(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2014-0015631
(43) 공개일자 2014년02월06일

(51) 국제특허분류(Int. Cl.)
G06F 9/48 (2006.01) *G06F 9/50* (2006.01)
(21) 출원번호 10-2014-7001412
(22) 출원일자(국제) 2012년05월18일
심사청구일자 2014년01월17일
(85) 번역문제출일자 2014년01월17일
(86) 국제출원번호 PCT/US2012/038659
(87) 국제공개번호 WO 2012/177343
국제공개일자 2012년12월27일
(30) 우선권주장
13/164,615 2011년06월20일 미국(US)

(71) 출원인
퀄컴 인코포레이티드
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
(72) 발명자
아르보 유까-빼까
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
(74) 대리인
특허법인코리아나

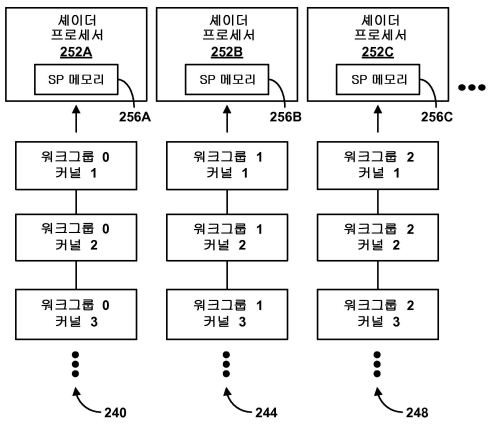
전체 청구항 수 : 총 29 항

(54) 발명의 명칭 **그래픽 프로세싱 유닛에서의 메모리 공유**

(57) 요 약

본 개시물의 양태들은 그래픽 프로세싱 유닛 (GPU) 을 사용하여 데이터를 프로세싱하는 방법에 관한 것이다. 본 개시물의 몇몇 양태들에 따르면, 이 방법은 셰이더 프로세서에 대한 실행 순서들을 정의하는 입력을 수신하는 단계를 포함하며, 실행 순서들은 복수의 커널 지정들 및 복수의 워크그룹 지정들을 포함한다. 이 방법은 또한 복수의 워크그룹 지정들 및 복수의 커널 지정들에서 식별되는 커널들의 워크그룹들을 셰이더 프로세서에 할당하는 단계를 포함할 수도 있다. 이 방법은 또한, 셰이더 프로세서에 의해, 복수의 워크그룹 지정들 및 복수의 커널 지정들에서 식별되는 커널들의 워크그룹들을 실행시켜 입력 데이터를 프로세싱하는 단계를 포함할 수도 있다.

대 표 도 - 도7



특허청구의 범위

청구항 1

그래픽 프로세싱 유닛 (GPU) 을 사용하여 데이터를 프로세싱하는 방법으로서,

셰이더 프로세서에 대한 실행 순서들을 정의하는 입력을 수신하는 단계로서, 상기 실행 순서들은 복수의 커널 지정들 (designations) 및 복수의 워크그룹 지정들을 포함하는, 상기 실행 순서들을 정의하는 입력을 수신하는 단계;

상기 복수의 워크그룹 지정들 및 상기 복수의 커널 지정들에서 식별된 커널들의 워크그룹들을 상기 셰이더 프로세서에 할당하는 단계; 및

상기 셰이더 프로세서에 의해, 입력 데이터를 프로세싱하기 위해 상기 복수의 워크그룹 지정들 및 상기 복수의 커널 지정들에서 식별된 상기 커널들의 워크그룹들을 실행하는 단계를 포함하는, 데이터를 프로세싱하는 방법.

청구항 2

제 1 항에 있어서,

상기 복수의 커널 지정들은 복수의 커널들을 식별하고, 상기 복수의 커널들의 각각의 커널은 상기 GPU 에 의해 실행되는 기능을 정의하는, 데이터를 프로세싱하는 방법.

청구항 3

제 1 항에 있어서,

상기 복수의 워크그룹 지정들은 복수의 워크그룹들을 식별하고, 상기 복수의 워크그룹들의 각각의 워크그룹은 상기 GPU 에 의해 입력 데이터에 대해 실행되는 명령들을 포함하는, 데이터를 프로세싱하는 방법.

청구항 4

제 1 항에 있어서,

상기 실행 순서들을 정의하는 입력을 수신하는 단계는 제 1 커널의 제 1 워크그룹을 제 2 커널의 제 1 워크그룹에 링크시키는 지정을 수신하는 단계를 포함하며, 상기 제 1 커널의 제 1 워크그룹 및 상기 제 2 커널의 제 1 워크그룹은 실질적으로 동일한 입력 데이터와 연관되는, 데이터를 프로세싱하는 방법.

청구항 5

제 4 항에 있어서,

상기 실행 순서들을 실행하는 단계는, 상기 셰이더 프로세서에 의해, 상기 제 1 커널의 제 1 워크그룹 이후에 상기 제 2 커널의 제 1 워크그룹을 실행하는 단계를 포함하는, 데이터를 프로세싱하는 방법.

청구항 6

제 1 항에 있어서

상기 실행 순서들의 스트림을 실행하는 단계는 상기 복수의 워크그룹 지정들에서 식별된 2 개 이상의 워크그룹들 사이에 입력 데이터를 공유하는 단계를 포함하는, 데이터를 프로세싱하는 방법.

청구항 7

제 6 항에 있어서,

상기 2 개 이상의 워크그룹들 사이에 입력 데이터를 공유하는 단계는 상기 2 개 이상의 워크그룹들 중 제 1 워크그룹 및 상기 2 개 이상의 워크그룹들 중 제 2 워크그룹의 실행을 위해 상기 셰이더 프로세서의 로컬 메모리에 상기 입력 데이터를 보유하는 단계를 포함하는, 데이터를 프로세싱하는 방법.

청구항 8

제 1 항에 있어서,

사용자 입력에 응답하여, 애플리케이션 프로그래밍 인터페이스 (API) 를 이용하여 상기 실행 순서들을 정의하는 입력을 생성하는 단계를 더 포함하는, 데이터를 프로세싱하는 방법.

청구항 9

그래픽 프로세싱 유닛 (GPU) 으로서,

셰이더 프로세서에 대한 실행 순서들을 정의하는 입력을 수신하도록 구성된 시퀀서 모듈을 포함하며, 상기 실행 순서들은 복수의 커널 지정들 및 복수의 워크그룹 지정들을 포함하고;

상기 시퀀서 모듈은 상기 복수의 워크그룹 지정들 및 상기 복수의 커널 지정들에서 식별된 커널들의 워크그룹들을 상기 셰이더 프로세서에 할당하도록 구성되고;

상기 셰이더 프로세서는 입력 데이터를 프로세싱하기 위해 상기 복수의 워크그룹 지정들 및 상기 복수의 커널 지정들에서 식별된 커널들의 워크그룹들을 실행하도록 구성된, 그래픽 프로세싱 유닛.

청구항 10

제 9 항에 있어서,

상기 복수의 커널 지정들은 복수의 커널들을 식별하고, 상기 복수의 커널들의 각각의 커널은 상기 GPU 에 의해 실행되는 기능을 정의하는, 그래픽 프로세싱 유닛.

청구항 11

제 9 항에 있어서,

상기 복수의 워크그룹 지정들은 복수의 워크그룹들을 식별하고, 상기 복수의 워크그룹들의 각각의 워크그룹은 상기 GPU 에 의해 입력 데이터에 대해 실행되는 명령들을 포함하는, 그래픽 프로세싱 유닛.

청구항 12

제 9 항에 있어서,

상기 스퀀서 모듈은 제 1 커널의 제 1 워크그룹을 제 2 커널의 제 1 워크그룹에 링크시키는 지정을 수신하도록 추가로 구성되며, 상기 제 1 커널의 제 1 워크그룹 및 상기 제 2 커널의 제 1 워크그룹은 실질적으로 동일한 입력 데이터와 연관되는, 그래픽 프로세싱 유닛.

청구항 13

제 12 항에 있어서,

상기 셰이더 프로세서는 상기 제 1 커널의 제 1 워크그룹 이후에 상기 제 2 커널의 제 1 워크그룹을 실행하도록 추가로 구성된, 그래픽 프로세싱 유닛.

청구항 14

제 9 항에 있어서,

상기 셰이더 프로세서는 셰이더 프로세서 메모리를 더 포함하고, 상기 셰이더 프로세서는 상기 복수의 워크그룹 지정들에서 식별된 2 개 이상의 워크그룹들과 연관된 입력 데이터를 저장하도록 구성된, 그래픽 프로세싱 유닛.

청구항 15

제 14 항에 있어서,

상기 셰이더 프로세서 메모리는, 상기 2 개 이상의 워크그룹들과 연관된 상기 저장된 입력 데이터를 보유하고 상기 저장된 데이터를 상기 2 개 이상의 워크그룹들 사이에 공유하도록 추가로 구성된, 그래픽 프로세싱 유닛.

청구항 16

그래픽 프로세싱 유닛 (GPU) 으로서,

셰이더 프로세서에 대한 실행 순서들을 정의하는 입력을 수신하는 수단으로서, 상기 실행 순서들은 복수의 커널 지정들 및 복수의 워크그룹 지정들을 포함하는, 상기 입력을 수신하는 수단;

상기 복수의 워크그룹 지정들 및 상기 복수의 커널 지정들에서 식별된 커널들의 워크그룹들을 상기 셰이더 프로세서에 할당하는 수단; 및

입력 데이터를 프로세싱하기 위해 상기 복수의 워크그룹 지정들 및 상기 복수의 커널 지정들에서 식별된 상기 커널들의 워크그룹들을 실행하는 수단을 포함하는, 그래픽 프로세싱 유닛.

청구항 17

제 16 항에 있어서,

상기 복수의 커널 지정들은 복수의 커널들을 식별하고, 상기 복수의 커널들의 각각의 커널은 상기 GPU 에 의해 실행되는 기능을 정의하는, 그래픽 프로세싱 유닛.

청구항 18

제 16 항에 있어서,

상기 복수의 워크그룹 지정들은 복수의 워크그룹들을 식별하고, 상기 복수의 워크그룹들의 각각의 워크그룹은 상기 GPU 에 의해 입력 데이터에 대해 실행되는 명령들을 포함하는, 그래픽 프로세싱 유닛.

청구항 19

제 16 항에 있어서,

상기 입력을 수신하는 수단은 제 1 커널의 제 1 워크그룹을 제 2 커널의 제 1 워크그룹에 링크시키는 지정을 수신하도록 추가로 구성되며, 상기 제 1 커널의 제 1 워크그룹 및 상기 제 2 커널의 제 1 워크그룹은 실질적으로 동일한 입력 데이터와 연관되는, 그래픽 프로세싱 유닛.

청구항 20

제 19 항에 있어서,

상기 실행하는 수단은 상기 제 1 커널의 제 1 워크그룹 이후에 상기 제 2 커널의 제 1 워크그룹을 실행하도록 추가로 구성된, 그래픽 프로세싱 유닛.

청구항 21

제 16 항에 있어서,

상기 실행하는 수단은 상기 복수의 워크그룹 지정들에서 식별된 2 개 이상의 워크그룹들과 연관된 입력 데이터를 저장하는 수단을 더 포함하는, 그래픽 프로세싱 유닛.

청구항 22

제 21 항에 있어서,

상기 입력 데이터를 저장하는 수단은, 상기 2 개 이상의 워크그룹들과 연관된 상기 저장된 입력 데이터를 보유하고 상기 저장된 데이터를 상기 2 개 이상의 워크그룹들 사이에 공유하도록 추가로 구성된, 그래픽 프로세싱 유닛.

청구항 23

명령들로 인코딩된 컴퓨터 판독가능 저장 매체로서,

상기 명령들은, 그래픽 프로세싱 유닛 (GPU) 을 갖는 컴퓨팅 디바이스의 하나 이상의 프로그래밍가능 프로세서들로 하여금,

셰이더 프로세서에 대한 실행 순서들을 정의하는 입력을 수신하게 하고, 상기 실행 순서들은 복수의 커널 지정들 및 복수의 워크그룹 지정들을 포함하고;

상기 복수의 워크그룹 지정들 및 상기 복수의 커널 지정들에서 식별된 커널들의 워크그룹들을 상기 셰이더 프로세서에 할당하게 하고;

상기 셰이더 프로세서에 의해, 입력 데이터를 프로세싱하기 위해 상기 복수의 워크그룹 지정들 및 상기 복수의 커널 지정들에서 식별된 상기 커널들의 워크그룹들을 실행하게 하는, 명령들로 인코딩된 컴퓨터 판독가능 저장 매체.

청구항 24

제 23 항에 있어서,

상기 복수의 커널 지정들은 복수의 커널들을 식별하고, 상기 복수의 커널들의 각각의 커널은 상기 GPU 에 의해 실행되는 기능을 정의하는, 명령들로 인코딩된 컴퓨터 판독가능 저장 매체.

청구항 25

제 23 항에 있어서,

상기 복수의 워크그룹 지정들은 복수의 워크그룹들을 식별하고, 상기 복수의 워크그룹들의 각각의 워크그룹은 상기 GPU 에 의해 입력 데이터에 대해 실행되는 명령들을 포함하는, 명령들로 인코딩된 컴퓨터 판독가능 저장 매체.

청구항 26

제 23 항에 있어서,

컴퓨팅 디바이스의 하나 이상의 프로그래밍가능 프로세서들로 하여금, 제 1 커널의 제 1 워크그룹을 제 2 커널의 제 1 워크그룹에 링크시키는 지정을 수신하게 하는 명령들을 더 포함하며, 상기 제 1 커널의 제 1 워크그룹 및 상기 제 2 커널의 제 1 워크그룹은 실질적으로 동일한 입력 데이터와 연관되는, 명령들로 인코딩된 컴퓨터 판독가능 저장 매체.

청구항 27

제 26 항에 있어서,

상기 실행 순서들을 실행하는 것은, 상기 셰이더 프로세서에 의해, 상기 제 1 커널의 제 1 워크그룹 이후에 상기 제 2 커널의 제 1 워크그룹을 실행하는 것을 포함하는, 명령들로 인코딩된 컴퓨터 판독가능 저장 매체.

청구항 28

제 23 항에 있어서,

상기 실행 순서들의 스트림을 실행하는 것은 상기 복수의 워크그룹 지정들에서 식별된 2 개 이상의 워크그룹들 사이에 입력 데이터를 공유하는 것을 포함하는, 명령들로 인코딩된 컴퓨터 판독가능 저장 매체.

청구항 29

제 28 항에 있어서,

상기 2 개 이상의 워크그룹들 사이에 입력 데이터를 공유하는 것은 상기 2 개 이상의 워크그룹들 중 제 1 워크그룹 및 상기 2 개 이상의 워크그룹들 중 제 2 워크그룹의 실행을 위해 상기 셰이더 프로세서의 로컬 메모리에 상기 입력 데이터를 보유하는 것을 포함하는, 명령들로 인코딩된 컴퓨터 판독가능 저장 매체.

명 세 서

기 술 분 야

[0001] 본 개시물은 그래픽 프로세싱 유닛 (GPU) 을 사용하여 데이터를 프로세싱하는 것에 관한 것이다.

배 경 기 술

[0002] 그래픽 프로세싱 디바이스들은 다양한 이미지 프로세싱 또는 다른 범용 프로세싱 애플리케이션들을 실행하도록 구현될 수도 있다. 예를 들어, 그래픽 프로세싱 유닛 (GPU, 때때로 범용 그래픽 프로세싱 유닛 (GPGPU) 이라고 지칭됨) 은 컬러 보정 알고리즘들, 안면 검출 알고리즘들, 패턴 인식 알고리즘들, 증강 현실 애플리케이션들, 다양한 알고리즘 애플리케이션들 (예컨대, 웨이블릿 변환들, 푸리에 변환들 등), 또는 다양한 기타 애플리케이션들과 같은 고도의 패러렐리즘으로부터 이익을 얻는 애플리케이션들을 실행할 수도 있다.

[0003] 일반적으로, GPU들은 GPU 에 상주하는 하나 이상의 셰이더 프로세서들을 사용하여 셰이더 명령들로 지칭될 수도 있는 일련의 명령들을 프로세싱하도록 디자인된다. 예시적 이미지 프로세싱 애플리케이션에서, 셰이더 명령들은 이미지를 형성하는 픽셀들 상에서 셰이더 프로세서들에 의해 수행될 하나 이상의 수학적 연산들을 정의할 수도 있다. 픽셀에 셰이더 명령을 적용함으로써, 픽셀 값은 셰이더 명령에 의해 정의된 수학적 연산에 따라 변경 또는 평가된다.

[0004] 셰이더 명령들은 커널로서 알려진 셰이더 프로그램 코드 내에 조직될 수도 있다. 커널은 GPU 에 의해 수행되는 기능 또는 태스크를 정의할 수도 있다. 커널을 실행하기 위해, 프로그램 코드는 하나 이상의 워크그룹들 (예컨대, 워크 아이템들의 세트) 내에 조직되는 워크 아이템들 (예컨대, GPU 에서의 워크의 기본 유닛) 로 분할된다.

발명의 내용

과제의 해결 수단

[0005] 일반적으로, 본 개시물의 양태들은 그래픽 프로세싱에 대한 커널 및 워크그룹 실행 순서들의 생성 및 프로세싱에 관련된다. 커널 및 워크그룹 실행 순서들은 그래픽 프로세싱 유닛 (GPU) 의 셰이더 프로세서 (SP) 와 연관된 메모리 리소스들의 관리를 제공할 수도 있다. 예를 들어, 커널 및 워크그룹 실행 순서들은 SP 의 로컬 메모리 리소스들에 저장된 데이터가 상이한 커널들의 워크그룹들에 의해 공유되게 한다. 일 실시예에서, 본 개시물의 양태들은 그래픽 프로세싱 유닛 (GPU) 을 사용하여 데이터를 프로세싱하는 방법에 관한 것이다. 이 방법은 셰이더 프로세서에 대한 실행 순서들을 정의하는 입력을 수신하는 단계를 포함하며, 실행 순서들은 복수의 커널 지정들 및 복수의 워크그룹 지정들을 포함한다. 이 방법은 또한 복수의 워크그룹 지정들 및 복수의 커널 지정들에서 식별되는 커널들의 워크그룹들을 셰이더 프로세서에 할당하는 단계를 포함한다. 이 방법은 또한, 셰이더 프로세서에 의해, 복수의 워크그룹 지정들 및 복수의 커널 지정들에서 식별되는 커널들의 워크그룹들을 실행시켜 입력 데이터를 프로세싱하는 단계를 포함한다.

[0006] 다른 실시예에서, 본 개시물의 양태들은 시퀀서 모듈을 포함하는 그래픽 프로세싱 유닛 (GPU) 에 관한 것이다. 이 시퀀서 모듈은 셰이더 프로세서에 대한 실행 순서들을 정의하는 입력을 수신하도록 구성되며, 실행 순서들은 복수의 커널 지정들 및 복수의 워크그룹 지정들을 포함한다. 시퀀서 모듈은 또한 복수의 워크그룹 지정들 및 복수의 커널 지정들에서 식별되는 커널들의 워크그룹들을 셰이더 프로세서에 할당하도록 구성된다. 셰이더 프로세서는 복수의 워크그룹 지정들 및 복수의 커널 지정들에서 식별되는 커널들의 워크그룹들을 실행시켜 입력 데이터를 프로세싱하도록 구성된다.

[0007] 다른 실시예에서, 본 개시물의 양태들은 컴퓨팅 디바이스의 하나 이상의 프로그래밍가능 프로세서들로 하여금 셰이더 프로세서에 대한 실행 순서들을 정의하는 입력을 수신하게 하는 명령들로 인코딩된 컴퓨터 판독가능 저장 매체에 관한 것이며, 실행 순서들은 복수의 커널 지정들 및 복수의 워크그룹 지정들을 포함한다. 명령들은 또한 컴퓨팅 디바이스의 하나 이상의 프로그래밍가능 프로세서들로 하여금 복수의 워크그룹 지정들 및 복수의 커널 지정들에서 식별되는 커널들의 워크그룹들을 셰이더 프로세서에 할당하게 한다. 명령들은 또한 컴퓨팅 디바이스의 하나 이상의 프로그래밍가능 프로세서들로 하여금, 셰이더 프로세서에 의해, 복수의 워크그룹 지정들 및 복수의 커널 지정들에서 식별되는 커널들의 워크그룹들을 실행시켜 입력 데이터를 프로세싱하게 한다.

[0008] 다른 실시예에서, 본 개시물의 양태들은 셰이더 프로세서에 대한 실행 순서들을 정의하는 입력을 수신하는 수단을 포함하는 그래픽 프로세싱 유닛 (GPU) 에 관한 것이며, 실행 순서들은 복수의 커널 지정들 및 복수의 워크그룹 지정들을 포함한다. GPU 는 또한 복수의 워크그룹 지정들 및 복수의 커널 지정들에서 식별되는 커널들의 워크그룹들을 셰이더 프로세서에 할당하는 수단을 포함한다. GPU 는 또한, 셰이더 프로세서에 의해, 복수의 워크그룹 지정들 및 복수의 커널 지정들에서 식별되는 커널들의 워크그룹들을 실행시켜 입력 데이터를 프로세싱

하는 수단을 포함한다.

[0009] 하나 이상의 실시예들의 세부사항들은 첨부된 도면 및 하기의 설명에서 설명된다. 다른 특징들, 목적들 및 이점들은 설명 및 도면들로부터 그리고 청구범위로부터 자명할 것이다.

도면의 간단한 설명

[0010] 도 1 은 본 개시물의 양태들을 구현하도록 구성될 수도 있는 그래픽 프로세싱 유닛 (GPU) 을 갖춘 컴퓨팅 디바이스를 예시한 블록도이다.

도 2 는, 본 개시물의 양태들에 따라, GPU 에 의해 실행될 수도 있는, 이미지를 프로세싱하는 관련된 명령들과 함께 이미지 데이터를 갖는 예시적 이미지를 예시한 블록도이다.

도 3 은, 본 개시물의 양태들에 따라, GPU 에 의해 실행될 수 있는 3 개의 커널들을 갖는 애플리케이션을 예시한 블록도이다.

도 4 는 본 개시물의 양태들을 실행하도록 구성될 수도 있는 GPU 를 예시한 블록도이다.

도 5 는, 본 개시물의 양태들에 따라, 제 1 커널, 제 2 커널, 및 제 3 커널 (146) 로 이루어진 워크그룹들을 분포시키도록 구성된 시퀀서 모듈의 실시예를 예시한 블록도이다.

도 6 은, 본 개시물의 양태들에 따라, 제 1 커널, 제 2 커널, 및 제 3 커널로 이루어진 워크그룹들을 분포시키도록 구성된 시퀀서 모듈의 실시예를 예시한 블록도이다.

도 7 은 본 개시물의 양태들에 따라 실행 순서들의 스트림들을 할당하는 실시예를 예시한 블록도이다.

도 8 은 본 개시물의 양태들에 따라 실행 순서들의 하나 이상의 스트림들을 생성하고 실행하는 방법을 예시한 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0011] 본 개시물의 양태들은 전반적으로 GPGPU 로서 구현될 수도 있는 GPU 의 셰이더 프로세서 (SP) 에 의한 실행을 위해 셰이더 명령들의 스트림들을 정의하고 생성하는 것에 관한 것이다. 일반적으로, GPU들은 하나 이상의 셰이더 프로그램들 (본 명세서에서 "커널들" 로 지칭됨) 을 실행하도록 디자인된 복수의 SP들을 포함한다. 커널들은 다양한 입력 데이터를 분석하거나 수정하도록 구현될 수 있는 기능들을 정의한다. 실시예들은 상대적으로 큰 수치 데이터 세트들을 병렬로 프로세싱하는 기능들을 포함한다. 이미지 프로세싱 콘텍스트에서, 기능들은, 예를 들어 컬러 보정 알고리즘들, 안면 검출 알고리즘들, 또는 증강 현실 애플리케이션들을 실행하기 위한 기능들을 포함할 수도 있다. 다른 실시예들은 변환 기능들, 광선 추적법 (ray tracing) 에 대한 기능들, 또는 다양한 다른 기능들을 포함한다.

[0012] 커널들은 워크그룹들 내에 그룹화된 개별 워크 아이템들 (예컨대, GPU 에서의 워크의 기본 유닛) 을 포함한다. GPU 가 이미지 (예컨대, 비디오 데이터의 프레임, 컴퓨터 생성된 그래픽 이미지, 정지 이미지 등) 를 프로세싱하도록 구현된 실시예에서, 입력 데이터는 이미지이고, 워크 아이템들은 이미지의 픽셀들을 분석하거나 수정하도록 구성된다. 복수의 워크 아이템들은 워크그룹들 (예컨대, 워크 아이템들의 세트) 내에 조직될 수 있다. 따라서, 이미지 프로세싱 실시예에서, 워크그룹들은 이미지 내의 픽셀들의 특정 그룹에 관련된 명령들을 포함할 수도 있다.

[0013] 커널들을 실행할 때, GPU 는 SP 가 워크그룹을 실행할 수 있게 되기 전에 그 SP 의 로컬 메모리 내에 워크그룹과 연관된 입력 데이터를 로딩한다. 일반적으로, GPU 가 커널을 실행할 때, GPU 는 어떤 SP 가 특정 워크그룹을 실행하는 지를 식별하거나 제어하지 않고 커널의 워크그룹들을 SP들에 할당한다. 예를 들어, GPU 는 GPU 애플리케이션 개발자 (예컨대, 컴퓨터 프로그래머) 에 의해 구성가능하지 않은 고정된 패턴으로 워크그룹들을 SP들에 분포시키는 하드웨어를 포함할 수도 있다. 이러한 실시예에서, GPU 는 다음 커널로 이동하기 전에 특정 커널과 연관된 모든 워크그룹들을 균일하게 분포시키고 실행함으로써 다수의 커널들을 갖는 애플리케이션을 순차적으로 실행한다.

[0014] 워크그룹 데이터는 일반적으로 SP들 사이에 공유될 수 없다. 예를 들어, SP들은 일반적으로 이산적이고, GPU 내의 유닛들을 물리적으로 분리시키며, GPU 애플리케이션 개발자는 어떤 SP 가 특정 워크그룹을 실행하는지를 제어하지 않는다. 따라서, 동일하거나 또는 실질적으로 동일한 입력 데이터를 프로세싱하는 다수의 커널들을 갖는 애플리케이션에서, 특정 워크그룹과 연관된 입력 데이터는 상이한 SP들의 로컬 메모리에 로딩될 필요

가 없을 수도 있다. 예를 들어, GPU 의 어떤 SP 가 특정 커널의 특정 워크그룹을 실행하는지를 제어하는 능력 없이, 제 2 커널의 워크그룹과 동일한 입력 데이터를 갖는 제 1 커널의 워크그룹은 GPU 의 상이한 SP들에 의해 프로세싱될 수도 있다.

[0015] 본 개시물의 양태들은 커널 및 워크그룹 실행 순서들을 생성하고 프로세싱하는 것에 관련된다. 커널 및 워크그룹 실행 순서들은 GPU 의 SP 와 연관된 로컬 메모리 리소스들의 관리를 지원한다. 예를 들어, GPU 에 의해 실행되는 커널 및 워크그룹 실행 순서들은 SP 로컬 메모리에 저장된 데이터가 상이한 커널들의 워크그룹들에 의해 공유되게 한다. 커널 및 워크그룹 실행 순서들은 "명령 스트림들" 로 지칭될 수도 있다. 명령 스트림은 단일 SP 를 이용하여 하나의 커널의 하나의 워크그룹과 연관된 입력 데이터가 다수의 다른 커널들에 의해 공유되고 직렬로 실행될 수 있도록 워크그룹들 및 커널들을 함께 결부시키거나 또는 사실상 링크시킨다.

명령 스트림들을 생성함으로써, 입력 데이터는 로컬 SP 메모리에 남아 있을 수 있고, 다수의 커널들의 워크그룹들에 이용가능할 수 있다. 명령 스트림들을 생성하는 것은 메모리 대역폭 소비뿐 아니라 SP 산술 로직 유닛 (ALU) 동작들을 감소시키는데, 이는 ALU 동작들이 동일한 데이터를 다수 회 페치하는 데 필수적인 것은 아니기 때문이다.

[0016] 몇몇 실시예들에서, 커널 및 워크그룹 실행 순서들은 GPU 가 동일한 또는 실질적으로 동일한 입력 데이터를 프로세싱하는 다수의 커널들을 갖는 애플리케이션을 실행시키고 있을 때 유용하다. 일 예시적 구현형태에서, GPU 는 이미지 (예컨대, 비디오 데이터의 프레임, 컴퓨터 생성된 그래픽 이미지, 정지 이미지 등) 를 프로세싱하도록 구현될 수도 있다. 이 실시예에서, 워크 아이템은 이미지의 특정 픽셀에 관련된 명령에 대응할 수도 있다. 복수의 워크 아이템들은 이미지의 픽셀들의 특정 그룹에 관련된 명령들을 포함하는 워크그룹들 내에 조직될 수 있다. 픽셀들의 그룹과 연관된 워크그룹을 프로세싱할 때, GPU 는 SP 의 로컬 메모리 내의 픽셀들의 그룹과 연관된 이미지 데이터를 로딩한다.

[0017] GPU 의 어떤 SP 가 특정 워크그룹을 실행하는지를 제어하는 능력 없이, 다수의 커널들을 연속으로 실행하는 것은 동일한 입력 이미지 영역이 상이한 SP들에 의해 프로세싱되게 할 수도 있다. 예를 들어, 제 1 커널을 실행하기 위해, GPU 는 실행을 위한 GPU 의 SP들의 로컬 메모리 내에 한 번에 하나의 워크 그룹씩 전체 이미지와 연관된 데이터를 로딩한다. 제 1 커널을 실행한 후 제 2 커널을 실행하기 위해, GPU 는 실행을 위해 GPU 의 SP들의 로컬 메모리 내에 동일한 이미지 데이터를 리로딩한다. 따라서, 입력 이미지 데이터는 각각의 커널에 대해 한 번씩 로컬 SP 메모리 내에 다수 회 로딩된다. 전체 이미지에 대한 SP 로컬 메모리 입력 대역폭 소비는 커널들의 수만큼 승산된 이미지 데이터 사이즈와 거의 동일하다 (예컨대, 3 개의 커널들을 갖는 64 MB 이미지를 분석하는 프로그램은 3 x 64MB 또는 192 MB 의 대역폭이 소비되게 한다). 커널들과 그들의 실행되는 워크그룹들 사이에 어떠한 입력 데이터도 공유하지 않으면서, 상대적으로 다량의 메모리 대역폭이 소비된다.

[0018] 이미지 프로세싱 실시예에서, 명령 스트림들을 생성하고 실행하는 것은 이미지의 특정 부분과 연관된 데이터가 특정 SP 의 로컬 메모리 리소스들에 한 번 로딩되게 하고 다수의 커널들로 프로세싱되게 한다. 3 개의 커널들 (예컨대, 제 1 커널, 제 2 커널, 및 제 3 커널) 을 갖는 예시적 프로그램이 제공된다. 제 1 커널의 제 1 워크그룹과 연관된 데이터가 특정 SP 의 로컬 메모리 내에 로딩되고, 제 1 워크그룹이 그 SP 에 의해 실행된다.

추가로, 동일한 SP 에게 제 2 커널의 제 1 워크그룹을 후속으로 실행하고 그 뒤에 제 3 커널의 제 1 워크그룹이 이어지도록 지시하는 커널 및 워크그룹 실행 순서들을 포함하는 명령 스트림이 제공된다. 따라서, 제 1 워크그룹과 연관된 이미지 데이터는 제 2 커널 및 제 3 커널을 실행하기 전에 특정 SP 의 로컬 메모리 내에 로딩될 필요가 없다. 대신, 제 2 커널 및 제 3 커널은 제 1 커널 동안 이전에 로딩된 입력 데이터를 이용한다. 이 방식으로, 메모리 대역폭 소비가 감소할 수도 있는데, 이는 입력 이미지의 특정 영역과 연관된 데이터가 한 번만 로컬 SP 내에 로딩될 필요가 있고 다수의 커널들로 프로세싱될 수 있기 때문이다. 위에 제공된 3 개 커널 실시예에서, 메모리 대역폭 소비는 2/3 만큼 감소한다.

[0019] 명령 스트림들은 다양한 방식들로 정의될 수 있다. 몇몇 양태들에 따르면, 사용자는 명령 스트림들을 이용하는 것으로부터 이익을 얻을 후보 커널들을 식별한다. 예를 들어, 사용자는 동일한 입력 데이터를 다수 회 이용하는 커널들을 식별할 수도 있다. 명령 스트림들을 이용하는 것은 입력 데이터가 로컬 메모리 리소스들 내에 로딩될 필요가 있는 횟수를 감소시킴으로써 SP들의 로컬 메모리 리소스들을 관리하는 것을 도울 수도 있다.

[0020] 후보들을 식별한 후, 사용자는 GPU 에 의해 실행되는 프로그램에서 명령 스트림들을 정의할 수 있다. 예를 들어, GPU 애플리케이션 프로그래밍은 다수의 플랫폼들, 운영체제들, 및 하드웨어에서 구동할 수 있는 표준 소프트웨어 인터페이스를 제공하는 애플리케이션 프로그램 인터페이스 (API) 를 이용하여 애플리케이션 개발자

(예컨대, 컴퓨터 프로그래머) 에 의해 수행되는 것이 일반적이다. API들의 실시예들은 오픈 그래픽 라이브러리 ("OpenGL", 2010 년 7 월 26 일에 발표되고 공개적으로 입수가능한 버전 4.1), 컴퓨터 통합형 디바이스 아키텍처 ("CUDA", 2010 년 9 월 17 일에 발표된 버전 3.2의 NVIDIA 코포레이션에 의해 개발됨), 및 DirectX (마이크로소프트, 인크에 의해 개발됨, 2009 년 10 월 27 일에 발표된 버전 11) 를 포함한다. 일반적으로, API 는 연관된 하드웨어에 의해 실행되는 커맨드들의 미리 정해진 표준화된 세트를 포함한다. API 커맨드들은 사용자가 GPU 의 하드웨어 콤포넌트들에게 그 하드웨어 콤포넌트들의 세부사항들에 대한 사용자 지식 없이 커맨드들을 실행할 것을 지시하게 한다.

[0021] 본 개시물의 양태들은 사용자가 명령 스트림들을 정의하게 하는 하나 이상의 API 커맨드들에 관련된다. 예를 들어, 하나 이상의 API 커맨드들은 디자인 환경에서 개발되고 작성될 수도 있다. 그 후, API 커맨드들은 전술된 API들과 같은 API 에 API 의 사용자들 (예컨대, 컴퓨터 프로그래머들) 을 위한 사전 구성된 옵션들로서 포함될 수도 있다.

[0022] 사용자는 애플리케이션의 개발 및 코딩 동안 GPU 에 의해 실행될 애플리케이션에서 명령 스트림들을 지정하도록 사전 구성된 명령 스트림 API 커맨드들을 구현할 수 있다. 예를 들어, 명령 스트림 API 커맨드들은 사용자가 특정 SP 에 의해 순차적으로 프로세싱될 다중 커널 애플리케이션의 상이한 커널들로부터 명령들 (예컨대, 하나 이상의 워크그룹들) 을 지정하게 한다. 명령 스트림 지정들을 포함하는 애플리케이션을 실행할 때, GPU 는 수신된 지정들에 따라 GPU 의 SP 에 명령들 (예컨대, 상이한 커널들의 워크그룹들) 을 라우팅한다.

[0023] 다른 실시예에서, 자동화된 시스템이 명령 스트림들을 생성하도록 구현될 수도 있다. 예를 들어, 컴파일러 프로그램과 같은 프로그램은 다수의 커널들로 동일한 입력 데이터를 반복적으로 프로세싱하는 애플리케이션에 대한 명령 스트림들을 자동으로 생성할 수도 있다. 이 실시예에서, 프로그램은 각각의 커널의 명령들을 파티셔닝하여 명령들의 각각의 그룹이 미리 정해진 양의 입력 데이터 (명령들의 워크그룹) 에 대응하도록 할 수도 있다. 그 후, 프로그램은 하나의 커널의 명령들의 하나의 그룹과 연관된 입력 데이터가 단일 SP 를 이용하여 다수의 다른 커널들에 의해 공유될 수 있고 직렬로 실행될 수 있도록 상이한 커널들로부터의 명령들의 그룹들을 링크시킴으로써 명령 스트림들을 생성할 수도 있다.

[0024] 비제한적인 이미지 프로세싱 실시예에서, GPU 애플리케이션 개발자는 입력 이미지 및 그 이미지를 프로세싱하기 위한 3 개의 커널들을 이용하여 명령 스트림 생성 프로그램을 제공할 수도 있다. 명령 스트림 생성 프로그램은 이미지의 사전 정의된 공간 영역들에 기초하여 명령 스트림들을 자동으로 생성할 수 있다. 예를 들어, 명령 스트림 생성 프로그램은 3 개의 커널들 각각의 명령들을 명령들의 그룹들로 파티셔닝할 수도 있는데, 명령들의 각각의 그룹은 입력 이미지의 사전 정의된 영역에 대응한다. 그 후, 명령 스트림 생성 프로그램은 동일한 입력 이미지 영역과 연관된 커널들 각각의 명령들의 그룹을 링크시킴으로써 명령 스트림들을 생성할 수 있다.

[0025] 예를 들어, 컴파일러 프로그램 또는 다른 개발자/분석 프로그램과 같은 프로그램은 명령 스트림들을 구현하는 것으로부터 이익을 얻을 후보 커널들을 식별할 수도 있다. 예를 들어, 프로그램은 메모리 액세스 패턴들을 모니터링할 수도 있고, 1 개를 초과하는 커널에 의해 이용되는 데이터를 식별할 수도 있다. 이 실시예에서, 프로그램은 다수의 커널들을 갖는 애플리케이션의 워크그룹들과 연관된 판독/기록 액세스 패턴들을 모니터링 및 로그할 수도 있다. 로그 후, 프로그램은 각각의 커널의 각각의 워크그룹들의 입력/출력 의존성을 검출할 수 있다. 즉, 프로그램은 다수의 커널 애플리케이션의 어떤 워크그룹들이 동일한 입력 데이터를 이용하는지를 결정할 수 있다. 이 데이터 공유 정보에 기초하여, 명령 스트림 지정들은 특정 SP 에 의해 순차적으로 프로세싱되고 있는 다중 커널 애플리케이션의 상이한 커널들로부터의 워크그룹들을 용이하게 하는 애플리케이션 내에 삽입될 수 있다. 예를 들어, 코드는 GPU 에게 동일한 SP 에 의해 순차적으로 실행될 동일한 입력 데이터를 공유하는 상이한 커널들로부터의 워크그룹들을 실행하도록 명령하는 애플리케이션 내에 삽입될 수 있다.

[0026] 후보 커널들을 식별하고 상이한 커널들의 워크그룹들을 명령 스트림에 지정하는 프로그램은 애플리케이션 개발 동안 실행될 수도 있고, 또는 GPU 애플리케이션 실행 동안 "즉석에서 (on the fly)" 실행될 수도 있다. 예를 들어, 몇몇 양태들에 따르면, GPU 애플리케이션 개발자는 후보 커널들을 식별하고 상이한 커널들의 워크그룹들을 명령 스트림에 지정하는 프로그램을 구현할 수도 있다. 그 후, 개발된 GPU 애플리케이션은 GPU 에 의해 실행될 명령 스트림 지정들을 포함할 수도 있다. 다른 실시예에서, 컴퓨팅 디바이스의 호스트 프로세서 또는 GPU 는 후보 커널들을 식별하고, GPU 애플리케이션을 실행하면서 상이한 커널들의 워크그룹들을 "즉석에서 (on the fly)" 명령 스트림에 지정하는 프로그램을 구현할 수도 있다.

[0027] 도 1 은 본 개시물의 양태들을 구현하도록 구성될 수도 있는 컴퓨팅 디바이스 (20) 를 예시한 블록도이다.

도 1 에 도시된 바와 같이, 컴퓨팅 디바이스 (20) 는 호스트 프로세서 (24), 스토리지 디바이스 (28), 메모리 (32), 네트워크 모듈 (36), 사용자 인터페이스 (40), 및 디스플레이 (44) 를 포함한다. 컴퓨팅 디바이스 (20) 는 또한 그래픽 프로세싱 유닛 (GPU)(48) 을 포함한다.

[0028] 컴퓨팅 디바이스 (20) 는, 몇몇 실시예들에서 포터블 컴퓨팅 디바이스 (예컨대, 모바일 폰, 넷북, 랩톱, 태블릿 디바이스, 디지털 미디어 플레이어, 게이밍 디바이스, 또는 다른 포터블 컴퓨팅 디바이스) 를 포함할 수도 있고 또는 그의 부분일 수도 있다. 대안으로, 컴퓨팅 디바이스 (20) 는 데스크톱 컴퓨터 또는 다른 고정된 컴퓨팅 디바이스로서 구성될 수도 있다. 컴퓨팅 디바이스 (20) 는 명료성을 위해 도 1 에 도시되지 않은 추가 콤포넌트들을 포함할 수도 있다. 예를 들어, 컴퓨팅 디바이스 (20) 는 컴퓨팅 디바이스 (20) 의 콤포넌트들 사이에 데이터를 이송하는 하나 이상의 통신 브리지들을 포함할 수도 있다. 또한, 도 1 에 도시된 컴퓨팅 디바이스 (20) 의 콤포넌트들은 컴퓨팅 디바이스 (20) 의 모든 실시예에서 필수적인 것은 아닐 수도 있다. 예를 들어, 사용자 인터페이스 (40) 및 디스플레이 (44) 는, 예컨대 컴퓨팅 디바이스 디바이스 (20) 가 데스크톱 컴퓨터인 경우의 실시예들에서, 컴퓨팅 디바이스 (20) 외부에 있을 수도 있다.

[0029] 호스트 프로세서 (24) 는 마이크로프로세서, 제어기, 디지털 신호 프로세서 (DSP), 주문형 반도체 (ASIC), 필드 프로그래밍가능 게이트 어레이 (FPGA), 또는 등가의 이산 또는 집적된 로직 회로 중 임의의 하나 이상을 포함할 수도 있다. 추가로, 본 개시물에서, 호스트 프로세서 (24) 에 기인된 기능들은 소프트웨어, 펌웨어, 하드웨어 또는 이들의 임의의 조합으로서 구현될 수도 있다.

[0030] 호스트 프로세서 (24) 는 컴퓨팅 디바이스 (20) 내에서의 실행을 위한 명령들을 프로세싱한다. 호스트 프로세서 (24) 는 스토리지 디바이스 (28) 에 저장된 명령들 또는 메모리 (32) 에 저장된 명령들을 프로세싱할 수도 있다. 예시적 애플리케이션들은 가시적 이미지들을 프로세싱하는 (예컨대, 이미지들을 필터링하는, 사전 정의된 피처들에 대한 이미지들을 분석하는 등) 애플리케이션들을 포함한다. 호스트 프로세서 (24) 는 사용자 인터페이스 (4) 를 통한 사용자의 선택에 기초하여 하나 이상의 애플리케이션들을 실행할 수도 있다. 몇몇 실시예들에서, 호스트 프로세서 (24) 는 사용자와의 상호작용 없이 하나 이상의 애플리케이션들을 실행할 수도 있다.

[0031] 본 개시물의 몇몇 양태들에 따르면, 그리고 GPU (48) 에 관하여 하기에서 보다 상세히 설명되는 바와 같이, 호스트 프로세서 (24) 는 GPU (48) 와 협력하여 하나 이상의 애플리케이션들과 연관된 다양한 태스크들을 실행할 수도 있다. 예를 들어, 호스트 프로세서 (24) 는 애플리케이션의 실행을 개시할 수도 있고, 그 애플리케이션과 연관된 어떤 프로세싱 기능들을 GPU (48) 에 오프로드하거나 또는 위임할 수도 있다. 일 실시예에서, 호스트 프로세서 (24) 는 이미지 프로세싱 애플리케이션의 실행을 개시할 수도 있고, 그 애플리케이션과 연관된 어떤 프로세싱 기능들을 GPU (48) 에 오프로드할 수도 있다.

[0032] 스토리지 디바이스 (28) 는 하나 이상의 컴퓨터 판독가능 스토리지 매체들을 포함할 수도 있다. 스토리지 디바이스 (28) 는 정보의 장기 스토리지를 위해 더 구성될 수도 있다. 몇몇 실시예들에서, 스토리지 디바이스들 (28) 은 불휘발성 스토리지 엘리먼트들을 포함할 수도 있다. 이러한 불휘발성 스토리지 엘리먼트들의 실시예들은 자기 하드디스크들, 광학 디스크들, 플로피 디스크들, 플래시 메모리들, 또는 EPROM 혹은 EEPROM 메모리들의 형태들을 포함할 수도 있다. 스토리지 디바이스 (28) 는, 몇몇 실시예들에서 비일시적 스토리지 매체로 간주될 수도 있다. 용어 "비일시적" 은 저장 매체가 반송파 또는 전파 신호로 구현되지 않음을 나타낼 수도 있다. 그러나, 용어 "비일시적" 은 스토리지 디바이스 (28) 가 탈착불가능함을 의미하도록 해석되어서는 안 된다. 일 실시예로서, 스토리지 디바이스 (28) 는 컴퓨팅 디바이스 (20) 로부터 제거될 수도 있고, 다른 디바이스로 이동될 수도 있다. 다른 실시예로서, 스토리지 디바이스 (28) 와 실질적으로 유사한 스토리지 디바이스가 컴퓨팅 디바이스 (20) 내에 삽입될 수도 있다.

[0033] 스토리지 디바이스 (28) 는 호스트 프로세서 (24) 또는 GPU (48) 에 의한 하나 이상의 애플리케이션들의 실행을 위한 명령들을 저장할 수도 있다. 스토리지 디바이스 (28) 는 또한 호스트 프로세서 (24) 또는 GPU (48) 에 의한 사용을 위한 데이터를 저장할 수도 있다. 예를 들어, 스토리지 디바이스 (28) 는 호스트 프로세서 (24) 또는 GPU (48) 에 의한 프로세싱을 위한 이미지 데이터를 저장할 수도 있다.

[0034] 메모리 (32) 는 동작 동안 컴퓨팅 디바이스 (20) 내에 정보를 저장하도록 구성될 수도 있다. 몇몇 실시예들에서, 메모리 (32) 는 임시 메모리인데, 이는 메모리 (32) 의 주 목적이 장기 저장이 아님을 의미한다. 메모리 (32) 는, 몇몇 실시예들에서, 컴퓨터 판독가능 저장 매체로서 설명될 수도 있다. 따라서, 메모리 (32) 는 또한 시간의 경과에 따라 변할 수 있는 데이터를 저장하고 있음에도 불구하고 "비일시적" 인 것으로 간주될 수도 있다. 메모리 (32) 는 또한, 몇몇 실시예들에서, 휘발성 메모리로 설명될 수도 있는데, 이는 컴퓨터가

턴 오프될 때 메모리 (32) 가 저장된 콘텐츠를 유지하지 않음을 의미한다. 휘발성 메모리들의 실시예들은 RAM, DRAM, SRAM, 및 당업계에 공지된 다른 형태들의 휘발성 메모리들을 포함한다.

[0035] 몇몇 실시예들에서, 메모리 (32) 는 호스트 프로세서 (24) 또는 GPU (48)에 의한 실행을 위한 프로그램 명령들을 저장하는 데 사용될 수도 있다. 메모리 (32) 는 프로그램 실행 동안 정보를 일시적으로 저장하기 위해 컴퓨팅 디바이스 (20) 상에서 구동하는 소프트웨어 또는 애플리케이션들에 의해 사용될 수도 있다. 이와 같이, 메모리 (32) 는 호스트 프로세서 (24) 및 GPU (48) 와 같은 컴퓨팅 디바이스 (20) 의 다른 콤포넌트들에 의해 액세스될 수도 있다.

[0036] 컴퓨팅 디바이스 (20) 는 하나 이상의 무선 네트워크들과 같은 하나 이상의 네트워크들을 통해 외부 디바이스들과 통신하도록 네트워크 모듈 (36) 을 이용할 수도 있다. 네트워크 모듈 (36) 은 이더넷 카드, 광 송수신기, 무선 주파수 송수신기, 또는 정보를 전송하고 수신할 수 있는 임의의 다른 타입의 디바이스와 같은 네트워크 인터페이스 카드일 수도 있다. 몇몇 실시예들에서, 컴퓨팅 디바이스 (20) 는 서버, 모바일 폰, 또는 다른 네트워킹된 컴퓨팅 디바이스와 같은 외부 디바이스와 무선으로 통신하도록 네트워크 모듈 (36) 을 이용할 수도 있다.

[0037] 컴퓨팅 디바이스 (20) 는 또한 사용자 인터페이스 (40) 를 포함한다. 사용자 인터페이스 (40) 의 실시예들은 트랙볼, 마우스, 키보드, 및 다른 타입들의 입력 디바이스들을 포함하지만, 이들로 제한되는 것은 아니다. 사용자 인터페이스 (40) 는 또한 디스플레이 (44) 의 일부로서 내장된 터치스크린을 포함할 수도 있다. 디스플레이 (44) 는 액정 디스플레이 (LCD), 유기 발광 다이오드 (OLED) 디스플레이, 플라즈마 디스플레이, 또는 다른 타입의 디스플레이 디바이스를 포함할 수도 있다.

[0038] 컴퓨팅 디바이스 (20) 의 GPU (48) 는 고정된 기능 및 GPU 애플리케이션들을 실행하기 위한 프로그래밍가능 콤포넌트들을 갖는 전용 하드웨어 유닛 일 수도 있다. GPU (48) 의 실시예들은 DSP, 범용 마이크로프로세서, ASIC, FPGA, 또는 다른 등가의 집적 또는 이산 로직 회로부를 포함할 수도 있다. GPU (48) 는 또한 도 4 에 관하여 보다 상세히 설명되는 바와 같이 전용 메모리와 같은 다른 콤포넌트들을 포함할 수도 있다. 또한, 도 1 에는 개별적인 콤포넌트들로서 도시되어 있지만, 몇몇 실시예들에서, GPU (48) 는 호스트 프로세서 (24) 의 부분으로서 형성될 수도 있다. GPU (48) 는 다양한 애플리케이션 프로그래밍 인터페이스들 (API들) 에 따라 프로세싱 기법들을 이용하도록 구성될 수도 있다. 예를 들어, 사용자는 다수의 플랫폼들, 운영체제들, 및 하드웨어에서 구동할 수 있는 표준 소프트웨어 인터페이스를 이용하여 GPU (48) 에 의해 실행되도록 애플리케이션을 프로그래밍할 수도 있다. 몇몇 실시예들에서, GPU (48) 는 (전술된 바와 같이) API들의 OpenCL, CUDA 또는 DirectX 의 집합을 이용하도록 구성될 수도 있다.

[0039] 몇몇 실시예들에 따르면, GPU (48) 는 범용 그래픽 프로세싱 유닛 (GPGPU) 로서 구현될 수 있다. 예를 들어, GPU (48) 는 호스트 프로세서 (24) 에 의해 일반적으로 실행되는 다양한 범용 컴퓨팅 기능들을 실행할 수도 있다. 실시예들은 비디오 디코딩 및 포스트 프로세싱 (예컨대, 디블로킹, 잡음 감소, 컬러 보정 등) 및 다른 애플리케이션 특정 이미지 프로세싱 기능들 (예컨대, 안면 검출/인식, 패턴 인식, 웨이블릿 변환들 등) 을 포함한 다양한 이미지 프로세싱 기능들을 포함한다. 몇몇 실시예들에서, GPU (48) 는 호스트 프로세서 (24) 와 협력하여 애플리케이션들을 실행하도록 할 수도 있다. 예를 들어, 호스트 프로세서 (24) 는 GPU (48) 에 의한 실행을 위해 명령들을 GPU (48) 에 제공함으로써 어떤 기능들을 GPU (48) 에 오프로드할 수도 있다.

[0040] GPGPU 로서 구현될 때, GPU (48) 는 본 명세서에서 커널들로 지칭되는 셰이더 프로그램들을 실행한다. 커널들은 전술된 예시적 API들과 같은 API 를 이용하여 사용자에 의해 정의될 수 있다. 커널들은 워크그룹들 내에 그룹화된 개별 워크 아이템들 (예컨대, GPU 에서의 워크의 기본 유닛) 을 포함할 수도 있다.

[0041] 본 개시물의 몇몇 양태들에 따르면, GPU (48) 는 본 명세서에서 명령 스트림들로 지칭되는 커널 및 워크그룹 실행 순서들을 수신하고 실행한다. GPU (48) 는 GPU (48) 의 SP (예컨대, 도 4 에 관하여 도시되고 설명됨) 와 연관된 로컬 메모리 리소스들을 관리하는 데 커널 및 워크그룹 실행 순서들을 이용할 수 있다. 예를 들어, GPU (48) 는 SP 로컬 메모리에 저장된 데이터를 상이한 커널들과 공유하는 데 커널 및 워크그룹 실행 순서들을 이용할 수도 있다.

[0042] 다음 특징들에서 제공되는 어떤 실시예들은 이미지 프로세싱 애플리케이션을 수행하기 위해 워크 아이템들 및 워크그룹들을 실행하는 CPU 를 지칭할 수도 있다. 예를 들어, 워크 아이템들 및 워크그룹들은 이미지의 픽셀들 (예컨대, 비디오 데이터의 프레임) 과 연관되는 것으로 하기에서 설명될 수도 있다. 그러나, GPU 는 다양한 입력 데이터 (예컨대, 병렬 프로세싱으로부터 이익을 얻는 임의의 기능들 및 데이터 세트들) 에 대한 이

미지 프로세싱 기능들외의 다양한 기능들을 실행하도록 구현될 수도 있다. 따라서, 워크그룹들 사이에 공유하는 명령 스트림들 및 메모리에 관하여 하기에 설명되는 실시예들 및 양태들은, 예를 들어 다양한 다른 입력 데이터 세트들에 대해 다양한 다른 기능들을 수행하는 GPU 에 의해 실행될 수 있다.

[0043] 도 2 는 이미지 (49) 를 프로세싱하는 관련된 명령들과 함께 이미지 데이터를 갖는 예시적 이미지 (49) 를 예시한 블록도이다. 명령들은 복수의 워크그룹들 (50A-50P)(총괄하여, 워크그룹들 (50)) 로 분할하는 것으로 나타내지며, 각각의 워크그룹은 복수의 워크 아이템들 (52) 을 포함한다. 명령들은 도 1 에 도시된 GPU (48) 와 같은 GPU 에 의해 실행될 수도 있다. 도 2 에 도시된 실시예에서, 이미지 (49) 를 프로세싱하는 명령들은 16 개의 워크그룹들 (50) 로 분할되며, 각각의 워크그룹 (50) 은 64 개의 개별 워크 아이템들 (52) 로 분할되지만, 다른 분할들이 가능하다.

[0044] 도 2 에 도시된 실시예에서, 이미지 (49) 는 1024 개의 픽셀들을 포함하는 정사각형의 약 16 메가바이트 (MB) 이미지이다. 워크 아이템들 (52) 각각은 GPU (48) 에 의해 실행될 수 있는 워크의 기본 유닛을 나타낸다. 몇몇 실시예들에서, 각각의 워크 아이템 (52) 은 이미지 (49) 의 특정 픽셀에 관련될 수도 있는 명령들을 포함한다. 따라서, GPU (48) 가 워크 아이템 (52) 을 실행할 때, 이미지 (49) 의 대응하는 픽셀이 프로세싱될 수도 있다 (예컨대, 명령들에 따라 분석되거나 변경될 수도 있다). 워크 아이템들 (52) 은 이미지 (49) 의 픽셀들의 특정 그룹에 관련된 명령들을 포함하는 워크그룹들 (50) 로 조직될 수도 있다. 워크그룹 (50) 을 프로세싱할 때, 워크그룹 (50) 과 연관된 픽셀들의 특정 그룹에 관련된 이미지 데이터는 (예를 들어, 도 4 에 관하여 하기에서 도시되고 설명된 바와 같이) SP 의 로컬 메모리 리소스들 내에 로딩될 수도 있다.

[0045] 도 2 에 관하여 설명된 픽셀 데이터, 워크 아이템들, 및 워크그룹들 사이의 관계는 단지 가능한 명령 구조들의 일 실시예에 불과하다. 다른 실시예들에서, 워크 아이템은 이미지 (49) 의 1 개 초과 또는 미만의 픽셀과 관련될 수도 있다.

[0046] 도 3 은 도 1 에 도시된 GPU (48) 와 같은 GPU 에 의해 실행될 수 있는 3 개의 커널들 (예컨대, 제 1 커널 (56), 제 2 커널 (57), 및 제 3 커널 (58)) 의 배열에서의 워크그룹들을 예시한 블록도이다. 또한, 각각의 커널은 주어진 애플리케이션에 관련된 특정 기능을 수행하도록 실행될 수도 있다. 몇몇 실시예들에서, 커널들 (56-58) 은 컬러 보정 알고리즘들, 안면 검출 알고리즘들, 패턴 인식 알고리즘들, 증강 현실 애플리케이션들, 다양한 알고리즘 애플리케이션들 (예컨대, 웨이블릿 변환들, 푸리에 변환들 등), 또는 다양한 기타 애플리케이션들에 대한 기능을 정의할 수도 있다. 단지 예시를 목적으로, 도 3 은 도 1 에 도시된 예시적 GPU (48) 및 도 2 에 도시된 예시적 이미지 (49) 에 관하여 설명된다.

[0047] GPU (48) 는 도 2 에 도시된 이미지 (49) 와 같은 이미지에 대해 특정 태스크를 실행하도록 커널들 (56-58) 을 실행할 수도 있다. 예를 들어, GPU (48) 는 안면 검출/인식, 패턴 인식, 및 병렬 프로세싱 (예컨대, 1 개를 초과하는 명령의 동시 프로세싱) 에 적합한 많은 다른 기능들과 같은 다양한 기능들을 실행하도록 GPGPU 로서 구현될 수도 있다. 단순화된 비제한적 실시예로서 제공되는 경우, 커널들 (56-58) 은 안면 검출 애플리케이션으로 구현될 수도 있다. 이 실시예에서, GPU (48) 는 이미지 (49) 에서 하나 이상의 안면을 검출하도록 커널들 (56-58) 을 구현할 수 있다. 커널들 (56-58) 의 각각은 특정 안면 검출 관련 기능을 수행하도록 구성될 수도 있다. 이러한 커널들 (56-58) 은 "분류기들" 로 지칭될 수도 있다. 즉, 커널들 (56-58) 은 특정한 사전 정의된 특징을 갖는 (또는 갖지 않는) 픽셀들을 분류한다. 커널들 (56-58) 은 다수의 트레이닝 이미지들을 이용하여 생성된 수학 공식들을 포함할 수도 있다. 예를 들어, 커널들 (56-58) 은 다수의 사전 정의된 이미지들과 함께 테스팅 환경에서 개발된 수학 공식들을 포함할 수도 있다.

[0048] 도 3 에 도시된 실시예에서, GPU (48) 는 각각의 픽셀이 커널들 (56-58) 에서 설명된 사전 정의된 속성들을 포함하는지를 결정하도록 커널들 (56-58) 을 연속으로 실행할 수도 있다. 즉, GPU (48) 에 의해 실행될 때, 각각의 커널 (56-58) 은 안면과 연관된 사전 정의된 속성을 식별하는 데 이용될 수 있는 부울 (Boolean) 값을 리턴할 수도 있다. 어떤 픽셀이 커널들 (56-58) 에서 설명된 모든 사전 정의된 속성들을 나타낸다면 (예컨대, 그 픽셀들과 연관된 부울 결과들이 몇몇 사전 정의된 기준들을 충족한다면), 그 픽셀은 후보 안면 픽셀로 간주된다. 어떤 픽셀이 커널들 (56-58) 에서 설명된 모든 사전 정의된 속성들을 나타내지 않는다면 (예컨대, 그 픽셀들과 연관된 부울 결과들이 몇몇 사전 정의된 기준들을 충족하지 않는다면), 그 픽셀은 안면 픽셀로 간주되는 것으로부터 배제된다.

[0049] 도 3 의 이미지 프로세싱 실시예에서, 이미지 (49) 와 연관된 데이터는 각각의 커널 (56-58) 에 대해 한 번씩 2 회 프로세싱된다. 예를 들어, 커널들 (56-58) 의 워크그룹들은 이미지 (49) 의 동일한 입력 이미지 영역에 대응할 수도 있다. 커널들 (56-58) 각각의 유사하게 넘버링된 워크그룹들은 이미지 (49) 의 동일한 입력 이

미지 영역에서 실행될 명령들의 세트를 포함할 수도 있다.

[0050] 본 개시물의 양태들은 커널들 (56-58) 의 유사하게 넘버링된 워크그룹들을 GPU (48) 에 의한 프로세싱을 위한 명령 스트림들로 결부시키는 명령들의 생성에 관련된다. 예를 들어, 사용자 (예컨대, 컴퓨터 또는 애플리케이션 프로그래머) 또는 프로그램은 동일한 SP 를 이용하여 커널 (56) 의 워크그룹 0, 그 뒤에 커널 (57) 의 워크그룹 0, 그리고 그 뒤에 커널 (58) 의 워크그룹 0 을 실행하도록 GPU (48) 에게 명령하는 명령 스트림을 생성할 수 있다. 이 방식으로, GPU (48) 는 (예를 들어, 도 4 에 관해 도시되고 설명된 바와 같이) 워크그룹 0 에 대응하는 이미지 (49) 의 입력 영역을 GPU (48) 의 셰이더 프로세서 (SP) 의 로컬 메모리 리소스들 내로의 로딩할 수 있고, 커널들 (56-58) 을 이용하여 그 입력 이미지 영역을 순차적으로 프로세싱할 수 있다.

[0051] 몇몇 실시예들에서, 사용자 (컴퓨터 또는 애플리케이션 프로그래머) 는 커널들 (56-58) 을 개발하면서 사전 구성된 API 커맨드를 이용하여 커널들 (56-58) 의 워크그룹 지정들을 포함하는 명령 스트림들을 정의할 수 있다. 예를 들어, 사용자는 커널들 (56-58) 의 워크그룹들을 GPU (48) 에 의해 실행될 명령 스트림들에 지정하도록 하는 사전 구성된 명령 스트림 API 커맨드들을 구현할 수 있다. 커널들 (56-58) 과 연관된 명령 스트림 지정들을 실행할 때, GPU (48) 는 GPU (48) 의 어떤 SP 에 커널들 (56-58) 의 워크그룹들을 라우팅한다.

[0052] 다른 실시예에서, 자동화된 시스템은 커널들 (56-58) 의 워크그룹 지정들을 포함하는 명령 스트림들을 생성하도록 구현될 수도 있다. 예를 들어, 컴파일러 프로그램 또는 다른 프로그램 (예컨대, 컴파일된 저레벨 머신 어셈블러 코드로부터 메모리 액세스 패턴들을 트레이싱하는 프로그램) 은 메모리 액세스 패턴들을 모니터링하거나 분석할 수도 있으며, 워크그룹 0 과 같은 워크그룹과 연관된 데이터가 커널들 (56-58) 에 의해 다수 회 액세스되는 것을 식별할 수도 있다. 그 후, 프로그램은 워크그룹들이 GPU (48) 의 SP 에 의해 순차적으로 프로세싱되도록 워크그룹들을 명령 스트림에 지정할 수도 있다. 커널들 (56-58) 과 연관된 명령 스트림 지정들을 실행할 때, GPU (48) 는 GPU (48) 의 GPU (48) 의 어떤 SP 에 커널들 (56-58) 의 워크그룹들을 라우팅한다.

[0053] 도 4 는 본 개시물의 양태들을 실행하도록 구성될 수도 있는 GPU (60) 를 예시한 블록도이다. 몇몇 실시예들에서, GPU (60) 는 도 1 에 도시된 GPU (48) 에 유사하게 또는 동일하게 구성될 수도 있다. 도 4 에 도시된 실시예에서, GPU (60) 는 메모리 (72), 셰이더 프로세서 메모리들 (78A-78C)(총괄하여, SP 메모리들 (78)) 을 각각 갖는 셰이더 프로세서 들 (76A-76C)(총괄하여, SP들 (76)), 및 시퀀서 모듈 (82) 를 포함한다.

[0054] 다른 실시예들에서, GPU (60) 는 명료성을 목적으로 도 4 에 도시되지 않은 다른 콤포넌트들을 포함할 수도 있다. 예를 들어, GPU (60) 는 또한 래스터화기, 텍스처 유닛들, 하나 이상의 버퍼들, 또는 다른 GPU 콤포넌트들과 같은, 이미지들을 분석하는 것 그리고 렌더링하는 것에 관련된 다양한 다른 모듈들을 포함할 수도 있다. 추가로, GPU (60) 는 도 4 에 도시된 것들보다 더 많거나 또는 더 적은 콤포넌트들을 포함할 수도 있다. 예를 들어, GPU (60) 는 3 개의 SP들 (76) 을 포함하는 것으로서 도 4 에 도시되어 있다. 그러나, 다른 실시예들에서, GPU (60) 는 도 4 에 도시된 것들보다 더 많거나 또는 더 적은 SP들을 포함할 수도 있다.

[0055] 몇몇 실시예들에서, GPU 메모리 (72) 는 도 1 에 도시된 메모리 (32) 와 유사할 수도 있다. 예를 들어, GPU 메모리 (72) 는 일시적 컴퓨터 판독가능 저장 매체일 수도 있다. GPU 메모리 (72) 의 실시예들은 RAM, DRAM, SRAM, 및 당업계에 공지된 다른 형태들의 휘발성 메모리들을 포함한다. GPU (60) 가 도 1 에 도시된 호스트 프로세서 (24) 와 같은 다른 프로세서의 부분으로서 형성되는 실시예들에서, GPU 메모리 (72) 는 GPU (60) 외의 콤포넌트들에 의해 액세스될 수도 있다.

[0056] GPU 메모리 (72) 는 GPU (60) 에 대한 글로벌 메모리로서 구성될 수도 있다. 예를 들어, GPU (72) 는 동작 동안 GPU (60) 내의 명령들 및 정보 (GPU (60) 에 의한 프로세싱을 위한 이미지 데이터 및 명령들) 을 저장하도록 구성될 수도 있다. GPU 메모리 (72) 는 또한 GPU (60) 에 의해 프로세싱된 데이터의 결과들을 저장하도록 구성될 수도 있다. 몇몇 실시예들에서, GPU 메모리 (72) 는 GPU (60) 외부의 컴퓨팅 디바이스 콤포넌트들과 인터페이싱한다. 예를 들어, GPU (60) 를 포함하는 컴퓨팅 디바이스의 콤포넌트는 초기에 데이터 (예컨대, 비디오 데이터의 하나 이상의 프레임들) 를 GPU (60) 에 의한 프로세싱을 위한 GPU 메모리 (78) 에 전달할 수도 있다. 그 후, GPU (60) 는 데이터를 프로세싱하고 그 결과들을 GPU 메모리 (72) 에 저장한다. 그 결과들은 후속으로 GPU 메모리 (72) 로부터 컴퓨팅 디바이스의 다른 콤포넌트로 판독될 수도 있다.

[0057] SP들 (76) 은 프로세싱 콤포넌트들의 프로그래밍가능 파이프라인으로서 구성된다. 몇몇 실시예들에서, SP들 (76) 은 SP들 (76) 이 기하학적 구조, 버텍스, 또는 픽셀 셰이딩 동작들을 수행하여 그래픽을 렌더링할 수 있다는 점에서 "통합된 셰이더들" 로 지칭될 수도 있다. SP들 (76) 은 또한 범용 계산을 수행하기 위한 GPGPU 애플리케이션들에서 이용될 수도 있다. 예를 들어, SP들 (76) 은 도 2 에 도시된 이미지 (49) 와 같은 이미

지를 분석하거나 또는 다른 방식으로 프로세싱하도록 구현될 수도 있다. SP들 (76) 은, 명령들을 페치하고 디코딩하는 콤포넌트들 및 산술 계산들을 실행하는 하나 이상의 산술 로직 유닛들 ("ALU들") 과 같이, 도 4 에 구체적으로 도시되지 않은 하나 이상의 콤포넌트들을 포함할 수도 있다. SP들 (76) 은 또한 SP 메모리들 (78) 과 같은 하나 이상의 메모리들, 캐시들, 또는 레지스터들을 포함한다.

[0058] SP 메모리들 (78) 은 SP들 (76) 에 의해 프로세싱되는 데이터를 저장하기 위한 레지스터들 또는 데이터 캐시들로서 구성될 수도 있다. 몇몇 실시예들에서, SP 메모리들 (78) 은 SP들 (76) 의 로컬 메모리들이다. 예를 들어, SP 메모리들 (78) 은 글로벌 GPU 메모리 (72) 보다 상대적으로 더 작을 수도 있으며, 실행 이전에 하나 이상의 워크그룹들과 연관된 데이터를 저장할 수도 있다. SP 메모리들 (78) 은 GPU 메모리 (72) 보다 상대적으로 느린 레이턴시를 가질 수도 있다. 예를 들어, SP 메모리들 (78) 은 SP들 (76) 에 의해 상대적으로 빠르게 액세스될 수 있다. 그러나, 글로벌 메모리 (72) 로부터 SP 메모리들 (78) 로의 데이터 이송과 관련된 레이턴시는 일반적으로 매우 크다. 예를 들어, 글로벌 메모리 (72) 로부터 SP 메모리들 (78) 로의 데이터 이송은 다수의 클록 사이클들을 소비할 수도 있고, 이에 의해 보틀넥을 생성하고 GPU (60) 의 전체 성능을 느리게 할 수도 있다.

[0059] SP 메모리들 (78) 은 GPU (60) 가 동작 중일 때 GPU 메모리 (72) 와 데이터를 교환할 수도 있다. 예를 들어, GPU (60) 는 GPU 메모리 (72) 로부터 SP 메모리들 (78) 로 워크그룹들과 연관된 데이터를 전송한다. 일단 SP 메모리들 (78) 에 저장되면, SP들 (76) 은 개별 SP 메모리들 (78) 에 저장된 데이터에 액세스하고 그를 프로세싱하도록 병렬로 동작한다. 데이터를 실행할 때, SP들 (76) 은 그 결과들을 GPU 메모리 (72) 로 리턴한다. 일반적으로, SP 메모리들 (78) 과 SP들 (76) 사이의 메모리 대역폭은 GPU 메모리 (72) 와 SP들 (76) 사이의 메모리 대역폭보다 크다. 따라서, SP (76) 는 SP (76) 가 GPU 메모리 (72) 로부터 데이터를 판독할 수 있는 것보다 더 빠르게 데이터를 관련 SP 메모리 (78) 로부터 판독할 수 있는 것이 일반적이다. 즉, GPU 메모리 (72) 는 일반적으로 SP 메모리들 (78) 과 연관된 것보다 더 높은 레이턴시를 나타낸다. 따라서, 데이터가 SP들 (76) 에 의해 실행되기 전에 SP 메모리들 (78) 에 이송되는 것이 이익이 될 수도 있다.

[0060] 시퀀서 모듈 (82) 은 GPU (60) 내에서의 명령 및 데이터 흐름을 제어한다. 시퀀서 모듈 (82) 은 SP들 (76) 에 의한 실행을 위해 워크 아이템들, 워크그룹들 및 관련 데이터를 SP 메모리들 (78) 에 분포시키는 고정된 기능 및 프로그래밍가능 콤포넌트들의 조합을 포함할 수도 있다. 따라서, 시퀀서 모듈 (82) 은 GPU 메모리 (72) 와 SP들 (76) 사이에서의 데이터 이송들을 관리한다. 단지 예시를 목적으로, 시퀀서 모듈 (82) 의 워크그룹은 도 3 과 관련하여 도시되고 설명된 애플리케이션에 관하여 설명된다.

[0061] 시퀀서 모듈 (82) 은 SP들 (76) 의 특정 SP 에 의해 어떤 워크그룹들이 실행되는지와는 무관하게, 고정된 분포 패턴으로 워크그룹들을 분포시킬 수도 있다. 예를 들어, 다수의 커널들 (56-58) 을 갖는 예시적 애플리케이션 (54)(도 3 에 도시됨) 을 프로세싱하기 위해, 시퀀서 모듈 (82) 은 GPU (60) 의 모든 SP들 (76) 에 워크그룹들을 균일하게 분포시킬 수도 있다. 추가로, 도 5 와 관련하여 하기에 보다 상세히 설명되는 바와 같이, 시퀀서 모듈 (82) 은 다음 커널로 이동하기 전에 커널의 모든 워크그룹들을 SP들 (76) 에 분포시킬 수도 있다. 예를 들어, 시퀀서 모듈 (82) 은, 커널 (56) 이 SP들 (76) 에 의해 프로세싱될 때까지, 커널 (56) 의 워크그룹 0 을 SP (76A) 에, 커널 (56) 의 워크그룹 1 을 SP (76B) 에, 커널 (56) 의 워크그룹 2 등을 SP (76C) 등에 분포시킬 수도 있다.

[0062] 다른 실시예들에서, 본 개시물의 몇몇 양태들에 따르면, 시퀀서 모듈 (82) 은 커널 및 워크그룹 실행 순서들을 수신하고 실행할 수도 있다. 예를 들어, 시퀀서 모듈 (82) 은 시퀀서 모듈 (82) 에게 커널들의 워크그룹들을 SP들 (76) 의 특정 SP 에 분포시키도록 지시하는 명령 스트림들을 정의하는 명령들을 수신할 수도 있다. 명령 스트림들은 상이한 커널들의 워크그룹들이 SP들 (76) 의 동일한 SP 에 의해 프로세싱되도록 그들을 함께 결부시킨다. 명령 스트림들은 SP 메모리들 (78) 의 리소스들을 관리하는 방식을 제공한다. 예를 들어, 명령 스트림들을 정의하는 명령들을 실행함으로써, 시퀀서 모듈 (82) 은 하나의 워크그룹과 연관된 입력 데이터가 다수의 다른 커널들의 워크그룹들에 의해 공유되고 직렬로 실행되게 한다.

[0063] 시퀀서 모듈 (82) 은 GPU (60) 가 동일하거나 또는 실질적으로 동일한 입력 데이터를 프로세싱하는 다수의 커널들을 갖는 애플리케이션을 실행하고 있을 때 명령 스트림들을 정의하는 명령들을 실행하도록 구현될 수 있다. 예를 들어, 도 3 과 관련하여 설명된 바와 같이, 애플리케이션 (54) 은 3 개의 커널들 (56-58) 을 포함하며, 각각의 커널은 복수의 관련된 워크그룹들을 갖는다. 커널 (56) 의 워크그룹 0 은 커널 (57) 의 워크그룹 0 및 커널 (58) 의 워크그룹과 동일한 입력 데이터에 대응한다. 따라서, 시퀀서 모듈 (82) 은 커널들 (56-58) 의 워크그룹 0 을 SP (76A) 에 순차적으로 분포시킬 수도 있다. 추가로, 시퀀서 모듈 (82) 은 모든 커널들

의 모든 워크그룹들이 SP들 (76) 에 의해 실행될 때까지 커널들 (56-58) 의 워크그룹 1 등을 SP (76B) 등에 분포시킬 수도 있다.

[0064] 이 방식으로, 시퀀서 모듈 (82) 은 SP 메모리들 (78) 의 로컬 메모리 리소스들을 관리할 수 있다. 예를 들어, 커널 (56) 의 워크그룹 0 을 실행하기 전, GPU (60) 는 GPU 메모리 (72) 로부터 SP 메모리 (78A) 로 커널 (56) 의 워크그룹 0 과 연관된 입력 데이터를 이송한다. 커널 (56) 의 워크그룹 0 을 실행한 후, 그리고 SP 메모리 (78A) 에 대한 새로운 데이터를 페치하는 대신, 시퀀서 모듈 (82) 은 SP (76A) 에게 커널 (57) 의 워크그룹 0 을 실행하고 그 뒤에 커널 (58) 의 워크그룹 0 을 실행하도록 지시한다. 워크그룹 0 의 입력 데이터는 커널들 (56-58) 사이에서 동일하여, 워크그룹 0 과 연관된 데이터는 SP 메모리 (78A) 에 남아 있을 수 있고, 모든 3 개의 커널들 (56-58) 의 워크그룹 0 에 의해 공유될 수 있다.

[0065] 명령 스트림들을 실행하는 것 그리고 상이한 커널들의 워크그룹들 사이에 데이터를 공유하는 것은 고정된 분포 패턴으로 워크그룹들을 분포시키는 시스템에 비해 로컬 메모리 대역폭 절감을 제공할 수도 있다. 예를 들어, 상이한 커널들의 워크그룹들 사이에 데이터를 공유하는 것은 GPU 메모리 (72) 와 SP 메모리들 (78) 사이에 보다 적은 데이터가 이송되게 한다. 도 3 에 도시된 3 개의 커널 실시예에서, GPU 메모리 (72) 와 SP 메모리들 (78) 사이의 메모리 대역폭 소비는 2/3 만큼 감소한다. 워크그룹 0 과 같은 워크그룹과 연관된 데이터를 로컬 SP 메모리에 3 회 이송 (예컨대, 각각의 커널마다 1 회 이송) 한다기보다, GPU (60) 는 워크그룹과 연관된 데이터를 로컬 SP 메모리에 한 번 이송할 수 있고, 모든 3 개의 커널들 (56-58) 사이에 데이터를 공유할 수 있다.

[0066] 본 개시물의 몇몇 실시예들에 따르면, 명령 스트림들을 실행하는 것과 연관된 로컬 메모리 대역폭 절감들은 또한 시간 절감을 제공할 수도 있다. 예를 들어, SP들이 프로그램 (54) 과 같은 주어진 프로그램을 실행하도록 워크그룹들과 연관된 동일한 수의 계산을 수행할 수도 있지만, 보다 적은 데이터가 GPU 메모리 (72) 와 SP 메모리들 (78) 사이에서 이송될 수도 있기 때문에 시간 절감이 달성될 수도 있다. 전술된 바와 같이, GPU 메모리 (72) 와 SP 메모리들 (78) 사이의 데이터 이송은 프로그램 (54) 을 실행하는 프로세스에 보틀넥을 도입하는 상대적으로 시간 집약적인 프로세스일 수도 있다. 따라서, GPU 메모리 (72) 와 SP 메모리들 (78) 사이에서 이송될 필요가 있는 데이터의 양을 감소시키는 것은 또한 GPU 메모리 (72) 와 SP 메모리들 (78) 사이에서의 데이터 이송과 연관된 보틀넥을 감소시킬 수도 있다.

[0067] 명령 스트림들을 정의하는 시퀀서 모듈 (82) 에 의해 수신된 명령들은 사용자에 의해 생성될 수도 있고, 또는 (예컨대, 컴파일러 프로그램에 의해) 자동으로 생성될 수도 있다. 예를 들어, 사용자 (예컨대, 소프트웨어 개발자) 는 하나 이상의 명령 스트림 커맨드들을 포함하는 API 를 이용하여 명령 스트림들을 정의하고 구현할 수도 있다. 명령 스트림 커맨드들을 갖는 애플리케이션들을 수신한 후, 시퀀서 모듈 (82) 은 명령 스트림 커맨드들을 실행하여 SP들 (76) 과 관련된 로컬 메모리 리소스들을 관리한다.

[0068] 명령 스트림들을 정의하는 명령들은 도 1 에 도시된 호스트 프로세서 (24) 와 같은 컴퓨팅 디바이스의 호스트 프로세서에 의해 시퀀서 모듈 (82) 에 송신될 수도 있다. GPU (60) 가 개별 디바이스인 (예컨대, 호스트 프로세서를 갖는 컴퓨팅 디바이스에 포함되지 않는) 실시예들에서, 다른 프로세싱 콤포넌트는 명령 스트림들을 포함하는 명령들을 시퀀서 모듈 (82) 에 송신하는 일을 담당할 수도 있다.

[0069] 도 5 는 제 1 SP 메모리 (156A) 를 갖는 제 1 SP (152A), 제 2 SP 메모리 (156B) 를 갖는 제 2 SP (152B), 및 제 3 SP 메모리 (156C) 를 갖는 제 3 SP (152C)(총괄하여, SP들 (152) 및 SP 메모리들 (156)) 에 제 1 커널 (142), 제 2 커널 (144), 및 제 3 커널 (146) 의 워크그룹들을 분포시키는 시퀀서 모듈 (140) 의 실시예를 예시한 블록도이다. 본 개시물의 몇몇 양태들에 따르면, 시퀀서 모듈 (140) 및 SP들 (152) 은 도 1 에 도시된 GPU (48) 또는 도 4 에 도시된 GPU (60) 와 같은 GPU 에 포함될 수도 있다.

[0070] 시퀀서 모듈 (140) 및 SP들 (152) 은 도 4 와 관련하여 도시되고 설명된 시퀀서 모듈 (82) 및 SP들 (76) 과 유사하게 또는 동일하게 구성될 수도 있다. 예를 들어, 시퀀서 모듈 (140) 은 GPU 내에서의 명령 및 데이터 흐름을 제어하는 일을 담당할 수도 있다. 시퀀서 모듈 (140) 은 워크 아이템들 및 워크그룹들을 SP들 (152) 및 관련 SP 메모리들 (156) 에 분포시키는 고정된 기능 및 프로그래밍가능 콤포넌트들의 조합을 포함할 수도 있다.

[0071] 시퀀서 모듈 (140) 은 상이한 커널들의 워크그룹들의 특정 지정을 제어하지 않으면서 커널들 (142-146) 의 워크그룹들을 고정된 분포 패턴으로 분포시킨다. 예를 들어, 시퀀서 모듈 (140) 은, 제 1 커널 (142) 이 분포되고 실행될 때까지, (라인 (160) 으로 나타내진 바와 같이) 제 1 워크그룹 WG0 을 SP (152A) 에, (라인 (161)

으로 나타내진 바와 같이) 제 2 워크그룹 WG1 을 SP (152B) 에, (라인 (162) 등으로 나타내진 바와 같이) 제 3 워크그룹 WG2 등을 SP (152C) 등에 순차적으로 분포시킴으로써 제 1 커널 (142) 의 워크그룹들을 분포시킨다.

그 후, 시퀀서 모듈 (140) 은 제 2 커널 (144) 및 제 3 커널 (146) 로 이동하고, 이들 워크그룹들을 SP들 (152) 에 분포시킨다. 예를 들어, 시퀀서 모듈 (140) 은 고정된 분포 패턴에서 이어질 수도 있으며, SP들 (152) 사이에 제 2 커널 (144) 의 모든 워크그룹들을 분포시킬 수도 있다. 그 후, 시퀀서 모듈 (140) 은 제 3 커널 (146) 로 진행할 수도 있고, SP들 (152) 사이에 제 3 커널 (146) 의 모든 워크그룹들을 분포시킬 수도 있다.

[0072] 어떤 SP (152) 가 특정 워크그룹을 실행할지를 제어할 수 있는 능력 없이, 워크그룹들과 연관된 데이터는 1 개를 초과하는 SP 메모리들 (156) 내에 로딩될 필요가 있을 수도 있다. 도 5 의 실시예에 도시된 바와 같이, 고정된 분포 패턴을 추종하여, 전체 제 1 커널을 프로세싱한 후, m 시퀀서 모듈 (140) 은 제 2 커널 (144) 의 워크그룹 WG0 을 SP (152B)(라인 (161)) 에 분포시킨다. 따라서, WG0 과 연관된 입력 데이터는 SP 메모리 (156B) 내에 로딩되어야 한다. 추가로, 전체 제 2 커널 (144) 을 프로세싱한 후, 분포 패턴의 고정된 특성으로 인해, 시퀀서 모듈 (140) 은 제 3 커널 (146) 의 워크그룹 WG0 을 SP (152C)(라인 (162)) 에 분포시킨다. 따라서, WG0 과 연관된 입력 데이터는 SP 메모리 (156C) 내에 로딩된다.

[0073] 도 4 와 관련하여 전술된 바와 같이, 특정 워크그룹과 연관된 데이터는 SP 가 그 워크그룹을 실행할 수 있게 되기 전에 SP 의 로멀 케모리 내에 로딩되어야 하는 것이 일반적이다. 어떤 SP (152) 가 특정 워크그룹을 실행할지를 제어할 수 있는 능력 없이, 워크그룹들과 연관된 데이터는 커널들 사이에 공유될 수 없다. 도 5 에 도시된 실시예에서, 워크그룹 WG0 과 연관된 데이터는 SP들 (152A-152C) 에 의한 프로세싱 이전의 여러 차례에 각각의 SP 메모리들 (156A, 156B, 156C) 내에 로딩되어야 한다. 따라서, SP들 (152) 에 대한 메모리 대역폭은 각각의 채널에 대한 입력 데이터의 3 배와 같다.

[0074] 도 6 은 제 1 SP 메모리 (204A) 를 갖는 제 1 SP (200A), 제 2 SP 메모리 (204B) 를 갖는 제 2 SP (200B), 및 제 3 SP 메모리 (204C) 를 갖는 제 3 SP (200C)(총괄하여, SP들 (200) 및 SP 메모리들 (204)) 에 제 1 커널 (184), 제 2 커널 (186), 및 제 3 커널 (188) 의 워크그룹들을 분포시키는 시퀀서 모듈 (180) 의 실시예를 예시한 블록도이다. 본 개시물의 몇몇 양태들에 따르면, 시퀀서 모듈 (180) 및 SP들 (200) 은 도 1 에 도시된 GPU (48) 또는 도 4 에 도시된 GPU (60) 와 같은 GPU 에 포함될 수도 있다.

[0075] 시퀀서 모듈 (180) 및 SP들 (200) 은 도 4 와 관련하여 도시되고 설명된 시퀀서 모듈 (82) 및 SP들 (76) 과 유사하게 또는 동일하게 구성될 수도 있다. 예를 들어, 시퀀서 모듈 (180) 은 GPU 내에서의 명령 및 데이터 흐름을 제어하는 일을 담당할 수도 있다. 시퀀서 모듈 (180) 은 SP들 (200) 에 의한 실행을 위해 워크 아이템들 및 워크그룹들을 SP 메모리들 (204) 에 분포시키는 고정된 기능 및 프로그래밍가능 콤포넌트들의 조합을 포함할 수도 있다.

[0076] 본 개시물의 몇몇 양태들에 따르면, 시퀀서 모듈 (82) 은 명령 스트림들에 워크그룹들을 지정하는 사전 정의된 명령들에 따라 커널들 (184-188) 의 워크그룹들을 분포시킨다. 예를 들어, 시퀀서 모듈 (82) 은 상이한 커널들의 워크그룹들이 동일한 SP 에 의해 실행되도록 그들의 워크그룹들을 결부시키는 명령 스트림들을 수신하도록 구성될 수도 있다. 따라서, 커널들 (184-188) 의 워크그룹들을 고정된 패턴 (예를 들어, 도 5 에 도시됨) 으로 분포시키기보다는, 시퀀서 모듈 (180) 은 커널들의 워크그룹들을 함께 결부시키는 명령 스트림들에 기초하여 워크그룹들을 분포시키도록 구성될 수도 있다.

[0077] 도 6 에 도시된 실시예에서, 시퀀서 모듈 (180) 은 커널 (184) 의 워크그룹 WG0 을 커널 (186) 의 워크그룹 WG0 에 그리고 커널 (188) 의 워크그룹 WG0 에 결부시키는 명령 스트림을 실행한다. 커널들 (184-186) 의 워크그룹들 WG0 은 동일한 입력 데이터와 모두 연관된다. 명령 스트림을 실행함으로써, 커널들 (184-188) 의 워크그룹 WG0 은 SP (200A) 를 이용하여 순차적으로 프로세싱된다. 따라서, 커널들 (184-188) 중에서 동일한 것일 수도 있는 WG0 과 연관된 입력 데이터는 커널들 (184-188) 의 워크그룹들 WG0 사이에 공유될 수 있다. 예를 들어, 워크그룹 WG0 과 연관된 데이터는, 커널 (184) 의 WG0 를 프로세싱할 때 그리고 커널 (186) 의 WG0 및 커널 (188) 의 WG0 에 의해 공유될 때, SP 메모리 (204A) 내로 로딩될 수 있다.

[0078] SP 메모리 (204A) 에 데이터를 보유하는 것 그리고 다수의 워크그룹들 사이에 그 데이터를 공유하는 것은 SP 메모리 (204A) 의 효율적인 관리를 제공한다. 예를 들어, 워크그룹의 모든 실행 이후 새로운 데이터를 페치하고 SP 메모리 (204A) 내에 이송해야 한다기보다, 그 데이터는 SP 메모리 (204A) 에 남아 있을 수 있고, 다수의 커널들의 다수의 워크그룹들에 의해 공유될 수 있다. 따라서, 로컬 메모리 대역폭 소비가 감소할 수도 있다. 도 6 에 도시된 3-커널 실시예에서, 로컬 메모리 대역폭 소비는 도 5 에 도시된 3-커널 실시예에 비

해 2/3 만큼 감소한다.

[0079] 본 개시물의 몇몇 실시예들에 따르면, 명령 스트림들을 실행하는 것과 연관된 로컬 메모리 대역폭 절감들은 또한 시간 절감을 제공한다. 예를 들어, SP들 (200) 이 주어진 프로그램과 같은 주어진 프로그램을 실행하도록 명령 스트림들을 이용하지 않는 워크그룹들 시스템으로서 워크그룹들과 연관된 동일한 수의 계산을 수행할 수도 있지만, 보다 적은 데이터가 GPU 글로벌 메모리와 SP 메모리들 (204) 사이에서 이송되어야 하기 때문에 시간 절감이 달성될 수도 있다. GPU 글로벌 메모리와 SP 메모리들 (204) 사이의 데이터 이송은 커널들 (184-188) 을 실행하는 프로세스에 보틀넥을 도입하는 상대적으로 시간 집약적인 프로세스일 수도 있다. 따라서, GPU 글로벌 메모리와 SP 메모리들 (204) 사이에서 이송될 필요가 있는 데이터의 양을 감소시키는 것은 또한 GPU 글로벌 메모리와 SP 메모리들 (204) 사이에서의 데이터 이송과 연관된 보틀넥을 감소시킨다.

[0080] 도 7 은 제 1 SP 메모리 (256A) 를 갖는 제 1 SP (252A), 제 2 SP 메모리 (256B) 를 갖는 제 2 SP (252B), 및 제 3 SP 메모리 (256C) 를 갖는 제 3 SP (252C)(총괄하여, SP들 (252) 및 SP 메모리들 (256)) 에 실행 순서들 (240, 244, 248) 의 스트림들을 할당하는 실시예를 예시한 블록도이다. 본 개시물의 몇몇 양태들에 따르면, SP들 (252) 은 도 1 에 도시된 GPU (48) 또는 도 4 에 도시된 GPU (60) 와 같은 GPU 에 포함될 수도 있다.

[0081] 도 7 에 도시된 실시예는 3 개의 커널들과 연관된 워크그룹들을 실행하는 3 개의 SP들 (252) 을 포함한다. 그러나, 실행 순서들의 스트림들은 도 7 에 도시된 것들보다 더 많거나 또는 더 적은 SP들 (예컨대, SP들 중 2 개의 SP들, 10 개의 SP들, 100s) 을 갖는 시스템들에서 구현될 수도 있음을 이해해야 한다. 추가로, 실행 순서들의 스트림들은 도 7 에 도시된 3 개보다 더 많거나 또는 더 적은 워크그룹들 및 커널들을 함께 링크시킬 수도 있다.

[0082] 실행 순서들의 스트림들 또는 명령 스트림들 (240-248) 은 도 4 에 도시된 시퀀서 모듈 (82) 과 같은 시퀀서 모듈에 의해 SP들 (252) 에 할당될 수도 있다. 명령 스트림들 (240-248) 은 상이한 커널들의 워크그룹들이 동일한 SP 에 의해 프로세싱되도록 그들을 함께 사실상 결부시킨다. 예를 들어, 도 7 에 도시된 바와 같이, 명령 스트림 (240) 은 커널 1 의 워크그룹 0 을 커널 2 의 워크그룹 0 및 커널 3 의 워크그룹 0 에 링크시킨다.
마찬가지로, 명령 스트림 (244) 은 커널 1 의 워크그룹 1 을 커널 2 의 워크그룹 1 및 커널 3 의 워크그룹 1 에 링크시키며, 명령 스트림 (248) 은 커널 1 의 워크그룹 2 를 커널 2 의 워크그룹 2 및 커널 3 의 워크그룹 2 에 링크시킨다.

[0083] 도 7 에 도시된 실시예에서, 유사하게 넘버링된 워크그룹들과 연관된 입력 데이터 중 적어도 일부는 일치한다.
예를 들어, 커널 1 의 워크그룹 0 과 연관된 입력 데이터는 커널 2 의 워크그룹 0 및 커널 3 의 워크그룹 0 과 연관된 입력 데이터와 동일하거나, 또는 그들과 적어도 일부 중복을 갖는다. 따라서, SP (252A) 는, 커널 1 의 워크그룹 0, 커널 2 의 워크그룹 0, 및 커널 3 의 워크그룹 0 을 실행할 때, 워크그룹 0 과 연관된 입력 데이터를 SP 메모리 (256A) 내에 로딩하고 그 입력 데이터를 공유함으로써, 명령 스트림 (240) 을 실행할 수 있다. 이 방식으로, 명령 스트림 (240) 을 실행하는 것은, 커널들 1, 2 및 3 의 실행 동안 SP 메모리 (256A) 내에 그리고 그 외부로 이송할 필요가 있는 데이터의 양을 감소시킨다. 유사한 동작들이 SP (252B) 및 SP 메모리 (256B) 에 대해 그리고 SP (252C) 및 SP 메모리 (256C) 에 대해 실행될 수도 있다.

[0084] 도 8 은, 도 7 에 도시된 스트림들 (240-248) 과 같이, 실행 순서들 (예컨대, "명령 스트림들") 을 생성하고 실행하는 방법 (300) 을 예시한 흐름도이다. 본 개시물의 몇몇 양태들에 따르면, 방법 (300) 은 도 1 에 도시된 GPU (48) 또는 도 4 에 도시된 GPU (60) 와 같은 GPU 에 의해 실행될 수도 있다. 오로지 예시를 목적으로, 방법 (300) 의 부분들이 도 4 에 도시된 예시적 GPU (60) 에 의해 실행되는 것으로 설명될 수도 있다.

[0085] 도 8 에 도시된 바와 같이, 실행 순서 스트림들에 대한 후보들이 초기 식별된다 (304). 몇몇 양태들에 따르면, 사용자는 명령 스트림들을 이용하는 것으로부터 이익을 얻을 후보 커널들을 식별한다. 예를 들어, 사용자는 동일한 입력 데이터를 다수 회 이용하는 커널들을 식별할 수도 있다. 다른 실시예에서, 컴파일러 프로그램과 같은 프로그램은 명령 스트림들을 구현하는 것으로부터 이익을 얻을 후보 커널들을 식별할 수도 있다.
예를 들어, 컴파일러 프로그램은 메모리 액세스 패턴들을 모니터링할 수도 있고, 1 개를 초과하는 커널에 의해 이용되는 데이터를 식별할 수도 있다. 프로그램의 1 개 초과의 커널에 의해 입력 데이터가 이용될 때, 명령 스트림은 동일한 데이터를 이용하는 워크그룹들이 동일한 SP 에 의해 실행되도록 그들 워크그룹들을 결부시키도록 구현될 수도 있다. 이 방식으로 명령 스트림들을 이용하는 것은 입력 데이터가 로컬 메모리 리소스들 내에 로딩될 필요가 있는 횟수를 감소시킴으로써 SP들의 로컬 메모리 리소스들을 관리하는 것을 도울 수도 있다. 예를 들어, 입력 데이터는 SP 의 메모리 내에 한 번 로딩될 수 있고, 다수의 커널들의 다수의 워크그룹들 사이에 공유될 수 있다.

[0086] 후보들이 식별된 후, 실행 순서 스트림 지정들이 생성된다 (308). 사용자는 명령 스트림 커맨드들을 포함하도록 적응된 API 를 이용하여 명령 스트림들을 정의할 수도 있다. 예를 들어, OpenGL, CUDA, DirectX, 또는 GPU 프로그램들을 생성하는 임의의 다른 API 와 같은 API 들은 사용자가 워크그룹들 및 그들의 커널들을 명령 스크림들에 지정하게 하는 하나 이상의 커맨드들을 포함하도록 적응될 수 있다. 다른 실시예에서, 컴파일러 프로그램과 같은 프로그램은 반복되는 메모리 액세스 패턴들을 식별한 후에 명령 스트림들을 자동으로 생성할 수도 있다.

[0087] 명령 스트림들이 생성된 후, 실행 순서 지정들이 GPU (60) 와 같은 GPU 로 송신되고 그에 의해 수신된다 (312). 몇몇 실시예들에서, 시퀀서 모듈 (82) 은 하나 이상의 커널 지정들 및 하나 이상의 워크그룹 지정들을 포함하는 실행 순서 스트림들을 정의하는 입력을 수신할 수도 있다. 명령 스트림들은 도 1 에 도시된 호스트 프로세서 (24) 와 같은 컴퓨팅 디바이스의 호스트 프로세서에 의해 시퀀서 모듈 (82) 에 송신될 수도 있다. GPU (60) 가 개별 디바이스인 (예컨대, 호스트 프로세서를 갖는 컴퓨팅 디바이스에 포함되지 않는) 실시예들에서, 다른 프로세싱 콤포넌트는 명령 스트림들을 수신하고 그들을 시퀀서 모듈 (82) 에 송신하는 일을 담당할 수도 있다.

[0088] 시퀀서 모듈 (82) 은 SP들 (76) 과 같은 SP 들에 스트림들을 할당함으로써 실행 순서들을 구현할 수도 있다 (316). 예를 들어, 시퀀서 모듈 (82) 은 GPU (60) 의 동일한 SP 에 의해 실행될 명령 스트림에서 지정된 워크그룹들을 할당할 수도 있다. SP들은 실행 순서들로 지정된 명령들을 실행함으로써 명령 스트림들을 실행한다 (320). 예를 들어, SP 는 명령 스트림에서 지정된 워크그룹들을 순차적으로 실행한다. 그렇게 함으로써, 명령 스트림에서 지정된 워크그룹들과 연관된 입력 데이터는 명령 스트림에서 지정된 워크그룹들 사이에 공유될 수 있다. 명령 스트림들을 실행하는 것은 GPU 메모리 (72) 와 SP 메모리들 (78) 사이에서 이송될 필요가 있는 데이터의 양을 감소시킬 수도 있고, 특정 프로그램을 실행하는 데 요구되는 전체 시간을 감소시킬 수도 있다.

[0089] 위에서 제공된 실시예들에서, 명령 스트림들은 상이한 커널들의 워크그룹들이 동이한 SP 에 의해 연속으로 실행되도록 상이한 커널들의 워크그룹들을 함께 결부시키는 것으로 설명된다. 이 방식으로 상이한 커널들의 워크그룹들을 함께 결부시키는 것은 SP들과 연관된 메모리 리소스들을 관리하는 것을 돕는데, 이는 워크그룹들과 연관된 데이터가 다수의 커널들에 의해 공유될 수 있기 때문이다. 그러나, 용어 "워크그룹" 은 일반적으로 명령들의 그룹을 지칭하는 점을 이해해야 한다. 예를 들어, "워크그룹" 은 "CUDA (Compute Unified Device Architecture)" (NVIDIA 코포레이션에 의해 개발됨, 2010 년 9 월 17 일에 발표된 버전 3.2) "스레드 블록" 으로 지칭될 수도 있다.

[0090] 또한, 워크그룹 및 커널 지정들은 단지 실시예로서 제공됨을 이해해야 한다. 본 개시물의 메모리 관리 양태들은 GPU 애플리케이션들의 다른 구성들에 적용될 수도 있다. 예를 들어, 다른 GPU 애플리케이션들은 실행 동안 동일한 입력 데이터를 1 회보다 많이 이용하는 명령들을 포함하는 단일의 상대적으로 보다 큰 "커널" 을 포함할 수도 있다. 이러한 실시예에서, 본 개시물의 양태들은 메모리 리소스들을 관리하는 데 여전히 적용될 수도 있다. 동일한 입력 데이터를 이용하는 명령들을 함께 결부시키는 명령 스트림들이 생성될 수도 있지만, 그 명령들은 동일한 커널에 속한다.

[0091] 하나 이상의 실시예들에서, 설명된 기능들은, 하드웨어, 하드웨어 상에서 실행되는 소프트웨어, 하드웨어 상에서 실행되는 펌웨어, 또는 이들의 임의의 조합으로 구현될 수도 있다. 몇몇 실시예들에서, 컴퓨터 판독가능 매체 상에 저장된 명령들은 하드웨어 콤포넌트들로 하여금 전술된 그들의 각각의 기능을 수행하게 할 수도 있다. 컴퓨터 판독가능 매체들은 컴퓨터 데이터 스토리지 매체들을 포함할 수도 있다. 데이터 스토리지 매체들은 본 명세서에서 설명된 기법들의 구현을 위한 명령들, 코드 및/또는 데이터 구조들을 취출하기 위해 하나 이상의 컴퓨터들 또는 하나 이상의 프로세서들에 의해 액세스될 수 있는 임의의 이용가능한 매체일 수도 있다. 비제한적인 실시예로서, 이러한 컴퓨터 판독가능 매체들은 RAM, ROM, EEPROM, CD-ROM 또는 다른 광학 디스크 스토리지, 자기 디스크 스토리지, 또는 다른 자기 스토리지 디바이스들, 플래시 메모리, 또는 희망하는 프로그램 코드를 명령들 또는 데이터 구조들의 형태로 전달하거나 또는 저장하는 데 사용될 수 있으며 컴퓨터에 의해 액세스될 수 있는 임의의 다른 매체들을 포함할 수 있다. 전술한 사항의 조합들은 또한 컴퓨터 판독가능 매체들의 범주 내에 포함되어야 한다.

[0092] 하나 이상의 DSP들, 범용 마이크로프로세서들, ASIC들, FPGA들, 또는 다른 등가의 집적 또는 이산 로직 회로부와 같은 하나 이상의 프로세서들에 의해 코드가 실행될 수도 있다. 따라서, 본 명세서에서 사용된 용어 "프로세서" 는 임의의 전술된 구조 또는 본 명세서에서 설명된 기법들의 구현에 적합한 임의의 다른 구조를 지칭할

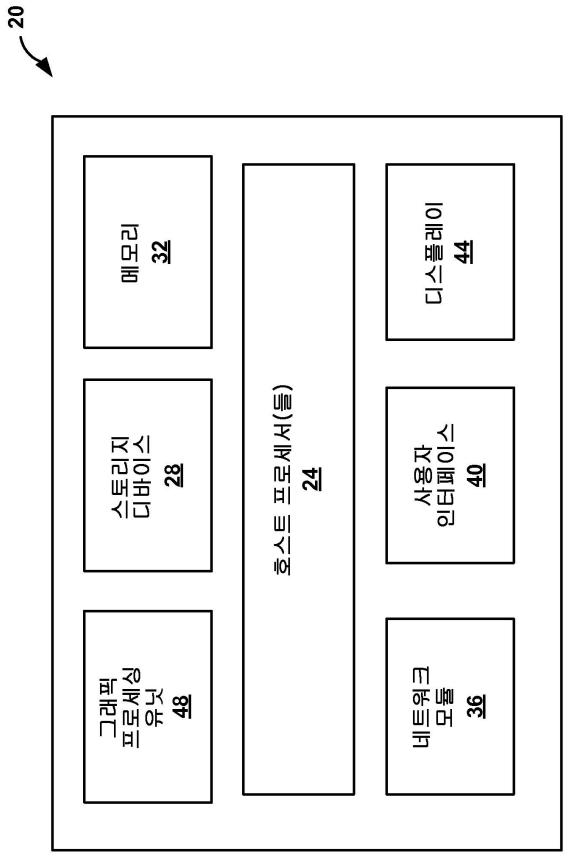
수도 있다. 추가로, 몇몇 양태들에서, 본 명세서에서 설명된 기능성은 인코딩 및 디코딩을 위해 구성된 전용 하드웨어 및/또는 소프트웨어 모듈들 내에 제공될 수도 있고, 또는 결합형 코덱에 통합될 수도 있다. 또한, 기법들은 하나 이상의 회로들 또는 로직 엘리먼트들에서 완전히 구현될 수 있다.

[0093] 본 개시물의 기법들은, 무선 핸드셋, 집적 회로 (IC) 또는 IC들의 세트 (예컨대, 칩 셋) 를 포함한 다양한 디바이스들 또는 장치들에서 구현될 수도 있다. 개시된 기법들을 수행하도록 구성된 디바이스들의 기능적 양태들을 강조하기 위해 다양한 콤포넌트들, 모듈들, 또는 유닛들이 본 개시물에서 설명되었지만, 반드시 상이한 하드웨어 유닛들에 의한 실현을 요구하는 것은 아니다. 오히려, 전술된 바와 같이, 다양한 유닛들은, 적절한 소프트웨어 및/또는 펌웨어와 연계하여, 전술된 바와 같은 하나 이상의 프로세서들을 포함하는 상호동작하는 하드웨어 유닛들의 집합에 의해 조합될 수도 있다.

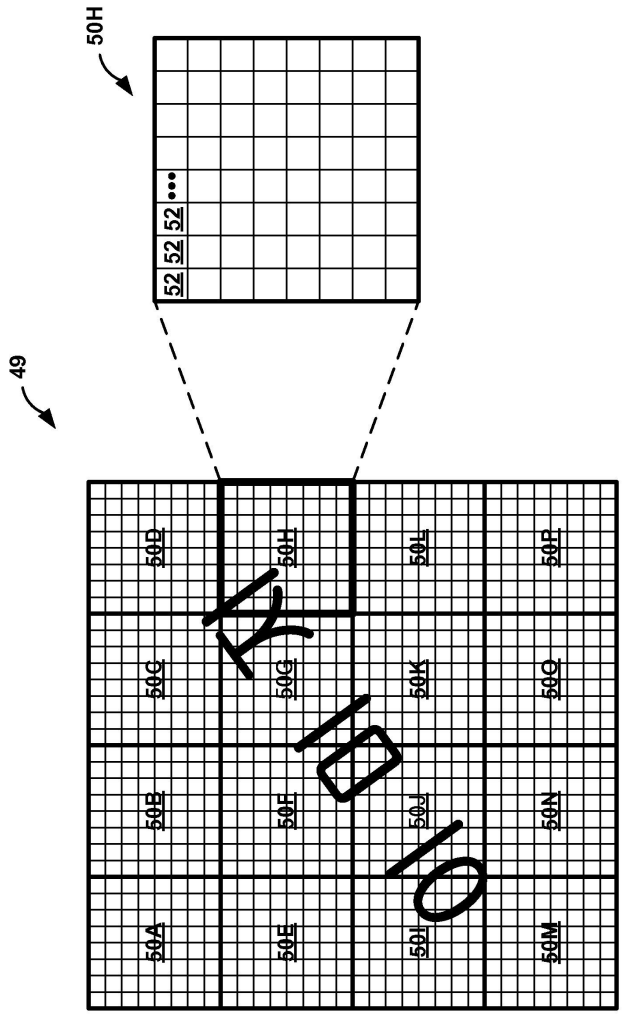
[0094] 다양한 실시예들이 설명되었다. 이들 및 다른 실시예들은 하기의 청구범위의 범주 내에 있다.

도면

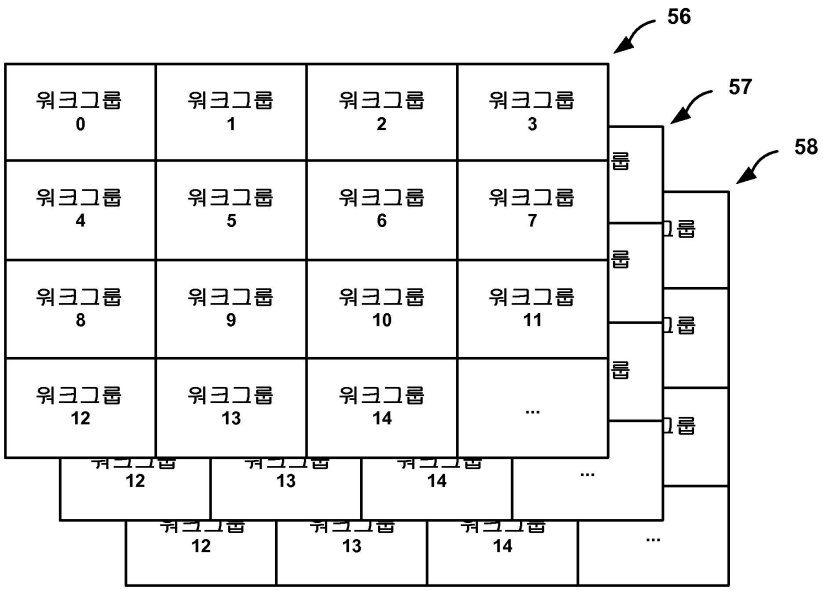
도면1



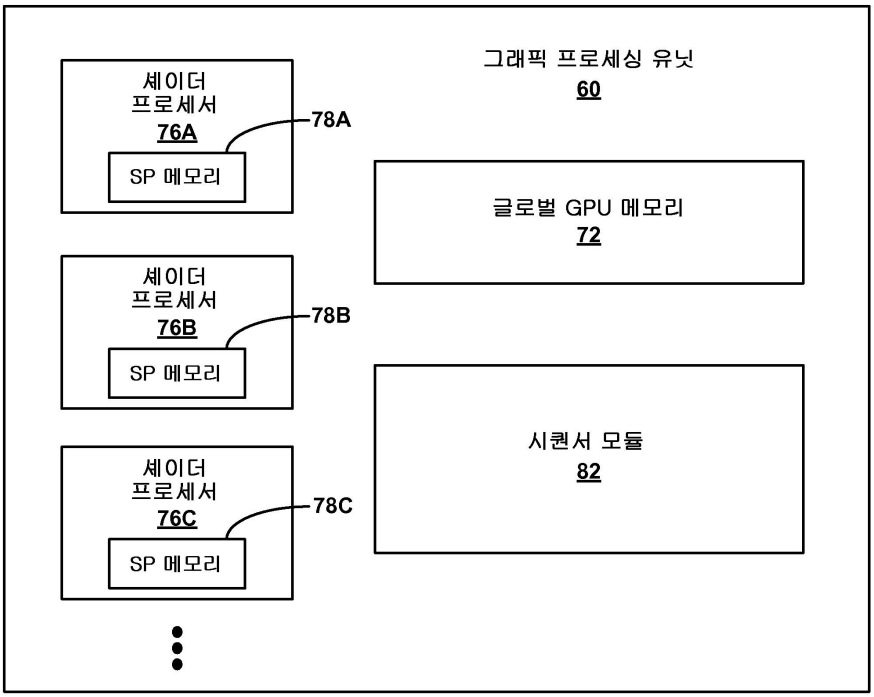
도면2


50H
52
52
52
49
50D
50H
50L
50P
50C
50G
50K
50O
50B
50F
50J
50N
50A
50E
50I
50M

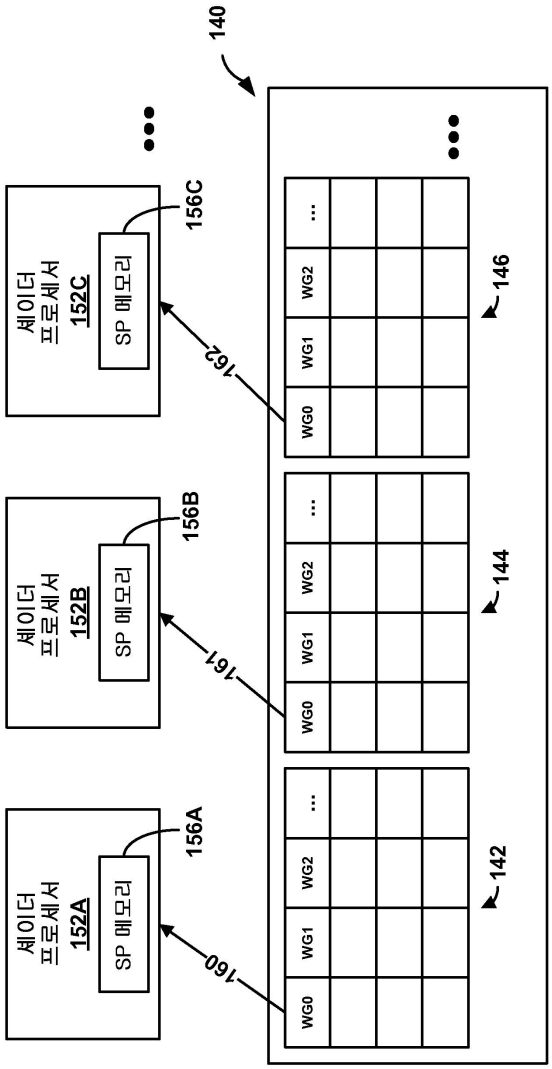
도면3


56
57
58
워크그룹 0
워크그룹 1
워크그룹 2
워크그룹 3
워크그룹 4
워크그룹 5
워크그룹 6
워크그룹 7
워크그룹 8
워크그룹 9
워크그룹 10
워크그룹 11
워크그룹 12
워크그룹 13
워크그룹 14
...

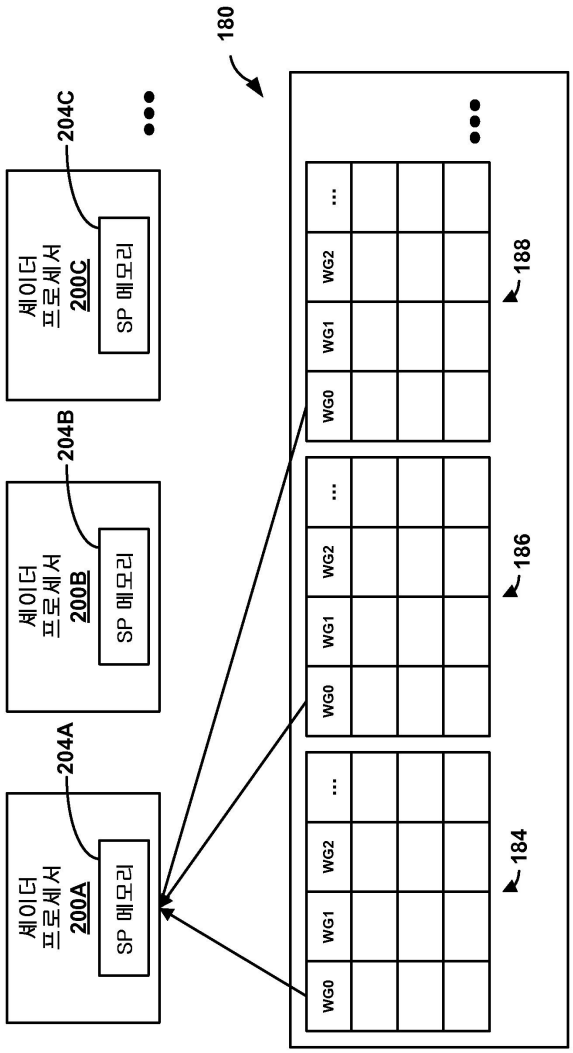
도면4


그래픽 프로세싱 유닛
60
셰이더 프로세서
76A
78A
SP 메모리
셰이더 프로세서
76B
78B
SP 메모리
셰이더 프로세서
76C
78C
SP 메모리
글로벌 GPU 메모리
72
시퀀서 모듈
82

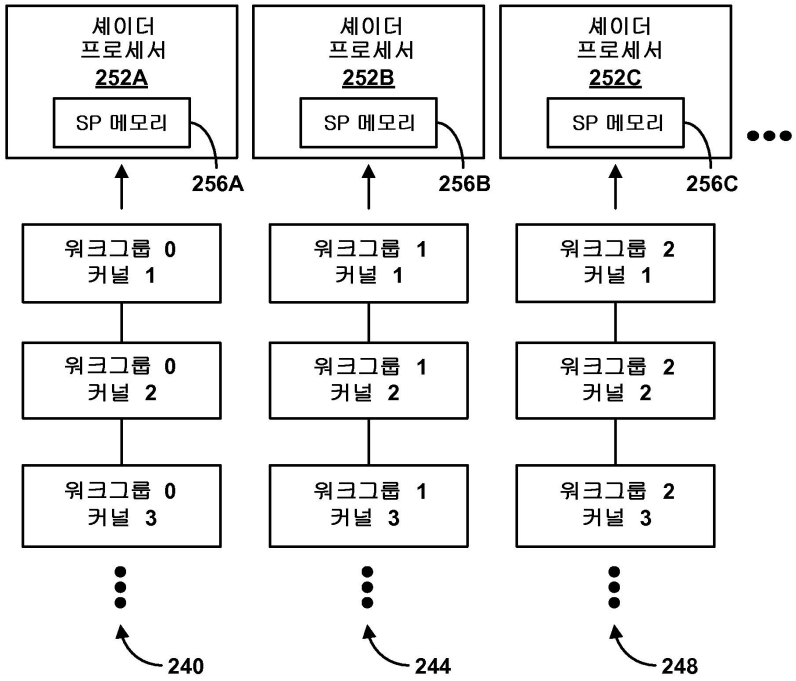
도면5


140
셰이더 프로세서 152C
SP 메모리
156C
162
셰이더 프로세서 152B
SP 메모리
156B
161
셰이더 프로세서 152A
SP 메모리
156A
160
WG0
WG1
WG2
...
142
144
146

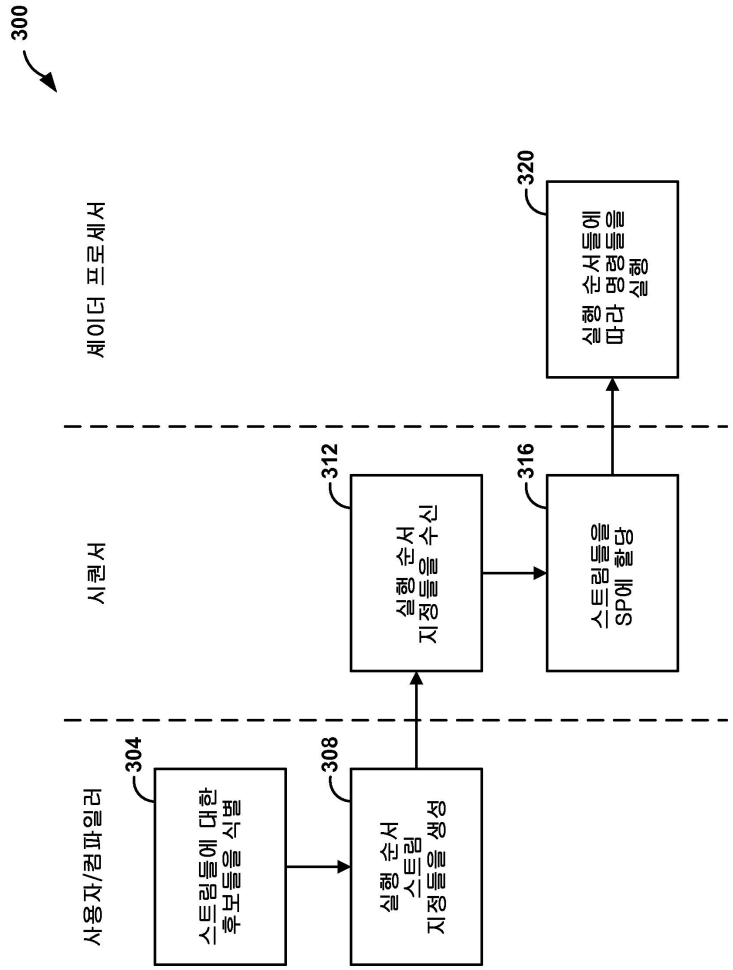
도면6


180
204C
셰이더 프로세서 200C
SP 메모리
204B
셰이더 프로세서 200B
SP 메모리
204A
셰이더 프로세서 200A
SP 메모리
WG0
WG1
WG2
...
184
186
188

도면7


셰이더
프로세서
252A
SP 메모리
256A
셰이더
프로세서
252B
SP 메모리
256B
셰이더
프로세서
252C
SP 메모리
256C
워크그룹 0
커널 1
워크그룹 0
커널 2
워크그룹 0
커널 3
240
워크그룹 1
커널 1
워크그룹 1
커널 2
워크그룹 1
커널 3
244
워크그룹 2
커널 1
워크그룹 2
커널 2
워크그룹 2
커널 3
248

도면8


300
쉐이더 프로세서
시퀀서
사용자/컴파일러
320
실행 순서들에 따라 명령들을 실행
312
실행 순서 지정들을 수신
316
스트림들을 SP에 할당
304
스트림들에 대한 후보들을 식별
308
실행 순서 스트림 지정들을 생성