(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2015년12월04일
(11) 등록번호 10-1574278
(24) 등록일자 2015년11월27일

(51) 국제특허분류(Int. Cl.)
G06F 13/42 (2006.01) *G06F 13/16* (2006.01)
(21) 출원번호 10-2010-7024483
(22) 출원일자(국제) 2009년03월27일
심사청구일자 2014년03월26일
(85) 번역문제출일자 2010년10월29일
(65) 공개번호 10-2010-0135283
(43) 공개일자 2010년12월24일
(86) 국제출원번호 PCT/US2009/038637
(87) 국제공개번호 WO 2009/123933
국제공개일자 2009년10월08일
(30) 우선권주장
61/041,210 2008년03월31일 미국(US)
(56) 선행기술조사문헌
US06956579 B1

(73) 특허권자
어드밴스드 마이크로 디바이시즈, 인코포레이티드
미국 캘리포니아 94088-3453 서니베일 피.오.박스 3453 원 에이엠디 플레이스
(72) 발명자
모레인 스테펜
미국 캘리포니아 94088 서니베일 원 에이엠디 플레이스
그로스만 마르크 에스.
미국 캘리포니아 94088 서니베일 원 에이엠디 플레이스
(뒷면에 계속)
(74) 대리인
박장원

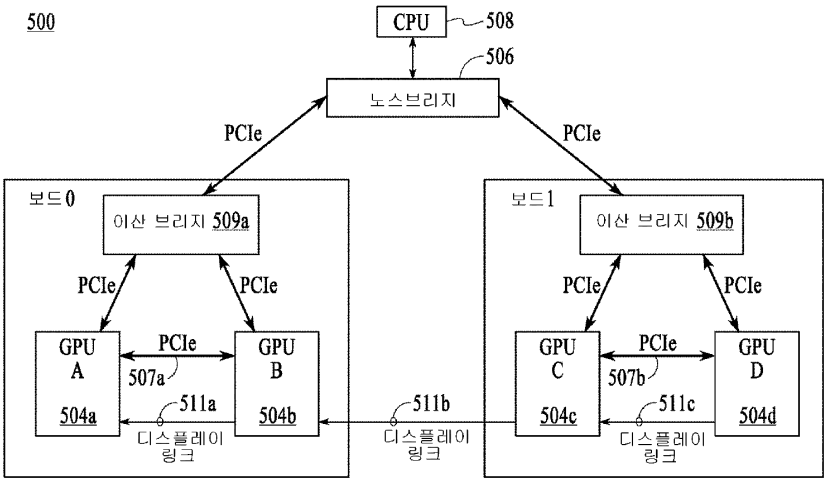
전체 청구항 수 : 총 21 항 심사관 : 고재용

(54) 발명의 명칭 **피어투피어 특수 목적 프로세서 아키텍처 및 방법**

(57) 요 약

피어투피어 특수 목적 프로세서 아키텍처 및 방법이 설명되고, 실시예들은 호스트 브리지 버스를 통해 중앙 처리 유닛에 결합된 복수의 특수 목적 프로세서들과, 상기 복수의 특수 목적 프로세서들 각각을 상기 복수의 특수 목적 프로세서들 중 적어도 하나의 다른 특수 목적 프로세서에 직접 결합시키는 다이렉트 버스와, 그리고 상기 복수의 특수 목적 프로세서들에 결합된 메모리 제어기를 포함하고, 적어도 하나의 상기 메모리 제어기가 데이터를 상기 호스트 버스를 통해 전송할지 또는 상기 다이렉트 버스를 통해 전송할지 여부 및 데이터를 상기 호스트 버스를 통해 수신할지 또는 상기 다이렉트 버스를 통해 수신할지 여부를 결정한다.

대 표 도



(72) 발명자

쿠루거 와렌 프리츠

미국 캘리포니아 94088 서니베일 원 에이엠디 플레이스

에트쉐이드 브라언

미국 캘리포니아 94088 서니베일 원 에이엠디 플레이스

명 세 서

청구범위

청구항 1

시스템으로서,

복수의 특수 목적 프로세서들과;

상기 복수의 특수 목적 프로세서들 각각을 브리지를 통해 상기 복수의 특수 목적 프로세서들 중 적어도 하나의 다른 특수 목적 프로세서에 결합시키는 호스트 브리지 버스와;

상기 복수의 특수 목적 프로세서들 각각을 상기 복수의 특수 목적 프로세서들 중 적어도 하나의 다른 특수 목적 프로세서에 직접 결합시키는 다이렉트 버스와;

적어도 하나의 로컬 메모리와;

적어도 하나의 시스템 메모리와;

적어도 하나의 피어 메모리(peer memory)와; 그리고

상기 복수의 특수 목적 프로세서들 각각에 결합된 메모리 제어기를 포함하여 구성되며,

상기 메모리 제어기는,

데이터를 상기 호스트 브리지 버스를 통해 전송할지 아니면 상기 다이렉트 버스를 통해 전송할지를 결정하도록 되어 있음과 아울러,

상기 메모리들 각각의 어드레스 범위에 근거하여 어떤 메모리에 액세스할지를 결정하도록 되어 있고,

상기 메모리 제어기는 또한,

상기 피어 메모리의 범위에 어드레스가 있을 때, 상기 호스트 브리지 버스 및 상기 다이렉트 버스 각각상의 트래픽의 양에 근거하여 어떤 버스를 사용할지를 결정하도록 되어 있는 것을 특징으로 하는 시스템.

청구항 2

제1항에 있어서,

상기 메모리 제어기는 상기 복수의 특수 목적 프로세서들 중 적어도 하나에 상주하는 것을 특징으로 하는 시스템.

청구항 3

제1항에 있어서,

상기 복수의 특수 목적 프로세서들은 그래픽 프로세싱 유닛(Graphics Processing Unit, GPU)들을 포함하는 것을 특징으로 하는 시스템.

청구항 4

제1항에 있어서,

상기 호스트 브리지 버스는 주변 컴포넌트 인터페이스 익스프레스(Peripheral Component Interface express, PCIe) 버스를 포함하는 것을 특징으로 하는 시스템.

청구항 5

제1항에 있어서,

상기 다이렉트 버스는 주변 컴포넌트 인터페이스 익스프레스(PCIe) 버스를 포함하는 것을 특징으로 하는 시스템.

청구항 6

제1항에 있어서,

상기 다이렉트 버스는 독점 버스(proprietary bus)를 포함하는 것을 특징으로 하는 시스템.

청구항 7

삭제

청구항 8

삭제

청구항 9

제1항에 있어서,

상기 메모리 제어기는 또한, 상기 피어 메모리의 범위에 어드레스가 있을 때, 상기 호스트 브리지 버스와 상기 다이렉트 버스 중 하나가 현재 가득 차 있는지 여부에 근거하여 어떤 버스를 사용할지를 결정하도록 되어 있는 것을 특징으로 하는 시스템.

청구항 10

제1항에 있어서,

상기 메모리 제어기는 또한, 상기 피어 메모리의 범위에 어드레스가 있을 때, 상기 호스트 브리지 버스 또는 상기 다이렉트 버스로의 고정된 맵핑(fixed mapping)에 근거하여 어떤 버스를 사용할지를 결정하도록 되어 있는 것을 특징으로 하는 시스템.

청구항 11

제1항에 있어서,

상기 메모리 제어기는 또한, 상기 호스트 브리지 버스 및 상기 다이렉트 버스 각각 상의 트래픽에 근거하여 다이나믹 로드 밸런싱(dynamic load balancing)을 수행하는 것을 특징으로 하는 시스템.

청구항 12

멀티-프로세서 시스템에서 데이터를 전달하는 방법으로서,

데이터를 호스트 프로세서 및 복수의 특수 목적 프로세서들에 의해 공유된 호스트 브리지 버스를 통해 전송 혹은 수신할 것인지 아니면 상기 복수의 특수 목적 프로세서들을 직접 결합시키는 다이렉트 버스를 통해 전송 혹은 수신할 것인지를 결정하는 단계를 포함하여 구성되고, 상기 결정하는 단계는 전송의 메모리 어드레스 범위에 의존하며, 상기 결정하는 단계는, 피어 메모리를 특정하는 범위 내에 어드레스가 있을 때, 상기 호스트 브리지 버스와 상기 다이렉트 버스 각각 상의 트래픽의 양에 근거하여 수행되는 것을 특징으로 하는 데이터 전달 방법.

청구항 13

제12항에 있어서,

상기 어드레스 범위들은 특수 목적 프로세서 상에 상주하는 피어 메모리, 상기 호스트 브리지 버스 상의 시스템 메모리, 및 상기 특수 목적 프로세서들 간에 결합된 로컬 메모리 중 하나를 특정하는 것을 특징으로 하는 데이터 전달 방법.

청구항 14

제13항에 있어서,

상기 피어 메모리를 특정하는 범위 내의 특정 어드레스들이 상기 호스트 브리지 버스 혹은 상기 다이렉트 버스 중 하나 상으로 할당되는 것을 특징으로 하는 데이터 전달 방법.

청구항 15

삭제

청구항 16

제13항에 있어서,

이력 트래픽 데이터(historical traffic data)에 근거하여 상기 호스트 브리지 버스와 상기 다이렉트 버스 각각 상에서의 트래픽 비율(percentage traffic)을 조정하기 위해 다이나믹 로드 밸런싱을 수행하는 단계를 더 포함하는 것을 특징으로 하는 데이터 전달 방법.

청구항 17

제13항에 있어서,

메모리 애퍼쳐(memory aperture)들을 상기 호스트 프로세서와, 그리고 상기 복수의 특수 목적 프로세서들 각각에 할당하는 단계를 더 포함하고, 상기 메모리 애퍼쳐들은 상기 피어 메모리, 상기 시스템 메모리, 및 상기 로컬 메모리에 적용되고, 그리고 상기 애퍼쳐는 각각의 프로세서가 메모리를 어떻게 뷰(view)할지를 결정하는 것을 특징으로 하는 데이터 전달 방법.

청구항 18

제17항에 있어서,

애퍼쳐 콤프레션(aperture compression)을 수행하는 단계를 더 포함하며, 상기 애퍼쳐 콤프레션을 수행하는 단계는, 오프셋(offset)을 변경시킴으로써 상기 피어 메모리 모두가 가시적(visible)이게 만들기 위해 오프셋 레지스터를 사용하는 것을 포함하는 것을 특징으로 하는 데이터 전달 방법.

청구항 19

삭제

청구항 20

컴퓨터 판독가능 매체로서,

상기 컴퓨터 판독가능 매체 상에는 명령들이 저장되어 있고, 상기 명령들은, 실행될 때, 멀티-프로세서 시스템에서 데이터를 전달하는 방법이 수행되도록 하며, 상기 방법은,

데이터를 호스트 프로세서 및 복수의 특수 목적 프로세서들에 의해 공유된 호스트 브리지 버스를 통해 전송 혹은 수신할 것인지 아니면 상기 복수의 특수 목적 프로세서들을 직접 결합시키는 다이렉트 버스를 통해 전송 혹은 수신할 것인지를 결정하는 단계를 포함하며, 상기 결정하는 단계는 전송의 메모리 어드레스 범위에 의존하며, 상기 어드레스 범위들은 특수 목적 프로세서 상에 상주하는 피어 메모리, 상기 호스트 브리지 버스 상의 시스템 메모리, 및 상기 특수 목적 프로세서에 결합된 로컬 메모리 중 하나를 특정하고, 상기 결정하는 단계는, 상기 피어 메모리를 특정하는 범위 내에 어드레스가 있을 때, 상기 호스트 브리지 버스와 상기 다이렉트 버스 각각 상의 트래픽의 양에 근거하여 수행되는 것을 특징으로 하는 컴퓨터 판독가능 매체.

청구항 21

삭제

청구항 22

제20항에 있어서,

상기 피어 메모리를 특정하는 범위 내의 특정 어드레스들이 상기 호스트 브리지 버스 혹은 상기 다이렉트 버스 중 하나 상으로 할당되는 것을 특징으로 하는 컴퓨터 판독가능 매체.

청구항 23

삭제

청구항 24

제20항에 있어서,

상기 방법은, 이력 트래픽 데이터에 근거하여 상기 호스트 브리지 버스와 상기 다이렉트 버스 각각 상에서의 트래픽 비율을 조정하기 위해 다이나믹 로드 밸런싱을 수행하는 단계를 더 포함하는 것을 특징으로 하는 컴퓨터 판독가능 매체.

청구항 25

제20항에 있어서,

상기 방법은, 메모리 애퍼쳐들을 상기 호스트 프로세서와, 그리고 상기 복수의 특수 목적 프로세서들 각각에 할당하는 단계를 더 포함하고, 상기 메모리 애퍼쳐들은 상기 피어 메모리, 상기 시스템 메모리, 및 상기 로컬 메모리에 적용되고, 그리고 상기 애퍼쳐는 각각의 프로세서가 메모리를 어떻게 뷰할지를 결정하는 것을 특징으로 하는 컴퓨터 판독가능 매체.

청구항 26

제25항에 있어서,

상기 방법은, 애퍼쳐 콤프레션을 수행하는 단계를 더 포함하며, 상기 애퍼쳐 콤프레션을 수행하는 단계는, 오프셋을 변경시킴으로써 상기 로컬 메모리 모두가 가시적이게 만들기 위해 오프셋 레지스터를 사용하는 것을 포함하는 것을 특징으로 하는 컴퓨터 판독가능 매체.

청구항 27

제20항에 있어서,

상기 명령들은, 상기 방법을 수행하는 애플리케이션 특정 집적 회로(Application Specific Integrated Circuit, ASIC)를 생성하는데 사용가능한 하드웨어 기술 언어 명령들을 포함하는 것을 특징으로 하는 컴퓨터 판독가능 매체.

청구항 28

삭제

발명의 설명

기 술 분 야

[0001] 본 발명은 컴퓨터 및 다른 디지털 시스템에서의 데이터 전달에 관한 것이다.

배 경 기 술

[0002] 본 출원은 미국 가특허출원 번호 제61/041,210호(2008년 3월 31일 출원)에 대한 우선권을 주장하며, 이 가특허출원은 그 전체가 참조로 본 명세서에 통합된다.

[0003] 컴퓨터 및 다른 디지털 시스템들이 더 복잡해지고 더 성능이 향상됨에 따라, 시스템 컴포넌트들 혹은 소자들 간의 데이터 전달 능력을 강화시키는 방법 및 하드웨어가 계속 발전하고 있다. 전달될 데이터에는, 데이터를 나타내는 신호들, 커맨드들, 혹은 임의의 다른 신호들이 있다. 데이터 전달의 속도 및 효율은, 그래픽 애플리케이션과 같은 데이터 집중도가 매우 큰 애플리케이션을 실행하는 시스템에서 특히 중요하다. 전형적인 시스템에서, 그래픽 프로세싱 능력은 중앙 처리 유닛(Central Processing Unit, CPU) 능력의 일부로서 제공되거나, 혹은 CPU와 통신하여 비디오 게임 등과 같은 애플리케이션에 대한 그래픽 데이터의 프로세싱을 보조하는 그래픽 프로세싱 유닛(Graphics Processing Unit, GPU)과 같은 개별의 특수 목적 프로세서에 의해 제공된다. 하나 또는 그 이상의 GPU들이 시스템에 포함될 수 있다. 종래 멀티 GPU 시스템(multi-GPU system)들에서, 브리지된 호스트 인터페이스(예를 들어, 주변 컴포넌트 인터페이스 익스프레스(Peripheral Component Interface express(PCIe®))는 호스트 트래픽(host traffic)과 대역폭을 공유해야만 한다. 도 1은 종래 기술의 시스템(100)을 나타낸 블럭도이

고, 이 시스템(100)은 루트 프로세서(root processor)(108)(예를 들어, CPU), 브리지(106), 및 두 개의 엔드포인트(endpoint)들인, EP0(104a) 및 EP1(104b)(예를 들어, GPU0 및 GPU1)을 포함한다. 각각의 엔드포인트(104)는 각각의 메모리 컴포넌트(102a 및 102b)에 결합된다.

[0004] 시스템에서의 하나 이상의 GPU의 존재를 이용하기 위해, 복수의 GPU들은 이들 간의 데이터의 전달을 요구하는 프로세싱 태스크들을 공유한다. 멀티 GPU 시스템에서의 양호한 성능 스케일링을 달성함에 있어 고려할 사항은 GPU들 간에 주고 받아야만 하는 데이터가 잠재적으로 많다는 것이다. 이러한 예시적 상황에서, 애플리케이션은 도로잉(drawing)에 많은 시간을 필요로 하는 백그라운드(background) 혹은 어떤 다른 텍스처(texture)의 드로잉을 요구한다. 이 백그라운드 혹은 다른 텍스처는, 그 다음 프레임 혹은 동일 프레임의 또 다른 부분에서 사용될 수 있도록, 또 다른 GPU로 복사되는 것이 바람직하다. 시스템(100)에서, GPU들(104) 간의 데이터 전달을 위한 유일한 경로는 브리지(106)까지의 각각의 경로(105a 및 105b)를 통하는 것이다. 루트(108)로의 경로는 각각의 GPU(104)로부터의 트래픽을 루트(108)로 운반해야하며, 뿐만 아니라 두 개의 GPU들(104) 간의 트래픽도 운반해야만 한다. 이것은 병목 현상을 일으킬 수 있는데, 왜냐하면 물리적 와이어(wire)들이 공유되어 있기 때문이다.

도면의 간단한 설명

[0005] 도 1은 종래 기술의 멀티-프로세서 시스템의 블럭도이다.

도 2는 일 실시예에 따른, 피어투피어 IO 경로가 부가된 멀티-프로세서 시스템의 부분들의 블럭도이다.

도 3은 일 실시예에 따른, 피어투피어 IO 경로가 부가된 멀티-프로세서 시스템의 보다 상세한 블럭도이다.

도 4는 일 실시예에 따른, IO 시스템에 의해 수행되는 방법의 플로우 다이어그램(flow diagram)이다.

도 5는 일 실시예에 따른, 비대칭 IO 시스템에서의 단계 정보(phase information)를 전달하는 방법을 나타낸 플로우차트(flowchart)이다.

도 6은 일 실시예에 따른, 비대칭 IO 시스템에서 클라이언트 단계 정보를 호스트 디바이스에 전달하는 것을 나타낸 타이밍 다이어그램(timing diagram)이다.

발명을 실시하기 위한 구체적인 내용

[0006] 피어투피어 특수 목적 프로세서 아키텍처 및 방법의 실시예들이 본 명세서에서 설명된다. 실시예들은 호스트 브리지 버스(host bridge bus)를 통해 적어도 하나의 중앙 처리 유닛에 결합된 복수의 특수 목적 프로세서들을 구비한 시스템을 포함한다. 다이렉트 버스(direct bus)는 상기 복수의 특수 목적 프로세서들 각각을 상기 복수의 특수 목적 프로세서들 중 적어도 하나의 다른 특수 목적 프로세서에 직접 결합시킨다. 메모리 제어기가 상기 복수의 특수 목적 프로세서들에 결합되고, 그리고 상기 메모리 제어기는 데이터를 호스트 버스를 통해 전송할지 또는 다이렉트 버스를 통해 전송할지 여부를 결정한다.

[0007] 도 2는 일 실시예에 따른 멀티 프로세서 시스템(200)의 부분들을 나타낸 블럭도이다. 이 시스템(200)은 브리지(206)에 결합된 루트 프로세서(root processor)(208)를 포함한다. 이 브리지(206)는 각각의 버스들(205a 및 205b)을 통해 버스 엔드포인트(bus endpoint)들인, EP0(204a) 및 EP1(204b)에 결합된다. 엔드포인트들(204) 각각은 각각의 메모리 디바이스(202a 혹은 202b)에 결합된다.

[0008] 일 실시예에서, 루트 프로세서(208)는 CPU이고, 엔드포인트들(204) 각각은 GPU들이지만, 실시예들이 이러한 것으로만 한정되는 것은 아니다. 아래에서 더 설명되는 바와 같이, 실시예들에서, 엔드포인트(204)는 PCIe® 포트들과 유사한 사이드포트(sideport)들을 구비한 각각의 GPU들이다. PCIe® 어드레싱 특징들, 및 다양한 로드 밸런싱 특징(load balancing feature)들이 성능 개선을 위해 시스템(200)의 아키텍처에서 사용된다. 전형적인 아키텍처들에서, 하나의 GPU로부터 또 다른 하나의 GPU로의 판독 동작은 특히 비용이 많이 든다. 판독 동작을 위해, 요청이 전송되고, 요청된 데이터가 일정 시간 이후에 수신되어 획득된다. 실시예들은 판독 동작을 포함하는 동작들에 대한 전형적인 성능을 개선시킨다.

[0009] 도 3은 일 실시예에 따른 멀티 프로세서 시스템(300)의 블럭도이다. 이 시스템(300)은 시스템(200)과 유사하지만, 더 상세하게 도시되었다. CPU(308)는 시스템(300)의 루트 프로세서이다. 브리지(306)는 CPU(308)를 시스템 메모리(309)에 결합시킨다. 브리지(306)는 또한 CPU(308)를 호스트 브리지 버스(305a)를 통해 GPU0(304a)에 결합시키고, 그리고 호스트 브리지 버스(305b)를 통해 GPU1(304b)에 결합시킨다.

[0010] GPU들(304)은 각각의 사이드 포트들을 통해 다이렉트 버스(307)를 경유해 서로 결합된다. GPU들(304) 각각은 개

별의 각각의 포트들을 통해 각각의 로컬 메모리들(313a 및 313b)에 더 결합된다. GPU0(304a)는 다양한 클라이언트들(311a)을 포함한다. 클라이언트들(311a)은 메모리 제어기(303a)와 통신한다. 메모리 제어기(303a)는 클라이언트(311)에 의한 로컬 메모리(313a)(예를 들어, 비디오 메모리)로의 액세스를 제어하고, 그리고 클라이언트(311)에 의한 버스(305a)로의 액세스를 제어한다. 특히, 메모리 제어기(303a)는 어떤 클라이언트들(311)이 로컬 메모리(313a)와 통신하는지, 그리고 어떤 클라이언트들(311)이 버스(305a)와 통신하는지, 및 그 시기를 결정한다.

[0011] 마찬가지로, GPU1(304b)은 메모리 제어기(303b)에 결합된 다양한 클라이언트들(311b)을 포함한다. 메모리 제어기(303b)는 로컬 메모리(313b)와 통신한다. GPU0(304a)와 GPU1(304b)는 다이렉트 버스(307)에 의해 서로 결합된다. 다이렉트 버스(307)는 GPU들(304)의 신규한 사이드 포트들 간의 피어투피어 경로이다.

[0012] 일 실시예에서, 클라이언트들(311)에 의한 다양한 버스들로의 액세스는 어드레스 범위에 의해 결정된다. 메모리들(309 및 313)로의 액세스는 어드레스 범위에 근거를 두고 있다. 일 실시예에서, 메모리 어드레스 공간은 세 개의 섹션들, 즉 시스템 메모리(309)를 위한 섹션, 로컬 메모리(313a)를 위한 섹션, 및 로컬 메모리(313b)를 위한 섹션으로 나누어진다. 메모리(313a)는 GPU(304) 관점에서 피어 메모리이고, 그 반대의 경우도 마찬가지이다. 피어 어드레스들은 또 다른 GPU 상의 메모리들을 참조하는 그러한 어드레스들이다. 로컬 메모리로의 액세스는 전형적으로 브리지(306)에 의해 처리된다. 일 실시예에서, 어드레스들이 피어 어드레스 범위에 있을 때, 양쪽의 이용가능한 경로들(다이렉트 버스(307)와 호스트 브리지 버스들(305))이 사용된다. 하나의 경로 혹은 또 다른 경로를 통해 전송되는 데이터의 양은, 서로 다른 실시예들에서 다양한 조건에 따라 달라진다. 예를 들어, 일 모드에 따르면, 데이터는 양쪽 경로들 간에 균등하게 분할 된다(데이터의 50%가 버스(307)에 의해 전송되고, 그리고 데이터의 50%가 버스(305)에 의해 전송됨). 이러한 모드는 종래 아키텍처로는 가능하지 않던 성능 개선을 제공한다. 그러나, 통상의 기술을 가진자들이 이해하고 있는 바와 같이, 버스(305)는 (앞서 설명된 바와 같이, 전송된 데이터에 더하여) 다른 트래픽에 의해 여전히 공유되어 있다. 이런 이유로, 성능을 더욱 개선시키기 위해, 다이나믹 로드 밸런싱(dynamic load balancing)이 이 모드에서 또한 사용될 수 있다. 예를 들어, (다이렉트 버스(307)를 사용하는) 다이렉트 경로가, 선호되는 경로일 수 있지만, 이 경로가 가득 차 있는 경우, (예를 들어, 경로(305a 및 305b)를 사용하는) 다른 경로가 사용된다. 그러나, 다음의 상황을 고려한다. 다음과 같은 데이터 기입("W")의 시퀀스를 가정한다(여기서, WA는 "메모리 어드레스/위치 A에 기입"을 나타냄).

[0013] 1. 305를 통해 GPU0에 WA(여기서, 도 3에 도시된 바와 같이, 데이터 = 4)

[0014] 2. GPU0에 WB

[0015] 3. GPU0에 WC

[0016] 4. 307을 통해 GPU0에 WA(여기서, 도 3에 도시된 바와 같이, 데이터 = 6)

[0017] 기입되는 데이터가 "6"이 될 것이 요구되는 경우이다. 앞서 설명된 시퀀스에서, "4"의 기입이 먼저 일어나지만, 호스트 브리지 버스(305a 및 305b)를 통해 일어난다. "6"의 기입이 이후 시작되지만 다이렉트 버스(307)(가장 빠른 길)를 통해 일어난다. "6"이 먼저 기입되며, 이후에 "4"가 도착하여 "6"을 오버라이팅(overwriting)한다. 만약 이러한 문제가 예측된다면, 이러한 문제를 피하는 일 모드는 어드레스 기반의 라우팅이다. 어드레스들은, 동일한 어드레스에 대한 데이터가 동일한 경로를 통해 전송될 수 있도록 나누어질 수 있다. 예를 들어, 짝수의 더블워드 어드레스(doubleword address)들이 다이렉트 경로(307)를 통해 전송될 수 있고, 홀수의 더블워드 어드레스들이 호스트 브리지 경로(305a 및 305b)를 통해 전송될 수 있다. 대안적으로, 여덟 개의 어드레스들 중 일곱 개(이들은 000 바이너리로 끝나지 않음)가 다이렉트 경로(307)를 통해 전송될 수 있고, 나머지가 호스트 브리지 경로(305a 및 305b)를 통해 전송될 수 있다.

[0018] 또 다른 모드에 따르면, 모든 데이터는 하나의 경로 혹은 다른 경로를 통해 전송된다. 이러한 모드는 가장 바람지하지 않는데, 왜냐하면 한 경로는 사용되지 않기 때문이다. 그러나, 복수의 클라이언트들(311a 및 311b) 각각은 자신의 규칙에 따라 허용가능할 수 있다. 이러한 방식으로, 서로 다른 경로들을 서로 다르게 사용하는 복수의 그래픽 하드웨어 기능을 갖는 것이 가능하고, 경로들의 완전한 이용이 실현된다.

[0019] 아래에서 더 상세히 설명되는 실시예들에서, 기입 동작이 설명된다. 즉, GPU0(304a)이 GPU1(304b)에 데이터를 푸시(push)하지만, GPU1(304b)로부터 다시 데이터를 풀(pull)하지는 않는다. 그러나, 판독 및 기입 동작은, 본 명세서에서 설명되는 동일한 방법을 따르는 다른 실시예들에서 어느 방향에서나 실현가능하다.

[0020] 도 4는 일 실시예에 따른 GPU가 두 개인 시스템(400)의 요소들을 나타낸 블럭도이다. 도 4는 일 실시예에 따른 전체적인 그래픽 서브시스템 상호접속을 포함한다(디스플레이될 수 있는 픽셀들을 마더보드 혹은 디스플레이 디

바이스로 전달하기 위한 디스플레이 링크는 도시되지 않음). 이 시스템(400)은 GPU0(404a) 및 GPU1(404b)을 포함한다. GPU들(404)이 다이렉트 버스(407) 및 각각의 사이드 포트들을 통해 직접적으로 서로 결합된다. "RC"로 표시된 블럭들은 루트 콤플렉스 코어(root complex core)들로서 구성된다. "EP"로 표시된 블럭들은 엔드포인트로서 구성된다. RC와 EP는 PCI 익스프레스(PCIe®) 포트들에 대해 공통 용어이다. 적절한 통신을 위해, RC는 또 다른 RC가 아닌 EP들과 교신해야만 한다. 공통 포트 설계로 동작하는 피어투피어 구현을 제공하는 실시예에서, 404a에서의 "RC" 또는 404b에서의 "RC"가 진정한 RC로서 동작할 수 있게 하고 다른 하나는 EP로서 동작할 수 있게 하는 스트랩핑 로직(strapping logic)이 포함된다. GPU들(404) 각각은 메모리 제어기(Memory Controller, MC) 및 호스트 데이터 경로 블럭(Host Data Path block, HDP)을 포함한다. 범례에서 제시된 바와 같이, 통신 경로들(421)은 판독/기입(Read/Write)(R/W) + d 요청 경로이고, 여기서, "W+d"는 "기입 요청 + 데이터"를 의미한다. "CPL 데이터" 화살표(413)는 데이터를 가진 판독 완료를 위한 경로를 나타낸다. 화살표(423)는 W+d 요청 경로들을 나타낸다. 내부 레지스터, 판독 전용 메모리(Read Only Memory, ROM), 오디오, 및 비디오 그래픽 어레이/어댑터(Video Graphics Array/Adapter, VGA)와 같은 다른 GPU 내부 블럭들로의 호스트 PCIe® 인터페이스들은 도시되지 않았다.

[0021] GPU들(404)은 또한, 각각의 호스트 브리지 버스들(405a 및 405b)을 경유해서 크로스바(coossbar)(417)를 통해 브리지(406)(본 예에서는 노스브리지(NorthBridge))에 결합된다. CPU(408)가 브리지(406)에 결합된다.

[0022] 도 5는 일 실시예에 따른 GPU가 4개인 시스템의 블럭도이다. 일 실시예에서, 두 개의 GPU들인, GPUA(504a) 및 GPUB(504b)가 하나의 보드(보드0)를 점유하고 있으며, 두 개의 추가적인 GPU들인, GPUC(504c) 및 GPUD(504d)가 또 다른 하나의 보드(보드1)를 점유하고 있다. GPU(504a)와 GPU(504b)가 다이렉트 버스(507a) 및 디스플레이 링크(511a)를 통해 결합된다. GPU(504a)와 GPU(504b) 각각은 각각의 PCIe® 버스들을 통해 이산 브리지(discrete bridge)(509a)에 결합된다.

[0023] GPU들, GPUC(504c) 및 GPUD(504d)이 다이렉트 버스(507b)와 디스플레이 링크(511c)를 통해 결합된다. GPU(504c)와 GPU(504d) 각각이 각각의 PCIe® 버스들을 통해 이산 브리지(509b)에 결합된다.

[0024] 보드0과 보드1이 디스플레이 링크(511b)에 의해 결합된다. 보드0과 보드1은 또한, 이산 브리지(509a)로부터 브리지(506)까지 PCIe® 버스를 경유하여 그리고 이산 브리지(509b)로부터 브리지(506)까지 또 다른 PCIe® 버스에 의해, 브리지(506)를 통해서 결합된다. CPU(508)가 브리지(506)에 결합된다. 피어투피어 전달이 시스템(500)에서 임의 쌍의 GPU들 간에 동시에 가능할 수 있다. 일 실시예에서, 만약 이 쌍들이 AB 및 CD라면, 각각의 전달은 최대 지원되는 값이 대략 12.2GB/초에서 수행된다. 만약 이 쌍들이 AC 및 BD 혹은 AD 및 BC이고, 전달의 방향이 반대 방향이라면, 각각의 전달은 대략 6.1 GB/초에서 수행된다. 만약 이 쌍들이 AC 및 BD 혹은 AD 및 BC이고, 전달의 방향이 동일한 방향이라면, 각각의 전달은 대략 3 GB/초에서 수행된다.

[0025] 도 6은 일 실시예에 따른 사이드 포트 서브시스템(side port subsystem)의 블럭도이다. 아래에 명명되는 섹션은, 도 6을 참조하여, 다양한 실시예들에 따른 세부적 구현을 제공하지만, 본원 발명이 이와 같이 설명되는 특정의 세부적 사항들로만 한정되는 것은 아니다.

[0026] *호스트 PCIe® 엔드포인트(Host PCIe® Endpoint)*

[0027] 일 실시예에서, 호스트 PCIe® 인터페이스는 버스 인터페이스 코어 로직(Bus Interface core logic, BIF), 아날로그 PHY, 및 래퍼(wrapper)로 형성된다. 이 인터페이스는 엔드포인트(Endpoint, EP)로서 구성된다. 이 버스는 표준 PCI 익스프레스 2.0 사양("PCIe2®")의 레이트(rate)인 핀당 5Gbps에서 동작하거나 또는 16 레인(lane) 인터페이스에 대해서 이론적 최대 레이트인 8GB/초에서 동작한다. 이 버스는 (호스트 혹은 피어로) 나가는 메모리 기입을 위한 128b 크레딧/데빗 마스터 인터페이스(credit/debit master interface) 및 (호스트 혹은 피어로부터) 들어오는 메모리 기입을 위한 128b 크레딧/데빗 슬레이브 인터페이스(credit/debit slave interface)를 갖는다. 이 버스는 판독 전용 메모리(Read Only Memory, ROM), 내부 레지스터 버스, 오디오, 및 비디오 그래픽 어레이(Video Graphics Array, VGA)에 대한 32b 인터페이스들을 갖는다. 이 버스는 호스트 요청들을 자신의 슬레이브 인터페이스(S로 표시됨)를 통해 복수의 GPU 목적지로 라우팅시킨다.

[0028] 요청 인터페이스들은 동반되는 완료 인터페이스들을 갖는다. BIF는 어드레스 애퍼쳐(address aperture)들의 세트(프레임 버퍼를 위한 것, 레지스터들을 위한 것, 그리고 ROM을 위한 것)로 트래픽을 디렉트(direct)시킨다. BIF는 구성 레지스터들 및 스트랩들을 위해 32b 인터페이스를 사용한다. BIF는 또한 ACPI 파워 상태 관리, 클럭, 리셋, 및 스캔 입력을 처리하기 위해 클럭 발생기(Clock Generator, CG)에 대한 인터페이스와, 그리고 디버그 버스 데이지 체인 인터페이스(debug bus daisy chain interface)를 갖는다. BIF는 또한 프레임버퍼 판독

및 기입을 위해 호스트 데이터 경로 블럭(Host Data Path block)(HDP)에 대한 인터페이스를 갖는다.

[0029] *사이드 포트 루트 콤플렉스(Side Port Root Complex)*

[0030] 사이드 포트를 위해 사용되는 PCI 익스프레스 로직(XSP)은 루트 콤플렉스(Root Complex, RC)로서 구성된다. 동작의 RC와 EP 모드들 간에는 구성 및 능력에서의 작은 차이가 있다. XSP 버스는 표준 PCIe2® 사양의 레이트인 핀당 0.5Gbps에서 동작하거나, 혹은 16 레인 인터페이스에 대해서 이론적 최대 레이트인 8GB/초에서 동작한다. 이 버스는 피어 GPU로 나가는 메모리 제어기(MC) 메모리 공간 기입 요청을 위한 128 데이터 비트의 크레딧/데빗 마스터 인터페이스, 및 피어 GPU로부터 들어오는 메모리 기입을 위한 128b 크레딧/데빗 슬레이브 인터페이스를 갖는다. 이들은 호스트 BIF 인스턴스 상에서 HDP에 의해 사용되는 인터페이스들이다. 이 버스는 구성 레지스터들 및 스트랩들을 위한 32b 인터페이스를 사용한다. 이 버스는 자기 자신의 링크 파워 상태를 호스트 인터페이스로부터 분리하여 유지시킨다. 호스트 인터페이스에서와 마찬가지로, 레인들의 수는 8 혹은 16이 되도록 소프트웨어 프로그래밍될 수 있다. 애플리케이션 특정 집적 회로(Application Specific Integrated Circuit, ASIC) 상의 두 개의 PCIe® PHY들이 동일 레이트에서 실행되기 때문에, 기준 클럭(reference clock, REFCLK) 입력 핀들의 단일 쌍이 사용될 수 있고, 이 경우 버퍼링된 REFCLK은 온-다이(on-die)로 두 개의 PHY들에 배분된다.

[0031] XSP 인스턴스는 다른 판독/기입 인터페이스들 중 어느 것도 필요로 하지 않으며, 또는 호스트 BIF에 의해 지원되는 인터럽트 핸들링을 필요로 하지 않는다. 사이드 포트는 ASIC 파워업에서 활성화될 필요가 없고, 따라서 대부분의 경우 핀 기반의 스트랩들을 사용하지 않는다. 사이드 포트의 구성은 SRBM 버스를 통한 소프트웨어 레지스터 기입을 통해 호스트 인터페이스와 분리되어 프로그래밍될 수 있다.

[0032] XSP는 레지스터 판독 혹은 기입, I/O 공간, 또는 구성 공간 싸이클을 지원할 필요가 없다. 단지 단일의 PCIe® 기능만이 이 디바이스 상에서 동작가능하도록 요구된다. 단지 포스팅(posting)된 기입들만이 발행되고(판독은 전혀 없음), 그리고 전체적 동기화 방식이 사용되는 실시예에서, 완료 메시지는 필요하지 않다.

[0033] MC 허브는 기입 요청을 XSP로 라우팅한다. 동일한 어드레스들이 호스트 BIF 요청에 대해서와 같이 XSP 요청에 대해 사용된다.

[0034] *호스트 데이터 경로 블럭(Host Data Path Block)*

[0035] 일 실시예에서, 호스트 데이터 경로 블럭(HDP)은 버스IP(BusIP)의 일부인 표준 블럭이다. HDP는 호스트 인터페이스에 의해 수행되는 프레임 버퍼 판독 및 기입을 핸들링한다. 멀티 GPU 시스템에서, 프레임 버퍼 판독 및 기입은 호스트로부터 나올 수 있거나 혹은 또 다른 GPU에 의해 개시되는 피어투피어 기입으로부터 나올 수 있다. 호스트 액세스들은 복수의 HDP 타일링 애퍼쳐들(tiling apertures) 중 하나를 사용할 수 있고, 반면, 피어투피어 액세스들은 사용할 수 있거나 사용하지 않을 수 있다(피어들은 요청자 ID에 의해 구별될 수 있음). HDP는 호스트 BIF의 슬레이브 포트에 그리고 MC에 판독 및 기입 클라이언트로서 인터페이스한다. BIF 및 HDP는 판독 대 기입 코히어런스(read to write coherency)를 유지하기 위해 협력하는바, 판독은 이전 기입들을 패스(pass)할 수 없다. 피어투피어 기입 동기화의 경우, 보존된 메모리 어드레스로의 특별한 기입이, HDP에 의해 인식되고, 그리고 MC 판독을 트리거하고 로컬 메모리로의 전체적인 특정의 "기입 단계"에 대한 이전 기입들의 플러시(flush)를 트리거하며, 이후에, 전송 클라이언트의 레지스터 공간에서의 단계별 메일박스로의 화답(writeback)이 수행된다.

[0036] 버스로부터 들어오는 판독 및 기입은 BIF에서의 특별한 애퍼쳐에 의해 HDP에 라우팅된다. 이 애퍼쳐는 로컬 메모리의 32MB 혹은 그 이상에 대응한다. 선택에 따라서는, HDP에 의해 핸들링되는 어드레스들은 가상 어드레스들로 다루어지고, 그리고 HDP 전용의 페이지 테이블에서 검색된다.

[0037] 각각의 GPU에 대해 이용가능한 PCIe® 애퍼쳐를 가장 잘 사용하기 위해 그리고 기입 단계들을 지원하기 위해, 피어투피어 기입 어드레스들은, 전송 GPU의 MC에 의해, 단계 및 오프셋으로서 인코딩되고, 산재된 자동적 단계별 베이스 어드레스 레지스터(Base Address Register, BAR) 업데이트들을 가질 수 있다. HDP는, BIF로부터의 요청자 ID + 전송된 비트들의 조합을 사용하여, 어떤 기입 단계 및 동적 어드레스 애퍼쳐를 사용해야 할지를 결정한다.

[0038] *메모리 제어기 블럭(Memory Controller (MC) Block)*

[0039] MC는 사이드 포트로의 결합을 위해 두 개의 인터페이스를 가지는바, MC 클라이언트들에 의해 요청된 기입들을 위한 128b BIF 마스터 인터페이스와, 다른 GPU로부터 로컬 메모리로의 들어오는 기입들을 위한 128b BIF 슬레이브 인터페이스를 갖는다. MC는 또한 마스터 요청을 XSP를 통해 피어 메모리로 라우팅시키기 위한 어드레싱 로직

을 포함한다. XDP 블럭은 인터페이스 기능을 처리한다. 이 블럭은 슬레이브 요청들을 검사하고, 어드레스 오프셋팅을 수행하거나 혹은 비디오 메모리(Video Memory, VM) 검색을 수행하며, 기입 완료 카운터들을 유지시키고, 그리고 기입 결합(32 BIF 요청들을 64B MC 요청들에 결합)을 제공한다.

[0040] 사이드 포트로의 요청 및 사이드 포트로부터의 요청은 (마스킹(masking)을 가진) 16 바이트 내지 64 바이트일 수 있다. 최상의 성능을 위해 요청들은 64B이어야만 한다.

[0041] 이 사이드 포트 BIF 인터페이스들은, 전체 그래픽 가상 메모리가 활성화되었는지 활성화되지 않았는지 여부에 상관없이 GART 변환 40b 시스템 어드레스들을 패스한다. GART는 시스템 콘텍스트 도메인(system context domain)과 동일하며, 36 비트 GPU 디바이스 어드레스들로부터의 변환을 수행한다.

[0042] MC는 멀티 GPU에 의해 보증된 복사 동작의 플러싱(flushing)을 가능하게 하는 방식으로 기입 완료 상태를 유지한다. HDP와 유사하게, XDP에 의해 인식된 특별한 어드레스는 특정 기입 단계의 기입 완료에 대한 점검을 트리거하는데 사용되고, 그리고 전송 클라이언트의 레지스터 공간에서의 메일박스로의 화답을 발생시키는데 사용된다.

[0043] *다이렉트 메모리 액세스 엔진(Direct Memory Access Engine (DRMDMA))*

[0044] 사이드 포드를 갖는 GPU는 메모리의 효과적인 호스트 블럭 전달(block transfer, BLT)들을 수행할 수 있는 DMA 엔진을 포함한다. DRMDMA는 로컬 메모리, 호스트 메모리, 및 또 하나의 GPU의 로컬 메모리 중 어느 둘 간의 전달을 위해 프로그래밍될 수 있다. (아래에서 더 설명되는) 어드레스들의 처리에 대한 개선과 함께, DRMDMA는 또한 XSP와 PCIe® 브리지 양쪽 모두를 사용하여 메모리의 블럭들을 이동시킬 수 있다. 일 실시예의 표준 전달 능력은 다음을 포함한다.

[0045] • 호스트에서 로컬로 (호스트 BLT 혹은 복사) - 클럭 당 16B의 피크 레이트(peak rate)

[0046] • 로컬을 상수값으로 채움 - 클럭당 32B의 피크 레이트

[0047] • 로컬에서 호스트로 (복사) - 클럭당 16B의 피크 레이트

[0048] • 로컬에서 원격 GPU로 (복사) - 이웃하는 GPU로는 클럭당 32B의 피크 레이트, 이웃하지 않는 GPU로는 클럭당 16B의 피크 레이트

[0049] DRMDMA는 클럭당 32B의 판독 및 32B 기입을 지원할 수 있고, 이는 두 개의 16B PCIe® 기입 인터페이스와 매칭된다.

[0050] *호스트 대 사이드 포트 로드 밸런싱(Host vs. Side Port Load Balancing)*

[0051] 사이드 포트의 부가는 이웃 GPU에 대한 더 큰 대역폭을 가능하게 하는바, 이는 이 대역폭의 대부분이 2 GPU & 4 GPU 사용 모델들에 대해 필요한 경우이다. 호스트 포트는 이용가능한 많은 대역폭을 또한 가지며, 이웃 GPU와의 통신을 위해 여전히 사용가능해야만 한다.

[0052] 로드 밸런싱의 가장 간단한 방법은 고정된 할당에 근거하여 데이터를 전달하는데 사용되는 포트를 다른 클라이언트들에 정적으로 할당하는 것을 포함하는 바, 예를 들어, 사이드 포트에 할당된 DRMDMA 및 호스트 포트에 할당된 컬러 버퍼 블럭(Color Buffer block)이 있다.

[0053] 로드 밸런싱의 두 번째 방법은 임의의 클라이언트로부터의 트래픽을, 사이드 포트가 얼마나 고밀도로 사용되고 있는지에 근거하여 두 개의 포트들 중 어느 하나에 동적으로 라우팅시키는 것을 포함한다. 아래의 "임의의(ANY)" 로드 밸런싱의 설명을 참조하기 바란다.

[0054] 앞서의 이 두 번째 방법을 사용하는 경우, 두 개의 서로 다른 경로들을 통해 동일 픽셀 어드레스로의 기입을 위한 레이스 조건(race condition)이 존재할 수 있다. 이는 기입들이 두 개의 서로 다른 프리미티브(primitive)들에 속하는 경우 픽셀 오더링 룰(pixel ordering rule)을 위반할 수 있다. 더 이른 프리미티브에 대한 기입이 더 이후의 프리미티브에 대한 기입 이후에 도달할 수 있다. 이 문제를 해결하기 위해, 어드레스에 근거한 라우팅의 옵션이 지원된다. 폴백 방법(fallback method)은 관련 클라이언트에 대해 단일 경로를 사용하는 것이다.

[0055] 만약 피어 기입 요청들(reqs)이 호스트 및 사이드 포트를 통해 이웃에 전송되고 있다면, 요청 오더(request order)는 이러한 2 포트들에 걸친 서로 다른 가변의 레이턴시(latency)들 때문에 유실될 수 있다. 호스트 포트는 어떤 상황에서는 심지어 스톨링(stalling)될 수도 있다. 이것이 의미하는 바는, 필요한 경우 요청 오더링이

보존되도록 하는 스텝들이 취해져야만 하는 것을 의미한다.

[0056] MC로부터의 유일한 오더링 보증은 동일 클라이언트로부터의 동일한 어드레스로의 기입이 리오더링(reordering)되지 않는 것이다. 그래서, 이것이 또한 피어 기입들에 대한 유일한 기입 후 기입(Write after Write, WAW) 요건이다. 실시예들에 따라 이것을 보증하는 수개의 방법이 있다. MC는 로컬 메모리뿐만 아니라 호스트 및 피어에 대한 판독/기입을 처리한다. 디폴트 MC 오더링 룰은 설명된 바와 같다. 추가적인 라우팅 룰 및 버스는 자체적으로 추가적인 오더링 보증을 제공한다. 모든 로드 밸런싱은 단지 이웃하는 GPU로 진행하는 요청들에만 적용된다. 이것은 애퍼쳐 점검에 의해 결정되고, 비이웃 요청(non-neighbor request)들은 호스트 포트로 모두 진행할 것이다.

[0057] *로드 밸런싱: 포트들로의 고정된 맵핑(Load Balancing: Fixed mapping to ports ("FXD"))*

[0058] 동일한 어드레스로의 두 개의 기입들이 비정렬되지 않도록 보장하는 한 가지 방법은 어드레스 점검에 의한 로드 밸런싱을 수행하는 것이다. 어드레스 점검은 소정의 어드레스가 언제나 동일한 포트를 사용함을 보장하도록 수행된다. 예를 들어, 어떤 입도(granularity)에 있어서, "짝수" 어드레스는 언제나 호스트 포트를 사용하고, "홀수" 어드레스는 언제나 사이드 포트를 사용한다. 이러한 방식으로, 동일한 어드레스로의 2개의 요청들은, 포트들을 크로싱(crossing)하면서, 비정렬될 수 없다.

[0059] 사용된 어드레스 점검 맵핑은, 어떤 어드레스 비트들이 검사되는지 및 어떤 쓰레시홀드(threshold)가 밸런싱을 위해 사용되는지에 관해, 프로그래밍 가능하다. 사용된 어드레스 비스트들이 밸런싱의 입도를 설정하는 데, 왜냐하면 더 큰 입도는 아마도 하나의 포트 혹은 다른 포트에 걸쳐 기입 버스트(burst)들의 함께 머무름을 의미할 것이기 때문이고, 입도가 너무 큰 것은 결과적으로 포트 스타빙(port starving)을 일으키며, 큰 버스트들은 하나의 포트 혹은 또 다른 하나의 포트로 전송된다. 사용된 쓰레시홀드는 호스트와 사이드 포트 간의 밸런싱의 더 세밀한 비율을 가능하게 한다. 예를 들어, 만약 세 개의 어드레스 비트들이 사용된다면, 호스트:사이드의 비율은, 사용된 쓰레시홀드에 따라, 0:8, 1:7, 2:6, ..., 7:1, 8:0일 수 있다. 모든 요청들을 단지 호스트 포트로만 혹은 사이드 포트로만 전송하는 것은 이러한 고정된 맵핑의 안 좋은 경우이다.

[0060] 이러한 타입의 고정된 로드 밸런싱과 함께, 만약 포트(호스트)가 매우 바쁘거나 혹은 스톨링되어 있다면, 백업(backup)이 다른 (사이드) 포트로 전송될 수 있는 대역폭을 쓰로틀(throttle)할 수 있다. 또한, 어떤 기입 패턴들은, 손쉽게, 요구된 밸런싱 비율이 파괴되도록 할 수 있다(예를 들어, 하나의 포트로 언제나 맵핑되고, 다른 포트는 비사용인 채로 남겨두는 스트라이딩 기입들(striding writes)). 따라서, 이러한 로드 밸런싱 방법의 바람직하지 않은 특성이 존재하는데, 그 원인은, 이용가능한 대역폭에 대해 반응하기보다는 오히려 정적이고, 소프트웨어 프로그래밍되기 때문이다. 동적 로드 밸런싱을 행하는 대안적 예가 아래에서 설명된다.

[0061] *로드 밸런싱: WAW가 보증되지 않는 무제약 혹은 임의의 오더링(Load Balancing: Unconstrained or any ordering where WAW is not guaranteed ("ANY"))*

[0062] 임의의(ANY) 로드 밸런싱이, 가득 찬 조건 혹은 거의 가득 찬 조건에 근거하여, 포트들 간에 스위칭된다. 이 사이드 포트는 결국, 예측된 더 높은 이용가능도로 인해 선호되도록 돼야만 한다. 각각의 포트 상에서의 다수의 미처리 크레딧들에 대한 제어가 밸런싱을 튜닝(tunning)하기 위해 제공되어야 한다. 어떤 클라이언트들은 WAW 코히어런스 보증을 필요로 하지 않는다. 예를 들어, 복사들은 동일한 복사에서 동일한 어드레스를 두 번 전달하지 않기 때문에, 코히어런스는 복사들 간의 플러싱에 의해 보증될 수 있다. 이것을 지원함으로써 소스 & 목적지는 전속력으로 진행할 수 있고, 단지 필요한 것은 잠재적으로 오버랩핑되는 전달들 간에 정규의 기입 완료 동기화를 행하는 것이다. 이러한 트래픽을 임의(ANY) 타입의 트래픽과 혼합하는 것은 또한, 이들 각각의 약점들의 일부를 완화시킬 수 있고, 잠재적으로는 이러한 기입들이 임의의 갭(gap)들을 채울 수 있게 한다. 사이드 포트가 가능한 한 많이 사용될 수 있고, 이 경우 호스트 포트는 사이드 포트가 가득 찬 경우에만 사용된다.

[0063] 로드 밸런싱의 임의의 구현 방법 및/또는 모든 구현 방법들은, 전달들의 독립적 스트림들을 위해 동시 사용될 수 있다. 소프트웨어는, 어떤 타입의 로드 밸런싱을 단계별로 사용할지를 구성할 수 있다.

[0064] 만약 임의의(ANY) 트래픽과 FXD 트래픽과의 혼합이 존재한다면, 임의의 트래픽은 FXD 옵션들의 약점에 의해 야기되는 임의의 버블(bubble)들을 채우는 것에 도움을 주어야만 한다. 하드웨어는 이러한 임의(ANY) 타입의 요청들이 일부 스톨링된 혹은 대기하는 FXD 요청들을 우회(bypass)하도록 지원한다.

[0065] FXD 및 ANY가 (지원되는) 단지 두 개의 로드 밸런싱 모드이지만, 다른 많은 실시예들이 가능하다.

[0066] *로드 밸런싱: 가능한 사용 시나리오(Load Balancing: Possible usage scenarios)*

[0067] 다음에 제시되는 것은 가능한 사용 모델들이지만, 실시예들이 이러한 것에 한정되는 것은 아니다.

[0068] 1) 피어 기입들은 DMA들이다(1-2 단계들이 데이터를 피어 GPU(들)에 복사하기 위해 DRMDMA에 의해 사용됨).

[0069] 1a) "임의(ANY)" 타입의 로드 밸런싱이 전달(들)을 위해 사용될 수 있지만, 이것은 오버랩핑의 위험이 있는 임의의 DMA들 간의 추가적인 동기화를 요구할 수 있다. 소프트웨어는 오버랩의 발생 여부를 결정하기 위해 점검을 수행할 수 있다.

[0070] 1b) 두 개의 DMA 엔진들을 위한 교번적 로드 밸런싱 옵션은 하나의 엔진은 "호스트 전용"으로 사용하고, 하나의 엔진은 "사이드 전용"으로 사용하는 것이다. 또는, "호스트 전용"은 이웃하지 않는 피어로의 기입들을 의미할 수 있다. 호스트 포트에 대해 이용가능한 대역폭이 보증되지 않아서, 소프트웨어가 최적으로 복사 작업을 나누는 것이 어려울 수 있지만, 만약 두 단계들이 동기화 레이턴시를 숨긴다면, 이것은 어떤 잠재적으로 오버랩핑되는 복사들 간의 어떤 추가적 동기화를 요구함이 없이 양쪽 포트들을 가득 찬 상태로 유지시킬 것이다. 하드웨어는 호스트 전용 단계 및 사이드 전용 단계들로 하여금 만약 관련된 리소스들이 오버랩핑하지 않는다면 동일한 단계 수를 사용하도록 할 수 있다.

[0071] 2) WAW 민감 클라이언트 기입(복사들이 또한 사용됨이 있는 경우 혹은 없는 경우)

[0072] 2a) 앞서의 1b와 마찬가지로, 만약 하나 혹은 그 이상의 피어 기입들이 전송된다면, 각각은 "호스트 전용" 혹은 "피어 전용"의 전용 경로에 할당될 수 있어, 양쪽 포트들이 완전히 사용되며, 포트들 간의 오더링 종속성 혹은 교차 결합 배압 경로(cross-coupled backpressure path)들이 존재하지 않는다.

[0073] 2b) 만약 이 WAW 민감 클라이언트가 전달을 행하는 유일한 클라이언트라면, 단지 "FXD"만이 이 포트들이 계속 완전히 이용될 수 있도록 하기 위해 사용될 필요가 있고, 이 경우 "사이드 전용"은 만약 다른 것들이 문제가 있다면(그리고/또는 만약 또 하나의 다른 클라이언트가 일부 시간 동안 "호스트 전용"을 사용할 수 있다면) 안전한 백업일 수 있다.

[0074] *어드레싱(Addressing)*

[0075] GPU의 호스트 인터페이스는 (메모리 공간에서 맵핑된) 레지스터들에 대한 버스 요청, 로컬 메모리 (프레임 버퍼)에 대한 버스 요청, 뿐만 아니라 다른 블럭들에 대한 버스 요청을 수신한다. BIF 내의 애퍼쳐들(또는 어드레스 범위들)의 세트는 PCIe® 공간 어드레스들(64b)을 GPU 내의 특정의 목적지로 맵핑시킨다. 로컬 메모리 판독 및 기입이 MC까지 진행하는 중 HDP를 통과한다.

[0076] GPU 발신 메모리 요청들은 가상 어드레스들일 수 있거나 아닐 수 있는 어드레스들을 사용한다. 윈도우 XP 혹은 비스타 그래픽 드라이버 모델의 경우, GPU "디바이스" 어드레스들은 시스템 메모리(GART를 통해 더 맵핑될 수 있음) 혹은 로컬 메모리를 포함하는 애퍼쳐 영역 플랫 어드레스 공간(aperture-regioned flat address space)을 형성한다. 장래의 드라이버 모델의 경우, GPU 어드레스들은 페이지 맵핑된 가상 어드레스들로서 다루어지고, 검색된 물리적 어드레스들은 시스템 메모리 및 로컬 메모리에 대해 분리된다.

[0077] *시스템 메모리 공간(System Memory Space)*

[0078] GPU의 관점으로부터, 시스템 메모리는 전형적으로, CPU에 부착된 다이나믹 랜덤 액세스 메모리(Dynamic Random Access Memory, DRAM)를 의미하지만, 실제 시스템에서 메모리는 임의의 액세스가능한 메모리 맵핑 디바이스들, 특히 다른 GPU 메모리에 속하는 임의의 메모리 공간을 포함한다. 시스템 메모리 액세스의 맵핑 및 라우팅은, 시스템 구성에 따라, 이산 브리지들과 칩셋의 브리지들(예를 들어, 노스브리지)의 조합에 의해 처리된다.

[0079] CPU로부터 관측되는 경우, 물리적 어드레스 공간(32b 혹은 64b)은 CPU에 부착된 진정한 시스템 메모리와, 그리고 "다른" 메모리로 파티션(partition)되고, 후자는 메모리 맵핑 디바이스들(즉, PCIe® 버스) 상에 존재하는 임의의 메모리이다. 이러한 파티션은 입력/출력(Input/Output, I/O) 시스템의 최상위 노드에서의 단일 베이스-리미트 구성 레지스터 쌍(single base-limit configuration register pair)을 통해 수행된다. 레지스터 공간을 위한 또 하나의 다른 쌍이 또한 존재한다. 그 아래에 있는 바이너리 트리(binary tree)의 각각의 레벨에서, 버스 어드레스 범위는 두 부분으로 나누어져, 각각의 다운스트림 레그(downstram leg)를 위한 베이스 리미트 쌍이 된다. GPU의 애퍼쳐 크기는 보드 스트랩(board strap)에 의해 설정된다. 오늘날, 이 크기는 32b 및 64b 운영 체제 양쪽 모두에 대해 유효하고, 얼마나 많은 GPU들이 시스템에 삽입될 것인지에 상관없이 유효하다.

[0080] GPU로부터 관측되는 경우, 시스템 어드레스들은 버스 어드레스 공간에 GART(시스템 페이지 테이블)을 통해 혹은 직접적으로 맵핑되고, 이 경우, 발생되지 않은 최상위 비트(Most Significant Bit, MSB)들은 0으로 설정된다.

각각의 업스트림 PCIe® 브리지 노드는, 베이스 리미트 쌍에 대한 점검을 수행하여, 후속 업스트림을 라우팅시키거나 혹은 피어 접속으로 라우팅시킨다.

[0081] 도 7은 두 개의 GPU들을 갖는 시스템 메모리 공간의 하이 레벨 다이어그램을 나타내며, 각각의 엔티티들이 어떻게 이것을 보고 있는지를 나타낸다. 도 7A는 CPU 뷰(view)이고, 도 7B는 GPU0 뷰이며, 그리고 도 7C는 GPU1 뷰이다.

[0082] *디스에이블된 비디오 메모리(Video Memory Disabled)*

[0083] GPU에 의해 MC에 제공된 36 비트 어드레스들은 다음의 동작 시퀀스를 겪게된다.

[0084] 1. 로컬 메모리 애퍼쳐에 대한 점검. 만약 어드레스가 이 범위 내에 있다면, 베이스 어드레스는 공제되고, 물리적 로컬 메모리 어드레스로서 사용된다.

[0085] 2. 만약 범위 밖에 있다면, 로컬 메모리 애퍼쳐는 사이드 포트 메모리 애퍼쳐에 대한 점검을 행한다. 이것은 일반적으로 이웃 GPU에 의해 사용되는 어드레스 범위에 대응한다. 만약 어드레스가 이 범위 내에 있다면, 이것은, 로드에 따라, 사이드 포트 혹은 호스트 인터페이스 상에서 수정되지 않은 채 사용된다. 만약 사이드 포트로 진행한다면, 원격 GPU 클라이언트는 자기 자신의 애퍼쳐 점검을 행하고, 이 검검으로 인해 결과적으로 로컬 메모리 애퍼쳐 내에서의 패스가 일어나게 된다. 만약 호스트 인터페이스로 진행한다면, 브리지 혹은 노스브리지에서의 애퍼쳐 점검은 요청을 원격 GPU로 라우팅시킬 것이다. 이 원격 호스트 인터페이스는 이러한 요청들을 자신의 HDP 혹은 SRBM으로 패스할 것이다.

[0086] 3. 만약 어드레스가 이전의 두 개의 애퍼쳐들 밖에 있다면, 이것은 시스템 어드레스이다. 만약 어드레스가, 비스눕핑된 애퍼쳐(non-snooped aperture) 안쪽에 있다면, 수정되지 않은 채 호스트 인터페이스로 패스된다. 만약 그렇지 않다면, 시스템 GART(예를 들어, 시스템 페이지 테이블)에서 검색되고 호스트 인터페이스로 패스된다.

[0087] *인에이블된 비디오 메모리(Video Memory Enabled)*

[0088] GPU 어드레스들은 가상으로서 다루어지고, 페이지 테이블에 의해 VM에서 시스템 물리적 어드레스들 혹은 로컬 비디오 메모리 어드레스들로 변환된다. 전통적으로, 시스템 물리적 어드레스들은 40b이고 요청의 목적지는 (GART(실제로는 시스템 페이지 테이블)를 통한 패스 이후) 언제나 호스트 인터페이스이다. 그러나, 피어투피어 전달을 위한 호스트 인터페이스와 사이드 포트 양쪽 모두의 사용을 지원하기 위해, 실시예들은 또한, 사이드 포트에 걸쳐 시스템 물리적 어드레스들의 라우팅을 가능하게 한다. 다음은 일 실시예에 따른 시퀀스이다.

[0089] 1. GPU 클라이언트(예를 들어, DMA 엔진)에 의해 만들어진 요청들이 적절한 페이지 테이블에서 검색된다.

[0090] 2. 만약 PTE에서의 (시스템 공간에 대해) "S" 비트가 설정되지 않았다면, 요청 GPU의 로컬 메모리가 액세스된다.

[0091] 3. 만약 "S" 비트가 설정되면, 사이드 포트 애퍼쳐 어드레스 범위 점검이 수행된다.

[0092] 4. 만약 어드레스가 SPA 밖에 있다면, 요청은 (GART 검색 이후) 호스트 인터페이스로 진행한다.

[0093] 5. 만약 어드레스가 SPA 안쪽에 있다면, 로드 밸런싱 결정이 행해지고, 이 요청은 사이드 포트로 라우팅되거나 호스트 인터페이스로 라우팅된다.

[0094] 6. 만약 요청이 호스트 인터페이스를 통해 라우팅된다면, 이는 현재 수행되는 바와 같은 정규 피어투피어 요청이다.

[0095] 7. 만약 요청이 사이드 포트를 통해 라우팅된다면, 시스템 어드레스는 원격 GPU로 패스되고, 이곳에서 시스템 어드레스는 HDP 공간 페이지 테이블에서 검색되고, 로컬 물리적 메모리 어드레스로 (거의 언제나) 맵핑된다.

[0096] *동기화(Synchronization)*

[0097] 복수의 GPU들을 사용하여 렌더링(rendering)하는 것은 데이터 코히어런스를 보장하기 위해 도로잉과 데이터 이동 스텝들 간의 동기화를 필요로 한다. 개별 커맨드 스트림들은 드로잉 및 복사를 수행한다. 교번적 프레임 렌더링(Alternate Frame Rendering, AFR)을 위한 전형적인 시퀀스는 다음과 같다.

[0098] 1. GPU0 상의 3D 엔진은 동적인 혹은 계속되는 텍스처 표면(texture surface)에 대해 드로잉을 행한다.

[0099] 2. 3D 엔진은 완료를 플러싱하고 그 신호를 보낸다(세마포어 신호 커맨드(semaphore Signal command)를 발행함).

[0100] 3. GPU0 상의 복사 엔진은 세마포어를 기다리고, 결과를 GPU1에 복사한다.

[0101] 4. 복사 엔진이 완료를 플러싱하고 그 신호를 보낸다.

[0102] 5. GPU1 상의 3D 엔진이 세마포어를 기다리고, 렌더링된 텍스처를 사용하여 드로잉을 행한다.

[0103] BIF 및 MC (뿐만 아니라 노스브리지 및 CPU 메모리 시스템) 각각은, 판독 및 기입 동작의 오더링을 위해 자기 자신의 코히어런스 룰을 갖는다. 소프트웨어 중재 없이 복수의 GPU들 간의 목적지 코히어런스에 대한 엔드투엔드 요청을 유지하기 위해, 이러한 룰들 모두를 고려하는 (또는 이들에 종속되지 않는) 동기화 방식이 필요하다.

[0104] 복수의 GPU에 대한 동기화의 필요는 단일 GPU에 대한 동기화의 필요와 유사하며, 둘 간에는 표면 데이터의 DMA들의 추가적인 스텝들이 있다. 이러한 DMA 경우에, 그리고 브로드캐스트 기입 능력의 경우에, 하나의 GPU 상에서의 클라이언트에 대해서, 기입이 또 하나의 다른 GPU 상에서 완료되었음을 보증하는 문제가 존재한다. MC 및 세마포어 블럭이 이러한 태스크에 도움을 준다.

[0105] *애퍼처 콤프레션(Aperture Compression)*

[0106] 효과적인 GPU 시스템들을 설계함에 있어서 한가지 해결할 문제는, GPU들의 수와 각각의 GPU를 가진 로컬 메모리의 양이 증가함에 따라, 시스템 메모리 공간에 모든 것을 맞추는 능력이 손상되고 그리고 디바이스 애퍼처가 크기 면에서 감소돼야 한다는 것이다. 만약 시스템 공간에 맵핑가능한 GPU 로컬 메모리의 양이 한정되어 있다면, 맵핑되지 않은 메모리의 유용성은 로컬 GPU로 한정된다. CPU 맵핑가능 GPU 공간의 한정된 범위를 갖는 것이 크게 중요하지않는 바, 폴백은, 로컬 비디오 메모리로의 락킹된 액세스(locked access) 대신에, 시스템 메모리에 표면들을 복사하는 것이다.

[0107] 일 실시예에 따른 해법은, 오프셋 레지스터(P2P BAR)를 부가함으로써 로컬 메모리의 나머지가 피어에 액세스가능하게 하는 것이다. 목적지 BIF가 그 소스로부터의 기입 요청을 관측할 때, HDP에 대해 36b 디바이스 어드레스를 생성한 메모리 어퍼처를 적용한 후에, 오프셋이 (4kB 페이지 입도로) 부가된다. 소스가 관측할 있는 인접하는 로컬 가상 메모리 공간의 양은 전에 존재했던 것보다 더 크지 않지만, 그러나 이것은 이제, P2P BAR를 재기입함으로써 변경될 수 있는 로컬 메모리 모두로의 윈도우이다. 이론적으로 BAR는 변경이 기입 스트림 내에서 오더링되는 동안은 어느 때나 이동될 수 있다. 복수의 오프셋들 모두가, 이들이 소스에 의해 구별될 수 있는 동안 (예를 들어, 기입 단계), 동기화되거나 혹은 공간적으로 분리될 필요가 없음에 주목해야 한다.

[0108] 도 8은 오프셋을 갖는 메모리 공간의 다이어그램이다. 도 8A는 시스템 메모리 공간의 GPU0 뷰이고, 도 8B는 오프셋이 없는 GPU1 LVM 맵핑이며, 그리고 도 8C는 오프셋이 있는 GPU1 맵핑이다.

[0109] *기입 단계(Write Phase)*

[0110] 단일 GPU가 기입들의 복수의 세트들(동시에 진행중인 기입 트래픽의 복수의 세트들(예를 들어, 3D 파이프 및 DRMDMA 복사) 혹은 동일한 클라이언트로부터의 기입들의 백투백(back to back) 세트들)을 발행할 때 기입 완료를 보증하는 것이 바람직하다. 후자의 경우에서의 목표는 플러시/동기화 동작의 레이턴시를 숨기는 것이다. 두번째로, 복수의 소스 GPU들로 하여금 단일 목적지 GPU에 기입을 행하도록 하는 것 및 각각의 소스에 대해 개별적으로 완료를 추적하도록 하는 것이 또한 바람직하다.

[0111] 요구된 결과를 제공하는 실시예들에 따르면, (목적지) BIF는, 각각의 세트에 대해 3 비트 단계 ID을 가진 베이스/리미트 레지스터들의 소정의 8 세트들(단계 범위)이다. 이들은 다양한 방식으로 소스 GPU들에 의해 기입될 수 있다. 예를 들면 다음과 같다.

[0112] • 최대 8개의 GPU들 각각은 그 배타적 사용을 위해 하나의 단계 범위를 보존할 수 있다. 단일 목적지 GPU 오버랩 상의 범위들을 갖는 것은 모호하다("스트림들을 크로싱함"). 더 좋은 방식은 단계가 단지 PCIe® 소스ID에 의해 결정되도록 하는 것이고, 이는 임의의 오버랩 제약을 제거한다.

[0113] • 최대 4개의 GPU들 각각은 두 개의 오버랩하지 않는 단계 범위를 보존할 수 있다.

[0114] • 2개의 GPU들 각각은 모두 8개의 단계 범위들을 사용할 수 있다(그러나, 실제로 GPU 소스 당 단지 2개만이 유용한 것으로 고려됨).

[0115] 일 실시예에서, 개별적 단계들에 할당될 수 있는 단일의 GPU에서의 트랙픽의 일곱 개의 소스가 존재한다.

[0116] • 컬러 버퍼 블럭 혹은 깊이 버퍼 블럭(Depth Buffer block)

[0117] • 셰이더 엑스포트 블럭(Shader Export block)

[0118] • 커맨드 프로세서

[0119] • DRMDMA(2개의 교번하는 단계들)

[0120] 따라서, 최종 단계 값은, 시스템에서의 GPU들의 수에 종속하는 범위 점검 결과들과 소소ID의 조합이어야 한다.

[0121] *기입 단계와 작은 애퍼처들을 결합시킴(Combining Write Phase and Small Apertures)*

[0122] 이러한 특징들(기입 단계와 작은 애퍼처들) 양쪽 모두를 동시에 고려하는 것이 가능하다. 즉, 소스 GPU로 하여금 목적지 표면 모두를 관측할 수 있도록 하고 GPU 별로 복수의 소스들을 갖도록 하는 것이 가능하다. 일 실시예는 단계별로 P2P BAR를 사용한다. 이것은 단일의 소스 GPU에 의해 사용되는 복수의 단계들을 갖는 능력을 제공하는데, 여기서 목적지 어드레스들은 단일 애퍼처들에 적합하기에는 너무 멀리 떨어져 있다. 그러나, P2P BAR를 선택하는 것은 문제가 될 수 있는데, 왜냐하면 PCIe® 공간 애퍼처는 목적지에서의 범위 점검이 동작하도록 하기에는 충분히 크지 않기 때문이다.

[0123] 한 가지 해법은 단계 범위 레지스터들을 복제하는 것, 그리고 전송 측 상에서 비교를 수행하는 것, 그리고 비교 결과를 사용하여, (a) 단계별 P2P BAR로부터의 오프셋과 함께 최종적인 단계 정보를 전송하는 것 및 (b) 요청이 해당 단계에 대한 현재 범위의 범위를 벗어날 때 P2P BAR를 업데이트 하는 것이다. P2P 애퍼처 크기는 개별적인 글로벌 레지스터를 통해 예정보다 빨리 고정된다. BAR가 움직이는 방식은, 업데이트의 빈도를 최소화하기 위해, 일정 양의 히스테리시스를 포함한다.

[0124] 해법의 두 번째 부분은, 원격 GPU 메모리 모두의 맵핑 가능성을 허용하기 위한 확장된 (36b) 가상 공간을 사용하는 것, 및 소스 BIF로 진행하는 PCIe® 공간의 분해된 뷰를 사용하는 것을 포함한다. 그러나, P2P BAR를 변경함이 없이 어드레싱될 수 있는 표면의 최대 크기는, 단계 비트들을 전송할 필요성으로 인해, 감소된다. GPU에 대한 실제 PCIe® 애퍼처가 어떤 특정 정렬을 가질 필요가 없음에 주목해야 하는바, 단계는, 합리적인 입도에 대한 이용가능한 어드레스 범위의 1/n 파티션들을 가짐으로써 (목적지 측 상에 추가적인 비교기들의 대가를 치르고) 결정된다. 물리적 표면 페이지들의 임의의 살포-수집이 목적지 HDP 페이지 테이블 혹은 GART에서 행해진다고 가정함에 주목해야한다(그래서 PCIe® 공간 내에서 표면 어드레스들은 인접하고 있음).

[0125] 도 9는 단계 콤프레션이 어떻게 동작하는지를 나타내는 다이어그램이다. 단지 두 개의 단계들만이 도시되었다. GPU 당 단지 128MB PCIe® 애퍼처 그리고 256MB의 로컬 메모리가 있다고 가정한다. 도 9A는 시스템 메모리 공간의 GPU0 뷰이다(GPU 가상 메모리 공간). 도 9B는 시스템 메모리 공간의 PCIe® 뷰이다(조직의 물리적 메모리 공간). 도 9C는 LVM의 GPU1 재구성 뷰이다(HDP 메모리 공간).

[0126] 한정된 PCIe® 애퍼처 크기를 최대한 사용하기 위해, 동일한 소스 GPU로부터의 복수의 단계들의 사용은 턴오프될 수 있다. 전형적인 커다란 표면은 2K × 2K × 32b/컴포넌트, 혹은 64MB임에 주목해야 한다. 이것은 512MB/8GPU 애퍼처에 단지 적합할 것이다. 일 실시예에서, 애퍼처의 인코딩의 적용은 전송 BIF보다 단지 앞서있다.

[0127] 본 명세서에서 설명되는 아키텍처 및 방법은 시스템을 포함하며, 이 시스템은, 호스트 브리지 버스에 결합하기 위한 복수의 특수 목적 프로세서들과; 상기 복수의 특수 목적 프로세서들 각각을 상기 복수의 특수 목적 프로세서들 중 적어도 하나의 다른 특수 목적 프로세서에 직접 결합시키는 다이렉트 버스와; 그리고 상기 복수의 특수 목적 프로세서들 중 적어도 하나에 결합된 메모리 제어기를 포함하며, 여기서, 적어도 하나의 상기 메모리 제어기가 데이터를 호스트 버스를 통해 전송할지 또는 다이렉트 버스를 통해 전송할지 여부 및 데이터를 호스트 버스를 통해 수신할지 또는 다이렉트 버스를 통해 수신할지 여부를 결정한다.

[0128] 일 실시예에서, 상기 메모리 제어기는 상기 복수의 특수 목적 프로세서들 중 적어도 하나에 상주한다.

[0129] 실시예들에서, 상기 복수의 특수 목적 프로세서들은 그래픽 프로세싱 유닛(Graphics Processing Unit, GPU)들을 포함한다.

[0130] 실시예들에서, 상기 호스트 버스는 주변 컴포넌트 인터페이스 익스프레스(Peripheral Component Interface express, PCIe®) 버스를 포함한다.

[0131] 실시예들에서, 상기 다이렉트 버스는 주변 컴포넌트 인터페이스 익스프레스(PCIe®) 버스를 포함한다.

[0132] 실시예들에서, 상기 다이렉트 버스는 독점 버스(proprietary bus)를 포함한다.

[0133] 실시예들에서, 상기 시스템은 적어도 하나의 로컬 메모리와, 적어도 하나의 시스템 메모리와, 그리고 적어도 하나의 피어 메모리(peer memory)를 더 포함하며, 여기서 상기 적어도 하나의 피어 메모리는 특수 목적 프로세서에 상주하고, 상기 메모리 제어기는 각각의 상기 메모리들의 어드레스 범위에 근거하여 어떤 메모리에 액세스할지를 결정하도록 구성된다.

[0134] 실시예들에서, 상기 피어 메모리의 범위에 어드레스가 있을 때, 상기 메모리 제어기는 상기 호스트 버스 및 상기 다이렉트 버스 각각 상의 트래픽의 양에 근거하여 어떤 포트를 사용할지를 결정한다.

[0135] 실시예들에서, 상기 피어 메모리의 범위에 어드레스가 있을 때, 상기 메모리 제어기는 상기 호스트 버스 및 상기 다이렉트 버스가 현재 가득 차 있는지 여부에 근거하여 어떤 포트를 사용할지를 결정한다.

[0136] 실시예들에서, 상기 피어 메모리의 범위에 어드레스가 있을 때, 상기 메모리 제어기의 결정은, 특정 어드레스들을 상기 호스트 버스 또는 상기 다이렉트 버스에 할당하는 것에 근거한다.

[0137] 실시예들에서, 상기 시스템은 적어도 하나의 로컬 메모리와, 적어도 하나의 시스템 메모리와, 그리고 적어도 하나의 피어 메모리를 더 포함하며, 여기서 상기 적어도 하나의 피어 메모리는 특수 목적 프로세서에 상주하고, 상기 메모리 제어기는 각각의 상기 메모리들의 어드레스 범위에 근거하여 어떤 메모리에 액세스할지를 결정하도록 구성되고, 상기 메모리 제어기는 또한, 상기 호스트 버스 및 상기 다이렉트 버스 각각 상의 트래픽에 근거하여 다이나믹 로드 밸런싱(dynamic load balancing)을 수행한다.

[0138] 실시예에서, 멀티-프로세서 시스템에서 데이터를 전달하는 방법이 제공되고, 상기 방법은, 데이터를 전송하거나 수신하는 것이 호스트 프로세서 및 복수의 특수 목적 프로세서들과 공유된 호스트 버스를 통해서인지 아니면 상기 복수의 특수 목적 프로세서들 간의 다이렉트 버스를 통해서인지를 결정하는 단계를 포함하며, 상기 결정하는 단계는 전송의 메모리 어드레스 범위에 의존한다.

[0139] 실시예들은, 특수 목적 프로세서 상에 상주하는 피어 메모리, 상기 호스트 버스 상의 시스템 메모리, 및 상기 특수 목적 프로세서들 간에 결합된 로컬 메모리 중 하나를 특정하는 어드레스 범위들을 포함한다.

[0140] 실시예들에서, 상기 범위 내에서 어드레스들을 특정하는 것은 상기 피어 메모리를 특정하는 것을 포함하고, 상기 피어 메모리를 특정하는 범위 내의 특정 어드레스들이 상기 호스트 버스 혹은 상기 다이렉트 버스 중 하나 상으로 할당된다.

[0141] 실시예들에서, 상기 피어 메모리를 특정하는 범위 내에 어드레스가 있을 때, 상기 결정은 상기 호스트 버스와 상기 다이렉트 버스 각각 상의 트래픽의 양에 근거하여 수행된다.

[0142] 일 실시예에서, 상기 방법은, 이력 트래픽 데이터(historical traffic data)에 근거하여 상기 호스트 버스와 상기 다이렉트 버스 각각 상에서의 트래픽 비율(percentage traffic)을 조정하기 위해 다이나믹 로드 밸런싱을 수행하는 단계를 더 포함한다.

[0143] 일 실시예에서, 상기 방법은, 메모리 애퍼쳐(memory aperture)들을 호스트 프로세서와, 그리고 상기 복수의 특수 목적 프로세서들 각각 상의 트랙픽 각각에, 할당하는 단계를 더 포함하고, 상기 메모리 애퍼쳐들은 상기 피어 메모리, 상기 시스템 메모리, 및 상기 로컬 메모리에 적용되고, 그리고 상기 애퍼쳐는 각각의 프로세서가 메모리를 어떻게 뷰(view)할지를 결정한다.

[0144] 일 실시예에서, 상기 방법은 애퍼쳐 콤프레션(aperture compression)을 수행하는 단계를 더 포함하며, 상기 애퍼쳐 콤프레션을 수행하는 단계는, 오프셋을 변경시킴으로써 상기 피어 메모리 모두가 가시적(visible)이게 만들기 위해 오프셋 레지스터를 사용하는 것을 포함한다.

[0145] 일 실시예에서, 명령들은, 상기 방법을 수행하는 애플리케이션 특정 집적 회로(Application Specific Integrated Circuit, ASIC)를 생성하는데 사용가능한 하드웨어 기술 언어 명령(hardware description language instruction)들을 포함한다.

[0146] 본 명세서에서 설명되는 아키텍처 및 방법은 또한 컴퓨터 판독가능 매체를 포함하고, 상기 컴퓨터 판독가능 매체 상에는 명령들이 저장되어 있고, 상기 명령들은, 실행될 때, 멀티-프로세서 시스템에서 데이터를 전달하는 방법이 수행되도록 하며, 상기 방법은, 데이터를 전송하거나 수신하는 것이 호스트 프로세서 및 복수의 특수 목

적 프로세서들과 공유된 호스트 버스를 통해서인지 아니면 상기 복수의 특수 목적 프로세서들 간의 다이렉트 버스를 통해서인지를 결정하는 단계를 포함하며, 상기 결정하는 단계는 전송의 메모리 어드레스 범위에 의존한다.

[0147] 일 실시예에서, 상기 어드레스 범위들은 특수 목적 프로세서 상에 상주하는 피어 메모리, 상기 호스트 버스 상의 시스템 메모리, 및 상기 특수 목적 프로세서에 결합된 로컬 메모리 중 하나를 특정한다.

[0148] 일 실시예에서, 상기 범위 내에서 어드레스들을 특정하는 것은 상기 피어 메모리를 특정하는 것을 포함하고, 상기 피어 메모리를 특정하는 범위 내의 특정 어드레스들이 상기 호스트 버스 혹은 상기 다이렉트 버스 중 하나 상으로 할당된다.

[0149] 일 실시예에서, 상기 피어 메모리를 특정하는 범위 내에 어드레스가 있을 때, 상기 결정은 상기 호스트 버스와 상기 다이렉트 버스 각각 상의 트래픽의 양에 근거하여 수행된다.

[0150] 일 실시예에서, 상기 방법은, 이력 트래픽 데이터에 근거하여 상기 호스트 버스와 상기 다이렉트 버스 각각 상에서의 트래픽 비율을 조정하기 위해 다이나믹 로드 밸런싱을 수행하는 단계를 더 포함한다.

[0151] 일 실시예에서, 상기 방법은, 메모리 애퍼처들을 호스트 프로세서와, 그리고 상기 복수의 특수 목적 프로세서들 각각 상의 트랙픽 각각에, 할당하는 단계를 더 포함하고, 상기 메모리 애퍼처들은 상기 피어 메모리, 상기 시스템 메모리, 및 상기 로컬 메모리에 적용되고, 그리고 상기 애퍼처는 각각의 프로세서가 메모리를 어떻게 뷰할지를 결정한다.

[0152] 일 실시예에서, 상기 방법은, 애퍼처 콤프레션을 수행하는 단계를 더 포함하며, 상기 애퍼처 콤프레션을 수행하는 단계는, 오프셋을 변경시킴으로써 상기 로컬 메모리 모두가 가시적이게 만들기 위해 오프셋 레지스터를 사용하는 것을 포함한다.

[0153] 일 실시예에서, 상기 명령들은, 상기 방법을 수행하는 애플리케이션 특정 집적 회로(Application Specific Integrated Circuit, ASIC)를 생성하는데 사용가능한 하드웨어 기술 언어 명령들을 포함한다.

[0154] 본 명세서에서 설명되는 아키텍처 및 방법은 또한, 멀티-프로세서 시스템에서 데이터를 전달하는 방법을 포함하고, 상기 방법은, 데이터를 전송하거나 수신하는 것이 호스트 프로세서 및 복수의 특수 목적 프로세서들과 공유된 호스트 버스를 통해서인지 아니면 상기 복수의 특수 목적 프로세서들 간의 다이렉트 버스를 통해서인지를 결정하는 단계와, 여기서 상기 결정하는 단계는 전송의 메모리 어드레스 범위에 의존하며; 그리고 상기 다이렉트 버스를 통해 프로세서들 간에 데이터를 직접 전달하는 단계를 포함하고, 상기 특수 목적 프로세서는 프레임 버퍼 애퍼처(frame buffer aperture)들을 갖는 그래픽 프로세싱 유닛들을 포함하고, 상기 데이터를 직접 전달하는 단계는 상기 다이렉트 버스를 통해 프레임 버퍼 애퍼처 밖의 액세스를 제공하는 것을 포함한다.

[0155] 앞서 설명된 실시예들의 실시형태들은 다양한 회로 중 어느 하나에 프로그래밍된 기능으로서 구현될 수 있고, 이러한 것으로는, 프로그래밍가능 로직 디바이스(Programmable Logic Device, PLD)들(예를 들어, 필드 프로그래밍가능 게이트 어레이(Field Programmable Gate Array, FPGA)들, 프로그래밍가능 어레이 로직(Programmable Array Logic, PAL) 디바이스들, 전기적으로 프로그래밍가능 로직 및 메모리 디바이스들), 그리고 표준 셀 기반의 디바이스들, 뿐만 아니라 애플리케이션 특정 집적 회로(Application Specific Integrated Circuit, ASIC)들, 그리고 풀리 커스텀 집적 회로(fully custom integrated circuit)들이 있지만, 이러한 것들로만 한정되는 것은 아니다. 이 실시예들의 실시형태를 구현하는 어떤 다른 가능한 것으로는, 메모리(예를 들어, 전자적으로 소거가능한 프로그래밍가능 판독 전용 메모리(Electronically Erasable Programmable Read Only Memory, EEPROM), 플래시 메모리, 등)를 구비한 마이크로제어기들, 임베디드 마이크로프로세서, 펌웨어, 소프트웨어, 등이 있다. 더욱이, 이 실시예들의 실시형태는, 소프트웨어 기반의 회로 에뮬레이션, 이산 로직(순차형 및 조합형), 커스텀 디바이스(custom device)들, 퍼지(fuzzy)(뉴럴(neural)) 로직, 양자 디아비스(quantum device)들, 및 앞서의 디바이스 타입들 중 임의 타입의 하이브리드(hybrid)들을 구비한 마이크로프로세서들로 구현될 수 있다. 물론, 기반이 되는 디바이스 기술들이 다양한 컴포넌트 타입들로 제공될 수 있는바, 예를 들어, 금속 산화물 반도체 전계 효과 트랜지스터(Metal-Oxide Semiconductor Field-Effect Transistor, MOSFET) 기술(예를 들어, 상보형 금속 산화물 반도체(Complementary Metal-Oxide Semiconductor, CMOS)), 바이폴라 기술(bipolar technologies)(예를 들어, 이미터 커플링 로직(Emitter-Coupled Logic, ECL)), 폴리머 기술(polymer technologies)(예를 들어, 실리콘-컨쥬게이션 폴리머(silicon-conjugated polymer) 및 금속-컨쥬게이션 폴리머-금속 구조), 아날로그 및 디지털 혼합형 등이 있다.

[0156] 본 명세서 및 청구항에서 사용되는 바와 같은 용어 "프로세서"는 프로세서 코어 혹은 프로세서의 일부를 포함한다. 더욱이, 비록 하나 이상의 GPU들 및 하나 이상의 CPU들이 일반적으로 본 명세서에서 분리되어

언급되었지만, 실시예들에서, GPU와 CPU 양쪽 모두는 단일 집적 회로 패키지에 포함되거나 단일의 모놀리식 다이(single monolithic die) 상에 포함된다. 따라서, 단일의 디바이스가 특허 청구 범위에서 청구된 방법을 이러한 실시예들에서 수행한다.

[0157] 문맥상 다른 방식으로 명확히 필요한 경우가 아니라면, 본 명세서 및 청구항 전체에 걸쳐, 용어 "포함한다", "포함하는" 등은, 배타적 의미 혹은 완벽히 구비된의 의미가 아닌, 포괄적 의미로서 해석돼야만 하는바, 즉, "포함하지만 이러한 것에만 한정되는 것은 아님"의 의미로 해석돼야만 한다. 단수 혹은 복수를 사용하는 용어들은 또한 각각 복수 혹은 단수의 의미도 포괄한다. 추가적으로, 용어 "본 명세서에서", "이하에서", "앞서", "아래에서" 등 유사한 의미의 용어는, 본 명세서에서 사용되는 경우, 본 명세서를 전체적으로 언급하는 것이지, 본 명세서의 어떤 특정 부분을 말하는 것이 아니다. 용어 "또는(혹은)"은 둘 또는 그 이상의 아이템들의 리스트를 참조하는데 사용되며, 이 용어의 해석은 다음과 같이 해석되는 것을 포괄하는바, 즉 리스트 내의 아이템들 중 어느 하나, 리스트 내의 아이템들 모두, 및 리스트 내의 아이템들의 임의의 조합을 모두 포괄한다.

[0158] 본 발명의 방법 및 시스템의 예시된 실시예들의 앞서의 설명을 통해 본 발명을 이처럼 개시되는 형태에 정확히 한정하려는 것이 아니며, 또는 본 발명의 실시예 모두를 말하려는 것도 아니다. 본 발명의 방법 및 시스템의 특정 실시예 및 그 예들은 예시적 목적으로 본 명세서에서 설명되며, 다양한 등가적 수정이, 종래의 관련 기술 분야에서 숙련된 자들이 인식하는 바와 같이, 본 발명의 범위 내에서 가능하다. 본 명세서에서 제공되는 공개된 가르침이 다른 시스템들에도 적용될 수 있는바, 앞서 설명된 바와 같은 그래픽 프로세싱 혹은 비디오 프로세싱을 포함하는 시스템에만 단지 적용되는 것은 아니다. 설명되는 다양한 동작들은 매우 광범위한 아키텍처로 수행될 수 있고, 설명된 바와는 다르게 분포될 수 있다. 추가적으로, 비록 많은 구성이 본 명세서에서 설명되었지만, 그 어떤 것도 한정적 의미 또는 배타적 의미로 사용되도록 의도되지 않았다.

[0159] 다른 실시예들에서, 본 명세서에서 설명되는 하드웨어 및 소프트웨어 능력 중 일부 혹은 모두는, 프린터, 카메라, 텔레비젼, 디지털 다기능 디스크(Digital Versatile Disc, DVD) 플레이어, DVR 혹은 PVR, 핸드헬드 디바이스(handheld device), 모발일 전화기, 혹은 어떤 다른 디바이스에 존재할 수 있다. 앞서 설명된 다양한 실시예들의 구성요소들 및 동작들이 결합되어 또 다른 실시예들을 제공할 수 있다. 앞서의 상세한 설명에 비추어 본 발명의 방법 및 시스템에 기타의 변경이 행해질 수 있다.

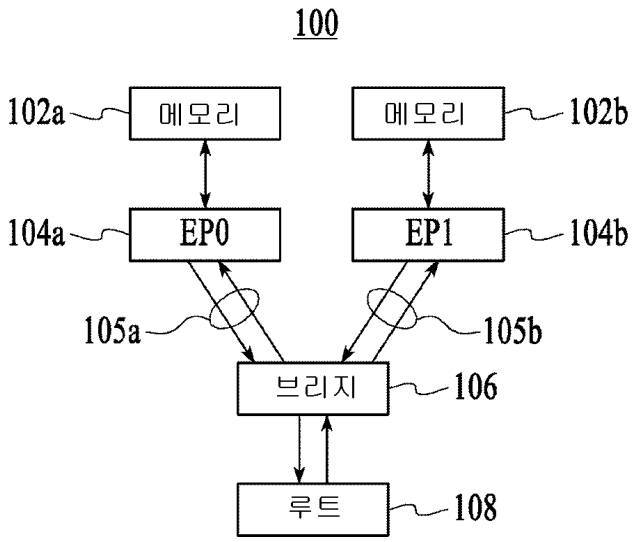
[0160] 일반적으로, 다음의 청구항들에서, 사용되는 용어들은 본 발명의 방법 및 시스템을 명세서 및 청구항들에서의 개시되는 특정 실시예들에 한정시키기 위한 것으로 해석되어서는 안 되며, 특허청구범위 하의 임의의 프로세싱 시스템 및 방법을 포함하도록 해석되어야 한다. 따라서, 본 발명의 방법 및 시스템이 이러한 개시에 의해 한정되지 않으며, 대신에 본 발명의 방법 및 시스템의 범위는 특허청구범위에 의해 전체적으로 결정돼야한다.

[0161] 본 발명의 방법 및 시스템의 특정 실시형태가 특정 청구 형태로 아래에서 제시되지만, 본 발명자는 임의 개수의 청구 형태로 본 발명의 방법 및 시스템의 다양한 실시형태들을 고려하고 있다. 예를 들어, 본 발명의 방법 및 시스템의 단지 일 실시형태만이 컴퓨터 판독가능 매체에서 구현되는 것으로 기재될 수 있지만, 다른 실시형태들도 마찬가지로 컴퓨터 판독가능 매체에서 구현될 수 있다. 이러한 컴퓨터 판독가능 매체들은 컴퓨팅 디바이스(예를 들어, 개인용 컴퓨터, PDA(Personal Digital Assistant), PVR, 모바일 디바이스 등)에 의해 실행될 명령들을 저장할 수 있거나, 또는 그 동작시 앞서 설명된 실시형태들을 수행하는 디바이스(GPU, ASIC, 등) 혹은 소프트웨어 애플리케이션을 그 실행시 생성하도록 설계된 명령들(예를 들어, 베리로그(Verilog) 혹은 하드웨어 기술 언어(hardware desciiption language))일 수 있다. 청구된 발명은, GDSⅡ 데이터(혹은 그 등가물)를 발생시키기 위해, 생성되고, 저장되고, 합성되고, 그리고 사용되는 컴퓨터 코드(예를 들어, HDL, 베리로그, 등)로 구현될 수 있다. ASIC가 이후 이러한 데이터를 근거로 제조될 수 있다.

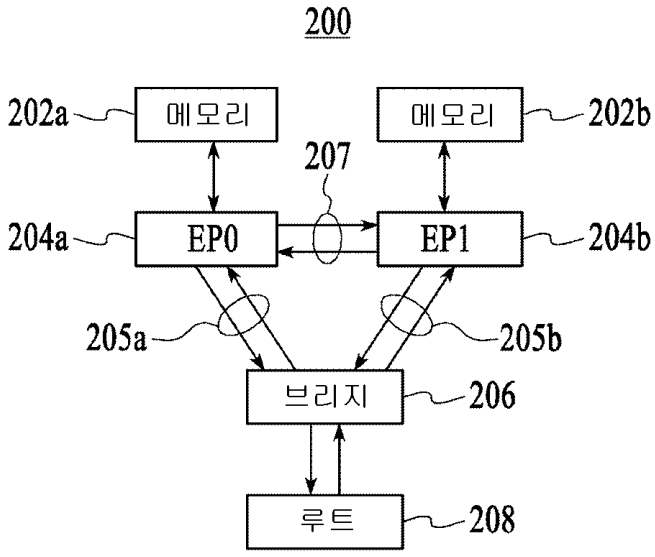
[0162] 따라서, 본 발명자는 본 발명의 방법 및 시스템의 다른 실시형태에 대한 이러한 추가적인 청구 형태를 추구하기 위해 본원의 출원 이후에 추가적인 청구항들을 부가할 권리를 유보하는 바이다.

도면

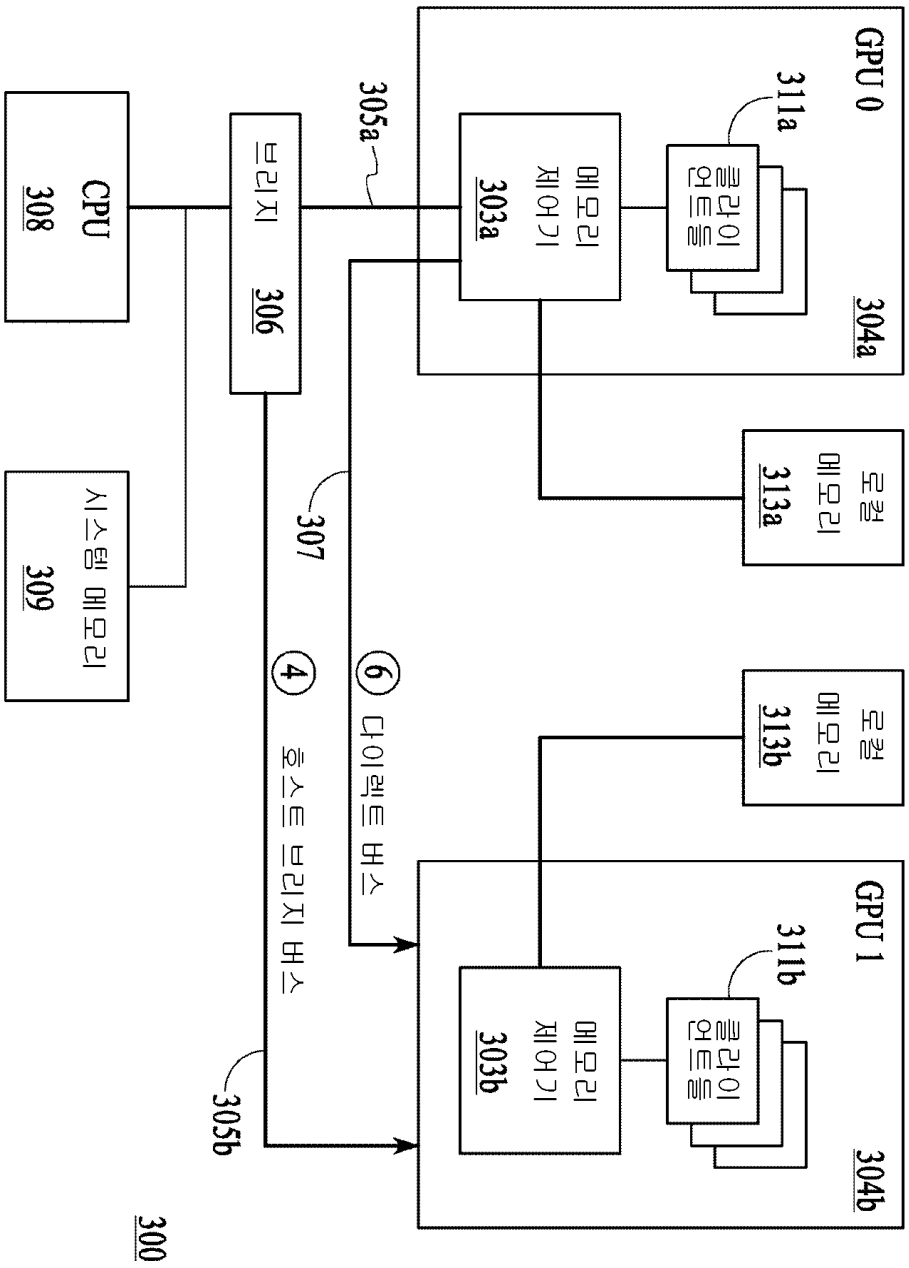
도면1


100
102a
메모리
메모리
102b
104a
EP0
EP1
104b
105a
105b
브리지
106
루트
108

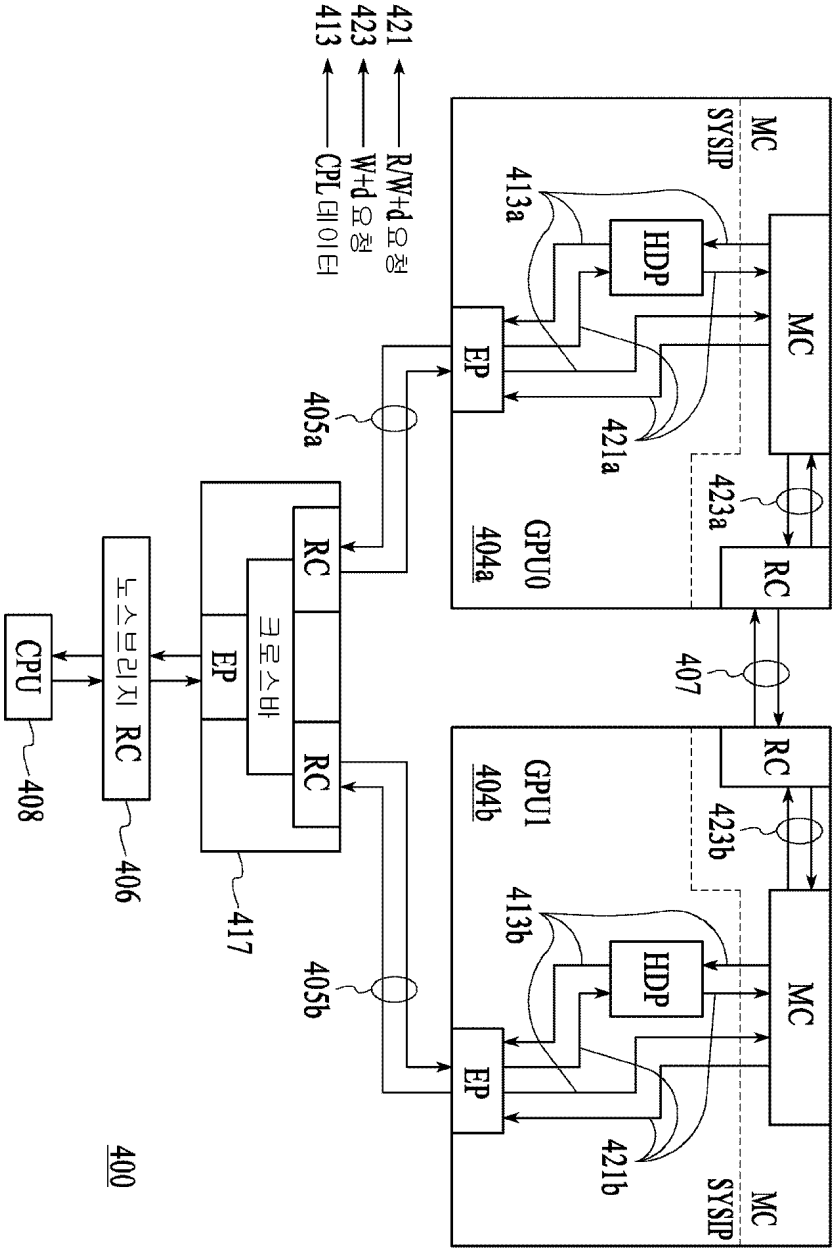
도면2


200
202a
메모리
메모리
202b
207
204a
EP0
EP1
204b
205a
205b
브리지
206
루트
208

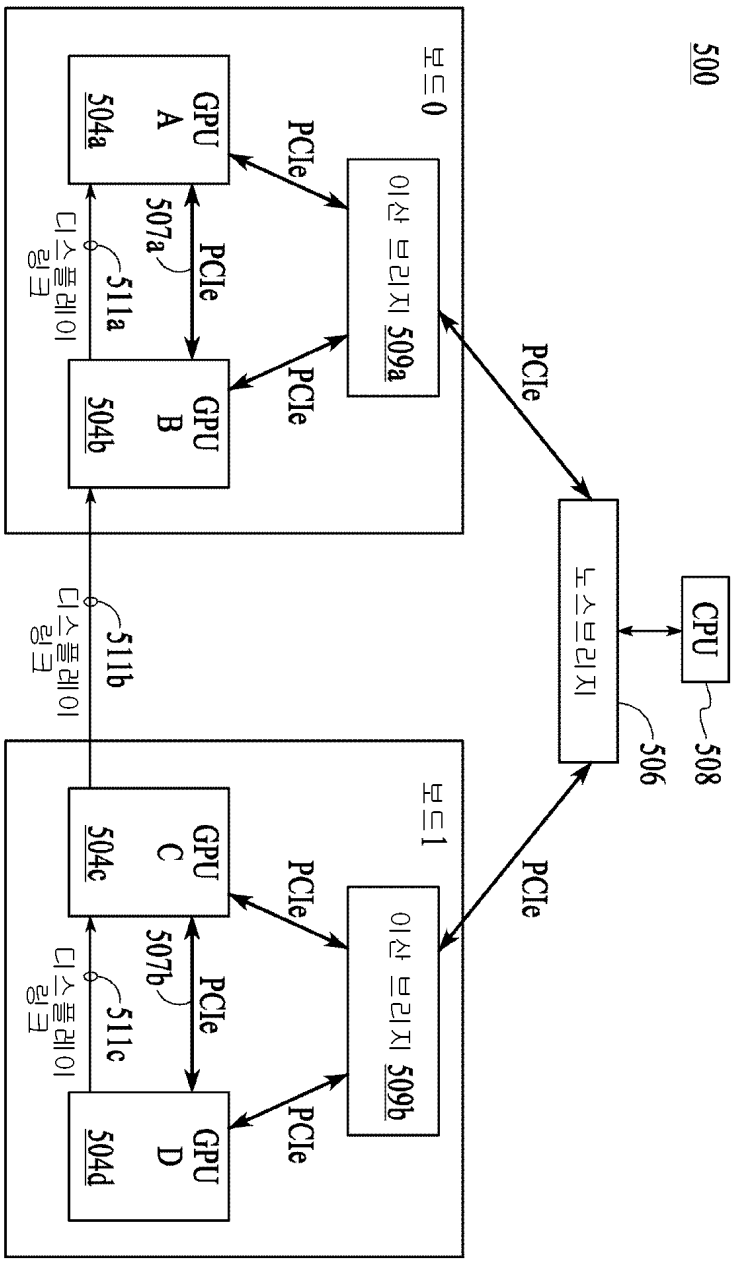
도면3


GPU 0
304a
311a
클라이언트들
메모리 제어기
303a
305a
로컬 메모리
313a
CPU
308
브리지
306
시스템 메모리
309
307
4
호스트 브리지 버스
6
다이렉트 버스
305b
로컬 메모리
313b
GPU 1
304b
311b
클라이언트들
메모리 제어기
303b
300

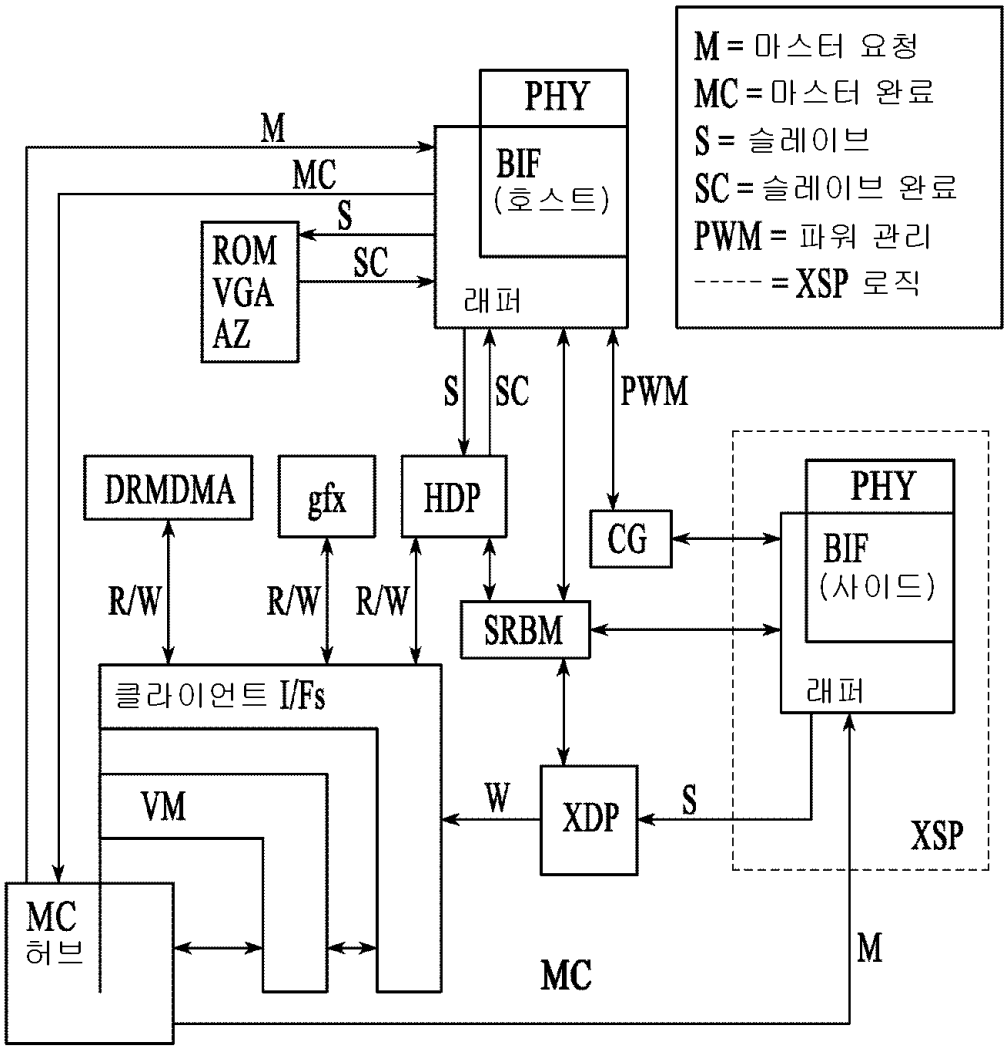
도면4


421
423
413
R/W+d 요청
W+d 요청
CPL 데이터
SYSIP
MC
413a
HDP
MC
EP
421a
405a
423a
RC
GPU0
404a
407
RC
423b
GPU1
404b
413b
HDP
MC
EP
421b
405b
SYSIP
MC
RC
크로스바
RC
EP
417
CPU
408
노스브리지 RC
406
400

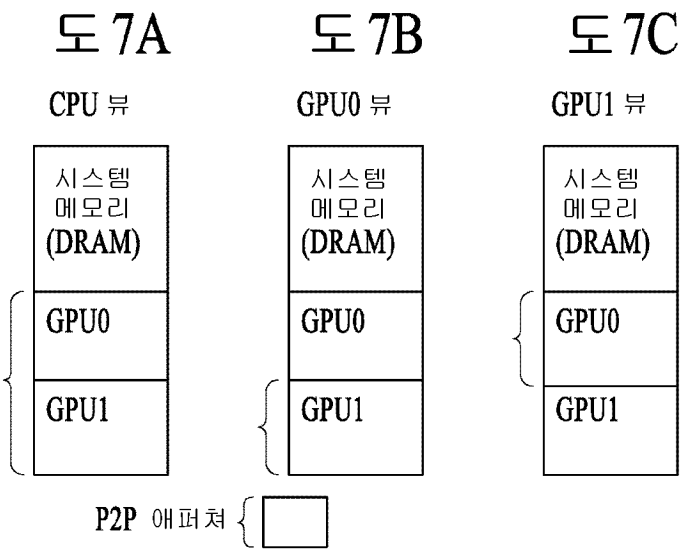
도면5


500
보드 0
GPU A
504a
GPU B
504b
이산 브리지 509a
PCIe
507a
디스플레이 링크
511a
511b
보드 1
GPU C
504c
GPU D
504d
이산 브리지 509b
507b
511c
노스브리지
506
CPU
508

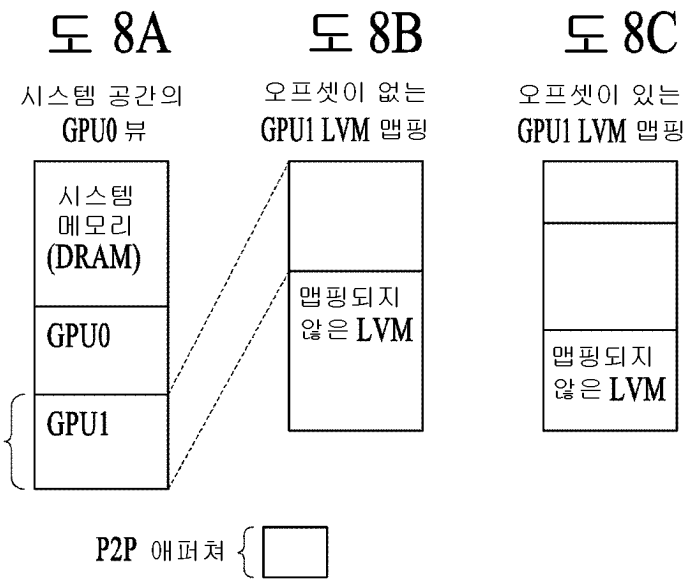
도면6


M = 마스터 요청
MC = 마스터 완료
S = 슬레이브
SC = 슬레이브 완료
PWM = 파워 관리
----- = XSP 로직
PHY
BIF
(호스트)
래퍼
M
MC
S
SC
ROM
VGA
AZ
S
SC
PWM
DRMDMA
gfx
HDP
CG
PHY
BIF
(사이드)
래퍼
R/W
R/W
R/W
SRBM
클라이언트 I/Fs
VM
W
XDP
S
XSP
MC
허브
MC
M

도면7


도 7A
도 7B
도 7C
CPU 뷰
GPU0 뷰
GPU1 뷰
시스템
메모리
(DRAM)
GPU0
GPU1
시스템
메모리
(DRAM)
GPU0
GPU1
시스템
메모리
(DRAM)
GPU0
GPU1
P2P 애퍼쳐

도면8


도 8A
시스템 공간의
GPU0 뷰
시스템
메모리
(DRAM)
GPU0
GPU1
도 8B
오프셋이 없는
GPU1 LVM 맵핑
맵핑되지
않은 LVM
도 8C
오프셋이 있는
GPU1 LVM 맵핑
맵핑되지
않은 LVM
P2P 애퍼쳐

도면9

도 9A

도 9B

도 9C