(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년07월22일
(11) 등록번호 10-1642105
(24) 등록일자 2016년07월18일

(51) 국제특허분류(Int. Cl.)
G06F 9/50 (2006.01) *G06F 9/38* (2006.01)
G06F 9/48 (2006.01) *G06T 1/20* (2006.01)
(21) 출원번호 10-2012-7008414
(22) 출원일자(국제) 2010년09월03일
심사청구일자 2015년09월02일
(85) 번역문제출일자 2012년03월30일
(65) 공개번호 10-2012-0064097
(43) 공개일자 2012년06월18일
(86) 국제출원번호 PCT/US2010/047786
(87) 국제공개번호 WO 2011/028986
국제공개일자 2011년03월10일
(30) 우선권주장
12/874,134 2010년09월01일 미국(US)
61/239,712 2009년09월03일 미국(US)
(56) 선행기술조사문헌
US20080109810 A1
US20080074433 A1
US5371849 A
US6252600 B1

(73) 특허권자
어드밴스드 마이크로 디바이시즈, 인코포레이티드
미국 캘리포니아 94088-3453 서니베일 피.오.박스 3453 원 에이엠디 플레이스
(72) 발명자
맨토 마이클
미국 플로리다 32825 올란도 피나 드라이브 1620
맥라리 렉스
미국 플로리다 32765 오비에도 섬머 오크 코트 758
(74) 대리인
박장원

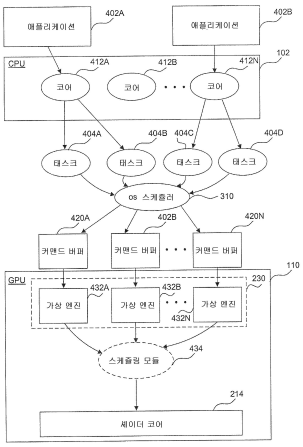
전체 청구항 수 : 총 27 항 심사관 : 유진태

(54) 발명의 명칭 **셰이더 코어 상에서 서로 다른 타입의 태스크들의 비동기 동시 디스패치를 가능하게 해주는 복수의 버퍼들을 구비한 커맨드 프로세서를 포함하는 그래픽 처리 유닛**

(57) 요 약

복수의 가상 엔진들 및 셰이더 코어를 포함하는 처리 유닛이 개시된다. 복수의 가상 엔진들은, (i) 운영 시스템(OS)으로부터, 복수의 태스크들을 실질적으로 서로 병렬로 수신하고 그리고 (ii) 복수의 태스크들 각각과 관련된 상태 데이터의 세트를 로딩하도록 되어 있다. 셰이더 코어는 복수의 태스크들 각각과 관련된 상태 데이터의 세트에 근거하여 실질적으로 병렬로 복수의 태스크들을 실행하도록 되어있다. 처리 유닛은 또한 셰이더 코어에 발행될 복수의 태스크들을 스케줄링하는 스케줄링 모듈을 포함할 수 있다.

대 표 도 - 도4



명 세 서

청구범위

청구항 1

컴퓨팅 시스템으로서,

제1 처리 유닛과 관련되고, 그리고 복수의 태스크들을 제2 처리 유닛과 관련된 스케쥴링 모듈(scheduling module)로부터 수신하도록 그리고 상기 복수의 태스크들 각각과 관련된 상태 데이터를 로딩하도록 된 복수의 엔진들과; 그리고

상기 제1 처리 유닛과 관련되고, 그리고 상기 복수의 태스크들을 상기 복수의 엔진들 중 적어도 하나로부터 수신하도록 그리고 제1 태스크 및 제2 태스크 각각과 관련된 각 상태 데이터에 기초하여 상기 복수의 태스크들로부터 제2 태스크를 실행하는 동안 상기 복수의 태스크들로부터 제1 태스크를 실행하도록 된 셰이더 코어(shader core)를 포함하는 것을 특징으로 하는 컴퓨팅 시스템.

청구항 2

제1 항에 있어서,

상기 셰이더 코어는,

상기 제1 태스크를 실행하도록 된 제1 복수의 처리 소자들 - 상기 제1 태스크는 상기 제1 태스크와 관련된 상태 데이터의 제1 세트에 기초하여 실행되며 - 과; 그리고

상기 제2 태스크를 실행하도록 된 제2 복수의 처리 소자들을 포함하고,

상기 제2 태스크는 상기 제2 태스크와 관련된 상태 데이터의 제2 세트에 기초하여 실행되는 것을 특징으로 하는 컴퓨팅 시스템.

청구항 3

제1 항에 있어서,

상기 복수의 엔진들은, 상기 제1 태스크를 수신하도록 된 제1 큐와, 그리고 상기 제2 태스크를 수신하도록 된 제2 큐를 포함하는 것을 특징으로 하는 컴퓨팅 시스템.

청구항 4

제1 항에 있어서,

상기 제1 태스크는 저-레이턴시(low-latency) 태스크를 포함하고 상기 제2 태스크는 정규 레이턴시(regular-latency) 태스크를 포함하는 것을 특징으로 하는 컴퓨팅 시스템.

청구항 5

제1 항에 있어서,

상기 제1 태스크는 그래픽-처리 태스크를 포함하고 상기 제2 태스크는 일반-컴퓨팅 태스크를 포함하는 것을 특징으로 하는 컴퓨팅 시스템.

청구항 6

제1 항에 있어서,

상기 셰이더 코어는 상기 복수의 태스크들 각각에 관련된 우선권(priority)에 기초하여 상기 복수의 태스크들을 수신하도록 된 것을 특징으로 하는 컴퓨팅 시스템.

청구항 7

제1 항에 있어서,

상기 셰이더 코어는 복수의 처리 소자들을 포함하고,

상기 처리 소자들 각각은 상기 복수의 엔진들로부터의 각 엔진에 대응하고, 상기 각 엔진으로부터 하나 이상의 태스크들을 실행하도록 된 것을 특징으로 하는 컴퓨팅 시스템.

청구항 8

제1 항에 있어서,

상기 셰이더 코어는 복수의 처리 소자들을 포함하고,

상기 처리 소자들 중 적어도 하나는, 제1 처리 시간을 상기 제1 태스크에 할당하고(dedicate), 제2 처리 시간을 상기 제2 태스크에 할당하도록 된 것을 특징으로 하는 컴퓨팅 시스템.

청구항 9

컴퓨팅 디바이스 상에서 실행시 컴퓨팅 시스템을 정의(define)하는 명령어들이 내포된 컴퓨터-판독가능 저장 매체로서, 상기 컴퓨팅 시스템은,

제1 처리 유닛과 관련되고, 그리고 복수의 태스크들을 제2 처리 유닛과 관련된 스케쥴링 모듈로부터 수신하도록 그리고 상기 복수의 태스크들 각각과 관련된 상태 데이터를 로딩하도록 된 복수의 엔진들과; 그리고

상기 제1 처리 유닛과 관련되고, 그리고 상기 복수의 태스크들을 상기 복수의 엔진들 중 적어도 하나로부터 수신하도록 그리고 제1 태스크 및 제2 태스크 각각과 관련된 각 상태 데이터에 기초하여 상기 복수의 태스크들로부터 제2 태스크를 실행하는 동안 상기 복수의 태스크들로부터 제1 태스크를 실행하도록 된 셰이더 코어를 포함하는 것을 특징으로 하는 컴퓨터-판독가능 저장 매체.

청구항 10

제9 항에 있어서,

상기 셰이더 코어는,

상기 제1 태스크를 실행하도록 된 제1 복수의 처리 소자들 - 상기 제1 태스크는 상기 제1 태스크와 관련된 상태 데이터의 제1 세트에 기초하여 실행되며 - 과; 그리고

상기 제2 태스크를 실행하도록 된 제2 복수의 처리 소자들을 포함하고,

상기 제2 태스크는 상기 제2 태스크와 관련된 상태 데이터의 제2 세트에 기초하여 실행되는 것을 특징으로 하는 컴퓨터-판독가능 저장 매체.

청구항 11

제9 항에 있어서,

상기 복수의 엔진들은, 상기 제1 태스크를 수신하도록 된 제1 큐와, 그리고 상기 제2 태스크를 수신하도록 된 제2 큐를 포함하는 것을 특징으로 하는 컴퓨터-판독가능 저장 매체.

청구항 12

제9 항에 있어서,

상기 제1 태스크는 저-레이턴시 태스크를 포함하고 상기 제2 태스크는 정규 레이턴시 태스크를 포함하는 것을 특징으로 하는 컴퓨터-판독가능 저장 매체.

청구항 13

제9 항에 있어서,

상기 제1 태스크는 그래픽-처리 태스크를 포함하고 상기 제2 태스크는 일반-컴퓨팅 태스크를 포함하는 것을 특징으로 하는 컴퓨터-판독가능 저장 매체.

청구항 14

제9 항에 있어서,

상기 셰이더 코어는 상기 복수의 태스크들 각각에 관련된 우선권에 기초하여 상기 복수의 태스크들을 수신하도록 된 것을 특징으로 하는 컴퓨터-판독가능 저장 매체.

청구항 15

컴퓨팅 시스템으로서,

메모리와;

제1 처리 유닛과; 그리고

상기 메모리 및 상기 제1 처리 유닛에 연결된 버스를 포함하며,

상기 제1 처리 유닛은,

복수의 태스크들을 제2 처리 유닛과 관련된 스케쥴링 모듈로부터 수신하도록 그리고 상기 복수의 태스크들 각각과 관련된 상태 데이터를 로딩하도록 된 복수의 엔진들과; 그리고

상기 복수의 태스크들을 상기 복수의 엔진들 중 적어도 하나로부터 수신하도록 그리고 제1 태스크 및 제2 태스크 각각과 관련된 각 상태 데이터에 기초하여 상기 복수의 태스크들로부터 상기 제2 태스크를 실행하는 동안 상기 복수의 태스크들로부터 상기 제1 태스크를 실행하도록 된 셰이더 코어를 포함하는 것을 특징으로 하는 컴퓨팅 시스템.

청구항 16

제15 항에 있어서,

상기 셰이더 코어는,

상기 제1 태스크를 실행하도록 된 제1 복수의 처리 소자들 - 상기 제1 태스크는 상기 제1 태스크와 관련된 상태 데이터의 제1 세트에 기초하여 실행되며 - 과; 그리고

상기 제2 태스크를 실행하도록 된 제2 복수의 처리 소자들을 포함하고,

상기 제2 태스크는 상기 제2 태스크와 관련된 상태 데이터의 제2 세트에 기초하여 실행되는 것을 특징으로 하는 컴퓨팅 시스템.

청구항 17

제15 항에 있어서,

상기 복수의 엔진들은, 상기 제1 태스크를 수신하도록 된 제1 큐와, 그리고 상기 제2 태스크를 수신하도록 된 제2 큐를 포함하는 것을 특징으로 하는 컴퓨팅 시스템.

청구항 18

제15 항에 있어서,

상기 제1 태스크는 저-레이턴시 태스크를 포함하고 상기 제2 태스크는 정규 레이턴시 태스크를 포함하는 것을 특징으로 하는 컴퓨팅 시스템.

청구항 19

제15 항에 있어서,

상기 제1 태스크는 그래픽-처리 태스크를 포함하고 상기 제2 태스크는 일반-컴퓨팅 태스크를 포함하는 것을 특징으로 하는 컴퓨팅 시스템.

청구항 20

제15 항에 있어서,

상기 셰이더 코어는 상기 복수의 태스크들 각각에 관련된 우선권에 기초하여 상기 복수의 태스크들을 수신하도록 된 것을 특징으로 하는 컴퓨팅 시스템.

청구항 21

컴퓨팅 시스템에서 태스크들을 처리하기 위한, 컴퓨터로 구현되는 방법으로서,

제1 처리 유닛과 관련된 복수의 엔진들로, 복수의 태스크들을 수신하는 단계 - 상기 복수의 엔진들은 상기 복수의 태스크들을 제2 처리 유닛과 관련된 스케쥴링 모듈로부터 수신하고 - 와;

상기 복수의 태스크들 각각과 관련된 상태 데이터를 로딩하는 단계와;

상기 제1 처리 유닛과 관련된 셰이더 코어로, 상기 복수의 태스크들을 상기 복수의 엔진들 중 적어도 하나로부터 수신하는 단계와; 그리고

제1 태스크 및 제2 태스크 각각과 관련된 상태 데이터에 기초하여 상기 복수의 태스크들로부터 상기 제2 태스크를 실행하는 동안 상기 복수의 태스크들로부터 상기 제1 태스크를, 상기 셰이더 코어로 실행하는 단계를 포함하는 것을 특징으로 하는 컴퓨팅 시스템에서 태스크들을 처리하기 위한, 컴퓨터로 구현되는 방법.

청구항 22

제21 항에 있어서,

상기 실행하는 단계는,

상기 제1 태스크와 관련된 상태 데이터의 제1 세트에 기초하여 상기 셰이더 코어의 제1 복수의 처리 소자들에서 상기 제1 태스크를 실행하는 단계와; 그리고

상기 제2 태스크와 관련된 상태 데이터의 제2 세트에 기초하여 상기 셰이더 코어의 제2 복수의 처리 소자들에서 상기 제2 태스크를 실행하는 단계를 포함하는 것을 특징으로 하는 컴퓨팅 시스템에서 태스크들을 처리하기 위한, 컴퓨터로 구현되는 방법.

청구항 23

제21 항에 있어서,

상기 복수의 엔진들로, 상기 복수의 태스크들을 수신하는 단계는,

상기 제1 태스크를 제1 큐에 큐잉하는 단계와; 그리고

상기 제2 태스크를 제2 큐에 큐잉하는 단계를 포함하는 것을 특징으로 하는 컴퓨팅 시스템에서 태스크들을 처리하기 위한, 컴퓨터로 구현되는 방법.

청구항 24

제21 항에 있어서,

상기 제1 태스크는 저-레이턴시 태스크를 포함하고 상기 제2 태스크는 정규 레이턴시 태스크를 포함하는 것을 특징으로 하는 컴퓨팅 시스템에서 태스크들을 처리하기 위한, 컴퓨터로 구현되는 방법.

청구항 25

제21 항에 있어서,

상기 제1 태스크는 그래픽-처리 태스크를 포함하고 상기 제2 태스크는 일반-컴퓨팅 태스크를 포함하는 것을 특징으로 하는 컴퓨팅 시스템에서 태스크들을 처리하기 위한, 컴퓨터로 구현되는 방법.

청구항 26

제21 항에 있어서,

상기 셰이더 코어로, 상기 복수의 태스크들을 상기 복수의 엔진들 중 적어도 하나로부터 수신하는 단계는, 상기

셰이더 코어로, 상기 복수의 태스크들 각각에 관련된 우선권에 기초하여 상기 복수의 태스크들을 수신하는 단계를 포함하는 것을 특징으로 하는 컴퓨팅 시스템에서 태스크들을 처리하기 위한, 컴퓨터로 구현되는 방법.

청구항 27

컴퓨팅 시스템에서 태스크들을 처리하기 위한, 컴퓨터로 구현되는 방법으로서,

제1 처리 유닛으로부터, 하나 이상의 애플리케이션들 중에서 복수의 태스크들을 수신하는 단계 - 각 태스크는 우선권 타입의 표시를 포함하고 - 와; 그리고

상기 복수의 태스크들 및 각각의 태스크와 관련된 상기 우선권 타입의 표시를 제2 처리 유닛에 제공하는 단계를 포함하고,

상기 제2 처리 유닛은:

상기 복수의 태스크들을 수신하도록 그리고 상기 복수의 태스크들 각각과 관련된 상태 데이터를 로딩하도록 된 복수의 엔진들과; 그리고

상기 복수의 태스크들을 상기 복수의 엔진들 중 적어도 하나로부터 수신하도록 그리고 제1 태스크 및 제2 태스크 각각과 관련된 각 상태 데이터에 기초하여 상기 복수의 태스크들로부터 상기 제2 태스크를 실행하는 동안 상기 복수의 태스크들로부터 상기 제1 태스크를 실행하도록 된 셰이더 코어를 포함하는 것을 특징으로 하는 컴퓨팅 시스템에서 태스크들을 처리하기 위한, 컴퓨터로 구현되는 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 일반적으로 컴퓨터 시스템에서 수행되는 컴퓨팅 동작(computing operation)에 관한 것이다. 보다 구체적으로, 본 발명은 컴퓨팅 동작을 수행하는 그래픽-처리 유닛(GPU)과 같은 처리 유닛 및 그 응용에 관한 것이다.

배 경 기 술

[0002] GPU는 그래픽-처리 태스크들과 같은 데이터-병렬 컴퓨팅 태스크들을 수행하도록된 복합 집적 회로이다. GPU는, 예를 들어, 비디오 게임 애플리케이션과 같은 최종 사용자 애플리케이션에 의해 요구되는 그래픽 처리 태스크들을 실행할 수 있다. GPU는 개별(즉, 분리된) 디바이스 및/또는 패키지이거나 또 다른 프로세서(예를 들어, 중앙 처리 유닛(CPU))로서 동일한 디바이스 및/또는 패키지에 포함될 수 있다. 예를 들어, GPU들은, 예컨대 노스브리지 디바이스들과 같은 라우팅 또는 브리지 디바이스에 종종 통합된다.

[0003] 최종 사용자 애플리케이션과 GPU 사이에는 몇개의 소프트웨어 계층들이 존재한다. 최종 사용자 애플리케이션은 애플리케이션-프로그래밍 인터페이스(API)와 통신한다. API는 최종 사용자 애플리케이션으로 하여금 그래픽 데이터 및 커맨드들을, GPU에 의존적인 포맷이 아닌 표준화된 포맷으로 출력할 수 있게 한다. Redmond, Washington의 Microsoft Corporation에 의해 개발된 DirectX®, Khronos Group에 의해 발표된 OpenGL®을 포함하여 몇가지 타입의 API들이 상용화되어 있다. API는 드라이버와 통신한다. 드라이버는 API로부터 수신된 표준 코드를 GPU에 의해 이해되는 네이티브 포맷의 명령들로 변환한다. 드라이버는 일반적으로 GPU의 제조사에 의해 작성된다. GPU는 드라이버로부터 명령들을 실행한다.

[0004] GPU에 의해 수행되는 그래픽-처리 태스크들은 일반적으로 매트릭스 및 벡터 동작들과 같은 복합적인 수학적 계산들을 수반한다. 단일의 그래픽-처리 태스크를 수행하기 위하여, GPU는 복수의 서로 다른 쓰레드들(명령들의 시퀀스)을 실행할 수 있다. 각각의 쓰레드는 기하 셰이더, 픽셀 셰이더, 버텍스 셰이더 등과 같은 셰이더 프로그램을 포함할 수 있다. 각각의 쓰레드(예를 들어, 셰이더 프로그램)는 일반적으로 GPU의 데이터 저장 유닛들에 국부적으로 저장되는 상태 데이터의 세트(예를 들어, 텍스쳐 핸들들(texture handles), 셰이더 상수들(shader constants), 변환 행렬들 등)와 관련된다. 국부적으로 저장된 상태 데이터는 콘텍스트(context)라 칭해진다.

[0005] 단일 그래픽-처리 태스크의 다양한 쓰레드들(예를 들어, 셰이더 프로그램들)을 효율적으로 실행하기 위하여, GPU는 처리 소자들의 어레이(셰이더 코어라 칭해짐)를 포함한다. 처리 소자들의 어레이는 단일-명령, 복수-데이터(SIMD) 디바이스들로 조직화(organize)된다. 셰이더 코어의 서로 다른 처리 소자들에 병렬적으로 분포되어있

는 각각의 쓰레드(예를 들어, 셰이더 프로그램)를 실행하기 위해 필요한 데이터와 함께, 복수의 쓰레드들(예를 들어, 셰이더 프로그램들)이 동시에 셰이더 코어에 발행(issue)될 수 있다. 서로 다른 처리 소자들은 데이터에 대한 연산들을 병렬적으로(in parallel) 수행할 수 있다. 이러한 식으로, GPU는 그래픽-처리 태스크에 요구되는 복잡한 수학적 계산들을 일반적인 중앙 처리 유닛(CPU)보다 빠르게 수행할 수 있다. 결과적으로, 컴퓨팅 시스템이 GPU를 포함하면, 그래픽-처리 태스크들(및 다른 타입의 데이터-병렬 처리 태스크들)은 일반적으로 CPU가 아닌 GPU로 전달(pass)된다.

[0006] 태스크들을 GPU로 전달하기 위하여, 운영-시스템(OS) 스케쥴러가 태스크들을 커맨드 버퍼에 저장한다. 종래의 GPU는 한번에 한 커맨드 버퍼를 처리한다. OS 스케쥴러가 커맨드 버퍼에 연속적으로(serially) 태스크들을 배치하고, 그리고 GPU는 일반적으로 태스크들이 커맨드 버퍼에 배치된 순서로 상기 태스크들을 처리한다. 그러나, 일부 경우에, GPU는 태스크들이 커맨드 버퍼에 배치된 순서에서 벗어나 태스크들을 처리한다. 예를 들어, GPU는, 제1 태스크 후 커맨드 버퍼에 배치된 더 중요한(예를 들어, 저-레이턴시(low-latency) 태스크를 실행하기 위하여 제1 태스크의 실행을 중단(interrupt)할 수 있다.

[0007] 종래의 GPU는 제1 태스크가 GPU의 셰이더 코어에서 완전히 완료되기 전에 더 중요한(예를 들어, 저-레이턴시(low-latency) 태스크)를 수행하기 위하여 콘텍스트 스위치(context switch)를 사용한다. 즉, 제1 태스크의 쓰레드들과 관련된 상태 데이터가 종래의 GPU에 의해 유지되는 백업 저장 유닛들로 스왑(swap)되고, 더 중요한(예를 들어, 저-레이턴시) 태스크의 쓰레드들(예를 들어, 셰이더 프로그램들)이 검색되어 셰이더 코어의 데이터 저장 유닛들에 배치된다. 셰이더 코어는 데이터-저장 유닛들의 새로운 상태 데이터에 근거하여 더-중요한(예를 들어, 저-레이턴시) 태스크의 쓰레드들(예를 들어, 셰이더 프로그램들)을 실행한다. 더-중요한(예를 들어, 저-레이턴시) 태스크가 실행을 종료한 후, 더-중요한(예를 들어, 저-레이턴시) 태스크의 쓰레드들과 관련된 상태 데이터가 데이터-저장 유닛들로부터 플러시(flush)되어, 제1 태스크의 쓰레드들로부터의 상태 데이터가 상기 셰이더-코어의 데이터-저장 유닛들로 다시 스왑된다. 그후, 셰이더 코어는 제1 태스크의 쓰레드들의 실행을 재개할 수 있다.

[0008] 콘텍스트 스위치는 GPU로 하여금 태스크들이 커맨드 버퍼에 배치된 순서에서 벗어나 태스크들을 처리할 수 있게 하지만, 콘텍스트 스위치는 몇가지 이유에서 문제가 있다. 첫번째로, 콘텍스트 스위치를 수행하기 위해서는 상당한 시간이 요구되며, 그럼으로써 GPU의 성능을 제약한다. 또한, 콘텍스트 스위치는 스위치되고 있는 콘텍스트를 저장하기 위한 추가의 로컬 메모리(예를 들어, 백업 저장 유닛들)을 필요로한다. 추가의 로컬 메모리는 고가의 칩 영역을 차지하며, 결과적으로 더 큰 GPU가 되게 한다.

[0009] 현저한 시간 및 영역을 요구하는 것에 부가하여, 콘텍스트 스위치는 저-레이턴시, 고-우선권(high-priority) 태스크들을 처리하는데 있어서 GPU를 비효율적이게 한다. 저-레이턴시, 고-우선권 태스크를 실행하기 위한 셰이더 코어를 준비하기 위하여, 종래의 GPU는 콘텍스트 스위치를 수행하여야만 한다. 콘텍스트 스위치와 관련된 시간은, 저-레이턴시, 고-우선권 태스크의 실행을 위한 실제 시간이 상대적으로 짧을(예를 들어, 수십 클럭 사이클) 수 있다하더라도, 저-레이턴시, 고-우선권 태스크를 실행하기 위한 유효 시간(effective time)을 상대적으로 길게 만든다.

[0010] 상기를 고려하여, 콘텍스트 스위치 없이 중요한(예를 들어, 저-레이턴시) 태스크들을 효율적으로 처리할 수 있는 처리 유닛이 요구된다.

발명의 내용

과제의 해결 수단

[0011] 본 발명의 실시예들은 비동기 태스크 디스패치를 가능하게 하는 방법, 장치, 시스템 및 그것의 응용을 제공함으로써, 위에서 기술된 요구들을 충족시킨다.

[0012] 예를 들어, 본 발명의 일 실시예는 복수의 가상 엔진들 및 셰이더 코어를 포함하는 처리 유닛을 제공한다. 복수의 가상 엔진들은 (i) 운영 시스템(OS)으로부터, 복수의 테스크들을 서로 실질적으로 병렬적으로(in parallel) 수신하고, (ii) 복수의 태스크들 각각과 관련된 상태 데이터의 세트를 로딩하도록 구성된다. 셰이더 코어는 복수의 태스크들 각각과 관련된 상태 데이터의 세트에 근거하여 복수의 태스크들을 실질적으로 병렬적으로 실행하도록 구성된다. 처리 유닛은 또한 셰이더 코어에 발행될 복수의 태스크들을 스케쥴링하는 스케쥴링 모듈을 포함할 수 있다.

[0013] 또 다른 실시예에서, 처리 유닛은 소프트웨어 내에서 정의된다. 이 실시예에서, 컴퓨터 프로그램 물(computer

program product)은, 컴퓨터 디바이스상에서 실행될 때 처리 유닛을 정의하는 명령들을 포함하고 있는 컴퓨터-판독가능 저장 매체를 포함한다.

[0014] 또 다른 실시예에서, 처리 유닛은 컴퓨팅 시스템에 포함된다. 이 실시예에서, 컴퓨팅 시스템은 메모리, 제1 처리 유닛, 제2 처리 유닛, 그리고 상기 메모리, 상기 제1 처리 유닛, 상기 처리 유닛에 연결된 버스를 포함한다. 예시적인 컴퓨팅 시스템은, 슈퍼컴퓨터, 데스크탑 컴퓨터, 랩탑 컴퓨터, 비디오 게임 콘솔, 임베디드 디바이스, 휴대형 디바이스(예를 들어, 모바일 폰, 스마트폰, MP3 플레이어, 카메라, GPS 디바이스 등) 또는 처리 유닛을 포함하거나 처리 유닛을 포함하도록 구성된 어떤 다른 디바이스를 포함하나, 이것들로 제한되는 것은 아니다.

[0015] 본 발명의 또 다른 실시예는 처리 유닛의 태스크들을 처리하기 위한 컴퓨터-구현 방법을 제공한다. 이 컴퓨터-구현 방법은 몇가지 동작들을 포함한다. 제1 동작에서, 운영 시스템(OS)으로부터, 복수의 태스크들이 서로 대해 병렬적으로 수신된다. 제2 동작에서, 복수의 태스크들 각각과 관련된 상태 데이터의 세트가 로딩된다. 제3 동작에서, 복수의 태스크들이, 상기 복수의 태스크들 각각과 관련된 상태 데이터의 세트에 근거하여 셰이더 코어에서 실질적으로 병렬적으로 실행된다. 이 컴퓨터-구현 방법은 또한 셰이더 코어에 발행될 복수의 태스크들을 스케쥴링하는 것을 포함한다.

[0016] 본 발명의 또 다른 실시예들은 태스크들을 처리 유닛에 제공하기 위한 컴퓨터-구현 방법을 제공한다. 이 방법은 몇개의 동작들을 포함한다. 제1 동작에서, 복수의 태스크들이 하나 이상의 애플리케이션들로부터 수신되며, 여기서 각각의 태스크는 우선권 타입(priority type)에 대한 표시를 포함한다. 제2 동작에서, 처리 유닛에는 복수의 태스크들 및 각각의 태스크와 관련된 우선권 타입의 표시가 제공된다. 실시예에서, 컴퓨터 프로그램 물의 컴퓨터-판독가능 저장 매체에 저장된 명령들은, 상기 명령들이 컴퓨팅 디바이스에 의해 실행되는 경우, 컴퓨팅 디바이스로 하여금 이 방법을 수행하게 할 수 있다.

[0017] 본 발명의 추가의 피쳐들 및 이점들은, 본 발명의 다양한 실시예들의 동작 및 구조와 함께, 첨부의 도면들을 참조로 하기에서 자세히 기술된다. 본 발명은 여기에 기술된 특정 실시예들로 제한된 것이 아니다. 그러한 실시예들은 여기에서 단지 예시적 목적으로 제시된다. 관련 기술 분야의 기술자들에게는, 본 명세서에 포함된 내용에 근거한 추가의 실시예들이 자명할 것이다.

도면의 간단한 설명

[0018] 여기에 포함되어있으며 본 명세서의 일부를 이루는 첨부의 도면들은 본 발명을 예시하고, 그리고 상세한 설명과 함께 본 발명의 원리를 설명하고 관련 기술 분야의 기술자들이 본 발명을 다루고 사용할 수 있게 하도록 기능한다.

도 1은 본 발명의 실시예에 따른 예시적인 컴퓨터 시스템을 도시하는 블록도이다.

도 2는 본 발명의 실시예에 따른 예시적인 GPU의 블록도이다.

도 3a 및 3b는 본 발명의 실시예에 따라 GPU의 가상 엔진들에 태스크들을 발행하기 위한 예시적인 작업 흐름을 도시한다.

도 4는 본 발명의 실시예에 따라 GPU의 가상 엔진들에 태스크들을 발행하기 위한 보다 상세한 예시적인 작업 흐름을 도시한다.

도 5는 본 발명의 실시예가 구현되는 예시적인 컴퓨터 시스템의 블록도를 도시한다.

본 발명의 피쳐들 및 이점들은 도면과 함께 고려될 때 하기에 설명된 상세한 설명으로부터 보다 명확해질 것이다. 도면에서 유사한 참조문자는 도면들 전체에 걸쳐 대응하는 구성요소들을 식별한다. 도면에서, 유사한 참조숫자들은 일반적으로, 동일한, 기능적으로 유사한, 그리고/또는 구조적으로 유사한 구성요소들을 표시한다. 구성요소가 처음으로 나타나는 도면은 대응하는 참조숫자의 최좌측의 번호(들)로 표시된다.

발명을 실시하기 위한 구체적인 내용

[0019] I. 개괄(Overview)

[0020] 본 발명의 이 실시예들은 비동기 태스크 디스패치를 가능하게 해주는 처리 유닛과 그 응용들을 제공한다. 하기의 상세한 설명에서, "일 실시예", "실시예", "예시적이 실시예" 등에 대한 참조는 기술된 실시예가 특정한 피쳐, 구조, 또는 특징을 포함할 수 있음을 나타내나, 모든 실시예가 반드시 그 특정한 피쳐, 구조, 또는 특징을 포함할 필요는 없다. 또한, 그러한 표현이 반드시 동일한 실시예를 참조하는 것일 필요는 없다. 또한, 특정한

피쳐, 구조, 또는 특징이 임의의 실시예와 관련하여 기술될 때, 명시적으로 설명되든 아니든 다른 실시예들과 관련되어 그러한 피쳐, 구조, 또는 특징에 영향을 주는 것은 본 기술분야의 기술자의 지식의 범위내에 있는 것으로 고려된다.

[0021] 실시예에 따라, 처리 유닛은 단일의 셰이더 코어에 구현된 복수의 가상 엔진들을 포함한다. 각각의 가상 엔진은 데이터-병렬 처리 태스크들(예를 들어, 그래픽-처리 태스크들 및 일반-컴퓨팅 태스크들)을 수신하고 단일 셰이더 코어 상에서 독립적으로 이 태스크들을 실행한다. 이러한 식으로, 처리 유닛은 콘텍스트 스위치를 요구함이 없이, 저-레이이턴시 처리 태스크들의 제1 스트림 및 표준 그래픽 처리 태스크들의 제2 스트림과 같은 두개 이상의 서로 다른 처리 태스크 스트림들을 실행할 수 있다. 두개 이상의 서로 다른 처리 태스크 스트림들을 실행하는 것은 정지 및 처리 유닛에서 데이터를 드레이닝(draining)하는 것과 관련된 오버헤드 없이 콘텍스트 스위치의 저-레이턴시 이점들을 제공한다. 사실, 본 발명의 실시예들은 복수의 콘텍스트들이 단일 셰이더 코어에 존재할 수 있게 하며 (실질적으로) 동시에 실행될 수 있게 해준다.

[0022] 제한이 아닌 단지 예시의 목적으로, 본 발명의 실시예들은 여기에서 GPU에 관하여 기술될 것이다. 그러나, 관련 기술분야의 통상의 기술자는, 본 발명의 실시예들이 처리 태스크들의 스트림들을 수신하는 다른 타입의 처리 유닛들(예를 들어, 중앙 처리 유닛들 및 코프로세서들)에 적용될 수 있음을 이해할 것이다. 이러한 다른 타입의 프로세서들은 본 발명의 정신 및 범주 내에 있는 것으로 고려된다.

[0023] 실시예들에서, GPU는 복수의 커맨드 버퍼들을 처리한다. 저-레이턴시 처리 태스크들은, 예를 들어, 제1 커맨드 버퍼에 배치되고, 표준 그래픽-처리 태스크들은, 예를 들어, 제2 커맨드 버퍼에 배치될 수 있다. GPU의 제1 가상 엔진은 저-레이턴시 처리 태스크들을 검색하고, GPU의 제2 가상 엔진은 표준 그래픽-처리 태스크들을 검색한다. 각각의 가상 엔진으로부터의 태스크들은 그후 서로에 대해 실질적으로 병렬적으로 단일의 셰이더 코어에 발행된다.

[0024] 단일 셰이더 코어로 하여금 두개 이상의 서로 다른 가상 엔진들로부터의 태스크들을 (실질적으로) 동시에 처리할 수 있게 하기 위하여, 셰이더 코어의 자원들이 공간 및/또는 시간으로 파티셔닝된다. 공간 파티셔닝을 달성하기 위하여, 예를 들어, 제1 가상 엔진으로부터의 제1 (예를 들어, 저-레이턴시) 태스크가 셰이더 코어의 처리 소자들(SIMD들)의 제1 서브세트로 발행되고, 제2 가상 엔진으로부터의 제2 (예를 들어, 표준 그래픽) 태스크가 셰이더 코어의 처리 소자들(SIMD들)의 제2 서브세트로 발행된다. 시간적 파티셔닝(temporal partitioning)을 달성하기 위하여, 예를 들어, 제1 및 제2 태스크들은 셰이더 코어의 처리 소자들(SIMD들)의 시간의 임의의 퍼센티지를 공유한다. 일 실시예에서, GPU는 셰이더 코어 상에서의 실행을 위하여 두개 이상의 서로 다른 가상 엔진들로부터의 태스크들을 스케쥴링하는 스케쥴링 모듈을 포함한다.

[0025] 본 발명의 실시예에 따라 복수의 가상 엔진들을 제공하기 위하여 GPU의 자원들을 공유하는 것은, 특히 큰 칩들에서 GPU 자원들의 사용을 개선한다. 두개 이상의 태스크 스트림들이 단일의 셰이더 코어에 발행될 수 있고, GPU로 하여금 컴퓨팅 및 입/출력 설비들을 효율적으로 사용할 수 있게 한다. 예를 들어, GPU 셰이더 코어의 자원들(예를 들어, SIMD들)이 수요, 우선권, 및/또는 프리셋 제한에 근거하여 동시 태스크들(concurrent tasks) 간에 나누어질 수 있고, 한편 임시적으로 임의의 일 태스크로 하여금 GPU의 자원들을 (실질적으로) 완전히 소비할 수 있게 할 수 있다.

[0026] 본 발명의 실시예에 따른 예시적인 GPU의 추가의 세부사항들이 하기에 기술된다. 그러나, 이 세부사항들을 제공하기에 앞서서, 그러한 GPU가 구현될 수 있는 예시적인 시스템을 기술하는 것이 도움이될 것이다.

[0027] II. 예시적인 시스템

[0028] 도 1은 실시예에 따른 컴퓨팅 시스템(100)의 블록도이다. 컴퓨팅 시스템(100)은 CPU(102), GPU(110)를 포함하며, 선택적으로 코프로세서(112)를 포함할 수 있다. 도 1의 실시예에서, CPU(102) 및 GPU(110)는 별개의 블록으로 보여진다. 이는 제한을 위한 것이 아닌 단지 설명을 위한 것이다. 관련 기술 분야의 기술자는 CPU(102)와 GPU(110)가 별개의 패키지들에 포함되거나 단일 패키지 또는 집적 회로 내에서 결합될 수 있음을 이해할 것이다.

[0029] 컴퓨팅 시스템(100)은 또한 CPU(102), GPU(110), 및 코프로세서(112)에 의해 액세스될 수 있는 시스템 메모리(104)를 포함한다. 실시예들에서, 컴퓨팅 시스템(100)은 슈퍼컴퓨터, 데스크탑 컴퓨터, 랩탑 컴퓨터, 비디오-게임 콘솔, 임베디드 디바이스, 휴대형 디바이스(예를 들어, 모바일 텔레폰, 스마트 폰, MP3 플레이어, 카메라,

GPS 디바이스 등), 또는 GPU를 포함하거나 GPU를 포함하도록 구성된 어떤 다른 디바이스를 포함할 수 있다.

[0030] GPU(110)는 특정한 특별 기능들(예를 들어, 그래픽-처리 태스크들 및 데이터-병렬, 일반-컴퓨팅 태스크들)을 CPU가 소프트웨어에서 그것들을 수행할 수 있는 것보다 일반적으로 더 빠르게 수행함으로써 CPU(102)를 보조한다. GPU(110)는 단일 셰이더 코어의 자원들을 공유하는 복수의 가상 엔진들을 포함한다. 이러한 식으로, GPU(110)의 복수의 가상 엔진들은 복수의 태스크들을 실질적으로 병렬적으로 실행할 수 있다. 실시예들에서, GPU(110)는 칩셋 및/또는 CPU(102)에 통합될 수 있다. GPU(110)의 추가의 세부사항들은 하기에 제공된다.

[0031] 코프로세서(112)는 또한 CPU(102)를 보조한다. 코프로세서(112)는 관련 기술분야의 기술자들에게 자명한 바와 같이 부동 소수점 코프로세서, GPU, 네트워킹 코프로세서, 및 다른 타입의 코프로세서들과 프로세서들을 포함할 수 있으나, 이것들에 국한되는 것은 아니다.

[0032] GPU(110) 및 코프로세서(112)는 버스(114)를 통해 CPU(102) 및 시스템 메모리와 통신한다. 버스(114)는 주변장치 인터페이스(PCI) 버스, 가속 그래픽 포트(AGP) 버스, PCI 익스프레스(PCIE) 버스 또는 현재 사용가능하거나 미래에 개발될 또 다른 타입의 버스를 포함하는, 컴퓨터 시스템에서 사용되는 임의의 타입의 버스일 수 있다.

[0033] 시스템 메모리(104)에 부가하여, 컴퓨팅 시스템(100)은 로컬 메모리(106) 및 로컬 메모리(108)를 더 포함한다. 로컬 메모리(106)는 GPU(110)에 연결되고 또한 버스(114)에 연결될 수도 있다. 로컬 메모리(108)는 코프로세서(112)에 연결되고 또한 버스(114)에 연결될 수 있다. 특정 데이터(예를 들어, 빈번하게 사용되는 데이터)에 대해 상기 데이터가 시스템 메모리(104)에 저장되었을 경우에 가능했을 액세스 속도보다 더 빠른 액세스를 제공하기 위하여, 로컬 메모리들(106, 108)이 GPU(110) 및 코프로세서(112)에서 각각 사용가능하다.

[0034] 실시예에서, GPU(110) 및 코프로세서(112)는 CPU(102)와 병렬적으로 명령들을 디코딩하여 그것들을 위한 명령들만을 실행한다. 또 다른 실시예에서, CPU(102)는 GPU(110) 및 코프로세서(112)를 위한 명령들을 각각의 커맨드 버퍼들로 송신한다.

[0035] 도 1에 구체적으로 도시되지는 않으나, 컴퓨팅 시스템(100)은 또한 디스플레이 디바이스(예를 들어, 캐쏘드-레이 튜브, 액정 디스플레이, 플라즈마 디스플레이 등)를 포함하거나 디스플레이 디바이스에 연결될 수 있다. 디스플레이 디바이스는 (예컨대, 컴퓨팅 시스템(100)이 컴퓨터, 비디오-게임 콘솔, 또는 휴대혀 디바이스를 포함할 때) 사용자에게 콘텐츠를 디스플레이하기 위하여 사용된다.

[0036] III. 예시적인 GPU

[0037] 위에서 언급된 바와 같이, GPU(110)는 셰이더 코어 상에 구현되는 복수의 가상 엔진들을 포함한다. 각각의 가상 엔진은 OS 스케쥴러에 의해 제공되는 처리 태스크들의 스트림을 실행하도록 구성되며, 여기서 주어진 스트림의 각각의 처리 태스크는 복수의 개별 처리 쓰레드들을 포함할 수 있다. GPU(110)가 복수의 가상 엔진들을 포함하므로, GPU(110)는 콘텍스트 스위치를 요구함이 없이 OS 스케쥴러로부터 서로 다른 처리 태스크들의 스트림들을 실행할 수 있다. 사실, 실시예들에서, GPU(110)는, 태스크들 사이에서 셰이더 코어의 자원들을 공유함으로써, 서로 다른 복수의 콘텍스트들에 대응하는 복수의 스트림들로부터의 태스크들을 단일 셰이더 코어에서 (실질적으로) 동시에 실행한다.

[0038] 도 2는 GPU(110)의 예시적인 하드웨어 컴포넌트들을 도시하는 블록도이다. 도 2를 참조하여, GPU(110)는 커맨드 프로세서(230), 입력 로직(버텍스 분석기(208), 스캔 변환기(212), 산술 로직(222)을 포함함), 셰이더 코어(214), 출력 로직(224), 및 메모리 시스템(210)을 포함한다. 이 컴포넌트들 각각은 하기에서 기술된다.

[0039] A. 커맨드 프로세서

[0040] 커맨드 프로세서(230)는 OS 스케쥴러에 의해 채워진 하나 이상의 커맨드 버퍼들로부터 태스크들(예를 들어, 그래픽-처리 및 일반-컴퓨팅 태스크들)을 수신한다. 도 3에 도시된 바와 같이, 커맨드 프로세서(230)는 GPU(110)의 자원들을 공유하는 복수의 가상 엔진들을 포함한다. 커맨드 프로세서(230)의 서로 다른 가상 엔진들은 서로 다른 타입의 태스크들을 처리한다.

[0041] 도 2의 실시예에서, 커맨드 프로세서(230)는 제1 백그라운드 엔진(202A), 제2 백그라운드 엔진(202B), 실시간 저-레이턴시 엔진(202C), 주요(primary) 3D 엔진(202D), 및 저-레이턴시 3D 엔진(202E)을 포함한다. 백그라운드 엔진들(202)은 저-우선권 태스크들을 처리한다. 그러나, 커맨드 프로세서(230)가 다른 타입의 가상 엔진들을 포함할 수 있음이 이해되어야 한다. 백그라운드 엔진들(202)은 다른 가상 엔진들이 GPU(110)의 자원들을 사용하고 있지 않을 때에만 GPU(110)의 자원들을 넘겨받는다(take over). 실시간 저-레이턴시 엔진(202C)은 고-우선권 태스크들을 처리하기 위하여, GPU(110)의 자원들에 대한 우선권 액세스를 가진다. 주요 3D 엔진(202D)은 표준

그래픽-처리 태스크들을 처리하고, 저-레이턴시 3D 엔진(202E)은 고-우선권 그래픽-처리 태스크들을 처리한다. 저-레이턴시 3D 엔진(202E)은 GPU(110)의 그래픽-처리 자원들에 대한 우선권 액세스를 가진다.

[0042] GPU(110)의 셰이더 코어(214)에 발행되기 전에, 커맨드 프로세서(230)로부터의 태스크들이 입력 로직에 제공된다.

[0043] B. 입력 로직

[0044] 입력 로직은 어떤 태스크들이 셰이더 코어(214)에 발행되는지를 중재(arbitration)한다. 실시예에서, 입력 로직은 셰이더 코어(214)의 자원들의 사용가능성 및 다양한 태스크들의 상대적 우선권에 근거하여 셰이더 코어(214)에서 실행하기 위한 태스크들을 스케쥴링하기 위한 소프트웨어 루틴을 구현한다. 도 3의 실시예에서, 입력 로직은 (셰이더 코어(214)에 발행하기 위한 그래픽-처리 태스크들을 준비하는) 그래픽 사전-처리(pre-processing) 로직 및 (셰이더 코어(214)에 태스크들을 제공하는) 중재 로직(222)을 포함한다.

[0045] 그래픽 사전-처리 로직은 버텍스 분석기(208) 및 스캔 변환기(212)를 포함한다. 주요 3D 엔진(202D) 및 저-레이턴시 3D 엔진(202E)으로부터의 태스크들이 그래픽 사전-처리 로직으로 송신된다. 선입 선출(FIFO) 버퍼(204A)는 주요 3D 엔진(202D)으로부터 태스크들을 수신하고, FIFO 버퍼(204B)는 저-레이턴시 3D 엔진(202E)으로부터 그 태스크들을 수신한다. 멀티플렉서(206)는 FIFO 버퍼들(204) 중 하나로부터 버텍스 분석기(208)로 태스크들을 제공한다.

[0046] 버텍스 분석기(208)는 그래픽-처리 및/또는 일반-컴퓨팅 태스크와 관련된 셰이더 프로그램들을 식별하고, 사용가능할 입력 및 출력 데이터에 근거하여 각각의 셰이더 프로그램이 셰이더 코어(214)에 론치될 수 있는 때를 스케쥴링한다. 론치될 셰이더 프로그램을 스케쥴링하는 것에 부가하여, 버텍스 분석기(208)는 또한 버텍스 버퍼에 대한 포인터들을 발생시키고 연결성 데이터를 포함한다. 포인터들은 버텍스 버퍼로부터 버텍스들을 판독하기 위하여 사용된다. 버텍스가 이미 처리되었고 버텍스 버퍼에 저장되어 있다면, 버텍스 분석기(310)는 버텍스 버퍼로부터 버텍스를 판독할 수 있으며, 따라서 버택스는 단지 한번 처리된다. 연결성 데이터는 프리미티브(예를 들어, 삼각형)를 만들기 위하여 어떻게 버텍스들이 함께 피팅되는지를 규정하며, 따라서 프리미티브가 적절히 래스터화(rasterize)될 수 있다.

[0047] 버텍스 분석기(208)는 그래픽-처리 태스크들을 스캔 변환기(212)로 송신하고 일반-컴퓨팅 태스크들을 중재 로직(222)으로 송신한다. 스캔 변환기(212)는 셰이더 코어(214)에 의해 처리될 픽셀들을 결정하기 위하여 프리미티브들을 횡단(traverse)한다. 스캔 변환기(212)는 픽셀들을 중재 로직(222)으로 송신한다. 중재 로직(222)은 커맨드 프로세서(230)의 서로 다른 가상 엔진들로부터의 태스크들을 셰이더 코어(214)에 제공하기 위하여 복수의 멀티플렉서들을 포함한다.

[0048] C. 셰이더 코어

[0049] 셰이더 코어(214)는 GPU(110)에 제공되는 태스크들을 실행하기 위한 복수의 처리 소자들(220)을 포함한다. 처리 소자들(220)은 SIMD 디바이스들로서 구성되고, 셰이더 코어(214)로 하여금 복수의 데이터-병렬 처리 태스크들을 (실질적으로) 동시에 실행할 수 있게 한다. 셰이더 코어(214)로 하여금 커맨드 프로세서(230)의 복수의 가상 엔진들로부터 (실질적으로) 동시에 태스크들을 처리할 수 있게 하기 위하여, 셰이더 코어(214)의 처리 소자들(220)은 공간 및/또는 시간으로 파티셔닝된다.

[0050] 공간 파티셔닝을 달성하기 위하여, 처리 소자들(220)의 서로 다른 서브세트들이 서로 다른 태스크들을 실행하도록 구성된다. 예를 들어, 제1 가상 엔진(예를 들어, 실시간 저-레이턴시 엔진(202C))으로부터의 제1(예를 들어, 저-레이턴시) 태스크는 셰이더 코어(214)의 처리 소자들(220)의 제1 서브세트에 발행될 수 있고, 제2 가상 엔진(예를 들어, 주요 3D 엔진(202D))으로부터의 제2(예를 들어, 표준 그래픽) 태스크는 셰이더 코어(214)의 처리 소자들(220)의 제2 서브세트에 발행될 수 있다. 처리 소자들(220)의 각각의 서브세트는 그것이 수신한 태스크를 독립적으로 실행한다.

[0051] 시간적인 파티셔닝을 달성하기 위하여, 서로 다른 가상 엔진들의 서로 다른 처리 태스크들이 셰이더 코어(214)의 처리 엔진들(220)의 임의 퍼센트의 시간을 공유한다. 위의 예로부터, 제1 및 제2 태스크들은 셰이더 코어(214)의 처리 소자들(220)의 임의 퍼센트의 시간을 공유한다.

[0052] 셰이더 코어(214)는 또한 OS 스케쥴러에 의해 제공되는 처리 태스크들을 실행하기 위하여 처리 소자들(220)에 의해 사용되는 데이터를 저장하기 위한 하나 이상의 로컬 데이터 공유들(LDS)(228)을 포함한다. LDS(228)는 셰이더 코어(214)에 의해 실행될 각각의 태스크와 관련된 상태 데이터를 저장한다. 실시예에서, LDS(228)는, 복수

의 서로 다른 콘텍스트들의 상태 데이터를 저장하고, 셰이더 코어(214)로 하여금, 콘텍스트 스위치를 요구함이 없이 복수의 서로 다른 콘텍스트들과 관련된 OS 스케쥴러로부터의 서로 다른 복수의 태스크들을 (실질적으로) 동시에 실행할 수 있게 한다.

[0053] 처리 소자들(220)의 중간 결과들이 셰이더 코어(214)에서 재처리될 수 있다. 예를 들어, 처리 소자들(220)은 OS 스케쥴러에 의해 제공되는 단일 그래픽-처리 태스크를 완료하기 위하여 복수의 서로 다른 셰이더 프로그램들(예를 들어, 기하 셰이더, 버텍스 셰이더, 픽셀 셰이더, 테셀레이션 셰이더, 등)을 실시할 수 있다. 서로 다른 셰이더 프로그램들의 중간 결과들이 버텍스 분석기(208) 및/또는 스캔 변환기(212)에 다시 송신되고, 종국적으로는 처리 소자들(220)로 재순환(recirculation)된다. 처리 소자들(220)이 OS 스케쥴러에 의해 제공된 태스크를 완료한후, 최종 결과들이 출력 로직(224)에 제공된다.

[0054] D. 출력 로직

[0055] 출력 로직(224)은 기록-결합 캐시들, 깊이 버퍼들, 및 컬러 버퍼들을 포함하는 복수의 버퍼들을 포함한다. 기록-결합 캐시들은 오프-칩 메모리에 기록될 데이터를 결합하고, 오프-칩 메모리에 대한 효율적인 액세스를 가능하게 한다. 깊이 버퍼들은 z-테스팅을 위한 결과들을 버퍼링한다. 컬러 버퍼들은 컬러 블렌딩을 위한 결과들을 버퍼링한다. 기록-결합 캐시들, 깊이 버퍼들, 및 컬러 버퍼들과 관련된 프로세스들을 수행한 후, 출력 로직(224)은 메모리 시스템(210)에 결과들을 제공한다.

[0056] E. 메모리 시스템

[0057] 메모리 시스템(210)은 하나 이상의 온-칩 캐시들 및 하나 이상의 오프-칩 메모리 인터페이스들을 포함한다. 메모리 시스템(210)은 커맨드 프로세서(230), 버텍스 분석기(208), 스캔 변환기(212), 셰이더 코어(214), 및 출력 로직(224) 각각에 연결된다. 셰이더 프로그램을 실행하기 위하여 이 컴포넌트들 중 임의의 컴포넌트들에 의해 데이터가 필요할 때, 메모리 시스템(210)의 온-칩 캐시로 요청이 이루어진다. 온-칩 캐시에 히트가 있는 경우(즉, 요청된 데이터가 온-칩 캐시에 있음), 데이터는 그것을 요청한 컴포넌트로 포워딩된다. 온-칩 캐시에 미스가 있는 경우(즉, 요청된 데이터가 온-칩 캐시에 없음), 요청된 데이터는 메모리 시스템(210)의 오프-칩 메모리 인터페이스를 통해 오프-칩 메모리(예를 들어, 도 1의 시스템 메모리(104))로부터 검색되어야 한다. 오프-칩 메모리로부터 데이터가 검색된 후, 데이터는 그것을 요청한 컴포넌트로 포워딩된다. 추가로, 데이터는 또한 관련 기술분야의 기술자들에게 잘 알려져 있는 캐시 메모리 기법들을 사용하여 온-칩 캐시에 저장된다.

[0058] IV. 예시적인 동작

[0059] GPU(110)는 OS 스케쥴러에 의해 제공되는 복수의 처리 태스크 스트림들을 실행하도록 구성된다. 복수의 처리 태스크 스트림들은 단일 애플리케이션에 의해 또는 하나 이상의 애플리케이션들에 의해 발생될 수 있다.

[0060] 도 3a는 셰이더 코어(214)가 2개의(또는 2개 이상의) 서로 다른 처리 태스크 스트림들을 (실질적으로) 동시에 실행하는 예를 도시하며, 여기서 처리 태스크들의 스트림들은 단일 애플리케이션(302)에 의해 생성된다. 애플리케이션(302)은, 예를 들어, GPU상에서 실행될, 그래픽-처리 태스크들(예를 들어, 비디오-게임 애플리케이션, 컴퓨터-보조 설계(CAD) 애플리케이션 등)을 발생시키는 최종-사용자 애플리케이션이거나 일반-컴퓨팅 태스크들(예를 들어, 수학적 알고리즘, 물리 시뮬레이션, 등)을 발생시키는 최종-사용자 애플리케이션일 수 있다. 도 3a를 참조하여, 애플리케이션(302)은 제1 태스크(308A) 및 제2 태스크(308B)를 발생시킨다. 애플리케이션(302)이 발생시키는 각각의 태스크(308)는 우선권 타입을 포함한다. 예를 들어, 애플리케이션(302)은 제1 태스크(308A)가 저-레이턴시, 고-우선권 태스크이고, 제 태스크(308B)가 표준 우선권 태스크임을 표시할 수 있다. OS 스케쥴러(310)는 애플리케이션(302)에 의해 발생된 태스크들을 수신하고 GPU(110)의 서로 다른 가상 엔진들에 태스크들을 발행한다. 예를 들어, OS 스케쥴러(310)는 먼저, 태스크(308A)를 제1 가상 엔진(312A)에 발행하고, 제2 태스크(308B)를 제2 가상 엔진(312B)에 발행한다. 각각의 가상 엔진(312)으로부터의 태스크들이 그후 (실질적으로) 동시에 GPU(110)의 셰이더 코어(214)에 의해 실행된다.

[0061] 도 3b는 셰이더 코어(214)가 두개(또는 두개 이상)의 서로 다른 처리 태스크 스트림들을 (실질적으로) 동시에 실행하는 예를 도시하며, 여기서 각각의 스트림은 서로 다른 애플리케이션에 의해 발생된다. 도 3b에 도시된 바와 같이, 제1 처리 태스크(330A)가 제1 애플리케이션(302A)에 의해 발생되고, 제2 처리 태스크(330B)가 제2 애플리케이션(302B)에 의해 발생된다. 각각의 태스크(330)는 우선권 타입을 포함한다. OS 스케쥴러(310)는 태스크들(330)을 수신하고 그것들을 GPU(110)의 서로 다른 가상 엔진들에 발행한다. 예를 들어, OS 스케쥴러(310)는 제1 태스크(330A)를 제1 가상 엔진(332A)에 발행하고 제2 태스크(330B)를 제2 가상 엔진(332B)에 발행한다. 각

각의 가상 엔진(332)으로부터의 태스크들이 그후 GPU(110)의 셰이더 코어(214)에 의해 (실질적으로) 동시에 실행된다.

[0062] 도 3a 및 3b의 예에서, GPU(110)는 태스크들 및 우선권 타입들을 모두 수신한다. GPU(110)에 우선권 타입을 제공하기 위하여, 애플리케이션(302)은 각각의 태스크의 우선권 타입을 표시하는 비트들을 API에 제공한다. API는 이 정보를 GPU(110)의 드라이버에 제공한다. 실시예에서, GPU(110)는, 적어도 부분적으로 애플리케이션에 의해 규정된 우선권 타입에 근거하여 셰이더 코어(214)에서 실행될 태스크들을 스케쥴링하는 스케쥴링 모듈을 포함한다.

[0063] 도 3a 및 3b의 예들이 2개의 처리 태스크 스트림들만을 실행하는 셰이더 코어(214)를 도시하나, 이는 단지 예시적인 목적이며, 제한적인 것이 아니다. 셰이더 코어(214)가 하나 이상의 애플리케이션들에 의해 발생된 두개 이상의 처리 태스크 스트림들을 (실질적으로) 동시에 실행할 수 있다는 것이 이해되어야 한다.

[0064] 예를 들어, 도 4는 컴퓨팅 시스템(예를 들어, 컴퓨팅 시스템(100)) 상에서 구동되는 하나 이상의 애플리케이션들과 상기 컴퓨팅 시스템에 포함된 GPU(110) 사이의 다양한 소프트웨어 및 하드웨어 계층들을 보여주는 예시적인 작업흐름이다. 도 4의 실시예에서, 두개의 서로 다른 서로 다른 처리 유닛 타입들, 즉, CPU(102)와 GPU(110)가 존재한다. 또한, 두개의 애플리케이션들-제1 애플리케이션(402A) 및 제2 애플리케이션(402B)-이 컴퓨터 시스템 상에서 구동된다. CPU(102)는 애플리케이션들(402)에 의해 요구되는 주요 기능을 제공한다. 예를 들어, CPU(102)는 복수의 코어들(412A-N)을 포함할 수 있으며, 여기서 제1 애플리케이션(402A)은 주로 제1 코어(412A) 상에서 구동되고 제2 애플리케이션(402B)은 주로 제2 코어(412N) 상에서 구동된다.

[0065] 구동 중에, 애플리케이션들(402)은 CPU(102)가 아닌 GPU(110) 상에서 실행될 복수의 태스크들(404A-D)을 발생시킬 수 있다. 태스크들(404)은 데이터-병렬 처리 태스크들(예를 들어, 그래픽-처리 태스크들, 일반-컴퓨팅 태스크들, 등)을 포함하는바, GPU(110)는 CPU(102)가 소프트웨어에서 상기 태스크들을 수행하였을 경우보다 빠르게 상기 태스크들을 수행할 수 있을 것이다. 각각의 태스크(404)는 애플리케이션들(402)에 의해 규정된 것과 같은 (도 3a 및 3b에 도시된 태스크들에 포함된 우선권 타입과 같은) 우선권 타입의 표시를 포함한다. 종래의 시스템에서, OS 스케쥴러는 단일 커맨드 버퍼에 직렬 방식으로 태스크들(404)을 제공할 것이고, 종래의 GPU는 그 태스크들을 직렬로 처리할 것이다. 그러한 종래의 시스템과 달리, OS 스케쥴러(310)는 애플리케이션들(402)에 의해 규정된 우선권 타입에 근거하여 각각의 태스크(404)를 복수의 커맨드 버퍼들(420A-N) 중 하나에 제공한다. 예를 들어, OS 스케쥴러(310)는 제1 타입의 태스크(예를 들어, 고-우선권 태스크들)를 제1 커맨드 버퍼(420A)에, 제2 타입의 태스크(예를 들어, 그래픽-처리 태스크들)을 제2 커맨드 버퍼(420B)에, 등으로 제공한다.

[0066] GPU(110)는 복수의 가상 엔진들(432A-N)을 포함하며, 각각은 커맨드 버퍼들(420A-N) 중 하나를 서비스하도록 구성된다. 예를 들어, 제1 가상 엔진(432A)은 제1 커맨드 버퍼(420A)를 서비스하고, 제2 가상 엔진(432B)은 제2 커맨드 버퍼(420B)를 서비스하고, 그리고 N-번째 가상 엔진(432N)은 N-번째 커맨드 버퍼(420N)를 서비스하도록 구성된다. 위에서 설명된 바와 같이, 가상 엔진들(432)로부터의 태스크들은 셰이더 코어(214)에 의해 (실질적으로) 동시에 실행된다. 실시예에서, GPU(110)의 스케쥴링 모듈(434)은 적어도 다음의 조건들에 근거하여 셰이더 코어(214)에 의해 실행될 태스크들을 스케쥴링한다: (i) 애플리케이션들(402)에 의해 규정된 우선권 타입, (ii) 가상 엔진들(432)에 의해 처리되는 태스크들(404) 사이의 상대적 우선권, 그리고 (iii) 셰이더 코어(214) 내의 자원들의 사용가능성. 예를 들어, 스케쥴링 모듈(434)은 수요, 우선권, 및/또는 프리셋 제한에 근거하여 동시 태스크들(404) 사이에서 셰이더 코어(214)의 자원들을을 분할할 수 있으며, 한편 태스크들(404) 중 임의의 태스크로 하여금 GPU(110)의 자원을 완전히 소비하게 할 수 있다. 셰이더 코어(214) 내의 두개 이상의 서로 다른 처리 태스크 스트림들을 실행함으로써, GPU(110)는 콘텍스트 스위칭과 관련된 데이터의 저장, 스와핑, 드레이닝(daining)과 관련된 오버헤드 없이 콘텍스트 스위치의 저-레이턴시 이점들을 제공한다.

[0067] V. 예시적인 컴퓨터 구현

[0068] 본 발명의 실시예들은 하드웨어, 소프트웨어, 그것의 조합을 사용하여 구현될 수 있으며, 하나 이상의 컴퓨터 시스템들 또는 다른 처리 시스템들에 구현될 수 있다. 컴퓨터 시스템(500)의 예는 도 5에 도시된다.

[0069] 컴퓨터 시스템(500)은 프로세서(504)와 같은 하나 이상의 프로세서들을 포함한다. 프로세서(504)는 범용 프로세서(예컨대, CPU(102)) 또는 특수 목적 프로세서(예컨대, GPU(110))일 수 있다. 프로세서(504)는 통신 인프라스트럭쳐(506)(예를 들어, 통신 버스, 크로스-오버 바, 또는 네트워크)에 연결된다. 다양한 소프트웨어 실시예들이 이 예시적인 컴퓨터 시스템에 대하여 설명된다. 이 설명을 읽은 후, 다른 컴퓨터 시스템들 및/또는 구조들을

사용하여 본 발명을 실시하는 방법은 관련 기술분야의 기술자들에게 자명하게 될 것이다.

[0070] 컴퓨터 시스템(500)은 디스플레이 유닛(530) 상에서의 디스플레이를 위하여통신 인프라스트럭처(506)로부터(또는 도시되지 않은 프레임 버퍼로부터)의 그래픽, 텍스트, 및 다른 데이터를 포워딩하는 디스플레이 인터페이스(502)를 포함한다.

[0071] 컴퓨터 시스템(500)은 또한 메인 메모리(508), 바람직하게는 랜덤 액세스 메모리(RAM)를 포함하며, 또한 제2 메모리(510)도 포함할 수 있다. 제2 메모리(510)는, 예를 들어, 플로피 디스크, 자기 테입 드라이브, 광학 디스크 드라이브 등을 나타내는 하드 디스크 드라이브(512) 및/또는 착탈식 저장 드라이브(514)를 포함할 수 있다. 착탈식 저장 드라이브(514)는 잘 알려진 방식으로 착탈식 저장 유닛(518)으로부터 판독하고 그리고/또는 상기 착탈식 저장 유닛(518)에 기록한다. 착탈식 저장 유닛(518)은, 착탈식 저장 드라이브(514)에 의해 판독되고 기록되는 플로피 디스크, 자기 테입, 광학 디스크 등을 나타낸다. 이해될 바와 같이, 착탈식 저장 유닛(518)은 내부에 컴퓨터 소프트웨어 및/또는 데이터가 수록된 컴퓨터 사용가능 저장 매체를 포함한다.

[0072] 대안적인 실시예들에서, 제2 메모리(510)는 컴퓨터 프로그램들 또는 다른 명령들이 컴퓨터 시스템(500)에 로딩될 수 있게 하기 위한 다른 유사한 디바이스들을 포함할 수 있다. 그러한 디바이스들은, 예를 들어, 착탈식 저장 유닛(522) 및 인터페이스(520)를 포함할 수 있다. 그러한 예들은 (비디오 게임 디바이스에서 발견되는 것과 같은) 프로그램 카트리지 및 카트리지 인터페이스, 착탈식 메모리 칩(예를 들어, 소거가능 프로그램가능 판독 전용 메모리(EEPROM), 또는 프로그램가능 판독 전용 메모리(PROM)) 및 관련된 소켓, 그리고 착탈식 저장 유닛(522)으로부터 컴퓨터 시스템(500)으로 소프트웨어 및 데이터가 전송될 수 있게 해주는 다른 착탈식 저장 유닛들(522)과 인터페이스들(520)을 포함할 수 있다.

[0073] 컴퓨터 시스템(500)은 또한 통신 인터페이스(524)를 포함할 수 있다. 통신 인터페이스(524)는 소프트웨어 및 데이터가 컴퓨터 시스템(500)과 외부 디바이스들 사이에서 전송될 수 있게 해준다. 통신 인터페이스(524)의 예는 모뎀, 네트워크 인터페이스(예컨대, 이더넷 카드), 통신 포트, PCMCIA(Personal Computer Memory Card International Association) 슬롯 및 카드 등을 포함할 수 있다. 통신 인터페이스(524)를 통해 전송되는 소프트웨어 및 데이터는 전자, 전자기, 광학 신호들, 또는 통신 인터페이스(524)에 의해 수신될 수 있는 다른 신호들일 수 있는 신호들(528)의 형태로 전송된다. 이 신호들(528)은 통신 경로(예를 들어, 채널)(526)를 통해 통신 인터페이스(524)에 제공된다. 이 채널(526)은 신호들(528)을 운반하며, 와이어 또는 케이블, 광섬유, 전화선, 셀룰러 링크, 라디오 주파수(RF) 링크 및 다른 통신 채널들을 사용하여 구현될 수 있다.

[0074] 본 명세서에서, 용어 "컴퓨터-판독가능 저장 매체"는 일반적으로 착탈식 저장 드라이브(514), 및 하드 디스크 드라이브(512)에 인스톨된 하드 디스크와 같은 매체를 지칭하는데 사용된다. 이 컴퓨터 프로그램 물은 컴퓨터 시스템(500)에 소프트웨어를 제공한다.

[0075] 컴퓨터 프로그램(또한 컴퓨터 제어 로직이라고도 지칭됨)은 주 메모리(508) 및/또는 제2 메모리(510)에 저장된다. 컴퓨터 프로그램은 또한 통신 인터페이스(524)를 통해 수신될 수 있다. 여기에서 논의되는 바와 같이, 그러한 컴퓨터 프로그램들은, 실행될 때, 컴퓨터 시스템(500)으로 하여금 본 발명의 피쳐들을 수행할 수 있게 한다. 특히, 컴퓨터 프로그램들은, 실행될 때, 프로세서(504)로 하여금 본 발명의 피쳐들을 수행할 수 있게 한다. 따라서, 그러한 컴퓨터 프로그램들은 컴퓨터 시스템(500)의 제어기들을 나타낸다.

[0076] 실시예에서, 소프트웨어는 컴퓨터 프로그램 물에 저장될 수 있고 착탈식 저장 드라이브(514), 하드 드라이브(512) 또는 통신 인터페이스(524)를 사용하여 컴퓨터 시스템(500)에 로딩될 수 있다. 제어 로직(소프트웨어)은, 프로세서(504)에의해 실행될 때, 프로세서(504)로 하여금 여기에 기술된 것과 같은 본 발명의 실시예의 기능들을 수행하게 한다.

[0077] VI. 예시적인 소프트웨어 구현(Example Software Implementations)

[0078] 처리 유닛들(예를 들어, CPU(102) 및/또는 GPU(110))의 하드웨어 구현예들에 부가하여, 그러한 처리 유닛들은 또한, 예를 들어 소프트웨어(예컨대, 컴퓨터-판독가능 프로그램 코드)를 저장하도록 된 컴퓨터-판독가능 매체에 배치될 수 있는 소프트웨어에서 구현될 수 있다. 프로그램 코드는, (i) 여기에 개시된 기법들 및 시스템들의 기능(예컨대, 태스크들을 GPU(110)에 제공, GPU(110)에서 태스크들을 스케쥴링, GPU(110)에서 태스크들을 실행, 등); (ii) 여기에 개시된 기법들 및 시스템들의 제작(예컨대, GPU(110)의 제작); 또는 (iii) 여기에 개시된 기법들 및 시스템들의 기능과 제작의 조합을 포함하여, 본 발명의 실시예들을 가능하게 한다.

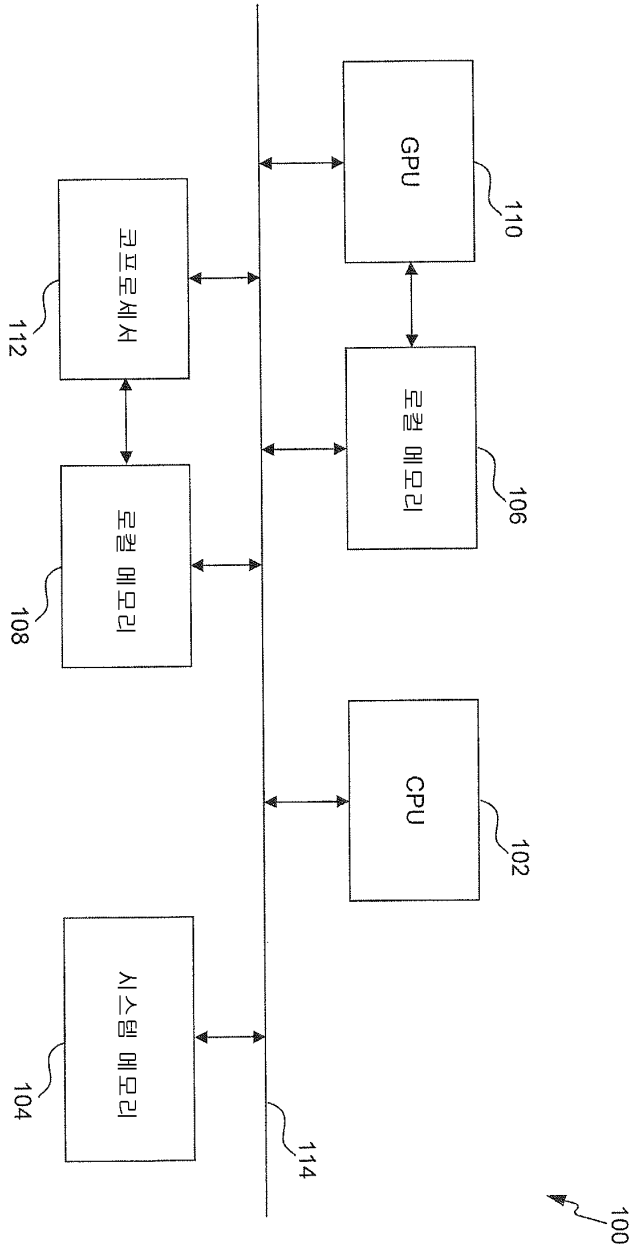
[0079] 이는, 예를 들어, 범용-프로그래밍 언어(예컨대, C 또는 C++); Verilong HDL을 포함하는 하드웨어-기술 언어(HDL), VHDL, 알테라 HDL(AHDL) 등을 포함하는 하드웨어-기술 언어; 또는 다른 사용가능한 프로그래밍 및/또는 스케메틱-캡쳐 툴들(예컨대, 회로-캡쳐 툴들)을 통해 달성될 수 있다. 프로그램 코드는 반도체, 자기 디스크, 또는 광학 디스크(예컨대, CD-ROM, DVD-ROM)와 같은 임의의 알려진 컴퓨터-판독가능 매체에 배치될 수 있다. 그런 식으로, 코드는 인터넷(Internet 및 internets)을 포함하는 통신 네트워크를 통해 전송될 수 있다. 위에서 기술된 시스템 및 기법들에 의해 제공된 구조 및/또는 달성된 기능들은 프로그램 코드에서 구현되는 코어(예컨대, GPU 코어)에 나타낼 수 있으며, 집적 회로 제품의 일부로서 하드웨어로 변환될 수 있다.

[0080] VII. 결론

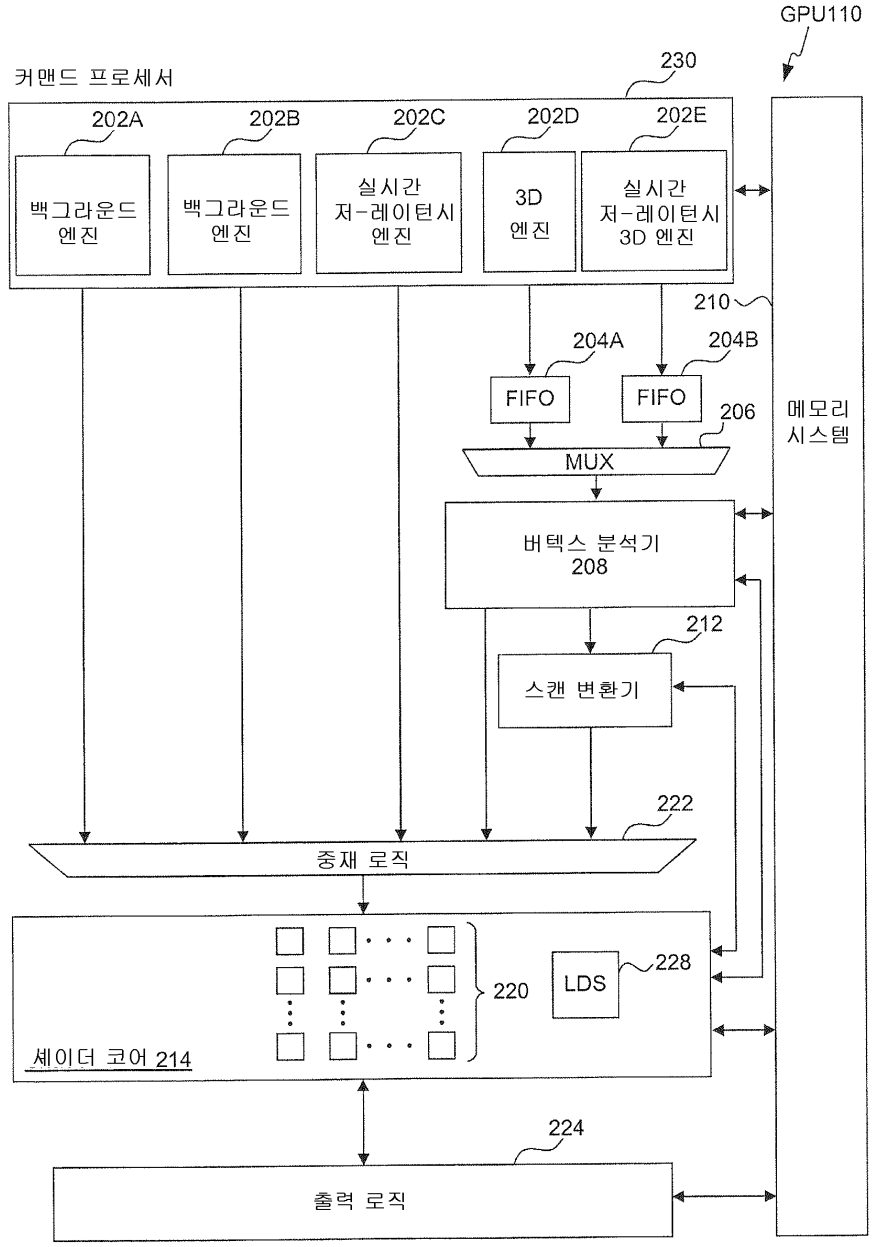
[0081] 개요 및 요약 부분이 아닌 상세한 설명 부분은 청구항들을 해석하는데 사용되도록 의도된 것임이 이해되어야 한다. 개요 및 요약 부분들은 발명자(들)에 의해 고려되는 본 발명의 모든 예시적 실시예들이 아닌 단지 한 두가지의 실시예들을 제시하는 것으로서, 본 발명 및 첨부된 청구항들을 어떠한 식으로든 제한하려 의도된 것이 아니다.

도면

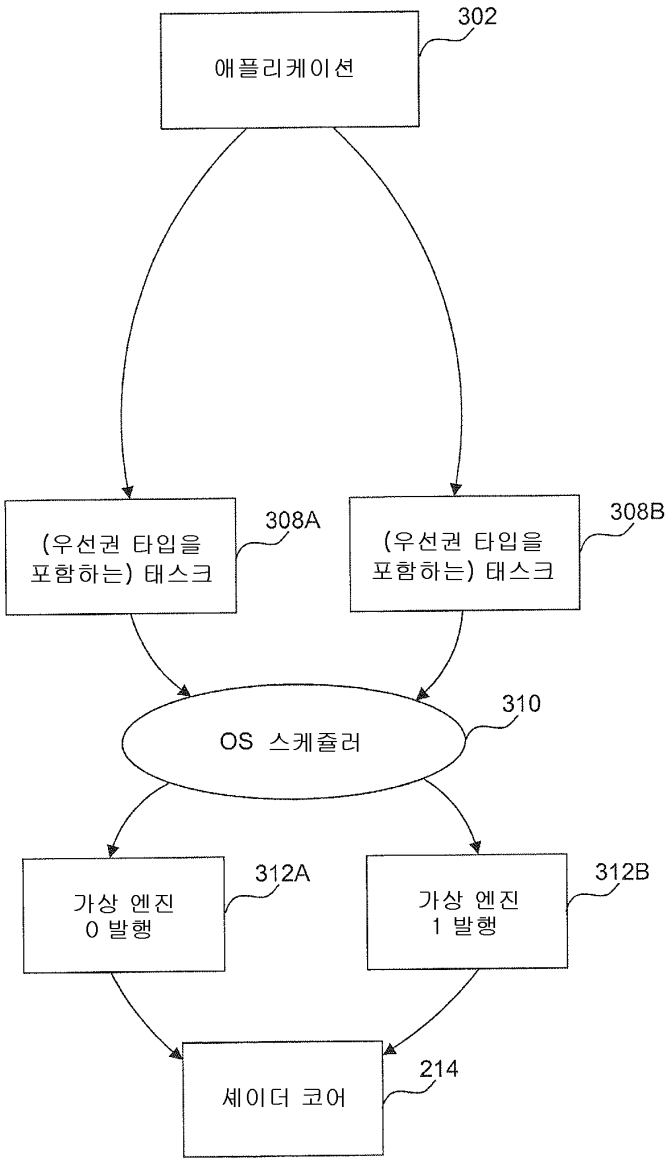
도면1


GPU
110
코프로세서
112
로컬 메모리
106
로컬 메모리
108
CPU
102
시스템 메모리
104
114
100

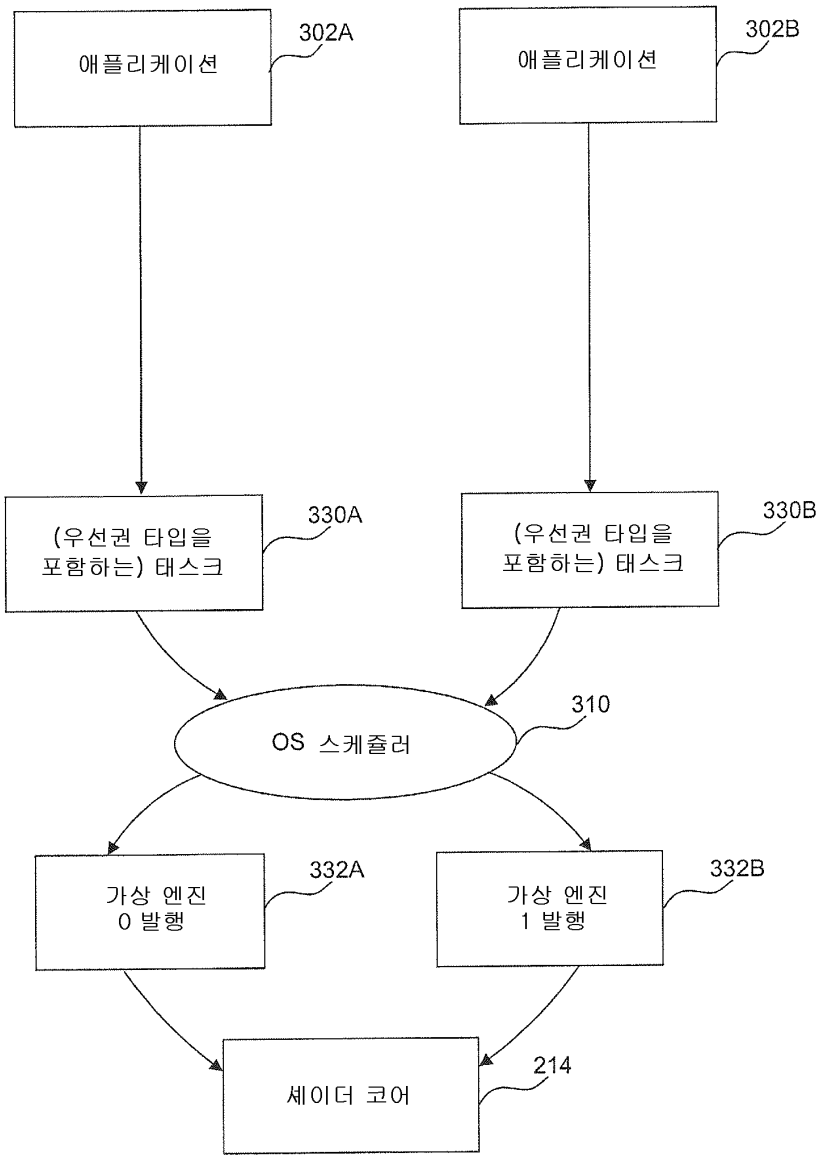
도면2


GPU110
커맨드 프로세서
230
202A
202B
202C
202D
202E
백그라운드 엔진
백그라운드 엔진
실시간 저-레이턴시 엔진
3D 엔진
실시간 저-레이턴시 3D 엔진
210
204A
204B
FIFO
FIFO
206
MUX
메모리 시스템
버텍스 분석기 208
212
스캔 변환기
222
중재 로직
220
LDS
228
셰이더 코어 214
224
출력 로직

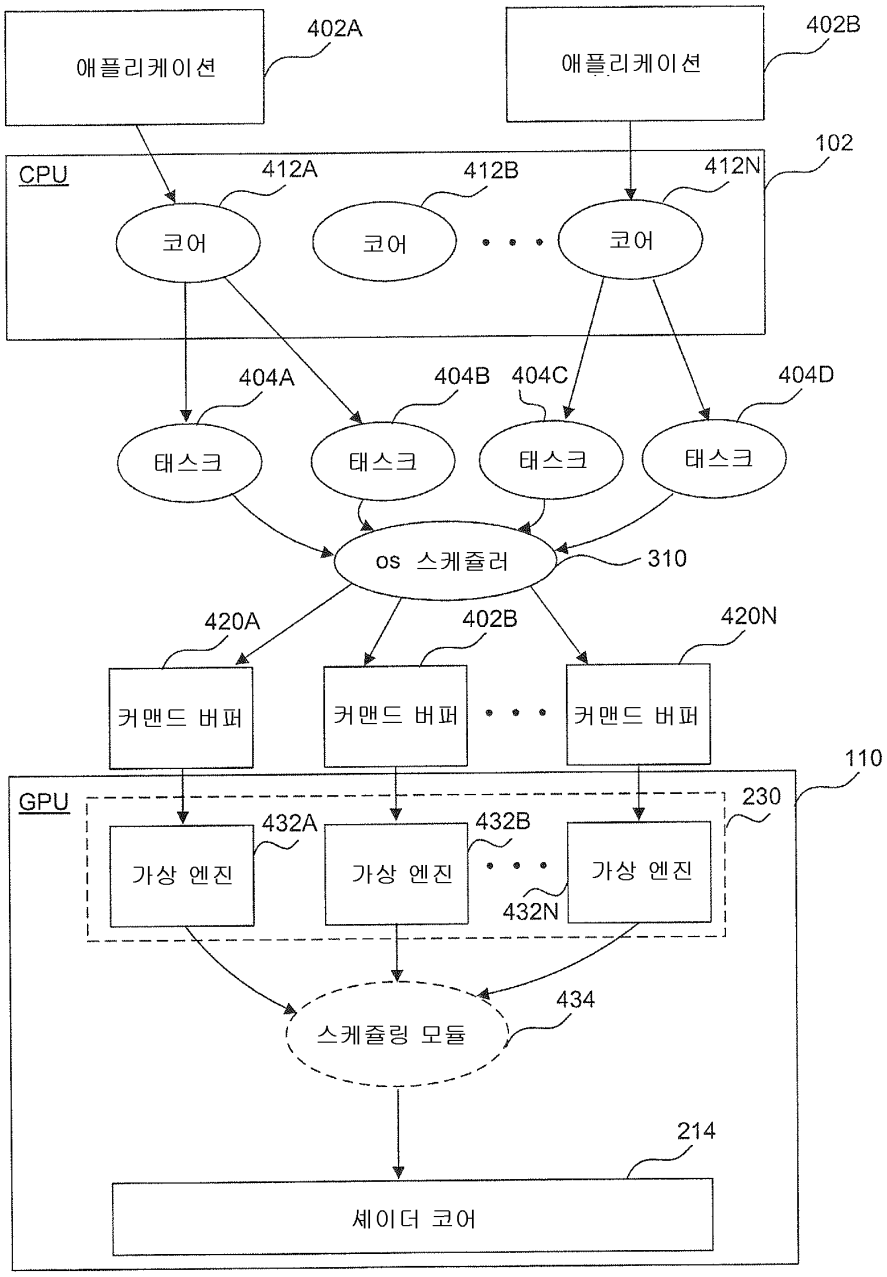
도면3a


302
애플리케이션
(우선권 타입을 포함하는) 태스크
308A
(우선권 타입을 포함하는) 태스크
308B
OS 스케쥴러
310
가상 엔진 0 발행
312A
가상 엔진 1 발행
312B
메이더 코어
214

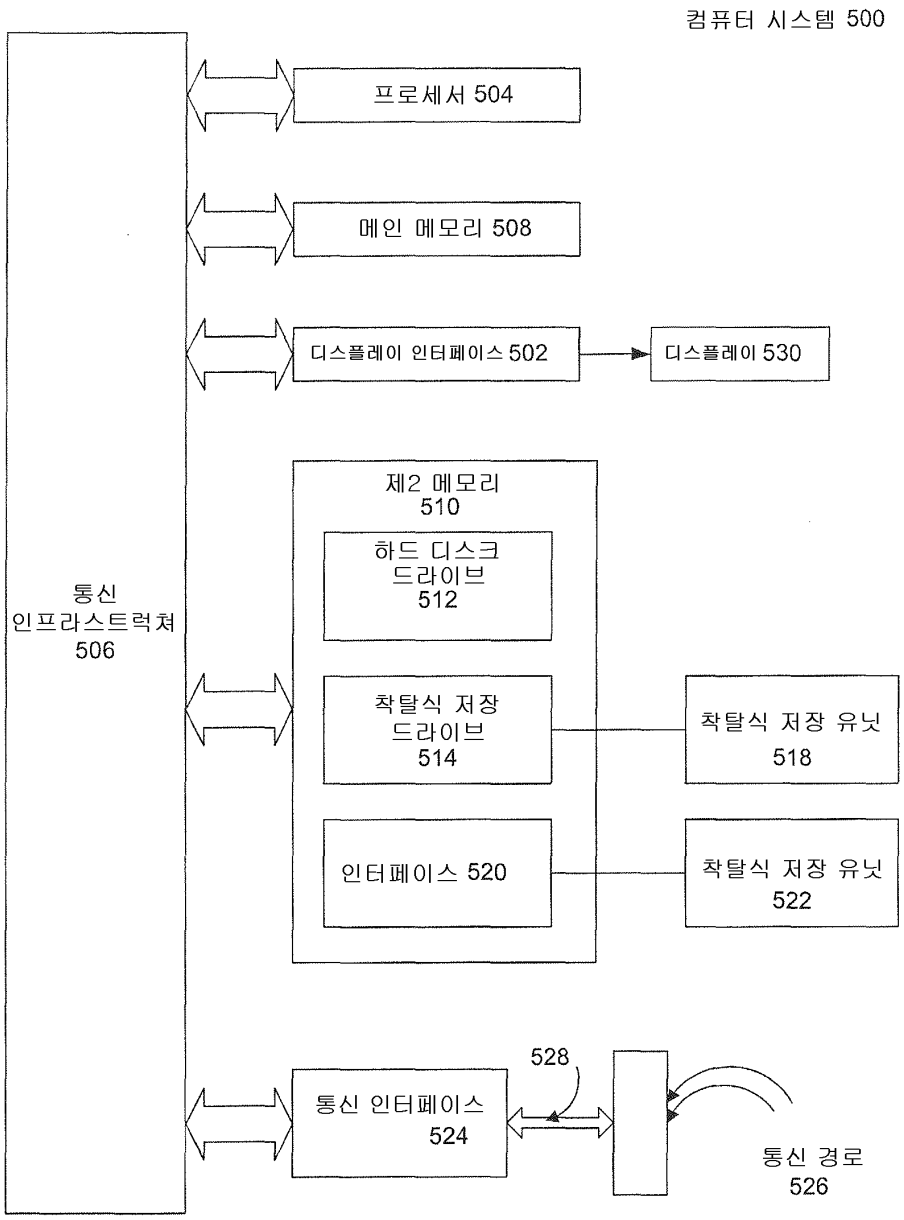
도면3b


애플리케이션
302A
애플리케이션
302B
(우선권 타입을 포함하는) 태스크
330A
(우선권 타입을 포함하는) 태스크
330B
OS 스케쥴러
310
가상 엔진 0 발행
332A
가상 엔진 1 발행
332B
셰이더 코어
214

도면4


애플리케이션
402A
애플리케이션
402B
CPU
102
412A
412B
412N
코어
코어
코어
404A
404B
404C
404D
태스크
태스크
태스크
태스크
os 스케쥴러
310
420A
402B
420N
커맨드 버퍼
커맨드 버퍼
커맨드 버퍼
110
GPU
230
432A
432B
가상 엔진
가상 엔진
가상 엔진
432N
434
스케쥴링 모듈
214
셰이더 코어

도면5


컴퓨터 시스템 500
프로세서 504
메인 메모리 508
디스플레이 인터페이스 502
디스플레이 530
제2 메모리
510
하드 디스크
드라이브
512
착탈식 저장
드라이브
514
착탈식 저장 유닛
518
인터페이스 520
착탈식 저장 유닛
522
통신
인프라스트럭쳐
506
528
통신 인터페이스
524
통신 경로
526