(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0018237
(43) 공개일자 2024년02월13일

(51) 국제특허분류(Int. Cl.)
G06F 9/30 (2018.01) *G06F 12/0895* (2016.01)
G06F 12/0897 (2016.01) *G06F 9/38* (2024.01)
(52) CPC특허분류
G06F 9/30138 (2013.01)
G06F 12/0895 (2013.01)
(21) 출원번호 10-2022-0096238
(22) 출원일자 2022년08월02일
심사청구일자 2022년08월02일

(71) 출원인
이화여자대학교 산학협력단
서울특별시 서대문구 이화여대길 52 (대현동, 이화여자대학교)
(72) 발명자
윤명국
서울특별시 종로구 통일로 246-20(무악동, 무악현대아파트) 102동 603호
(74) 대리인
특허법인 무한

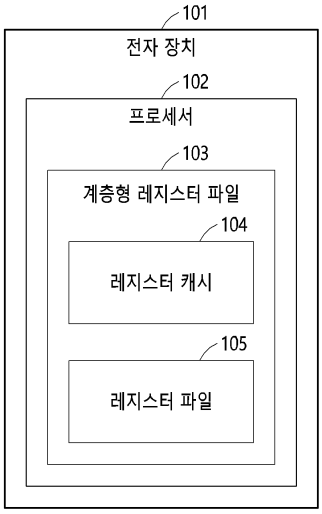
전체 청구항 수 : 총 10 항

(54) 발명의 명칭 **프로세서의 레지스터 캐시 인덱스 결정 방법 및 이를 수행하는 전자 장치**

(57) 요 약

프로세서의 레지스터 캐시 인덱스 결정 방법 및 이를 수행하는 전자 장치가 개시된다. 본 발명의 일실시예에 따른 프로세서의 레지스터 캐시 인덱스 결정 방법에 있어서, 상기 프로세서에 대한 명령어 연산을 수행하는 워프를 나타내는 워프 ID를 식별하는 단계; 상기 워프에 의해 사용되는 레지스터를 나타내는 레지스터 ID를 식별하는 단계; 및 상기 워프 ID 및 상기 레지스터 ID 간 논리 연산을 통해, 상기 명령어 연산의 결과를 저장하는 레지스터 캐시를 나타내는 레지스터 캐시 인덱스를 결정하는 단계를 포함하고, 상기 논리 연산은 상기 워프 ID 및 상기 레지스터 ID 중 비트 순서가 반전된 어느 하나의 최하위 비트(LSB)와 다른 하나의 최상위 비트(MSB)가 서로 정렬(align)되어 수행될 수 있다.

대 표 도 - 도1



(52) CPC특허분류
G06F 12/0897 (2013.01)
G06F 9/30141 (2013.01)
G06F 9/3824 (2013.01)
G06F 9/3842 (2013.01)
G06F 9/3851 (2023.08)
G06F 2212/6042 (2013.01)

이 발명을 지원한 국가연구개발사업
과제고유번호 1711152863
과제번호 2021-0-02068-002
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 정보통신방송혁신인재양성
연구과제명 인공지능 혁신 허브 연구 개발
기 여 율 1/2
과제수행기관명 고려대학교산학협력단
연구기간 2022.01.01 ~ 2022.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1711169967
과제번호 2021R1G1A1092196
부처명 과학기술정보통신부
과제관리(전문)기관명 한국연구재단
연구사업명 개인기초연구(과기정통부)
연구과제명 대규모 병렬 프로세서를 위한 미세 전력 관리 기법
기 여 율 1/2
과제수행기관명 이화여자대학교
연구기간 2022.03.01 ~ 2023.02.28

명 세 서

청구범위

청구항 1

프로세서의 레지스터 캐시 인덱스 결정 방법에 있어서,

상기 프로세서에 대한 명령어 연산을 수행하는 워프를 나타내는 워프 ID를 식별하는 단계;

상기 워프에 의해 사용되는 레지스터를 나타내는 레지스터 ID를 식별하는 단계; 및

상기 워프 ID 및 상기 레지스터 ID 간 논리 연산을 통해, 상기 명령어 연산의 결과를 저장하기 위한 레지스터 캐시의 위치를 나타내는 레지스터 캐시 인덱스를 결정하는 단계

를 포함하고,

상기 논리 연산은

상기 워프 ID 및 상기 레지스터 ID 중 비트 순서가 반전된 어느 하나의 최하위 비트(LSB)와 다른 하나의 최상위 비트(MSB)가 서로 정렬(align)되어 수행되는,

레지스터 캐시 인덱스 결정 방법.

청구항 2

제1항에 있어서,

상기 워프 ID 및 상기 레지스터 ID 중 어느 하나의 하위 비트일수록 다른 하나의 상위 비트와 상기 논리 연산이 수행되는, 레지스터 캐시 인덱스 결정 방법.

청구항 3

제1항에 있어서,

상기 논리 연산은

연산 결과가 제1 값일 확률과 상기 연산 결과가 제2 값일 확률이 동일한 연산인, 레지스터 캐시 인덱스 결정 방법.

청구항 4

제1항에 있어서,

상기 레지스터 캐시 인덱스를 결정하는 단계는

상기 워프 ID의 전체 비트와 상기 레지스터 ID의 전체 비트를 이용하여 상기 레지스터 캐시 인덱스를 결정하는, 레지스터 캐시 인덱스 결정 방법.

청구항 5

제1항에 있어서,

상기 레지스터 캐시 인덱스를 결정하는 단계는

상기 워프 ID 및 상기 레지스터 ID 중 비트 길이가 짧은 하나의 비트 순서를 반전(reverse)시키고, 상기 반전된

하나와 다른 하나 간 상기 논리 연산을 수행함으로써, 상기 레지스터 캐시 인덱스를 결정하는, 레지스터 캐시 인덱스 결정 방법.

청구항 6

전자 장치에 있어서,

상기 전자 장치에 포함된 프로세서에 대한 명령어 연산을 수행하는 워프를 나타내는 워프 ID를 식별하고, 상기 워프에 의해 사용되는 레지스터를 나타내는 레지스터 ID를 식별하며, 상기 워프 ID 및 상기 레지스터 ID 간 논리 연산을 통해, 상기 명령어 연산의 결과를 저장하기 위한 레지스터 캐시의 위치를 나타내는 레지스터 캐시 인덱스를 결정하는 하나 이상의 프로세서

를 포함하고,

상기 논리 연산은

상기 워프 ID 및 상기 레지스터 ID 중 비트 순서가 반전된 어느 하나의 최하위 비트(LSB)와 다른 하나의 최상위 비트(MSB)가 서로 정렬(align)되어 수행되는,

전자 장치.

청구항 7

제6항에 있어서,

상기 워프 ID 및 상기 레지스터 ID 중 어느 하나의 하위 비트일수록 다른 하나의 상위 비트와 상기 논리 연산이 수행되는, 전자 장치.

청구항 8

제6항에 있어서,

상기 논리 연산은

연산 결과가 제1 값일 확률과 상기 연산 결과가 제2 값일 확률이 동일한 연산인, 전자 장치.

청구항 9

제6항에 있어서,

상기 프로세서는

상기 워프 ID의 전체 비트와 상기 레지스터 ID의 전체 비트를 이용하여 상기 레지스터 캐시 인덱스를 결정하는, 전자 장치.

청구항 10

제6항에 있어서,

상기 프로세서는

상기 워프 ID 및 상기 레지스터 ID 중 비트 길이가 짧은 하나의 비트 순서를 반전(reverse)시키고, 상기 반전된 하나와 다른 하나 간 상기 논리 연산을 수행함으로써, 상기 레지스터 캐시 인덱스를 결정하는, 전자 장치.

발명의 설명

기 술 분 야

[0001] 본 발명은 프로세서의 레지스터 캐시 인덱스 결정 방법 및 이를 수행하는 전자 장치에 관한 것이다.

배 경 기 술

[0002] 일반적으로 그래픽 프로세싱 유닛의 경우 레지스터 파일은 프로세서의 전체 전력 사용량 중에서 상당한 양을 사용할 수 있다. 따라서, 레지스터 파일의 전력 소모를 줄일 필요가 있다. 전력 소모를 줄이기 위해 비휘발성 메모리를 이용한 레지스터 파일 구조가 제시된다. 비휘발성 메모리는 높은 집적도와 낮은 전력 사용량의 장점을 갖는다. 다만, 비휘발성 메모리는 쓰기 연산 시간이 오래 걸리고, 쓰기 전력 소모량이 커서 단독으로 레지스터 파일로 사용하기엔 적합하지 않다. 따라서, 비휘발성 메모리와 휘발성 메모리를 함께 사용하는 혼합형 레지스터 파일 구조가 제안되었다.

[0003] 혼합형 레지스터 파일 구조 중에서 특히 계층형 레지스터 파일 구조는 휘발성 메모리를 사용하는 레지스터 캐시 및 비휘발성 메모리를 사용하는 레지스터 파일을 포함한다. 계층형 레지스터 파일 구조는 명령어 연산 결과를 레지스터 캐시에 저장할 수 있다. 이때, 계층형 레지스터 파일 구조는 명령어 연산 결과를 저장하기 위해 레지스터 캐시 인덱스 연산을 수행할 수 있다. 레지스터 캐시 인덱스 연산 결과에 해당하는 레지스터 캐시에 이미 이전에 계산된 결과가 저장되어 있는 경우, 컨텐션(contention)이 발생하여 라이트 백(write-back) 연산이 발생할 수 있다. 따라서, 라이트 백 연산을 감소시킬 필요가 있다.

발명의 내용

해결하려는 과제

[0004] 본 발명은 계층형 레지스터 파일 구조에서 컨텐션으로 발생하는 라이트 백 연산을 감소시키기 위한 프로세서의 레지스터 캐시 인덱스 결정 방법 및 이를 수행하는 전자 장치를 제공한다.

과제의 해결 수단

[0005] 본 발명의 일실시예에 따른 프로세서의 레지스터 캐시 인덱스 결정 방법에 있어서, 상기 프로세서에 대한 명령어 연산을 수행하는 워프를 나타내는 워프 ID를 식별하는 단계; 상기 워프에 의해 사용되는 레지스터를 나타내는 레지스터 ID를 식별하는 단계; 및 상기 워프 ID 및 상기 레지스터 ID 간 논리 연산을 통해, 상기 명령어 연산의 결과를 저장하기 위한 레지스터 캐시의 위치를 나타내는 레지스터 캐시 인덱스를 결정하는 단계를 포함하고, 상기 논리 연산은 상기 워프 ID 및 상기 레지스터 ID 중 비트 순서가 반전된 어느 하나의 최하위 비트(LSB)와 다른 하나의 최상위 비트(MSB)가 서로 정렬(align)되어 수행될 수 있다.

[0006] 상기 워프 ID 및 상기 레지스터 ID 중 어느 하나의 하위 비트일수록 다른 하나의 상위 비트와 상기 논리 연산이 수행될 수 있다.

[0007] 상기 논리 연산은 연산 결과가 제1 값일 확률과 상기 연산 결과가 제2 값일 확률이 동일한 연산일 수 있다.

[0008] 상기 레지스터 캐시 인덱스를 결정하는 단계는 상기 워프 ID의 전체 비트와 상기 레지스터 ID의 전체 비트를 이용하여 상기 레지스터 캐시 인덱스를 결정할 수 있다.

[0009] 상기 레지스터 캐시 인덱스를 결정하는 단계는 상기 워프 ID 및 상기 레지스터 ID 중 비트 길이가 짧은 하나의 비트 순서를 반전(reverse)시키고, 상기 반전된 하나와 다른 하나 간 상기 논리 연산을 수행함으로써, 상기 레지스터 캐시 인덱스를 결정할 수 있다.

[0010] 본 발명의 일실시예에 따른 전자 장치에 있어서, 상기 전자 장치에 포함된 프로세서에 대한 명령어 연산을 수행하는 워프를 나타내는 워프 ID를 식별하고, 상기 워프에 의해 사용되는 레지스터를 나타내는 레지스터 ID를 식별하며, 상기 워프 ID 및 상기 레지스터 ID 간 논리 연산을 통해, 상기 명령어 연산의 결과를 저장하기 위한 레지스터 캐시 위치를 나타내는 레지스터 캐시 인덱스를 결정하는 하나 이상의 프로세서를 포함하고, 상기 논리 연산은 상기 워프 ID 및 상기 레지스터 ID 중 비트 순서가 반전된 어느 하나의 최하위 비트(LSB)와 다른 하나의 최상위 비트(MSB)가 서로 정렬(align)되어 수행될 수 있다.

[0011] 상기 워프 ID 및 상기 레지스터 ID 중 어느 하나의 하위 비트일수록 다른 하나의 상위 비트와 상기 논리 연산이

수행될 수 있다.

[0012] 상기 논리 연산은 연산 결과가 제1 값일 확률과 상기 연산 결과가 제2 값일 확률이 동일한 연산일 수 있다.

[0013] 상기 프로세서는 상기 워프 ID의 전체 비트와 상기 레지스터 ID의 전체 비트를 이용하여 상기 레지스터 캐시 인덱스를 결정할 수 있다.

[0014] 상기 프로세서는 상기 워프 ID 및 상기 레지스터 ID 중 비트 길이가 짧은 하나의 비트 순서를 반전(reverse)시키고, 상기 반전된 하나와 다른 하나 간 상기 논리 연산을 수행함으로써, 상기 레지스터 캐시 인덱스를 결정할 수 있다.

발명의 효과

[0015] 본 발명의 일실시예에 의하면, 계층형 레지스터 파일 구조에서 명령어 연산시에 컨텐션이 발생하는 것을 막아 라이트 백 연산이 발생하지 않도록 할 수 있다.

도면의 간단한 설명

[0016] 도 1은 본 발명의 일실시예에 따른 전자 장치를 설명하기 위한 도면이다.

도 2는 본 발명의 일실시예에 따른 라이트 백 연산을 설명하기 위한 도면이다.

도 3 내지 도 7은 본 발명의 일실시예에 따라 레지스터 캐시 인덱스를 결정할 때 고려될 수 있는 사항들을 설명하기 위한 도면이다.

도 8은 본 발명의 일실시예에 따른 레지스터 캐시 인덱스를 결정하는 방법을 설명하기 위한 플로우차트이다.

도 9는 본 발명의 일실시예에 따른 레지스터 캐시 인덱스를 결정하는 방법을 설명하기 위한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0017] 이하, 실시예들을 첨부된 도면을 참조하여 상세하게 설명한다. 그러나, 특허출원의 범위가 이러한 실시예들에 의해 제한되거나 한정되는 것은 아니다. 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타낸다.

[0018] 아래 설명하는 실시예들에는 다양한 변경이 가해질 수 있다. 아래 설명하는 실시예들은 실시 형태에 대해 한정하려는 것이 아니며, 이들에 대한 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다.

[0019] 제1 또는 제2 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 이런 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 이해되어야 한다. 예를 들어, 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소는 제1 구성요소로도 명명될 수 있다.

[0020] 실시예에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로, 실시예를 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수 개의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 명세서 상에 기재된 특징, 숫자, 단계, 동작, 구성 요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0021] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 실시예가 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가지고 있다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥 상 가지는 의미와 일치하는 의미를 가지는 것으로 해석되어야 하며, 본 출원에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0022] 또한, 첨부 도면을 참조하여 설명함에 있어, 도면 부호에 관계없이 동일한 구성 요소는 동일한 참조 부호를 부여하고 이에 대한 중복되는 설명은 생략하기로 한다. 실시예를 설명함에 있어서 관련된 공지 기술에 대한 구체적인 설명이 실시예의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우 그 상세한 설명을 생략한다.

[0023] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0025] 도 1은 본 발명의 일실시예에 따른 전자 장치를 설명하기 위한 도면이다.

[0026] 도 1을 참조하면 전자 장치(101)가 도시된다. 전자 장치(101)는 프로세서(102)를 포함하는 장치로서, 예를 들어, 휴대폰, 스마트 폰, 태블릿, 전자북 장치, 랩탑, 퍼스널 컴퓨터, 데스크탑, 워크스테이션 또는 서버와 같은 다양한 컴퓨팅 장치, 스마트 시계, 스마트 안경, HMD(Head-Mounted Display), 또는 스마트 의류와 같은 다양한 웨어러블 기기, 스마트 스피커, 스마트 TV, 또는 스마트 냉장고와 같은 다양한 가전장치, 스마트 자동차, 스마트 키오스크, IoT(Internet of Things) 기기, WAD(Walking Assist Device), 드론, 또는 로봇일 수 있다.

[0027] 프로세서(102)는 전자 장치(101)에서 수행되는 동작이나 처리를 수행하는 엘리먼트로서, 예를 들어, CPU(central processing unit), GPU(graphic processing unit), NPU(neural processing unit) 및 TPU(tensor processing unit) 등일 수 있다. 프로세서(102)는 레지스터 캐시(104)을 포함할 수 있다.

[0028] 계층형 레지스터 파일(103)은 레지스터 캐시(104) 및 레지스터 파일(105)을 포함할 수 있다. 레지스터 파일(105)은 계층형 레지스터 파일(103)의 메인 레지스터 파일일 수 있다. 레지스터 캐시(104)는, 예를 들어, SRAM(Static Random Access Memory, SRAM) 기반의 휘발성 메모리일 수 있다. 레지스터 파일(105)은, 예를 들어, STT-MRAM(Spin-Transfer-Torque Magnetoresistive Random Access Memory, STT-MRAM) 및 SOT-MRAM(Spin-Orbit Torque MRAM)기반의 비휘발성 메모리일 수 있다.

[0029] 프로세서(102)는 명령어 연산 결과를 레지스터 캐시(104)에 더 이상 저장할 수 없는 경우, 프로세서(102)는 레지스터 캐시(104)의 일부 데이터를 레지스터 파일(105)로 이동시키는 라이트 백(write-back) 연산을 수행할 수 있다.

[0030] 이하에서는 계층적 레지스터 파일 구조에서 라이트 백 연산을 수행하는 방법에 대해서 설명하겠다.

[0032] 도 2는 본 발명의 일실시예에 따른 라이트 백 연산을 설명하기 위한 도면이다.

[0033] 계층형 레지스터 파일 구조는 하이브리드 레지스터 파일 구조와는 달리 레지스터 캐시에 새로운 연산 결과를 작성할 수 없는 경우에만 라이트 백 연산을 수행할 수 있다.

[0034] 도 2를 참조하면 연산 장치(201) 및 계층형 레지스터 파일(예: 도 1의 계층형 레지스터 파일(103))이 도시된다. 계층형 레지스터 파일(103)은 레지스터 캐시(예: 도 1의 레지스터 캐시(104)) 및 메인 레지스터 파일인 레지스터 파일(예: 도 1의 레지스터 파일(105))을 포함할 수 있다.

[0035] 먼저, 연산 장치(201)는 레지스터 파일(105) 및/또는 레지스터 캐시(104)로부터 데이터를 읽어 연산을 수행할 수 있다. 연산 결과는 휘발성 메모리인 레지스터 캐시(104)에 먼저 저장될 수 있다.

[0036] 이때, 레지스터 캐시(104)에 저장 공간이 부족한 경우, 프로세서는 레지스터 캐시(104)에 저장되어 있던 데이터를 레지스터 파일(105)로 이동하는 연산인 라이트 백 연산을 수행할 수 있다. 라이트 백 연산에 의해 레지스터 파일(105)로 이동된 데이터는 레지스터 캐시(104)에서 제거될 수 있다.

[0037] 레지스터 캐시(104)는 저장 공간이 제한적일 수 있다. 따라서, 라이트 백 연산이 자주 발생하여 성능 저하가 발생하는 문제가 있을 수 있다. 그러므로, 라이트 백 연산을 줄일 필요가 있다.

[0038] 이하에서는 레지스터 캐시 인덱스를 계산하는 방법을 설명하도록 하겠다.

[0040] 도 3 내지 도 7은 본 발명의 일실시예에 따라 레지스터 캐시 인덱스를 결정할 때 고려될 수 있는 사항들을 설명하기 위한 도면이다.

[0041] 도 3을 참조하면, 레지스터 캐시 인덱스를 결정하는 일례가 도시된다.

[0042] 레지스터 캐시(예: 도 1의 104)는 Direct-Mapped 캐시 구조를 사용할 수 있다. Direct-Mapped 캐시 구조에서는 명령어 연산 결과를 저장하기 위한 위치를 결정하기 위해 워프 ID와 레지스터 ID의 일부를 이용할 수 있다.

[0043] 레지스터 캐시 인덱스를 결정하는 방법을 설명하기 위해 프로세서는 그래픽 프로세싱 유닛으로 가정하겠다. 레지스터 캐시는 총 256개의 엔트리로 구성되었다고 가정하겠다. 또한, 워프 ID(301)는 총 6비트, 레지스터 ID(302)는 총 8비트로 구성되었다고 가정하겠다. 워프는 복수의 스레드를 포함할 수 있다. 하나의 워프에 포함되는 스레드는 같은 명령어로 동시에 동작할 수 있다. 워프 ID(301)는 명령어 연산을 수행하는 워프의 인덱스일 수 있다. 레지스터 ID(302)는 워프의 연산 결과를 저장하기 위한 레지스터 파일의 인덱스 일 수 있다.

[0044] 즉, 레지스터 캐시의 엔트리를 나타내기 위해서 총 8 비트($log_2(256)= 8$)의 레지스터 캐시 인덱스(303)가 필요함을 의미한다. 그리고, 그래픽 프로세싱 유닛의 스트리밍 멀티프로세서가 동시에 실행할 수 있는 워프의 수는 총 64개($log_2(64)= 6$)이고, 하나의 스레드가 사용할 수 있는 레지스터의 개수는 총 256($log_2(256) = 8$)개임을 의미한다.

[0045] 프로세서는 워프 ID(301) 및 레지스터 ID(302)를 식별할 수 있다. 프로세서는 워프 ID(301) 및 레지스터 ID(302)로부터 일부 비트를 추출하여 레지스터 캐시 인덱스(303)를 계산할 수 있다.

[0046] 도 3을 참조하면, 프로세서는 총 6 비트인 워프 ID(301)에서 3 비트를 추출하고, 총 8비트인 레지스터 ID(302)에서 5 비트를 추출할 수 있다. 프로세서는 추출된 워프 ID의 3 비트와 추출된 레지스터 ID의 5 비트를 결합하여 총 8 비트의 레지스터 캐시 인덱스(303)를 결정할 수 있다. 이때, 워프 ID(301)와 레지스터 ID(302)에서 추출하는 비트의 수는 하드웨어 구성마다 다를 수 있다.

[0047] 다만, 프로세서가 위 방법으로 레지스터 캐시 인덱스를 결정하는 경우, 컨텐션(contention)이 자주 발생하여 도 2에서 설명한 라이트 백 연산이 발생할 수 있으므로, 이러한 라이트 백 연산 발생이 최소화되도록 하는 레지스터 캐시 인덱스 계산 방법이 요구될 수 있다. 도 4 내지 도 7을 통해 레지스터 캐시 인덱스 계산 중 발생할 수 있는 컨텐션에 대해 설명한다.

[0049] 도 4를 참조하면, 인터 워프 컨텐션(inter-warp contention)을 설명하기 위한 예시가 도시된다.

[0050] 도 4를 참조하면, 워프 ID를 모두 표현하기 위해서 5 비트가 필요하고, 레지스터 ID를 모두 표현하기 위해서 5 비트가 필요하다고 가정하겠다. 그리고, 워프 1번(Warp #1)과 워프 9번(Warp #9)이 명령어 연산을 수행한 후 연산 결과를 레지스터 2번(Reg #2)에 저장한다고 가정하겠다.

[0051] 또한, 레지스터 캐시 인덱스를 결정하기 위해 워프 ID의 3 비트, 레지스터 ID의 5 비트를 샘플링한다고 가정하겠다. 따라서, 워프 1번(Warp #1)과 워프 9번(Warp #9)은 레지스터 캐시 인덱스 결정을 위하여 동일한 (001) 비트를 사용할 수 있다. 그리고, 레지스터 2번(Reg #2)은 모든 비트를 사용할 수 있다.

[0052] 그 결과, 레지스터 캐시 인덱스는 (00100010), 즉 레지스터 캐시 인덱스 34번으로 결정될 수 있다. 이때, 워프 1번(Warp #1)과 워프 9번(Warp #9)은 모두 동일한 34번 레지스터 캐시에 결과를 작성하려 하므로, 프로세서는 워프 1번(Warp #1)과 워프 9번(Warp #9) 중에서 먼저 연산이 완료된 결과를 레지스터 캐시에서 레지스터 파일로 이동시키는 라이트 백 연산을 수행할 수 있다. 즉, 인터 워프 컨텐션이 발생함에 따라 라이트 백 연산이 수행될 수 있다.

[0053] 결과적으로, 레지스터 캐시 인덱스 연산을 위해 사용되는 워프 ID의 비트 수가 감소할수록 인터 워프 컨텐션은 더 많이 발생할 수 있다.

[0055] 도 5를 참조하면, 인트라 워프 컨텐션(intra-warp contention)을 설명하기 위한 예시가 도시된다.

[0056] 도 5를 참조하면, 워프 ID를 모두 표현하기 위해서 5 비트가 필요하고, 레지스터 ID를 모두 표현하기 위해서 5 비트가 필요하다고 가정하겠다. 그리고, 워프 9번(Warp #1) 이 명령어 연산을 수행한 후 연산 결과를 레지스터 2번(Reg #2)과 레지스터 10번(Reg #10)에 저장한다고 가정하겠다.

[0057] 다만, 인터 워프 컨텐션과는 달리 레지스터 캐시 인덱스를 결정하기 위해 워프 ID의 5 비트, 레지스터 ID의 3 비트를 샘플링한다고 가정하겠다. 따라서, 레지스터 2번(Warp #2)과 레지스터 10번(Warp #10)은 레지스터 캐시 인덱스 결정을 위하여 동일한 (001) 비트를 사용할 수 있다. 그리고, 워프 9번(Warp #9)은 모든 비트를 사용할 수 있다.

[0058] 그 결과, 레지스터 캐시 인덱스는 (01001010), 즉 레지스터 캐시 인덱스 74번으로 결정될 수 있다. 이때, 두 레지스터는 모두 동일한 74번 레지스터 캐시에 결과를 작성하려 하므로, 프로세서는 두 레지스터 중에서 먼저 연산이 완료된 결과를 레지스터 캐시에서 레지스터 파일로 업데이트시키는 라이트 백 연산을 수행할 수 있다. 즉, 인트라 워프 컨텐션이 발생함에 따라 라이트 백 연산이 수행될 수 있다.

[0059] 결과적으로, 레지스터 캐시 인덱스 연산을 위해 사용되는 레지스터 ID의 비트 수가 감소할수록 인트라 워프 컨텐션은 더 많이 발생할 수 있다.

[0061] 도 6를 참조하면, 폴스 컨텐션을 설명하기 위한 예시가 도시된다.

[0062] 폴스 컨텐션(false contention)은 레지스터 캐시 인덱스의 결정을 위해 샘플링하는 워프 ID와 레지스터 ID 중 일부 비트가 항상 0인 경우 발생할 수 있다.

[0063] 폴스 컨텐션을 설명하기 위해 도 5와 마찬가지로 워프 ID를 모두 표현하기 위해서 5 비트가 필요하고, 레지스터 캐시 인덱스의 결정을 위해 워프 ID의 5 비트 모두가 사용된다고 가정하겠다.

[0064] 만약 애플리케이션 특성상 실행 가능한 워프가 16개로 워프 ID가 0 ~ 15번만 존재하는 경우, 4 비트만으로 애플리케이션이 실행 가능한 워프를 모두 표현할 수 있으므로 MSB(Most Significant Bit, MSB)는 항상 0으로 결정될 수 있다. 여기서, MSB는 워프 ID의 가장 왼쪽에 위치한 비트로 최상위 비트일 수 있다.

[0065] 결국, 최상위 비트가 항상 0으로 결정되어 레지스터 캐시의 엔트리 중에서 일부는 연산이 종료할 때까지 한 번도 사용되지 않을 수 있다. 도 6의 y축 값은 레지스터 캐시의 엔트리에 연산 결과가 작성된 횟수를 일반화한 값이다. 즉, y축 값은 특정 레지스터 캐시의 엔트리에 연산 결과가 작성된 횟수를 전체 레지스터 캐시의 엔트리에 연산 결과가 작성된 횟수로 나눈 값이다. 만약 y 값이 0이라면, 한 번도 연산 결과가 작성되지 않았다는 의미이다. x축 값은 레지스터 캐시의 엔트리 번호를 나타낸다.

[0066] 따라서, 레지스터 캐시의 엔트리 16번부터 32번까지는 한 번도 연산 결과가 작성되지 않았음을 알 수 있다. 그 결과, 어떤 레지스터 캐시의 엔트리에는 한 번도 연산 결과가 작성되지 않는 반면, 다른 엔트리에서는 계속해서 연산 결과가 작성되는 것을 확인할 수 있다.

[0067] 이를 해결하기 위한 방법은 Direct-Mapped 캐시 구조를 Fully-Associative 캐시 구조로 변경하는 것이다. 다만, Fully-Associative 캐시 구조는 하드웨어적으로 복잡하여 구현하기에 어려울 수 있다.

[0068] 앞서 설명한 다양한 컨텐션들의 발생이 레지스터 캐시 인덱스 계산 시 방지되어야 하며, 이하에서는 Direct-Mapped 캐시 구조에서 컨텐션을 줄일 수 있는 방법에 대해서 설명하도록 하겠다.

[0070] 도 7은 본 발명의 일실시예에 따른 레지스터와 워프의 관계를 설명하기 위한 그래프이다.

[0071] 도 7은 프로세서의 예시인 그래픽 프로세싱 유닛에서 레지스터와 워프의 관계를 설명하기 위한 그래프이다. 그래픽 프로세싱 유닛은 많은 수의 스레드를 동시에 스케줄링 할 수 있다. 그래픽 프로세싱 유닛에서 실행되는 스레드의 수는 3가지 요소로 결정될 수 있다. 3가지 요소는 총 실행가능한 스레드 수의 제한, 레지스터 파일의 크기 및 공유 메모리(shared memory)의 크기일 수 있다.

[0072] 특히, 총 실행가능한 스레드의 수와 레지스터 파일의 크기는 관련이 있다. 각 스레드가 사용하는 레지스터의 수가 적을수록 그래픽 프로세싱 유닛에서 사용되는 스레드(즉, 워프)의 수는 증가할 수 있다. 반대로, 각 스레드가 사용하는 레지스터의 수가 많을수록 그래픽 프로세싱 유닛에서 사용되는 스레드(즉, 워프)의 수는 감소할 수 있다.

[0073] 구체적으로, 도 7을 참조하면, 도 7은 그래픽 프로세싱 유닛에서 스레드의 레지스터 사용량과 총 실행되는 스레드(즉, 워프)의 관계를 나타낸 그래프이다.

[0074] 여기서, 워프 ID는 총 6비트, 레지스터 ID는 총 8비트로 구성될 수 있다. 이는 스레드의 레지스터 사용량과 총 실행되는 스레드(즉, 워프)의 관계를 설명하기 위한 예시일뿐 이에 한정되지 않는다.

[0075] 명령어 연산 실행 시 사용되는 레지스터 ID의 비트 수를 나타내는, 원으로 표시된 선을 제1 선이라고 하겠다. 명령어 연산 실행 시 사용되는 워프 ID의 비트 수를 나타내는, 삼각형으로 표시된 선을 제2 선이라고 하겠다. 제1 선과 제2선의 합인, 명령어 연산 실행 시 사용되는 레지스터 ID와 워프 ID의 비트 수를 합한 비트 수를 나타내는 점으로 표시된 선을 제3 선이라고 하겠다. 워프 ID 와 레지스터 ID의 총 비트 수를 합한 실선으로 표시된 선을 제4 선이라고 하겠다.

[0076] 왼쪽에서 두번째 바(8에 해당함)를 살피면, 하나의 스레드가 연산을 위해서 8개의 레지스터를 사용하는 경우이

다. 이 경우, 총 실행되는 스레드의 개수는 총 2048개로 가정하면, 일반적으로 1개의 워프에는 32개의 스레드가 포함될 수 있으므로 64개의 워프가 사용될 수 있다. 따라서, 명령어 연산 실행 시 사용되는 64개의 워프를 표현하기 위해서 워프 ID는 6비트(제2 선)가 필요할 수 있다. 또한, 명령어 연산 실행 시 사용되는 8개의 레지스터를 표현하기 위해서는 레지스터 ID는 3비트(제1 선)가 필요 할 수 있다. 따라서, 제3은 9 비트 일 수 있다. 제4 선은 워프 ID와 레지스터 ID의 총 비트 수의 합이므로 항상 14 비트 일 수 있다.

[0077] 도 7을 살피면, 제3선은 8 비트 내지 12 비트를 가질 수 있다. 즉, 제3 선은 항상 14 비트인 제4 선보다 아래에 있다. 이것은 실제 애플리케이션은 워프 ID와 레지스터 ID의 모든 비트를 사용하지 않음을 의미할 수 있다.

[0078] 또한, 제1 선이 증가하면 제2 선이 감소하는 것을 확인할 수 있다. 즉, 각 스레드가 사용하는 레지스터의 수가 많을수록 그래픽 프로세싱 유닛에서 사용되는 스레드(즉, 워프)의 수는 감소할 수 있다.

[0079] 결과적으로, 그래픽 프로세싱 유닛에서 실행되는 각 어플리케이션은 그 특성에 따라서 총 실행되는 스레드(즉, 워프)의 수와 각 스레드가 사용하는 레지스터의 수가 다를 수 있다. 이러한 특성을 이용하여 레지스터 캐시의 인덱스를 결정하면 컨텐션 문제를 개선할 수 있다.

[0080] 따라서, 이하에서는, 컨텐션 문제를 개선하기 위해 스레드의 레지스터 사용량과 총 실행되는 스레드(즉, 워프)의 수를 고려한 레지스터 캐시 인덱스 결정 방법에 대해서 설명하도록 하겠다.

[0082] 도 8은 본 발명의 일실시예에 따른 레지스터 캐시 인덱스를 결정하는 방법을 설명하기 위한 플로우차트이다.

[0083] 단계(801)에서, 프로세서는 프로세서에 대한 명령어 연산을 수행하는 워프를 나타내는 워프 ID를 식별할 수 있다.

[0084] 워프 ID는 2진법으로 표시될 수 있다. 예를 들어, 총 5 비트를 갖는 워프 ID에서 워프 9번의 경우 (01001)로 표현될 수 있다. 따라서, 프로세서는 2진법으로 표시된 워프 ID를 식별할 수 있다.

[0085] 단계(802)에서, 프로세서는 워프에 의해 사용되는 레지스터를 나타내는 레지스터 ID를 식별할 수 있다.

[0086] 레지스터 ID는 이진법으로 표시될 수 있다. 예를 들어, 총 5 비트를 갖는 레지스터 ID에서 레지스터 10번의 경우 (01010)로 표현될 수 있다. 따라서, 프로세서는 2진법으로 표시된 레지스터 ID를 식별할 수 있다.

[0087] 단계(803)에서, 워프 ID 및 레지스터 ID 간 논리 연산을 통해, 명령어 연산의 결과를 저장하기 위한 레지스터 캐시의 위치를 나타내는 레지스터 캐시 인덱스를 결정할 수 있다.

[0088] 레지스터 캐시 인덱스를 결정할 때, 워프 ID의 전체 비트와 레지스터 ID의 전체 비트가 사용될 수 있다. 즉, 도 3에서 워프 ID와 레지스터 ID에서 일부 비트를 추출하여 레지스터 캐시 인덱스를 계산하는 것과 달리 전체 비트를 사용할 수 있다.

[0089] 도 7에 따르면, 각 스레드(즉, 워프)가 사용하는 레지스터의 수가 많을수록 그래픽 프로세싱 유닛에서 사용되는 스레드(즉, 워프)의 수가 감소하는 특징이 있음을 알 수 있다. 따라서, 워프가 사용하는 레지스터의 수가 많을수록 레지스터 ID의 상위 비트는 상대적으로 워프 ID의 상위 비트보다 레지스터 캐시 인덱스의 결정에 영향을 더 미칠 수 있다. 반대로, 워프가 사용하는 레지스터의 수가 적을수록 워프 ID의 상위 비트는 레지스터 ID의 상위 비트보다 레지스터 캐시 인덱스의 결정에 영향을 더 미칠 수 있다.

[0090] 그러므로, 워프 ID 및 레지스터 ID 중 어느 하나의 레지스터 캐시 인덱스 결정에 영향을 주지 못하는 비트와 다른 하나의 레지스터 캐시 인덱스 결정에 영향을 주는 비트를 서로 정렬시키기 위해서, 워프 ID 와 레지스터 ID 중 어느 하나의 비트 순서를 반전시킬 필요가 있다.

[0091] 그러고나서, 프로세서는 워프 ID 및 레지스터 ID 중 비트 순서가 반전된 어느 하나의 최하위 비트(Least Significant Bit, LSB)와 다른 하나의 최상위 비트(Most Significant Bit, MSB)가 서로 정렬되도록 하여 논리 연산을 수행할 수 있다.

[0092] 또한, 프로세서는 정렬을 통해 워프 ID 및 레지스터 ID 중 어느 하나의 하위 비트일수록 다른 하나의 상위 비트와 논리 연산을 수행할 수 있다.

[0093] 특히, 프로세서는 워프 ID 및 레지스터 ID 중 비트 길이가 짧은 하나의 비트 순서를 반전시키고, 반전된 하나와 다른 하나 간 논리 연산을 수행하여 레지스터 캐시 인덱스를 결정할 수 있다.

[0094] 여기서, 논리 연산은 연산 결과가 제1 값일 확률과 연산 결과가 제2 값일 확률이 동일한 연산이다. 즉, 논리 연산은 연산 결과인 0 과 1이 나올 확률이 동일한 연산일 수 있다. 따라서, 논리 연산은 결과가 0과 1이 나올 확률이 동일한 게이트인 XOR 게이트를 이용한 연산은 물론, 결과가 0과 1이 나올 확률이 동일하도록 여러 게이트를 적절히 결합한 연산일 수 있다.

[0095] 결론적으로, 단순히 워프 ID와 레지스터 ID의 일부를 결합하여 레지스터 캐시 인덱스를 결정하는 방법과는 달리, 본 발명의 방법은 비트의 반전 및 논리 연산을 이용함으로써, 레지스터 캐시 인덱스가 중복되는 경우가 줄어 컨텐션의 발생을 줄일 수 있다. 따라서, 그 결과 라이트 백 연산이 줄어들 수 있다.

[0096] 이하에서는 예시를 이용하여 컨텐션의 발생을 줄이는 레지스터 캐시 인덱스 결정 방법을 설명하도록 하겠다.

[0098] 도 9는 본 발명의 일실시예에 따른 레지스터 캐시 인덱스를 결정하는 방법을 설명하기 위한 도면이다.

[0099] 도 9를 참조하면, 워프 ID(예: 도 3의 워프 ID(301)) 및 레지스터 ID(예: 도 3의 레지스터 ID(302))에 기반하여 레지스터 캐시 인덱스가 결정되는 예시가 도시된다.

[0100] 여기서, 워프 ID(301)는 총 6 비트를 가지고 64개의 워프를 표현할 수 있으나, 워프 ID(301)에 포함되는 비트수가 전술한 예로 한정되지 않는다. 레지스터 ID(302)는 총 8 비트를 가지고 256개의 레지스터를 표현할 수 있으나, 레지스터 ID(302)에 포함되는 비트수가 전술한 예로 한정되지 않는다.

[0101] 먼저, 워프 ID(301)와 레지스터 ID(302) 중에서 어느 하나의 비트 순서가 반전될 수 있다. 예를 들어, 워프 ID(301)와 레지스터 ID(302) 중에서 비트 길이가 짧은 어느 하나(예: 도 9의 예시에서 워프 ID(301))의 비트 순서가 반전될 수 있다. 따라서, 가장 왼쪽에 있던 워프 ID(301)의 최상위 비트가 반전에 의해서 가장 오른쪽으로 이동되고, 가장 오른쪽에 있던 워프 ID(301)의 최하위 비트가 가장 왼쪽으로 이동될 수 있다.

[0102] 다음으로, 워프 ID(301)와 레지스터 ID(302) 중 비트 순서가 반전된 어느 하나의 최상위 비트와 다른 하나의 최하위 비트가 정렬될 수 있다. 다시 말해, 워프 ID(301)와 레지스터 ID(302) 중 어느 하나의 하위 비트일수록 다른 하나의 상위 비트와 정렬될 수 있다.

[0103] 예를 들어, 도 9를 참조하면, 비트 순서가 반전된 워프 ID(301)의 최상위 비트와 레지스터 ID(302)의 최하위 비트가 정렬될 수 있다. 예를 들어, 워프 ID(301)의 최하위 비트인 '0' 비트와 레지스터 ID(302)의 최상위 비트인 '7' 비트가 정렬될 수 있다.

[0104] 또한, 워프 ID(301)의 하위 비트일수록 레지스터 ID(302)의 상위비트와 정렬될 수 있다. 예를 들어, 워프 ID(301)는 상위 비트인 '5' 비트에서 하위 비트인 '0' 비트로 갈수록, 레지스터 ID(302)의 하위 비트인 '2' 비트에서 상위 비트인 '7' 비트와 정렬될 수 있다. 결국, 워프 ID(301)의 '1' 비트와 레지스터 ID(302)의 '6' 비트가 정렬되고, 워프 ID(301)의 '2' 비트와 레지스터 ID(302)의 '5' 비트가 정렬되고, 워프 ID(301)의 '3' 비트와 레지스터 ID(302)의 '4' 비트가 정렬되고, 워프 ID(301)의 '4' 비트와 레지스터 ID(302)의 '3' 비트가 정렬되고, 워프 ID(301)의 '5' 비트와 레지스터 ID(302)의 '2' 비트가 정렬될 수 있다.

[0105] 그 다음, 정렬된 워프 ID(301)와 레지스터 ID(302) 간 논리 연산이 수행되어 명령어 연산의 결과를 저장하는 레지스터 캐시를 나타내는 레지스터 캐시 인덱스(예: 도 3의 레지스터 캐시 인덱스(303))가 결정될 수 있다.

[0106] 도 9를 살피면, 반전된 워프 ID(301)와 레지스터 ID(302) 간 XOR 게이트를 이용한 논리 연산이 도시되어 있다.

[0107] 논리 연산은 연산 결과가 제1 값일 확률과 연산 결과가 제2 값일 확률이 동일한 연산이거나, 두 확률의 차이가 미리 정해진 범위 내인 연산일 수 있다. 즉, 연산 결과가 0 또는 1일 확률이 동일한 연산 일 수 있다. 따라서, 논리 연산은 연산 결과가 0 또는 1일 확률이 동일한 XOR 게이트를 이용한 연산일 수 있다. 다만, 논리 연산은 하나의 게이트로 이루어진 XOR 게이트뿐만 아니라 여러 논리 게이트를 적절히 결합하여 연산 결과가 0 또는 1일 확률이 동일하거나, 두 확률의 차이가 미리 정해진 범위 내인 논리 연산일 수 있다.

[0108] 논리 연산의 결과는 캐시 인덱스(302)의 대응하는 비트 위치로 전달될 수 있다. 이때, 워프 ID(301)와 레지스터 ID(302) 중 비트 길이가 긴 어느 하나의 비트 위치가 고려될 수 있다. 도 9에 도시된 예시에서, 워프 ID(301)의 '0' 비트와 레지스터 ID(302)의 '7' 비트 간 논리 연산 결과는 레지스터 ID(302)의 비트 위치에 따라 캐시 인덱스(302)의 최상위 비트로 전달될 수 있다.

[0109] 레지스터 ID(302)가 워프 ID(301)보다 비트 길이가 길기 때문에, 레지스터 ID(302) 중 일부는 워프 ID(301)와

짝을 이루지 않을 수 있다. 이러한 레지스터 ID(302)의 비트(예: '0', '1' 비트)는 논리 연산 없이 그대로 캐시 인덱스(302)에 포함될 수 있다. 이때도 마찬가지로, 레지스터 ID(302)의 비트 위치에 따라 워프 ID(301)와 짝을 이루지 않는 레지스터 ID(302)의 일부 비트가 캐시 인덱스(302)의 대응하는 비트로 전달될 수 있다.

[0110] 도 9에서는 설명의 편의를 위해 레지스터 ID(302)가 워프 ID(301)보다 비트 길이가 긴 예시가 도시되어 있으나, 실시예가 이로 한정되는 것은 아니고, 레지스터 ID(302)가 워프 ID(301)보다 비트 길이가 짧을 수도 있다. 또한, 실시예에 따라서는 레지스터 ID(302)가 반전되어 워프 ID(301)와 정렬될 수도 있다. 이 경우에도 앞서 설명한 사항들이 마찬가지로 적용될 수 있으므로, 보다 상세한 사항은 생략한다.

[0111] 결론적으로, 본 발명에 의하면 스레드(즉, 워프)가 레지스터를 많이 사용하는 애플리케이션의 경우, 레지스터 ID의 비트가 워프 ID의 비트보다 레지스터 캐시 인덱스의 계산에 더 큰 영향을 미칠 수 있다. 반대로, 스레드가 레지스터를 많이 사용하지 않는 애플리케이션의 경우, 워프 ID의 비트가 레지스터 ID의 비트보다 레지스터 캐시 인덱스의 계산에 더 큰 영향을 미칠 수 있다.

[0112] 따라서, 이러한 특성을 고려한 본 발명의 방법에 따르면 컨텐션이 줄어 들어 라이트 백 연산이 감소할 수 있다.

[0113]

[0114] 한편, 본 발명에 따른 방법은 컴퓨터에서 실행될 수 있는 프로그램으로 작성되어 마그네틱 저장매체, 광학적 판독매체, 디지털 저장매체 등 다양한 기록 매체로도 구현될 수 있다.

[0115] 본 명세서에 설명된 각종 기술들의 구현들은 디지털 전자 회로조직으로, 또는 컴퓨터 하드웨어, 펌웨어, 소프트웨어로, 또는 그들의 조합들로 구현될 수 있다. 구현들은 데이터 처리 장치, 예를 들어 프로그램가능 프로세서, 컴퓨터, 또는 다수의 컴퓨터들의 동작에 의한 처리를 위해, 또는 이 동작을 제어하기 위해, 컴퓨터 프로그램 제품, 즉 정보 캐리어, 예를 들어 기계 판독가능 저장 장치(컴퓨터 판독가능 매체) 또는 전파 신호에서 유형적으로 구체화된 컴퓨터 프로그램으로서 구현될 수 있다. 상술한 컴퓨터 프로그램(들)과 같은 컴퓨터 프로그램은 컴파일된 또는 인터프리트된 언어들을 포함하는 임의의 형태의 프로그래밍 언어로 기록될 수 있고, 독립형 프로그램으로서 또는 모듈, 구성요소, 서브루틴, 또는 컴퓨팅 환경에서의 사용에 적절한 다른 유닛으로서 포함하는 임의의 형태로 전개될 수 있다. 컴퓨터 프로그램은 하나의 사이트에서 하나의 컴퓨터 또는 다수의 컴퓨터들 상에서 처리되도록 또는 다수의 사이트들에 걸쳐 분배되고 통신 네트워크에 의해 상호 연결되도록 전개될 수 있다.

[0116] 컴퓨터 프로그램의 처리에 적절한 프로세서들은 예로서, 범용 및 특수 목적 마이크로프로세서들 둘 다, 및 임의의 종류의 디지털 컴퓨터의 임의의 하나 이상의 프로세서들을 포함한다. 일반적으로, 프로세서는 판독 전용 메모리 또는 랜덤 액세스 메모리 또는 둘 다로부터 명령어들 및 데이터를 수신할 것이다. 컴퓨터의 요소들은 명령어들을 실행하는 적어도 하나의 프로세서 및 명령어들 및 데이터를 저장하는 하나 이상의 메모리 장치들을 포함할 수 있다. 일반적으로, 컴퓨터는 데이터를 저장하는 하나 이상의 대량 저장 장치들, 예를 들어 자기, 자기-광 디스크들, 또는 광 디스크들을 포함할 수 있거나, 이것들로부터 데이터를 수신하거나 이것들에 데이터를 송신하거나 또는 양쪽으로 되도록 결합될 수도 있다. 컴퓨터 프로그램 명령어들 및 데이터를 구체화하는데 적절한 정보 캐리어들은 예로서 반도체 메모리 장치들, 예를 들어, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(Magnetic Media), CD-ROM(Compact Disk Read Only Memory), DVD(Digital Video Disk)와 같은 광 기록 매체(Optical Media), 플롭티컬 디스크(Floptical Disk)와 같은 자기-광 매체(Magneto-Optical Media), 롬(ROM, Read Only Memory), 램(RAM, Random Access Memory), 플래시 메모리, EPROM(Erasable Programmable ROM), EEPROM(Electrically Erasable Programmable ROM) 등을 포함한다. 프로세서 및 메모리는 특수 목적 논리 회로조직에 의해 보충되거나, 이에 포함될 수 있다.

[0117] 또한, 컴퓨터 판독가능 매체는 컴퓨터에 의해 액세스될 수 있는 임의의 가용매체일 수 있고, 컴퓨터 저장매체 및 전송매체를 모두 포함할 수 있다.

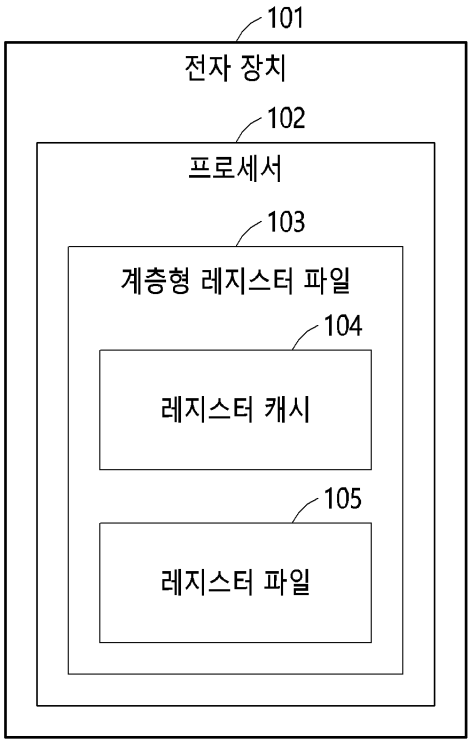
[0118] 본 명세서는 다수의 특정한 구현물의 세부사항들을 포함하지만, 이들은 어떠한 발명이나 청구 가능한 것의 범위에 대해서도 제한적인 것으로서 이해되어서는 안되며, 오히려 특정한 발명의 특정한 실시형태에 특유할 수 있는 특징들에 대한 설명으로서 이해되어야 한다. 개별적인 실시형태의 문맥에서 본 명세서에 기술된 특정한 특징들은 단일 실시형태에서 조합하여 구현될 수도 있다. 반대로, 단일 실시형태의 문맥에서 기술한 다양한 특징들 역시 개별적으로 혹은 어떠한 적절한 하위 조합으로도 복수의 실시형태에서 구현 가능하다. 나아가, 특징들이 특정한 조합으로 동작하고 초기에 그와 같이 청구된 바와 같이 묘사될 수 있지만, 청구된 조합으로부터의 하나 이상의 특징들은 일부 경우에 그 조합으로부터 배제될 수 있으며, 그 청구된 조합은 하위 조합이나 하위 조합의 변형물로 변경될 수 있다.

[0119] 마찬가지로, 특정한 순서로 도면에서 동작들을 묘사하고 있지만, 이는 바람직한 결과를 얻기 위하여 도시된 그 특정한 순서나 순차적인 순서대로 그러한 동작들을 수행하여야 한다거나 모든 도시된 동작들이 수행되어야 하는 것으로 이해되어서는 안 된다. 특정한 경우, 멀티태스킹과 병렬 프로세싱이 유리할 수 있다. 또한, 상술한 실시형태의 다양한 장치 컴포넌트의 분리는 그러한 분리를 모든 실시형태에서 요구하는 것으로 이해되어서는 안되며, 설명한 프로그램 컴포넌트와 장치들은 일반적으로 단일의 소프트웨어 제품으로 함께 통합되거나 다중 소프트웨어 제품에 패키징 될 수 있다는 점을 이해하여야 한다.

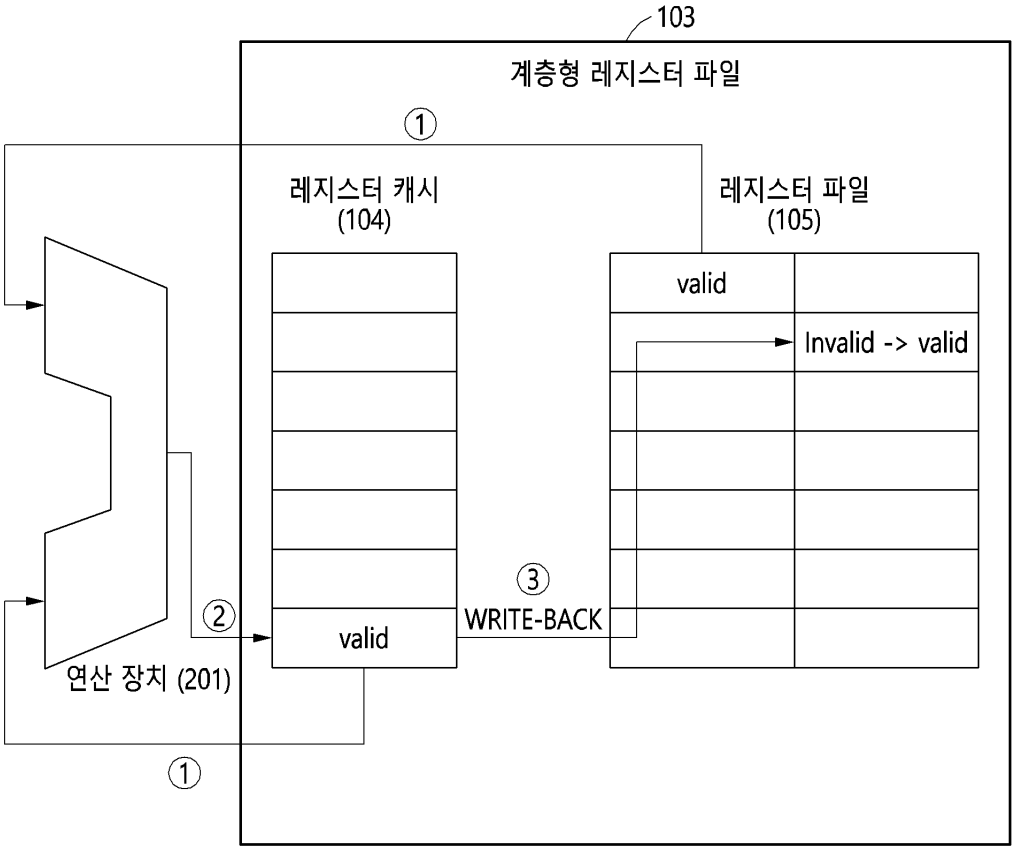
[0120] 한편, 본 명세서와 도면에 개시된 본 발명의 실시 예들은 이해를 돕기 위해 특정 예를 제시한 것에 지나지 않으며, 본 발명의 범위를 한정하고자 하는 것은 아니다. 여기에 개시된 실시 예들 이외에도 본 발명의 기술적 사상에 바탕을 둔 다른 변형 예들이 실시 가능하다는 것은, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 자명한 것이다.

도면

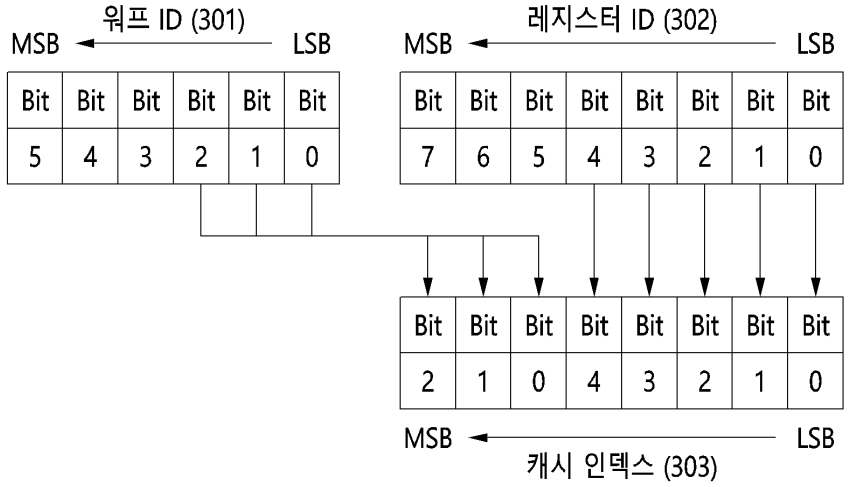
도면1



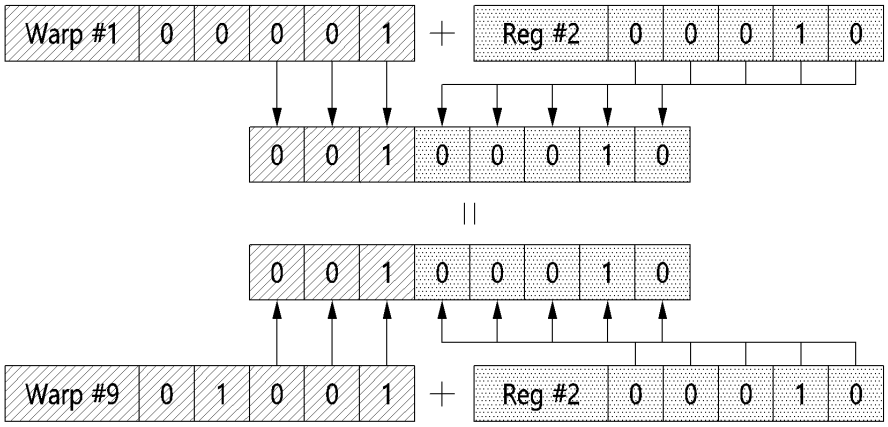
도면2


103
계층형 레지스터 파일
①
레지스터 캐시
(104)
레지스터 파일
(105)
valid
Invalid -> valid
③
WRITE-BACK
②
valid
연산 장치 (201)
①

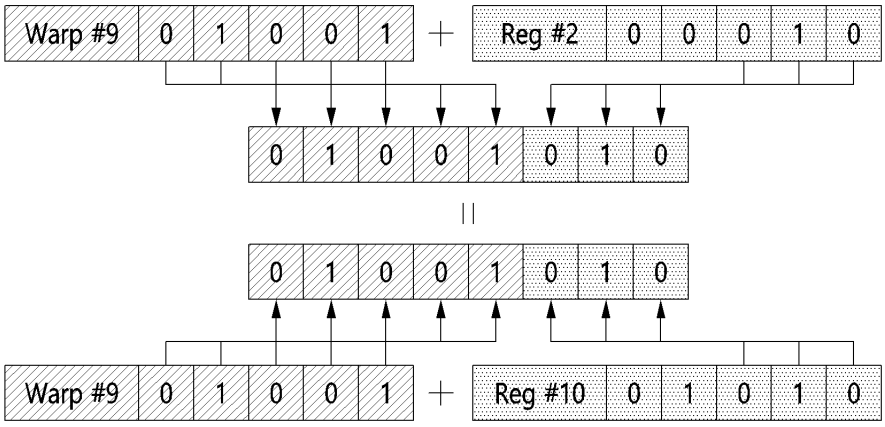
도면3


워프 ID (301)
MSB
LSB
Bit 5 4 3 2 1 0
레지스터 ID (302)
MSB
LSB
Bit 7 6 5 4 3 2 1 0
Bit 2 1 0 4 3 2 1 0
MSB
LSB
캐시 인덱스 (303)

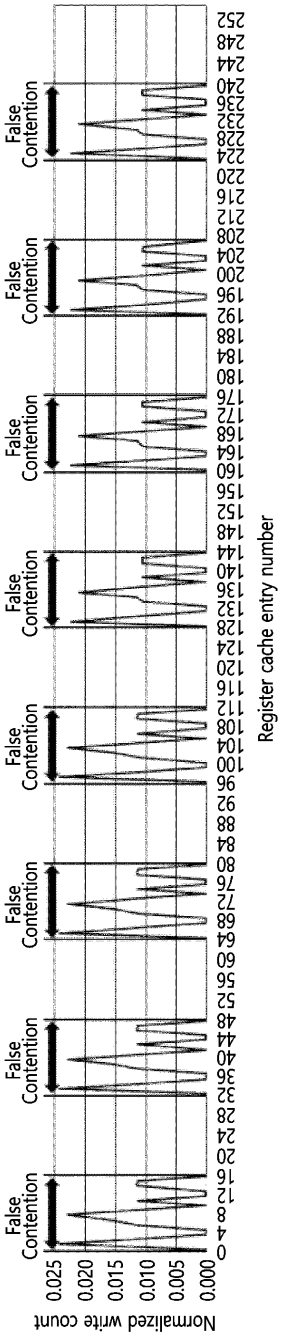
도면4


Warp #1
0 0 0 0 1
+
Reg #2
0 0 0 1 0
0 0 1 0 0 0 1 0
||
0 0 1 0 0 0 1 0
Warp #9
0 1 0 0 1
+
Reg #2
0 0 0 1 0

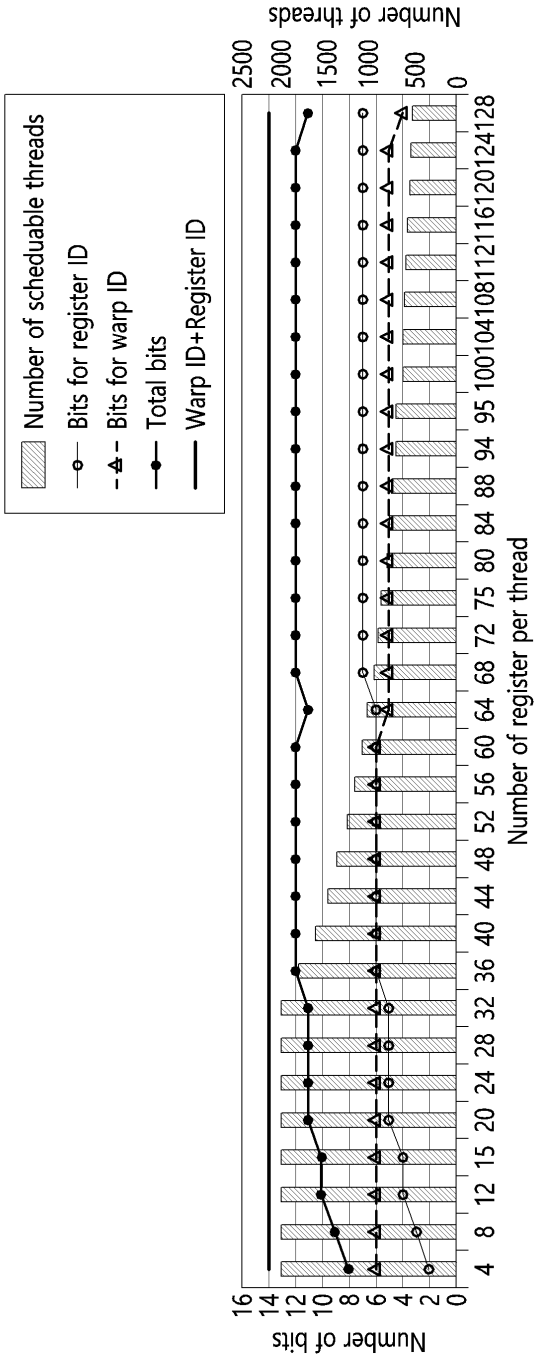
도면5


Warp #9
0 1 0 0 1
+
Reg #2
0 0 0 1 0
0 1 0 0 1 0 1 0
||
0 1 0 0 1 0 1 0
Warp #9
0 1 0 0 1
+
Reg #10
0 1 0 1 0

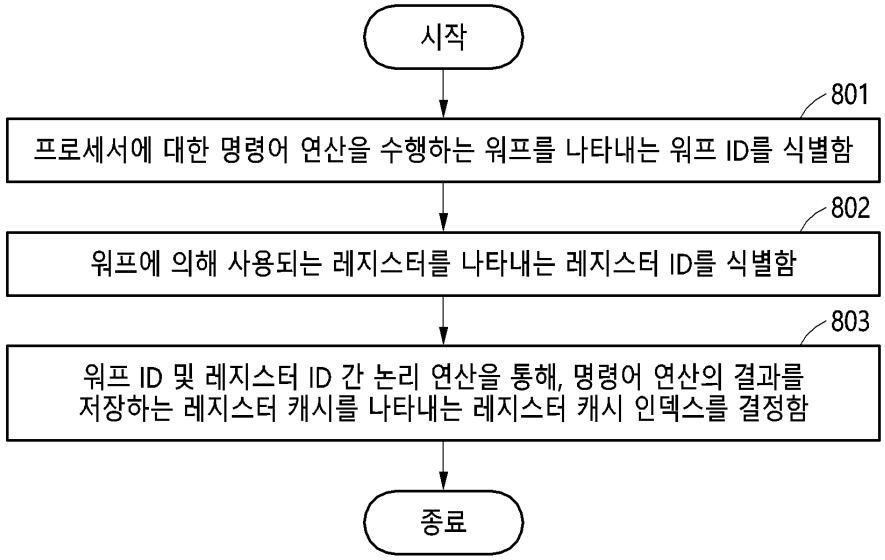
도면6


Normalized write count
0.025
0.020
0.015
0.010
0.005
0.000
Register cache entry number
0
4
8
12
16
20
24
28
32
36
40
44
48
52
56
60
64
68
72
76
80
84
88
92
96
100
104
108
112
116
120
124
128
132
136
140
144
148
152
156
160
164
168
172
176
180
184
188
192
196
200
204
208
212
216
220
224
228
232
236
240
244
248
252
False Contention
False Contention
False Contention
False Contention
False Contention
False Contention
False Contention
False Contention

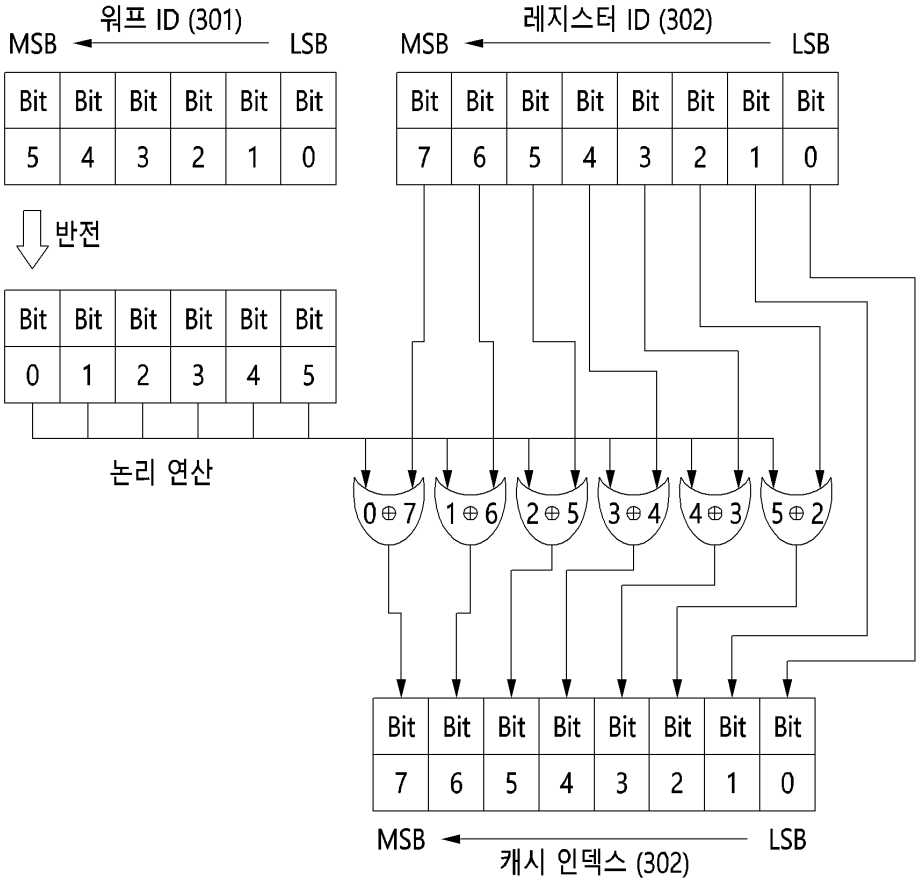
도면7


Number of scheduable threads
Bits for register ID
Bits for warp ID
Total bits
Warp ID+Register ID
Number of threads
2500
2000
1500
1000
500
0
Number of bits
16
14
12
10
8
6
4
2
0
4 8 12 15 20 24 28 32 36 40 44 48 52 56 60 64 68 72 75 80 84 88 94 95 100 104 108 112 116 120 124 128
Number of register per thread

도면8


시작
801
프로세서에 대한 명령어 연산을 수행하는 워프를 나타내는 워프 ID를 식별함
802
워프에 의해 사용되는 레지스터를 나타내는 레지스터 ID를 식별함
803
워프 ID 및 레지스터 ID 간 논리 연산을 통해, 명령어 연산의 결과를 저장하는 레지스터 캐시를 나타내는 레지스터 캐시 인덱스를 결정함
종료

도면9


워프 ID (301)
MSB
LSB
Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 Bit 0
반전
Bit 0 Bit 1 Bit 2 Bit 3 Bit 4 Bit 5
논리 연산
레지스터 ID (302)
MSB
LSB
Bit 7 Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 Bit 0
0⊕7
1⊕6
2⊕5
3⊕4
4⊕3
5⊕2
Bit 7 Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 Bit 0
MSB
LSB
캐시 인덱스 (302)