(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2015년10월30일
(11) 등록번호 10-1564815
(24) 등록일자 2015년10월26일

(51) 국제특허분류(Int. Cl.)
G06F 9/54 (2006.01) *G06F 9/38* (2006.01)
(21) 출원번호 10-2013-7010120
(22) 출원일자(국제) 2011년09월19일
심사청구일자 2013년04월19일
(85) 번역문제출일자 2013년04월19일
(65) 공개번호 10-2013-0060337
(43) 공개일자 2013년06월07일
(86) 국제출원번호 PCT/US2011/052196
(87) 국제공개번호 WO 2012/040121
국제공개일자 2012년03월29일
(30) 우선권주장
13/235,236 2011년09월16일 미국(US)
(뒷면에 계속)

(56) 선행기술조사문헌
The OpenCL specification 1.0 (2009.06.10.) [출처: https://www.khronos.org/registry/cl/specs/opencl-1.0.pdf]*
Ian Foster. Message Passing Interface (1995) [출처: http://www.mcs.anl.gov/~itf/dbpp/text/node94.html]*
US05604909 A
US20050138249 A1
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
퀄컴 인코포레이티드
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
(72) 발명자
부르드 알렉세이 브이
미국 92121 캘리포니아주 샌디에고 모어하우스 드라이브 5775
샤프 콜린 크리스토퍼
미국 92121 캘리포니아주 샌디에고 모어하우스 드라이브 5775
(뒷면에 계속)
(74) 대리인
특허법인코리아나

전체 청구항 수 : 총 34 항 심사관 : 유진태

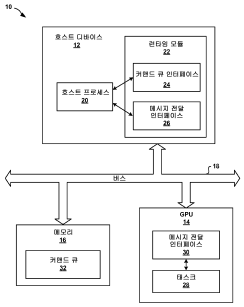
(54) 발명의 명칭 **다중-프로세서 컴퓨팅 플랫폼에서의 프로세서간 통신 기법들**

(57) 요 약

본 개시물은 다중-프로세서 컴퓨팅 플랫폼 내에 사용될 수도 있는 통신 기법들을 기술한다. 이 기법들은 일부 예들에서, 커맨드 큐들을 이용하여 태스크들을 개시하는 다중-프로세서 컴퓨팅 플랫폼 내에서 메시지 전달을 지원하는데 사용될 수도 있는 소프트웨어 인터페이스들을 제공할 수도 있다. 이 기법들은 추가적인

(뒷면에 계속)

대 표 도 - 도1



예들에서, 다중-프로세서 컴퓨팅 플랫폼 내에서 공유 메모리 프로세서간 통신을 위해 사용될 수도 있는 소프트웨어 인터페이스들을 제공할 수도 있다. 추가적인 예들에서, 이 기법들은 그래픽 프로세싱 유닛 (GPU) 과 호스트 CPU 사이에 메시지 전달 및/또는 공유 메모리 통신을 지원하는 하드웨어를 포함하는 GPU 를 제공할 수도 있다.

(72) 발명자

가르시아 가르시아 다비드 리헬

미국 92121 캘리포니아주 샌디에고 모어하우스 드라이브 5775

장 치홍

미국 92121 캘리포니아주 샌디에고 모어하우스 드라이브 5775

(30) 우선권주장

13/235,266 2011년09월16일 미국(US)

61/384,571 2010년09월20일 미국(US)

61/515,182 2011년08월04일 미국(US)

명 세 서

청구범위

청구항 1

호스트 디바이스로서,

하나 이상의 프로세서들;

상기 하나 이상의 프로세서들 상에서 실행하며, 상기 호스트 디바이스 상에서 실행하는 프로세스로부터 하나 이상의 인큐 명령 (enqueue instruction) 들을 수신하는 것에 응답하여, 복수의 커맨드들을 커맨드 큐에 배치하도록 구성된 커맨드 큐 인터페이스로서, 상기 복수의 커맨드들은, 상기 호스트 디바이스와 연관된 제 1 메모리 공간과 그래픽 프로세싱 유닛 (GPU) 과 연관된 제 2 메모리 공간 사이에 데이터를 전송하도록 상기 호스트 디바이스에 명령하는 제 1 커맨드를 포함하며, 상기 복수의 커맨드들은, 상기 GPU 상에서 태스크의 실행을 개시하도록 상기 호스트 디바이스에 명령하는 제 2 커맨드를 더 포함하는, 상기 커맨드 큐 인터페이스; 및

상기 하나 이상의 프로세서들 상에서 실행하며, 상기 태스크가 상기 GPU 상에서 실행중인 동안, 그리고 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 상기 호스트 디바이스 상에서 실행하는 상기 프로세스와 상기 GPU 상에서 실행하는 태스크 사이에 하나 이상의 메시지들을 전달하도록 구성된 메시지 전달 인터페이스를 포함하는, 호스트 디바이스.

청구항 2

제 1 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 메시지를 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 상기 GPU 상에서 실행하는 상기 태스크로 전송하도록 상기 메시지 전달 인터페이스에 명령하는 전송 명령을 포함하며,

상기 메시지 전달 인터페이스는 또한, 상기 전송 명령을 수신하는 것에 응답하여, 상기 태스크가 상기 GPU 상에서 실행중인 동안 상기 메시지를 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 상기 GPU 상에서 실행하는 상기 태스크로 전송하도록 구성되는, 호스트 디바이스.

청구항 3

제 1 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 상기 GPU 로부터, 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했다는 것을 나타내는 신호를 수신하는 것에 응답하여, 콜백 루틴을 인보크하도록 상기 메시지 전달 인터페이스에 명령하는 레지스터 콜백 루틴 명령을 포함하며,

상기 메시지 전달 인터페이스는 또한, 상기 GPU 로부터, 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했다는 것을 나타내는 상기 신호를 수신하는 것에 응답하여, 상기 레지스터 콜백 루틴 명령에 특정된 상기 콜백 루틴의 실행을 개시하도록 구성되는, 호스트 디바이스.

청구항 4

제 1 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했는지 여부를 나타내는 메시지 상태 정보에 대해 상기 GPU 를 폴링하도록 상기 메시지 전달 인터페이스에 명령하는 폴링 명령을 포함하며,

상기 메시지 전달 인터페이스는 또한, 상기 폴링 명령을 수신하는 것에 응답하여, 상기 메시지 상태 정보에 대해 상기 GPU 를 폴링하고, 상기 메시지 상태 정보가 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했다는 것을 나타낸다면, 상기 GPU 로부터 상기 메시지를 획득하도록 구성되는, 호스트 디바이스.

청구항 5

제 1 항에 있어서,

상기 GPU 상에서 실행하는 상기 태스크는, 메시지를 상기 GPU 상에서 실행하는 상기 태스크로부터 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로 전송하도록 상기 GPU 에 명령하는 명령을 포함하는, 호스트 디바이스.

청구항 6

제 1 항에 있어서,

상기 GPU 상에서 실행하는 상기 태스크는, 이용가능하다면, 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 상기 태스크로 전송된 메시지를 상기 태스크에 제공하도록 상기 GPU 에 명령하는 명령을 포함하는, 호스트 디바이스.

청구항 7

제 1 항에 있어서,

상기 메시지 전달 인터페이스는 또한, 상기 커맨드 큐 내에 어떠한 커맨드들도 배치하지 않고 상기 하나 이상의 메시지 전달 명령들을 실행하도록 구성되는, 호스트 디바이스.

청구항 8

호스트 디바이스의 하나 이상의 프로세서들 상에서 실행하는 커맨드 큐 인터페이스에 의해, 상기 호스트 디바이스 상에서 실행하는 프로세스로부터 하나 이상의 인큐 명령 (enqueue instruction) 들을 수신하는 것에 응답하여, 복수의 커맨드들을 커맨드 큐에 배치하는 단계로서, 상기 복수의 커맨드들은, 상기 호스트 디바이스와 연관된 제 1 메모리 공간과 그래픽 프로세싱 유닛 (GPU) 과 연관된 제 2 메모리 공간 사이에 데이터를 전송하도록 상기 호스트 디바이스에 명령하는 제 1 커맨드를 포함하며, 상기 복수의 커맨드들은, 상기 GPU 상에서 태스크의 실행을 개시하도록 상기 호스트 디바이스에 명령하는 제 2 커맨드를 더 포함하는, 상기 복수의 커맨드들을 커맨드 큐에 배치하는 단계; 및

상기 호스트 디바이스의 상기 하나 이상의 프로세서들 상에서 실행하는 메시지 전달 인터페이스에 의해, 상기 태스크가 상기 GPU 상에서 실행중인 동안, 그리고 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 상기 호스트 디바이스 상에서 실행하는 상기 프로세스와 상기 GPU 상에서 실행하는 태스크 사이에 하나 이상의 메시지들을 전달하는 단계를 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 방법.

청구항 9

제 8 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 메시지를 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 상기 GPU 상에서 실행하는 상기 태스크로 전송하도록 상기 메시지 전달 인터페이스에 명령하는 전송 명령을 포함하며,

상기 방법은, 상기 메시지 전달 인터페이스에 의해, 상기 태스크가 상기 GPU 상에서 실행중인 동안, 그리고 상기 전송 명령을 수신하는 것에 응답하여, 상기 메시지를 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 상기 GPU 상에서 실행하는 상기 태스크로 전송하는 단계를 더 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 방법.

청구항 10

제 8 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 상기 GPU 로부터, 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했다는 것을 나타내는 신호를 수신하는 것에 응답하여, 콜백 루틴을 인보크하도록 상기 메시지 전달 인터페이스에 명령하는 레지스터 콜백 루틴 명령을 포함하며,

상기 방법은, 상기 GPU 로부터, 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했다는 것을 나타내는 상기 신호를 수신하는 것에 응답하여, 상기 레지스터 콜백 루틴 명령에 특정된 상기 콜백 루틴의 실행을 개시하는 단계를 더 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 방법.

청구항 11

제 8 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했는지 여부를 나타내는 메시지 상태 정보에 대해 상기 GPU 를 폴링하도록 상기 메시지 전달 인터페이스에 명령하는 폴링 명령을 포함하며,

상기 방법은 :

상기 메시지 전달 인터페이스에 의해, 상기 폴링 명령을 수신하는 것에 응답하여, 상기 메시지 상태 정보에 대해 상기 GPU 를 폴링하는 단계; 및

상기 메시지 상태 정보가 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했다는 것을 나타낸다면, 상기 GPU 로부터 상기 메시지를 획득하는 단계를 더 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 방법.

청구항 12

제 8 항에 있어서,

상기 GPU 상에서 실행하는 상기 태스크는, 메시지를 상기 GPU 상에서 실행하는 상기 태스크로부터 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로 전송하도록 상기 GPU 에 명령하는 명령을 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 방법.

청구항 13

제 8 항에 있어서,

상기 GPU 상에서 실행하는 상기 태스크는, 이용가능하다면, 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 상기 태스크로 전송된 메시지를 상기 태스크에 제공하도록 상기 GPU 에 명령하는 명령을 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 방법.

청구항 14

제 8 항에 있어서,

상기 메시지 전달 인터페이스에 의해, 상기 커맨드 큐 내에 어떠한 커맨드들도 배치하지 않고 상기 하나 이상의 메시지 전달 명령들을 실행하는 단계를 더 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 방법.

청구항 15

호스트 디바이스 상에서 실행하는 프로세스로부터 하나 이상의 인큐 명령 (enqueue instruction) 들을 수신하는 것에 응답하여, 복수의 커맨드들을 커맨드 큐에 배치하는 수단으로서, 상기 복수의 커맨드들은, 상기 호스트 디바이스와 연관된 제 1 메모리 공간과 그래픽 프로세싱 유닛 (GPU) 과 연관된 제 2 메모리 공간 사이에 데이터를 전송하도록 상기 호스트 디바이스에 명령하는 제 1 커맨드를 포함하며, 상기 복수의 커맨드들은, 상기 GPU 상에서 태스크의 실행을 개시하도록 상기 호스트 디바이스에 명령하는 제 2 커맨드를 더 포함하는, 상기 복수의 커맨드들을 커맨드 큐에 배치하는 수단; 및

상기 태스크가 상기 GPU 상에서 실행중인 동안, 그리고 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 상기 호스트 디바이스 상에서 실행하는 상기 프로세스와 상기 GPU 상에서 실행하는 태스크 사이에 하나 이상의 메시지들을 전달하는 수단을 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 장치.

청구항 16

제 15 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 메시지를 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 상기 GPU 상에서 실행하는 상기 태스크로 전송하도록 상기 하나 이상의 메시지들을 전달하는 수단에 명령하는 전송 명령을 포함하며,

상기 장치는, 상기 전송 명령을 수신하는 것에 응답하여, 상기 태스크가 상기 GPU 상에서 실행중인 동안 상기 메시지를 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 상기 GPU 상에서 실행하는 상기 태스크로 전송하는 수단을 더 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 장치.

청구항 17

제 15 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 상기 GPU 로부터, 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했다는 것을 나타내는 신호를 수신하는 것에 응답하여, 콜백 루틴을 인보크하도록 상기 하나 이상의 메시지들을 전달하는 수단에 명령하는 레지스터 콜백 루틴 명령을 포함하며,

상기 장치는, 상기 GPU 로부터, 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했다는 것을 나타내는 상기 신호를 수신하는 것에 응답하여, 상기 레지스터 콜백 루틴 명령에 특정된 상기 콜백 루틴의 실행을 개시하는 수단을 더 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 장치.

청구항 18

제 15 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했는지 여부를 나타내는 메시지 상태 정보에 대해 상기 GPU 를 폴링하도록 상기 하나 이상의 메시지들을 전달하는 수단에 명령하는 폴링 명령을 포함하며,

상기 장치는 :

상기 폴링 명령을 수신하는 것에 응답하여, 상기 메시지 상태 정보에 대해 상기 GPU 를 폴링하는 수단; 및

상기 메시지 상태 정보가 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했다는 것을 나타낸다면, 상기 GPU 로부터 상기 메시지를 획득하는 수단을 더 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 장치.

청구항 19

명령들을 포함하는 컴퓨터 판독가능 매체로서,

상기 명령들은, 하나 이상의 프로세서들로 하여금 :

호스트 디바이스 상에서 실행하는 프로세스로부터 하나 이상의 인큐 명령 (enqueue instruction) 들을 수신하는 것에 응답하여, 복수의 커맨드들을 커맨드 큐에 배치하는 것으로서, 상기 복수의 커맨드들은, 상기 호스트 디바이스와 연관된 제 1 메모리 공간과 그래픽 프로세싱 유닛 (GPU) 과 연관된 제 2 메모리 공간 사이에 데이터를 전송하도록 상기 호스트 디바이스에 명령하는 제 1 커맨드를 포함하며, 상기 복수의 커맨드들은, 상기 GPU 상에서 태스크의 실행을 개시하도록 상기 호스트 디바이스에 명령하는 제 2 커맨드를 더 포함하는, 상기 복수의 커맨드들을 커맨드 큐에 배치하도록 하고;

상기 태스크가 상기 GPU 상에서 실행중인 동안, 그리고 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 상기 호스트 디바이스 상에서 실행하는 상기 프로세스와 상기 GPU 상에서 실행하는 태스크 사이에 하나 이상의 메시지들을 전달하도록 하는, 컴퓨터 판독가능 매체.

청구항 20

제 19 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 메시지를 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터

상기 GPU 상에서 실행하는 상기 태스크로 전송하도록 상기 하나 이상의 프로세서들에 명령하는 전송 명령을 포함하며,

상기 컴퓨터 판독가능 매체는, 상기 하나 이상의 프로세서들로 하여금, 상기 전송 명령을 수신하는 것에 응답하여, 상기 태스크가 상기 GPU 상에서 실행중인 동안 상기 메시지를 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 상기 GPU 상에서 실행하는 상기 태스크로 전송하도록 하는 명령들을 더 포함하는, 컴퓨터 판독가능 매체.

청구항 21

제 19 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 상기 GPU 로부터, 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했다는 것을 나타내는 신호를 수신하는 것에 응답하여, 콜백 루틴을 인보크하도록 상기 하나 이상의 프로세서들에 명령하는 레지스터 콜백 루틴 명령을 포함하며,

상기 컴퓨터 판독가능 매체는, 상기 하나 이상의 프로세서들로 하여금, 상기 GPU 로부터, 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했다는 것을 나타내는 상기 신호를 수신하는 것에 응답하여, 상기 레지스터 콜백 루틴 명령에 특정된 상기 콜백 루틴의 실행을 개시하도록 하는 명령들을 더 포함하는, 컴퓨터 판독가능 매체.

청구항 22

제 19 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했는지 여부를 나타내는 메시지 상태 정보에 대해 상기 GPU 를 폴링하도록 상기 하나 이상의 프로세서들에 명령하는 폴링 명령을 포함하며,

상기 컴퓨터 판독가능 매체는, 상기 하나 이상의 프로세서들로 하여금 :

상기 폴링 명령을 수신하는 것에 응답하여, 상기 메시지 상태 정보에 대해 상기 GPU 를 폴링하도록 하고;

상기 메시지 상태 정보가 상기 GPU 상에서 실행하는 상기 태스크가 메시지를 전송했다는 것을 나타낸다면, 상기 GPU 로부터 상기 메시지를 획득하도록 하는 명령들을 더 포함하는, 컴퓨터 판독가능 매체.

청구항 23

그래픽 프로세싱 유닛 (GPU) 으로서,

태스크를 실행하도록 구성된 하나 이상의 프로세서들;

호스트 디바이스에 의해 액세스가능한 하나 이상의 레지스터들; 및

상기 태스크가 상기 하나 이상의 프로세서들 상에서 실행중인 동안, 그리고 상기 하나 이상의 프로세서들 상에서 실행하는 상기 태스크로부터 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 상기 하나 이상의 레지스터들을 통해, 상기 하나 이상의 프로세서들 상에서 실행하는 상기 태스크와 상기 호스트 디바이스 상에서 실행하는 프로세스 사이에 하나 이상의 메시지들을 전달하도록 구성된 메시지 전달 모듈을 포함하는, 그래픽 프로세싱 유닛 (GPU).

청구항 24

제 23 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 메시지를 상기 GPU 상에서 실행하는 상기 태스크로부터 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로 전송하도록 상기 메시지 전달 모듈에 명령하는 전송 명령을 포함하며,

상기 메시지 전달 모듈은 또한, 상기 하나 이상의 레지스터들에 상기 메시지와 연관된 메시지 데이터를 저장하도록 구성되는, 그래픽 프로세싱 유닛 (GPU).

청구항 25

제 23 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 이용가능하다면, 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 상기 태스크로 전송된 메시지를 상기 태스크에 제공하도록 상기 메시지 전달 모듈에 명령하는 수신 명령을 포함하며,

상기 메시지 전달 모듈은 또한, 상기 하나 이상의 레지스터들로부터 상기 메시지와 연관된 메시지 데이터를 획득하도록 구성되는, 그래픽 프로세싱 유닛 (GPU).

청구항 26

그래픽 프로세싱 유닛 (GPU) 의 메시지 전달 모듈에 의해, 상기 GPU 상에서 실행하는 태스크로부터 하나 이상의 메시지 전달 명령들을 수신하는 단계; 및

상기 태스크가 상기 GPU 상에서 실행중인 동안, 그리고 상기 GPU 상에서 실행하는 상기 태스크로부터 상기 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 호스트 디바이스에 의해 액세스가능한 상기 GPU 내의 하나 이상의 레지스터들을 통해, 상기 GPU 상에서 실행하는 상기 태스크와 상기 호스트 디바이스 상에서 실행하는 프로세스 사이에 하나 이상의 메시지들을 전달하는 단계를 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 방법.

청구항 27

제 26 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 메시지를 상기 GPU 상에서 실행하는 상기 태스크로부터 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로 전송하도록 상기 메시지 전달 모듈에 명령하는 전송 명령을 포함하며,

상기 방법은, 상기 하나 이상의 레지스터들에 상기 메시지와 연관된 메시지 데이터를 저장하는 단계를 더 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 방법.

청구항 28

제 26 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 이용가능하다면, 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 상기 태스크로 전송된 메시지를 상기 태스크에 제공하도록 상기 메시지 전달 모듈에 명령하는 수신 명령을 포함하며,

상기 방법은, 상기 하나 이상의 레지스터들로부터 상기 메시지와 연관된 메시지 데이터를 획득하는 단계를 더 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 방법.

청구항 29

그래픽 프로세싱 유닛 (GPU) 상에서 실행하는 태스크로부터 하나 이상의 메시지 전달 명령들을 수신하는 수단; 및

상기 태스크가 상기 GPU 상에서 실행중인 동안, 그리고 상기 GPU 상에서 실행하는 상기 태스크로부터 상기 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 호스트 디바이스에 의해 액세스가능한 상기 GPU 내의 하나 이상의 레지스터들을 통해, 상기 GPU 상에서 실행하는 상기 태스크와 상기 호스트 디바이스 상에서 실행하는 프로세스 사이에 하나 이상의 메시지들을 전달하는 수단을 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 장치.

청구항 30

제 29 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 메시지를 상기 GPU 상에서 실행하는 상기 태스크로부터 상기 호스트

디바이스 상에서 실행하는 상기 프로세스로 전송하도록 상기 전달하는 수단에 명령하는 전송 명령을 포함하며,

상기 장치는, 상기 하나 이상의 레지스터들에 상기 메시지와 연관된 메시지 데이터를 저장하는 수단을 더 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 장치.

청구항 31

제 29 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 이용가능하다면, 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 상기 태스크로 전송된 메시지를 상기 태스크에 제공하도록 상기 전달하는 수단에 명령하는 수신 명령을 포함하며,

상기 장치는, 상기 하나 이상의 레지스터들로부터 상기 메시지와 연관된 메시지 데이터를 획득하는 수단을 더 포함하는, 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세서간 통신을 위한 장치.

청구항 32

명령들을 포함하는 컴퓨터 판독가능 매체로서,

상기 명령들은, 하나 이상의 프로세서들로 하여금 :

그래픽 프로세싱 유닛 (GPU) 상에서 실행하는 태스크로부터 하나 이상의 메시지 전달 명령들을 수신하도록 하고;

상기 태스크가 상기 GPU 상에서 실행중인 동안, 그리고 상기 GPU 상에서 실행하는 상기 태스크로부터 상기 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 호스트 디바이스에 의해 액세스가능한 상기 GPU 내의 하나 이상의 레지스터들을 통해, 상기 GPU 상에서 실행하는 상기 태스크와 상기 호스트 디바이스 상에서 실행하는 프로세스 사이에 하나 이상의 메시지들을 전달하도록 하는, 컴퓨터 판독가능 매체.

청구항 33

제 32 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 메시지를 상기 GPU 상에서 실행하는 상기 태스크로부터 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로 전송하도록 상기 하나 이상의 프로세서들에 명령하는 전송 명령을 포함하며,

상기 컴퓨터 판독가능 매체는, 상기 하나 이상의 프로세서들로 하여금, 상기 하나 이상의 레지스터들에 상기 메시지와 연관된 메시지 데이터를 저장하도록 하는 명령들을 더 포함하는, 컴퓨터 판독가능 매체.

청구항 34

제 32 항에 있어서,

상기 하나 이상의 메시지 전달 명령들은, 이용가능하다면, 상기 호스트 디바이스 상에서 실행하는 상기 프로세스로부터 상기 태스크로 전송된 메시지를 상기 태스크에 제공하도록 상기 하나 이상의 프로세서들에 명령하는 수신 명령을 포함하며,

상기 컴퓨터 판독가능 매체는, 상기 하나 이상의 프로세서들로 하여금, 상기 하나 이상의 레지스터들로부터 상기 메시지와 연관된 메시지 데이터를 획득하도록 하는 명령들을 더 포함하는, 컴퓨터 판독가능 매체.

발명의 설명

기 술 분 야

[0001] 본 개시물은 컴퓨팅 플랫폼들에 관한 것으로, 보다 상세하게는, 다중 프로세서들을 포함하는 컴퓨팅 플랫폼들에 관한 것이다.

배 경 기 술

[0002] 다중 프로세서들을 포함하는 컴퓨팅 플랫폼들은 높은 계산 강도 요구사항들 및/또는 높은 데이터 스루풋 요구사

항들을 갖는 애플리케이션들의 성능을 향상시키는데 사용된다. 다중-프로세서 컴퓨팅 플랫폼은 계산 집약적인 태스크들의 성능을 오프로드하기 위해 호스트 중앙 처리 유닛 (CPU) 이 이용할 수도 있는 하나 이상의 컴퓨팅 디바이스들 및 호스트 디바이스의 역할을 할 수도 있는 범용 CPU 를 포함하여, 전체 시스템의 성능을 향상시킬 수도 있다. 일부 경우들에서, 하나 이상의 컴퓨팅 디바이스들은 소정의 유형들의 태스크들을 호스트 CPU 보다 더 효율적으로 프로세싱하도록 특별히 설계될 수도 있으며, 이는 전체 시스템에 추가적인 성능 향상들을 제공할 수도 있다. 예를 들어, 하나 이상의 컴퓨팅 디바이스들은 병렬 알고리즘들을 호스트 CPU 보다 더 효율적으로 실행하도록 특별히 설계될 수도 있다.

[0003] 다중-프로세서 컴퓨팅 시스템에 사용될 수도 있는 컴퓨팅 디바이스의 하나의 유형은 그래픽 프로세싱 유닛 (GPU) 이다. 전통적으로, GPU들은 디스플레이 디바이스에 대한 3 차원 (3D) 그래픽들의 실시간 렌더링을 위해 특별히 설계되었지만 통상 프로그래밍가능하지 않은 (즉, 컴파일된 프로그램이 GPU 로 다운로드되어 GPU 상에서 실행될 수 없었다) 고정된 기능 하드웨어를 포함하였다. 그러나 보다 최근에는, 프로그래밍가능 셰이더 유닛들 (shader units) 의 개발로 인해, 많은 GPU 의 아키텍처가 많은 병렬 프로세싱 엘리먼트들을 포함하는 프로그래밍가능 아키텍처로 시프트하였다. 프로그래밍가능 아키텍처는 GPU 가 그래픽들 동작들 뿐만 아니라 범용 컴퓨팅 태스크들의 실행을 고도 병렬 방식으로 용이하게 하는 것을 허용한다.

[0004] 범용 넌-그래픽 특정의 컴퓨팅 태스크들을 실행하는데 GPU 를 이용하는 것은 본원에서 GPGPU (General-Purpose computation on Graphics Processing Units) 로, 또는 대안적으로는, GPU 컴퓨팅으로 지칭될 수도 있다. 일부 경우들에서, GPU들은 그래픽 특정이 아닌 애플리케이션 프로그래밍 인터페이스들 (API) 을 이용가능하게 함으로써, 범용 컴퓨팅 태스크들의 실행을 위한 GPU 의 프로그래밍을 용이하게 할 수도 있다. GPU 컴퓨팅 태스크들은 계산 집약적인 태스크들을 포함하고/하거나 고도의 병렬성, 예컨대, 매트릭스 계산들, 신호 프로세싱 계산들, 통계적 알고리즘들, 분자 모델링 애플리케이션들, 재무 애플리케이션들, 메디칼 이미징, 암호 해독 애플리케이션들 등을 포함할 수도 있다.

[0005] GPU 는 다중-프로세서 컴퓨팅 플랫폼에 사용될 수 있는 컴퓨팅 디바이스의 하나의 유형일 뿐이며, GPU 에 더해서 또는 대신에, 다른 유형들의 컴퓨팅 디바이스들이 또한 사용될 수도 있다. 예를 들어, 다중-프로세서 컴퓨팅 플랫폼에 사용될 수도 있는 다른 유형들의 컴퓨팅 디바이스들은 예컨대, 추가적인 CPU, 디지털 신호 프로세서 (DSP), 셀 광대역 엔진 (셀/BE) 프로세서 또는 임의의 다른 유형의 프로세싱 유닛을 포함한다.

[0006] 다수의 컴퓨팅 디바이스들을 갖는 다중-프로세서 컴퓨팅 플랫폼은 동종 플랫폼 또는 이종 플랫폼 중 어느 하나일 수도 있다. 동종 플랫폼에서, 모든 컴퓨팅 디바이스들은 공통 명령 세트 아키텍처 (ISA; instruction set architecture) 를 공유한다. 이에 반해, 이종 플랫폼은 상이한 ISA들을 가진 2 개 이상의 컴퓨팅 디바이스들을 포함할 수도 있다. 일반적으로, 상이한 유형들의 컴퓨팅 디바이스들은 상이한 ISA들을 가질 수도 있고, 또한 동일한 유형의 상이한 브랜드들의 컴퓨팅 디바이스들은 상이한 ISA들을 가질 수도 있다.

[0007] 다중-프로세서 컴퓨팅 플랫폼의 성능은 멀티-코어 컴퓨팅 디바이스들 및/또는 매니-코어 컴퓨팅 디바이스들을 이용함으로써 추가로 향상될 수도 있다. 멀티-코어 컴퓨팅 디바이스의 일 예는 복수의 프로세싱 코어들을 갖는 프로그래밍가능 셰이더 유닛을 포함하는, 위에서 설명한 GPU 이다. 그러나, CPU들은 또한 다수의 프로세싱 코어들을 포함하도록 설계될 수도 있다. 일반적으로, 다수의 프로세싱 코어들을 포함하는 임의의 칩 또는 다이 (die) 가 멀티-코어 프로세서인 것으로 간주될 수도 있다. 프로세싱 코어는 특정 피스의 데이터에 대한 명령을 실행가능한 프로세싱 유닛을 지칭할 수도 있다. 예를 들어, GPU 내의 단일 ALU (arithmetic logic unit) 유닛 또는 벡터 프로세서가 프로세싱 코어인 것으로 간주될 수도 있다. 매니-코어 프로세서들은 일반적으로 상대적으로 높은 수의 코어들, 예컨대, 10개보다 많은 코어들을 갖는 멀티-코어 프로세서들을 지칭하며, 더 작은 수의 코어들을 가진 멀티-코어 프로세서들을 설계하는데 사용되는 기법들과는 상이한 기법들을 이용하여 통상 설계된다. 멀티-코어 프로세서들은 소프트웨어 프로그램이, 단일 칩 상의 다수의 코어들 상에서 병렬로, 예컨대, 동시발생적으로 실행하는 것을 허용함으로써, 성능 향상을 제공한다.

[0008] 병렬 프로그래밍 모델은 프로그램이 다수의 프로세싱 코어들 상에서 동시발생적으로 실행되는 것을 허용하도록 설계된 프로그래밍 모델을 지칭한다. 그 프로그램은 멀티-쓰레드된 프로그램일 수도 있으며, 이 경우, 단일 쓰레드가 각각의 프로세싱 코어 상에서 동작할 수도 있다. 일부 예들에서, 단일 컴퓨팅 디바이스는 프로그램을 실행하는데 사용되는 프로세싱 코어들의 모두를 포함할 수도 있다. 다른 예들에서, 프로그램을 실행하는데 사용되는 프로세싱 코어들의 일부는 동일한 유형 또는 상이한 유형의 상이한 컴퓨팅 디바이스들 상에 로케이트될 수도 있다.

[0009] 크로스-플랫폼, 크로스-벤더, 이종 컴퓨팅 플랫폼, 병렬 프로그래밍 모델 애플리케이션 프로그래밍 인터페이스

(API) 가 상이한 ISA들을 구현하는 상이한 벤더들에 의해 잠재적으로 만들어진 상이한 유형들의 컴퓨팅 디바이스들을 포함하는 이종, 멀티-코어 컴퓨팅 플랫폼의 병렬 프로그래밍에 공통 언어 사양을 제공하기 위해 사용될 수도 있다. 개방 컴퓨팅 언어 (OpenCL™) 는 크로스-플랫폼, 크로스-벤더, 이종 컴퓨팅 플랫폼, 병렬 프로그래밍 API 의 일 예이다. 이러한 API들은 GPU 상에서 더 일반화된 데이터 프로세싱이 가능하도록 설계될 수도 있다. 예를 들어, 계산 언어를 통해서 확장된 세이더 서브시스템 능력들을 노출하는 것을 넘어서, 이들 API들은 데이터 흐름 및 제어 경로들을 GPU 로 넌-그래픽 특정의 방식으로 일반화할 수도 있다. 그러나, 현재, 이러한 API들에 의해 제공되는 명령 세트들은 GPU 의 하드웨어 아키텍처에 기초하며, 따라서, 기존의 GPU 아키텍처들과 호환가능한 기능성에 제한된다.

발명의 내용

과제의 해결 수단

[0010] 본 개시물은 다중-프로세서 컴퓨팅 플랫폼 내에서 사용될 수도 있는 통신 기법들을 설명한다. 이 기법들은 일부 예들에서, 커맨드 큐들을 이용하여 태스크들을 개시하는 다중-프로세서 컴퓨팅 플랫폼 내에서 메시지 전달 (message passing) 을 지원하는데 사용될 수도 있는 소프트웨어 인터페이스들을 제공할 수도 있다. 이 기법들은 추가적인 예들에서, 다중-프로세서 컴퓨팅 플랫폼 내에서 공유 메모리 프로세서간 통신을 위해 사용될 수도 있는 소프트웨어 인터페이스들을 제공할 수도 있다. 추가적인 예들에서, 기법들은 GPU 와 호스트 CPU 사이에 메시지 전달 및/또는 공유 메모리 통신을 지원하는 하드웨어를 포함하는 그래픽 프로세싱 유닛 (GPU) 을 제공할 수도 있다.

[0011] 일 예에서, 본 개시물은 하나 이상의 프로세서들을 포함하는 호스트 디바이스를 설명한다. 이 디바이스는 하나 이상의 프로세서들 상에서 실행하며, 호스트 디바이스 상에서 실행하는 프로세스로부터 하나 이상의 인큐 명령 (enqueue instruction) 들을 수신하는 것에 응답하여 복수의 커맨드들을 커맨드 큐에 배치하도록 구성된 커맨드 큐 인터페이스를 더 포함한다. 복수의 커맨드들은 호스트 디바이스와 연관된 제 1 메모리 공간과 그래픽 프로세싱 유닛 (GPU) 과 연관된 제 2 메모리 공간 사이에 데이터를 전송하도록 호스트 디바이스에 명령하는 제 1 커맨드를 포함한다. 복수의 커맨드들은 GPU 상에서 태스크의 실행을 개시하도록 호스트 디바이스에 명령하는 제 2 커맨드를 더 포함한다. 이 디바이스는 하나 이상의 프로세서들 상에서 실행하며, 태스크가 GPU 상에서 실행중인 동안, 그리고 호스트 디바이스 상에서 실행하는 프로세스로부터 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 호스트 디바이스 상에서 실행하는 프로세스와 GPU 상에서 실행하는 태스크 사이에 하나 이상의 메시지들을 전달하도록 구성된 메시지 전달 인터페이스를 더 포함한다.

[0012] 또다른 예에서, 본 개시물은 호스트 디바이스의 하나 이상의 프로세서들 상에서 실행하는 커맨드 큐 인터페이스에 의해, 호스트 디바이스 상에서 실행하는 프로세스로부터 하나 이상의 인큐 명령들을 수신하는 것에 응답하여 복수의 커맨드들을 커맨드 큐에 배치하는 단계를 포함하는 방법을 설명한다. 복수의 커맨드들은 호스트 디바이스와 연관된 제 1 메모리 공간과 그래픽 프로세싱 유닛 (GPU) 과 연관된 제 2 메모리 공간 사이에 데이터를 전송하도록 호스트 디바이스에 명령하는 제 1 커맨드를 포함한다. 복수의 커맨드들은 GPU 상에서 태스크의 실행을 개시하도록 호스트 디바이스에 명령하는 제 2 커맨드를 더 포함한다. 이 방법은, 호스트 디바이스의 하나 이상의 프로세서들 상에서 실행하는 메시지 전달 인터페이스에 의해, 태스크가 GPU 상에서 실행중인 동안, 그리고 호스트 디바이스 상에서 실행하는 프로세스로부터 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 호스트 디바이스 상에서 실행하는 프로세스와 GPU 상에서 실행하는 태스크 사이에 하나 이상의 메시지들을 전달하는 단계를 더 포함한다.

[0013] 또다른 예에서, 본 개시물은 호스트 디바이스 상에서 실행하는 프로세스로부터 하나 이상의 인큐 명령들을 수신하는 것에 응답하여 복수의 커맨드들을 커맨드 큐에 배치하는 수단을 포함하는 장치를 설명한다. 복수의 커맨드들은 호스트 디바이스와 연관된 제 1 메모리 공간과 그래픽 프로세싱 유닛 (GPU) 과 연관된 제 2 메모리 공간 사이에 데이터를 전송하도록 호스트 디바이스에 명령하는 제 1 커맨드를 포함한다. 복수의 커맨드들은 GPU 상에서 태스크의 실행을 개시하도록 호스트 디바이스에 명령하는 제 2 커맨드를 더 포함한다. 이 장치는 태스크가 GPU 상에서 실행중인 동안, 그리고 호스트 디바이스 상에서 실행하는 프로세스로부터 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 호스트 디바이스 상에서 실행하는 프로세스와 GPU 상에서 실행하는 태스크 사이에 하나 이상의 메시지들을 전달하는 수단을 더 포함한다.

[0014] 또다른 예에서, 본 개시물은, 하나 이상의 프로세서로 하여금, 호스트 디바이스 상에서 실행하는 프로세스로부터 하나 이상의 인큐 명령들을 수신하는 것에 응답하여, 복수의 커맨드들을 커맨드 큐에 배치하도록 하는 명령

들을 포함하는 컴퓨터 판독가능 저장 매체를 설명한다. 복수의 커맨드들은 호스트 디바이스와 연관된 제 1 메모리 공간과 그래픽 프로세싱 유닛 (GPU) 과 연관된 제 2 메모리 공간 사이에 데이터를 전송하도록 호스트 디바이스에 명령하는 제 1 커맨드를 포함한다. 복수의 커맨드들은 GPU 상에서 태스크의 실행을 개시하도록 호스트 디바이스에 명령하는 제 2 커맨드를 더 포함한다. 컴퓨터 판독가능 저장 매체는, 하나 이상의 프로세서들로 하여금, 태스크가 GPU 상에서 실행중인 동안, 그리고 호스트 디바이스 상에서 실행하는 프로세스로부터 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 호스트 디바이스 상에서 실행하는 프로세스와 GPU 상에서 실행하는 태스크 사이에 하나 이상의 메시지들을 전달하도록 하는 명령들을 더 포함한다.

[0015] 또다른 예에서, 본 개시물은 태스크를 실행하도록 구성된 하나 이상의 프로세서들을 포함하는 그래픽 프로세싱 유닛 (GPU) 을 설명한다. GPU 는 호스트 디바이스에 의해 액세스가능한 하나 이상의 레지스터들을 더 포함한다. GPU 는 태스크가 하나 이상의 프로세서들 상에서 실행중인 동안, 그리고 하나 이상의 프로세서들 상에서 실행하는 태스크로부터 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 하나 이상의 레지스터들을 통해, 하나 이상의 프로세서들 상에서 실행하는 태스크와 호스트 디바이스 상에서 실행하는 프로세스 사이에 하나 이상의 메시지들을 전달하도록 구성된 메시지 전달 모듈을 더 포함한다.

[0016] 또다른 예에서, 본 개시물은 그래픽 프로세싱 유닛 (GPU) 의 메시지 전달 모듈에 의해, GPU 상에서 실행하는 태스크로부터 하나 이상의 메시지 전달 명령들을 수신하는 단계를 포함하는 방법을 설명한다. 이 방법은 태스크가 GPU 상에서 실행중인 동안, 그리고 GPU 상에서 실행하는 태스크로부터 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 호스트 디바이스에 의해 액세스가능한 GPU 내의 하나 이상의 레지스터들을 통해, GPU 상에서 실행하는 태스크와 호스트 디바이스 상에서 실행하는 프로세스 사이에 하나 이상의 메시지들을 전달하는 단계를 더 포함한다.

[0017] 또다른 예에서, 본 개시물은 그래픽 프로세싱 유닛 (GPU) 상에서 실행하는 태스크로부터 하나 이상의 메시지 전달 명령들을 수신하는 수단을 포함하는 장치를 설명한다. 이 장치는 태스크가 GPU 상에서 실행중인 동안, 그리고 GPU 상에서 실행하는 태스크로부터 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 호스트 디바이스에 의해 액세스가능한 GPU 내의 하나 이상의 레지스터들을 통해, GPU 상에서 실행하는 태스크와 호스트 디바이스 상에서 실행하는 프로세스 사이에 하나 이상의 메시지들을 전달하는 수단을 더 포함한다.

[0018] 또다른 예에서, 본 개시물은, 하나 이상의 프로세서들로 하여금, 그래픽 프로세싱 유닛 (GPU) 상에서 실행하는 태스크로부터 하나 이상의 메시지 전달 명령들을 수신하도록 하는 명령들을 포함하는 컴퓨터 판독가능 매체를 설명한다. 컴퓨터 판독가능 저장 매체는, 하나 이상의 프로세서들로 하여금, 태스크가 GPU 상에서 실행중인 동안, 그리고 GPU 상에서 실행하는 태스크로부터 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 호스트 디바이스에 의해 액세스가능한 GPU 내의 하나 이상의 레지스터들을 통해, GPU 상에서 실행하는 태스크와 호스트 디바이스 상에서 실행하는 프로세스 사이에 하나 이상의 메시지들을 전달하도록 하는 명령들을 더 포함한다.

[0019] 또다른 예에서, 본 개시물은 호스트 디바이스의 하나 이상의 프로세서들 상에서 실행하는 메모리 버퍼 인터페이스에 의해, 호스트 디바이스에 의해 그리고 그래픽 프로세싱 유닛 (GPU) 에 의해 액세스가능한 공유 메모리 공간에 대해 즉시 모드가 인에이블되어야 하는지 여부를 특정하는 정보를 포함하는 명령을 수신하는 단계를 포함하는 방법을 설명한다. 이 방법은, 메모리 버퍼 인터페이스에 의해, 즉시 모드가 인에이블되어야 하는지 여부를 특정하는 정보에 기초하여, 공유 메모리 공간에 대해 즉시 모드를 선택적으로 인에이블하는 단계를 더 포함한다.

[0020] 또다른 예에서, 본 개시물은 하나 이상의 프로세서들을 포함하는 호스트 디바이스를 설명한다. 이 디바이스는 하나 이상의 프로세서들 상에서 실행하며, 공유 메모리 공간에 대해 즉시 모드가 인에이블되어야 하는지 여부를 특정하는 정보를 포함하는 명령을 수신하고, 즉시 모드가 인에이블되어야 하는지 여부를 특정하는 정보에 기초하여, 공유 메모리 공간에 대해 즉시 모드를 선택적으로 인에이블하도록 구성된 메모리 버퍼 인터페이스를 더 포함하며, 공유 메모리 공간은 호스트 디바이스에 의해 그리고 그래픽 프로세싱 유닛 (GPU) 에 의해 액세스가능하다.

[0021] 또다른 예에서, 본 개시물은 호스트 디바이스에 의해 그리고 그래픽 프로세싱 유닛 (GPU) 에 의해 액세스가능한 공유 메모리 공간에 대해 즉시 모드가 인에이블되어야 하는지 여부를 특정하는 정보를 포함하는 명령을 수신하는 수단을 포함하는 장치를 설명한다. 이 장치는 즉시 모드가 인에이블되어야 하는지 여부를 특정하는 정보에 기초하여, 공유 메모리 공간에 대해 즉시 모드를 선택적으로 인에이블하는 수단을 더 포함한다.

[0022] 또 다른 예에서, 본 개시물은, 하나 이상의 프로세서들로 하여금, 호스트 디바이스에 의해 그리고 그래픽 프로세싱 유닛 (GPU) 에 의해 액세스가능한 공유 메모리 공간에 대해 즉시 모드가 인에이블되어야 하는지 여부를 특정하는 정보를 포함하는 명령을 수신하도록 하는 명령들을 포함하는 컴퓨터 판독가능 매체를 설명한다. 컴퓨터 판독가능 저장 매체는, 하나 이상의 프로세서들로 하여금, 즉시 모드가 인에이블되어야 하는지 여부를 특정하는 정보에 기초하여, 공유 메모리 공간에 대해 즉시 모드를 선택적으로 인에이블하도록 하는 명령들을 더 포함한다.

[0023] 또다른 예에서, 본 개시물은 메모리와 연관된 그래픽 프로세싱 유닛 (GPU) 캐시를 포함하는 GPU 를 설명한다. 이 디바이스는 메모리 공간에 대해 판독 동작들 및 기록 동작들 중 적어도 하나를 실행하기 위해 캐싱 서비스들이 이용되어야 하는지 여부를 특정하는 정보를 수신하는 것에 응답하여, 메모리의 메모리 공간에 대해 판독 동작 및 기록 동작 중 적어도 하나를 실행하기 위해 GPU 캐시의 캐싱 서비스들을 선택적으로 이용하도록 구성된 하나 이상의 프로세싱 모듈들을 더 포함한다.

[0024] 또다른 예에서, 본 개시물은 메모리 공간에 대해 판독 동작들 및 기록 동작들 중 적어도 하나를 실행하기 위해 캐싱 서비스들이 이용되어야 하는지 여부를 특정하는 정보를 수신하는 것에 응답하여, 메모리의 메모리 공간에 대해 판독 동작 및 기록 동작 중 적어도 하나를 실행하기 위해 메모리와 연관된 그래픽 프로세싱 유닛 (GPU) 캐시의 캐싱 서비스들을 선택적으로 이용하는 단계를 포함하는 방법을 설명한다.

[0025] 또다른 예에서, 본 개시물은 메모리와 연관되는 GPU 캐시를 포함하는 장치를 설명한다. 이 장치는 메모리 공간에 대해 판독 동작들 및 기록 동작들 중 적어도 하나를 실행하기 위해 캐싱 서비스들이 이용되어야 하는지 여부를 특정하는 정보를 수신하는 것에 응답하여, 메모리의 메모리 공간에 대해 판독 동작 및 기록 동작 중 적어도 하나를 실행하기 위해 GPU 캐시의 캐싱 서비스들을 선택적으로 이용하는 수단을 더 포함한다.

[0026] 또다른 예에서, 본 개시물은, 하나 이상의 프로세서들로 하여금, 메모리 공간에 대해 판독 동작들 및 기록 동작들 중 적어도 하나를 실행하기 위해 캐싱 서비스들이 이용되어야 하는지 여부를 특정하는 정보를 수신하는 것에 응답하여, 메모리의 메모리 공간에 대해 판독 동작 및 기록 동작 중 적어도 하나를 실행하기 위해 메모리와 연관된 그래픽 프로세싱 유닛 (GPU) 캐시의 캐싱 서비스들을 선택적으로 이용하도록 하는 명령들을 포함하는 컴퓨터 판독가능 매체를 설명한다.

도면의 간단한 설명

[0027] 도 1 은 본 개시물에 따른, 메시지 전달 기법들을 수행하는데 사용될 수도 있는 예시적인 컴퓨팅 시스템을 도시하는 블록도이다.

도 2 는 본 개시물에 따른, 도 1 의 컴퓨팅 시스템에 사용될 수도 있는 예시적인 GPU 를 도시하는 블록도이다.

도 3 은 본 개시물에 따른, 다중-프로세서 플랫폼 환경에서 메시지 전달을 위한 예시적인 기법을 도시하는 흐름도이다.

도 4 는 본 개시물에 따른, 호스트 디바이스 상에서 실행하는 프로세스에 의해 이슈된 전송 명령을 실행하는 예시적인 기법을 도시하는 흐름도이다.

도 5 및 도 6 은 본 개시물에 따른 도 4 에 도시된 기법의 부분들을 구현하는데 사용될 수도 있는 예시적인 기법들을 도시하는 흐름도들이다.

도 7 은 본 개시물에 따른, 컴퓨팅 디바이스, 예컨대, GPU 에서 수신된 메시지를 프로세싱하는 예시적인 기법을 도시하는 흐름도이다.

도 8 은 본 개시물에 따른, 컴퓨팅 디바이스, 예컨대, GPU 상에서 실행하는 태스크에 의해 이슈된 수신 명령을 실행하는 예시적인 기법을 도시하는 흐름도이다.

도 9 및 도 10 은 본 개시물에 따른, 도 8 에 도시된 기법의 부분들을 구현하는데 사용될 수도 있는 예시적인 기법들을 도시하는 흐름도들이다.

도 11 은 본 개시물에 따른, 컴퓨팅 디바이스, 예컨대, GPU 상에서 실행하는 프로세스에 의해 이슈된 전송 명령을 실행하는 예시적인 기법을 도시하는 흐름도이다.

도 12 및 도 13 은 본 개시물에 따른, 도 11 에 도시된 기법의 부분들을 구현하는데 사용될 수도 있는 예시적인 기법들을 도시하는 흐름도들이다.

도 14 는 본 개시물에 따른, 호스트 디바이스 상에서 실행하는 프로세스에 의해 이슈된 레지스터 콜백 루틴 명령 (register callback routine instruction) 을 실행하는 예시적인 기법을 도시하는 흐름도이다.

도 15 는 본 개시물에 따른, 컴퓨팅 디바이스로부터 수신된 인터럽트를 프로세싱하는 예시적인 기법을 도시하는 흐름도이다.

도 16 및 도 17 은 본 개시물에 따른, 도 15 에 도시된 기법의 부분들을 구현하는데 사용될 수도 있는 예시적인 기법들을 도시하는 흐름도들이다.

도 18 은 본 개시물에 따른, 호스트 디바이스 상에서 실행하는 프로세스에 의해 이슈된 판독 명령을 실행하는 예시적인 기법을 도시하는 흐름도이다.

도 19 는 본 개시물에 따른, 도 18 에 도시된 기법의 부분들을 구현하는데 사용될 수도 있는 예시적인 기법을 도시하는 흐름도이다.

도 20 은 본 개시물에 따른, 즉시 메모리 오브젝트들의 사용을 용이하게 할 수도 있는 예시적인 컴퓨팅 시스템을 도시하는 블록도이다.

도 21 은 본 개시물에 따른, 호스트 디바이스 상에서 실행하는 프로세스에 의해 이슈된 메모리 오브젝트 생성 명령을 실행하는 예시적인 기법을 도시하는 흐름도이다.

도 22 는 본 개시물에 따른, 호스트 디바이스 상에서 실행하는 프로세스에 의해 이슈된 메모리 오브젝트 생성 명령을 실행하는 또다른 예시적인 기법을 도시하는 흐름도이다.

도 23 내지 도 26 은 본 개시물에 따른, 캐시 모드 및 즉시 모드 명령들을 프로세싱하는 예시적인 기법들을 도시하는 흐름도들이다.

도 27 은 본 개시물에 따른, 도 20 의 컴퓨팅 시스템에서 사용될 수도 있는 예시적인 GPU 를 도시하는 블록도이다.

도 28 은 본 개시물에 따른, 캐시 모드 및 즉시 모드 명령들을 프로세싱하는 예시적인 기법을 도시하는 흐름도이다.

도 29 는 본 개시물에 따른, 호스트 디바이스 상에서 실행하는 프로세스에 의해 이슈된 메모리 오브젝트 생성 명령을 실행하는 또다른 예시적인 기법을 도시하는 흐름도이다.

도 30 은 본 개시물에 따른, GPU 가 제 1 컴파일 (compilation) 기법에 따라 컴파일된 명령들의 시퀀스를 프로세싱할 수도 있는 방법을 나타내는 흐름도이다.

도 31 은 본 개시물에 따른, 태스크에 대한 소스 코드를 컴파일하는 예시적인 기법을 도시하는 흐름도이다.

도 32 는 본 개시물에 따른, 캐싱 서비스들을 선택적으로 이용하기 위해 GPU 에 의해 사용될 수도 있는 예시적인 기법을 도시하는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0028] 본 개시물은 다중-프로세서 컴퓨팅 플랫폼 내에서 사용될 수도 있는 통신 기법들을 설명한다. 이 기법들은 일부 예들에서, 커맨드 큐 (command queue) 들을 이용하여 태스크들을 개시하는 다중-프로세서 컴퓨팅 플랫폼 내에서 메시지 전달을 지원하는데 사용될 수도 있는 소프트웨어 인터페이스들을 제공할 수도 있다. 이 기법들은 추가적인 예들에서, 다중-프로세서 컴퓨팅 플랫폼 내에서 공유 메모리 프로세서간 통신을 위해 사용될 수도 있는 소프트웨어 인터페이스들을 제공할 수도 있다. 추가적인 예들에서, 이 기법들은 GPU 와 호스트 CPU 사이에 메시지 전달 및/또는 공유 메모리 통신을 지원하는 하드웨어를 포함하는 그래픽 프로세싱 유닛 (GPU) 을 제공할 수도 있다.

[0029] 최근, 실시간 3D 그래픽들의 프로세싱용으로 원래 설계된 프로세서들, 예컨대, 그래픽 프로세싱 유닛 (GPU) 이, 범용 컴퓨팅 태스크들 (GPGPU) 을 수행하기 위해 일반화되었다. GPGPU들의 가치는 예컨대, OpenCL™ (Open Computing Language) 표준과 같은 산업-전체적인 표준들의 채택에 의해, 부분적으로 입증되었다. OpenCL 은 다중-프로세서 컴퓨팅 플랫폼 상에서 태스크-레벨 병렬성 및/또는 데이터-레벨 병렬성을 갖는 프로그램들을 실행하는데 사용될 수도 있는, 크로스-플랫폼, 크로스-벤더, 이종 컴퓨팅 플랫폼, 병렬 프로그래밍 API 의 일 예이다. API 는 GPU의 데이터 흐름 및 제어 경로들을 넌-그래픽 특정의 방식으로 일반화함으로써, GPU 상에서

더 일반화된 데이터 프로세싱을 허용하도록 특별히 설계된다. 이 접근법의 하나의 제한은 호스트 CPU 와 컴퓨팅 디바이스들, 예컨대, GPU 사이의 데이터 통신의 코어스 그래뉼래러티 (coarse granularity) 이다.

[0030] 예를 들어, OpenCL API 는 호스트 디바이스와 하나 이상의 컴퓨팅 디바이스들 사이에 통신의 태스크-레벨 그래뉼래러티 (task-level granularity) 를 지원하는 커맨드 큐 인터페이스를 제공한다. 각각의 커맨드 큐는 일반적으로 특정의 컴퓨팅 디바이스에 의해 실행될 커맨드들을 유지한다. 호스트 디바이스 상에서 실행하는 호스트 프로세스는 메모리 전송을 수행하도록 호스트 디바이스에 명령하는 커맨드를 커맨드 큐에 배치함으로써, 호스트 메모리 공간과 디바이스 메모리 공간 사이에 데이터를 전송할 수도 있다. 이와 유사하게, 호스트 프로세스는, 태스크로 하여금, 컴퓨팅 디바이스 상에서 태스크를 실행하도록 호스트 디바이스에 명령하는 커맨드를 커맨드 큐에 배치함으로써, 컴퓨팅 디바이스 상에서 실행을 시작하도록 할 수도 있다.

[0031] 커맨드 큐 인터페이스는 커맨드들의 순차 (in-order) 실행 또는 커맨드들의 비순차 (out-of-order) 실행 중 어느 하나를 제공하도록 구성될 수도 있다. 커맨드 큐 인터페이스가 커맨드들의 순차 실행을 제공하도록 구성될 때, 커맨드 큐 인터페이스는, 커맨드들이 커맨드 큐에 배치된 순서로 커맨드들이 실행되고 이전 커맨드가 실행을 완료할 때까지 후속 커맨드의 실행이 시작되지 않는 것을 보장한다. 따라서, 호스트 프로세스가 태스크를 실행하기 위해 커맨드를 커맨드 큐에 배치할 때, 커맨드 큐는 후속하여 커맨드 큐에 배치될 수도 있는 임의의 추가적인 커맨드들을 실행하기 전에 그 태스크가 실행을 완료하기를 대기한다.

[0032] 호스트 CPU 및 GPU 를 수반하는 간단한 설정 및 순차 커맨드 큐에서, 호스트 CPU 와 GPU 사이의 통신 방식은 다음 동작들을 포함할 수도 있다: (1) 호스트 CPU 가 데이터를 준비해서 GPU-액세스가능 메모리에 배치하는 동작; (2) 호스트 CPU 가 태스크를 실행하도록 GPU 에 커맨드하는 동작; (3) 호스트 CPU 가 GPU 가 태스크의 실행을 종료하기를 대기하는 동작; 및 (4) 호스트 CPU 가 데이터를 GPU-액세스가능 메모리로부터 호스트 메모리로 복사하는 동작. 이러한 구성에서는, GPU 상에서의 태스크의 실행에 요구되는 모든 데이터가 태스크의 실행의 시작 전에 GPU-액세스가능 메모리로 전송되며, GPU 상에서 실행하는 태스크가 실행을 완료한 이후까지, GPU 상에서 실행하는 태스크에 의해 생성된 데이터는 호스트 CPU 에 이용가능하지 않다. 호스트 CPU 와 GPU 사이의 데이터 공유에서의 이 코어스니스 (coarseness) 는 예를 들어, 호스트 디바이스 상에서 실행하는 프로세스와 GPU 상에서 실행하는 태스크 사이에 프로세스간 메시지들을 전달하는 것과 같은, 병렬-기반 애플리케이션들에 대한 많은 유용한 동작들의 효과적인 구현들을 방해할 수도 있다. 이러한 메시지들은 예를 들어, GPU 상에서 실행하는 태스크가 호스트 CPU 상에서 원격 프로시저 호출 (RPC; Remote Procedure Call) 을 실행하는 능력을 갖는 것을 허용하는데 유용할 수도 있다.

[0033] 커맨드 큐 인터페이스가 커맨드들의 비순차 실행을 제공하도록 구성될 때, 호스트 프로세스는 특정 태스크의 실행 동안, 특정 커맨드의 실행이 일어날 때 제어가능하지 않다. 이로써, 커맨드 큐에 대한 비순차 실행 모드는 또한 호스트 디바이스 상에서 실행하는 프로세스와 GPU 상에서 실행하는 태스크 사이에 프로세스간 메시지 전달의 구현을 사실상 허용하지 않는다.

[0034] OpenCL 에 사용되는 메모리 모델에 관하여, API 는 호스트 CPU 와 GPU 사이에 데이터를 공유하거나 또는 다수의 OpenCL 계산 디바이스들 사이에 데이터를 공유하기 위해 사용될 수도 있는, 소위 글로벌 CL 버퍼들 및 글로벌 CL 이미지들을 정의한다. 그러나, CPU 및 GPU 는 동시에 버퍼로부터 판독하거나 또는 버퍼에 기록할 수 없다. 통상, CPU 는 소스 데이터를 포함하는 하나 이상의 버퍼들을 준비하고 그 버퍼들을 프로세싱을 위해 GPU 로 전달한다. GPU 는 이들 버퍼들을 변경하거나 또는 결과들을 GPU 데이터 변경들을 수신할 목적으로 CPU 상에서 실행하는 소프트웨어에 의해 선험적으로 또한 할당된 다른 버퍼들에 배치하거나 한다.

[0035] 현재 OpenCL 에서의 메모리 오브젝트들은 컴퓨팅 디바이스에 의해 사용되는 버퍼 데이터를 저장하는데 호스트 메모리 공간의 영역이 사용되는 것을 허용하지만, 이 사양은 컴퓨팅 디바이스들이 태스크의 보다 효율적인 실행을 위해 이 데이터를 캐시하는 것을 허용한다. 호스트 디바이스는 일반적으로 버퍼 데이터를 캐시하는데 사용되는 컴퓨팅 디바이스 캐시를 즉시 무효화할 수 없다. 따라서, 호스트 디바이스가 호스트 메모리 공간에 저장된 소정의 메모리 버퍼 데이터를 중복 기록하였더라도, 컴퓨팅 디바이스에 그 변경된 데이터에의 즉시 액세스를 제공하기 위해 컴퓨팅 디바이스 내의 캐시가 업데이트된다는 보장도 없다. 게다가, 컴퓨팅 디바이스에 의해 수행된 계산들의 결과들이 컴퓨팅 디바이스 캐시에 저장될 수도 있기 때문에, 호스트 디바이스 상에서 실행하는 호스트 프로세스는 이러한 데이터가 컴퓨팅 디바이스 캐시에 저장된 더 새로운 데이터로 인해 무효일 수도 있기 때문에, 버퍼로부터 임의의 부분 결과들을 판독할 수 없다. 따라서, OpenCL 에서의 메모리 관리 모델은 용이하게 공유 메모리를 통한 인-플라이트 (in-flight) 데이터 공유를 허용하지 않는다.

[0036] 본 개시물에서 설명하는 기법들은 일부 예들에서, OpenCL API 의 위에서 언급된 제한들 중 하나 이상을 극복하

는데 사용될 수도 있다. 예를 들어, 본 개시물의 기법들은 태스크-레벨 그래뉼래러티 커맨드 큐들을 이용하여 태스크들을 개시하는 다중-프로세서 컴퓨팅 플랫폼 내에서 프로세스간 메시지 전달을 지원하는데 사용될 수 있는 소프트웨어 인터페이스들을 제공할 수도 있다. 또 다른 예로서, 본 개시물의 기법들은 다중-프로세서 컴퓨팅 플랫폼 내에서 공유 메모리를 통한 인-플라이트 데이터 공유를 지원하는데 사용될 수 있는 소프트웨어 인터페이스들을 제공할 수도 있다.

[0037] 일부 예들에서, 본 개시물의 기법들은 소프트웨어-레벨 메시지 전달을 용이하게 하는 GPU 하드웨어 아키텍처를 제공할 수도 있다. 예를 들어, 본 개시물의 기법들은 소프트웨어-레벨 메시지 전달 명령들의 실행을 지원하도록 구성되는 GPU 하드웨어 아키텍처를 제공할 수도 있다. 추가적인 예들에서, 본 개시물의 기법들은 GPU 와 호스트 CPU 사이에 공유 메모리 통신을 용이하게 하는 GPU 하드웨어 아키텍처를 제공할 수도 있다. 예를 들어, 본 개시물의 기법들은 공유 메모리 공간에 대해 캐싱 서비스들을 선택적으로 인에이블 및 디스에이블하고/하거나 공유 메모리 공간에 대해 캐시-코히런시 메커니즘 (cache-coherency mechanism) 을 선택적으로 인에이블 및 디스에이블하도록 구성되는 GPU 하드웨어 아키텍처를 제공할 수도 있다.

[0038] 본 개시물의 제 1 양태에 따르면, 컴퓨팅 디바이스에 의한 태스크의 실행 동안 호스트 디바이스와 하나 이상의 컴퓨팅 디바이스들 사이에 메시지 전달 명령들의 실행을 용이하게 하는 메시지 전달 인터페이스가 제공된다. 메시지 전달은 통신중인 프로세스들이 각각 메시지를 성공적으로 전달하기 위해 상보적인 세트들의 동작들을 수행하는, 프로세스간, 및 잠재적으로는, 디바이스간 통신의 형태를 나타낼 수도 있다. 예를 들어, 메시지 전달 프로토콜에 따라 통신하는 프로세스들 각각이 전송 동작 및 수신 동작을 구현할 수도 있다. 본 개시물에서의 메시지 전달 기법들은 CPU 및 컴퓨팅 디바이스, 예컨대, GPU 가, 컴퓨팅 디바이스 상에서의 태스크의 실행 동안 서로 사이에 메시지들을 전달하는 것을 허용할 수도 있다. 이 방식으로, 태스크-레벨 그래뉼래러티 커맨드 큐 통신 방식을 구현하는 다중-프로세서 컴퓨팅 플랫폼은 프로세스간 및/또는 디바이스간 통신을 용이하게 하는 것을 가능하게 할 수도 있다.

[0039] 일부 예들에서, 본 개시물에서 설명하는 메시지 전달 기법들은, 이 기법들이 호스트 디바이스와 예컨대, GPU 와 같은 컴퓨팅 디바이스 사이의 통신을 위해 OpenCL 에 통상 사용되는 커맨드 큐 인터페이스와는 다른 인터페이스를 사용할 수도 있기 때문에, "대역외 시그널링" 기법들로 지칭될 수도 있다. 즉, 본 개시물의 기법들은 OpenCL 내에 포함된 대역내 커맨드 큐 인터페이스와는 논리적으로 별개인 새로운 대역외 통신 인터페이스를 포함할 수도 있다. 대역외 통신 인터페이스는 커맨드 큐 인터페이스가 영향을 받는 동일한 태스크-레벨 그래뉼래러티에 영향을 받지 않으며, 이에 따라, 커맨드 큐의 태스크-레벨 그래뉼래러티에 대해 위에서 설명한 하나 이상의 제한들에 대한 솔루션을 제공할 수도 있다.

[0040] 본 개시물의 기법들에 따라 CPU 와 GPU 사이에 전송되는 메시지들은 임의의 유형의 메시지일 수도 있다. 상이한 유형들의 메시지들의 예들은 신호들, 메모리 할당 요청들, 메모리 할당해제 요청들, 통지 메시지들, 동기화 메시지들, 원격 프로시저 호출 메시지들 (예컨대, 원격 프로시저 호출 (RPC) 의 부분인 메시지들), 데이터 패킷들, 레포팅 메시지들, 어서션 (assertion) 메커니즘 메시지들, 및 로깅 (logging) 메시지들을 포함한다.

[0041] 현재의 OpenCL 패러다임에서, 호스트 CPU 로부터 GPU 로의 모든 요청들은 OpenCL 커맨드 큐들에 큐 업 (queue up) 되며 그후 GPU 로 전송된다. 특히, 애플리케이션은 다수의 커널 실행들 및 버퍼 동작들을 커맨드 큐에 인큐할 것이다. 한편, 첫번째 인큐된 태스크, 예컨대, 커널 실행이, 예를 들어, CPU 로부터 추가적인 메모리 할당을 요청할 필요가 있다면, 다수의 이슈들이 발생한다. 첫째, GPU 가 어떻게 메모리 할당을 발생시킬 필요가 있다는 것을 실행중인 커널 내로부터 CPU 에 통지하는가? 둘째, CPU 가 어떻게 메모리 할당 및 새로 할당된 메모리 블록의 어드레스의 완료에 관해 GPU 에 통지하는가? 그러나 본 개시물의 메시지 전달 인터페이스 기법들은, CPU 와 GPU 사이에 위에서 설명한 통지들 및 정보를 포함하는 하나 이상의 메시지들이 전달되는 것을 허용함으로써, 이들 이슈들을 해결가능할 수도 있다.

[0042] 본 개시물의 대역외 시그널링 기법들은 일부 예들에서, 호스트 CPU 와 하나 이상의 컴퓨팅 디바이스들, 예컨대, OpenCL 계산 디바이스들 사이에 시그널링을 구현하는데 사용될 수도 있다. 대역외 시그널링은 빠른 대역외 통지들을, 예컨대, 푸시 또는 풀 메커니즘 (push or pull mechanism) 을 이용하여 제공할 수도 있다. 일부 예들에서, 대역외 시그널링 기법들은 상대적으로 적은 양들의 데이터를 운반할 수도 있다.

[0043] 본 개시물의 제 2 양태에 따르면, 메시지들을 GPU 와는 다른 프로세서들 상에서 실행하는 프로세스들로 전송하고 메시지들을 GPU 와는 다른 프로세서들 상에서 실행하는 프로세스들로부터 수신하는 것이 가능한 GPU 가 제공된다. 예를 들어, GPU 는 메시지들을 전송하고 수신하는 하나 이상의 동작들을 구현하도록 구성된 하드웨어를 포함할 수도 있다. 일부 예들에서, 본 개시물에 따라서 설계된 GPU 는 메시지 전달 프로토콜과 연관된

상태 및 데이터 정보를 저장하도록 구성된 하나 이상의 호스트-액세스가능 레지스터들을 포함할 수도 있다. 하나 이상의 레지스터들은 GPU 상에서 실행하는 태스크와 GPU 와는 다른 디바이스 상에서 실행하는 프로세스 사이에 메시지 전달을 용이하게 하도록 구성될 수도 있다. 추가적인 예들에서, GPU 의 ALU 프로세싱 블록 (예컨대, 프로그래밍가능 셰이더 유닛) 은 호스트-액세스가능 레지스터들을 통해서 메시지들을 전송하고 수신하기 위해 호스트-액세스가능 레지스터들에 통신적으로 커플링될 수도 있다. GPU 는 또한 동기적 및/또는 비동기적 메시지 전달 기법들을 구현하는 여러 폴링 및/또는 인터럽트 메커니즘들을 포함하도록 설계될 수도 있다.

[0044] 본 개시물의 제 3 양태에 따르면, 즉시 메모리 오브젝트들 (immediate memory objects) 이 생성되는 것을 허용하는 메모리 버퍼 인터페이스가 제공된다. 즉시 메모리 오브젝트들은, 태스크가 컴퓨팅 디바이스 상에서 실행중인 동안 호스트 디바이스 상에서 실행하는 프로세스와 컴퓨팅 디바이스 상에서 실행하는 태스크 사이에 데이터를 공유하기 위하여, 캐시불가능한 공유 메모리 공간 및/또는 캐시-코히런트 공유 메모리 공간을 구현하는데 사용될 수도 있다. 공유 메모리 공간은 컴퓨팅 디바이스에 의한 태스크의 실행 동안 호스트 디바이스와 컴퓨팅 디바이스, 예컨대, GPU 양자에 의해 액세스가능한 메모리 공간일 수도 있다. 캐시불가능한 공유 메모리 공간은, 본원에서 사용한 바와 같이, 호스트 디바이스 및 컴퓨팅 디바이스 중 하나 또는 양자에서의 하나 이상의 대응하는 캐시들이 메모리 공간에 대해 디스에이블되는 공유 메모리 공간을 지칭할 수도 있다. 캐시-코히런트 공유 메모리 공간은, 본원에서 사용한 바와 같이, 공유 메모리 캐시 코히런시 기법들이 호스트 디바이스 및 컴퓨팅 디바이스 중 하나 또는 양자에서의 하나 이상의 대응하는 캐시들 내에서 캐시 코히런시를 유지하는데 사용되는 공유 메모리 공간을 지칭할 수도 있다. 캐시불가능한 공유 메모리 공간 및 캐시-코히런트 공유 메모리 공간은 언제라도 데이터 공유를 허용할 수도 있다. 즉시 메모리 오브젝트들은 일부 예들에서, 호스트 디바이스 및 컴퓨팅 디바이스에 대해, 캐시불가능한 휘발성 공유 메모리로서 및/또는 캐시-코히런트 휘발성 공유 메모리로서 구현될 수도 있다.

[0045] 일부 예들에서, 본 개시물의 즉시 메모리 오브젝트들은 메모리 오브젝트 메모리 관리 방식을 포함하는, 크로스-플랫폼, 크로스-벤더, 이종 컴퓨팅 플랫폼, 병렬 프로그래밍 API 내에 통합될 수도 있다. 예를 들어, 즉시 메모리 오브젝트들은 OpenCL 메모리 오브젝트들, 예컨대, OpenCL 버퍼 오브젝트들 또는 OpenCL 이미지 오브젝트들의 추가적인 속성으로서 OpenCL 에 통합될 수도 있다. 이러한 예들에서, 즉시 메모리 오브젝트들은 기능 호출에 의해 생성된 결과의 메모리 오브젝트가 표준 모드 메모리 오브젝트이어야 하는지 또는 즉시 모드 메모리 오브젝트이어야 하는지 여부를 특정하는 파라미터 또는 플래그를 포함하도록 메모리 오브젝트 생성 기능들을 변경함으로써 생성될 수도 있다. 이 방식으로, 본 개시물의 기법들은 OpenCL 과 같은, 메모리 오브젝트 메모리 관리 방식들을 포함하는 API들을 구현하는 멀티-프로세서 컴퓨팅 시스템들이 캐시 코히런시 이슈들에 영향을 받지 않는 공유 메모리 공간을 통한 인-플라이트 데이터 공유를 구현하는 것을 허용할 수도 있다.

[0046] 추가적인 예들에서, 본 개시물의 즉시 메모리 오브젝트들이 호스트 CPU 와 OpenCL 계산 디바이스 사이 또는 상이한 OpenCL 계산 디바이스들 사이의 인-플라이트 데이터 공유를 위해 사용될 수도 있다. 추가적인 예들에서, 즉시 메모리 오브젝트들은 내부 동기화 마커들을 포함할 수도 있다. 추가적인 예들에서, 즉시 메모리 오브젝트들이 동기화를 위해 대역외 신호들과 함께 사용될 수도 있다.

[0047] 본 개시물의 제 4 양태에 따르면, 캐시불가능한 공유 메모리 공간을 제공하기 위하여, 특정 메모리 어드레스 공간들에 대해 선택적으로 디스에이블될 수도 있는 공유 메모리 공간에 대응하는 캐시를 포함하는 GPU 가 제공된다. 예를 들어, GPU 는 공유 메모리 공간에 대해 판독 동작들 및/또는 기록 동작들을 실행하기 위해 캐싱 서비스들이 이용되어야 하는지 여부를 특정하는 정보를 수신하는 것에 응답하여, 공유 메모리 공간과 연관된 캐시에 의해 제공된 캐싱 서비스들을 선택적으로 인에이블 및 디스에이블할 수도 있다. 일부 예들에서, 공유 메모리 공간에 대해 판독 동작들 및/또는 기록 동작들을 실행하기 위해 캐싱 서비스들이 이용되어야 하는지 여부를 특정하는 정보는 특정 명령을 실행하기 위해 캐시 모드 또는 즉시 모드가 사용되어야 하는지 여부를 특정하는 캐시 모드 명령 또는 즉시 모드 명령일 수도 있다. 추가적인 예들에서, 공유 메모리 공간에 대해 판독 동작들 및/또는 기록 동작들을 실행하기 위해 캐싱 서비스들이 이용되어야 하는지 여부를 특정하는 정보는 메모리 오브젝트에 대해 즉시 모드가 인에이블되는지 여부를 특정하는 즉시 모드 메모리 오브젝트 속성일 수도 있다.

[0048] 추가적인 예들에서, 본 개시물의 기법들은 캐시-코히런트 공유 메모리 공간을 제공하기 위해 선택적으로 인에이블될 수도 있는 캐시 코히런시 모드를 포함하는 GPU 를 제공할 수도 있다. 일부 예들에서, GPU 는 호스트 디바이스로부터 수신된 하나 이상의 명령들에 기초하여, 그 공유 메모리 공간에 대응하는 캐시의 부분에 대해 캐시 코히런시 모드를 선택적으로 인에이블할 수도 있다. 호스트 디바이스는 호스트 프로세스에 의해 특정

된 즉시 모드 파라미터에 기초하여 호스트 디바이스에 의한 공유 메모리 공간의 할당 시 그 공유 메모리 공간에 대응하는 캐시의 부분에 대해 공유 메모리 공간 캐시 코히런시 모드를 선택적으로 인에이블하기 위해 하나 이상의 명령들을 GPU 에 이슈할 수도 있다.

[0049] 본 개시물의 대역외 시그널링 및 즉시 버퍼링 (buffering) 기법들은, OpenCL 커맨드 큐 인터페이스를 단독으로 이용하여 달리 획득가능한 것에 비해, 호스트 CPU 와 GPU 사이에 또는 2 개의 OpenCL 계산 디바이스들 사이에 보다 파인 그레인드 (fine grained) 태스크 커플링을 제공할 수도 있다. 본 개시물의 기법들은 병렬 및/또는 멀티-쓰레드된 프로그램들의 효율적인 실행을 돕기 위하여 다중-프로세서 컴퓨팅 플랫폼이 다양한 동작들을 수행하는 것을 허용할 수도 있다. 예를 들어, 본 개시물의 기법들은 GPU 상에서 실행하는 태스크가 RPC 를 착수하는 것을 허용할 수도 있다. 또 다른 예로서, 본 개시물의 기법들은 GPU 상에서 실행하는 태스크가 CPU 을 통해, 또다른 GPU 태스크를 착수하는 것을 허용할 수도 있다. 추가 예로서, 본 개시물의 기법들은 GPU 상에서 실행하는 태스크가 예를 들어, 메모리 할당 및/또는 메모리 할당해제 요청들과 같은 리소스 관리 요청들을 CPU 및/또는 그 CPU 상에서 실행하는 드라이버에 이슈하는 것을 허용할 수도 있다. 또한, 또다른 예로서, 본 개시물의 기법들은 GPU 상에서 실행하는 태스크가 예를 들어, 어서션 메커니즘, 중간 레포팅, 및/또는 진단 로깅의 구현과 같은, 상태 체크들 및 CPU 로의 일반 메시지 전달을 수행하는 것을 허용할 수도 있다.

[0050] 도 1 은 본 개시물에 따른, 예시적인 컴퓨팅 시스템 (10) 을 도시하는 블록도이다. 컴퓨팅 시스템 (10) 은 다수의 프로세싱 디바이스들 상에서 하나 이상의 소프트웨어 애플리케이션들을 프로세싱하도록 구성된다. 일부 예들에서, 하나 이상의 소프트웨어 애플리케이션들은 호스트 프로세스를 포함할 수도 있으며, 컴퓨팅 시스템 (10) 은 호스트 프로세스를 실행하고 컴퓨팅 시스템 (10) 내의 다른 컴퓨팅 디바이스들 상에 호스트 프로세스에 의해 개시된 하나 이상의 태스크들의 실행을 분산하도록 구성될 수도 있다. 추가적인 예들에서, 컴퓨팅 시스템 (10) 에 의해 실행된 호스트 프로세스 및/또는 태스크들은 병렬 프로그래밍 모델에 따라 프로그래밍될 수도 있다. 예를 들어, 애플리케이션들은 근본적인 하드웨어 시스템들의 태스크-레벨 병렬성 및/또는 데이터-레벨 병렬성을 레버리지하도록 설계되는 명령들을 포함할 수도 있다.

[0051] 컴퓨팅 시스템 (10) 은 개인용 컴퓨터, 데스크탑 컴퓨터, 랩탑 컴퓨터, 컴퓨터 워크스테이션, 비디오 게임 플랫폼 또는 콘솔, 예컨대, 셀룰러 또는 위성 전화기와 같은 모바일 전화기, 유선 전화기, 인터넷 전화기, 휴대형 비디오 게임 디바이스 또는 개인 휴대정보 단말기 (PDA) 와 같은 핸드헬드 디바이스, 개인 뮤직 플레이어, 비디오 플레이어, 디스플레이 디바이스 또는 텔레비전과 같은 디지털 미디어 플레이어, 텔레비전 셋톱 박스, 서버, 중간 네트워크 디바이스, 메인프레임 컴퓨터 또는 정보를 프로세싱하는 임의의 다른 유형의 디바이스일 수도 있다.

[0052] 컴퓨팅 시스템 (10) 는 호스트 디바이스 (12), 그래픽 프로세싱 유닛 (GPU; 14), 메모리 (16) 및 상호접속 네트워크 (18) 를 포함한다. 호스트 디바이스 (12) 는 호스트 프로세스의 실행을 위한 플랫폼 및 다중-프로세서 컴퓨팅 플랫폼 API 를 위한 런타임 모듈을 제공하도록 구성된다. 통상, 호스트 디바이스 (12) 는 범용 CPU 이지만, 호스트 디바이스 (12) 는 프로그램들을 실행가능한 임의의 유형의 디바이스일 수도 있다. 호스트 디바이스 (12) 는 상호접속 네트워크 (18) 를 통해 GPU (14) 및 메모리 (16) 에 통신적으로 커플링된다. 호스트 디바이스 (12) 는 호스트 프로세스 (20) 및 런타임 모듈 (22) 을 포함하며, 이들 각각은 하나 이상의 프로그래밍가능 프로세서들의 임의의 조합 상에서 실행할 수도 있다.

[0053] 호스트 프로세스 (20) 는 컴퓨팅 시스템 (10) 의 컴퓨팅 시스템 플랫폼 상에서의 실행을 위한 소프트웨어 프로그램을 형성하는 명령들의 세트를 포함한다. 소프트웨어 프로그램은 최종-사용자에 대한 하나 이상의 특정의 태스크들을 수행하도록 설계될 수도 있다. 이러한 태스크들은 일부 예들에서, 컴퓨팅 시스템 (10) 에 의해 제공된 다수의 프로세싱 디바이스들 및 병렬 아키텍처들을 활용할 수 있는 계산 집약적인 알고리즘들을 수반할 수도 있다.

[0054] 런타임 모듈 (22) 은 호스트 프로세스 (20) 에 포함된 명령들 중 하나 이상을 서비스하도록 구성된 하나 이상의 인터페이스들을 구현하는 호스트 디바이스 (12) 상에서 실행하는 소프트웨어 모듈일 수도 있다. 런타임 모듈 (22) 에 의해 구현된 인터페이스들은 커맨드 큐 인터페이스 (24) 및 호스트 메시지 전달 인터페이스 (26) 를 포함한다. 일부 예들에서, 런타임 모듈 (22) 은 본 개시물에서 설명하는 인터페이스들에 더해, 표준 다중-프로세서 시스템 API 내에 포함되는 하나 이상의 인터페이스들을 구현할 수도 있다. 일부 예들에서, 표준 API 는 이종 컴퓨팅 플랫폼 API, 크로스-플랫폼 API, 크로스-벤더 API, 병렬 프로그래밍 API, 태스크-레벨 병렬 프로그래밍 API 및/또는 데이터-레벨 병렬 프로그래밍 API 일 수도 있다. 추가적인 예들에서, 표준 API 는 OpenCL API 일 수도 있다. 이러한 예들에서, 런타임 모듈 (22) 은 OpenCL 사양들 중 하나 이상에 따르는 것

으로 설계될 수도 있다. 추가적인 예들에서, 런타임 모듈 (22) 은 드라이버 프로그램, 예컨대, GPU 드라이버의 부분으로서 구현될 수도 있고 또는 드라이버 프로그램, 예컨대, GPU 드라이버일 수도 있다.

[0055] 커맨드 큐 인터페이스 (24) 는 호스트 프로세스 (20) 로부터 하나 이상의 인큐 명령들을 수신하고 수신된 명령들에 의해 특정된 기능들을 실행하도록 구성된다. 일부 예들에서, 커맨드 큐 인터페이스 (24) 는 OpenCL 사양에 따라 설계될 수도 있다. 예를 들어, 커맨드 큐 인터페이스 (24) 는 커맨드 큐들과 상호작용하기 위해 OpenCL 사양에 특정된 인큐 명령들 중 하나 이상을 구현할 수도 있다.

[0056] 본 개시물에 따르면, 호스트 메시지 전달 인터페이스 (26) 는 호스트 프로세스 (20) 로부터 하나 이상의 메시지 전달 명령들을 수신하고 수신된 명령들에 의해 특정된 기능들을 실행하도록 구성된다. 일부 예들에서, 호스트 메시지 전달 인터페이스 (26) 는 예컨대, OpenCL API 와 같은 기존의 표준 API 에 대한 확장으로서 구현될 수도 있다. 추가적인 예들에서, 호스트 메시지 전달 인터페이스 (26) 는 예컨대 OpenCL API 와 같은 기존의 표준 API 에 통합될 수도 있다.

[0057] GPU (14) 는 호스트 디바이스 (12) 로부터 수신된 명령들에 응답하여 하나 이상의 태스크들을 실행하도록 구성된다. GPU (14) 는 하나 이상의 프로그래밍가능 프로세싱 엘리먼트들을 포함하는 임의의 유형의 GPU 일 수도 있다. 예를 들어, GPU (14) 는 병렬로 태스크에 대한 복수의 실행 인스턴스들을 실행하도록 구성되는 하나 이상의 프로그래밍가능 셰이더 유닛들을 포함할 수도 있다. 프로그래밍가능 셰이더 유닛들은 정점 셰이더 유닛, 단편 셰이더 유닛, 지오메트리 셰이더 유닛 및/또는 통합 셰이더 유닛을 포함할 수도 있다. GPU (14) 는 상호접속 네트워크 (18) 를 통해 호스트 디바이스 (12) 및 메모리 (16) 에 통신적으로 커플링된다. GPU (14) 는 태스크 (28) 및 디바이스 메시지 전달 인터페이스 (30) 를 포함한다. 태스크 (28) 및 디바이스 메시지 전달 인터페이스 (30) 는 하나 이상의 프로그래밍가능 프로세싱 엘리먼트들의 임의의 조합 상에서 실행할 수도 있다.

[0058] 태스크 (28) 는 컴퓨팅 시스템 (10) 에서 컴퓨팅 디바이스 상에서의 실행을 위한 태스크를 형성하는 명령들의 세트를 포함한다. 일부 예들에서, 태스크 (28) 에 대한 명령들의 세트는 호스트 프로세스 (20) 에서 정의될 수도 있으며, 일부 경우들에서, 호스트 디바이스 (12) 상에서 실행하는 호스트 프로세스 (20) 에 포함된 명령들에 의해 컴파일될 수도 있다. 추가적인 예들에서, 태스크 (28) 는 GPU (14) 상에서 병렬로 실행하는 다수의 실행 인스턴스들을 갖는 커널 프로그램일 수도 있다. 이러한 예들에서, 호스트 프로세스 (20) 는 커널 실행 인스턴스들을 실행하기 위해 커널 실행 인스턴스들을 각각의 프로세싱 엘리먼트들에 맵핑하는 커널에 대한 인덱스 공간을 정의할 수도 있으며, GPU (14) 는 그 커널에 대해 정의된 인덱스 공간에 따라 태스크 (28) 에 대한 다수의 커널 실행 인스턴스들을 실행할 수도 있다.

[0059] 본 개시물에 따르면, 디바이스 메시지 전달 인터페이스 (30) 는 호스트 프로세스 (20) 로부터 하나 이상의 메시지 전달 명령들을 수신하고 수신된 명령들에 의해 특정된 기능들을 실행하도록 구성된다. 일부 예들에서, 디바이스 메시지 전달 인터페이스 (30) 는 기존의 표준 API 에 대한 확장으로서 구현될 수도 있다. 예를 들어, 표준 API 는 예컨대, OpenCL C API 와 같은 표준 컴퓨팅 디바이스 API 일 수도 있다. 추가적인 예들에서, 디바이스 메시지 전달 인터페이스 (30) 는 예컨대, OpenCL C API 와 같은 기존의 표준 API 에 통합될 수도 있다.

[0060] 메모리 (16) 는 호스트 디바이스 (12) 와 GPU (14) 중 하나 또는 양자에 의한 사용을 위한 데이터를 저장하도록 구성된다. 메모리 (16) 는 예를 들어, 랜덤 액세스 메모리 (RAM), 정적 RAM (SRAM), 동적 RAM (DRAM), 판독 전용 메모리 (ROM), 소거가능한 프로그래밍가능 ROM (EPROM), 전기적으로 소거가능한 프로그래밍가능 ROM (EEPROM), 플래시 메모리, 자기 데이터 저장 매체들 또는 광학 저장 매체들과 같은, 하나 이상의 휘발성 또는 비휘발성 메모리들 또는 저장 디바이스들의 임의의 조합을 포함할 수도 있다. 메모리 (16) 는 상호접속 네트워크 (18) 를 통해 호스트 디바이스 (12) 및 GPU (14) 에 통신적으로 커플링된다. 메모리 (16) 는 커맨드 큐 (32) 를 포함한다.

[0061] 커맨드 큐 (32) 는 커맨드 큐 인터페이스 (24) 로부터 수신된 커맨드들을 저장하고 취출하는 메모리 (16) 에 구현되는 데이터 구조일 수도 있다. 일부 예들에서, 커맨드 큐 (32) 는 실행을 위해 특정 순서로 커맨드들을 저장하는 버퍼일 수도 있다.

[0062] 상호접속 네트워크 (18) 는 호스트 디바이스 (12), GPU (14) 및 메모리 (16) 사이에 통신을 용이하게 하도록 구성된다. 상호접속 네트워크 (18) 는 당업계에서 알려진 임의의 유형의 상호접속 네트워크일 수도 있다. 도 1 의 예시적인 컴퓨팅 시스템 (10) 에서, 상호접속 네트워크 (18) 는 버스이다. 버스는 임의의 다양한

버스 구조들, 예컨대, 3 세대 버스 (예컨대, HyperTransport 버스 또는 InfiniBand 버스), 2 세대 버스 (예컨대, 진보된 그래픽들 포트 버스, PCIe (Peripheral Component Interconnect Express) 버스, 또는 AXI (Advanced eXentisible Interface) 버스) 또는 임의의 다른 유형의 버스 중 하나 이상을 포함할 수도 있다. 상호접속 네트워크 (18) 는 호스트 디바이스 (12), GPU (14) 및 메모리 (16) 에 커플링된다.

[0063] 이하, 컴퓨팅 시스템 (10) 내의 컴포넌트들의 구조들 및 기능성들이 더 상세히 설명될 것이다. 위에서 설명한 바와 같이, 호스트 프로세스 (20) 는 명령들의 세트를 포함한다. 명령들의 세트는 예를 들어, 하나 이상의 인큐 명령들, 및 하나 이상의 호스트 메시지 전달 명령들을 포함할 수도 있다. 추가적인 예들에서, 명령들의 세트는 GPU (14) 상에서 실행될 태스크들 또는 커널들을 특정하는 명령들, 커맨드 큐들을 생성하여 커맨드 큐들을 특정 디바이스들과 연관시키는 명령들, 프로그램들을 컴파일하고 바인딩하는 명령들, 커널 파라미터들을 셋업하는 명령들, 인덱스 공간들을 정의하는 명령들, 디바이스 콘텍스트를 정의하는 명령들 및 호스트 프로세스 (20) 에 의해 제공되는 기능성을 지원하는 다른 명령들을 포함할 수도 있다.

[0064] 호스트 프로세스 (20) 는 하나 이상의 커맨드들을 커맨드 큐 (32) 에 배치하도록 커맨드 큐 인터페이스 (24) 에 명령하는 하나 이상의 인큐 명령들을 커맨드 큐 인터페이스 (24) 에 이슈함으로써, 커맨드 큐 인터페이스 (24) 와 상호작용할 수도 있다. 하나 이상의 인큐 명령들은 메모리 전송 커맨드를 커맨드 큐 (32) 에 인큐하도록 커맨드 큐 인터페이스 (24) 에 명령하는 메모리 전송 인큐 명령들을 포함할 수도 있다. 예를 들어, 하나 이상의 인큐 명령들은 호스트 디바이스 (12) 와 연관된 메모리 공간과 GPU (14) 와 연관된 메모리 공간 사이에 데이터를 전송하도록, 호스트 디바이스 (12), 예컨대, 호스트 디바이스 (12) 상에서 실행하는 런타임 모듈 (22) 에 명령하는 커맨드를 인큐하는 명령을 포함할 수도 있다.

[0065] 메모리 공간은 호스트 디바이스 (12) 에 의한 호스트 프로세스 (20) 의 실행 동안 메모리 공간이 호스트 디바이스 (12) 에 의해 액세스가능하다면, 호스트 디바이스 (12) 와 연관될 수도 있다. 이와 유사하게, 메모리 공간은 GPU (14) 에 의한 태스크 (28) 의 실행 동안 메모리 공간이 GPU (14) 에 의해 액세스가능하다면, GPU (14) 와 연관될 수도 있다. 호스트 디바이스 (12) 와 연관된 메모리 공간은 본원에서 호스트 메모리 공간으로 지칭될 수도 있으며, GPU (14) 와 연관된 메모리 공간은 본원에서 디바이스 메모리 공간으로 지칭될 수도 있다.

일부 예들에서, 메모리 (16) 는 호스트 메모리 공간 및 디바이스 메모리 공간 양자의 부분들을 포함할 수도 있다. 추가적인 예들에서, 호스트 메모리 공간과 디바이스 메모리 공간 중 하나 또는 양자의 부분들은 도 1 의 컴퓨팅 시스템 (10) 에 나타내지지 않은 하나 이상의 다른 메모리 디바이스들 상에 로케이트될 수도 있다.

[0066] 일부 예들에서, 호스트 디바이스 (12) 와 연관된 메모리 공간과, GPU (14) 와 연관된 메모리 공간 사이에 데이터를 전송하도록 호스트 디바이스 (12) 에 명령하는 커맨드는 호스트 메모리 공간의 부분에 저장된 데이터를 디바이스 메모리 공간에 할당된 버퍼 오브젝트로 전송하도록 런타임 모듈 (22) 에 명령하는 커맨드일 수도 있다.

이러한 커맨드를 인큐하도록 호스트 프로세스 (20) 에 의해 이슈된 명령은 본원에서 기록 버퍼 인큐 명령으로 지칭될 수도 있다. 일부 경우들에서, 기록 버퍼 인큐 명령은 OpenCL API 사양에 의해 특정된 *clEnqueueWriteBuffer()* 함수의 형태를 취할 수도 있다.

[0067] 추가적인 예들에서, 호스트 디바이스 (12) 와 연관된 메모리 공간과, GPU (14) 와 연관된 메모리 공간 사이에 데이터를 전송하도록 호스트 디바이스 (12) 에 명령하는 커맨드는 디바이스 메모리 공간에 할당된 버퍼 오브젝트에 저장된 데이터를 호스트 메모리 공간의 부분으로 전송하도록 런타임 모듈 (22) 에 명령하는 커맨드일 수도 있다. 이러한 커맨드를 인큐하도록 호스트 프로세스 (20) 에 의해 이슈된 명령은 본원에서 판독 버퍼 인큐 명령으로 지칭될 수도 있다. 일부 경우들에서, 판독 버퍼 인큐 명령은 OpenCL API 사양에 의해 특정된 *clEnqueueReadBuffer()* 함수의 형태를 취할 수도 있다.

[0068] 하나 이상의 인큐 명령들은 또한 태스크 실행 커맨드를 커맨드 큐 (32) 에 인큐하도록 커맨드 큐 인터페이스 (24) 에 명령하는 태스크 실행 인큐 명령들을 포함할 수도 있다. 예를 들어, 하나 이상의 인큐 명령들은 호스트 디바이스 (12), 예컨대, 호스트 디바이스 (12) 상에서 실행하는 런타임 모듈 (22) 에, GPU (14) 상에서 태스크를 실행하도록 명령하는 커맨드를 인큐하는 명령을 포함할 수도 있다. 일부 예들에서, 태스크를 실행하는 커맨드는 태스크의 다수의 실행 인스턴스들을 GPU (14) 내의 복수의 프로세싱 엘리먼트들 상에서 병렬로 실행하는 커맨드일 수도 있다. 예를 들어, 태스크는 커널일 수도 있으며, 호스트 프로세스 (20) 는 커널 실행 인스턴스들을 실행하기 위해 커널 실행 인스턴스들을 GPU (14) 내의 각각의 프로세싱 엘리먼트들에 맵핑하는 커널에 대한 인덱스 공간을 정의할 수도 있다. 이러한 예에서, 태스크를 실행하는 커맨드는 GPU (14) 에 대해 정의된 인덱스 공간을 따라 GPU (14) 상에서 커널을 실행하는 커맨드일 수도 있다. 일부 경우들에서, 태스크 실행 인큐 명령은 OpenCL API 에 의해 특정된 *clEnqueueNDRangeKernel()* 함수의 형태를 취할 수도 있다.

[0069] 본 개시물에 따르면, 호스트 프로세스 (20) 는 또한 호스트 디바이스 (12) 상에서 실행하는 호스트 프로세스 (20) 와 GPU (14) 상에서 실행하는 태스크 (28) 사이에 하나 이상의 메시지들을 전달하도록 호스트 메시지 전달 인터페이스 (26) 에 명령하는 하나 이상의 호스트 메시지 전달 명령들을 호스트 메시지 전달 인터페이스 (26) 에 이슈함으로써, 호스트 메시지 전달 인터페이스 (26) 와 상호작용할 수도 있다. 호스트 메시지 전달 명령들은 호스트 디바이스 (12) 에 의해 실행될 수도 있다.

[0070] 호스트 메시지 전달 명령들은 일부 예들에서, 특정된 데이터를 특정된 디바이스로 전송하도록 호스트 디바이스 (12) 에 명령하는 전송 명령을 포함할 수도 있다. 예를 들어, 전송 명령은 메시지를 호스트 디바이스 (12) 상에서 실행하는 호스트 프로세스 (20) 로부터 GPU (14) 상에서 실행하는 태스크 (28) 로 전송하도록 호스트 메시지 전달 인터페이스 (26) 에 명령할 수도 있다. 일부 예들에서, 전송 명령은 메시지가 전송되어야 하는 특정 디바이스를 특정하는 제 1 입력 파라미터 및 전송될 메시지의 콘텐츠를 특정하는 제 2 입력 파라미터를 포함할 수도 있다.

[0071] 전송 명령은 블록킹 전송 명령 또는 넌-블록킹 전송 명령 중 어느 하나일 수도 있다. 전송 명령은 일부 예들에서, 전송 명령이 블록킹 전송 명령인지 또는 넌-블록킹 전송 명령인지 여부를 특정하는 제 3 입력 파라미터를 포함할 수도 있다. 블록킹 전송 명령은 호출 프로세스, 예컨대, 호스트 디바이스 (12) 상에서 실행하는 호스트 프로세스 (20) 로 리턴하기 전에 전송 동작이 완료될 때까지 대기할 수도 있다. 넌-블록킹 전송 명령은 전송 동작이 완료될 때까지 대기하지 않고, 호출 프로세스로 리턴할 수도 있다. 예를 들어, 넌-블록킹 전송 명령은 전송 동작이 성공적이었는지를 결정하기 위해 후속하여 호출 프로세스에 의해 질의될 수 있는 특정 전송 동작에 대한 핸들을 리턴할 수도 있다. 넌-블록킹 전송 동작은 실패할 수도 있으며, 실패의 경우에, 호출 프로세스는 전송 동작을 재시도하기 위해 다시 전송 명령을 이슈할 필요가 있을 수도 있다.

[0072] 일부 예들에서, 전송 명령에 대한 인터페이스는 다음의 형태를 취할 수도 있다:

[0073] clSendOutOfBandData(

[0074] cl_device *deviceId,

[0075] int OOB_data,

[0076] bool blocking)

[0077] 여기서, clSendOutOfBandData 는 명령 식별자이고, cl_device *deviceId 는 메시지가 전송되어야 하는 특정 OpenCL 디바이스를 특정하는 입력 파라미터이고, int OOB_data 는 전송될 메시지의 콘텐츠를 특정하는 입력 파라미터이며, bool blocking 은 그 명령이 블록킹 전송 명령인지 또는 넌-블록킹 전송 명령인지 여부를 특정하는 입력 파라미터이다. 블록킹 명령의 경우에, 이 명령은 전송 동작이 성공적으로 완료되었는지 여부를 나타내는 파라미터를 리턴할 수도 있다. 넌-블록킹 명령의 경우에, 이 명령은 호출 프로세스에 의해 질의하는 후속 상태에 대한 핸들 파라미터를 리턴할 수도 있다.

[0078] 호스트 메시지 전달 명령들은 일부 예들에서, 특정된 디바이스로부터 비동기적 방식으로 데이터를 수신하기 위해 콜백을 등록하도록 호스트 디바이스 (12) 에 명령하는 레지스터 콜백 루틴 명령을 포함할 수도 있다. 예를 들어, 레지스터 콜백 루틴 명령은 GPU (14) 로부터, GPU (14) 상에서 실행하는 태스크가 메시지를 호스트 프로세스 (20) 로 전송했다는 것을 나타내는 신호를 수신하는 것에 응답하여, 콜백 루틴을 인보크하도록 호스트 메시지 전달 인터페이스 (26) 에 명령할 수도 있다. 레지스터 콜백 루틴 명령은 콜백 루틴이 등록되어야 하는 특정 디바이스를 특정하는 제 1 입력 파라미터 및 콜백 루틴의 메모리 로케이션을 특정하는 제 2 입력 파라미터를 포함할 수도 있다.

[0079] 일부 예들에서, 레지스터 콜백 루틴 명령에 대한 인터페이스는 다음의 형태를 취할 수도 있다:

[0080] clRegisterOutOfBandDataCallback(

[0081] cl_device *deviceId,

[0082] void(*)(int) callBackPtr)

[0083] 여기서, clRegisterOutOfBandDataCallback 은 명령 식별자이고, cl_device *deviceId 는 메시지가 전송되어야 하는 특정 OpenCL 디바이스를 특정하는 입력 파라미터이며, void(*)(int) callBackPtr 은 콜백 루틴의 메모리 로케이션을 특정하는 입력 파라미터이다. 레지스터 콜백 루틴 명령은 콜백 루틴 등록 동작이 성공적으로 완

료되었는지 여부를 나타내는 파라미터를 리턴할 수도 있다.

[0084] 호스트 메시지 전달 명령들은 일부 예들에서, 특정된 디바이스로부터 데이터를 판독하는 시도를 행하도록 호스트 디바이스 (12) 에 명령하는 폴링 명령을 포함할 수도 있다. 예를 들어, 폴링 명령은 GPU (14) 상에서 실행하는 태스크 (28) 가 메시지를 전송했는지 여부를 나타내는 메시지 상태 정보에 대해 GPU (14) 를 폴링하도록 호스트 메시지 전달 인터페이스 (26) 에 명령할 수도 있다. 폴링 명령은 폴링될 특정 디바이스를 특정하는 입력 파라미터, 및 만약에 있다면, 폴링의 결과로서 획득된 데이터를 특정하는 출력 파라미터를 포함할 수도 있다.

[0085] 일부 예들에서, 폴링 명령에 대한 인터페이스는 다음의 형태를 취할 수도 있다:

[0086] clTryReadOutOfBandData(

[0087] cl_device *deviceId,

[0088] int *OOB_data)

[0089] 여기서, clTryReadOutOfBandData 는 명령 식별자이고, cl_device *deviceId 는 폴링될 특정 OpenCL 디바이스를 특정하는 입력 파라미터이며, int *OOB_data 는 만약에 있다면, 폴링의 결과로서 획득된 데이터를 특정하는 출력 파라미터이다. 폴링 명령은 데이터가 폴링 동작으로부터 성공적으로 획득되었는지 여부를 나타내는 파라미터를 리턴할 수도 있다.

[0090] 호스트 프로세스 (20) 와 유사하게, 태스크 (28) 는 컴퓨팅 디바이스에 의해 실행되는 하나 이상의 디바이스 메시지 전달 명령들을 포함할 수도 있다. 디바이스 메시지 전달 명령들은 특정된 데이터를 호스트 디바이스 (12) 로 전송하도록 컴퓨팅 디바이스에 명령하는 전송 명령을 포함할 수도 있다. 예를 들어, 전송 명령은 메시지를 GPU (14) 상에서 실행하는 태스크 (28) 로부터 호스트 디바이스 (12) 상에서 실행하는 호스트 프로세스 (20) 로 전송하도록 GPU (14) 에 명령할 수도 있다.

[0091] 전송 명령은 블록킹 전송 명령 또는 넌-블록킹 전송 명령 중 어느 하나일 수도 있다. 전송 명령은 일부 예들에서, 전송 명령이 블록킹 전송 명령인지 또는 넌-블록킹 전송 명령인지 여부를 특정하는 제 1 입력 파라미터를 포함할 수도 있다. 블록킹 전송 명령은 호출 프로세스, 예컨대, GPU (14) 상에서 실행하는 태스크 (28) 를 스톨 (stall) 하고, 호출 프로세스로 리턴하기 전에 전송 동작이 완료되기를 대기할 수도 있다. 넌-블록킹 전송 명령은 전송 동작이 완료될 때까지 대기하지 않고, 호출 프로세스로 리턴할 수도 있다. 예를 들어, 넌-블록킹 전송 명령은 전송 동작이 성공적이었는지를 결정하기 위해 후속하여 호출 프로세스에 의해 질의될 수 있는 특정 전송 동작에 대한 핸들을 리턴할 수도 있다. 넌-블록킹 전송 동작은 실패할 수도 있으며, 실패의 경우에, 호출 프로세스는 전송 동작을 재시도하기 위해 다시 전송 명령을 이슈할 필요가 있을 수도 있다. 전송 명령은 호스트 디바이스로 전송될 메시지의 콘텐츠를 특정하는 제 2 입력 파라미터를 포함할 수도 있다.

[0092] 일부 예들에서, 전송 명령에 대한 인터페이스는 다음의 형태를 취할 수도 있다:

[0093] send_oobdata (

[0094] bool blocking,

[0095] int data)

[0096] 여기서, send_oobdata 는 명령 식별자이고, bool blocking 은 그 명령이 블록킹 전송 명령인지 또는 넌-블록킹 전송 명령인지 여부를 특정하는 입력 파라미터이며, int data 는 전송될 메시지의 콘텐츠를 특정하는 입력 파라미터이다. 블록킹 명령의 경우에, 이 명령은 전송 동작이 성공적으로 완료되었는지 여부를 나타내는 파라미터를 리턴할 수도 있다. 넌-블록킹 명령의 경우에, 이 명령은 호출 프로세스에 의해 질의하는 후속 상태에 대한 핸들 파라미터를 리턴할 수도 있다.

[0097] 디바이스 메시지 전달 명령들은 일부 예들에서, 호스트 디바이스 (12) 로부터 데이터를 수신하도록 컴퓨팅 디바이스에 명령하는 수신 명령을 포함할 수도 있다. 예를 들어, 수신 명령은 이용가능하다면, 호스트 디바이스 (12) 상에서 실행하는 호스트 프로세스 (20) 로부터 태스크 (28) 로 전송된 메시지를 GPU (14) 상에서 실행하는 태스크 (28) 에 제공하도록 GPU (14), 예컨대, 디바이스 메시지 전달 인터페이스 (30) 에 명령할 수도 있다. 이러한 명령은 폴링 메커니즘을 지원하는데 사용될 수도 있다.

[0098] 수신 명령은 블록킹 수신 명령 또는 넌-블록킹 수신 명령 중 어느 하나일 수도 있다. 수신 명령은 일부 예

들에서, 그 수신 명령이 블록킹 수신 명령인지 또는 넌-블록킹 수신 명령인지 여부를 특정하는 입력 파라미터를 포함할 수도 있다. 블록킹 수신 명령은 호출 프로세스, 예컨대, GPU (14) 상에서 실행하는 태스크 (28) 를 스톨하고, 호출 프로세스로 리턴하기 전에 그 메시지가 이용가능할 때까지 대기할 수도 있다. 넌-블록킹 수신 명령은 메시지가 이용가능할 때까지 대기하지 않고, 호출 프로세스로 리턴할 수도 있다. 예를 들어, 메시지가 이용가능하다면, 넌-블록킹 수신 명령은 메시지를 리턴할 수도 있다. 그러나, 메시지가 이용가능하지 않다면, 넌-블록킹 수신 명령은 실패할 수도 있다. 실패의 경우에, 호출 프로세스는 수신 동작을 재시도하기 위해 다시 수신 명령을 이슈할 필요가 있을 수도 있다. 수신 명령은 만약에 있다면, 수신 동작의 결과로서 획득된 데이터를 특정하는 출력 파라미터를 포함할 수도 있다.

[0099] 일부 예들에서, 수신 명령에 대한 인터페이스는 다음의 형태를 취할 수도 있다:

[0100] receive_oobdata(

[0101] bool blocking,

[0102] int data)

[0103] 여기서, receive_oobdata 는 명령 식별자이고, bool blocking 은 그 명령이 블록킹 수신 명령인지 또는 넌-블록킹 수신 명령인지 여부를 특정하는 입력 파라미터이며, int data 는 만약에 있다면, 수신 동작의 결과로서 획득된 데이터를 특정하는 출력 파라미터이다. 명령은 수신 동작이 성공적으로 완료되었는지 여부를 나타내는 파라미터를 리턴할 수도 있다.

[0104] 커맨드 큐 인터페이스 (24) 는 커맨드들을 커맨드 큐 (32) 에 인큐하도록 구성된다. 예를 들어, 커맨드 큐 인터페이스 (24) 는 호스트 프로세스 (20) 로부터 하나 이상의 인큐 명령들을 수신하고, 호스트 프로세스 (20) 로부터 하나 이상의 인큐 명령들을 수신하는 것에 응답하여, 하나 이상의 커맨드들을 커맨드 큐 (32) 에 배치할 수도 있다. 하나 이상의 인큐 명령들은 태스크 실행 커맨드들 및 데이터 전송 커맨드들을 각각 인큐하도록 커맨드 큐 인터페이스 (24) 에 명령하는 태스크 실행 인큐 명령들 및 데이터 전송 인큐 명령들을 포함할 수도 있다.

[0105] 커맨드 큐 인터페이스 (24) 는 또한 커맨드 큐 (32) 에 저장된 커맨드들을 실행하도록 구성된다. 데이터 전송 커맨드들에 대해, 커맨드 큐 인터페이스 (24) 는 호스트 메모리 공간과 디바이스 메모리 공간 사이에 데이터를 전송할 수도 있다. 예를 들어, 기록 버퍼 커맨드에 대해, 커맨드 큐 인터페이스 (24) 는 호스트 메모리 공간의 부분에 저장된 데이터를 디바이스 메모리 공간에 할당된 버퍼 오브젝트로 전송할 수도 있다. 또 다른 예로서, 판독 버퍼 커맨드에 대해, 커맨드 큐 인터페이스 (24) 는 디바이스 메모리 공간에 할당된 버퍼 오브젝트에 저장된 데이터를 호스트 메모리 공간의 부분으로 전송할 수도 있다. 디바이스 메모리 공간은 커맨드 큐 (32) 가 연관되는 디바이스에 대응할 수도 있다.

[0106] 태스크 실행 커맨드들에 대해, 커맨드 큐 인터페이스 (24) 는 태스크의 실행이 커맨드 큐와 연관된 디바이스 상에서 시작하도록 할 수도 있다. 예를 들어, 도 1 의 예에서, 커맨드 큐 (32) 는 런타임 모듈 (22) 의 콘텍스트 내에서 GPU (14) 와 연관된다. 따라서, 태스크 실행 커맨드를 실행할 때, 커맨드 큐 인터페이스 (24) 는 태스크가 GPU (14) 상에서 실행하기 시작하도록 할 수도 있다. 일부 예들에서, 커맨드 큐 인터페이스 (24) 는 하나 이상의 커맨드들을 GPU (14) 내에 포함된 로컬 커맨드 큐에 배치함으로써, 태스크가 GPU (14) 상에서 실행하기 시작하도록 할 수도 있다. 다른 예들에서, 커맨드 큐 인터페이스 (24) 는, 태스크로 하여금, 태스크의 실행을 시작하도록 GPU (14) 에 명령하는 하나 이상의 명령들을 GPU (14) 로 전송함으로써, GPU (14) 상에서 실행하기 시작하도록 할 수도 있다. 커맨드 큐 인터페이스 (24) 는 GPU (14), 메모리 (16), 및 호스트 및 디바이스 메모리 공간들과 통신하기 위해, 상호접속 네트워크 (18) 를 이용할 수도 있다.

[0107] 일부 예들에서, 커맨드 큐 인터페이스 (24) 는 커맨드들을 순차로 실행할 수도 있다. 이러한 예들에서, 제 1 커맨드가 제 2 커맨드 이전에 인큐된다면, 제 1 커맨드가 실행을 완료한 후에 제 2 커맨드의 실행이 시작된다. 추가적인 예들에서, 커맨드 큐 인터페이스 (24) 는 커맨드들을 비순차로 실행할 수도 있다. 이러한 예들에서, 제 1 커맨드가 제 2 커맨드 이전에 인큐된다면, 제 2 커맨드의 실행이 반드시 제 1 커맨드가 실행을 완료한 후에 시작될 필요는 없다.

[0108] 호스트 메시지 전달 인터페이스 (26) 는 호스트 프로세스 (20) 로부터 수신된 하나 이상의 메시지 전달 명령들을 실행하도록 구성된다. 예를 들어, 호스트 프로세스 (20) 로부터 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 호스트 메시지 전달 인터페이스 (26) 는 태스크 (28) 가 GPU (14) 상에서 실행중인 동안 호스트 디바이스 (12) 상에서 실행하는 호스트 프로세스 (20) 와 GPU (14) 상에서 실행하는 태스크 (28) 사이에

하나 이상의 메시지들을 전달할 수도 있다. 일부 예들에서, 호스트 메시지 전달 인터페이스 (26) 는 임의의 커맨드들을 커맨드 큐 (32) 에 배치하지 않고, 하나 이상의 메시지 전달 명령들을 실행할 수도 있다.

[0109] 제 1 예에 따르면, 호스트 프로세스 (20) 로부터 전송 명령을 수신하는 것에 응답하여, 호스트 메시지 전달 인터페이스 (26) 는 태스크 (28) 가 GPU (14) 상에서 실행중인 동안 메시지를 호스트 프로세스 (20) 로부터 태스크 (28) 로 전송할 수도 있다. 예를 들어, 호스트 메시지 전달 인터페이스 (26) 는 그 전송 명령 내에 포함된 메시지 데이터에 기초하여 인출 (outgoing) 메시지를 구성하고, 그 특정된 디바이스 상에서 실행하는 태스크, 예컨대, 태스크 (28) 로의 전달을 위해, 그 인출 메시지를 상호접속 네트워크 (18) 를 통해 그 전송 명령에 특정된 디바이스, 예컨대, GPU (14) 로 전송할 수도 있다.

[0110] 제 2 예에 따르면, 호스트 프로세스 (20) 로부터 레지스터 콜백 루틴 명령을 수신하는 것에 응답하여, 호스트 메시지 전달 인터페이스 (26) 는 그 명령에 특정된 콜백 루틴을 그 특정된 디바이스 상에서 실행하는 태스크, 예컨대, 태스크 (28) 가 메시지를 전송했다는 것을 나타내는, 그 명령에 특정된 디바이스, 예컨대, GPU (14) 로부터의 신호와 연관시킬 수도 있다. 일부 예들에서, 그 디바이스로부터의 신호는 인터럽트 신호일 수도 있다. 인터럽트 신호는, 일부 예들에서, 전용 인터럽트 신호 라인을 통해 전달될 수도 있다. 그 디바이스 상에서 실행하는 태스크가 메시지를 전송했다는 것을 나타내는 신호를 특정된 디바이스로부터 수신하는 것에 응답하여, 호스트 메시지 전달 인터페이스 (26) 는 레지스터 콜백 루틴 명령에 특정된 콜백 루틴의 실행을 개시할 수도 있다. 콜백 루틴은 태스크, 예컨대, 태스크 (28) 에 의해 전송된 메시지를, 특정된 디바이스, 예컨대, GPU (14) 로부터 획득하고, 추가적인 프로세싱을 위해 그 메시지를 호스트 프로세스 (20) 로 리턴할 수도 있다.

[0111] 제 3 예에 따르면, 폴링 명령을 수신하는 것에 응답하여, 호스트 메시지 전달 인터페이스 (26) 는 메시지 상태 정보에 대해 그 명령에 특정된 디바이스, 예컨대, GPU (14) 를 폴링할 수도 있다. 호스트 메시지 전달 인터페이스 (26) 는 그 디바이스를 폴링하기 위해 상호접속 네트워크 (18) 또는 또다른 하드웨어 기반의 통신 경로를 이용할 수도 있다. 특정된 디바이스, 예컨대, GPU (14) 상에서 실행하는 태스크, 예컨대, 태스크 (28) 가 메시지를 전송했다는 것을 메시지 상태 정보가 나타낸다면, 호스트 메시지 전달 인터페이스 (26) 는 특정된 디바이스로부터 메시지를 획득하고, 추가적인 프로세싱을 위해 그 메시지를 호스트 프로세스 (20) 로 리턴할 수도 있다.

[0112] 디바이스 메시지 전달 인터페이스 (30) 는 태스크 (28) 로부터 수신된 하나 이상의 디바이스 메시지 전달 명령들을 실행하도록 구성된다. 예를 들어, 태스크 (28) 로부터 하나 이상의 디바이스 메시지 전달 명령들을 수신하는 것에 응답하여, 디바이스 메시지 전달 인터페이스 (30) 는 태스크 (28) 가 GPU (14) 상에서 실행중인 동안, GPU (14) 상에서 실행하는 태스크 (28) 와 호스트 디바이스 (12) 상에서 실행하는 호스트 프로세스 (20) 사이에 하나 이상의 메시지들을 전달할 수도 있다.

[0113] 제 1 예에 따르면, 전송 명령을 수신하는 것에 응답하여, 디바이스 메시지 전달 인터페이스 (30) 는 메시지를 GPU (14) 상에서 실행하는 태스크 (28) 로부터 호스트 디바이스 (12) 상에서 실행하는 호스트 프로세스 (20) 로 전송할 수도 있다. 예를 들어, 디바이스 메시지 전달 인터페이스 (30) 는 그 전송 명령 내에 포함된 메시지 데이터에 기초하여, 인출 메시지를 구성하고, 호스트 프로세스 (20) 로의 전달을 위해 인출 메시지를 상호접속 네트워크 (18) 를 통해 호스트 디바이스 (12) 로 전송할 수도 있다.

[0114] 제 2 예에 따르면, 태스크 (28) 로부터 수신 명령을 수신하는 것에 응답하여, 디바이스 메시지 전달 인터페이스 (30) 는 호스트 프로세스 (20) 로부터의 메시지가 이용가능한지를 결정할 수도 있다. 일부 예들에서, 디바이스 메시지 전달 인터페이스 (30) 는 메시지가 이용가능한지를 결정하기 위해 하나 이상의 호스트-액세스가능 레지스터들을 체크할 수도 있다. 호스트 프로세스 (20) 로부터의 메시지가 이용가능하다면, 디바이스 메시지 전달 인터페이스 (30) 는 그 메시지를 태스크 (28) 에 제공할 수도 있다.

[0115] 커맨드 큐 인터페이스 (24) 및 호스트 메시지 전달 인터페이스 (26) 가 도 1 에 호스트 프로세스 (20) 와 별개인 컴포넌트들로서 도시되지만, 일부 예들에서, 커맨드 큐 인터페이스 (24) 및 호스트 메시지 전달 인터페이스 (26) 중 하나 또는 양자의 기능성은 호스트 프로세스 (20) 로 부분적으로 및/또는 완전히 컴파일될 수도 있다.
이와 유사하게, 일부 예들에서, 디바이스 메시지 전달 인터페이스 (30) 의 기능성은 태스크 (28) 로 부분적으로 및/또는 완전히 컴파일될 수도 있다.

[0116] 예시의 용이성을 위해, 도 1 에 도시된 예시적인 컴퓨팅 시스템 (10) 은 GPU (14) 를 컴퓨팅 디바이스로서 이용하는 본 개시물의 메시지 전달 기법들을 설명한다. 그러나, 본 개시물의 기법들은 GPU (14) 에 더하여 또는 대신에, GPU 와는 다른 컴퓨팅 디바이스들을 갖는 다중-프로세서 컴퓨팅 시스템들에 적용될 수도 있는 것으로

인식되어야 한다. 일부 예들에서, 컴퓨팅 디바이스들은 OpenCL 계산 디바이스들일 수도 있다. OpenCL 계산 디바이스는 하나 이상의 계산 유닛들을 포함한다. 계산 유닛들 각각은 하나 이상의 프로세싱 엘리먼트들을 포함한다. 예를 들어, 계산 유닛은 계산 유닛에서 프로세싱 엘리먼트들의 모두에 의해 사용될 수 있는 온칩 공유 메모리를 갖는 프로세싱 엘리먼트들, 예컨대, ALU들의 클러스터일 수도 있다. 작업 아이템은 커맨드 큐에 배치된 커맨드에 의해 OpenCL 계산 디바이스 상에 인보크되는 커널 또는 태스크의 복수의 병렬 실행들 중 하나일 수도 있다. 각각의 작업 아이템은 계산 유닛의 개개의 프로세싱 엘리먼트 상에서 다른 프로세싱 엘리먼트들 상에서 실행하는 다른 작업 아이템들과 병렬로 실행할 수도 있다. 작업 그룹은 계산 디바이스 내의 단일 계산 유닛 상에서 단일 커널 실행 커맨드의 부분으로서 프로세싱되는 하나 이상의 작업-아이템들의 컬렉션일 수도 있다. OpenCL 호스트는 OpenCL 런타임 레이어를 실행하는데 사용되는 플랫폼의 중앙 CPU 일 수도 있다.

[0117] OpenCL API 는 호스트 디바이스와 상이한 유형들의 계산 디바이스들 사이의 상호작용을 위해 공통 세트의 인터페이스들을 제공할 수도 있다. 예를 들어, OpenCL API 는 호스트와 GPU 계산 디바이스 사이, 그리고 호스트와 넌-GPU 계산 디바이스 사이의 상호작용을 위해 공통 인터페이스를 제공할 수도 있다. OpenCL API 는 호스트가 여러 계산 디바이스들 상에서 태스크들 (예컨대, OpenCL 커널들) 을 실행하는 공통 인터페이스를 이용하는 것을 허용한다. 일부 예들에서, 태스크들은 범용 컴퓨팅 태스크들일 수도 있으며, OpenCL API 는 호스트가 범용 컴퓨팅 태스크로 하여금 GPU 계산 디바이스 상에서 실행하도록 하는 것을 허용할 수도 있다.

[0118] 도 1 에 나타낸 예시적인 컴퓨팅 시스템 (10) 은 호스트 디바이스와 컴퓨팅 디바이스 사이에 메시지 전달 및/또는 대역외 시그널링을 용이하게 하는 인프라스트럭처 및 기법들을 도시한다. 그러나, 다른 예시적인 컴퓨팅 시스템들에서, 이 기법들은 하나 보다 많은 컴퓨팅 디바이스를 갖는 컴퓨팅 시스템에서 상이한 컴퓨팅 디바이스들 (예컨대, OpenCL 계산 디바이스들) 사이에 인-플라이트 메시지 전달을 제공하도록 용이하게 확장될 수도 있다. 이러한 예들에서, 하나 이상의 인터럽트 라인들은 상이한 컴퓨팅 디바이스들 사이에 유선으로 연결될 수도 있다.

[0119] 도 2 는 본 개시물에 따른, 도 1 의 컴퓨팅 시스템 (10) 에 사용될 수도 있는 예시적인 GPU (40) 를 도시하는 블록도이다. 일부 예들에서, GPU (40) 는 도 1 에 도시된 GPU (14) 를 구현하는데 사용될 수도 있다. GPU (40) 는 GPU 프로세싱 블록 (42), 호스트-액세스가능 GPU 레지스터들 (44) 및 버스 제어기 (46) 를 포함한다. GPU (40) 는 상호접속 네트워크 (18) 를 통해 하나 이상의 다른 호스트 디바이스들 또는 컴퓨팅 디바이스들과 통신할 수도 있다.

[0120] GPU 프로세싱 블록 (42) 은 태스크들을 실행하고, GPU 프로세싱 블록 (42) 상에서 실행하는 태스크들과 다른 호스트 또는 컴퓨팅 디바이스들 상에서 실행하는 프로세스들 사이에 메시지 전달을 용이하게 하도록 구성된다. GPU 프로세싱 블록 (42) 은 예컨대, 하나 이상의 제어 및/또는 데이터 라인들을 통해 호스트-액세스가능 GPU 레지스터들 (44) 에 통신적으로 커플링된다. 일부 예들에서, GPU 프로세싱 블록 (42) 은 산술 로직 유닛 (ALU) 블록으로 지칭될 수도 있다. GPU 프로세싱 블록 (42) 은 태스크 (48), 메시지 전달 모듈 (50), 인입 데이터 레지스터 (52) 및 인출 데이터 레지스터 (54) 를 포함한다.

[0121] 호스트-액세스가능 GPU 레지스터들 (44) 은 호스트 디바이스로 통신되거나 또는 호스트 디바이스로부터 통신될 수도 있는 데이터를 저장하도록 구성된다. 호스트-액세스가능 GPU 레지스터들 (44) 은 메시지 상태 레지스터 (56), 메시지 카운트 레지스터 (58), 인입 메시지 레지스터 (60), 인출 메시지 레지스터 (62), 인터럽트 상태 레지스터 (64) 및 인터럽트 확인응답 레지스터 (66) 를 포함한다. 호스트-액세스가능 GPU 레지스터들 (44) 각각은 호스트 디바이스, 예컨대, 도 1 의 호스트 디바이스 (12) 에 의해 액세스가능할 수도 있다. 일부 예들에서, 호스트-액세스가능 GPU 레지스터들 (44) 은 메모리-맵핑된 레지스터들, 즉, 호스트 디바이스의 메모리 공간에 맵핑되어 어드레스가능한 레지스터들일 수도 있다. 추가적인 예들에서, 호스트-액세스가능 GPU 레지스터들 (44) 은 입력/출력-맵핑된 (I/O-맵핑된) 레지스터들, 즉, 호스트 디바이스의 I/O 공간에 맵핑된 레지스터들일 수도 있다. 호스트-액세스가능 GPU 레지스터들 (44) 은 하나 이상의 제어 및/또는 데이터 라인들을 통해 GPU 프로세싱 블록 (42) 에 통신적으로 커플링된다. 호스트-액세스가능 GPU 레지스터들 (44) 은 또한 상호접속 네트워크 (18) 를 통해 버스 제어기 (46) 에 통신적으로 커플링된다.

[0122] 태스크 (48) 는 하나 이상의 프로그래밍가능 프로세서들 상에서 실행할 수도 있다. 일부 예들에서, GPU 프로세싱 블록 (42) 은 태스크 (48) 의 다수의 실행 인스턴스들을 실행하도록 구성된 다중 프로세서들 또는 프로세싱 엘리먼트들을 포함할 수도 있다. 태스크 (48) 는 도 1 에 대해 위에서 설명한 태스크 (28) 와 실질적으로 유사할 수도 있으며, 따라서 더 상세히 설명되지 않을 것이다.

[0123] 메시지 전달 모듈 (50) 은 GPU (40) 에 의해 수행되는 메시지 전달 동작들을 제어하도록 구성된다. 메시지 전달 모듈 (50) 은 하드웨어, 소프트웨어, 펌웨어 또는 이들의 임의의 조합으로 구현될 수도 있다. 일부 예들에서, 메시지 전달 모듈 (50) 의 기능성의 부분 또는 모두가 소프트웨어로 구현되면, 이러한 구현을 위한 소프트웨어 명령들은 동일한 실행가능한 파일 내에, 태스크 (48) 에 대한 소프트웨어 명령들을 포함하는 실행가능한 파일로서 포함될 수도 있다. 메시지 전달 모듈 (50) 은 태스크 (48), 인입 (incoming) 데이터 레지스터 (52) 및 인출 (outgoing) 데이터 레지스터 (54) 에 통신적으로 커플링된다.

[0124] 메시지 전달 모듈 (50) 은 태스크 (48) 가 하나 이상의 프로세서들 상에서 실행중인 동안, 그리고 태스크 (48) 로부터 하나 이상의 메시지 전달 명령들을 수신하는 것에 응답하여, 하나 이상의 프로세서들 상에서 실행하는 태스크 (48) 와 호스트 디바이스 상에서 실행하는 프로세스 사이에 호스트-액세스가능 GPU 레지스터들 (44) 을 통해, 하나 이상의 메시지들을 전달할 수도 있다. 일부 예들에서, 하나 이상의 메시지 전달 명령들은 메시지를 태스크 (48) 로부터 호스트 디바이스 상에서 실행하는 프로세스로 전송하도록 메시지 전달 모듈 (50) 에 명령하는 전송 명령을 포함할 수도 있다. 이러한 예들에서, 메시지 전달 모듈 (50) 은 호스트-액세스가능 GPU 레지스터들 (44) 의 하나에 그 메시지와 연관된 메시지 데이터를 저장할 수도 있다. 추가적인 예들에서, 하나 이상의 메시지 전달 명령들은, 이용가능하다면, 호스트 디바이스 상에서 실행하는 프로세스로부터 태스크 (48) 로 전송된 메시지를 태스크 (48) 에 제공하도록 메시지 전달 모듈 (50) 에 명령하는 수신 명령을 포함할 수도 있다. 이러한 예들에서, 메시지 전달 모듈 (50) 은 호스트-액세스가능 GPU 레지스터들 (44) 의 하나 이상으로부터 그 메시지와 연관된 메시지 데이터를 획득할 수도 있다.

[0125] 인입 데이터 레지스터 (52) 는 도 2 의 예에서, 인입 메시지 레지스터 (60) 를 통해 외부 디바이스로부터 수신된 인입 데이터 (incoming data) 를 저장하는 하드웨어 레지스터이다. 인입 데이터 레지스터 (52) 는 또한 인입 데이터 레지스터 (52) 내의 데이터가 소비되었는지 여부 및/또는 인입 데이터 레지스터 (52) 내의 데이터가 판독하는데 이용가능한지 여부를 나타내는 상태 비트를 저장할 수도 있다. 인입 데이터 레지스터 (52) 는 하나 이상의 데이터 라인들을 통해, 인입 메시지 레지스터 (60) 에 통신적으로 커플링된다. 일부 예들에서, 데이터 라인들의 수는 인입 데이터 레지스터 (52) 내의 비트들의 수와 동일할 수도 있으며, 이들 양자는 메시지 내의 비트들의 수와 동일할 수도 있다. 추가적인 예들에서, 비트들의 수는 32 비트들일 수도 있다. 일부 예들에서, GPU 프로세싱 블록 (42) 은 인입 데이터 레지스터 (52) 로부터 수신된 복수의 인입 메시지들을 저장하기 위해 내부 선입선출 (FIFO) 버퍼를 구현할 수도 있다.

[0126] 인출 데이터 레지스터 (54) 는, 도 2 의 예에서, 태스크 (48) 에 의해 이슈된 하나 이상의 메시지 전달 명령들로부터 수신된 인출 데이터를 저장하는 하드웨어 레지스터이다. 인출 데이터 레지스터 (54) 는 하나 이상의 데이터 라인들을 통해 인출 메시지 레지스터 (62) 에 통신적으로 커플링된다. 일부 예들에서, 데이터 라인들의 수는 인출 데이터 레지스터 (54) 내의 비트들의 수와 동일할 수도 있으며, 이들 양자는 메시지 내의 비트들의 수와 동일할 수도 있다. 일부 예들에서, 인출 데이터 레지스터 (54) 및 인출 메시지 레지스터 (62) 는 메시지 전달 모듈 (50) 이 인출 데이터 레지스터 (54) 에 데이터를 기록할 때, 인출 메시지 레지스터 (62) 가 인출 데이터 레지스터 (54) 에 기록된 데이터로 자동적으로 업데이트되도록 구성될 수도 있다. 일부 예들에서, GPU 프로세싱 블록 (42) 은 인출 데이터 레지스터 (54) 에 기록될 복수의 인출 메시지들을 저장하기 위해 내부 선입선출 (FIFO) 버퍼를 구현할 수도 있다.

[0127] 메시지 상태 레지스터 (56) 는, 도 2 의 예에서, 인입 메시지가 GPU (40) 에 의해 수락되었는지 여부를 나타내는 데이터를 저장하도록 구성된다. 메시지 상태 레지스터 (56) 는 메시지가 성공적으로 송신되었는지 여부를 결정하기 위해, 그리고 일부 예들에서는, 백-오프 및/또는 오버플로우 메커니즘을 구현하기 위해, 호스트 디바이스에 의해 사용될 수도 있다. 인입 메시지를 수락한 후, 메시지 전달 모듈 (50) 은 메시지 상태 레지스터 (56) 를, 인입 메시지가 수락되었다는 것을 나타내는 특정 값으로 설정할 수도 있다.

[0128] 메시지 카운트 레지스터 (58) 는, 도 2 의 예에서, 인입 메시지 레지스터 (60) 가 인입 메시지를 포함하는지 여부를 나타내는 데이터를 저장하도록 구성된다. 일부 예들에서, 메시지 카운트 레지스터 (58) 는 메시지 카운트 레지스터 (58) 가 호스트 디바이스에 의해 증분되었을 때 메시지 도달을 나타내는 신호를 메시지 전달 모듈 (50) 로 전송할 수도 있다. 일부 경우들에서, 신호는 1 비트 펄스 라인일 수도 있다. 추가적인 예들에서, 메시지 전달 모듈 (50) 은 인입 데이터 레지스터 (52) 로부터의 메시지를 판독한 후 메시지 카운트 레지스터 (58) 를 감분시킬 수도 있다.

[0129] 인입 메시지 레지스터 (60) 는, 도 2 의 예에서, 인입 메시지 데이터를 저장하도록 구성된다. 예를 들어, 호스트 디바이스는 메시지를 태스크 (48) 로 전송하기 위하여 인입 메시지 데이터를 인입 메시지 레지스터 (60)

에 배치할 수도 있다. 인입 메시지 레지스터 (60) 는 인입 데이터 레지스터 (52) 에 통신적으로 커플링된다.

[0130] 인출 메시지 레지스터 (62) 는, 도 2 의 예에서, 인출 데이터 레지스터 (54) 로부터 수신된 인출 메시지 데이터를 저장하도록 구성된다. 인출 메시지 레지스터 (62) 는 새로운 데이터가 인출 데이터 레지스터 (54) 에 기록될 때 인출 데이터 레지스터 (54) 에 대응시키기 위해 인출 메시지 레지스터 (62) 내의 데이터를 자동적으로 업데이트할 수도 있다. 일부 예들에서, 메시지 전달 모듈 (50) 은 인출 메시지가 인출 메시지 레지스터 (62) 에 기록되는 것에 응답하여 인터럽트 신호를 생성할 수도 있다. 인터럽트 신호는 호스트 디바이스로 전송될 수도 있으며, 메시지 전달 모듈 (50) 이 메시지를 전송했다는 것을 나타낼 수도 있다.

[0131] 인터럽트 상태 레지스터 (64) 는, 도 2 의 예에서, 인출 메시지가 인출 메시지 레지스터 (62) 에 기록되었는지 여부를 나타내는 상태 비트를 저장하도록 구성된다. 예를 들어, 인터럽트 상태 레지스터 (64) 및 인출 메시지 레지스터 (62) 는 인출 메시지가 인출 메시지 레지스터 (62) 에 기록될 때 인터럽트 상태 레지스터 (64) 내의 상태 비트가 설정되도록 구성될 수도 있다. 상태 비트는 메시지가 이용가능한지 여부를 인식하기 위해 호스트 디바이스 상에서 실행하는 프로세스가 GPU (40) 를 폴링하는 것을 허용할 수도 있다.

[0132] 인터럽트 확인응답 레지스터 (66) 는, 도 2 의 예에서, 호스트 디바이스가 인출 메시지 레지스터 (62) 에 저장된 인출 메시지를 판독했는지 여부를 나타내는 확인응답 비트 (acknowledgement bit) 를 저장하도록 구성된다. 예를 들어, 인출 메시지 레지스터 (62) 및 인터럽트 확인응답 레지스터 (66) 는 인출 메시지가 인출 메시지 레지스터 (62) 에 기록될 때, 인터럽트 확인응답 레지스터 (66) 내의 확인응답 비트가 설정되도록 구성될 수도 있다. 이러한 예에서, 호스트 디바이스가 인출 메시지 레지스터 (62) 를 판독한 후, 호스트 디바이스는 확인응답 비트를 소거함으로써, 호스트 디바이스가 인출 메시지를 판독했고 새로운 인출 메시지가 인출 메시지 레지스터 (62) 에 기록될 수 있다는 것을 나타낼 수도 있다. 확인응답 비트는 인출 메시지 데이터에 대한 흐름 제어 방식을 구현하는데 사용될 수도 있다.

[0133] 버스 제어기 (46) 는, 도 2 의 예에서, 외부 디바이스들이 상호접속 네트워크 (18) 를 통해 호스트-액세스가능 GPU 레지스터들 (44) 에 액세스하는 것을 허용하도록 구성된다. 예를 들어, 버스 제어기 (46) 는 버스 신호들을 멀티플렉싱 및 디멀티플렉싱하고, 버스 신호들에 의해 특정된 여러 수신 및 송신 동작들을 수행할 수도 있다. 버스 제어기 (46) 는 하나 이상의 공공 (public) 또는 사유 (proprietary) 버스 표준들에 따라 동작할 수도 있다.

[0134] 이하, 다중-프로세서 컴퓨팅 시스템들에서의 메시지 전달에 대한 여러 기법들이 본 개시물의 소정의 양태들에 따라 설명될 것이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 은 도 3 내지 도 19 에 나타낸 예시적인 기법들을 구현하는데 사용될 수도 있다. 설명의 용이성을 위해, 기법들이 도 1 에 나타낸 예시적인 컴퓨팅 시스템 (10) 의 컴포넌트들에 대해 설명될 수도 있지만, 이 기법들은 동일한 또는 상이한 컴포넌트들을 가진 다른 시스템들 상에서 동일한 또는 상이한 구성으로 수행될 수도 있는 것으로 해석되어야 한다. 추가적인 예들에서, 도 3 내지 도 19 에 나타낸 기법들 중 일부는 도 2 의 GPU (40) 의 특정의 컴포넌트들에 대해 설명될 수도 있다. 또, 도 2 가 본 개시물의 기법들을 구현할 수도 있는 GPU 의 일 예일 뿐이며, 이러한 기법들이 동일한 또는 상이한 컴포넌트들을 가진 다른 GPU들에 의해 동일한 또는 상이한 구성으로 수행될 수도 있는 것으로 해석되어야 한다.

[0135] 도 3 은 본 개시물에 따른, 다중-프로세서 플랫폼 환경에서 메시지 전달을 위한 예시적인 기법을 도시한다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 은 도 3 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 커맨드 큐 인터페이스 (24) 는 메모리 전송 커맨드를 커맨드 큐 (32) 에 배치한다 (70). 커맨드 큐 인터페이스 (24) 는 태스크 실행 커맨드를 커맨드 큐 (32) 에 배치한다 (72). 커맨드 큐 인터페이스 (24) 는 GPU (14) 상에서 태스크의 실행을 개시하기 위해 태스크 실행 커맨드를 실행한다 (74). 호스트 메시지 전달 인터페이스 (26) 는 태스크 (28) 가 GPU (14) 상에서 실행중인 동안 호스트 디바이스 (12) 와 GPU (14) 사이에 하나 이상의 메시지들을 전달한다 (76). 예를 들어, 호스트 메시지 전달 인터페이스 (26) 는 호스트 프로세스 (20) 에 의해 이슈된 하나 이상의 전송 명령들로부터 기인하는 메시지를 GPU (14) 로 전달할 수도 있다. 하나 이상의 전송 명령들은 GPU (14) 또는 GPU (14) 상에서 실행하는 태스크가 그 메시지에 대한 목적지라는 것을 특정할 수도 있다.

[0136] 도 4 는 본 개시물에 따른, 호스트 디바이스 상에서 실행하는 프로세스에 의해 이슈된 전송 명령을 실행하는 예시적인 기법이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 이 도 4 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 호스트 메시지 전달 인터페이스 (26) 는 호스트 프로세스 (20) 로부터 전송 명령을

수신한다 (78). 호스트 메시지 전달 인터페이스 (26) 는 그 전송 명령에 포함된 메시지 데이터에 기초하여 인출 메시지를 생성한다 (80). 일부 예들에서, 인출 메시지는 그 전송 명령에 포함된 메시지 데이터와 동일할 수도 있다. 추가적인 예들에서, 호스트 메시지 전달 인터페이스 (26) 는 인출 메시지를 생성하기 위해, 헤더 정보 및/또는 라우팅 정보의 하나 이상의 피스들을 그 전송 명령에 포함된 메시지 데이터에 첨부할 수도 있다. 추가적인 예들에서, 호스트 메시지 전달 인터페이스 (26) 는 인출 메시지를 생성하기 위해, 그 전송 명령에 포함된 메시지 데이터에 대해 하나 이상의 코딩 또는 변환 동작들을 수행할 수도 있다. 호스트 메시지 전달 인터페이스 (26) 는 인출 메시지를 GPU (14) 로 전송할 수도 있다 (82).

[0137] 호스트 메시지 전달 인터페이스 (26) 는 전송 명령이 블록킹 명령인지 또는 넌-블록킹 명령인지 여부를 결정할 수도 있다 (84). 일부 예들에서, 호스트 메시지 전달 인터페이스 (26) 는 전송 명령에 특정된 입력 파라미터에 기초하여, 전송 명령이 블록킹 명령인지 또는 넌-블록킹 명령인지 여부의 결정을 행할 수도 있다. 다른 예들에서, 2 개의 상이한 유형들의 전송 명령들이 사용될 수도 있으며, 호스트 메시지 전달 인터페이스 (26) 는 명령의 유형, 예컨대, 명령의 연산 코드 (opcode) 에 기초하여, 전송 명령이 블록킹 명령인지 또는 넌-블록킹 명령인지 여부의 결정을 행할 수도 있다. 전송 명령이 넌-블록킹 명령이라고 호스트 메시지 전달 인터페이스 (26) 가 결정한다면, 호스트 메시지 전달 인터페이스 (26) 는 호출 프로세스로 핸들을 리턴할 수도 있다 (86). 핸들은 메시지가 추후에 성공적으로 전송되었는지를 결정하기 위해 호출 프로세스가 핸들에 대해 질의하는 것을 허용할 수도 있다. 후속 질의가 전송이 실패했다는 것을 나타낸다면, 호출 프로세스는 전송 동작을 재시도하기 위해 후속 전송 명령을 이슈할 필요가 있을 수도 있다. 일부 예들에서, 호출 프로세스는 실패된 전송 동작에 응답하여 백-오프 (back-off) 루틴 또는 오버플로우 (overflow) 메커니즘을 구현할 수도 있다.

[0138] 전송 명령이 블록킹 명령이라고 호스트 메시지 전달 인터페이스 (26) 가 결정한다면, 호스트 메시지 전달 인터페이스 (26) 는 인출 메시지가 GPU (14) 에 의해 성공적으로 수신되었는지 여부를 결정할 수도 있다 (88). 인출 메시지가 성공적으로 수신되었다고 호스트 메시지 전달 인터페이스 (26) 가 결정한다면, 호스트 메시지 전달 인터페이스 (26) 는 전송 명령에 포함된 메시지가 성공적으로 전송되었다는 것을 나타내는 값을 호출 프로세스로 리턴할 수도 있다 (90). 그렇지 않고, 인출 메시지가 성공적으로 수신되지 않았다고 호스트 메시지 전달 인터페이스 (26) 가 결정한다면, 호스트 메시지 전달 인터페이스 (26) 는 프로세스 블록 (82) 으로 진행하여 인출 메시지를 GPU (14) 로 재전송할 수도 있다. 블록킹 명령은 일부 예들에서, 메시지가 성공적으로 수신되었거나 또는 미성공 전달 시도들의 임계 개수에 도달하였다고 호스트 메시지 전달 인터페이스 (26) 가 결정할 때 완료할 수도 있다.

[0139] 도 5 는 본 개시물에 따른, 도 4 의 프로세스 블록 (82) 을 구현하는데 사용될 수도 있는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 및/또는 도 2 의 GPU (40) 가 도 5 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 호스트 메시지 전달 인터페이스 (26) 는 그 인출 메시지를 GPU (40) 의 인입 메시지 레지스터 (60) 에 배치하거나 또는 저장할 수도 있다 (92). 호스트 메시지 전달 인터페이스 (26) 는 GPU (14) 의 메시지 전달 모듈 (50) 에 새로운 메시지가 도달하였다는 것을 나타내기 위해 GPU (40) 의 메시지 카운트 레지스터 (58) 를 증분시킬 수도 있다 (94). 일부 예들에서, 호스트 메시지 전달 인터페이스 (26) 는 프로세스 블록들 (92 및 94) 중 하나 이상을 수행하기 위해 당업계에 알려진 메모리-맵핑된 레지스터 하드웨어 및/또는 I/O-맵핑된 레지스터 하드웨어를 이용할 수도 있다.

[0140] 도 6 은 본 개시물에 따른, 도 4 의 결정 블록 (88) 을 구현하는데 사용될 수도 있는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 및/또는 도 2 의 GPU (40) 가 도 6 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 호스트 메시지 전달 인터페이스 (26) 는 GPU (40) 의 메시지 상태 레지스터 (56) 내의 상태 비트를 체크할 수도 있다 (96). 호스트 메시지 전달 인터페이스 (26) 는 메시지 상태 레지스터 (56) 내의 상태 비트에 기초하여, 전송된 메시지가 GPU (14) 에 의해 수락되었는지를 결정할 수도 있다 (98). 전송된 메시지가 GPU (14) 에 의해 수락되었다고 상태 비트가 나타낸다면, 호스트 메시지 전달 인터페이스 (26) 는 인출 메시지가 성공적으로 수신되었다고 결정할 수도 있다 (100). 한편, 전송된 메시지가 GPU (14) 에 의해 수락되지 않았다고 상태 비트가 나타낸다면, 호스트 메시지 전달 인터페이스 (26) 는 인출 메시지가 성공적으로 수신되지 않았다고 결정할 수도 있다 (102).

[0141] 도 7 은 수신된 메시지를 예컨대, GPU 와 같은 컴퓨팅 디바이스에서 프로세싱하는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 2 의 GPU (40) 가 도 7 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. GPU (40) 의 메시지 전달 모듈 (50) 이 메시지 도달 신호를 수신한다 (104). 예를 들어, 메시지 카운트 레지스터 (58) 는 호스트 디바이스가 메시지 카운트 레지스터 (58) 를 증분시킬 때마다, 메시지 도달 펄

스가 메시지 전달 모듈 (50) 로 전송되도록 구성될 수도 있다. 메시지 전달 모듈 (50) 은 인입 메시지 레지스터 (60) 에 저장된 데이터로 하여금 인입 데이터 레지스터 (52) 로 전송되도록 할 수도 있다 (106). 예를 들어, 메시지 전달 모듈 (50) 은 인입 데이터 레지스터 (52) 로 하여금 인입 데이터 레지스터 (52) 에 저장된 현재의 데이터를 인입 메시지 레지스터 (60) 에 저장된 데이터에 중복 기록하도록 하는 제어 신호를 인입 데이터 레지스터 (52) 에 이슈할 수도 있다. 메시지 전달 모듈 (50) 은 데이터가 인입 데이터 레지스터 (52) 에서 이용가능하다는, 예컨대, 소비되지 않는다는 것을 나타내도록 인입 데이터 레지스터 (52) 내의 상태 비트를 설정할 수도 있다 (108). 메시지 전달 모듈 (50) 은 인입 메시지가 GPU (40) 에 의해 수락되었음을 나타내도록 메시지 상태 레지스터 (56) 내의 상태 비트를 설정할 수도 있다 (110).

[0142] 도 8 은 본 개시물에 따른, 컴퓨팅 디바이스 상에서 실행하는 태스크에 의해 이슈된 수신 명령을 실행하는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 이 도 8 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 디바이스 메시지 전달 인터페이스 (30) 는 태스크 (28) 로부터 수신 명령을 수신한다 (112). 디바이스 메시지 전달 인터페이스 (30) 는 호스트 디바이스로부터 메시지가 이용가능한지를 결정한다 (114).

[0143] 메시지가 이용가능하지 않다고 메시지 전달 모듈 (50) 이 결정한다면, 메시지 전달 모듈 (50) 은 그 수신 명령이 블록킹 수신 명령인지 또는 넌-블록킹 수신 명령인지 여부를 결정할 수도 있다 (116). 일부 예들에서, 메시지 전달 모듈 (50) 은 수신 명령에 특정된 입력 파라미터에 기초하여 수신 명령이 블록킹 명령인지 또는 넌-블록킹 명령인지 여부의 결정을 행할 수도 있다. 다른 예들에서, 2 개의 상이한 유형들의 수신 명령들이 사용될 수도 있으며, 메시지 전달 모듈 (50) 은 명령의 유형, 예컨대, 명령의 연산 코드 (opcode) 에 기초하여, 수신 명령이 블록킹 명령인지 또는 넌-블록킹 명령인지 여부의 결정을 행할 수도 있다. 수신 명령이 블록킹 명령이라고 메시지 전달 모듈 (50) 이 결정한다면, 메시지 전달 모듈 (50) 은 인입 메시지가 이용가능한지를 결정하기 위해 결정 블록 (114) 으로 리턴할 수도 있다. 그렇지 않고, 수신 명령이 넌-블록킹 명령이라고 메시지 전달 모듈 (50) 이 결정한다면, 메시지 전달 모듈 (50) 은 수신 명령이 실패했다는 것을 나타내는 값을 호출 프로세스로 리턴할 수도 있다 (118).

[0144] 메시지가 호스트 디바이스로부터 이용가능하다고 메시지 전달 모듈 (50) 이 결정한다면, 메시지 전달 모듈 (50) 은 메시지 데이터를 호출 프로세스로 리턴할 수도 있다 (120). 메시지 전달 모듈 (50) 은 메시지 데이터가 소비된 것으로 마크되어야 하는지 여부를 결정한다 (122). 메시지 전달 모듈 (50) 은 하나 이상의 소비 모드들에 기초하여, 데이터가 소비된 것으로 마크되어야 하는지 여부를 결정할 수도 있다. 일부 예들에서, 소비 모드는 GPU (14) 에 하드와이어링될 수도 있다. 추가적인 예들에서, 소비 모드는 태스크 (28) 및/또는 호스트 프로세스 (20) 에 의해 프로그래밍될 수도 있다. 예를 들어, 태스크 (28) 또는 호스트 프로세스 (20) 중 어느 하나에서의 전송 및/또는 수신 명령은 특정 소비 모드를 특정하는 파라미터를 포함할 수도 있다. 예를 들어, 하나의 소비 모드는 그 태스크의 적어도 하나의 실행 인스턴스가 데이터를 판독했을 때 메시지 데이터가 소비된 것으로 마크되어야 한다고 특정할 수도 있다. 또 다른 예로서, 하나의 소비 모드는 적어도 태스크의 임계 개수의 실행 인스턴스들이 데이터를 판독했을 때 메시지 데이터가 소비된 것으로 마크되어야 한다고 특정할 수도 있다.

[0145] 메시지 데이터가 소비된 것으로 마크되어야 한다고 메시지 전달 모듈 (50) 이 결정한다면, 메시지 전달 모듈 (50) 은 메시지 데이터를 소거할 수도 있다 (124). 예를 들어, 메시지 전달 모듈 (50) 은 인입 데이터 레지스터 (52) 내의 상태 비트를 소거할 수도 있다. 한편, 메시지 데이터가 소비된 것으로 마크되지 않아야 한다고 메시지 전달 모듈 (50) 이 결정한다면, 메시지 전달 모듈 (50) 은 메시지 데이터를 유지할 수도 있다 (126). 예를 들어, 메시지 전달 모듈 (50) 은 인입 데이터 레지스터 (52) 내의 상태 비트를 소거하지 않을 수도 있다.

[0146] 도 9 는 본 개시물에 따른, 도 8 의 결정 블록 (114) 을 구현하는데 사용될 수도 있는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 및/또는 도 2 의 GPU (40) 가 도 9 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 메시지 전달 모듈 (50) 은 GPU (40) 의 인입 데이터 레지스터 (52) 내의 상태 비트를 판독할 수도 있다 (128). 메시지 전달 모듈 (50) 은 상태 비트가 설정되는지를 결정할 수도 있다 (130). 인입 데이터 레지스터 (52) 내의 상태 비트가 설정된다면, 메시지 전달 모듈 (50) 은 인입 메시지가 이용가능하다고 결정할 수도 있다 (132). 한편, 인입 데이터 레지스터 (52) 내의 상태 비트가 설정되지 않는다면, 메시지 전달 모듈 (50) 은 인입 메시지가 이용가능하지 않다고 결정할 수도 있다 (134).

[0147] 도 10 은 본 개시물에 따른, 도 8 의 프로세스 블록 (120) 을 구현하는데 사용될 수도 있는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 및/또는 도 2 의 GPU (40) 가 도 10 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 메시지 전달 모듈 (50) 은 GPU (40) 내의 인입 데이터 레지스터 (52) 로부터 인입 메시지 데이터를 취출할 수도 있다 (136). 메시지 전달 모듈 (50) 은 인입 데이터 레지스터 (52) 로부터 취출된 메시지 데이터에 기초하여 태스크 (48) 에 대한 리턴 메시지 데이터를 생성할 수도 있다 (138). 일부 예들에서, 리턴된 메시지 데이터는 인입 데이터 레지스터 (52) 에 포함된 메시지 데이터와 동일할 수도 있다. 추가적인 예들에서, 메시지 전달 모듈 (50) 은 리턴 메시지 데이터를 생성하기 위해, 인입 데이터 레지스터 (52) 에 포함된 메시지 데이터로부터 헤더 정보 및/또는 라우팅 정보의 하나 이상의 피스들을 제거할 수도 있다. 추가적인 예들에서, 메시지 전달 모듈 (50) 은 리턴 메시지 데이터를 생성하기 위해, 인입 데이터 레지스터 (52) 에 포함된 메시지 데이터에 대해 하나 이상의 디코딩 또는 변환 동작들을 수행할 수도 있다. 메시지 전달 모듈 (50) 은 메시지 데이터를 태스크 (48) 에 제공한다 (140).

[0148] 도 11 은 본 개시물에 따른, 컴퓨팅 디바이스, 예컨대, GPU (14) 상에서 실행하는 프로세스에 의해 이슈된 전송 명령을 실행하는 예시적인 기법이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 이 도 11 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 메시지 전달 모듈 (50) 은 태스크 (28) 로부터 전송 명령을 수신한다 (142). 메시지 전달 모듈 (50) 은 그 전송 명령에 포함된 메시지 데이터에 기초하여 인출 메시지를 생성한다 (144). 일부 예들에서, 인출 메시지는 그 전송 명령에 포함된 메시지 데이터와 동일할 수도 있다. 추가적인 예들에서, 메시지 전달 모듈 (50) 은 인출 메시지를 생성하기 위해 헤더 정보 및/또는 라우팅 정보의 하나 이상의 피스들을 그 전송 명령에 포함된 메시지 데이터에 첨부할 수도 있다. 추가적인 예들에서, 메시지 전달 모듈 (50) 은 인출 메시지를 생성하기 위해, 그 전송 명령에 포함된 메시지 데이터에 대해 하나 이상의 코딩 또는 변환 동작들을 수행할 수도 있다. 메시지 전달 모듈 (50) 은 인출 메시지를 호스트 디바이스 (12) 로 전송할 수도 있다 (146).

[0149] 메시지 전달 모듈 (50) 은 전송 명령이 블록킹 명령인지 또는 넌-블록킹 명령인지 여부를 결정할 수도 있다 (148). 일부 예들에서, 메시지 전달 모듈 (50) 은 전송 명령에 특정된 입력 파라미터에 기초하여, 전송 명령이 블록킹 명령인지 또는 넌-블록킹 명령인지 여부의 결정을 행할 수도 있다. 다른 예들에서, 2 개의 상이한 유형들의 전송 명령들이 사용될 수도 있으며, 메시지 전달 모듈 (50) 은 명령의 유형, 예컨대, 명령의 연산 코드 (opcode) 에 기초하여, 전송 명령이 블록킹 명령인지 또는 넌-블록킹 명령인지 여부의 결정을 행할 수도 있다. 전송 명령이 넌-블록킹 명령이라고 메시지 전달 모듈 (50) 이 결정한다면, 메시지 전달 모듈 (50) 은 호출 프로세스, 예컨대, 태스크 (28) 로 핸들을 리턴할 수도 있다 (150). 핸들은 메시지가 추후에 성공적으로 전송되었는지를 결정하기 위해 호출 프로세스가 그 핸들에 대해 질의하는 것을 허용할 수도 있다. 전송 동작이 실패했다고 후속 질의가 나타낸다면, 호출 프로세스는 전송 동작을 재시도하기 위해 후속 전송 명령을 이슈할 필요가 있을 수도 있다.

[0150] 전송 명령이 블록킹 명령이라고 메시지 전달 모듈 (50) 이 결정한다면, 메시지 전달 모듈 (50) 은 인출 메시지가 호스트 디바이스 (12) 에 의해 성공적으로 수신되었는지 여부를 결정할 수도 있다 (152). 예를 들어, 메시지 전달 모듈 (50) 은 메시지가 수락되었는지 여부를 나타내는, 호스트 디바이스 (12) 내에 포함된 상태 레지스터를 폴링할 수도 있다. 인출 메시지가 성공적으로 수신되었다고 메시지 전달 모듈 (50) 이 결정한다면, 메시지 전달 모듈 (50) 은 전송 명령에 포함된 메시지가 성공적으로 전송되었다는 것을 나타내는 값을 호출 프로세스로 리턴할 수도 있다 (154). 그렇지 않고, 인출 메시지가 성공적으로 수신되지 않았다고 메시지 전달 모듈 (50) 이 결정한다면, 메시지 전달 모듈 (50) 은 프로세스 블록 (146) 으로 진행하여 인출 메시지를 호스트 디바이스 (12) 로 재전송할 수도 있다. 블록킹 명령은 일부 예들에서, 메시지가 성공적으로 수신되었거나 또는 미성공 전달 시도들의 임계 개수에 도달하였다고 메시지 전달 모듈 (50) 이 결정할 때 완료할 수도 있다.

[0151] 도 12 는 본 개시물에 따른, 도 11 의 프로세스 블록 (146) 을 구현하는데 사용될 수도 있는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 및/또는 도 2 의 GPU (40) 가 도 12 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 메시지 전달 모듈 (50) 은 인출 메시지를 인출 데이터 레지스터 (54) 에 배치하거나 또는 저장할 수도 있다 (156). 인출 메시지 레지스터 (62) 는 새로운 데이터가 인출 데이터 레지스터 (54) 에 배치되는 것에 응답하여, 인출 데이터 레지스터 (54) 에 대응하도록 인출 메시지 레지스터 (62) 내의 데이터를 업데이트할 수도 있다 (158). 메시지 전달 모듈 (50) 은 메시지가 GPU (40) 의 태스크 (28) 로부터 이용가능하다는 것을 나타내는 인터럽트 신호를 생성하여 호스트 디바이스 (12) 로 전송할 수도 있다 (160).

[0152] 도 13 은 본 개시물에 따른, 도 11 의 프로세스 블록 (146) 을 구현하는데 사용될 수도 있는 또다른 예시적인

기법이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 및/또는 도 2 의 GPU (40) 가 도 13 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 메시지 전달 모듈 (50) 은 인출 메시지를 인출 데이터 레지스터 (54) 에 배치하거나 또는 저장할 수도 있다 (162). 인출 메시지 레지스터 (62) 는 새로운 데이터가 인출 데이터 레지스터 (54) 에 배치되는 것에 응답하여, 인출 데이터 레지스터 (54) 에 대응하도록 인출 메시지 레지스터 (62) 내의 데이터를 업데이트할 수도 있다 (164). 메시지 전달 모듈 (50) 은 메시지가 GPU (40) 의 태스크 (28) 로부터 이용가능하다는 것을 나타내기 위해, 인터럽트 상태 레지스터 (64) 내의 상태 비트를 설정할 수도 있다. 상태 비트는 메시지가 이용가능한지를 결정하기 위해 호스트 디바이스 (12) 가 GPU (40) 를 폴링하는 것을 허용하도록 설정될 수도 있다 (166).

[0153] 도 14 는 본 개시물에 따른, 호스트 디바이스 상에서 실행하는 프로세스에 의해 이슈된 레지스터 콜백 루틴 명령을 실행하는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 이 도 14 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 호스트 메시지 전달 인터페이스 (26) 는 호스트 프로세스 (20) 로부터 레지스터 콜백 루틴 명령을 수신한다 (168). 호스트 메시지 전달 인터페이스 (26) 는 레지스터 콜백 루틴 명령에 특정된 콜백 루틴을 그 명령에 특정된 디바이스, 예컨대, GPU (14) 로부터의 인터럽트 신호와 연관시킨다 (170). 일부 예들에서, 인터럽트 신호는 특정된 디바이스 상에서 실행하는 태스크, 예컨대, GPU (14) 상에서 실행하는 태스크 (28) 가 메시지를 전송했다는 것을 나타낼 수도 있다. 인터럽트 신호는, 일부 예들에서, 호스트 디바이스 (12) 와 GPU (14) 사이에 커플링된 전용 인터럽트 신호 라인을 통해 전달될 수도 있다. 추가적인 예들에서, 인터럽트 신호는 태스크 (28) 가 메시지를 전송하는 것에 더해서, 다른 이벤트들을 나타낼 수도 있다. 이러한 예들에서, 호스트 메시지 전달 인터페이스 (26) 는 다수의 이벤트들 중 어느 이벤트가 그 신호에 의해 나타내지는지를 결정하기 위해, 인터럽트 신호를 수신한 후 추가적인 프로세싱을 수행할 수도 있다.

[0154] 호스트 메시지 전달 인터페이스 (26) 는 콜백 루틴이 인터럽트 신호와 성공적으로 연관되었는지를 결정한다 (172). 콜백 루틴이 인터럽트 신호와 성공적으로 연관되었다면, 호스트 메시지 전달 인터페이스 (26) 는 레지스터 콜백 루틴 동작이 성공적으로 완료되었다는 것을 나타내는 값을 호출 프로세스로 리턴할 수도 있다 (174). 그렇지 않고, 콜백 루틴이 인터럽트 신호와 성공적으로 연관되지 않았다면, 예컨대, 에러가 일어났다면, 호스트 메시지 전달 인터페이스 (26) 는 레지스터 콜백 루틴 동작이 실패했다는 것을 나타내는 값을 호출 프로세스로 리턴할 수도 있다 (176).

[0155] 도 15 는 본 개시물에 따른, 컴퓨팅 디바이스로부터 수신된 인터럽트를 프로세싱하는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 이 도 15 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 호스트 메시지 전달 인터페이스 (26) 는 컴퓨팅 디바이스, 예컨대, GPU (14) 로부터 인터럽트 신호를 수신한다 (178). 호스트 메시지 전달 인터페이스 (26) 는 메시지 수신 이벤트에 응답하여 인터럽트 신호가 전송되었는지를 결정한다 (180). 즉, 호스트 메시지 전달 인터페이스 (26) 는 그 디바이스 상에서 실행하는 태스크, 예컨대, GPU (14) 상에서 실행하는 태스크 (28) 가 메시지를 전송했다는 것을 인터럽트 신호가 나타내는지를 결정할 수도 있다.

[0156] 일부 예들에서, 인터럽트 신호는 메시지 수신 이벤트들을 시그널링하고 다른 이벤트들은 시그널링하지 않는 전용 인터럽트 신호일 수도 있다. 이러한 예들에서, 호스트 메시지 전달 인터페이스 (26) 는 인터럽트 신호 자체를 수신하기 때문에 메시지 수신 이벤트에 응답하여 인터럽트 신호가 전송되었으며 다른 동작들도 반드시 수행될 필요는 없다고 결정할 수도 있다. 인터럽트 신호가 복수의 퍼텐셜 이벤트들을 시그널링하는 예들에서, 호스트 메시지 전달 인터페이스 (26) 는 어느 이벤트가 시그널링되었는지를 결정하기 위해, 컴퓨팅 디바이스에 질의할 필요가 있을 수도 있다.

[0157] 메시지 수신 이벤트에 응답하여 인터럽트 신호가 전송되지 않았다고 호스트 메시지 전달 인터페이스 (26) 가 결정한다면, 호스트 메시지 전달 인터페이스 (26) 는 다른 유형들의 이벤트들을 체크할 수도 있다 (182). 그렇지 않고, 메시지 수신 이벤트에 응답하여 인터럽트 신호가 전송되었다고 호스트 메시지 전달 인터페이스 (26) 가 결정한다면, 호스트 메시지 전달 인터페이스 (26) 는 메시지가 수신된 디바이스와 연관된 콜백 루틴을 실행할 수도 있다 (184).

[0158] 도 16 은 본 개시물에 따른, 도 15 의 결정 블록 (180) 을 구현하는데 사용될 수도 있는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 및/또는 도 2 의 GPU (40) 가 도 16 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 호스트 메시지 전달 인터페이스 (26) 는 GPU (40) 내의 인터럽트 상태 레지스터 (64) 를 판독할 수도 있다 (186). 호스트 메시지 전달 인터페이스 (26) 는 새로운

메시지가 호스트 디바이스에 이용가능하다는 것을 인터럽트 상태 레지스터 (64) 내의 상태 비트가 나타내는지 여부를 결정할 수도 있다 (188). 예를 들어, 메시지 전달 모듈 (50) 은 메시지가 이용가능할 때 인터럽트 상태 레지스터 (64) 내의 상태 비트를 설정할 수도 있으며, 호스트 메시지 전달 인터페이스 (26) 는 새로운 메시지가 호스트 디바이스에 이용가능한지를 결정하기 위하여, 상태 비트가 설정되는지를 결정하도록 인터럽트 상태 레지스터 (64) 를 폴링할 수도 있다. 새로운 메시지가 호스트 디바이스에 이용가능하다는 것을 상태 비트가 나타낸다면, 호스트 메시지 전달 인터페이스 (26) 는 메시지 수신 이벤트에 응답하여 인터럽트 신호가 전송되었다고 결정할 수도 있다 (190). 한편, 새로운 메시지가 호스트 디바이스에 이용가능하지 않다고 상태 비트가 나타낸다면, 호스트 메시지 전달 인터페이스 (26) 는 메시지 수신 이벤트에 응답하여 인터럽트 신호가 전송되지 않았다고 결정할 수도 있다 (192).

[0159] 도 17 은 본 개시물에 따른, 도 15 의 프로세스 블록 (184) 을 구현하는데 사용될 수도 있는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 및/또는 도 2 의 GPU (40) 가 도 17 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 호스트 메시지 전달 인터페이스 (26) 는 GPU (40) 내의 인출 메시지 레지스터 (62) 로부터 메시지를 취출할 수도 있다 (194). 호스트 메시지 전달 인터페이스 (26) 는 인터럽트 확인응답 레지스터 (66) 내의 확인응답 비트를 소거할 수도 있다 (196). 확인응답 비트를 소거하는 것은 GPU (40) 의 흐름 제어를 도울 수도 있다. 예를 들어, GPU (40) 는 인출 메시지가 인출 메시지 레지스터 (62) 에 기록될 때 인터럽트 확인응답 레지스터 (66) 내의 확인응답 비트를 설정할 수도 있으며, 추가적인 데이터를 인출 메시지 레지스터 (62) 에 기록하기 전에 확인응답 비트가 소거될 때까지 대기할 수도 있다.

[0160] 도 18 은 본 개시물에 따른, 호스트 디바이스 상에서 실행하는 프로세스에 의해 이슈된 판독 명령을 실행하는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 이 도 18 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 호스트 메시지 전달 인터페이스 (26) 는 데이터를 판독할 특정 디바이스를 특정하는 판독 명령을 수신한다 (198). 호스트 메시지 전달 인터페이스 (26) 는 판독 명령에 특정된 디바이스를 폴링한다 (200). 호스트 메시지 전달 인터페이스 (26) 는 폴링 동작으로부터 수신된 폴링 데이터에 기초하여 메시지가 수신 명령에 특정된 디바이스로부터 이용가능한지를 결정한다 (202). 메시지가 수신 명령에 특정된 디바이스로부터 이용가능하다고 호스트 메시지 전달 인터페이스 (26) 가 결정한다면, 호스트 메시지 전달 인터페이스 (26) 는 판독 명령에 특정된 디바이스로부터 메시지를 취출할 수도 있다 (204).

일부 예들에서, 호스트 메시지 전달 인터페이스 (26) 는 호스트 디바이스 (12) 에 액세스가능한 디바이스 내의 레지스터, 예컨대, GPU (40) 내의 인출 메시지 레지스터 (62) 로부터 메시지를 취출할 수도 있다. 호스트 메시지 전달 인터페이스 (26) 는 메시지 데이터를 호출 프로세스, 예컨대, 호스트 프로세스 (20) 로 리턴할 수도 있다 (206). 메시지가 수신 명령에 특정된 디바이스로부터 이용가능하지 않다고 호스트 메시지 전달 인터페이스 (26) 가 결정한다면, 호스트 메시지 전달 인터페이스 (26) 는 판독 명령이 실패했다는 것을 나타내는 값을 리턴할 수도 있다 (208). 호출 프로세스는 판독 동작을 재시도하기 위해 다시 판독 명령을 이슈할 필요가 있을 수도 있다.

[0161] 도 19 는 본 개시물에 따른, 도 18 의 결정 블록 (202) 을 구현하는데 사용될 수도 있는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 1 의 컴퓨팅 시스템 (10) 및/또는 도 2 의 GPU (40) 가 도 19 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 호스트 메시지 전달 인터페이스 (26) 는 GPU (40) 내의 인터럽트 상태 레지스터 (64) 를 판독할 수도 있다 (210). 호스트 메시지 전달 인터페이스 (26) 는 새로운 메시지가 호스트 디바이스에 이용가능하다는 것을 인터럽트 상태 레지스터 (64) 내의 상태 비트가 나타내는지 여부를 결정할 수도 있다 (212). 예를 들어, 메시지 전달 모듈 (50) 은 메시지가 이용가능할 때 인터럽트 상태 레지스터 (64) 내의 상태 비트를 설정할 수도 있으며, 호스트 메시지 전달 인터페이스 (26) 는 새로운 메시지가 호스트 디바이스에 이용가능한지를 결정하기 위하여, 상태 비트가 설정되었는지를 결정하도록 인터럽트 상태 레지스터 (64) 를 폴링할 수도 있다. 상태 비트가 설정된다면, 호스트 메시지 전달 인터페이스 (26) 는 메시지가 이용가능하다고 결정할 수도 있다 (214). 한편, 상태 비트가 설정되지 않는다면, 호스트 메시지 전달 인터페이스 (26) 는 메시지가 이용가능하지 않다고 결정할 수도 있다 (216).

[0162] 호스트 메시지 전달 인터페이스 (26) 및 디바이스 메시지 전달 인터페이스 (30) 에 의해 구현된 메시지 전달 기법들은 호스트 디바이스 (12) 와 GPU (14) 사이에 대역외 시그널링을 제공하는 것으로 위에서 설명되었지만, 다른 예시적인 시스템들에서는, 다른 기법들이 대역외 시그널링을 제공하는데 이용될 수도 있다. 예를 들어, 일부 예들에서는, 대역외 메시지들을 전송하는데 사용될 수도 있는 특수한 높은 우선순위 큐가 정의될 수도 있다.

[0163] 도 20 은 본 개시물에 따른, 즉시 메모리 오브젝트들의 사용을 용이하게 할 수도 있는 예시적인 컴퓨팅 시스템 (310) 을 도시하는 블록도이다. 컴퓨팅 시스템 (310) 은 다수의 프로세싱 디바이스들 상에서 하나 이상의 소프트웨어 애플리케이션들을 프로세싱하도록 구성된다. 일부 예들에서, 하나 이상의 애플리케이션들은 호스트 프로세스를 포함할 수도 있으며, 컴퓨팅 시스템 (310) 은 호스트 프로세스를 실행하고 컴퓨팅 시스템 (310) 내의 다른 컴퓨팅 디바이스들 상에 호스트 프로세스에 의해 개시되는 하나 이상의 태스크들의 실행을 분산하도록 구성될 수도 있다. 추가적인 예들에서, 컴퓨팅 시스템 (310) 에 의해 실행된 호스트 프로세스 및/또는 태스크들은 병렬 프로그래밍 모델에 따라 프로그래밍될 수도 있다. 예를 들어, 애플리케이션들은 근본적인 하드웨어 시스템들의 태스크-레벨 병렬성 및/또는 데이터-레벨 병렬성을 레버리지하도록 설계되는 명령들을 포함할 수도 있다.

[0164] 컴퓨팅 시스템 (310) 은 개인용 컴퓨터, 데스크탑 컴퓨터, 랩탑 컴퓨터, 컴퓨터 워크스테이션, 비디오 게임 플랫폼 또는 콘솔, 예컨대, 셀룰러 또는 위성 전화기와 같은 모바일 전화기, 유선 전화기, 인터넷 전화기, 휴대형 비디오 게임 디바이스 또는 개인 휴대정보 단말기 (PDA) 와 같은 핸드헬드 디바이스, 개인 뮤직 플레이어, 비디오 플레이어, 디스플레이 디바이스 또는 텔레비전과 같은 디지털 미디어 플레이어, 텔레비전 셋톱 박스, 서버, 중간 네트워크 디바이스, 메인프레임 컴퓨터 또는 정보를 프로세싱하는 임의의 다른 유형의 디바이스를 포함할 수도 있다.

[0165] 컴퓨팅 시스템 (310) 은 호스트 디바이스 (312), GPU (314), 메모리 (316) 및 상호접속 네트워크 (318) 를 포함한다. 호스트 디바이스 (312) 는 호스트 프로세스의 실행을 위한 플랫폼 및 다중-프로세서 컴퓨팅 플랫폼 API 를 위한 런타임 모듈을 제공하도록 구성된다. 통상, 호스트 디바이스 (312) 는 범용 CPU 이지만, 호스트 디바이스 (312) 는 프로그램들을 실행할 수 있는 디바이스의 유형 중 임의의 유형일 수도 있다. 호스트 디바이스 (312) 는 상호접속 네트워크 (318) 를 통해 GPU (314) 및 메모리 (316) 에 통신적으로 커플링된다. 호스트 디바이스 (312) 는 호스트 프로세스 (320), 런타임 모듈 (322), 호스트 캐시 (324) 및 호스트 캐시 제어 모듈 (326) 을 포함한다. 호스트 프로세스 (320) 및 런타임 모듈 (322) 은 하나 이상의 프로그래밍가능 프로세서들의 임의의 조합 상에서 실행할 수도 있다.

[0166] 호스트 프로세스 (320) 는 컴퓨팅 시스템 (310) 플랫폼 상에서의 실행을 위한 소프트웨어 프로그램을 형성하는 명령들의 세트를 포함한다. 소프트웨어 프로그램은 최종-사용자에 대해 하나 이상의 특정의 태스크들을 수행하도록 설계될 수도 있다. 이러한 태스크들은 일부 예들에서, 컴퓨팅 시스템 (310) 에 의해 제공되는 다수의 프로세싱 디바이스들 및 병렬 아키텍처들을 활용할 수 있는 계산 집약적인 알고리즘들을 수반할 수도 있다.

[0167] 런타임 모듈 (322) 은 호스트 프로세스 (320) 에 포함된 명령들 중 하나 이상을 서비스하도록 구성된 하나 이상의 인터페이스들을 구현하는 소프트웨어 모듈일 수도 있다. 런타임 모듈 (322) 에 의해 구현된 인터페이스들은 메모리 버퍼 인터페이스 (328) 를 포함한다. 일부 예들에서, 런타임 모듈 (322) 은 메모리 버퍼 인터페이스 (328) 에 더해, 도 1 에 나타낸 커맨드 큐 인터페이스 (24) 및 도 1 에 나타낸 호스트 메시지 전달 인터페이스 (26) 중 하나 이상을 구현할 수도 있다. 추가적인 예들에서, 런타임 모듈 (322) 은 본 개시물에서 설명하는 인터페이스들에 더해, 표준 다중-프로세서 시스템 API 내에 포함된 하나 이상의 인터페이스들을 구현할 수도 있다. 일부 예들에서, 표준 API 는 이종 컴퓨팅 플랫폼 API, 크로스-플랫폼 API, 크로스-벤더 API, 병렬 프로그래밍 API, 태스크-레벨 병렬 프로그래밍 API 및/또는 데이터-레벨 병렬 프로그래밍 API 일 수도 있다. 추가적인 예들에서, 표준 API 는 OpenCL API 일 수도 있다. 이러한 예들에서, 런타임 모듈 (322) 은 OpenCL 사양들 중 하나 이상에 따르는 것으로 설계될 수도 있다. 추가적인 예들에서, 런타임 모듈 (322) 은 드라이버 프로그램, 예컨대, GPU 드라이버의 부분으로서 구현될 수도 있고, 또는 드라이버 프로그램, 예컨대, GPU 드라이버일 수도 있다.

[0168] 메모리 버퍼 인터페이스 (328) 는 호스트 프로세스 (320) 로부터 하나 이상의 메모리 오브젝트 생성 명령들을 수신하고 수신된 명령들에 의해 특정된 기능들을 실행하도록 구성된다. 일부 예들에서, 메모리 버퍼 인터페이스 (328) 는 예컨대, OpenCL API 와 같은 기존의 표준 API 에 대한 확장으로서 구현될 수도 있다. 추가적인 예들에서, 메모리 버퍼 인터페이스 (328) 는 예컨대, OpenCL API 와 같은 기존의 표준 API 에 통합될 수도 있다.

[0169] 호스트 캐시 (324) 는 호스트 디바이스 (312) 내에서 실행하는 프로세스들에 의한 사용을 위한 데이터를 저장하도록 구성된다. 일부 예들에서, 호스트 캐시 (324) 에 저장된 데이터와 연관된 메모리 공간은 메모리 (316) 내의 메모리 공간의 부분과 중첩할 수도 있다. 호스트 캐시 (324) 는 당업계에 알려진 임의의 유형의 캐시

일 수도 있다. 예를 들어, 호스트 캐시 (324) 는 캐시-레벨들 (예컨대, L1, L2, 등) 및/또는 맵핑 방식들 (예컨대, 직접 맵핑 (direct mapped), 완전-연관 (fully-associative), 세트 연관 (set associative) 등) 의 임의의 조합을 포함할 수도 있다. 호스트 캐시 제어 모듈 (326) 은 호스트 캐시 (324) 의 동작을 제어하도록 구성된다.

[0170] GPU (314) 는 호스트 디바이스 (312) 로부터 수신된 명령들에 응답하여 하나 이상의 태스크들을 실행하도록 구성된다. GPU (314) 는 하나 이상의 프로그래밍가능 프로세서들 또는 프로세싱 엘리먼트들을 포함하는 임의의 유형의 GPU 일 수도 있다. 예를 들어, GPU (314) 는 병렬로 태스크에 대한 복수의 실행 인스턴스들을 실행하도록 구성되는 하나 이상의 프로그래밍가능 셰이더 유닛들을 포함할 수도 있다. 프로그래밍가능 셰이더 유닛들은 정점 셰이더 유닛, 단편 셰이더 유닛, 지오메트리 셰이더 유닛 및/또는 통합 셰이더 유닛을 포함할 수도 있다. GPU (314) 는 상호접속 네트워크 (318) 를 통해 호스트 디바이스 (312) 및 메모리 (316) 에 통신적으로 커플링된다. GPU (314) 는 태스크 (330), GPU 캐시 (332) 및 GPU 캐시 제어 모듈 (334) 을 포함한다. 태스크 (330) 는 하나 이상의 프로그래밍가능 프로세싱 엘리먼트들의 임의의 조합 상에서 실행할 수도 있다.

[0171] 태스크 (330) 는 컴퓨팅 시스템 (310) 내의 컴퓨팅 디바이스 상에서의 실행을 위한 태스크를 형성하는 명령들의 세트를 포함한다. 일부 예들에서, 태스크 (330) 에 대한 명령들의 세트는 호스트 프로세스 (320) 에 정의될 수도 있으며, 일부 경우들에서, 호스트 프로세스 (320) 에 포함된 명령들에 의해 컴파일될 수도 있다. 추가적인 예들에서, 태스크 (330) 는 GPU (314) 상에서 병렬로 실행하는 다수의 실행 인스턴스들을 갖는 커널 프로그램일 수도 있다. 이러한 예들에서, 호스트 프로세스 (320) 는 커널 실행 인스턴스들을 실행하기 위해 커널 실행 인스턴스들을 각각의 프로세싱 엘리먼트들에 맵핑하는 커널에 대한 인덱스 공간을 정의할 수도 있으며, GPU (314) 는 그 커널에 대해 정의된 인덱스 공간에 따라 태스크 (330) 에 대한 다수의 커널 실행 인스턴스들을 실행할 수도 있다.

[0172] GPU 캐시 (332) 는 GPU (314) 내에서 실행하는 태스크들에 의한 사용을 위한 데이터를 저장하도록 구성된다. 일부 예들에서, GPU 캐시 (332) 에 저장된 데이터와 연관된 메모리 공간은 메모리 (316) 내의 메모리 공간의 부분과 중첩할 수도 있다. GPU 캐시 (332) 는 당업계에 알려진 임의의 유형의 캐시일 수도 있다. 예를 들어, GPU 캐시 (332) 는 캐시-레벨들 (예컨대, L1, L2, 등) 및/또는 맵핑 방식들 (예컨대, 직접 맵핑, 완전-연관, 세트 연관 등) 의 임의의 조합을 포함할 수도 있다. GPU 캐시 제어 모듈 (334) 은 GPU 캐시 (332) 의 동작을 제어하도록 구성된다.

[0173] 메모리 (316) 는 호스트 디바이스 (312) 및 GPU (314) 중 하나 또는 양자에 의한 사용을 위한 데이터를 저장하도록 구성된다. 메모리 (316) 는 예를 들어, 랜덤 액세스 메모리 (RAM), 정적 RAM (SRAM), 동적 RAM (DRAM), 판독 전용 메모리 (ROM), 소거가능한 프로그래밍가능 ROM (EPROM), 전기적으로 소거가능한 프로그래밍가능 ROM (EEPROM), 플래시 메모리, 자기 데이터 저장 매체들 또는 광학 저장 매체들과 같은, 하나 이상의 휘발성 또는 비휘발성 메모리들 또는 저장 디바이스들의 임의의 조합을 포함할 수도 있다. 메모리 (316) 는 상호접속 네트워크 (318) 를 통해 호스트 디바이스 (312) 및 GPU (314) 에 통신적으로 커플링된다. 메모리 (316) 는 공유 메모리 공간 (336) 을 포함한다. 공유 메모리 공간 (336) 은 호스트 디바이스 (312) 와 GPU (314) 양자에 의해 액세스가능한 메모리 공간일 수도 있다.

[0174] 상호접속 네트워크 (318) 는 호스트 디바이스 (312), GPU (314) 및 메모리 (316) 사이에 통신을 용이하게 하도록 구성된다. 상호접속 네트워크 (318) 는 당업계에 알려진 임의의 유형의 상호접속 네트워크일 수도 있다. 도 20 의 예시적인 컴퓨팅 시스템 (310) 에서, 상호접속 네트워크 (318) 는 버스이다. 버스는 임의의 다양한 버스 구조들, 예컨대, 3 세대 버스 (예컨대, HyperTransport 버스 또는 InfiniBand 버스), 2 세대 버스 (예컨대, 진보된 그래픽들 포트 버스, PCIe (Peripheral Component Interconnect Express) 버스, 또는 AXI (Advanced eXentisible Interface) 버스) 또는 임의의 다른 유형의 버스 중 하나 이상을 포함할 수도 있다. 상호접속 네트워크 (318) 는 호스트 디바이스 (312), GPU (314) 및 메모리 (316) 에 커플링된다.

[0175] 이하, 컴퓨팅 시스템 (310) 내의 컴포넌트들의 구조들 및 기능성들이 더 상세히 설명될 것이다. 위에서 설명한 바와 같이, 호스트 프로세스 (320) 는 명령들의 세트를 포함한다. 명령들의 세트는 예를 들어, 하나 이상의 메모리 오브젝트 생성 명령들을 포함할 수도 있다. 추가적인 예들에서, 명령들의 세트는 GPU (314) 상에서 실행될 태스크들 또는 커널들을 특정하는 명령들, 커맨드 큐들을 생성하여 커맨드 큐들을 특정 디바이스들과 연관시키는 명령들, 프로그램들을 컴파일하고 바인딩하는 명령들, 커널 파라미터들을 셋업하는 명령들, 인덱스 공간들을 정의하는 명령들, 디바이스 콘텍스트를 정의하는 명령들, 인큐 명령들, 메시지 전달 명령들 및

호스트 프로세스 (320) 에 의해 제공되는 기능성을 지원하는 다른 명령들을 포함할 수도 있다.

[0176] 본 개시물에 따르면, 호스트 프로세스 (320) 는 메모리 오브젝트에 대해 즉시 모드가 인에이블되는지 여부를 특정하는 명령에 포함된 정보에 기초하여, 메모리 오브젝트를 생성하도록 메모리 버퍼 인터페이스 (328) 에 명령하는 하나 이상의 메모리 오브젝트 생성 명령들을 메모리 버퍼 인터페이스 (328) 에 이슈함으로써, 메모리 버퍼 인터페이스 (328) 와 상호작용할 수도 있다. 본원에서 사용한 바와 같이, 메모리 오브젝트는 GPU (314) 에 의해 액세스가능한 메모리 공간의 영역을 나타내는 소프트웨어 오브젝트를 지칭할 수도 있다. 일부 예들에서, 메모리 공간의 영역은 또한 호스트 디바이스 (312) 에 의해 액세스가능할 수도 있다. 메모리 오브젝트는 메모리 오브젝트와 연관된 메모리 공간에 포함된 데이터를 포함할 수도 있다. 메모리 오브젝트는 메모리 공간과 연관된 하나 이상의 특성들을 더 포함할 수도 있다. 일부 예들에서, 메모리 오브젝트는 글로벌 메모리, 예컨대, 메모리 (316) 의 참조 카운트 영역에 대한 핸들을 포함할 수도 있다.

[0177] 메모리 오브젝트들은 버퍼 오브젝트들 및 이미지 오브젝트들을 포함할 수도 있다. 버퍼 오브젝트는 바이트들의 1 차원 컬렉션을 저장하는 메모리 오브젝트일 수도 있다. 바이트들의 1 차원 컬렉션은 메모리 오브젝트와 연관되는 데이터일 수도 있다. 버퍼 오브젝트는 또한 예컨대, 버퍼 오브젝트와 연관되는 메모리 공간의 바이트 (bytes) 단위 사이즈, 버퍼 오브젝트에 대한 사용 정보, 및 버퍼 오브젝트에 할당된 메모리 공간의 영역과 같은, 정보를 포함할 수도 있다. 이미지 오브젝트는 예컨대, 텍스쳐, 프레임-버퍼 또는 이미지와 같은 데이터의 2 차원 또는 3 차원 어레이를 저장한다. 이미지 오브젝트는 또한 예컨대, 이미지의 디멘젼들, 이미지 내의 각각의 엘리먼트의 설명, 이미지 오브젝트에 대한 사용 정보, 및 이미지 오브젝트에 할당된 메모리 공간의 영역과 같은 정보를 포함할 수도 있다.

[0178] 본 개시물의 일부 양태들에 따르면, 메모리 오브젝트 생성 명령들은 생성되는 메모리 오브젝트에 대해 즉시 모드가 인에이블되어야 하는지 여부를 특정하는 입력 파라미터를 포함할 수도 있다. 본원에서 더 상세히 설명하는 바와 같이, 즉시 모드가 인에이블될 때, 메모리 오브젝트는 캐시불가능한 공유 메모리로서 및/또는 캐시-코히런트 공유 메모리로서 구현될 수도 있다. 즉시 모드가 디스에이블될 때, 메모리 오브젝트는 반드시 캐시불가능한 공유 메모리로서 또는 캐시-코히런트 공유 메모리로서 구현될 필요가 없을 수도 있다.

[0179] 일부 예들에서, 메모리 오브젝트는 메모리 오브젝트가 즉시 모드 메모리 오브젝트인지 여부를 나타내는 즉시 모드 속성을 포함할 수도 있다. 이러한 예들에서, 메모리 버퍼 인터페이스 (328) 는 메모리 오브젝트에 대해 즉시 모드가 인에이블되어야 하는지 여부를 특정하는 정보에 기초하여, 생성되는 메모리 오브젝트에 대한 즉시 모드 속성을 메모리 오브젝트에 대해 즉시 모드가 인에이블되는지 여부를 나타내는 값으로 설정하도록 구성될 수도 있다. 메모리 오브젝트의 즉시 모드 속성은 메모리 오브젝트를 캐시불가능한 공유 메모리로서 및/또는 캐시-코히런트 공유 메모리로서 구현할지 여부를 결정하기 위해 컴퓨팅 시스템 (310) 에 의해 사용될 수도 있다.

[0180] 메모리 오브젝트 생성 명령들은 일부 예들에서, 버퍼 오브젝트에 대해 즉시 모드가 인에이블되는지 여부를 특정하는 명령에 포함된 정보에 기초하여, 버퍼 오브젝트를 생성하도록 메모리 버퍼 인터페이스 (328) 에 명령하는 버퍼 오브젝트 생성 명령을 포함할 수도 있다. 메모리 오브젝트 생성 명령들은 추가적인 예들에서, 이미지 오브젝트에 대해 즉시 모드가 인에이블되는지 여부를 특정하는 명령에 포함된 정보에 기초하여, 이미지 오브젝트를 생성하도록 메모리 버퍼 인터페이스 (328) 에 명령하는 이미지 오브젝트 생성 명령을 포함할 수도 있다.

[0181] 일부 예들에서, 버퍼 오브젝트 생성 명령에 대한 인터페이스는 다음의 형태를 취할 수도 있다:

[0182]
```
cl_mem clCreateBuffer (
```

[0183]
```
        cl_context context,
```

[0184]
```
        cl_mem_flags flags,
```

[0185]
```
        size_t size,
```

[0186]
```
        void *host_ptr,
```

[0187]
```
        cl_int *errcode_ret)
```

[0188] 여기서, clCreateBuffer 는 명령 식별자이고, cl_context context 는 버퍼 오브젝트를 생성하는데 사용되는 유효한 콘텍스트, 예컨대, OpenCL 콘텍스트이고, cl_mem_flags flags 는 버퍼 오브젝트에 대한 할당 및 사용 정보를 특정하는데 사용되는 비트-필드이고, size_t size 는 할당되는 버퍼 메모리 오브젝트의 바이트 단위 사이즈

를 특정하는 파라미터이고, void *host_ptr 은 애플리케이션에 의해 이미 할당될 수도 있는 버퍼 데이터에 대한 포인터이며, cl_int *errcode_ret 은 하나 이상의 에러 코드들을 리턴한다. 이 명령은 생성된 버퍼 오브젝트를 cl_mem 메모리 오브젝트로서 리턴할 수도 있다. 이 예에서, 이미지 오브젝트에 대해 즉시 모드가 인에이블되어야 하는지 여부를 특정하는 입력 파라미터는 예컨대, cl_mem_flags flags 필드에 특정되는 CL_IMMEDIATE 플래그일 수도 있다.

[0189] 추가적인 예들에서, 이미지 오브젝트 생성 명령에 대한 인터페이스는 다음의 형태를 취할 수도 있다:

[0190] cl_mem clCreateImage2D (

[0191] cl_context context,

[0192] cl_mem_flags flags,

[0193] const cl_image_format *image_format,

[0194] size_t image_width,

[0195] size_t image_height,

[0196] size_t image_row_pitch,

[0197] void *host_ptr,

[0198] cl_int *errcode_ret)

[0199] 여기서, clCreateImage2D 는 명령 식별자이고, cl_context context 는 버퍼 오브젝트를 생성하는데 사용되는 유효한 콘텍스트, 예컨대, OpenCL 콘텍스트이고, cl_mem_flags flags 는 이미지 오브젝트에 대한 할당 및 사용 정보를 특정하는데 사용되는 비트-필드이고, const cl_image_format *image_format 은 할당되는 이미지의 포맷 성질들을 기술하는 구조에 대한 포인터이고, size_t image_width 는 픽셀 단위의 이미지의 폭이고, size_t image_height 는 픽셀 단위의 이미지의 높이이고, size_t image_row_pitch 는 바이트 단위의 스캔-라인 피치이고, void *host_ptr 은 애플리케이션에 의해 이미 할당될 수도 있는 이미지 데이터에 대한 포인터이며, cl_int *errcode_ret 은 하나 이상의 에러 코드들을 리턴한다. 이 명령은 생성된 이미지 오브젝트를 cl_mem 메모리 오브젝트로서 리턴할 수도 있다. 이 예에서, 이미지 오브젝트에 대해 즉시 모드가 인에이블되어야 하는지 여부를 특정하는 입력 파라미터는 예컨대, cl_mem_flags flags 필드에 특정되는 CL_IMMEDIATE 플래그일 수도 있다.

[0200] 일부 예들에서, 메모리 오브젝트 생성 인터페이스들은 판독/기록 속성들의 관점에서 단지 WRITE_ONLY 속성 또는 READ_ONLY 속성 중 어느 하나만을 허용하도록 구성될 수도 있다. 즉, 이러한 예들에서, 메모리 버퍼 인터페이스 (328) 는 READ_WRITE 속성을 허용하지 않을 수도 있다. 비-즉시 CL 이미지들은 OpenCL 사양에 의해 제공된 이러한 피쳐를 이미 가질 수도 있다. READ_WRITE 속성을 허용하지 않는 것은 캐시 코히런시를 유지하는데 있어 복잡성을 감소시킬 수도 있다.

[0201] 본 개시물에 따르면, 메모리 버퍼 인터페이스 (328) 는 호스트 디바이스 (312) 와 GPU (314) 양자에 의해 액세스가능한 공유 메모리 공간 (336) 에 대해 즉시 모드가 인에이블되어야 하는지 여부를 특정하는 명령을 수신하고, 공유 메모리 공간 (336) 에 대해 즉시 모드가 인에이블되어야 하는지 여부를 특정하는 수신된 명령에 기초하여, 공유 메모리 공간 (336) 에 대해 즉시 모드를 선택적으로 인에이블하도록 구성된다. 예를 들어, 메모리 버퍼 인터페이스 (328) 는, 공유 메모리 공간 (336) 에 대해 즉시 모드가 인에이블되어야 한다고 그 명령이 특정한다면, 공유 메모리 공간 (336) 에 대해 즉시 모드를 인에이블하고, 공유 메모리 공간 (336) 에 대해 즉시 모드가 디스에이블되어야 한다고 그 명령이 특정한다면, 공유 메모리 공간 (336) 에 대해 즉시 모드를 디스에이블할 수도 있다. 이 명령은 예를 들어, 메모리 오브젝트 생성 명령, 버퍼 오브젝트 생성 명령 또는 이미지 오브젝트 생성 명령 중 하나일 수도 있다. 공유 메모리 공간 (336) 은 예를 들어, 메모리 오브젝트, 버퍼 오브젝트 또는 이미지 오브젝트에 대응할 수도 있다.

[0202] 일부 예들에서, 메모리 버퍼 인터페이스 (328) 가 공유 메모리 공간 (336) 에 대해 즉시 모드를 인에이블할 때, 메모리 버퍼 인터페이스 (328) 는 공유 메모리 공간 (336) 에 대한 캐싱 서비스들이 디스에이블되게 할 수도 있다. 이와 유사하게, 메모리 버퍼 인터페이스 (328) 가 공유 메모리 공간 (336) 에 대해 즉시 모드를 디스에이블할 때, 메모리 버퍼 인터페이스 (328) 는 공유 메모리 공간 (336) 에 대한 캐싱 서비스들이 공유 메모리 공간 (336) 에 대해 인에이블되도록 할 수도 있다. 캐싱 서비스들은 호스트 캐시 (324) 및 GPU 캐시 (332) 중

하나 또는 양자에 의해 수행될 수도 있다. 캐싱 서비스들은, 본원에서 사용한 바와 같이, 당업계에 알려진 캐시에 의해 통상 수행되는 서비스들을 지칭할 수도 있다.

[0203] 추가적인 예들에서, 메모리 버퍼 인터페이스 (328) 는 공유 메모리 공간 (336) 과 연관된 즉시 모드 속성을 공유 메모리 공간에 대해 즉시 모드가 인에이블되는지 여부를 나타내는 값으로 설정함으로써, 공유 메모리 공간 (336) 에 대해 즉시 모드를 인에이블 및 디스에이블할 수도 있다. 예를 들어, 메모리 버퍼 인터페이스 (328) 는 공유 메모리 공간 (336) 과 연관된 즉시 모드 속성을 공유 메모리 공간 (336) 에 대해 즉시 모드가 인에이블된다는 것을 나타내는 값으로, 예컨대, 즉시 모드 속성 = 참 (true) 으로 설정함으로써, 공유 메모리 공간 (336) 에 대해 즉시 모드를 인에이블할 수도 있다. 이와 유사하게, 메모리 버퍼 인터페이스 (328) 는 공유 메모리 공간 (336) 과 연관된 즉시 모드 속성을 공유 메모리 공간 (336) 에 대해 즉시 모드가 디스에이블된다는 것을 나타내는 값으로, 예컨대, 즉시 모드 속성 = 거짓 (false) 으로 설정함으로써, 공유 메모리 공간 (336) 에 대해 즉시 모드를 디스에이블할 수도 있다. 즉시 모드 속성은 일부 경우들에서, GPU (314) 상에서 실행하는 태스크 (330) 에 의해 액세스가능한 글로벌 변수, 예컨대, 부울 변수 (boolean variable) 일 수도 있다. 일부 예들에서, 즉시 모드 속성은 공유 메모리 공간 (336) 내에 저장될 수도 있다. 다른 예들에서, 즉시 모드 속성은 공유 메모리 공간 (336) 과는 다른 GPU (314) 상에서 실행하는 태스크 (330) 에 의해 액세스가능한 로케이션에 저장될 수도 있다. 공유 메모리 공간 (336) 이 메모리 오브젝트의 부분인 경우들에서, 즉시 모드 속성은 메모리 오브젝트의 다른 속성들이 저장되는 메모리 공간의 로케이션에 저장될 수도 있다.

[0204] 공유 메모리 공간 (336) 과 연관된 즉시 모드 속성을 설정함으로써 메모리 버퍼 인터페이스 (328) 가 공유 메모리 공간 (336) 에 대해 즉시 모드를 인에이블 및 디스에이블하는 예들에서, 일부의 경우, 태스크 (330) 에 대한 소스 코드는, 공유 메모리 공간 (336) 에 대해 메모리 판독 또는 기록 동작을 수행하기 전에, 태스크 (330) 가 공유 메모리 공간 (336) 과 연관된 즉시 모드 속성에 액세스하여 공유 메모리 공간 (336) 에 대한 즉시 모드 속성에 기초하여 공유 메모리 공간 (336) 에 대해 즉시 모드가 인에이블되는지 여부를 결정하도록 컴파일될 수도 있다. 공유 메모리 공간 (336) 에 대해 즉시 모드가 인에이블된다면, 태스크 (330) 는 공유 메모리 공간 (336) 으로부터 데이터를 판독하거나 또는 공유 메모리 공간 (336) 에 데이터를 기록하기 위해 즉시 모드 판독 또는 기록 명령을 실행하도록 프로그래밍될 수도 있다. 한편, 공유 메모리 공간에 대해 즉시 모드가 인에이블되지 않는다면, 태스크 (330) 는 공유 메모리 공간 (336) 으로부터 데이터를 판독하거나 또는 공유 메모리 공간 (336) 에 데이터를 기록하기 위해 캐시 모드 판독 또는 기록 명령, 예컨대, 캐시 판독 또는 기록 명령을 실행하도록 프로그래밍될 수도 있다.

[0205] 즉시 모드 판독 및 기록 명령들은 예를 들어, 캐싱 서비스들을 이용하지 않고, 판독 및 기록 동작들을 각각 수행할 수도 있다. 예를 들어, 즉시 모드 판독 명령은 판독 동작을 수행하기 전에 캐시가 무효화되게 할 수도 있거나, 및/또는 판독 동작을 수행할 때 캐시를 바이패스할 수도 있다. 즉시 모드 기록 명령은, 예를 들어, 기록 동작을 수행할 때 캐시가 즉시 후기록 (immediate write-back) 을 수행하게 할 수도 있거나, 및/또는 기록 동작을 수행할 때 캐시를 바이패스할 수도 있다. 캐시 판독 및 기록 명령들은 예를 들어, GPU 캐시 (332) 중 하나 또는 양자의 캐싱 서비스들을 이용하여 판독 및 기록 동작들을 각각 실행할 수도 있다.

[0206] 추가적인 경우들에서, 태스크 (330) 에 대한 컴파일러는 태스크 (330) 에 대한 소스 코드를 컴파일할 때, 공유 메모리 공간 (336) 에 대해 즉시 모드가 인에이블되는지 여부를 나타내는 정보에 액세스할 수도 있다. 예를 들어, 태스크 (330) 에 대한 소스 코드, 예컨대, 커널 소스 코드는, 태스크 (330) 에 의해 사용되고 공유 메모리 공간 (336) 과 연관되는 메모리 오브젝트에 대해 즉시 모드가 인에이블되는지 여부를 나타내는 플래그를 포함할 수도 있다. 일부 예들에서, 플래그는 예컨대, _cl_immediate 속성 수식자 (attribute qualifier) 와 같은 OpenCL 속성 수식자의 형태를 취할 수도 있다. 공유 메모리 공간 (336) 과 연관된 메모리 오브젝트에 대해 즉시 모드가 인에이블된다면, 컴파일러는 태스크 (330) 에 대한 컴파일된 코드가 공유 메모리 공간 (336) 에 대해 일어나는 판독 또는 기록 동작들에 대한 즉시 모드 판독 및/또는 기록 명령들을 포함하도록 태스크 (330) 를 컴파일할 수도 있다. 그렇지 않고, 공유 메모리 공간 (336) 과 연관된 메모리 오브젝트에 대해 즉시 모드가 인에이블되지 않는다면, 컴파일러는 태스크 (330) 에 대한 컴파일된 코드가 공유 메모리 공간 (336) 에 대해 일어나는 판독 또는 기록 동작들에 대한 즉시 모드 판독 및/또는 기록 명령들을 포함하지 않도록 태스크 (330) 를 컴파일할 수도 있다. 예를 들어, 컴파일러는, 태스크 (330) 에 대한 컴파일된 코드가 공유 메모리 공간 (336) 에 대해 일어나는 판독 또는 기록 동작들에 대한 캐시 판독 및/또는 기록 명령들을 포함하도록 태스크 (330) 를 컴파일할 수도 있다.

[0207] 추가적인 예들에서, 메모리 버퍼 인터페이스 (328) 는 호스트 디바이스 (312) 내의 호스트 캐시 (324) 및 GPU (314) 내의 GPU 캐시 (332) 중 적어도 하나에 의해 공유 메모리 공간 (336) 에 대한 캐싱 서비스들의 수행을 인

에이블 및 디스에이블함으로써, 공유 메모리 공간 (336) 에 대해 즉시 모드를 인에이블 및 디스에이블할 수도 있다. 예를 들어, 메모리 버퍼 인터페이스 (328) 는 호스트 디바이스 (312) 내의 호스트 캐시 (324) 및 GPU (314) 내의 GPU 캐시 (332) 중 적어도 하나에 의해 공유 메모리 공간 (336) 에 대한 캐싱 서비스들의 수행을 디스에이블함으로써, 공유 메모리 공간 (336) 에 대해 즉시 모드를 인에이블할 수도 있다. 이와 유사하게, 메모리 버퍼 인터페이스 (328) 는 호스트 디바이스 (312) 내의 호스트 캐시 (324) 및 GPU (314) 내의 GPU 캐시 (332) 중 적어도 하나에 의해 공유 메모리 공간 (336) 에 대한 캐싱 서비스들의 수행을 인에이블함으로써, 공유 메모리 공간 (336) 에 대해 즉시 모드를 디스에이블할 수도 있다.

[0208] 이러한 예들에서, 메모리 버퍼 인터페이스 (328) 는 공유 메모리 공간 (336) 에 대한 캐싱 서비스들을 수행하는 캐시와 연관된 하드웨어-기반 캐시 제어 모듈 및/또는 하드웨어-기반 메모리 관리 유닛을 구성함으로써, 공유 메모리 공간 (336) 에 대한 캐싱 서비스들의 수행을 인에이블 및 디스에이블할 수도 있다. 예를 들어, 호스트 캐시 (324) 에 의해 공유 메모리 공간 (336) 에 대한 캐싱 서비스들의 수행을 인에이블하기 위해, 메모리 버퍼 인터페이스 (328) 는 캐싱 서비스들이 공유 메모리 공간 (336) 에 대해 호스트 캐시 (324) 에 의해 제공되도록 호스트 캐시 제어 모듈 (326) 을 구성할 수도 있다. 호스트 캐시 (324) 에 의해 공유 메모리 공간 (336) 에 대한 캐싱 서비스들의 수행을 디스에이블하기 위해, 메모리 버퍼 인터페이스 (328) 는 예를 들어, 캐싱 서비스들이 공유 메모리 공간 (336) 에 대해 호스트 캐시 (324) 에 의해 제공되지 않도록 호스트 캐시 제어 모듈 (326) 을 구성할 수도 있다. 이와 유사하게, GPU 캐시 (332) 에 의해 공유 메모리 공간 (336) 에 대한 캐싱 서비스들의 수행을 인에이블하기 위해, 메모리 버퍼 인터페이스 (328) 는 예를 들어, 캐싱 서비스들이 공유 메모리 공간 (336) 에 대해 호스트 캐시 (324) 에 의해 제공되도록 GPU 캐시 제어 모듈 (334) 을 구성할 수도 있다. GPU 캐시 (332) 에 의해 공유 메모리 공간 (336) 에 대한 캐싱 서비스들의 수행을 디스에이블하기 위해, 메모리 버퍼 인터페이스 (328) 는 예를 들어, 캐싱 서비스들이 공유 메모리 공간 (336) 에 대해 GPU 캐시 (332) 에 의해 제공되지 않도록 GPU 캐시 제어 모듈 (334) 을 구성할 수도 있다.

[0209] 일부 예들에서, 메모리 버퍼 인터페이스 (328) 는 공유 메모리 공간 (336) 과 연관된 하나 이상의 하드웨어-기반 즉시 플래그들을 공유 메모리 공간 (336) 에 대해 캐싱 서비스들이 제공되어야 하는지 여부를 나타내는 값으로 설정함으로써, 호스트 캐시 제어 모듈 (326) 및 GPU 캐시 제어 모듈 (334) 중 하나 또는 양자를 구성할 수도 있다. 하나 이상의 하드웨어-기반 즉시 플래그들은 일부 예들에서, 하나 이상의 레지스터들일 수도 있다.

추가적인 예들에서, 하드웨어-기반 즉시 플래그들은 즉시 플래그들의 테이블에서의 각각의 즉시 플래그가 메모리 (316) 내의 특정 어드레스 공간에 대응하는 즉시 플래그들의 테이블의 부분일 수도 있다. 어느 경우나, 캐싱 서비스들이 제공되어야 한다는 것을 나타내는 값들로 공유 메모리 공간 (336) 과 연관된 하나 이상의 즉시 플래그들이 설정될 때, 호스트 캐시 제어 모듈 (326) 및/또는 GPU 캐시 제어 모듈 (334) 은 호스트 캐시 (324) 및/또는 GPU 캐시 (332) 를 이용하여 공유 메모리 공간 (336) 에 대한 캐싱 서비스들을 제공할 수도 있다. 이와 유사하게, 캐싱 서비스들이 제공되지 않아야 한다는 것을 나타내는 값들로 공유 메모리 공간 (336) 과 연관된 하나 이상의 플래그들이 설정될 때, 호스트 캐시 제어 모듈 (326) 및/또는 GPU 캐시 제어 모듈 (334) 은 공유 메모리 공간 (336) 에 대한 캐싱 서비스들을 제공하지 않을 수도 있다.

[0210] 이러한 예들에서, GPU 캐시 제어 모듈 (334) 은 메모리 (316) 의 어드레스 공간 내의 메모리 어드레스들에 대해 판독 명령들 및/또는 기록 명령들을 프로세싱하도록 구성될 수도 있다. 판독 및 기록 명령들은 예를 들어, GPU (314) 상에서 실행하는 태스크 (330) 에 의해 GPU 캐시 제어 모듈 (334) 에 이슈될 수도 있다. 메모리 (316) 의 주어진 어드레스 공간 내의 메모리 로케이션으로부터 데이터를 판독하고 메모리 (316) 의 주어진 어드레스 공간 내의 메모리 로케이션에 데이터를 기록하기 위해 판독 또는 기록 명령을 수신하는 것에 응답하여, GPU 캐시 제어 모듈 (334) 은 어드레스 공간과 연관된 하드웨어-기반 플래그를 식별하고, 하드웨어-기반 플래그의 값에 기초하여, 판독 또는 기록 명령을 프로세싱할 때 GPU 캐시 (332) 의 캐싱 서비스들을 이용할지 여부를 결정할 수도 있다. GPU 캐시 제어 모듈 (334) 이 GPU 캐시 (332) 의 캐싱 서비스들을 이용하기로 결정한다면, GPU 캐시 제어 모듈 (334) 은 예를 들어, 그 데이터가 유효하다면 GPU 캐시 (332) 로부터 데이터를 판독하고 및/또는 GPU 캐시 (332) 에 데이터를 기록하려고 시도할 수도 있다. GPU 캐시 제어 모듈 (334) 이 GPU 캐시 (332) 의 캐싱 서비스들을 이용하지 않기로 결정한다면, GPU 캐시 제어 모듈 (334) 은 일부 예들에서, GPU 캐시 (332) 를 바이패스하고, 직접 메모리 (316) 로부터 데이터를 판독하거나 메모리 (316) 에 데이터를 기록할 수도 있다. 추가적인 예들에서, GPU 캐시 제어 모듈 (334) 이 GPU 캐시 (332) 의 캐싱 서비스들을 이용하지 않기로 결정한다면, GPU 캐시 제어 모듈 (334) 은 판독 명령을 실행하기 전에 어드레스 공간과 연관되는 캐시 (332) 의 부분을 무효화하거나, 및/또는 기록 명령을 실행할 때 캐시 후기록 또는 캐시 연속기록 (write-through) 기법을 수행할 수도 있다. 호스트 캐시 제어 모듈 (334) 은 호스트 디바이스 (312) 상에서 실행하는 호스트 프로세스 (320) 로부터 수신된 판독 및 기록 명령들에 응답하여, 호스트 캐시 (324) 에 대해 유사한

방식으로 동작할 수도 있다.

[0211] 추가적인 예들에서, 메모리 버퍼 인터페이스 (328) 는 호스트 디바이스 (312) 내의 호스트 캐시 (324) 및 GPU (314) 내의 GPU 캐시 (332) 중 적어도 하나에 대해 공유 메모리 캐시 코히런시 모드를 인에이블 및 디스에이블함으로써, 공유 메모리 공간 (336) 에 대해 즉시 모드를 인에이블 및 디스에이블할 수도 있다. 예를 들어, 공유 메모리 공간 (336) 에 대해 즉시 모드를 인에이블하기 위해, 메모리 버퍼 인터페이스 (328) 는 호스트 디바이스 (312) 내의 호스트 캐시 (324) 및 GPU (314) 내의 GPU 캐시 (332) 중 적어도 하나에 대해 공유 메모리 캐시 코히런시 모드를 인에이블할 수도 있다. 이와 유사하게, 공유 메모리 공간 (336) 에 대해 즉시 모드를 디스에이블하기 위해, 메모리 버퍼 인터페이스 (328) 는 호스트 디바이스 (312) 내의 호스트 캐시 (324) 및 GPU (314) 내의 GPU 캐시 (332) 중 적어도 하나에 대해 공유 메모리 캐시 코히런시 모드를 디스에이블할 수도 있다.

이러한 예들에서, 메모리 버퍼 인터페이스 (328) 는 일부 경우들에, 공유 메모리 캐시 코히런시 모드를 인에이블하도록 호스트 캐시 제어 모듈 (326) 및 GPU 캐시 제어 모듈 (334) 중 하나 또는 양자를 구성함으로써 호스트 캐시 (324) 에 대해 공유 메모리 캐시 코히런시 모드를 인에이블하고, 공유 메모리 캐시 코히런시 모드를 디스에이블하도록 호스트 캐시 제어 모듈 (326) 및 GPU 캐시 제어 모듈 (334) 중 하나 또는 양자를 구성함으로써 호스트 캐시 (324) 에 대해 공유 메모리 캐시 코히런시 모드를 디스에이블할 수도 있다.

[0212] 호스트 캐시 (324) 에 대한 공유 메모리 캐시 코히런시 모드가 인에이블될 때, 호스트 캐시 제어 모듈 (326) 은 기지의 방법들에 따라 공유 메모리 공간 (336) 에 대해 공유 메모리 캐시 코히런시 기법들을 수행할 수도 있다.

호스트 캐시 (324) 에 대한 공유 메모리 캐시 코히런시 모드가 디스에이블될 때, 호스트 캐시 (324) 는 공유 메모리 공간 (336) 에 대해 공유 메모리 캐시 코히런시 기법들을 수행하지 않을 수도 있다. 이와 유사하게, GPU 캐시 (332) 에 대한 공유 메모리 캐시 코히런시 모드가 인에이블될 때, GPU 캐시 제어 모듈 (334) 은 기지의 방법들에 따라 공유 메모리 공간 (336) 에 대해 공유 메모리 캐시 코히런시 기법들을 수행할 수도 있다. GPU 캐시 (332) 에 대한 공유 메모리 캐시 코히런스 모드가 디스에이블될 때, GPU 캐시 제어 모듈 (334) 은 공유 메모리 공간 (336) 에 대해 공유 메모리 캐시 코히런시 기법들을 수행하지 않을 수도 있다.

[0213] 예시의 용이성을 위해, 도 20 에 도시된 예시적인 컴퓨팅 시스템 (310) 은 GPU (314) 를 컴퓨팅 디바이스로서 이용하는 본 개시물의 즉시 버퍼링 기법들을 설명한다. 본 개시물의 기법들이 GPU (314) 에 더하여 또는 대신에 GPU 와는 다른 컴퓨팅 디바이스들을 갖는 다중-프로세서 컴퓨팅 시스템들에 적용될 수도 있는 것으로 인식되어야 한다. 일부 예들에서, 컴퓨팅 시스템 (310) 내의 컴퓨팅 디바이스들은 OpenCL 계산 디바이스들일 수도 있다. 게다가, 도 20 에 나타낸 예시적인 컴퓨팅 시스템 (310) 은 호스트 디바이스와 컴퓨팅 디바이스 사이에 인-플라이트 데이터 공유를 용이하게 하는 즉시 메모리 오브젝트들을 구현하는 인프라스트럭처 및 기법들을 도시한다. 그러나, 다른 예시적인 컴퓨팅 시스템들에서, 이 기법들은 하나 보다 많은 컴퓨팅 디바이스를 갖는 컴퓨팅 시스템 내의 상이한 컴퓨팅 디바이스들 (예컨대, OpenCL 계산 디바이스들) 사이에 인-플라이트 데이터 공유를 제공하도록 용이하게 확장될 수도 있다. 이러한 예들에서, 하나 이상의 인터럽트 라인들 상이한 컴퓨팅 디바이스들 사이에 유선으로 연결될 수도 있다.

[0214] 도 21 은 본 개시물에 따른, 호스트 디바이스 상에서 실행하는 프로세스에 의해 이슈된 메모리 오브젝트 생성 명령을 실행하는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 20 의 컴퓨팅 시스템 (310) 은 도 21 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 메모리 오브젝트 생성 명령은 버퍼 오브젝트 생성 명령 또는 이미지 오브젝트 생성 명령일 수도 있다. 메모리 버퍼 인터페이스 (328) 는 메모리 오브젝트 생성 명령을 수신한다 (340). 메모리 버퍼 인터페이스 (328) 는 메모리 오브젝트에 대해 즉시 모드가 인에이블되어야 한다고 메모리 오브젝트 생성 명령들이 특정하는지를 결정한다 (342). 예를 들어, 메모리 버퍼 인터페이스 (328) 는 즉시 플래그 파라미터가 메모리 오브젝트 생성 명령에 대한 파라미터 리스트에 포함되는지 여부를 결정할 수도 있다.

[0215] 메모리 오브젝트에 대해 즉시 모드가 인에이블되어야 한다는 것을 메모리 오브젝트 생성 명령이 특정하지 않는다고 메모리 버퍼 인터페이스 (328) 가 결정한다면, 메모리 버퍼 인터페이스 (328) 는 생성되는 메모리 오브젝트에 대해 공유 메모리 공간 (336) 을 할당하고 (344), 호스트 캐시 (324) 및 GPU 캐시 (332) 중 하나 또는 양자에 의한 캐싱 서비스들의 수행이 공유 메모리 공간 (336) 에 대해 인에이블되도록 하며 (346), 생성된 메모리 오브젝트에 대한 참조를 리턴할 수도 있다 (348). 메모리 오브젝트 생성 명령은 예컨대, 즉시 플래그 파라미터를 포함하지 않거나 또는 즉시 모드가 인에이블되지 않아야 한다는 것을 또다른 파라미터 값으로 특정함으로써, 즉시 모드가 인에이블되지 않아야 한다는 것을 특정할 수도 있다. 반대로, 메모리 오브젝트에 대해 즉시 모드가 인에이블되어야 한다고 메모리 오브젝트 생성 명령이 특정한다고 메모리 버퍼 인터페이스 (328) 가 결정한다면, 메모리 버퍼 인터페이스 (328) 는 생성되는 메모리 오브젝트에 대해 공유 메모리 공간 (336) 을 할

당하고 (350), 호스트 캐시 (324) 및 GPU 캐시 (332) 중 하나 또는 양자에 의한 캐싱 서비스들의 수행이 공유 메모리 공간 (336) 에 대해 디스에이블되도록 하며 (352), 생성된 메모리 오브젝트에 대한 참조를 리턴할 수도 있다 (354). 메모리 오브젝트 생성 명령은 예컨대, 즉시 플래그 파라미터를 포함하거나 또는 즉시 모드가 인에이블되어야 한다는 것을 또다른 파라미터 값으로 특정함으로써, 즉시 모드가 인에이블되어야 한다는 것을 특정할 수도 있다.

[0216] 일부 예들에서, 메모리 버퍼 인터페이스 (328) 는 공유 메모리 공간 (336) 과 연관된 메모리 오브젝트의 즉시 모드 속성을 공유 메모리 공간 (336) 과 연관된 메모리 오브젝트에 대해 캐싱 서비스들이 제공되어야 한다는 것을 나타내는 값으로 설정함으로써, 캐싱 서비스들의 수행이 공유 메모리 공간 (336) 에 대해 인에이블되도록 할 수도 있다. 이와 유사하게, 메모리 버퍼 인터페이스 (328) 는 공유 메모리 공간 (336) 과 연관된 메모리 오브젝트의 즉시 모드 속성을 공유 메모리 공간 (336) 과 연관된 메모리 오브젝트에 대해 캐싱 서비스들이 제공되지 않아야 한다는 것을 나타내는 값으로 설정함으로써, 캐싱 서비스들의 수행이 공유 메모리 공간 (336) 에 대해 디스에이블되도록 할 수도 있다. 리턴된 메모리 오브젝트는 즉시 모드 속성을 포함할 수도 있다. 이러한 예들에서, 메모리 오브젝트에 대한 즉시 모드 속성은 호스트 디바이스 (312) 상에서 실행하는 호스트 프로세스 (320) 및 GPU (314) 상에서 실행하는 태스크 (330) 중 하나 또는 양자에 의해 액세스가능할 수도 있다. 호스트 프로세스 (320) 및/또는 태스크 (330) 는 공유 메모리 공간 (336) 과 연관된 메모리 오브젝트의 즉시 모드 속성에 기초하여, 공유 메모리 공간 (336) 에 대해 특정 판독 및 기록 명령들을 실행할 때 캐싱 서비스들을 이용할지 여부를 결정할 수도 있다.

[0217] 추가적인 예들에서, 메모리 버퍼 인터페이스 (328) 는 공유 메모리 공간 (336) 과 연관된 하나 이상의 하드웨어-기반 즉시 플래그들을 공유 메모리 공간 (336) 에 대해 캐싱 서비스들이 제공되어야 한다는 것을 나타내는 값으로 구성함으로써, 캐싱 서비스들의 수행이 공유 메모리 공간 (336) 에 대해 인에이블되도록 할 수도 있다. 이와 유사하게, 메모리 버퍼 인터페이스 (328) 는 공유 메모리 공간 (336) 과 연관된 하나 이상의 하드웨어-기반 즉시 플래그들을 공유 메모리 공간 (336) 에 대해 캐싱 서비스들이 제공되지 않아야 한다는 것을 나타내는 값으로 설정함으로써, 캐싱 서비스들의 수행이 공유 메모리 공간 (336) 에 대해 디스에이블되도록 할 수도 있다. 하나 이상의 하드웨어-기반 즉시 플래그들은 호스트 캐시 제어 모듈 (326) 및 GPU 캐시 제어 모듈 (334) 중 하나 이상 또는 또다른 로컬 또는 글로벌 메모리 관리 유닛 (미도시) 에 로케이트될 수도 있다.

[0218] 추가적인 예들에서, 메모리 버퍼 인터페이스 (328) 는 그 데이터를 저장하기 위해 메모리 (316) 에 물리적 메모리 공간을 할당하기 전에 메모리 오브젝트를 호출 프로세스, 예컨대, 호스트 프로세스 (320) 로 리턴할 수도 있다. 이러한 예들에서, 메모리 버퍼 인터페이스 (328) 는 리턴된 메모리 오브젝트에 즉시 모드 속성을 포함할 수도 있다. 그 후, 추후에 메모리 오브젝트에 메모리 (316) 가 할당될 때, 메모리 버퍼 인터페이스 (328) 또는 또다른 모듈은 메모리 오브젝트의 즉시 모드 속성에 기초하여, 하나 이상의 하드웨어-기반 즉시 플래그들을 구성할 수도 있다.

[0219] 도 22 는 본 개시물에 따른, 호스트 디바이스 상에서 실행하는 프로세스에 의해 이슈된 메모리 오브젝트 생성 명령을 실행하는 또다른 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 20 의 컴퓨팅 시스템 (310) 이 도 22 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 메모리 오브젝트 생성 명령은 버퍼 오브젝트 생성 명령 또는 이미지 오브젝트 생성 명령일 수도 있다. 메모리 버퍼 인터페이스 (328) 는 메모리 오브젝트 생성 명령을 수신한다 (356). 메모리 버퍼 인터페이스 (328) 는 메모리 오브젝트에 대해 즉시 모드가 인에이블되어야 한다고 메모리 오브젝트 생성 명령이 특정한다고 결정한다 (358). 예를 들어, 메모리 버퍼 인터페이스 (328) 는 즉시 플래그 파라미터가 메모리 오브젝트 생성 명령에 대한 파라미터 리스트에 포함되는지 여부를 결정할 수도 있다.

[0220] 메모리 오브젝트에 대해 즉시 모드가 인에이블되어야 한다고 메모리 오브젝트 생성 명령이 특정하지 않는다고 메모리 버퍼 인터페이스 (328) 가 결정한다면, 메모리 버퍼 인터페이스 (328) 는 생성되는 메모리 오브젝트에 대해 공유 메모리 공간 (336) 을 할당하고 (360), 공유 메모리 공간 (336) 에 대해 공유 메모리 캐시 코히런시 모드를 디스에이블하며 (362), 생성된 메모리 오브젝트에 대한 참조를 리턴할 수도 있다 (364). 반대로, 메모리 오브젝트에 대해 즉시 모드가 인에이블되어야 한다고 메모리 오브젝트 생성 명령이 특정한다고 메모리 버퍼 인터페이스 (328) 가 결정한다면, 메모리 버퍼 인터페이스 (328) 는 생성되는 메모리 오브젝트에 대해 공유 메모리 공간 (336) 을 할당하고 (366), 공유 메모리 공간 (336) 에 대해 공유 메모리 캐시 코히런시 모드를 인에이블하며 (368), 생성된 메모리 오브젝트에 대한 참조를 리턴할 수도 있다 (370).

[0221] 일부 예들에서, 메모리 버퍼 인터페이스 (328) 는 그 데이터를 저장하기 위해 메모리 (316) 에 물리적 메모리

공간을 할당하기 전에 메모리 오브젝트를 호출 프로세스, 예컨대, 호스트 프로세스 (320) 로 리턴할 수도 있다.

이러한 예들에서, 메모리 버퍼 인터페이스 (328) 는 리턴된 메모리 오브젝트에 즉시 모드 속성을 포함할 수도 있다. 그 후, 추후에 메모리 오브젝트에 대해 메모리 (316) 가 할당될 때, 메모리 버퍼 인터페이스 (328) 또는 또다른 모듈은 메모리 오브젝트의 즉시 모드 속성에 기초하여 공유 메모리 공간 캐시 코히런시 모드를 인에이블 또는 디스에이블할 수도 있다.

[0222] 도 23 내지 도 28 은 본 개시물에 따른, GPU 가 즉시 모드 및 캐시 모드 로드 및 저장 명령들을 프로세싱하기 위해 사용할 수도 있는 예시적인 기법들을 도시한다. 위에서 설명한 바와 같이, 태스크 (330) 에 대한 소스 코드는 일부 예들에서, 즉시 메모리 오브젝트들과 캐시 메모리 오브젝트들 양자를 지원하기 위하여 컴파일된 코드가 캐시 모드 명령들 및 즉시 모드 명령들 양자를 포함할 수 있도록 컴파일될 수도 있다. 캐시 모드 명령들은 근본적인 메모리와 연관된 캐시의 캐싱 서비스들을 이용하여 메모리에 대해 판독 및 기록 동작들을 실행할 수도 있고, 즉시 모드 명령들은 근본적인 메모리와 연관된 캐시의 캐싱 서비스들을 이용하지 않고, 메모리에 대해 판독 및 기록 동작들을 실행할 수도 있다. 캐시 모드 명령들은 대안으로는 본원에서 비-즉시 모드 명령들로 지칭될 수도 있다. 로드 및 저장 명령들은 대안으로는 본원에서 판독 및 기록 명령들로 각각 지칭될 수도 있다.

[0223] 일부 예들에서, 로드 (load) 또는 저장 명령의 캐시 모드 버전, 및 로드 또는 저장 명령의 즉시 모드 버전은 예컨대, 상이한 연산 코드, 즉, opcode 를 각각 갖는 상이한 명령들일 수도 있다. 추가적인 예들에서, 로드 또는 저장 명령의 캐시 모드 버전, 및 로드 또는 저장 명령의 즉시 모드 버전은 동일한 명령일 수도 있으며, 예컨대, 양자는 동일한 opcode 를 갖는다. 이러한 예들에서, 명령에 제공되는 파라미터는 그 명령이 캐시 모드인지 또는 즉시 모드인지 여부를 특정할 수도 있다.

[0224] 도 23 은 본 개시물에 따른, 캐시 모드 및 즉시 모드 명령들을 프로세싱하는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 20 의 컴퓨팅 시스템 (310) 은 도 23 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 도 23 의 예에서, 즉시 모드는 바이패스 캐시 모드로 지칭되며, 즉시 모드 명령들은 바이패스 캐시 모드 명령들에 대응한다. GPU 캐시 제어 모듈 (334) 은 메모리 로케이션 및 바이패스 캐시 모드가 인에이블되는지 여부를 특정하는 로드 명령을 수신한다 (372). GPU 캐시 제어 모듈 (334) 은 바이패스 캐시 모드가 인에이블된다고 로드 명령이 특정하는지를 결정한다 (374). 일부 경우들에, GPU 캐시 제어 모듈 (334) 은 명령의 유형, 예컨대, 명령의 opcode 에 기초하여, 바이패스 캐시 모드가 인에이블된다고 로드 명령이 특정하는지를 결정할 수도 있다. 추가적인 경우들에서, GPU 캐시 제어 모듈 (334) 은 바이패스 캐시 모드가 인에이블되는지 여부를 나타내는 로드 명령에 포함되는 파라미터에 기초하여, 바이패스 캐시 모드가 인에이블된다고 로드 명령이 특정하는지를 결정할 수도 있다. 바이패스 캐시 모드가 인에이블되지 않는다고 GPU 캐시 제어 모듈 (334) 이 결정한다면, GPU 캐시 제어 모듈 (334) 은 캐시, 예컨대, GPU 캐시 (332) 로부터, 로드 명령에 특정된 메모리 로케이션과 연관된 캐시 로케이션에서 데이터를 취출한다 (376). 한편, 바이패스 캐시 모드가 인에이블된다고 GPU 캐시 제어 모듈 (334) 이 결정한다면, GPU 캐시 제어 모듈 (334) 은 메모리, 예컨대, 공유 메모리 공간 (336) 으로부터, 로드 명령에 특정된 메모리 로케이션에서 데이터를 취출한다 (378).

[0225] 도 24 는 본 개시물에 따른, 캐시 모드 및 즉시 모드 명령들을 프로세싱하는 또다른 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 20 의 컴퓨팅 시스템 (310) 이 도 24 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 도 24 의 예에서, 즉시 모드는 바이패스 캐시 모드로 지칭되며, 즉시 모드 명령들은 바이패스 캐시 모드 명령들에 대응한다. GPU 캐시 제어 모듈 (334) 은 메모리 로케이션, 저장할 데이터 및 바이패스 캐시 모드가 인에이블되는지 여부를 특정하는 저장 명령을 수신한다 (380). GPU 캐시 제어 모듈 (334) 은 바이패스 캐시 모드가 인에이블된다고 저장 명령이 특정하는지를 결정한다 (382). 일부 경우들에서, GPU 캐시 제어 모듈 (334) 은 명령의 유형, 예컨대, 명령의 opcode 에 기초하여, 바이패스 캐시 모드가 인에이블된다고 저장 명령이 특정하는지를 결정할 수도 있다. 추가적인 경우들에서, GPU 캐시 제어 모듈 (334) 은 바이패스 캐시 모드가 인에이블되는지 여부를 나타내는 로드 명령에 포함되는 파라미터에 기초하여, 바이패스 캐시 모드가 인에이블된다고 저장 명령이 특정하는지를 결정할 수도 있다. 바이패스 캐시 모드가 인에이블되지 않는다고 GPU 캐시 제어 모듈 (334) 이 결정한다면, GPU 캐시 제어 모듈 (334) 은 저장 명령에 특정된 데이터를 캐시, 예컨대, GPU 캐시 (332) 에, 저장 명령에 특정된 메모리 로케이션과 연관된 캐시 로케이션에 저장한다 (384). 한편, 바이패스 캐시 모드가 인에이블된다고 GPU 캐시 제어 모듈 (334) 이 결정한다면, GPU 캐시 제어 모듈 (334) 은 저장 명령에 특정된 데이터를 메모리, 예컨대, 공유 메모리 공간 (336) 에, 저장 명령에 특정된 메모리 로케이션에 저장한다 (386).

[0226] 도 25 는 본 개시물에 따른, 캐시 모드 및 즉시 모드 명령들을 프로세싱하는 또다른 예시적인 기법을 도시하는

흐름도이다. 일부 예들에서, 도 20 의 컴퓨팅 시스템 (310) 이 도 25 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. GPU 캐시 제어 모듈 (334) 은 메모리 로케이션, 저장할 데이터 및 즉시 모드가 인에이블되는지 여부를 특정하는 저장 명령을 수신한다 (388). GPU 캐시 제어 모듈 (334) 은 저장 명령에 특정된 데이터를 캐시, 예컨대, GPU 캐시 (332) 에, 저장 명령에 특정된 메모리 로케이션과 연관된 캐시 로케이션에 저장한다 (390). GPU 캐시 제어 모듈 (334) 은 즉시 모드가 인에이블되는지 여부를 특정하는 저장 명령 내의 정보에 기초하여, 즉시 모드가 인에이블되는지를 결정한다 (392). 즉시 모드가 인에이블되는지를 특정하는 정보는 일부 예들에서, 명령의 유형, 예컨대, 명령에 대한 opcode 및/또는 그 명령에 대해 즉시 모드가 인에이블되는지 여부를 특정하는 명령 내에 포함된 파라미터일 수도 있다. 즉시 모드가 인에이블되지 않는다면, GPU 캐시 제어 모듈 (334) 은 즉시 캐시 후기록 동작 (immediate cache write-back operation) 을 수행하지 않는다 (394). 한편, 즉시 모드가 인에이블된다면, GPU 캐시 제어 모듈 (334) 은 즉시 캐시 후기록 동작을 수행한다 (396).

[0227] 도 26 은 본 개시물에 따른, 캐시 모드 및 즉시 모드 명령들을 프로세싱하는 또다른 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 20 의 컴퓨팅 시스템 (310) 이 도 26 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. GPU 캐시 제어 모듈 (334) 은 메모리 로케이션 및 즉시 모드가 인에이블되는지 여부를 특정하는 로드 명령을 수신한다 (398). GPU 캐시 제어 모듈 (334) 은 즉시 모드가 인에이블되는지 여부를 특정하는 로드 명령 내의 정보에 기초하여, 즉시 모드가 인에이블되는지를 결정한다 (400). 즉시 모드가 인에이블되는지 여부를 특정하는 정보는 일부 예들에서, 명령의 유형, 예컨대, 명령에 대한 opcode 및/또는 그 명령에 대해 즉시 모드가 인에이블되는지 여부를 특정하는 명령에 포함되는 파라미터일 수도 있다. 즉시 모드가 인에이블되지 않는다면, GPU 캐시 제어 모듈 (334) 은 캐시를 플러시 및 무효화하지 않는다 (402). GPU 캐시 제어 모듈 (334) 은 데이터가 캐시에서 이용가능하다면, 캐시, 예컨대, GPU 캐시 (332) 로부터, 또는, 데이터가 캐시에서 이용가능하지 않다면, 근본적인 메모리로부터, 로드 명령에 특정된 데이터를 취출한다 (404). 즉시 모드가 인에이블된다면, GPU 캐시 제어 모듈 (334) 은 캐시를 플러시 및 무효화한다 (406). GPU 캐시 제어 모듈 (334) 은 근본적인 메모리로부터 로드 명령에 특정된 데이터를 취출한다 (408). 캐시는 캐시가 플러시 및 무효화되었기 때문에 데이터를 리턴하지 않는다.

[0228] 도 27 은 본 개시물에 따른, 도 20 의 컴퓨팅 시스템 (310) 에 사용될 수도 있는 예시적인 GPU (420) 를 도시하는 블록도이다. 일부 예들에서, GPU (420) 는 도 20 에 도시된 GPU (314) 를 구현하는데 사용될 수도 있다. GPU (420) 는 GPU 프로세싱 모듈 (422), GPU 캐시 제어 모듈 (424), GPU 캐시 (426), 캐시 버스 (428) 및 바이패스 버스 (430) 를 포함한다. GPU 프로세싱 모듈 (422) 은 캐시 버스 (428) 를 통해 GPU 캐시 제어 모듈 (424) 에 통신적으로 커플링된다. GPU 프로세싱 모듈 (422) 은 또한 바이패스 버스 (430) 를 통해 메모리 (316) 에 통신적으로 커플링된다. GPU 캐시 제어 모듈 (424) 및 GPU 캐시 (426) 는 도 20 의 GPU 캐시 제어 모듈 (334) 및 GPU 캐시 (332) 와 실질적으로 유사하며, 더 상세히 설명되지 않을 것이다. GPU 프로세싱 모듈 (422) 은 프로세싱 엘리먼트 (432) 및 버스 제어기 (434) 를 포함한다. 프로세싱 엘리먼트 (432) 는 로드 및 저장 명령들을 버스 제어기 (434) 에 이슈하도록 구성된다.

[0229] 버스 제어기 (434) 는 캐시 버스 (428) 및 바이패스 버스 (430) 를 통해 로드 및 저장 명령들을 적합한 로케이션들로 포워딩하도록 구성될 수도 있다. 버스 제어기 (434) 는 그 명령이 즉시 모드인지 또는 캐시 모드 명령인지 여부를 나타내는 로드 또는 저장 명령 내의 정보에 기초하여, 즉시 모드 또는 비-즉시 모드에서 동작하도록 구성될 수도 있다. 버스 제어기 (434) 가 비-즉시 모드, 즉, 캐시 모드에서 동작하도록 구성될 때, 버스 제어기 (434) 는 실행을 위해 로드 및 저장 명령들을 GPU 캐시 제어 모듈 (424) 로 포워딩하기 위해 캐시 버스 (428) 를 이용할 수도 있다. 한편, 버스 제어기 (434) 가 즉시 모드에서 동작하도록 구성될 때, 버스 제어기 (434) 는 실행을 위해 로드 및 저장 명령들을 메모리 (316) 로 포워딩하기 위해 바이패스 버스 (430) 를 이용할 수도 있다.

[0230] 도 28 은 본 개시물에 따른, 캐시 모드 및 즉시 모드 명령들을 프로세싱하는 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 27 의 GPU (420) 가 도 28 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 버스 제어기 (434) 는 로드 또는 저장 명령을 수신한다 (440). 버스 제어기 (434) 는 즉시 모드가 인에이블되는지 여부를 특정하는 로드 또는 저장 명령 내의 정보에 기초하여, 즉시 모드가 인에이블되는지를 결정한다 (442). 즉시 모드가 인에이블되는지 여부를 특정하는 정보는 일부 예들에서, 명령의 유형, 예컨대, 명령에 대한 opcode 및/또는 그 명령에 대해 즉시 모드가 인에이블되는지 여부를 특정하는 명령에 포함되는 파라미터일 수도 있다. 즉시 모드가 인에이블되지 않는다고 버스 제어기 (434) 가 결정한다면, 버스 제어기 (434) 는 수신된 명령을 GPU 캐시 제어 모듈 (424) 로 포워딩하기 위해 캐시 버스 (428) 를 이용한다 (444).

그렇지 않고, 즉시 모드가 인에이블된다고 버스 제어기 (434) 가 결정한다면, 버스 제어기 (434) 는 수신된 명령을 메모리 (316) 로 포워딩하기 위해 바이패스 버스 (430) 를 이용한다 (446).

[0231] 도 29 는 본 개시물에 따른, 호스트 디바이스 상에서 실행하는 프로세스에 의해 이슈된 메모리 오브젝트 생성 명령을 실행하는 또다른 예시적인 기법을 도시하는 흐름도이다. 일부 예들에서, 도 20 의 컴퓨팅 시스템 (310) 이 도 29 에 나타낸 예시적인 기법을 구현하는데 사용될 수도 있다. 메모리 오브젝트 생성 명령은 버퍼 오브젝트 생성 명령 또는 이미지 오브젝트 생성 명령일 수도 있다. 메모리 버퍼 인터페이스 (328) 는 메모리 오브젝트 생성 명령을 수신한다 (448). 메모리 버퍼 인터페이스 (328) 는 메모리 오브젝트에 대해 즉시 모드가 인에이블되어야 한다고 메모리 오브젝트 생성 명령이 특정하는지를 결정한다 (450). 예를 들어, 메모리 버퍼 인터페이스 (328) 는 즉시 플래그 파라미터가 메모리 오브젝트 생성 명령에 대한 파라미터 리스트에 포함되는지 여부를 결정할 수도 있다.

[0232] 메모리 오브젝트에 대해 즉시 모드가 인에이블되어야 한다고 메모리 오브젝트 생성 명령이 특정하지 않는다고 메모리 버퍼 인터페이스 (328) 가 결정한다면, 메모리 버퍼 인터페이스 (328) 는 생성된 메모리 오브젝트에 대한 즉시 모드 속성을, 즉시 모드가 인에이블되지 않는다는 것을 나타내는 값, 예컨대, "거짓" 으로 설정한다 (452). 한편, 메모리 오브젝트에 대해 즉시 모드가 인에이블되어야 한다고 메모리 오브젝트 생성 명령이 특정한다고 메모리 버퍼 인터페이스 (328) 가 결정한다면, 메모리 버퍼 인터페이스 (328) 는 생성된 메모리 오브젝트에 대한 즉시 모드 속성을 즉시 모드가 인에이블된다는 것을 나타내는 값, 예컨대, "참" 으로 설정한다 (454). 메모리 오브젝트의 즉시 모드 속성은 일부 예들에서, 특정 메모리 오브젝트에 저장된 데이터에 액세스할 때 캐시 모드 또는 즉시 모드 판독 및 기록 동작들을 실행할지 여부를 결정하기 위해, 호스트 디바이스 (312) 및/또는 GPU (314) 에 의해 사용될 수도 있다.

[0233] 일부 예들에서, 호스트 프로세스 (320) 및/또는 태스크 (330) 는 일부 메모리 오브젝트들을 즉시 메모리 오브젝트들인 것으로, 그리고, 다른 오브젝트들을 캐시 메모리 오브젝트들, 즉, 비-즉시 메모리 오브젝트들인 것으로 프로그래밍하기를 원할 수도 있다. 본 개시물의 기법들은 일부 예들에서, 컴파일된 태스크 (330) 가 캐시 메모리 오브젝트들과 즉시 메모리 오브젝트들 양자에 대해 판독 및 기록 동작들을 수행하는 것을 허용하는 특수화된 컴파일 기법들을 포함할 수도 있다. 제 1 예시적인 컴파일 기법은 주어진 판독 동작 또는 기록 동작을 명령들의 시퀀스로 컴파일할 수도 있다. 명령들의 시퀀스는 판독하거나 기록할 메모리 오브젝트에 대한 즉시 모드 속성의 값을 체크하고, 즉시 모드 속성의 값에 기초하여 캐시 모드 명령을 실행할지 또는 즉시 모드 명령을 실행할지 여부를 결정할 수도 있다. 제 2 예시적인 컴파일 기법은 캐시 모드 명령들 또는 즉시 모드 명령들 중 어느 하나를 선택하여, 메모리 오브젝트에 액세스하는 컴파일된 코드에 사용하기 위해, 메모리 오브젝트가 즉시 모드 오브젝트인지 여부를 나타내는 소스 코드 내의 정보를 이용할 수도 있다.

[0234] 제 1 예시적인 컴파일 기법에 따르면, 컴파일러는 태스크 (330) 에 대한 컴파일된 코드가 다음의 예시적인 의사-코드에 따른 판독 시퀀스를 포함하도록 태스크 (330) 에 대한 소스 코드를 컴파일할 수도 있다:

[0235]
```
if( isImmediate ){
```

[0236]
```
        immediate_read(...)
```

[0237]
```
}
```

[0238]
```
else{
```

[0239]
```
        cached_read(...)
```

[0240]
```
}
```

[0241] 여기서, "isImmediate" 는 데이터가 판독될 메모리 오브젝트에 대한 부울 즉시 모드 속성을 나타내고, "immediate_read(...)" 는 즉시 모드 판독 명령을 나타내며, "cached_read(...)" 는 캐시 모드 판독 명령을 나타낸다.

[0242] GPU 캐시 제어 모듈 (334) 은 immediate_read(...) 명령을, 예를 들어, 사용된다면, GPU 캐시 (332) 로부터 데이터를 판독하기 전에 GPU 캐시 (332) 를 무효화함으로써 프로세싱할 수도 있다. GPU 캐시 제어 모듈 (334) 은 예컨대, 판독을 수행하기 전에 캐시를 무효화하지 않고, 예를 들어, GPU 캐시 (332) 로부터 데이터를 통상의 방식으로 판독함으로써, cached_read(...) 명령을 프로세싱할 수도 있다.

[0243] 제 1 예시적인 컴파일 기법에 따르면, 컴파일러는 태스크 (330) 에 대한 컴파일된 코드가 다음의 예시적인 의사

-코드에 따른 기록 시퀀스를 포함하도록, 태스크 (330) 에 대한 소스 코드를 컴파일할 수도 있다:

[0244] if(isImmediate){

[0245] immediate_write(...)

[0246] }

[0247] else{

[0248] cached_write(...)

[0249] }

[0250] 여기서, "isImmediate" 는 데이터가 기록될 메모리 오브젝트에 대한 부울 즉시 모드 속성을 나타내고, "immediate_write(...)" 는 즉시 모드 기록 명령을 나타내며, "cached_write(...)" 는 캐시 모드 기록 명령을 나타낸다.

[0251] GPU 캐시 제어 모듈 (334) 은 일부 예들에서, 캐시가 사용된다면, GPU 캐시 (332) 에 대한 연속기록 모드 (write-through mode) 를 이용하여, immediate_write(...) 명령을, 프로세싱할 수도 있다. 추가적인 예들에서, GPU 캐시 제어 모듈 (334) 은 캐시가 사용된다면 GPU 캐시 (332) 에 데이터를 기록하고, GPU 캐시 (332) 에 데이터를 기록하는 것에 응답하여 GPU 캐시 (332) 에 대한 캐시 플러시 (cache flush) 를 수행함으로써, immediate_write(...) 명령을 프로세싱할 수도 있다. GPU 캐시 제어 모듈 (334) 은 예컨대, 연속기록 모드를 이용하지 않거나 및/또는 기록 동작에 응답하여 캐시를 플러시하지 않고, GPU 캐시 (332) 에 데이터를 통상의 방식으로 기록함으로써, cached_write(...) 명령을 프로세싱할 수도 있다.

[0252] 도 30 은 GPU 가 위에서 설명한 제 1 컴파일 기법에 따라 컴파일된 명령들의 시퀀스를 프로세싱할 수도 있는 방법을 도시하는 흐름도이다. 일부 예들에서, 도 30 에 도시된 기법은 판독 및 기록 시퀀스들에 대해 위에서 제공된 예시적인 의사-코드를 구현하는데 사용될 수도 있다. 태스크 (330) 는 판독 시퀀스 또는 기록 시퀀스를 시작한다 (456). 예를 들어, 태스크 (330) 는 태스크 (330) 가 특정 메모리 오브젝트에 대한 판독 또는 기록 명령이 발생해야 하는 태스크 (330) 의 실행의 포인트에 도달할 때 판독 시퀀스 또는 기록 시퀀스를 시작할 수도 있다. 태스크 (330) 는 데이터가 판독되거나 또는 데이터가 기록될 메모리 오브젝트와 연관된 즉시 모드 속성에 액세스한다 (458). 태스크 (330) 는 메모리 오브젝트에 대한 속성이 즉시 모드가 인에이블된다는 것을 나타내는 값, 예컨대, "참" 으로 설정되는지 여부를 결정한다 (460). 메모리 오브젝트에 대한 속성이 즉시 모드가 인에이블되지 않는다는 것을 나타내는 값으로 설정된다고 태스크 (330) 가 결정한다면, 태스크 (330) 는 캐시 판독 또는 기록 명령을 이용하여 메모리 오브젝트에 대한 캐시 판독 또는 기록 동작을 수행한다 (462). 그렇지 않고, 메모리 오브젝트에 대한 속성이 즉시 모드가 인에이블된다는 것을 나타내는 값으로 설정된다고 태스크 (330) 가 결정한다면, 태스크 (330) 는 즉시 판독 또는 기록 명령을 이용하여 메모리 오브젝트에 대한 즉시 판독 또는 기록 동작을 수행한다 (464).

[0253] 제 2 예시적인 컴파일 기법에 따르면, 소스 코드를 컴파일할 때, 컴파일러는 태스크 (330) 에 의해 판독되거나 기록되는 특정 메모리 오브젝트에 대해 즉시 모드가 인에이블되는지 여부를 나타내는 정보에 액세스할 수도 있다. 컴파일러는 태스크 (330) 가 특정 메모리 오브젝트로부터 판독하거나 특정 메모리 오브젝트에 기록할 때, 태스크 (330) 에 대한 소스 코드를 컴파일하여 캐시 모드 판독 및 기록 명령들 사이 또는 즉시 모드 판독 및 기록 명령들 사이 중 어느 하나에서 선택하기 위해 이 정보를 이용할 수도 있다.

[0254] 일부 예들에서, 특정 메모리 오브젝트에 대해 즉시 모드가 인에이블되는지 여부를 나타내는 정보는 태스크 (330) 에 대한 소스 코드에 의해 액세스되는 하나 이상의 메모리 오브젝트들에 대해 즉시 모드가 인에이블되는지 여부를 나타내는 컴파일-시간 속성일 수도 있다. 예를 들어, 태스크 (330) 에 대한 소스 코드, 예컨대, 커널 소스 코드는, 태스크 (330) 에 의해 사용되는 하나 이상의 메모리 오브젝트들에 대해 즉시 모드가 인에이블되는지 여부를 나타내는 컴파일-시간 속성을 포함할 수도 있다. 컴파일-시간 속성은 일부 경우들에서, 예컨대, _cl_immediate 와 같은 OpenCL 속성 수식자의 형태를 취할 수도 있다. 속성 수식자는 하나 이상의 특정 메모리 오브젝트들 및/또는 하나 이상의 메모리 오브젝트들 내에 저장된 하나 이상의 변수들과 연관될 수도 있다. 속성 수식자가 특정 메모리 오브젝트와 연관될 때, 컴파일러는 메모리 오브젝트에 대해 즉시 모드가 인에이블된다고 결정할 수도 있다. 이와 유사하게, 속성 수식자가 특정 메모리 오브젝트와 연관되지 않을 때, 컴파일러는 메모리 오브젝트에 대해 즉시 모드가 인에이블되지 않는다고 결정할 수도 있다. 이러한 속성을 이용하는 것은 컴파일러에 대한 작업을 감소시켜 잠재적으로 커널 사이즈를 감소시킬 수도 있다. 일부

예들에서, 소프트웨어 애플리케이션들은 즉시 버퍼들의 사용을 이러한 버퍼들이 요구되는 경우들에 제한할 수도 있다. 이러한 예들에서, 즉시 버퍼들을 사용할지 여부의 결정은 컴파일 시간 결정일 수도 있다.

[0255] 공유 메모리 공간 (336) 과 연관된 메모리 오브젝트에 대해 즉시 모드가 인에이블된다고 컴파일-시간 속성이 나타낸다면, 컴파일러는 태스크 (330) 에 대한 컴파일된 코드가 공유 메모리 공간 (336) 에 대해 일어나는 판독 또는 기록 동작들에 대한 즉시 모드 판독 및/또는 기록 명령들을 포함하도록 태스크 (330) 를 컴파일할 수도 있다. 그렇지 않고, 공유 메모리 공간 (336) 과 연관된 메모리 오브젝트에 대해 즉시 모드가 인에이블되지 않는다면, 컴파일러는 태스크 (330) 에 대한 컴파일된 코드가 공유 메모리 공간 (336) 에 대해 일어나는 판독 또는 기록 동작들에 대한 즉시 모드 판독 및/또는 기록 명령들을 포함하지 않도록 태스크 (330) 를 컴파일할 수도 있다. 예를 들어, 컴파일러는 태스크 (330) 에 대한 컴파일된 코드가 공유 메모리 공간 (336) 에 대해 일어나는 판독 또는 기록 동작들에 대한 캐시 판독 및/또는 기록 명령들을 포함하도록 태스크 (330) 를 컴파일할 수도 있다.

[0256] 도 31 은 본 개시물에 따른, 태스크에 대한 소스 코드를 컴파일하는 예시적인 기법을 도시하는 흐름도이다. 도 31 의 기법들을 이용하여 컴파일된 결과의 코드는 일부 예들에서, 도 20 의 태스크 (330) 에 대응할 수도 있다. 도 31 의 예시적인 기법에서, 태스크 (330) 는 커널로 지칭된다. 컴파일러는 메모리 오브젝트에 의해 구현되는 커널 인수를 프로세싱한다 (466). 컴파일러는 메모리 오브젝트가 즉시 모드 메모리 오브젝트인지 여부를 결정한다 (468). 일부 예들에서, 컴파일러는 커널의 소스 코드에 포함된 정보, 예컨대, 커널 인수와 연관된 컴파일-시간 속성에 기초하여, 메모리 오브젝트가 즉시 모드 메모리 오브젝트인지 여부를 결정할 수도 있다. 메모리 오브젝트가 즉시 모드 메모리 오브젝트가 아니라고 컴파일러가 결정한다면, 컴파일러는 캐시 판독 및 기록 명령들을 이용하여 특정 커널 인수와 연관된 판독 동작들 및 기록 동작들을 컴파일한다 (470). 한편, 메모리 오브젝트가 즉시 모드 메모리 오브젝트라고 컴파일러가 결정한다면, 컴파일러는 즉시 모드 판독 및 기록 명령들을 이용하여 특정 커널 인수와 연관된 판독 동작들 및 기록 동작들을 컴파일한다 (472).

[0257] 도 32 는 본 개시물에 따른, 캐싱 서비스들을 선택적으로 이용하기 위해 GPU 에 의해 사용될 수도 있는 예시적인 기법을 도시하는 흐름도이다. 예를 들어, 기법들은 메모리 공간에 대해 판독 동작들 및 기록 동작들 중 적어도 하나를 실행하기 위해 캐싱 서비스들이 이용되어야 하는지 여부를 특정하는 정보를 수신하는 것에 응답하여, 메모리의 메모리 공간에 대해 판독 동작 및 기록 동작 중 적어도 하나를 실행하기 위해, GPU 가 메모리와 연관된 GPU 캐시를 선택적으로 이용하는 것을 허용할 수도 있다. 일부 예들에서, 도 20 에 도시된 GPU (314) 및/또는 도 27 에 도시된 GPU (420) 가 도 32 에 도시된 기법들을 구현하는데 사용될 수도 있다.

[0258] GPU (314) 는 프로세싱할 판독 명령 또는 기록 명령 중 적어도 하나를 수신한다 (474). 수신된 명령은 메모리의 메모리 공간에 대해 판독 동작 및 기록 동작 중 적어도 하나를 실행하도록 GPU (314) 에 명령할 수도 있다. GPU (314) 는 메모리 공간에 대해 판독 동작들 및 기록 동작들 중 적어도 하나를 실행하기 위해 캐싱 서비스들이 이용되어야 하는지 여부를 특정하는 캐시 모드 정보를 수신한다 (476). 일부 예들에서, 캐시 모드 정보는 수신된 명령 내에 포함될 수도 있다. 추가적인 예들에서, 캐시 모드 정보는 메모리 공간과 연관된 메모리 오브젝트의 즉시 모드 속성일 수도 있다. GPU (314) 는 캐시 모드 정보에 기초하여 캐싱 서비스들을 이용할지 여부를 결정한다 (478). 캐싱 서비스들이 수신된 명령을 실행하는데 사용되어야 한다고 특정하는 정보를 수신하는 것에 응답하여, GPU (314) 는 수신된 명령을 실행하기 위해 캐싱 서비스들을 이용할 수도 있다 (480). 캐싱 서비스들이 수신된 명령을 실행하는데 사용되지 않아야 한다고 특정하는 정보를 수신하는 것에 응답하여, GPU (314) 는 수신된 명령을 실행하기 위해 캐싱 서비스들을 이용하지 않을 수도 있다 (482).

일부 예들에서, GPU (314) 는 결정 박스 (478) 및 프로세스 박스들 (480 및 482) 중 하나 이상을 구현하기 위해 도 23 내지 도 28 및 도 30 에 도시된 기법들 중 하나 이상을 이용할 수도 있다. 일부 경우들에서, GPU 캐시 제어 모듈 또는 메모리 관리 유닛, 예컨대, 도 20 에 도시된 GPU 캐시 제어 모듈 (334) 이 도 32 에 나타낸 기법들을 구현하는데 사용될 수도 있다. 추가적인 경우들에서, 버스 제어기, 예컨대, 도 27 에 도시된 버스 제어기 (434) 가 도 32 에 나타낸 기법들을 구현하는데 사용될 수도 있다.

[0259] 일부 예들에서, 즉시 메모리 오브젝트들을 구현하기 위해, GPU ALU 는 그 명령에 특정된 글로벌 메모리 캐시 및/또는 글로벌 메모리 캐시의 특정의 부분을 무효화하는 ALU 명령을 실행하도록 설계될 수도 있다. 일반적으로, 호스트 디바이스 (312) 는 즉시 메모리 오브젝트들을 구현하기 위해 기존의 CPU 능력들을 이용할 수도 있다.

[0260] 이하, 본 개시물에서 설명되는 대역외 시그널링 기법들, 예컨대, 본원에서 설명되는 메시지 전달 기법들, 및 본

개시물에서 설명되는 즉시 메모리 오브젝트들에 대한 여러 유즈 케이스들이 더 상세히 설명될 것이다. 제 1 유즈 케이스 (use case) 에 따르면, 대역외 시그널링은 대역외 시그널링 기법들에 더해 즉시 메모리 오브젝트들을 반드시 이용하지 않고, 독립형 (stand-alone) 피쳐로서 사용될 수도 있다. 대역외 시그널링은 동기화 및 상대적으로 적은 양의 데이터를 빨리 전달하기 위해 사용될 수도 있다. 일부 예들에서, 대역외 시그널링은 즉시 메모리 오브젝트들보다 낮은 레이턴시를 갖지만, 즉시 메모리 오브젝트들보다 낮은 대역폭을 가질 수도 있다.

[0261] 대역외 시그널링은 또한 메모리 할당 동작들에 대해 제 1 유즈 케이스에 따라 사용될 수도 있다. 예를 들어, GPU 는 호스트 CPU 가 새로운 버퍼를 할당하도록 요청하기 위해 대역외 시그널링을 이용할 수도 있다. GPU 는 또한 호스트 CPU 에 요청된 버퍼 길이를 특정하기 위해 대역외 시그널링을 이용할 수도 있다. 또 다른 예로서, CPU 는 버퍼에 대한 메모리 로케이션을 특정하는 포인터를 GPU 로 전송하기 위해 버퍼를 할당한 후 대역외 시그널링을 이용할 수도 있다.

[0262] 대역외 시그널링은 또한 적은 양의 데이터가 교환되는 원격 프로시저 호출들에 대해 제 1 유즈 케이스에 따라 사용될 수도 있다. 예를 들어, 컴퓨팅 디바이스 내의 계산 유닛 상에서 실행하는 커널이 동일한 컴퓨팅 디바이스 또는 또다른 컴퓨팅 디바이스 중 어느 하나 내의 또다른 계산 유닛 상에서 또다른 커널을 착수하기 위해 RPC 를 이용하는 경우에, RPC 에 대한 데이터는 그 착수하는 계산 유닛의 로컬 메모리에 저장될 것이다. 본 개시물의 대역외 시그널링 기법들은 데이터를 그 착수하는 계산 유닛의 로컬 메모리로부터 새로 착수된 커널을 실행하는 계산 유닛의 로컬 메모리로 전송하는데 사용될 수도 있다.

[0263] 대역외 시그널링은 또한 중간 레포팅에 대해 제 1 유즈 케이스에 따라 사용될 수도 있다. 예를 들어, GPU 는 현재의 태스크의 완료의 퍼센티지를 호스트 CPU 에 레포팅하기 위해 대역외 시그널링을 이용할 수도 있다.

[0264] 대역외 시그널링은 또한 에러 레포팅에 대해 제 1 유즈 케이스에 따라 사용될 수도 있다. 예를 들어, GPU 는 에러 코드를 호스트 CPU 에 레포팅하기 위해 대역외 시그널링을 이용할 수도 있다.

[0265] 대역외 시그널링은 또한 콘텍스트 스위치 (context switche) 들을 돕기 위해 제 1 유즈 케이스에 따라 사용될 수도 있다. 예를 들어, 호스트 CPU 는 GPU 가 콘텍스트 스위치를 준비하는 상태를 세이브하도록 요청하기 위해 대역외 시그널링을 이용할 수도 있다.

[0266] 제 2 유즈 케이스에 따르면, 즉시 메모리 오브젝트들은 즉시 메모리 오브젝트들에 더해서 대역외 시그널링을 반드시 이용하지 않고, 독립형 피쳐로서 사용될 수도 있다. 예를 들어, 즉시 버퍼들은 상대적으로 큰 양의 데이터의 교환을 달성하는데 사용될 수도 있다. 즉시 버퍼들은 데이터 뿐만 아니라, 동기화 마커들도 포함할 수도 있다. 이 경우, 데이터 생산자는 먼저 데이터를 버퍼에 기록하고, 데이터의 준비도 (readiness) 및/또는 로케이션을 소비자에게 나타내는 동기화 마커 (synchronization marker) 를 기록할 수도 있다. 소비자는 이 메모리 로케이션을 폴링함으로써, 예를 들어, 버퍼의 헤더 섹션에서와 같이, 선험적으로 결정된 로케이션에서, 동기화 데이터를 찾는다. 일단 동기화 마커가 획득되면, 소비자는 데이터를 판독한다. 유사한 기법들이 즉시 이미지 오브젝트들에 적용될 수도 있다.

[0267] 다양한 동기화 프로토콜들이 이들 기법들에 채용될 수도 있다. 예를 들어, 동기화 마커들이 데이터 버퍼 내부에 내장되거나, 또는 별개의 버퍼들에 로케이트될 수도 있다. 이러한 기법들이 가변 길이 인코딩 또는 실행 길이 (run length) 인코딩 방식들을 이용하여 압축된 압축 데이터의 송신에 적용될 수도 있다.

[0268] 제 3 유즈 케이스에 따르면, 즉시 메모리 오브젝트들이 예를 들어, 상대적으로 큰 양의 데이터의 교환을 달성하기 위해 대역외 시그널링과 함께 사용될 수도 있다. 이 경우, 대역외 시그널링이 즉시 메모리 오브젝트들이 데이터를 저장하는 동안 동기화를 위해 사용될 수도 있다. 예를 들어, 데이터 생산자는 데이터를 즉시 버퍼에 배치하고 대역외 시그널링을 이용하여 소비자에게 데이터의 사이즈 및/또는 준비도 및 로케이션을 통지할 수도 있다. 흐름 제어 시나리오에서, 소비자는 데이터를 판독하고 생산자에게 버퍼가 재사용될 수 있다는 것을 통지한다. 통지는 또한 대역외 시그널링을 이용하여 달성될 수도 있다.

[0269] 이러한 기법들은 흐름 제어 데이터 파이프라이닝 (pipelining) 을 필요로 하는 알고리즘들에 사용될 수도 있다. 호스트 CPU 및 GPU 에 있어, 이러한 기법들은 예를 들어, 진단의 로깅 (logging) 에 사용될 수도 있다. 다수의 OpenCL 계산 디바이스들에 있어, 이들 기법들은 다수의 디바이스들을 비동기적 흐름-제어 데이터 파이프라인에 접속하는데 사용될 수도 있다. 이것은 애플리케이션으로 하여금, 각각의 CPU 또는 GPU 에 더 적합한 블록들로 분할가능하게 하여, 다수의 디바이스들 상에서 여러 파이프라인 프로세싱 단계들을 착수하거나, 및/또는 호스트 CPU 로부터 대부분, 또는 심지어 모든 데이터 동기화를 오프-로드할 수도 있다.

[0270] 일부 예들에서, 본 개시물의 기법들은 커맨드 큐들을 이용하여 태스크들을 개시하는 다중-프로세서 컴퓨팅 플랫폼에 대해 호스트 디바이스 상에서 실행하는 프로세스와 컴퓨팅 디바이스 상에서 실행하는 태스크 사이에 메시지들의 전송 및 수신을 용이하게 하는 메시지 전달 인터페이스를 제공할 수도 있다. 컴퓨팅 디바이스는 일부 경우들에서, GPU 일 수도 있다. 추가적인 경우들에서, 컴퓨팅 디바이스는 크로스-플랫폼, 크로스-벤더, 이종 컴퓨팅 플랫폼 API 에 의해 정의되는 임의의 유형의 컴퓨팅 디바이스일 수도 있다.

[0271] 추가적인 예들에서, 본 개시의 기법들은 호스트 디바이스에 의해 액세스가능한 하나 이상의 레지스터들을 포함하는 GPU 를 제공할 수도 있다. 하나 이상의 레지스터들은 GPU 상에서 실행하는 태스크와 GPU 와는 다른 디바이스 상에서 실행하는 프로세스 사이에 메시지 전달을 용이하게 하도록 구성될 수도 있다.

[0272] 추가적인 예들에서, 본 개시의 기법들은 즉시 메모리 오브젝트들이 생성되는 것을 허용하는 메모리 버퍼 인터페이스를 제공할 수도 있다. 즉시 메모리 오브젝트들은 태스크가 컴퓨팅 디바이스 상에서 실행중인 동안 호스트 디바이스 상에서 실행하는 프로세스와 컴퓨팅 디바이스 상에서 실행하는 태스크 사이에 데이터를 공유하기 위하여 캐시불가능한 공유 메모리 공간 및/또는 캐시-코히런트 공유 메모리 공간을 구현하는데 사용될 수도 있다. 컴퓨팅 디바이스는, 일부 경우들에서, 그래픽 프로세싱 유닛 (GPU) 일 수도 있다. 추가적인 경우들에서, 컴퓨팅 디바이스는 크로스-플랫폼, 크로스-벤더, 이종 컴퓨팅 플랫폼 API 에 의해 정의된 임의의 유형의 컴퓨팅 디바이스일 수도 있다.

[0273] 또한 추가적인 예들에서, 본 개시의 기법들은 캐시불가능한 공유 메모리 공간을 제공하기 위하여 선택적으로 디스에이블될 수도 있는 공유 메모리 공간에 대한 캐시를 포함하는 GPU 를 제공할 수도 있다. 추가적인 예들에서, 본 개시물의 기법들은 캐시-코히런트 공유 메모리 공간을 제공하기 위해 선택적으로 인에이블될 수도 있는 캐시 코히런시 모드를 포함하는 GPU 를 제공할 수도 있다.

[0274] 본 개시물에서 설명하는 기법들은 적어도 부분적으로, 하드웨어, 소프트웨어, 펌웨어 또는 이들의 임의의 조합으로 구현될 수도 있다. 예를 들어, 설명되는 기법들의 여러 양태들은 하나 이상의 마이크로프로세서들, 디지털 신호 프로세서 (DSP) 들, 주문형 집적회로 (ASIC) 들, 필드 프로그래밍가능 게이트 어레이 (FPGA) 들, 또는 임의의 다른 등가의 통합 또는 이산 로직 회로 뿐만 아니라, 이러한 컴포넌트들의 임의의 조합들을 포함한, 하나 이상의 프로세서들 내에 구현될 수도 있다. 용어 "프로세서" 또는 "프로세싱 회로" 는 일반적으로 단독으로 또는 다른 로직 회로와 조합한, 임의의 전술한 로직 회로, 또는 프로세싱을 수행하는 별개의 하드웨어와 같은 임의의 다른 등가 회로를 지칭할 수도 있다.

[0275] 이러한 하드웨어, 소프트웨어, 및 펌웨어는 본 개시물에서 설명하는 여러 동작들 및 기능들을 지원하기 위해 동일한 디바이스 내에 또는 별개의 디바이스들 내에 구현될 수도 있다. 게다가, 설명되는 유닛들, 모듈들 또는 컴포넌트들 중 임의의 것이 별개의 공용가능한 로직 디바이스들과 함께 또는 개별적으로 구현될 수도 있다. 모듈들 또는 유닛들로서 상이한 피쳐들의 서술은 상이한 기능적 양태들을 강조하는 것으로 의도된 것이며 이러한 모듈들 또는 유닛들이 별개의 하드웨어 또는 소프트웨어 컴포넌트들에 의해 실현되어야 한다는 것을 반드시 암시하지는 않는다. 더 정확히 말하면, 하나 이상의 모듈들 또는 유닛들과 연관된 기능성은 별개의 하드웨어, 펌웨어, 및/또는 소프트웨어 컴포넌트들에 의해 수행되거나, 또는 공통 또는 별개의 하드웨어 또는 소프트웨어 컴포넌트들과 통합될 수도 있다.

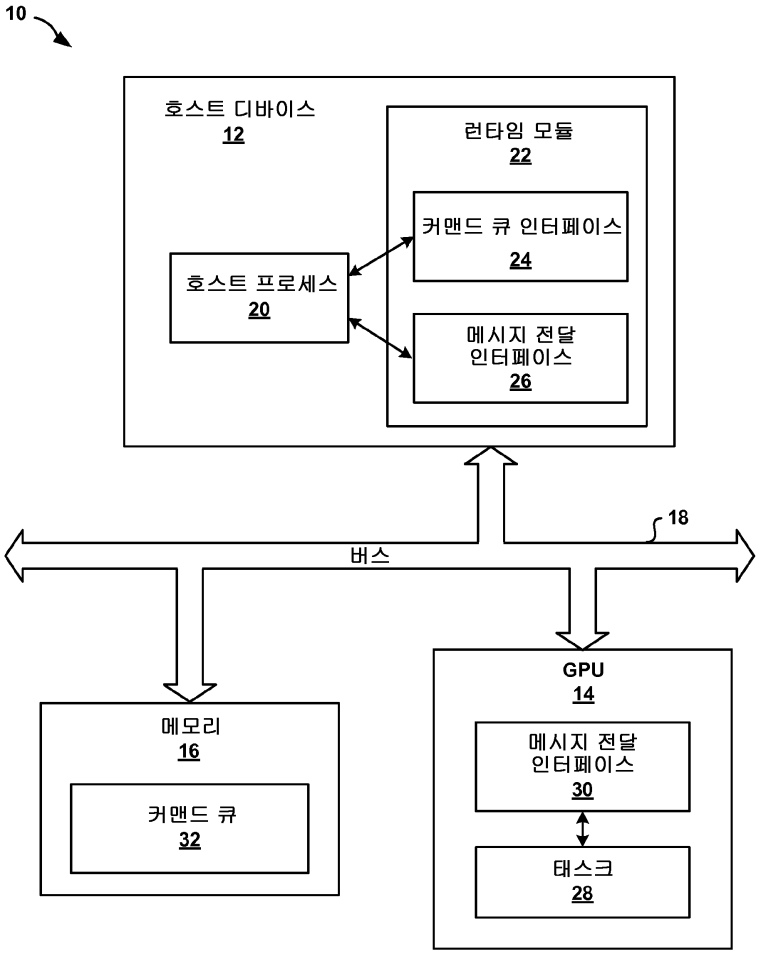
[0276] 본 개시물에서 설명하는 기법들은 또한 컴퓨터 판독가능 매체, 예컨대 명령들을 저장하는 컴퓨터 판독가능 저장 매체에 저장되거나, 수록되거나 또는 인코딩될 수도 있다. 컴퓨터 판독가능 매체에 내장되거나 또는 인코딩된 명령들은 예컨대, 명령들이 하나 이상의 프로세서들에 의해 실행될 때 하나 이상의 프로세서들로 하여금 본원에서 설명되는 기법들을 수행하도록 할 수도 있다. 컴퓨터 판독가능 저장 매체들은 랜덤 액세스 메모리 (RAM), 판독전용 메모리 (ROM), 프로그래밍가능 판독전용 메모리 (PROM), 소거가능한 프로그래밍가능 판독전용 메모리 (EPROM), 전자적으로 소거가능한 프로그래밍가능 판독전용 메모리 (EEPROM), 플래시 메모리, 하드 디스크, CD-ROM, 플로피 디스크, 카세트, 자기 매체들, 광학 매체들, 또는 유형인 다른 컴퓨터 판독가능 저장 매체들을 포함할 수도 있다.

[0277] 컴퓨터 판독가능 매체는 위에 리스트된 것과 같이, 유형의 저장매체에 대응하는 컴퓨터 판독가능 저장 매체들을 포함할 수도 있다. 컴퓨터 판독가능 매체들은 또한 예컨대, 통신 프로토콜에 따라 한 장소로부터 다른 장소로의 컴퓨터 프로그램의 전송을 용이하게 하는 임의의 매체를 포함한, 통신 매체들을 포함할 수도 있다. 이 방법에서, 어구 "컴퓨터 판독가능 매체들" 은 일반적으로 (1) 비일시적인 유형의 컴퓨터 판독가능 저장 매체, 및 (2) 일시적 (transitory) 신호 또는 반송파와 같은 비유형의 컴퓨터 판독가능 통신 매체에 대응할 수도

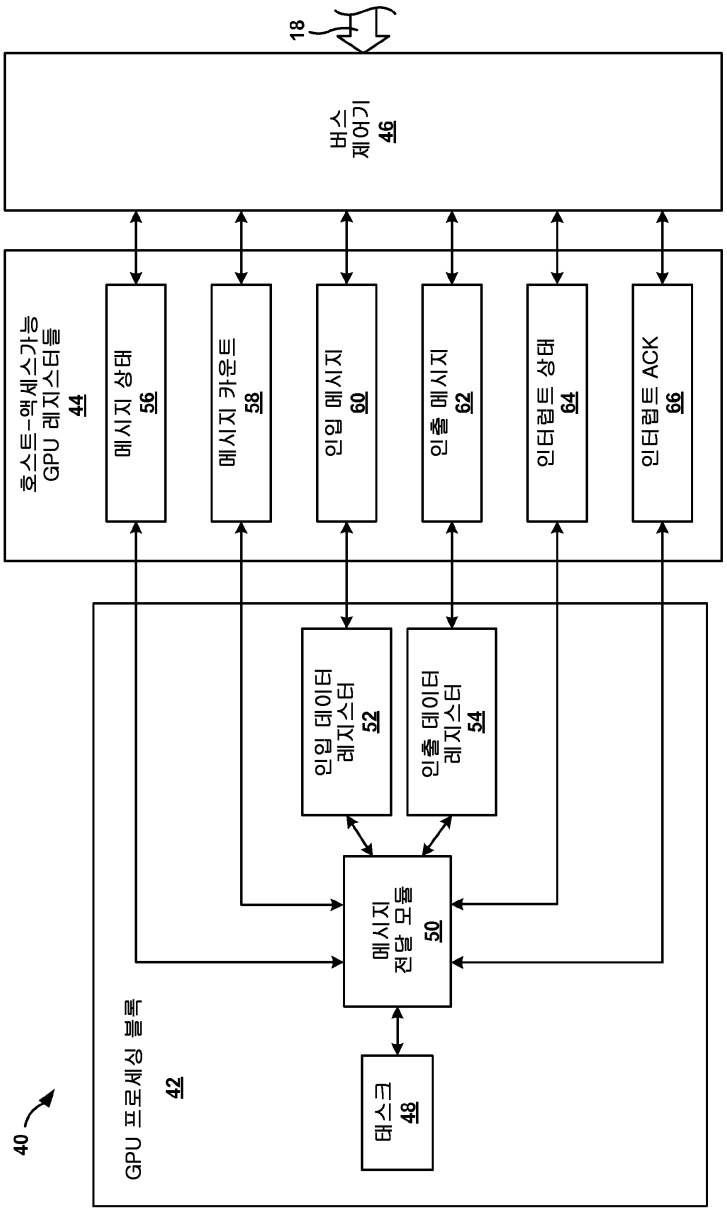
있다.

도면

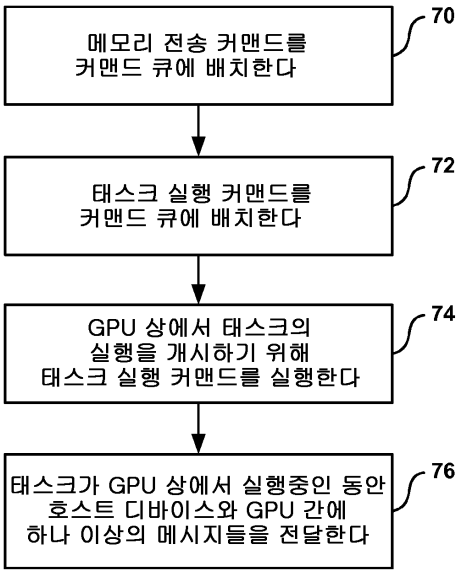
도면1


10
호스트 디바이스
12
런타임 모듈
22
커맨드 큐 인터페이스
24
호스트 프로세스
20
메시지 전달
인터페이스
26
18
버스
GPU
14
메모리
16
메시지 전달
인터페이스
30
커맨드 큐
32
태스크
28

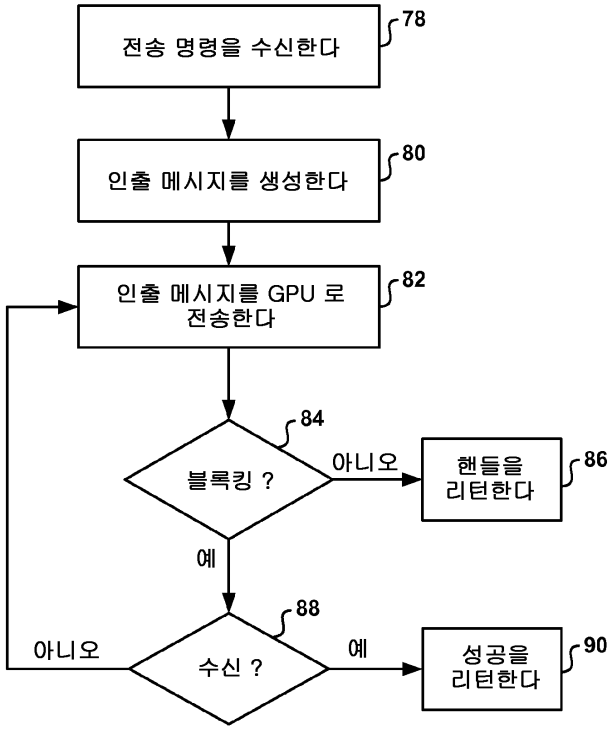
도면2


18
버스 제어기 46
호스트-액세스가능 GPU 레지스터들 44
메시지 상태 56
메시지 카운트 58
인입 메시지 60
인출 메시지 62
인터럽트 상태 64
인터럽트 ACK 66
인입 데이터 레지스터 52
인출 데이터 레지스터 54
메시지 전달 모듈 50
태스크 48
GPU 프로세싱 블록 42
40

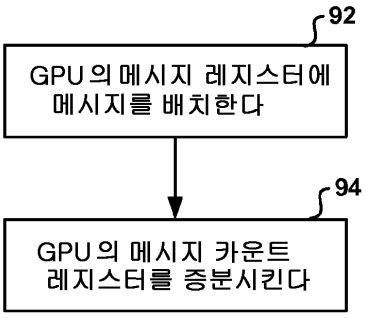
도면3


메모리 전송 커맨드를
커맨드 큐에 배치한다
70
태스크 실행 커맨드를
커맨드 큐에 배치한다
72
GPU 상에서 태스크의
실행을 개시하기 위해
태스크 실행 커맨드를 실행한다
74
태스크가 GPU 상에서 실행중인 동안
호스트 디바이스와 GPU 간에
하나 이상의 메시지들을 전달한다
76

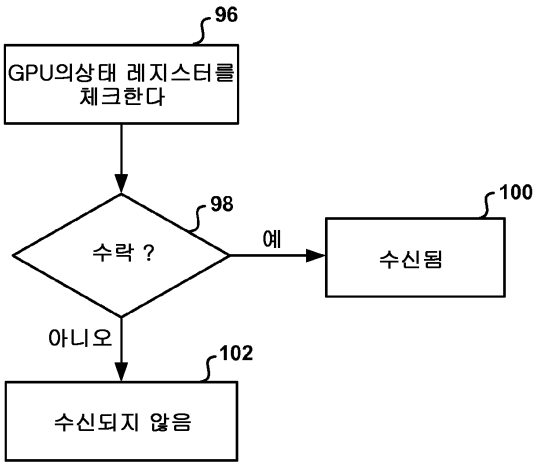
도면4


전송 명령을 수신한다
78
인출 메시지를 생성한다
80
인출 메시지를 GPU 로
전송한다
82
84
블록킹 ?
아니오
핸들을
리턴한다
86
예
88
아니오
수신 ?
예
성공을
리턴한다
90

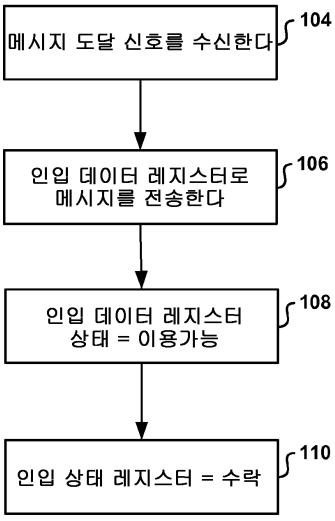
도면5


92
GPU의 메시지 레지스터에 메시지를 배치한다
94
GPU의 메시지 카운트 레지스터를 증분시킨다

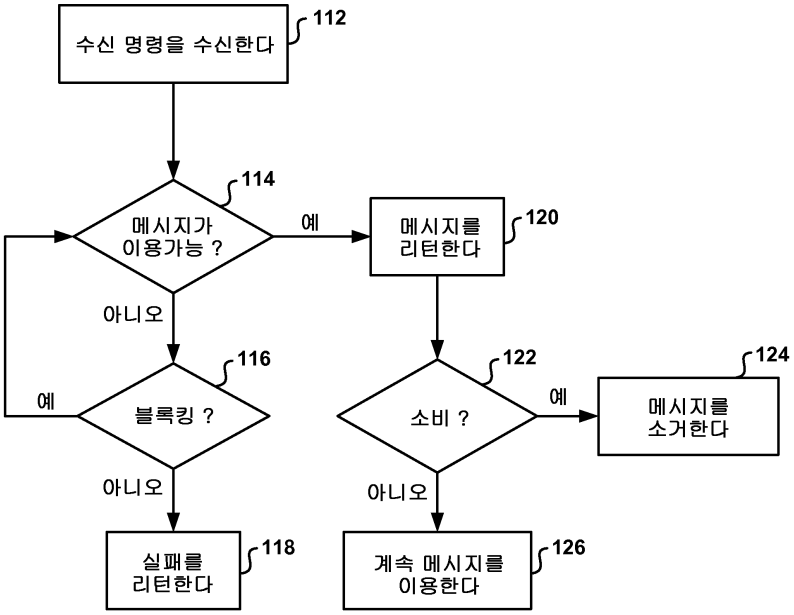
도면6


96
GPU의상태 레지스터를 체크한다
98
수락 ?
예
100
수신됨
아니오
102
수신되지 않음

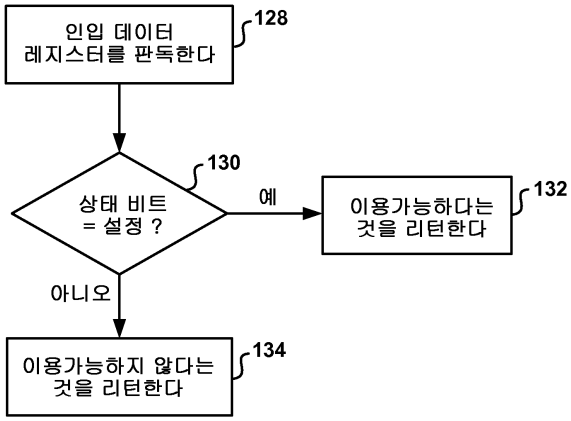
도면7


메시지 도달 신호를 수신한다
104
인입 데이터 레지스터로 메시지를 전송한다
106
인입 데이터 레지스터 상태 = 이용가능
108
인입 상태 레지스터 = 수락
110

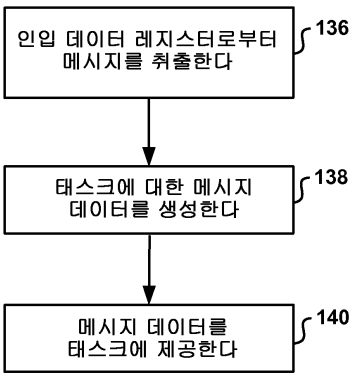
도면8


수신 명령을 수신한다
112
메시지가 이용가능 ?
114
예
메시지를 리턴한다
120
아니오
블록킹 ?
116
예
아니오
실패를 리턴한다
118
소비 ?
122
예
메시지를 소거한다
124
아니오
계속 메시지를 이용한다
126

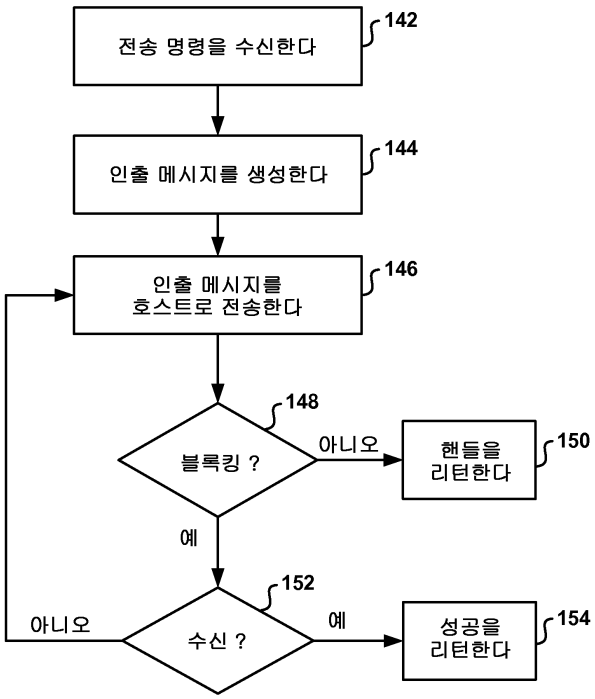
도면9


인입 데이터 레지스터를 판독한다
128
상태 비트 = 설정 ?
130
예
이용가능하다는 것을 리턴한다
132
아니오
이용가능하지 않다는 것을 리턴한다
134

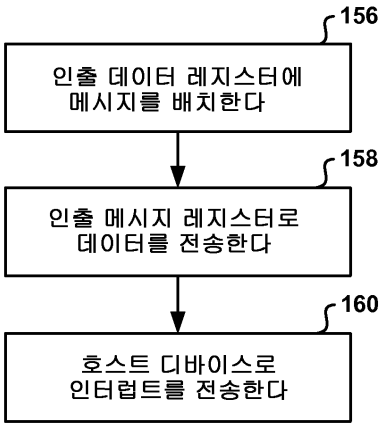
도면10


인입 데이터 레지스터로부터 메시지를 취출한다
136
태스크에 대한 메시지 데이터를 생성한다
138
메시지 데이터를 태스크에 제공한다
140

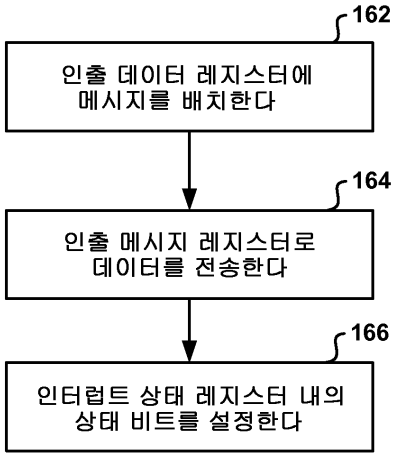
도면11


전송 명령을 수신한다
142
인출 메시지를 생성한다
144
인출 메시지를
호스트로 전송한다
146
148
블록킹 ?
아니오
핸들을
리턴한다
150
예
152
수신 ?
아니오
예
성공을
리턴한다
154

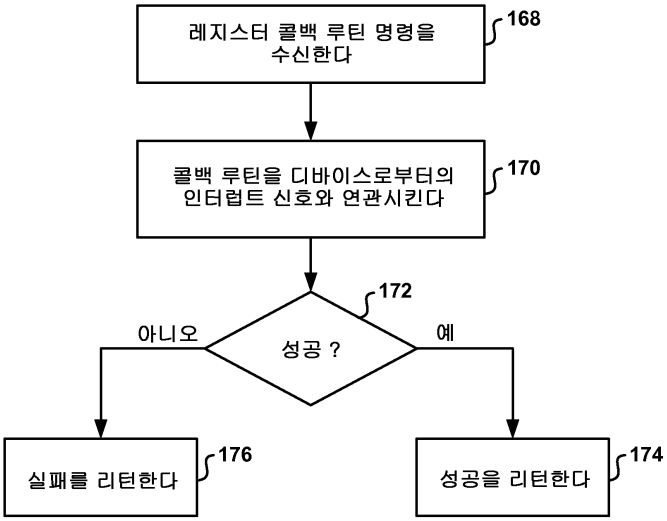
도면12


156
인출 데이터 레지스터에
메시지를 배치한다
158
인출 메시지 레지스터로
데이터를 전송한다
160
호스트 디바이스로
인터럽트를 전송한다

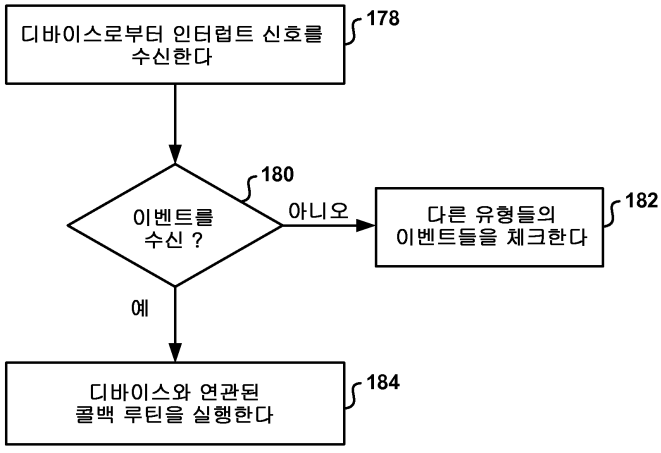
도면13



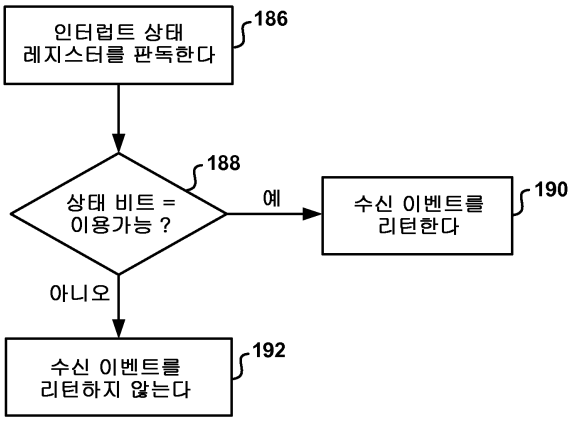
도면14



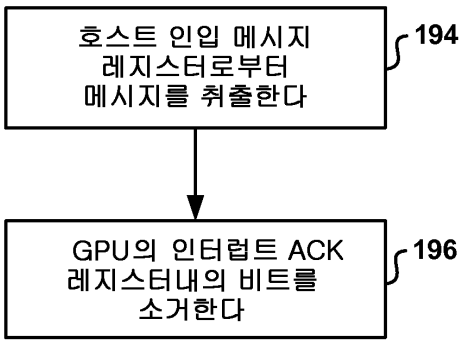
도면15



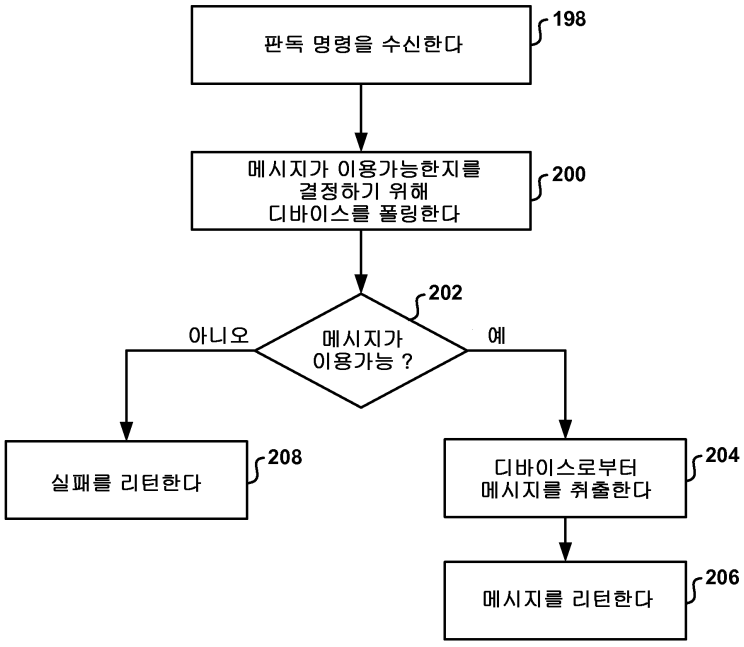
도면16



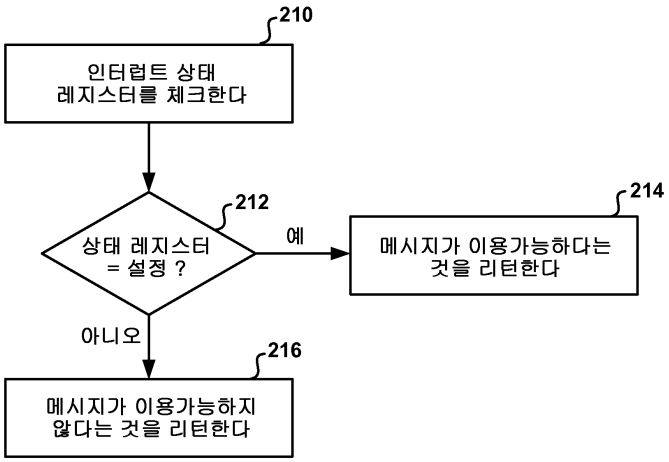
도면17



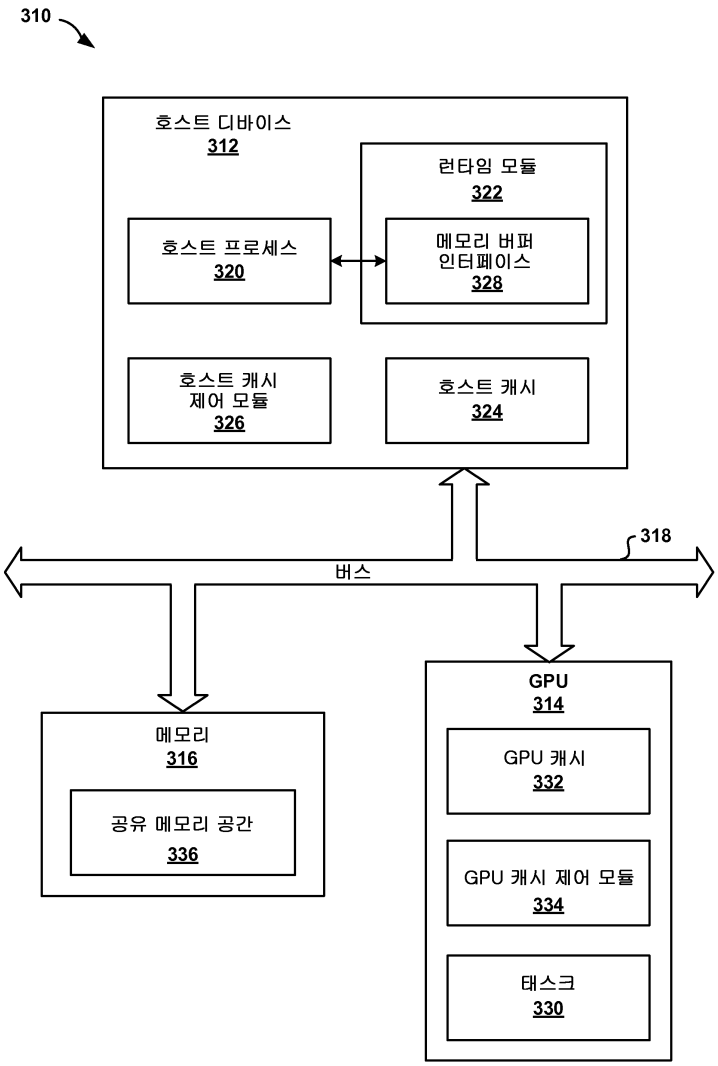
도면18



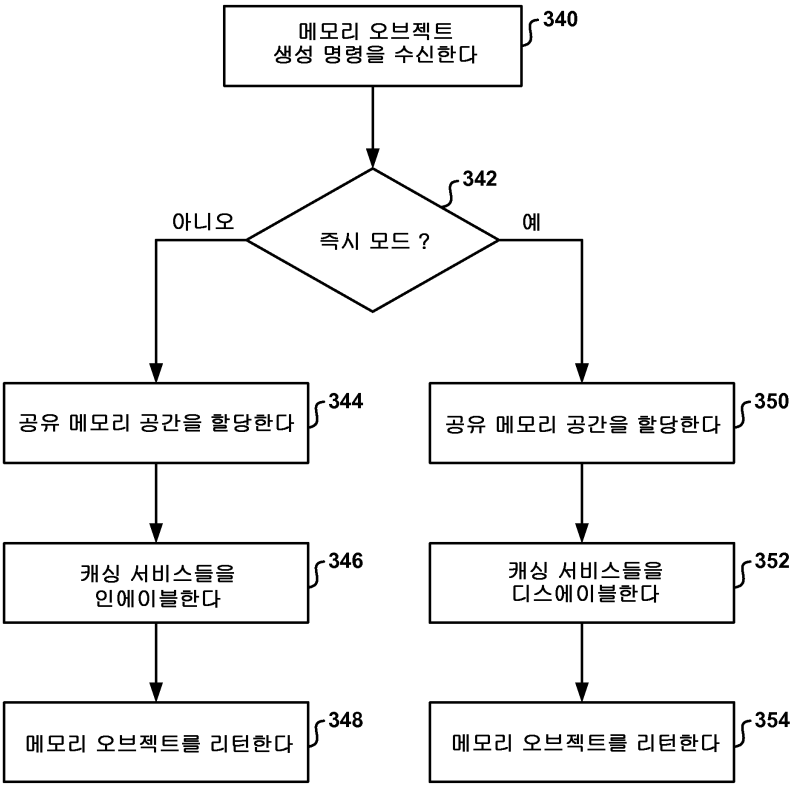
도면19


210
인터럽트 상태
레지스터를 체크한다
212
상태 레지스터
= 설정 ?
예
214
메시지가 이용가능하다는
것을 리턴한다
아니오
216
메시지가 이용가능하지
않다는 것을 리턴한다

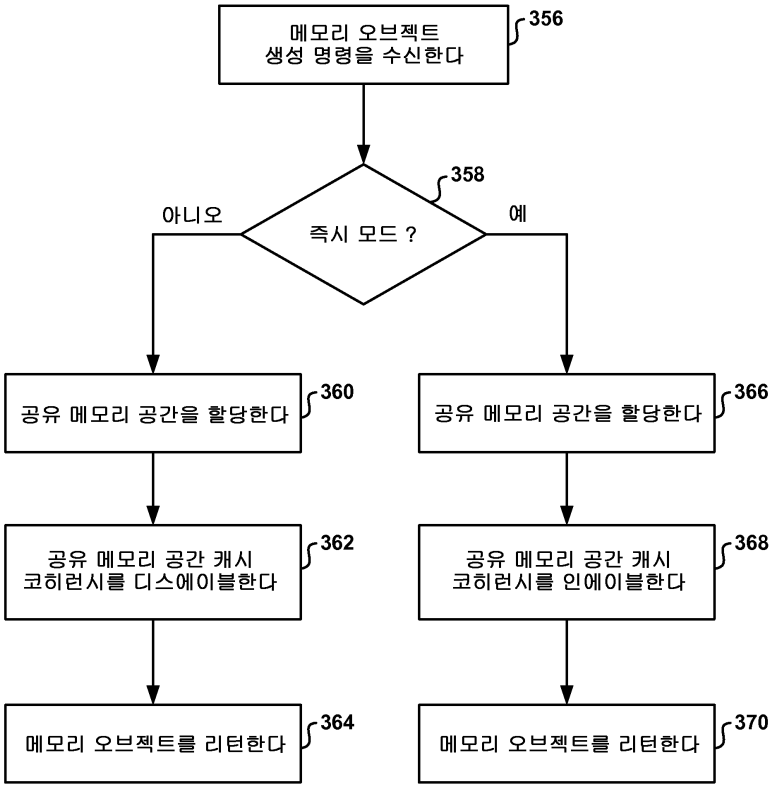
도면20


310
호스트 디바이스
312
런타임 모듈
322
호스트 프로세스
320
메모리 버퍼
인터페이스
328
호스트 캐시
제어 모듈
326
호스트 캐시
324
318
버스
GPU
314
메모리
316
공유 메모리 공간
336
GPU 캐시
332
GPU 캐시 제어 모듈
334
태스크
330

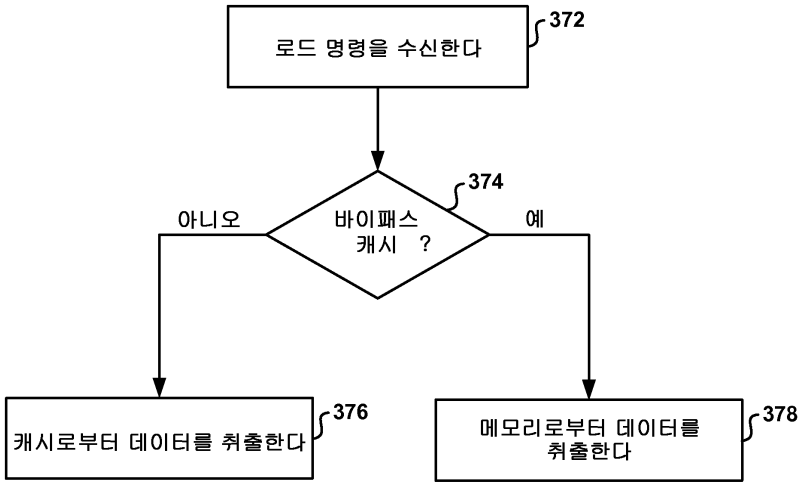
도면21


메모리 오브젝트
생성 명령을 수신한다
340
342
즉시 모드 ?
아니오
예
공유 메모리 공간을 할당한다
344
캐싱 서비스들을
인에이블한다
346
메모리 오브젝트를 리턴한다
348
공유 메모리 공간을 할당한다
350
캐싱 서비스들을
디스에이블한다
352
메모리 오브젝트를 리턴한다
354

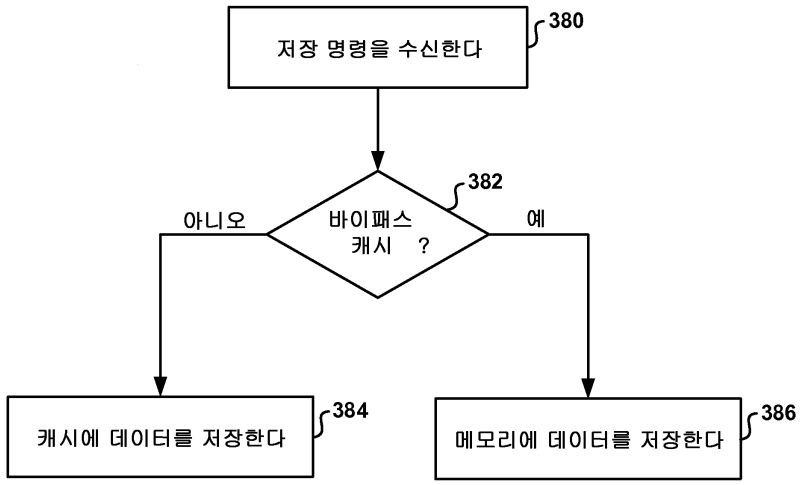
도면22


메모리 오브젝트
생성 명령을 수신한다
356
358
즉시 모드 ?
아니오
예
공유 메모리 공간을 할당한다
360
공유 메모리 공간 캐시
코히런시를 디스에이블한다
362
메모리 오브젝트를 리턴한다
364
공유 메모리 공간을 할당한다
366
공유 메모리 공간 캐시
코히런시를 인에이블한다
368
메모리 오브젝트를 리턴한다
370

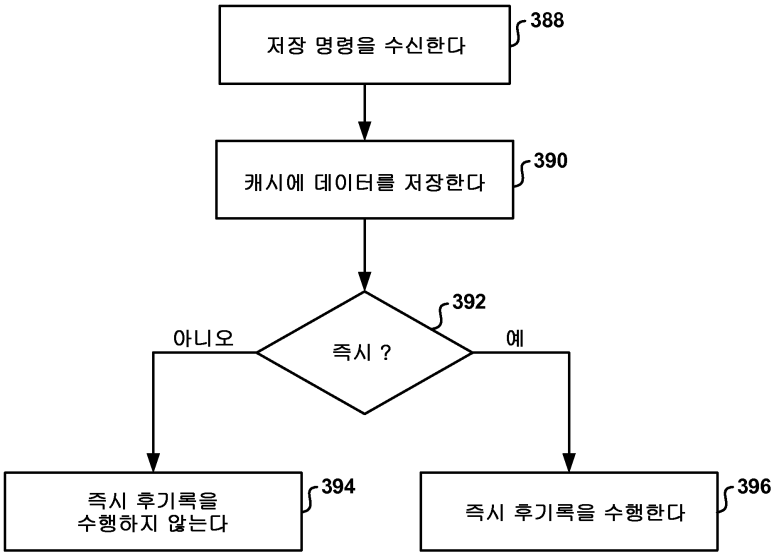
도면23


로드 명령을 수신한다
372
374
바이패스
캐시 ?
아니오
예
캐시로부터 데이터를 취출한다
376
메모리로부터 데이터를
취출한다
378

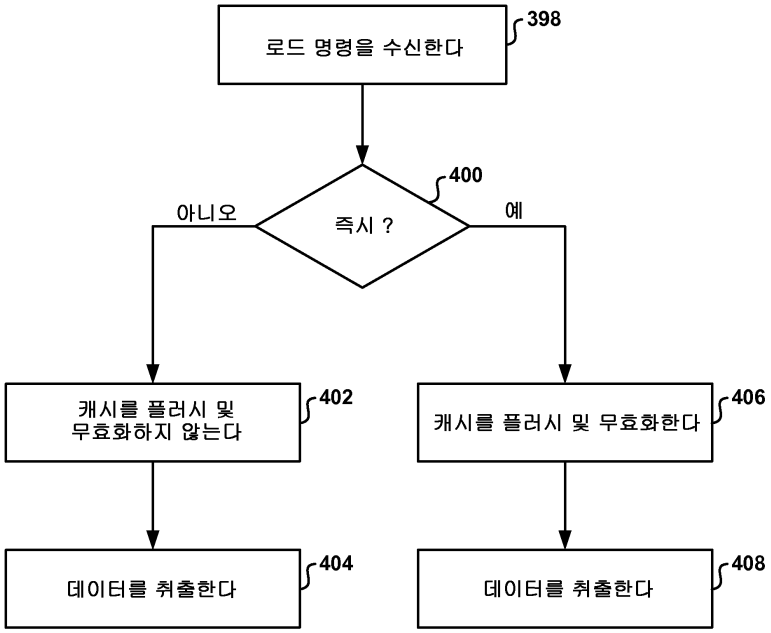
도면24


저장 명령을 수신한다
380
382
바이패스
캐시 ?
아니오
예
캐시에 데이터를 저장한다
384
메모리에 데이터를 저장한다
386

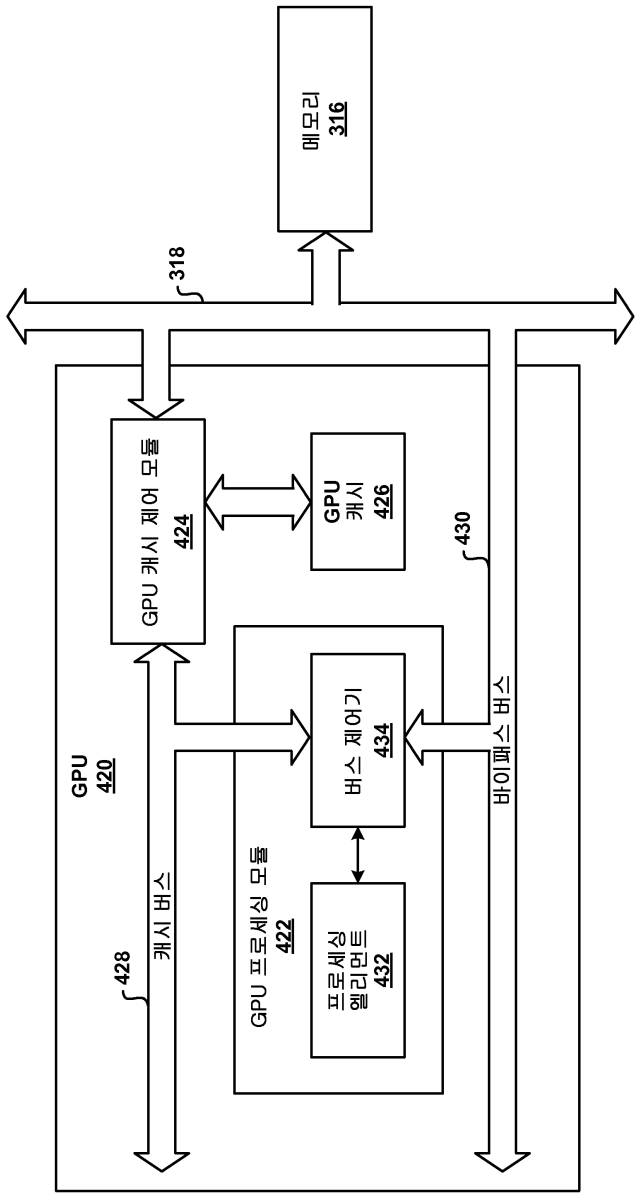
도면25


저장 명령을 수신한다
388
캐시에 데이터를 저장한다
390
392
즉시 ?
아니오
예
즉시 후기록을
수행하지 않는다
394
즉시 후기록을 수행한다
396

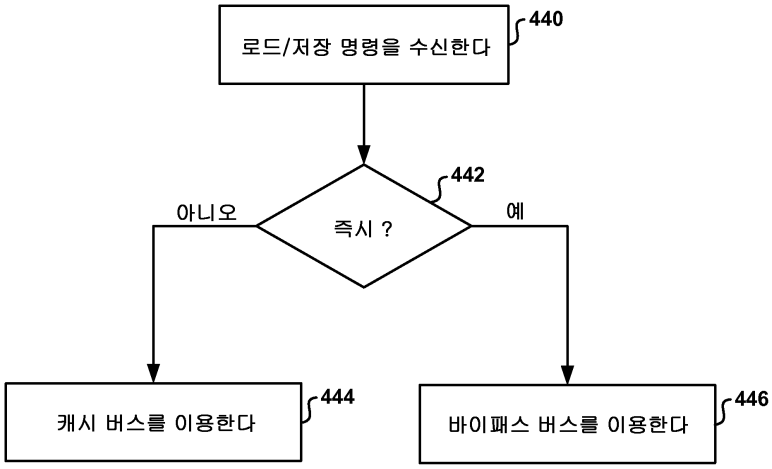
도면26


로드 명령을 수신한다
398
400
즉시 ?
아니오
예
캐시를 플러시 및
무효화하지 않는다
402
캐시를 플러시 및 무효화한다
406
데이터를 취출한다
404
데이터를 취출한다
408

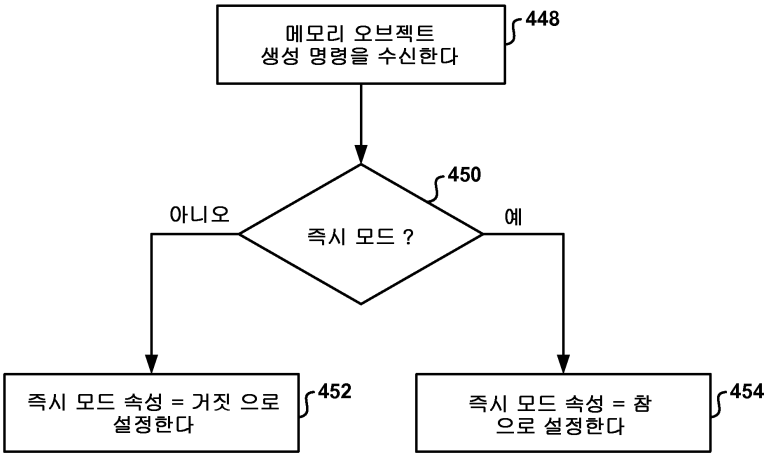
도면27


메모리
316
318
GPU
420
GPU 캐시 제어 모듈
424
GPU
캐시
426
430
버스 제어기
434
바이패스 버스
GPU 프로세싱 모듈
422
프로세싱
엘리먼트
432
428
캐시 버스

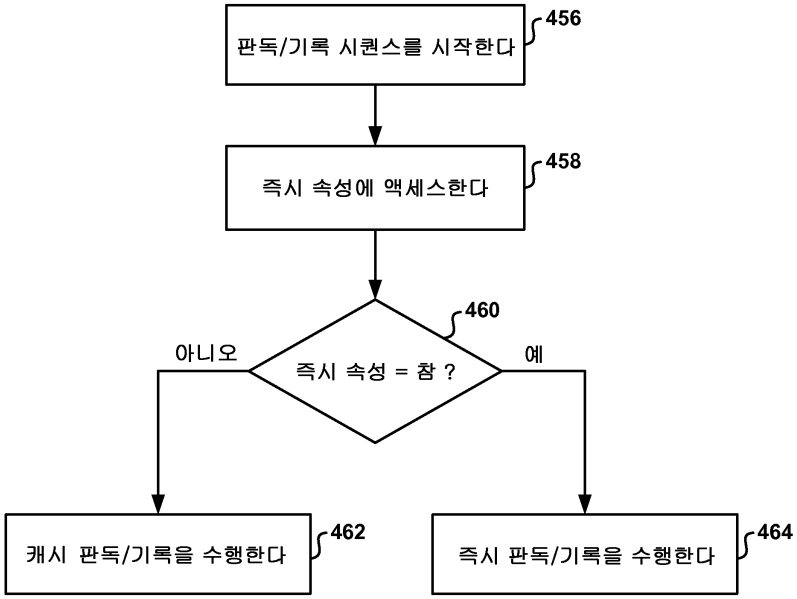
도면28


로드/저장 명령을 수신한다
440
즉시 ?
442
아니오
예
캐시 버스를 이용한다
444
바이패스 버스를 이용한다
446

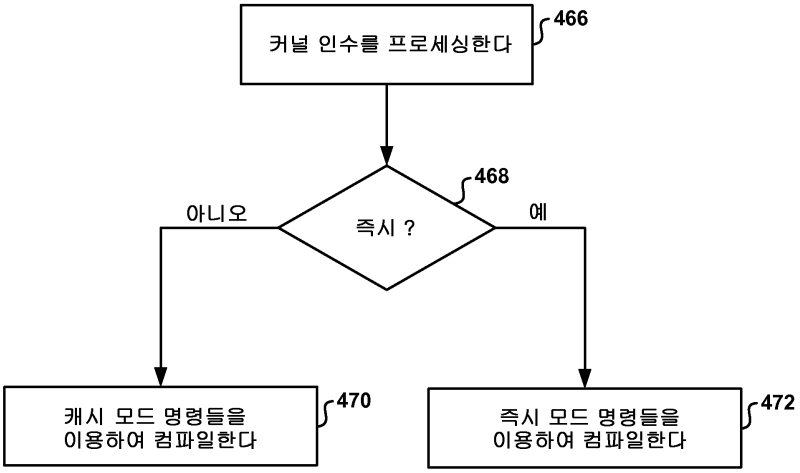
도면29


메모리 오브젝트
생성 명령을 수신한다
448
즉시 모드 ?
450
아니오
예
즉시 모드 속성 = 거짓 으로
설정한다
452
즉시 모드 속성 = 참
으로 설정한다
454

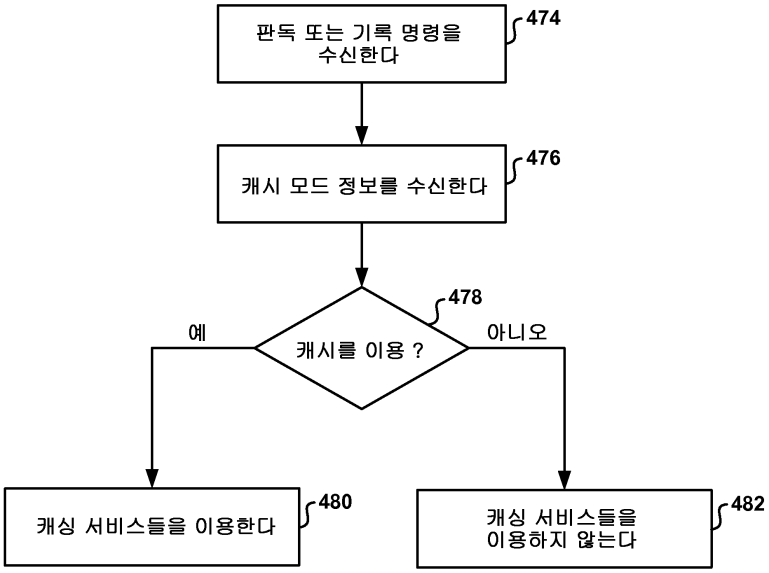
도면30


판독/기록 시퀀스를 시작한다
456
즉시 속성에 액세스한다
458
460
아니오
즉시 속성 = 참 ?
예
캐시 판독/기록을 수행한다
462
즉시 판독/기록을 수행한다
464

도면31


커널 인수를 프로세싱한다
466
468
아니오
즉시 ?
예
캐시 모드 명령들을
이용하여 컴파일한다
470
즉시 모드 명령들을
이용하여 컴파일한다
472

도면32


판독 또는 기록 명령을 수신한다
474
캐시 모드 정보를 수신한다
476
캐시를 이용 ?
478
예
아니오
캐싱 서비스들을 이용한다
480
캐싱 서비스들을 이용하지 않는다
482