(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2008년09월03일
(11) 등록번호 10-0856605
(24) 등록일자 2008년08월28일

(51) Int. Cl.
G06F 9/46 (2006.01) *G06F 1/32* (2006.01)
(21) 출원번호 10-2006-7000942
(22) 출원일자 2006년01월13일
심사청구일자 2006년01월13일
번역문제출일자 2006년01월13일
(65) 공개번호 10-2006-0031868
(43) 공개일자 2006년04월13일
(86) 국제출원번호 PCT/US2004/022354
국제출원일자 2004년07월14일
(87) 국제공개번호 WO 2005/010737
국제공개일자 2005년02월03일
(30) 우선권주장
10/621,228 2003년07월15일 미국(US)
(56) 선행기술조사문헌
US5968167(1999.10.19)
전체 청구항 수 : 총 17 항

(73) 특허권자
인텔 코오퍼레이션
미합중국 캘리포니아 산타클라라 미션 칼리지 블러바드 2200
(72) 발명자
베일리, 다니엘
미국 01532 매사추세츠주 노쓰보로우 데이비스 스트리트 327
더튼, 토드
미국 01772 매사추세츠주 사우스보로우 메타케밋 레인 3
포썸, 트리그브
미국 01532 매사추세츠주 노쓰보로우 인크리즈 워드 드라이브 19
(74) 대리인
백만기, 이중희, 주성민

심사관 : 안철용

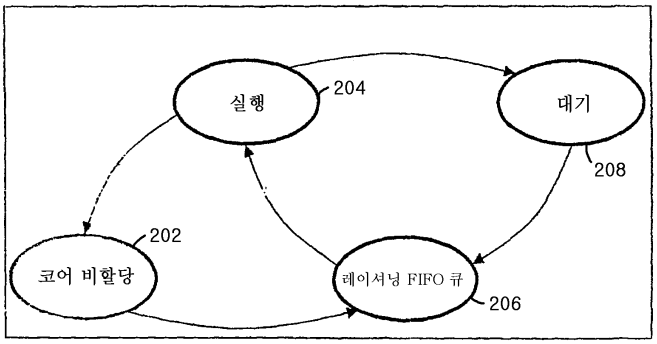
(54) 다중 코어 프로세서 성능을 개선하기 위한 방법, 시스템 및장치

(57) 요 약

다중 코어 프로세서의 코어들이 다양한 전력 및 열 제약 조건들을 고수할 수 있도록 하기 위한 코어 레이셔닝 로직을 위한 시스템, 장치 및 방법이 제공된다.

대표도 - 도1

스레드가 완료되어 코어 비할당 큐로 다시 전환될 때까지의 나중의 상태들



도 1. 각 코어에 대한 CRSO 상태도

특허청구의 범위

청구항 1

복수의 프로세서 코어들 중 적어도 하나의 프로세서 코어에 대해 클록들을 디스에이블(disable)하기 위한 방법으로서,

적어도 부분적으로 작업 부하량에 기초하여 실행 코어 한계(executing core limit)를 계산하는 단계;

m개의 프로세서 코어들을 인에이블(enable)하는 단계;

n개의 이용 가능한 스레드(thread)들을 실행하는 단계; 및

상기 프로세서 코어가 메모리 동작을 기다리는 유휴(idle) 기간 동안 적어도 하나의 프로세서에 대해 클록들을 디스에이블하는 단계

를 포함하고,

n은 정수이고, m은 n개의 이용 가능한 스레드들의 개수 이하의 정수인, 방법.

청구항 2

삭제

청구항 3

제1항에 있어서,

적어도 하나의 프로세서 코어에 대해 클록들을 디스에이블하는 단계는 전력 소비를 감소시키는 방법.

청구항 4

제1항에 있어서,

적어도 하나의 프로세서 코어에 대해 클록들을 디스에이블하는 단계는 해당 프로세서 코어의 동작 주파수를 증가시키는 방법.

청구항 5

삭제

청구항 6

삭제

청구항 7

삭제

청구항 8

삭제

청구항 9

삭제

청구항 10

삭제

청구항 11

삭제

청구항 12

삭제

청구항 13

삭제

청구항 14

삭제

청구항 15

삭제

청구항 16

삭제

청구항 17

삭제

청구항 18

삭제

청구항 19

삭제

청구항 20

삭제

청구항 21

삭제

청구항 22

복수의 프로세스 코어 중 제1 프로세스 코어를 실행 큐에 위치시켜 상기 프로세스 코어에 할당되는 스레드의 실행을 위한 인에이블을 대기하도록 하는 단계;

코어 각각에 할당되는 스레드를 실행하고 있는 상기 복수의 프로세스 코어로부터 코어들의 수를 결정하는 단계;

스레드를 실행하고 있는 상기 복수의 프로세스 코어로부터의 상기 코어들의 수가 실행 코어 한계보다 작은지 결정하는 단계;

스레드를 실행하고 있는 상기 복수의 프로세스 코어로부터의 상기 수가 상기 실행 코어 한계보다 작고, 상기 제1 프로세스 코어가 상기 실행 큐로부터 다음에 실행될 경우, 상기 제1 프로세스 코어를 인에이블하고, 상기 제1 프로세스 코어를 상기 실행 큐에서 제거하는 단계

를 포함하는 방법.

청구항 23

제22항에 있어서,

상기 실행 코어 한계는 식 N x (E/(E+M))에 적어도 부분적으로 기초하며, N은 컨텍스트를 갖는 스레드들의 개수를 표현하고, E는 실행 시간의 백분율을 표현하며, M은 메모리 참조 시간의 백분율을 표현하고, 상기 식의 결과는 가장 가까운 정수로 버림되어(rounded down) 상기 실행 코어 한계를 나타내는 방법.

청구항 24

요청들 및 응답들을 수신하기 위해 적어도 두 개의 방향성 버스들에 연결되고, 캐시 메모리에 연결된 적어도 하나의 다중 코어 프로세서; 및

메모리 동작이 완료되기를 기다리는 코어들의 클록을 디스에이블하고, 상기 메모리 동작이 완료되고 현재 실행되고 있는 코어들의 수가 실행 코어 한계 이하가 될 때까지 클록을 인에이블하지 않음으로써, 인에이블된 코어들의 개수가 상기 실행 코어 한계 이하가 되도록 관리하기 위한 코어 레이셔닝 로직(core rationing logic)

을 포함하고,

상기 코어 한계는 컨텍스트를 갖는 스레드들의 수, 실행 시간 및 메모리 참조 시간에 기초하는 시스템.

청구항 25

제24항에 있어서,

상기 실행 코어 한계는, N은 컨텍스트를 갖는 스레드들의 개수를 표현하고, %E는 실행 시간의 백분율을 표현하고, %M은 메모리 참조 시간의 백분율을 표현하는 식,

int(N x (%E/(%E+%M)))

에 적어도 부분적으로 기초하는 시스템.

청구항 26

제24항에 있어서,

메모리 DIMM들을 위한 복수의 메모리 컨트롤러;

다른 프로세서 다이들 또는 I/O 서브시스템들에 대한 상호연결 링크들을 관리하기 위한 라우터 로직; 및

상기 코어 레이셔닝 로직

을 포함하는 시스템 인터페이스를 더 포함하는 시스템.

청구항 27

제24항에 있어서,

요청들 및 응답들을 수신하기 위한 적어도 두 개의 반시계 방향 버스들을 더 포함하는 시스템.

청구항 28

제24항에 있어서,

상기 적어도 두 개의 방향성 버스들은 적어도 두 개의 시계 방향 버스들을 포함하는 시스템.

청구항 29

복수의 프로세서 코어 - 임의의 순간에 실행될 수 있는 상기 프로세서 코어들의 수는 실행 코어 한계 이하임 -; 및

메모리 트랜잭션이 완료되기를 기다리고 있는 상기 복수의 프로세서 코어 중에서 하나의 프로세서 코어를 디스에이블하고, 실행되고 있는 코어들의 총 수가 실행 코어 한계 이하인 경우 메모리 트랜잭션 상에서 기다리고 있지 않은 하나의 프로세서 코어를 인에이블하기 위한 코어 레이셔닝 로직

을 포함하는 장치.

청구항 30

제29항에 있어서,

상기 실행 코어 한계는, N은 컨텍스트를 갖는 스레드들의 개수를 표현하고, %E는 실행 시간의 백분율을 표현하고, %M은 메모리 참조 시간의 백분율을 표현하는 식,

int(N x (%E/(%E+%M)))

에 적어도 부분적으로 기초하는 장치.

청구항 31

제29항에 있어서,

요청들 및 응답들을 수신하기 위한 적어도 두 개의 반시계 방향 버스들을 더 포함하는 장치.

청구항 32

제29항에 있어서,

적어도 두 개의 시계 방향 버스들을 포함하는 적어도 두 개의 방향성 버스들을 더 포함하는 장치.

청구항 33

복수의 프로세스 코어에 대한 실행 코어 한계를 계산하는 단계 - 상기 실행 코어 한계는 상기 프로세스 코어들 상에서 실행 가능하도록 준비된 스레드들의 수 이하인 정수 값임 -

상기 실행 코어 한계에 의해 특정되는 수의 실행되는 프로세서 코어들 및 상기 복수의 실행 코어 중 실행되지 않는 나머지 코어들을 갖도록 전압-주파수 동작점을 선택하는 단계;

메모리 동작이 완료되기를 기다리고 있는 임의의 프로세스 코어로의 클록을 디스에이블하는 단계; 및

상기 메모리 동작이 완료되면, 현재 실행되는 코어들의 수가 상기 실행 코어 한계 이하인 경우 상기 프로세스 코어로의 클록을 인에이블하는 단계

를 포함하는 방법.

청구항 34

제33항에 있어서,

상기 실행 코어 한계는, N은 컨텍스트를 갖는 스레드들의 개수를 표현하고, %E는 실행 시간의 백분율을 표현하고, %M은 메모리 참조 시간의 백분율을 표현하는 식,

int(N x (%E/(%E+%M)))

에 적어도 부분적으로 기초하는 방법.

청구항 35

제33항에 있어서,

상기 실행 코어 한계를 다시 계산하는 단계;

상기 실행 코어 한계가 감소하는 경우 상기 전압-주파수 동작점을 증가시키는 단계;

상기 실행 코어 한계가 증가하는 경우, 상기 전압-주파수 동작점을 감소시키는 단계

를 더 포함하는 방법.

명 세 서

기 술 분 야

<1> 본 명세서는 전원 관리 분야에 속한다. 더욱 자세하게는, 본 명세서는 전력 제약 조건들에도 불구하고 다중 코어 프로세서(multi-core processor) 성능을 개선하기 위한 새로운 방법 및 장치에 속한다.

배 경 기 술

<2> 전원 관리 기법들은 서버들, 랩톱 컴퓨터들, 프로세서들 및 데스크톱 컴퓨터들과 같은 다양한 유형의 시스템들 및 집적 디바이스들(integrated devices)을 위한 저전력 애플리케이션들을 달성하기 위해 전력 소비를 감소시키

는 것을 고려한다. 통상적으로, CPU(Central Processing Unit) 활동에 적어도 부분적으로 근거하여 성능을 최적화하기 위해 다수의 전원 상태들을 지원하기 위한 시스템들 및 집적 디바이스들에 소프트웨어 방법들이 채용된다.

<3> 현재의 전원 관리 기법들은 전력 소비를 감소시키기 위해 전압과 주파수 중 하나를 감소시키거나 둘 다를 감소시킨다. 그러나 이것은 전체 성능을 감소시킨다. 또한, 몇몇 방법들은 튜닝, 캘리브레이션 및 일시적인 작업 부하량에 대한 루프 안정성(loop stability)에 관련된 다양한 도전들을 받는 아날로그 설계들을 포함한다.

<4> 고 주파수들 및 고전압들에서 동작하는 코어들의 증가 및 열적 한계들, 최대 전류 및 Vcc 범위와 같은 다양한 전력 제약 조건들을 고수해야할 필요 때문에, 다수의 코어들을 갖는 프로세서들의 도입에 있어서, 전원 관리는 주된 관심이 된다.

발명의 상세한 설명

<10> 다음의 설명은 전력 제약 조건들에도 불구하고 개선된 다중 코어 프로세서 성능을 위한 방법 및 장치를 제공한다. 다음의 설명에서, 본 발명의 보다 완전한 이해를 제공하기 위해 수많은 특정한 세부사항들이 설명된다. 그러나 본 기술분야의 당업자에 의해서, 그러한 특정한 세부사항들 없이도 본 발명이 실시될 수 있다는 것이 이해될 것이다. 본 기술분야의 당업자들은 포함된 설명들을 가지고, 과도한 실험없이 적절한 논리회로들을 구현할 수 있을 것이다.

<11> 전술된 바와 같이, 전력 제약 조건들을 고수하면서 프로세서 성능을 개선하는데는 문제가 있다. 현재의 방법들은 전체 성능을 희생시켜 전압 또는 주파수를 낮추는 것을 포함한다. 대조적으로, 본 발명은 전력 제약 조건들을 고수하면서 전체 성능을 개선한다. 예를 들면, "프로세서 시스템에 대한 실행 코어들의 개수의 레이셔닝(rationing)"이라는 개념은 메모리 트랜잭션이 완료하기를 기다릴 때와 같은 유휴(idle) 상태인 코어들에 대해 클록들을 디스에이블하는 결과로서 주파수를 증가시키는 것을 고려한다. 예를 들면, 본 발명은 프로세서 코어에 대해 클록들을 디스에이블함으로써 그 코어들의 유휴 기간을 이용하여 전력 낭비를 감소시킨다. 따라서, 전력 낭비에서의 감소의 결과, 보다 높은 주파수가 이용될 수 있다. 일 실시예에서, 적절한 실행 코어 한계(executing core limit)는 작업 부하량에 대해 계산된다. 또한, 동일한 실시예에서, 실행 코어들의 개수는 이용 가능하고 스레드(thread)들에 대해 준비되어 있는 개수 이하이다. 스레드는 특정한 적용에 대한 명령들의 독립적인 세트이다.

<12> 일 실시예에서, 본 발명은 모든 코어들에서 집합적으로 진행되는 스레드들의 활동 레벨의 예측에 근거하여 전압/주파수 동작점을 선택하는 것을 돕는다. 예를 들어, TPC-C 스레드들은 시간의 50-60% 정도 활동적이고, 그들 시간의 40-50%를 메모리 참조들이 완료되기를 기다리면서 유휴상태로 보낸다. 그러한 환경에서, 실행 코어 한계를, 일 실시예에서, 다이에 있는 코어의 총 개수의 60%가 되도록 특정할 수 있다; 만약 8개의 코어가 있다면, 실행 코어 한계를, 이 경우에서는, 5개로 설정할 수 있다. 그 후 동시에 오직 5개의 코어들이 활동적이고 3개의 코어들이 비활동적인(저전원 상태인) 것에 해당하는 전압-주파수 동작점을 특정할 수 있고, 이것은 8개의 코어 모두를 동시에 활동적이게 특정하는 것보다 훨씬 더 높은 동작 주파수이다. 코어 레이셔닝 로직은 다이의 동작들을 제한하여, (이 경우에서는) 5개보다 많은 코어들이 임의의 소정의 순간에 활동적이지 않는 것을 보장한다. (도 1과 관련하여 더 논의될) 대기 및 레이셔닝 큐들의 점유에 관하여 통계치들이 수집된다; 휴지기들에서 이러한 통계치들은 동작점(실행 코어 한계 및 그와 관련된 전압/주파수 쌍)이 변경되어야 하는지를 결정하기 위해 분석된다. 만약 대기 큐가 비어있고 레이셔닝 큐가 채워져 있는 경향이 있으면, 이는, 코어들이 진행될 수 있을 때 진행하고 있지 않고, 성능을 개선하기 위해서 실행 코어 한계가 상승되고 전압/주파수는 감소되어야 한다는 표시이다. 반대로, 레이셔닝 큐가 비어 있고 대기 큐가 채워져 있는 경향이 있으면, 이것은 실행 코어 한계를 감소시키고 전압/주파수 포인트를 증가시킴으로써 성능을 개선할 수 있다는 표시일 수 있다.

<13> 도 1은 일 실시예에 따라 이용된 방법의 흐름도를 예시한다. 일 실시예에서, 흐름도는 상태도(state diagram)를 위한 방법을 표현한다.

<14> 동일한 실시예에서, 상태도는 시스템의 프로세서 코어에 대한 소정의 상태 머신(state machine)을 예시한다. 이 동일한 실시예에서, 상태 머신은 "코어들의 레이셔닝"을 용이하게 하여 메모리 트랜잭션이 완료하기를 기다리고 있는 코어들에 대한 클록들을 디스에이블한 결과로서 프로세서 성능을 개선한다.

<15> 일 실시예에서, 상태도는 코어 비할당 상태(202), 실행 상태(204), 레이셔닝 FIFO 큐 상태(206), 및 대기 상태(208)와 같은 네 개의 정의된 상태들을 갖는다. 처음에, 코어 비할당 상태는 다음과 같이 각 코어가 할당된 스레드를 갖고 있지 않는 것으로 정의된다. 다음에, 마침내 코어가 그에 할당된 스레드를 갖게 되면, 본 발명은

레이셔닝 FIFO 큐 상태(206)로 전환한다. 일 실시예에서, FIFO는 선입선출(First In First Out)로 정의된다.

<16> 레이셔닝 FIFO 큐 상태로의 전환시, 실행 코어들의 개수와 실행 코어 한계(ECL) 사이의 비교가 판단된다. 일 실시예에서, 프로세서 또는 시스템 사양은 열 전력 고려사항을 준수하기 위해 적절한 실행 코어의 한계를 결정한다. 일 실시예에서, ECL은 본 출원에서 후술된 공식에 의해 결정된다. 만약 실행 코어들의 개수가 ECL보다 적다면, 특정한 코어가, FIFO 큐에서 다음에 진행될 코어였던 경우에 실행 상태(204)로 전환한다. 또는, 코어는 레이셔닝 FIFO 큐(206)에 남아 있는다.

<17> 실행 상태로의 진입시, 코어는 메모리 참조 및 과열 이벤트, 및/또는 공평한 타임아웃(fairness timeout)과 같은 이벤트가 일어나지 않은 한 이 상태로 남아 있는다. 예를 들어, 공평한 타임아웃은 가능한 무한 반복(live lock) 상태를 방지하기 위해 이용될 수 있다. 이와 관련하여, 메모리 참조는 프로세서에 결합된 임의의 캐시 메모리에 있지 않은 특정한 메모리 어드레스(캐시 메모리의 모든 레벨에서의 미스)에 대한 판독 또는 기입 동작을 나타낸다. 따라서, 주메모리에 대한 액세스가 개시된다.

<18> 전술된 바와 같이 이벤트가 일어나면, 코어는 대기 상태(208)로 전환한다. 이벤트 완료시, 코어는 레이셔닝 FIFO 큐 상태로 전환한다. 상태들(204, 206 및 208) 사이의 이러한 일련의 순환은 특정한 스레드가 완료될 때까지 일어난다. 스레드 완료시, 코어는 코어 비할당 상태로 전환한다.

<19> 그러나 본 발명은 상태도의 네 개의 정의된 상태들에 한정되지 않는다. 본 발명은 서로 다른 상태량을 지원한다. 도 1은 단순히 실행 코어들의 개수가 스레드들의 이용 가능한 개수보다 적게 되도록 제한하는 것의 예를 예시한다. 예를 들면, 일 실시예는 다수의 대기 상태를 고려할 수 있다. 또는, 대기 상태들은 또 다른 큐 상태로 대체될 수도 있다. 또한, 상태도들의 다른 실시예들은, 코어들에 대한 다수의 우선순위 레벨들을 허용할 뿐만 아니라, 실행 상태로부터의 탈출을 유발시키는 이벤트의 성질에 따라 서로 다른 대기 큐들(메모리 대기, 열적 대기, ACPI 대기 등)을 가질 수 있다.

<20> 통상적으로, 코어는 메모리 판독 또는 기입 동작을 실행하고, 다음에 그 동작에 따른 동작을 실행한다(예를 들면, 메모리 판독 동작에 의해 반환된 데이터를 활용한다). 다음에, 코어는 메모리 동작이 완료되기를 기다리면서 "정지(stall)"한다. 이런 경우에, 정지되었다는 것을 가리키는 신호를 중앙 코어 레이셔닝 로직(core rationing logic)에 어서트한다; 이것은 코어 레이셔닝 로직에 의해 디스에이블되는 것이 바람직하다는 표시이다. 코어 레이셔닝 로직은 해당 코어를 "자게 함(napping)"으로써 이 신호에 응답한다 - 그것은 코어에 "낮잠" 신호를 어서트(assert)하여 코어가 명령 발행을 막게 하고 그 후 (캐시-코히어런트) 저전력 상태로 전환하게 하는 "수면(nap)" 신호를 코어에 어서트한다. 더구나, 코어 레이셔닝 로직은 그 코어에 대한 식별자를 대기 큐에 넣는다. 메모리 동작이 완료되었을 때, 코어는 "정지" 신호를 디어서트(deassert)한다; 코어 레이셔닝 로직은 그 코어에 대한 식별자를 대기 큐에서 레이셔닝 큐로 옮김으로써 이에 반응한다. 만약 현재의 실행 ("잠들지" 않은) 코어들의 개수가 실행 코어 한계 이하이면, 코어 레이셔닝 로직은 가장 오래된 식별자를 레이셔닝 큐에서 제거하고, 그 코어에 대해 "수면" 신호를 디어서트한다.

<21> 도 2는 일 실시예에 따라 이용된 막대 그래프를 예시한다. 일 실시예에서, 막대 그래프는 다양한 작업 부하량에 대해 몬테카를로 시뮬레이션(Monte Carlo simulation)에 의해 계산된 16-코어 멀티프로세서의 실행에 소비된 시간의 백분율을 표시한다. 독립된 축은 2, 4, 6, 8, 10, 12, 14 및 16에 대한 ECL을 예시한다. 또한, (실행 시간에 대해) 1%, 30%, 40%, 및 50%의 메모리 참조 듀티 사이클(memory reference duty cycle)로 시뮬레이션된 서로 다른 작업 부하량에서의 각 ECL에 대한 막대들이 있다.

<22> 50%의 메모리 참조 듀티 사이클을 분석하는 것은 실행 시간의 백분율이 50%에서 포화한다는 사실을 강조한다. 따라서, ECL이 이용 가능한 스레드들의 수와 같을 때 메모리 참조들의 처리는 실행 시간의 반을 소비한다.

<23> 도 3은 일 실시예에 따라 이용된 막대 그래프를 예시한다. 도 2에 더하여서, 도 3은 실행 시간의 백분율과 주파수의 곱으로써 계산된 전체 성능을 예시한다. 전체 성능은 주파수가 ECL과 반비례한다는 사실도 포함한다. 전술된 바와 같이, 실행 코어들의 개수가 줄어들면, 전력 낭비를 감소시키기 때문에 이러한 관계가 존재한다. 따라서, 정상 상태 열 한계에 남아있도록 주파수가 증가될 수 있다.

<24> 또한, 도 3은 30%의 메모리 참조 듀티 사이클에 대한 실행 시간의 최대 백분율이 70%임을 표현한다. 또한, 포화 한계와 스레드들의 개수의 곱은 포화의 시작의 경계를 정한다. 이것이 개선된 또는 최적의 성능을 위한 영역일 수 있기 때문에 포화의 시작은 특히 중요하다.

<25> 일 실시예에서, 적절한 ECL을 결정하기 위해 자가 최적화 식이 이용된다. 이 식에서, N은 컨텍스트(context)를

갖는 스레드들의 개수를 표현한다: %E는 실행 시간의 백분율을 표현한다: 그리고 %M은 메모리 참조 시간의 백분율을 표현한다. 그 식은

<26> int(Nx(%E/(%E+%M))) 이다.

<27> 도 4는 일 실시예에 따른 장치를 표현한다. 일 실시예에서, 장치는 레벨 3(L3) 캐시 메모리의 독립적인 뱅크에 개별적으로 연결된 복수의 프로세서들(410)을 갖는 다중 코어 프로세서 시스템을 표현한다. 동일한 실시예에서, 복수의 4개의 버스들은 2개의 카운터 회전 "링들"(counter rotating rings) - 시계방향으로 요청/응답(REQ0/RSP0) 링(402 및 404), 및 반시계 방향으로 요청/응답 링(REQ1/RSP1)(406 및 408)을 만든다.

<28> "P"들 및 "C"들 사이의 원은 각 링에 대한 상태 디바이스들의 쌍을 표현한다. 따라서, 원형 파이프라인(pipeline)들의 세트는 각 프로세서 코어/캐시 뱅크로부터 임의의 다른 프로세서 코어/캐시 뱅크로 정보를 전달하기 위해 이용된다. 시스템 인터페이스 로직은 메모리 DIMM들을 위한 메모리 컨트롤러들, 다른 프로세서 다이들 및/또는 I/O 서브시스템들에 대한 상호연결 링크들을 다루기 위한 라우터 로직, 및 (중앙 코어 레이셔닝 컨트롤러를 포함하는) 여러 가지 종류의 다른 시스템 제어 로직을 포함한다.

<29> 소정의 예시적인 실시예들이 기술되고 첨부 도면들에서 도시되었지만, 본 명세서를 학습할 때 본 기술분야의 당업자들에게 다양한 다른 변경들이 떠오를 수 있기 때문에 그러한 실시예들은 단순히 예시적인 것이고 넓은 발명에 대해 제한적이지 않다는 것과, 본 발명은 도시되고 설명된 특정한 구성들 및 배열들에 한정되지 않는다는 것이 이해되어야 한다.

도면의 간단한 설명

<5> 첨부 도면들에서 본 발명은 한정이 아닌 예로서 예시된다.

<6> 도 1은 일 실시예에 따라 이용된 방법에 대한 흐름도(flowchart)를 예시한다.

<7> 도 2는 일 실시예에 따라 이용된 막대 그래프를 예시한다.

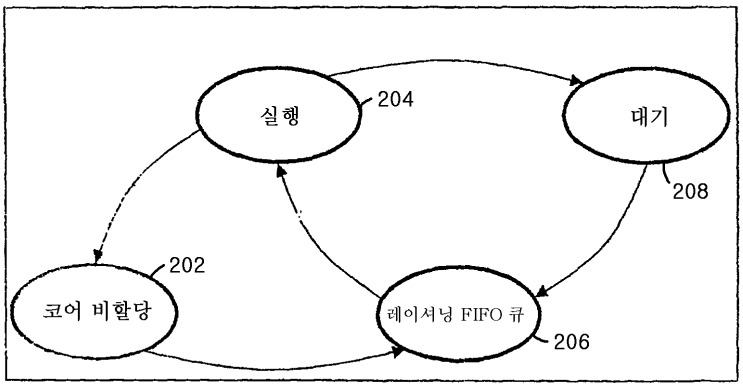
<8> 도 3은 일 실시예에 따라 이용된 막대 그래프를 예시한다.

<9> 도 4는 일 실시예에 따른 장치를 예시한다.

도면

도면1

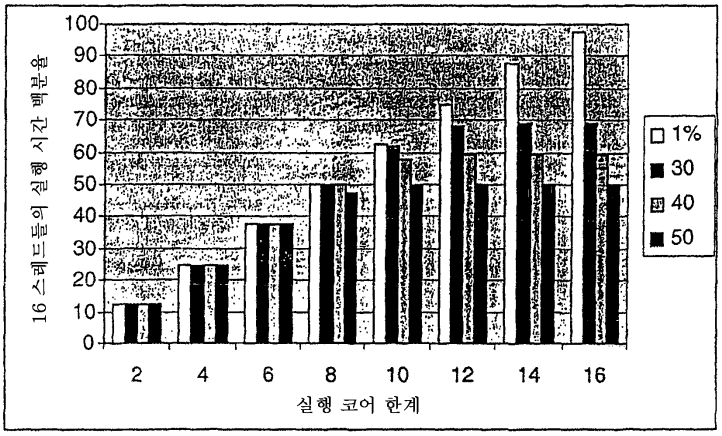
스레드가 완료되어 코어 비할당 큐로 다시 전환될 때까지의 나중의 상태들



도 1. 각 코어에 대한 CRSO 상태도

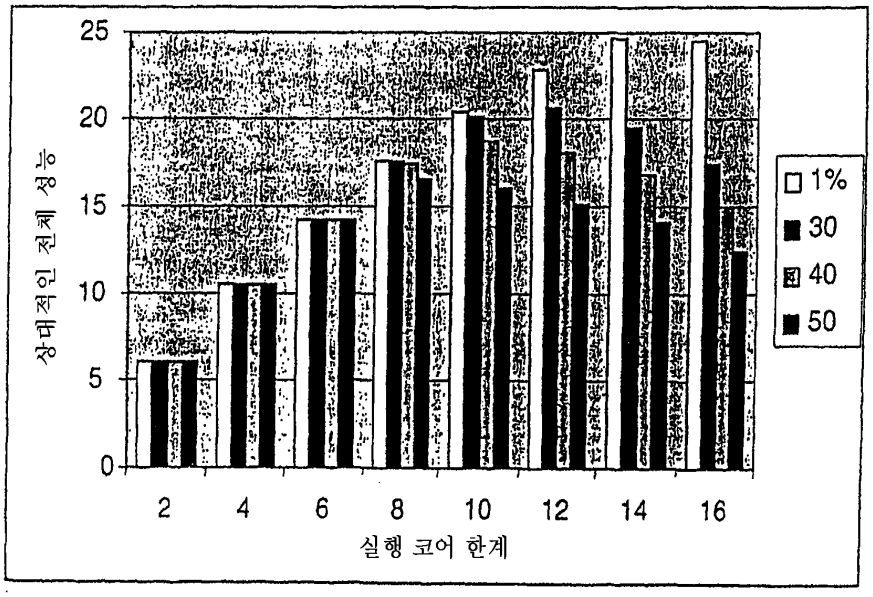
도면2

주파수. 전술된 바와 같이, 실행 코어들의 개수가 감소함에 따라, 동적 전력 소모가 감소하기 때문에, f는 ECL에 반비례하고, 따라서 정상 상태 열 한계로 남아있도록 f가 증가될 수 있다. 그러므로 전체 성능은 ECL이 감소됨에 따라 증가된 f의 효과들을 포함한다.



도 2. 1%, 30%, 40%, 및 50%의 메모리 참조 듀티 사이클로 16 스레드를 몬테카를로 시뮬레이션한 실행 시간 백분율

도면3



도면4

