(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2023년07월07일
(11) 등록번호 10-2553932
(24) 등록일자 2023년07월05일

(51) 국제특허분류(Int. Cl.)
G06T 1/20 (2018.01) *G06F 13/16* (2006.01)
G06F 9/38 (2006.01) *G09G 5/36* (2006.01)
G09G 5/397 (2006.01)
(52) CPC특허분류
G06T 1/20 (2013.01)
G06F 13/16 (2013.01)
(21) 출원번호 10-2021-7040695(분할)
(22) 출원일자(국제) 2014년08월06일
심사청구일자 2022년01월10일
(85) 번역문제출일자 2021년12월10일
(65) 공개번호 10-2021-0156845
(43) 공개일자 2021년12월27일
(62) 원출원 특허 10-2021-7006168
원출원일자(국제) 2014년08월06일
심사청구일자 2021년03월23일
(86) 국제출원번호 PCT/IB2014/002541
(87) 국제공개번호 WO 2015/019197
국제공개일자 2015년02월12일
(30) 우선권주장
1314263.3 2013년08월08일 영국(GB)
(뒷면에 계속)
(56) 선행기술조사문헌
John Alexander."Evolution and use of the VME subsystem bus -VSB HISTORY OF VME SUBSYSTEM BUSES", vol 10 n 6,1986.08.31.,(제307면 내지 제312면)*
(뒷면에 계속)
전체 청구항 수 : 총 24 항

(73) 특허권자
몰로니, 다비드
아일랜드, 더블린 9, 글라스네빈, 아이오나 로드 42
리치몬드, 리차드
영국, 벨파스트 비티5 5엔유, 안트림, 썬버리 애비뉴 14
(뒷면에 계속)
(72) 발명자
몰로니, 다비드
아일랜드, 더블린 9, 글라스네빈, 아이오나 로드 42
리치몬드, 리차드
영국, 벨파스트 비티5 5엔유, 안트림, 썬버리 애비뉴 14
(뒷면에 계속)
(74) 대리인
김태홍, 김진희

심사관 : 이후락

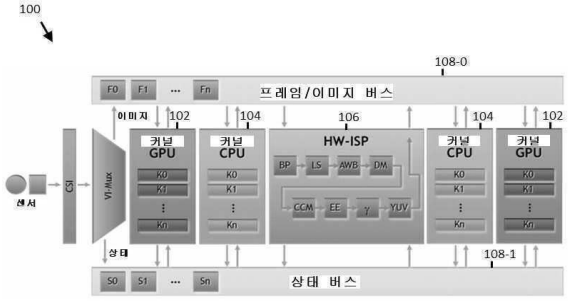
(54) 발명의 명칭 **컴퓨터 이미징 파이프라인**

(57) 요 약

본 출원은 일반적으로는 병렬 프로세싱 디바이스에 관한 것이다. 병렬 프로세싱 디바이스는 복수의 프로세싱 엘리먼트, 메모리 서브시스템 및 상호접속 시스템을 포함할 수 있다. 메모리 서브시스템은 복수의 메모리 슬라이스를 포함할 수 있고, 그 중 적어도 하나는 복수의 프로세싱 엘리먼트 중 하나와 연관되고 복수의 랜덤 액세스 메

(뒷면에 계속)

대 표 도 - 도1



모리(RAM) 타일을 포함하며, 각각의 타일은 개개의 판독 및 기록 포트를 갖는다. 상호접속 시스템은 복수의 프로세싱 엘리먼트와 메모리 서브시스템을 결합하도록 구성된다. 상호접속 시스템은 로컬 상호접속부 및 글로벌 상호접속부를 포함한다.

(52) CPC특허분류
G06F 9/3867 (2013.01)
G06F 9/3885 (2013.01)
G09G 5/363 (2013.01)
G09G 5/397 (2013.01)
G09G 2360/08 (2013.01)
G09G 2360/122 (2013.01)

(73) 특허권자
도노호, 다비드
캐나다, 온타리오, 케이2에스 오케이, 스티츠빌, 템페스트 드라이브 206
바리, 브렌든
아일랜드, 더블린 14, 처치타운, 렌드스케이프 가든스 8
브릭, 코맥
아일랜드, 더블린 7, 카브라, 칸로 로드 57
베사, 오비디우, 안드레이
루마니아, 알-300362 티미소아라, 엔트리 비. 아파트먼트 9, 블록 비27, 마티리이 데 라 판타나 알바 스트리트

(72) 발명자
도노호, 다비드
캐나다, 온타리오, 케이2에스 오케이, 스티츠빌, 템페스트 드라이브 206
바리, 브렌든
아일랜드, 더블린 14, 처치타운, 렌드스케이프 가든스 8
브릭, 코맥
아일랜드, 더블린 7, 카브라, 칸로 로드 57
베사, 오비디우, 안드레이
루마니아, 알-300362 티미소아라, 엔트리 비. 아파트먼트 9, 블록 비27, 마티리이 데 라 판타나 알바 스트리트

(56) 선행기술조사문헌
GB2348971 A
US20030160794 A1
US20080007562 A1
KR1020120104364 A
KR1020020088063 A
JP2004178056 A
JP2007272357 A
KR1020020018071 A
KR1020110058410 A
KR1020130024860 A
KR1020130031258 A
*는 심사관에 의하여 인용된 문헌

(30) 우선권주장
A/00812 2013년11월06일 루마니아(RO)
14/082,396 2013년11월18일 미국(US)
14/082,645 2013년11월18일 미국(US)

명 세 서

청구범위

청구항 1

삭제

청구항 2

병렬 프로세싱 디바이스에 있어서,

프로세서 회로부(circuitry);

랜덤 액세스 메모리(RAM) 타일을 갖는 메모리 슬라이스 - 상기 프로세서 회로부는 클라이언트 요청 벡터를 통해 상기 RAM 타일에 액세스하기 위한 요청들을 발행함 - ; 및

포트 선택 회로부를 포함하고,

상기 포트 선택 회로부는,

상기 클라이언트 요청 벡터의 제1 리딩 원(leading one)의 제1 위치에 기초하여 상기 RAM 타일에 액세스하기 위한 제1 요청을 검출하고,

상기 클라이언트 요청 벡터의 마스킹된 버전의 제2 리딩 원의 제2 위치에 기초하여 상기 RAM 타일에 액세스하기 위한 제2 요청을 검출하고,

상기 제1 요청의 제1 우선순위와 상기 제2 요청의 제2 우선순위에 기초하여 상기 제1 요청 또는 상기 제2 요청 중 어느 것이 상기 RAM 타일에 배정될지를 결정하는 것인, 병렬 프로세싱 디바이스.

청구항 3

제2항에 있어서,

상기 RAM 타일은 제1 RAM 타일이고, 상기 메모리 슬라이스는 제2 RAM 타일을 포함하고, 상기 클라이언트 요청 벡터의 상기 마스킹된 버전은 제1 마스킹된 버전이고, 상기 포트 선택 회로부는 제1 포트 선택 회로부이고, 상기 병렬 프로세싱 디바이스는 또한 제2 포트 선택 회로부를 포함하고,

상기 제2 포트 선택 회로부는,

상기 클라이언트 요청 벡터의 제1 트레일링 원(trailing one)의 제3 위치에 기초하여 상기 제2 RAM 타일에 액세스하기 위한 제3 요청을 검출하고,

상기 클라이언트 요청 벡터의 제2 마스킹된 버전의 제2 트레일링 원의 제4 위치에 기초하여 상기 제2 RAM 타일에 액세스하기 위한 제4 요청을 검출하고,

상기 제3 요청의 제3 우선순위와 상기 제4 요청의 제4 우선순위에 기초하여 상기 제3 요청 또는 상기 제4 요청 중 어느 것이 상기 제2 RAM 타일에 배정될지를 결정하는 것인, 병렬 프로세싱 디바이스.

청구항 4

제2항에 있어서,

상기 RAM 타일은 제1 RAM 타일이고, 상기 메모리 슬라이스는 제2 RAM 타일을 갖고, 상기 병렬 프로세싱 디바이스는 또한 승자(winner) 검출 회로부를 포함하고,

상기 승자 검출 회로부는 상기 제1 요청이 상기 제1 RAM 타일과 상기 제2 RAM 타일에 배정된 것에 응답하여,

상기 제1 RAM 타일과 상기 제2 RAM 타일 중 상기 제1 요청을 배정할 첫 번째 타일을 선택하고,

상기 제1 RAM 타일과 상기 제2 RAM 타일 중 두 번째 타일에 제3 요청 - 상기 제3 요청은 상기 제1 요청의 상기 제1 우선순위와 비교하여 다음으로 가장 높은 우선순위를 가짐 - 을 배정하는 것인, 병렬 프로세싱 디

바이스.

청구항 5

제2항에 있어서,

상기 클라이언트 요청 벡터는 상기 RAM 타일에 액세스하기 위한 각각의 요청을 발행한 것이 제1 프로세서 회로부인지 제2 프로세서 회로부인지 여부를 나타내는 적어도 두 개의 비트들을 포함하는 것인, 병렬 프로세싱 디바이스.

청구항 6

제2항에 있어서,

상기 포트 선택 회로부는 상기 제1 요청, 상기 제2 요청, 또는 제3 요청 중 어느 것이 상기 RAM 타일에 배정될지 (a) 상기 제1 요청의 상기 제1 우선순위, (b) 상기 제2 요청의 상기 제2 우선순위, 또는 (c) 상기 제3 요청의 제3 우선순위 중 적어도 하나에 기초하여 결정하고, 상기 제3 요청은 최우선순위(super-priority) 벡터에 포함되고, 상기 제2 우선순위는 우선순위 벡터에 기초하여 결정되는 것인, 병렬 프로세싱 디바이스.

청구항 7

제6항에 있어서,

상기 최우선순위 벡터에 포함된 상기 제3 요청의 상기 제3 우선순위는 상기 제1 우선순위, 상기 제2 우선순위, 및 상기 제3 우선순위 중 가장 높은 레벨의 우선순위이고,

상기 클라이언트 요청 벡터의 상기 마스킹된 버전에 포함된 상기 제2 요청의 상기 제2 우선순위는 상기 우선순위 벡터에 기초하여 우선순위 레벨이 배정되고, 상기 우선순위 벡터는 상기 클라이언트 요청 벡터의 상기 마스킹된 버전을 우선순위 및 비-우선순위(non-priority) 요청들로 분할하고,

상기 클라이언트 요청 벡터에 포함된 상기 제1 요청의 상기 제1 우선순위는 상기 제1 우선순위, 상기 제2 우선순위, 및 상기 제3 우선순위 중 가장 낮은 레벨의 우선순위를 갖는 것인, 병렬 프로세싱 디바이스.

청구항 8

제6항에 있어서,

상기 최우선순위 벡터는 런타임 스케줄러에 의해 제공되고, 상기 최우선순위 벡터는 상기 RAM 타일에 액세스하기 위한 상기 요청들 중 애플리케이션에 관련된 트래픽 클래스들과 연관된 요청들을 우선화하고(prioritize), 상기 트래픽 클래스들은 비디오 트래픽 클래스들을 포함하는 것인, 병렬 프로세싱 디바이스.

청구항 9

제2항, 제3항, 제4항, 제5항, 제6항, 제7항, 또는 제8항 중 어느 한 항에 있어서,

상기 포트 선택 회로부는 상기 클라이언트 요청 벡터와 비트마스크에 AND 연산을 수행함으로써 상기 클라이언트 요청 벡터의 상기 마스킹된 버전을 생성하는 것인, 병렬 프로세싱 디바이스.

청구항 10

명령어들을 포함하는 컴퓨터 판독가능 저장 매체에 있어서,

상기 명령어들은 실행시 하나 이상의 프로세서로 하여금 적어도,

클라이언트 요청 벡터의 제1 리딩 원(leading one)의 제1 위치에 기초하여, 메모리 슬라이스의 랜덤 액세스 메모리(RAM) 타일에 액세스하기 위한 제1 프로세서로부터의 제1 요청을 검출하고,

상기 클라이언트 요청 벡터의 마스킹된 버전의 제2 리딩 원의 제2 위치에 기초하여, 상기 RAM 타일에 액세스하기 위한 제2 프로세서로부터의 제2 요청을 검출하고,

상기 제1 요청의 제1 우선순위와 상기 제2 요청의 제2 우선순위에 기초하여 상기 제1 요청 또는 상기 제2 요청 중 어느 것이 상기 RAM 타일에 배정될지를 결정하게 하는 것인, 컴퓨터 판독가능 저장 매체.

청구항 11

제10항에 있어서,

상기 RAM 타일은 제1 RAM 타일이고, 상기 메모리 슬라이스는 제2 RAM 타일을 포함하고, 상기 클라이언트 요청 벡터의 상기 마스킹된 버전은 제1 마스킹된 버전이고, 상기 명령어들은 실행시 상기 하나 이상의 프로세서로 하여금,

상기 클라이언트 요청 벡터의 제1 트레일링 원(trailing one)의 제3 위치에 기초하여, 상기 제2 RAM 타일에 액세스하기 위한 제3 프로세서로부터의 제3 요청을 검출하고,

상기 클라이언트 요청 벡터의 제2 마스킹된 버전의 제2 트레일링 원의 제4 위치에 기초하여, 상기 제2 RAM 타일에 액세스하기 위한 제4 프로세서로부터의 제4 요청을 검출하고,

상기 제3 요청의 제3 우선순위와 상기 제4 요청의 제4 우선순위에 기초하여 상기 제3 요청 또는 상기 제4 요청 중 어느 것이 상기 제2 RAM 타일에 배정될지를 결정하게 하는 것인, 컴퓨터 판독가능 저장 매체.

청구항 12

제10항에 있어서,

상기 RAM 타일은 제1 RAM 타일이고, 상기 메모리 슬라이스는 제2 RAM 타일을 갖고, 상기 명령어들은 실행시 상기 하나 이상의 프로세서로 하여금 상기 제1 요청이 상기 제1 RAM 타일과 상기 제2 RAM 타일에 배정된 것에 응답하여,

상기 제1 RAM 타일과 상기 제2 RAM 타일 중 상기 제1 요청을 배정할 첫 번째 타일을 선택하고,

상기 제1 RAM 타일과 상기 제2 RAM 타일 중 두 번째 타일에 제3 요청 - 상기 제3 요청은 상기 제1 요청의 상기 제1 우선순위와 비교하여 다음으로 가장 높은 우선순위를 가짐 - 을 배정하게 하는 것인, 컴퓨터 판독가능 저장 매체.

청구항 13

제10항에 있어서,

상기 클라이언트 요청 벡터는 상기 RAM 타일에 액세스하기 위한 각각의 요청을 발행한 것이 제1 프로세서인지 제2 프로세서인지 여부를 나타내는 적어도 두 개의 비트들을 포함하는 것인, 컴퓨터 판독가능 저장 매체.

청구항 14

제10항에 있어서,

상기 명령어들은 실행시 상기 하나 이상의 프로세서로 하여금, 상기 제1 요청, 상기 제2 요청, 또는 제3 요청 중 어느 것이 상기 RAM 타일에 배정될지 (a) 상기 제1 요청의 상기 제1 우선순위, (b) 상기 제2 요청의 상기 제2 우선순위, 또는 (c) 상기 제3 요청의 제3 우선순위 중 적어도 하나에 기초하여 결정하게 하고, 상기 제3 요청은 최우선순위(super-priority) 벡터에 포함되고, 상기 제2 우선순위는 우선순위 벡터에 기초하여 결정되는 것인, 컴퓨터 판독가능 저장 매체.

청구항 15

제14항에 있어서,

상기 최우선순위 벡터에 포함된 상기 제3 요청의 상기 제3 우선순위는 상기 제1 우선순위, 상기 제2 우선순위, 및 상기 제3 우선순위 중 가장 높은 레벨의 우선순위이고,

상기 클라이언트 요청 벡터의 상기 마스킹된 버전에 포함된 상기 제2 요청의 상기 제2 우선순위는 상기 우선순위 벡터에 기초하여 우선순위 레벨이 배정되고, 상기 우선순위 벡터는 상기 클라이언트 요청 벡터의 상기 마스킹된 버전을 우선순위 및 비-우선순위(non-priority) 요청들로 분할하고,

상기 클라이언트 요청 벡터에 포함된 상기 제1 요청의 상기 제1 우선순위는 상기 제1 우선순위, 상기 제2 우선순위, 및 상기 제3 우선순위 중 가장 낮은 레벨의 우선순위를 갖는 것인, 컴퓨터 판독가능 저장 매체.

청구항 16

제14항에 있어서,

상기 최우선순위 벡터는 런타임 스케줄러에 의해 제공되고, 상기 최우선순위 벡터는 상기 RAM 타일에 액세스하기 위한 상기 요청들 중 애플리케이션에 관련된 트래픽 클래스들과 연관된 요청들을 우선화하고(prioritize), 상기 트래픽 클래스들은 비디오 트래픽 클래스들을 포함하는 것인, 컴퓨터 판독가능 저장 매체.

청구항 17

제10항, 제11항, 제12항, 제13항, 제14항, 제15항, 또는 제16항 중 어느 한 항에 있어서,

상기 명령어들은 실행시 상기 하나 이상의 프로세서로 하여금, 상기 클라이언트 요청 벡터와 비트마스크에 AND 연산을 수행함으로써 상기 클라이언트 요청 벡터의 상기 마스킹된 버전을 생성하게 하는 것인, 컴퓨터 판독가능 저장 매체.

청구항 18

컴퓨팅 디바이스에 의해 수행되는 방법에 있어서,

클라이언트 요청 벡터의 제1 리딩 원(leading one)의 제1 위치에 기초하여, 메모리 슬라이스의 랜덤 액세스 메모리(RAM) 타일에 액세스하기 위한 제1 프로세서 회로부로부터의 제1 요청을 검출하는 단계;

상기 클라이언트 요청 벡터의 마스킹된 버전의 제2 리딩 원의 제2 위치에 기초하여, 상기 RAM 타일에 액세스하기 위한 제2 프로세서 회로부로부터의 제2 요청을 검출하는 단계; 및

상기 제1 요청의 제1 우선순위와 상기 제2 요청의 제2 우선순위에 기초하여 상기 제1 요청 또는 상기 제2 요청 중 어느 것이 상기 RAM 타일에 배정될지를 결정하는 단계를 포함하는 방법.

청구항 19

제18항에 있어서,

상기 RAM 타일은 제1 RAM 타일이고, 상기 메모리 슬라이스는 제2 RAM 타일을 포함하고, 상기 클라이언트 요청 벡터의 상기 마스킹된 버전은 제1 마스킹된 버전이고, 상기 방법은 또한,

상기 클라이언트 요청 벡터의 제1 트레일링 원(trailing one)의 제3 위치에 기초하여, 상기 제2 RAM 타일에 액세스하기 위한 제3 프로세서 회로부로부터의 제3 요청을 검출하는 단계;

상기 클라이언트 요청 벡터의 제2 마스킹된 버전의 제2 트레일링 원의 제4 위치에 기초하여, 상기 제2 RAM 타일에 액세스하기 위한 제4 프로세서 회로부로부터의 제4 요청을 검출하는 단계; 및

상기 제3 요청의 제3 우선순위와 상기 제4 요청의 제4 우선순위에 기초하여 상기 제3 요청 또는 상기 제4 요청 중 어느 것이 상기 제2 RAM 타일에 배정될지를 결정하는 단계를 포함하는 방법.

청구항 20

제18항에 있어서,

상기 RAM 타일은 제1 RAM 타일이고, 상기 메모리 슬라이스는 제2 RAM 타일을 갖고, 상기 방법은 또한, 상기 제1 요청이 상기 제1 RAM 타일과 상기 제2 RAM 타일에 배정된 것에 응답하여,

상기 제1 RAM 타일과 상기 제2 RAM 타일 중 상기 제1 요청을 배정할 첫 번째 타일을 선택하는 단계; 및

상기 제1 RAM 타일과 상기 제2 RAM 타일 중 두 번째 타일에 제3 요청 - 상기 제3 요청은 상기 제1 요청의 상기 제1 우선순위와 비교하여 다음으로 가장 높은 우선순위를 가짐 - 을 배정하는 단계를 포함하는 방법.

청구항 21

제18항에 있어서,

상기 클라이언트 요청 벡터는 상기 RAM 타일에 액세스하기 위한 각각의 요청을 발행한 것이 제1 프로세서인지 제2 프로세서인지 여부를 나타내는 적어도 두 개의 비트들을 포함하는 것인, 방법.

청구항 22

제18항에 있어서,

상기 방법은 또한, 상기 제1 요청, 상기 제2 요청, 또는 제3 요청 중 어느 것이 상기 RAM 타일에 배정될지 (a) 상기 제1 요청의 상기 제1 우선순위, (b) 상기 제2 요청의 상기 제2 우선순위, 또는 (c) 상기 제3 요청의 제3 우선순위 중 적어도 하나에 기초하여 결정하는 단계를 포함하고, 상기 제3 요청은 최우선순위(super-priority) 벡터에 포함되고, 상기 제2 우선순위는 우선순위 벡터에 기초하여 결정되는 것인, 방법.

청구항 23

제22항에 있어서,

상기 최우선순위 벡터에 포함된 상기 제3 요청의 상기 제3 우선순위는 상기 제1 우선순위, 상기 제2 우선순위, 및 상기 제3 우선순위 중 가장 높은 레벨의 우선순위이고,

상기 클라이언트 요청 벡터의 상기 마스킹된 버전에 포함된 상기 제2 요청의 상기 제2 우선순위는 상기 우선순위 벡터에 기초하여 우선순위 레벨이 배정되고, 상기 우선순위 벡터는 상기 클라이언트 요청 벡터의 상기 마스킹된 버전을 우선순위 및 비-우선순위(non-priority) 요청들로 분할하고,

상기 클라이언트 요청 벡터에 포함된 상기 제1 요청의 상기 제1 우선순위는 상기 제1 우선순위, 상기 제2 우선순위, 및 상기 제3 우선순위 중 가장 낮은 레벨의 우선순위를 갖는 것인, 방법.

청구항 24

제22항에 있어서,

상기 최우선순위 벡터는 런타임 스케줄러에 의해 제공되고, 상기 최우선순위 벡터는 상기 RAM 타일에 액세스하기 위한 상기 요청들 중 애플리케이션에 관련된 트래픽 클래스들과 연관된 요청들을 우선화하고(prioritize), 상기 트래픽 클래스들은 비디오 트래픽 클래스들을 포함하는 것인, 방법.

청구항 25

제18항, 제19항, 제20항, 제21항, 제22항, 제23항, 또는 제24항 중 어느 한 항에 있어서,

상기 방법은 또한, 상기 클라이언트 요청 벡터와 비트마스크에 AND 연산을 수행함으로써 상기 클라이언트 요청 벡터의 상기 마스킹된 버전을 생성하는 단계를 포함하는 방법.

청구항 26

삭제

청구항 27

삭제

청구항 28

삭제

청구항 29

삭제

청구항 30

삭제

청구항 31

삭제

청구항 32

삭제

청구항 33

삭제

청구항 34

삭제

청구항 35

삭제

청구항 36

삭제

청구항 37

삭제

청구항 38

삭제

청구항 39

삭제

청구항 40

삭제

청구항 41

삭제

청구항 42

삭제

청구항 43

삭제

청구항 44

삭제

청구항 45

삭제

청구항 46

삭제

청구항 47

삭제

청구항 48

삭제

청구항 49

삭제

청구항 50

삭제

발명의 설명

기 술 분 야

[0001] **관련 출원에 대한 상호 참조**

[0002] 본 출원은, 2013년 11월 6일자로 제출된 루마니아 특허 출원 OSIM 등록 A/00812(발명의 명칭: "APPARATUS, SYSTEMS, AND METHODS FOR PROVIDING CONFIGURABLE AND COMPOSABLE COMPUTATIONAL IMAGING PIPELINE") 및 2013년 8월 8일자로 제출된 영국 특허 출원 제GB1314263.3호(발명의 명칭: "CONFIGURABLE AND COMPOSABLE COMPUTATIONAL IMAGING PIPELINE")에 관한 우선권을 주장하는, 2013년 11월 18일자로 제출된 미국 특허 출원 제14/082,396호(발명의 명칭: "APPARATUS, SYSTEMS, AND METHODS FOR PROVIDING COMPUTATIONAL IMAGING PIPELINE")의 앞선 우선권 일자의 이익을 주장한다. 본 출원은 또한, 2013년 11월 6일자로 제출된 루마니아 특허 출원 OSIM 등록 A/00812(발명의 명칭: "APPARATUS, SYSTEMS, AND METHODS FOR PROVIDING CONFIGURABLE AND COMPOSABLE COMPUTATIONAL IMAGING PIPELINE") 및 2013년 8월 8일자로 제출된 영국 특허 출원 제GB1314263.3호(발명의 명칭: "CONFIGURABLE AND COMPOSABLE COMPUTATIONAL IMAGING PIPELINE")에 관한 우선권을 주장하는, 2013년 11월 18일자로 제출된 미국 특허 출원 제14/082,645호(발명의 명칭: "APPARATUS, SYSTEMS, AND METHODS FOR PROVIDING CONFIGURABLE COMPUTATIONAL IMAGING PIPELINE")의 앞선 우선권 일자의 이익을 주장한다. 이들 기초 출원들 중 각각의 출원은 그들 전문이 참고로 본 명세서에 편입된다.

[0003] **기술분야**

[0004] 본 출원은 일반적으로는 이미지 및 비디오 프로세싱에 적합한 프로세싱 디바이스에 관한 것이다.

배 경 기 술

[0005] 컴퓨터 이미지 및 비디오 프로세싱은, 초당 높은 수백 메가픽셀에서의 취합 픽셀 레이트가 흔하면서, 이미지 해상도 및 프레임 레이트가 높으므로 메모리 대역폭의 관점에서 매우 요구가 많아지고 있다. 더욱, 이러한 분야는 비교적 초창기이므로, 알고리즘이 끊임없이 유동하고 있다. 그래서, 알고리즘에 대한 변경은 하드웨어가 적응할 수 없음을 의미하므로 그것들을 하드웨어에서 전적으로 구현하는 것은 어렵다. 동시에, 프로세서 단독으로의 구현에 의존하는 소프트웨어 접근법은 비현실적이다. 따라서, 일반적으로는, 프로세서 및 하드웨어 가속기를 수용할 수 있는, 융통성 있는 아키텍처/기반구조를 제공하는 것이 바람직하다.

[0006] 동시에, 그러한 비디오 및 이미지 프로세싱에 대한 수요는, 전력 소모가 핵심 고려사항인, 휴대용 전자 디바이스, 예를 들어 태블릿 컴퓨터 및 모바일 디바이스로부터 크게 다가오고 있다. 결과로서, 프로그래밍 가능한 멀티코어 프로세서 및 하드웨어 가속기를 그것들이 휴대용 전자 디바이스에 필요한 저전력 레벨로 지속적 데이터 전송률을 전달할 수 있게 하는 높은 대역폭 메모리 서브시스템과 결합할 융통성 있는 기반구조에 대한 일반적 필요성이 있다.

발명의 내용

[0007] 개시된 주제 사항에 의하면, 설정가능하고 구성가능한 컴퓨터 이미징 파이프라인을 제공하기 위한 장치, 시스템 및 방법이 제공된다.

[0008] 개시된 주제 사항은 병렬 프로세싱 디바이스를 포함한다. 프로세싱 디바이스는 명령어들을 실행하도록 각각 구성된 복수의 프로세싱 엘리먼트 및 복수의 프로세싱 엘리먼트 중 하나와 연관된 제1 메모리 슬라이스를 포함하는 복수의 메모리 슬라이스를 포함하는 메모리 서브시스템을 포함한다. 제1 메모리 슬라이스는 개개의 판독 및

기록 포트를 각각 갖는 복수의 랜덤 액세스 메모리(RAM) 타일을 포함한다. 병렬 프로세싱 디바이스는 복수의 프로세싱 엘리먼트와 메모리 서브시스템을 결합하도록 구성된 상호 접속 시스템을 포함할 수 있다. 상호접속 시스템은 복수의 프로세싱 엘리먼트 중 하나와 제1 메모리 슬라이스를 결합하도록 구성된 로컬 상호접속부, 및 복수의 프로세싱 엘리먼트 중 나머지와 상기 제1 메모리 슬라이스를 결합하도록 구성된 글로벌 상호접속부를 포함할 수 있다.

[0009] 일부 실시예에 있어서, 복수의 RAM 타일 중 하나는 중재 블록과 연관되되, 중재 블록은 복수의 프로세싱 엘리먼트 중 하나로부터 메모리 액세스 요청들을 수신하고, 복수의 프로세싱 엘리먼트 중 하나에, 복수의 RAM 타일 중 하나로의 액세스를 승인하도록 구성된다.

[0010] 일부 실시예에 있어서, 중재 블록은 라운드-로빈 방식으로 복수의 RAM 타일 중 하나로의 액세스를 승인하도록 구성된다.

[0011] 일부 실시예에 있어서, 중재 블록은 복수의 RAM 타일 중 하나로의 메모리 액세스 요청들을 모니터링하고 복수의 프로세싱 엘리먼트 중 2개 이상이 동시에 복수의 RAM 타일 중 하나에 액세스하려 시도하고 있는지 결정하도록 구성된 충돌 검출기를 포함한다.

[0012] 일부 실시예에 있어서, 충돌 검출기는 복수의 주소 디코더에 결합되되, 복수의 주소 디코더의 각각은 복수의 프로세싱 엘리먼트 중 하나에 결합되고 복수의 프로세싱 엘리먼트 중 하나가 중재 블록과 연관된 복수의 RAM 타일 중 하나에 액세스하려 시도하고 있는지 결정하도록 구성된다.

[0013] 일부 실시예에 있어서, 복수의 프로세싱 엘리먼트는 적어도 하나의 벡터 프로세서 및 적어도 하나의 하드웨어 가속기를 포함한다.

[0014] 일부 실시예에 있어서, 병렬 프로세싱 디바이스는 복수의 메모리 슬라이스 중 하나로의 액세스를 제공하도록 각각 구성된 복수의 메모리 슬라이스 컨트롤러를 포함한다.

[0015] 일부 실시예에 있어서, 상호접속 시스템은 적어도 하나의 벡터 프로세서와 메모리 서브시스템 간 통신을 제공하도록 구성된 제1 버스를 포함한다.

[0016] 일부 실시예에 있어서, 상호접속 시스템은 적어도 하나의 하드웨어 가속기와 메모리 서브시스템 간 통신을 제공하도록 구성된 제2 버스 시스템을 포함한다.

[0017] 일부 실시예에 있어서, 제2 버스 시스템은 적어도 하나의 하드웨어 가속기로부터 메모리 액세스 요청을 수신함으로써 그리고, 적어도 하나의 하드웨어 가속기에, 메모리 서브시스템으로의 액세스를 승인함으로써 적어도 하나의 하드웨어 가속기와 메모리 서브시스템 간 통신을 중재하도록 구성된 슬라이스 주소 요청 필터를 포함한다.

[0018] 일부 실시예에 있어서, 복수의 프로세싱 디바이스 중 하나는 메모리 서브시스템의 처리율을 증가시키도록 버퍼를 포함하되, 버퍼 내 엘리먼트의 수는 메모리 서브시스템으로부터 데이터를 검색하기 위한 사이클의 수보다 더 크다.

[0019] 개시된 주제 사항은 병렬 프로세싱 시스템을 동작시키기 위한 방법을 포함한다. 방법은 제1 프로세싱 엘리먼트 및 제2 프로세싱 엘리먼트를 포함하는 복수의 프로세싱 엘리먼트를 제공하는 단계를 포함하되, 복수의 프로세싱 엘리먼트의 각각은 명령어들을 실행하도록 구성된다. 방법은 또한 제1 프로세싱 엘리먼트와 연관된 제1 메모리 슬라이스를 포함하는 복수의 메모리 슬라이스를 포함하는 메모리 서브시스템을 제공하는 단계를 포함하되, 제1 메모리 슬라이스는 개개의 판독 및 기록 포트를 각각 갖는 복수의 랜덤 액세스 메모리(RAM) 타일을 포함한다. 방법은 제1 프로세싱 엘리먼트로부터의 제1 메모리 액세스 요청을, 상호접속 시스템의 로컬 상호접속부를 통하여 복수의 RAM 타일 중 하나와 연관된 중재 블록에 의해, 수신하는 단계를 더 포함한다. 방법은 부가적으로는 복수의 RAM 타일 중 하나로의 액세스 권한을 제1 프로세싱 엘리먼트에 부여하도록 제1 프로세싱 엘리먼트에 제1 수권 메시지를, 글로벌 상호접속부를 통하여 중재 블록에 의해, 보내는 단계를 포함한다.

[0020] 일부 실시예에 있어서, 방법은 제2 프로세싱 엘리먼트로부터의 제2 메모리 액세스 요청을, 상호접속 시스템의 글로벌 상호접속부를 통하여 중재 블록에 의해, 수신하는 단계; 및 복수의 RAM 타일 중 하나로의 액세스 권한을 제2 프로세싱 엘리먼트에 부여하도록 제2 프로세싱 엘리먼트에 제2 수권 메시지를, 글로벌 상호접속부를 통하여 중재 블록에 의해, 보내는 단계를 더 포함한다.

[0021] 일부 실시예에 있어서, 방법은 라운드-로빈 방식으로 복수의 RAM 타일 중 하나로의 액세스 권한을 부여하도록 복수의 프로세싱 엘리먼트에 복수의 수권 메시지를, 중재 블록에 의해, 보내는 단계를 더 포함한다.

[0022] 일부 실시예에 있어서, 방법은 복수의 RAM 타일 중 하나로의 메모리 액세스 요청들을, 중재 블록에서의 충돌 검출기에 의해, 모니터링하는 단계; 및 복수의 프로세싱 엘리먼트 중 2개 이상이 동시에 복수의 RAM 타일 중 하나에 액세스하려 시도하고 있는지 결정하는 단계를 더 포함한다.

[0023] 일부 실시예에 있어서, 복수의 프로세싱 엘리먼트는 적어도 하나의 벡터 프로세서 및 적어도 하나의 하드웨어 가속기를 포함한다.

[0024] 일부 실시예에 있어서, 방법은 복수의 메모리 슬라이스 중 하나로의 액세스를 제공하도록 각각 구성된 복수의 메모리 슬라이스 컨트롤러를 제공하는 단계를 더 포함한다.

[0025] 일부 실시예에 있어서, 방법은 상호접속 시스템의 제1 버스 시스템을 통하여 적어도 하나의 벡터 프로세서와 메모리 서브시스템 간 통신을 제공하는 단계를 더 포함한다.

[0026] 일부 실시예에 있어서, 방법은 상호접속 시스템의 제2 버스 시스템을 통하여 적어도 하나의 하드웨어 가속기와 메모리 서브시스템 간 통신을 제공하는 단계를 더 포함한다.

[0027] 일부 실시예에 있어서, 제2 버스 시스템은 적어도 하나의 하드웨어 가속기로부터 메모리 액세스 요청을 수신함으로써 그리고, 적어도 하나의 하드웨어 가속기에, 메모리 서브시스템으로의 액세스를 승인함으로써 적어도 하나의 하드웨어 가속기와 메모리 서브시스템 간 통신을 중재하도록 구성된 슬라이스 주소 요청 필터를 포함한다.

[0028] 개시된 주제 사항은 전자 디바이스를 포함한다. 전자 디바이스는 병렬 프로세싱 디바이스를 포함한다. 프로세싱 디바이스는 명령어들을 실행하도록 각각 구성된 복수의 프로세싱 엘리먼트 및 복수의 프로세싱 엘리먼트 중 하나와 연관된 제1 메모리 슬라이스를 포함하는 복수의 메모리 슬라이스를 포함하는 메모리 서브시스템을 포함한다. 제1 메모리 슬라이스는 개개의 판독 및 기록 포트를 각각 갖는 복수의 랜덤 액세스 메모리(RAM) 타일을 포함한다. 병렬 프로세싱 디바이스는 복수의 프로세싱 엘리먼트와 메모리 서브시스템을 결합하도록 구성된 상호접속 시스템을 포함할 수 있다. 상호접속 시스템은 복수의 프로세싱 엘리먼트 중 하나와 제1 메모리 슬라이스를 결합하도록 구성된 로컬 상호접속부, 및 복수의 프로세싱 엘리먼트 중 나머지와 상기 제1 메모리 슬라이스를 결합하도록 구성된 글로벌 상호접속부를 포함할 수 있다. 전자 디바이스는 또한 메모리에 저장된 모듈을 실행하도록 구성된, 병렬 프로세싱 디바이스와 통신하고 있는, 프로세서를 포함한다. 모듈은 데이터 프로세싱 프로세스와 연관된 흐름 그래프를 수신하고 복수의 노드 중 제1 노드를 병렬 프로세싱 디바이스의 제1 프로세싱 엘리먼트에 그리고 복수의 노드 중 제2 노드를 병렬 프로세싱 디바이스의 제2 프로세싱 엘리먼트에 배정하고, 그로써 제1 노드 및 제2 노드와 연관된 연산들을 병렬화하도록 구성되되, 흐름 그래프는 복수의 노드 및 복수의 노드 중 2개 이상을 접속하는 복수의 에지를 포함하고, 각각의 노드는 연산을 식별시키고 각각의 에지는 접속된 노드들 간 관계를 식별시킨다.

[0029] 일부 실시예에 있어서, 흐름 그래프는 확장성 마크업 언어(XML) 포맷으로 제공된다.

[0030] 일부 실시예에 있어서, 모듈은 병렬 프로세싱 디바이스에서의 메모리 서브시스템의 과거 성능에 기반하여 복수의 노드 중 제1 노드를 제1 프로세싱 엘리먼트에 배정하도록 구성된다.

[0031] 일부 실시예에 있어서, 병렬 프로세싱 디바이스의 메모리 서브시스템은 사전 결정된 시간 기간에 걸쳐 메모리 충돌의 수를 카운팅하도록 구성되는 카운터를 포함하고, 메모리 서브시스템의 과거 성능은 카운터에 의해 측정된 메모리 충돌의 수를 포함한다.

[0032] 일부 실시예에 있어서, 모듈은 병렬 프로세싱 디바이스가 흐름 그래프의 적어도 일부를 동작시키고 있는 동안 복수의 노드 중 제1 노드를 제1 프로세싱 엘리먼트에 배정하도록 구성된다.

[0033] 일부 실시예에 있어서, 모듈은 복수의 흐름 그래프를 수신하고, 복수의 흐름 그래프와 연관된 모든 연산을 병렬 프로세싱 디바이스에서의 단일 프로세싱 엘리먼트에 배정하도록 구성된다.

[0034] 일부 실시예에 있어서, 모듈은 메모리 충돌을 감축하도록 프로세싱 엘리먼트들에 의한 메모리 액세스들을 스태거링(staggering)하도록 구성된다.

[0035] 일부 실시예에 있어서, 전자 디바이스는 모바일 디바이스를 포함한다.

[0036] 일부 실시예에 있어서, 흐름 그래프는 병렬 프로세싱 디바이스와 연관된 애플리케이션 프로그래밍 인터페이스(API)를 사용하여 특정된다.

[0037] 일부 실시예에 있어서, 모듈은 입력 이미지 데이터를 스트립들로 분할하고 입력 이미지 데이터의 하나의 스트립

을 복수의 프로세싱 엘리먼트 중 하나에 제공함으로써 입력 이미지 데이터를 복수의 프로세싱 엘리먼트에 제공하도록 구성된다.

[0038] 일부 실시예에 있어서, 입력 이미지 데이터의 스트립의 수는 복수의 프로세싱 엘리먼트의 수와 동일하다.

[0039] 개시된 주제 사항은 방법을 포함한다. 방법은 데이터 프로세싱 프로세스와 연관된 흐름 그래프를, 병렬 프로세싱 디바이스와 통신하고 있는 프로세서에서, 수신하는 단계를 포함하되, 흐름 그래프는 복수의 노드 및 복수의 노드 중 2개 이상을 접속하는 복수의 에지를 포함하고, 각각의 노드는 연산을 식별시키고 각각의 에지는 접속된 노드들 간 관계를 식별시킨다. 방법은 또한 복수의 노드 중 제1 노드를 병렬 프로세싱 디바이스의 제1 프로세싱 엘리먼트에 그리고 복수의 노드 중 제2 노드를 병렬 프로세싱 디바이스의 제2 프로세싱 엘리먼트에 배정하고, 그로써 제1 노드 및 제2 노드와 연관된 연산을 병렬화하는 단계를 포함한다. 병렬 프로세싱 디바이스는 또한 제1 프로세싱 엘리먼트와 연관된 제1 메모리 슬라이스를 포함하는 복수의 메모리 슬라이스를 포함하는 메모리 서브시스템으로서, 제1 메모리 슬라이스는 개개의 판독 및 기록 포트를 각각 갖는 복수의 랜덤 액세스 메모리(RAM) 타일을 포함하는 메모리 서브시스템; 및 제1 프로세싱 엘리먼트, 제2 프로세싱 엘리먼트, 및 메모리 서브시스템을 결합하도록 구성된 상호접속 시스템을 포함한다. 상호접속 시스템은 제1 프로세싱 엘리먼트와 제1 메모리 슬라이스를 결합하도록 구성된 로컬 상호접속부, 및 제2 프로세싱 엘리먼트와 제1 메모리 슬라이스를 결합하도록 구성된 글로벌 상호접속부를 포함한다.

[0040] 일부 실시예에 있어서, 복수의 노드 중 제1 노드를 병렬 프로세싱 디바이스의 제1 프로세싱 엘리먼트에 배정하는 단계는 병렬 프로세싱 디바이스에서의 제1 메모리 슬라이스의 과거 성능에 기반하여 복수의 노드 중 제1 노드를 제1 프로세싱 엘리먼트에 배정하는 단계를 포함한다.

[0041] 일부 실시예에 있어서, 방법은 또한 사전 결정된 시간 기간에 걸쳐 제1 메모리 슬라이스에서의 메모리 충돌의 수를, 메모리 서브시스템에서의 카운터에서, 카운팅하는 단계를 포함하고, 제1 메모리 슬라이스의 과거 성능은 제1 메모리 슬라이스에서의 메모리 충돌의 수를 포함한다.

[0042] 일부 실시예에 있어서, 복수의 노드 중 제1 노드를 제1 프로세싱 엘리먼트에 배정하는 단계는 병렬 프로세싱 디바이스가 흐름 그래프의 적어도 일부를 동작시키고 있는 동안 수행된다.

[0043] 일부 실시예에 있어서, 방법은 또한 메모리 충돌을 감축하기 위해 제1 메모리 슬라이스로의 프로세싱 엘리먼트들에 의한 메모리 액세스들을 스태거링하는 단계를 포함한다.

[0044] 일부 실시예에 있어서, 흐름 그래프는 병렬 프로세싱 디바이스와 연관된 애플리케이션 프로그래밍 인터페이스(API)를 사용하여 특정된다.

[0045] 일부 실시예에 있어서, 방법은 또한 입력 이미지 데이터를 복수의 스트립으로 분할하고 상기 입력 이미지 데이터의 복수의 스트립 중 하나를 복수의 프로세싱 엘리먼트 중 하나에 제공함으로써 입력 이미지 데이터를 복수의 프로세싱 엘리먼트에 제공하는 단계를 포함한다.

[0046] 일부 실시예에 있어서, 입력 이미지 데이터의 복수의 스트립의 수는 복수의 프로세싱 엘리먼트의 수와 동일하다.

도면의 간단한 설명

[0047] 개시된 주제 사항의 다양한 목적, 특징 및 이점은 유사한 참조 숫자가 유사한 구성요소를 식별시키는 이하의 도면과 연관하여 고려될 때 개시된 주제 사항의 이하의 상세한 설명을 참조하여 더 충분히 인식될 수 있다.

도 1은 키메라의 컴퓨터 이미징 플랫폼의 설명도;

도 2는 셀 프로세서의 멀티코어 아키텍처의 설명도;

도 3은 효율적 저-전력 마이크로프로세서(ELM) 아키텍처의 설명도;

도 4는 일부 실시예에 따른 개선된 메모리 서브시스템의 예시도;

도 5는 일부 실시예에 따른 병렬 프로세싱 디바이스의 섹션의 예시도;

도 6은 일부 실시예에 따른 타일 제어 로직 내 중앙집중형 충돌 검출 시스템의 예시도;

도 7은 일부 실시예에 따른 타일 제어 로직 내 분산형 충돌 검출 시스템의 예시도;

도 8은 일부 실시예에 따라 요청기에 충돌 신호를 보고하기 위한 중재 블록도;

도 9는 일부 실시예에 따른 사이클-지향 중재 블록의 예시도;

도 10은 일부 실시예에 따라 메모리 액세스 중재에 기인하는 메모리 액세스 레이턴시를 감축하기 위한 메커니즘의 예시도;

도 11은 일부 실시예에 따른 스케줄링 소프트웨어의 애플리케이션의 예시도;

도 12는 일부 실시예에 따라 병렬 프로세싱 디바이스를 갖는 시스템의 계층적 구조도;

도 13은 일부 실시예에 따라 방향성 비사이클 그래프(DAG) 또는 흐름 그래프의 기술이 어떻게 병렬 프로세싱 디바이스의 연산을 제어하는데 사용될 수 있는지의 예시도;

도 14a 내지 도 14b는 일부 실시예에 따라 컴파일러 및 스케줄러에 의한 태스크의 스케줄링 및 발행의 예시도;

도 15는 일부 실시예에 따른 실시간 DAG 컴파일러의 연산의 예시도;

도 16은 일부 실시예에 따라 제안된 온라인 DAG 스케줄러에 의해 발생된 스케줄 대 개방형 컴퓨팅 언어(OpenCL) 스케줄러에 의해 발생된 스케줄의 비교도;

도 17은 일부 실시예에 따라 프로세서 및/또는 필터 가속기의 연산을 동기화하기 위한 배리어 메커니즘의 예시도;

도 18은 일부 실시예에 따라 여러 다른 유형의 프로세싱 엘리먼트를 갖는 병렬 프로세싱 디바이스의 예시도;

도 19는 일부 실시예에 따라 제안된 멀티코어 메모리 서브시스템의 예시도;

도 20은 일부 실시예에 따라 접속 매트릭스(CMX) 기반구조의 단일 슬라이스의 예시도;

도 21은 일부 실시예에 따라 가속기 메모리 컨트롤러(AMC) 크로스바 아키텍처의 예시도;

도 22는 일부 실시예에 따라 AMC 크로스바 포트 컨트롤러의 예시도;

도 23은 일부 실시예에 따라 AMC를 사용하는 판독 연산의 예시도;

도 24는 일부 실시예에 따라 AMC를 사용하는 기록 연산의 예시도;

도 25는 일부 실시예에 따른 병렬 프로세싱 디바이스의 예시도; 및

도 26은 일부 실시예에 따라 병렬 프로세싱 디바이스를 포함하는 전자 디바이스의 예시도.

발명을 실시하기 위한 구체적인 내용

[0048] 이하의 설명에서는, 개시된 주제 사항의 철저한 이해를 제공하기 위해 개시된 주제 사항의 시스템 및 방법과 그러한 시스템 및 방법이 동작 등을 할 수 있는 환경에 관하여 수많은 특정 상세가 제시된다. 그렇지만, 개시된 주제 사항이 그러한 특정 상세 없이 실시될 수 있음과, 당업계에 주지되어 있는 소정 특징이 개시된 주제 사항의 주제 사항의 복잡화를 회피하기 위해 상세히 설명되지는 않음은 당업자에게 명백할 것이다. 부가적으로, 아래에서 제공되는 예들은 예시적인 것임과, 개시된 주제 사항의 범위 내에 드는 다른 시스템 및 방법이 있다고 생각됨을 이해할 것이다.

[0049] 그러한 여러 다른 프로세싱 자원(예를 들어, 프로세서 및 하드웨어 가속기)을 상호접속하는 하나의 가능한 방식은 엔비디아(NVidia)에 의해 개발된 컴퓨터 포토그래피 엔진 키메라(Chimera)에서 개괄된 바와 같은 버스를 사용하는 것이다. 도 1은 컴퓨터 포토그래피 엔진 키메라를 예시하고 있다. 키메라 컴퓨터 포토그래피 엔진(100)은 플랫-레벨 버스 기반구조(108)(예를 들어, 모든 프로세싱 엘리먼트를 접속하는 단일 계층 버스 시스템)를 통하여 멀티코어 ARM 프로세서 서브-시스템(104) 및 하드웨어(HW) 이미지 신호 프로세싱(ISP) 가속기(106)에 접속되는 다중 그래픽 프로세싱 유닛(GPU) 코어(102)를 포함한다. 키메라 컴퓨터 포토그래피 엔진은 프로그래머로부터 기저 GPU 코어(102), CPU(104) 및 ISP 블록(106)의 상세를 추상화하는 소프트웨어 프레임워크로서 일반적으로 제시된다. 더욱, 키메라 컴퓨터 포토그래피 엔진(100)은 2개의 정보 버스(108-0, 108-1), 이미지 또는 프레임 데이터를 반송하는 제1 버스(108-0) 및 각각의 프레임과 연관된 상태 정보를 반송하는 제2 버스(108-1)를 통하여 그들 컴퓨터 포토그래피 엔진을 통하는 데이터 흐름을 기술한다.

[0050] 키메라에서와 같은, 플랫-레벨 버스 기반구조의 사용은 구현하기가 저렴하고 편리할 수 있다. 그렇지만, 플랫-

레벨 버스 기반구조의 사용은, GPU 코어(102), CPU(104) 및 ISP 블록(106)과 같은, 이종 프로세싱 엘리먼트(예를 들어, 다양한 유형의 프로세싱 엘리먼트)를 상호접속하는 수단으로서 여러 주목할만한 단점을 가질 수 있다. 첫째로, 컴퓨터 자원을 상호접속하는데 버스의 사용은 각각의 중앙 프로세싱 유닛(CPU)(104), 그래픽 프로세싱 유닛(GPU)(102), 및/또는 이미지 신호 프로세서(ISP) 블록(106)에 로컬인 시스템 도처에 메모리가 분산될 수 있음을 의미한다. 그래서, 메모리는 프로그래머가 구현하기를 바라는 컴퓨터 포토그래피 파이프라인의 요건에 따라 프로세싱 파이프라인 내에서 융통성 있게 할당될 수 없다. 이러한 융통성 부족은 어떤 이미지 또는 비디오 프로세싱이 구현하기가 어렵거나 불가능함 또는 구현이 프레임-레이트, 이미지 품질 또는 그 밖의 관점에서 제한됨을 의미할 수 있다.

[0051] 둘째로, 플랫-레벨 버스 기반구조의 사용은 또한 여러 다른 컴퓨터 자원(CPU(104), GPU(102) 및 ISP 블록(106))이 버스 대역폭을 두고 경합하여야 함을 의미한다. 이러한 경합은 이용가능한 버스 대역폭의 양을 감축하는 중재를 필요로 한다. 그래서, 실제 작업에는 이론적 대역폭 중 점점 더 적은 부분이 이용가능하다. 대역폭의 감축은 프로세싱 파이프라인이 프레임 레이트, 이미지 품질 및/또는 전력의 관점에서 애플리케이션의 성능 요건을 충족하지 못함을 의미할 수 있다.

[0052] 셋째로, 특정 컴퓨터 자원에 근접한 불충분한 메모리는 주어진 GPU(102), CPU(104), 또는 하드웨어 ISP 블록(106) 및 다른 컴퓨터 자원과 연관된 메모리 사이에서 데이터가 이리저리 전송되어야 함을 의미할 수 있다. 이러한 집약성 부족은 부가적 버스 대역폭 및 중재 오버헤드가 초래될 수 있음을 의미한다. 더욱, 집약성 부족은 또한 부가적 전력이 소모됨을 의미한다. 그래서, 특정 목표 프레임-레이트로 특정 알고리즘을 지원하는 것이 어렵거나 불가능할 수 있다.

[0053] 넷째로, 플랫-레벨 버스 기반구조의 사용은 또한, 각각 다른 레이턴시 특성을 가질 수 있는, 이종 프로세싱 엘리먼트로부터 파이프라인을 구성하는데 어려움을 악화시킬 수 있다. 예를 들면, GPU 코어(102)는 레이턴시를 커버하기 위해 메모리(보통은 외부 DRAM)로의 다중의 현저한 액세스를 취급하도록 다중 중첩 프로세스 스레드를 실행함으로써 레이턴시를 감내하도록 설계되는 반면, 정규 CPU(104) 및 하드웨어 ISP 블록(106)은 레이턴시를 감내하도록 설계되지 않는다.

[0054] 여러 다른 프로세싱 자원을 상호접속하는 다른 방식은, 도 2에 예시되는, IBM에 의해 개발된 셀 프로세서 아키텍처에 의해 제공된다. 셀 프로세서 아키텍처(200)는 시너지 실행 유닛(SXU)이라고도 알려져 있는 각각의 프로세서(204)가 이용가능한 로컬 저장소(LS)(202)를 포함한다. 셀 프로세서(200)는 하나의 프로세서의 LS(202)와 다른 하나의 프로세서의 LS(202) 사이의 데이터-전송을 프로그램으로 스케줄링하도록 시간-공유 기반구조 및 직접 메모리 액세스(DMA)(206) 전송에 의존한다. 셀 아키텍처(200)에 대한 어려움은 필요할 때 각각의 프로세서(204)가 공유 데이터를 이용가능함을 보장하도록 (셀 아키텍처(200)에서의 높은 레이턴시에 기인하여) 미리 수백 사이클 배경 데이터-전송을 명시적으로 스케줄링하도록 프로그래머가 직면한 복잡도이다. 프로그래머가 배경 데이터 전송을 명시적으로 스케줄링하지 않으면, 프로세서(204)는 멎어서, 성능을 저하시킬 것이다.

[0055] 여러 다른 프로세싱 자원을 상호접속하는 다른 방식은 멀티코어 프로세싱 시스템에서 프로세서들 사이에 효율적으로 데이터를 공유하도록 공유 멀티코어 메모리 서브시스템을 사용하는 것이다. 이러한 공유 멀티코어 메모리 서브시스템은 효율적 저-전력 마이크로프로세서(ELM) 시스템에서 사용된다. 도 3은 ELM 시스템을 예시한다. ELM 시스템(300)은 앙상블(302), ELM 시스템에서 자원을 컴퓨팅하기 위한 일차적 물리적 설계 유닛을 포함한다. 앙상블(302)은 소결합되는 4개의 프로세서(304)의 클러스터를 포함한다. 4개의 프로세서(304)의 클러스터는 상호접속 네트워크로의 인터페이스 및 앙상블 메모리(306)와 같은 로컬 자원을 공유한다. 앙상블 메모리(306)는 프로세서(304) 가까이에 있는 명령어 및 데이터 작업 세트를 캡처링하고, 로컬 프로세서(304) 및 네트워크 인터페이스 컨트롤러가 동시다발적으로 그것에 액세스할 수 있게 하도록 뱅킹된다. 앙상블(302) 내 각각의 프로세서(304)에는 앙상블 메모리(306)에서 선호되는 뱅크가 배정된다. 그 선호되는 뱅크로의 프로세서(304)에 의한 액세스는 다른 프로세서 및 네트워크 인터페이스에 의한 액세스에 앞서(그리고 액세스를 차단할 것임) 우선시된다. 단일 프로세서(304)에 사유인 명령어 및 데이터는 결정론적 액세스 시간을 제공하도록 그 선호되는 뱅크에 저장될 수 있다. 판독 및 기록 포트로의 액세스를 제어하는 중재기는 프로세서(304)와 메모리 뱅크(306) 사이의 친화도를 확립하도록 바이어싱된다. 이것은 소프트웨어가 다중 프로세서에 의해 공유될 수 있는 데이터에 액세스할 때 대역폭 가용성 및 레이턴시에 대한 더 강한 가정을 할 수 있게 한다.

[0056] 그렇지만, ELM 아키텍처(300)는 물리적으로 큰 랜덤 액세스 메모리(RAM) 블록에 기인하여 많은 전력을 소모할 수 있다. 더욱, ELM 아키텍처(300)는 프로세서(304)들 사이에 많은 데이터-공유가 있는 경우 낮은 처리율에 시달릴 수 있다. 부가적으로, 프로비저닝은, 어떤 경우에는 전력 및 성능의 관점에서 유익할 수 있는, 프로세서

(304)와 하드웨어 가속기 사이의 데이터-공유에 대해서는 이루어지지 않는다.

[0057] 본 발명은 다중 프로세서 및 하드웨어 가속기가 다른 프로세서 및 하드웨어 가속기와 동시에 공유 데이터에 액세스 가능하게 하기 위한 장치, 시스템 및 방법에 관한 것이다. 본 발명은, 로컬 저장소에 액세스하는데 더 높은 친화도(예를 들어, 더 높은 우선순위)를 갖는 로컬 프로세서에 의해 차단됨이 없이, 공유 데이터에 동시에 액세스하기 위한 장치, 시스템 및 방법을 제공한다.

[0058] 개시된 장치, 시스템 및 방법은 기존 멀티코어 메모리 아키텍처에 비해 실질적 이점을 제공한다. 기존 멀티코어 메모리 아키텍처는, 데이터가 액세스될 수 있는 대역폭을 제한할 수 있는, 프로세서당 RAM의 단일 모놀리식 블록을 사용한다. 개시된 아키텍처는 RAM의 단일 모놀리식 블록을 사용하는 기존 멀티코어 메모리 아키텍처에 비해 실질적으로 더 높은 대역폭으로 메모리에 액세스하기 위한 메커니즘을 제공할 수 있다. 개시된 아키텍처는, 프로세서당 단일의 큰 RAM 블록을 인스턴스 생성하는 대신에, 프로세서당 다중 물리적 RAM 블록을 인스턴스 생성함으로써 이러한 더 높은 대역폭을 획득한다. 각각의 RAM 블록은 전용 액세스 중재 블록 및 주위 기반구조를 포함할 수 있다. 그래서, 메모리 서브시스템 내 각각의 RAM 블록은 시스템 내 다중 프로세싱 엘리먼트, 예를 들어, 벡터 프로세서, 축소형 명령어 세트 컴퓨팅(RISC) 프로세서, 하드웨어 가속기 또는 DMA 엔진에 의해 다른 것들과 독립적으로 액세스될 수 있다.

[0059] 다중의 작은 RAM 인스턴스의 사용이 단일의 큰 RAM 인스턴스를 사용하는 것에 비해 유익하다는 이러한 것은 단일의 큰 RAM 인스턴스에 기반하는 메모리 뱅크가 다중의 더 작은 RAM 인스턴스에 기반하는 메모리 뱅크보다 더 면적 효율적이기 때문에 다소 반직관적이다. 그렇지만, 더 작은 RAM 인스턴스에 대한 소비 전력은 전형적으로는 단일의 큰 RAM 인스턴스에 대한 것보다 상당히 더 낮다. 더욱, 단일의 큰 물리적 RAM 인스턴스가 멀티-인스턴스 RAM 블록과 동일한 대역폭을 달성한다면, 단일의 큰 물리적 RAM 인스턴스는 다중의 물리적 RAM 인스턴스로 이루어진 것보다 실질적으로 더 높은 전력에 처하게 될 것이다. 그래서, 적어도 소비 전력 관점으로부터, 메모리 서브시스템은 단일의 큰 RAM 인스턴스를 사용하는 것보다 다중의 물리적 RAM 인스턴스를 사용하는 것으로부터 혜택을 볼 수 있다.

[0060] 다중의 물리적 RAM 인스턴스를 갖는 메모리 서브시스템은 더 작은 RAM 블록의 RAM 액세스당 비용, 예를 들어, 메모리 액세스 시간 또는 전력 소모가 전형적으로는 더 큰 RAM 블록의 그것보다 훨씬 더 낮다는 점에서 부가 이점을 가질 수 있다. 이것은 RAM으로부터 데이터를 판독/기록하는데 사용된 단축된 비트-라인에 기인한다. 더욱, 더 작은 RAM 블록에 대한 판독 및 기록을 위한 액세스-시간도 (더 짧은 비트-라인과 연관된 감축된 저항-커패시턴스(RC) 시상수에 기인하여) 더 낮다. 그래서, 멀티-RAM 기반 메모리 서브시스템에 결합된 프로세싱 엘리먼트는 더 높은 주파수에서 동작할 수 있어서, 순차적으로 정적 누설 전류에 기인하는 정적 전력을 감축한다. 이것은 프로세서와 메모리가 전력 도메인으로 격리될 때 특히 유용할 수 있다. 예를 들어, 주어진 프로세서 또는 필터 가속기가 그 태스크를 완료하였을 때, 주어진 프로세서 또는 필터 가속기와 연관된 전력 도메인은 유익하게는 게이트 오프될 수 있다. 그래서, 개시된 아키텍처에서의 메모리 서브시스템은 이용가능한 대역폭 및 소비 전력의 관점에서 우수한 특성을 갖는다.

[0061] 부가적으로, 중재된 액세스를 각각 갖는 다중 RAM 인스턴스를 갖는 메모리 서브시스템은, RAM 블록을 로킹함으로써 RAM 블록을 특정 프로세서에 전용시킴이 없이, 프로세서와 하드웨어 가속기 사이에 데이터가 공유되도록 여러 방식을 제공할 수 있다. 원칙적으로, 더 큰 RAM이 N개의 서브-블록으로 세분되면, 그때 이용가능한 데이터 대역폭은 대략 N의 인수만큼 증가된다. 이것은 다중 프로세싱 엘리먼트에 의한 동시발생적 공유(예를 들어, 액세스 충돌)를 감축하도록 데이터가 적절히 파티셔닝될 수 있다는 가정에 기반한다. 예를 들어, 소비기 프로세서 또는 소비기 가속기가 생산기 프로세서 또는 생산기 가속기에 의해 채워지고 있는 데이터 버퍼로부터 데이터를 판독할 때, 그때 데이터 버퍼의 동시발생적 공유가 있어, 액세스 충돌을 초래한다.

[0062] 일부 실시예에 있어서, 개시된 아키텍처는 데이터의 동시발생적 공유를 감축하기 위한 메커니즘을 제공할 수 있다. 특히, 개시된 아키텍처는 정적 메모리 할당 메커니즘 및/또는 동적 메모리 할당 메커니즘을 통하여 동시발생적 공유를 감축하도록 순종적일 수 있다. 예를 들어, 정적 메모리 할당 메커니즘에 있어서, 데이터는 데이터의 동시발생적 공유를 감축하기 위해, 프로그램이 시작되기 전에, 예를 들어, 프로그램 컴파일 스테이지 동안 메모리의 여러 다른 부분에 매핑된다. 다른 한편으로, 동적 메모리 할당 기법에 있어서, 데이터는 프로그램 실행 동안 메모리의 여러 다른 부분에 매핑된다. 정적 메모리 할당 메커니즘은 데이터에 메모리를 할당하기 위한 예측가능한 메커니즘을 제공하고, 그것은 전력 또는 성능의 관점에서 어떠한 실질적 오버헤드도 초래하지 않는다.

[0063] 다른 일례로서, 개시된 아키텍처는 다중 RAM 블록에 걸쳐 파티셔닝된 데이터-구조로의 액세스를 중재하는 하나

이상의 프로세서 또는 컨트롤러(예를 들어, 감독하는 RISC 프로세서) 상에서 실행되는 런타임 스케줄러와 함께 사용될 수 있다. 런타임 스케줄러는 공유 데이터로의 동시 액세스를 감축하기 위해 데이터(예를 들어, 이미지 프레임)의 부분(예를 들어, 라인 또는 타일)들 상에 동작하는 여러 다른 프로세싱 엘리먼트의 시작-시간을 스태거링하도록 구성될 수 있다.

[0064] 일부 실시예에 있어서, 런타임 스케줄러는 하드웨어 중재 블록으로 보완될 수 있다. 예를 들어, 하드웨어 중재 블록은 멎음을 감축하도록 설계된 공유 결정론적 상호접속부를 통하여 (벡터 프로세서와 같은) 프로세서에 의해 공유 메모리 액세스를 중재하도록 구성될 수 있다. 일부 경우에 있어서, 하드웨어 중재 블록은 사이클-지향 스케줄링을 수행하도록 구성될 수 있다. 사이클-지향 스케줄링은, 다중 클록-사이클을 필요로 할 수 있는 태스크-레벨 입도로 자원의 사용을 스케줄링하는 것과는 대조적으로, 클록-사이클 입도로 자원의 사용을 스케줄링하는 것을 포함할 수 있다. 클록 사이클 입도로 자원 할당을 스케줄링하는 것은 더 높은 성능을 제공할 수 있다.

[0065] 다른 실시예에 있어서, 런타임 스케줄러는 데이터를 저장할 입력 버퍼 및 출력 버퍼를 각각 포함할 수 있는 다수의 하드웨어 가속기로 보완될 수 있다. 입력 및 출력 버퍼는 외부 메모리와 같은 외부 자원에 액세스함에 있어서 지연의 분산을 흡수하도록(또는 감추도록) 구성될 수 있다. 입력 및 출력 버퍼는 선입선출(FIFO) 버퍼를 포함할 수 있고, FIFO 버퍼는 외부 자원에 액세스함에 있어서 지연의 분산을 흡수하기에 충분한 양의 데이터 및/또는 명령어를 저장하기에 충분한 수의 슬롯을 포함할 수 있다.

[0066] 일부 실시예에 있어서, 개시된 장치, 시스템 및 방법은 병렬 프로세싱 디바이스를 제공한다. 병렬 프로세싱 디바이스는 명령어를 각각 실행할 수 있는, 병렬 프로세서와 같은, 복수의 프로세서를 포함할 수 있다. 병렬 프로세싱 디바이스는 또한 복수의 메모리 슬라이스를 포함할 수 있으며, 각각의 메모리 슬라이스는 병렬 프로세싱 디바이스 중 하나와 연관되고 병렬 프로세싱 디바이스 내 다른 프로세싱 디바이스에 비해 그 프로세서에 우선적 액세스를 부여한다. 각각의 메모리 슬라이스는 복수의 RAM 타일을 포함할 수 있으며, 이 경우 각각의 RAM 타일은 판독 포트 및 기록 포트를 포함할 수 있다. 일부 경우에 있어서, 각각의 메모리 슬라이스에는 관련된 메모리 슬라이스로의 액세스를 제공하기 위한 메모리 슬라이스 컨트롤러가 제공될 수 있다. 프로세서 및 RAM 타일은 버스를 통하여 서로 결합될 수 있다. 일부 경우에 있어서, 버스는 프로세서 중 어느 하나를 메모리 슬라이스 중 어느 하나와 결합할 수 있다. 적합하게는, 각각의 RAM 타일은 타일로의 액세스를 승인하기 위한 타일 제어 로직 블록을 포함할 수 있다. 타일 제어 로직 블록은 때로는 타일 제어 로직 또는 중재 블록이라고 지칭된다.

[0067] 일부 실시예에 있어서, 병렬 프로세싱 디바이스는 사전 정의된 프로세싱 기능, 예를 들어, 이미지 프로세싱을 수행하도록 구성된 적어도 하나의 하드웨어 가속기를 더 포함할 수 있다. 일부 경우에 있어서, 사전 정의된 프로세싱 기능은 필터링 연산을 포함할 수 있다.

[0068] 일부 실시예에 있어서, 적어도 하나의 하드웨어 가속기는 별개의 버스를 통하여 메모리 슬라이스에 결합될 수 있다. 별개의 버스는, 적어도 하나의 하드웨어 가속기로부터의 요청을 수신하고 관련된 메모리 슬라이스 컨트롤러를 통해 메모리 슬라이스로의 액세스를 하드웨어 가속기에 승인하도록 구성되는, 연관된 가속기 메모리 컨트롤러(AMC)를 포함할 수 있다. 그리하여, 하드웨어 가속기에 의해 채용된 메모리 액세스 경로는 벡터-프로세서에 의해 채용된 경로와는 다를 수 있음을 인식할 것이다. 일부 실시예에 있어서, 적어도 하나의 하드웨어 가속기는 메모리 슬라이스에 액세스함에 있어서 지연을 처리하도록 내부 버퍼(예를 들어, FIFO 메모리)를 포함할 수 있다.

[0069] 일부 실시예에 있어서, 병렬 프로세싱 디바이스는 호스트 프로세서를 포함할 수 있다. 호스트 프로세서는 호스트 버스를 통하여 AMC와 통신하도록 구성될 수 있다. 병렬 프로세싱 디바이스에는 또한 애플리케이션-프로그래밍 인터페이스(API)가 제공될 수 있다. API는 벡터 프로세서 및/또는 하드웨어 가속기에 하이 레벨 인터페이스를 제공한다.

[0070] 일부 실시예에 있어서, 병렬 프로세싱 디바이스는 병렬 프로세싱 디바이스를 위한 명령어를 제공하는 컴파일러와 함께 동작할 수 있다. 일부 경우에 있어서, 컴파일러는 벡터 프로세서 또는 하드웨어 가속기와 같은 프로세싱 엘리먼트와는 구별되는 호스트 프로세서 상에서 실행되도록 구성된다. 일부 경우에 있어서, 컴파일러는 이미지 프로세싱 프로세스를 특정하는, 이미지/비디오 API(1206)(도 12)를 통하여, 흐름 그래프를 수신하도록 구성된다. 컴파일러는 벡터 프로세서 또는 하드웨어 가속기와 같은 프로세싱 엘리먼트 중 하나 이상에 흐름 그래프의 하나 이상의 태양을 매핑하도록 더 구성될 수 있다. 일부 실시예에 있어서, 흐름 그래프는 노드 및 에지를 포함할 수 있으며, 이 경우 각각의 노드는 연산을 식별시키고, 각각의 에지는, 연산이 수행되는 순서와 같은, 노드(예를 들어, 연산)들 간 관계를 식별시킨다. 컴파일러는 흐름 그래프의 컴퓨터 계산을 병렬화하기 위해 프로세싱 엘리먼트 중 하나에 노드(예를 들어, 연산)를 배정하도록 구성될 수 있다. 일부 실시예에 있어서, 흐름

그래프는 확장성 마크-업 언어(XML) 포맷으로 제공될 수 있다. 일부 실시예에 있어서, 컴파일러는 단일 프로세싱 엘리먼트에 다중 흐름 그래프를 할당하도록 구성될 수 있다.

[0071] 일부 실시예에 있어서, 병렬 프로세싱 디바이스는 그 성능을 측정하고 정보를 컴파일러에 제공하도록 구성될 수 있다. 그래서, 컴파일러는 병렬 프로세싱 디바이스 내 프로세싱 엘리먼트에 현재 태스크의 할당을 결정하도록 병렬 프로세싱 디바이스로부터 수신된 과거 성능 정보를 사용할 수 있다. 일부 실시예에 있어서, 성능 정보는 프로세싱 디바이스에서 하나 이상의 프로세싱 엘리먼트에 의해 경험된 소정 수의 액세스 충돌을 나타낼 수 있다.

[0072] 일부 경우에 있어서, 병렬 프로세싱 디바이스는 컴퓨터 계산 비용이 많이 들 수 있는 비디오 애플리케이션에서 사용될 수 있다. 비디오 애플리케이션의 컴퓨터 계산 요구를 다루기 위해, 병렬 프로세싱 디바이스는 메모리 액세스 동안 프로세싱 유닛들 간 액세스 충돌을 감축하도록 그 메모리 서브시스템을 구성할 수 있다. 이러한 목적을 위해, 앞서 논의된 바와 같이, 병렬 프로세싱 디바이스는, 모놀리식 메모리 뱅크를 메모리의 단일 물리적 블록으로서 사용하는 대신에, 모놀리식 메모리 뱅크를 다중의 물리적 RAM 인스턴스로 세분할 수 있다. 이러한 세분으로, 각각의 물리적 RAM 인스턴스는 판독 및 기록 연산에 대해 중재될 수 있고, 그로써 메모리 뱅크 내 물리적 RAM 인스턴스의 수만큼 이용가능한 대역폭을 증가시킬 수 있다.

[0073] 일부 실시예에 있어서, 하드웨어 사이클-지향 중재는 또한 다중 트래픽 클래스 및 프로그래밍 가능한 스케줄링 마스크를 제공할 수 있다. 다중 트래픽 클래스 및 프로그래밍 가능한 스케줄링 마스크는 런타임 스케줄러를 사용하여 제어될 수 있다. 하드웨어 사이클-지향 중재 블록은, 라운드-로빈 스케줄링 기법으로 단일의 공유 자원의 다중 요청기를 할당하도록 구성될 수 있는, 포트 중재 블록을 포함할 수 있다. 라운드-로빈 스케줄링 기법에 있어서, 요청기(예를 들어, 프로세싱 엘리먼트)는 요청이 요청기들로부터 수신된 순서로 자원(예를 들어, 메모리)으로의 액세스가 승인된다. 일부 경우에 있어서, 포트 중재 블록은 다중 트래픽 클래스를 처리하도록 라운드-로빈 스케줄링 기법을 증강시킬 수 있다. 단일의 공유 자원은 벡터-프로세서, 필터 가속기 및 RISC 프로세서가 데이터를 공유하도록 액세스할 수 있는 RAM-타일, 공유 레지스터 또는 다른 자원을 포함할 수 있다. 부가적으로, 중재 블록은 우선순위 벡터 또는 최-우선순위 벡터로 자원의 라운드-로빈 할당에 우선하는 것에 대해 허용할 수 있다. 우선순위 벡터 또는 최우선순위 벡터는 관심 있는 특정 애플리케이션에 의한 필요에 따라 소정 트래픽 클래스(예를 들어, 비디오 트래픽 클래스)를 우선시하기 위해 런타임 스케줄러에 의해 제공될 수 있다.

[0074] 일부 실시예에 있어서, 프로세싱 엘리먼트는 벡터 프로세서 또는 스트리밍 하이브리드 아키텍처 벡터 엔진 프로세서와 같은 프로세서, 하드웨어 가속기, 및 하드웨어 필터 연산기 중 하나 이상을 포함할 수 있다.

[0075] 도 4는, 일부 실시예에 따라 다중 프로세서(예를 들어, 스트리밍 하이브리드 아키텍처 벡터 엔진(SHAVE) 프로세서)가 멀티포트 메모리 서브시스템을 공유할 수 있게 하는, 메모리 서브시스템을 갖는 병렬 프로세싱 디바이스를 예시하고 있다. 구체적으로, 도 4는 이미지 및 비디오 데이터를 프로세싱하는데 적합한 병렬 프로세싱 디바이스(400)를 도시하고 있다. 프로세싱 디바이스(400)는 프로세서와 같은 복수의 프로세싱 엘리먼트(402)를 포함한다. 도 4의 예시적 구성에 있어서, 프로세싱 디바이스(400)는 8개의 프로세서(SHAVE 0(402-0) 내지 SHAVE 7(402-7))를 포함한다. 각각의 프로세서(402)는 데이터가 메모리(412)로부터 로딩되고 그에 저장될 수 있게 하는 2개의 로드 스토어 유닛(404, 406)(LSU0, LSU1)을 포함할 수 있다. 각각의 프로세서(402)는 또한 명령어가 로딩될 수 있는 명령어 유닛(408)을 포함할 수 있다. 프로세서가 SHAVE를 포함하는 특정 실시예에 있어서, SHAVE는 축소형 명령어 세트 컴퓨터(RISC), 디지털 신호 프로세서(DSP), 매우 긴 명령어 워드(VLIW), 및/또는 그래픽 프로세싱 유닛(GPU) 중 하나 이상을 포함할 수 있다. 메모리(412)는 여기에서는 접속 매트릭스(CMX) 슬라이스라고 지칭되는 복수의 메모리 슬라이스(412-0 ... 412-7)를 포함한다. 각각의 메모리 슬라이스(412-N)는 대응하는 프로세서(402-7)와 연관된다.

[0076] 병렬 프로세싱 디바이스(400)는 또한 프로세서(402)와 메모리 슬라이스(412)를 결합하는 상호접속 시스템(410)을 포함한다. 상호접속 시스템(410)은 여기에서는 SHAVE-간 상호접속부(ISI)라고 지칭된다. ISI는 프로세서(402)가 메모리 슬라이스(412) 중 어느 하나의 어느 부분에 데이터를 기록 또는 판독할 수 있게 통하는 버스를 포함할 수 있다.

[0077] 도 5는 일부 실시예에 따른 병렬 프로세싱 디바이스의 섹션을 예시하고 있다. 섹션(500)은 단일 프로세서(402-N), 단일 프로세서(402-N)와 연관된 메모리 슬라이스(412-N), 단일 프로세서(402-N)와 다른 메모리 슬라이스(도시되지 않음)를 결합하는 ISI(410), 및 메모리 슬라이스(412-N) 내 타일과 프로세서(402) 간 통신을 중재하기 위한 타일 제어 로직(506)을 포함한다. 섹션(500)에서 예시된 바와 같이, 프로세서(402-N)는 프로세서(402-N)와

연관된 메모리 슬라이스(412-N)에 직접 액세스하도록 구성될 수 있고; 프로세서(402-N)는 ISI를 통하여 다른 메모리 슬라이스(도시되지 않음)에 액세스할 수 있다.

[0078] 일부 실시예에 있어서, 각각의 메모리 슬라이스(412-N)는 복수의 RAM 타일 또는 물리적 RAM 블록(502-0 ... 502-N)을 포함할 수 있다. 예를 들면, 128kB의 사이즈를 갖는 메모리 슬라이스(412-N)는 4k x 32-비트 워드로서 조직된 4개의 32kB 단일-포트형 RAM 타일(예를 들어, 물리적 RAM 엘리먼트)을 포함할 수 있다. 일부 실시예에 있어서, 타일(502)은 논리적 RAM 블록이라고도 지칭될 수 있다. 일부 실시예에 있어서, 타일(502)은 단일 포트형 상보적 금속-산화막-반도체(CMOS) RAM을 포함할 수 있다. 단일 포트형 CMOS RAM의 이점은 그것이 대부분의 반도체 프로세스에서 일반적으로 이용가능하다는 것이다. 다른 실시예에 있어서, 타일(502)은 멀티-포트형 CMOS RAM을 포함할 수 있다.

[0079] 일부 실시예에 있어서, 각각의 타일(502)은 타일 제어 로직(506)과 연관될 수 있다. 타일 제어 로직(506)은 프로세서(402)로부터 요청을 수신하도록 구성되고 연관된 타일(502)의 개개의 판독 및 기록-포트로의 액세스를 제공한다. 예를 들어, 프로세싱 엘리먼트(402-N)가 RAM 타일(502-0) 내 데이터에 액세스하기를 원할 때, 프로세싱 엘리먼트(402-N)가 직접 RAM 타일(502-0)에 메모리 데이터 요청을 보내기 전에, 프로세싱 엘리먼트(402-N)는 RAM 타일(502-0)과 연관된 타일 제어 로직(506-0)에 메모리 액세스 요청을 보낼 수 있다. 메모리 액세스 요청은 프로세싱 엘리먼트(402-N)에 의해 요청된 데이터의 메모리 주소를 포함할 수 있다. 후속하여, 타일 제어 로직(506-0)은 메모리 액세스 요청을 분석하고 프로세싱 엘리먼트(402-N)가 요청된 메모리에 액세스할 수 있는지 결정할 수 있다. 프로세싱 엘리먼트(402-N)가 요청된 메모리에 액세스할 수 있으면, 타일 제어 로직(506-0)은 프로세싱 엘리먼트(402-N)에 액세스 승인 메시지를 보낼 수 있고, 후속하여, 프로세싱 엘리먼트(402-N)는 RAM 타일(502-0)에 메모리 데이터 요청을 보낼 수 있다.

[0080] 다중 프로세싱 엘리먼트에 의한 동시 액세스에 대한 잠재력이 있으므로, 일부 실시예에 있어서, 타일 제어 로직(506)은 프로세서 또는 가속기와 같은 2개 이상의 프로세싱 엘리먼트가 메모리 슬라이스 내 타일 중 어느 하나에 액세스하려고 시도하는 인스턴스를 검출하도록 구성되는 충돌 검출기를 포함할 수 있다. 충돌 검출기는 시도된 동시 액세스에 대한 각각의 타일(502)로의 액세스를 모니터링할 수 있다. 충돌 검출기는 액세스 충돌이 일어났고 해결될 필요가 있음을 런타임 스케줄러에 보고하도록 구성될 수 있다.

[0081] 도 6은 일부 실시예에 따른 타일 제어 로직 내 중앙집중형 충돌 검출 시스템을 예시하고 있다. 충돌 검출 시스템은 복수의 원-핫 주소 인코더(602) 및 복수의 충돌 검출기(604)를 포함하는 중앙집중형 중재 블록(608)을 포함할 수 있다. 일부 실시예에 있어서, 원-핫 주소 인코더(602)는 프로세싱 엘리먼트(402) 중 하나로부터 메모리 액세스 요청을 수신하도록 그리고 메모리 액세스 요청이 원-핫 주소 인코더(602)와 연관된 RAM 타일(502)에 저장된 데이터에 대한 것인지 결정하도록 구성된다. 각각의 충돌 검출기(604)는, 충돌 검출기(602)와 연관된 타일(502)에 액세스할 수 있는 프로세싱 엘리먼트(402) 중 하나에 또한 결합되는, 하나 이상의 원-핫 주소 인코더(602)에 결합될 수 있다. 일부 실시예에 있어서, 충돌 검출기(604)는 특정 RAM 타일(502)과 연관된 모든 원-핫 주소 인코더(602)에 결합될 수 있다.

[0082] 메모리 액세스 요청이 원-핫 주소 인코더(602)와 연관된 RAM 타일(502)에 저장된 데이터에 대한 것이면, 그때 원-핫 주소 인코더(602)는 특정 RAM 타일의 충돌 검출기(604)에 비트 값 "1"을 제공할 수 있고; 메모리 액세스 요청이 원-핫 주소 인코더(602)와 연관된 RAM 타일(502)에 저장된 데이터에 대한 것이 아니면, 그때 원-핫 주소 인코더(602)는 특정 RAM 타일의 충돌 검출기(604)에 비트 값 "0"을 제공할 수 있다.

[0083] 일부 실시예에 있어서, 원-핫 주소 인코더(602)는 메모리 액세스 요청의 표적 주소를 분석함으로써 메모리 액세스 요청이 원-핫 주소 인코더(602)와 연관된 RAM 타일(502)에 저장된 데이터에 대한 것인지 결정하도록 구성된다. 예를 들어, 원-핫 주소 인코더(602)와 연관된 RAM 타일(502)이 0x0000과 0x00ff의 메모리 주소 범위로 설계될 때, 그때 원-핫 주소 인코더(602)는 메모리 액세스 요청의 표적 주소가 0x0000과 0x00ff의 범위 내에 드는지 결정할 수 있다. 그러하다면, 메모리 액세스 요청은 원-핫 주소 인코더(602)와 연관된 RAM 타일(502)에 저장된 데이터에 대한 것이고; 그러하지 않다면, 메모리 액세스 요청은 원-핫 주소 인코더(602)와 연관된 RAM 타일(502)에 저장된 데이터에 대한 것이 아니다. 일부 경우에 있어서, 원-핫 주소 인코더(602)는 메모리 액세스 요청의 표적 주소가 RAM 타일(502)과 연관된 주소 범위 내에 드는지 결정하도록 범위 비교 블록을 사용할 수 있다.

[0084] 충돌 검출기(604)가 모든 원-핫 주소 인코더(602)로부터 비트 값을 수신하고 나면, 충돌 검출기(604)는 동일한 RAM 타일(502)로의 액세스를 현재 요청하고 있는 프로세싱 엘리먼트(402)가 하나보다 많은지 결정하도록 그 수신된 비트 값에서 "1"의 수를 카운팅(예를 들어, 비트 값을 합산)할 수 있다. 동일한 RAM 타일(502)로의 액세스

를 현재 요청하고 있는 프로세싱 엘리먼트가 하나보다 많으면, 충돌 검출기(604)는 충돌을 보고할 수 있다.

[0085] 도 7은 일부 실시예에 따른 타일 제어 로직 내 분산형 충돌 검출 시스템을 예시하고 있다. 분산형 충돌 검출 시스템은 복수의 충돌 검출기(704)를 포함하는 분상형 중재기(702)를 포함할 수 있다. 분산형 충돌 검출 시스템의 동작은 실질적으로는 중앙집중형 충돌 검출 시스템의 동작과 유사하다. 이러한 경우에 있어서, 충돌 검출기(704)는 분산된 방식으로 배열된다. 특히, 분산형 중재기(702)는 직렬 방식으로 배열되는 충돌 검출기(704)를 포함할 수 있고, 이 경우 각각의 충돌 검출기(704)는 특정 RAM 타일(502)과 연관된 원-핫 주소 인코더(602)의 서브세트에만 결합된다. 이러한 배열은 충돌 검출기(704)가 특정 RAM 타일(502)과 연관된 모든 원-핫 주소 인코더(602)에 결합되는 중앙집중형 충돌 검출 시스템과 다르다.

[0086] 예를 들어, 특정 RAM 타일(502)이 64개의 프로세싱 엘리먼트(402)에 의해 액세스될 수 있을 때, 제1 충돌 검출기(704-0)는 32개의 프로세싱 엘리먼트로부터 메모리 액세스 요청을 수신할 수 있고 제2 충돌 검출기(704-1)는 잔여 32개의 프로세싱 엘리먼트로부터 메모리 액세스 요청을 수신할 수 있다. 제1 충돌 검출기(704-0)는 그 자신에 결합된 32개의 프로세싱 엘리먼트로부터의 하나 이상의 메모리 액세스 요청을 분석하고, 그 자신에 결합된 32개의 프로세싱 엘리먼트 중, 특정 RAM 타일(502-0)로의 액세스를 요청하고 있는 엘리먼트의 제1 수를 결정하도록 구성될 수 있다. 병렬로, 제2 충돌 검출기(704-1)는 그 자신에 결합된 32개의 프로세싱 엘리먼트로부터의 하나 이상의 메모리 액세스 요청을 분석하고, 그 자신에 결합된 32개의 프로세싱 엘리먼트 중, 특정 RAM 타일(502-0)로의 액세스를 요청하고 있는 엘리먼트의 제2 수를 결정하도록 구성될 수 있다. 그 후 제2 충돌 검출기(704)는 64개의 프로세싱 엘리먼트 중 얼마나 많은 것이 특정 RAM 타일(502-0)로의 액세스를 요청하고 있는지 결정하도록 제1 수와 제2 수를 가산할 수 있다.

[0087] 충돌 검출 시스템이 충돌을 검출하고 나면, 충돌 검출 시스템은 요청기(402)에 정지 신호를 보낼 수 있다. 도 8은 일부 실시예에 따라 요청기에 충돌 신호를 보고하기 위한 중재 블록을 도시하고 있다. 더 구체적으로, 충돌 검출 시스템에서의 범위-비교 블록의 출력은 요청기에 정지 신호를 발생시키도록 OR 게이트를 사용하여 조합된다. 정지 신호는 하나보다 많은 프로세싱 엘리먼트가 요청기와 연관된 메모리 슬라이스 내 동일한 물리적 RAM 서브-블록에 액세스하려고 시도하고 있음을 나타낸다. 정지 신호 수신시, 요청기는 충돌이 치워질 때까지 메모리 액세스 연산을 정지할 수 있다. 일부 실시예에 있어서, 충돌은 프로그램 코드와는 독립적으로 하드웨어에 의해 치워질 수 있다.

[0088] 일부 실시예에 있어서, 중재 블록은 사이클 입도로 동작할 수 있다. 그러한 실시예에 있어서, 중재 블록은 다중 클록 사이클을 포함할 수 있는 태스크 레벨 입도로보다는 클록 사이클 입도로 자원을 할당한다. 그러한 사이클-지향 스케줄링은 시스템의 성능을 개선할 수 있다. 중재 블록은 중재 블록이 실시간으로 사이클-지향 스케줄링을 수행할 수 있도록 하드웨어에서 구현될 수 있다. 예를 들어, 어느 특정 인스턴스에서, 하드웨어에서 구현된 중재 블록은 다음 클록 사이클에 대한 자원을 할당하도록 구성될 수 있다.

[0089] 도 9는 일부 실시예에 따른 사이클-지향 중재 블록을 예시하고 있다. 사이클-지향 중재 블록은 포트 중재 블록(900)을 포함할 수 있다. 포트 중재 블록(900)은 제1 포트 선택 블록(930) 및 제2 포트 선택 블록(932)을 포함할 수 있다. 제1 포트 선택 블록(930)은 슬라이스 포트[0]에 결합된 메모리 슬라이스에 액세스하도록 (클라이언트 요청 벡터에서 비트 위치로서 식별된) 메모리 액세스 요청 중 어느 것이 슬라이스 포트[0]에 배정되는지 결정하도록 구성되는 반면, 제2 선택 블록(932)은 슬라이스 포트[1]에 결합된 메모리 슬라이스에 액세스하도록 클라이언트 요청 벡터 중 어느 것이 슬라이스 포트[1]에 배정되는지 결정하도록 구성된다.

[0090] 제1 포트 선택 블록(930)은 제1 리딩 원 검출기(LOD)(902-0) 및 제2 LOD(902-1)를 포함한다. 제1 LOD(902-0)는, 복수의 비트를 포함할 수 있는, 클라이언트 요청 벡터를 수신하도록 구성된다. 클라이언트 요청 벡터에서의 각각의 비트는 메시지 액세스 요청이 그 비트 위치와 연관된 요청기로부터 수신되었는지 여부를 나타낸다. 일부 경우에 있어서, 클라이언트 요청 벡터는 "액티브 하이" 모드로 동작한다. 제1 LOD(902-0)가 클라이언트 요청 벡터를 수신하고 나면, 제1 LOD(902-0)는, 좌측으로부터 우측으로 카운팅하여, 요청이 처음으로 영이 아니게 되는 비트 위치를 검출하고, 그로써, 좌측으로부터 우측으로 카운팅하여, 제1 포트 선택 블록(930)에 제1 메모리 액세스 요청을 식별시키도록 구성된다. 병렬로, 클라이언트 요청 벡터는 마스크 레지스터(906) 및 마스크 좌측 시프터(904)에 의해 발생된 마스크를 사용하여 마스킹된 클라이언트 요청 벡터를 발생시키도록 AND 로직 연산기(912)에 의해 마스킹될 수 있다. 마스크 레지스터(906)는 마스크 레지스터(906)와 통신하고 있는 프로세서에 의해 설정될 수 있고, 마스크 좌측 시프터(904)는 마스크 레지스터(906)에 의해 표현된 마스크를 좌측으로 시프트하도록 구성될 수 있다. 제2 LOD(902-1)는 AND 로직 연산기(912)로부터 마스킹된 클라이언트 요청 벡터를 수신하고, 마스킹된 클라이언트 요청 벡터에서 리딩 1을 검출할 수 있다.

[0091] 제1 LOD(902-0) 및 제2 LOD(902-1)로부터의 출력은 그 후 포트[0] 승자 선택 블록(908)에 제공된다. 포트[0] 승자 선택 블록(908)은 2개의 부가적 입력: 우선순위 벡터 및 최-우선순위 벡터를 더 수신한다. 포트[0] 승자 선택 블록(908)은, 입력의 우선순위에 기반하여, 수신된 메모리 액세스 요청 중 어느 것이 슬라이스 포트[0]에 배정되어야 하는지 결정하도록 구성된다. 일부 실시예에 있어서, 입력의 우선순위는 다음과 같이 랭킹될 수 있다: 가장 높은 우선순위를 갖는 최-우선순위 벡터로 시작하여, 마스킹된 LOD 벡터를 우선순위 요청과 비-우선순위 요청으로 분할하는 우선순위 벡터, 그 다음에 가장 낮은 우선순위를 갖는 마스킹되지 않은 LOD 벡터. 다른 실시예에서는, 다른 우선순위가 특정될 수 있다.

[0092] 제1 포트 선택 블록(930)은 클라이언트 요청 벡터가 슬라이스 포트[0]에 배정될 수 있는지 결정하도록 구성될 수 있는 한편, 제2 포트 선택 블록(932)은 클라이언트 요청 벡터가 슬라이스 포트[1]에 배정될 수 있는지 결정하도록 구성될 수 있다. 제2 포트 선택 블록(932)은 제1 트레일링 원 검출기(TOD)(912-0), 제2 TOD(912-1), 마스크 레지스터(914), 마스크 우측 시프터(916), 포트[1] 승자 선택 블록(918), 및 마스킹 AND 로직 블록(920)을 포함한다. TOD(912)는 복수의 비트를 포함할 수 있는 클라이언트 요청 벡터를 수신하고, 우측으로부터 좌측으로 카운팅하여, 벡터가 처음으로 영이 아니게 되는 비트 위치를 검출하도록 구성된다. 제2 포트 선택 블록(932)의 동작은 그것이 입력 벡터의 우측으로부터 좌측으로 동작하여 트레일링-원 검출기(912-0)를 사용하여 입력 요청 벡터에서 트레일링 원을 선택한다는 것을 제외하고는 제1 포트 선택 블록(930)과 실질적으로 유사하다.

[0093] 포트 승자 선택 블록(908, 918)의 출력은 또한, 동일한 메모리 액세스 요청이 슬라이스 포트[0]와 슬라이스 포트[1] 양자로의 액세스를 획득하였는지 결정하도록 구성되는, 동일한 승자 검출 블록(910)에 제공된다. 동일한 클라이언트 요청 벡터가 슬라이스 포트[0]와 슬라이스 포트[1] 양자로의 액세스를 획득하였으면, 동일한 승자 검출 블록(910)은 요청을 라우팅하도록 슬라이스 포트 중 하나를 선택하고 다른 포트를 입력 벡터에서의 다음의 가장 높은 랭킹 요청에 할당한다. 이것은 특정 요청에 대한 자원의 과잉-할당을 회피하고, 그로써 경쟁하는 요청에 대한 자원의 할당을 개선한다.

[0094] 포트 중재 블록(900)의 동작은 32-비트 클라이언트 요청 벡터의 좌변으로부터 시작함으로써 작업하고 마스킹된 LOD(902-1)는 제1 마스킹된 요청 벡터의 위치를, 이러한 마스킹된 요청 벡터가 LOD 위치에 대응하는 요청기가 획득하는 우선순위 또는 최-우선순위 벡터를 통하여 더 높은 우선순위 입력에 의해 대체되지 않고 포트[0]로의 액세스를 승인받으면, 출력한다. LOD 위치는 또한 32-비트 좌측-시프터(904)를 통하여 마스크 위치를 전진시키는데 사용되고 그리고 또한 동일한 요청기가 양 포트로의 액세스를 부여받았는지 체크하도록 포트 1 LOD 배정과 비교하는데 사용되고 이러한 경우에 있어서 연속하는 동일한 승자 검출의 경우에 포트 0과 1 간 교번 기반으로 액세스를 승인하도록 토글링되는 플립-플롭으로 포트 중 하나만이 승인된다. 마스킹된 검출기(902-1)로부터 출력된 LOD에 우선순위 벡터에서의 대응하는 일 비트를 통하여 우선순위가 배정된 경우에, 요청하는 클라이언트는 포트 0으로의 2 백-투-백 사이클 액세스를 승인받는다. 마스킹된 클라이언트 요청 벡터에서 리딩 원이 없고 더 높은 우선순위 요청이 존재하지 않는 경우에, 마스킹되지 않은 LOD가 포트 0로의 액세스를 획득하고 배정받는다. 위 경우들 중 어느 경우에서라도, 최-우선순위 벡터에서의 1-비트는 이전 요청 중 어느 것보다도 우선하고 포트 0로의 무제한 액세스를 요청기에 승인할 것이다.

[0095] 선도의 하위 부분에서의 로직은 요청 벡터의 우변으로부터 시작하고 그밖에는 요청 벡터의 좌변으로부터 시작하는 상위 부분과 동일한 방식으로 동작한다. 이러한 경우에 있어서, 우선순위 등의 관점에서 포트 1 중재 블록의 동작은 로직의 포트 0 부분의 동작과 똑같다.

[0096] 일부 실시예에 있어서, 프로세싱 엘리먼트(402)는 메모리 액세스 중재에 기인하는 메모리 액세스의 레이턴시를 감축하도록 버퍼를 포함할 수 있다. 도 10은 일부 실시예에 따라 메모리 액세스 중재에 기인하는 메모리 액세스 레이턴시를 감축하기 위한 메커니즘을 예시하고 있다. 전형적 메모리 액세스 중재 기법에 있어서, 메모리 액세스 중재 블록은 파이프라인형이어서, RAM 타일(502)과 같은 공유 자원을 복수의 프로세싱 엘리먼트(예를 들어, 요청기) 중 하나에 할당할 때 고정된 오버헤드 중재 페널티를 초래한다. 예를 들어, 요청기(402)가 중재 블록(608/702)에 메모리 액세스 요청을 보낼 때, 이하의 단계의 각각에서 적어도 하나의 사이클이 걸리기 때문에 요청기(402)가 액세스 승인 메시지를 수신하는데 적어도 4개의 사이클이 걸린다: (1) 원-핫 주소 인코더(602)에서 메모리 액세스 요청을 분석한다, (2) 중재 블록(608/702)에서 원-핫 주소 인코더(602)의 출력을 분석한다, (3) 중재 블록(608/702)에 의해 원-핫 주소 인코더(602)에 액세스 승인 메시지를 보낸다, 그리고 (4) 원-핫 주소 인코더(602)에 의해 요청기(402)에 액세스 승인 메시지를 보낸다. 그 후, 후속하여, 요청기(402)는 RAM 타일(502)에 메모리 데이터 요청을 보내고 RAM 타일(502)로부터 데이터를 수신하여야 하는데, 그 각각은 적어도 하나의 사이클이 걸린다. 그래서, 메모리 액세스 연산은 적어도 6개의 사이클의 레이턴시를 갖는다. 이러한 고정된 페

널티는 메모리 서브시스템의 대역폭을 감축할 것이다.

[0097] 이러한 레이턴시 문제는 프로세싱 엘리먼트(402)에 유지된 메모리 액세스 요청 버퍼(1002)로 다뤄질 수 있다. 예를 들어, 메모리 액세스 요청 버퍼(1002)는 클록 사이클마다 프로세싱 엘리먼트로부터 메모리 액세스 요청을 수신하고, 수신된 메모리 액세스 요청을 그것들이 메모리 중재 블록(608/702)에 보내질 준비가 될 때까지 저장할 수 있다. 버퍼(1002)는 메모리 액세스 요청이 메모리 중재 블록(608/702)에 보내지는 레이트와 데이터가 메모리 서브시스템으로부터 수신되는 레이트를 사실상 동기화한다. 일부 실시예에 있어서, 버퍼는 큐를 포함할 수 있다. 버퍼(1002)에서의 엘리먼트의 수(예를 들어, 버퍼의 깊이)는 메모리 서브시스템으로부터 데이터를 검색하기 위한 사이클의 수보다 더 클 수 있다. 예를 들어, RAM 액세스 레이턴시가 6개의 사이클일 때, 버퍼(1002)에서의 엘리먼트의 수는 10일 수 있다. 버퍼(1002)는 중재 레이턴시 페널티를 감축하고 메모리 서브시스템의 처리율을 개선할 수 있다. 메모리 액세스 요청 버퍼로는, 원칙적으로, 총 메모리 대역폭의 100%까지 요청기 간에 할당될 수 있다.

[0098] 다중 RAM 인스턴스를 사용하는 것과의 잠재적 문제는, 뱅크 내 서브-인스턴스로의 다중 프로세싱 엘리먼트에 의한 동시 액세스를 허용함으로써, 메모리 경합이 초래될 수 있다는 것임을 이해할 것이다.

[0099] 본 발명은 메모리 경합을 다루도록 적어도 2개의 접근법을 제공한다. 첫째로, 추후 설명될 바와 같이, 경합 및/또는 메모리 충돌을 감축하도록 메모리 서브시스템에서 데이터를 주의깊게 레이아웃함으로써 메모리 경합 및/또는 메모리 충돌을 회피하도록 소프트웨어 설계에 주의한다. 더욱, 병렬 프로세싱 디바이스와 연관된 소프트웨어 개발 툴은 메모리 경합 또는 메모리 충돌이 소프트웨어 설계 단계 동안 보고될 수 있게 할 수 있다. 그래서, 메모리 경합 문제 또는 메모리 충돌 문제는 소프트웨어 설계 단계 동안 보고된 메모리 경합 또는 메모리 충돌에 응답하여 데이터 레이아웃을 개선함으로써 정정될 수 있다.

[0100] 둘째로, 아래에서 더 설명되는 바와 같이, 아키텍처 내 ISI 블록은 하드웨어에서의 포트-충돌(경합)을 검출하고 더 낮은 우선순위를 갖는 프로세싱 엘리먼트를 멎게 하도록 구성된다. 예를 들어, ISI 블록은 프로세싱 엘리먼트로부터의 메모리 액세스 요청을 분석하고, 메모리 액세스 요청의 시퀀스에 서비싱하고, 모든 프로세싱 엘리먼트로부터의 모든 데이터 판독 또는 기록이 우선순위 순서로 완료되도록 우선순위 순서에 따라 메모리 액세스 요청을 라우팅하도록 구성된다.

[0101] 프로세싱 엘리먼트 간 우선순위 순서는 여러 방식으로 확립될 수 있다. 일부 실시예에 있어서, 우선순위 순서는 시스템 설계 시간에서 정적으로 정의될 수 있다. 예를 들어, 우선순위 순서는 시스템 파워업 때 시스템이 사전-배정된 우선순위 세트로 파워업하도록 시스템 레지스터에 대한 재설정 상태로서 코딩될 수 있다. 다른 실시예에 있어서, 우선순위 순서는 사용자-프로그래밍 가능한 레지스터를 통하여 동적으로 결정될 수 있다.

[0102] 일부 실시예에 있어서, 프로그래머는 메모리 슬라이스 내 메모리의 공유 서브-블록에 대한 경합을 감축하기 위해 그들 소프트웨어 애플리케이션에 대한 데이터-레이아웃을 계획할 수 있다. 일부 경우에 있어서, 데이터-레이아웃의 계획은 중재 블록에 의해 조력받을 수 있다. 예를 들어, 중재 블록은 메모리 경합을 검출하고, 가장 높은 우선순위 태스크와 연관된 프로세싱 엘리먼트에, 우선순위에 기반하여, 메모리로의 액세스를 승인하고, 경합하고 있는 다른 프로세싱 엘리먼트를 멎게 하고, 경합이 해결될 때까지 프로세스마다 일일이 경합을 풀 수 있다.

[0103] 도 11은 일부 실시예에 따른 스케줄링 소프트웨어의 애플리케이션을 예시하고 있다. 이러한 애플리케이션에 있어서, 스케줄링 소프트웨어는 프로세싱 파이프라인 내 3x3 블러 커널의 구현을 조정할 수 있다. 스케줄링 소프트웨어는, 런타임에서, 연산의 순서화를 결정하고 프로세싱 엘리먼트에 의한 연산을 조정할 수 있다. 파이프라인에 대한 흐름-그래프(1100)는 엘리먼트1 내지 엘리먼트5(1102 내지 1110)를 포함한다. 엘리먼트1(1102)는 입력 버퍼(1112), 프로세싱 블록(1144) 및 출력 버퍼(1114)를 포함할 수 있다. 입력 버퍼(1112) 및 출력 버퍼(1114)는 플립-플롭을 사용하여 구현될 수 있다. 일부 실시예에 있어서, 다른 엘리먼트(1104 내지 1110)의 각각은 엘리먼트1(1102)와 실질적으로 유사한 구조를 가질 수 있다.

[0104] 일부 실시예에 있어서, 엘리먼트2(1104)는 3x3 블러 필터로 입력을 필터링할 수 있는 프로세싱 엘리먼트(예를 들어, 벡터 프로세서 또는 하드웨어 가속기)를 포함할 수 있다. 엘리먼트2(1104)는, 엘리먼트1(1102)의 출력을 일시적으로 유지하는, 공유 버퍼(1118)로부터 입력을 수신하도록 구성될 수 있다. 3x3 블러 커널을 입력에 적용하기 위해, 엘리먼트2(1104)는 그것이 연산을 개시할 수 있기 전에 공유 입력 버퍼(1118)로부터 데이터의 적어도 3개 라인을 수신할 수 있다. 그리하여, RISC 프로세서(1122) 상에서 실행될 수 있는 SW 스케줄러(1120)는 그것이 필터링 연산을 개시할 수 있음을 엘리먼트2(1104)에 신호보내기 전에 공유 버퍼(1118)에 데이터의 라인의

올바른 수가 포함되어 있음을 검출할 수 있다.

[0105] 데이터의 3개 라인이 존재한다는 초기 신호에 뒤이어, SW 스케줄러(1120)는 각각의 부가적 새로운 라인이 롤링 3-라인 버퍼(1118)에 부가되었을 때를 엘리먼트2(1104)에 신호보내도록 구성될 수 있다. 라인-대-라인 동기화에 부가하여, 사이클-대-사이클 중재 및 동기화가 파이프라인에서의 각각의 엘리먼트에 대해 수행된다. 예를 들면, 엘리먼트1(1102)는 사이클당 하나의 완전 출력 픽셀을 산출하는 하드웨어 가속기를 포함할 수 있다. 이러한 처리율을 달성하기 위해, 하드웨어 가속기는 입력 버퍼(1112)를 가득 차게 유지할 수 있고 그래서 프로세싱 블록(1114)은 그 연산을 계속하기에 충분한 데이터를 갖는다. 이러한 식으로, 프로세싱 블록(1114)은 엘리먼트(1102)의 처리율을 가능한 높게 유지하기에 충분한 출력을 산출할 수 있다.

[0106] 일부 실시예에 있어서, 소프트웨어 툴 체인은 메모리 서브시스템을 사용하는 소프트웨어 프로그램을 분석하는 것으로부터 메모리 상충을 예측할 수 있다. 소프트웨어 툴 체인은 개발자가 코드를 편집하고, 컴파일러, 어셈블러를 호출하고, 필요할 때 소스-레벨 디버깅을 수행할 수 있는 그래픽 사용자 인터페이스(GUI)-기반 통합 개발 환경(IDE)(예를 들어, 이클립스(Eclipse)-기반 IDE)을 포함할 수 있다. 소프트웨어 툴 체인은, 모든 프로세싱, 버스, 메모리 엘리먼트 및 주변장치를 모델링하는, 시스템 시뮬레이터를 사용하여 다중 프로세서 상에서 실행중인 프로그램의 동적 분석을 통하여 메모리 상충을 예측하도록 구성될 수 있다. 또한, 소프트웨어 툴 체인은, 여러 다른 프로세서 또는 하드웨어 자원 상에서 실행 중인 여러 다른 프로그램이 메모리 슬라이스의 특정 블록으로의 동시발생적 액세스를 시도하면, 로그 파일 또는 디스플레이 디바이스에 로깅하도록 구성될 수 있다. 소프트웨어 툴 체인은 사이클-대-사이클 기반으로 로깅하도록 구성될 수 있다.

[0107] 일부 실시예에 있어서, 파이프라인(1100)은 메모리 충돌이 일어날 때마다 증분되는 하나 이상의 하드웨어 카운터(예를 들어, 각각의 메모리 인스턴스에 대해 하나의 카운터)를 또한 포함할 수 있다. 그때 이들 카운터는 하드웨어 디버거(예를 들어, JTAG)에 의해 판독되고 스크린 상에 디스플레이되거나 파일에 로깅될 수 있다. 시스템 프로그래머에 의한 로그 파일의 후속 분석은 메모리 포트 충돌 가능성을 감축하도록 메모리 액세스가 서로 다르게 스케줄링될 수 있게 할 수 있다.

[0108] (도 2에 예시된) IBM의 셀 아키텍처의 프로그래머에 대한 하나의 핵심적 어려움은 벡터 프로세서가 데이터에 액세스할 때까지 데이터가 DMA에 의해 제어되어 로컬 저장소(LS)에 저장될 수 있도록 미리 수백 사이클 데이터-전송을 프로그램으로 스케줄링하는 것이다. 개시된 아키텍처의 일부 실시예는 하드웨어에서 액세스의 중재 및 스케줄링을 취급하고 사용자-판독가능한 하드웨어 카운터에서 충돌을 로깅함으로써 이러한 문제를 다룰 수 있다. 이것은 개시된 아키텍처가 고성능 비디오/이미지 프로세싱 파이프라인을 생성하는데 사용될 수 있게 한다.

[0109] 도 12는 일부 실시예에 따라 병렬 프로세싱 디바이스를 갖는 시스템의 계층적 구조를 제공하고 있다. 시스템(1200)은, 필터와 같은, 복수의 프로세싱 엘리먼트를 갖는 병렬 컴퓨팅 시스템(1202), 및 병렬 컴퓨팅 시스템(1204) 상에서 실행되는 소프트웨어 애플리케이션(1204), 병렬 컴퓨팅 시스템(1202)과 애플리케이션(1204)을 인터페이싱하기 위한 애플리케이션 프로그래밍 인터페이스(API)(1206), 병렬 컴퓨팅 시스템(1202) 상에서 실행되도록 소프트웨어 애플리케이션(1204)을 컴파일링하는 컴파일러(1208), 및 병렬 컴퓨팅 시스템(1202)에서의 프로세싱 엘리먼트의 연산을 제어하기 위한 런타임 스케줄러(1210)를 포함할 수 있다.

[0110] 일부 실시예에 있어서, 개시된 병렬 프로세싱 디바이스는, 이미지-프로세싱 파이프라인이 흐름-그래프로서 기술될 수 있게 하는, 파이프라인 기술 툴(예를 들어, 소프트웨어 애플리케이션)(1204)과 함께 동작하도록 구성될 수 있다. 파이프라인 기술 툴(1204)은 기저 하드웨어/소프트웨어 플랫폼과는 독립적인 융통성 있는 방식으로 이미지/비전 프로세싱 파이프라인을 기술할 수 있다. 특히, 파이프라인 기술 툴에 의해 사용된 흐름-그래프는 흐름-그래프를 구현하는데 사용될 수 있는 프로세싱 엘리먼트(예를 들어, 프로세서 및 필터 가속기 자원)와는 독립적으로 태스크가 기술될 수 있게 한다. 파이프라인 기술 툴의 결과적 출력은 방향성 비사이클 그래프(DAG) 또는 흐름 그래프의 기술을 포함할 수 있다. DAG 또는 흐름 그래프의 기술은 XML과 같은 적합한 포맷으로 저장될 수 있다.

[0111] 일부 실시예에 있어서, DAG 또는 흐름 그래프의 기술은 시스템(1200)에서의 모든 다른 툴이 액세스가능할 수 있고, DAG에 따라 병렬 프로세싱 디바이스의 연산을 제어하는데 사용될 수 있다. 도 13은 일부 실시예에 따라 DAG 또는 흐름 그래프의 기술이 어떻게 병렬 프로세싱 디바이스의 연산을 제어하는데 사용될 수 있는지 예시하고 있다.

[0112] 컴퓨팅 디바이스의 실제 연산 이전에, 병렬 프로세싱 디바이스(1202)에 대한 컴파일러(1208)는 (1) 흐름-그래프의 기술(1306) 및 (2) 이용가능한 자원의 기술(1302)을 취하고, DAG가 다중 프로세싱 엘리먼트에 걸쳐 어떻게

수행될 수 있는지 나타내는 태스크 리스트(1304)를 발생시킬 수 있다. 예를 들어, 태스크가 단일 프로세싱 엘리먼트 상에서 수행될 수 없을 때, 컴파일러(1208)는 다중 프로세싱 엘리먼트에 걸쳐 태스크를 분할할 수 있고; 태스크가 단일 프로세싱 엘리먼트 상에서 수행될 수 있을 때, 컴파일러(1208)는 태스크를 단일 프로세싱 엘리먼트에 배정할 수 있다.

[0113] 일부 경우에 있어서, 태스크가 프로세싱 엘리먼트의 능력의 일부만을 사용할 것일 때, 컴파일러(1208)는, 프로세싱 엘리먼트에 의해 지원될 수 있는 한도까지, 순차적 방식으로 단일 프로세싱 엘리먼트 상에서 실행될 다중 태스크를 융합 및 스케줄링할 수 있다. 도 14a는 일부 실시예에 따라 컴파일러 및 스케줄러에 의한 태스크의 스케줄링 및 발행을 예시하고 있다. 컴파일러 및 스케줄러를 사용하여 태스크를 스케줄링하는 이점은 컴파일러 및 스케줄러가 태스크에 의해 수행되는 연산에 기반하여 태스크를 자동으로 스케줄링할 수 있다는 것이다. 이것은 프로그래머가, 주변장치로부터 CMX로, CMX로부터 CMX 블록으로 그리고 CMX로부터 다시 주변장치로 DMA에 의한 데이터 전송을 언제 스케줄링할지를 포함하여, 특정 태스크를 실행하는 프로세싱 엘리먼트 또는 프로세싱 엘리먼트 그룹 상에서 실행 중인 코드에 대한 스케줄을 수동으로 결정해야 했었던 종래 기술에 비해 큰 이점이다. 이것은 고되고 오류나기 쉬운 태스크였고 DFG의 사용은 이러한 프로세스가 자동화될 수 있게 하여 시간을 절약하고 생산성을 증가시킨다.

[0114] 컴퓨팅 디바이스의 런타임 동안, 런타임 스케줄러(1210)는 컴파일러(1208)에 의해 발생된 태스크 리스트(1304)에 기반하여 이용가능한 프로세싱 엘리먼트에 걸쳐 태스크를 동적으로 스케줄링할 수 있다. 런타임 스케줄러(1210)는 멀티코어 시스템에서 호스트 RISC 프로세서(1306) 상에서 동작할 수 있고, 하드웨어 성능 모니터 및 타이머(1308)로부터의 통계를 사용하여, 복수의 벡터 프로세서, 필터 가속기 및 직접 메모리 액세스(DMA) 엔진과 같은 프로세싱 엘리먼트에 걸쳐 태스크를 스케줄링할 수 있다. 일부 실시예에 있어서, 하드웨어 성능 모니터 및 타이머(1308)는, 런타임 스케줄러(1210)에 의해 판독될 수 있는, 멎음 카운터, CMX-충돌 카운터, 버스 사이클-카운터(ISI, APB 및 AXI) 밍 사이클-카운터를 포함할 수 있다.

[0115] 일부 실시예에 있어서, 런타임 스케줄러(1210)는 하드웨어 성능 모니터 및 타이머(1308)로부터의 통계에 기반하여 이용가능한 프로세싱 엘리먼트에 태스크를 배정할 수 있다. 하드웨어 성능 모니터 및 타이머(1308)는 프로세싱 엘리먼트의 효율을 증가시키도록, 또는 전력을 절약하거나 다른 태스크가 병렬로 컴퓨팅될 수 있게 하기 위해 더 적은 수의 프로세싱 엘리먼트를 사용하여 태스크를 수행하도록 사용될 수 있다.

[0116] 이러한 목적으로, 하드웨어 성능 모니터 및 타이머(1308)는 성능 메트릭을 제공할 수 있다. 성능 메트릭은 프로세싱 엘리먼트의 활동 레벨을 나타내는 수일 수 있다. 성능 메트릭은 태스크를 수행하도록 인스턴스 생성된 프로세싱 엘리먼트의 수를 제어하는데 사용될 수 있다. 예컨대, 특정 프로세싱 엘리먼트와 연관된 성능 메트릭이 사전 결정된 임계치보다 더 클 때, 그때 런타임 스케줄러(1210)는 특정 프로세싱 엘리먼트와 동일한 유형의 부가적 프로세싱 엘리먼트를 인스턴스 생성하여, 그로써 더 많은 프로세싱 엘리먼트에 걸쳐 태스크를 분산시킬 수 있다. 다른 일례로서, 특정 프로세싱 엘리먼트와 연관된 성능 메트릭이 사전 결정된 임계치보다 더 작을 때, 그때 런타임 스케줄러(1210)는 특정 프로세싱 엘리먼트와 동일한 유형의 인스턴스 생성된 프로세싱 엘리먼트 중 하나를 제거하여, 소정 태스크를 수행하는 프로세싱 엘리먼트의 수를 감축할 수 있다.

[0117] 일부 실시예에 있어서, 런타임 스케줄러(1210)는 프로세싱 엘리먼트의 사용의 우선순위를 결정할 수 있다. 예를 들어, 런타임 스케줄러(1210)는 태스크가 프로세서에 배정되어야 하는 것이 바람직한지 하드웨어 필터 가속기에 배정되어야 하는 것이 바람직한지 결정하도록 구성될 수 있다.

[0118] 일부 실시예에 있어서, 런타임 스케줄러(1210)는 시스템이 런타임 구성 기준을 준수할 수 있도록 메모리 서브시스템에서의 CMX 버퍼 레이아웃을 변경하도록 구성될 수 있다. 런타임 구성 기준은, 예를 들어, 이미지 프로세싱 처리율(초당 프레임), 에너지 소모, 시스템에 의해 사용되는 메모리의 양, 동작하는 프로세서의 수, 및/또는 동작하는 필터 가속기의 수를 포함할 수 있다.

[0119] 출력 버퍼는 수개의 방식 중 하나로 메모리에서 레이아웃될 수 있다. 일부 경우에 있어서, 출력 버퍼는 메모리에서 물리적으로 연속할 수 있다. 다른 경우에 있어서, 출력 버퍼는 "청크" 또는 "슬라이스"될 수 있다. 예를 들어, 출력 버퍼는 N개의 수직 스트립으로 분할될 수 있으며, 여기서 N은 이미지 프로세싱 애플리케이션에 배정된 프로세서의 수이다. 각각의 스트립은 서로 다른 CMX 슬라이스에 위치하고 있다. 이러한 레이아웃은, 각각의 프로세서가 입력 및 출력 버퍼에 로컬 액세스할 수 있으므로, 프로세서에 유리할 수 있다. 그렇지만, 이러한 레이아웃은, 그러한 레이아웃이 필터 가속기에 대한 많은 충돌을 야기할 수 있기 때문에, 필터 가속기에는 유해할 수 있다. 필터 가속기는 흔히 좌측으로부터 우측으로 데이터를 프로세싱한다. 그래서, 모든 필터 가속기는 이미지의 처음 스트립에 액세스함으로써 그들 프로세스를 개시할 것이어서, 시작부터 많은 충돌을 야기할 수 있다.

다른 경우에 있어서, 출력 버퍼는 인터리빙될 수 있다. 예를 들어, 출력 버퍼는, 사전 결정된 사이즈 인터리빙으로, 모든 16개의 CMX 슬라이스에 걸쳐 분할될 수 있다. 사전 결정된 사이즈는 128 비트일 수 있다. 출력 버퍼의 인터리빙된 레이아웃은 CMX에 걸쳐 액세스를 분산시키는 것이 충돌의 가능성을 감축하기 때문에 필터 가속기에 유리할 수 있다.

[0120] 일부 실시예에 있어서, 입력 버퍼 또는 출력 버퍼와 같은 버퍼는 그 생산기 및 소비기가 하드웨어인지 그리고/또는 소프트웨어인지에 기반하여 할당될 수 있다. 소비기는, 그것들이 전형적으로는 더 많은 대역폭을 필요로 하므로, 더 중요하다(필터는 보통 다수의 라인을 판독하고 하나의 라인을 출력한다). 하드웨어 필터는 버퍼의 레이아웃에 따라 프로그래밍된다(그것들은 연속하는, 인터리빙된 그리고 슬라이스된 메모리 주소지정을 지원한다).

[0121] 도 14b는 일부 실시예에 따라 컴파일러 및 스케줄러를 사용하여 태스크를 자동으로 스케줄링하기 위한 프로세스를 도시하고 있다. 컴파일러는 DAG에 기반하여 병렬 프로세싱 디바이스에 의해 수행될 태스크의 리스트를 결정한다. 단계(1402)에서, 런타임 스케줄러는 태스크의 리스트를 수신하고 태스크의 리스트를 별개의 큐에 유지하도록 구성된다. 예를 들어, 태스크의 리스트가 (1) DMA에 의해 수행될 태스크, (2) 프로세서에 의해 수행될 태스크, 및 (3) 하드웨어 필터에 의해 수행될 태스크를 포함할 때, 런타임 스케줄러는 3개의 별개의 큐, 예를 들어, DMA에 대해서는 제1 큐, 프로세서에 대해서는 제2 큐, 및 하드웨어 필터에 대해서는 제3 큐에 태스크를 저장할 수 있다.

[0122] 단계(1404 내지 1408)에서, 런타임 컴파일러는 연관된 하드웨어 컴포넌트가 새로운 태스크에 이용가능하게 됨에 따라 연관된 하드웨어 컴포넌트에 태스크를 발행하도록 구성된다. 예를 들어, 단계(1404)에서, DMA가 새로운 태스크를 수행하도록 이용가능하게 될 때, 런타임 컴파일러는 DMA에 대한 제1 큐를 큐 해제하고, 큐 해제된 태스크를 DMA에 제공하도록 구성된다. 마찬가지로, 단계(1406)에서, 프로세서가 새로운 태스크를 수행하도록 이용가능하게 될 때, 런타임 컴파일러는 프로세서에 대한 제2 큐를 큐 해제하고, 큐 해제된 태스크를 프로세서에 제공하도록 구성된다. 또한, 단계(1408)에서, 하드웨어 필터가 새로운 태스크를 수행하도록 이용가능하게 될 때, 런타임 컴파일러는 하드웨어 필터에 대한 제3 큐를 큐 해제하고, 큐 해제된 태스크를 하드웨어 필터에 제공하도록 구성된다.

[0123] 일부 실시예에 있어서, 런타임 스케줄러(1210)는, 특히 하나보다 많은 파이프라인(예를 들어, 소프트웨어 애플리케이션(1204))이 동시에 프로세싱 엘리먼트의 어레이 상에서 실행 중인 경우, 이들 파이프라인은 반드시 공동-설계되지는 않았으므로, 프로세싱 엘리먼트의 사용을 조절하도록 하드웨어 성능 모니터 및 타이머(1308)로부터의 카운터 값을 사용할 수 있다. 예를 들면, 각각의 파이프라인에 할당된 유효 버스 대역폭이 예상된 것보다 더 작고, CMX 메모리에 액세스함에 있어서 일어나는 충돌의 수가 크면, 런타임 스케줄러(1210)는 2개 파이프라인 큐로부터 태스크가 취해지는 순서를 수정함으로써 2개 파이프라인의 실행을 스태거링하도록 이러한 정보를 사용하고, 그로써 메모리 충돌을 감축할 수 있다.

[0124] 일부 실시예에 있어서, DAG 컴파일러는 실시간(예를 들어, 온라인)으로 동작할 수 있다. 도 15는 일부 실시예에 따른 실시간 DAG 컴파일러의 연산을 예시하고 있다. 실시간 DAG 컴파일러(1502)는 DAG의 입력 XML 기술, 이용가능한 프로세싱 엘리먼트의 기술, 및 프로세서의 수, 프레임-레이트, 소비-전력 목표 등과 같은 어느 사용자-정의된 제약이라도 수신하도록 구성될 수 있다. 그때, 실시간 DAG 컴파일러(1502)는 시스템 자원에 매핑될 때 특정된 바와 같은 DAG가 사용자-정의된 제약을 충족할 수 있음을 보장하도록, 예를 들어, DMA 엔진, 프로세서, 하드웨어 필터 및 메모리를 포함하는 프로세싱 엘리먼트에 걸쳐 DAG 컴포넌트를 스케줄링하도록 구성될 수 있다. 일부 실시예에 있어서, 실시간 DAG 컴파일러(1502)는 DAG에서의 태스크가 너비-우선 방식으로 병렬로 수행될 수 있는지 결정할 수 있다. DAG의 너비가 태스크를 병렬로 수행하는데 이용가능한 프로세싱 엘리먼트의 수보다 더 크면(예를 들어, 이용가능한 프로세싱 능력의 양이 DAG의 병렬도보다 더 적으면), 실시간 DAG 컴파일러(1502)는 이용가능한 프로세싱 엘리먼트 상에서 태스크가 순차적으로 수행되도록 태스크를 "폴딩"할 수 있다.

[0125] 도 16은 일부 실시예에 따라 제안된 온라인 DAG 스케줄러에 의해 발생된 스케줄과 OpenCL 스케줄러에 의해 발생된 스케줄을 비교하고 있다. 제안된 스케줄러(1208/1502)에 의해 산출된 스케줄은 전형적 OpenCL 스케줄에 존재하는 중복 사본 및 DMA 전송을 없앨 수 있다. 이들 데이터-전송은 DAG 태스크 상의 프로세싱을 수행하는데 사용된 GPU가 스케줄을 실행하는 프로세서로부터 원격에 있기 때문에 OpenCL 스케줄 내에 존재한다. 모바일 디바이스에서 사용되는 전형적 애플리케이션 프로세서에서는, 데이터의 큰 블록이 스케줄을 실행하는 프로세서와 프로세싱을 하는 GPU 사이에서 이리저리 전송된다. 제안된 설계에서는, 프로세싱 엘리먼트 전부가 동일한 메모리 공간을 공유하고 그래서 이리저리 복사가 필요하지 않아 상당한 시간, 대역폭 및 소비 전력을 절약한다.

[0126] 일부 실시예에 있어서, 태스크가 프로세싱 엘리먼트의 능력의 일부만을 사용할 것일 때, 도 14에 예시된 바와 같이, 런타임 스케줄러(1210)는 또한, 프로세싱 엘리먼트에 의해 지원될 수 있는 한도까지, 순차적 방식으로 단일 프로세싱 엘리먼트 상에서 실행될 다중 태스크를 융합 및 스케줄링하도록 구성될 수 있다.

[0127] 이미지 프로세싱 애플리케이션에 있어서, 스케줄러는 이미지를 스트립으로 분할함으로써 프로세서들 간에 프로세싱 태스크를 분배하도록 구성될 수 있다. 예를 들어, 이미지는 사전 결정된 폭의 수직 스트립 또는 수평 스트립으로 분할될 수 있다.

[0128] 일부 실시예에 있어서, 스케줄러는 특정 이미지 프로세싱 애플리케이션에 사용되는 프로세서의 수를 미리 결정할 수 있다. 이것은 스케줄러가 이미지에 대한 스트립의 수를 미리 결정할 수 있게 한다. 일부 실시예에 있어서, 필터링 연산은 직렬로 프로세서에 의해 수행될 수 있다. 예를 들어, 애플리케이션에 의해 실행되는 5개의 소프트웨어 필터가 있을 때, 프로세서(402)들은 각각 제1 타임 인스턴스에서 동시에 제1 소프트웨어 필터를, 제2 타임 인스턴스에서 동시에 제2 소프트웨어 필터를 등등 실행하도록 구성될 수 있다. 이것은 특정 이미지 프로세싱 애플리케이션에 배정된 프로세서들 간에 컴퓨터 계산 부하가 더 균등하게 균형을 이룬다는 것을 의미한다. 이것은 프로세서들이 동일한 순서로 필터의 동일한 리스트를 동시에 실행하도록 구성되기 때문이다.

[0129] 너무 많은 프로세서가 이미지 프로세싱 애플리케이션에 배정될 때, 프로세서는, 하드웨어 필터 가속기가 태스크를 완료하도록 기다려, 유휴에 많은 시간을 소비할 수 있다. 다른 한편으로, 너무 적은 프로세서가 애플리케이션에 배정될 때, 하드웨어 필터 가속기는 유휴에 많은 시간을 소비할 수 있다. 일부 실시예에 있어서, 런타임 스케줄러(1210)는 이들 상황을 검출하고 그에 따라 적응하도록 구성될 수 있다. 다른 실시예에 있어서, 스케줄러(1210)는 특정 이미지 프로세싱 애플리케이션에 프로세서를 초과-배정하고, 프로세서가 하드웨어 필터 가속기에 앞서 그 태스크를 완료하고 나면 프로세서가 파워 다운 할 수 있게 하도록 구성될 수 있다.

[0130] 일부 실시예에 있어서, 스케줄러는 하드웨어 필터 가속기 및 프로세서와 같은 프로세싱 엘리먼트를 동기화하도록 배리어 메커니즘을 사용할 수 있다. 스케줄러의 출력은 커맨드 스트림을 포함할 수 있다. 이들 커맨드는 (1) 하드웨어 필터 가속기 및 프로세서와 같은 프로세싱 엘리먼트에 대한 시작 커맨드, 및 (2) 배리어 커맨드를 포함할 수 있다. 배리어 커맨드는, 프로세싱 엘리먼트 중 일부가 그들 태스크를 실제로 완료하였더라도, 그룹 내 모든 프로세싱 엘리먼트가 배리어 커맨드에 도달할 때까지 프로세싱 엘리먼트가 다음 커맨드 세트로 진행하는 것으로부터 기다려야 함을 나타낸다. 일부 실시예에 있어서, 스케줄러는 프로세싱 엘리먼트에 의해 수행되는 태스크들 간 의존도에 기반하여 배리어 커맨드를 제공할 수 있다.

[0131] 도 17은 일부 실시예에 따라 프로세싱 엘리먼트를 동기화하기 위한 배리어 메커니즘을 예시하고 있다. 커맨드 스트림은 배리어 커맨드(1702, 1712) 및 태스크 커맨드(1704, 1706, 1708, 1710)를 포함한다. 각각의 태스크 커맨드는 소정 프로세싱 엘리먼트와 연관될 수 있고, 아래 그래프에서 나타낸 바와 같이, 태스크 커맨드는 여러 다른 시간에 완료될 수 있다. 그래서, 스케줄러는 배리어 커맨드(1712)가 치워질 때까지 프로세싱 엘리먼트가 장래 태스크로 진행하지 않도록 배리어 커맨드(1712)를 삽입할 수 있다. 이러한 배리어 메커니즘은 병렬 태스크의 시간적 파이프라인으로 고려될 수 있다.

[0132] 일부 실시예에 있어서, 배리어 메커니즘은 인터럽트 신호(1714)를 사용하여 하드웨어에서 구현된다. 예를 들어, 스케줄러는, 어느 프로세싱 엘리먼트가 소정 그룹에 속하는지 특정하는, 비트 마스크를 프로그래밍할 수 있다. 프로세싱 엘리먼트가 배정된 태스크를 완료함에 따라, 각각의 프로세싱 엘리먼트와 연관된 인터럽트 신호가 표명된다. 그룹 내 프로세싱 엘리먼트와 연관된 모든 인터럽트 신호가 표명되고 나면, 프로세싱 엘리먼트의 컨트롤러는 모든 프로세싱 엘리먼트가 배리어 커맨드에 도달하였음을 나타내는 글로벌 인터럽트 신호를 수신할 수 있다.

[0133] 일부 실시예에 있어서, 인터럽트 소스는 SHAVE 벡터 프로세서, RISC 프로세서, 하드웨어 필터 또는 외부 이벤트를 포함할 수 있다. 특히, 하드웨어 필터는 입력/출력 버퍼가 프레임을 포함하고 있고 필터가 그것이 전체 입력 프레임을 프로세싱하였든 완전 대응하는 출력 프레임을 기록하였든 때 단일 인터럽트를 발행하도록 구성될 수 있는 경우 비-원형 버퍼 모드를 포함하는 다양한 모드를 지원할 수 있다. 또한, 필터는 이미지 치수, 버퍼 베이스 주소/라인 스트라이드 등에 적합한 설정을 사용하여 프레임으로부터의 라인, 패치 또는 타일 상에 연산하도록 프로그래밍 가능하다.

[0134] 복합 병렬 프로세싱 디바이스와의 하나의 중요한 도전과제는, 특히 컴퓨터 자원 및 메모리와 같은 자원의 관점에서 매우 전력에 민감하고 희소한 매립형 시스템에 대해, 병렬 프로세싱 디바이스에서의 프로세싱 엘리먼트를 어떻게 프로그래밍할 것인가이다. 컴퓨터 이미징, 비디오 및 이미지 프로세싱은 프레임 치수 및 레이트가 매우

높고 해마다 강하게 증가해 감에 따라 매립형 시스템 상의 성능의 관점에서 특히 매우 요구가 많아지고 있다.

[0135] 여기에서 제시된 이러한 문제에 대한 해법은 멀티코어 프로세서 아키텍처(1202)의 상세에 대한 정통한 지식 없이 프로그래머에 의해 하이 레벨로 애플리케이션이 기록될 수 있게 하는 애플리케이션 프로그래밍 인터페이스(API)(1206)를 제공하는 것이다. 소프트웨어 API(1206)를 사용하면, 기능이 프로그래밍 가능한 프로세서 상의 소프트웨어에서 구현되는지 하드웨어에서 구현되는지에 대한 상세가 프로그래머로부터 도외시되므로 프로그래머는 구현에 대한 정통한 상세를 알고 있지 않고도 새로운 이미지 또는 비디오 프로세싱 파이프라인을 신속히 생성할 수 있다. 예를 들면, 블러 필터 커널의 구현은 하나 이상의 프로세서 또는 하드웨어 가속기 필터 상에서 실행 중인 참조 소프트웨어 구현으로서 제공된다. 프로그래머는 처음에는 소프트웨어 블러 필터 구현을 사용할 수 있고 전반적 파이프라인 구현에 대한 변경 없이 하드웨어 필터를 사용하는 것으로 스위칭할 수 있는데, 프로그래머가 아니라, ISI, AMC 및 CMX 중재 블록이 어느 프로세서 및 HW 자원이 그리고 어느 순서로 물리적 메모리 블록으로의 액세스를 획득하는지를 다루기 때문이다.

[0136] 위에서 설명된 멀티-포트형 메모리 접근법이 똑같은 프로세서들 간 높은 대역폭 및 낮은 레이턴시로 메모리를 공유하는데 충분하기는 하지만, 다른 디바이스들과 대역폭을 공유하는데 이상적이지는 않다. 이들 다른 디바이스는 특히 컴퓨터 비디오 및 이미지 프로세싱과 같은 매우 높은 대역폭 애플리케이션에서 다른 레이턴시 요건들을 갖는 다른 프로세서 및 하드웨어 가속기일 수 있다.

[0137] 개시된 아키텍처는 고도로 결정론적인 레이턴시 요건을 갖는 다수의 프로그래밍 가능한 VLIW 프로세서, 큰 집단의 프로그래밍 가능한 이미지/비디오 프로세싱 하드웨어 필터, 및 관용적 호스트-프로세서 및 주변장치에 의한 제어 및 데이터-액세스를 가능하게 하는 버스 인터페이스로부터의 더 많은 동시 액세스를 지원하기 위해 더 높은 대역폭을 제공하도록 멀티-포트형 메모리 서브시스템과 함께 사용될 수 있다. 도 18은 일부 실시예에 따라 여러 다른 유형의 프로세싱 엘리먼트를 갖는 병렬 프로세싱 디바이스를 예시하고 있다. 병렬 프로세싱 디바이스는 복수의 프로세서(1802) 및 복수의 필터 가속기(1804)를 포함하고, 복수의 프로세서(1802) 및 복수의 필터 가속기(1804)는, 각각, ISI(410) 및 가속기 메모리 컨트롤러(AMC)(1806)를 통하여 메모리 서브시스템(412)에 결합될 수 있다.

[0138] AMC(1806)의 서브시스템 및 멀티코어 메모리(CMX) 서브시스템(412)은 특정 이미지/비디오 프로세싱 애플리케이션에 대해 하드웨어 필터 가속기(1804)뿐만 아니라 프로세서(1802) 상의 소프트웨어에서 저전력 스트리밍 디지털 신호 프로세싱을 용이하게 하는 온 칩 스토리지를 제공한다. 일부 실시예에 있어서, CMX 메모리(412)는 64-비트 워드로서 조직된 128kB의 16개 슬라이스로 조직된다(총 2MB). 각각의 프로세서(1802)는 메모리 서브시스템(412) 내 소정 슬라이스로의 직접 액세스 및 메모리 서브시스템(412) 내 모든 다른 슬라이스로의 간접 (더 높은 레이턴시) 액세스를 가질 수 있다. 하드웨어 필터 가속기(1804)가 데이터를 저장하도록 CMX 메모리(412)를 사용하는 동안 프로세서(1802)는 명령어 또는 데이터를 저장하도록 CMX 메모리(412)를 사용할 수 있다.

[0139] 레이턴시를 감내하지 못하는 프로세서(1802)가 HW 필터 가속기(1804)와 공유 CMX 메모리(412)에 액세스할 때 고성능을 달성할 수 있게 하면서 이종 프로세싱 엘리먼트들 간 데이터-공유를 용이하게 하기 위해, HW 필터 가속기(1804)는 레이턴시를 감내하도록 설계된다. 이것은, 일부 실시예에 따라 도 10에 예시된 바와 같이, ISI가 HW 필터 가속기로부터의 경합 없이 자유롭게 SHAVE-간 통신을 지원하게 두면서 CMX로의 액세스를 공유하도록 크로스-바 스위치뿐만 아니라, 타이밍을 더 탄력적으로 하는 로컬 FIFO도 각각의 HW 필터 가속기(필터)(1804)에 제공함으로써 달성된다.

[0140] 외부로 나가는 포트 충돌에 부가하여, 들어오는 ISI(410) 포트 액세스와 충돌하는 것이 가능하다. 하나보다 많은 외부 슬라이스가 어느 하나의 사이클에서 메모리의 동일한 슬라이스에 액세스하려고 시도하면, 그때 포트 충돌이 일어날 수 있다. LSU 포트 대 ISI 상호접속 포트의 매핑이 고정되고, 그래서 어떠한 충돌도 없이 SHAVE 0(1802-0)가 LSU 포트 0를 통해 슬라이스 2에 액세스하고 SHAVE 11(1702-11)이 LSU 포트 1을 통해 슬라이스 2에 액세스하는 것이 가능하다. ISI 매트릭스는 8x2포트x 매 사이클마다 전송될 데이터의 64-비트에 대해 허용할 수 있다. 예를 들어, SHAVE N(1802)는 LSU 포트 0 및 1을 통해 슬라이스 N+1에 액세스할 수 있고, 모든 8개의 SHAVE 프로세서는 어떠한 멎음도 없이 동시에 액세스할 수 있다.

[0141] 일부 실시예에 있어서, 메모리 서브시스템(412)은 슬라이스(블록)로 논리적으로 분할될 수 있다. 도 19는 일부 실시예에 따라 제안된 멀티코어 메모리 서브시스템을 예시하고 있다. 도 19는 AXI, AHB, SHAVE, ISI와 CMX는 물론 필터 가속기, AMC와 CMX 간 상세한 버스 상호-접속을 도시하고 있다. 선도는 2개의 AMC 입력 및 2개의 AMC 출력 포트, 2개의 ISI 입력 및 2개의 ISI 출력 포트, L2 캐시 및 상호 배제(뮤텍스) 블록으로의 접속은 물론 FIFO 및 4개의 메모리 타일을 주소지정하는 내부 판독/기록 중재 및 소스 멀티플렉싱과 더불어 ISI 및 AMC로의

4개의 메모리 블록 출력의 출력 수신지 선택도 도시하고 있다.

[0142] 각각의 슬라이스는, 12개의 SHAVE, DMA, 텍스처 관리 유닛(TMU), 및 온-보드 호스트 프로세서로의 AHB 버스 인터페이스를 포함하는, 16개의 가능한 ISI 입력 소스 중 2개에 접속될 수 있다. 유사하게, 각각의 슬라이스는 12개의 SHAVE, DMA, 텍스처 관리 유닛(TMU), 및 온-보드 호스트 프로세서로의 AXI 및 AHB 버스 인터페이스를 포함하는 16개의 가능한 수신지 중 2개에 슬라이스가 데이터를 보낼 수 있게 하는 2개의 출력 ISI 포트를 갖는다. 바람직한 구현에 있어서, 슬라이스는 로컬 SHAVE 프로세서(2xLSU 및 2x64-비트 명령어 포트), 2개의 ISI 입력 포트, 2개의 AMC 입력 포트, 및 SHAVE-간 메시징에 사용되는 FIFO는 물론 메시징 FIFO, L2 캐시 및 뮤텍스 블록에도 차례로 접속되는 입력 중재 블록을 갖는 4개의 물리적 RAM 블록을 포함하고 있다.

[0143] CMX 슬라이스로부터의 출력 경로 상에서, 수신지 선택 블록으로의 입력은, L2 캐시 및 하드웨어 뮤텍스 블록과 함께, 4개의 RAM 인스턴스에 접속된다. 수신지 선택 블록으로부터의 출력은, 블록(2002)으로서 도 20에 예시된 바와 같이, 2개의 로컬 LSU 포트 및 명령어 포트(SP_1, SP_0)는 물론 2xISI 출력 포트 및 2xAMC 출력 포트에도 접속된다. 2개의 ISI 포트는 로컬 슬라이스가 12개의 가능한 프로세서, DMA, TMU, AXI 및 AHB 호스트 버스로부터의 2개의 수신지에 접속될 수 있게 한다. 프로세서에는 멀티코어 메모리 서브시스템 슬라이스에 포함되어 있는 2x64-비트 ISI 입력 및 2x64-비트 ISI 출력 포트에 접속되는 SHAVE 간 상호접속부(ISI)를 통하여 메모리로의 액세스가 제공된다. ISI 상호접속부에 의해 제공되는 높은-대역폭, 낮은-레이턴시 결정론적 액세스는 프로세서에서의 멎음을 감축하고 높은 컴퓨터 계산 처리율을 제공한다.

[0144] 도 21은 일부 실시예에 따라 AMC 크로스바 아키텍처를 예시하고 있다. AMC 크로스바(1806)는 하드웨어 이미지-프로세싱 필터(1804)를 CMX 멀티코어 메모리 슬라이스(412)의 AMC 포트에 접속하도록 구성될 수 있다. AMC(1806)는 하나 이상의 슬라이스 포트 컨트롤러(2102)를, 바람직하게는 각각의 CMX 슬라이스(412)에 대해 하나씩, 포함할 수 있다. 슬라이스 포트 컨트롤러(2102)는 차례로 슬라이스 주소 요청 필터(SARF)(2104)에 접속된다. SARF(2104)는 차례로 AMC 클라이언트에 접속된다(이 실시예에서 AMC 클라이언트는 하드웨어 이미지-프로세싱 가속기이다). SARF(2104)는 필터 가속기로부터의 판독/기록 요청을 수락하고 그것들로의 요청 또는 승인 신호보내기, 기록 액세스를 승인받은 SIPP 블록으로부터의 데이터 및 주소 수락하기, 및 판독 액세스를 승인받은 것들에 판독 데이터 제공하기를 제공한다. 부가적으로, SARF는 시스템 호스트 프로세서에 AXI 호스트 버스에 대한 AXI 마스터링을 제공하여, 호스트가 AMC 크로스바 스위치를 통하여 CMX 메모리에 질의(판독/기록)할 수 있게 한다.

[0145] 일부 실시예에서는, 도 21에 도시된 바와 같이 슬라이스 포트 컨트롤러(2102)가 AMC(1806)에 제공되어 연관된 CMX 메모리 슬라이스(412)에서의 2x판독 및 2x기록 포트와 통신한다. CMX 메모리 서브시스템(412)의 필터 가속기측으로부터 보면, 각각의 하드웨어 필터는 가속기 메모리 컨트롤러(AMC) 크로스바(1806) 상의 소정 포트에 접속된다. AMC(1806)는 한 쌍의 64 비트 판독 및 한 쌍의 64-비트 기록-포트를 가지며, 그것을 CMX 메모리(412)의 각각의 슬라이스에 접속한다(바람직한 구현에서는 총 2MB에 대해 16개 슬라이스가 있다). 가속기 내 로컬 버퍼링 및 판독 또는 기록-클라이언트 인터페이스를 통하여 AMC(1806)에 이미지 프로세싱 하드웨어 가속기를 접속하는 것은 프로세서의 멎음이 감축될 수 있게 하는 고도로 결정론적인 타이밍으로 더 많은 대역폭을 ISI 및 프로세서가 이용가능하게 두어 더 완화된 레이턴시 요건을 허용한다.

[0146] 도 20은 일부 실시예에 따라 CMX 기반구조의 단일 슬라이스를 예시하고 있다. 슬라이스는 CMX 슬라이스에서의 4개의 물리적 SRAM 타일로의 액세스를 8개의 가능한 64-비트 소스 중으로부터 4개까지 허용하는 중재기 및 소스 멀티플렉싱, 공유 L2 캐시, 스레드 간 상호 배제의 프로세서-간 교섭을 위한 공유 뮤텍스 하드웨어 블록은 물론, SHAVE-간 낮은 대역폭 메시징에 사용되는 64-비트 FIFO도 포함하고 있다. 6개의 입력 소스는: 도 21에 도시된 ACM로부터의 대응하는 슬라이스_포트[1] 및 슬라이스_포트[0] 포트에 접속되는 AMCout1 및 AMCout0; 부가적으로 2개의 ISI 포트, ISIout1 및 ISIout0; 2x LSU 포트 LSU_1 및 LSU_0; 및 마지막으로 조합될 때 128-비트 명령어가 CMX로부터 판독될 수 있게 하는 2x명령어 포트 SP_1 및 SP_0이다. 중재기 및 소스 멀티플렉싱은 8개의 입력 소스로부터의 우선순위 액세스에 응답하여 4xSRAM 타일을 제어하도록 판독/기록 주소 및 64-비트 데이터를 발생시킨다. 64-비트 SHAVE-간 통신 FIFO의 입력은 중재기 및 소스 멀티플렉서에 접속되는 한편, 그 출력은 CMX 슬라이스에서의 로컬 프로세서에 의해서만 판독가능하다. 실제로, 프로세서는, CMX, 슬라이스 및 프로세서를 함께 접속하는, CMX 슬라이스 외부 ISI 기반구조 및 ISIout1 및 ISIout0 포트를 통하여 서로의 메시징 FIFO와 통신한다.

[0147] 도 20에 도시된 슬라이스에서의 2개의 AMC 포트의 각각 상의 64 요청기들 간 중재에 부가하여, 부가적 2:1 중재기가 AMC 포트 1와 0 간 중재하도록 제공된다. 이러한 2:1 중재기의 목적은 2개의 AMC 포트 중 하나 또는 다른

하나가, 요청하는 포트 중 하나 상에서의 과도한 멎음을 초래할 수 있는, AMC 포트 대역폭 전부를 포화시키는 것을 방지하는 것이다. 이러한 강화는 포트 대역폭의 다중의 과중한 요청기의 존재시 더 균형을 이루는 자원 할당 및 그리하여 전반적 아키텍처에 대한 더 높은 지속적 처리율을 제공한다. 유사하게, 2:1 중재기는 유사한 이유로 2개의 프로세서 포트 SP1와 SP0 간 중재한다.

[0148] 또한, 중재 및 멀티플렉싱 로직은, 하나의 64-비트 포트가 16개의 CMX 슬라이스의 각각과 제2 레벨 중재기 사이에 접속되는 경우, 16개의 가능한 소스 간 엄밀한 라운드-로빈 기반으로 액세스를 공유하는 제2 레벨의 중재를 통하여, 공유 L2 캐시로 ISI를 통하여서든 직접적이든, 프로세서에 의한 액세스를 제어한다. 유사하게, 동일한 로직은 (ISI 상에서의 AHB 및 AXI 버스 접속을 통하여) 12개의 온-보드 프로세서 및 2x32-비트 RISC 프로세서 상에서 실행되는 스레드 간 상호 배제의 프로세서-간 교섭에 사용되는 32개의 하드웨어 뮤텍스로의 액세스를 허용한다.

[0149] 우선순위는 바람직한 구현에서는 SP_1와 SP_0가 가장 높은 우선순위를 갖고, 그 후 LSU_1와 LSU_0, 그 다음에 ISIout1와 ISIout0 그리고 마지막으로 AMCout1와 AMCout0이고 마지막으로 FIFO는 가장 낮은 우선순위를 갖는다. 이러한 우선순위 배정에 대한 이유는 SP_1와 SP_0가 CMX로의 프로세서에 의한 프로그래밍 액세스를 제어하고 프로세서는 다음 명령어가 이용가능하지 않으면 즉시 멎을 것이고, 또다시 프로세서가 멎게 야기할 LSU_1와 LSU_0가 뒤따르고; 유사하게 ISIout1와 ISIout0가 다른 프로세서로부터 유래하고 데이터가 즉시 이용가능하지 않으면 그것들이 멎게 야기할 것이기 때문이다. AMCout1와 AMCout0 포트는 그것들이 내장 FIFO를 갖고 그리하여 멎기 전에 많은 레이턴시를 감내할 수 있으므로 가장 낮은 우선순위를 갖는다. 프로세서 FIFO는 낮은 대역폭 프로세서-간 메시징에 필요로 될 뿐이고 그리하여 모든 것 중 가장 낮은 우선순위를 갖는다.

[0150] 중재기가 4개 소스까지 4개 SRAM 타일, L2 캐시, 뮤텍스 및 FIFO로의 액세스를 허용하고 나면, 4개 SRAM 타일, L2 캐시 및 뮤텍스를 포함하는 6개 판독-데이터 소스로부터의 출력 데이터는 8개의 가능한 64-비트 수신지 포트; 슬라이스와 연관된 프로세서 상의 4개(SP_1, SP_0, LSU_1, LSU_0), ISI와 연관된 2개(ISIout1, ISIout0) 및 마지막으로 AMC와 연관된 2개(AMCout1, AMCout0) 중으로부터 4개까지로 선택 및 지향된다. 출력 멀티플렉서에서는 8개의 수신지 포트에 4개의 64-비트 소스만이 분배될 필요가 있으므로 우선순위결정은 필요하지 않다.

[0151] 도 22는 일부 실시예에 따라 AMC 크로스바 포트 컨트롤러를 예시하고 있다. AMC 크로스바 포트 컨트롤러(2202)는 프로세서를 통하여 요청이 필터링된 가속기(1804)에 포트 컨트롤러(2202)를 접속하는 라운드-로빈 중재기(2204)를 포함한다. 그 후, 중재기는 포트-컨트롤러 FIFO 상에 AMC 클라이언트로부터의 유효 요청을 푸싱할 수 있다. 판독 요청의 경우에, 요청에 대한 응답(판독-클라이언트 ID 및 라인 인덱스)이 판독 TX ID FIFO 상에 푸싱된다. 반환된 슬라이스 포트 데이터 및 유효 신호는 판독 TX ID FIFO로부터 판독 클라이언트 ID 및 라인 인덱스를 팝핑하고 대응하는 슬라이스 포트 판독 데이터 및 유효 신호를 그것이 요청하는 AMC 클라이언트에 의해 판독되어 나올 수 있는 판독 데이터 FIFO 상에 푸싱하는 포트-컨트롤러 판독 로직 내로 입력된다. FIFO의 CMX측 상에서, 포트 멎음 로직은 FIFO로부터 요청을 팝핑하고 연관된 CMX 메모리 슬라이스 상의 2xAMC 입력 포트에 슬라이스 포트 제어를 제공한다.

[0152] CMX로의 판독 및 기록 클라이언트 인터페이스의 수는 별개로 구성가능하다. 어느 클라이언트라도 CMX의 어느 슬라이스(또는 슬라이스들)라도 주소지정할 수 있다. CMX에서의 16개 슬라이스, 슬라이스당 2개 포트 및 600MHz의 시스템 클록 주파수로는, 클라이언트에 공급될 수 있는 최대 총 데이터 메모리 대역폭은 143GB/s이다: 최대 대역폭 = 600MHz * (64/8) * 2 * 16 = 1.536e11 B/s = 143GB/s.

[0153] 800MHz의 더 높은 클록-레이트에서 대역폭은 191GB/s로 상승한다. AMC는 그것에 접속된 하드웨어 가속기 블록의 그 클라이언트 판독/기록 인터페이스 상의 동시 액세스 간 중재한다. 2개의 판독/기록 액세스 중 최대치가 클록 사이클당 슬라이스당 승인되어, 600MHz의 시스템 클록 주파수에서 8.9GB/s의 최대 슬라이스 메모리 대역폭을 줄 수 있다. 클라이언트 액세스는 CMX 주소 공간으로 제한되지 않는다. CMX 주소 공간 밖에 드는 어느 액세스라도 AMC의 AXI 버스 마스터에 포워딩된다.

[0154] 도 23은 일부 실시예에 따라 AMC(1806)를 사용하는 판독 연산을 예시하고 있다. 이 예시에서는, 4개의 데이터 워드가 주소 범위 A0-3로부터 판독된다. AMC 클라이언트(예를 들어, 필터 가속기(1804))는 우선 포트 컨트롤러 입력 상에 요청을 표명한다. 포트 컨트롤러(2202)는 차례로 클라이언트가 주소 A0, A1, A2 그리고 마지막으로 A3를 방출하게 야기하는 승인 신호(gnt)를 발행함으로써 응답한다. 대응하는 rindex 값은 각각의 승인에 대응하여 클록(clk)의 상승 에지 상에 나타난다. CMX 슬라이스로부터 포트-컨트롤러로 출력되는, 인덱스 주소 및 데이터에 비교될 때, 클라이언트측 상의 타이밍은 아주 탄력적일 수 있음을 알 수 있다. 포트-컨트롤러의 CMX측상의 결정론적 타이밍은 레이턴시에 매우 민감한 프로세서와 AMC 클라이언트 간 CMX로의 효율적 공유 액세스를 가능

하게 하는 한편, FIFO가 가능하고 AMC 클라이언트에서의 로컬 저장은 타이밍이 CMX 메모리 서브시스템(412)의 AMC 클라이언트(예를 들어, 필터 가속기)측 상에서 고도로 가변일 수 있게 한다.

[0155] 도 24는 일부 실시예에 따라 AMC(1806)를 사용하는 기록 연산을 예시하고 있다. 타이밍 선도에서, AMC를 통하여 CMX로 4개의 데이터 워드의 전송이 도시되어 있다. 요청이 AMC 클라이언트에 의해 발행되고 클록(clk)의 다음의 상승 에지 상에서 승인 신호(gnt)는 하이로 가서 AMC를 통하여 주소 A0와 연관된 데이터-워드 D0를 전송한다. 그 후 gnt 신호는 한번의 클록 사이클 동안 로우로 가고 clk의 다음 상승 에지 상에서 gnt는 2번의 클록 사이클 동안 하이로 가서 gnt가 다시 로우로 가기 전에 D1 및 D2가 각각 A1 및 A2를 주소지정할 수 있게 한다. 다음의 상승 clk 에지 상에서 gnt는 다시 하이로 가서, 데이터 워드 D3가 주소 A3에 전송될 수 있고 하고, 그 결과 req 및 gnt는 clk의 다음 상승 에지 상에서 로우 가서 다음 판독/기록 요청을 기다린다.

[0156] 도 12의 예시적 배열과 함께 사용되는 스트리밍 이미지 프로세싱 파이프라인(SIPP) 소프트웨어 프레임워크는 이미지/비디오 프로세싱 다이가 배속되는 기판에 와이어-본딩된 패키지에 외부 DRAM 다이의 부속으로 고해상도로 스캔-라인 버퍼, 프레임 타일(프레임의 서브섹션) 또는 참으로 프레임 전체에 대해 CMX 메모리(412)를 사용하여 이미지 프로세싱 파이프라인을 구현하는 것으로의 융통성 있는 접근법을 제공한다. SIPP 프레임워크는 원형 라인 버퍼 관리 및 이미지 보더(픽셀의 복제) 취급과 같은 복잡성을 처리하여 (프로세서 상의) 소프트웨어에서의 ISP(이미지 신호-프로세싱) 기능의 구현을 더 단순하고 더 일반적이게 한다.

[0157] 도 25는 일부 실시예에 따른 병렬 프로세싱 디바이스(400)를 예시하고 있다. 병렬 프로세싱 디바이스(400)는 메모리 서브시스템(CMX)(412), 복수의 필터 가속기(1804), 및 메모리 서브시스템(412)으로의 액세스를 중재하기 위한 버스 구조(1806)를 포함할 수 있다. 메모리 서브시스템(CMX)(412)은 복수의 프로세싱 엘리먼트(402)가 멎음 없이, 병렬로, 데이터 및 프로그램 코드 메모리에 액세스할 수 있게 하도록 구축된다. 이들 프로세싱 엘리먼트(402)는, 예를 들어, SHAVE(스트리밍 하이브리드 아키텍처 벡터 엔진) 프로세서, 적합하게는 VLIW(매우 긴 명령어 워드) 프로세서, 멎음 없이 데이터 및 프로그램 코드 메모리로의 병렬 액세스, 또는 필터 가속기를 포함할 수 있다. 부가적으로, 메모리 서브시스템(CMX)(412)은 호스트 프로세서(도시되지 않음)가 AXI(도시되지 않음)와 같은 병렬 버스를 통하여 CMX 메모리 서브시스템(412)에 액세스하도록 준비할 수 있다. 일부 실시예에 있어서, 각각의 프로세싱 엘리먼트(402)는 그 LSU 포트를 통해 사이클당 128-비트까지 판독/기록하고 그 명령어 포트를 통해 사이클당 128 비트 프로그램 코드까지 판독할 수 있다. 각각 프로세서 및 필터 가속기에 대한 ISI 및 AMC 인터페이스에 부가하여, CMX(412)는 AHB 및 AXI 버스 인터페이스를 통해 메모리로의 동시 판독/기록 액세스를 제공한다. AHB 및 AXI는 프로세서, 메모리 및 주변장치가 공유 버스 기반구조(1806)를 사용하여 접속될 수 있게 하는 표준 ARM 병렬 인터페이스 버스이다. CMX 메모리 서브시스템(412)은 사이클당 18x128-비트 메모리 액세스 피크를 취급하도록 구성될 수 있다.

[0158] 가속기(1804)는 SIPP 소프트웨어 프레임워크(1200)에서 사용될 수 있는 하드웨어 이미지 프로세싱 필터 집합을 포함한다. 가속기(1804)는 가장 컴퓨터 계산 집약적인 기능성 중 일부가 프로세싱 엘리먼트(1802)로부터 분담 가능하게 할 수 있다. 선도는 주소 필터링, 중재 및 멀티플렉싱을 수행하는 AMC(1804)에 복수의 필터 가속기(1804)가 어떻게 접속될 수 있는지 보여주고 있다. 또한, AMC(1804)에는 다중 MIPI 카메라 직렬 인터페이스(2502)가 접속될 수 있고, 바람직한 구현에서는 총 12개의 MIPI 직렬 레인이 2개 레인의 6개 그룹으로 접속된다. 또한, AMC(1804)는 참조 구현에서는 2개의 시스템 RISC 프로세서가 AMC를 통하여 CMX 메모리에 액세스할 수 있게 하도록 AXI 및 APB 인터페이스에 접속된다. 선도의 마지막 엘리먼트는 CMX(412)로서 AMC(1804)가 그로의 액세스를 중재하여, CMX 메모리(412)에서의 물리적 RAM 인스턴스로의 다중 하드웨어 필터 가속기(1804)에 의한 동시 액세스를 가능하게 한다. 참조 필터 가속기(1804)가 또한 도시되어 있고, 이 경우에서는 5x5 2D 필터 커널로서, fp16(IEEE754-같은 16-비트 부동-소수점 포맷) 산술 파이프라인, 연관된 파이프라인 멎음 컨트롤러, fp16 파이프라인으로의 입력의 라인을 저장하는 라인 버퍼 판독 클라이언트, 라인 시작 제어 입력, 및 fp16 파이프라인으로부터의 출력을 저장하는 라인 버퍼 기록 클라이언트를 포함하고 있다. 가속기가 SIPP 프레임워크에 들어맞을 수 있게 하기 위해 그것들은 CMX 메모리로의 높은 대역폭 액세스를 필요로 하고, 이것은 가속기 메모리 컨트롤러(AMC)에 의해 제공된다.

[0159] 일부 실시예에 있어서, CMX 메모리 서브시스템(412)은 고속, 저전력 액세스를 위해 그 이웃하는 프로세싱 엘리먼트(402)와 연관된 128kB 블록 또는 슬라이스로 분할될 수 있다. 슬라이스 내에서, 메모리는 소정 수의 더 작은 타일, 예를 들어 3x32kB, 1x16kB 및 2x8KB 독립 SRAM 블록으로서 조직된다. 물리적 RAM 사이즈는 면적 이용도 및 구성 융통성의 절충점으로서 선택될 수 있다. 어느 프로세싱 엘리먼트(402)라도 동일한 레이턴시(3 사이클)로 메모리 서브시스템(CMX)(412)에서의 어느 곳에서라도 물리적 RAM에 액세스할 수 있지만, 로컬 프로세서 슬라이스 밖의 액세스는 대역폭이 제한되고 로컬 슬라이스의 액세스보다 더 많은 전력을 소모할 것이다. 일반적

으로, 전력 소모를 감소시키기 위해 그리고 성능을 증가시키기 위해, 프로세싱 엘리먼트(402)는 데이터를 전용 메모리 슬라이스에 로컬 저장할 수 있다.

[0160] 일부 실시예에 있어서, 각각의 물리적 RAM은 64-비트 폭일 수 있다. 하나보다 많은 프로세싱 엘리먼트(402)가 단일 물리적 RAM에 액세스하려고 시도하면, 충돌이 일어나 프로세서 멎음을 초래할 수 있다. CMX는 포트 충돌들 간 자동으로 중재하고 데이터가 손실되지 않음을 보장할 것이다. 모든 포트 충돌마다, 프로세싱 엘리먼트(402)가 소정 사이클 동안 멎어 더 낮은 처리율을 초래한다. CMX(412) 내에서의 (프로그래머에 의한) 주의깊은 데이터 레이아웃으로, 포트 충돌은 회피되고 프로세서 사이클은 더 잘 이용될 수 있다.

[0161] 일부 실시예에 있어서, 복수의 프로세서에는 가속기 및 CMX 메모리가 제공된다.

[0162] 각각의 필터 가속기(1804)는 CMX 메모리(412)에 액세스하도록 적어도 하나의 AMC 판독 및/또는 기록 클라이언트 인터페이스를 포함할 수 있음을 도 25에 도시된 이미지-프로세싱 HW 아키텍처로부터 알 수 있다. AMC(1806) 상의 판독/기록 클라이언트 인터페이스의 수는 적합하게 구성가능하다. AMC(1806)는 CMX 메모리(412)의 각각의 슬라이스로의 한 쌍의 64 비트 포트를 포함할 수 있다. AMC(1806)는 (부분적 주소 디코딩에 의해) 적합한 CMX 슬라이스(412)로 그 클라이언트로부터의 요청을 라우팅한다. 여러 다른 클라이언트로부터 동일 슬라이스로의 동시 요청은 라운드-로빈 방식으로 중재될 수 있다. CMX(412)로부터 반환된 판독 데이터는 다시 요청하는 AMC 판독 클라이언트로 라우팅된다.

[0163] AMC 클라이언트(가속기)(1804)는 AMC(1806)에 풀 32 비트 주소를 제시한다. CMX 메모리 공간에 매핑하지 않는 클라이언트로부터의 액세스는 AMC의 AXI 마스터에 포워딩된다. 여러 다른 클라이언트로부터의 (CMX 메모리 공간 밖의) 동시 액세스는 라운드-로빈 방식으로 중재된다.

[0164] AMC(1806)는 필터 가속기(1804)에 CMX(412) 액세스를 제공하는 것으로 한정되지 않고; 어느 하드웨어 가속기 또는 제3자 엘리먼트라도 CMX 및 플랫폼의 더 넓은 메모리 공간에, 그 메모리 인터페이스가 AMC 상의 판독/기록 클라이언트 인터페이스에 적합하게 적응되면, 액세스하도록 AMC(1806)를 사용할 수 있다.

[0165] 하드웨어 이미지 프로세싱 파이프라인(SIPP)은 필터 가속기(1804), 중재 블록(1806), MIPI 제어(2502), APB 및 AXI 인터페이스 및 CMX 멀티포트 메모리(412)로의 접속은 물론 일례의 하드웨어 5x5 필터 커널을 포함할 수 있다. 이러한 배열은 이미지-프로세싱 애플리케이션에 대한 복수의 프로세서(1802) 및 하드웨어 필터 가속기(1804)가 복수의 단일-포트형 RAM(Random Access Memory) 물리적 블록으로 이루어진 메모리 서브시스템(412)을 공유할 수 있게 한다.

[0166] 단일-포트형 메모리의 사용은 메모리 서브시스템의 전력 및 면적 효율을 증가시키지만 대역폭을 제한한다. 제안된 배열은 이들 RAM 블록이 다중 물리적 RAM 인스턴스를 사용하고 그것들로의 중재된 액세스를 제공하여 다중 소스에 서비싱함으로써 다중 소스(프로세서 및 하드웨어 블록)로부터의 다중 동시 판독 및 기록 요청에 서비싱할 수 있는 가상의 멀티-포트형 메모리 서브시스템으로서 거동할 수 있게 한다.

[0167] 애플리케이션 프로그래밍 인터페이스(API)의 사용, 및 애플리케이션 레벨에서의 데이터 파티셔닝은 프로세서와 필터 가속기 간, 또는 프로세서 자신들 간 물리적 RAM 블록에 대한 경합이 감축되고 그리하여 프로세서 및 하드웨어에 전달되는 데이터 대역폭이 주어진 메모리 서브시스템 구성에 대해 증가됨을 보장하기 위해 중요하다.

[0168] 일부 실시예에 있어서, 병렬 프로세싱 디바이스(400)는 전자 디바이스에 거주할 수 있다. 도 26은 일부 실시예에 따라 병렬 프로세싱 디바이스를 포함하는 전자 디바이스를 예시하고 있다. 전자 디바이스(2600)는 프로세서(2602), 메모리(2604), 하나 이상의 인터페이스(2606) 및 병렬 프로세싱 디바이스(400)를 포함할 수 있다.

[0169] 전자 디바이스(2600)는 컴퓨터 판독가능한 매체, 플래시 메모리, 자기 디스크 드라이브, 광학 드라이브, 프로그래밍가능 판독 전용 메모리(PROM), 및/또는 판독 전용 메모리(ROM)와 같은 메모리(2604)를 가질 수 있다. 전자 디바이스(2600)는 명령어를 프로세싱하고 메모리(2604)에 저장될 수 있는 소프트웨어를 실행하는 하나 이상의 프로세서(2602)로 구성될 수 있다. 프로세서(2602)는 또한 다른 디바이스와 통신하도록 인터페이스(2606) 및 메모리(2604)와 통신할 수 있다. 프로세서(2602)는 CPU, 애플리케이션 프로세서 및 플래시 메모리를 조합하는 시스템-온-칩, 또는 축소형 명령어 세트 컴퓨팅(RISC) 프로세서와 같은 어느 적용가능한 프로세서라도 될 수 있다.

[0170] 일부 실시예에 있어서, 컴파일러(1208) 및 런타임 스케줄러(1210)는 메모리(2604)에 저장된 소프트웨어에서 구현되고, 프로세서(2602) 상에서 동작할 수 있다. 메모리(2604)는 비-일시적 컴퓨터 판독가능한 매체, 플래시 메모리, 자기 디스크 드라이브, 광학 드라이브, 프로그래밍가능 판독 전용 메모리(PROM), 판독 전용 메모리(ROM),

또는 어느 다른 메모리 또는 메모리 조합이라도 될 수 있다. 소프트웨어는 컴퓨터 명령어 또는 컴퓨터 코드를 실행할 수 있는 프로세서 상에서 실행될 수 있다. 프로세서는 또한 주문형 반도체(ASIC), 프로그래밍가능 로직 어레이(PLA), 필드 프로그래밍가능 게이트 어레이(FPGA), 또는 어느 다른 집적 회로를 사용하여 하드웨어에서 구현될 수 있다.

[0171] 일부 실시예에 있어서, 컴파일러(1208)는 인터페이스(2606)를 통하여 전자 디바이스(2600)와 통신하고 있는 별개의 컴퓨팅 디바이스에서 구현될 수 있다. 예를 들어, 컴파일러(1208)는 전자 디바이스(2600)와 통신하고 있는 서버에서 동작할 수 있다.

[0172] 인터페이스(2606)는 하드웨어 또는 소프트웨어에서 구현될 수 있다. 인터페이스(2606)는, 텔레비전에 대한 리모컨과 같은, 로컬 소스뿐만 아니라 네트워크로부터도 데이터 및 제어 정보 양자를 수신하는데 사용될 수 있다. 전자 디바이스는 또한 키보드, 터치 스크린, 트랙볼, 터치 패드 및/또는 마우스와 같은 다양한 사용자 인터페이스를 제공할 수 있다. 전자 디바이스는 일부 실시예에서는 스피커 및 디스플레이 디바이스를 포함할 수도 있다.

[0173] 일부 실시예에 있어서, 병렬 프로세싱 디바이스(400)에서의 프로세싱 엘리먼트는 컴퓨터 명령어 또는 컴퓨터 코드를 실행할 수 있는 집적 칩을 포함할 수 있다. 프로세서는 또한 주문형 반도체(ASIC), 프로그래밍가능 로직 어레이(PLA), 필드 프로그래밍가능 게이트 어레이(FPGA), 또는 어느 다른 집적 회로를 사용하여 하드웨어에서 구현될 수 있다.

[0174] 일부 실시예에 있어서, 병렬 프로세싱 디바이스(400)는 시스템 온 칩(SOC)으로서 구현될 수 있다. 다른 실시예에 있어서, 병렬 프로세싱 디바이스에서의 하나 이상의 블록은 별개의 칩으로서 구현될 수 있고, 병렬 프로세싱 디바이스는 시스템 인 패키지(SIP)에 패키징될 수 있다. 일부 실시예에 있어서, 병렬 프로세싱 디바이스(400)는 데이터 프로세싱 애플리케이션에 사용될 수 있다. 데이터 프로세싱 애플리케이션은 이미지 프로세싱 애플리케이션 및/또는 비디오 프로세싱 애플리케이션을 포함할 수 있다. 이미지 프로세싱 애플리케이션은 이미지 필터링 연산을 포함하는 이미지 프로세싱 프로세스를 포함할 수 있고; 비디오 프로세싱 애플리케이션은 비디오 디코딩 연산, 비디오 인코딩 연산, 비디오 내 물체 또는 모션을 검출하기 위한 비디오 분석 연산을 포함할 수 있다. 본 발명의 부가적 애플리케이션은 비디오, 물체 또는 이미지의 시퀀스에 기반하는 기계 학습 및 분류, 및 게이밍 애플리케이션이 심도 가능 카메라를 포함하는 다중 카메라 뷰로부터 기하구조를 추출하고 와이어프레임 기하구조(예를 들면, 포인트-클라우드를 통하여)가 GPU에 의한 후속 정점 셰이딩을 위해 추출되어 나올 수 있는 다중 뷰로부터 특징을 추출하는 것들을 포함하는 증강 현실 애플리케이션을 포함한다.

[0175] 전자 디바이스(2600)는 셀룰러 폰과 같은 모바일 디바이스를 포함할 수 있다. 모바일 디바이스는 복수의 액세스 기술을 사용하는 복수의 라디오 액세스 네트워크와 그리고 유선 통신 네트워크와 통신할 수 있다. 모바일 디바이스는 워드 프로세싱, 웹 브라우징, 게이밍, 전자책 능력, 운영 체제 및 풀 키보드와 같은 고급 능력을 제공하는 스마트 폰일 수 있다. 모바일 디바이스는 심비안 운영 체제, 아이폰 운영 체제, 림의 블랙베리, 윈도우즈 모바일, 리눅스, 팜 웹 운영 체제 및 안드로이드와 같은 운영 체제를 실행할 수 있다. 스크린은 모바일 디바이스에 데이터를 입력하는데 사용될 수 있는 터치 스크린일 수 있고 스크린은 풀 키보드 대신에 사용될 수 있다. 모바일 디바이스는 애플리케이션을 실행하거나 또는 통신 네트워크에서의 서버에 의해 제공되는 애플리케이션과 통신할 수 있는 능력을 가질 수 있다. 모바일 디바이스는 네트워크 상의 이들 애플리케이션으로부터 업데이트 및 다른 정보를 수신할 수 있다.

[0176] 또한, 전자 디바이스(2600)는 텔레비전(TV), 비디오 프로젝터, 셋-톱 박스 또는 셋-톱 유닛, 디지털 비디오 레코더(DVR), 컴퓨터, 넷북, 랩톱, 태블릿 컴퓨터, 및 네트워크와 통신할 수 있는 어느 다른 청각/시각 장비와 같은 여러 다른 디바이스를 망라할 수 있다. 또한, 전자 디바이스는 전역 측위 좌표, 프로파일 정보, 또는 다른 위치 정보를 그 스택 또는 메모리에 유지할 수 있다.

[0177] 여기에서 수개의 다른 배열이 설명되었기는 하지만, 각각의 특징은 이점을 달성하도록 다양한 형태로 함께 조합되는 것이 유익할 수도 있음을 인식할 것이다.

[0178] 상기 명세서에서, 본 출원이 특정 예를 참조하여 설명되었다. 그렇지만, 첨부 청구범위에서 제시된 바와 같이 본 발명의 더 넓은 취지 및 범위로부터 벗어남이 없이 다양한 수정 및 변경이 여기에 이루어질 수 있음은 명백할 것이다. 예를 들어, 접속은, 예를 들어 중간 디바이스를 통하여, 각각의 노드, 유닛 또는 디바이스로 또는 그로부터 신호를 전송하는데 적합한 어느 유형의 접속이라도 될 수 있다. 따라서, 달리 내포되거나 서술되지 않는 한, 접속은 예를 들어 직접 접속일 수도 있고 간접 접속일 수도 있다.

[0179] 여기에서 묘사된 아키텍처는 단지 예시일 뿐이고, 실제로는 동일한 기능성을 달성하는 많은 다른 아키텍처가 구

현될 수 있다고 이해되는 것이다. 추상적이지만 그래도 확실한 의미로, 동일한 기능성을 달성하는 컴포넌트의 어느 배열이라도 소망 기능성이 달성되도록 효과적으로 "연관된다". 그리하여, 특정 기능성을 달성하도록 조합된 여기에서의 어느 2개의 컴포넌트라도, 아키텍처 또는 중간 컴포넌트와 무관하게, 소망 기능성이 달성되도록 서로 "연관된" 것으로 보일 수 있다. 마찬가지로, 그렇게 연관된 어느 2개의 컴포넌트라도 또한 소망 기능성을 달성하도록 서로 "동작가능하게 접속된" 또는 "동작가능하게 결합된" 것으로 보일 수 있다.

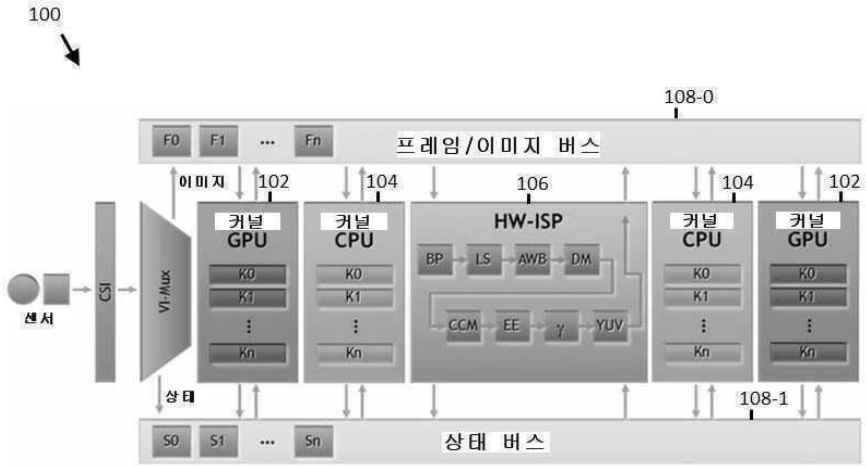
[0180] 더욱, 당업자는 위에서 설명된 연산의 기능성 간 경계가 단지 예시적인 것임을 인식할 것이다. 다중 연산의 기능성은 단일 연산으로 조합될 수 있고, 그리고/또는 단일 연산의 기능성은 부가적 연산으로 분산될 수 있다. 더욱, 대안의 실시예는 특정 연산의 다중 인스턴스를 포함할 수 있고, 연산의 순서는 다양한 다른 실시예에서는 바뀔 수 있다.

[0181] 그렇지만, 다른 수정, 변형 및 대안도 가능하다. 따라서, 명세서 및 도면은 제한적 의미라기보다는 예시적인 것으로 간주되는 것이다.

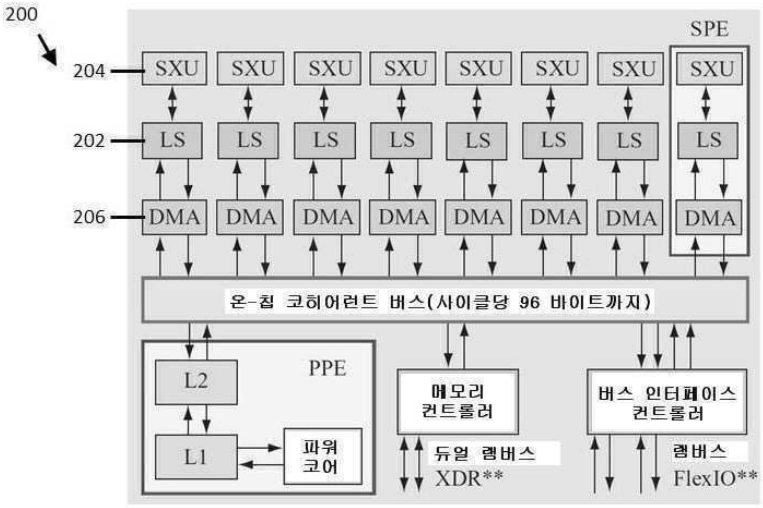
[0182] 청구범위에서, 어구들 사이에 놓이는 어느 참조 기호라도 청구범위를 한정하는 것으로 해석되어서는 안 된다. 단어 '포함한다'는 청구범위에서 열거된 것들 이외의 다른 구성요소 또는 단계의 존재를 배제하지는 않는다. 더욱, 단수형 부정관사는, 여기에서 사용될 때, 하나보다는 하나 이상으로서 정의된다. 또한, 청구범위에서 "적어도 하나" 및 "하나 이상"과 같은 도입구의 사용은 단수형 부정 관사에 의한 다른 청구항 구성요소의 도입이 그러한 도입된 청구항 구성요소를 포함하고 있는 어느 특정 청구항이라도 단 하나의 그러한 구성요소를 포함하고 있는 발명으로 한정함을 내포하는 것으로 해석되어서는 안 되며, 동일한 청구항이 도입구 "하나 이상" 또는 "적어도 하나"와 단수형 부정관사를 포함할 때라도 그렇다. 정관사의 사용에 대해서도 동일하다. 달리 서술되지 않는 한, "제1" 및 "제2"와 같은 용어는 그러한 용어가 기술하는 구성요소들 간 임의로 구별하도록 사용된다. 그리하여, 이들 용어는 반드시 그러한 구성요소의 시간적 또는 다른 우선순위결정을 나타내려는 의도는 아니다. 어떤 조치들이 서로 다른 청구항에서 나열되고 있다는 사실만으로 이들 조치의 조합이 이롭게 사용될 수 없음을 나타내지는 않는다.

도면

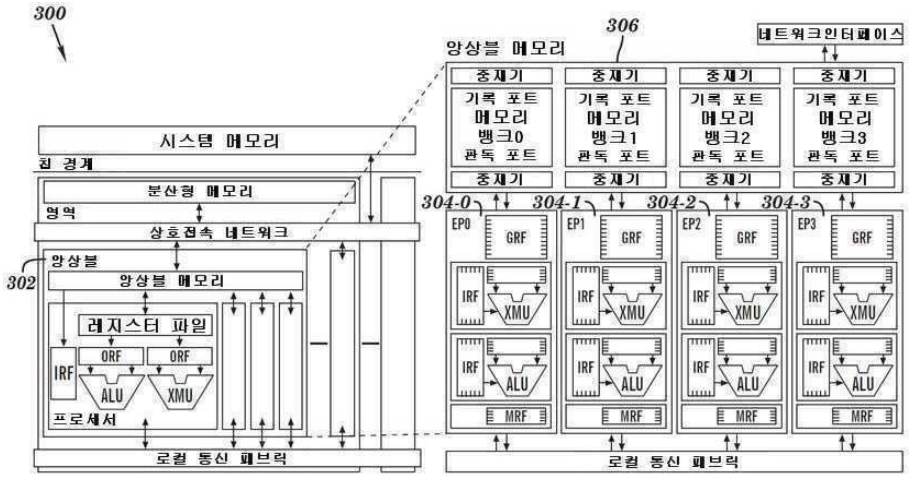
도면1



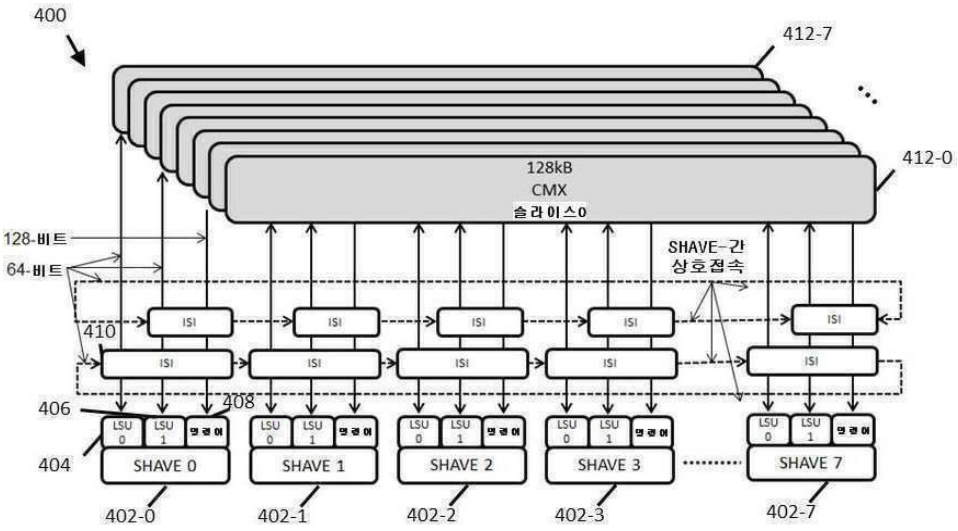
도면2


200
204
202
206
SPE
SXU
LS
DMA
온-칩 코히어런트 버스(사이클당 96 바이트까지)
PPE
L2
L1
파워 코어
메모리 컨트롤러
듀얼 램버스 XDR**
버스 인터페이스 컨트롤러
램버스 FlexIO**

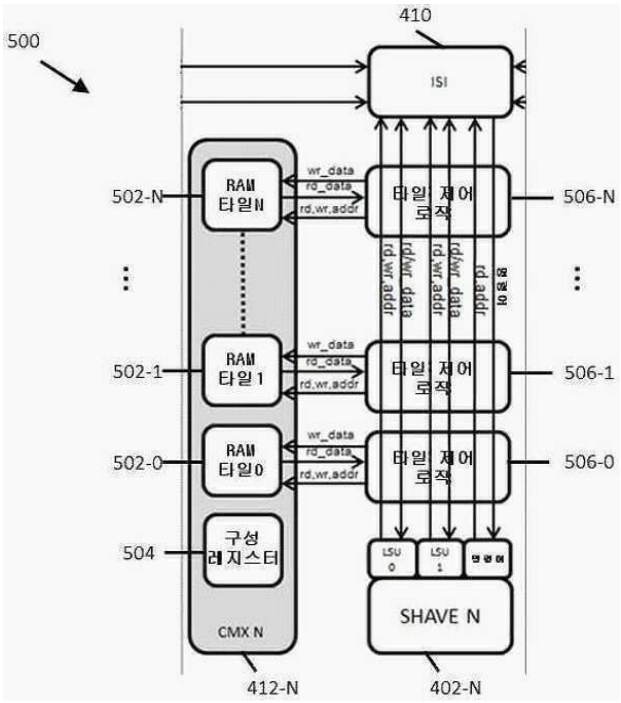
도면3


300
306
앙상블 메모리
네트워크인터페이스
중재기
기록 포트 메모리 뱅크0 관독 포트
기록 포트 메모리 뱅크1 관독 포트
기록 포트 메모리 뱅크2 관독 포트
기록 포트 메모리 뱅크3 관독 포트
304-0
304-1
304-2
304-3
EP0
EP1
EP2
EP3
GRF
IRF
XMU
ALU
MRF
시스템 메모리
칩 경계
분산형 메모리
영역
상호접속 네트워크
302
앙상블
앙상블 메모리
레지스터 파일
ORF
프로세서
로컬 통신 패브릭

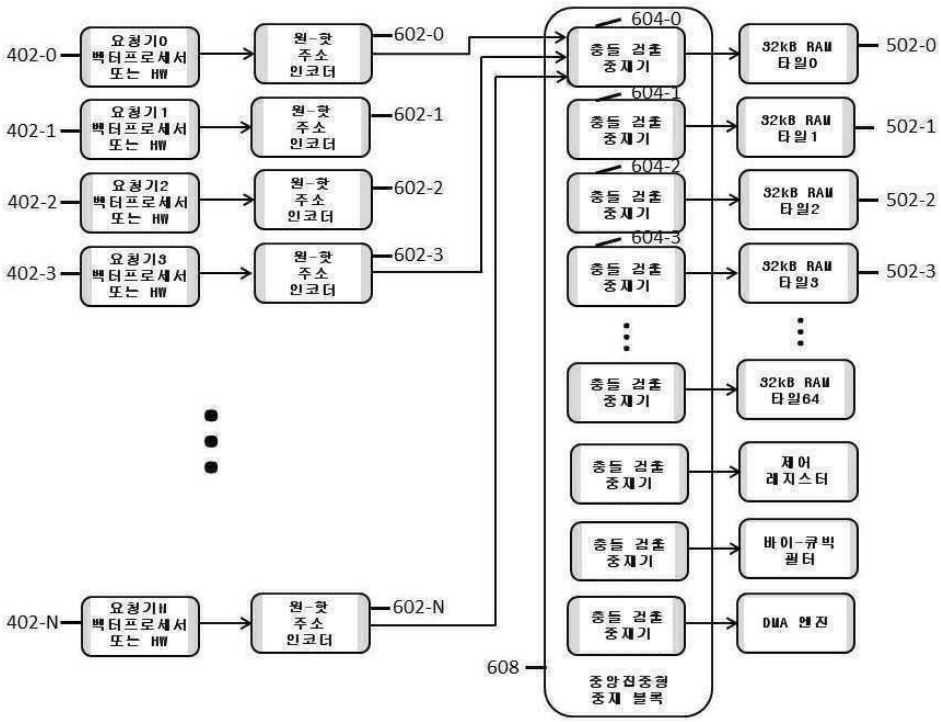
도면4


400
412-7
412-0
128kB CMX 슬라이스0
128-비트
64-비트
SHAVE-간 상호접속
410
ISI
406
408
404
LSU 0
LSU 1
명령어
SHAVE 0
SHAVE 1
SHAVE 2
SHAVE 3
SHAVE 7
402-0
402-1
402-2
402-3
402-7

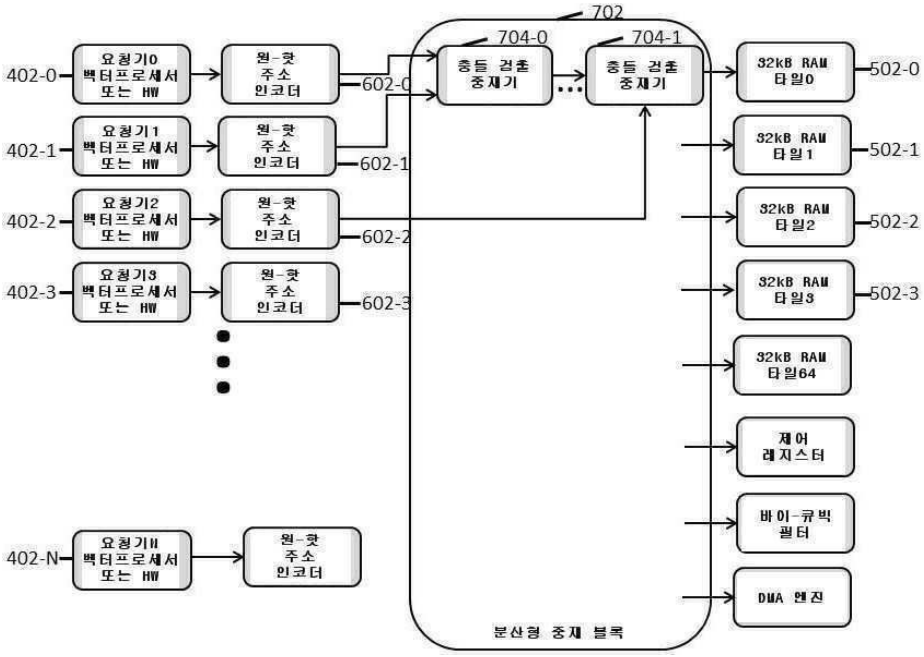
도면5


500
410
ISI
502-N
RAM 타일N
wr_data
rd_data
rd,wr,addr
타일 제어 로직
506-N
rd/wr addr
rd/wr data
rd/wr addr
rd addr
응답
502-1
RAM 타일1
506-1
502-0
RAM 타일0
506-0
504
구성 레지스터
LSU 0
LSU 1
CMX N
SHAVE N
412-N
402-N

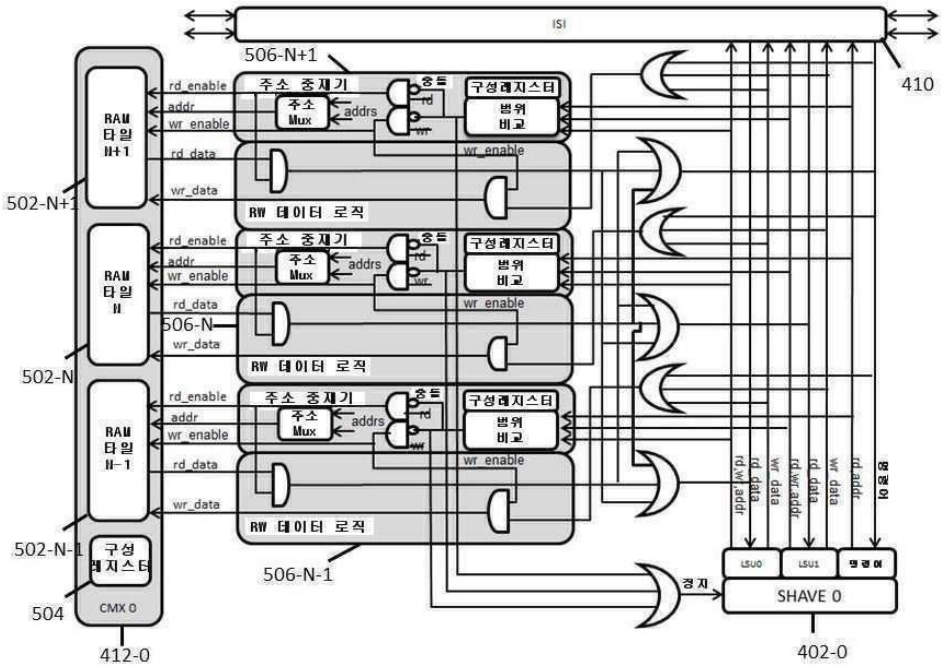
도면6


402-0
요청기0 백터프로세서 또는 HW
원-핫 주소 인코더
602-0
604-0
충돌 검출 중재기
32kB RAM 타일0
502-0
402-1
요청기1 백터프로세서 또는 HW
602-1
604-1
32kB RAM 타일1
502-1
402-2
요청기2 백터프로세서 또는 HW
602-2
604-2
32kB RAM 타일2
502-2
402-3
요청기3 백터프로세서 또는 HW
602-3
604-3
32kB RAM 타일3
502-3
32kB RAM 타일64
제어 레지스터
바이-큐빅 필터
DMA 엔진
402-N
요청기N 백터프로세서 또는 HW
602-N
608
중앙집중형 중재 블록

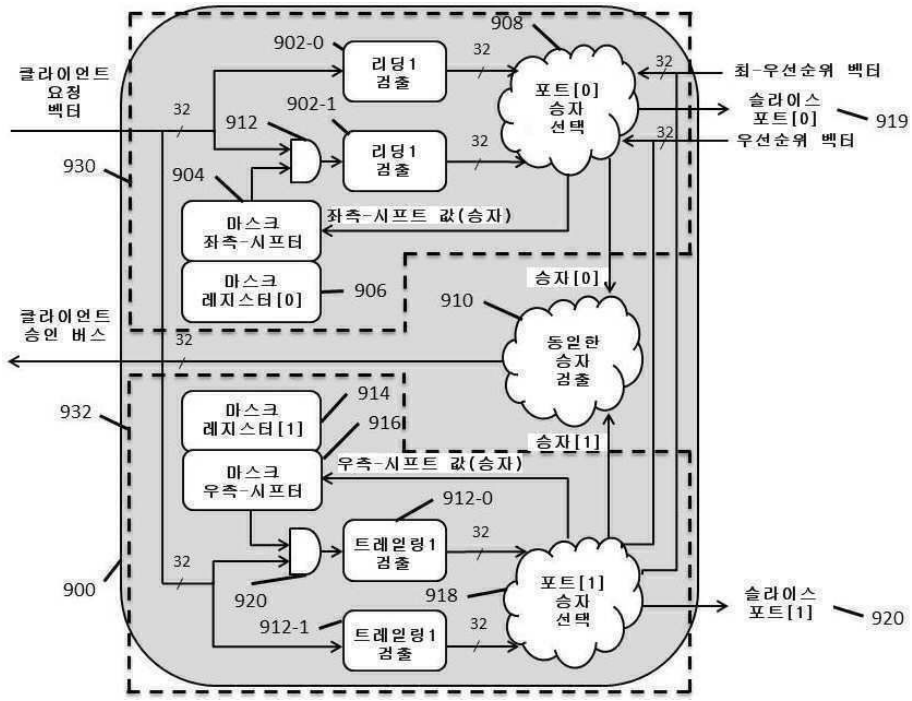
도면7


702
704-0
704-1
402-0
요청기0 벡터프로세서 또는 HW
원-핫 주소 인코더
602-0
충돌 검출 중재기
32kB RAM 타일0
502-0
402-1
요청기1 벡터프로세서 또는 HW
602-1
32kB RAM 타일1
502-1
402-2
요청기2 벡터프로세서 또는 HW
602-2
32kB RAM 타일2
502-2
402-3
요청기3 벡터프로세서 또는 HW
602-3
32kB RAM 타일3
502-3
32kB RAM 타일64
제어 레지스터
바이-큐빅 필터
DMA 엔진
402-N
요청기N 벡터프로세서 또는 HW
분산형 중재 블록

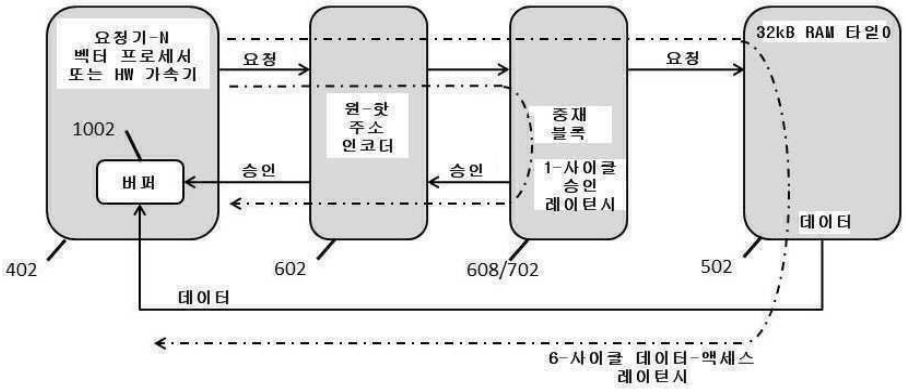
도면8


ISI
410
506-N+1
주소 중재기
주소 Mux
addrs
충돌
구성레지스터
범위 비교
rd_enable
addr
wr_enable
rd_data
wr_data
RW 데이터 로직
RAM 타일 N+1
502-N+1
RAM 타일 N
506-N
502-N
RAM 타일 N-1
506-N-1
502-N-1
구성 레지스터
504
CMX 0
412-0
정지
SHAVE 0
LSU0
LSU1
명령어
402-0

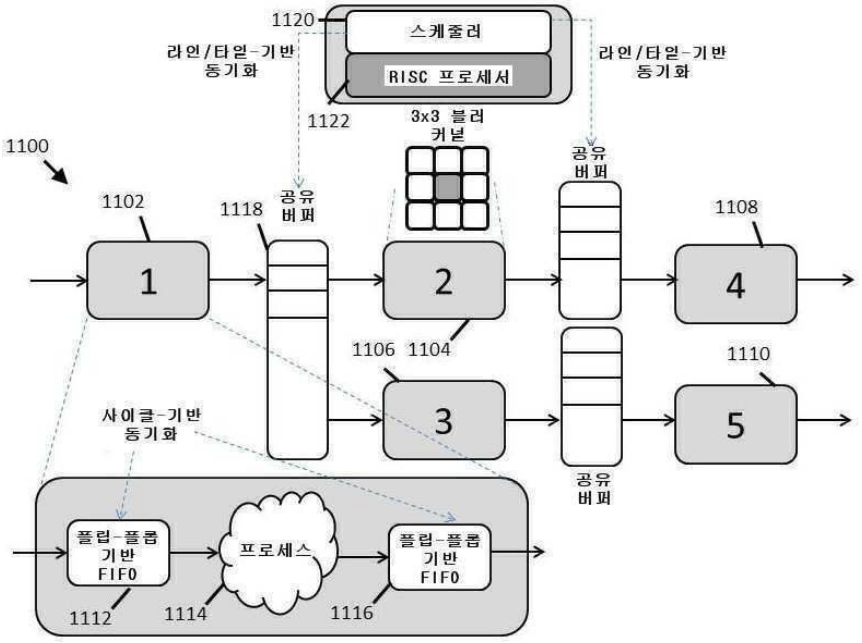
도면9


908
902-0
리딩1 검출
32
포트[0] 승자 선택
최-우선순위 벡터
슬라이스 포트[0]
919
우선순위 벡터
클라이언트 요청 벡터
32
912
902-1
리딩1 검출
930
904
마스크 좌측-시프터
좌측-시프트 값(승자)
마스크 레지스터[0]
906
승자[0]
910
동일한 승자 검출
클라이언트 승인 버스
932
914
916
마스크 레지스터[1]
마스크 우측-시프터
우측-시프트 값(승자)
승자[1]
912-0
트레일링1 검출
900
920
918
912-1
포트[1] 승자 선택
슬라이스 포트[1]
920

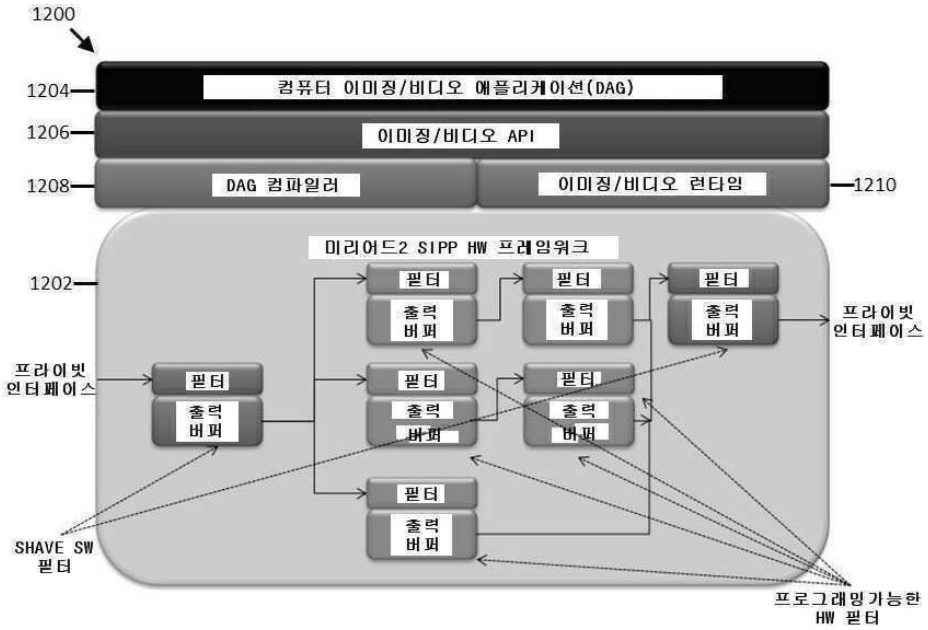
도면10


요청기-N 벡터 프로세서 또는 HW 가속기
요청
원-핫 주소 인코더
중재 블록
1-사이클 승인 레이턴시
32kB RAM 타입0
1002
버퍼
승인
승인
데이터
402
602
608/702
502
데이터
6-사이클 데이터-액세스 레이턴시

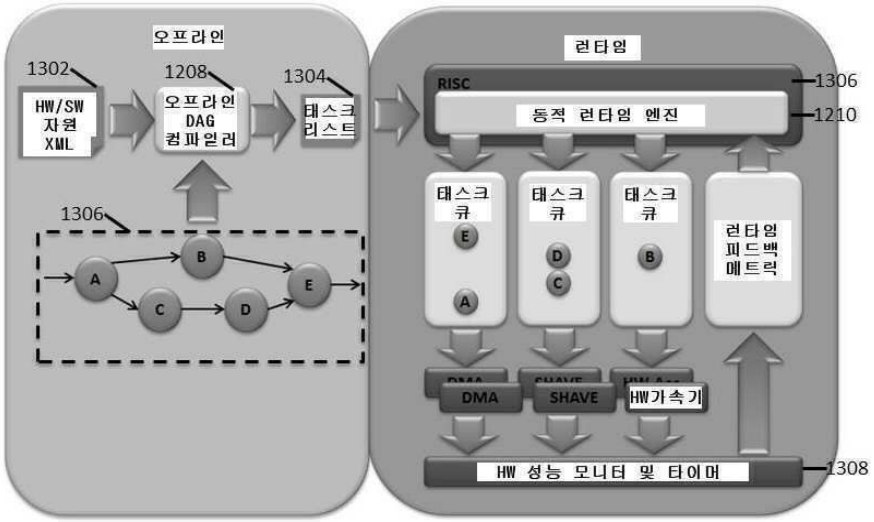
도면11


1120
스케줄러
RISC 프로세서
1122
라인/타일-기반 동기화
라인/타일-기반 동기화
3x3 블러 커널
1100
1102
1118
공유 버퍼
공유 버퍼
1108
1
2
4
1106
1104
1110
3
5
사이클-기반 동기화
공유 버퍼
플립-플롭 기반 FIFO
프로세스
플립-플롭 기반 FIFO
1112
1114
1116

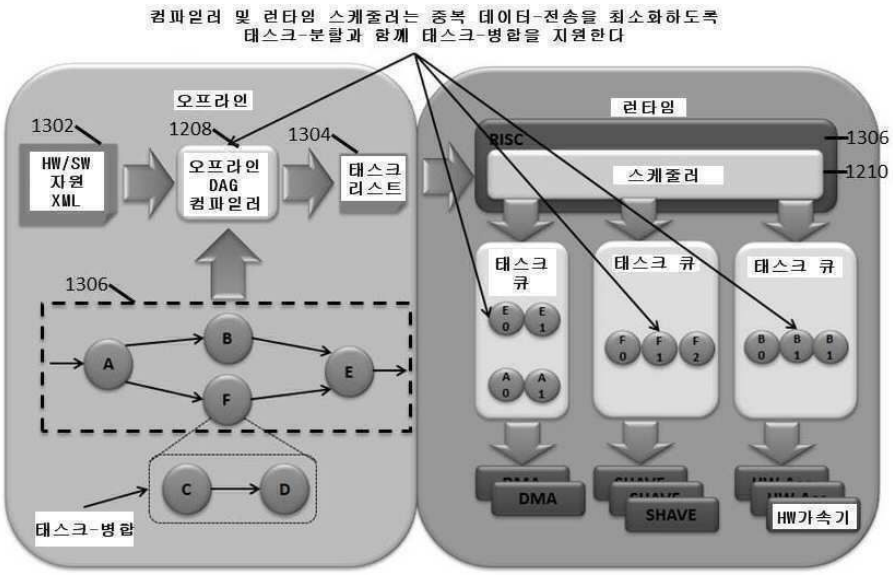
도면12


1200
1204
컴퓨터 이미징/비디오 애플리케이션(DAG)
1206
이미징/비디오 API
1208
DAG 컴파일러
이미징/비디오 런타임
1210
미리어드2 SIPP HW 프레임워크
1202
필터
출력 버퍼
프라이빗 인터페이스
프라이빗 인터페이스
SHAVE SW 필터
프로그래밍가능한 HW 필터

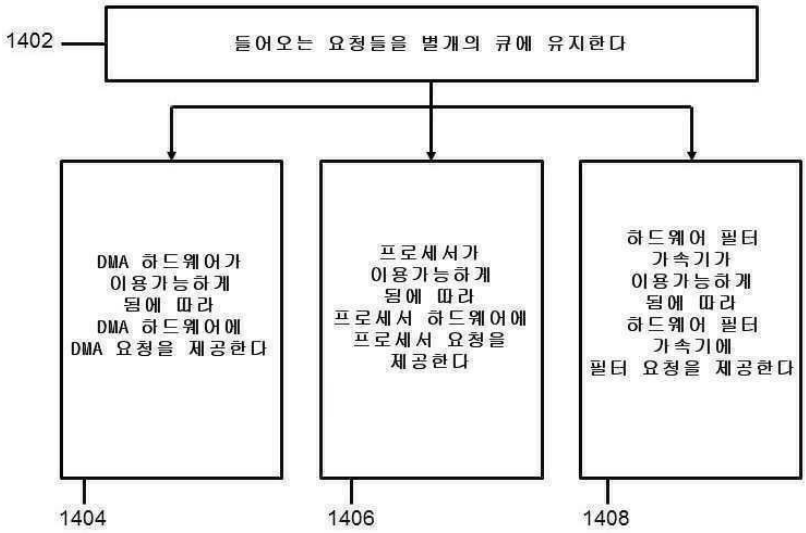
도면13


오프라인
1302
HW/SW 자원 XML
1208
오프라인 DAG 컴파일러
1304
태스크 리스트
1306
A
B
C
D
E
런타임
RISC
1306
동적 런타임 엔진
1210
태스크 큐
E
A
태스크 큐
D
C
태스크 큐
B
런타임 피드백 메트릭
DMA
SHAVE
HW가속기
HW 성능 모니터 및 타이머
1308

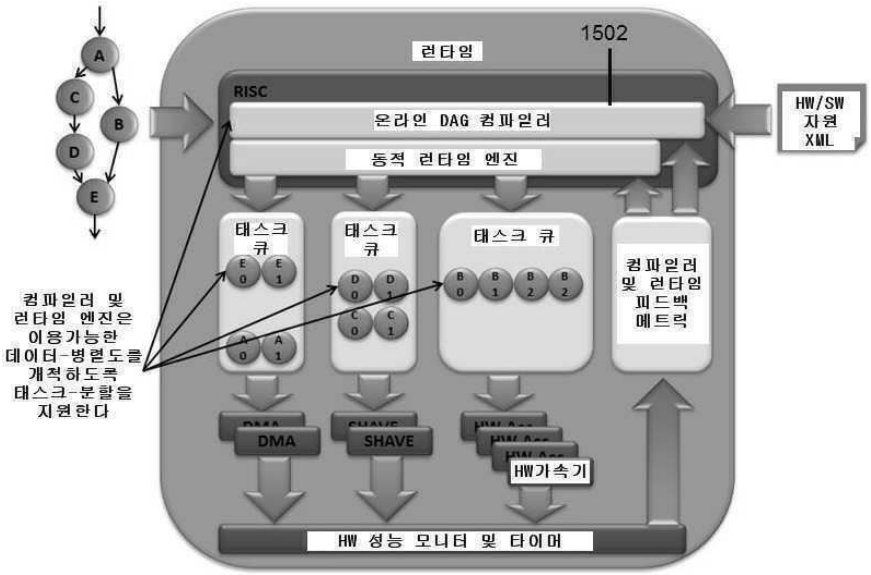
도면14a


컴파일러 및 런타임 스케줄러는 중복 데이터-전송을 최소화하도록
태스크-분할과 함께 태스크-병합을 지원한다
오프라인
1302
HW/SW 자원 XML
1208
오프라인 DAG 컴파일러
1304
태스크 리스트
1306
A
B
C
D
E
F
태스크-병합
런타임
RISC
1306
스케줄러
1210
태스크 큐
태스크 큐
태스크 큐
DMA
SHAVE
HW가속기

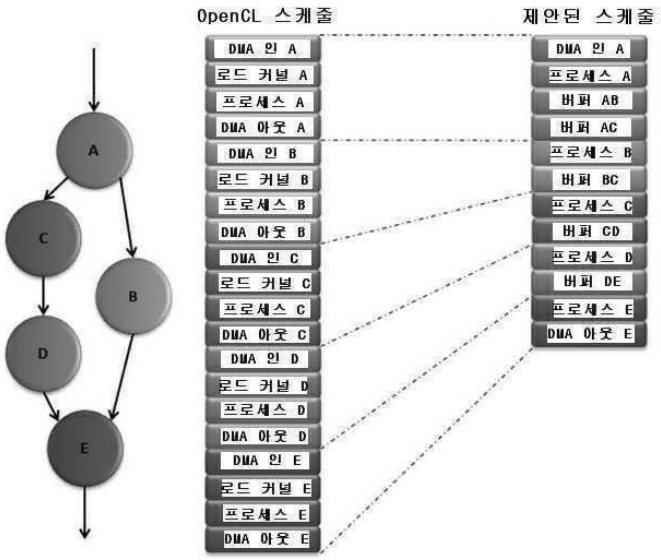
도면14b


1402
들어오는 요청들을 별개의 큐에 유지한다
DMA 하드웨어가 이용가능하게 됨에 따라 DMA 하드웨어에 DMA 요청을 제공한다
프로세서가 이용가능하게 됨에 따라 프로세서 하드웨어에 프로세서 요청을 제공한다
하드웨어 필터 가속기가 이용가능하게 됨에 따라 하드웨어 필터 가속기에 필터 요청을 제공한다
1404
1406
1408

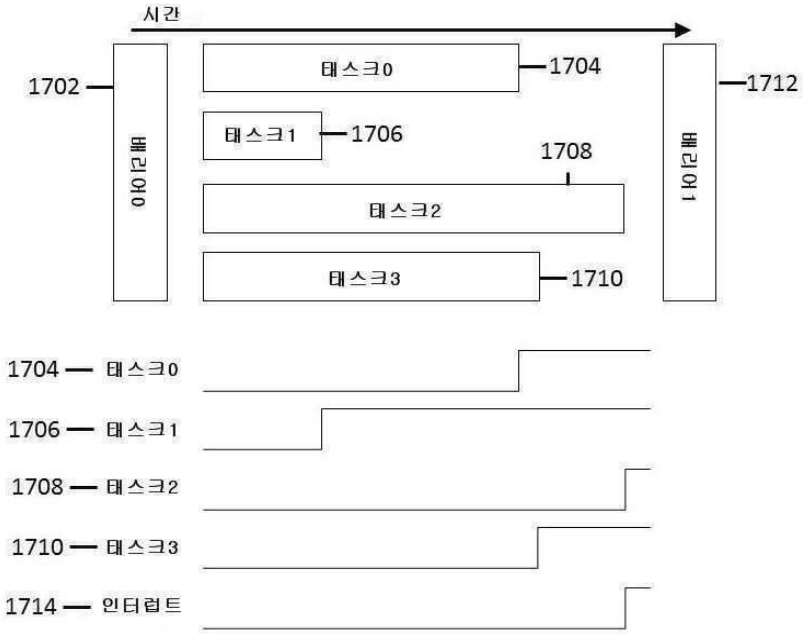
도면15


1502
런타임
RISC
온라인 DAG 컴파일러
동적 런타임 엔진
HW/SW 자원 XML
태스크 큐
태스크 큐
태스크 큐
컴파일러 및 런타임 피드백 메트릭
컴파일러 및 런타임 엔진은 이용가능한 데이터-병렬도를 개척하도록 태스크-분할을 지원한다
DMA
SHAVE
HW가속기
HW 성능 모니터 및 타이머

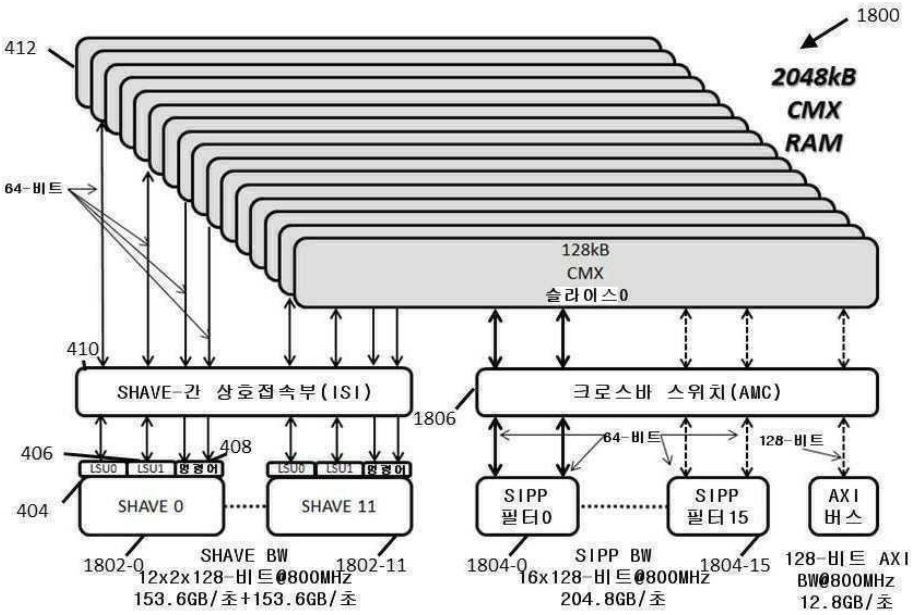
도면16


OpenCL 스케줄
제안된 스케줄
A
B
C
D
E
DMA 인 A
로드 커널 A
프로세스 A
DMA 아웃 A
DMA 인 B
로드 커널 B
프로세스 B
DMA 아웃 B
DMA 인 C
로드 커널 C
프로세스 C
DMA 아웃 C
DMA 인 D
로드 커널 D
프로세스 D
DMA 아웃 D
DMA 인 E
로드 커널 E
프로세스 E
DMA 아웃 E
DMA 인 A
프로세스 A
버퍼 AB
버퍼 AC
프로세스 B
버퍼 BC
프로세스 C
버퍼 CD
프로세스 D
버퍼 DE
프로세스 E
DMA 아웃 E

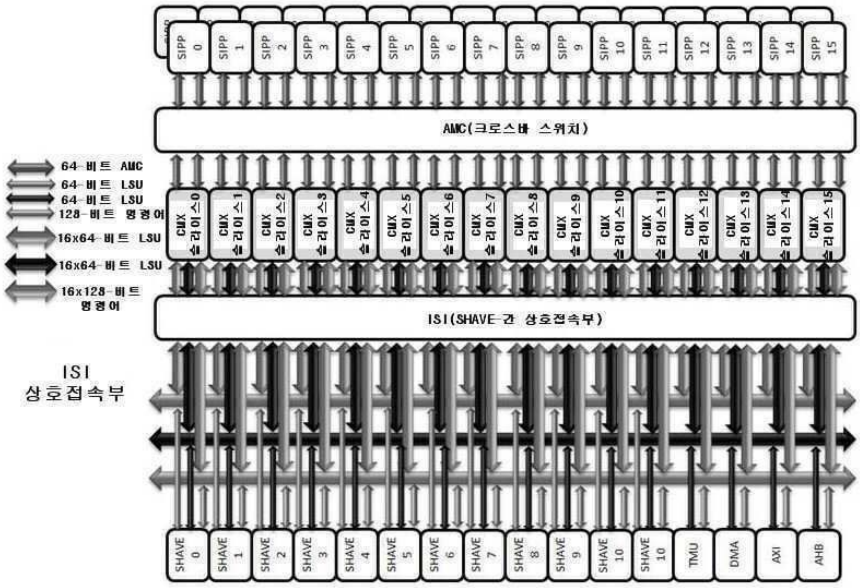
도면17


시간
1702
버퍼 0
태스크0
1704
태스크1
1706
1708
태스크2
태스크3
1710
버퍼 1
1712
1704 태스크0
1706 태스크1
1708 태스크2
1710 태스크3
1714 인터럽트

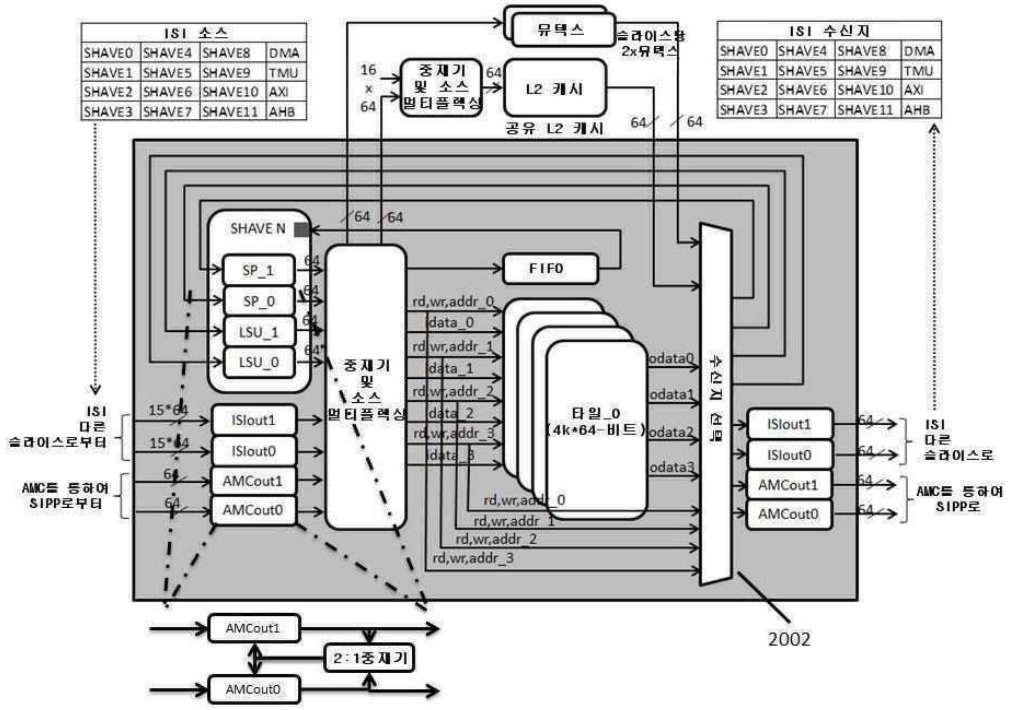
도면18


1800
412
2048kB
CMX
RAM
64-비트
128kB
CMX
슬라이스0
410
SHAVE-간 상호접속부(ISI)
1806
크로스바 스위치(AMC)
406
408
64-비트
128-비트
LSU0
LSU1
명령어
404
SHAVE 0
SHAVE 11
SIPP
필터0
SIPP
필터15
AXI
버스
1802-0
SHAVE BW
12x2x128-비트@800MHz
153.6GB/초+153.6GB/초
1802-11
1804-0
SIPP BW
16x128-비트@800MHz
204.8GB/초
1804-15
128-비트 AXI
BW@800MHz
12.8GB/초

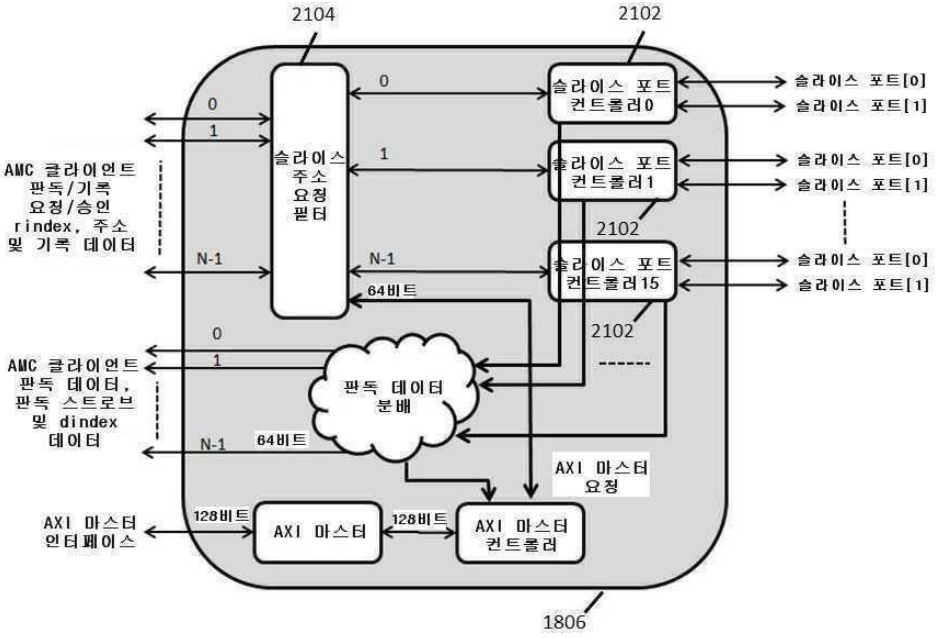
도면19


SIPP 0
SIPP 1
SIPP 2
SIPP 3
SIPP 4
SIPP 5
SIPP 6
SIPP 7
SIPP 8
SIPP 9
SIPP 10
SIPP 11
SIPP 12
SIPP 13
SIPP 14
SIPP 15
AMC(크로스바 스위치)
64-비트 AMC
64-비트 LSU
64-비트 LSU
128-비트 명령어
16x64-비트 LSU
16x64-비트 LSU
16x128-비트 명령어
CMX 슬라이스0
CMX 슬라이스1
CMX 슬라이스2
CMX 슬라이스3
CMX 슬라이스4
CMX 슬라이스5
CMX 슬라이스6
CMX 슬라이스7
CMX 슬라이스8
CMX 슬라이스9
CMX 슬라이스10
CMX 슬라이스11
CMX 슬라이스12
CMX 슬라이스13
CMX 슬라이스14
CMX 슬라이스15
ISI(SHAVE-간 상호접속부)
ISI
상호접속부
SHAVE 0
SHAVE 1
SHAVE 2
SHAVE 3
SHAVE 4
SHAVE 5
SHAVE 6
SHAVE 7
SHAVE 8
SHAVE 9
SHAVE 10
SHAVE 11
TMU
DMA
AXI
AHB

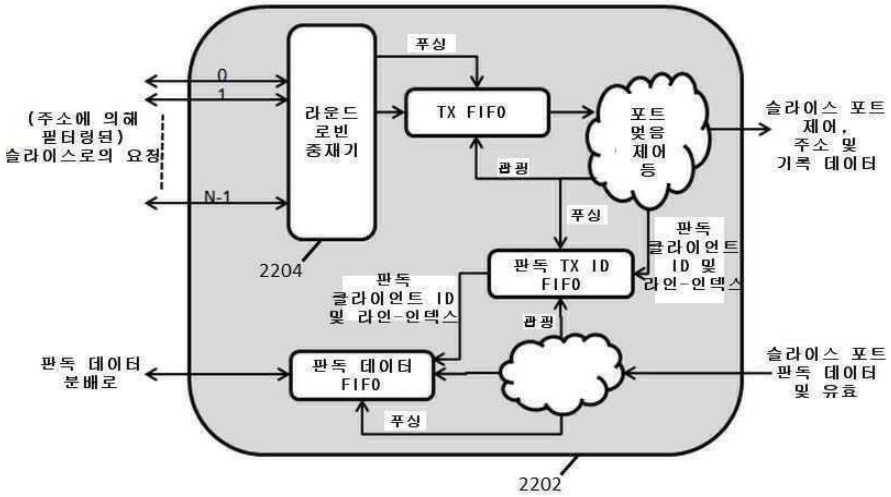
도면20


ISI 소스
SHAVE0 SHAVE4 SHAVE8 DMA
SHAVE1 SHAVE5 SHAVE9 TMU
SHAVE2 SHAVE6 SHAVE10 AXI
SHAVE3 SHAVE7 SHAVE11 AHB
뮤텍스
슬라이스당 2x뮤텍스
중재기 및 소스 멀티플렉싱
L2 캐시
공유 L2 캐시
ISI 수신지
SHAVE0 SHAVE4 SHAVE8 DMA
SHAVE1 SHAVE5 SHAVE9 TMU
SHAVE2 SHAVE6 SHAVE10 AXI
SHAVE3 SHAVE7 SHAVE11 AHB
SHAVE N
SP_1
SP_0
LSU_1
LSU_0
FIFO
중재기 및 소스 멀티플렉싱
타일_0 (4k*64-비트)
수신지 선택
ISI 다른 슬라이스로부터
AMC를 통하여 SIPP로부터
ISIout1
ISIout0
AMCout1
AMCout0
ISI 다른 슬라이스로
AMC를 통하여 SIPP로
2:1중재기
2002

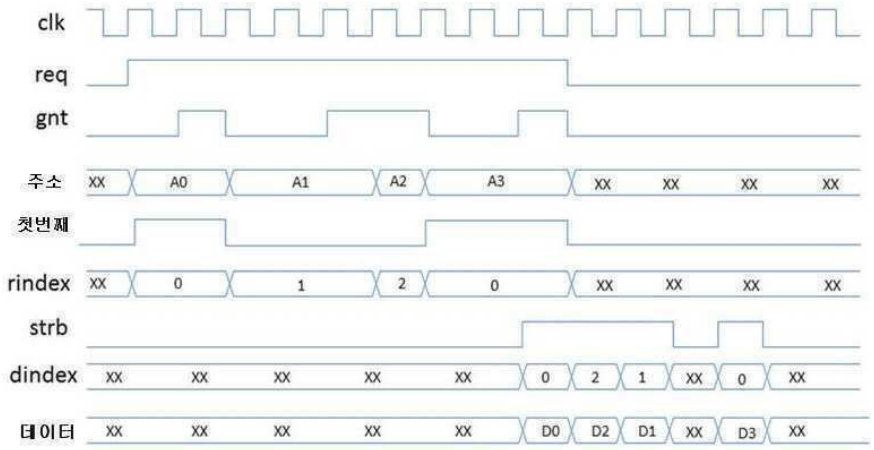
도면21


2104
2102
슬라이스 주소 요청 필터
슬라이스 포트 컨트롤러0
슬라이스 포트 컨트롤러1
슬라이스 포트 컨트롤러15
슬라이스 포트[0]
슬라이스 포트[1]
AMC 클라이언트 판독/기록 요청/승인 rindex, 주소 및 기록 데이터
AMC 클라이언트 판독 데이터, 판독 스트로브 및 dindex 데이터
판독 데이터 분배
64비트
AXI 마스터 요청
AXI 마스터 인터페이스
128비트
AXI 마스터
AXI 마스터 컨트롤러
1806

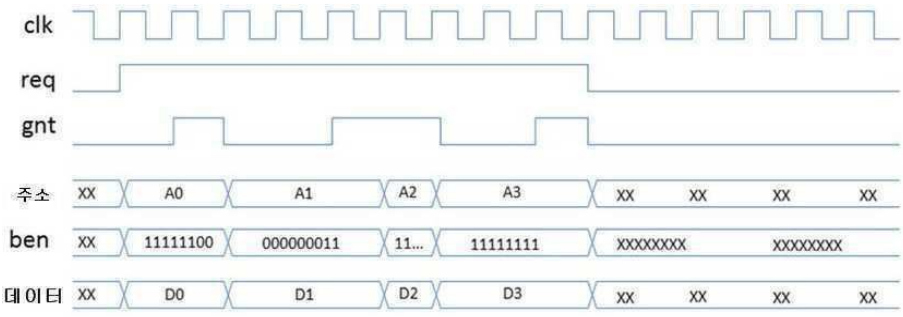
도면22



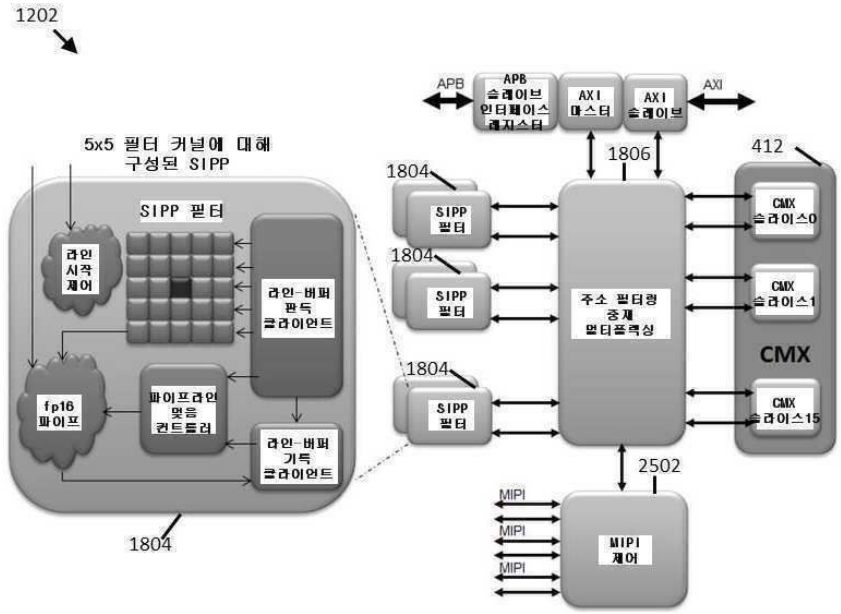
도면23



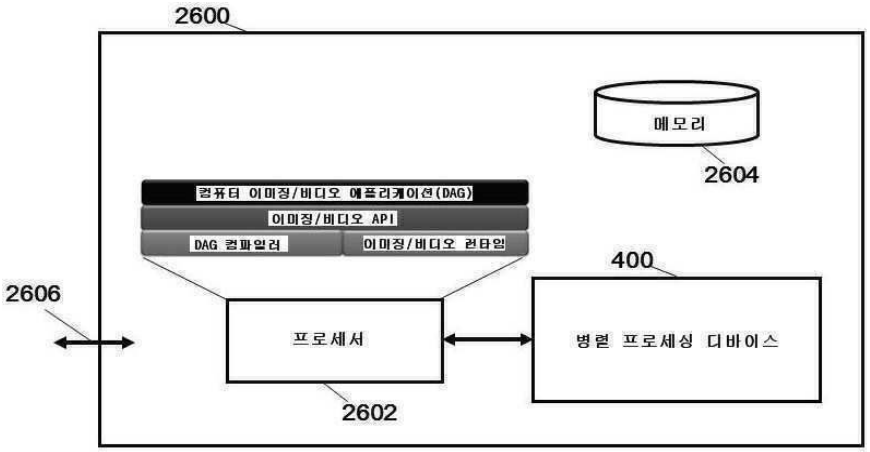
도면24



도면25


1202
5x5 필터 커널에 대해
구성된 SIPP
SIPP 필터
라인
시작
제어
라인-버퍼
판독
클라이언트
fp16
파이프
파이프라인
멈춤
컨트롤러
라인-버퍼
기록
클라이언트
1804
APB
APB
슬레이브
인터페이스
레지스터
AXI
마스터
AXI
슬레이브
AXI
1806
412
1804
SIPP
필터
1804
SIPP
필터
1804
SIPP
필터
주소 필터링
중재
멀티플렉싱
CMX
슬라이스0
CMX
슬라이스1
CMX
CMX
슬라이스15
2502
MIPI
MIPI
MIPI
MIPI
제어

도면26


2600
메모리
2604
컴퓨터 이미징/비디오 애플리케이션(DAG)
이미징/비디오 API
DAG 컴파일러
이미징/비디오 런타임
400
2606
프로세서
병렬 프로세싱 디바이스
2602