(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2021년05월17일
(11) 등록번호 10-2253426
(24) 등록일자 2021년05월12일

(51) 국제특허분류(Int. Cl.)
G06F 9/48 (2018.01) *G06F 9/52* (2018.01)
G06T 1/20 (2018.01)
(52) CPC특허분류
G06F 9/4843 (2013.01)
G06F 9/522 (2013.01)
(21) 출원번호 10-2016-7009971
(22) 출원일자(국제) 2014년09월10일
심사청구일자 2019년08월23일
(85) 번역문제출일자 2016년04월15일
(65) 공개번호 10-2016-0065121
(43) 공개일자 2016년06월08일
(86) 국제출원번호 PCT/US2014/054966
(87) 국제공개번호 WO 2015/050681
국제공개일자 2015년04월09일
(30) 우선권주장
14/043,562 2013년10월01일 미국(US)
(56) 선행기술조사문헌
Minsoo Rhu 외 1명. 'CAPRI: Prediction of Compaction-Adequacy for Handling Control-Divergence in GPGPU Architectures'. ACM SIGARCH Computer Architecture News, 2012.06., pp.1-11.
(뒷면에 계속)

(73) 특허권자
퀄컴 인코포레이티드
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
(72) 발명자
메이 춘휘
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
부르드 알렉세이 블라디미로비치
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
천 린
미국 92121-1714 캘리포니아주 샌디에고 모어하우스 드라이브 5775
(74) 대리인
특허법인코리아나

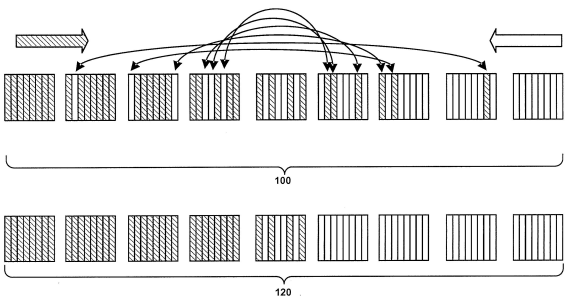
전체 청구항 수 : 총 30 항 심사관 : 유진태

(54) 발명의 명칭 **GPU 다이버전스 배리어**

(57) 요 약

디바이스는, 메모리, 및 적어도 하나의 프로그래밍가능 프로세서를 포함하며, 적어도 하나의 프로그래밍가능 프로세서는 복수의 와프들 중 각각의 와프에 대하여, 불 연산식이 각각의 와프의 대응하는 스레드에 대하여 참인지의 여부를 결정하고, 연산식이 참인 대응하는 스레드를 갖는 각각의 와프의 실행을 중지하고; 연산식이 참인 복수의 와프들 각각에 대하여 액티브 스레드들의 수를 결정하고; 복수의 와프들 각각에서의 액티브 스레드들의 수에 기초하여 연산식이 참인 복수의 와프들을 분류하고; 복수의 와프들 중 제 1 와프의 액티브 스레드의 스레드 데이터를 복수의 와프들 중 제 2 와프의 비액티브 스레드의 스레드 데이터와 교환하고; 그리고 연산식이 참인 복수의 와프들 중 적어도 하나의 와프의 실행을 재개하도록 구성된다.

대 표 도



(52) CPC특허분류
G06T 1/20 (2013.01)

(56) 선행기술조사문헌
Wilson W. L. Fung 외 3명. 'Dynamic Warp Formation and Scheduling for Efficient GPU Control Flow'. 40th IEEE/ACM International Symposium on Microarchitecture, pp.407-418.
Michael Steffen 외 1명. 'Improving SIMT Efficiency of Global Rendering Algorithms ~'. Annulal IEEE/ACM International Symposium on Microarchitecture, 2010, pp.239-248.
KR1020090050977 A
Wilson W. L. Fung 외 1명. 'Thread Block Compaction for Efficient SIMT Control Flow'. IEEE International Symposium on High Performance Computer Architecture, 2011, pp.25-36.

명 세 서

청구범위

청구항 1

적어도 하나의 프로그래밍가능한 프로세서에 의해 수행되는 방법으로서,

복수의 와프들 중 각각의 와프에 대하여, 각각의 와프의 대응하는 스레드에 대해 다이버전스 배리어 명령의 불 연산식 인수 (Boolean expression argument) 가 참인지의 여부를 결정하는 단계;

연산식이 참인 상기 대응하는 스레드를 갖는 각각의 와프의 실행을 중지하는 단계;

상기 연산식이 참인 상기 복수의 와프들의 각각에 대하여 액티브 스레드들의 수를 결정하는 단계로서, 상기 액티브 스레드들은 브랜치 (branch) 를 취하는, 상기 액티브 스레드들의 수를 결정하는 단계;

복수의 분류된 와프들을 생성하기 위해 상기 복수의 와프들의 각각에서 상기 액티브 스레드들의 수에 기초하여 상기 연산식이 참인 상기 복수의 와프들을 분류하는 단계;

상기 복수의 분류된 와프들 각각에서의 상기 액티브 스레드들의 수에 기초하여 상기 복수의 와프들 중 제 1 와프의 액티브 스레드의 스레드 데이터를 상기 복수의 와프들 중 제 2 와프의 비액티브 스레드의 스레드 데이터와 교환하는 단계; 및

상기 연산식이 참인 복수의 스레드들 중 한 스레드를 갖는 상기 복수의 분류된 와프들 중 적어도 하나의 와프의 실행을 재개하는 단계를 포함하는, 적어도 하나의 프로그래밍가능한 프로세서에 의해 수행되는 방법.

청구항 2

제 1 항에 있어서,

상기 방법은,

상기 복수의 분류된 와프들 중 적어도 하나의 와프의 실행을 재개하기 전에, 상기 연산식이 참인 상기 복수의 스레드들의 각각의 스레드에 대하여 컨텍스트 데이터를 교환하는 단계를 더 포함하는, 적어도 하나의 프로그래밍가능한 프로세서에 의해 수행되는 방법.

청구항 3

제 1 항에 있어서,

상기 적어도 하나의 프로그래밍가능 프로세서는 그래픽 프로세싱 유닛 (GPU) 을 포함하는, 적어도 하나의 프로그래밍가능한 프로세서에 의해 수행되는 방법.

청구항 4

제 1 항에 있어서,

상기 액티브 스레드의 상기 스레드 데이터는 상기 액티브 스레드의 레지스터 데이터를 포함하고, 그리고

상기 비액티브 스레드의 상기 스레드 데이터는 상기 비액티브 스레드의 레지스터 데이터를 포함하는, 적어도 하나의 프로그래밍가능한 프로세서에 의해 수행되는 방법.

청구항 5

제 1 항에 있어서,

상기 복수의 와프들을 분류하는 단계는 큐에 삽입된 와프들을 분류하기 위한 삽입 분류를 이용하여 상기 복수의 와프들을 분류하는 단계를 포함하는, 적어도 하나의 프로그래밍가능한 프로세서에 의해 수행되는 방법.

청구항 6

제 1 항에 있어서,

상기 방법은, 상기 불 연산식 인수가 참인지의 여부를 결정하는 단계 이후에 그리고 상기 복수의 와프들을 분류하는 단계 이전에,

상기 연산식이 참인 상기 복수의 와프들의 각각의 와프에 대해, 복수의 다이버전스 배리어들 중 연관된 다이버전스 배리어를 결정하는 단계; 및

각각의 와프의 상기 연관된 다이버전스 배리어에 기초하여 상기 복수의 와프들의 각각의 와프를 복수의 압축 풀들로 그룹화하는 단계를 더 포함하고,

상기 복수의 와프들을 분류하는 단계는, 상기 복수의 압축 풀들 중 동일한 하나에 속하는 상기 복수의 와프들을 분류하는 단계를 포함하고,

상기 제 1 와프 및 상기 제 2 와프는 상기 복수의 압축 풀들 중 동일한 하나에 속하는 와프들을 포함하고,

상기 연산식이 참인 상기 복수의 와프들 중 적어도 하나의 와프의 실행을 재개하는 단계는, 상기 동일한 하나의 압축 풀의 적어도 하나의 와프의 실행을 재개하는 단계를 포함하는, 적어도 하나의 프로그래밍가능한 프로세서에 의해 수행되는 방법.

청구항 7

제 6 항에 있어서,

상기 방법은, 상기 연관된 다이버전스 배리어를 결정하는 단계 이후에 그리고 상기 복수의 압축 풀들로 그룹화하는 단계 이전에,

상기 복수의 와프들의 각각과 연관된 상기 다이버전스 배리어에 기초하여 상기 복수의 와프들의 각각에 프리픽스를 배정하는 단계를 더 포함하고,

상기 복수의 와프들을 적어도 하나의 압축 풀들로 그룹화하는 것은, 배정된 상기 프리픽스에 기초하여 상기 복수의 와프들을 적어도 하나의 압축 풀들로 그룹화하는 것을 포함하는, 적어도 하나의 프로그래밍가능한 프로세서에 의해 수행되는 방법.

청구항 8

제 1 항에 있어서,

상기 액티브 스레드의 스레드 데이터를 비액티브 스레드의 스레드 데이터와 교환하는 단계는, 비액티브 와프들이 형성될 수 없을 때까지 계속되는, 적어도 하나의 프로그래밍가능한 프로세서에 의해 수행되는 방법.

청구항 9

제 1 항에 있어서,

상기 교환하는 단계는 상기 복수의 와프들이 모든 액티브 스레드들을 갖는 와프를 포함하는 것으로 결정하는 단계를 포함하고,

상기 복수의 분류된 와프들 중 적어도 하나의 와프의 실행을 재개하는 단계는, 모든 액티브 스레드들을 갖는 상기 와프의 실행을 재개하는 단계를 포함하는, 적어도 하나의 프로그래밍가능한 프로세서에 의해 수행되는 방법.

청구항 10

제 1 항에 있어서,

상기 연산식이 참인 상기 복수의 와프들을 분류하는 단계는:

큐에서 상기 복수의 와프들을 저장하는 단계;

상기 액티브 스레드들의 수에 기초하여 상기 연산식이 참인 상기 복수의 와프들을 상기 큐에서 분류하는 단계; 및

상기 큐에서 분류된 상기 복수의 와프들을 저장하는 단계를 포함하는, 적어도 하나의 프로그래밍가능한 프로세

서에 의해 수행되는 방법.

청구항 11

제 1 항에 있어서,

상기 불 연산식 인수가 참인지의 여부를 결정하는 단계는:

다이버전스가 발생할 가능성이 있는 로케이션, 및 복수의 와프들을 실행하는 커넬 내에서 성능에 영향을 주는 로케이션 중 적어도 하나의 로케이션을 결정하는 단계; 및

상기 적어도 하나의 로케이션에서 다이버전스 배리어 명령을 상기 커넬 내에 삽입하는 단계를 더 포함하고,

상기 불 연산식은 상기 다이버전스 배리어 명령과 연관되는, 적어도 하나의 프로그래밍가능한 프로세서에 의해 수행되는 방법.

청구항 12

복수의 와프들 중 각각의 와프에 대하여, 각각의 와프의 대응하는 스레드에 대해 다이버전스 배리어 명령의 불 연산식 인수가 참인지의 여부를 결정하는 수단;

연산식이 참인 상기 대응하는 스레드를 갖는 각각의 와프의 실행을 중지하는 수단;

상기 연산식이 참인 상기 복수의 와프들의 각각에 대하여 액티브 스레드들의 수를 결정하는 수단으로서, 상기 액티브 스레드들은 브랜치를 취하는, 상기 액티브 스레드들의 수를 결정하는 수단;

복수의 분류된 와프들을 생성하기 위해 상기 복수의 와프들 각각에서 상기 액티브 스레드들의 수에 기초하여 상기 연산식이 참인 상기 복수의 와프들을 분류하는 수단;

상기 복수의 분류된 와프들 각각에서의 상기 액티브 스레드들의 수에 기초하여 상기 복수의 와프들 중 제 1 와프의 액티브 스레드의 스레드 데이터를 상기 복수의 와프들 중 제 2 와프의 비액티브 스레드의 스레드 데이터와 교환하는 수단; 및

상기 연산식이 참인 복수의 스레드들 중 한 스레드를 갖는 상기 복수의 분류된 와프들 중 적어도 하나의 와프의 실행을 재개하는 수단을 포함하는, 장치.

청구항 13

제 12 항에 있어서,

상기 장치는,

상기 복수의 분류된 와프들 중 적어도 하나의 와프의 실행을 재개하기 전에, 상기 연산식이 참인 상기 복수의 스레드들의 각각의 스레드에 대하여 컨텍스트 데이터를 교환하는 수단을 더 포함하는, 장치.

청구항 14

제 12 항에 있어서,

상기 장치는 그래픽 프로세싱 유닛 (GPU) 을 포함하는, 장치.

청구항 15

제 12 항에 있어서,

상기 액티브 스레드의 상기 스레드 데이터는 상기 액티브 스레드의 레지스터 데이터를 포함하고, 그리고

상기 비액티브 스레드의 상기 스레드 데이터는 상기 비액티브 스레드의 레지스터 데이터를 포함하는, 장치.

청구항 16

제 12 항에 있어서,

상기 복수의 와프들을 분류하는 수단은 큐에 삽입된 와프들을 분류하기 위한 삽입 분류를 이용하여 상기 복수의

와프들을 분류하는 수단을 포함하는, 장치.

청구항 17

제 12 항에 있어서,

상기 연산식이 참인 상기 복수의 와프들의 각각의 와프에 대해, 복수의 다이버전스 배리어들 중 연관된 다이버전스 배리어를 결정하는 수단; 및

각각의 와프의 상기 연관된 다이버전스 배리어에 기초하여 상기 복수의 와프들의 각각의 와프를 복수의 압축 풀들로 그룹화하는 수단을 더 포함하고,

상기 복수의 와프들을 분류하는 수단은, 상기 복수의 압축 풀들 중 동일한 하나에 속하는 상기 복수의 와프들을 분류하는 수단을 포함하고,

상기 제 1 와프 및 상기 제 2 와프는 상기 복수의 압축 풀들 중 동일한 하나에 속하는 와프들을 포함하고,

상기 연산식이 참인 상기 복수의 와프들 중 적어도 하나의 와프의 실행을 재개하는 수단은, 상기 동일한 하나의 압축 풀의 적어도 하나의 와프의 실행을 재개하는 수단을 포함하는, 장치.

청구항 18

제 17 항에 있어서,

상기 복수의 와프들의 각각과 연관된 상기 다이버전스 배리어에 기초하여 상기 복수의 와프들의 각각에 프리픽스를 배정하는 수단을 더 포함하고,

상기 복수의 와프들을 적어도 하나의 압축 풀들로 그룹화하는 수단이, 배정된 상기 프리픽스에 기초하여 복수의 와프들을 적어도 하나의 압축 풀들로 그룹화하는 수단을 포함하는, 장치.

청구항 19

제 12 항에 있어서,

상기 액티브 스레드의 스레드 데이터를 비액티브 스레드의 스레드 데이터와 교환하는 수단은, 비액티브 와프들이 형성될 수 없을 때까지 계속 진행하는, 장치.

청구항 20

제 12 항에 있어서,

상기 복수의 와프들이 모든 액티브 스레드들을 갖는 와프를 포함하는 것으로 결정하는 수단; 및

모든 액티브 스레드들을 갖는 상기 와프의 실행을 재개하는 수단을 더 포함하는, 장치.

청구항 21

제 12 항에 있어서,

상기 연산식이 참인 상기 복수의 와프들을 분류하는 수단은:

큐에서 상기 복수의 와프들을 저장하는 수단;

상기 액티브 스레드들의 수에 기초하여 상기 연산식이 참인 상기 복수의 와프들을 상기 큐에서 분류하는 수단; 및

상기 큐에서 분류된 상기 복수의 와프들을 저장하는 수단을 포함하는, 장치.

청구항 22

제 12 항에 있어서,

다이버전스가 발생할 가능성이 있는 로케이션, 및 복수의 와프들을 실행하는 커넬 내에서 성능에 영향을 주는 로케이션 중 적어도 하나의 로케이션을 결정하는 수단; 및

상기 적어도 하나의 로케이션에서 다이버전스 배리어 명령을 상기 커넬 내에 삽입하는 수단을 더 포함하고,

상기 불 연산식은 상기 다이버전스 배리어 명령과 연관되는, 장치.

청구항 23

명령들을 포함하는 비일시적 컴퓨터 판독가능 저장 매체로서,

상기 명령들은 실행될 때 적어도 하나의 프로그래밍가능 프로세서로 하여금,

복수의 와프들 중 각각의 와프에 대하여, 각각의 와프의 대응하는 스레드에 대해 다이버전스 배리어 명령의 불 연산식 인수가 참인지의 여부를 결정하게 하고;

연산식이 참인 상기 대응하는 스레드를 갖는 각각의 와프의 실행을 중지하게 하는 것으로서, 액티브 스레드들은 브랜치를 취하는, 상기 각각의 와프의 실행을 중지하게 하고;

상기 연산식이 참인 상기 복수의 와프들의 각각에 대하여 액티브 스레드들의 수를 결정하게 하고;

복수의 분류된 와프들을 생성하기 위해 상기 복수의 와프들의 각각에서 상기 액티브 스레드들의 수에 기초하여 상기 연산식이 참인 상기 복수의 와프들을 분류하게 하고;

상기 복수의 분류된 와프들 각각에서의 상기 액티브 스레드들의 수에 기초하여 상기 복수의 와프들 중 제 1 와프의 액티브 스레드의 스레드 데이터를 상기 복수의 와프들 중 제 2 와프의 비액티브 스레드의 스레드 데이터와 교환하게 하고; 그리고

상기 연산식이 참인 복수의 스레드들 중 한 스레드를 갖는 상기 복수의 분류된 와프들 중 적어도 하나의 와프의 실행을 재개하게 하는, 명령들을 포함하는 비일시적 컴퓨터 판독가능 저장 매체.

청구항 24

메모리; 및

적어도 하나의 프로그래밍가능 프로세서를 포함하고,

상기 적어도 하나의 프로그래밍 프로세서는:

복수의 와프들 중 각각의 와프에 대하여, 각각의 와프의 대응하는 스레드에 대해 다이버전스 배리어 명령의 불 연산식 인수가 참인지의 여부를 결정하고;

연산식이 참인 상기 대응하는 스레드를 갖는 각각의 와프의 실행을 중지하고;

상기 연산식이 참인 상기 복수의 와프들의 각각에 대하여 액티브 스레드들의 수를 결정하는 것으로서, 상기 액티브 스레드들은 브랜치를 취하는, 상기 액티브 스레드들의 수를 결정하고;

복수의 분류된 와프들을 생성하기 위해 상기 복수의 와프들의 각각에서 상기 액티브 스레드들의 수에 기초하여 상기 연산식이 참인 상기 복수의 와프들을 분류하고;

상기 복수의 분류된 와프들 각각에서의 상기 액티브 스레드들의 수에 기초하여 상기 복수의 와프들 중 제 1 와프의 액티브 스레드의 스레드 데이터를 상기 복수의 와프들 중 제 2 와프의 비액티브 스레드의 스레드 데이터와 교환하고; 그리고

상기 연산식이 참인 복수의 스레드들 중 한 스레드를 갖는 상기 복수의 분류된 와프들 중 적어도 하나의 와프의 실행을 재개하도록 구성되는, 장치.

청구항 25

제 24 항에 있어서,

상기 적어도 하나의 프로그래밍 프로세서는 또한,

상기 복수의 분류된 와프들 중 적어도 하나의 와프의 실행을 재개하기 전에, 상기 연산식이 참인 상기 복수의 스레드들의 각각의 스레드에 대하여 컨텍스트 데이터를 교환하도록 계속 진행되는, 장치.

청구항 26

제 24 항에 있어서,

상기 장치는 그래픽 프로세싱 유닛 (GPU) 을 포함하는, 장치.

청구항 27

제 24 항에 있어서,

상기 액티브 스레드의 상기 스레드 데이터는 상기 액티브 스레드의 레지스터 데이터를 포함하고, 그리고

상기 비액티브 스레드의 상기 스레드 데이터는 상기 비액티브 스레드의 레지스터 데이터를 포함하는, 장치.

청구항 28

제 24 항에 있어서,

상기 적어도 하나의 프로그래밍가능 프로세서는 또한,

상기 연산식이 참인 상기 복수의 와프들의 각각의 와프에 대해, 복수의 다이버전스 배리어들 중 연관된 다이버전스 배리어를 결정하고;

각각의 와프의 상기 연관된 다이버전스 배리어에 기초하여 상기 복수의 와프들의 각각의 와프를 복수의 압축 풀들로 그룹화하도록 구성되고,

상기 복수의 와프들을 분류하는 것은, 상기 적어도 하나의 프로그래밍가능 프로세서가 또한, 복수의 압축 풀들 중 동일한 하나에 속하는 상기 복수의 와프들을 분류하도록 구성되는 것을 포함하고,

상기 제 1 와프 및 상기 제 2 와프는 상기 복수의 압축 풀들 중 동일한 하나에 속하는 와프들을 포함하고,

상기 연산식이 참인 상기 복수의 와프들 중 적어도 하나의 와프의 실행을 재개하기 위해, 상기 적어도 하나의 프로그래밍가능 프로세서는 상기 동일한 하나의 압축 풀의 적어도 하나의 와프의 실행을 재개하도록 구성되는, 장치.

청구항 29

제 28 항에 있어서,

상기 적어도 하나의 프로그래밍가능 프로세서는 또한,

상기 복수의 와프들의 각각과 연관된 상기 다이버전스 배리어에 기초하여 상기 복수의 와프들의 각각에 프리픽스를 배정하고,

상기 적어도 하나의 프로그래밍가능 프로세서로 하여금 상기 복수의 와프들을 적어도 하나의 압축 풀로 그룹화하도록 하는 명령들이, 상기 적어도 하나의 프로그래밍가능 프로세서로 하여금 배정된 상기 프리픽스에 기초하여 상기 복수의 와프들을 상기 적어도 하나의 압축 풀로 그룹화하게 하는 명령들을 포함하는, 장치.

청구항 30

제 24 항에 있어서,

상기 연산식이 참인 상기 복수의 오프들을 분류하기 위해, 상기 적어도 하나의 프로그래밍가능 프로세서는:

큐에서 상기 복수의 와프들을 저장하고;

상기 액티브 스레드들의 수에 기초하여 상기 연산식이 참인 상기 복수의 와프들을 상기 큐에서 분류하고; 그리고

상기 큐에서 분류된 상기 복수의 와프들을 저장하도록 구성되는, 장치.

발명의 설명

기 술 분 야

[0001] 본 개시물은 그래픽 프로세싱에 관한 것이고, 보다 구체적으로, 그래픽 프로세싱 유닛 (GPU) 의 스레드들의 실

행을 관리하는 기술들에 관한 것이다.

배 경 기 술

[0002] 최근, 소위 범용 GPU들 (GPGPU들) 을 향한 움직임이 있어 왔다. 그래픽 렌더링들을 수행하는 통상적인 GPU들과 달리, GPGPU들은 "커널"로서 종종 지칭되는 범용 태스크 또는 프로그램을 실행하도록 구성될 수도 있다.

태스크들의 일부 유형들은 중앙 프로세싱 (CPU) 또는 GPU 와 같은 특정 유형의 프로세서에 보다 잘 적합할 수도 있다. CPU들은 보다 많은 브랜치들, 점프들, 및 조건적 논리를 갖는 태스크들에 적합한 반면, GPU들은 많은 부동소수점 계산들을 갖는 고병렬 태스크들에 적절할 수도 있다. 많은 GPU들이 SIMD 하드웨어 아키텍처를 가질 수도 있기 때문에, GPU들은 또한 SIMD (Single Instruction multiple Data) 명령들을 실행하는 능력을 포함할 수도 있다. GPU 가 SIMD 명령을 실행할 때, GPU 는 다수의 데이터 값들에 대하여 명령에 의해 표시되는 동일 동작을 실행할 수도 있다. 통상적으로, GPU 는 SIMD 명령에 의해 표시되는 동작들을 병렬로 실행가능한 다수의 실행 유닛들을 갖는다.

발명의 내용

해결하려는 과제

과제의 해결 수단

[0003] 본 개시물의 기술들은 그래픽 프로세싱 유닛 (GPU) 상에서 실행하는 스레드들 간의 다이버전스를 감소시키기 위한 기술들을 제공한다. GPU 는 "다이버전스 배리어" 명령으로서 지칭되는 명령에 대한 서포트를 포함할 수도 있다. 다이버전스 배리어 명령은 스레드들이 동일한 명령을 실행하게 하도록 다수의 와프들로부터 새로운 와프들로 다이버전트 스레드들을 그룹화하도록 시도하여, 이에 의해 GPU 성능을 개선시킨다.

[0004] 일 예에서, 본 개시물은, 복수의 와프들 중 각각의 와프에 대하여, 불 연산식 (Boolean expression) 이 각각의 와프의 대응하는 스레드에 대하여 참인지의 여부를 결정하는 단계; 연산식이 참인 대응하는 스레드를 갖는 각각의 와프의 실행을 중지하는 단계; 연산식이 참인 복수의 와프들 각각에 대하여 액티브 스레드들의 수를 결정하는 단계; 복수의 와프들 각각에서의 액티브 스레드들의 수에 기초하여 연산식이 참인 복수의 와프들을 분류하는 단계; 복수의 와프들 중 제 1 와프의 액티브 스레드의 스레드 데이터를 복수의 와프들 중 제 2 와프의 비액티브 스레드의 스레드 데이터와 교환하는 단계; 및 연산식이 참인 복수의 와프들 중 적어도 하나의 와프의 실행을 재개하는 단계를 포함하는 방법을 설명한다.

[0005] 다른 예에서, 본 개시물은, 메모리, 및 적어도 하나의 프로그래밍가능 프로세서를 포함하는 디바이스를 설명하며, 적어도 하나의 프로그래밍가능 프로세서는 복수의 와프들 중 각각의 와프에 대하여, 불 연산식이 각각의 와프의 대응하는 스레드에 대하여 참인지의 여부를 결정하고; 연산식이 참인 대응하는 스레드를 갖는 각각의 와프의 실행을 중지하고; 연산식이 참인 복수의 와프들 각각에 대하여 액티브 스레드들의 수를 결정하고; 복수의 와프들 각각에서의 액티브 스레드들의 수에 기초하여 연산식이 참인 복수의 와프들을 분류하고; 복수의 와프들 중 제 1 와프의 액티브 스레드의 스레드 데이터를 복수의 와프들 중 제 2 와프의 비액티브 스레드의 스레드 데이터와 교환하고; 그리고 연산식이 참인 복수의 와프들 중 적어도 하나의 와프의 실행을 재개하도록 구성된다.

[0006] 다른 예에서, 본 개시물은, 복수의 와프들 중 각각의 와프에 대하여, 불 연산식이 각각의 와프의 대응하는 스레드에 대하여 참인지의 여부를 결정하는 수단; 연산식이 참인 대응하는 스레드를 갖는 각각의 와프의 실행을 중지하는 수단; 연산식이 참인 복수의 와프들 각각에 대하여 액티브 스레드들의 수를 결정하는 수단; 복수의 와프들 각각에서의 액티브 스레드들의 수에 기초하여 연산식이 참인 복수의 와프들을 분류하는 수단; 복수의 와프들 중 제 1 와프의 액티브 스레드의 스레드 데이터를 복수의 와프들 중 제 2 와프의 비액티브 스레드의 스레드 데이터와 교환하는 수단; 및 연산식이 참인 복수의 와프들 중 적어도 하나의 와프의 실행을 재개하는 수단을 포함하는 장치를 설명한다.

[0007] 다른 예에서, 본 개시물은, 명령들을 저장한 비일시적 컴퓨터 판독가능 저장 매체를 설명하며, 명령들은 실행될 때, 적어도 하나의 프로그래밍가능 프로세서로 하여금, 연산식이 참인 복수의 와프들의 각각의 와프에 대하여, 복수의 다이버전스 배리어들 중 연관된 다이버전스 배리어를 결정하게 하고; 각각의 와프의 연관된 다이버전스 배리어에 기초하여 복수의 와프들의 각각의 와프를 복수의 압축 풀들로 그룹화하게 하며, 적어도 하나의 프로세서로 하여금 복수의 와프들을 분류하게 하는 명령들은 적어도 하나의 프로세서로 하여금 복수의 압축 풀들 중

동일한 하나에 속하는 복수의 와프들을 분류하게 하는 명령들을 포함하고, 제 1 와프와 제 2 와프는 복수의 압축 풀들 중 동일한 풀에 속하며, 적어도 하나의 프로세서로 하여금 조건이 참인 복수의 와프들 중 적어도 하나의 와프의 실행을 재개하게 하는 명령들은 동일한 하나의 압축 풀의 적어도 하나의 와프의 실행을 재개하는 것을 포함한다.

[0008] 본 개시물의 하나 이상의 예들의 세부 사항들은 첨부된 도면과 하기의 설명으로부터 설명된다. 다른 특징들, 목적들 및 이점들은 하기의 설명 및 도면들, 및 하기의 특허청구범위로부터 명확해질 것이다.

도면의 간단한 설명

[0009] 도 1 은 본 개시물의 기법들에 따라 GPU 다이버전스 배리어 명령의 실행을 지원할 수도 있는 예시적인 컴퓨팅 디바이스를 예시하는 블록도이다.

도 2 는 본 개시물의 기법들에 따라 복수의 프로세싱 엘리먼트 상에서 실행하는 와프를 예시하는 블록도이다.

도 3 은 본 개시물의 기법들에 따라 각각의 와프 내에서 액티브 스레드들의 수에 기초하여 와프들을 분류하는 것을 예시하는 개념도이다.

도 4 는 한 와프로부터의 액티브 스레드들을 다른 와프들로부터 비액티브 스레드들과 교환하기 위한 기법들을 예시하는 개념도이다.

도 5 는 본 개시물의 기법들에 따라 다수의 다어버전스 배리어 명령들을 핸들링하는 기법들을 예시하는 개념도이다.

도 6 은 본 개시물의 기법들에 따라 다이버전스 배리어 명령들을 실행하기 위한 기법들을 예시하는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0010] 본 개시물은 그래픽 프로세싱 유닛 (GPU) 상에서 실행하는 스레드들의 다이버전스를 감소시키기 위한 기법들에 대하여 교시된다. GPU 는 프로세싱 엘리먼트들 (PE들) 이라 지칭되는 다수의 실행 유닛들을 포함할 수도 있다. "커넬"이라고 지칭되는 프로그램은 GPU 의 하나 이상의 PE들 상에서 실행할 수도 있다. 애플리케이션은 GPU 의 기본 작업 유닛을 구성하는 다수의 스레드들로 커넬을 분할할 수도 있다. GPU 스케줄러는 또한, "와프" 라고 지칭되는 스레드 그룹으로 스레드들을 함께 그룹화할 수도 있다. 와프는 일부 그래픽 아키텍처들 상에서 특정 수의 스레드들, 예를 들어, 32 개의 스레드들을 포함할 수도 있다.

[0011] GPU 의 드라이버 또는 스케줄러는 GPU 상에서 커넬을 실행하도록 스레드들을 생성한다. 스레드는 GPU 상에서 프로세싱될 기본 데이터 유닛이며, CPU 스레드와는 혼동하지 않아야 한다. 스케줄러는 GPU 의 실행 유닛에 각각의 스레드를 배정할 수도 있다. 또한 프로세싱 엘리먼트들 ("PE들") 이라 지칭되는 실행 유닛들은 또한 다수의 데이터 값들에 대하여 동일한 명령의 병렬 실행을 할 수 있는 SIMD 유닛들이다.

[0012] 일반적으로, 와프의 각각의 스레드는 동일한 명령을 실행한다. 프로그램 카운터 (PC) 는 각각의 스레드가 실행할 명령의 메모리 어드레스를 저장한다. 일반적으로, 와프의 스레드들 각각에 대하여 단일의 PC 가 존재할 수도 있다. 각각의 와프에 대해 단일의 PC 를 갖는 것은, 스레드들 각각이 상이한 명령들을 실행할 필요가 없는 한은, 스레드들 각각이 동시에 실행하는 것을 허용한다.

[0013] 이때, 많은 GPU들은 플로우 제어 명령들을 실행하는 능력, 예를 들어, 브랜치, 점프, 진행 (goto) 그리고 다른 플로우 제어 명령들을 실행하는 능력을 포함한다. 플로우 제어 명령들은 복수의 방식들로 프로그램 실행의 플로우를 변경할 수 있다. 플로우 제어 명령들이 없는 프로그램 또는 커넬에서, PE 는 시작에서부터 종료까지 커넬의 명령들을 실행할 수도 있다. PE 가 명령을 실행하는 것을 종료한 후, GPU 는 (예를 들어, PC 값을 하나만큼 증분시킴으로써) PC 의 값을 메모리에서의 다음 명령의 어드레스로 설정하고 PE 는 다음 명령을 실행한다. 프로그램이 종료 시점에 도달하여 이 시점에서 실행이 만료할 때까지 프로그램을 실행하는 프로세스는 이 방식으로 플로우 제어 명령들 없이 프로그램을 계속한다.

[0014] 플로우 제어 명령을 실행하는 것은 PE 로 하여금, 증분된 PC 값 이외의 어드레스에서의 후속하는 명령을 실행하게 할 수도 있다. 증분된 PC 값의 어드레스에서 후속하는 명령을 실행하는 것 대신에, 플로우 제어 명령을 실행하는 PE 는 상이한 PC 어드레스, 이를 테면, 서브루틴의 어드레스 등을 갖는 후속하는 명령을 실행할 수도 있다. 따라서, 플로우 제어 명령은 프로그램의 실행 "플로우" 를 변경한다라고 한다.

[0015] 플로우 제어 명령들의 예들은 서브루틴 호출, 브랜치들, 리턴들, 점프들 등을 포함한다. 여러 예들에서, PE

가 "점프"하는 명령 어드레스, 즉, PC 에 배정된 어드레스는 런타임에서 스레드들 간에 변화하는 데이터의 값에 기초할 수도 있다. 플로우 제어 명령들은 또한 각각의 PE 가 개별적으로 평가하는 불 연산식과 연관될 수도 있다. 불 연산식은 참 또는 거짓에 대하여 평가하는 불 값을 생성하는 연산식이다. 불 연산식은 불 연산자들, 이를 테면, "and," "or", "not", "exclusive or (XOR)" 등을 포함할 수도 있다. 불 연산식은 또한 산술 테스트들, 이를 테면, 보다 더 큰, 보다 더 작은, 같은, 같지 않은, 이상, 이하 등을 포함할 수도 있다. 불 연산식의 참 또는 거짓은 스레드마다 변화하는 데이터 또는 값들에 의존할 수도 있다.

[0016] 따라서, 한 스레드가 점프하는 것도 가능하고 동일한 와프 내에서 다른 스레드와는 상이한 명령을 실행하는 것도 가능할 수도 있다. 그러나, 위에 진술한 바와 같이, 와프에 대하여 오직 하나의 PC 만이 존재한다. 와프의 둘 이상의 스레드들이 상이한 명령들을 실행하는 조건이 "다이버전스"로서 지칭된다. 다이버전스가 발생할 때, 일부 세트들의 스레드들은 동일한 명령들을 실행하는 것을 계속할 수도 있다. 그러나, 또한 상이한 명령들을 실행하는 다수의 세트의 스레드들이 존재할 수도 있다.

[0017] 스레드 다이버전스의 일 예로서, 와프의 제 1 스레드와 제 2 스레드는 플로우 제어 명령, 이를 테면, "if-else" 스테이트먼트, 또는 루프 스테이트먼트를 실행할 수도 있다. 제 1 스레드가 실행하는 후속하는 명령은 제 1 스레드의 레지스터에 저장된 데이터의 값들에 기초할 수도 있다. 이와 유사하게, 제 2 스레드의 후속하는 명령은 제 2 스레드의 레지스터에 저장된 데이터의 값에 기초할 수도 있다. 제 1 스레드 및 제 2 스레드가 상이한 레지스터 데이터를 가지면, 제 1 스레드 및 제 2 스레드는 상이한 명령 어드레스들과 연관된 상이한 후속하는 명령들로 점프할 수도 있다.

[0018] 와프 스레드가 다이버전스하는 경우, 스레드들은 "if-else" 스테이트먼트와 같이, 제어 플로우 블록들의 상이한 브랜치들을 취할 수도 있다. 루프 스테이트먼트의 경우, 와프 스레드들은 또한, 상이한 횟수들로 예를 들어, 상이한 횟수들의 루프의 반복들을 실행한 후에 루프 스테이트먼트를 종료할 수도 있다.

[0019] 예를 들어, if-else 스테이트먼트의 상이한 브랜치들을 취하거나 또는 루프의 상이한 횟수의 반복들을 수행하는 것에 의해 와프 스레드들이 다이버전스할 때, GPU 는 다이버전스에 의해 야기되는 상이한 실행 경로들 각각을 직렬화한다. 즉, GPU 는 "액티브"이고 동일한 명령들을 실행하는 스레드들을 결정한다. 스레드가 실행을 끝내거나 배리어, 이를 테면, 아래 보다 자세히 논의될 다이버전스 배리어 명령에 도달할 때까지 액티브 스레드들은 각각의 스레드와 연관된 PE들 상에서 실행하는 것을 계속한다.

[0020] 직렬화 동안에, GPU 는 현재 실행중이지 않은 스레드들을 결정하고, 이들 비액티브 스레드들 및 이들의 연관된 PE들을 유휴 상태로 설정한다. PE들이 유휴상태로 설정되는 동안, 비액티브 스레드들은 실행을 행하지 않아 GPU 성능을 손상시킨다. 일부 경우들에서, 다이버전스 스레드들은 추가로 다이버즈할 수도 있다, 즉, 다수의 "레벨들" 또는 "네스트된 다이버전스" 가 존재할 수도 있다. 네스트된 다이버전스를 핸들링하기 위해, GPU 는 네스트된 브랜치들 및 루프들을 추적하기 위해 컨버전스 스택을 이용한다. GPU 는 먼저 다이버전스의 가장 깊거나 또는 가장 안쪽의 층을 핸들링하고, 실행이 완료 또는 중지할 때까지 다이버전스의 가장 깊은 레벨을 가진 스레드들을 실행시킨다. 그 후, GPU 는 컨버전스 스택으로부터 다이버전스의 레벨을 제거하고, 컨버전스 스택 상의 가장 안쪽의 나머지 스레드를 실행하고 컨버전스 스택으로부터 완료된 스레드를 제거하는 프로세스를 반복한다. 일단 스레드가 브랜치 또는 루프를 실행하는 것을 끝내면, GPU 는 더 이상 다이버전스하지 않는 와프들을 형성하기 위해 스레드들을 함께 다시 재결합하거나 또는 수렴할 수도 있다.

[0021] 본 개시물의 기법들은 "다이버전스 배리어" 명령으로 지칭되는, GPU 가 지원할 수도 있는 명령을 도입한다. 여러 예들에서, 애플리케이션 프로그래밍 인터페이스 (API) 는 다이버전스 배리어 명령에 대한 서포트를 포함할 수도 있다. 이러한 API들은 OpenCL (Open Compute Language), OpenGL (Open Graphics Language), 및 Microsoft DirectX API 들을 포함할 수도 있다. 특정 API 로 GPU 를 프로그래밍할 때, 프로그래머는 다이버전스 배리어 함수 호출들을 삽입할 수도 있으며, 이는 GPU 로 하여금, 다이버전스가 성능에 상당히 영향을 줄 수 있는 코드 포인트들에서, 다이버전스 배리어 명령을 실행하게 한다. GPU 드라이버 또는 컴파일러는 또한, 다이버전스가 성능에 상당한 영향을 줄 수 있는 코드 포인트들을 자동으로 검출할 수도 있고, 이들 코드 포인트들에서 다이버전스 배리어 명령들을 삽입할 수도 있다.

[0022] 그 후, CPU 는 실행을 위하여 GPU 에 다이버전스 배리어 명령들을 포함하는 커널의 코드를 송신한다. 그 후, GPU 는 이것이 다이버전스 배리어 명령에 접할 때까지 커널 코드를 실행한다. 각각의 다이버전스 배리어 명령은 GPU 로 하여금 불 연산식을 평가하게 한다. GPU 가 불 연산식을 참으로서 평가하면, GPU 는 와프의 실행을 중지한다. GPU 는 다른 와프로 스위칭하여 다른 와프의 실행을 시작한다. GPU 는 (다이버전스 배리어 명령을 실행하는 것으로 인하여) 커널의 와프들 모두가 실행을 끝내거나 중지될 때까지 와프들을 실

행하는 프로세스를 계속한다. 와프들 모두가 실행을 끝내거나 중지되면, GPU 는 다이버전스 배리어 명령을 실행하는 결과로서, 현재 중지되는 와프들 중에서 다이버전스를 제거하도록 시도한다.

[0023] GPU 가 다이버전스 배리어 명령을 실행하고, 와프의 실행을 중지할 때, GPU 는 다이버전스 배리어 명령을 실행하였던 것으로 인하여 현재 중지되는 와프들의 큐 내에 와프를 삽입한다. 큐 내에 위치될 때, GPU 는 삽입 분류를 이용하여 각각의 와프에서의 액티브 스레드들의 수에 기초하여 큐에서의 와프들을 분류하고, 삽입 분류를 이용하여 큐에 중지된 와프들의 각각을 분류시킨다. 모든 와프들이 중지되고 큐에서 분류된 후 (또는 마무리된 후), GPU 는 커넬을 실행하는 와프들의 스레드들 간의 다이버전스를 제거하도록 시도한다. 와프들의 스레드들 간에 다이버전스를 제거하는 프로세스는 "스레드 압축"이라 지칭된다.

[0024] 스레드 압축 동안에, GPU 는 더 많은 비액티브 스레드들을 갖는 현재 액티브 와프들을 더 많은 액티브 스레드들을 갖는 와프들로부터의 비액티브 스레드들과 교환함으로서 다이버전스가 거의 없거나 전혀 없는 스레들을 갖는 와프들을 형성하도록 시도한다. GPU 는 상이한 와프들로부터의 스레드들을 교환할 때 교환되는 데이터 양을 최소화하기 위하여 와프 분류 큐를 이용한다. 새로운 와프들의 형성을 가져오는 GPU 스레드 압축 동안에, GPU 는 모든 액티브 스레드들을 갖는 와프가 형성되자마자 각각의 새로운 와프의 실행을 계속할 수도 있다. 이 방식으로, 다이버전스 배리어 명령을 지원하도록 구성되는 GPU 는 와프 스레드 다이버전스를 감소시켜 GPU 성능을 개선시킬 수도 있다.

[0025] 도 1 은 본 개시물의 기법들에 따라 GPU 다이버전스 배리어 명령의 실행을 지원할 수도 있는 예시적인 컴퓨팅 디바이스를 예시하는 블록도이다. 도 1 은 컴퓨팅 디바이스 (2) 를 포함한다. 컴퓨팅 디바이스 (2) 는 예를 들어, 퍼스널 컴퓨터, 데스크톱 컴퓨터, 랩톱 컴퓨터, 테블릿 컴퓨터, 컴퓨터 워크스테이션, 테블릿 컴퓨팅 디바이스, 비디오 게임 플랫폼 또는 콘솔, 무선 통신 디바이스 (이를 테면, 예를 들어, 모바일 전화기, 셀룰라 전화기, 위성 전화기, 및/또는 모바일 전화기 핸드셋), 랜드라인 전화기, 인터넷 전화기, 핸드헬드 디바이스, 이를 테면, 포터블 비디오 게임 디바이스 또는 개인 휴대정보 단말기 (PDA), 퍼스널 뮤직 플레이어, 비디오 플레이어, 디스플레이 디바이스, 텔레비전, 텔레비전 셋톱 박스, 서버, 중간 네트워크 디바이스, 메인프레임 컴퓨터, 또는 그래픽 데이터를 프로세싱 및/또는 디스플레이하는 임의의 다른 유형의 디바이스를 포함할 수도 있다.

[0026] 도 1 의 예에 예시되는 바와 같이, 컴퓨팅 디바이스 (2) 는 CPU (16), 시스템 메모리 (14), 그래픽 프로세싱 유닛 (GPU)(12) 및 컴파일러/드라이버 (18) 를 포함한다. CPU (16) 는 여러 유형들의 애플리케이션들을 실행할 수도 있다. 애플리케이션들의 예들은 웹브라우저들, 이메일 애플리케이션들, 스프레드시트들, 비디오게임들, 또는 디스플레이의 뷰 가능한 오브젝트들을 생성하는 다른 애플리케이션들을 포함한다. 하나 이상의 애플리케이션들의 실행을 위한 명령들은 시스템 메모리 (14) 내에 저장될 수도 있다.

[0027] CPU (16) 는 또한 컴파일러/드라이버 (18) 를 실행할 수도 있다. 컴파일러/드라이버 (18) 는 GPU (12) 의 명령을 제어하는 드라이버 및/또는 컴파일러를 포함할 수도 있다. 컴파일러/드라이버 (18) 는 특정 그래픽 애플리케이션 프로그래밍 인터페이스 (API) 에서 기록되는 코드와 같은 프로그램 코드를 취할 수도 있고 이 코드를 커넬 (20) 로 변환할 수도 있다. 커넬 (20) 은 GPU (12) 가 실행가능한 네이티브 코드, 예를 들어, 바이너리 명령들로 구성된다. 컴파일러/드라이버 (18) 는 또한 GPU (12) 의 런타임 실행을 관리할 수도 있다. 아래 보다 자세하게 설명될 바와 같이, 컴파일러/드라이버 (18) 는 본 개시물의 기술들에 따라 런타임에서 다이버전스 배리어 명령들을 커넬 (20) 내에 삽입할 수도 있다. CPU (16) 는 추가의 처리를 위하여 GPU (12) 에 커넬 (20) 을 송신할 수도 있다.

[0028] GPU (12) 는 그래픽 데이터를 프로세싱하기 위하여 매우 적합화된 대용량 병렬 프로세싱을 허용하는 직렬화된 하드웨어일 수도 있다. 이 방식으로, CPU (16) 는 GPU (12) 에 의해 보다 잘 핸들링되는 그래픽 프로세싱을 오프로드한다. CPU (16) 는 특정 애플리케이션 프로세싱 인터페이스 (API) 에 따라, GPU (12) 와 통신할 수도 있다. 이러한 API 의 예들은 Microsoft® 사에 의한 DirectX® API 및 Khronos 그룹에 의한 OpenGL® 을 포함하고 있지만, 그러나, 본 개시물의 양태들은 DirectX 및 OpenGL API들로 제한되지 않으며, 현재 개발되고 있거나 또는 향후 개발될 다른 유형들의 API들로 확장될 수도 있다.

[0029] GPU (12) 가 CPU (16) 로부터 그래픽 데이터를 수신하기 위한 방식을 정의하는 것에 더하여, API들은 GPU (12) 가 구현하고자 하는 특정 그래픽 프로세싱 파이프라인을 정의할 수도 있다. GPU (12) 는 도 1 에서 Direct3D 11 API 에 의해 정의되는 그래픽 프로세싱 파이프라인을 예시한다. 보다 자세하게 설명될 바와 같이, 도 2 는 OpenGL 4.x API 의 그래픽 프로세싱 파이프라인을 예시한다.

[0030] CPU (16) 및 GPU (12) 의 예들은 디지털 신호 프로세서 (DSP), 범용 마이크로프로세서, 응용 주문형 집적 회로 (ASIC), 필드 프로그래밍가능 로직 어레이 (FPGA), 또는 다른 등가의 통합된 또는 별개의 논리 회로를 포함하지만, 이들에 제한되는 것은 아니다. 일부 예들에서, GPU (12) 는 그래픽 프로세싱에 적절한 대용량 병렬 프로세싱 능력들을 GPU (12) 에 제공하는 통합된 및/또는 별개의 논리 회로를 포함하는 특수화된 하드웨어일 수도 있다. 일부 경우들에서, GPU (12) 는 또한 범용 프로세싱을 포함할 수도 있고, 범용 GPU (12; GPGPU) 로서 지칭될 수도 있다. 본 개시물에 설명된 기법들은 GPU (12) 가 GPGPU 인 예들에 적용가능할 수도 있다.

[0031] 시스템 메모리 (14) 는 하나 이상의 컴퓨터 판독가능 저장 매체를 포함할 수도 있다. 시스템 메모리 (14) 의 예들은 랜덤 액세스 메모리 (RAM), 판독 전용 메모리 (ROM), 전기적 소거가능 프로그래밍가능 판독전용 메모리 (EEPROM), 플래시 메모리, 또는 명령들 및/또는 데이터 구조물의 형태로 원하는 프로그램 코드를 운반 또는 저장하는데 이용될 수 있고 컴퓨터 또는 프로세서에 의해 액세스될 수 있는 임의의 다른 매체를 포함하지만, 이들에 제한되지 않는다.

[0032] 일부 양태들에서, 시스템 메모리 (14) 는 본 개시물에서, CPU (16) 및/또는 GPU (12) 로 하여금 CPU (16) 및 GPU (12) 에 기인하는 함수들을 수행하게 하는 명령들을 포함할 수도 있다. 따라서, 시스템 메모리 (14) 는 하나 이상의 프로세서들, 예를 들어, CPU (16) 및 GPU (12) 로 하여금, 여러 기능들을 수행하게 하는 명령들을 포함하는 컴퓨터 판독가능 저장 매체일 수도 있다.

[0033] 시스템 메모리 (14) 는 일부 예들에서, 비일시적 저장 매체로서 고려될 수도 있다. 용어 "비일시적인" 은 저장 매체가 반송파 또는 전파 신호로 구현되지 않음을 표시할 수도 있다. 그러나, 용어 "비일시적인" 은 시스템 메모리 (14) 가 비가동성임을 의미하는 것으로 해석되어서는 안된다. 일 예로서, 시스템 메모리 (14) 는 디바이스 (10) 로부터 제거되어, 다른 디바이스로 이동될 수도 있다. 다른 예로서, 시스템 메모리 (14) 와 실질적으로 유사한 시스템 메모리는 디바이스 (10) 에 삽입될 수도 있다. 특정 예들에서, 비일시적 저장 매체는 시간에 따라 (예를 들어, RAM 에서) 변할 수 있는 데이터를 저장할 수도 있다.

[0034] CPU (16) 는 또한 GPGPU 애플리케이션들에 대한 커맨드들 및 데이터, 예를 들어, 레이 추적 애플리케이션용의 커맨드 및 장면 데이터, 물리적 시뮬레이션, 또는 임의의 유형의 GPGPU 커널에 대한 데이터를 생성할 수도 있다. GPGPU 애플리케이션들, 예를 들어, 커널 (20) 은 또한 그래픽스 API, 이를 테면, DirectX, 또는 OpenGL 을 이용하여, 또는 보다 범용의 컴퓨트 API, 이를 테면, OpenCL (Open Compute Language), 또는 OpenCompute, 또는 DirectCompute 를 이용하여 컴파일될 수도 있다. CPU (16) 는 프로세싱을 위하여 커맨드 버퍼에 커널 (20) 에 대한 데이터를 송신할 수도 있다. 여러 예들에서, 커맨드 버퍼는 시스템 메모리 (14) 의 부분 또는 GPU (12) 의 부분일 수도 있다. 일부 예들에서, CPU (16) 는 커널 (20) 의 데이터 및 커맨드들을 GPU (12) 에 대해 송신하여 특수 목적 버스, 이를 테면, PCI-Express 버스 또는 다른 범용 시리얼 또는 패럴렐 버스를 통하여 프로세싱할 수도 있다.

[0035] 커맨드 버퍼에 커널 (20) 의 저장된 동작들을 수행하기 위하여, GPU (12) 는 그래픽 프로세싱 파이프라인을 구현할 수도 있다. 그래픽 프로세싱 파이프라인은 GPU (12) 상에서 실행할 소프트웨어 또는 펌웨어에 의해 정의되는 바와 같이 수행하는 것, 및 매우 특수한 기능들을 수행하도록 하드와이어된 고정된 기능 유닛들에 의해 기능들을 수행하는 것을 포함한다. GPU (12) 상에서 실행하는 소프트웨어 또는 펌웨어는 쉐이더들, 예를 들어, 쉐이더 (22) 로서 지칭될 수도 있다. 쉐이더들 (22) 은 GPU (12) 의 하나 이상의 프로세싱 엘리먼트들 (또한, "쉐이더 코어들" 또는 "PE들" 로서 지칭됨) 을 실행할 수도 있다. 사용자는 임의의 다른 프로세서에서와 같이 임의의 모든 방식으로 원하는 태스크들을 실행하기 위하여 쉐이더들을 프로그래밍할 수 있기 때문에, 쉐이더들 (22) 은 기능적 유연성을 사용자들에게 제공한다. 그러나, 고정된 기능 유닛들은 고정된 기능 유닛들이 태스크들을 수행하는 방식에 대해 하드와이어된다. 따라서, 고정된 기능 유닛들은 더 많은 기능 유연성을 제공하지 못할 수도 있다. 본 개시물의 기법들은 GPU 쉐이더들 (22) 상에서 커널, 이를 테면, 커널 (20) 의 실행에 대하여 교시된다.

[0036] CPU (16) 가 커맨드 버퍼에, 커널을 실행하거나 또는 그래픽 장면을 렌더링하는 것과 연관되는 데이터 및/또는 커맨드들을 송신하면, GPU (12) 는 GPU (12) 의 그래픽스 파이프라인을 통하여 커맨드들의 실행을 시작한다. GPU (12) 의 스케줄러 (24) 는 커널과 연관된 작업의 기본 유닛을 수행하는 스레드들을 생성한다. 스케줄러 (24) 는 쉐이더들 (22) 의 특정 프로세싱 엘리먼트에 스레드들을 배정한다. 스케줄러 (24) 는 또한 스레드들을 실행을 위한 와프들로 그룹화하고 그 와프들의 실행을 시작한다.

[0037] 위에 논의된 바와 같이, 상이한 스레드들이 플로우 제어 명령을 실행한 결과로서, 상이한 명령들로 점핑하면, 와프의 스레드들은 다이버즈한다. 다이버전스한 와프의 경우, 스케줄러는 스레드들의 각각의 세트를 직렬로

실행한다. 즉, GPU (12) 는 더 이상 스레드들의 와프 모두를 병렬로 실행하지 않고 그룹들로 직렬로 실행하는데 이는 GPU 성능을 손상시킨다.

[0038] 와프들이 다이버전스할 때 GPU 성능을 개선시키기 위하여, 프로그래머 또는 컴파일러/드라이버 (18) 는 커널 (20) 내에 다이버전스 배리어 명령을 삽입할 수도 있다. 다이버전스 배리어는 GPU (12) 가 런타임에서 평가하는 불 연산식과 연관된다. 불 연산식은 참 또는 거짓을 평가하는 연산식이다. 불 연산식은 여러 예들에서 산술 연산자, 비트와이즈 논리 연산자들, 및/또는 논리 연산자들을 포함할 수도 있다. 불 연산식에 기초하여 다이버전스 배리어 명령을 실행할지의 여부를 결정함으로써, 불 연산식은 GPU 가 다이버전스 배리어를 실행할 때를 제어하는데 있어서 유연성을 제공한다. 불 연산식 평가들은 다이버전스 배리어 명령이 통상의 배리어 명령과는 상이한 일 방식이다. 즉, GPU 가 배리어 명령을 실행할 때 와프들의 실행을 항상 정지시키는 통상의 다이버전스 배리어 명령을 실행하는 것과 달리, 다이버전스 배리어들이 불 조건과 연관되고 다이버전스 배리어들이 종종, 불 연산식과 또한 연관되어 있는 제어 플로우 블록들에 위치되기 때문에 와프들은 각각의 다이버전스 배리어에서 정지할 필요가 없다. 다이버전스 배리어 명령에 대한 의사 코드의 일 예는 다음과 같다;

[0039] divergence_barrier(Boolean expression);

[0040] 다이버전스 배리어 명령은 GPU 로 하여금, 다이버전스 배리어 명령과 연관된 불 연산식이 다이버전스 배리어 명령에 도달하는 와프의 각각에서의 적어도 하나의 스레드에 대하여 참인지의 여부를 결정하게 한다. 적어도 하나의 스레드에 대하여 조건이 참이면, GPU (12) 는 복수의 와프들의 각각의 실행을 중지하고, 액티브 스레드들의 수에 기초하여 와프들을 분류한 다음, 새로운 액티브/비액티브 와프들을 형성하기 위해 비액티브 스레드들을 액티브 스레드들과 교환한다. GPU (12) 는 모든 비액티브 스레드들을 갖는 비액티브 와프들이 생성되지 않을 때까지 비액티브 스레드들을 액티브 스레드들과 교환하는 것을 계속한다. 어떠한 비액티브 와프들도 생성될 수 없다면, GPU (12) 는 와프들의 실행을 재개한다. GPU (12) 가 모든 액티브 스레드들을 갖는 와프를 형성하면, GPU (12) 는 또한, 큐로부터 즉시 릴리즈되고 그 와프의 실행을 시작할 수도 있다.

[0041] 본 개시물의 기법들에 따른 일 예로서, 컴퓨팅 디바이스 (2) 의 GPU (12) 는 복수의 와프들의 각각의 와프에 대하여 불 연산식이 각각의 와프의 대응하는 스레드에 대하여 참인지의 여부를 결정하는 단계; 연산식이 참인 대응하는 스레드를 갖는 각각의 와프의 실행을 중지하는 단계, 및 연산식이 참인 복수의 와프들 각각에 대한 액티브 스레드들의 수를 결정하는 단계를 포함하는 방법을 수행하도록 구성될 수도 있다. 본 방법은 복수의 와프들 각각에서의 액티브 스레드들의 수에 기초하여 연산식이 참인 복수의 와프들을 분류하는 단계, 복수의 와프들 중 제 1 와프의 액티브 스레드의 스레드 데이터를, 복수의 와프들 중 제 2 와프의 액티브 스레드의 스레드 데이터와 교환하는 단계, 및 연산식이 참인 복수의 와프들 중 적어도 하나의 와프의 실행을 재개하는 단계를 더 포함할 수도 있다.

[0042] 도 2 는 본 개시물의 기법들에 따라 복수의 프로세싱 엘리먼트 상에서 실행하는 와프를 예시하는 블록도이다. 도 2 는 복수의 프로세싱 엘리먼트들 (42A-42N; PE들 (42)) 상에서 실행하는 스레드 와프 (40) 를 예시한다. PE들 (42) 은 하나 이상의 쉐이더들 (22)(도 1) 의 일부일 수도 있다. 스레드 와프, 이를 테면, 와프 (40) 는 GPU 스케줄러 (24) 가 실행을 위하여 복수의 프로세싱 엘리먼트들, 예를 들어, PE들 (42) 에 배정할 수도 있는 스레드들의 그룹을 포함할 수도 있다. 도 2 의 각각의 PE 는 특정 시간에 다수의 데이터 값들에 대하여, 벡터 명령과 같은 단일 명령을 실행할 수 있는 단일 명령 다중 데이터 (SIMD) 유닛을 포함할 수도 있다. PE들 (42) 은 또한 단일 부동 소수점 값에 대한 단일의 연산과 같이 단일의 데이터 값에 대한 단일의 명령의 실행을 지원할 수도 있다.

[0043] 와프 (40) 는 또한 실행을 위하여 GPU (12) 의 스케줄러가 PE들 (42) 을 배정하는 명령들 (44) 을 포함한다. 일부 예들에서, 명령들 (44) 은 커맨드 버퍼에 저장될 수도 있다. 명령들 (44) 은 각각의 PE 가 실행하도록 구성되는 커널의 명령들의 세트를 포함할 수도 있다. 프로그램 카운터 (PC)(50) 는 PE들 (42) 중 하나 이상이 실행할 현재 명령을 나타낸다. 명령이 PE들 (42) 상에서 실행하는 것을 끝낸 후, PC (50) 의 값은 커널 (20) 의 다음 명령의 어드레스에 대해 증분될 수도 있다. 와프 (40) 는 또한 레지스터 (46) 를 포함한다. 레지스터들 (46A-46N)(레지스터 (46)) 는 다수의 데이터 값들 또는 단일 값을 유지가능한 범용 레지스터들일 수도 있다. 레지스터들 (46) 은 "뱅킹될" 수도 있다, 즉, 특정 PE 에 데이터를 로딩하여 저장할 수도 있다. 일 예로서, 레지스터 (46A) 는 PE (42A) 에 대해 데이터를 저장하도록 제한될 수도 있고 다른 PE들에 대해 데이터를 로딩하거나 저장하지 않을 수도 있다. 레지스터들 (46) 각각은 PE들 (42) 중 하나에 및/또는 하나로부터, PE들 (42) 이 이후에 프로세싱할 수도 있는 데이터를 공급할 수도 있다. 와프 (40) 는

또한 와프 컨텍스트 데이터 (48) 를 포함할 수도 있다. 와프 컨텍스트 데이터 (48) 는 와프 (40) 의 상이한 스레드들 간에 공통이거나 또는 공유되는 데이터를 포함할 수도 있다. 일 예로서, 컨텍스트 데이터 (48) 는 와프 (40) 의 PE들 (42) 상에서 실행하는 각각의 스레드에 대한 데이터를 포함할 수도 있는 특정 레지스터의 데이터를 포함할 수도 있다.

[0044] 와프 (40), PE들 (42), 명령 (44), 레지스터 (46), 컨텍스트 (48), 및 PC (50) 는 GPU (12) 의 쉐이더들 (22) 의 코어 또는 코어의 부분을 포함할 수도 있다. 여러 예들에서, 와프 (40) 는 GPU (12) 의 그래픽 파이프라인의 부분일 수도 있는 지오메트리 쉐이더, 픽셀 쉐이더, 및/또는 버텍스 쉐이더와 같은 쉐이더를 포함할 수도 있다. 일부 예들에서, GPU (12) 는 추가적인 프로세싱을 위하여 그래픽 파이프라인의 다른 스테이지 내에 와프에 의해 생성된 결과들을 피드할 수도 있다.

[0045] 와프 (40) 상의 커널의 실행 동안에, PE들 (42) 중 하나 이상은 PC (50) 에 의해 표시되는 어드레스에 위치된 명령들 (44) 중 하나를 실행한다. 명령의 실행 동안에, PE들 (42) 은 레지스터들 (46) 로부터 하나 이상의 데이터 값들을 판독할 수도 있다. PE들 (42) 은 데이터 값들에 대한 하나 이상의 연산들을 수행할 수도 있고, 레지스터들 (46) 에 다시 새로운 값을 저장할 수도 있다. PE들 (42) 은 플로우 제어 명령들, 이를 테면, 브랜치들, 점프들, 진행 (goto) 등을 실행할 수도 있다. 플로우 제어 명령들은 하나의 PE, 예를 들어, PE (42B) 외에, PE (42A) 로 하여금 명령들 (44) 중 상이한 명령으로 점프하게 할 수도 있다, 즉, PE들을 실행하는 스레드들이 플로우 제어의 상이한 평가들로 인하여 다이버전스하게 될 수도 있다. 그러나, 단일의 PC (50) 가 존재하기 때문에, PE들 (42) 은 주어진 시간에 특정 하나에서 PC (50) 에 의해 표시되는 명령들 (44) 중 하나만을 실행할 수도 있다.

[0046] 와프의 스레드들이 다이버즈하면, PE들 (42) 은 여전히, 특정 시간에 PC (50) 의 값에 의해 표시되는 하나의 명령만을 실행할 수도 있다. 다이버전트 실행을 지원하기 위하여, 와프 (40) 는 PE들 (42) 중 어느 것이 PC (50) 의 어드레스에서 명령을 실행해야 하는지를 나타내는 상태, 이를 테면, 비트마스크를 유지한다. 일 예로서, PE (42A 및 42B) 는 "if-else" 스테이트먼트의 상이한 브랜치들을 취하는 것으로부터 야기되는, 상이한 명령들을 실행하도록 스케줄링될 수도 있다. 이 예에서, PE (42A) 는 명령들 (44) 중 제 1 명령을 실행하고, 이후 시간에, PE (42B) 는 명령들 (44) 중 제 2 의 상이한 명령을 실행한다. PE (42A) 가 제 1 명령을 실행할 때, 와프 (40) 는 PE (42A) 가 명령의 실행 동안에 액티브 상태인 한편, PE (42B) 는 비액티브 상태임을 나타내도록 비트마스크를 설정한다. 그 후, PE (42A) 의 스레드가 실행을 끝내거나 또는 다이버전스 배리어 명령을 중지하고 스레드의 실행을 중지할 때까지 PE (42A) 는 명령들 (44) 을 실행하는 것을 계속한다. 일단 PE (42A) 가 실행을 끝내면, 와프 (40) 는 PE (42B) 만이 액티브 상태임을 나타내도록 비트마스크를 변경하고, PC (50) 의 값을 PE (42B) 가 실행해야 하는 명령의 어드레스로 변경한 다음, PE (42B) 는 스레드가 실행을 중지하거나 또는 끝낼 때까지, PC (50) 에 의해 특정된 명령들을 실행한다.

[0047] 위에 언급된 바와 같이, 본 개시물의 기법들은 실행될 때, 다수의 와프들의 스레드들, 이를 테면, 와프 (40) 가 다이버즈할 때, GPU (12) 의 성능을 개선시킬 수도 있는 다이버전스 배리어 명령을 포함한다. 다이버전스 배리어 명령은 애플리케이션 프로그래밍 인터페이스 (API), 이를 테면, DirectX 11 API, OpenGL API, OpenCL, 및/또는 DirectCompute 등의 부분을 포함할 수도 있다. 이러한 API 에 기록된 프로그램은 GPU (12) 로 하여금 다이버전스 배리어 명령을 실행하게 하는 커널 (20) 내에, 다비어전스 배리어 함수에 대한 호출을 삽입할 수도 있다.

[0048] 컴파일러/드라이버 (18) 또는 오퍼레이팅 시스템은 또한, 커널 (20) 의 코드 내에 다이버전스 배리어 명령에 대한 호출들을 삽입할 수도 있다. 여러 예들에서, 사용자는 컴파일러/드라이버 (18) 를 이용하여 커널 (20) 을 컴파일할 수도 있다. 컴파일화 동안에, 컴파일러/드라이버 (18) 는 커널 (20) 을 분석하고 로케이션, 다이버전스가 발생할 가능성이 있는 프로그램, 및 성능에 상당히 영향을 주는 로케이션 중 적어도 하나를 결정할 수도 있고 이들 로케이션들 중 적어도 하나에서 다이버전스 배리어 명령들을 삽입할 수도 있다. 컴파일러/드라이버 (18) 는 스레드 다이버전스가 발생할 가능성이 있는 로케이션, 및 성능에 상당히 영향을 주는 로케이션 중 적어도 하나에서 런타임 (또한, "바인드 타임"으로 지칭됨) 으로 커널 (20) 의 명령들 내에 다이버전스 배리어 명령들을 삽입할 수도 있다.

[0049] 다이버즈할 가능성이 있는 코드의 일 예는, 아래 포함된, 레이 추적 (raytracing) 애플리케이션의 코드일 수도 있다. 이 예에서, 예를 들어, (컴파일러 또는 사용자에 의해) 다이버전스 배리어 명령이 삽입되어, 다음 레이 추적 의사코드를 실행할 때 다이버전스가 감소된다.

```
i = 0;
While ( i < dynamic_limit) {   // dynamic_limit goes from 0 to 30
   divergence_barrier(i%10==0); //eliminate divergence after each 10 loops
   //traverse scene tree and do ray intersection calculation
}
```

[0050]

[0051] 위의 의사코드는 GPU (12) 의 다수의 스레드들 또는 와프들이 실행할 수도 있는 루프의 일 예이다. 각각의 스레드는 상이한 횟수로, 예를 들어, 레이추적 장면에서 레이가 만드는 바운스들의 수에 기초하여 루프를 실행할 수도 있다. 따라서, 일부 스레드들은 루프의 수회의 반복을 수행한 후 종료할 수도 있는 한편, 루프의 30 회의 반복만큼 루프의 실행을 계속할 수도 있다.

[0052] 이 예에서, GPU (12) 는 각각의 루프의 반복 동안에 다이버전스 배리어 명령을 실행한다. 다이버전스 배리어 명령은 GPU 가 루프의 각각의 반복에 의해 평가한 불 연산식을 포함한다. GPU (12) 는 불 연산식이 와프의 적어도 하나의 스레드에 대해 참으로 평가하면, 다이버전스 배리어 명령과 연관된 연산들만을, 예를 들어, 와프 분류 및 스레드 압축만을 실행한다. 이 예에서, 불 연산식 i%10 == 0 은 루프의 매 10회째 반복 동안에 참으로 평가한다. 불 연산식이 와프의 하나의 스레드에 대해 참일 때, GPU (12) 는 보다 액티브한 스레드들을 갖는 새로운 와프들을 형성하기 위해 상이한 와프들로부터의 스레드들을 교환할 수도 있으며, 이 프로세스는 "스레드 압축"으로서 지칭된다.

[0053] 하나의 와프 스레드의 다이버전스 배리어와 연관된 불 연산식이 참으로 평가하면, GPU (12) 는 그 스레드와 연관된 와프, 예를 들어, 와프 (40) 를 버퍼 또는 큐 내에 둔다. 와프가 큐 내에 놓이면, GPU (12) 는 와프 (40) 가 실행하는 것을 정지하게 하고, 큐에서 와프들을 분류한다.

[0054] 각각의 와프의 액티브 스레드들의 수에 기초하여 와프를 분류하는 것은 도 3 에 보다 자세하게 예시되어 있으며, GPU (12) 는 삽입 분류를 이용하여 각각의 와프에서의 액티브 스레드들의 수에 기초하여 와프들을 분류할 수도 있다. GPU (12) 는 보다 높게 액티브한 스레드들이 큐의 전방으로 분류되고 보다 낮게 액티브한 스레드들을 갖는 와프들을 큐의 후방에 있도록 와프들을 분류한다.

[0055] 모든 와프들이 큐 내에 추가되거나 또는 배리어에서 중단됨이 없이 완료된 후에, GPU (12) 는 큐 내에 와프들에 대하여 스레드 압축을 수행한다, 즉, 보다 많은 수의 액티브 스레들을 갖는 와프들로부터의 비액티브 스레드들을, 보다 적은 수의 스레드들을 갖는 와프들과 교환한다. GPU (12) 는 GPU (12) 가 "비액티브" 와프를 생성할 수 없을 때까지 보다 많은 수의 액티브 스레드들을, 보다 적은 수의 액티브 스레드들을 갖는 와프들과 교환하는 것을 계속한다. 비액티브 와프는 모든 비액티브 스레드들을 갖는 와프이다. GPU (12) 는 또한, 비액티브 스레드를 액티브 스레드와 교환할 때 필요에 따라 스레드 당 컨텍스트 데이터 (48) 를 교환할 수도 있다. 모든 액티브 스레드를 갖는 "완전 액티브 와프 (fully active warp)" 가 스레드들을 교환함으로써 생성되면, GPU (12) 는 큐로부터 모든 액티브 와프를 제거하고, 그 상태를 액티브로 설정하고, 현재 명령으로부터 모든 액티브 와프의 실행을 재개한다. GPU (12) 가 스레드 압축을 끝낸 후, 부분적 액티브 와프들 및 모든 비액티브 와프들을 포함하는 모든 와프들이 준비 상태 또는 액티브 상태로 설정된다. 부분적 액티브 와프들은 또한 현재 명령으로부터 재개된다. 모든 비액티브 스레드들은 현재 제어 플로우 블록의 끝으로 고속 진행 (fast forward) 할 수 있고, 어떠한 명령도 현재 제어 블록을 뒤따르지 않으면, 모든 비액티브 와프가 즉시 실행을 끝낼 수 있다. 와프들 간에 스레드들을 교환하는 프로세스는 도 4 에 대하여 보다 자세하게 예시된다.

[0056] 액티브 스레드와 비액티브 스레드를 교환하기 위하여, GPU (12) 는 일부 예들에서, 비액티브 스레드 및 액티브 스레드의 저장된 레지스터 데이터를 레지스터 교환 버퍼 (52) 에 저장할 수도 있다. 그 후, GPU (12) 는 이전의 비액티브 스레드의 레지스터 데이터를 이전의 액티브 스레드의 대응하는 레지스터에 저장한다. GPU (12) 는 또한 멀티플렉서 (54)("MUX (54)") 를 이용하여 이전의 비액티브 스레드의 대응하는 레지스터에 이전의 액티브 스레드의 레지스터 데이터를 저장한다. 보다 구체적으로, 각각의 스레드와 연관된 각각의 레지스터에 대하여, 멀티플렉서 (54)("MUX (54)") 는 비액티브 및 액티브 스레드들의 저장된 레지스터 값들 사이를 멀티플렉싱하고 교환될 와프들의 레지스터 파일들에 값들을 다시 저장한다. 교환 프로세스 동안에, DBS (50) 는 또한 제 1 및 제 2 와프들로부터의 스레드 당 컨텍스트 데이터 (48) 를 교환할 수도 있다. 일부 예들에서, GPU (12) 는 레지스터 데이터를 교환하기 위해 레지스터 교환 버퍼 (52) 를 이용하지 않을 수도 있다. 다만, GPU (12) 는 버퍼에 값들을 저장하기 보다는, 레지스터 값들을 동시에 교환할 수도 있다.

[0057] 일부 예들에서, 각각의 와프는 레지스터 포인터를 이용하여 "뱅크"로서 지칭되는, 특정 스레드와 연관된 레지스

터들 (46) 의 세트를 지칭할 수도 있다. GPU (12) 는 포인터들의 맵핑 테이블을 저장할 수도 있다. 테이블의 각각의 로우 또는 컬럼은 특정 와프에 대응할 수도 있으며, (테이블 레이아웃에 의존하여) 와프에 대응하는 로우 또는 컬럼 내의 각각의 엔트리는 레지스터들 (46) 내의 레지스터 뱅크에 특정 스레드를 맵핑하는 포인터 값을 저장할 수도 있다. GPU (12) 는 콘텍스트 데이터 (48) 에서 와프의 스레드들에 대한 레지스터 뱅크들에 대한 포인터들의 맵핑을 저장할 수도 있다. 일부 예들에서, 레제스터들 (46) 이 스레드당 레지스터 뱅크 포인터들에 의해 참조되면, GPU (12) 는 레지스터 교환 버퍼 (52) 및 mux (54) 를 이용하여 2 개의 스레드들의 대응하는 레지스터 값들의 각각을 교환하기 보다는, 2 개의 스레드들의 스레드 당 뱅크 포인터 값들을 단순히 교환함으로써 스레드당 레지스터 데이터를 교환할 수도 있다.

[0058] 일부 예들에서, 커넬을 실행하는 것은 예를 들어, GPU (12) 의 글로벌 메모리 및/또는 시스템 메모리 (14) 에 빈번하게 액세스할 수도 있거나 또는 많은 양의 액세스 시간 또는 레이턴스를 갖는 다른 연산들을 수행할 수도 있다. 이 경우에, 다이버전스 배리어 연산들을 포함하는 배리어 연산들은 너무 많은 와프들을 중지시킬 수도 있어, 이들 긴 레이턴시 연산들을 은닉시키지 못할 수도 있고, 실행 성능이 악화될 수도 있다. 긴 레이턴시 연산들을 갖는 커넬들의 실행을 고속화하기 위하여, GPU (12) 는 액티브 와프들 (액티브 와프 풀) 의 수가 특정 임계값에 도달하면 즉시 스레드 압축을 수행할 수도 있다.

[0059] 일부 커넬들은 "통상의" 배리어 연산들 및 다이버전스 배리어 연산들의 혼합을 포함할 수도 있다. 통상의 배리어 연산들은 배리어에 도달하는 모든 와프들로 하여금 중지하게 하고, 다이버전스 배리어들과 달리, GPU (12) 가 런타임에서 평가하는 불 조건과 연관되지 않는다. 통상의 다이버전스 배리어 연산들은 GPU (12) 로 하여금, 스레드 분류 및 스레드 압축을 수행하지 않게 한다. 통상의 배리어들과 다이버전스 배리어들의 혼합을 포함하는 커넬들에 대하여, 다이버전스 배리어 명령들은 통상의 배리어 연산들로 대체되어야 한다. 통상의 배리어 및 다이버전스 배리어 양쪽 모두의 혼합을 갖는 커넬에서, GPU (12) 는 통상의 배리어 연산을 실행하는 것으로 인하여 와프가 중지하는 것을 대기함이 없이 스레드 압축을 수행할 수도 있다.

[0060] 일부 커넬들은 또한 서브루틴들 호출을 포함할 수도 있다. 서브루틴 호출 동안에, GPU 는 스레드 데이터를, 호출된 서브루틴과 연관된 상이한 호출 스택들을 갖는 와프와 교환할 수도 있다. 서브루틴 호출은 다이버전스 배리어 연산들이 이러한 호출 내에 포함될 때 문제가 될 수도 있다. 예를 들어, 제 1 와프의 스레드는 커넬의 제 1 라인, 예를 들어, 라인 (10) 에서 서브루틴을 호출할 수도 있다. 제 2 와프는 커넬의 이후의 실행 포인트, 예를 들어, 라인 (20) 에서 동일한 서브루틴을 호출할 수도 있다. 서브루틴은 다이버전스 배리어 명령을 포함한다.

[0061] 명령들 및/또는 다른 팩터들을 개재한 실행으로 인하여, 제 1 와프 및 제 2 와프가 서브루틴 내의 다이버전스 배리어 명령을 실행할 때, 제 1 와프 및 제 2 와프의 스택들은 서로 상이할 수도 있다. 서브루틴들 내에 다이버전스 배리어들을 갖는 문제에 대한 일 예시적인 솔루션에서, GPU (12) 는 서브루틴들 내부에서 전체적으로 다이버전스 배리어들을 갖는 것을 금지시킬 수도 있다. 다른 예시적인 솔루션에서, GPU (12) 는 서브루틴 호출들 내에서 다이버전스 배리어 명령들을 실행할 때 다이버전스 배리어 명령들을 갖는 서브루틴들을 실행하는 와프들이 동일한 스택들을 갖는 것을 보장하기 위한 노직을 구현할 수도 있다.

[0062] 도 3 은 본 개시물의 기법들에 따라 각각의 와프 내에서 액티브 스레드들의 수에 기초하여 와프들을 분류하는 것을 예시하는 개념도이다. 도 3 의 예는 미분류된 와프들 (80) 의 수를 예시한다. GPU (12) 는 위에 설명된 바와 같이, 참과 동일한, 다이버전스 배리어 명령과 연관된 불 연산식을 평가하는 것에 응답하여 미분류 와프들 (80) 을 분류한다. 도 3 의 예에서, 미분류 와프들 (80) 은 와프들 (82, 84, 86, 88, 90, 92, 94, 96, 및 98) 을 포함한다. 미분류 와프들 (80) 에서, 액티브 와프들은 대각선 해싱으로서 예시된다. 비액티브 와프 스레드들은 해싱이 없이 예시된다.

[0063] GPU (12) 는 각각의 와프에서의 액티브 스레드들의 수에 기초하여, 미분류된 와프들 (82) 을 분류한다. 결과적으로 분류된 와프들은 도 3 에서, 분류된 와프들 (100) 로서 예시되어 있다. 미분류된 와프들 (80) 중, 와프 (82) 는 가장 액티브한 스레드들 (완전히 액티브) 이고, 그 뒤를 이어서 순서대로 와프 (90), 와프 (88), 와프 (94), 와프 (84), 와프 (98), 와프 (92), 와프 (86), 및 와프 (96)(완전히 비액티브) 로 이어진다. 도 3 에 예시되는 바와 같이, GPU (12) 는 삽입 분류를 이용하여 미분류된 와프들 (80) 을 분류한다. 각각의 와프에서의 액티브 스레드들의 수에 기초한 삽입 분류의 결과는 분류된 와프 (100) 로서 도 3 에 예시되어 있다. 여러 예들에서, GPU (12) 는 큐에서의 미분류된 와프들 (80) 을 큐에 저장할 수도 있고, 그 후, 큐에서 가동할 준비에 있는 와프들을 분류하여, 그 결과, 분류된 와프들 (100) 이 큐가 된다. 여러 예들에서, 큐는 포인터들의 링크된 리스트로서 구현될 수도 있다. 각각의 포인터는 특정 와프를 적시할 수도 있다.

링크된 리스트를 분류하기 위하여, GPU (12) 는 링크된 리스트에서 와프들과 연관된 포인터들을 교환할 수도 있다.

[0064] 도 4 는 한 와프로부터의 액티브 스레드들을 다른 와프들로부터 비액티브 스레드들과 교환하기 위한 기법들을 예시하는 개념도이다. 도 4 의 예에서, GPU (12) 는 이전에, 미분류된 와프들 (80) 을 분류된 와프들 (100) 로 분류하였다. GPU (12) 는 비액티브 스레드들을 분류된 와프들 (100) 의 액티브 스레드들과 교환한다. GPU (12) 는 "비액티브 와프들" 즉, 모든 비액티브 스레드들을 갖는 와프들이 더 이상 생성될 수 없을 때까지 비액티브 스레드들을 액티브 스레드들과 교환한다. 비액티브 스레드들을 액티브 스레드들과 교환하는 프로세스는 "스레드 압축" 으로 지칭된다. 비액티브 스레드들을 분류된 와프들 (100) 의 액티브 스레드들과 교환한 결과는 압축된 와프들 (120) 로서 예시된다.

[0065] GPU (12) 는 2 개의 와프들에서의 액티브 및 비액티브 스레드들의 수에 기초하여 비액티브 스레드들을 액티브 스레드들과 교환한다. 도 4 의 예에서, GPU (12) 는 보다 많은 액티브 스레드들을 갖는 와프들로부터의 스레드들을 보다 적은 수의 액티브 스레드들을 갖는 와프들로부터의 스레드들과 교환한다. 도 4 에서, GPU (12) 는 비액티브 스레드를 갖는 최좌측 와프의 스레드들을 액티브 스레드를 갖는 최우측 와프의 스레드와 교환한다. GPU (12) 는 비액티브 와프들이 더 이상 생성될 수 없을 때까지 상이한 와프들로부터의 스레드들을 뒤집어서 교환하는 것, 즉, 보다 많은 액티브 스레드들을 갖는 와프들로부터의 비액티브 스레드들을 보다 많은 비액티브 스레드들을 갖는 와프들로부터의 액티브 스레드들과 교환하는 것을 계속한다. 스케줄러 (24) 는 그 때 여전히 큐에서 유지되는 임의의 그리고 모든 와프들의 실행을 재개한다. 추가로, 큐의 헤드에서의 와프가 모든 액티브 스레드들을 포함할 때마다, 스케줄러 (24) 는 큐의 헤드에 위치된 모든 액티브 스레드들을 갖는 와프를 릴리즈하고 그 와프의 실행을 시작한다.

[0066] 비액티브 스레드를 액티브 스레드와 교환함으로써, 본 개시물의 기법들은 모든 비액티브 스레드들을 갖는 와프들 뿐만 아니라 보다 많은 수의 액티브 스레드들을 갖는 와프들을 형성한다. 보다 많은 수의 액티브 스레드들을 갖는 와프들은 GPU (12) 의 사용률 및 스루풋을 증가시킨다. 어떠한 명령들도 현재 제어 블록을 후속하지 않는다면, 비액티브 와프들이 현재 제어 플로우 블록으로 "고속 진행"을 할 수도 있거나 또는 실행을 끝낼 수도 있기 때문에, 모든 비액티브 스레드들을 갖는 와프들은 또한 GPU (12) 의 스루풋을 증가시킬 수도 있다. 따라서, 일부 경우들에서, 모든 비액티브 스레드들을 갖는 와프들은 즉시 실행을 끝낼 수도 있다. 따라서, GPU (12) 는 이러한 비액티브 와프의 실행을 중단 또는 실행을 단축시킬 수도 있고, 비액티브 와프와 연관된 PE들을 이용하여, 스케줄러 (24) 가 이들 PE들 상에서 실행할 수 있는 것으로 결정한 상이한 와프를 실행할 수도 있다.

[0067] 도 5 는 본 개시물의 기법들에 따라 다수의 다어버전스 배리어 명령들을 핸들링하는 기법들을 예시하는 개념도이다. 다이버전스 배리어가 종종, 불 조건과 또한 연관된 제어 플로우 블록들에 위치되기 때문에, GPU 가 불 조건이 해당 와프의 임의의 스레드에 대하여 참이라고 평가하면 그 와프들은 다이버전스 캐리어가 위치되는 제어 플로우 브랜치에 진입할 수도 있거나, 또는 GPU 는 다이버전스 캐리어가 위치되는 제어 플로우 블록을 통과하여 실행을 계속하도록 허용할 수도 있다. 와프들은 배리어에 위치되어 있는 제어 플로우 브랜치에 이들 와프가 진입하지 못하면 또는 다이버전스 배리어의 불 조건이 와프에서의 모든 스레드들에 대하여 거짓이면 다이버전스를 통과할 수도 있다. 도 5 의 예에서, GPU (12) 는 "DB1," "DB2," 및 "DB3" 로서 지칭되는 다수의 다이버전스 배리어들을 포함하는 커널 (20) 을 실행한다. 일 예로서, 커널은 레이 추적 애플리케이션으로 이루어질 수도 있다. 도 5 는 커널을 통하여 8 개의 스레드들 (스레드들 (140, 142, 144, 146, 148, 150, 152, 및 154)) 의 진행을 예시한다. 스레드들 (140-154) 의 바들의 길이는 각각의 스레드가 다이버전스 배리어들 (DB1-DB3) 중 하나에 도달하는지 또는 커널의 실행을 완전히 끝냈는지 ("END") 의 여부를 나타낸다.

[0068] 도 5 의 예에서, DB1, DB2, 및 DB3 의 각각은 커널에서의 상이한 포인트들에 위치된다. DB1, DB2, 및 DB3 에 대응하는 3 개의 다이버전스 배리어들을 포함하는 샘플의 일 예시적인 의사코드는 아래에 포함된다.

```
i = 0;
while(i < dynamic_limit){   // dynamic_limit goes from 0 to 30
   divergence_barrier(i%10==0); //DB1 for each 10 loops
   i++;
   // do some work …
}
if (dynamic_codition) {
   divergence_barrier(true);  // DB2 for long control flow block
   // do some heavy work
}
else {
   divergence_barrier(true); // DB3 for long flow control block
   // do some heavy work
}
```

[0069]

[0070] 커넬 의사코드는 다수의 다이버전스 배리어 명령들 (DB1, DB2, 및 DB3) 을 포함한다. 다이버전스 배리어 명령들의 각각은 브랜치 스테이트먼트 또는 루프 스테이트먼트에서 발생한다. 커넬 (20) 을 실행하는 와프는 이들이 어느 제어 블로우 블록에 진입하는지 및 불 조건들, 예를 들어, 다이버전스 배리어 명령들과 연관된 배리어 조건들의 평가에 의존하여 제어 상이한 다이버전스 배리어들에 도달할 수도 있다. 커넬 (20) 을 실행하는 스레드들은 커넬 (20) 의 실행시 먼저 DB1 에 마주칠 수도 있고 그 다음 이어서 DB2 또는 DB3 에 마주칠 수도 있다.

[0071] GPU (12) 는 단일의 다이버전스 배리어 명령에 반대되어, 다수의 다이버전스 배리어 명령들이 커넬에 존재할 때, 유사하게 와프들을 분류하고 스레드 압축을 수행하는 것을 핸들링할 수도 있다. 구체적으로, 스케줄러 (24) 는 "압축 풀" 로서 지칭되는 것 내의 동일한 다이버전스 배리어에 도달하는 와프들을 함께 그룹화할 수도 있다. GPU (12) 는 동일한 다이버전스 배리어 명령에 도달하였던 압축 풀에서의 와프들의 스레드들을 압축할 수도 있다.

[0072] 보다 구체적으로, GPU (12) 는 각각의 와프와 함께 와프가 도달한 다이버전스 배리어와 연관된 프리픽스를 연관시킨다. 일 예로서, 제 1 다이버전스 배리어에 도달한 와프는 프리픽스 "1" 를 가질 수도 있고, 제 2 다이버전스 배리어에 도달한 와프들은 프리픽스 "2"에 도달할 수 있으며 이하 동일하게 이루어진다. 각각의 와프는 또한 그 와프에서의 액티브 스레드들의 수를 나타내는 제 2 수, 예를 들어, 서픽스를 배정받는다. 일 예로서, 와프가 3 개의 액티브 와프들을 갖는다면, 와프는 서픽스 "삼 (3)" 을 배정받는다.

[0073] 프리픽스 및 서픽스의 조합은 GPU (12) 가 와프 큐에서의 와프들을 분류하는데 이용하는 수를 형성한다. 일 예로서, 와프 큐에서 GPU (12) 가 분류하기 위한 3 개의 스레드들이 존재할 수도 있다. 제 1 와프는 다이버전스 배리어 "2" 에 도달하고, 네 (4) 개의 액티브 스레드들을 갖는다. GPU (12) 는 분류 목적을 위하여 수 "24" 를 제 1 와프에 배정할 수도 있다. 제 2 와프는 다이버전스 배리어 "1" 에 도달할 수도 있고, 하나 (1) 의 액티브 스레드들을 갖는다. GPU (12) 는 값 "11" 을 제 2 와프에 배정할 수도 있다. 제 3 와프는 다이버전스 배리어 "1" 에 도달할 수도 있고, 세개 (3) 의 액티브 스레드들을 갖는다. GPU (12) 는 와프에 대한 분류 값으로서 13 을 배정할 수도 있다. GPU (12) 는 각각의 와프의 값들에 의해 큐에서의 와프들을 분류한다. 분류의 결과는 (분류 값 11 을 갖는) 제 3 와프가 큐의 헤드에 있도록 하고, (분류 값 13 을 갖는) 제 2 와프가 큐에서 두번째에 있도록 하고, (분류 값 24 를 갖는) 제 1 와프가 큐의 꼬리에 있도록 한다. 분류 값들 11 및 13 을 갖는 와프들이 동일한 프리픽스 "1" 을 갖고 있기 때문에 GPU (12) 는 압축 그룹을 형성할 수도 있다.

[0074] GPU (12) 가 모든 와프들을 중지시키고 와프들을 큐에 삽입한 후 (또는 와프들이 배리어 상에서 중지되지 않으면 실행을 끝낸 후), GPU (12) 는 액티브 스레드들을 비액티브 스레드들과 교환함으로써 제 1 와프 그룹에서의 와프들에 대한 스레드 압축을 수행한다. 즉, GPU (12) 는 후속하는 다이버전스 배리어들에 대한 다이버전스를 제거하기 전에 제 1 다이버전스 배리어 상의 모든 다이버전스를 제거한다, 즉, GPU (12) 는 다이버전스 배리어들 (DB2, DB3 등) 상으로 이동하기 전에 다이버전스 배리어 (DB1) 상의 다이버전스를 제거한다.

[0075] 특정 다이버전스 배리어와 연관된 다이버전스를 제거하기 위해, GPU (12) 는 제 1 와프 그룹을 큐로부터 분리시

키고, 압축 풀을 형성하고, 풀에서의 와프들에 대해 압축을 수행한다. GPU (12) 가 압축을 실행할 때, GPU (12) 는 압축 풀로부터 와프들을 릴리즈하고, 실행시 그 와프들을 재개하여, GPU (12) 가 임의의 후속하는 다이버전스 배리어들에 도달할 때 다시 릴리즈된 와프들을 중지시킬 수도 있다. 한편, 나머지 와프들을 포함하는 큐는 임의의 다이버전스 배리어들에 대해 중지된 추가적인 와프들을 수신하는 것을 계속한다. GPU (12) 는 이들 와프들이 배리어들 (DB2, DB3) 에 또는 루프의 경우 심지어 배리어 (DB1) 에 다시 도달하면, 재개된 와프들을 중지시킬 수도 있다. GPU (12) 는 위에 설명된 바와 같이, 큐에 와프들을 추가하고, 큐에서 와프들을 다른 와프들과 분류한다.

[0076] 모든 이들 와프들이 중지되고 큐에 삽입될 때, 즉, 큐가 전체를 다시 얻을 때 GPU (12) 는 예를 들어, DB2 에 대한 것일 수도 있는 큐에서의 현재 제 1 그룹에 대하여 동일한 압축 프로세스를 반복한다. GPU (12) 가 이전 압축을 완료하고 이전 압축 풀로부터 모든 와프들을 릴리즈하기 전에, GPU (12) 는 큐에 모든 와프들을 갖는 것은 아니고 압축의 다른 어라운드를 시작할 수도 있음을 주지한다. 따라서, 연속하는 압축 프로세스들 사이에서 충돌이 존재하지 않는다. 모든 와프들이 큐에서의 오직 하나의 그룹만을 형성하는 동일한 배리어 상에서 중지되면, GPU (12) 는 큐로부터 중지된 와프들 모두를 분리시키고 큐를 빈 상태로 둘 수도 있다.

[0077] 큐의 전방에서의 배리어들, 예를 들어, DB1 에서 중지된 와프들이 후속하는 배리어들, 예를 들어, 이후에 있는 DB2/DB3 를 힛팅할 가능성이 있기 때문에, GPU (12) 는 후속하는 배리어들 (예를 들어, DB2, DB3 등) 에 대해 압축을 수행할 때 가능한 많은 다이버전트 와프들을 함께 압축 풀들로 그룹화할 수 있기 위하여, 제 1 와프 그룹만을 압축하는 기술을 이용할 수도 있다. 한번에 하나의 배리어를 압축함으로써, 이 기법은 후속하는 다이버전스 배리어들의 압축 동안에 압축 풀에서의 보다 많은 수의 압축을 가능하게 하여 스레드 압축의 효율성을 개선할 수도 있다.

[0078] 다수의 배리어들의 경우에, GPU (12) 는 위에 설명된 동일한 조건 하에서 그리고 동일한 방식으로, 보다 조기에, 즉, 큐가 완전하지 않을 때, 다이버전스 배리어들에 대한 압축을 수행하는 것을 시작할 수도 있다. 이들 조건들은 예를 들어, 커넬 프로그램이 통상의 배리어들을 포함하거나 또는 빈번하고 긴 레이턴스 동작들을 발생시키는 것을 포함할 수도 있다.

[0079] 다수의 다이버전스 배리어들이 커넬에 존재하고, 와프와 연관된 불 연산식이 적어도 하나의 와프 스레드에 대해 참으로 평가할 때, GPU (12) 는 와프를 큐 내에 위치시키고 프리픽스를 와프와 연관시킨다. 프리픽스는 와프가 도달하였던 특정 다이버전스 배리어를 나타낸다. 일 예로서, 스케줄러 (24) 는 프리픽스, 이를 테면, "1" 을 각각의 와프들 (140, 146, 148, 152, 및 154) 과 연관된 식별자에 첨부하여, 이들 와프들이 다이버전스 배리어 (DB1) 에 도달하였음을 나타낼 수도 있다. 스케줄러 (24) 는 유사한 프리픽스 (예를 들어, "2", "3") 를 와프들 (144 및 150) 에 추가하여, 이들 와프들이 각각 다이버전스 배리어들 (DB3 및 DB2) 에 도달하였음을 나타낼 수도 있다.

[0080] DBM (52) 은 큐에 와프들 (140, 142, 144, 146, 148, 150, 152, 및 154) 각각을 저장한다. 와프들 (140-154) 은 초기에 분류되지 않으며, 와프가 어느 다이버전스 배리어에 도달하는지에 기초하여 프리픽스들과 연관된다. DBM (52) 은 초기에, 스레드들 각각과 연관된 프리픽스에 기초하여 와프들 (140, 142, 144, 146, 148, 150, 152, 및 154) 을 분류하고, 프리픽스 수에 기초하여 압축 그룹들로 와프들을 함께 그룹화한다.

[0081] 가장 이른 다이버전스 배리어, 예를 들어, DB1 에 대응하는 프리픽스를 갖는 와프들의 그룹은 "압축 풀" 로서 지칭된다. 도 1 의 예에서, 압축 풀 (156) 은 와프들 (140, 146, 148, 152, 및 154) 을 포함하며, 이 와프들 모두는 다이버전스 배리어 (DB1) 에 도달하고, 이에 따라 동일한 프리픽스를 포함한다.

[0082] 위에 설명된 바와 같이, GPU (12) 는 도달된 다이버전스 배리어 수에 기초하여 유도되는 프리픽스, 및 각각의 와프에서의 액티브 스레드들의 수에 관련된 서픽스에 기초하여 압축 풀 (156) 의 와프들을 분류한다. GPU (12) 가 배리어들 상에서 모든 와프들 (실행을 끝낸 와프들을 제외함) 을 중지하고, 중지된 와프들을 큐 내에 삽입하고 큐에서의 와프들을 분류한 후, GPU (12) 는 큐에서의 최전방 배리어를 표현하는 제 1 와프 그룹을 큐로부터 분리하고, 이 그룹과 압축 풀을 형성한다. 그 후, GPU (12) 는 비액티브 와프들이 압축 풀의 와프들로부터 더 이상 생성될 수 없을 때까지, 더 많은 수의 액티브 스레드들을 갖는 압축 풀에서의 와프들의 비액티브 스레드들을, 보다 많은 수의 비액티브 스레드들을 갖는 압축 풀에서의 와프들의 액티브 스레드들과 교환한다. GPU (12) 가 임의의 새로운 와프들의 스레드 압축을 끝내면, DBM (52) 는 실행을 위하여 압축 풀 (156) 로부터 와프들을 릴리즈한다.

[0083] 동시에, 큐는 위에 설명된 바와 같이, 와프들이 실행에 대하여 재개한 후, 임의의 배리어 상에서 중지된 와프들

을 수신하는 것을 계속할 수도 있다. GPU (12) 는 위에 설명된 바와 같이 삽입 분류를 이용하여 새롭게 수신된 와프를 분류하고 이들을 큐에서의 기존의 와프들과 함께 큐에서 분류할 수도 있다. 모든 와프들이 다이버전스 배리어 상에서 중지되어 큐로 이동하거나 또는 실행을 끝낸 다음 종료하면, GPU (12) 는 큐로부터 현재의 제 1 와프 그룹을 분리하여 압축 풀을 형성하고, 압축 풀에 대해 압축을 수행한다.

[0084] 일부 커널 애플리케이션들은 커널을 실행하는 스레드들의 균일한 페이싱을 요구할 수도 있다. 스레드가 실행하는 페이스에 대하여 민감한 커널들은 또한 다이버전스 배리어들의 이용을 복잡하게 할 수도 있다. 예를 들어, 다이버전스 배리어들이 추가되었던 이러한 페이스 감응형 커널을 실행할 때, 일부 와프들은 다이버전스 배리어들에 도달 및 중지할 수도 있는 한편, 다른 와프들은 커널의 명령 시퀀스에서 훨씬 뒤에까지 다이버전스 배리어들에서 중지되지 않을 수도 있다. 따라서, 다이버전스 배리어들은 불균일한 스레드 및 와프 페이싱을 야기할 수도 있다. 제 1 연관된 불 조건을 갖는 제 1 다이버전스 배리어스레드 페이싱을 주변에서의 균일한 스레드 페이싱을 위해, 프로그래머는 제 1 불 조건의 불 보수 (complement) 인 제 2 연관된 불 연산식을 갖는 제 2 다이버전스 배리어 명령에 삽입할 수도 있다.

[0085] 다음 의사코드는 이 기법을 예시한다.

```
i = 0;
while(i < dynamic limit) {
 i++;
 //to eliminate divergence
 divergence barrier(diverg conditon==true);
 // do some work
 }
//warps do not hit barrier wait for those do
divergence barrier(diverg conditon==false);
// some heavy work.
```

[0086]

[0087] 위에 의사코드에서, GPU (12) 는 제 1 불 조건과 연관된 제 1 다이버전스 배리어 명령을 포함하는 루프를 실행한다. 코드는 제 1 불 조건의 보수인 제 2 불 조건을 갖는 루프 외부에 제 2 다이버전스 배리어 명령을 포함한다. 제 2 불 조건이 제 1 불 조건의 보수이면, GPU (12) 는 제 1 또는 제 2 다이버전스 배리어 명령에서 각각의 와프를 중지시킴으로써 일관성있는 스레드 페이싱을 보장한다.

[0088] 도 6 은 본 개시물의 기법들에 따라 다이버전스 배리어 명령들을 실행하기 위한 기법들을 예시하는 흐름도이다. GPU (12) 는 도 6 에 예시된 방법을 수행하도록 구성될 수도 있다. 일부 예들에서, GPU (12) 는 복수의 와프들 중 각각의 와프에 대하여, 각각의 와프의 대응하는 스레드에 대해 불 연산식이 참인지의 여부를 결정할 수도 있다 (200). GPU (12) 가 연산식이 참인 대응하는 스레드를 갖는 각각의 와프의 실행을 중지시킬 수도 있고 (202), 연산식이 참인 복수의 와프들의 각각에 대해 액티브 스레드들의 수를 결정할 수도 있다 (204). GPU (12) 는 복수의 와프들의 각각에서의 액티브 스레드들의 수에 기초하여 연산식이 참인 복수의 와프들을 분류할 수도 있다 (206). 그 후, GPU (12) 는 복수의 와프들 중 제 1 와프의 액티브 스레드의 스레드 데이터를, 복수의 와프들 중 제 2 와프의 비액티브 스레드의 스레드 데이터와 교환할 수도 있고 (208), 연산식이 참인 복수의 와프들 중 적어도 하나의 와프의 실행을 재개할 수도 있다 (210).

[0089] 여러 예들에서, 도 6 의 방법은 복수의 와프들 중 적어도 하나의 와프의 실행을 재개하기 전에 연산식이 참인 복수의 스레드들에 대하여 스레드 당 컨텍스트 데이터를 교환하는 것을 더 포함할 수도 있다. 액티브 스레드의 스레드 데이터는 액티브 스레드의 레지스터 데이터를 포함할 수도 있고, 비액티브 스레드의 스레드 데이터는 비액티브 스레드의 레지스터 데이터를 포함할 수도 있다. 일부 예들에서, 복수의 와프들을 분류하는 것은 삽입 분류를 이용하여 복수의 와프들을 분류하는 것을 포함할 수도 있다.

[0090] 일부 예들에서, GPU (12) 는 연산식이 참인 복수의 와프들의 각각의 와프에 대해 복수의 다이버전스 배리어들의 연관된 다이버전스 배리어를 추가로 결정할 수도 있고, 각각의 와프의 연관된 다이버전스 배리어에 기초하여 복수의 와프들의 각각의 와프를 복수의 압축 풀들로 그룹화할 수도 있다. 복수의 와프들을 분류하기 위해, GPU (12) 는 또한, 복수의 와프들을 분류하도록 구성될 수도 있고, 이는 복수의 압축 풀들 중 동일한 하나에 속

하는 복수의 와프들을 분류하는 것을 포함한다. 여러 예들에서, 제 1 와프 및 제 2 와프가 복수의 압축 풀들 중 동일한 압축 풀에 속하는 와프들을 포함하고, 조건이 참인 복수의 와프들 중 적어도 하나의 와프의 실행을 재개하기 위해, GPU (12) 는 동일한 하나의 압축 풀의 적어도 하나의 압축의 실행을 재개하도록 구성될 수도 있다.

[0091] 일부 예들에서, GPU (12) 는 복수의 와프들의 각각과 연관된 다이버전스 배리어에 기초하여 복수의 와프들의 각각에 프리픽스를 추가로 할당하고, 복수의 와프들을 적어도 하나의 압축 풀로 그룹화하기 위해, GPU (12) 는 할당된 프리픽스에 기초하여 적어도 하나의 압축 풀로 복수의 와프들을 그룹화할 수도 있다.

[0092] 또 다른 예들에서, GPU (12) 는 복수의 와프들이 모든 액티브 스레드들을 갖는 와프를 포함하는 것으로 추가로 결정할 수도 있고, 모든 액티브 스레드들을 갖는 와프의 실행을 재개할 수도 있다. 또 다른 예에서, 연산식이 참인 복수의 와프들을 분류하기 위해, GPU (12) 는 큐에서 복수의 와프들을 저장하고, 액티브 스레드들의 수에 기초하여 연산식이 참인 복수의 와프들을 분류하고, 큐에서 분류된 복수의 와프들을 저장하도록 구성될 수도 있다.

[0093] 또 다른 예에서, 컴파일러/드라이버 (18) 는 또한, 다이버전스가 발생할 가능성이 있는 로케이션 및 복수의 와프들 상에서 실행하는 커널 (20) 내에서 성능에 상당한 영향을 주는 로케이션 중 적어도 하나를 결정하도록 추가로 구성될 수도 있다. 컴파일러/드라이버 (18) 는 적어도 하나의 로케이션에서 커널 내에 다이버전스 배리어 명령을 삽입할 수도 있다. 도 6 의 방법의 불 연산식은 이 예에서 다이버전스 배리어 명령과 연관될 수도 있다.

[0094] 본 개시물에서 설명된 기술들은, 적어도 부분적으로, 하드웨어, 소프트웨어, 펌웨어, 또는 이들의 임의의 조합으로 구현될 수도 있다. 예를 들면, 설명된 기술들의 여러 양태들은, 하나 이상의 마이크로프로세서들을 포함하는 하나 이상의 프로세서들, 디지털 신호 프로세서들 (DSPs), ASIC들, FPGA들, 또는 다른 등가적인 집적 또는 이산 논리 회로부 뿐만 아니라 이와 같은 컴포넌트들의 임의의 조합들 내에서 구현될 수도 있다. 일반적으로, 용어 "프로세서" 또는 "프로세싱 회로부"는 임의의 전술한 논리 회로부만을 지칭할 수도 있거나, 또는 다른 논리 회로부 또는 임의의 다른 등가의 회로부, 이를 테면, 프로세싱을 수행하는 별개의 하드웨어와 조합하여 지칭할 수도 있다.

[0095] 이러한 하드웨어, 소프트웨어, 및 펌웨어는 본 개시물에서 설명된 여러 동작들, 및 기능들을 지원하기 위해 동일한 디바이스 내에서 또는 별개의 디바이스들 내에서 구현될 수도 있다. 또한, 설명된 유닛들, 모듈들 또는 컴포넌트들의 어떠한 것도 함께 또는 독립된 그러나 상호 동작가능한 로직 디바이스들로서 별개로 구현될 수도 있다. 모듈들 또는 유닛들로서의 상이한 피쳐들의 묘사는 상이한 기능의 양태들을 강조하기 위해 의도된 것이며 이러한 모듈들 또는 유닛들이 반드시 별개의 하드웨어, 또는 소프트웨어 컴포넌트들에 의해 실현되어야만 하는 것을 의미하는 것은 아니다. 대신, 하나 이상의 모듈들 또는 유닛들과 관련된 기능성은 별개의 하드웨어, 펌웨어, 및/또는 소프트웨어 컴포넌트들에 의해 수행될 수도 있거나, 또는 공통의 또는 별개의 하드웨어, 또는 소프트웨어 컴포넌트들 내에 통합될 수도 있다.

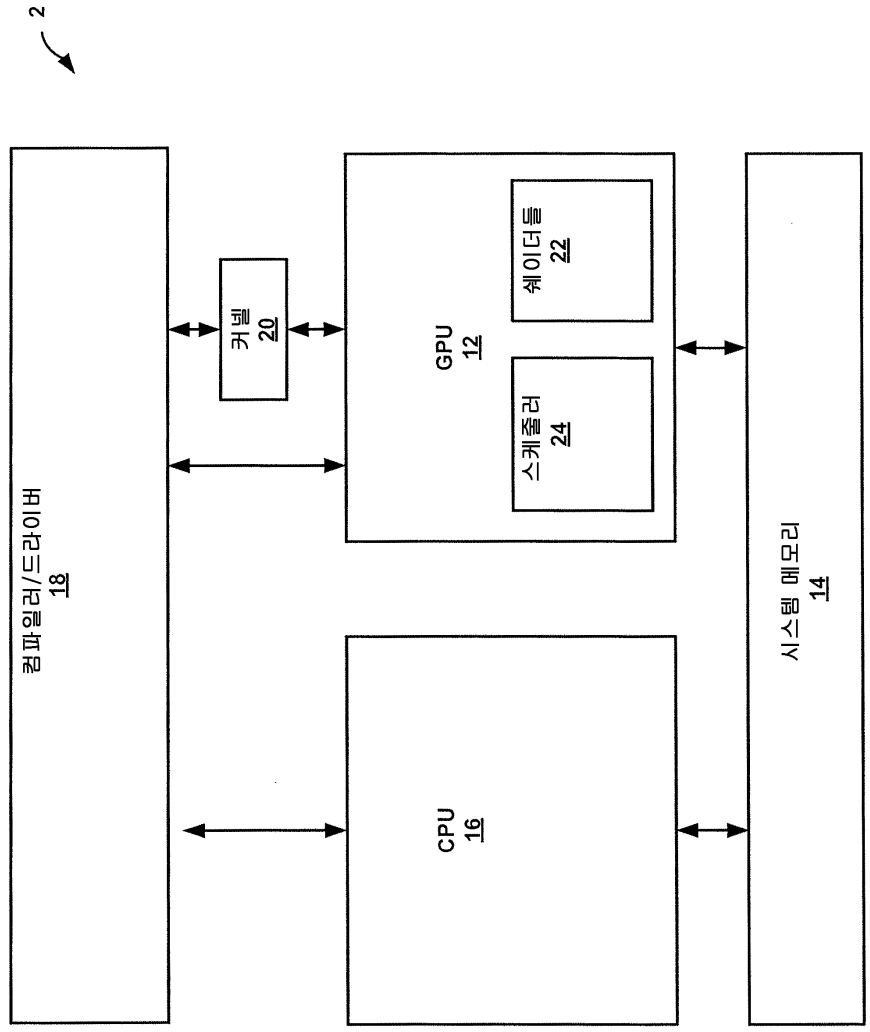
[0096] 본 개시물에 설명된 기법들이 또한 컴퓨터 판독가능 매체, 이를 테면, 명령들을 저장한 컴퓨터 판독가능 저장 매체에 저장, 내장 또는 인코딩될 수도 있다. 컴퓨터 판독가능 매체에 내장 또는 인코딩된 명령들은 명령들이 예를 들어, 하나 이상의 프로세스들에 의해 실행될 때 하나 이상의 프로세서들로 하여금, 본원에 설명된 기법들을 수행하게 할 수도 있다. 컴퓨터 판독 가능한 저장 매체들은 랜덤 액세스 메모리 (RAM), 판독 전용 메모리 (ROM), 프로그래밍가능 판독전용 메모리 (PROM), 소거가능 프로그래밍가능 판독 전용 메모리 (EPROM), 전기적 소거가능 프로그래밍 판독 전용 메모리 (EEPROM), 플래시 메모리, 하드디스크, CD-ROM, 플로피 디스크, 카세트, 자기 매체들, 광학 매체들, 또는 유형인 다른 컴퓨터 판독 가능한 매체들을 포함할 수도 있다.

[0097] 컴퓨터 판독가능 매체는 위에 나열된 것과 같은 유형의 저장 매체에 대응하는 컴퓨터 판독가능 저장 매체를 포함할 수도 있다. 컴퓨터 판독가능 매체는 예를 들어, 또한 통신 프로토콜에 따라 한 위치에서 다른 위치로 컴퓨터 프로그램의 전달을 용이하게 하는 임의의 매체를 포함한 통신 매체를 포함할 수도 있다. 이 방식으로, 어구 "컴퓨터 판독가능 매체"는 일반적으로 (1) 비일시적인 유형의 컴퓨터 판독가능 저장 매체들 또는 (2) 신호 또는 반송파와 같은 비유형의 컴퓨터 판독가능 통신 매체에 대응할 수도 있다.

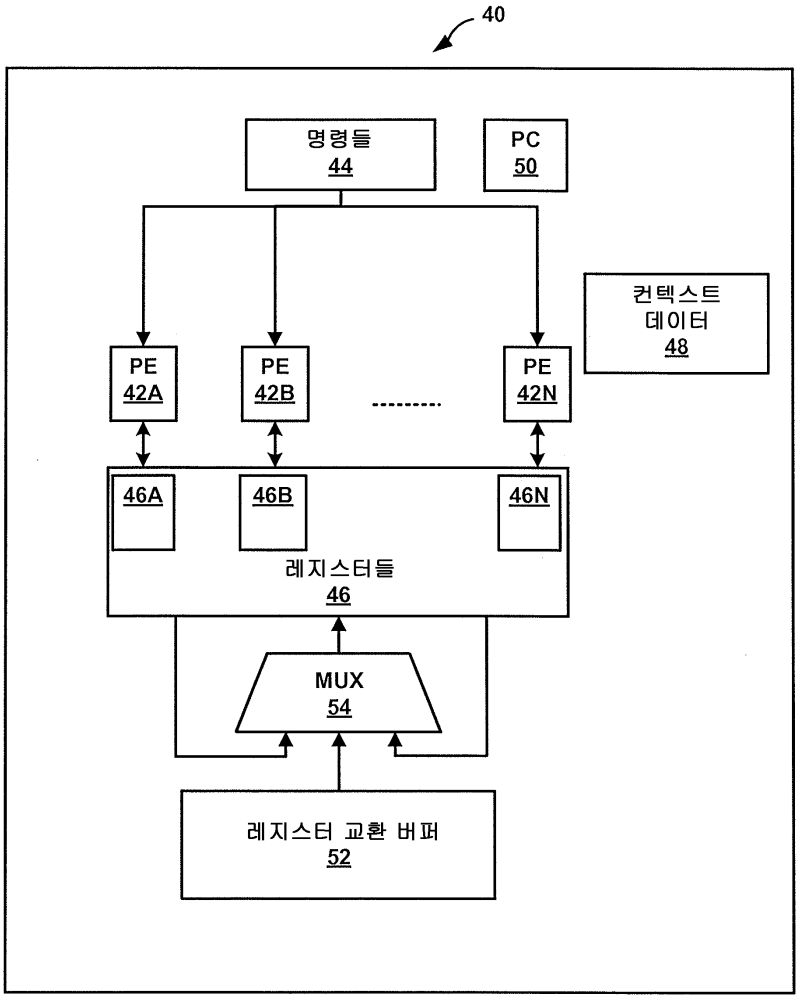
[0098] 여러 양태들이 설명되었다. 그러나, 변경들이 다음 청구항들의 범위로부터 벗어남이 없이 본 개시물의 기법들 또는 구조에 대해 행해질 수도 있다.

도면

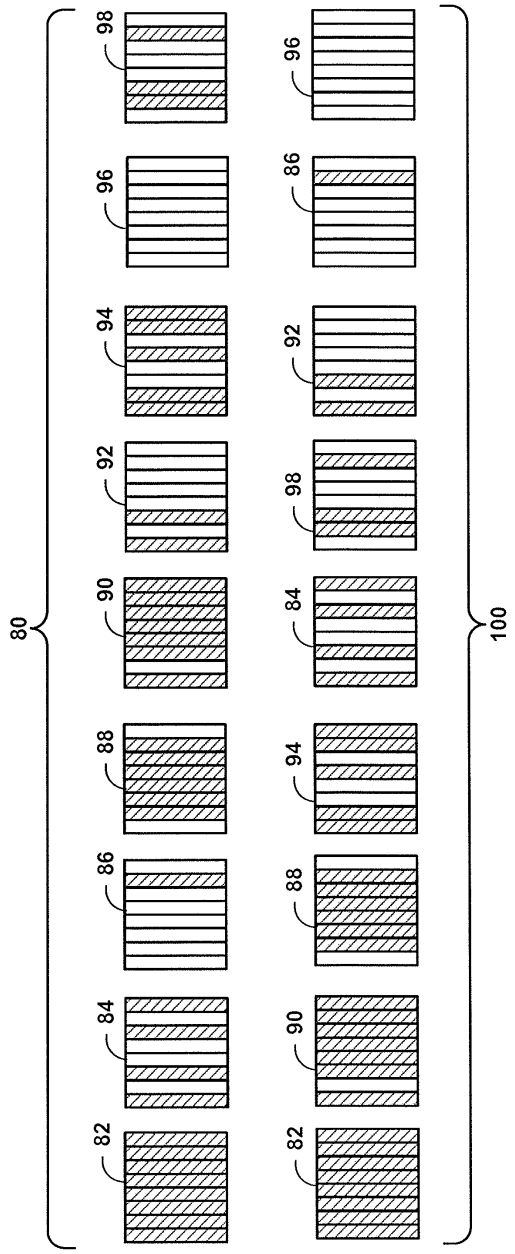
도면1



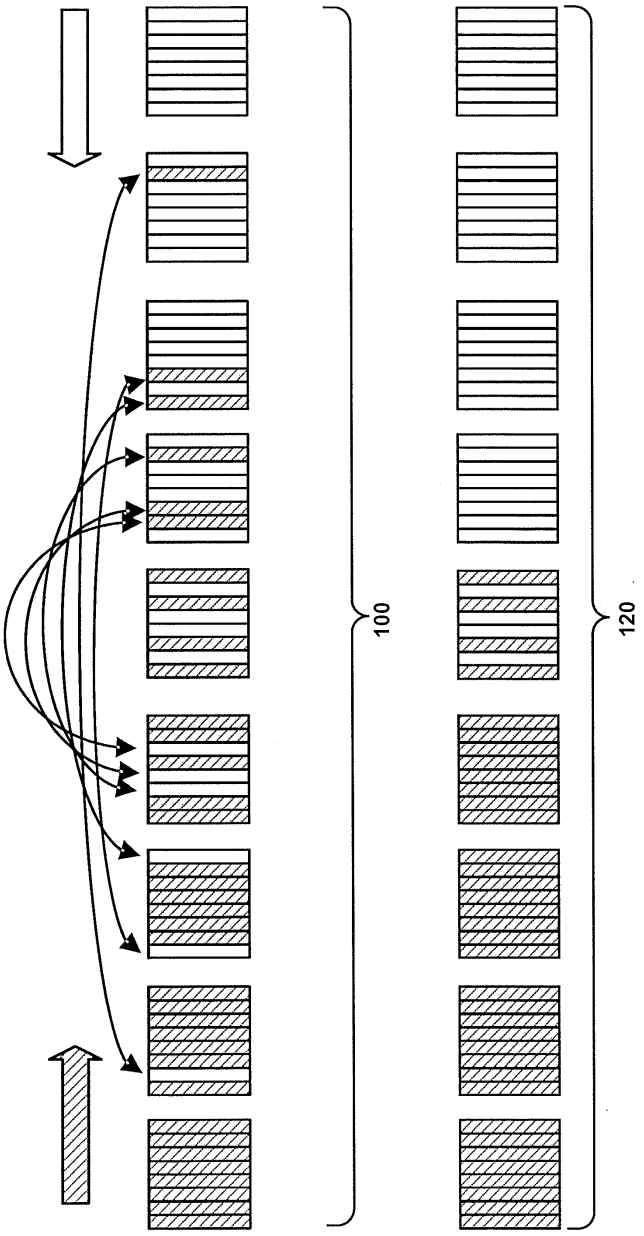
도면2


40
명령들
44
PC
50
컨텍스트
데이터
48
PE
42A
PE
42B
PE
42N
46A
46B
46N
레지스터들
46
MUX
54
레지스터 교환 버퍼
52

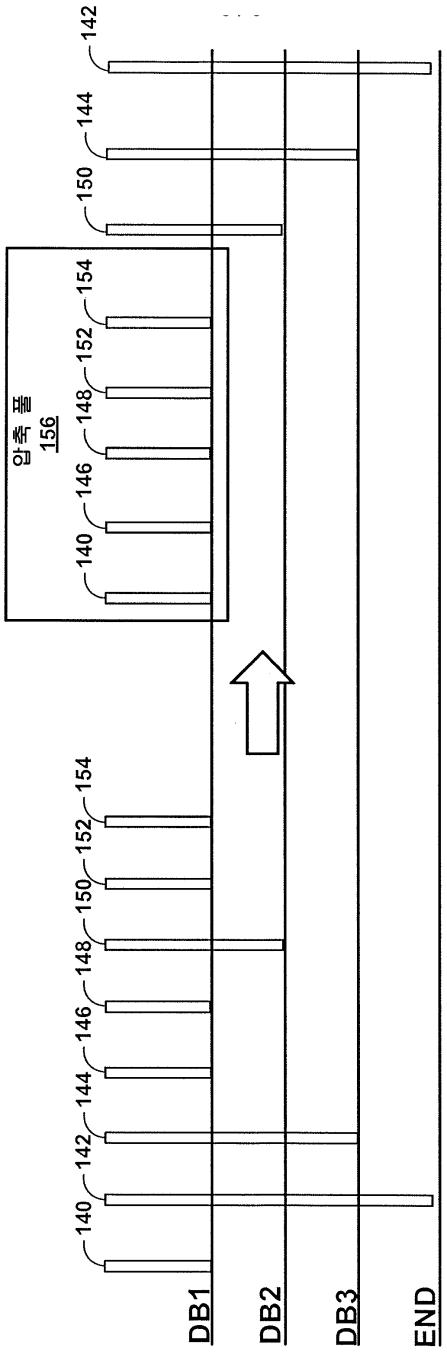
도면3



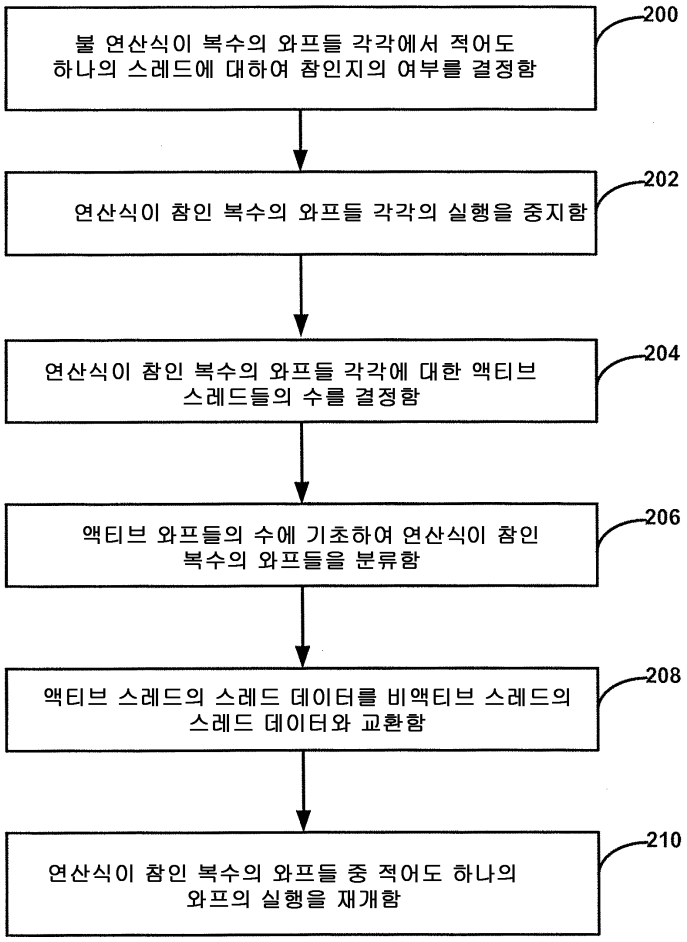
도면4



도면5


142
144
150
154
152
148
146
140
압축 풀
156
154
152
150
148
146
144
142
140
DB1
DB2
DB3
END

도면6


불 연산식이 복수의 와프들 각각에서 적어도 하나의 스레드에 대하여 참인지의 여부를 결정함
200
연산식이 참인 복수의 와프들 각각의 실행을 중지함
202
연산식이 참인 복수의 와프들 각각에 대한 액티브 스레드들의 수를 결정함
204
액티브 와프들의 수에 기초하여 연산식이 참인 복수의 와프들을 분류함
206
액티브 스레드의 스레드 데이터를 비액티브 스레드의 스레드 데이터와 교환함
208
연산식이 참인 복수의 와프들 중 적어도 하나의 와프의 실행을 재개함
210