)가 이미 IPREDICT 입력 토큰 처리를 끝냈으면, CPU(122)는 ITBRDY 상태 비트가 "1"일때까지 기다리고 SWAP 명령 레지스터에 기록한다. 결과적으로 버퍼(905 및 910)의 내용은 그것의 대응 교환 레지스터(920 및 922)에 복사된다. 토큰 입력 버퍼(270)는 스위칭되어, CPU(122) 및 블록 매칭 프로세서(250)에 의해 액세스될 새로운 입력 토큰을 허용한다. 그러나, 교환은 ITBRDY 상태 비트가 "0"으로 설정되게 하고 MEC(20)가 이미지 압축 코프로세서(945)로부터 프로세서 판독 사이클을 요구함으로써 IPREDICT 명령을 마치도록한다.

만약 버퍼(910)에 저장된 프로세서 패킷 워드 0의 각각의 비트 13:12가 "0"이면, 프로세서 패킷 워드는 판독 사이클이 버퍼(910)로부터 직접적으로 복사되는 동안 MEC(20)에 의해 이미지 압축 코프로세서(45)에 다시 보내진다. 그러나, 만약 비트 13:12의 어느것이 "0"이 아니면, IPREDICT 명령은 목적지를 가지며 이미지 압축 코프로세서(45)로 되돌려 보내진 프로 세서 패킷 워드 0의 비트 16은 출력 제어 토큰을 가리키기 위하여 "0"이 된다. 상기 경우에, MEC(20)는 제 1 기술자 워드의 최상위 비트가 제어 토큰을 가리키기 위하여 "0"이 되는 것을 제외하고 버퍼(905)에복사된 4개의 24 비트 토큰 기술자 워드를 가지는 프로세서 패킷 워드 1의 버스 전송을 따른다.

모션 평가로부터 발생하는 예측은 단지 동작 벡터만 포함하는 제어 토큰이나 양쪽 예측된 화소 및 동작 벡터를 포함하는 데이터 토큰으로서 OPREDICT 명령을 통하여 이미지 압축 코프로세서(45)에 되돌려 보내진다. OPREDICT는 오퍼랜드로서 제어 또는 데이터 토큰을 선택적으로 사용할수 있다. 그와같이 IPREDICT 명령 MEC(20)은 OPREDICT 명령에서 파라미터 필드를 미리 한정하지 않는다. CPU(122)는 테이블 21의 로드 구성 레지스터를 사용하는 OPREDICT 버퍼(265)를 로드할수 있다.

제 39 도는 OPREDICT 메모리 인터페이스의 블록 다이어그램이다. IPREDICT와 다르게, OPREDICT 명령을 지지하기 위한 MEC(20) 메모리 인터페이스는 이중 버퍼되지 않는다. 명령의 시작에서 버스로부터 수시된 두 개의 24 비트 프로세서 패킷 워드는 OPREDICT 프로세서 패킷 버퍼(950)에서 저장된다. 명령의 단에서 이미지 압축 코프로세서(45)에 전송되는 결과 토큰의 기술자 및 화소 데이터 부분은 각각 OPREDICT 토큰 기술자 버퍼(906) 및 토큰 출력 화소 버퍼(265)에 저장된다. CPU(122)의 방향하에서, 버퍼(960)는 그것이 블록 보간법 프로세서(19)를 통하여 통과된후 서치 메모리(60)로부터 화소가 기록된다. CPU(122)는 각각 테이블 21 및 22에서 기술된 레지스터를 통하여 버퍼(950)에 기록하고 버퍼(960)로부터 판독한다.

OPREDICT 명령을 시작하는 프로세서 기록 사이클동안, MEC(20)는 버퍼(950)에서 프로세서 패킷 워드 0 및 프로세서 패킷 워드 1을 저장하고 ORQST 상태 비트를 "1"로 설정한다. CPU(122)는 테이블 22에서 기술된 레지스터를 통하여 버퍼(950)를 자유로이 판독할수있다. MEC(20)는 만약 OPREDICT 명령이 목적지를 가지

지 않으면 에러를 검출한다 ; 즉, 프로세서 패킷 워드 0의 비트 12 또는 13의 어느것도 "1"이 아니다. 만약 프로세서 패킷 워드 0의 비트 11이 "1"이면, MEC(20)는 이미지 압축 코프로세서(45)로 부터 오퍼랜드를 판독하고 상기 오퍼랜드를 버린다. 만약 프로세서 패킷 워드 0의 비트 16이 "0"이 아니면 오퍼랜드는 그것의 길이가 프로세서 패킷 워드 0의 비트 15:14에 의해 지시된 데이터 토큰이다.

CPU(122)는 이미지 압축 코프로세서(45)로 돌아가는 결과 토큰을 보낼 때, CPU(122)는 어떤 사전 결과가 전송되는 것을 보장하도록 OTBRDY 상태 비트가 "1"일때까지 기다린다. CPU(122)는 OTBY 레지스터에 기록함으로써 버퍼(960)를 임의적으로 로드한다. OTBY에 기록하는 것은 LDOTBRDY 상태 비트가 "0"으로 지정되도록 한다. 버퍼(960)가 충전되는 동안, CPU는 테이블 23에서 목록된 레지스터를 경유하여 결과 기술자를 역시 기록할 수 있다. 일단 CPU(122)가 결과 기술자를 기록하는 것을 끝마치면, CPU(122)는 OPREDICT 명령의 수행을 마치기 위하여 SEND 명령 레지스터를 기록한다.

SEND 명령에 응답하여, MEC(20)는 OTBRDY를 "0"에 지정하고 양쪽 ORQST 및 LDOTBRDY 가 "1"로 지정될 때까지 기다린다. 나중 신호는 버퍼(960)가 로딩을 끝내는 것을 가리킨다. MEC(20)는 ORQST 상태 비트를 "0"으로 지정하고 이미지 압축 코프로세서(45)로부터 프로세서 판독 사이클을 요구한다. 판독 사이클동안, MEC(20)는 프로세서 패킷 워드 0의 비트 16이 OTD로부터 비트 11(토큰 기술자 형태 비트)과 동일하게 되고 프로세서 패킷 워드의 비트 15:14가 "11"로 되는 것을 제외하고 버퍼(950)로부터 프로세서 패킷 워드 0 및 프로세서 패킷 워드 1의 이미지 압축 코프로세서(45) 복사본을 보낸다.

MEC(20)는 기술자의 비트 93:92가 "11"로 되는 것을 제외하고 OPREDICT 토큰 기술자 레지스터로부터 복사된 결과 토큰 기술자를 가지는 프로세서 패킷 워드 1의 전송을 따른다. 만약 OTD의 비트 11이 "1"이면, MEC(20)는 버퍼(960)로부터 복사된 토큰 데이터 256 화소를 가지는 버스상의 토큰 기술자를 따른다. MEC(20)는 판독 사이클의 결말에서 OTBRDY 상태 비트를 "1"로 지정한다.

이미지 압축 코프로세서(45) 프로그램은 휘도 화소의 4개의 8×8 블록을 포함하는 토큰을 예측 메모리에 기록하기 위한 MEC(20)에 보내기 위하여 통상적으로 WRMEM 명령을 사용한다. 상기 명령은 그것이 모션 평가 서치를 위해 추후에 사용되도록 재구성된 인코더 입력 이미지를 예측 메모리에 기록하기 위하여 일반적으로 사용된다. 만약 WRMEM 명령이 이미지 압축 코프로세서(45) 프로그램에서 목적지를 가진다면, 그것은 제어 토큰올 출력하고, 그렇지 않으면 결과를 생산하지 않는다. WRMEM은 미리 한정된 파라미터 필드를 가진다. 이들 필드는 이미지 압축 코프로세서(45) 비디오 메모리 기록 명령에 의해 사용된 것과 유사하다.

제 40 도는 WRMEM 메모리 인터페이스의 블록 다이어그램이다. OPREDICT와 같이, WRMEM 명령을 유지하기 위한 MEC 메모리 인터페이스는 이중 버퍼되지 않는다. 제 40 도에서 도시된 것처럼, 명령의 시작에서 보조 인터페이스 버스(281)로부터 수신된 두 개의 24 비트 프로세서 패킷 워드는 WRMEM 프로세서 패킷 버퍼(980)에 저장된다. MEC의 비디오 버스에 메모리에 기록되는 오퍼랜드 데이터 토큰의 기술자 및 블록 데이터 부분은 각각 WRMEM 토큰 기술자 버퍼(985) 및 토큰 화소 버퍼(175)에 저장된다. 양쪽 IPREDTCT 및 OPREDTCT 명령과 다르게, CPU(122)는 WRMEM 명령을 실행시키지 않는다. MEC(20) 보조 인터페이스 유니트(281) 및 비디오 인터페이스 유니트(185)는 분명히 WRMEM 명령을 실행한다.

WRMEM 명령의 프로세서 기록 사이클동안, MEC(20)는 버퍼(980)에서 프로세서 패킷 워드 0 및 프로세서 패킷 워드 1을 저장한다. 만약 프로세서 패킷 워드 0의 비트 11이 "1"이면, MEC(20)는 버퍼(985)에서 입력 오퍼랜드로부터 기술자를 역시 저장한다. 마지막으로, 만약 프로세서 패킷 워드 0의 비트 16이 "1"이면, 오퍼랜드는 데이터 토큰이고, MEC(20)는 버퍼(175)에 256 화소를 저장한다. 에러 조건은 만약 프로세서 패킷 워드 0의 비트 11이 "0"이거나 프로세서 패킷 워드 0의 비트 16:14가 모두 "1"로 지정되지 않으면 존재한다.

프로세서 기록 사이클의 결말에서, 보조 인터페이스 유니트(281)는 버퍼(980)로부터 파라미터 및 버퍼(985)로부터 hpos와 vpos 기술자 필드를 사용하는 16×16 블록으로서 버퍼(175)의 화소를 비디오 메모리에 기록하는 비디오 인터페이스 유니트(185)에 통보한다. 비디오 인터페이스 유니트(185)는 그것이 끝났을 때 보조 인터 페이스 유니트(281)에 통보하고, 보조 인터페이스 유니트(281)는 이미지 압축 코프로세서(45)로부터 프로세서 판독 사이클을 요구함으로써 MRMEM 명령을 끝낸다.

만약 버퍼(980)에 저장된 프로세서 패킷 워드 0의 각각의 비트 13:12가 "0"이면, 프로세서 패킷 워드는 판독 사이클이 버퍼(980)으로부터 직접적으로 복사되는 동안 MEC(20)에 의해 이미지 압축 코프로세서(45)로 되돌려 보내진다. 그러나, 만약 비트 13:12의 어느것이 "0"이 아니면, WRMEM 명령은 목적지를 가지고, 이미지 압축 코프로세서(45)에 되돌려 보내진 프로세서 패킷워드 0의 비트 16은 출력 제어 토큰을 가리키도록 "0"이 된다. 상기 경우에, MEC(20)는 제 1 기술자 워드의 최상위 비트가 제어 토큰을 가리키도록 "0"이 되는 것을 제외하고 버퍼(985)로부터 복사된 4개의 24 비트 토큰 기술자 워드를 가지는 프로세서 패킷 워드 0 및 프로세서 패킷 워드 1의 전송에 따른다.

비록 본 발명이 바람직한 실시예 및 어떤 변화 및 변형에 의해 기술 되었지만, 다른 변화 및 변형은 사용될수 있고, 본 발명은 다음 청구 범위에 의해 정의된다.

테이블 1 : 비디오 버스 인터페이스 신호

VA(10 : 0)

비디오 메모리 어드레스(3-상태 출력) - 액세스된 메모리 행 또는 열 어드레스를 가리킨다 ; DRAM 또는 VRAM 상의 어드레스 핀에 직접 연결될수 있다. 이들 핀은 VDIS 핀이 주장되거나 MEC가 비디오 버스(버스 중재 섹션 4.8에서 논의되었다)을 처리하지 않을때마다 높은 임피더스 상태에 있다.

VMSEL(2 : 0)

비디오 메모리 선택(3-상태 출력) - 화소가 액세스 되고 MEC가 비디오 버스상 메모리를 리프레시할 때 또한 지시되는 메모리 "뱅크"를 선택한다. 만약 필요하다면, VMSEL(2:0)은 제어 신호를 제한하기 위하여 버스상 각 메모리에 의해 사용된다. 이들 핀응 VDIS 핀이 주장되거나 MEC가 비디오 버스를 점유하지 않을때

마다 높은 임피던스 상태에 있다.

VD0(7 : 0)

비디오 메모리 데이터 버스 0(입력/출력) - 16비트 및 32 비트 액세스 양쪽을 위하여 화소를 전송하기 위하여 사용된다 ; DRAM 도는 VRAM 데이터 핀에 직접적으로 접속된다. 이들 핀상의 출력 버퍼는 MEC가 비디오 데이터를 판독하고, VDIS 핀이 주장되거나, MEC가 비디오 버스를 점유하지 않을때마다 높은 임피던스 상태에 있다.

VD1(7 : 0)

비디오 메모리 데이터 버스 1(입력/출력) - 16비트 및 32 비트 액세스 양쪽을 위하여 화소를 전송하기 위하여 사용된다 ; DRAM 도는 VRAM 데이터 핀에 직접적으로 접속될 수 있다. 이들 핀상의 출력 버퍼는 MEC가 비디오 데이터를 판독하고, VDIS 핀이 주장되거나, MEC가 비디오 버스를 점유하지 않을때마다 높은 임피던스 상태에 있다.

VD2(7 : 0)

비디오 메모리 데이터 버스 2(입력/출력) - 16비트 및 32 비트 액세스 모드 양쪽을 위하여 화소를 전송하기 위하여 사용된다 ; DRAM 또는 VRAM 데이터 핀에 직접적으로 접속될 수 있다. 이들 핀상의 출력 버퍼는 MEC가 비디오 데이터를 판독하고, VDIS 핀이 주장되거나, MEC가 비디오 버스를 점유하지 않을때마다 높은 임피던스 상태에 있다.

VD3(7 : 0)

비디오 메모리 데이터 버스 2(입력/출력) - 16비트 및 32 비트 액세스 양쪽을 위하여 화소를 전송하기 위하여 사용된다 ; DRAM 도는 VRAM 데이터 핀에 직접적으로 접속될 수 있다. 이들 핀상의 출력 버퍼는 MEC가 비디오 데이터를 판독하고, VDIS 핀이 주장되거나, MEC가 비디오 버스를 소요하지 않을때마다 높은 임피던스 상태에 있다.

VRAS-(3 : 0)

비디오 메모리 행 어드레스 스트로브(strobes) (3-상태 출력) - VRAS - (k)는 비디오 데이터 버스 VDK에 결합된 메모리를 위한 행 어드레스 스트로브이고 필요하다면 VMSEL(2:0)에 의해 제한된다 ; 그럼에도 불구하고, DRAM 또는 VRAM "RAS" 핀에 직접 연결될수 있다. 이들 핀은 VDIS 핀이 주장되거나 MEC가 비디오 버스를 점유하지 않을때마다 높은 임피던스 상태에 있다.

VCAS0 - (3 : 0)

비디오 메모리 열 어드레스 스트로브 출력(3-출력 상태) - VCAS0 - (k)는 비디오 데이터 버스 VDK에 결합된 메모리를 위한 열 어드레스 스트로브이다 ; 그것은 직접 DRAM 또는 VRAM "CAS" 핀에 직접 연결될수 있다. 이들 핀은 VDIS 핀이 주장되거나 MEC가 비디오 버스를 점유하지 않을때마다 높은 임피던스 상태에 있다.

VCAS1 - (3 : 0)

비디오 메모리 열 어드레스 스트로브 입력(입력) - VCASI - (k)는 비디오 데이터 버스 VDK에 결합된 메모리 칩의 CAS 핀들에 결합되어야 한다 ; 메모리 판독동안 비디오 데이터 버스 VDK로부터 데이터를 래치하기 위하여 사용된 MEC 내의 분명한 래치로서 동작한다.

VWE-

비디오 메모리 기록 인에이블(3-상태 출력) - 데이터 버스가 상이함에도 불구하고 모든 메모리를 위해 사용된다 ; DRAM 또는 VRAM "기록" 핀에 직접 연결될수 있다. 상기 핀은 VDIS 핀이 나타되거나 MEC가 비디오 버스를 점유하지 않을 때 마다 언제든지 높은 임피던스 상태에 있다.

VOE-

비디오 메모리 기록 인에이블(3-상태 출력) - 데이터 버스가 상주함에도 불구하고 모든 메모리를 위해 사용된다 ; DRAM 또는 VRAM "출력 인에이블" 핀에 직접 연결될수 있다. 상기 핀은 VDIS 핀이 주장되거나 MEC가 비디오 버스를 점유하지 않을 때 마다 언제든지 높은 임피던스 상태에 있다.

VDIS

비디오버스 입력 디스에이블(입력) - 신호가 주장될 때(높게), 비디오 버스 상의 3-상태(및 양방향성) 데이터, 어드레스, 및 제어 출력은 높은 임피던스 상태로 무조건적으로 놓인다.

VRQSTI

비디오 버스는 요구 입력(입력) - 버스 요구 데이지 체인용 입력 신호 ; 다음 낮은 우선권 버스 마스터(master)의 VRQST0에 연결된다. 버스 중재는 섹션 4.8에서 논의된다.

VGQST0

비디오 버스 요구 입력(출력) - 버스 요구 이미지 체인 출력 ; 다음 높은 우선권 버스 마스터의 VRQSTI에 연결된다.

VGRNTI

비디오 버스 승인 입력(입력) - 버스 승인 데이지 체인용 입력 신호 ; 다음 높은 우선권 버스 마스터의 VGRNT0에 연결된다.

VGRNTO

비디오 버스 승인 입력(출력) - 버스 승인 데이지 체인용 출력신호 ; 다음 보다 낮은 우선권 버스 마스터의 VGRNTI에 연결된다.

VCLK

비디오 버스 클럭(입력) - 비디오 버스상의 모든 동작을 동조화하기 위하여 사용된다.

테이블 2 : VIB 세그먼트 구성 레지스터

VkORIGIN(6 : 0) 세그먼트 k의 시작 페이지 어드레스(범위=[0, 127])

VkLENGTH(6 : 0) 세그먼트 k에서 시작 페이지 행의 수(범위=[1, 64])

VkWIDTH(4 : 0) 세그먼트 k에서 시작 페이지 열의 수(범위=[1, 16])

VkX(11 : 0)

세그먼트 k의 상부 왼쪽 모서리의 이미지 메모리에서 논리 시작 X(열) 어드레스 ; VkX(0)는 "0"으로 하드웨어화된다(즉 모든 페치는 이미지 메모리의 짝수 열에서 시작된다).

VkY(11 : 0)

세그먼트 k의 상부 왼쪽 모서리의 이미지 메모리에서 논리 시작 Y (행) 어드레스 ; VkY(0)는 "0"으로 하드웨어화된다(즉 모든 페치는 이미지 메모리의 짝수 열에서 시작된다).

VkPCNT(7 : 0)

최고의 해상도 데이터로 세그먼트 k를 채울 때 이미지 메모리의 각 행으로부터 판독되는 화소의 수, 또는 4분 해상도 데이터로 세그먼트 k를 채울 때 이미지 메모리의 각 행으로부터 판독되는 화소수의 1/2 ; VkPCNT(0)는 "0"으로 하드웨어화된다(즉 화소의 짝수는 항상 열로부터 래치된다), 그리고 VkPCNT는 8 * VKWIDTH보다 작거나 같아야 한다.

VkMSEL(2 : 0)

세그먼트 k의 패치동안 출력한다.

VkROW(5 : 0)

VIB(범위=[0, 63])의 세그먼트 k로부터 복사한 화소에 대한 서치 메모리의 세그먼트 k에서 시작 페이지 행수.

VkCOL(3 : 0)

VIB(범위=[0, 15])의 세그먼트 k로부터 복사한 화소에 대한 서치 메모리의 세그먼트 k에서 시작 페이지 행수.

테이블 3 : LDVIB 명령 레지스터의 정의

LDVIB(2k+1:2k)	정의
00	세그먼트 k를 로딩하지 마시오
01	최고 해상도 화소를 가지는 세그먼트 k만 로딩
10	4분 해상도 화소를 가지는 세그먼트 k만 로딩
11	최고 해상도 화소를 가지는 세그먼트 k 및 최고 해상도 화소로부터 유도된 4분 해상도 화소를 가지는 세그먼트 k+1을 로딩

테이블 4 : 비디오 인터페이스 시간 제어 레지스터

레지스터	설명	심볼	용도
VCNT1(4 : 0)	RAS - 능동 지연에 대한 행 어드레스	Tasr	N
VCNT2(4 : 0)	열 어드레스 지연에 대한 행 어드레스	TASR+Trah	N
VCNT3(4 : 0)	CAS - 능동 지연에 대한 RAS 능동	Trod	N
VCNT4(4 : 0)	CAS - 사이클 기간	Tcp+Tcas	N
VCNT5(4 : 0)	RAS - 사전 충전 기간	Trp	N,R
VCNT8(4 : 0)	CAS - 펄스 폭	Tcas	N
VCNT9(4 : 0)	CAS - RAS 수동 지연에 대한 능동	Trsh	N
VCNT10(4 : 0)	CAS - RAS 수동 지연에 대한 능동	Tcsr	R
VCNT11(4 : 0)	RAS - CAS 수동 지연에 대한 능동	Tchr	R
VCNT12(4 : 0)	RAS 수동 지연에 대한 능동	Tras	R

제한 : VCNT2는 VCNT1 보다 커야 한다 ;

VCNT1 + VCNT3는 VCNT2 보다 커야 한다 ;

VCNTx는 항상 0보다 커야한다.

레지스터 용도 : N=정상 페이지(??) 판독 및 기록 사이클

R=리프레시 사이클

테이블 5-1 : MAD 트래킹 레지스터

MADMS(10 : 0) : 현재 최소 MAD의 비트 15:5

MADLS(4 : 0) : 현재 최소 MAD의 비트 4:0

MADX(7 : 0) : 현재 최소 MAD의 상대 X 좌표

MADY(7 : 0) : 현재 최소 MAD의 상대 Y 좌표

테이블 5-2 : 블록 매칭 수행

BMP 모드	보간법	부산물
00	적용하지 않음	5
01	적용하지 않음	6
10	없음	18
10	수평 또는 수직	27
11	없음	20
11	수평, 또는 수직	30

테이블 6 : 조사 패턴 구성 레지스터

SPX(7 : 0) : 제 1 서치 좌표 트리플리트(triplet)의 상대 X 좌표.

SPY(7 : 0) : 제 1 서치 좌표 트리플리트의 상대 Y 좌표.

네츳(7 : 0) : 서치될 트리플리트의 총수보다 작음

SPINC(4 : 0) : 하나의 트리플리트로부터 다음 트리플티르로 수평(X) 증가(비트 1:0 사이의 이진 포인트를 가지는 표시되지 않은 아주 작은 양의 수).

SPMASK(2 : 0) : 최소 MAD의 계산에서 사용하기 위한 각 트리플리트로 부터 MAD를 선택한다 ; 만약 SPMASK(n) = 1이면, 각 트리플리트의 n번째 서치 좌표로부터 MAD는 최소 MAD 계산으로 분해되고, 여기서 좌표 0은 트리플리에서 가장 큰 좌표로 정의된다.

테이블 7 : 블록 매칭 구성 레지스터

BMSEG(1 : 0) 서치된(0, 1, 2 또는 3) 검색 메모리 세그먼트

BMXORG(6 : 0) 세크먼트 BMSEG에서 서치 원점의 절대 X 좌표.

BMYORG(6 : 0) 세크먼트 BMSEG에서 서치 원점의 절대 Y 좌표.

BMPMODE(1 : 0) 블록 매칭 프로세서 동작 모드

00 = 연속 행의 8×8 블록 매치

01 = 선택 행의 8×8 블록 매치

10 = 연속 행의 16×16 블록 매치

11 = 교번 행의 16×16 블록 매치

테이블 8 : 보간법을 위해 필요한 블록 크기

xisb	visb	보간법	보간법 블록 크기
0	0	없음	적용하지 않음
0	1	수직	17 행 ×16 열
1	0	수평	16 행 ×17 열
1	1	수평 및 수직	17 행 ×17 열

테이블 9 : 디버그 어드레스 매핑

RADDR 또는 WADDR(이진수에서)	선택된 CPU 레지스터 또는 메모리
0 XXXX XXXX XXXX	CPU 어드레스 "XXX"(hex)에서 레지스터 또는 메모리

1 000X XXX0 0000	레지스터 파일에서 R0
1 000X XXX0 0001	레지스터 파일에서 R1
1 000X XXX0 0010	레지스터 파일에서 R2
. . .	.
1 000X XXX1 1111	레지스터 파일에서 R31
1 001X XXXX XXXX	nPC(액세스만 판독)
1 010X XXXX XXXX	PC(액세스만 판독)
1 011X XXXX XXXX	저장
. . .	.
1 111X XXXX XXXX	저장

테이블 10 : CPU 명령 세트

명령		정의	Opcode
ADD	Rc, Ra, rb	Ro=Ra+Rb	00
ADDI	Rc, Ra, 상수	Rc=Ra+상수	01
SUB	Rc, Ra, Rb	Rc=Ra-Rb	02
SUBI	Rc, Ra, 상수	Rc=Ra-상수	03
AND	Rc, Ra, Rb	Rc=Ra & Rb	04
ANDI	Rc, Ra, 상수	Rc=Ra & 상수	05
OR	Rc, Ra, Rb	Rc=Ra & Rb	08
ORI	Rc, Ra, 상수	Rc=Ra & 상수	09
XOR	Rc, Ra, Rb	Rc=Ra & Rb	06
XORI	Rc, Ra, 상수	Rc=Ra & 상수	07
NOR	Rc, Ra, Rb	Rc= ~(Ra, Rb)	0A
NORI	Rc, Ra, 상수	Rc= ~(Ra, 상수)	0B
SRA	Rc, Ra, 상수	Rc= Ra〉〉상수(산술 우측 시프트)	0C
SRL	Rc, Ra, 상수	Rc= Ra〉〉상수(논리 우측 시프트)	0D
SLL	Rc, Ra, 상수	Rc= Ra〈〈상수	0E
BEQAL	Ra, Rb, 어드레스	Rc에서 무조건 저장(nPC+1)	10
	Rc	만약 Ra+Rb이면 브랜치에서 어드레스로	
BNEAL	Ra, Rb, 어드레스	Rc에서 무조건 저장(nPC+1)	11
	Rc	만약 Ra !=Rb이면 브랜치에서 어드레스로	
BGTAL	Ra, Rb, 어드레스	Rc에서 무조건 저장(nPC+1)	13
		만약 Ra〉Rb이면 브랜치에서 어드레스로	
BGEAL	Ra, Rb, 어드레스	Rc에서 무조건 저장(nPC+1)	15
	Rc	만약 Ra〉Rb이면 브랜치에서 어드레스로	
JALR	(Ra), Rc	Ro에서 저장(nPC+1) 및 (Ra)로 점프	16
ST	(Ra+상수), Rb	Rb에 위치(Ra+상수)로 저장	19
LD	Rc, (Ra+상수)	위치(Ra+상수)에서 Rc로 로딩	1B

테이블 11 : MEC 호스트 메모리 맵

어드레서	내용	형태	리셋	비트	해설
CPU 프로그램 메모리					
000	PGM(0)(31 : 16)	RW	없음	15:0	명령 0, msw
001	PGM(0)(15 : 0)	RW	없음	7.2 15:0	명령 0, lsw
002	PGM(1)(31 : 16)	RW	없음	15:0	명령 1, msw
003	PGM(1)(15 : 0)	RW	없음	15:0	명령 1, lsw

★ ★ ★ ★ ★ ★

3FE PGM(511) RW 없음 15:0 명령 511, msw

3FF PGM(511) RW 없음 15:0 명령 511, lsw

400 프로그램 메모리 확장을 위한 비축

★

5FF 프로그램 메모리 확장을 위한 비축

호스트 인터페이스 레지스터

600	HDIN(11 : 0)	W	없음	11:0	
601	HDOUT(11 : 0)	R	없음	11:0	0으로 판독된 비트 15:12
602	HDINACK	R			0으로 판독된 비트 15:1
603	HDOUTRDY	R			0으로 판독된 비트 15:1
610	에러	R			0으로 판독된 비트 15:1
611	HITEN	W			0
612	CLRHINT	W			무시된 데이터
613	리셋	W			무시된 데이터
620	RADDR(12 : 0)	W	없음	12:0	
621	DBDAT(11 : 0)	RW	없음	11:0	0으로 판독된 비트 15:2
622	WADDR(12 : 0)	W	없음	12:0	
623	DBRDY	R			0으로 판독된 비트 15:1
624	단계	W			무시된 데이터
625	RUN	W			무시된 데이터
626	HALT	W			무시된 데이터; CPU는 PRST에 응답하는 것을 중지한다.
630	BLENGTH	RW			판독상의 정의되지 않은 비트 14:7

테이블에서 열은 아래와 같이 정의된다.

어드레스

hex에서 어드레스

내용

MEC 메모리 위치 또는 레지스터 이름.

형태 :

만약 있다면, 호스트 판독(R), 호스트 기록(W), 또는 호스 판독/기록(RW) 리셋 상태 : "없음"은 질문의 어드레스에서 레지스터가 미리 정의된 초기 상태를 가지지 않고 현재 상태는 리셋에 의해 변화하지 않는다는 것을 가리킨다 ; "ha"는 리셋이 호스트 "명령"에 대응하고 레지스터는 대응하지 않으므로 어드레스에 적용할 수 없는 것을 가리킨다. 그럼에도 불구하고 지시되지 않으면, 레지스터는 PRST - 핀을 주장함으로써 표시된 "하드웨어" 리셋이나 리셋에 대한 호스트 기록에 의해 표시된 "소프트웨어" 리셋을 경유하여 리셋된다. 그것의 리셋 열에서 ★★★로 표시된 레지스터는 하드웨어 리셋에 응답하여 리셋되지만 소프트웨어 리셋에 응답하여 변하지 않는다(미리 정의된 초기 상태를 취하지 않거나).

비트

특정 어드레스에 정의된 데이터 비트.

설명

갖가지의 추가 정보.

테이블 12 : 호스트 인터페이스 버스 신호

신호 이름 정의

HCLK 호스트 인터페이스 클럭(입력) - 모든 버스 어드레스, 데이터, 및 제어신호(HDIS를 제외하고)는 상기 클럭의 상승 모서리에 대해 샘플링되고 주장된다.

HA(10:0) 11 비트 어드레스 버스(입력) - 완전한 16 비트 워드에 대해 항상 기술된다.

HD(15:0) 16 비트 데이터 버스(양방향성) - 1sb인 비트 0에 대해, 16 비트 데이터를 항상 전달하고, 비트 0은 1sb이다.

HADS - 어드레스 스트로브(입력) - 새로운 버스 액세스가 시작되는 것을 가리키도록 주장(낮은)된다.

HMODE 전송 시간 모드(입력) - 액세를 위해 사용된 데이터 전송 시간을 명기하기 위하여 HADS와 관련하여 주장된다.

HBRST 버스트 사이클 요구(입력) - 다음 연속 어드레스에서 다른 전송을 요구하기 위하여 데이터의 전송단에서 주장된다.

HWE - 기록 인에이블(입력) - 기록 사이클이 수행되는 것을 가리키기 위하여 HADS - 와 관련하여 주장(낮은)된다.

HDIS 버스 디스에이블(입력) - 만약 주장(높은)된다면, HD(15:0) 핀 출력 구동기를 무조건적으로 3-상태가 된다 ; 상기 신호는 사용되기 전에 MEC에 의해 내부적으로 등록되지 않는다.

HRDY - 데이터 준비(개방 드레인 출력) - 기록 동작, 또는 판독 동작을 위한 버스로부터 데이터를 샘플할 때, 유효 데이터가 데이터 버스에 주장될 때 MEC에 의해 표시(낮은) 된다. 상기 신호는 HCLK의 하나의 상승 가장 자리를 위해 유효한 것이 보장되고 레지스터를 사용하는 Vcc에 외부적으로 인출되어야 한다.

HINT - 호스트 인터럽트(출력) - 인에이블될 때, 상기 핀은 HINT 명령 레지스터에 기록하는 CPU에 응답하는 MEC 또는 하드웨어 에러를 검출하는 MEC의 보조 일시 중단 장치에 의해 주장(낮은)된다.

테이블 13

어드레스	내용	형태	리셋	버스	설명
메모철	메모리				
000	M(0) (11 : 0)	RW	없음	11:0	
001	M(1) (11 : 0)	RW	없음	11:0	
.	.	.	.	.	
03F	M(63) (11 : 0)	RW	없음	11:0	
CPU 시간 기록기 호스트 인터페이스 데이터 레지스터					
800	TIMERMS(3:0)	RW	0	3:0	0으로 판독된 비트 11:4
801	TIMERRLS(11:0)	RW	0	11:0	
802	HDIN(11:0)	R	없음	11:0	
803	HDOUT(11:0)	W	없음	11:0	
명령 레지스터					
810	INITMAD	W	없음		무시된 데이터

811	LDVIB(7:0)	W	00	7:0	
812	LDSM	W	없음		무시된 데이터
813	SWAP	W	없음		무시된 데이터
814	SEND	W	없음		무시된 데이터
815	HINT	W	없음		무시된 데이터

상태 비트

820	VIBRDY	R	1	0	0으로 판독된 비트 11:1
821	SMRDY	R	1	0	0으로 판독된 비트 11:1
822	ITBRDY	R	1	0	0으로 판독된 비트 11:1
823	SPRDY	R	1	0	0으로 판독된 비트 11:1
824	BMPIDLE	R	1	0	0으로 판독된 비트 11:1
825	OTBRDY	R	1	0	0으로 판독된 비트 11:1
826	ORQST	R	0	0	0으로 판독된 비트 11:1
827	LKOTBRDY	R	1	0	0으로 판독된 비트 11:1
828	TZERO	R	1	0	0으로 판독된 비트 11:1
829	HDIMRDY	R	0	0	0으로 판독된 비트 11:1
82A	HDOUTACK	R	1	0	0으로 판독된 비트 11:1

비디오 인터페이스 시간 및 제어 레지스터

900	VCNT1(4:0)	RW	없음	4.5	정의되지 않은 것으로 판독된 비트 5, 11
900	VCNT2(4:0)	RW	없음	4.5	정의되지 않은 것으로 판독된 비트 5, 11
901	VCNT3(4:0)	RW	없음	4.5	정의되지 않은 것으로 판독된 비트 5, 11
901	VCNT4(4:0)	RW	없음	4.5	정의되지 않은 것으로 판독된 비트 5, 11
902	VCNT5(4:0)	RW	없음	4.5	정의되지 않은 것으로 판독된 비트 5, 11
902	VCNT8(4:0)	RW	없음	4.5	정의되지 않은 것으로 판독된 비트 5, 11
903	VCNT9(4:0)	RW	없음	4.5	정의되지 않은 것으로 판독된 비트 5, 11
903	VCNT10(4:0)	RW	없음	4.5	정의되지 않은 것으로 판독된 비트 5, 11
904	VCNT11(4:0)	RW	없음	4.5	정의되지 않은 것으로 판독된 비트 5, 11
904	VCNT12(4:0)	RW	없음	4.5	정의되지 않은 것으로 판독된 비트 5, 11
905	VARB	W	0*	4.8	
905	VBUSMODE	W	0*	4.3.1	
905	VREM	W	0*	4.7	
906	VRFPMS(7:0)	RW	없음	4.7	정의되지 않은 것으로 판독된 비트 11:8
907	VRFPLS(7:0)	RW	없음	4.7	정의되지 않은 것으로 판독된 비트 11:8

테이블 13

비디오 인터페이스 버퍼 구성 레지스트

A00	VOORIGIN (6 : 0)	RW	없음	4. 1	정의 되지 않고 판독된 비트 11:7
A01	VOLENGTH (6 : 0)	RW	없음	4. 1	
A01	VOWIDTH (11 : 0)	RW	없음	4. 1	
A02	VOX (11 : 0)	RW	없음	4. 1	기록되지않고 0로 판독된 비트 0
A03	VOY (11 : 0)	RW	없음	4. 1	기록되지않고 0로 판독된 비트 0
A04	VOPCNT(7 : 0)	RW	없음	4. 1	0로 판독된 0 및 비트 11:8 ; 기록

되지 않은 비트 0

A05 VOROW(5 : 0) RW 없음 4. 1 0로 판독된 비트 11 : 6

A06 VOCOL(3 : 0) RW 없음 4. 1 0로 판독된 비트 11 : 4

A07 VOMSEL(2 : 0) RW 없음 4. 1 정의 되지 않고 판독된 비트 11 : 3

A08 V1ORIGIN(6 : 0) RW 없음 4. 1 0로 판독된 비트 11 : 7

A09 V1LENGTH(6 : 0) RW 없음 4. 1

A09 V1WIDTH(4 : 0) RW 없음 4. 1

A0A V1X(11 : 0) RW 없음 4. 1 기록되지않고 0로 판독된 비트 0

A0B V1Y(11 : 0) RW 없음 4. 1 기록되지않고 0로 판독된 비트 0

A0C V1PCNT(7 : 0) RW 없음 4. 1 0로 판독된 0 및 비트 11:8 ; 기록

되지 않은 비트 0

A0D V1ROW(5 : 0) RW 없음 4. 1 0로 판독된 비트 11 : 6

A0E V1COL(3 : 0) RW 없음 4. 1 0로 판독된 비트 11 : 4

A0F V1MSEL(2 : 0) RW 없음 4. 1 정의 되지 않고 판독된 비트 11 : 3

A10 V2ORIGIN(6 : 0) RW 없음 4. 1 0로 판독된 비트 11 : 7

A11 V2LENGTH(6 : 0) RW 없음 4. 1

A11 V2WIDTH(4 : 0) RW 없음 4. 1

A12 V2X(11 : 0) RW 없음 4. 1 기록되지않고 0로 판독된 비트 0

A13 V2Y(11 : 0) RW 없음 4. 1 기록되지않고 0로 판독된 비트 0

A14 V2PCNT(7 : 0) RW 없음 4. 1 0로 판독된 0 및 비트 11:8 ; 기록

되지 않은 비트 0

A15 V2ROW(5 : 0) RW 없음 4. 1 0로 판독된 비트 11 : 6

A16 V2COL(3 : 0) RW 없음 4. 1 0로 판독된 비트 11 : 4

A17 V2MSEL(2 : 0) RW 없음 4. 1 정의 되지 않고 판독된 비트 11 : 3

A18 V3ORIGIN(6 : 0) RW 없음 4. 1 0로 판독된 비트 11 : 7

A19 V3LENGTH(6 : 0) RW 없음 4. 1

A19 V3WIDTH(4 : 0) RW 없음 4. 1

A1A V3X(11 : 0) RW 없음 4. 1 기록되지않고 0로 판독된 비트 0

A1B V3Y(11 : 0) RW 없음 4. 1 기록되지않고 0로 판독된 비트 0

A1C V3PCNT(7 : 0) RW 없음 4. 1 0로 판독된 0 및 비트 11:8 ; 기록

되지 않은 비트 0

A1D V3ROW(5 : 0) RW 없음 4. 1 0로 판독된 비트 11 : 6

A1E V3COL(3 : 0) RW 없음 4. 1 0로 판독된 비트 11 : 4

A1F V3MSEL(2 : 0) RW 없음 4. 1 정의 되지 않고 판독된 비트 11 : 3

서치 메모리 구성 레지스트

B00 S0ORIGIN (8 : 0) RW 없음 5 .1 정의 되지 않고 판독된 비트 11 : 9

B01 S0WIDTH (4 : 0) RW 없음 5 .1 정의 되지 않고 판독된 비트 11 : 5

B02 S1ORIGIN (8 : 0) RW 없음 5 .1 정의 되지 않고 판독된 비트 11 : 9

B03 S1WIDTH (4 : 0) RW 없음 5 .1 정의 되지 않고 판독된 비트 11 : 5

B04 S2ORIGIN (8 : 0) RW 없음 5 .1 정의 되지 않고 판독된 비트 11 : 9

B05 S2WIDTH (4 : 0) RW 없음 5 .1 정의 되지 않고 판독된 비트 11 : 5

B06 S3ORIGIN (8 : 0) RW 없음 5 .1 정의 되지 않고 판독된 비트 11 : 9

B07 S3WIDTH (4 : 0) RW 없음 5 .1 정의 되지 않고 판독된 비트 11 : 5

블록 매칭 프로세스 제어 및 MAD 레지스트

C00 BMPMODE (1 : 0) RW 없음 5 .2 정의 되지 않고 판독된 비트 11:4

C00 BMSEG (1 : 0) RW 없음 5 .2 정의 되지 않고 판독된 비트 11:4

C01 BMXORG (6 : 0) RW 없음 5 .2 정의 되지 않고 판독된 비트 11:7

C02 BMYORG (6 : 0) RW 없음 5 .2 정의 되지 않고 판독된 비트 11:7

C03 MADX (7 : 0) R 없음 5 .2 비트 7과 동일한 비트 11 : 8

C04 MADY (7 : 0) R 없음 5 .2 비트 7과 동일한 비트 11 : 8

C05 NADMS (10) R 7FF 5 .2 0으로 판독된 비트 11

C06 MADLS (4 : 0) R 1F 5 .2 0으로 판독된 비트 11 : 5

C07 SPX (7 : 0) RW 없음 5 .2 비트 7과 동일한 비트 11 : 8

C08 SPY (7 : 0) RW 없음 5 .2 비트 7과 동일한 비트 11 : 8

C09 SPCNT(7 : 0) RW 없음 5 .2 정의 되지 않고 판독된 비트 11 : 8

C0A SPINC(4 : 0) RW 없음 5 .2 정의 되지 않고 판독된 비트 11 : 8

C0A SPMASK(2 : 0) RW 없음 5 .2 정의 되지 않고 판독된 비트 11 : 8

C0B OTBSEG(1 : 0) RW 없음 5 .4 정의 되지 않고 판독된 비트 11 : 2

C0C OTBXORG(6 : 0) RW 없음 5 .4 정의 되지 않고 판독된 비트 11 : 7

C0D OTBXORG(6 : 0) RW 없음 5 .4 정의 되지 않고 판독된 비트 11 : 7

C0E OTBX (7 : 0) RW 없음 5 .4 비트 7과 동일한 비트 11 : 8

C0F OTBY (7 : 0) RW 없음 5 .4 비트 7과 동일한 비트 11 : 8

IPREDICT 레지스트

D00 IPP0 (11 : 0) R 없음 3 .3

D01 IPP2 (11 : 0) R 없음 3 .3

D02 IPP2 (11 : 0) R 없음 3 .3

D03 ITD0 (11 : 0) R 없음 3 .3

D04 ITD1 (11 : 0) R 없음 3 .3

D05 ITD2 (11 : 0) R 없음 3 .3

D06 ITD3 (11 : 0) R 없음 3 .3

D07 IBITS (3 : 0) R 없음 3 .3 정의되지않고 판독된 비트 11 : 4

D08 IFMVX (11 : 0) R 없음 3 .3 비트 10과 동일한 비트 11

D09 IBMVX (11 : 0) R 없음 3 .3 비트 10과 동일한 비트 11

D0A IBMVY (11 : 0) R 없음 3 .3 비트 10과 동일한 비트 11

D0B IBMVY (11 : 0) R 없음 3 .3 비트 10과 동일한 비트 11

OPREDICT 레지스트

E00 OPP0 (11 : 0) R 없음 3 .4

E01 OPP1 (11 : 0) R 없음 3 .4

E02 OPP2 (11 : 0) R 없음 3 .4

E03 OTD0 (11 : 0) W 없음 3 .4

E04 OTD1 (11 : 0) W 없음 3 .4

E05 OTD2 (11 : 0) W 없음 3 .4

E06 OTD3 (3 : 0) W 없음 3 .4

E07 OBITS (10 : 0) W 없음 3 .4

E08 OFMVX (10 : 0) W 없음 3 .4

E09 OFMVY (10 : 0) W 없음 3 .4

E09 OFMVY (10 : 0) W 없음 3 .4
E0A OBMVX (10 : 0) W 없음 3 .4

테이블 14 : MEC 보조 인터페이스 버스 신호 정의

XCLK 보조 인터페이스 클럭 (입력) - 모든 어드레스, 데이터, 및 제어 신호는 이 클럭의 상승 끝부분에 대해 상대적으로 주장되거나 표본화 된다.

XAD(23:0) 24비트 멀티플렉스된 어드레스/데이터 버스 (입력/출력)

XADS- 보조 프로세서 어드레스 /데이터 스트로브(입력 - XADS-는 모든 프로세스 판독 또는 기록 전송 시퀀스의 초기에 XCLK 사이클을 위한 ICC에 의해 (로우)로 표시된다(이미지 압축 코프로세스).

XOE- 출력 인에이블(입력)- XOE-는 판독 사이클중에 ICC에 의해 MEC 의 출력 버퍼가 XAD로 도달하게 (로우)로 표시된다; 만약 XOE가 하이되면 이 출력 버퍼들은 고 임피던스 상태로 된다.

XRDY 데이터 준비 (3 상태 출력)- 이 신호는 데이터 전송 스퀀스 전체를 통하여 MEC에 의해 데이터를 공급하기 위한 준비(판독의 경우에) 또는 데이터의 접수 준비(기재의 경우)중 하나를 나타내기 위하여 (로우)로 표시된다. MEC는 XRDY를 3 상태로 하지않고, XCLK 클럭 사이클이 전송 시퀀스의 초기에 XRDY의 표시에 앞서 그것을 소거하고 전송 시퀀스 끝부분에 그것의 소거후에 고 임피던스 상태의 한 클럭에 다시 되돌려 보낸다.(그림 3-1 및 3 -2 참조)

XRQST- 프로세스 서비스 요구(출력) - 이 신호는 ICC에 의해 그것에 빨리 전송된 명령의 실행이 끝나자마다 MEC에 의해 주장되고; 이 출력은 ICC의 XRQST-(3:0) 핀에 연결되어야 한다.

XDIS 출력 디스에이블(입력)- 이 신호가 (하이)로 주장될 때, MEC에 XRDY 및 XAD(23:0) 출력 구동기가 고 임피던스 상태로 된다.

XID(1:0) 프로세스 ID 선택(입력) - 이 핀들은 ICC와 통신할 목적으로 MEC의 "어드레스" 또는 프로세스 ID를 표시한다. 그들은 시스템에서 하드웨어화된다. 그것이 ICC에 의해 선택되는지를 결정하기 위하여, MEC는 프로세서 판독 또는 기재 사이클의 시작에서 XAD(23:0) 버스의 비트 1:0와 이 핀 들을 비교한다.

* ICC = 이미지 압축 코프로세스

테이블 15 : MEC 명령 오피코드

기계어	오피코드	보통 정의
IPREDICT	XX10XX	입력 토큰에서 모션 평가 수행
OPREDICT	XX01XX	이미지 압축 코프로세스에 모션 평가로부터의 결과 토큰 전송
WRMEM	XX11XX	예측 메모리에 입력 신호를 기록

테이블 16 : 프로세서 패킷 워드 정의

단어	비트 위치	내용
0	(23 : 17)	명령 어드레스
0	(16)	오퍼랜드 토큰 기술자(기재 사이클용) 또는 결과 토큰 기술자 (판독 사이클용)로부터 타입 필드
0	(15 : 14)	오퍼랜드 토큰 기술자(기재 사이클용) 또는 결과신호 기술자(판독 사이클용)로부터의 nblock 필드
0	(13 : 12)	ICC 명령 비트 30 : 29 (ND 필드)
0	(11)	ICC 명령 비트 32 (MD(0) 필드)
0	(10 : 6)	ICC 명령 비트 4 : 0 (PARAMS(4:0)필드)
0	(5 : 0)	ICC 명령 비트 39 : 34 (오피코드 필드)
1	(23 : 16)	ICC 명령 비트 28 : 21 (DEST1 필드)
1	(15 : 8)	ICC 명령 비트 20 : 13 (DEST2 필드)
1	(7 : 0)	ICC 명령 비트 12 : 5 (DEST3 필드)

* ICC* 이미지 압축 코프로세서

테이블 17 : 보조 인터페이스 버스에 토큰 전송 포맷

버스 전송 번호	XAD(23:12) 내용	XAD(11:0) 내용
0	기술 비트 95 : 84	기술자 비트 83 : 72
1	기술자 비트 71 : 60	기술자 비트 59 : 48
2	기술자 비트 47 : 36	기술자 비트 35 : 24
3	기술자 비트 23 : 12	기술자 비트 11 : 0
4	블록 0, 행 0, 열 0	블록 0, 행 0, 열 1
.	.	
35	블록 0, 행 7, 열 6	블록 0, 행 7, 열 7
.	.	.
.	.	.
nblock * 32 + 4	블록 nblock, 행 0 열 0	블록 nblock, 행 0 열 0
	열 0	
nblock * 32 + 35	블록 nblock, 행 7 열 6	블록 nblock, 행 7 열 7

테이블 18 : MEC 하드웨어 에러 조건

에러 조건	기재
PR(3 :2) ==0	정의되지 않는 MEC 오피코드
{RP0 (3)==1 & & RP0(11)==0} ‖ {RP0 (3)==1 & & PR0(16)==1& & PP0(15 :14) !* 3}	IPREDICT 또는 WRMEM 명령은 4개의 블록보다 작은 데이터 신호 오퍼랜드, 또는 전혀 오퍼랜드를 가지고 있지 않다.
PP0 (3:2)==3 & & PR0(16)==0	WEMEM 명령은 제어 토큰 오퍼랜드를 갖고 있다.
PP0 (3:2)==1 & & PR0(13 : 12)==0	OPREDICT 명령은 목적지를 가지고 있지 않다.

주 : PP0 : 프로세스 패킷 워드 0

PP1 : 프로세스 패킷 워드 1

테이블 19 : IPREDICT 토큰 기술자 교환 레지스트

레지스터	기술 비트	내용
ITD0(11 : 0)	95 : 84	QUANT(4)와 기술자 비트 95 : 85의 연결
ITD1(11 : 0)	83 : 72	QUANT(3 :0)과 VPOS의 연결
ITD2(11 : 0)	71 : 60	cntr1(7:4)와 hpos의 연결
ITD3(11 : 0)	59 : 48	cntr2와 cntr1(3, 0)의 연결
IBITS(3 : 0)	47 : 44	USR비트
IFMVX(10 : 0)	43 : 3	순방향 모션 벡터의 수평 성분
IBMVY(10 : 0)	32 : 22	순방향 모션 벡터의 수직 성분
IBMV4(10 : 0)	21 : 11	역방향 모션 벡터의 수평 성분
IBMVY(10 : 0)	10 : 0	역방향 모션 벡터의 수평 성분

테이블 20 : OTPB 로드 구성 레지스트

OTBSEG(1 : 0) : (0, 1, 2, , 3)에 엑세스될 검색 메모리 세그먼트

OTBXORG(6 : 0) : 세그먼트 OTBSEG에서 원점의 절대 X 좌표

OTBYORG(6 : 0) : 세그먼트 OTBSEG에서 원점의 절대 Y 좌표

OTBX (7 : 0) : OTPB로 복사될 16 * 16 블록의 상대적 X 좌표

OTBX (7 : 0) : OTPB로 복사될 16 * 16 블록의 상대적 Y 좌표

테이블 21 : IPREDIT 프로세스 패킷 교환 레지스터

레지스터	내용
IPR0(11 : 0)	IPPB로부터 PP0(11 : 0)
IPR1(11 : 0)	IPPB로부터 PP1(23 : 12)
IPR2(11 : 0)	IPPB로부터 PP1(11 : 0)

테이블 22 : OPRKDICT 프로세스 패킷 레지스터

레지스트	내용
OPP0(11 : 0)	OPPB로부터 PP0(11 : 0)
OPP1(11 : 0)	OPPB로부터 PP1(23 : 12)
OPP2(11 : 0)	OPPB로부터 PP1(11 : 0)

테이블 23 : OPREDIT 토큰 기술자 레지스터

레지스터	기술자 비트	내용
OTD0(11 : 0)	95 : 84	QUANT(4)와 기술자 비트 95 : 85의 연결
OTD1(11 : 0)	83 : 72	QUANT(3 :0)과 VPOS의 연결
OTD2(11 : 0)	71 : 60	CNTR1(7:4)와 hpos의 연결
OTD3(11 : 0)	59 : 48	CNTR2와 cntr1(3:0)의 연결
OBITS(3 : 0)	47 : 44	USRbit
OFMVX(10 : 0)	43 : 33	순방향 모션 벡터의 수평 성분
OFMVY(10 : 0)	32 : 22	역방향 모션 벡터의 수직 성분
OBMV4(10 : 0)	21 : 11	순방향 모션 벡터의 수평 성분
OBMVY(10 : 0)	10 : 0	역방향 모션 벡터의 수직 성분

(57) 청구의 범위

청구항 1

모션 비디오 압축 장치에 있어서, 현재 프레임을 기술하는 데이터 토큰을 출력하기 위한 이미지 압축 코프로세서; P행 Q열의 물리적인 어레이에 비디오 이미지 데이터를 저장하기 위한 비디오 메모리; 및 상기 이미지 압축 코프로세서 및 상기 비디오 메모리와 결합된 모션 평가 코프로세서를 포함하며, 상기 모션 평가 코프로세서는, 상기 비디오 메모리로부터 복수의 서치 윈도우를 검색하기 위한 수단; 상기 검색 수단에 결합되고, 상기 현재 프레임과 상기 복수의 서치 윈도우를 비교하고 최적 매칭 블록을 식별하고 상기 최적 매칭 블록과 상기 현재 프레임 사이의 차이를 기술하는 벡터를 계산하기 위한 수단; 및 상기 비교 수단에 결합되고, 상기 벡터 및 상기 최적 매칭 블록 둘 모두를 포함하는 출력 토큰을 상기 이미지 압축 코프로세서에 출력하기 위한 수단을 포함하는 것을 특징으로 하는 모션 비디오 압축 장치.

청구항 2

P행 Q열로 물리적으로 배열된 비디오 메모리를 갖는 비디오 시스템에서의 모션 평가 프로세서에 있어서, 현재 프레임을 기술하는 데이터 토큰을 수신하기 위한 수단; 상기 비디오 메모리에 결합되어 있고, 상기 비디오 메모리로부터 P행 Q열을 갖도록 논리적으로 구성된 복수의 이미지 데이터 타일을 검색하기 위한 수단; 상기 검색 수단과 결합되어 있고, 상기 P행 중 짝수로 번호 매겨진 것 및 상기 Q열 중 짝수로 번호 매겨진 것으로부터 검색된 이미지 데이터를 저장하기 위한 제 1 저장 장치; 상기 검색 수단과 결합되어 있고, 상기 P행 중 홀수로 번호 매겨진 것 및 상기 Q 열 중 홀수로 번호 매겨진 것으로부터 검색된 이미지 데이터를 저장하기 위한 제 2 저장 장치; 상기 제 1 및 상기 제 2 저장 장치와 결합되고 서치 윈도우를 저장하기 위한 서치 메모리; 상기 검색 수단에 결합되어 있고, 상기 현재 프레임과 상기 서치 윈도우를 비교하고 최적 매칭 블록을 식별하고 상기 최적 매칭 블록과 상기 현재 프레임 사이의 차이를 기술하는 벡터를 계산하기 위한 블록 매칭 프로세서; 및 상기 블록 매칭 프로세서와 결합되어 있고, 상기 벡터 및 상기 최적 매칭 서치 윈도우를 포함하는 출력 토큰을 출력하기 위한 수단을 포함하는 것을 특징으로 하는 모션 평가 프로세서.

청구항 3

제2항에 있어서, 상기 서치 메모리는 0 〈 n ≤4에 대하여 n 다중 서치 윈도우를 저장하도록 형성할 수 있는 것을 특징으로 하는 모션 평가 프로세서.

청구항 4

제2항에 있어서, 상기 서치 메모리와 상기 블록 매칭 프로세서 사이에 배치된 보간기를 더 포함하는 것을 특징으로 하는 모션 평가 프로세서.

청구항 5

제2항에 있어서, 데이터 토큰을 수신하기 위한 상기 수단 중 적어도 하나와 결합하거나 감소된 스케일 이미지를 형성하기 위하여 개별적으로 상기 현재 프레임 또는 상기 이미지 데이터 중 하나를 데시메이션 필터링하도록 비디오 메모리로부터 검색하기 위한 상기 수단과 결합된 데시메이션 필터를 더 포함하고, 상기 블록 매칭 프로세서는 상기 감소된 스케일 이미지를 사용하여 상기 벡터 및 상기 출력 토큰을 계산하는 것을 특징으로 하는 모션 평가 프로세서.

청구항 6

제2항에 있어서, 상기 블록 매칭 프로세서는 상기 서치 윈도우의 n 행의 주어진 열 위치에서 시작하는 n 블럭 매칭을 동시에 수행되고 n ≥2이고 상기 행은 복수의 연속적인 또는 교번하는 행 중 하나인 것을 특징으로 하는 모션 평가 프로세서.

청구항 7

제2항에 있어서, 상기 제어 신호에 응답하는 상기 블록 매칭 프로세서의 동작을 지시하기 위하여, 상기 블록 매칭 프로세서와 결합되어 있고, 상기 모션 평가 프로세서 외부의 CPU에 의하여 상기 모션 평가 프로세서에 대한 제어 신호 입력을 수신하기에 적합한 수단을 더 포함하는 것을 특징으로 하는 모션 평가 프로세서.

청구항 8

제6항에 있어서, 상기 제어 신호에 응답하고, 상기 서치 윈도우내에서 kxn 블록 매칭을 수행하도록 상기 블록 매칭 프로세서를 지시하기 위하여, 상기 블록 매칭 프로세서와 결합되어 있고, 상기 모션 평가 프로세서 외부의 CPU에 의하여 상기 모션 평가 프로세서에 대한 제어 신호 입력을 수신하기에 적합한 수단을 더 포함하는 것을 특징으로 하는 모션 평가 프로세서.

청구항 9

제6항에 있어서, 상기 블록 매칭 프로세서는, 상기 n 블록 매칭 중 하나를 계산하기 위한 가산기 트리를 각각 갖는 n 모듈; 상기 n 모듈에 결합되어 있고, P=a 상수인 경우 K=1에서 K=n-1까지 의 클럭 사이클의 pik 번호에 의하여 모듈 n=K+1으로 입력을 지연시키기 위한 수단을 더 포함하는 것을 특징으로 하는 모션 평가 프로세서.

청구항 10

제9항에 있어서, 상기 n 모듈 각각으로부터의 출력의 합은 차례로 레지스터에 저장된 값과 비교되고 상기 값은 상기 출력 합이 상기 저장된 값보다 작을 때 오버라이트되는 것을 특징으로 하는 모션 평가 프로세

서.

청구항 11

제10항에 있어서, 상기 출력 합 및 변위 벡터는 상기 출력 합이 상기 저장된 값보다 작을 때 상기 레지스터에 기록되는 것을 특징으로 하는 모션 평가 프로세서.

청구항 12

모션 평가 프로세서에서, 페이지 각각이 j행×N열 블록을 포함하는 페이지의 P/j행×K열 어레이를 포함하고, 여기서 K=하나의 정수인 동안 Q=N*K인 것을 특징으로 하는 M 비트 화소의 P행×Q열 이미지를 저장하기 위한 장치.

청구항 13

제12항에 있어서, 상기 페이지를 저장하고 상기 페이지로부터 짝수로 번호 매겨진 행을 저장하기 위한 제 1 뱅크 및 상기 페이지로부터 홀수로 매겨진 열을 저장하기 위한 제 2 뱅크를 갖는 메모리 구조를 더 포함하는 것을 특징으로 하는 장치.

청구항 14

제12항에 있어서, j=1인 것을 특징으로 하는 장치.

청구항 15

제12항에 있어서, j=2인 것을 특징으로 하는 장치.

청구항 16

제12항에 있어서, 상기 페이지를 저장하기 위한 B 뱅크를 갖는 메모리 구조를 더 포함하며, 상기 B 뱅크 중 하나는: M 비트 폭×(kxP)/B 워드의 N 독립 메모리를 포함하는데, n=1 내지 n=N인 동안, 상기 N개의 독립적인 메모리 중 n번째 메모리는 단일 뱅크에서 모든 페이지 행의 n번째 열로부터 복수의 화소를 저장하고 주어진 페이지로 부터의 모든 화소들은 주어진 어드레스에 저장되고, N×M 비트 워드로써의 상기 출력을 원형으로 회전시키도록 시프트 카운트를 사용하기 위하여 상기 N개의 독립적인 메모리의 출력에 결합된 크로스바; 및 N개의 멀티플렉서 각각은 상기 N 메모리의 어드레스 입력에 결합된 출력을 갖고 제 1 어드레스 라인에 결합된 제 1 입력을 가지며 제 2 어드레스 라인에 결합된 제 2 입력을 가지며, 상기 N개의 멀티플렉서 각각은 분리된 선택기 입력을 가지는 N개의 멀티플렉서 세트를 포함하는 것을 특징으로 하는 장치.

청구항 17

제16항에 있어서, 화소 어드레스를 수신하도록 결합되어 있고 상기 시프트 카운트, 상기 화소 어드레스 및 상기 N 메모리에 대한 상기 어드레스를 출력하고, 상기 N 메모리로부터 화소를 출력하기 위한 행 또는 열 시작 위치중 하나를 제어하기 위한 메모리 제어기를 더 포함하는 것을 특징으로 하는 장치.

청구항 18

P 행 ×Q 열의 비디오 메모리를 갖는 비디오 데이터 시스템에서, 모션 평가 방법에 있어서, 현재 이미지 데이터 프레임을 수신하는 단계; 제 1 메모리의 상기 P 행 중 홀수행 및 제 2 메모리의 상기 P행 중 짝수행을 저장하는 단계; 서치 윈도우를 얻기 위하여 상기 제 1 및 제 2 메모리를 어드레싱하는 단계; 상기 서치 윈도우와 상기 현재 이미지 데이터 프레임 사이의 평균 절대 차를 계산하는 단계; 및 모션 벡터 및 최적 매칭 블록을 출력하는 단계를 포함하는 것을 특징으로 하는 모션 평가 방법.

청구항 19

제18항에 있어서, 상기 저장 단계는: 상기 P ×Q 비디오 메모리를 P행 및 Q열 타일의 어레이로 논리적으로 부분 분리하는 단계; 및 상기 제 1 및 제 2 메모리의 상기 타일 중 주어진 하나를 저장하는 단계를 포함하는 것을 특징으로 하는 모션 평가 방법.

청구항 20

이미지 데이터를 저장하기 위한 비디오 메모리를 갖는 비디오 시스템에서, 모션 평가 처리 방법에 있어서, 화소 데이터의 현재 이미지를 수신하는 단계; 상기 비디오 메모리로부터 비디오 이미지 데이터의 최고 해상도 서치 윈도우를 검색하는 단계; 감소된 해상도 서치 윈도우를 얻기 위하여 상기 서치 윈도우를 데시메이션 필터링하는 단계; 및 상기 현재 이미지 및 상기 감소된 해상도 서치 윈도우를 블록 매칭하는 단계를 포함하는 것을 특징으로 하는 모션 평가 방법.

청구항 21

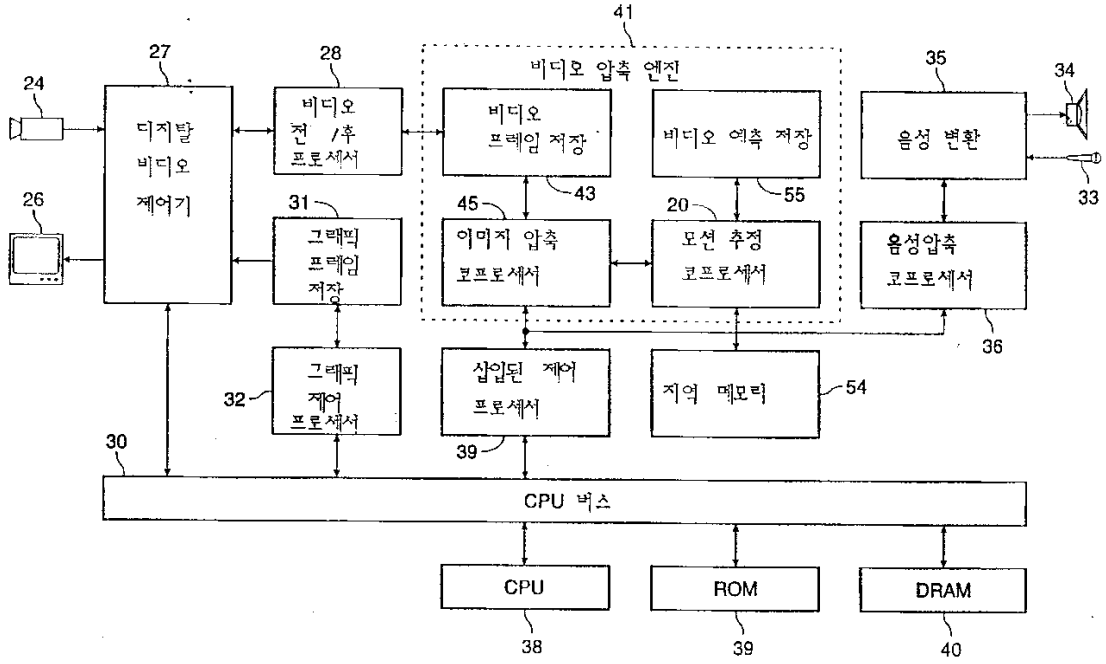
제20항에 있어서, 상기 현재 이미지와 가장 작은 최소 절대 차이를 갖는 감소된 해상도 서치 윈도우를 식별하는 단계; 및 상기 현재 이미지와 상기 최고 해상도의 서치 윈도우를 블록 매칭하는 단계를 더 포함하는 것을 특징으로 하는 모션 평가 방법.

청구항 22

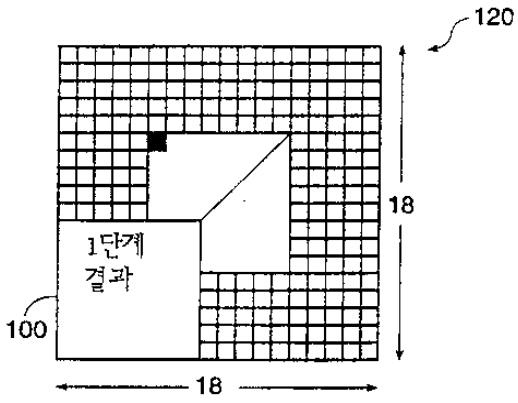
제20항에 있어서, 상기 감소된 해상도 서치 윈도우의 최종 해상도를 제어하기 위하여 필터값을 선택하는 단계를 더 포함하는 것을 특징으로 하는 모션 평가 방법.

도면

도면1

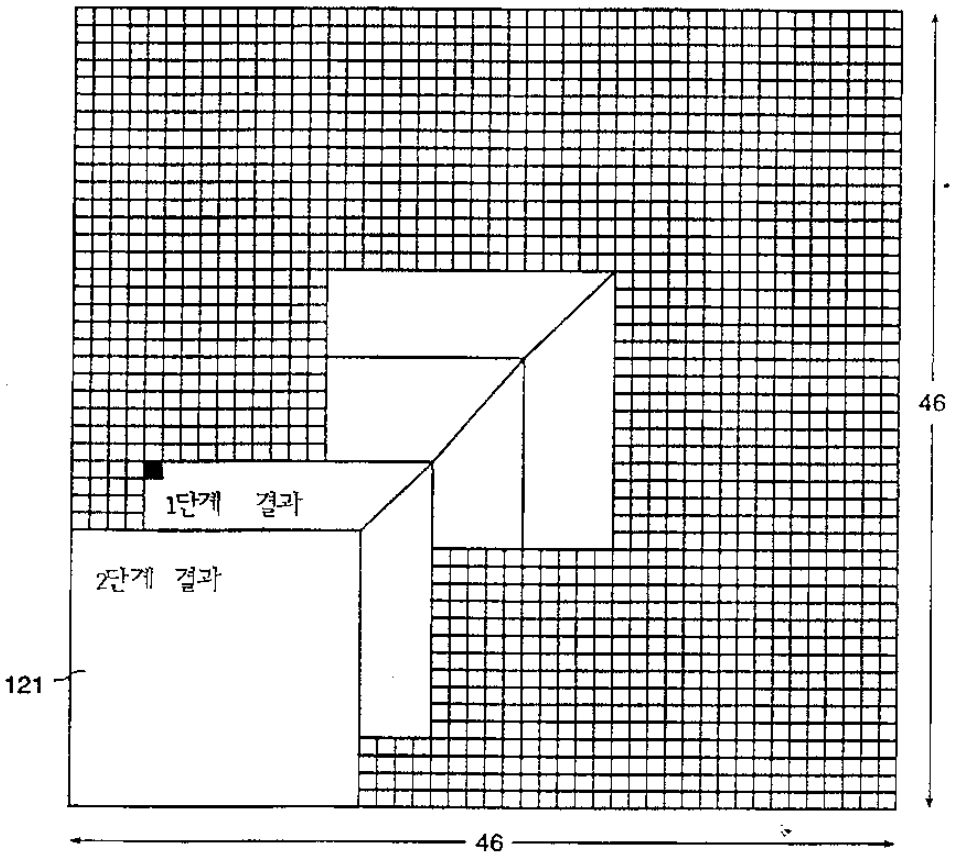


도면2a

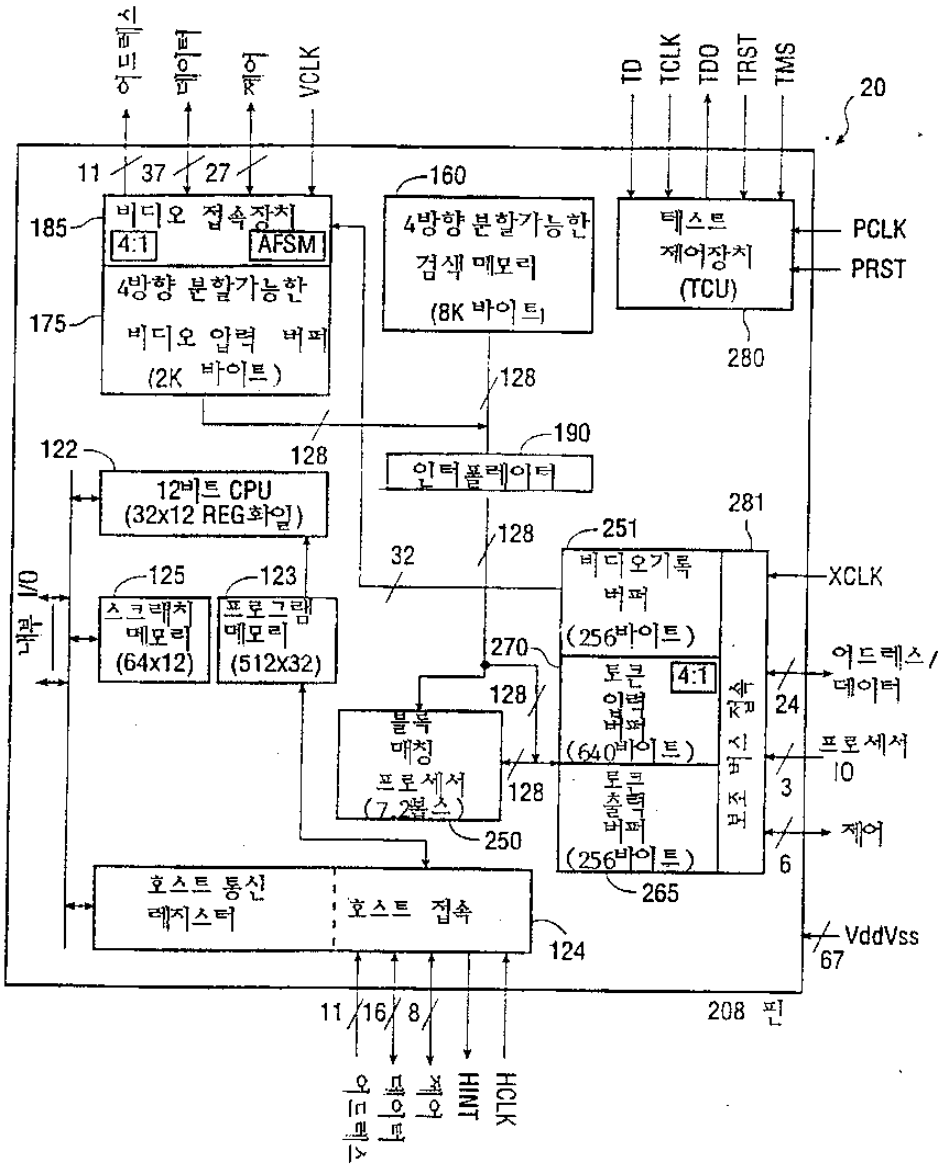


□ = 현재 검색 좌표 ■ = 현재 윤곽 검색 영역

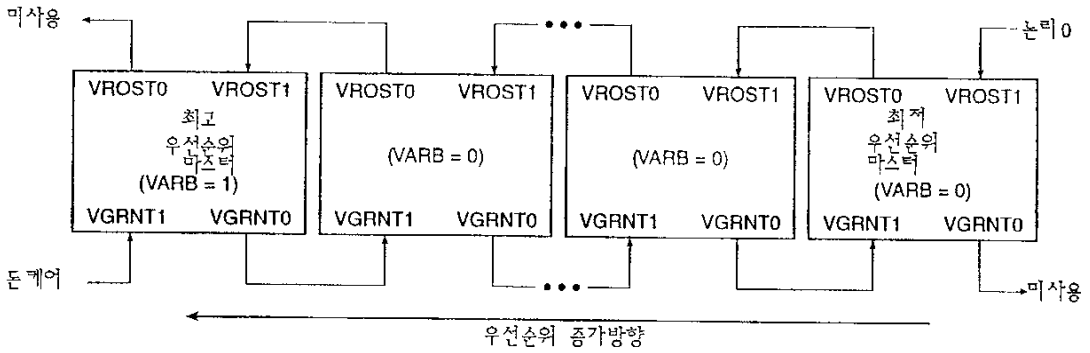
도면2b



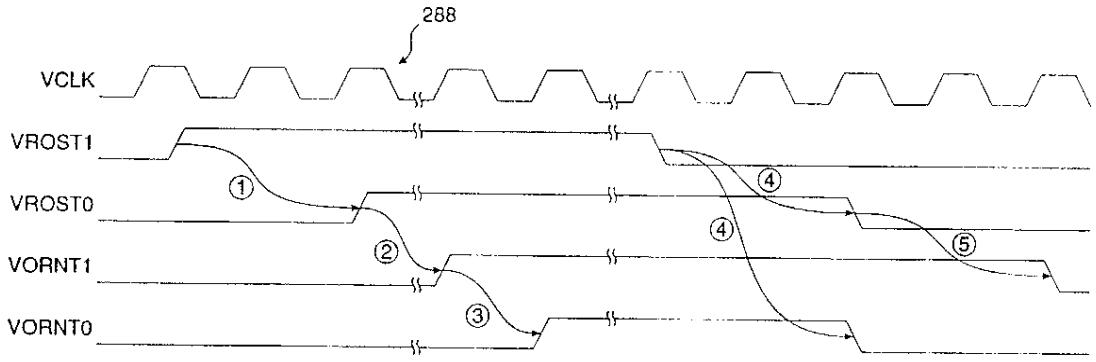
도면3


비디오 데이터 입력
VCLK
TD
TCLK
TDO
TRST
TMS
20
11
37
27
160
185
비디오 접속장치
4:1
AFSM
4방향 분할가능한
검색 메모리
(8K 바이트)
테스트
제어장치
(TCU)
PCLK
PRST
4방향 분할가능한
비디오 입력 버퍼
(2K 바이트)
175
280
128
128
190
122
12비트 CPU
(32x12 REG화일)
인터폴레이터
281
251
128
32
비디오기록
버퍼
(256바이트)
XCLK
I/O 버퍼
125
123
스크래치
메모리
(64x12)
프로그램
메모리
(512X32)
270
토큰
이력
버퍼
(640바이트)
4:1
128
어드레스/
데이터
24
블록
매칭
프로세서
(7.2봅스)
128
프로세서
IO
3
250
토큰
출력
버퍼
(256바이트)
제어
6
265
호스트 통신
레지스터
호스트 접속
124
VddVss
67
11
16
8
208 핀
어드레스
데이터
HINT
HCLK

도면4


미사용
논리 0
VROST0
VROST1
최고
우선순위
마스터
(VARB = 1)
VGRNT1
VGRNT0
(VARB = 0)
최저
우선순위
마스터
돈케어
미사용
우선순위 증가방향

도면5



도면6

289

272

	0	1	2	3
0	00 01 02 03 04 05 06 07			
	10 11 12 13 14 15 16 17			
1	20 21 22 23 24 25 26 27			
	30 31 32 33 34 35 36 37			
2	40 41 42 43 44 45 46 47			
	50 51 52 53 54 55 56 57			
3	60 61 62 63 64 65 66 67			
	70 71 72 73 74 75 76 77			

도면7

290

	0	1	2	3
0	00	02	04	06
1	20	22	24	26
2	40	42	44	46
3	60	62	64	66

비디오 데이터 버스# 0

292

	0	1	2	3
0	01	03	05	07
1	21	23	25	27
2	41	43	45	47
3	61	63	65	67

비디오 데이터버스 # 2

291

	0	1	2	3
0	10	12	14	16
1	30	32	34	36
2	50	52	54	56
3	70	72	74	76

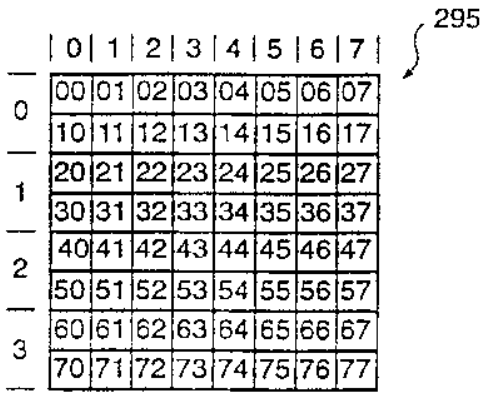
비디오 데이터 버스 # 1

293

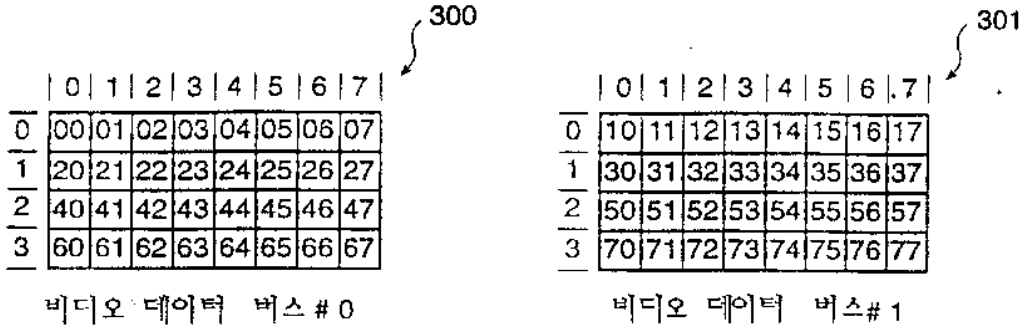
	0	1	2	3
0	11	13	15	17
1	31	33	35	37
2	51	53	55	57
3	71	73	75	77

비디오 데이터버스 # 3

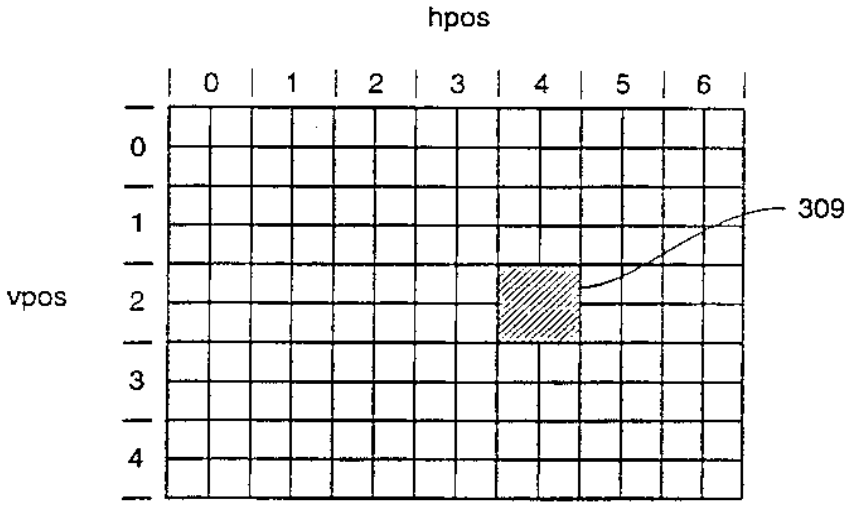
도면8


295
0 1 2 3 4 5 6 7
0 00 01 02 03 04 05 06 07
10 11 12 13 14 15 16 17
1 20 21 22 23 24 25 26 27
30 31 32 33 34 35 36 37
2 40 41 42 43 44 45 46 47
50 51 52 53 54 55 56 57
3 60 61 62 63 64 65 66 67
70 71 72 73 74 75 76 77

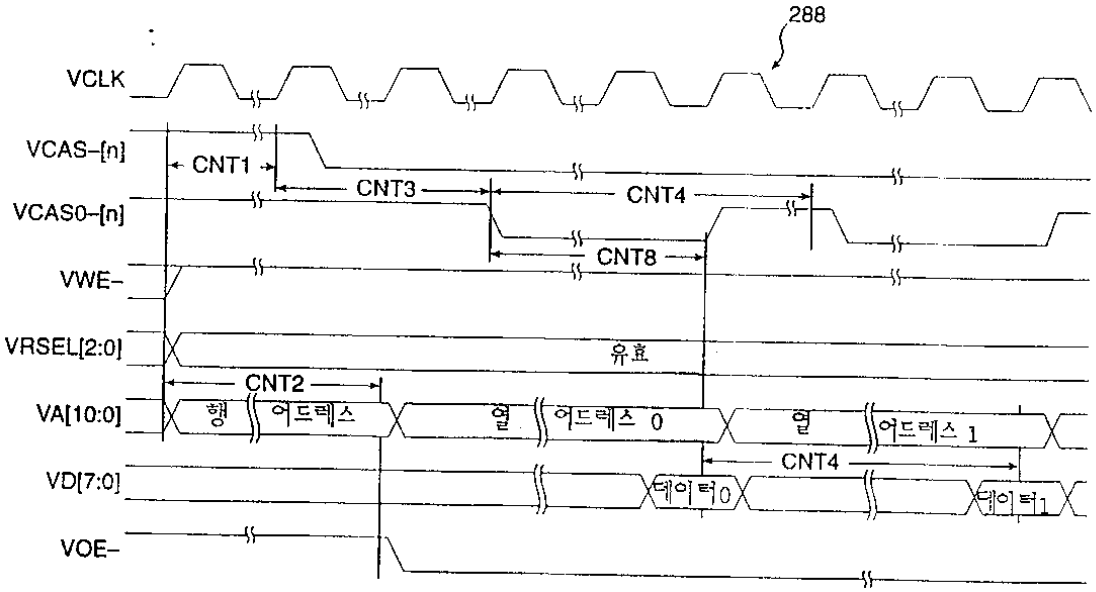
도면9


300
0 1 2 3 4 5 6 7
0 00 01 02 03 04 05 06 07
1 20 21 22 23 24 25 26 27
2 40 41 42 43 44 45 46 47
3 60 61 62 63 64 65 66 67
비디오 데이터 버스 # 0
301
0 1 2 3 4 5 6 7
0 10 11 12 13 14 15 16 17
1 30 31 32 33 34 35 36 37
2 50 51 52 53 54 55 56 57
3 70 71 72 73 74 75 76 77
비디오 데이터 버스# 1

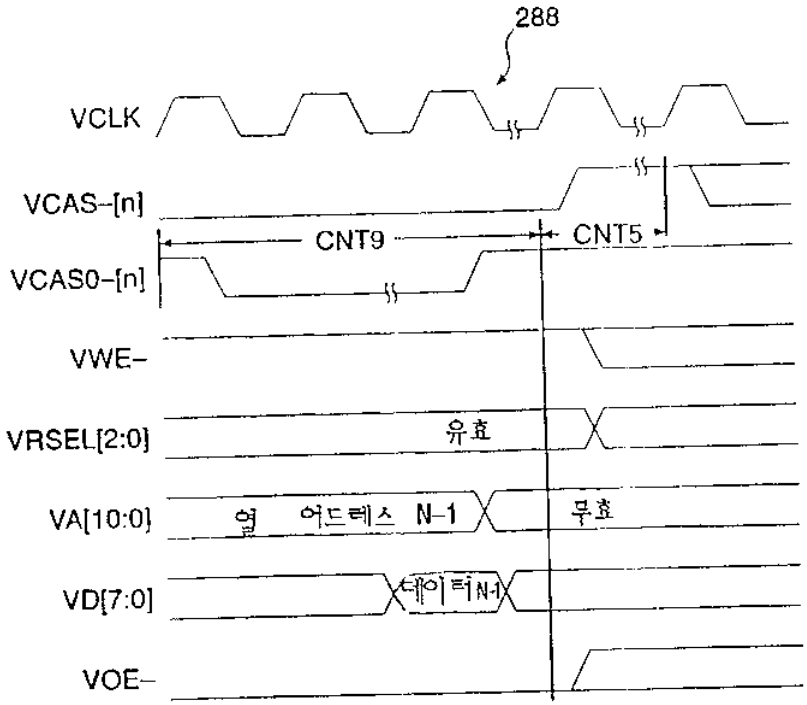
도면10


hpos
0 1 2 3 4 5 6
0
1
2
3
4
vpos
309

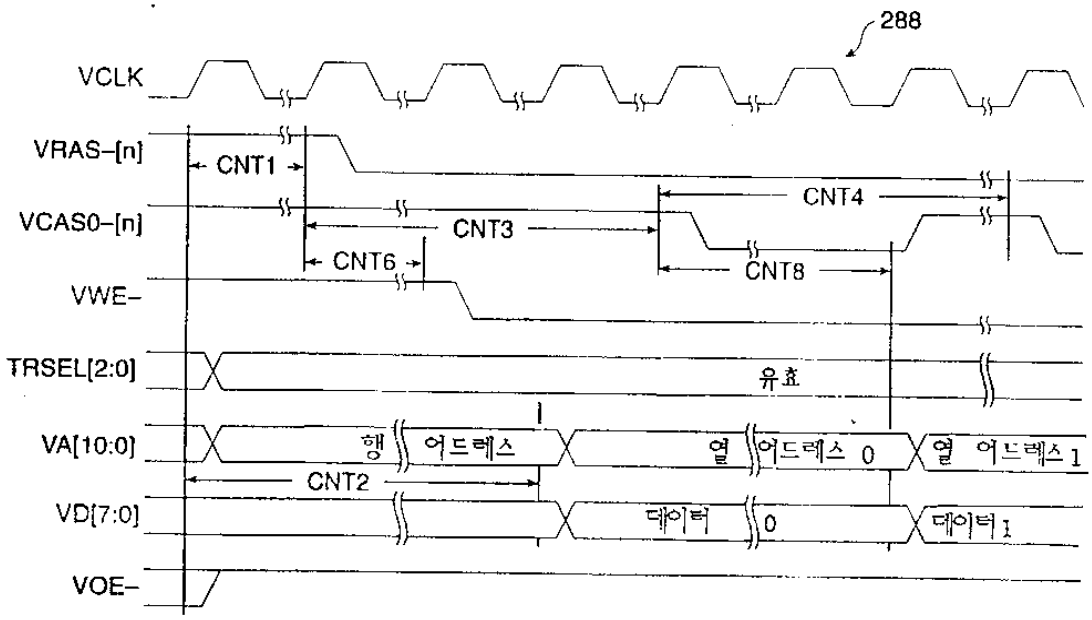
도면11a


288
VCLK
VCAS-[n]
CNT1
CNT3
CNT4
VCAS0-[n]
CNT8
VWE-
VRSEL[2:0]
유효
CNT2
VA[10:0]
행 어드레스
열 어드레스 0
열 어드레스 1
CNT4
VD[7:0]
데이터0
데이터1
VOE-

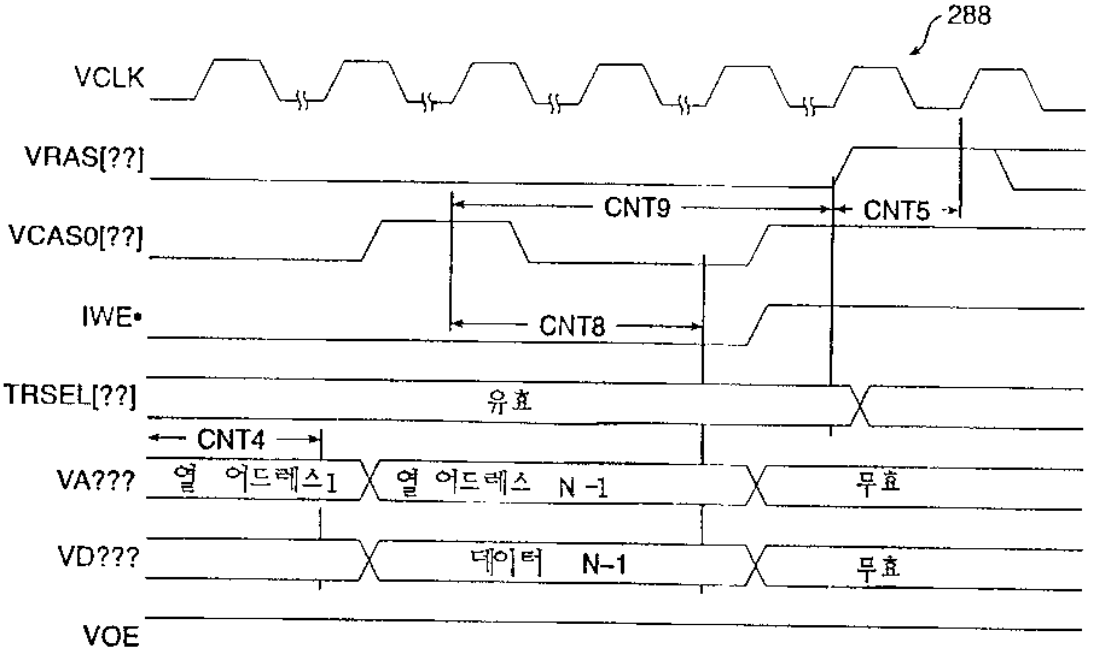
도면11b


288
VCLK
VCAS-[n]
CNT9
CNT5
VCAS0-[n]
VWE-
VRSEL[2:0]
유효
VA[10:0]
열 어드레스 N-1
무효
VD[7:0]
데이터N-1
VOE-

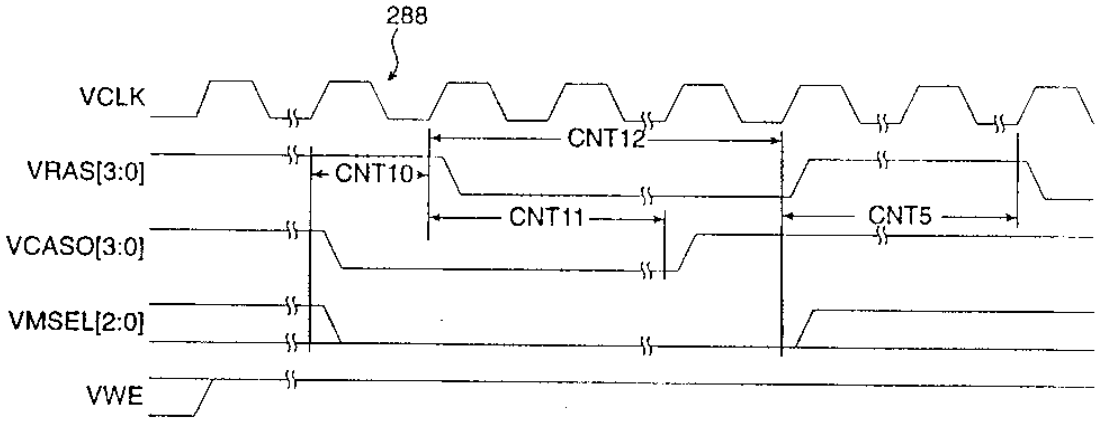
도면12a


288
VCLK
VRAS-[n]
VCAS0-[n]
VWE-
TRSEL[2:0]
VA[10:0]
VD[7:0]
VOE-
CNT1
CNT2
CNT3
CNT4
CNT6
CNT8
유효
행 어드레스
열 어드레스 0
열 어드레스 1
데이터 0
데이터 1

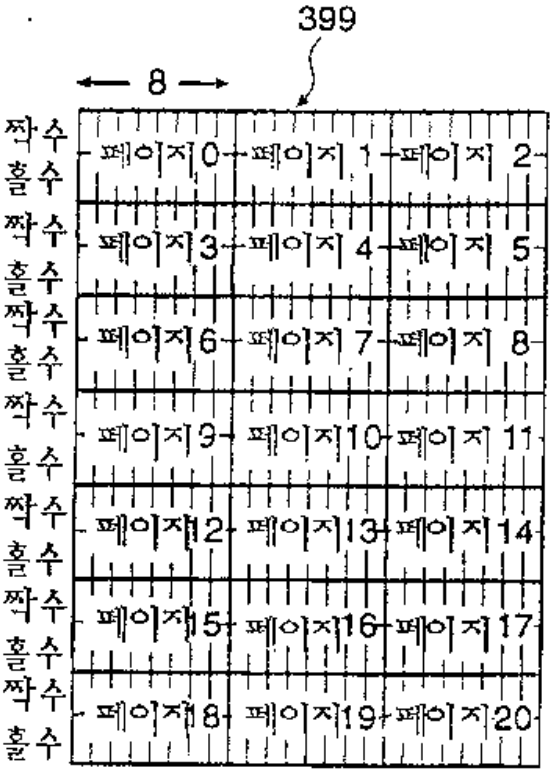
도면12b


288
VCLK
VRAS[??]
VCAS0[??]
IWE•
TRSEL[??]
VA???
VD???
VOE
CNT4
CNT5
CNT8
CNT9
유효
열 어드레스1
열 어드레스 N -1
무효
데이터 N-1
무효

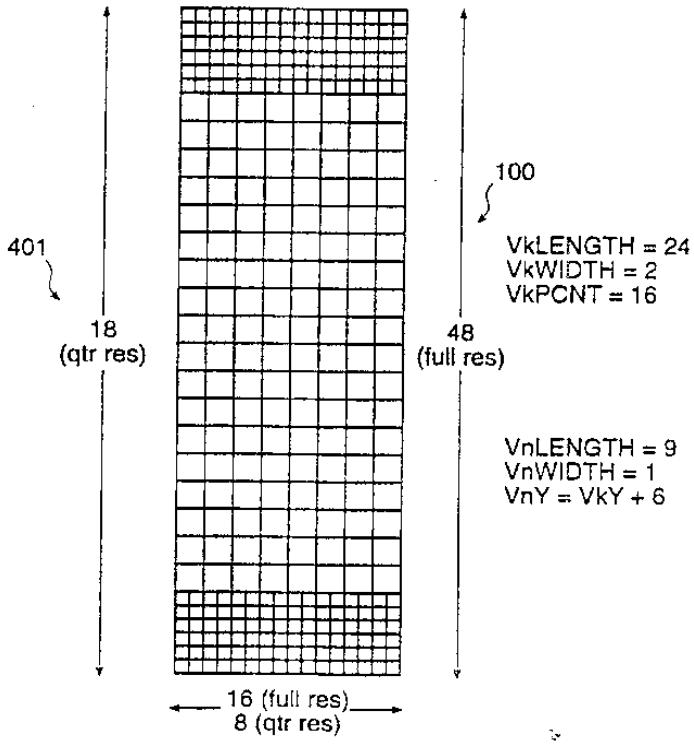
도면13


288
VCLK
VRAS[3:0]
VCASO[3:0]
VMSEL[2:0]
VWE
CNT12
CNT10
CNT11
CNT5

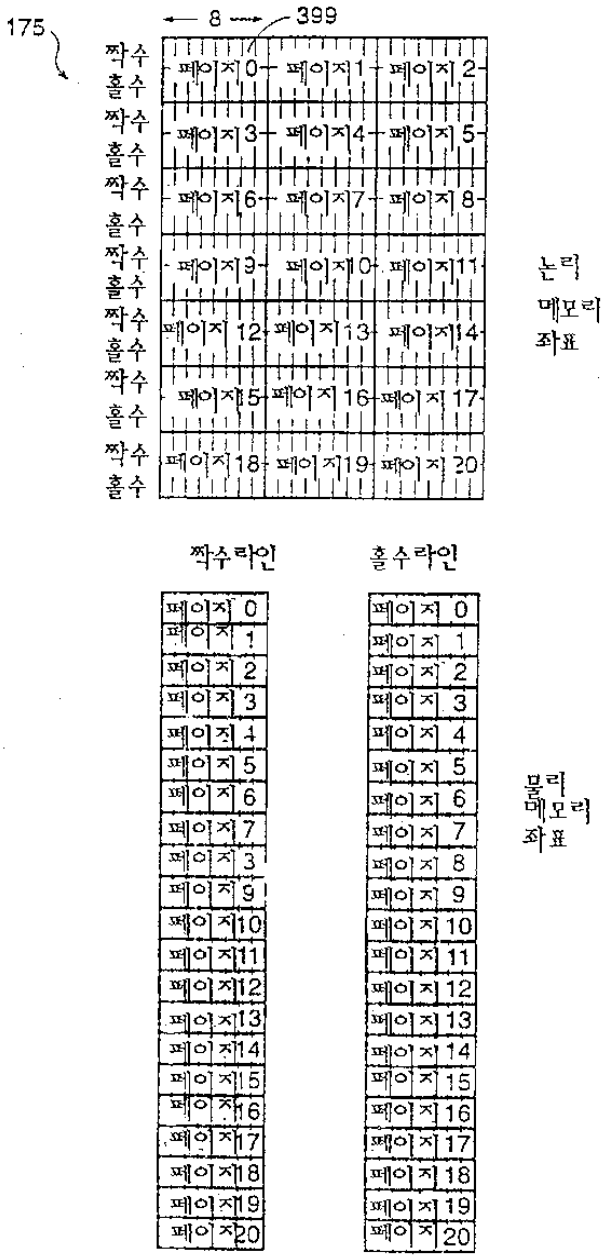
도면14


399
8
짝수
홀수
짝수
홀수
짝수
홀수
짝수
홀수
짝수
홀수
짝수
홀수
짝수
홀수
페이지0
페이지1
페이지2
페이지3
페이지4
페이지5
페이지6
페이지7
페이지8
페이지9
페이지10
페이지11
페이지12
페이지13
페이지14
페이지15
페이지16
페이지17
페이지18
페이지19
페이지20

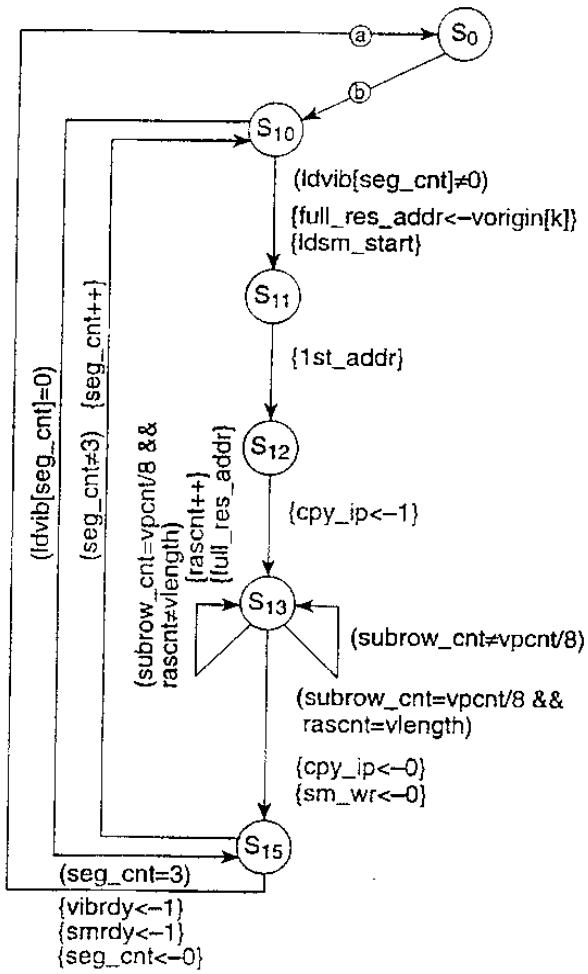
도면15a


401
18
(qtr res)
48
(full res)
100
VkLENGTH = 24
VkWIDTH = 2
VkPCNT = 16
VnLENGTH = 9
VnWIDTH = 1
VnY = VkY + 6
16 (full res)
8 (qtr res)

도면15

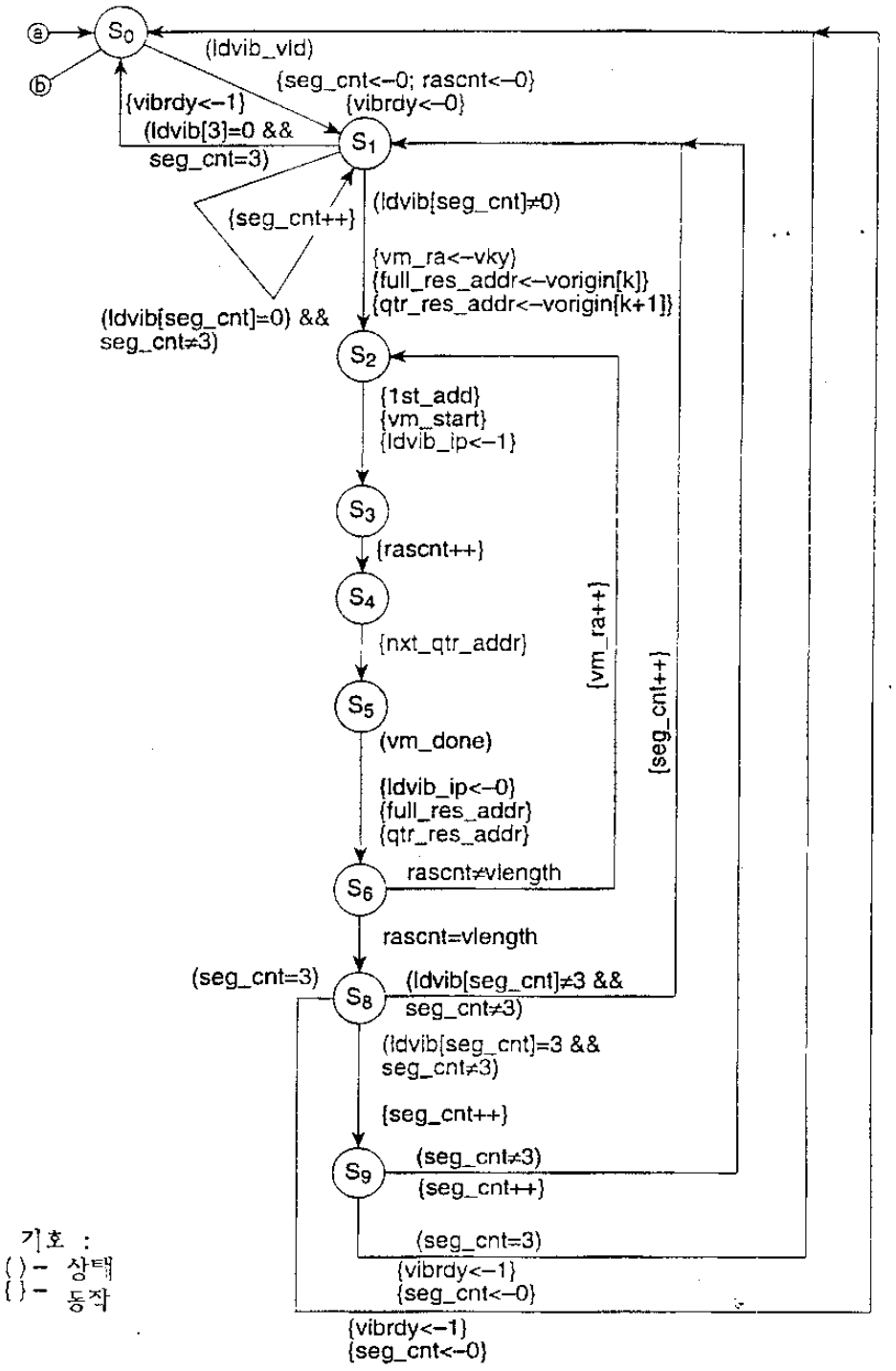

175
8
399
짝수
홀수
짝수
홀수
짝수
홀수
짝수
홀수
짝수
홀수
짝수
홀수
짝수
홀수
페이지 0
페이지 1
페이지 2
페이지 3
페이지 4
페이지 5
페이지 6
페이지 7
페이지 8
페이지 9
페이지 10
페이지 11
페이지 12
페이지 13
페이지 14
페이지 15
페이지 16
페이지 17
페이지 18
페이지 19
페이지 20
논리 메모리 좌표
짝수라인
홀수라인
페이지 0
페이지 1
페이지 2
페이지 3
페이지 4
페이지 5
페이지 6
페이지 7
페이지 3
페이지 9
페이지 10
페이지 11
페이지 12
페이지 13
페이지 14
페이지 15
페이지 16
페이지 17
페이지 18
페이지 19
페이지 20
페이지 0
페이지 1
페이지 2
페이지 3
페이지 4
페이지 5
페이지 6
페이지 7
페이지 8
페이지 9
페이지 10
페이지 11
페이지 12
페이지 13
페이지 14
페이지 15
페이지 16
페이지 17
페이지 18
페이지 19
페이지 20
물리 메모리 좌표

도면16a

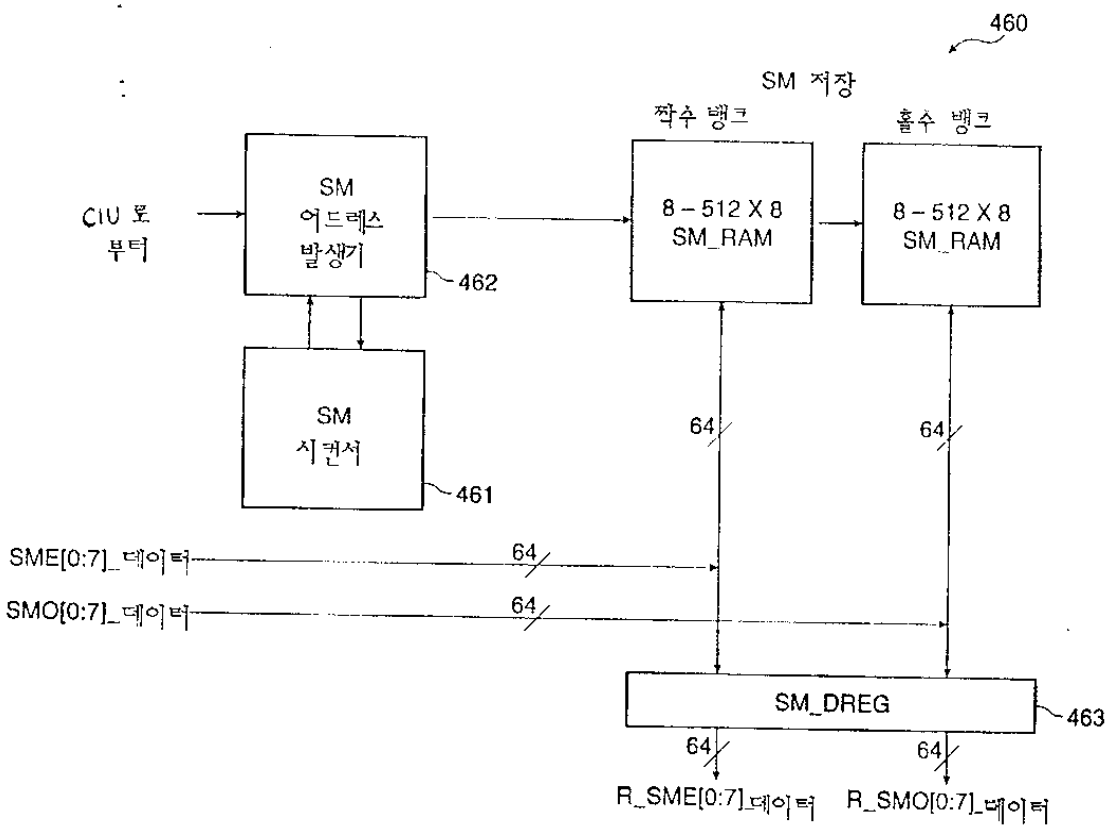


기호 :
( ) – 상태
{ } – 동작

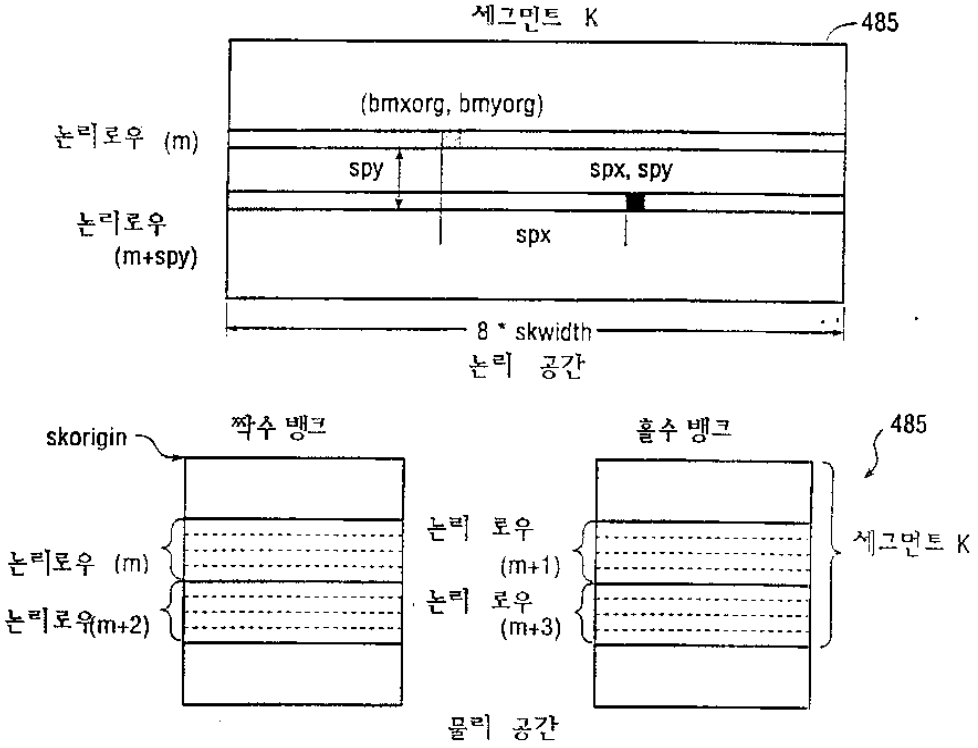
도면16b


ⓐ
ⓑ
S0
(ldvib_vld)
{seg_cnt<–0; rascnt<–0}
{vibrdy<–0}
{vibrdy<–1}
(ldvib[3]=0 &&
seg_cnt=3)
S1
{seg_cnt++}
(ldvib[seg_cnt]≠0)
{vm_ra<–vky}
{full_res_addr<–vorigin[k]}
{qtr_res_addr<–vorigin[k+1]}
(ldvib[seg_cnt]=0) &&
seg_cnt≠3)
S2
{1st_add}
{vm_start}
{ldvib_ip<–1}
S3
{rascnt++}
S4
{nxt_qtr_addr}
S5
(vm_done)
{ldvib_ip<–0}
{full_res_addr}
{qtr_res_addr}
{vm_ra++}
{seg_cnt++}
S6
rascnt≠vlength
rascnt=vlength
(seg_cnt=3)
S8
(ldvib[seg_cnt]≠3 &&
seg_cnt≠3)
(ldvib[seg_cnt]=3 &&
seg_cnt≠3)
{seg_cnt++}
S9
(seg_cnt≠3)
{seg_cnt++}
(seg_cnt=3)
{vibrdy<–1}
{seg_cnt<–0}
{vibrdy<–1}
{seg_cnt<–0}
기호 :
( ) – 상태
{ } – 동작

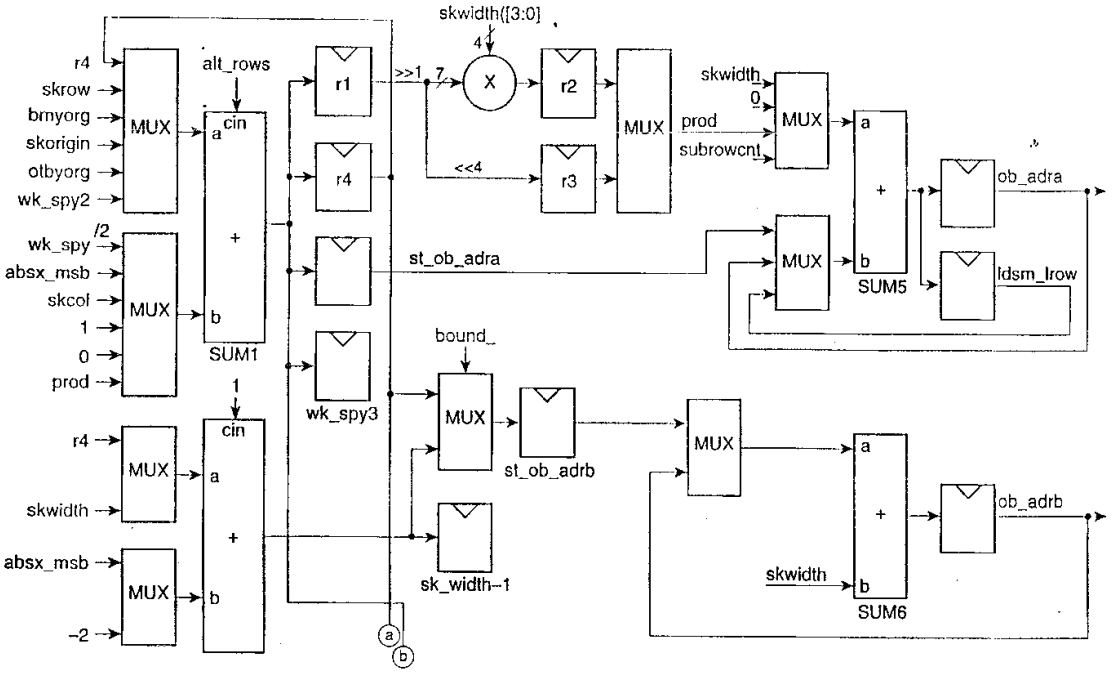
도면17


460
SM 저장
짝수 뱅크
홀수 뱅크
CIU 로부터
SM 어드레스 발생기
462
8-512 X 8 SM_RAM
8-512 X 8 SM_RAM
SM 시퀀서
461
64
64
SME[0:7]_데이터
64
SMO[0:7]_데이터
64
SM_DREG
463
64
64
R_SME[0:7]데이터
R_SMO[0:7]_데이터

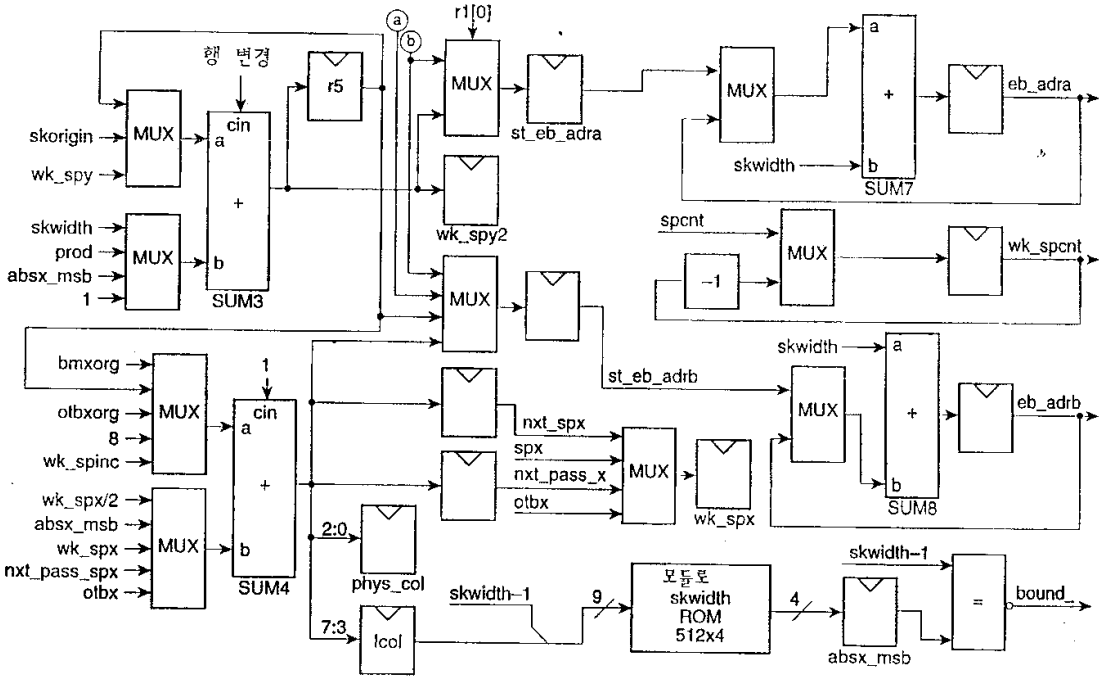
도면18


세그먼트 K
485
(bmxorg, bmyorg)
논리로우 (m)
spy
spx, spy
논리로우 (m+spy)
spx
8 * skwidth
논리 공간
skorigin
짝수 뱅크
홀수 뱅크
485
논리로우 (m)
논리로우(m+2)
논리 로우 (m+1)
논리 로우 (m+3)
세그먼트 K
물리 공간

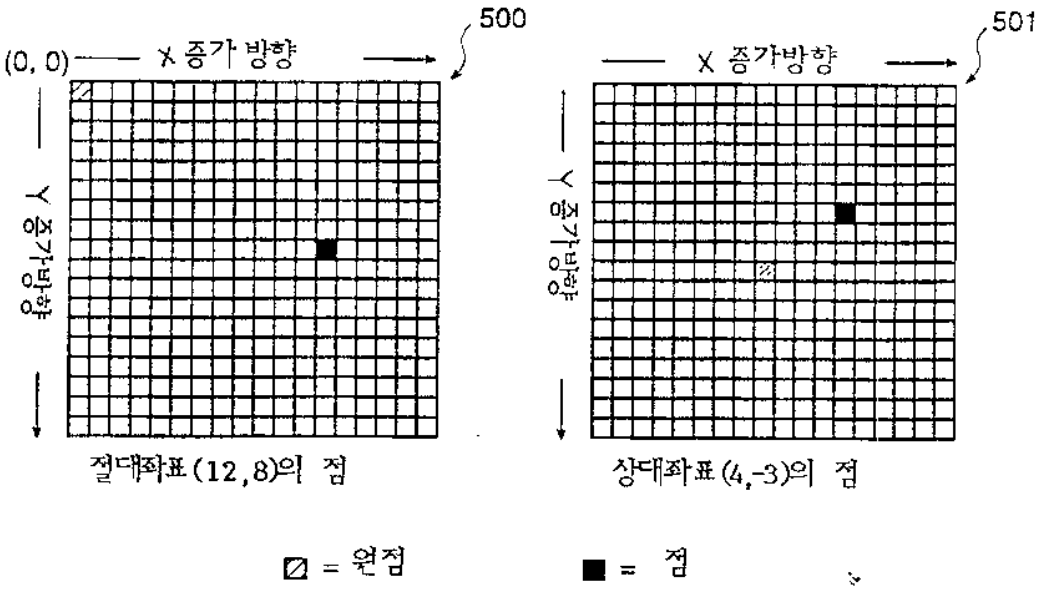
도면19a


skwidth[3:0]
4
r4
skrow
bmyorg
skorigin
otbyorg
wk_spy2
MUX
alt_rows
cin
a
+
b
SUM1
r1
>>1
7
X
r2
r4
<<4
r3
MUX
prod
subrowcnt
skwidth
0
MUX
a
+
b
SUM5
ob_adra
ldsm_lrow
/2
wk_spy
absx_msb
skcol
1
0
prod
MUX
st_ob_adra
wk_spy3
bound_
MUX
st_ob_adrb
MUX
a
+
b
SUM6
ob_adrb
skwidth
r4
skwidth
absx_msb
-2
MUX
1
cin
a
+
b
sk_width-1
a
b

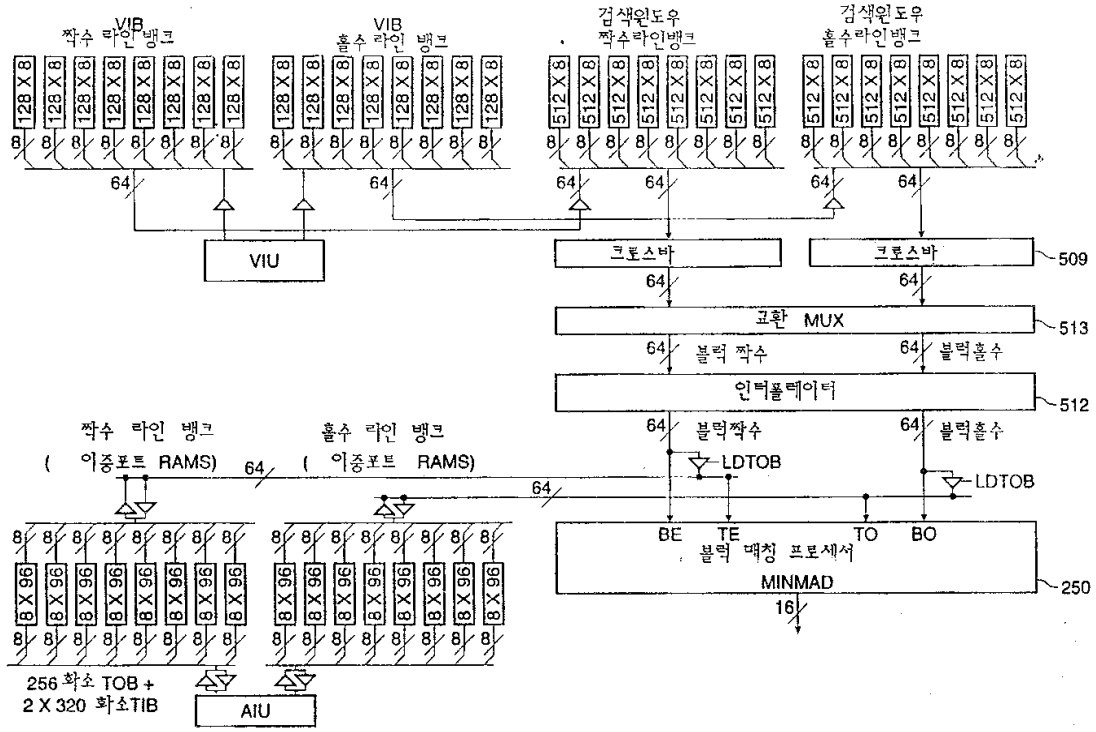
도면19b


a
b
r1[0]
행 변경
r5
skorigin
wk_spy
skwidth
prod
absx_msb
1
MUX
cin
a
+
b
SUM3
MUX
st_eb_adra
MUX
a
+
b
SUM7
eb_adra
skwidth
wk_spy2
MUX
spcnt
-1
MUX
wk_spcnt
bmxorg
otbxorg
8
wk_spinc
wk_spx/2
absx_msb
wk_spx
nxt_pass_spx
otbx
MUX
1
cin
a
+
b
SUM4
nxt_spx
spx
nxt_pass_x
otbx
MUX
wk_spx
st_eb_adrb
skwidth
MUX
a
+
b
SUM8
eb_adrb
2:0
phys_col
7:3
lcol
skwidth-1
9
모듈로
skwidth
ROM
512x4
4
absx_msb
skwidth-1
=
bound_

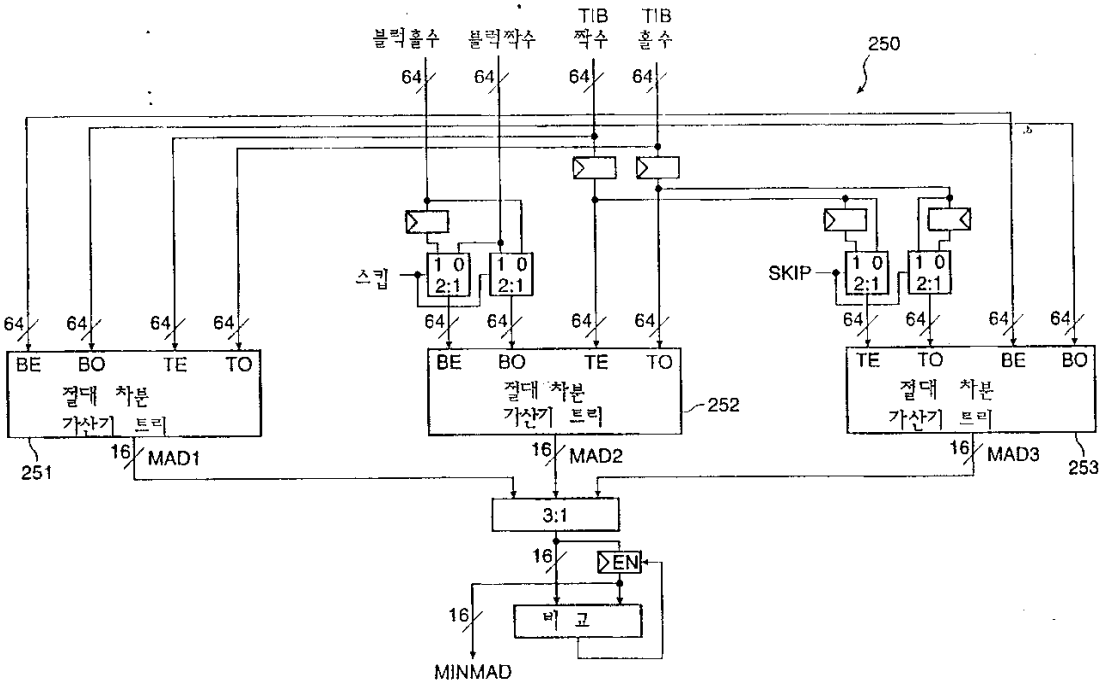
도면20


500
(0, 0)
X 증가 방향
Y 증가방향
절대좌표(12,8)의 점
501
X 증가방향
Y 증가방향
상대좌표(4,-3)의 점
= 원점
= 점

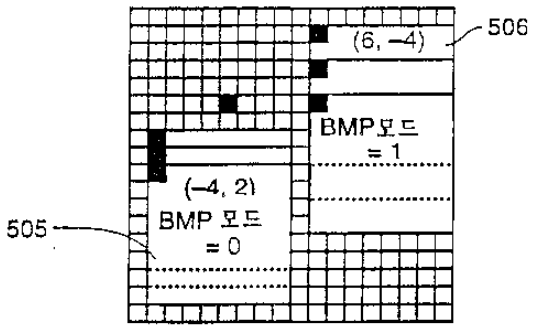
도면21


VIB
짝수 라인 뱅크
VIB
홀수 라인 뱅크
검색윈도우
짝수라인뱅크
검색윈도우
홀수라인뱅크
128 X 8
512 X 8
64
VIU
크로스바
509
교환 MUX
513
블럭 짝수
블럭홀수
인터폴레이터
512
LDTOB
BE
TE
TO
BO
블럭 매칭 프로세서
MINMAD
250
16
짝수 라인 뱅크
(이중포트 RAMS)
홀수 라인 뱅크
(이중포트 RAMS)
8 X 96
256 확소 TOB +
2 X 320 확소TIB
AIU

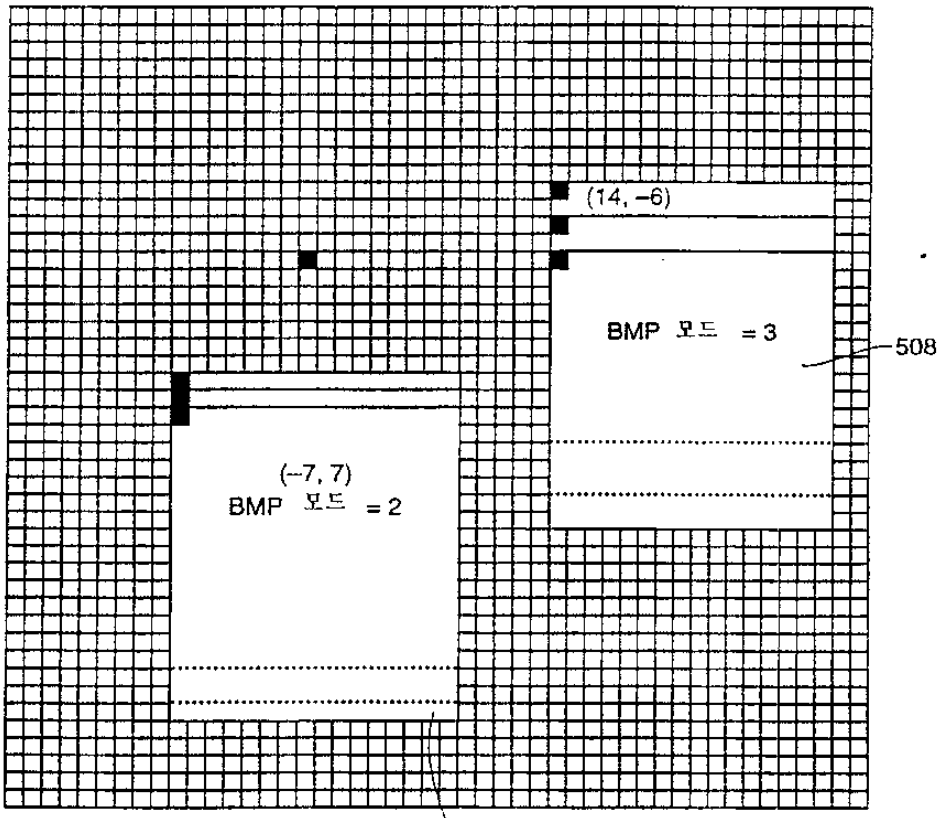
도면21a


블럭홀수
블럭짝수
TIB 짝수
TIB 홀수
250
64
스킵
SKIP
1 0 2:1
BE BO TE TO
절대 차분 가산기 트리
251
252
253
16 MAD1
16 MAD2
16 MAD3
3:1
EN
비 고
MINMAD

도면22



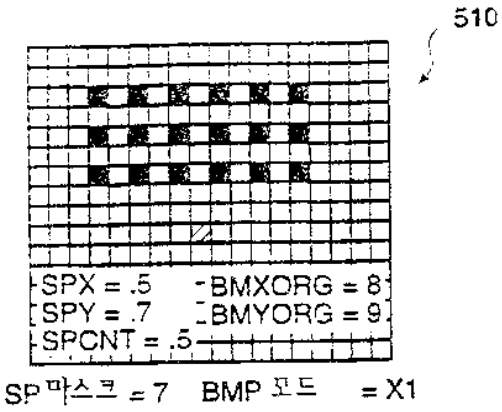
8 X 8 블럭 비교



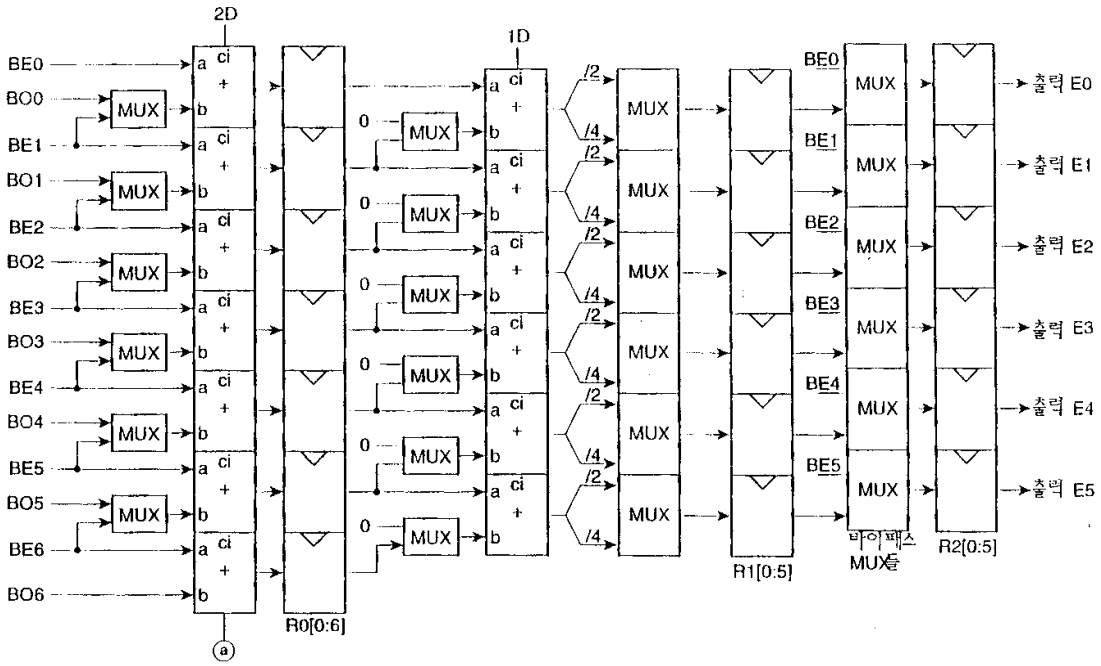
16 x 16 블럭 비교

■ =원점

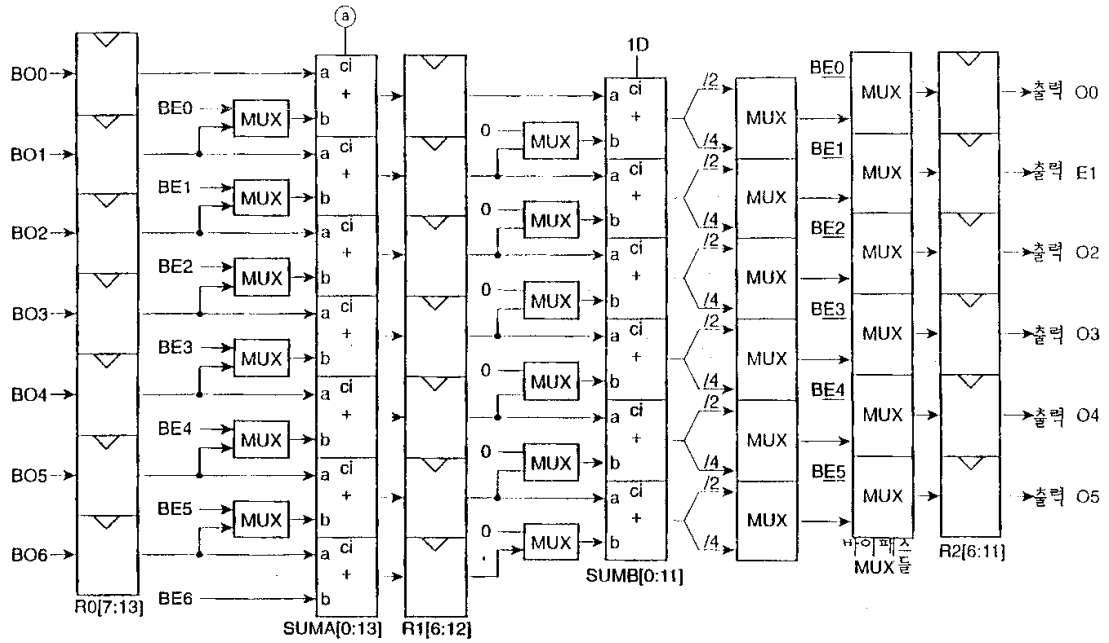
도면23


510
SPX = .5
SPY = .7
SPCNT = .5
BMXORG = 8
BMYORG = 9
SP 마스크 = 7
BMP 모드 = X1

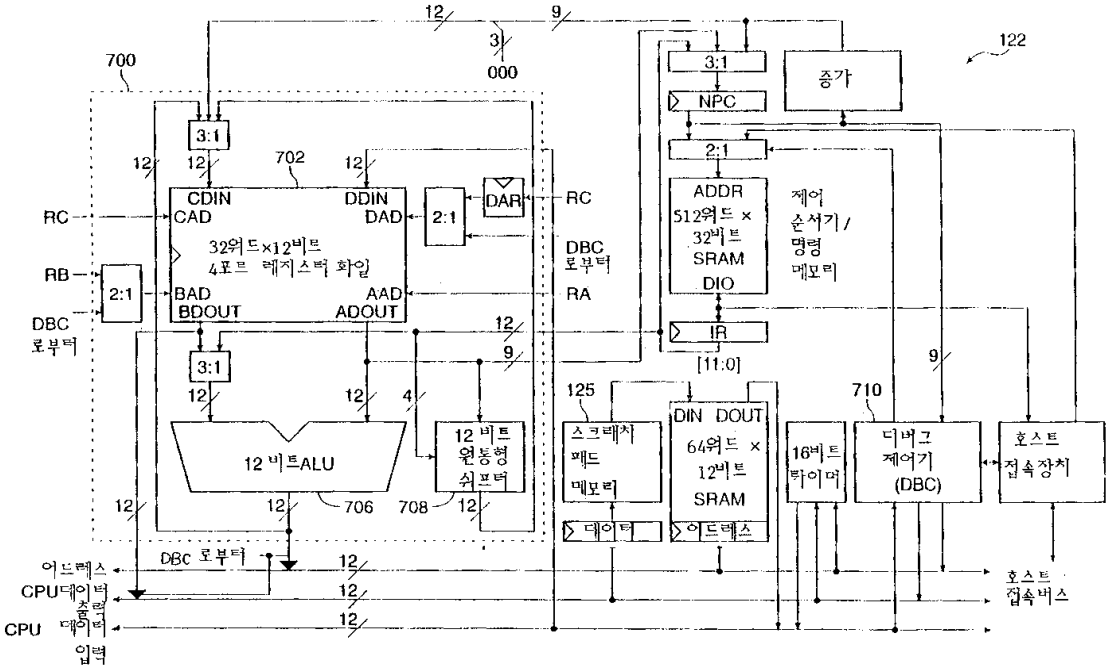
도면24a


2D
1D
BE0
BO0
BE1
BO1
BE2
BO2
BE3
BO3
BE4
BO4
BE5
BO5
BE6
BO6
MUX
R0[0:6]
R1[0:5]
R2[0:5]
/2
/4
바이패스 MUX들
출력 E0
출력 E1
출력 E2
출력 E3
출력 E4
출력 E5

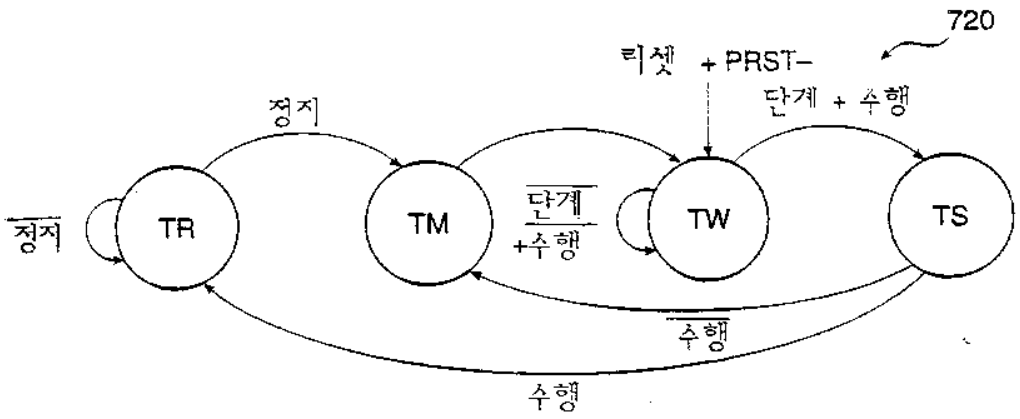
도면24b


BO0
BO1
BO2
BO3
BO4
BO5
BO6
R0[7:13]
BE0
BE1
BE2
BE3
BE4
BE5
BE6
MUX
SUMA[0:13]
R1[6:12]
1D
SUMB[0:11]
/2
/4
BE0
BE1
BE2
BE3
BE4
BE5
바이패스 MUX들
R2[6:11]
출력 O0
출력 E1
출력 O2
출력 O3
출력 O4
출력 O5

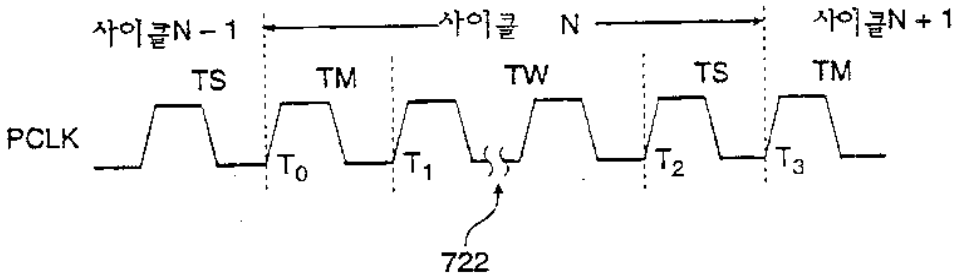
도면25


700
12
9
3
000
122
증가
3:1
NPC
2:1
ADDR
512워드 × 32비트 SRAM
DIO
제어 순서기 / 명령 메모리
IR
[11:0]
702
CDIN
CAD
DDIN
DAD
32워드×12비트 4포트 레지스터 화일
BAD
BDOUT
AAD
ADOUT
DAR
RC
RB
DBC 로부터
RA
12 비트ALU
12 비트 원통형 쉬프터
706
708
125
스크래치 패드 메모리
DIN DOUT
64워드 × 12비트 SRAM
데이터
어드레스
710
16비트 타이머
디버그 제어기 (DBC)
호스트 접속장치
호스트 접속버스
어드레스
CPU데이터 출력
CPU 데이터 입력

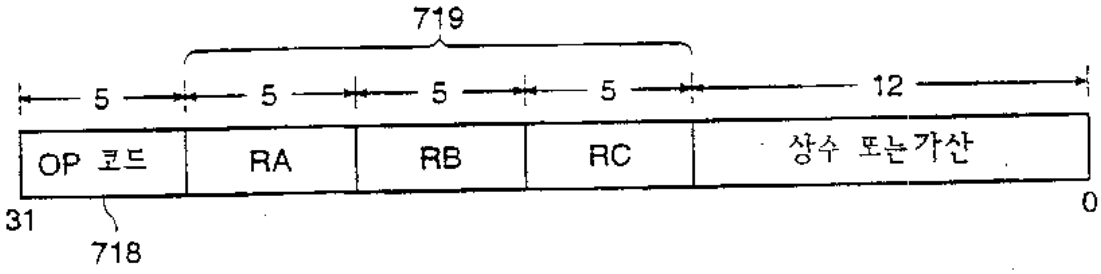
도면26



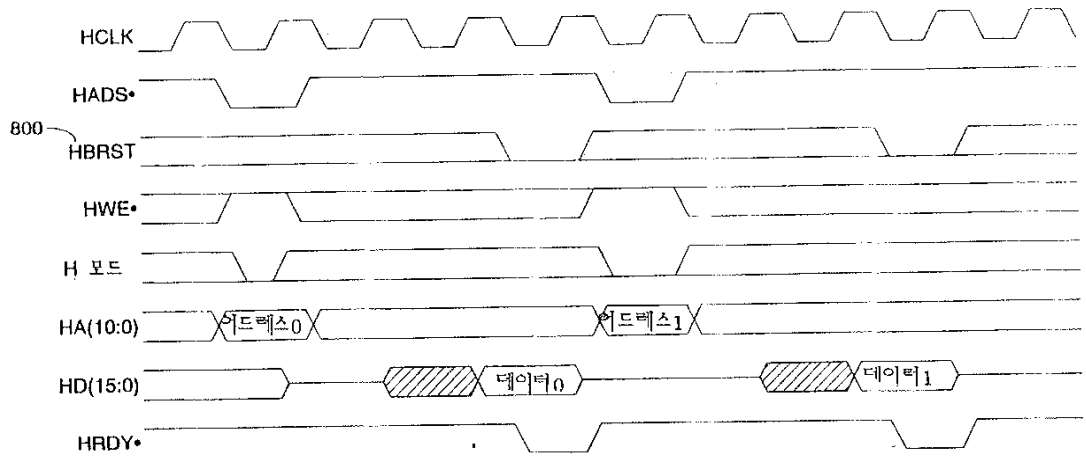
도면27



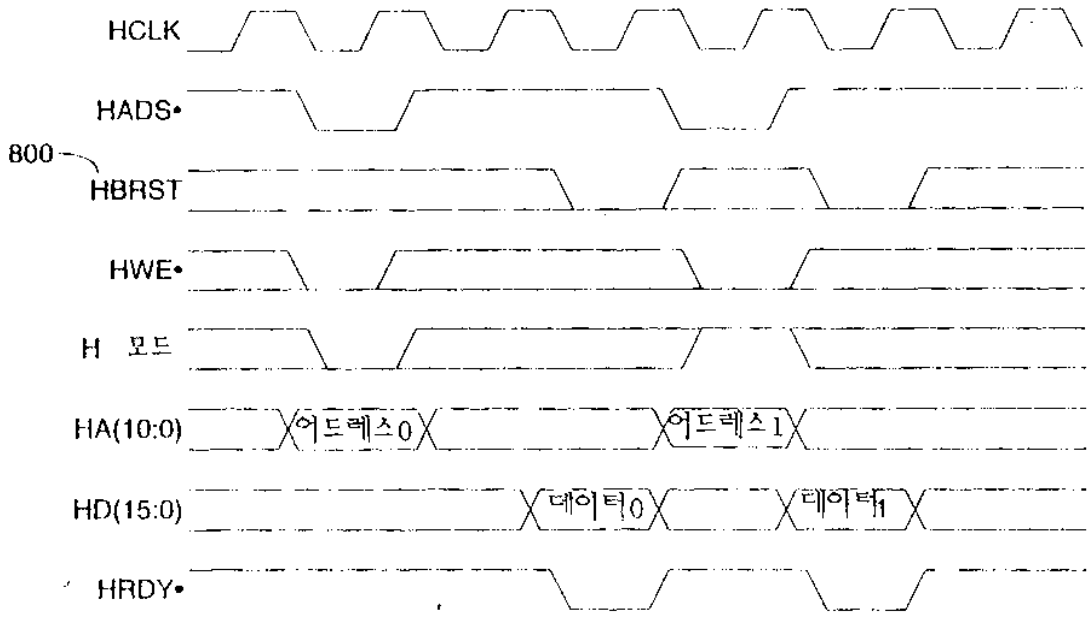
도면28



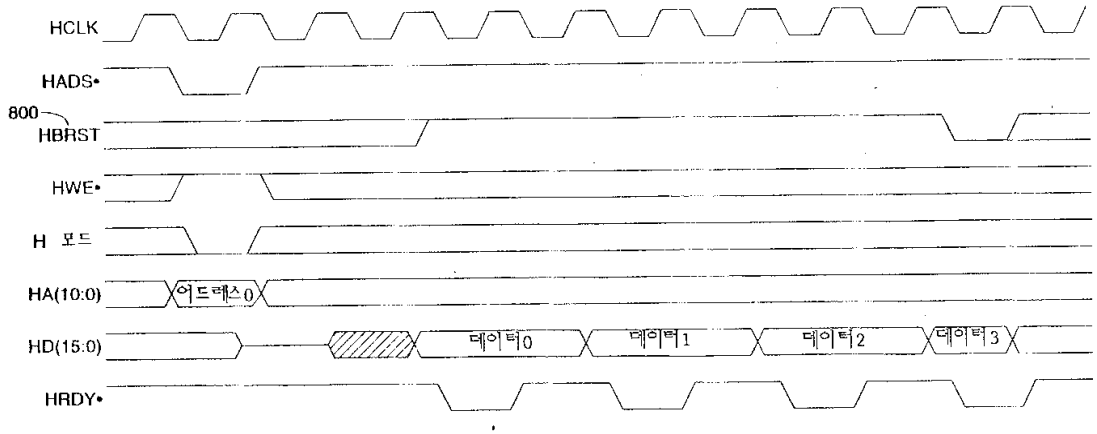
도면29



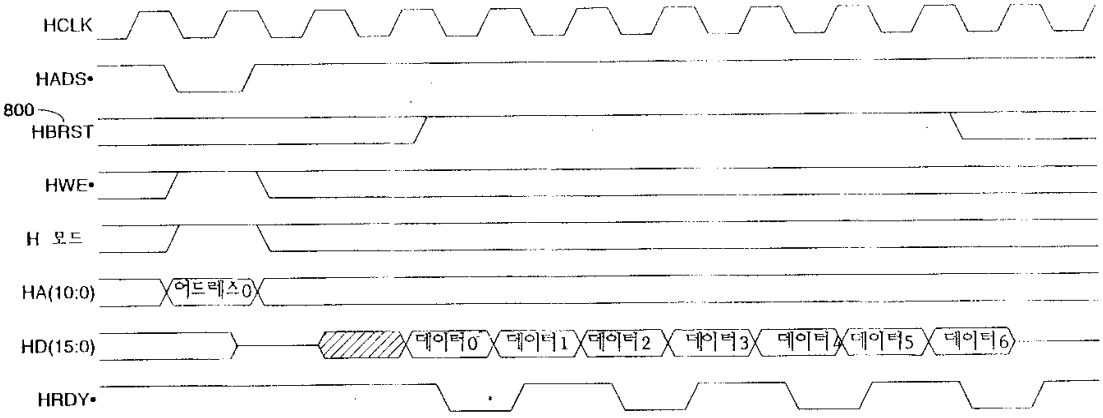
도면30



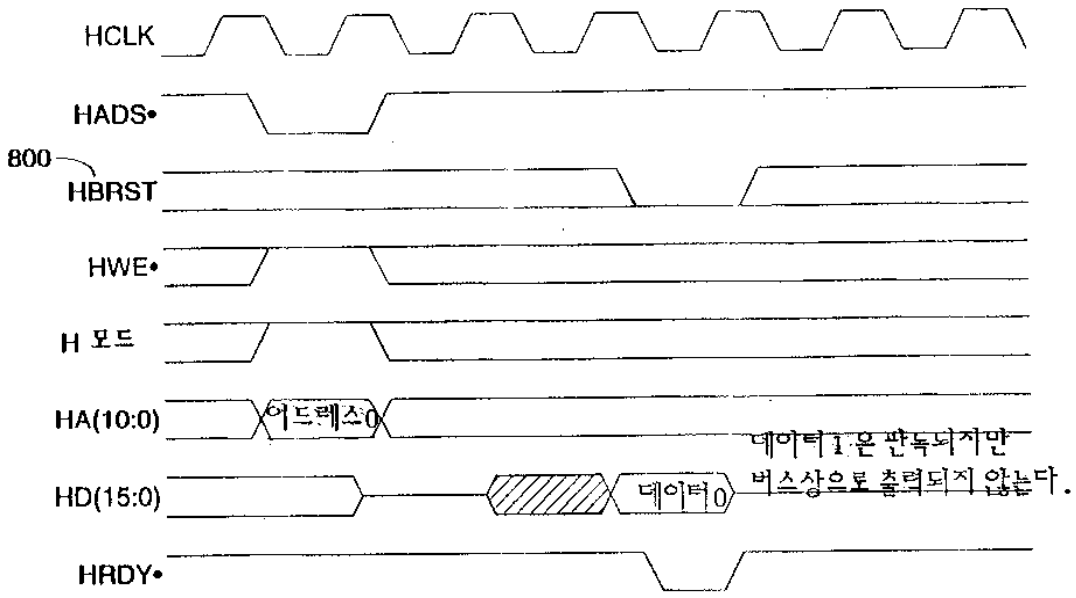
도면31



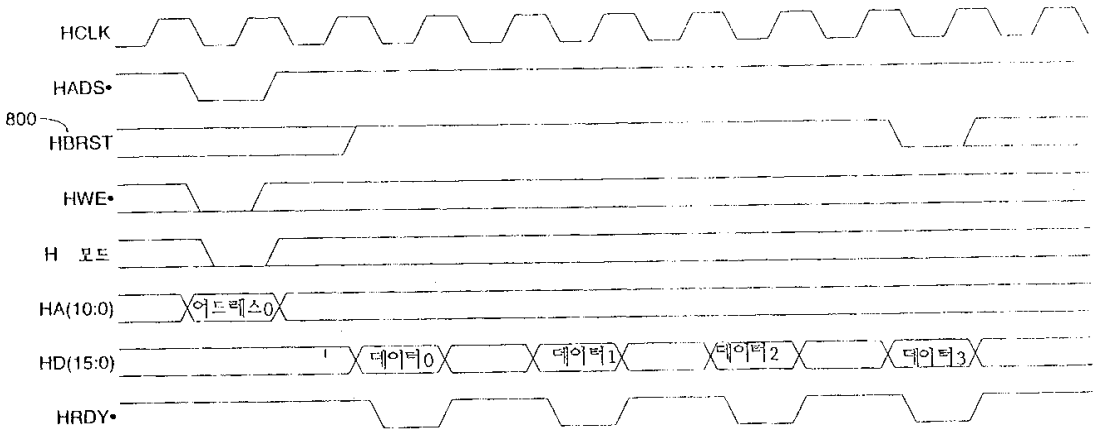
도면32



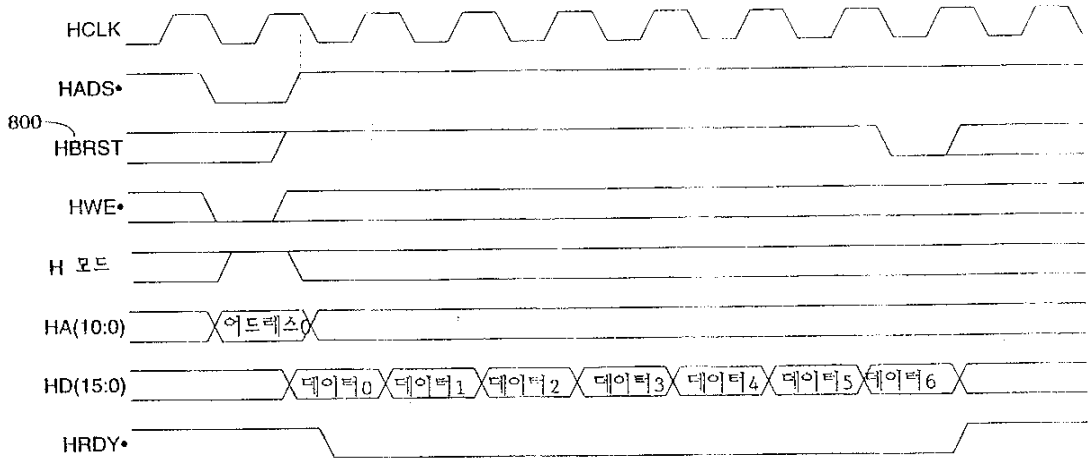
도면33



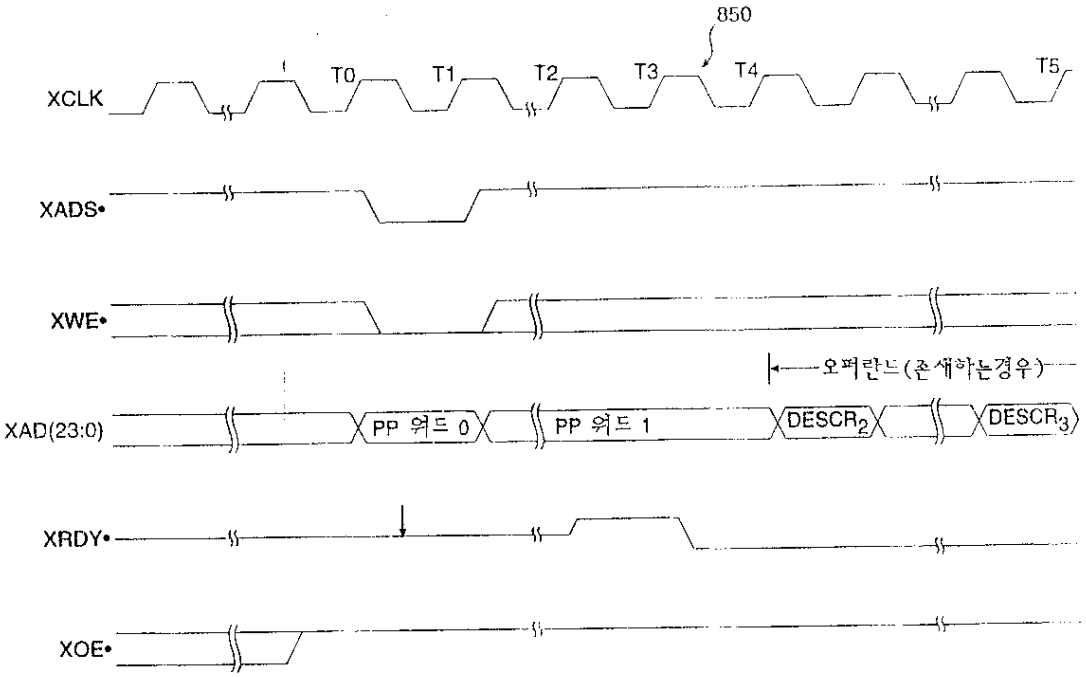
도면34



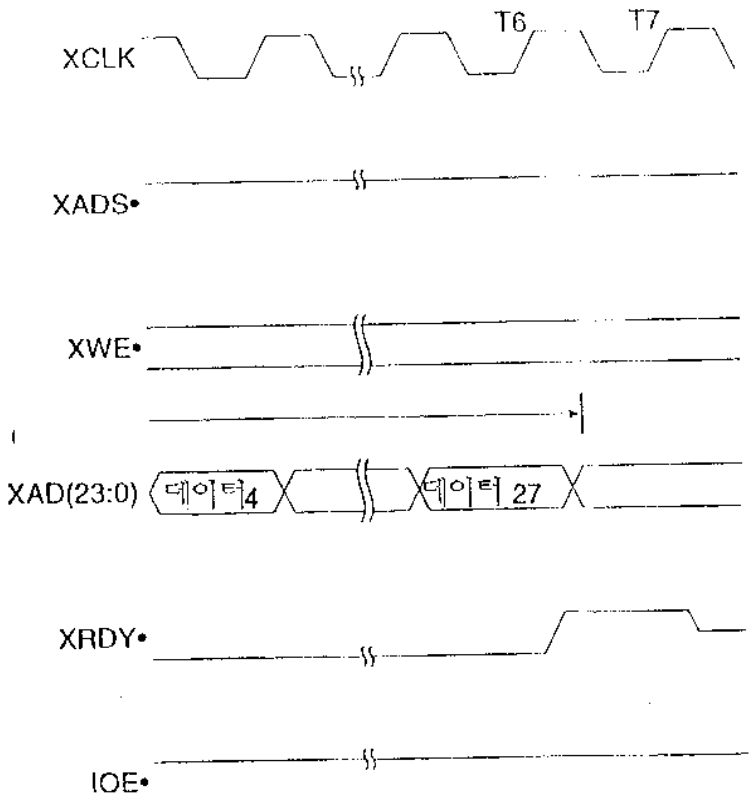
도면35



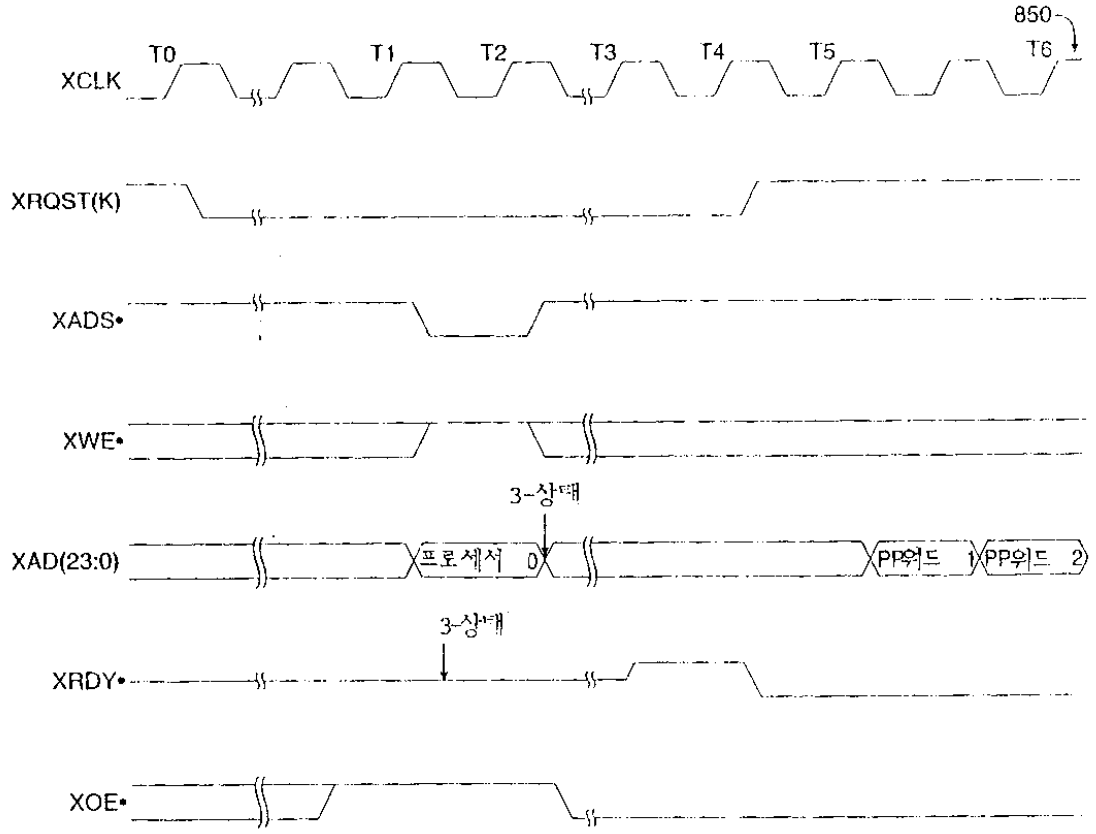
도면36a


850
T0
T1
T2
T3
T4
T5
XCLK
XADS•
XWE•
오퍼란드(존재하는경우)
XAD(23:0)
PP 워드 0
PP 워드 1
DESCR2
DESCR3
XRDY•
XOE•

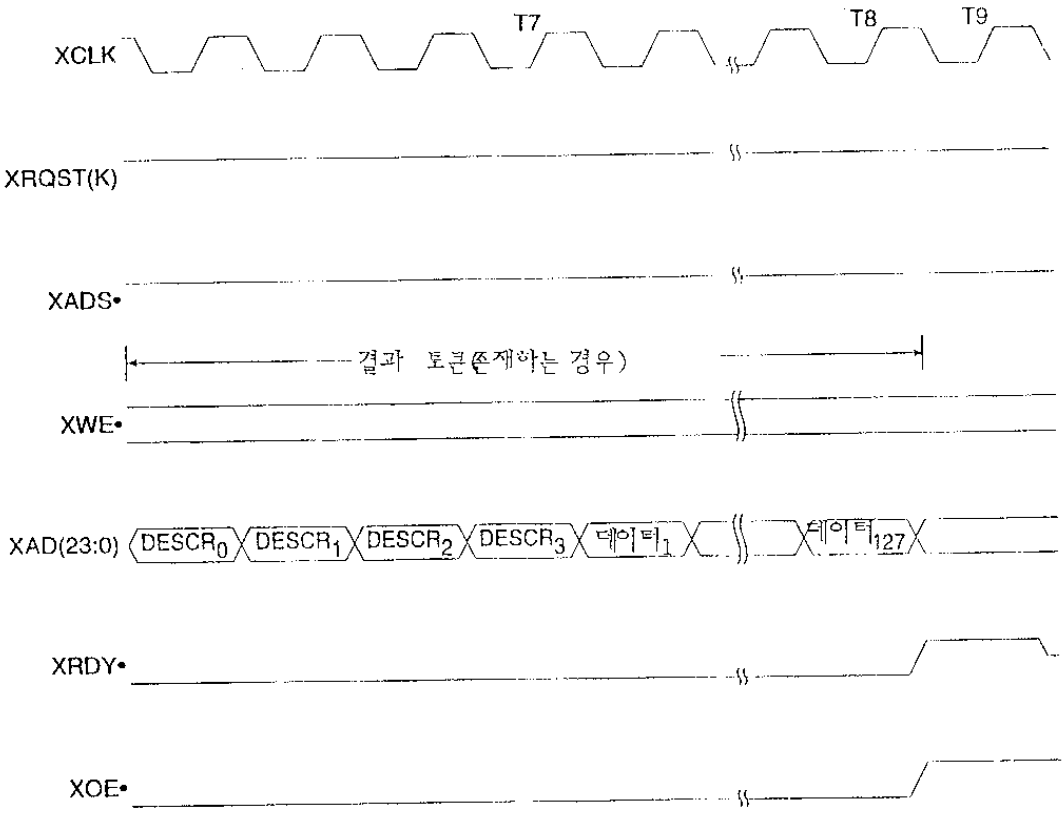
도면36b


T6
T7
XCLK
XADS•
XWE•
XAD(23:0)
데이터4
데이터 27
XRDY•
IOE•

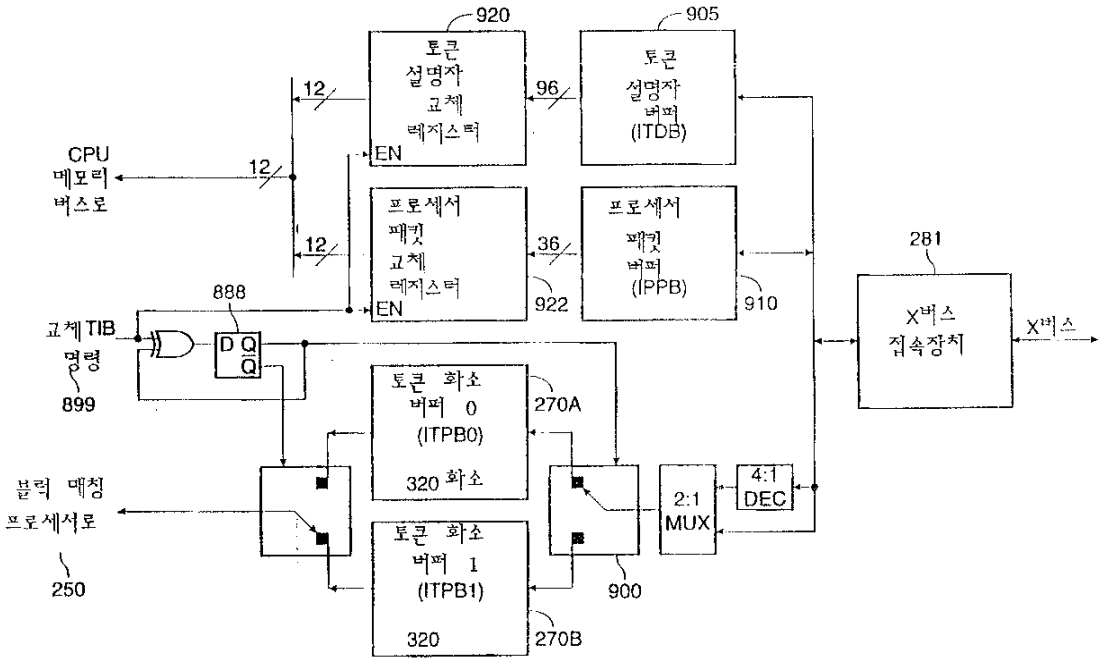
도면37a


850
T0
T1
T2
T3
T4
T5
T6
XCLK
XRQST(K)
XADS•
XWE•
3-상태
XAD(23:0)
프로세서 ID
PP워드 1
PP워드 2
3-상태
XRDY•
XOE•

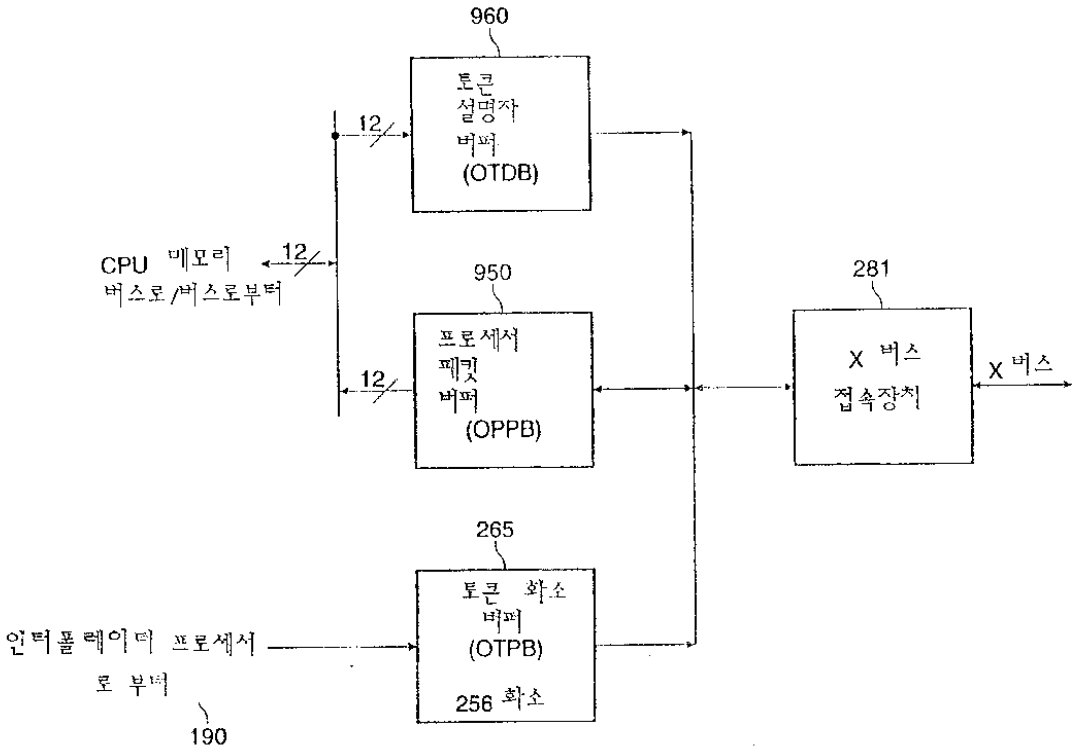
도면37b


T7
T8
T9
XCLK
XRQST(K)
XADS•
결과 토큰(존재하는 경우)
XWE•
XAD(23:0)
DESCR0
DESCR1
DESCR2
DESCR3
데이터1
데이터127
XRDY•
XOE•

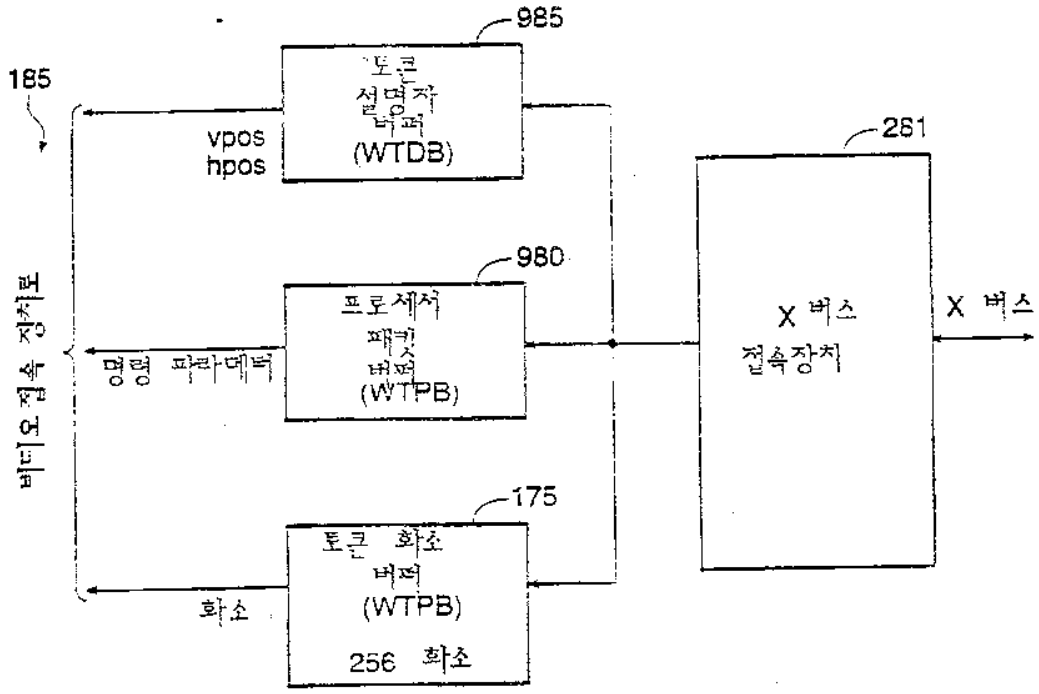
도면38


920
토큰 설명자 교체 레지스터
EN
905
토큰 설명자 버퍼 (ITDB)
96
12
CPU 메모리 버스로
12
12
프로세서 패킷 교체 레지스터
EN
922
36
프로세서 패킷 버퍼 (IPPB)
910
281
X버스 접속장치
X버스
888
교체TIB 명령
899
D Q
Q
토큰 확소 버퍼 0 (ITPB0)
320 확소
270A
블럭 매칭 프로세서로
250
토큰 확소 버퍼 1 (ITPB1)
320
270B
900
2:1 MUX
4:1 DEC

도면39


960
토큰 설명자 버퍼 (OTDB)
12
CPU 메모리 버스로/버스로부터
12
950
프로세서 패킷 버퍼 (OPPB)
12
281
X 버스 접속장치
X 버스
265
토큰 확소 버퍼 (OTPB)
256 확소
인터폴레이터 프로세서로 부터
190

도면40


185
985
토큰
설명자
버퍼
(WTDB)
vpos
hpos
281
980
프로세서
패킷
버퍼
(WTPB)
X 버스
접속장치
X 버스
명령 파라메터
비디오 접속 장치로
175
토큰 화소
버퍼
(WTPB)
256 화소
화소