(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2011년11월21일
(11) 등록번호 10-1084806
(24) 등록일자 2011년11월11일

(51) Int. Cl.
G06T 1/20 (2006.01) *G09G 5/36* (2006.01)
G09G 5/39 (2006.01)
(21) 출원번호 10-2011-7001766(분할)
(22) 출원일자(국제출원일자) 2005년11월14일
심사청구일자 2011년01월24일
(85) 번역문제출일자 2011년01월24일
(65) 공개번호 10-2011-0011758
(43) 공개일자 2011년02월08일
(62) 원출원 특허 10-2010-7018048
원출원일자(국제출원일자) 2005년11월14일
심사청구일자 2010년11월15일
(86) 국제출원번호 PCT/US2005/041329
(87) 국제공개번호 WO 2006/055546
국제공개일자 2006년05월26일
(30) 우선권주장
11/267,599 2005년11월04일 미국(US)
(뒷면에 계속)
(56) 선행기술조사문헌
US3614740 A
US4541046 A
전체 청구항 수 : 총 22 항

(73) 특허권자
엔비디아 코포레이션
미국 캘리포니아 95050 산타 클라라 산 토마스 익스프레스웨이 2701
(72) 발명자
가드레, 시리쉬
미국 94539 캘리포니아주 프레몬트 오막 스트리트 45302
카란디카르, 아시쉬
미국 94085 캘리포니아주 서니베일 아파트먼트 넘버108 아노 뉴보 애비뉴 395
(뒷면에 계속)
(74) 대리인
양영준, 백만기

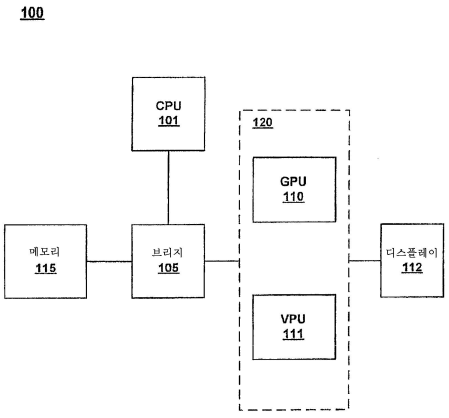
심사관 : 박상철

(54) 비디오 처리

(57) 요 약

비디오 처리 동작을 실행하기 위한 지연 감내 시스템이 설명된다. 스칼라 및 벡터 컴포넌트를 구비한 비디오 프로세서, 비디오 프로세서에서의 스트림 처리, 및 비디오 프로세서에서의 다차원 데이터 경로 처리도 설명된다.

대 표 도 - 도1



(72) 발명자

류, 스테펀

미국 94085 캘리포니아주 서니베일 아나카파 테라스 627

칭, 크리스토퍼, 티.

미국 95051 캘리포니아주 산타 클라라 로욜라 드라이브 1232

(30) 우선권주장

11/267,638 2005년11월04일 미국(US)
11/267,700 2005년11월04일 미국(US)
11/267,875 2005년11월04일 미국(US)
60/628,414 2004년11월15일 미국(US)

특허청구의 범위

청구항 1

비디오 처리 동작들을 실행하기 위한 지연 감내 시스템(latency tolerant system)으로서,

비디오 프로세서와 호스트 CPU 간의 통신을 구현하기 위한 호스트 인터페이스;

상기 호스트 인터페이스에 연결되고 스칼라 비디오 처리 동작들을 실행하도록 구성된 스칼라 실행 유닛;

상기 호스트 인터페이스에 연결되고 벡터 비디오 처리 동작들을 실행하도록 구성된 벡터 실행 유닛;

커멘드 FIFO에 액세스함으로써 상기 벡터 실행 유닛이 요구 구동 기반(demand driven basis)으로 동작하는 것을 가능하게 하기 위한 커맨드 FIFO - 상기 벡터 실행 유닛은 상기 요구 구동 기반에 기초하여 상기 벡터 실행 유닛의 수행 레벨(performance level)을 조정하도록 동작할 수 있음 -;

상기 비디오 프로세서와 프레임 버퍼 메모리 간의 통신을 구현하기 위한 메모리 인터페이스; 및

상기 메모리 인터페이스 내에 구축되고, 복수의 상이한 메모리 위치들 간의 DMA 전달들을 구현하며, 상기 벡터 실행 유닛에 대한 데이터 및 명령들을 데이터 저장 메모리 및 명령 캐시에 로딩하기 위한 DMA 엔진

을 포함하는 지연 감내 시스템.

청구항 2

제1항에 있어서,

상기 벡터 실행 유닛은 상기 요구 구동 기반으로 동작하기 위해 상기 커맨드 FIFO를 액세스함으로써 상기 스칼라 실행 유닛에 관하여 비동기적으로 동작하도록 구성되는, 지연 감내 시스템.

청구항 3

제1항에 있어서,

상기 요구 구동 기반은 상기 상이한 메모리 위치들에서 상기 벡터 실행 유닛의 상기 커맨드 FIFO로의 데이터 전달의 지연을 숨기도록 구성되는, 지연 감내 시스템.

청구항 4

제1항에 있어서,

상기 스칼라 실행 유닛은 알고리즘 흐름 제어 처리(algorithm flow control processing)를 구현하도록 구성되는, 지연 감내 시스템.

청구항 5

제4항에 있어서,

상기 스칼라 실행 유닛은 데이터 전달 지연을 숨기기 위하여 상기 벡터 실행 유닛에 대한 작업 파라미터들을 사전 계산하도록 구성되는, 지연 감내 시스템.

청구항 6

제1항에 있어서,

상기 벡터 실행 유닛은 벡터 서브 루틴의 후속 실행을 위한 커맨드들을 프리페치하기 위해 상기 DMA 엔진을 통해 메모리 판독을 스케쥴링하도록 구성되는, 지연 감내 시스템.

청구항 7

제6항에 있어서,

상기 메모리 판독은 상기 스칼라 실행 유닛에 의한 상기 벡터 서브 루틴의 호출들 이전에 상기 벡터 서브 루틴

의 실행을 위한 커맨드들을 프리페치하도록 스케쥴링되는, 지연 감내 시스템.

청구항 8

지연 감내 비디오 처리 동작들을 위한 방법으로서,

호스트 인터페이스를 이용하여 비디오 프로세서와 호스트 CPU 간의 통신을 구현하는 단계;

상기 호스트 인터페이스에 연결된 스칼라 실행 유닛을 이용하여 스칼라 비디오 처리 동작들을 실행하는 단계;

상기 호스트 인터페이스에 연결된 벡터 실행 유닛을 이용하여 벡터 비디오 처리 동작들을 실행하는 단계;

메모리 커맨드 FIFO를 액세스함으로써 상기 벡터 실행 유닛이 요구 구동 기반으로 동작할 수 있게 하는 단계 - 상기 벡터 실행 유닛은 상기 요구 구동 기반에 기초하여 상기 벡터 실행 유닛의 수행 레벨을 조정하도록 동작할 수 있음 -;

메모리 인터페이스를 이용하여 상기 비디오 프로세서와 프레임 버퍼 메모리 간의 통신을 구현하는 단계; 및

상기 메모리 인터페이스 내에 구축되고 상기 벡터 실행 유닛에 대한 데이터 및 명령들을 데이터 저장 메모리 및 명령 캐시에 로딩하도록 구성된 DMA 엔진을 이용하여 복수의 상이한 메모리 위치들 간의 DMA 전달들을 구현하는 단계

를 포함하는 지연 감내 비디오 처리 동작들을 위한 방법.

청구항 9

제8항에 있어서,

상기 벡터 실행 유닛은 상기 요구 구동 기반으로 동작하기 위하여 상기 메모리 커맨드 FIFO를 액세스함으로써 상기 스칼라 실행 유닛에 관하여 비동기적으로 동작하도록 구성되는, 지연 감내 비디오 처리 동작들을 위한 방법.

청구항 10

제8항에 있어서,

상기 요구 구동 기반은 상기 상이한 메모리 위치들에서 상기 벡터 실행 유닛의 상기 메모리 커맨드 FIFO로의 데이터 전달의 지연을 숨기도록 구성되는, 지연 감내 비디오 처리 동작들을 위한 방법.

청구항 11

제8항에 있어서,

상기 스칼라 실행 유닛은 알고리즘 흐름 제어 처리를 구현하도록 구성되는, 지연 감내 비디오 처리 동작들을 위한 방법.

청구항 12

제11항에 있어서,

상기 스칼라 실행 유닛은 데이터 전달 지연을 숨기기 위하여 상기 벡터 실행 유닛에 대한 작업 파라미터들을 사전 계산하도록 구성되는, 지연 감내 비디오 처리 동작들을 위한 방법.

청구항 13

제8항에 있어서,

상기 벡터 실행 유닛은 벡터 서브 루틴의 후속 실행을 위한 커맨드들을 프리페치하기 위해 상기 DMA 엔진을 통해 메모리 판독을 스케쥴링하도록 구성되는, 지연 감내 비디오 처리 동작들을 위한 방법.

청구항 14

제13항에 있어서,

상기 메모리 판독은 상기 스칼라 실행 유닛에 의한 상기 벡터 서브 루틴의 호출들 이전에 상기 벡터 서브 루틴의 실행을 위한 커맨드들을 프리페치하도록 스케쥴링되는, 지연 감내 비디오 처리 동작들을 위한 방법.

청구항 15

비디오 처리 동작들을 실행하기 위한 지연 감내 시스템으로서,

마더보드;

상기 마더보드에 연결된 호스트 CPU;

상기 마더보드 및 상기 호스트 CPU에 연결된 비디오 프로세서;

비디오 프로세서와 상기 호스트 CPU 간의 통신을 구현하기 위한 호스트 인터페이스;

상기 호스트 인터페이스에 연결되고 스칼라 비디오 처리 동작들을 실행하도록 구성된 스칼라 실행 유닛;

상기 호스트 인터페이스에 연결되고 벡터 비디오 처리 동작들을 실행하도록 구성된 벡터 실행 유닛;

커맨드 FIFO에 액세스함으로써 상기 벡터 실행 유닛이 요구 구동 기반으로 동작하는 것을 가능하게 하기 위한 커맨드 FIFO - 상기 벡터 실행 유닛은 상기 요구 구동 기반에 기초하여 상기 벡터 실행 유닛의 수행 레벨을 조정하도록 동작할 수 있음 -;

상기 비디오 프로세서와 프레임 버퍼 메모리 간의 통신을 구현하기 위한 메모리 인터페이스; 및

상기 메모리 인터페이스 내에 구축되고, 복수의 상이한 메모리 위치들 간의 DMA 전달들을 구현하며, 상기 벡터 실행 유닛에 대한 데이터 및 명령들을 데이터 저장 메모리 및 명령 캐시에 로딩하기 위한 DMA 엔진

을 포함하는 지연 감내 시스템.

청구항 16

제15항에 있어서,

상기 벡터 실행 유닛은 상기 요구 구동 기반으로 동작하기 위해 상기 커맨드 FIFO를 액세스함으로써 상기 스칼라 실행 유닛에 관하여 비동기적으로 동작하도록 구성되는, 지연 감내 시스템.

청구항 17

제15항에 있어서,

상기 요구 구동 기반은 상기 상이한 메모리 위치들에서 상기 벡터 실행 유닛의 상기 커맨드 FIFO로의 데이터 전달의 지연을 숨기도록 구성되는, 지연 감내 시스템.

청구항 18

제15항에 있어서,

상기 스칼라 실행 유닛은 알고리즘 흐름 제어 처리를 구현하도록 구성되는, 지연 감내 시스템.

청구항 19

제18항에 있어서,

상기 스칼라 실행 유닛은 데이터 전달 지연을 숨기기 위하여 상기 벡터 실행 유닛에 대한 작업 파라미터들을 사전 계산하도록 구성되는, 지연 감내 시스템.

청구항 20

제15항에 있어서,

상기 벡터 실행 유닛은 벡터 서브 루틴의 후속 실행을 위한 커맨드들을 프리페치하기 위해 상기 DMA 엔진을 통해 메모리 판독을 스케쥴링하도록 구성되고, 상기 메모리 판독은 상기 스칼라 실행 유닛에 의한 상기 벡터 서브 루틴의 호출들 이전에 상기 벡터 서브 루틴의 실행을 위한 커맨드들을 프리페치하도록 스케쥴링되는, 지연 감내

시스템.

청구항 21

제15항에 있어서,

상기 벡터 실행 유닛은 상기 커맨드 FIFO에 기초하여 제1 전력 레벨 및 제2 전력 레벨에서 동작할 수 있고, 상기 제2 전력 레벨은 상기 제1 전력 레벨보다 적은 전력을 사용하는, 지연 감내 시스템.

청구항 22

제15항에 있어서,

상기 벡터 지연 유닛은 상기 커맨드 FIFO의 컨텐츠들에 기초하여 상기 요구 구동 기반으로 동작할 수 있는, 지연 감내 시스템.

명 세 서

기 술 분 야

[0001] 본 발명은 디지털 전자 컴퓨터 시스템에 관한 것이다. 구체적으로는, 본 발명은 컴퓨터 시스템 상에서 비디오 정보를 효율적으로 처리하기 위한 시스템에 관한 것이다. 일 양태에서는, 비디오 처리 동작을 실행하기 위한 지연 감내 시스템이 기술된다. 다른 양태에서는, 비디오 프로세서에서의 스트림 처리가 기술된다. 또한, 비디오 프로세서에서의 다차원 데이터 경로 처리가 기술된다. 스칼라 및 벡터 컴포넌트들을 구비한 비디오 프로세서도 기술된다.

배 경 기 술

[0002] 본 출원은 "비디오 처리 방법 및 시스템(A METHOD AND SYSTEM FOR VIDEO PROCESSING)"라는 명칭으로 가드레(Gadre) 등에 의해 2004년 11월 15일자로 출원된 미국 가출원 번호 제60/628,414호의 35 U.S.C. Section 119(e) 하의 이익을 청구하며, 이 가출원의 전체 내용은 본 명세서에 포함되어 있다.

[0003] 이미지 및 풀-모션 비디오의 디스플레이는 최근에 크게 발전하고 있는 전자 산업의 한 영역이다. 고품질 비디오, 특히 고선명도 디지털 비디오의 디스플레이 및 렌더링은 현대 비디오 기술 애플리케이션 및 장치의 주 목표이다. 비디오 기술은 셀룰러 전화, 개인용 비디오 레코더, 디지털 비디오 프로젝터, 고선명도 텔레비젼 등의 다양한 제품에 이용된다. 고선명 비디오 생성 및 디스플레이가 가능한 장치들의 출현 및 성장 보급은 고도의 혁신 및 진보를 겪는 전자 산업의 한 영역이다.

[0004] 많은 소비자 전자 타입 및 전문가 레벨 장치들로 보급되는 비디오 기술은 디스플레이를 위해 비디오 신호를 포맷 및/또는 향상시키기 위하여 하나 이상의 비디오 프로세서에 의존한다. 이것은 특히 디지털 비디오 애플리케이션에 대해 사실이다. 예를 들어, 하나 이상의 비디오 프로세서는 일반적인 셋톱 박스 내에 포함되고, HDTV 방송 신호를 디스플레이에 의해 이용 가능한 비디오 신호로 변환하는 데 이용된다. 이러한 변환은 예를 들어 비디오 신호가 트루 16x9(예를 들어, 와이드 스크린) 디스플레이 상의 적절한 디스플레이를 위해 비(non-) 16x9 비디오 이미지로부터 변환되는 스케일링을 포함한다. 하나 이상의 비디오 프로세서는, 비디오 신호가 홀수 및 짝수 스캔 라인들이 개별적으로 디스플레이되는 인터레이스 포맷에서 전체 프레임이 단일 스위프 내에 묘화되는 진행 포맷으로 변환되는 스캔 변환을 수행하는 데 이용될 수 있다.

[0005] 비디오 프로세서 애플리케이션의 추가 예는 예를 들어 비디오 신호가 압축 포맷(예를 들어, MPEG-2)으로 수신되고 디스플레이를 위해 압축 해제되고 포맷되는 신호 압축 해제를 포함한다. 또 하나의 예는 리인터레이싱 스캔 변환(re-interlacing scan conversion)인데, 이는 수신 디지털 비디오 신호를 DVI(Digital Visual Interface) 포맷에서 시장에 설치된 많은 수의 구식 텔레비젼 디스플레이와 호환 가능한 합성 비디오 포맷으로 변환하는 것을 포함한다.

[0006] 보다 세련된 사용자들은 예를 들어 루프내/루프외 디블록킹 필터, 개량된 모션 적응 디인터레이싱, 인코딩 동작을 위한 입력 잡음 필터링, 다위상 스케일링/리샘플링, 서브-픽처 합성, 및 칼라 공간 변환, 조정, 픽셀 포인트 조작(예를 들어, 샤프닝, 히스토그램 조정 등) 및 다양한 비디오 표면 포맷 변환 지원 동작과 같은 프로세서-증폭기 동작들과 같은 보다 정교한 비디오 프로세서 기능들을 필요로 한다.

발명의 내용

해결하려는 과제

[0007] 이러한 정교한 비디오 프로세서 기능을 제공하는 것과 관련된 문제는 이러한 기능을 구현하기에 충분히 강력한 아키텍처를 가진 비디오 프로세서가 많은 타입의 장치에 포함되기에는 과도하게 비용이 많이 들 수 있다는 것이다. 비디오 처리 기능이 더 정교할수록, 이러한 기능을 구현하는 데 필요한 집적 회로는 실리콘 다이 면적, 트랜지스터 수, 메모리 속도 요건 등의 면에서 더 비용이 증가할 것이다.

[0008] 따라서, 종래의 시스템 설계자들은 비디오 프로세서 성능과 비용에 관련하여 절충점을 찾아야만 했다. 수용 가능한 비용/성능 비를 갖는 것으로 널리 간주되는 종래의 비디오 프로세서는 종종 지연 제한(예를 들어, 비디오 떨림 또는 비디오 처리 애플리케이션의 기능 정지를 방지하기 위해) 및 연산 밀도(예를 들어, 다이(die) 제곱 밀리미터당 프로세서 연산 횟수) 면에서 거의 충분하지 못했었다. 더욱이, 종래의 비디오 프로세서들은 일반적으로, 비디오 장치가 다수의 비디오 스트림을 처리(예를 들어, 다수의 수신 스트림 및 발신 디스플레이 스트림의 동시 처리)할 것으로 예상되는 경우에서와 같이 선형 스케일링 성능 요건에 적합하지 않다.

[0009] 따라서, 종래 기술의 한계를 극복하는 새로운 비디오 프로세서 시스템이 필요하다. 새로운 비디오 프로세서 시스템은 더욱 세련된 사용자들에 의해 기대되는 정교한 비디오 프로세서 기능들을 처리하기 위해 스케일링이 가능해야 하고, 높은 연산 밀도를 가져야 한다.

과제의 해결 수단

[0010] 본 발명의 실시예들은 집적 회로 실리콘 다이 면적, 트랜지스터 수, 메모리 속도 요건 등을 효율적으로 이용하면서 정교한 비디오 처리 기능들을 지원하는 새로운 비디오 프로세서 시스템을 제공한다. 본 발명의 실시예들은 다수의 비디오 스트림을 처리하기 위해 높은 연산 밀도를 유지하며, 쉽게 스케일링이 가능하다.

[0011] 일 실시예에서는, 비디오 프로세서에서 비디오 처리 동작들을 실행하기 위한 지연 감내 시스템이 구현된다. 이 시스템은 비디오 프로세서와 호스트 CPU 간의 통신을 구현하기 위한 호스트 인터페이스, 호스트 인터페이스에 연결되고, 스칼라 비디오 처리 동작을 실행하도록 구성된 스칼라 실행 유닛, 및 호스에 연결되고, 벡터 비디오 처리 동작을 실행하도록 구성된 벡터 실행 유닛을 포함한다. 벡터 실행 유닛이 메모리 커맨드 FIFO를 액세스함으로써 요구 구동 기반으로 동작하는 것을 가능하게 하기 위한 커맨드 FIFO가 포함된다. 비디오 프로세서와 프레임 버퍼 메모리 간의 통신을 구현하기 위한 메모리 인터페이스가 포함된다. 복수의 상이한 메모리 위치 간의 DMA 전달을 구현하고, 벡터 실행 유닛에 대한 데이터 및 명령을 데이터 저장 메모리 및 명령 캐시에 로딩하기 위한 DMA 엔진이 메모리 인터페이스 내에 구축된다.

[0012] 일 실시예에서, 벡터 실행 유닛은 요구 구동 기반으로 동작하기 위해 커맨드 FIFO를 액세스함으로써 스칼라 실행 유닛에 관하여 비동기적으로 동작하도록 구성된다. 요구 구동 기반은 상이한 메모리 위치들(예를 들어, 프레임 버퍼 메모리, 시스템 메모리, 캐시 메모리 등)에서 벡터 실행 유닛의 커맨드 FIFO로의 데이터 전달의 지연을 숨기도록 구성될 수 있다. 커맨드 FIFO는 벡터 실행 유닛의 기능 정지를 방지하기 위한 파이프라인 FIFO일 수 있다.

[0013] 일 실시예에서, 본 발명은 비디오 처리 동작을 실행하기 위한 비디오 프로세서로서 구현된다. 비디오 프로세서는 비디오 프로세서와 호스트 CPU 간의 통신을 구현하기 위한 호스트 인터페이스를 포함한다. 비디오 프로세서는 비디오 프로세서와 프레임 버퍼 메모리 간의 통신을 구현하기 위한 메모리 인터페이스를 포함한다. 스칼라 실행 유닛이 호스트 인터페이스 및 메모리 인터페이스에 연결되고, 스칼라 비디오 처리 동작을 실행하도록 구성된다. 벡터 실행 유닛이 호스트 인터페이스 및 메모리 인터페이스에 연결되고, 벡터 비디오 처리 동작을 실행하도록 구성된다. 비디오 프로세서는 독립형 비디오 프로세서 집적 회로이거나, GPU 집적 회로 내에 집적된 컴포넌트일 수 있다.

[0014] 일 실시예에서, 스칼라 실행 유닛은 비디오 프로세서의 제어기로서 기능하여, 벡터 실행 유닛의 동작을 제어한다. 스칼라 실행 유닛은 애플리케이션의 흐름 제어 알고리즘을 실행하도록 구성될 수 있고, 벡터 실행 유닛은 애플리케이션의 픽셀 처리 동작을 실행하도록 구성될 수 있다. 스칼라 실행 유닛과 벡터 실행 유닛을 인터페이스하기 위한 벡터 인터페이스 유닛이 비디오 프로세서 내에 포함될 수 있다. 일 실시예에서, 스칼라 실행 유닛 및 벡터 실행 유닛은 비동기적으로 동작하도록 구성된다. 스칼라 실행 유닛은 제1 클럭 주파수로 실행할 수 있고, 벡터 실행 유닛은 상이한 클럭 주파수(예를 들어, 더 빠르거나, 더 느린 등)로 실행할 수 있다. 벡터 실행

유닛은 스칼라 실행 유닛의 제어하에 요구 구동 기반으로 동작할 수 있다.

[0015] 일 실시예에서, 본 발명은 비디오 처리 동작을 실행하기 위한 비디오 프로세서용의 다차원 데이터 경로 처리 시스템으로서 구현된다. 비디오 프로세서는 스칼라 비디오 처리 동작을 실행하도록 구성된 스칼라 실행 유닛 및 벡터 비디오 처리 동작을 실행하도록 구성된 벡터 실행 유닛을 포함한다. 벡터 실행 유닛에 대한 데이터를 저장하기 위한 데이터 저장 메모리가 포함된다. 데이터 저장 메모리는 어레이로 배열된 대칭 뱅크 데이터 구조들을 가진 복수의 타일을 포함한다. 뱅크 데이터 구조들은 각 뱅크의 상이한 타일들로의 액세스를 지원하도록 구성된다.

[0016] 특정 구성의 요건에 따라, 뱅크 데이터 구조들 각각은 복수의 타일(예를 들어, 4x4, 8x8, 8x16, 16x24 등)을 포함할 수 있다. 일 실시예에서, 뱅크들은 각 뱅크의 상이한 타일들로의 액세스를 지원하도록 구성된다. 이것은 단일 액세스가 2개의 인접 뱅크로부터 타일들의 행(row) 또는 열(column)을 검색하는 것을 가능하게 한다. 일 실시예에서, 복수의 뱅크 데이터 구조의 타일들(예를 들어, 행, 열, 블록 등)을 액세스하기 위한 구성을 선택하기 위해 크로스바가 사용된다. 크로스바에 의해 액세스되는 뱅크들의 타일들을 수신하고 타일들을 클럭 단위로 벡터 데이터 경로의 프론트 엔드에 제공하기 위한 콜렉터가 포함될 수 있다.

[0017] 일 실시예에서, 본 발명은 비디오 프로세서용의 스트림 기반 메모리 액세스 시스템으로서 구현된다. 비디오 프로세서는 스칼라 비디오 처리 동작을 실행하도록 구성된 스칼라 실행 유닛 및 벡터 비디오 처리 동작을 실행하도록 구성된 벡터 실행 유닛을 포함한다. 스칼라 실행 유닛 및 벡터 실행 유닛에 대한 데이터를 저장하기 위한 프레임 버퍼 메모리가 포함된다. 스칼라 실행 유닛 및 벡터 실행 유닛과 프레임 버퍼 메모리 간의 통신을 구현하기 위한 메모리 인터페이스가 포함된다. 프레임 버퍼 메모리는 복수의 타일을 포함한다. 메모리 인터페이스는 스칼라 실행 유닛에 대한 제1 순차적 타일 액세스의 제1 스트림을 구현하고, 벡터 실행 유닛에 대한 제2 순차적 타일 액세스의 제2 스트림을 구현한다.

[0018] 일 실시예에서, 제1 스트림 및 제2 스트림은 발원 메모리 위치(예를 들어, 프레임 버퍼 메모리, 시스템 메모리 등)로부터의 액세스 지연을 숨기는 방식으로 프리페치되는 일련의 순차적 프리페치 타일들을 포함한다. 일 실시예에서, 메모리 인터페이스는 복수의 상이한 발원 위치로부터의 그리고 복수의 상이한 종료 위치로의 복수의 상이한 스트림을 관리하도록 구성된다. 일 실시예에서, 메모리 인터페이스 내에 구축된 DMA 엔진이 다수의 스트림을 지원하기 위해 복수의 메모리 판독 및 복수의 메모리 기입을 구현하는 데 사용된다.

[0019] 대체로, 본 발명은 적어도 다음 4개의 방법을 개시한다.

[0020] A) 본 명세서에 광범위하게 교시되는 방법은 비디오 처리 동작을 실행하기 위한 비디오 프로세서에서의 다차원 데이터 경로 처리 시스템의 방법으로서, 스칼라 실행 유닛을 이용하여 스칼라 비디오 처리 동작을 실행하는 단계; 벡터 실행 유닛을 이용하여 벡터 비디오 처리 동작을 실행하는 단계; 데이터 저장 메모리를 이용하여 벡터 실행 유닛에 대한 데이터를 저장하는 단계를 포함하고, 데이터 저장 메모리는 어레이로 배열된 대칭 뱅크 데이터 구조들을 포함하는 복수의 타일을 포함하고, 뱅크 데이터 구조들은 각 뱅크의 상이한 타일들로의 액세스를 지원하도록 구성된다. 또한, 상기 A 방법은 4x4 패턴으로 배열된 복수의 타일을 포함하는 뱅크 데이터 구조들 각각을 포함한다. 또한, 상기 A 방법은 8x8, 8x16 또는 16x24 패턴으로 배열된 복수의 타일을 포함하는 뱅크 데이터 구조들 각각을 포함한다. 또한, 상기 A 방법은 각각의 뱅크 데이터 구조의 상이한 타일들로의 액세스를 지원하도록 구성된 뱅크 데이터 구조들을 포함하고, 적어도 하나의 액세스는 2개의 뱅크 데이터 구조의 타일들의 행을 포함하는 2개의 인접한 뱅크 데이터 구조에 대한 것이다. 상기 A 방법은 또한 각각의 뱅크 데이터 구조의 상이한 타일들로의 액세스를 지원하도록 구성된 타일들을 포함하며, 적어도 하나의 액세스는 2개의 인접 뱅크 데이터 구조의 타일들의 열을 포함하는 2개의 인접 뱅크 데이터 구조에 대한 것이다. 또한, 상기 A 방법은 데이터 저장 메모리에 연결된 크로스바를 이용하여 복수의 뱅크 데이터 구조의 타일들을 액세스하기 위한 구성을 선택하는 단계를 포함한다. 이 선택 단계에서, 크로스바는 클럭 단위로 벡터 데이터 경로에 데이터를 제공하기 위해 복수의 뱅크 데이터 구조의 타일들을 액세스한다. 또한, 콜렉터를 이용하여 크로스바에 의해 액세스되는 복수의 뱅크 데이터 구조의 타일들을 수신하는 단계, 및 클럭 단위로 벡터 데이터 경로의 프론트 엔드에 타일들을 제공하는 단계를 포함한다.

[0021] B) 본 명세서에서 광범위하게 교시되는 방법은 또한 비디오 처리 동작을 실행하기 위한 방법으로서, 컴퓨터 판독 가능 코드를 실행하는 컴퓨터 시스템의 비디오 프로세서를 이용하여 구현되며, 호스트 인터페이스를 이용하여 비디오 프로세서와 호스트 CPU 간의 통신을 설정하는 단계; 메모리 인터페이스를 이용하여 비디오 프로세서와 프레임 버퍼 메모리 간의 통신을 설정하는 단계; 호스트 인터페이스 및 메모리 인터페이스에 연결된 스칼라 실행 유닛을 이용하여 스칼라 비디오 처리 동작을 실행하는 단계; 및 호스트 인터페이스 및 메모리 인터페이스

에 연결된 벡터 실행 유닛을 이용하여 벡터 비디오 처리 동작을 실행하는 단계를 포함한다. 상기 B 방법은 비디오 프로세서의 제어기로서 기능하여 벡터 실행 유닛의 동작을 제어하는 스칼라 실행 유닛을 더 포함한다. 상기 B 방법은 스칼라 실행 유닛과 벡터 실행 유닛을 인터페이스하기 위한 벡터 인터페이스 유닛을 더 포함한다. 상기 B 방법은 비동기적으로 동작하도록 구성된 스칼라 실행 유닛 및 벡터 실행 유닛을 더 포함한다. 또한, 스칼라 실행 유닛은 제1 클럭 주파수로 실행하고, 벡터 실행 유닛은 제2 클럭 주파수로 실행한다. 상기 B 방법은 애플리케이션의 흐름 제어 알고리즘을 실행하도록 구성된 스칼라 실행 유닛 및 애플리케이션의 픽셀 처리 동작을 실행하도록 구성된 벡터 실행 유닛을 포함한다. 또한, 벡터 실행 유닛은 스칼라 실행 유닛의 제어하에 요구 구동 기반으로 동작하도록 구성된다. 또한, 스칼라 실행 유닛은 메모리 커맨드 FIFO를 이용하여 벡터 실행 유닛에 기능 호출들을 전송하도록 구성되며, 벡터 실행 유닛은 메모리 커맨드 FIFO를 액세스함으로써 요구 구동 기반으로 동작한다. 또한, 비디오 프로세서의 비동기 동작은 애플리케이션의 벡터 서브 루틴 또는 스칼라 서브 루틴의 개별적인 독립적 갱신을 지원하도록 구성된다. 마지막으로, 상기 B 방법은 VLIW(very long instruction word) 코드를 이용하여 동작하도록 구성된 스칼라 실행 유닛을 포함한다.

[0022] C) 본 명세서에 설명되는 방법은 또한 비디오 처리 동작을 실행하기 위한 비디오 프로세서에서의 스트림 기반 메모리 액세스를 위한 방법을 광범위하게 교시하고 있는데, 이 방법은 스칼라 실행 유닛을 이용하여 스칼라 비디오 처리 동작을 실행하는 단계; 벡터 실행 유닛을 이용하여 벡터 비디오 처리 동작을 실행하는 단계; 프레임 버퍼 메모리를 이용하여 스칼라 실행 유닛 및 벡터 실행 유닛에 대한 데이터를 저장하는 단계; 및 메모리 인터페이스를 이용하여 스칼라 실행 유닛 및 벡터 실행 유닛과 프레임 버퍼 메모리 간의 통신을 구현하는 단계를 포함하고, 상기 프레임 버퍼 메모리는 복수의 타일을 포함하고, 상기 메모리 인터페이스는 벡터 실행 유닛 또는 스칼라 실행 유닛에 대해 제1 순차적 타일 액세스를 포함하는 제1 스트림을 구현하고, 제2 순차적 타일 액세스를 포함하는 제2 스트림을 구현한다. 상기 C 방법은 또한 적어도 하나의 프리페치 타일을 포함하는 제1 스트림 및 제2 스트림을 갖는다. 상기 C 방법은 프레임 버퍼 메모리 내의 제1 위치로부터 발원되는 제1 스트림, 및 프레임 버퍼 메모리의 제2 위치로부터 발원되는 제2 스트림을 더 포함한다. 상기 C 방법은 복수의 상이한 발원 위치로부터의 그리고 복수의 상이한 종료 위치로의 복수의 스트림을 관리하도록 구성된 메모리 인터페이스를 더 포함한다. 이와 관련하여, 발원 위치들 중 적어도 하나 또는 종료 위치들 중 적어도 하나는 시스템 메모리 내에 있다. 상기 C 방법은 메모리 인터페이스 내에 구축된 DMA 엔진을 이용하여 제1 스트림 및 제2 스트림을 지원하기 위해 복수의 메모리 판독을 구현하고, 제1 스트림 및 제2 스트림을 지원하기 위해 복수의 메모리 기입을 구현하는 단계를 더 포함한다. 또한, 상기 C 방법은 제2 스트림보다 높은 양의 지연을 경험하는 제1 스트림을 포함하고, 제1 스트림은 제2 스트림보다 많은 수의 타일 저장용 버퍼를 포함한다. 상기 C 방법은 또한 제1 스트림 또는 제2 스트림의 지연을 보상하기 위하여 제1 스트림 또는 제2 스트림의 조정 가능한 수의 타일들을 프리페치하도록 구성된 메모리 인터페이스를 포함한다.

[0023] D) 본 명세서에 설명되는 방법은 또한 지연 감내 비디오 처리 동작을 위한 방법을 광범위하게 포함하는데, 이 방법은 호스트 인터페이스를 이용하여 비디오 프로세서와 호스트 CPU 간의 통신을 구현하는 단계; 호스트 인터페이스에 연결된 스칼라 실행 유닛을 이용하여 스칼라 비디오 처리 동작을 실행하는 단계; 호스트 인터페이스에 연결된 벡터 실행 유닛을 이용하여 벡터 비디오 처리 동작을 실행하는 단계; 벡터 실행 유닛이 메모리 커맨드 FIFO를 액세스함으로써 요구 구동 기반으로 동작할 수 있게 하는 단계; 메모리 인터페이스를 이용하여 비디오 프로세서와 프레임 버퍼 메모리 간의 통신을 구현하는 단계; 및 메모리 인터페이스 내에 구축되고 벡터 실행 유닛에 대한 데이터 및 명령을 데이터 저장 메모리 및 명령 캐시에 로딩하도록 구성된 DMA 엔진을 이용하여 복수의 상이한 메모리 위치 간의 DMA 전달을 구현하는 단계를 포함한다. 상기 D 방법은 요구 구동 기반으로 동작하기 위하여 커맨드 FIFO를 액세스함으로써 스칼라 실행 유닛에 관하여 비동기적으로 동작하도록 구성된 벡터 실행 유닛을 더 포함한다. 상기 D 방법은 상이한 메모리 위치들에서 벡터 실행 유닛의 커맨드 FIFO로의 데이터 전달의 지연을 숨기도록 구성된 요구 구동 기반을 더 포함한다. 또한, 상기 D 방법은 알고리즘 흐름 제어 처리를 구현하도록 구성된 스칼라 실행 유닛을 포함하고, 벡터 실행 유닛은 비디오 처리 작업 부하의 대부분을 구현하도록 구성된다. 이와 관련하여, 스칼라 실행 유닛은 데이터 전달 지연을 숨기기 위하여 벡터 실행 유닛에 대한 작업 파라미터를 사전 계산하도록 구성된다. 상기 D 방법은 벡터 서브 루틴의 후속 실행을 위한 커맨드를 프리페치하기 위해 DMA 엔진을 통해 메모리 판독을 스케줄링하도록 구성된 벡터 실행 유닛을 포함한다. 여기서, 메모리 판독은 스칼라 실행 유닛에 의한 벡터 서브 루틴의 호출 이전에 벡터 서브 루틴의 실행을 위한 커맨드를 프리페치하도록 스케줄링된다.

[0024] 본 발명의 또다른 특징에 따르면, 제1 양태는, 스칼라 비디오 처리 동작을 실행하도록 구성된 스칼라 실행 유닛; 벡터 비디오 처리 동작을 실행하도록 구성된 벡터 실행 유닛; 및 상기 벡터 실행 유닛에 대한 데이터를 저장하기 위한 데이터 저장 메모리를 포함하고, 상기 데이터 저장 메모리는 어레이로 배열된 대칭 뱅크 데이터

구조들을 포함하는 복수의 타일을 포함하고, 상기 뱅크 데이터 구조들은 각각의 뱅크의 상이한 타일들로의 액세스를 지원하도록 구성되는 시스템이다.

[0025] 본 발명의 제2 양태는, 제1 양태에 있어서, 상기 시스템은 비디오 처리 동작을 실행하기 위한 비디오 프로세서용의 다차원 데이터 경로 처리 시스템이다.

[0026] 본 발명의 제3 양태는, 비디오 처리 동작을 지원하기 위한 다차원 데이터 경로 처리용 시스템으로서, 마더보드; 상기 마더보드에 연결된 호스트 CPU; 및 상기 마더보드 및 상기 CPU에 연결되고, 제1항의 시스템을 포함하는 비디오 프로세서를 포함하는 시스템이다.

[0027] 본 발명의 제4 양태는, 제1 내지 제3 양태 중 어느 한 양태에 있어서, 상기 뱅크 데이터 구조들 각각은 4x4 패턴으로 배열된 복수의 타일을 포함한다.

[0028] 본 발명의 제5 양태는, 제1 내지 제3 양태 중 어느 한 양태에 있어서, 상기 뱅크 데이터 구조들 각각은 8x8 또는 8x16 또는 16x24 패턴으로 배열된 복수의 타일을 포함한다.

[0029] 본 발명의 제6 양태는, 제1 내지 제3 양태 중 어느 한 양태에 있어서, 상기 뱅크 데이터 구조들은 각각의 뱅크 데이터 구조의 상이한 타일들로의 액세스를 지원하도록 구성되고, 적어도 하나의 액세스는 2개의 상기 뱅크 데이터 구조들의 타일들의 행을 포함하는 2개의 인접한 뱅크 데이터 구조들에 대한 것이다.

[0030] 본 발명의 제7 양태는, 제1 내지 제3 양태 중 어느 한 양태에 있어서, 상기 타일들은 각각의 뱅크 데이터 구조의 상이한 타일들로의 액세스를 지원하도록 구성되고, 적어도 하나의 액세스는 2개의 인접한 상기 뱅크 데이터 구조들의 타일들의 열을 포함하는 2개의 인접한 뱅크 데이터 구조들에 대한 것이다.

[0031] 본 발명의 제8 양태는, 제1 내지 제3 양태 중 어느 한 양태에 있어서, 상기 데이터 저장 메모리에 연결되고, 상기 복수의 뱅크 데이터 구조의 타일들을 액세스하기 위한 구성을 선택하기 위한 크로스바를 더 포함한다.

[0032] 본 발명의 제9 양태는, 제8 양태에 있어서, 상기 크로스바는 데이터를 벡터 데이터 경로에 클럭 단위 기반으로 제공하기 위해 상기 복수의 뱅크 데이터 구조의 타일들을 액세스한다.

[0033] 본 발명의 제10 양태는, 제9 양태에 있어서, 상기 크로스바에 의해 액세스되는 상기 복수의 뱅크 데이터 구조의 타일들을 수신하고, 상기 타일들을 벡터 데이터 경로의 프론트 엔드에 클럭 단위 기반으로 제공하는 콜렉터를 더 포함한다.

[0034] 본 발명의 제11 양태는, 비디오 처리 동작을 실행하기 위한 비디오 프로세서로서, 상기 비디오 프로세서와 호스트 CPU 간의 통신을 구현하기 위한 호스트 인터페이스; 상기 비디오 프로세서와 프레임 버퍼 메모리 간의 통신을 구현하기 위한 메모리 인터페이스; 상기 호스트 인터페이스 및 상기 메모리 인터페이스에 연결되고, 스칼라 비디오 처리 동작을 실행하도록 구성된 스칼라 실행 유닛; 및 상기 호스트 인터페이스 및 상기 메모리 인터페이스에 연결되고, 벡터 비디오 처리 동작을 실행하도록 구성된 벡터 실행 유닛을 포함하는 비디오 프로세서이다.

[0035] 본 발명의 제12 양태는, 비디오 처리 동작을 실행하기 위한 시스템으로서, 마더보드; 상기 마더보드에 연결된 호스트 CPU; 및 상기 마더보드 및 상기 CPU에 연결된 제11 양태의 상기 비디오 프로세서를 포함하는 시스템이다.

[0036] 본 발명의 제13 양태는, 제11 양태에 있어서, 상기 스칼라 실행 유닛은 상기 비디오 프로세서의 제어기로서 기능하고, 상기 벡터 실행 유닛의 동작을 제어한다.

[0037] 본 발명의 제14 양태는, 제11 양태에 있어서, 상기 스칼라 실행 유닛을 상기 벡터 실행 유닛과 인터페이스하기 위한 벡터 인터페이스 유닛을 더 포함한다.

[0038] 본 발명의 제15 양태는, 제11 양태에 있어서, 상기 스칼라 실행 유닛 및 상기 벡터 실행 유닛은 비동기적으로 동작하도록 구성된다.

[0039] 본 발명의 제16 양태는, 제15 양태의 비디오 프로세서 또는 제12 양태의 시스템으로서, 상기 스칼라 실행 유닛은 제1 클럭 주파수로 실행하고, 상기 벡터 실행 유닛은 제2 클럭 주파수로 실행한다.

[0040] 본 발명의 제17 양태는, 제11 양태의 비디오 프로세서 또는 제12 양태의 시스템으로서, 상기 스칼라 실행 유닛은 애플리케이션의 흐름 제어 알고리즘을 실행하도록 구성되고, 상기 벡터 실행 유닛은 상기 애플리케이션의 픽셀 처리 동작을 실행하도록 구성된다.

[0041] 본 발명의 제18 양태는, 제17 양태에 있어서, 상기 벡터 실행 유닛은 상기 스칼라 실행 유닛의 제어하에 요구

구동 기반으로(on a demand driven basis) 동작하도록 구성된다.

[0042] 본 발명의 제19 양태는, 제17 양태에 있어서, 상기 스칼라 실행 유닛은 커맨드 FIFO를 이용하여 상기 벡터 실행 유닛에 기능 호출들(function calls)을 전송하도록 구성되고, 상기 벡터 실행 유닛은 상기 커맨드 FIFO를 액세스함으로써 요구 구동 기반으로 동작한다.

[0043] 본 발명의 제20 양태는, 제17 양태에 있어서, 상기 비디오 프로세서의 비동기적인 동작은 상기 애플리케이션의 벡터 서브 루틴 또는 스칼라 서브 루틴의 개별적인 독립적 갱신을 지원하도록 구성된다.

[0044] 본 발명의 제21 양태는, 제11 양태에 있어서, 상기 스칼라 실행 유닛은 VLIW(very long instruction word) 코드를 이용하여 동작하도록 구성된다.

[0045] 본 발명의 제22 양태는, 비디오 처리 동작을 실행하기 위한 비디오 프로세서용 스트림 기반 메모리 액세스 시스템으로서, 스칼라 비디오 처리 동작을 실행하도록 구성된 스칼라 실행 유닛; 벡터 비디오 처리 동작을 실행하도록 구성된 벡터 실행 유닛; 상기 스칼라 실행 유닛 및 상기 벡터 실행 유닛에 대한 데이터를 저장하기 위한 프레임 버퍼 메모리; 및 상기 스칼라 실행 유닛 및 상기 벡터 실행 유닛과 상기 프레임 버퍼 메모리 간의 통신을 구현하기 위한 메모리 인터페이스를 포함하고, 상기 프레임 버퍼 메모리는 복수의 타일을 포함하고, 상기 메모리 인터페이스는 상기 벡터 실행 유닛 또는 상기 스칼라 실행 유닛에 대해 제1 순차적 타일 액세스를 포함하는 제1 스트림을 구현하고, 제2 순차적 타일 액세스를 포함하는 제2 스트림을 구현하는 시스템이다.

[0046] 본 발명의 제23 양태는, 비디오 처리 동작을 지원하기 위해 스트림 기반 메모리 액세스들을 실행하는 시스템으로서, 마더보드; 상기 마더보드에 연결되는 호스트 CPU; 상기 마더보드 및 상기 CPU에 연결된 비디오 프로세서 - 상기 비디오 프로세서는, 상기 비디오 프로세서와 상기 호스트 CPU 간의 통신을 설정하기 위한 호스트 인터페이스, 상기 호스트 인터페이스에 연결되고, 스칼라 비디오 처리 동작을 실행하도록 구성된 스칼라 실행 유닛, 및 상기 호스트 인터페이스에 연결되고, 벡터 비디오 처리 동작을 실행하도록 구성된 벡터 실행 유닛을 포함함 - ; 및 상기 스칼라 실행 유닛 및 상기 벡터 실행 유닛에 연결되고, 상기 스칼라 실행 유닛과 상기 벡터 실행 유닛과 프레임 버퍼 메모리 간의 스트림 기반 통신을 설정하기 위한 메모리 인터페이스를 포함하고, 상기 프레임 버퍼 메모리는 복수의 타일을 포함하고, 상기 메모리 인터페이스는 상기 벡터 실행 유닛 또는 상기 스칼라 실행 유닛에 대해 제1 순차적 타일 액세스를 포함하는 제1 스트림을 구현하고, 제2 순차적 타일 액세스를 포함하는 제2 스트림을 구현하는 시스템이다.

[0047] 본 발명의 제24 양태는, 제22 양태에 있어서, 상기 제1 스트림 및 상기 제2 스트림은 적어도 하나의 프리페치된 타일(prefetched tile)을 포함한다.

[0048] 본 발명의 제25 양태는, 제22 양태에 있어서, 상기 제1 스트림은 상기 프레임 버퍼 메모리 내의 제1 위치로부터 발원되고, 상기 제2 스트림은 상기 프레임 버퍼 메모리 내의 제2 위치로부터 발원된다.

[0049] 본 발명의 제26 양태는, 제22 또는 제23 양태에 있어서, 상기 메모리 인터페이스는 복수의 상이한 발원 위치로부터, 그리고 복수의 상이한 종료 위치까지의 복수의 스트림을 관리하도록 구성된다.

[0050] 본 발명의 제27 양태는, 제26 양태에 있어서, 상기 발원 위치들 중 적어도 하나 또는 상기 종료 위치들 중 적어도 하나는 시스템 메모리 내에 있다.

[0051] 본 발명의 제28 양태는, 제22 또는 제23 양태에 있어서, 상기 메모리 인터페이스 내에 구축되고, 상기 제1 스트림 및 상기 제2 스트림을 지원하기 위해 복수의 메모리 판독을 구현하고, 상기 제1 스트림 및 상기 제2 스트림을 지원하기 위해 복수의 메모리 기입을 구현하도록 구성된 DMA 엔진을 더 포함한다.

[0052] 본 발명의 제29 양태는, 제22 또는 제23 양태에 있어서, 상기 제1 스트림은 상기 제2 스트림보다 더 높은 양의 지연을 경험하고, 상기 제1 스트림은 상기 제2 스트림보다 많은 수의 타일 저장용 버퍼들을 통합한다.

[0053] 본 발명의 제30 양태는, 제22 또는 제23 양태에 있어서, 상기 메모리 인터페이스는 상기 제1 스트림 또는 상기 제2 스트림의 지연을 보상하기 위하여 상기 제1 스트림 또는 상기 제2 스트림의 조정 가능한 수의 타일을 프리페치하도록 구성된다.

[0054] 본 발명의 제31 양태는, 비디오 프로세서와 호스트 CPU 간의 통신을 구현하기 위한 호스트 인터페이스; 상기 호스트 인터페이스에 연결되고, 스칼라 비디오 처리 동작을 실행하도록 구성된 스칼라 실행 유닛; 상기 호스트 인터페이스에 연결되고, 벡터 비디오 처리 동작을 실행하도록 구성된 벡터 실행 유닛; 커맨드 FIFO로서, 상기 벡터 실행 유닛이 상기 커맨드 FIFO를 액세스함으로써 요구 구동 기반으로 동작할 수 있게 하는 커맨드 FIFO; 상

기 비디오 프로세서와 프레임 버퍼 메모리 간의 통신을 구현하기 위한 메모리 인터페이스; 및 상기 메모리 인터페이스 내에 구축되고, 복수의 상이한 메모리 위치 간의 DMA 전달을 구현하고, 데이터 저장 메모리 및 명령 캐시에 상기 벡터 실행 유닛에 대한 데이터 및 명령을 로딩하기 위한 DMA 엔진을 포함하는 시스템이다.

[0055] 본 발명의 제32 양태는, 제31 양태에 있어서, 상기 시스템은 비디오 처리 동작을 실행하기 위한 지연 감내 시스템(latency tolerant system)이다.

[0056] 본 발명의 제33 양태는, 제32 양태에 있어서, 마더보드; 상기 마더보드에 연결된 호스트 CPU; 상기 마더보드 및 상기 CPU에 연결된 비디오 프로세서를 더 포함한다.

[0057] 본 발명의 제34 양태는, 제31 내지 제33 양태 중 어느 한 양태에 있어서, 상기 벡터 실행 유닛은 요구 구동 기반으로 동작하기 위하여 상기 커맨드 FIFO를 액세스함으로써 상기 스칼라 실행 유닛에 대하여 비동기적으로 동작하도록 구성된다.

[0058] 본 발명의 제35 양태는, 제31 내지 제33 양태 중 어느 한 양태에 있어서, 상기 요구 구동 기반은 상기 상이한 메모리 위치들로부터 상기 벡터 실행 유닛의 커맨드 FIFO로의 데이터 전달의 지연을 숨기도록 구성된다.

[0059] 본 발명의 제36 양태는, 제31 내지 제33 양태 중 어느 한 양태에 있어서, 상기 스칼라 실행 유닛은 알고리즘 흐름 제어 처리를 구현하도록 구성되고, 상기 벡터 실행 유닛은 비디오 처리 작업 부하의 대부분을 구현하도록 구성된다.

[0060] 본 발명의 제37 양태는, 제36 양태에 있어서, 상기 스칼라 실행 유닛은 데이터 전달 지연을 숨기기 위해 상기 벡터 실행 유닛에 대한 작업 파라미터들을 사전 계산하도록 구성된다.

[0061] 본 발명의 제38 양태는, 제31 양태에 있어서, 상기 벡터 실행 유닛은 벡터 서브 루틴의 후속 실행을 위한 커맨드들을 프리페치하기 위해 상기 DMA 엔진을 통해 메모리 판독을 스케쥴링하도록 구성된다.

[0062] 본 발명의 제39 양태는, 제38 양태에 있어서, 상기 메모리 판독은 상기 스칼라 실행 유닛에 의한 상기 벡터 서브 루틴에 대한 호출 이전에 상기 벡터 서브 루틴의 실행을 위한 커맨드들을 프리페치하도록 스케쥴링된다.

[0063] 본 발명의 제40 양태는, 제33 양태에 있어서, 상기 벡터 실행 유닛은 벡터 서브 루틴의 후속 실행을 위한 커맨드들을 프리페치하기 위해 상기 DMA 엔진을 통해 메모리 판독을 스케쥴링하도록 구성되고, 상기 메모리 판독은 상기 스칼라 실행 유닛에 의한 상기 벡터 서브 루틴에 대한 호출 이전에 상기 벡터 서브 루틴의 실행을 위한 커맨드들을 프리페치하도록 스케쥴링된다.

발명의 효과

[0064] 본 발명의 실시예들은 집적 회로 실리콘 다이 면적, 트랜지스터 수, 메모리 속도 요건 등을 효율적으로 이용하면서 정교한 비디오 처리 기능들을 지원하는 새로운 비디오 프로세서 아키텍처를 제공한다. 본 발명의 실시예들은 다수의 비디오 스트림을 처리하기 위하여 높은 연산 밀도를 유지하며, 쉽게 스케일링이 가능하다. 본 발명의 실시예들은 예를 들어 MPEG-2/WMV9/H.264 인코드 지원(예를 들어, 루프내 디코더), MPEG-2/WMV9/H.264 디코드(예를 들어, 포스트 엔트로피 디코딩), 및 루프내/루프외 비블록킹 필터와 같은 다수의 정교한 비디오 처리 동작을 제공할 수 있다.

[0065] 본 발명의 실시예들에 의해 제공되는 추가적인 비디오 처리 동작은 예를 들어 향상된 모션 적응 디인터레이싱, 인코딩을 위한 입력 잡음 필터링, 다위상 스케일링/리샘플링, 및 서브 픽처 합성을 포함한다. 본 발명의 비디오 프로세서 아키텍처는 또한 예를 들어 칼라 공간 변환, 칼라 공간 조정, 샤프닝, 히스토그램 조정과 같은 픽셀 포인트 조작, 및 다양한 비디오 표면 포맷 변환과 같은 소정의 비디오 프로세서-증폭기(프로캠프) 애플리케이션에 이용될 수 있다.

도면의 간단한 설명

[0066] 본 발명은 한정적이 아니라 예시적으로 설명되며, 첨부 도면들에서 동일한 참조 번호는 유사한 요소를 나타낸다.

도 1은 본 발명의 일 실시예에 따른 컴퓨터 시스템의 기본 컴포넌트를 나타내는 개략도.

도 2는 본 발명의 일 실시예에 따른 비디오 프로세서의 내부 컴포넌트를 나타내는 도면.

도 3은 본 발명의 일 실시예에 따른 비디오 프로세서에 대한 예시적인 소프트웨어 프로그램을 나타내는 도면.

도 4는 본 발명의 일 실시예에 따른 비디오 프로세서를 이용하는 비디오와의 서브 픽처 혼합의 일례를 나타내는 도면.

도 5는 본 발명의 일 실시예에 따른 벡터 실행 유닛의 내부 컴포넌트를 나타내는 도면.

도 6은 본 발명의 일 실시예에 따른 대칭 타일 어레이를 구비한 데이터 저장 메모리의 레이아웃을 나타내는 도면.

발명을 실시하기 위한 구체적인 내용

[0067] 이제 본 발명의 바람직한 실시예들을 상세히 참조하는데, 그 예는 첨부 도면들에 도시되어 있다. 본 발명은 바람직한 실시예들과 관련하여 설명되지만, 본 발명은 이들 실시예에 한정되는 것은 아니라는 것을 이해할 것이다. 오히려, 본 발명은 첨부된 청구범위에 의해 정의되는 발명의 사상 및 범위 내에 포함될 수 있는 대안, 변형 및 균등물을 포함하는 것으로 의도한다. 더욱이, 아래의 본 발명의 실시예들의 상세한 설명에서, 다양한 특정 상세는 본 발명의 완전한 이해를 제공하기 위해 설명된다. 그러나, 본 발명은 이러한 특정 상세 없이도 실시될 수 있다는 것을 이 분야의 전문가들은 이해할 것이다. 다른 사례에서, 본 발명의 실시예들의 양태를 불필요하게 불명확하게 하지 않기 위하여 공지 방법, 프로시저, 컴포넌트 및 회로는 상세히 설명되지 않는다.

[0068] 기호 및 용어

[0069] 이어지는 상세한 설명의 몇몇 부분은 프로시저, 단계, 논리 블록, 처리 및 컴퓨터 메모리 내의 데이터 비트 상의 동작의 다른 기호 표현을 이용하여 제공된다. 이들 설명 및 표현은 데이터 처리 분야의 전문가들이 그들의 작업의 내용을 다른 전문가에게 가장 효과적으로 전달하기 위해 사용하는 수단이다. 본 명세서에서, 일반적으로, 프로시저, 컴퓨터 실행 단계, 논리 블록, 프로세서 등은 원하는 결과를 도출하는 단계들 또는 명령들의 모순 없는 시퀀스인 것으로 고려된다. 이 단계들은 물리량의 물리적 조작을 요구하는 단계들이다. 통상적으로, 반드시 그렇지는 않지만, 이들 양은 컴퓨터 시스템에서 저장, 전송, 조합, 비교 및 조작될 수 있는 전기 또는 자기적 신호의 형태를 갖는다. 이들 신호를 비트, 값, 요소, 기호, 문자, 용어, 수 등으로 지칭하는 것이 때때로 주로 일반적인 사용을 위해 편리한 것으로 입증되었다.

[0070] 그러나, 이들 및 유사한 용어들 모두는 적절한 물리량과 연관되며 이들 양에 적용되는 편리한 라벨일 뿐이라는 것에 유의해야 한다. 아래의 설명으로부터 명백하듯이, 특별히 달리 언급되지 않는 한, 본 발명을 통해, "처리" 또는 "액세스" 또는 "실행" 또는 "저장" 또는 "렌더링" 등과 같은 용어를 이용하는 설명은 컴퓨터 시스템의 레지스터 및 메모리 내의 물리(전자)량으로 표현되는 데이터를, 컴퓨터 시스템 메모리 또는 레지스터 또는 다른 그러한 정보 저장, 전송 또는 디스플레이 장치 내의 물리량으로 유사하게 표현되는 다른 데이터로 조작하고 변환하는 컴퓨터 시스템(예를 들어, 도 1의 컴퓨터 시스템(100)) 또는 유사한 전자 컴퓨팅 장치의 동작 및 프로세스를 지칭하는 것으로 이해된다.

[0071] 컴퓨터 시스템 플랫폼

[0072] 도 1은 본 발명의 일 실시예에 따른 컴퓨터 시스템(100)을 나타낸다. 컴퓨터 시스템(100)은 소정의 하드웨어 기반 및 소프트웨어 기반 기능을 위한 실행 플랫폼을 제공하는 본 발명의 실시예들에 따른 기본 컴퓨터 시스템의 컴포넌트들을 나타낸다. 일반적으로, 컴퓨터 시스템(100)은 적어도 하나의 CPU(101), 시스템 메모리(115), 적어도 하나의 그래픽 프로세서 유닛(GPU;110) 및 하나의 비디오 프로세서 유닛(VPU;111)을 포함한다. CPU(101)는 브리지 컴포넌트(105)를 통해 시스템 메모리(115)에 연결되거나 CPU(101) 내부의 메모리 제어기(도시되지 않음)를 통해 시스템 메모리(115)에 직접 연결될 수 있다. 브리지 컴포넌트(105; 예를 들어, 노스브리지)는 다양한 I/O 장치들(예를 들어, 하나 이상의 하드 디스크 드라이브, 이더넷 어댑터, CD ROM, DVD 등)을 접속시키는 확장 버스들을 지원할 수 있다. GPU(110) 및 비디오 프로세서 유닛(111)은 디스플레이(112)에 연결된다. 연산 능력을 더 향상시키기 위하여 하나 이상의 추가 GPU가 옵션으로 시스템(100)에 연결될 수 있다. GPU(110) 및 비디오 프로세서 유닛(111)은 브리지 컴포넌트(105)를 통해 CPU(101) 및 시스템 메모리(115)에 연결된다. 시스템(100)은 예를 들어 전용 그래픽 렌더링 GPU(110)에 연결된 강력한 범용 CPU(101)를 구비한 데스크탑 컴퓨터 시스템 또는 서버 컴퓨터 시스템으로서 구현될 수 있다. 이러한 실시예에서, 주변 버스, 특수 그래픽 메모리 및 시스템 메모리, I/O 장치 등을 추가하는 컴포넌트들이 포함될 수 있다. 마찬가지로, 시스템(100)은 핸드헬드 장치(예를 들어, 셀폰 등) 또는 예를 들어 워싱톤 레드먼드의 마이크로소프트사로부터 입수할 수 있는 Xbox(등록 상표) 또는 일본, 도꾜의 소니 컴퓨터 엔터테인먼트사로부터 입수할 수 있는 PlayStation3(등록 상표)와 같은 셋톱 비디오 게임 콘솔 장치로서 구현될 수 있다.

[0073] GPU(110)는 개별 컴포넌트, 커넥터(예를 들어, AGP 슬롯, PCI 익스프레스 슬롯 등)를 통해 컴퓨터 시스템에 연결되도록 설계된 개별 그래픽 카드, 개별 집적 회로 다이(예를 들어, 마더보드에 직접 실장됨), 또는 컴퓨터 시스템 칩셋 컴포넌트의 집적 회로 다이 내에 포함된 집적 GPU(예를 들어, 브리지 칩(105) 내에 집적됨)로서 구현될 수 있다는 것을 이해해야 한다. 또한, 높은 대역폭 그래픽 데이터 저장을 위해 GPU(110)에 대해 로컬 그래픽 메모리가 포함될 수 있다. 또한, GPU(110) 및 비디오 프로세서 유닛(111)은 동일 집적 회로 다이 상에 집적되거나(예를 들어, 컴포넌트(120)로서) 컴퓨터 시스템(100)의 마더보드에 접속되거나 그 위에 실장되는 개별 집적 회로 컴포넌트들일 수 있다는 것을 이해해야 한다.

[0074] 본 발명의 실시예들

[0075] 도 2는 본 발명의 일 실시예에 따른 비디오 프로세서 유닛(111)의 내부 컴포넌트를 나타내는 도면이다. 도 2에 도시된 바와 같이, 비디오 프로세서 유닛(111)은 스칼라 실행 유닛(201), 벡터 실행 유닛(202), 메모리 인터페이스(203) 및 호스트 인터페이스(204)를 포함한다.

[0076] 도 2의 실시예에서, 비디오 프로세서 유닛(이하 간단히 비디오 프로세서라 함; 111)은 비디오 처리 동작을 실행하기 위한 기능 컴포넌트들을 포함한다. 비디오 프로세서(111)는 호스트 인터페이스(204)를 이용하여 브리지(105)를 통해 비디오 프로세서(111)와 호스트 CPU(101) 간의 통신을 설정한다. 비디오 프로세서(111)는 메모리 인터페이스(203)를 이용하여 비디오 프로세서(111)와 프레임 버퍼 메모리(205)(예를 들어, 도시되지 않은 연결된 디스플레이(112)를 위한 것임) 간의 통신을 설정한다. 스칼라 실행 유닛(201)은 호스트 인터페이스(204) 및 메모리 인터페이스(203)에 연결되고, 스칼라 비디오 처리 동작을 실행하도록 구성된다. 벡터 실행 유닛은 호스트 인터페이스(204) 및 메모리 인터페이스(203)에 연결되고, 벡터 비디오 처리 동작을 실행하도록 구성된다.

[0077] 도 2의 실시예는 비디오 프로세서(111)가 그의 실행 기능을 스칼라 동작 및 벡터 동작으로 분할하는 방식을 설명한다. 스칼라 동작은 스칼라 실행 유닛(201)에 의해 구현된다. 벡터 동작은 벡터 실행 유닛(202)에 의해 구현된다.

[0078] 일 실시예에서, 벡터 실행 유닛(202)은 스칼라 실행 유닛(201)에 대해 슬레이브 코프로세서로서 기능하도록 구성된다. 이러한 실시예에서, 스칼라 실행 유닛은 벡터 실행 유닛(202)에 제어 스트림을 공급하고 벡터 실행 유닛(202)에 대한 데이터 입출력을 관리함으로써 벡터 실행 유닛(202)의 작업 부하를 관리한다. 제어 스트림은 일반적으로 기능 파라미터, 서브루틴 독립 변수 등을 포함한다. 대표적인 비디오 처리 애플리케이션에서, 애플리케이션의 처리 알고리즘의 제어 흐름은 스칼라 실행 유닛(210) 상에서 실행되는 반면, 실제 픽셀/데이터 처리 동작은 벡터 실행 유닛(202) 상에서 구현된다.

[0079] 도 2를 계속 참조하면, 스칼라 실행 유닛(201)은 RISC 기반 실행 기술을 포함하는 RISC 스타일 스칼라 실행 유닛으로서 구현될 수 있다. 벡터 실행 유닛(202)은 예를 들어 하나 이상의 SIMD 파이프라인을 구비한 SIMD 머신으로서 구현될 수 있다. 예를 들어, SIMD 파이프라인 실시예에서, 각각의 SIMD 파이프라인은 16 픽셀 폭의 데이터 경로(또는 더 넓음)로 구현되며, 따라서 벡터 실행 유닛(202)에 클럭당 최대 32 픽셀의 결과적인 데이터 출력을 생성하는 순수한 컴퓨팅 능력을 제공할 수 있다. 일 실시예에서, 스칼라 실행 유닛(201)은 클럭 단위로 스칼라 동작의 병렬 실행을 최적화하기 위해 VLIW 소프트웨어 코드를 이용하여 동작하도록 구성된 하드웨어를 포함한다.

[0080] 도 2의 실시예에서, 스칼라 실행 유닛(201)은 스칼라 프로세서(210)에 연결된 명령 캐시(211) 및 데이터 캐시(212)를 포함한다. 캐시들(211, 212)은 예를 들어 프레임 버퍼(205)와 같은 외부 메모리로의 액세스를 위해 메모리 인터페이스(203)와 인터페이스한다. 스칼라 실행 유닛(201)은 벡터 실행 유닛(202)과의 통신을 설정하기 위한 벡터 인터페이스 유닛(213)을 더 포함한다. 일 실시예에서, 벡터 인터페이스 유닛(213)은 스칼라 실행 유닛(201)과 벡터 실행 유닛(202) 간의 비동기 통신을 가능하게 하도록 구성된 하나 이상의 동기 메일 박스(214)를 포함할 수 있다.

[0081] 도 2의 실시예에서, 벡터 실행 유닛(202)은 벡터 실행 데이터 경로, 즉 벡터 데이터 경로(221)의 동작을 제어하도록 구성된 벡터 제어 유닛(220)을 포함한다. 벡터 제어 유닛(220)은 스칼라 실행 유닛(201)으로부터 명령 및 데이터를 수신하기 위한 커맨드 FIFO(225)를 포함한다. 벡터 제어 유닛(220)에 명령을 제공하도록 명령 캐시가 연결된다. 벡터 데이터 경로(221)에 입력 데이터를 제공하고 벡터 데이터 경로(221)로부터 결과적인 데이터를 수신하도록 데이터 저장 메모리(223)가 연결된다. 데이터 저장 메모리(223)는 벡터 데이터 경로(221)에 대한 명령 캐시 및 데이터 RAM으로서 기능한다. 명령 캐시(222) 및 데이터 저장 메모리(223)는 프레임 버퍼(205)와 같은 외부 메모리를 액세스하기 위해 메모리 인터페이스(203)에 연결된다. 도 2의 실시예는 또한 제2 데이터

벡터 경로(231) 및 각각의 제2 데이터 저장 메모리(223)(예를 들어, 점선)를 나타낸다. 제2 벡터 데이터 경로(231) 및 제2 데이터 저장 메모리(233)는 벡터 실행 유닛(202)이 2개의 벡터 실행 파이프라인(예를 들어, 듀얼 SIMD 파이프라인 구성)을 갖는 사례를 나타내도록 도시된다는 것을 이해해야 한다. 본 발명의 실시예들은 많은 수의 벡터 실행 파이프라인(예를 들어, 4, 8, 16 등)을 가진 벡터 실행 유닛에 적합하다.

[0082] 스칼라 실행 유닛(201)은 벡터 실행 유닛(202)에 대해 데이터 및 커맨드 입력을 제공한다. 일 실시예에서, 스칼라 실행 유닛(201)은 메모리 맵핑된 커맨드 FIFO(225)를 이용하여 벡터 실행 유닛(202)에 기능 호출들을 전송한다. 벡터 실행 유닛(202) 커맨드는 커맨드 FIFO(225) 내에 큐잉된다.

[0083] 커맨드 FIFO(225)의 사용은 스칼라 실행 유닛(201)과 벡터 실행 유닛(202)을 효과적으로 분리한다. 스칼라 실행 유닛(201)은 그 자신의 개별 클럭 상에서 기능하여, 벡터 실행 유닛(202)의 클럭 주파수와 다르고 개별적으로 제어될 수 있는 그 자신의 개별 클럭 주파수로 동작할 수 있다.

[0084] 커맨드 FIFO(225)는 벡터 실행 유닛(202)이 요구 구동 유닛으로서 동작하는 것을 가능하게 한다. 예를 들어, 작업이 스칼라 실행 유닛(201)에서 커맨드 FIFO(225)로 핸드 오프된 후, 분리 비동기 방식으로 처리하기 위해 벡터 실행 유닛(202)에 의해 액세스될 수 있다. 벡터 실행 유닛(202)은 이런 식으로 필요에 따라 또는 요구에 따라 스칼라 실행 유닛(201)에 의해 그의 작업 부하를 처리한다. 이러한 기능은 벡터 실행 유닛(202)이 최대 성능이 요구되지 않을 때 (예를 들어, 하나 이상의 내부 클럭을 줄이거나 정지시킴으로써) 능력을 보존하는 것을 가능하게 한다.

[0085] 비디오 처리 기능의 스칼라 부분(예를 들어, 스칼라 실행 유닛(201)에 의한 실행을 위해) 및 벡터 부분(예를 들어, 벡터 실행 유닛(202)에 의한 실행을 위해)으로의 분할은 비디오 프로세서(111)에 대해 구축된 비디오 처리 프로그램들이 개별 스칼라 소프트웨어 코드 및 벡터 소프트웨어 코드로 컴파일되는 것을 가능하게 한다. 스칼라 소프트웨어 코드 및 벡터 소프트웨어 코드는 개별적으로 컴파일된 후, 함께 링크되어 코히런트 애플리케이션을 형성할 수 있다.

[0086] 분할(partitioning)은 벡터 소프트웨어 코드 함수들이 스칼라 소프트웨어 코드 함수들과 별개로 다르게 작성되는 것을 가능하게 한다. 예를 들어, 벡터 함수들은 개별적으로(예를 들어, 상이한 시간에, 상이한 엔지니어 팀에 의해 등) 작성될 수 있으며, 스칼라 함수들(스칼라 스레드, 프로세스 등)에 의해/과 함께 사용하기 위해 하나 이상의 서브루틴 또는 라이브러리 함수로서 제공될 수 있다. 이것은 스칼라 소프트웨어 코드 및/또는 벡터 소프트웨어 코드의 개별적인 독립적 갱신을 가능하게 한다. 예를 들어, 벡터 서브루틴이 스칼라 서브루틴과 독립적으로 (예를 들어, 사전 배포된 프로그램, 배포된 프로그램의 기능을 향상시키기 위해 추가된 새로운 특징 등의 갱신을 통해) 갱신될 수 있으며, 그 역도 성립한다. 분할은 스칼라 프로세서(210)의 개별 캐시(예를 들어, 캐시 211, 212) 및 벡터 제어 유닛(220) 및 벡터 데이터 경로(221)의 개별 캐시(예를 들어, 캐시 222, 223)에 의해 용이해진다. 전술한 바와 같이, 스칼라 실행 유닛(201) 및 벡터 실행 유닛(202)은 커맨드 FIFO(225)를 통해 통신한다.

[0087] 도 3은 본 발명의 일 실시예에 따른 비디오 프로세서(111)용의 예시적인 소프트웨어 프로그램(300)을 나타내는 도면이다. 도 3에 도시된 바와 같이, 소프트웨어 프로그램(300)은 비디오 프로세서(111)에 대한 프로그래밍 모델의 속성을 나타내며, 이에 의해 스칼라 제어 스레드(301)가 벡터 데이터 스레드(302)와 함께 비디오 프로세서(111)에 의해 실행된다.

[0088] 도 3의 실시예의 소프트웨어 프로그램(300)의 예는 비디오 프로세서(111)에 대한 프로그래밍 모델을 나타내며, 이에 의해 스칼라 실행 유닛(301) 상의 스칼라 제어 프로그램(예를 들어, 스칼라 제어 스레드 301)이 벡터 실행 유닛(202) 상에서 서브루틴 호출(예를 들어, 벡터 데이터 스레드(302)을 실행한다. 소프트웨어 프로그램(300)의 예는 컴파일러 또는 소프트웨어 프로그래머가 비디오 처리 애플리케이션을 스칼라 부분(예를 들어, 제1 스레드) 및 벡터 부분(예를 들어, 제2 스레드)으로 분할한 사례를 나타낸다.

[0089] 도 3에 도시된 바와 같이, 스칼라 실행 유닛(201) 상에서 실행되는 스칼라 제어 스레드(301)는 재빨리 작업 파라미터를 계산하고 이들 파라미터를 처리 작업의 대부분을 수행하는 벡터 실행 유닛(202)에 공급한다. 전술한 바와 같이, 2개의 스레드(301, 302)에 대한 소프트웨어 코드는 개별적으로 작성되고 컴파일될 수 있다.

[0090] 스칼라 스레드는 다음을 담당한다.

[0091] 1. 호스트 유닛(204)과의 인터페이싱 및 클래스 인터페이스의 구현

[0092] 2. 벡터 실행 유닛(202)의 초기화, 셋업 및 구성

[0093] 3. 루프 내의 작업 유닛들, 청크들 또는 작업 세트들 내의 알고리즘을 실행하여, 각각의 반복에 의해

[0094] a. 현재 작업 세트에 대한 파라미터가 계산되고,

[0095] b. 벡터 실행 유닛으로의 입력 데이터 전달이 개시되고,

[0096] c. 벡터 실행 유닛으로부터의 출력 데이터의 전달이 개시된다.

[0097] 스칼라 스레드의 대표적인 실행 모델은 "파이어 앤드 포겟(fire-and-forget"이다. 파이어 앤드 포겟이라는 용어는, 비디오 기저 대역 처리 애플리케이션에 대한 대표적인 모델에 대해 커맨드 및 데이터가 스칼라 실행 유닛(201)으로부터 (예를 들어, 커맨드 FIFO(225)를 통해) 벡터 실행 유닛(202)으로 전송되고 알고리즘이 완료될 때까지 벡터 실행 유닛(202)으로부터 리턴 데이터가 존재하지 않는 속성을 지칭한다.

[0098] 도 3의 프로그램(300) 예에서, 스칼라 실행 유닛(201)은 커맨드 FIFO(225) 내에 어떠한 공간도 더 이상 존재하지 않을 때까지(예를 들어, lend_of_alg &!cmd_fifo_full) 벡터 실행 유닛(202)에 대한 스케쥴링 작업을 유지한다. 스칼라 실행 유닛(201)에 의해 스케쥴링된 작업은 파라미터를 계산하고 파라미터를 벡터 서브루틴으로 전송한 후, 벡터 서브루틴을 호출하여 작업을 수행한다. 벡터 실행 유닛(202)에 의한 서브루틴(예를 들어, vector_funcB)의 실행은 주로 주 메모리(예를 들어, 시스템 메모리 115)로부터의 지연을 숨기기 위하여 적시에 지연된다. 따라서, 비디오 프로세서(111)의 아키텍처는 명령 및 데이터 트래픽 양자에 대해 벡터 실행 유닛(202) 측 상의 지연 보상 메카니즘을 제공한다. 이러한 지연 보상 메카니즘은 후술한다.

[0099] 소프트웨어 프로그램(300)의 예는 2개 이상의 파이프라인(예를 들어, 도 2의 벡터 데이터 경로(221) 및 제2 벡터 데이터 경로(231))이 존재하는 사례에서 더 복잡하다는 점에 유의해야 한다. 마찬가지로, 소프트웨어 프로그램(300)의 예는 프로그램(300)이 2개의 벡터 실행 파이프라인을 가진 컴퓨터 시스템에 대해 작성되지만 단일 벡터 실행 파이프라인을 가진 시스템 상에서 실행될 수 있는 능력을 여전히 보유하는 상황에 대해 더 복잡할 것이다.

[0100] 따라서, 도 2 및 도 3의 설명에서 전술한 바와 같이, 스칼라 실행 유닛(201)은 벡터 실행 유닛(202) 상에서의 연산 개시를 담당한다. 일 실시예에서, 스칼라 실행 유닛(201)에서 벡터 실행 유닛(202)으로 전달된 커맨드는 아래의 주요 타입을 갖는다.

[0101] 1. 메모리로부터 벡터 실행 유닛(202)의 데이터 RAM으로 현재의 작업 설정 데이터를 전달하기 위해 스칼라 실행 유닛(201)에 의해 개시되는 판독 커맨드(예를 들어, memRd)

[0102] 2. 스칼라 실행 유닛(201)에서 벡터 실행 유닛(202)으로 전달되는 파라미터

[0103] 3. 실행될 벡터 서브루틴의 PC(예를 들어, 프로그램 카운터) 형태의 실행 커맨드

[0104] 4. 벡터 연산의 결과를 메모리로 복사하기 위해 스칼라 실행 유닛(201)에 의해 개시되는 기입 커맨드(예를 들어, memWr)

[0105] 일 실시예에서, 벡터 실행 유닛(202)은 이들 커맨드의 수신시에 즉시 메모리 인터페이스(203)에 대해 memRd 커맨드를 스케쥴링한다(예를 들어, 프레임 버퍼(205)로부터 요청된 데이터를 판독하기 위해). 벡터 실행 유닛(202)은 또한 실행 커맨드를 검사하고, (캐시(222) 내에 존재하지 않는 경우) 실행될 벡터 서브루틴을 프리페치한다.

[0106] 이 상황에서 벡터 실행 유닛(202)의 목적은 벡터 실행 유닛(202)이 현재의 실행 커맨드 상에서 작업하면서 다음 소수의 실행 커맨드의 명령 및 데이터 스트림을 미리 스케쥴링하는 것이다. 사전 스케쥴링 특징은 메모리 위치들로부터 명령/데이터를 페치하는 데 수반되는 지연을 효과적으로 숨긴다. 이러한 판독 요청을 재빨리 행하기 위하여, 벡터 실행 유닛(202), 데이터 저장 메모리(예를 들어, 223), 및 명령 캐시(예를 들어, 222)는 고속 최적 하드웨어를 이용하여 구현된다.

[0107] 전술한 바와 같이, 데이터 저장 메모리(예를 들어, 223)는 벡터 실행 유닛(202)의 작업 RAM으로서 기능한다. 스칼라 실행 유닛(201)은 데이터 저장 메모리가 FIFO의 집합인 것처럼 인식하고 그와 상호작용한다. FIFO는 비디오 프로세서(111)가 동작하는 "스트림"을 포함한다. 일 실시예에서, 스트림은 일반적으로 스칼라 실행 유닛(201)이 전송(예를 들어, 벡터 실행 유닛(202)에 대해)을 개시하는 입출력 FIFO이다. 전술한 바와 같이, 스칼라 실행 유닛(201) 및 벡터 실행 유닛(202)의 동작은 분리된다.

[0108] 입출력 스트림이 가득 찬 경우, 벡터 제어 유닛(220) 내의 DMA 엔진이 커맨드 FIFO(225)의 처리를 중지한다.

이것은 곧 커맨드 FIFO(225)가 가득 차게 한다. 스칼라 실행 유닛(201)은 커맨드 FIFO(225)가 가득 찬 때 벡터 실행 유닛(202)에 대한 추가 작업의 발행을 중지한다.

[0109] 일 실시예에서, 벡터 실행 유닛(202)은 입출력 스트림 외에 중간 스트림을 필요로 할 수 있다. 따라서, 전체 데이터 저장 메모리(223)는 스칼라 실행 유닛(201)과의 상호작용과 관련하여 스트림들의 집합으로 보여질 수 있다.

[0110] 도 4는 본 발명의 일 실시예에 따른 비디오 프로세서를 이용한 비디오와의 서브 픽처 혼합의 일례를 나타낸다. 도 4는 비디오 표면이 서브 픽처와 혼합된 후 ARGB 표면으로 변환되는 예시적인 사례를 나타낸다. 표면을 포함하는 데이터는 프레임 버퍼 메모리(205) 내에 휘도(Luma) 파라미터(412) 및 색도(Chroma) 파라미터(413)로서 위치한다. 도시된 바와 같이, 서브 픽처 픽셀 요소(414)도 프레임 버퍼 메모리(205) 내에 위치한다. 벡터 서브 루틴 명령 및 파라미터(411)는 도시된 바와 같이 메모리(205) 내에 인스턴스화되어 있다.

[0111] 일 실시예에서, 각각의 스트림은 "타일"이라고 하는 데이터의 작업 2D 청크들의 FIFO를 포함한다. 이러한 실시예에서, 벡터 실행 유닛(202)은 각각의 스트림에 대해 판독 타일 포인터 및 기입 타일 포인터를 유지한다. 예를 들어, 입력 스트림에 대해, 벡터 서브 루틴이 실행될 때, 벡터 서브 루틴은 현재(판독) 타일로부터 소비하거나 판독할 수 있다. 그 배경에서, memRd 커맨드에 의해 데이터가 현재(기입) 타일로 전송된다. 벡터 실행 유닛은 또한 출력 스트림에 대한 출력 타일을 생성할 수 있다. 이어서, 이들 타일은 실행 커맨드에 이어지는 memWr0 커맨드에 의해 메모리로 이동된다. 이것은 타일들을 효과적으로 프리페치하며, 타일들이 조작될 준비가 되게 하여 지연을 효과적으로 숨긴다.

[0112] 도 4의 서브 픽처 혼합 예에서, 벡터 데이터 경로(221)는 벡터 서브 루틴 명령 및 파라미터(411)(예를 들어, &v_subp_blend)의 인스턴스화된 인스턴스에 의해 구성된다. 이것은 라인 421로 도시되어 있다. 스칼라 실행 유닛(201)은 표면들의 청크들(예를 들어, 타일들)을 판독하고, 이들을 DMA 엔진(401)(예를 들어, 메모리 인터페이스(203) 내의)을 이용하여 데이터 저장 메모리(223)에 로딩한다. 로딩 동작은 라인 422, 423 및 424에 의해 도시된다.

[0113] 도 4를 계속 참조하면, 다수의 입력 표면이 존재하므로, 다수의 입력 스트림이 유지되는 것이 필요하다. 각각의 스트림은 대응 FIFO를 갖는다. 각각의 스트림은 상이한 수의 타일을 가질 수 있다. 도 4의 예는 서브 픽처 표면이 시스템 메모리(115)(예를 들어, 서브 픽처 픽셀 요소 414) 내에 있고, 따라서 추가 버퍼링(예를 들어, n, n+1, n+2, n+3 등)을 갖는 반면, 비디오 스트림(예를 들어, 휘도(412), 색도(413) 등)이 보다 적은 수의 타일을 가질 수 있는 사례를 나타낸다. 사용되는 버퍼/FIFO의 수는 스트림이 겪는 지연의 정도에 따라 조정될 수 있다.

[0114] 전술한 바와 같이, 데이터 저장 메모리(223)는 지연을 숨기기 위하여 예측 프리페치 방법을 이용한다. 이 때문에, 데이터가 적절한 벡터 데이터 경로 실행 하드웨어(예를 들어, FIFO n, n+1, n+2 등으로 도시)에 대해 프리페치될 때, 스트림이 둘 이상의 타일 내에 데이터를 가질 수 있다.

[0115] 데이터 저장 메모리가 로딩되면, FIFO는 벡터 데이터 경로 하드웨어(221)에 의해 액세스되고, 벡터 서브 루틴(예를 들어, 430)에 의해 조작된다. 벡터 데이터 경로 동작의 결과는 출력 스트림(403)을 포함한다. 이 출력 스트림은 스칼라 실행 유닛(201)에 의해 DMA 엔진(401)을 통해 프레임 버퍼 메모리(205)(예를 들어, ARGB_OUT 415)로 복사된다. 이것은 라인 425로 도시되어 있다.

[0116] 이와 같이, 본 발명의 실시예들은 데이터 저장 메모리가 복수의 메모리 타일로서 추상화되는 스트림 처리의 중요한 양태를 이용한다. 따라서, 스트림은 순차적으로 액세스되는 타일들의 집합으로 볼 수 있다. 스트림들은 데이터를 프리페치하는 데 사용된다. 이 데이터는 타일의 형태를 갖는다. 타일들은 데이터가 발원되는 특정 메모리 소스(예를 들어, 시스템 메모리, 프레임 버퍼 메모리 등)로부터의 지연을 숨기기 위해 프리페치된다. 마찬가지로, 스트림들은 상이한 위치들(예를 들어, 벡터 실행 유닛에 대한 캐시, 스칼라 실행 유닛에 대한 캐시, 프레임 버퍼 메모리, 시스템 메모리 등)을 향할 수 있다. 스트림들의 또 하나의 특징은 이들이 일반적으로 예측 프리페치 모드로 타일들을 액세스한다는 것이다. 전술한 바와 같이, 지연이 클수록, 프리페치는 더 어렵고, 스트림에 대해 사용되는 버퍼링은 더 많다(예를 들어, 도 4에 도시된 바와 같이).

[0117] 도 5는 본 발명의 일 실시예에 따른 벡터 실행 유닛의 내부 컴포넌트들을 나타내는 도면이다. 도 5는 프로그램의 관점에서 벡터 실행 유닛(202)의 다양한 기능 유닛 및 레지스터/SRAM 자원의 배열을 나타낸다.

[0118] 도 5의 실시예에서, 벡터 실행 유닛(202)은 비디오 기저 대역 처리의 성능 및 다양한 코덱(압축-압축 해제 알고리즘)의 실행을 위해 최적화된 VLIW 디지털 신호 프로세서를 포함한다. 따라서, 벡터 실행 유닛(202)은 비디오

처리/코덱 실행의 효율 향상을 지향하는 다수의 속성을 갖는다.

[0119] 도 5의 실시예에서, 속성들은 다음을 포함한다.

[0120] 1. 다수의 벡터 실행 파이프라인의 포함을 위한 옵션을 제공함에 의한 스케일링 가능한 성능

[0121] 2. 파이프당 2개의 데이터 어드레스 생성기(DAG)의 할당

[0122] 3. 메모리/레지스터 피연산자

[0123] 4. 2D (x,y) 포인터/반복기

[0124] 5. 디프(deep) 파이프라인(예를 들어, 11-12) 스테이지

[0125] 6. 스칼라(정수)/분기 유닛

[0126] 7. 가변 명령 폭(장/단 명령)

[0127] 8. 피연산자 추출을 위한 데이터 정렬기

[0128] 9. 대표적인 피연산자 및 결과의 2D 데이터 경로(4x4) 형상

[0129] 10. 원격 프로시저 호출을 실행하는 스칼라 실행 유닛에 대한 슬레이브 벡터 실행 유닛

[0130] 일반적으로, 벡터 실행 유닛(202)의 프로그래머의 뷰는 2개의 DAG(503)를 구비한 SIMD 데이터 경로로서이다. 명령들은 VLIW 방식으로 발행되며(예를 들어, 명령들은 벡터 데이터 경로(504) 및 어드레스 생성기(503)에 대해 동시에 발행된다), 명령 디코더(501)에 의해 디코딩되어 적절한 실행 유닛으로 발송된다. 명령들은 가변 길이를 가지며, 가장 일반적으로 사용되는 명령들은 짧은 형태로 인코딩된다. 풀 명령 세트는 VLIW 타입의 명령과 같이 긴 형태로 이용 가능하다.

[0131] 레전드(502)는 3개의 VLIW 명령을 가진 3개의 클럭 사이클을 나타낸다. 레전드(510)에 따르면, VLIW 명령들(502)의 최상위는 2개의 어드레스 명령(예를 들어, 2개의 DAG(503)에 대해) 및 벡터 데이터 경로(504)에 대한 하나의 명령을 포함한다. 중간 VLIW 명령은 하나의 정수 명령(예를 들어, 정수 유닛 505에 대해), 하나의 어드레스 명령 및 하나의 벡터 명령을 포함한다. 최하위 VLIW 명령은 분기 명령(예를 들어, 분기 유닛(506)에 대해), 하나의 어드레스 명령 및 하나의 벡터 명령을 포함한다.

[0132] 벡터 실행 유닛은 단일 데이터 파이프 또는 다수의 데이터 파이프를 갖도록 구성될 수 있다. 각각의 데이터 파이프는 로컬 RAM(예를 들어, 데이터 저장 메모리 511), 크로스바(516), 2개의 DAG(503), 및 SIMD 실행 유닛(예를 들어, 벡터 데이터 경로 504)으로 구성된다. 도 5는 하나의 데이터 파이프만이 인스턴스화된 설명을 위한 기본 구성을 나타낸다. 2개의 데이터 파이프가 인스턴스화될 때, 이들은 개별 스레드로서 또는 협동 스레드로서 실행될 수 있다.

[0133] 6개의 상이한 포트(예를 들어, 4개의 판독 및 2개의 기입)가 어드레스 레지스터 파일 유닛(515)에 의해 액세스될 수 있다. 이들 레지스터는 스칼라 실행 유닛으로부터 또는 정수 유닛(505) 또는 어드레스 유닛(503)의 결과로부터 파라미터를 수신한다. DAG(503)는 또한 수집 제어기로서 기능하며, 데이터 저장 메모리(511; 예를 들어 RA0, RA1, RA2, RA3, WA0 및 WA1)의 내용에 어드레스하기 위해 레지스터들의 분배를 관리한다. 크로스바(516)가 주어진 명령을 구현하기 위해 출력 데이터 포트들(R0, R1, R2, R3)을 임의의 순서/조합으로 벡터 데이터 경로(504)에 할당하도록 연결된다. 벡터 데이터 경로(504)의 출력은 지시되는 바와 같이(예를 들어, W0) 데이터 저장 메모리(511)로 피드백될 수 있다. 상수 RAM(517)이 정수 유닛(505)에서 벡터 데이터 경로(504) 및 데이터 저장 메모리(511)로 자주 사용되는 피연산자를 제공하는 데 사용된다.

[0134] 도 6은 본 발명의 일 실시예에 따른 메모리(600)의 복수의 뱅크(601-604) 및 대칭 타일 어레이를 가진 데이터 저장 메모리(610)의 레이아웃을 나타낸다. 도 6에 도시된 바와 같이, 설명을 위해, 데이터 저장 메모리(610)의 일부만이 도시된다. 데이터 저장 메모리(610)는 논리적으로 타일들의 어레이(또는 어레이들)를 포함한다. 각각의 타일은 4x4 형태의 서브 타일들의 어레이이다. 물리적으로, 메모리(600)로 도시된 바와 같이, 데이터 저장 메모리(610)는 "N" 개의 물리적 메모리 뱅크(예를 들어, 뱅크 601-604)의 어레이 내에 저장된다.

[0135] 또한, 데이터 저장 메모리(610)는 스트림 내의 논리적 타일을 시각적으로 도시한다. 도 6의 실시예에서, 이 타일은 16 바이트의 높이와 16 바이트의 폭을 갖는다. 이 타일은 서브 타일들의 어레이이다(이 예에서는 4x4). 각각의 서브 타일은 물리적 뱅크 내에 저장된다. 8개의 물리적 메모리 뱅크(예를 들어, 뱅크 0 내지 7)가 존재하는 경우, 이것은 도 6에서 각각의 4x4 서브 타일 내의 수로 표시된다. 뱅크들 내의 서브 타일들의 구성은 서

브 타일들의 2x2 배열 내에 공통 뱅크가 존재하지 않도록 이루어진다. 이것은 어떠한 뱅크 충돌 없이 임의의 비정렬 액세스(예를 들어, x 및 y 방향 양자에서)를 가능하게 한다.

[0136] 뱅크들(601-604)은 각각의 뱅크의 상이한 타일들로의 액세스를 지원하도록 구성된다. 예를 들어, 하나의 사례에서, 크로스바(516)는 뱅크(601)로부터 2x4 타일 세트(예를 들어, 뱅크(601)의 최초 2행)를 액세스할 수 있다. 다른 사례에서, 크로스바(516)는 2개의 인접 뱅크로부터 1x8 타일 세트를 액세스할 수 있다. 마찬가지로, 다른 사례에서, 크로스바(516)는 2개의 인접 뱅크로부터 8x1 타일 세트를 액세스할 수 있다. 각각의 사례에서, DAG/콜렉터(503)는 뱅크들이 크로스바(516)에 의해 액세스될 때 타일들을 수신하여, 이 타일들을 클럭 단위로 벡터 데이터 경로(504)의 프론트 엔드에 제공할 수 있다.

[0137] 이러한 방식으로, 본 발명의 실시예들은 집적 회로 실리콘 다이 면적, 트랜지스터 수, 메모리 속도 요건 등을 효율적으로 이용하면서 정교한 비디오 처리 기능들을 지원하는 새로운 비디오 프로세서 아키텍처를 제공한다. 본 발명의 실시예들은 다수의 비디오 스트림을 처리하기 위하여 높은 연산 밀도를 유지하며, 쉽게 스케일링이 가능하다. 본 발명의 실시예들은 예를 들어 MPEG-2/WMV9/H.264 인코드 지원(예를 들어, 루프내 디코더), MPEG-2/WMV9/H.264 디코드(예를 들어, 포스트 엔트로피 디코딩), 및 루프내/루프외 비블록킹 필터와 같은 다수의 정교한 비디오 처리 동작을 제공할 수 있다.

[0138] 본 발명의 실시예들에 의해 제공되는 추가적인 비디오 처리 동작은 예를 들어 향상된 모션 적응 디인터레이싱, 인코딩을 위한 입력 잡음 필터링, 다위상 스케일링/리샘플링, 및 서브 픽처 합성을 포함한다. 본 발명의 비디오 프로세서 아키텍처는 또한 예를 들어 칼라 공간 변환, 칼라 공간 조정, 샤프닝, 히스토그램 조정과 같은 픽셀 포인트 조작, 및 다양한 비디오 표면 포맷 변환과 같은 소정의 비디오 프로세서-증폭기(프로캠프) 애플리케이션에 이용될 수 있다.

[0139] 광범위하게 그리고 제한 없이, 본 발명은 다음을 개시하였다. 비디오 처리 동작을 실행하기 위한 지연 감내 시스템이 설명되었다. 이 시스템은 비디오 프로세서와 호스트 CPU 간의 통신을 구현하기 위한 호스트 인터페이스, 호스트 인터페이스에 연결되고 스칼라 비디오 처리 동작을 실행하도록 구성된 스칼라 실행 유닛, 및 호스트 인터페이스에 연결되고 벡터 비디오 처리 동작을 실행하도록 구성된 벡터 실행 유닛을 포함한다. 벡터 실행 유닛이 메모리 커맨드 FIFO를 액세스함으로써 요구 구동 기반으로 동작하게 하기 위한 커맨드 FIFO가 포함된다. 비디오 프로세서와 프레임 버퍼 메모리 간의 통신을 구현하기 위한 메모리 인터페이스가 포함된다. 복수의 상이한 메모리 위치 간의 DMA 전달을 구현하고 벡터 실행 유닛에 대한 데이터 및 명령을 커맨드 FIFO에 로딩하기 위한 DMA 엔진이 메모리 인터페이스 내에 구축된다. 비디오 처리 동작을 실행하기 위한 비디오 프로세서가 설명되었다. 비디오 프로세서는 비디오 프로세서와 호스트 CPU 간의 통신을 구현하기 위한 호스트 인터페이스를 포함한다. 비디오 프로세서와 프레임 버퍼 메모리 간의 통신을 구현하기 위한 메모리 인터페이스가 포함된다. 스칼라 실행 유닛이 호스트 인터페이스 및 메모리 인터페이스에 연결되고, 스칼라 비디오 처리 동작을 실행하도록 구성된다. 벡터 실행 유닛이 호스트 인터페이스 및 메모리 인터페이스에 연결되고, 벡터 비디오 처리 동작을 실행하도록 구성된다. 비디오 처리 동작을 실행하기 위한 비디오 프로세서용의 다차원 데이터 경로 처리 시스템이 설명되었다. 비디오 프로세서는 스칼라 비디오 처리 동작을 실행하도록 구성된 스칼라 실행 유닛 및 벡터 비디오 처리 동작을 실행하도록 구성된 벡터 실행 유닛을 포함한다. 벡터 실행 유닛에 대한 데이터를 저장하기 위한 데이터 저장 메모리가 포함된다. 데이터 저장 메모리는 어레이로 배열된 대칭 뱅크 데이터 구조들을 가진 복수의 타일을 포함한다. 뱅크 데이터 구조들은 각각의 뱅크의 상이한 타일들로의 액세스를 지원하도록 구성된다. 비디오 처리 동작을 실행하기 위한 비디오 프로세서용의 스트림 기반 메모리 액세스 시스템이 설명되었다. 비디오 프로세서는 스칼라 비디오 처리 동작을 실행하도록 구성된 스칼라 실행 유닛 및 벡터 비디오 처리 동작을 실행하도록 구성된 벡터 실행 유닛을 포함한다. 스칼라 실행 유닛 및 벡터 실행 유닛에 대한 데이터를 저장하기 위한 프레임 버퍼 메모리가 포함된다. 스칼라 실행 유닛 및 벡터 실행 유닛과 프레임 버퍼 메모리 간의 통신을 설정하기 위한 메모리 인터페이스가 포함된다. 프레임 버퍼 메모리는 복수의 타일을 포함한다. 메모리 인터페이스는 벡터 실행 유닛 또는 스칼라 실행 유닛에 대해 제1 순차적 타일 액세스를 포함하는 제1 스트림을 구현하고 제2 순차적 타일 액세스를 포함하는 제2 스트림을 구현한다.

[0140] 본 발명의 특정 실시예들의 전술한 설명은 예시 및 설명을 위해 제공되었다. 이들은 고갈적이거나 본 발명을 개시된 특정 형태로 한정하려는 것은 아니며, 위의 가르침에 비추어 많은 수정 및 변형이 가능하다. 실시예들은 발명의 원리 및 그의 실제 이용을 가장 잘 설명하여 이 분야의 전문가들이 고려된 특정 이용에 적합할 때 발명 및 다양한 수정이 이루어진 다양한 실시예를 가장 잘 이용할 수 있도록 하기 위하여 선택되고 설명되었다. 본 발명의 범위는 첨부된 청구범위 및 그의 균등물에 의해 정의되는 것을 의도한다.

[0141] 부록 A

[0142] **1 개요**

[0143] VP2는 스칼라 제어 프로세서에 연결된 VLIW SIMD 비디오 DSP이다. 그의 주 초점은 비디오 코덱 및 비디오 기저대역 처리이다.

[0144] **1.1 VP2.0의 정신**

[0145] - 효율: VP2.0은 perf/mm2 및 perf/mW의 면에서 비디오 애플리케이션들에 대한 연산 효율적인 머신이다.

[0146] - 프로그래밍 가능성: 매우 프로그래머블하고, 쉽게 호환 가능하며, 머신을 프로그래밍하는 데 보다 안전하다.

[0147] - 스케일링 가능성: VP2.0 설계/아키텍처는 다수의 애플리케이션 영역에 대한 성능 요건과 매칭되도록 스케일링 가능해야 한다.

[0148] **1.2 설계 목표**

[0149] - 연산 밀도

[0150] - VP1.0보다 상당한 perf/mm2 이점을 제공한다.

[0151] - H.264와 같은 새로운 애플리케이션 영역의 효율적인 구현

[0152] - SW 개발자의 부담을 덜어주는 HW에서의 지연 감내

[0153] - 메모리 액세스 및 연산을 재배열함으로써 데이터 페치 지연을 숨김

[0154] - 명령 스트림의 자동 프리페치

[0155] - 데이터 경로 지연의 숨김

[0156] - 중간 결과의 선택적 전달

[0157] - 스트리밍 연산 모델

[0158] - 스케일링 가능성

[0159] - 구조적으로 VP2 벡터 유닛은 2x 만큼 위로, 그리고 1/2x 만큼 아래로 그의 데이터 경로를 스케일링할 수 있다.

[0160] - 선택적 리파이프라이닝에 의해 주파수 개선이 달성될 수 있다.

[0161] **1.3 애플리케이션 타겟**

[0162] VP2.0 설계 및 명령 세트는 다음의 애플리케이션을 매우 효율적으로 수행하도록 최적화된다.

[0163] - mpeg2/wmv9/H.264 인코드 지원(루프내 디코더)

[0164] - mpeg2/wmv9/H.264 디코드(포스트 엔트로피 디코딩)

[0165] - 루프내/루프외 디블록킹 필터

[0166] - 향상된 모션 적응 디인터레이싱

[0167] - 인코딩을 위한 입력 잡음 필터링

[0168] - 다위상 스케일링/리샘플링

[0169] - 서브 픽처 합성

[0170] - 프로캠프, 칼라 공간 변환, 조정, 샤프닝, 히스토그램 조정과 같은 픽셀 포인트 동작들 등

[0171] - 다양한 비디오 표면 포맷 변환 지원

[0172] 구조적으로 VP2.0은 다음 영역에서 효율적일 수 있다.

[0173] - 2D 원시 함수, 블릿, 로테이트 등

[0174] - 정밀 기반 소프트웨어 모션 추정 알고리즘

[0175] - 16/32 비트 MAC 애플리케이션

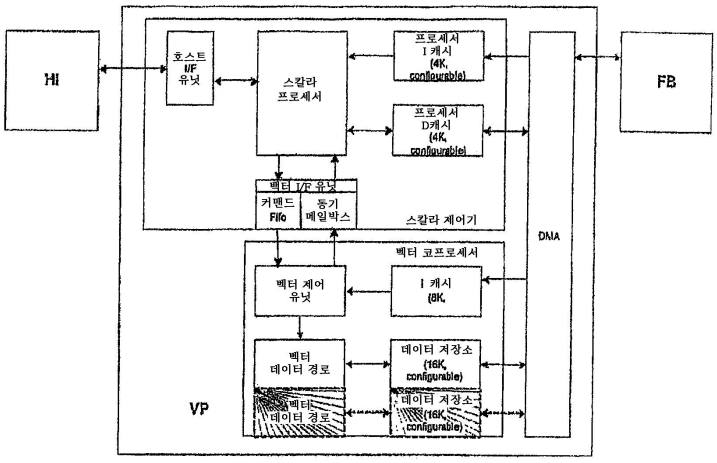
[0176] **2 최상위 레벨 아키텍처**

[0177] VP2.0 머신은 스칼라 및 벡터 프로세서로 분할되어 있다. 벡터 프로세서는 스칼라 프로세서에 대해 슬레이브 코프로세서로서 동작한다. 스칼라 프로세서는 벡터 프로세서에 제어 스트림(파라미터, 서브 루틴 독립 변수)을 공급하고 벡터 프로세서로의 데이터 I/O를 관리하는 것을 담당한다. 알고리즘의 모든 제어 흐름은 스칼라 머신 상에서 실행되는 반면, 실제의 픽셀/데이터 처리 동작은 벡터 프로세서 상에서 행해진다.

[0178] 스칼라 프로세서는 대표적인 RISC 스타일 스칼라이고, 벡터 코프로세서는 1 또는 2개의 SIMD 파이프(각 SIMD 파이프는 16 픽셀 데이터 경로를 가짐)를 가진 SIMD 머신이다. 따라서, 벡터 코프로세서는 순수 연산 능력으로서 결과의 최대 32 픽셀을 생성할 수 있다.

[0179] 스칼라 프로세서는 메모리 맵핑된 커맨드 FIFO를 이용하여 벡터 코프로세서에 기능 호출들을 전송한다. 코프로세서 커맨드는 이 FIFO에 큐잉된다. 스칼라 프로세서는 이 FIFO를 이용하여 벡터 프로세서로부터 완전히 분리된다. 스칼라 프로세서는 그 자신의 클럭 상에서 실행될 수 있다. 벡터 프로세서는 요구 구동 유닛으로서 동작한다.

[0180] VP2.0의 최상위 레벨도가 아래에 주어진다.

[0181] 

[0182] VP2.0은 개별 스칼라 코드 및 벡터 코드로 컴파일된 후, 후에 함께 링크될 수 있다. 별개로, 벡터 함수들은 개별적으로 작성될 수 있으며, 서브 루틴 또는 라이브러리 함수로서 스칼라 스레드에 제공될 수 있다. 스칼라 프로세서는 그 자신의 명령 및 데이터 캐시를 갖는다. 벡터 유닛도 명령 캐시 및 데이터 RAM(데이터 저장 메모리로 지칭됨)을 갖는다. 이들 2개의 엔진은 분리되고, FIFO를 통해 통신한다.

[0183] **3 단순 프로그래밍 모델**

[0184] VP2.0에 대한 가장 간단한 프로그래밍 모델은 벡터 슬레이브 코프로세서 상에서 서브 루틴 호출을 실행하는 스칼라 제어 프로그램이다. 여기서 프로그래머는 문제를 2개의 스레드로 분해하였다는 고유한 가정이 존재한다. 스칼라 프로세서 상에서 실행되는 스레드는 재빨리 작업 파라미터를 계산하고 이들을 주 기계인 벡터 프로세서에 공급한다. 이들 2개의 스레드를 위한 프로그램은 개별적으로 작성되고 컴파일되는 것으로 예상된다.

[0185] 스칼라 스레드는 다음을 담당한다.

[0186] 1. 호스트 유닛과의 인터페이싱 및 클래스 인터페이스 구현

[0187] 2. 벡터 유닛의 초기화, 셋업 및 구성

[0188] 3. 루프 내의 작업 유닛들, 청크들 또는 작업 세트들 내의 알고리즘을 실행하여 각각의 반복은

[0189] a. 현재 작업 세트에 대한 파라미터를 계산한다.

[0190] b. 입력 데이터의 벡터 프로세서로의 전달을 개시한다.

[0191] c. 벡터 프로세서로부터의 출력 데이터의 전달을 개시한다.

[0192] 스칼라 스레드의 대표적인 실행 모델은 파이어 앤드 포겟이다. 이것은 벡터 코프로세서로부터 리턴 데이터가 존재하지 않는 비디오 기저 대역 처리에 대한 대표적인 모델일 것으로 예상된다. 스칼라 프로세서는 커맨드 FIFO 내에 공간이 존재할 때까지 벡터 프로세서에 대한 스케쥴링 작업을 유지할 것이다. 벡터 프로세서에 의한 서브 루틴의 실행은 주로 주 메모리로부터의 지연으로 인해 적시에 지연된다. 따라서, 벡터 측에 지연 보상 메카니즘을 제공하는 것이 중요하다. VP2.0에서, 벡터 프로세서는 명령 및 데이터 트래픽 양자에 대한 지연 보상을 제공한다. 그를 위한 메카니즘이 섹션에서 요약된다.

[0193] 대표적인 VP 프로그램은 아래와 같을 것이다.

[0194]
```
스칼라 제어 스레드                    벡터 스레드
parse_methods();                      vector_functionA {
setup_vector_engine();
                                      }
while (!end_of_alg &                  vector_functionB{
!cmd_fifo_full ) {
    compute_param_for_tile(i)
    send_params_to_vector();
    fetch-data-to-vector();
    vector-function-call();
    write-data-from-vector();
}                                     }
                                      .....
```

[0195] 보다 복잡한 프로그래밍 모델은 2개의 데이터 파이프를 가질 때이다. 또는, 2개의 파이프에 대한 코드를 작성하고 이를 하나의 데이터 파이프 머신 상에서 실행시키기를 원할 때이다. 그를 위한 프로그래밍 모델은 섹션 6에서 조사된다.

[0196] **4 스트리밍 모델**

[0197] 앞에서 요약된 바와 같이, 스칼라 엔진은 벡터 프로세서 상의 연산 개시를 담당한다. 스칼라 엔진에서 벡터 엔진으로 전달되는 커맨드는 다음 주요 타입을 갖는다.

[0198] 1. 현재 작업 세트 데이터를 메모리에서 벡터 엔진의 데이터 RAM으로 전달하기 위해 스칼라에 의해 개시되는 판독 커맨드(memRd)

[0199] 2. 스칼라에서 벡터로 전달되는 파라미터

[0200] 3. 실행될 벡터 서브 루틴의 PC 형태의 실행 커맨드

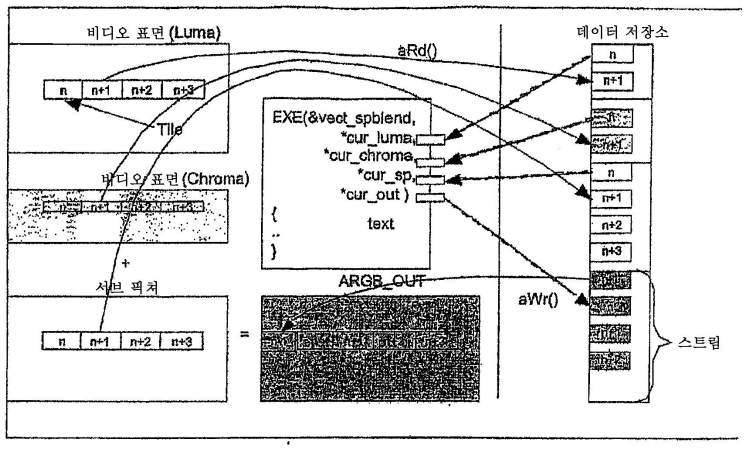
[0201] 4. 벡터 연산의 결과를 메모리에 복사하기 위해 스칼라에 의해 개시되는 기입 커맨드(memWr)

[0202] 벡터 프로세서는 이들 커맨드의 수신시에 즉시 프레임 버퍼(FB) 인터페이스에 대해 memRd 커맨드를 스케쥴링한다. 또한, 실행 커맨드를 검사하고, 실행될 벡터 서브 루틴을 프리페치한다(캐시 내에 존재하지 않는 경우). 하나의 목적은 벡터 엔진이 현재 실행 커맨드 상에서 작업하고 있는 동안 다음 소수의 실행 커맨드의 명령 및 데이터 스트림을 미리 스케쥴링하는 것이다. 이러한 판독 요청을 재빨리 행하기 위하여, 벡터 엔진은 하드웨어에서 데이터 저장 메모리 및 명령 캐시를 관리한다.

[0203] 데이터 저장 메모리는 벡터 프로세서의 작업 RAM이다. 스칼라 프로세서는 이 데이터 저장 메모리를 FIFO 또는 스트림의 집합으로 본다. 스트림들은 본질적으로 스칼라가 전달을 개시하는 입출력 FIFO이다. 입출력 스트림이 가득 차면, 벡터 DMA 엔진은 스칼라로부터 커맨드 FIFO의 처리를 중지하며, 곧 가득 차게 된다. 따라서, 스칼라는 벡터 엔진에 대한 더 이상의 작업 발행을 중지한다. 입출력 스트림 외에, 벡터는 중간 스트림을 필요로 할 수 있다. 따라서, 전체 데이터 저장 메모리는 스칼라 측으로부터 스트림의 집합으로서 보여질 수 있다. 각각의 스트림은 타일이라고 하는 작업 2D 청크들의 FIFO이다. 벡터 프로세서는 각각의 스트림에 대해 판독 타일 포인터 및 기입 타일 포인터를 유지한다. 입력 스트림에 대해, 벡터 서브 루틴이 실행될 때, 현재(판독) 타일로부터 소비하거나 판독할 수 있다. 그 배경에서, 데이터는 memRd 커맨드에 의해 현재(기입) 타일로 전송된다. 벡터 프로세서는 또한 출력 스트림에 대한 출력 타일을 생성할 수 있다. 이어서, 이 타일들은 실행 커맨드에 이어지는 memWr() 커맨드에 의해 메모리로 이동된다.

[0204] 이 모델은 비디오와의 서브 픽처 혼합에 대한 예로 설명된다. 예를 들어, 비디오 표면(예를 들어, NV12 포맷)이 서브 픽처와 혼합된 후 ARGB 표면으로 변환되는 간단한 예를 고려한다. 이들 표면은 메모리 내에 위치한다. 스칼라 프로세서는 이들 표면의 청크들(타일들)을 판독하여 이들을 데이터 저장 메모리 내에 로딩한다. 다수의

입력 표면이 존재하므로, 다수의 입력 스트림을 유지해야 한다. 각각의 스트림은 상이한 수의 타일을 가질 수 있는 반면(예를 들어, 이 예에서, 서브 픽처 표면이 시스템 메모리 내에 있는 것으로 가정할 수 있고, 따라서 그것을 더 버퍼링해야 한다), 비디오 스트림은 보다 적은 수의 타일들을 가질 수 있다.



[0205]

[0206] **5 벡터 코프로세서**

[0207] VP2의 벡터 코프로세서는 비디오 기저대역 처리 및 코덱을 위해 설계된 VLIW DSP이다. 이 프로세서의 몇 가지 중요한 설계 속성은 다음을 포함한다.

[0208] 1. 스케일링 가능한 성능, 1 또는 2개의 데이터 파이프.

[0209] 2. 각각의 파이프가 2개의 데이터 어드레스 생성기(DAG)를 구비한다.

[0210] 3. 메모리/레지스터 피연산자

[0211] 4. 2D (x,y) 포인터/반복기

[0212] 5. 디프 파이프라인 (11-12) 스테이지

[0213] 6. 스칼라(정수)/분기 유닛

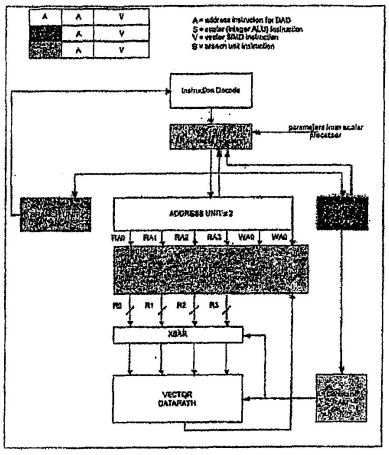
[0214] 7. 가변 명령 폭(긴/짧은 명령)

[0215] 8. 피연산자 추출을 위한 데이터 정렬기

[0216] 9. 대표적인 피연산자 및 결과의 2D 데이터 경로 (4x4) 형상

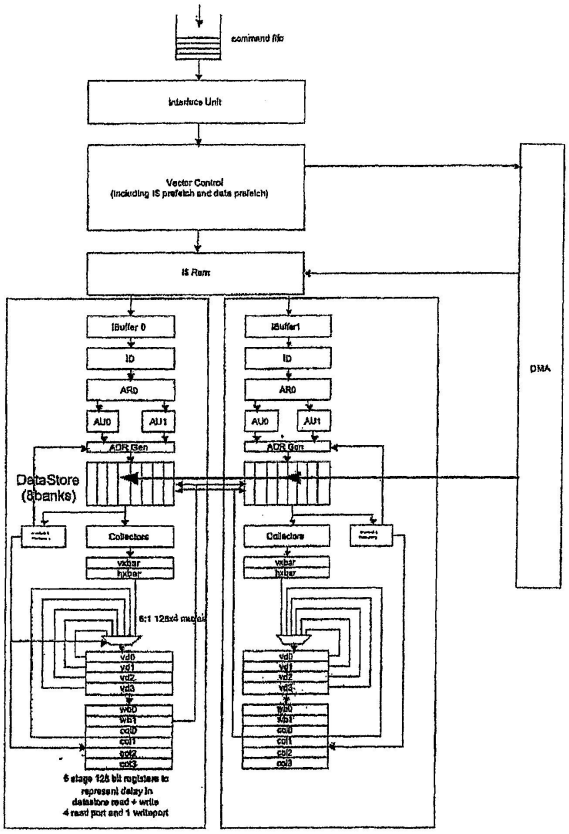
[0217] 10. 스칼라 프로세서에 대한 슬레이브 프로세서, 원격 프로시저 호출을 실행한다.

[0218] 벡터 코프로세서의 프로그래머의 뷰는 가장 간단한 용어로 2개의 DAG를 구비한 SIMD 데이터 경로이다. 명령들은 VLIW 방식으로 발행된다(즉, 명령들은 벡터 데이터 경로 및 어드레스 생성기에 대해 동시에 발행된다). 명령들은 가변 길이를 가지며, 가장 일반적으로 사용되는 명령들은 짧은 형태로 인코딩된다. 풀 명령 세트는 긴 형태로 이용 가능하다. 예를 들어, 프로그래머의 관점에서, 다양한 기능 유닛 및 레지스터/SRAM 자원들의 배열은 아래 나타낸 바와 같다.



[0219]

[0220] 벡터 유닛은 단일 데이터 파이프 또는 듀얼 데이터 파이프를 인스턴스화한다. 각각의 데이터 파이프는 로컬 RAM(데이터 저장 메모리), 2개의 DAG 및 SIMD 실행 유닛으로 구성된다. 기본 구성에 있어서는 하나의 데이터 파이프만이 존재한다. 2개의 데이터 파이프가 존재할 때, 이들은 독립적인 스레드로서 또는 협동 스레드로서 실행될 수 있다. 벡터 프로세서의 완전한 파이프라인도가 아래에 기술된다. 여기서, 전체 구성은 2개의 데이터 파이프를 갖는다.



[0221]

[0222] **6 향상된 프로그래밍 모델**

[0223] 섹션 3에서, 기본 아키텍처를 설명하기 위하여 RPC 모델이 소개되었다. 이 섹션에서는 보다 향상된 개념이 소개된다.

[0224] **6.1 듀얼 데이터 파이프 구성**

[0225] 듀얼 파이프 구성에서는 아래의 프로세서 자원들이 공유된다.

[0226] - 스칼라 제어기

[0227] - 벡터 코프로세서 내의 벡터 제어 유닛

[0228] - 명령/데이터 페치를 위한 DMA 엔진

[0229] - 명령 캐시(듀얼 포트화될 수 있다)

[0230] 다음의 자원들이 복제된다.

[0231] - 데이터 파이프(어드레스/분기/벡터 실행 유닛)

[0232] - 데이터 저장 메모리

[0233] - 레지스터 파일

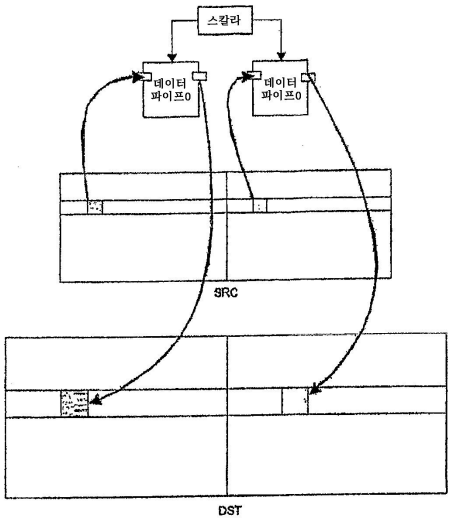
[0234] 다음과 같은 점에 유의해야 한다.

[0235] 1. 1 파이프만을 가진 인스턴스 상에서 2 파이프에 대해 프로그램이 작성될 수 있다. 벡터 제어 유닛은 동일 물리적 파이프 상에 각 파이프에 대한 실행 커맨드를 맵핑할 것이다. 그러나, 양 파이프에 대한 스트림이 단지 하나의 데이터 저장 메모리에만 존재하므로, 데이터 저장소의 크기는 조정되어야 한다. 간단한 방법은 타일 크기를 줄이거나 스트림 내의 타일들의 수를 반감시키는 것이다. 이것은 구성시에 스칼라 스레드에 의해 행해진다. 마이크로 아키텍처 스테이지에서 해결되어야 하는 글로벌 레지스터의 복제 및 스트림 맵핑과 같은 문제가 존재한다.

[0236] 2. 1 파이프에 대해 작성된 프로그램은 2 파이프를 가진 인스턴스 상에서 실행될 수 있다. 그러나, 이 코드는 하나의 파이프 상에서만 실행되고 나머지 파이프는 사용하지 않는다. 머신은 절반이 유휴 상태가 된다.

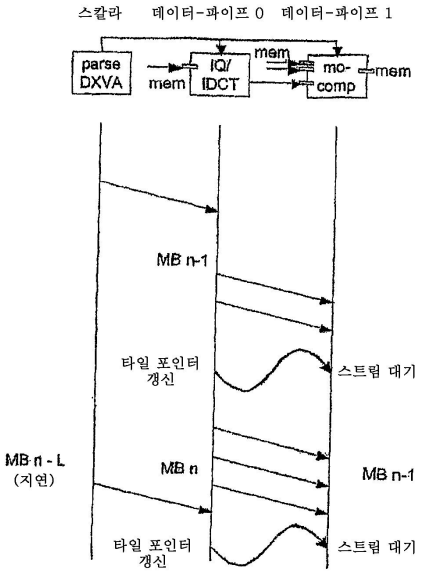
[0237] 3. 2개의 완전히 상이한 스레드를 각각 실행하는 2 파이프에 대해 프로그램이 작성될 수 있다. 멀티 스레드화되지 않은 단일 스칼라만을 구비하므로, 이것은 바람직하지 않을 수 있다. 하나의 스칼라 실행 스레드만을 지원하므로, 이것은 바람직하지 않을 수 있지만, 이 모델은 지원될 수도 있다.

[0238] 4. 동일 스레드를 각각 실행하는 2 파이프에 대해 프로그램이 작성될 수 있다. 이것은 대부분의 비디오 기저대역 처리와 같은 병렬 가능 알고리즘에 대해 예상되는 대표적인 모델이다. 이것은 비디오의 2개의 스트림 또는 2개의 절반 등에서 동작하기 위해 동일 명령 스트림을 사용하는 것을 허용한다. 각각의 데이터 파이프는 그 자신의 실행 유닛 및 데이터 저장소를 갖는다. 스칼라 제어기는 2개의 데이터 파이프를 제공해야 한다. 그러나, 파라미터, 판독 및 기입 커맨드는 서로 관련되며(오프셋), 따라서 스칼라 성능 요건이 정확하게 2배로 상승하는 것은 아니다. 이 모델의 일례가 아래에 도시된다.



[0239]

[0240] 5. 2개의 협동 스레드를 갖는 프로그램이 작성될 수 있다. 이것은 단일 스칼라 제어 스레드를 갖지만 다수의 기능적인 벡터 기능 블록이 함께 접속되는 것이 필요할 수 있는 코덱에 대해 예상되는 모델이다. 이것은 기능 블록들의 다이렉트-쇼우 핀 모델과 유사하다. 이러한 애플리케이션의 일례가 아래 도시된다. 이 모델은 단지 2개의 협동 스레드로 제한되는데, 이는 2개의 데이터 파이프만을 갖기 때문이다. 또 하나의 제한은 스레드들이 2개의 스레드 사이에 균형을 이루어야 한다는 것이다. 그렇지 않으면 성능 손실이 발생한다. 이러한 제한 내에서, 이 모델은 2개의 데이터 파이프 상에서 동작하며, 또한 단일 파이프로 스케일링될 수 있다.



[0241]

[0242] 6. 2개의 데이터 파이프가 서로 동기화될 수 있다. 동기화에 대한 기본 접근법은 데이터 구동 방식이다. 벡터 함수들은 데이터가 프로세스에 이용 가능할 때 실행된다. 스트림들은 메모리로부터의 판독들 또는 다른 데이터 파이프로부터의 기입들에 의해 채워진다. 데이터가 이용 가능하면, 벡터 제어 유닛은 실행 커맨드를 활성화시켜 실행한다. 스트림들은 또한 카운팅 세마포어로서 사용될 수 있다. 스칼라 제어기 및 벡터 데이터 파이프 양자는 타일 포인터들을 증감시킬 수 있으며, 데이터 전달이 존재하지 않는 경우에도 스트림 기술자를 카운팅 세마포어로서 사용할 수 있다.

[0243] 보충 개요:

[0244] 일반적으로, 본 발명의 실시예들은 다음을 수행한다.

[0245] 1. 매체 알고리즘의 스칼라 및 벡터 부분으로의 분해

[0246] 오프 더 셸프 스칼라 설계, 이는 또한 전력 및 성능 요건에 기초하여 상이한 클럭 속도로 스칼라 및 벡터 부분을 실행할 수 있는 능력을 제공한다.

[0247] 2. 스트림 처리

[0248] 3. 2D 데이터 경로 처리

[0249] 4. 지연 숨김(데이터 및 커맨드 페치 양자에 대해)

[0250] 애플리케이션 영역

[0251] 숨김:

[0252] 연산 코드 숨김

[0253] 암호화 프로그램은 단지 칩을 숨길 수 있다. 스칼라/제어기 블록은 단지 특정 동작이 수행되는 것을 요구하며, 암호화 엔진은 명령 등을 페치할 것이다. 스칼라는 심지어 알고리즘이 실행되고 있는지를 알 수 없으므로, 매우 안전하다. 이것은 사용자로부터 암호화 알고리즘을 숨기기 위한 메카니즘을 제공한다.

[0254] 2D

[0255] VP2 명령 세트 아키텍처는 2D 처리를 위한 명령들을 지원한다. 이들은 많은 GUI/윈도우 시스템에 사용되는 ROP3 및 ROP4 지원을 포함한다. 이것은 매체 프로세서가 매체 프로세서 상에서 2D 연산을 실행하는 것을 가능하게 한다. 여기서 고유한 이점은 전력 절감이다.

[0256] ISA

[0257] 명령 슬롯으로서의 조건 코드:

[0258] 조건 코드 연산을 위해 (다중 발행 명령 번들 내의) 개별 발행 슬롯을 구비한다. 종래 기술은 조건 코드/술어

레지스터에 영향을 미칠 수도 있는 사람 사용 SIMD 명령이다. 그러나, VP2에서 취해지는 접근법에서, 데이터 처리 및 술어 레지스터 처리는 독립적으로 스케쥴링되어, 보다 높은 성능을 얻을 수 있다.

[0259] 메모리 I/O

[0260] 마이크로코드화된 DMA 엔진:

[0261] DMA 엔진은 스트림에 대한 데이터 프리페치, 데이터 스트림 포맷팅, 에지 패딩 등과 같은 다양한 동작을 수행하도록 프로그래밍될 수 있다(또는 그 자신의 작은 마이크로코드를 가질 수 있다). 일반적으로, 프로그래머블 DMA 엔진이며, 유선 기능은 없다. 따라서, 메모리 I/O 프로세서와 매체 처리 코어의 조합은 전체 시스템 레벨 성능을 향상시킨다. 매체 프로세서 코어는 데이터 I/O 처리를 행해야 하는 부담이 없다.

[0262] 메모리 계층 구조:

[0263] VP2 아키텍처에서, 메모리 계층 구조는 메모리 BW를 최소화하는 것은 물론 지연 보상을 제공하도록 최적화된다. 다음과 같은 많은 상이한 스킴이 제공된다.

[0264] - 스크래치 RAM으로서 벡터 코어에게 보일 수 있는 스트리밍 데이터 저장소의 제1 레벨. 스칼라 프로세서에 의해 생성된 요구 스트림을 사전에 조사하도록 HW에 의해 관리된다. 이 데이터 저장소는 데이터 재사용을 위한 L2 캐시에 의해 옵션으로 보강된다. L2 캐시는 스트림 기반으로 개별 섹터들로 분할될 수 있다.

[0265] - 스트리밍 데이터 저장소에 의해 보강된 L1 캐시. 데이터 저장소는 다음 관련 데이터 세트를 프리페치하였다.

[0266] - 스트림 포인터를 데이터 태그로서 사용하는 캐시

[0267] - 스칼라 생성 스트림 어드레스를 이용한 L1 데이터 저장소 및 L2 캐시의 프리페치/캐싱

[0268] 최적화된 스칼라-벡터 통신 링크:

[0269] MemRd/Wr 포맷:

[0270] 시스템 메모리를 로컬 메모리로 판독 및 기입하기 위한 스칼라로부터의 콤팩트 커맨드. DMA 엔진을 관리하는데 필요한 제어 흐름 대역폭 상에 저장한다. 동시에 지원 트랜잭션들의 타입을 제한하지 않는다.

[0271] 벡터 L2에 대한 스칼라-벡터에 관한 고찰:

[0272] 통신 대역폭을 줄이기 위해 파라미터 변경자 및 반복기를 지원하는 파라미터 압축.

[0273] 파이프라인 캐시:

[0274] 파이프라인화된 명령 캐시. 다음과 같이 다양한 스킴이 지원된다.

[0275] - 벡터와 스칼라 프로세서 사이의 인 플라이트 실행을 추적함으로써 각 캐시 라인의 수명을 관리한다. 이것은 벡터 프로세서가 실행을 시작하기 전에 명령들이 준비되는 것을 가능하게 한다. 명령들이 아직 캐시 내에 있지 않은 경우, 명령들은 프리페치된다.

[0276] - 적은 지연 구성에 대해, 명령 캐시는 작은 FIFO로 변경됨으로써 최소화된다. FIFO 내의 기존 실행들이 재이용될 수 있으며, 그렇지 않은 경우 다시 페치된다.

[0277] 전체 아키텍처:

[0278] 데이터 저장소는 다양한 처리 요소 사이에 공유될 수 있다. 이들은 스트림을 통해 통신하며 서로 공급할 수 있다. 이 아키텍처는 SIMD 벡터 코어들, DMA 엔진들, 스트림을 통해 접속된 고정 함수 유닛들과 같은 한 세트의 이종 기능 유닛들을 구상한다.

[0279] 연산/DP

[0280] 임의/유연 형상/절반 파이프:

[0281] 데이터 경로는 가변 형상들 상에서 동작한다. 데이터 경로의 형상은 문제 세트에 매칭되도록 구성될 수 있다. 일반적으로, 사람들은 1D 데이터 경로를 행한다. VP2는 알고리즘에 매칭되도록 4x4, 8x4 또는 16x1 등의 가변 크기를 가질 수 있는 형상을 처리할 수 있다.

[0282] 스케일링 가능성:

[0283] VP2 데이터 경로 아키텍처는 면적을 줄이기 위하여 다수의 사이클을 통해 보다 좁은 데이터 경로 상에서 보다 넓은 SIMD 명령들을 실행하기 위하여 명령 안내 기술을 이용한다(주: 16 웨이 SIMD 파이프를 갖는데, 각각의 피연산자는 1바이트 폭을 갖는다. 8 웨이 SIMD 파이프(2개의 파이프를 함께 그룹화함) 및 각각의 피연산자가 2 바이트인 보다 넓은 SIMD 데이터 경로를 가질 수 있으며, 마찬가지로 4 웨이 SIMD 파이프(4개의 파이프를 함께 그룹화함) 및 각각의 피연산자가 4 바이트인 보다 넓은 SIMD 데이터 경로를 가질 수 있다).

[0284] 예를 들어, VP2는 데이터 경로를 16 웨이 SIMD에서 8 웨이 SIMD로 스케일링할 수 있다.

[0285] 바이트 레인들의 연결:

[0286] 피연산자 폭을 증가시키기 위한 SIMD 웨이들의 연결. 예를 들어, 현재 16 웨이 SIMD는 8 비트 피연산자를 갖는다. 이것을 8 웨이 SIMD의 16 비트 피연산자 및 4 웨이 SIMD의 32 비트 피연산자로 증가시킬 수 있다.

[0287] SIMD 어드레스 생성기

[0288] SIMD 파이프의 각 웨이에 대한 개별 스트림 어드레스 생성기.

[0289] VP2는 요청들이 데이터 저장소에 대한 최소 액세스로 합체된 SIMD 어드레스 생성기를 이용할 수 있다.

[0290] 크로스바 및 콜렉터를 이용한 데이터 확장

[0291] 크로스바를 이용하여 보다 많은 데이터 피연산자를 생성할 수 있는 능력. 판독 포트 압력을 줄이고 전력을 줄인다.

[0292] X2 명령들:

[0293] 모든 명령들이 데이터 경로 내의 모든 HW 요소(가산기/승산기)를 이용할 수 있는 것은 아니다. 따라서, 복잡한 명령보다 넓은 데이터 형상을 처리할 수 있는 덧셈/뺄셈과 같은 간단한 명령에 대해 가능하다. 따라서, 성능을 최소 공통 크기로 제한하는 대신에, VP2는 판독 포트들이 동작 대역폭을 유지할 수 있는 한 보다 넓은 형상 상에서 동작하도록 적절히 시도하는 유연한 명령 세트를 이용한다.

[0294] 멀티스레드/멀티코어 매체 처리

[0295] VP2 아키텍처은 다음과 같은 다양한 멀티스레딩 옵션을 지원한다.

[0296] - 멀티스레드 스칼라 프로세서는 스트림을 통해 접속된 다수의 벡터 유닛 상에서 프로시저 호출을 스케쥴링한다.

[0297] - 다수의 스레드는 명령 단위 또는 실행 단위의 스레드 스위칭에 의해 단일 벡터 엔진 상에서 실행된다.

[0298] 상이한 벡터/스칼라를 이용한 전력 관리

[0299] 스칼라 및 벡터 부분이 분리되는 경우, 전력 및 성능 요건에 기초하여 이들 2 블록을 상이한 속도로 실행할 수 있다.

[0300] 콘텍스트 스위치:

[0301] 이 매체 프로세서는 레지스터가 없는 구조를 가지므로 매우 빠른 콘텍스트 스위치를 지원하는 능력을 갖고 있다. HW 지원이 스칼라-벡터 커맨드 큐를 추적하고, 이것을 저장하고 재생하여 콘텍스트 스위칭을 달성하기 위해 존재한다. 또한, 콘텍스트 스위치는 페이지 고장시에 개시될 수 있다.

[0302] 이것은 매체 프로세서가 입출력 디스플레이 처리와 같은 실시간 처리 태스크를 유지하면서 디스플레이 파이프라인을 공급하기 위해 2D 가속화 또는 최적시의 비디오 향상과 같은 비 실시간 태스크를 지원할 수 있게 한다.

[0303] 이러한 콘텍스트 스위치 능력은 그의 명령 세트와 함께 VP2가 통합 픽셀/코덱 처리하는 것을 가능하게 한다.

[0304] 데이터 구성:

[0305] VP2는 다음과 같은 특성을 가진 데이터 저장소 구성을 이용한다.

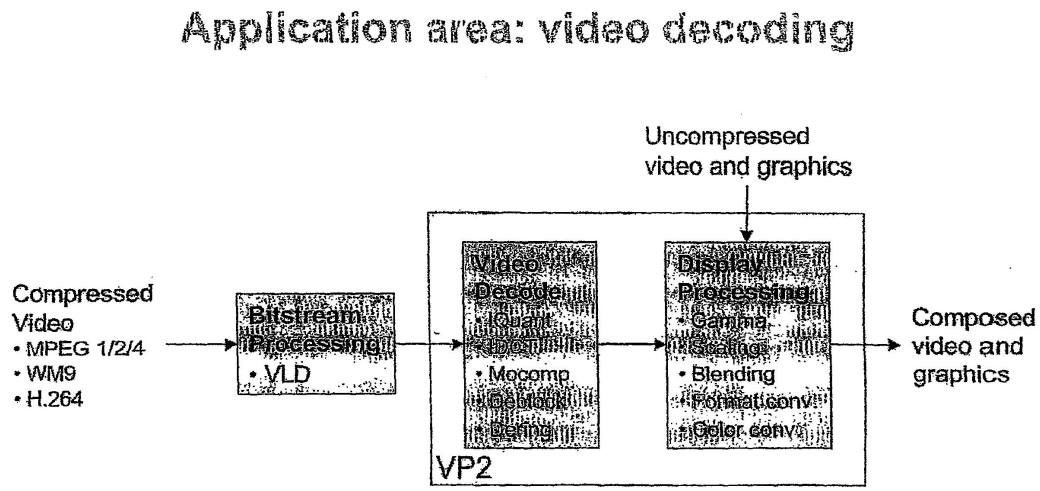
[0306] 각 방향으로 최대 16 픽셀이 뱅크 충돌없이 액세스될 수 있다. 이것은 스트라이드 요건을 최소로 유지하면서 행해진다.

[0307] 데이터 저장소 구성은 데이터 형상들의 효율적인 전이를 가능하게 한다.

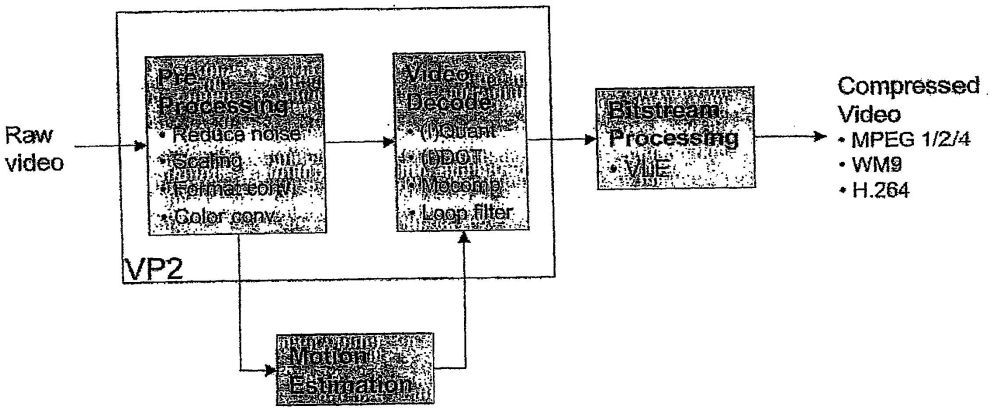
[0308] 2D 어드레싱이 데이터 저장소 내에서 지원되어, 비디오와 같은 대부분의 매체 처리 애플리케이션에서 선형 어드레스의 SW 연산을 제거한다.

[0309]

[0310] Applications



[0311]



[0312]

[0313] Architecture

High compute density

- 2x perf/mm2 of VP1 architecture
- Efficient on new apps (compare to fixed function HW)

Programmable

- "reasonably" compilable : C for scalar, intrinsics for vector
- Latency hiding managed in HW
- Protection against hangs and bad addresses

Scalable

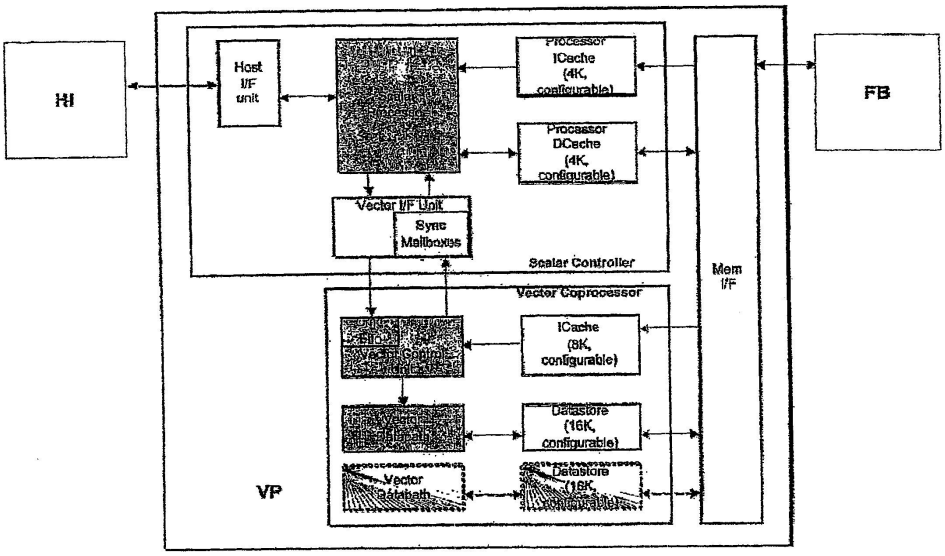
- 2x and _x scaling options
- Scalable via clock frequency

High precision operations support

- 10b + 20b integer datatypes

[0314]

Top level block diagram



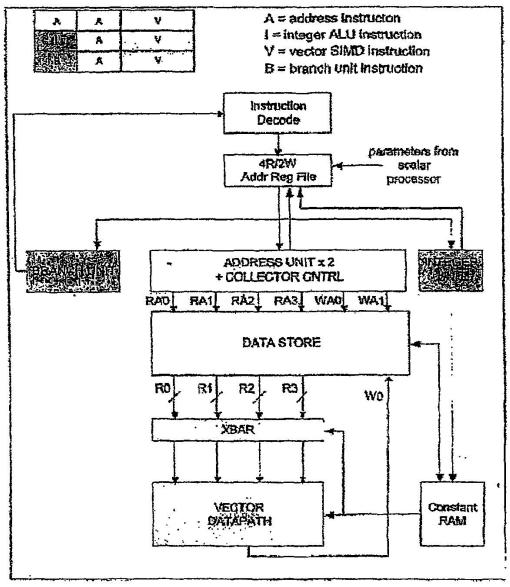
[0315]

Block overview

- Scalar processor:
  - Program flow control
  - Tile rasterization and address calc
  - Feeds loosely coupled vector unit
- Vector processor:
  - 3-issue VLIW (V + I,A,B)
  - 4x4 SIMD datapath
  - Vector operands directly from datastore (no regfile transfer)
- Scalar / vector interface:
  - Loosely coupled processors – scalar feeds a command fifo
  - Instruction and data streaming (latency hiding) managed by HW
  - Vector commands issue when instructions and data are ready

[0316]

Vector datapath



[0317]

inherited from VP1

- Dot product oriented datapath
  - VP2 goes from 2-tap to 4-tap math
- VLIW with A/I/B/V instructions. (Modified issue rules)
- Datapath arch: accumulator, data forwarding, HW interlock
- Free transposes. Changed from 16x16 -> 4x4
- Dedicated address units
- Complex instruction set (optimized for video)

[0318]

VP2 Features 1

- Decoupled scalar processor
  - Good target for compilable outer loop code in C
- Instruction streaming
  - HW-managed instruction prefetches (using pipelined I$)
  - Applications can have larger footprint than instruction cache
- Data prefetching
  - HW-managed streams in datastore
  - Programmer does not have to do outer loop unrolling
- Precision improvements
  - 10b single precision
  - New 20b double precision
- Better ISA
- Better error management
  - memory protection, instruction traps

[0319]

VP2 Features 2

- 4X4 SIMD datapath
  - VP1 was 16x1 SIMD machine
  - Very efficient organization for advanced codecs (H.264, WMV9)
  - Reduces bandwidth requirements for 2D processing
- Improved datastore organization
  - Tiled 4x4 structure to match datapath and provide a much cheaper transpose capability vs VP1
  - collector structure like SPA to emulate multi-port RAM
  - vector pipe operates directly from datastore (no vector regfile)
- Dedicated crossbar stage
  - Extract un-aligned operands
  - Create multiple operands from few read ports
- Constant RAM
  - reduces datastore read bandwidth pressure
- Very flexible condition code / predicate support

[0320]

Target applications

- Codecs
  - mpeg2/wmv9/H.264 encode assist (in-loop decoder)
  - mpeg2/wmv9/H.264 decode (post VLD decoding)
  - In-loop/out-of-loop de-blocking filters
- Image processing / enhancement
  - Advanced motion adaptive deinterlacing
  - Input noise filtering for encoding
  - Polyphase scaling/resampling
  - Sub-picture compositing
  - Per-pixel transforms: procamp, color space conversion, gamma, LCD overdrive, histogram adjustment etc.
  - Video surface format conversions

[0321]

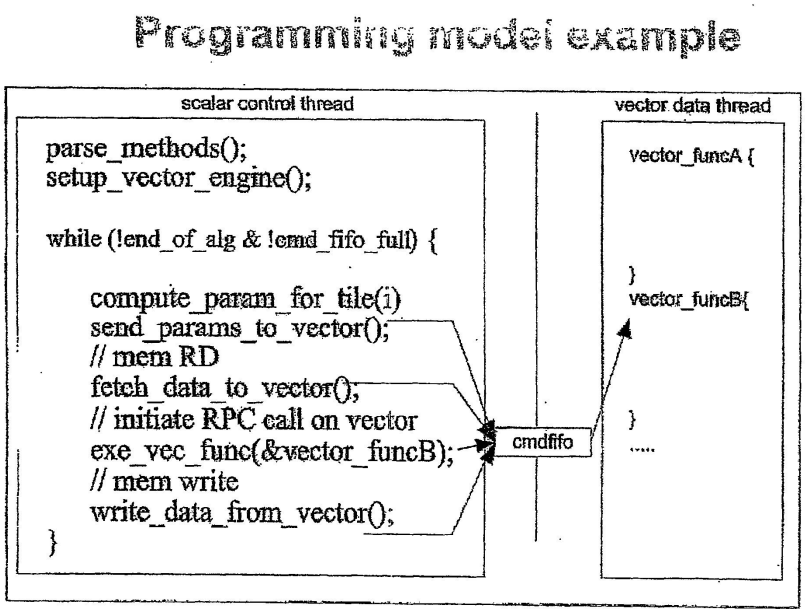
Potential target applications

- 2D primitives, blits, rotates etc.
- Refinement-based software motion est algorithms
- 16/32 bit MAC applications (audio?)

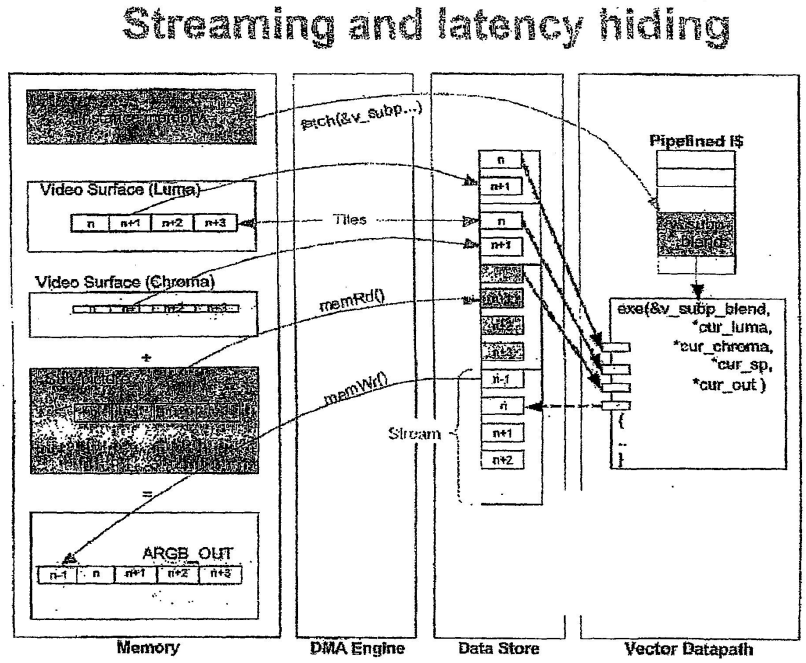
[0322]

Programming Model

[0323]



[0324]



[0325]

부호의 설명

[0326] 101: CPU

105: 브리지

112: 디스플레이

115: 메모리

210: 스칼라 프로세서

211: 프로세서 명령 캐시

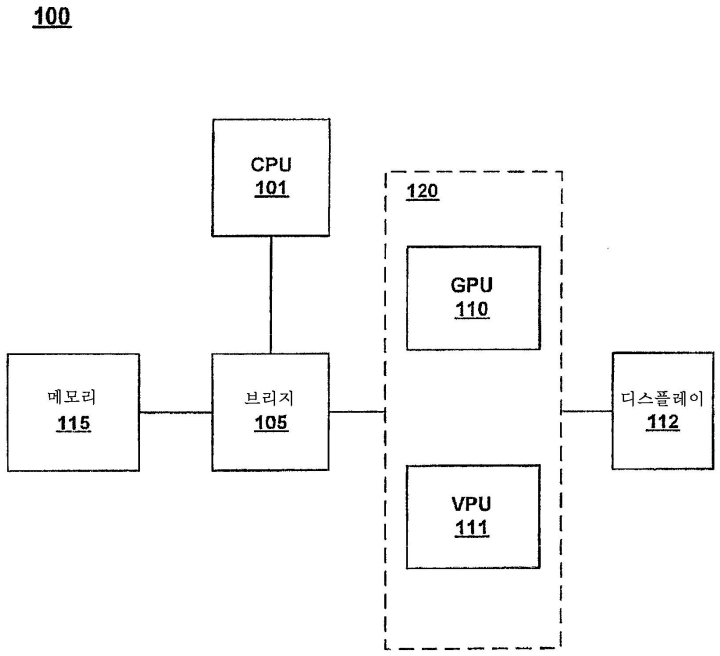
212: 프로세서 데이터 캐시

511: 데이터 저장 메모리

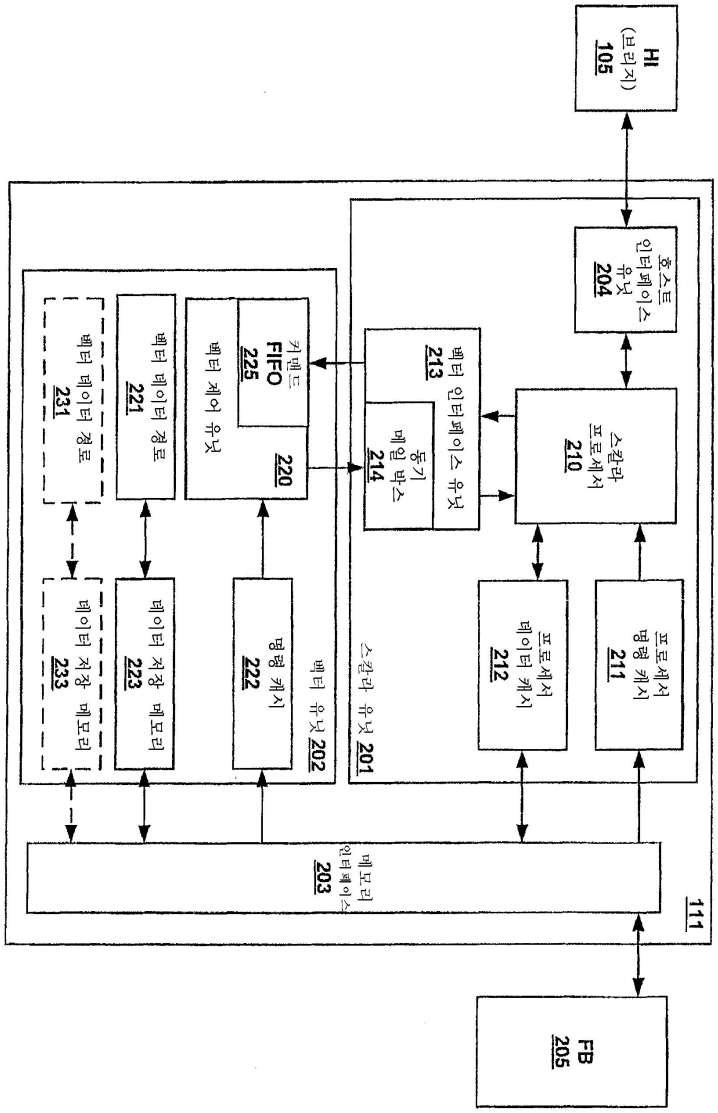
515: 어드레스 레지스터 파일

도면

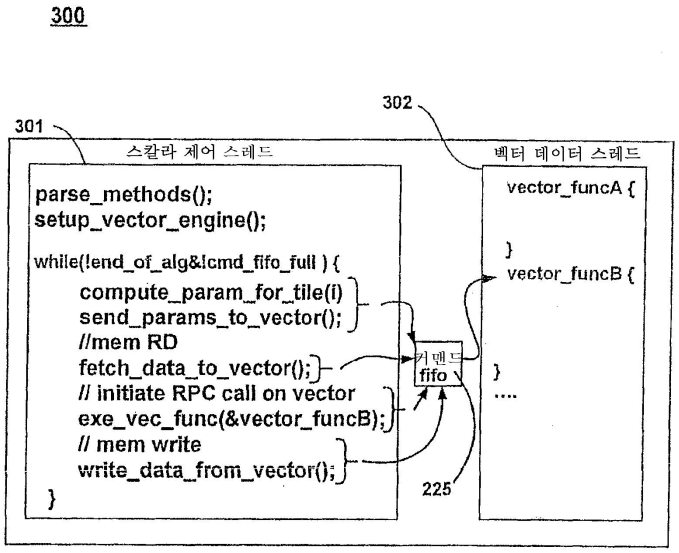
도면1



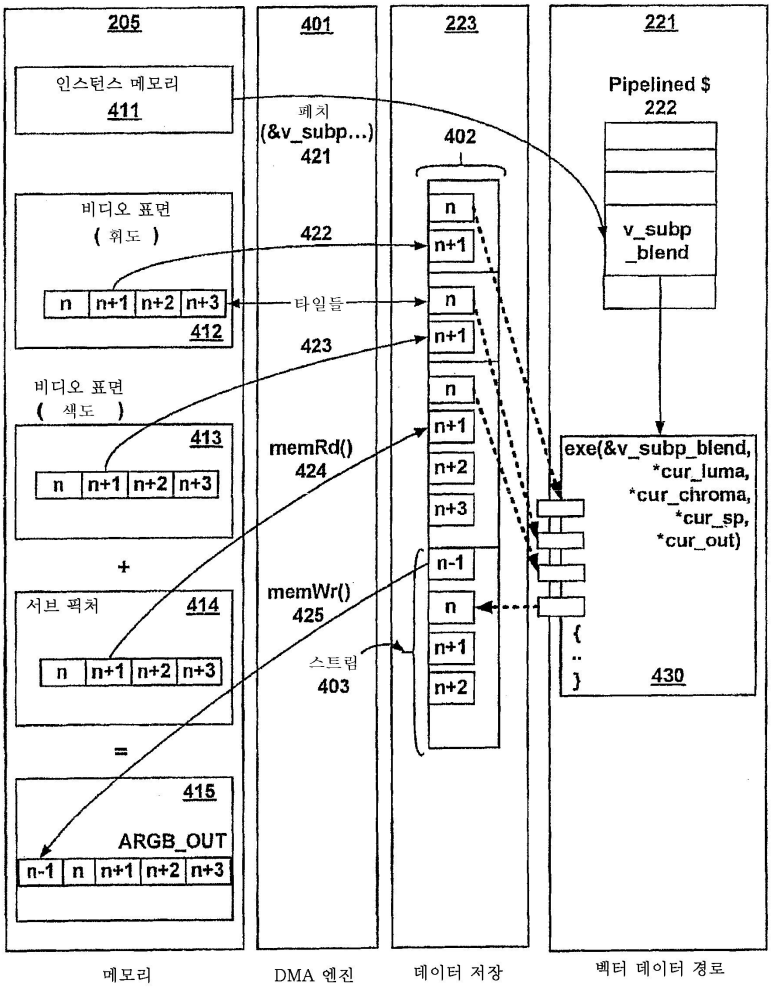
도면2


HI
(브리지)
105
호스트
인터페이스
유닛
204
벡터 인터페이스 유닛
213
동기
메일 박스
214
스칼라
프로세서
210
커맨드
FIFO
225
220
벡터 제어 유닛
벡터 데이터 경로
221
벡터 데이터 경로
231
프로세서
명령 캐시
211
프로세서
데이터 캐시
212
명령 캐시
222
데이터 저장 메모리
223
데이터 저장 메모리
233
벡터 유닛 202
스칼라 유닛 201
메모리
인터페이스
203
111
FB
205

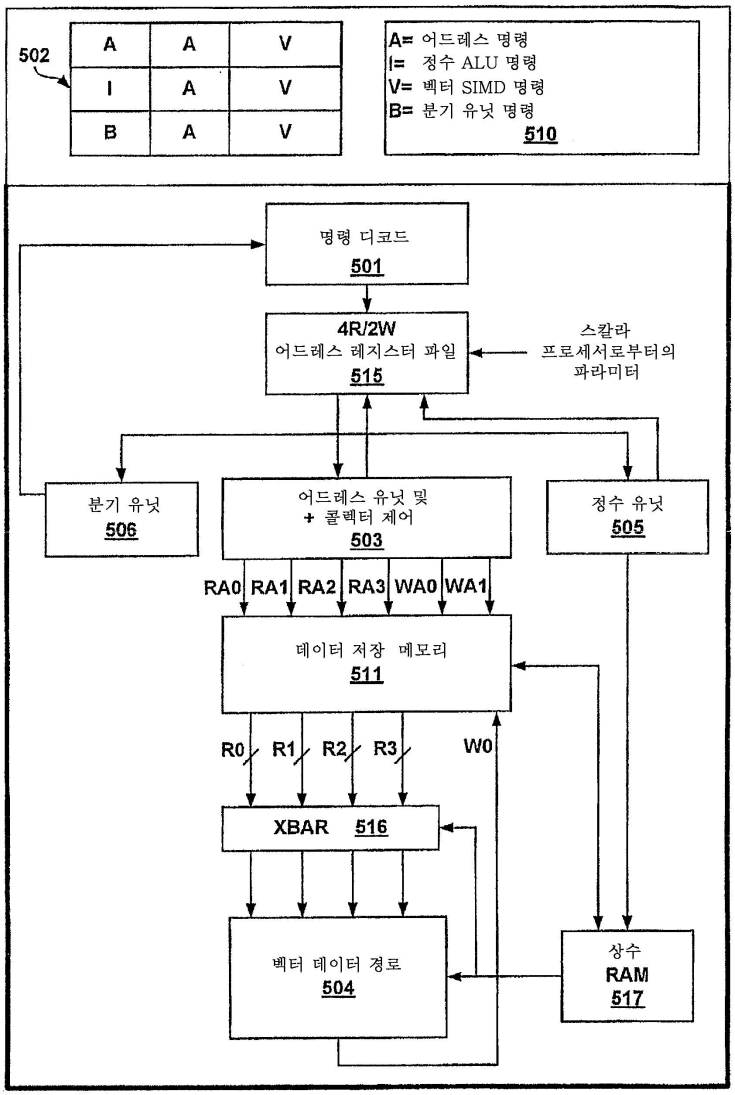
도면3


300
301
스칼라 제어 스레드
302
벡터 데이터 스레드
parse_methods();
setup_vector_engine();
while(!end_of_alg&!cmd_fifo_full) {
compute_param_for_tile(i)
send_params_to_vector();
//mem RD
fetch_data_to_vector();
// initiate RPC call on vector
exe_vec_func(&vector_funcB);
// mem write
write_data_from_vector();
}
vector_funcA {
}
vector_funcB {
}
....
커맨드
fifo
225

도면4


205
401
223
221
인스턴스 메모리
411
페치
(&v_subp...)
421
402
Pipelined $
222
비디오 표면
(휘도)
422
n
n+1
n+2
n+3
412
타일들
v_subp
_blend
423
비디오 표면
(색도)
413
memRd()
424
exe(&v_subp_blend,
*cur_luma,
*cur_chroma,
*cur_sp,
*cur_out)
+
서브 픽처
414
memWr()
425
n-1
{
..
}
430
스트림
403
=
415
ARGB_OUT
메모리
DMA 엔진
데이터 저장
벡터 데이터 경로

도면5


502
A A V
I A V
B A V
A= 어드레스 명령
I= 정수 ALU 명령
V= 벡터 SIMD 명령
B= 분기 유닛 명령
510
명령 디코드
501
4R/2W
어드레스 레지스터 파일
515
스칼라
프로세서로부터의
파라미터
분기 유닛
506
어드레스 유닛 및
+ 콜렉터 제어
503
정수 유닛
505
RA0 RA1 RA2 RA3 WA0 WA1
데이터 저장 메모리
511
R0 R1 R2 R3
W0
XBAR 516
벡터 데이터 경로
504
상수
RAM
517

도면6