(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2018년03월08일
(11) 등록번호 10-1835056
(24) 등록일자 2018년02월27일

(51) 국제특허분류(Int. Cl.)
G06F 9/50 (2018.01)
(21) 출원번호 10-2013-7030265
(22) 출원일자(국제) 2012년04월13일
심사청구일자 2017년04월13일
(85) 번역문제출일자 2013년11월14일
(65) 공개번호 10-2014-0027270
(43) 공개일자 2014년03월06일
(86) 국제출원번호 PCT/US2012/033660
(87) 국제공개번호 WO 2012/142512
국제공개일자 2012년10월18일
(30) 우선권주장
13/087,206 2011년04월14일 미국(US)
(56) 선행기술조사문헌
KR1020070049226 A*
(뒷면에 계속)
전체 청구항 수 : 총 19 항

(73) 특허권자
어드밴스드 마이크로 디바이시즈, 인코포레이티드
미국 캘리포니아 95054 산타 클라라 어거스틴 드라이브 2485
(72) 발명자
오스본 마이클 제이.
미국 뉴햄프셔 03049 홀리스 블랙 오크 드라이브 50
너쓰바움 세바스티엔 제이.
미국 매사추세츠 02420 레싱톤 파이퍼 레인 99
(74) 대리인
박장원

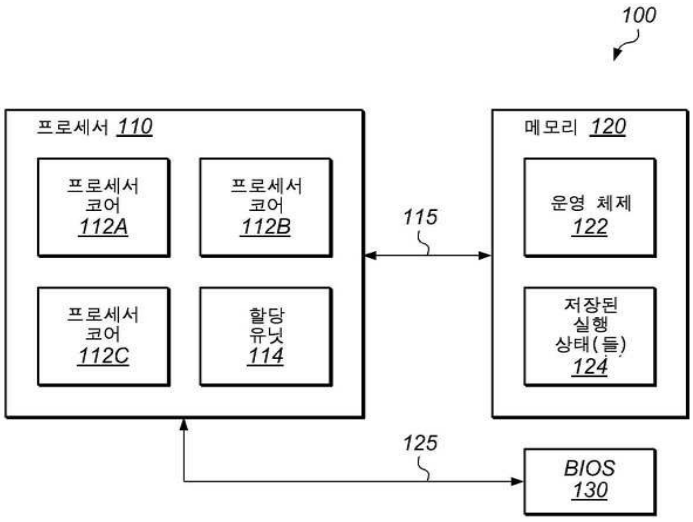
심사관 : 유진태

(54) 발명의 명칭 **논리적 코어들의 동적 맵핑**

(57) 요 약

논리적 코어들을 물리적 코어들로 동적으로 재맵핑하는 프로세서가 개시된다. 일 실시예에서, 상기 프로세서는 복수의 물리적 코어들을 포함하며, 논리적 코어들의 상기 복수의 물리적 코어들로의 맵핑을 저장하도록 구성된다. 상기 프로세서는 상기 프로세서의 부트 프로세스에 이어 상기 논리적 코어들을 상기 복수의 물리적 코어들로 재맵핑하도록 구성된 할당 유닛을 더 포함한다. 몇몇 실시예들에서, 상기 할당 유닛은 상기 복수의 물리적 코어들 중 하나 이상이 유휴 상태에 들어간다는 표시를 수신하는 것에 응답하여 상기 논리적 코어들을 재맵핑하도록 구성된다. 상기 프로세서는 상기 제 1 물리적 코어가 유휴 상태를 빠져나올 때 상기 복수의 물리적 코어들 중 제 2의 실행 상태를 가진 상기 복수의 물리적 코어들의 제 1을 로딩하도록 구성될 수 있다.

대 표 도 - 도1



(56) 선행기술조사문헌
KR1020070054138 A*
JP2008234191 A
US20100146513 A1
WO2009027153 A1
KR1020120104380 A
*는 심사관에 의하여 인용된 문헌

명 세 서

청구범위

청구항 1

프로세서로서,

복수의 물리적 코어들, 상기 프로세서는 상기 복수의 물리적 코어들로의 논리적 코어들의 맵핑을 저장하도록 구성되고, 상기 맵핑은 상기 복수의 물리적 코어들에게 태스크들을 할당하도록 운영 체제에 의해서 이용가능하며; 그리고

할당 유닛을 포함하며, 상기 할당 유닛은,

상기 프로세서의 부트(boot) 프로세스에 후속하여 상기 논리적 코어들을 상기 복수의 물리적 코어들로 재맵핑하도록 구성되고;

상기 논리적 코어들의 이전(previous) 재맵핑을 수행한 이후로 경과한 시간의 양을 결정하도록 구성되며; 그리고

상기 시간의 양이 임계 값을 초과하는 것에 응답하여 상기 논리적 코어들을 재맵핑하도록 구성되는 것을 특징으로 하는 프로세서.

청구항 2

제1항에 있어서,

상기 할당 유닛은 상기 복수의 물리적 코어들 중 하나 이상이 유휴 상태(idle state)에 들어갔다는 표시를 수신하도록 구성되며, 상기 할당 유닛은 상기 표시를 수신하는 것에 응답하여 상기 논리적 코어들을 재맵핑하도록 구성되는 것을 특징으로 하는 프로세서.

청구항 3

제2항에 있어서,

상기 프로세서는 제 1 물리적 코어가 상기 유휴 상태에 들어가는 것에 응답하여 상기 복수의 물리적 코어들 중 상기 제 1 물리적 코어의 실행 상태를 저장하도록 구성되며, 상기 프로세서는 상기 제 1 물리적 코어가 상기 유휴 상태를 빠져나오는 것에 응답하여 상기 복수의 물리적 코어들 중 제 2 물리적 코어의 실행 상태를 상기 제 1 물리적 코어에 로딩하도록 구성되는 것을 특징으로 하는 프로세서.

청구항 4

삭제

청구항 5

제1항에 있어서,

상기 할당 유닛은 상기 복수의 물리적 코어들 중 하나 이상의 물리적 코어의 각각의 작업 부하(work load)를 결정하도록 구성되며, 상기 할당 유닛은 상기 결정된 하나 이상의 작업 부하들에 기초하여 상기 논리적 코어들 중 하나 이상을 상기 복수의 물리적 코어들 중 하나 이상의 물리적 코어에 재맵핑하도록 구성되는 것을 특징으로 하는 프로세서.

청구항 6

제1항에 있어서,

상기 할당 유닛은 상기 복수의 물리적 코어들 중 하나 이상의 물리적 코어에 대한 온도 정보를 수신하도록 구성되며, 상기 할당 유닛은 상기 수신된 온도 정보에 기초하여 상기 논리적 코어들 중 하나 이상의 논리적 코어를 상기 복수의 물리적 코어들 중 하나 이상의 물리적 코어에 재맵핑하도록 구성되는 것을 특징으로 하는

프로세서.

청구항 7

제1항에 있어서,

상기 프로세서는 상기 복수의 물리적 코어들 중 제 1 물리적 코어의 실행 상태 및 상기 복수의 물리적 코어들 중 제 2 물리적 코어의 실행 상태를 저장하도록 구성되며, 상기 프로세서는 상기 할당 유닛이 상기 제 1 및 제 2 물리적 코어들에 맵핑된 논리적 코어들을 재맵핑하는 것에 응답하여 상기 제 2 물리적 코어의 상기 저장된 실행 상태를 상기 제 1 물리적 코어에 로딩하고 그리고 상기 제 1 물리적 코어의 상기 저장된 실행 상태를 상기 제 2 물리적 코어에 로딩하도록 구성되는 것을 특징으로 하는 프로세서.

청구항 8

제1항에 있어서,

상기 프로세서는 상기 복수의 물리적 코어들 중 하나를 오버클록(overclock)하도록 구성되며, 상기 할당 유닛은 상기 물리적 코어가 오버클록된 시간의 길이에 기초하여 논리적 코어를 상기 오버클록된 물리적 코어로 재맵핑하도록 구성되는 것을 특징으로 하는 프로세서.

청구항 9

제1항에 있어서,

상기 할당 유닛은 상기 논리적 코어들의 모두를 재맵핑하지 않고 상기 복수의 물리적 코어들 중 물리적 코어들(ones)로의 상기 논리적 코어들의 서브세트의 재맵핑을 수행하도록 구성되는 것을 특징으로 하는 프로세서.

청구항 10

논리적 코어들의 동적 맵핑을 위한 방법으로서,

프로세서 상에서의 복수의 물리적 코어들이 한 세트의 태스크들을 수행하는 단계, 논리적 코어들의 상기 복수의 물리적 코어들로의 맵핑에 기초하여 상기 한 세트의 태스크들이 할당되며;

상기 프로세서 내의 할당 유닛이 상기 논리적 코어들의 매핑 이후로 경과한 시간의 양을 결정하는 단계; 및

상기 시간의 양이 임계 값을 초과하는 것에 응답하여 상기 할당 유닛이 운영 체제가 실행 중인 동안 상기 논리적 코어들을 상기 복수의 물리적 코어들로 재맵핑하는 단계를 포함하며,

추가적인 세트의 태스크들이 상기 재맵핑에 기초하여 할당되는 것을 특징으로 하는 논리적 코어들의 동적 맵핑을 위한 방법.

청구항 11

제10항에 있어서,

논리적 코어가 제 1 물리적 코어로부터 제 2 물리적 코어로 재맵핑되는 것에 응답하여 상기 프로세서가 상기 복수의 물리적 코어들 중 제 1 물리적 코어로부터 상기 복수의 물리적 코어들 중 제 2 물리적 코어로 현재 실행 상태를 이동하는(transfer) 단계를 더 포함하는 것을 특징으로 하는 논리적 코어들의 동적 맵핑을 위한 방법.

청구항 12

제11항에 있어서,

상기 이동하는 단계는 상기 프로세서에 결합된 메모리에 상기 실행 상태를 저장하는 단계 및 상기 제 2 물리적 코어 상에 상기 실행 상태를 재로딩하는 단계를 포함하는 것을 특징으로 하는 논리적 코어들의 동적 맵핑을 위한 방법.

청구항 13

제12항에 있어서,

상기 이동하는 단계는 상기 제 1 물리적 코어의 상기 실행 상태를 재로딩하기 위해 상기 제 2 물리적 코어가 유

휴 상태를 빠져나오는 단계를 포함하는 것을 특징으로 하는 논리적 코어들의 동적 맵핑을 위한 방법.

청구항 14

제10항에 있어서,

상기 프로세서는 상기 복수의 물리적 코어들 중 물리적 코어들(ones) 각각의 성능 상태에 기초하여 상기 재맵핑을 결정하는 것을 특징으로 하는 논리적 코어들의 동적 맵핑을 위한 방법.

청구항 15

제14항에 있어서,

상기 재맵핑하는 단계는 상위 성능 상태에서의 제 1 물리적 코어로부터 하위 성능 상태에서의 제 2 물리적 코어로 논리적 코어를 재맵핑하는 단계를 포함하는 것을 특징으로 하는 논리적 코어들의 동적 맵핑을 위한 방법.

청구항 16

제10항에 있어서,

상기 프로세서는 상기 복수의 물리적 코어들로부터 수집된 하나 이상의 온도들에 기초하여 상기 재맵핑을 결정하는 것을 특징으로 하는 논리적 코어들의 동적 맵핑을 위한 방법.

청구항 17

제10항에 있어서,

상기 프로세서는 규칙적인 간격들로 상기 논리적 코어들의 상기 복수의 물리적 코어들로의 재맵핑을 수행하는 것을 특징으로 하는 논리적 코어들의 동적 맵핑을 위한 방법.

청구항 18

컴퓨터 시스템 상에서 실행가능한 명령들이 저장되어 있는 비-일시적인 컴퓨터 판독가능한 저장 매체로서, 상기 실행가능한 명령들은 상기 컴퓨터 시스템에서 실행될 때, 회로부를 포함한 프로세서를 제작하기 위한 공정의 일부를 수행하며, 상기 회로부는,

복수의 물리적 코어들, 상기 프로세서는 상기 복수의 물리적 코어들로의 논리적 코어들의 맵핑을 저장하도록 구성되며; 그리고

할당 유닛을 포함하며, 상기 할당 유닛은,

상기 프로세서의 부트(boot) 프로세스에 후속하여 상기 논리적 코어들을 상기 복수의 물리적 코어들로 재맵핑하도록 구성되고;

상기 논리적 코어들의 이전(previous) 재맵핑을 수행한 이후로 경과한 시간의 양을 결정하도록 구성되며; 그리고

상기 시간의 양이 임계 값을 초과하는 것에 응답하여 상기 논리적 코어들을 재맵핑하도록 구성되는 것을 특징으로 하는 비-일시적인 컴퓨터 판독가능한 저장 매체.

청구항 19

제18항에 있어서,

상기 프로세서는 논리적 코어가 제 1 물리적 코어로부터 제 2 물리적 코어로 재맵핑되는 것에 응답하여, 상기 복수의 물리적 코어들 중 제 1 물리적 코어로부터의 실행 상태를 메모리에 저장하고 그리고 상기 복수의 물리적 코어들 중 제 2 물리적 코어 상에 상기 실행 상태를 로딩하도록 구성되는 것을 특징으로 하는 비-일시적인 컴퓨터 판독가능한 저장 매체.

청구항 20

제18항에 있어서,

상기 저장 매체는 하드웨어 기술 언어(HDL, hardware description language) 데이터, 베릴로그(Verilog) 데이터, 또는 그래픽 데이터베이스 시스템 Ⅱ(GDS Ⅱ, graphic database system Ⅱ) 데이터를 저장하는 것을 특징으로 하는 비-일시적인 컴퓨터 판독가능한 저장 매체.

발명의 설명

기 술 분 야

[0001] 본 개시는 일반적으로 프로세서들에 관한 것으로, 보다 구체적으로는 다수의 프로세서 코어들에 걸쳐 작업 부하들을 분산하는 것에 관한 것이다.

배 경 기 술

[0002] 집적 회로들의 수명을 연장하는 것은, 많은 인스턴스들에서 고객들이 광범위한 기간들 동안 및 상당한 스트레스 하에서 신뢰성 있게 동작할 수 있는 회로들을 원하기 때문에, 종종 중요한 설계 목표이다. 따라서 개발자들은 집적 회로들의 수명에 영향을 미치는 다양한 인자들을 고려하려고 시도한다.

[0003] 프로세서들의 기대 수명에 영향을 미치는 하나의 중요한 인자는 프로세서가 시간에 걸쳐 경험하는 작업 부하이다. 높은 작업 부하(서버 시스템들에서 사용된 프로세서들과 같은) 하에서의 광범위한 기간들 동안 구동되는 프로세서들은 전자이주, 절연 파괴, 또는 열 유도 마모와 관련 있는 보다 많은 양의 실리콘 스트레스를 경험할 수 있다. 이들 인자들은, 결국 프로세서의 수명을 상당히 단축시킬 수 있다.

발명의 내용

해결하려는 과제

[0004] 따라서, 프로세서의 작업 부하를 감소시키는 것은 몇몇 인스턴스들에서, 그것의 기대 수명을 향상시키기 위해 바람직할 수 있다.

과제의 해결 수단

[0005] 본 개시는 논리적 코어들을 물리적 코어들로 동적으로 재맵핑하기 위한 구조들 및 방법들의 다양한 실시예들을 설명한다.

[0006] 일 실시예에서, 프로세서가 개시된다. 상기 프로세서는 복수의 물리적 코어들을 포함하며, 논리적 코어들의 상기 복수의 물리적 코어들로의 맵핑을 저장하도록 구성된다. 상기 프로세서는 프로세서의 부트 프로세서에 이어 상기 논리적 코어들을 복수의 물리적 코어들로 재맵핑하도록 구성된 할당 유닛을 더 포함한다.

[0007] 일 실시예에서, 방법이 개시된다. 상기 방법은 프로세서상에서의 복수의 물리적 코어들이 한 세트의 태스크들을 수행하는 단계를 포함하며, 여기에서 상기 세트의 태스크들은 논리적 코어들의 복수의 물리적 코어들로의 맵핑에 기초하여 할당된다. 상기 방법은 상기 프로세서가 운영 체제인 실행 중인 동안 상기 논리적 코어들의 상기 복수의 물리적 코어들로의 재맵핑 단계를 더 포함한다. 부가적인 세트의 태스크들이 상기 재맵핑에 기초하여 할당된다.

[0008] 일 실시예에서, 컴퓨터 판독가능한 저장 매체가 개시된다. 상기 저장 매체는 컴퓨터 시스템상에서 실행가능한 프로그램에 의해 동작되는 데이터 구조를 포함한다. 상기 데이터 구조상에서 동작하는 프로그램은 데이터 구조에 의해 설명된 회로를 포함한 프로세서를 제작하기 위해 프로세스의 일부를 수행하도록 실행가능하다. 상기 데이터 구조에 의해 설명된 회로는 복수의 물리적 코어들을 포함한다. 상기 프로세서는 논리적 코어들의 상기 복수의 물리적 코어들로의 맵핑을 저장하도록 구성된다. 상기 회로는 또한 상기 프로세서의 부트 프로세스에 이어 상기 논리적 코어들을 상기 복수의 물리적 코어들로 재맵핑하도록 구성된 할당 유닛을 포함한다.

도면의 간단한 설명

[0009] 도 1은 논리적 코어들을 물리적 코어들로 동적으로 재맵핑하도록 구성된 다중-코어 프로세서를 포함하는 컴퓨터 시스템의 일 실시예를 예시한 블록도이다.

도 2는 다중-코어 프로세서에 포함될 수 있는 할당 유닛의 일 실시예를 예시한 블록도이다.

도 3은 할당 유닛에 포함될 수 있는 결정 유닛의 일 실시예를 예시한 블록도이다.

도 4는 논리적 코어들을 물리적 코어들로 동적으로 재맵핑하기 위한 방법의 일 실시예를 예시한 흐름도이다.

도 5는 대표적인 컴퓨터 시스템의 일 실시예를 예시한 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0010] 본 명세서는 "일 실시예" 또는 "실시예"에 대한 참조들을 포함한다. 상기 구절들("일 실시예에서" 또는 "실시예에서")의 외관은 반드시 동일한 실시예를 나타내는 것은 아니다. 특정한 특징들, 구조들, 또는 특성들은 본 개시와 일치하는 임의의 적절한 방식으로 조합될 수 있다.

[0011] 용어들. 다음의 단락들은 본 개시(첨부된 청구항들을 포함하여)에서 발견된 용어들에 대한 정의들 및/또는 문맥을 제공한다.

[0012] "포함하는". 이 용어는 제약을 두지 않는다. 첨부된 청구항들에 사용된 바와 같이, 이 용어는 부가적인 구조 또는 단계들을 배제하지 않는다. "하나 이상의 프로세서 유닛들을 포함한 장치 ...."를 나열하는 청구항을 고려하자. 이러한 청구항은 부가적인 구성요소들(예로서, 네트워크 인터페이스 유닛, 그래픽스 회로 등)을 포함하는 것으로부터 상기 청구항을 배제하지 않는다.

[0013] "하도록 구성된". 다양한 유닛들, 회로들, 또는 다른 구성요소들은 "하도록 구성된"이 태스크 또는 태스크들을 수행하는 것으로서 설명되거나 또는 주장될 수 있다. 이러한 문맥들에서, "하도록 구성된"은 상기 유닛들/회로들/구성요소들이 동작 동안 이들 태스크 또는 태스크들을 수행하는 구조(예로서, 회로)를 포함한다는 것을 표시함으로써 구조를 함축하기 위해 사용된다. 이와 같이, 상기 유닛/회로/구성요소는 상기 특정된 유닛/회로/구성요소가 현재 동작 중이 아닐(예로서, 온이 아닐) 때조차 상기 태스크를 수행하도록 구성된다고 말하여질 수 있다. 상기 "하도록 구성된" 언어와 함께 사용된 상기 유닛들/회로들/구성요소들은 하드웨어, 예를 들면, 회로들, 상기 동작을 구현하도록 실행가능한 프로그램 명령들을 저장한 메모리, 등을 포함한다. 유닛/회로/구성요소가 하나 이상의 태스크들을 수행"하도록 구성된다"고 나열하는 것은 상기 유닛/회로/구성요소에 대해 35 U.S.C. § 112, 6번째 단락을 호출하지 않도록 명확하게 의도된다. 부가적으로, "하도록 구성된"은 문제가 되고 있는 태스크(들)를 수행할 수 있는 방식으로 동작하기 위해 소프트웨어 및/또는 펌웨어(예로서, FPGA 또는 범용 프로세서 실행 소프트웨어)에 의해 조작되는 포괄적인 구조(예로서, 포괄적인 회로)를 포함할 수 있다. "하도록 구성한"은 또한 하나 이상의 태스크들을 구현하거나 또는 수행하도록 적응되는 디바이스들(예로서, 집적 회로들)을 제작하기 위해 제조 프로세스(예로서, 반도체 제작 설비)를 적응시키는 것을 포함할 수 있다.

[0014] "제 1", "제 2" 등. 여기에 사용된 바와 같이, 이들 용어들은 그것들이 선행하는 명사들에 대한 라벨들로서 사용되며, 임의의 유형의 순서(예로서, 공간적, 시간적, 논리적 등)를 암시하지 않는다. 예를 들면, 8개의 프로세싱 요소들 또는 코어들을 가진 프로세서에서, 상기 용어들, 즉 "제 1" 및 제 2" 프로세싱 요소들은 상기 8개의 프로세싱 요소들 중 임의의 두 개를 나타내기 위해 사용될 수 있다. 다시 말해서, 상기 "제 1" 및 "제 2" 프로세싱 요소들은 논리적 프로세싱 요소들(0, 1)에 제한되지 않는다.

[0015] "에 기초한". 여기에 사용된 바와 같이, 이 용어는 결정에 영향을 미치는 하나 이상의 인자들을 설명하기 위해 사용된다. 이 용어는 결정에 영향을 미칠 수 있는 부가적인 인자들을 배제하지 않는다. 즉, 결정은 단지 이들 인자들에 기쵸하거나 또는 이들 인자들에 적어도 부분적으로 기초할 수 있다. 구절 "B에 기초하여 A를 결정하다"를 고려하자. B는 A의 결정에 영향을 미치는 인자일 수 있지만, 이러한 구절은 A의 결정이 또한 C에 기초하는 것을 배제하지 않는다. 다른 인스턴스들에서, A는 단지 B에 기초하여 결정될 수 있다.

[0016] "프로세서". 이 용어는 이 기술분야에서 그것의 보통의 및 허용된 의미를 가지며, 명령들을 실행할 수 있는 디바이스를 포함한다. 프로세서는, 제한 없이, 중앙 프로세싱 유닛(CPU), 코-프로세서, 산술 프로세싱 유닛, 그래픽스 프로세싱 유닛, 디지털 신호 프로세서(DSP) 등을 나타낼 수 있다. 프로세서는 단일 또는 다수의 파이프라인들을 가진 슈퍼스칼라 프로세서일 수 있다. 프로세서는 각각이 명령들을 실행하도록 구성되는 단일의 또는 다수의 코어들을 포함할 수 있다.

[0017] "물리적 코어". 이 용어는 이 기술분야에서의 보통의 및 허용된 의미를 가지며, 프로세서를 가진 물리적(즉, 유형의) 회로를 포함하고, 여기에서 상기 회로는 명령들을 실행하도록 구성된다. 프로세서는 다수의 물리적 코어들을 포함할 수 있다.

[0018] "논리적 코어". 이 용어는 이 기술분야에서 보통의 및 허용된 의미를 가지며, 다중-코어 프로세서에서의 물리적

코어와 연관되며 상기 코어에 태스크들을 할당하기 위해 사용되는 식별자를 포함한다. 예를 들면, 일 실시예에서, 태스크는 태스크가 물리적 코어(P0) 상에서 수행된다는 것을 직접 특정하는 대신에, 그것이 P0에 맵핑되는 논리적 코어(L0)에 의해 수행된다는 것을 특정할 수 있다. 논리적 코어(L0)가 후속하여 또 다른 물리적 코어(P1)에 재맵핑된다면, 상기 태스크는 이제 코어(P1) 상에서 수행될 수 있다. 따라서, "논리적 코어"는 상이한 시간들에서 동일한 물리적 코어에 대응하거나 또는 대응할 수 없는 코어에 대한 식별자이다.

[0019] "부트 프로세스". 이 용어는 이 기술분야에서 보통의 및 허용된 의미를 가지며, 컴퓨터 시스템을 초기화하고 상기 컴퓨터 시스템에 대한 운영 체제를 로딩하는 프로세스를 포함한다. 부트 프로세스는 또한 I/O 디바이스들, 테스팅 메모리 등을 초기화하는 것을 포함할 수 있다.

[0020] 최신 운영 체제들은 각각의 코어의 각각의 작업 부하들에 기초하여 다중-코어 프로세서에서의 상이한 프로세서 코어들에 태스크들을 분산하기 위한 능력을 지원한다. 예를 들면, 프로세서가 두 개의 프로세서 코어들을 포함하며 다른 것은 그렇지 않지만 하나의 코어가 막대한 작업 부하 하에서 동작한다면, 운영 체제(또는 몇몇 다른 메커니즘)는 상기 프로세서 코어들 사이에서의 태스크들을 재분배할 것이다. 코어들 사이에서의 태스크들을 재분배하는 것은 태스크들의 성능뿐만 아니라, 혹사당하는 코어가 초기에 고장날 기회들을 감소시킴으로써 프로세서의 기대 수명 또한 향상시킬 수 있다.

[0021] 운영 시스템은 특정 태스크들이 오직 특정 코어들 상에서 수행되도록 허용할 수 있기 때문에, 작업 부하들의 균일한 분배를 획득하는 것은 어려울 수 있다. 예를 들면, 특정 운영 체제들 하에서 실시간 프로그램 케이던스(cadence)를 설정하도록 작용하는 타이머 틱 인터럽트들은 논리적 부트스트랩 프로세서 코어로 안내되며, 다른 코어들로 분배될 수 없다. 다른 예들은 주어진 애플리케이션 프로세스(예로서, 친화성을 향상시키기 위한 의도를 갖고, 즉 프로세스가 그것의 캐시에 이미 존재하는 동일한 캐시 메모리 액세스 데이터세트를 발견할 수 있는 가능도)를 지속적으로 처리하도록 특정 코어에 요청하는 프로세서-간 인터럽트들 및 디바이스 프로그래밍을 통해 특정 코어들로 안내될 수 있는 IO 디바이스 인터럽트들을 포함한다. 특정한 코어가 이러한 방식으로 여러 개의 태스크들을 수행하도록 특정될 때, 상기 코어는 다른 코어들보다 더 높은 작업 부하를 가질 수 있으며, 따라서 보다 짧은 기대 수명을 가질 수 있다.

[0022] 부가적으로, 전력 관리에서의 최근의 진보들은 단일 CPU 코어가 활성인 시스템들의 전압 및 주파수를 열 한계들 내에서 달성 가능한 최고 전압 및 주파수로 올리고자 한다. 이것이 증가된 성능을 달성하는 동안, 그것은 또한 실리콘 열화율을 증가시킨다.

[0023] 시간에 걸쳐 실리콘 열화의 영향을 완화시키기 위한 여러 개의 방식들이 있다. 통상적으로, 실리콘 열화는 칩 규격의 일부인 한 세트의 작업 부하 및 환경적 동작 조건들을 사용하여 모델링된다. 이들 가정들은 시간에 걸친 실리콘 열화의 영향, 특히 트랜지스터 임계 전압, 포화 전류들, 및 이행 속도에 대한 효과를 추정하기 위해 사용된다. 이것은 장기 손상에 대하여 동작 주파수를 충분히 가드-밴딩함으로써 상기 부분에 할당된 성능을 제한하기 위해 사용될 수 있다. 이것은 실리콘 열화의 몇몇 양상들을 커버할 수 있지만, 그것은 열 유도된 기계적 마모를 커버하지 않으며, 보장된 수명에 관하여, 출시할 때 시스템의 최대 성능을 제한하는 것을 요구한다.

[0024] 몇몇 시스템들은 실리콘 스트레스를 감소시키기 위해 유휴 기간들 동안 동작의 전압 및 주파수를 적극적으로 아래로 관리하도록 시도한다. 이것은 통상적인 시스템들이 높은 활동 및 낮은 활동의 기간들을 검토하기 위해 예측될 수 있는 바와 같은 문제를 완화시키도록 돕지만, 그것은 단지 부분적인 해결책이며, 이전 단락에서 참조된 동작 주파수 가드-밴드를 감소시키기 위해 사용될 수 있다.

[0025] 몇몇 시스템들은 논리적 코어들을 물리적 코어들로 맵핑시키고 BIOS 파워-온 자기-테스트 단계 동안 상기 맵핑을 변경함으로써 이 문제를 완화시키려고 시도한다. 예를 들면, 두 개의 물리적 코어들(P0, P1)을 가진 프로세서는 제 1 부트 동안 각각 이들 코어들을 각각 논리적 코어들(L0, L1)에 맵핑시킨다. 논리적 코어(L0)를 특정하는 태스크들은 그 후 물리적 코어(P0) 상에서 수행될 수 있으며, 논리적 코어(L1)를 특정하는 태스크들은 그 후 물리적 코어(P1) 상에서 수행될 수 있다. 리셋시, 프로세서는 논리적 코어들(L0, L1)을 각각 물리적 코어들(P1, P0)에 재맵핑할 수 있다. 논리적 코어(L0)가 일반적으로 L1보다 더 막대한 작업 부하를 가진다면, 각각의 리셋 동안 상기 논리적 코어들을 재맵핑하는 것은 상기 물리적 코어들(P0, P1)이 상기 코어들의 수명에 걸쳐 보다 양호한 작업 부하 분산을 경험하게 할 수 있다. 리셋 동안 이러한 재맵핑을 수행하는 것은 운영 체제가 아직 로딩되지 않았기 때문에 비교적 단순하며, 소프트웨어는 특정 코어가 상기 인터럽트를 처리하도록 기대할 수 없다. 본 개시는 이러한 방식으로 논리적 코어들을 재맵핑하는 것은 리셋 이벤트들이 통상적으로 서버들 및 어느 정도까지, 데스크탑들에 대해 드물다는 것을 인지한다.

[0026] 본 개시는 논리적 코어들을 물리적 코어들에 동적으로 재맵핑하기 위한 다양한 기술들을 설명한다. 이하에 설명될 바와 같이, 다중-코어 프로세서는 논리적 코어들의 복수의 물리적 코어들로의 맵핑을 저장할 수 있다. 이러한 맵핑은 운영 체제, I/O 디바이스들 등에 의해 상기 물리적 코어들의 다양한 것들에 태스크들을 할당하기 위해 사용될 수 있다. 일 실시예에서, 상기 다중-코어 프로세서는 라이브 시스템(즉, 상기 프로세서의 부트 프로세스에 이어) 상에서 논리적 코어들을 물리적 코어들로 재맵핑하도록 구성된 할당 유닛을 포함한다. 예를 들면, 제 1 논리적 코어는 초기에 제 1 물리적 코어에 맵핑될 수 있다. 상기 제 1 논리적 코어로 어드레싱된 태스크들은 그 후 성능을 위해 제 1 물리적 코어에 할당될 수 있다. 상기 할당 유닛은 그 후 상기 물리적 코어들 중 제 2의 것으로 상기 제 1 논리적 코어의 재맵핑을 수행할 수 있다. 그 결과, 제 1 논리적 코어로 어드레싱된 태스크들은 그 후 제 1 물리적 코어 대신에 성능을 위해 제 2 물리적 코어에 할당될 수 있다. 제 1 논리적 코어가 통상적으로 다른 논리적 코어들보다 높은 작업 부하를 가진다면, 많은 인스턴스들에서 그것의 더 높은 작업 부하는 상기 프로세서를 재부팅할 필요 없이 물리적 코어들에 걸쳐 보다 균일하게 분산될 수 있다.

[0027] 따라서, 여기에 설명된 프로세서는 많은 인스턴스들에서, 동적 재맵핑을 수행하도록 구성되지 않은 다른 프로세서들보다 더 빈번하게 물리적 코어 할당들에 국한되어 스위칭할 수 있다. 다양한 실시예들에서, 이러한 보다 빈번한 재맵핑은 기능적 아키텍처 규격을 여전히 만족하면서 OS, I/O, 및 시스템-디바이스 프로그래밍에 관계없이 코어들에 걸쳐 작업 부하들의 보다 균일한 분산을 야기할 수 있다. 그 결과, 상기 프로세서는 단일 컴퓨터 엔티티 상에서 감소된 실리콘 스트레스를 경험할 수 있으며 전체 수명을 연장한다. 실리콘 수명을 증가시키는 것은 또한 실리콘 열화에 대한 동작 주파수 가드-밴드가 감소될 수 있는 바와 같이 부가적인 신뢰성 및 성능 업사이드로 나타날 수 있다. 부가적으로 제조시 부품당 실리콘 스트레스 테스팅은(보통 "번-인(burn-in)" 테스트로서 불리우는) 위험이 감소되기 때문에 보다 제한될 수 있다 - 이것은 제조 비용들을 감소시킨다.

[0028] 이제 도 1로 돌아가면, 컴퓨터 시스템(100)의 일 실시예의 블록도가 묘사된다. 예시된 실시예에서, 컴퓨터 시스템(100)은 프로세서(110), 메모리(120), 및 기본 입력/출력 시스템(BIOS)(130)을 포함한다. 프로세서(110)는 프로세서 코어들(112A 내지 112C)(몇몇 실시예들에서, 프로세서(110)는 도시된 것보다 더 많거나 또는 더 적은 코어들(112)을 포함할 수 있다) 및 할당 유닛(114)을 포함한다. 프로세서(110)는 상호연결(115)을 통해 메모리(120) 및 상호연결(125)을 통해 BIOS(130)에 결합된다. 예시된 실시예에서, 메모리(120)는 운영 체제(122) 및 하나 이상의 저장된 실행 상태들(124)을 저장하도록 구성된다.

[0029] 프로세서(110)는 임의의 적절한 유형의 다중-코어 프로세서일 수 있다. 프로세서(110)는 중앙 프로세싱 유닛(CPU)과 같은 범용 프로세서일 수 있다. 프로세서(110)는 가속 프로세싱 유닛(accelerated processing unit; APU), 디지털 신호 프로세서(DSP), 그래픽스 프로세싱 유닛(GPU) 등과 같은 특수-목적 프로세서일 수 있다. 프로세서(110)는 애플리케이션-특정 집적 회로(ASIC), 필드-프로그램가능한 게이트 어레이(FPGA) 등과 같은 가속도 로직일 수 있다. 프로세서(110)는 다중-스레딩 슈퍼스칼라 프로세서일 수 있다.

[0030] 메모리(120)는 임의의 적절한 유형의 메모리일 수 있다. 메모리(120)는 하드 디스크 저장 장치, 플로피 디스크 저장 장치, 착탈 가능한 디스크 저장 장치, 플래시 메모리, 랜덤 액세스 메모리(RAM - 정적 RAM(SRAM), EDO(extended data out) RAM, 동기식 동적 RAM(SDRAM), 이중 데이터 레이트(DDR) SRAM, RAMBUS RAM 등), 판독 전용 메모리(ROM - 프로그램가능한 ROM(PROM), 전기적으로 삭제가능한 프로그램가능한 ROM(EEPROM) 등) 등과 같이, 상이한 물리적 메모리 미디어를 사용하여 구현될 수 있다.

[0031] BIOS(130)는 프로세서(110)의 부트 프로세스를 수행하도록 구성되는 기본 입력/출력 시스템의 일 실시예이다. 다양한 실시예들에서, 이러한 부트 프로세스는 비디오 카드, 마우스, 키보드 등과 같은 컴퓨터 시스템(100)에서의 I/O 디바이스들을 식별 및 초기화하는 것을 포함할 수 있다. 이러한 부트 프로세스는 메모리(120)에서 디바이스들의 무결성을 테스트하는 것을 포함할 수 있다. 일 실시예에서, BIOS(130)는 영구 저장 장치로부터의 운영 체제(122)를 메모리(120)에서의 RAM에 로딩하고 프로세서(110)가 운영 체제(122)의 실행을 시작하게 함으로써 상기 부트 프로세스를 마칠 수 있다. 이하에 설명될 바와 같이, 다양한 실시예들에서, 할당 유닛(114)은 BIOS(130)가 프로세서(110)의 부트 프로세스를 수행하는 것을 완료한 후 논리적 코어들을 코어들(112)로 재맵핑하도록 구성된다.

[0032] 프로세서 코어들(112)은 물리적 코어들의 일 실시예이며, 이것은 하나 이상의 태스크들을 수행하기 위해 명령들을 실행하도록 구성된다. 다양한 실시예들에서, 이들 태스크들은 운영 체제(122)에 의해 할당될 수 있다(예로서, 타임 티커 인터럽트를 프로세싱하기 위해). 이들 태스크들은 다른 코어들(112)에 의해 할당될 수 있다(예로서, 프로세서-간 인터럽트(IPI)를 프로세싱하기 위해). 이들 태스크들은 I/O 기기들에 의해 할당될 수 있다(예로서, 수신된 입력을 서비스하기 위해). 상기 주지된 바와 같이, 다양한 실시예들에서, 몇몇 태스크들은

물리적 코어들(112)로의 논리적 코어들의 맵핑에 기초하여 할당될 수 있다. 예를 들면, 운영 체제(122)는 일 실시예에서 부트스트랩 프로세서에 대응할 수 있는 제 1 논리적 코어(L0)에 특정 태스크를 할당할 수 있다. 논리적 코어(L0)가 현재 프로세서(112A)에 맵핑된다면, 프로세서(112A)는 그 후 상기 태스크를 수행할 수 있다. 그러나, 논리적 코어(L0)가 프로세서(112B)에 맵핑된다면, 프로세서(112B)는 대신에 상기 태스크를 수행할 수 있다.

[0033] 몇몇 실시예들에서, 코어들(112)은 상이한 동작 상태들에서 동작하도록 구성될 수 있으며, 프로세서(110)에 의해 미리 정의되며 상이한 전원 상태들(power states) 및 상이한 성능 상태들(performance states)을 포함할 수 있다. 몇몇 실시예들에서, 이들 전원 상태들 및/또는 성능 상태들은 개선된 구성 및 전력 인터페이스(advanced configuration and power interface; ACPI) 표준에 의해 정의된 "C" 및 "P" 상태들에 각각 대응한다. 성능 상태는 코어(112)가 특정 전압/주파수에서 명령들을 실행하는 상태이다. 예를 들면, 일 실시예에서, 상당한 프로세싱 요구들이 존재한다면, 코어(112)는 그것의 최고 성능 상태에서 동작할 수 있으며, 이것은 성능 상태(P0)로서 불리울 수 있다. 이러한 실시예에서, P0는 코어(112)의 최대 동작 주파수 및 최고 전력 설정에 대응한다. 보다 적은 요구가 존재한다면, 코어(112)는 보다 낮은 성능 상태(예로서, 성능 상태(P1, P2 등))에서 동작할 수 있으며, 여기에서 코어(112)는 보다 낮은 동작 주파수들 및 보다 낮은 전력 설정들에서 동작한다. 전원 상태는 예를 들면, 코어(112)가 완전히 동작적인지 또는 전체적으로 또는 부분적으로 정지되는지를 표시할 수 있다. 예를 들면, 일 실시예에서, 코어(112)에 대한 몇몇 프로세싱 요구가 존재한다면, 상기 코어(112)는 명령들의 실행을 허용하는 전원 상태에서 동작할 수 있으며; 이러한 상태는 전원 상태(C0)로서 불리울 수 있다. 코어(112)에 대한 요구가 적거나 또는 없다면, 상기 코어(112)는, 일 실시예에서, 중단 상태 또는 클록-정지 상태와 같은 보다 낮은 전원 상태에서 동작할 수 있으며; 이러한 상태들은 각각 전원 상태들(C1 또는 C2)로서 불리울 수 있다. 따라서, 전원 상태는 코어(112)가 완전히 동작적이거나(즉, 명령들을 실행하는) 또는 코어(112)가 명령들을 실행하지 않는 복수의 상이한 유휴 상태들 중 하나인 상태를 나타낼 수 있다.

[0034] 일 실시예에서, 각각의 코어(112)는 유휴 상태에 들어갈 때 그것의 실행 상태(124)(상술된 코어의 동작 상태와 혼동되지 않도록)를 메모리(120)에 저장하도록 구성된다. 다양한 실시예들에서, 코어들(112)의 저장된 실행 상태들(124)은 레지스터들의 콘텐트들, 프로그램 카운터들, 스택 포인터들 등을 포함할 수 있다. 저장된 실행 상태들(124)은 또한 명령들 캐시들(I-캐시들)의 콘텐트들, 데이터 캐시들(D-캐시들)의 콘텐트들, 변환-색인-버퍼(translation-lookaside-buffer; TLB) 정보 등을 포함할 수 있다. 일 실시예에서, 코어(112)가 유휴 상태를 빠져나올 때, 상기 코어(112)는 그것의 실행 상태(124)를 재로딩하고 상기 재로딩된 상태(124)를 사용하여 실행 명령들을 재개하도록 구성된다. 이하에 설명될 바와 같이, 몇몇 실시예들에서, 코어(112)는 논리적 코어들의 물리적 코어들로의 재맵핑에 응답하여 또 다른 코어(112)의 저장된 상태(124)를 로딩하도록 구성될 수 있다.

[0035] 일 실시예에서, 할당 유닛(114)은 논리적 코어들의 물리적 코어들(112)로의 맵핑(즉, 논리적-코어 맵핑)을 생성하고 상기 논리적 코어들의 물리적 코어들(112)로의 동적 재맵핑을 수행하도록 구성된다. 여기에 사용된 바와 같이, 동적 재맵핑은 BIOS(130)가 프로세서(110)에 대한 부트 프로세스를 수행한 후 수행되는 재맵핑을 나타낸다. 또 다른 방식으로 말하면, 할당 유닛(114)은 일 실시예에서 운영 체제(122)가 로딩되고 실행한 후, 논리적 코어들을 물리적 코어들(112)로 재맵핑하도록 구성된다. 예시된 실시예에서, 할당 유닛(114)은 프로세서(110)에 위치되지만 특정 코어(112) 내에서는 위치되지 않는다. 몇몇 실시예들에서, 할당 유닛(114)은 코어(112) 내에 위치될 수 있거나 또는 다수의 코어들(112)(즉, 코어들(112)은 할당 유닛(114)의 동작들을 수행할 수 있다) 중에서 분배될 수 있다. 몇몇 실시예들에서, 할당 유닛(114)은 프로세서(110)의 노스 브리지 로직(north bridge logic) 내에 위치될 수 있다. 몇몇 실시예들에서, 할당 유닛(114)은 BIOS(130) 내에서와 같이 프로세서(110)의 외부에 위치될 수 있다. 몇몇 실시예들에서, 할당 유닛(114)에 의해 수행되는 것처럼 설명된 동작들은 컴퓨터 시스템(100) 내에서의 다수의 블록들 중에서 분할될 수 있다.

[0036] 일 실시예에서, 프로세서(110)는 할당 유닛(114)이 상이한 논리적 코어를 상기 코어(112)에 재맵핑할 때 코어(112)의 실행 상태를 또 다른 코어(112)로 전달하도록 구성된다. 예를 들면, 코어(112A)가 논리적 코어(L0)에 할당된다면, 프로세서(110)는 L0이 코어(112B)에 재맵핑될 때 코어(112A)의 상태를 코어(112B)에 전달할 수 있다. 일 실시예에서, 프로세서(110)는 코어들(112) 사이에서의 상호 연결을 통해 코어들(112) 사이에서 실행 상태들을 전달하도록 구성된다. 일 실시예에서, 프로세서(110)는 상기 상태를 메모리(120)(상태(124)로서)에 저장하고 그 후 상기 상태를 또 다른 코어(112)로 재로딩함으로써 실행 상태를 전달하도록 구성될 수 있다. 몇몇 실시예들에서, 프로세서(110)는 코어들(112)이 유휴 전원 상태(상술된 바와 같은)에 들어가고 빠져나올 때 실행 상태들을 저장 및 재로딩하도록 구서된다. 이하에 설명될 바와 같이, 프로세서(110)는 일 실시예에서, 하나 이상의 코어들(112)로 하여금 할당 유닛(114)이 재맵핑을 수행하는 것에 응답하여 유휴 상태에 들어가고 나오게

하도록 구성될 수 있다. 대안적으로, 할당 유닛(114)은 하나 이상의 코어들(112)이 유휴 상태로 또는 그로부터 이행할 때 재맵핑을 수행하도록 구성될 수 있다. 다양한 실시예들에서, 프로세서(110)는 운영 시스템 또는 소프트웨어 지식 없이 실행 상태들을 전달하도록 구성될 수 있다.

[0037] 할당 유닛(114)은 도 2와 함께 다음에 설명된다.

[0038] 이제 도 2로 돌아가면, 할당 유닛(114)의 일 실시예가 묘사된다. 예시된 실시예에서, 할당 유닛(114)은 맵핑 레지스터들(210), 결정 유닛(220), 및 카운터 레지스터들(230)을 포함한다. 유닛들(210 내지 230)이 단일 유닛(114) 내에 도시되지만, 몇몇 실시예들에서, 유닛들(210 내지 230)은 컴퓨터 시스템(100)에서의 유닛들 가운데 분산될 수 있다는 것을 주의하자.

[0039] 일 실시예에서, 맵핑 레지스터들(210)은 논리적 코어들의 물리적 코어들로의 맵핑(논리적-코어 맵핑(212)으로서 도시된)을 대표하는 식별자들을 저장하도록 구성된다. 몇몇 실시예들에서, 레지스터들(210)은 각각의 코어(112)에 대한 각각의 레지스터를 포함할 수 있으며 상기 각각의 레지스터(210)는 상기 코어(112)에 맵핑된 논리적 코어를 저장하도록 구성될 수 있다. 따라서, 레지스터들(210)은 코어(112A)에 할당된 논리적 코어를 저장하는 제 1 레지스터, 코어(112B)에 할당된 논리적 코어를 저장하는 제 2 레지스터 등을 포함할 수 있다. 다양한 실시예들에서, 운영 체제(122), 코어들(112), I/O 디바이스들 등은 태스크들을 할당할 곳(즉, 코어들(112) 중 어떤 것들이 태스크들을 수행할지)을 결정하기 위해 레지스터들(210)로부터 논리적 맵핑(212)을 판독하도록 구성될 수 있다.

[0040] 일 실시예에서, 결정 유닛(220)은 논리적 코어들의 물리적 코어들로의 재맵핑이 수행될 때를 식별하고, 상기 논리적-코어 재맵핑을 결정하도록 구성된다. 결정 유닛(220)은 다양한 적절한 조건들 중 임의의 것 상에서 재맵핑을 수행할 때에 대한 결정에 기초할 수 있다. 일 실시예에서, 결정 유닛(220)은 하나 이상의 코어들(112)이 유휴 상태에 들어간다는 표시를 수신하는 것에 응답하여 재맵핑을 수행하도록 구성된다. 상기 주지된 바와 같이, 프로세서(110)는 재맵핑 동안 하나의 코어(112)의 실행 상태(124)를 또 다른 코어(112)로 전달할 수 있다. 몇몇 실시예들에서, 프로세서(110)는 코어(112)가 유휴 상태를 빠져나올 때 또 다른 코어(112)의 실행 상태(124)를 로딩하게 함으로써, 이러한 전달을 수행하도록 구성된다. 코어(112)가 유휴 상태에 들어간 후 재맵핑을 수행함으로써, 일 실시예에서, 프로세서(110)는 그것이 이미 거기에 있기 때문에 상기 코어(112)를 유휴 상태에 들어가게 하는 것 없이 하나의 코어(112)의 상태를 또 다른 것에 전달할 수 있다. 일 실시예에서, 결정 유닛(220)은 규칙적인 간격들로 재맵핑을 수행하도록 구성된다(예로서, 이전 재맵핑 이래 특정한 양의 시간이 경과되었다고 결정한 후). 이러한 실시예에서, 할당 유닛(114)은 하나 이상의 코어(112)가 재맵핑 동안 실행 상태들(124)의 전달을 용이하게 하기 위해 유휴 상태에 들어가고 나오게 하도록 구성될 수 있다. 다른 실시예들에서, 할당 유닛(114)은 상기 설명된 것들과 같이 실행 상태의 전달을 용이하게 하기 위해 다른 기술들을 사용하도록 구성될 수 있다. 일 실시예에서, 결정 유닛(220)은 코어들(112)에 대한 결정된 작업 부하들에 기초하여 재맵핑을 수행하도록 구성된다. 예를 들면, 결정 유닛(220)은 특정 코어(112)의 작업 부하가 임계 값(예로서, 코어(112A)가 특정 양의 시간 동안 특정 성능 상태에서 동작하였다)을 초과한다고 결정한 후 재맵핑을 수행하도록 구성될 수 있다. 상기 예시된 실시예에서, 결정 유닛(220)은 운영 체제(122)로부터 수신된 코어 정보(222) 및/또는 코어들(112)로부터 수신된 코어 정보(224)에 기초하여 재맵핑을 수행할지 여부를 결정하도록 구성된다. 도 3과 함께 이하에 설명될 바와 같이, 이러한 정보는 코어들(112)에 대한 전원 및/또는 성능 상태들을 표시하는 동작 상태 정보, 코어들(112)에 대한 온도들을 특정하는 온도 정보 등을 포함할 수 있다.

[0041] 결정 유닛(220)은 또한 다양한 적절한 기준들 중 임의의 것에 기초하여 논리적-코어 재맵핑을 결정할 수 있다. 일 실시예에서, 결정 유닛(220)은 물리적 코어들(112)의 미리 결정된 순서에 기초하여 논리적 코어들을 재맵핑하도록 구성된다. 예를 들면, 결정 유닛(220)은 초기에 논리적 코어를 초기 물리적 코어(112)(예로서, 코어(112A)로 맵핑하고 그 후 논리적 코어를 순서에서 다음의 물리적 코어(112)(예로서, 코어(112B)로 재맵핑할 수 있다. 또 다른 실시예에서, 결정 유닛(220)은 랜덤하게 생성된 시퀀스에 기초하여 논리적 코어들을 물리적 코어들(112)에 할당하도록 구성될 수 있다. 몇몇 실시예들에서, 결정 유닛(220)은 수신된 정보(222 및/또는 224)에 기초하여 재맵핑을 결정하도록 구성될 수 있다. 이하에 설명될 바와 같이, 일 실시예에서, 결정 유닛(220)은 하나 이상의 코어들(112)의 동작 상태들에 기초하여 재맵핑을 결정하도록 구성될 수 있다. 일 실시예에서, 결정 유닛(220)은 하나 이상의 코어들(112)로부터 측정된 온도들에 기초하여 재맵핑을 결정하도록 구성될 수 있다. 몇몇 실시예들에서, 결정 유닛(220)은 코어들(112)에 대한 동작-상태 정보 및 온도 정보의 합성물에 기초하여 로컬-코어 재맵핑을 결정하도록 구성될 수 있다. 다양한 실시예들에서, 결정 유닛(220)은 상기 논리적 코어들의 일부 또는 모두를 물리적 코어들로 재맵핑할지 여부를 결정하도록 구성될 수 있다. 예를 들면, 결정 유닛(220)은 초기에 단지 논리적 코어들의 서브세트만을 재맵핑하도록 결정하며(예로서, 코어들(112)에 대한 온도 정보에

기초하여) 그 후 코어들(112)의 모두를 재맵핑하도록 결정할 수 있다(예로서, 코어들(112)의 현재 작업 부하들에 기초하여).

[0042] 일 실시예에서, 카운터 레지스터들(230)은 논리적-코어 재맵핑을 수행할지 여부를 결정할 때 결정 유닛(220)에 의해 사용가능한 정보를 저장하도록 구성된다. 일 실시예에서, 카운터 레지스터들(230)은 마지막 맵핑 또는 재맵핑 이래 경과된 시간을 표시하는 값으로 구성된다. 일 실시예에서, 카운터 레지스터들(230)은 코어(112)가 특정 전원 및/또는 성능 상태에서 얼마나 오래 동작하는지를 표시하는 각각의 코어(112)에 대한 값을 저장하도록 구성된다. 예를 들면, 레지스터(230)는 코어(112A)가 마지막 N 초 동안 오버클록 성능 상태에서 동작한다는 것을 특정하는 값을 저장할 수 있다. 일 실시예에서, 카운터 레지스터들(230)은 코어(112)가 특정 임계 온도 이상의 온도에서 얼마나 오래 동작하는지를 표시하는 각각의 코어(112)에 대한 값을 저장하도록 구성된다.

[0043] 이제 도 3으로 가면, 결정 유닛(220)의 블록도가 묘사된다. 상기 주지된 바와 같이, 다양한 실시예들에서, 결정 유닛(220)은 언제 재맵핑을 수행할지를 식별하고, 상기 재맵핑을 결정하도록 구성된다. 예시된 실시예에서, 결정 유닛(220)은 동작-상태 유닛(310) 및 온도 유닛(320)을 포함한다.

[0044] 일 실시예에서, 동작-상태 유닛(310)은 결정 유닛(220)에 대한 전원 및 성능 정보(312)를 프로세싱하도록 구성된다. 다양한 실시예들에서, 정보(312)는 각각의 코어(112)에 대한 전원 상태들 및 성능 상태들의 표시들을 포함한다. 정보(312)는 또한 코어(112)가 하나의 상태에서 또 다른 상태로 이행할 때를 특정하는(예로서, 코어(112A)가 성능 상태(P1)에서 성능 상태(P0)로 이행한다는 것을 특정하는) 표시를 포함할 수 있다. 일 실시예에서, 동작-상태 유닛(310)은 운영 체제(122)로부터 정보(312)를 수신하도록 구성된다. 또 다른 실시예에서, 동작-상태 유닛(310)은 코어들(112)로부터 정보를 직접 수신하도록 구성될 수 있다. 몇몇 실시예들에서, 동작-상태 유닛(310)은 카운터 레지스터들(230)에서의 정보(312)와 연관된 정보(카운터 정보(332)로서 도시된)를 저장 및 검색하도록 구성된다.

[0045] 일 실시예에서, 동작-상태 유닛(310)은 새로운 논리적-코어 맵핑(332)을 생성할지 여부를 결정할 때 결정 유닛(220)을 돕기 위해 프로세싱된 정보(312)를 사용하도록 구성된다. 몇몇 실시예들에서, 동작-상태 유닛(310)은 하나 이상의 코어들(112)이 새로운 동작 상태들로 이행하는 것에 응답하여 논리적-코어 재맵핑을 야기하도록 구성된다. 예를 들면, 일 실시예에서, 동작-상태 유닛(310)은 적어도 하나의 코어(112)가 유휴 상태에 들어가는 것에 응답하여 재맵핑을 야기하도록 구성된다. 상기 주지된 바와 같이, 몇몇 실시예들에서, 코어(112)는 유휴 상태에 들어갈 때 그것의 실행 상태(124)를 저장하도록 구성될 수 있으며 재맵핑 후 상기 유휴 상태를 빠져나올 때 또 다른 코어(112)의 실행 상태(124)를 재로딩할 수 있다. 몇몇 실시예들에서, 동작-상태 유닛(310)은 코어(112)가 얼마나 오래 특정 성능 상태 또는 전원 상태에 있는지에 기초하여 재맵핑을 야기하도록 구성된다. 예를 들면, 동작-상태 유닛(310)은 특정 코어(112)가 특정 시간 이상 동안 고 성능 상태(예로서, P0)에 있다고 결정할 때 재맵핑을 야기하도록 구성될 수 있다. 유사하게, 동작-상태 유닛(310)은 특정 코어(112)가 특정 시간 이상 동안 오버클록 상태에서 동작하고 있다고 결정할 때 재맵핑을 야기하도록 구성될 수 있다. 몇몇 실시예들에서, 동작-상태 유닛(310)은 상기 코어의 동작 상태 및 상기 상태에서 소비된 시간에 기초하여 각각의 코어(112)에 대한 작업 부하 점수를 계산하도록 구성될 수 있다. 동작-상태 유닛(310)은 개개의 점수들이 임계 값을 초과한다면 또는 상기 점수들의 함계가 총 임계값을 초과한다면 그 후 재맵핑을 야기하도록 구성될 수 있다. 몇몇 실시예들에서, 이러한 점수들은 재맵핑이 수행되어야 할 때를 결정하기 위해 온도 유닛(320)(이하에 설명된)에 의해 생성된 점수들과 조합될 수 있다.

[0046] 일 실시예에서, 동작-상태 유닛(310)은 또한 상기 논리적-코어 맵핑(332)을 결정할 때 결정 유닛(220)을 돕기 위해 프로세싱된 정보(312)를 사용하도록 구성된다. 일 실시예에서, 동작-상태 유닛(310)은 결정 유닛(220)으로 하여금 논리적 코어들을 유휴 상태에서 동작하는 물리적 코어들(112)로 재맵핑하게 하도록, 및 유휴 상태에서 동작하지 않는 물리적 코어들(112)에 대한 논리적 코어들을 재맵핑하지 않게 하도록 구성된다. 일 실시예에서, 동작-상태 유닛(310)은 결정 유닛(220)이 상위 동작 상태들에서의 코어들(112)로부터 하위 동작 상태들에서의 코어들(112)로 논리적-코어들을 재맵핑하게 하도록 구성된다. 일 실시예에서, 동작-상태 유닛(310)은 결정 유닛(220)으로 하여금 논리적 코어들을 보다 작은 기간들 동안 상기 상태들에 있는 코어들(112)로 재맵핑하기 전에 논리적 코어들을 보다 긴 기간들 동안 상위 동작 상태에 있는 코어들(112)로 재맵핑하게 하도록 구성된다.

[0047] 일 실시예에서, 온도 유닛(320)은 결정 유닛(220)에 대한 온도 정보(322)를 프로세싱하도록 구성된다. 다양한 실시예들에서, 정보(322)는 프로세서(110) 전체에 걸쳐 수집된 하나 이상의 온도들을 포함한다. 일 실시예에서, 온도 유닛(320)은 코어들(112) 상에 위치된 다수의 센서들로부터 정보(322)를 수신하도록 구성된다. 몇몇 실시예들에서, 온도 유닛(320)은 카운터 레지스터들(320)에서의 정보(322)와 연관된 정보(332)를 저장 및 검색하도

록 구성될 수 있다.

[0048] 일 실시예에서, 온도 유닛(320)은 새로운 논리적-코어 맵핑(332)을 생성할지 여부를 결정할 때 결정 유닛(220)을 돕기 위해 프로세싱된 정보(322)를 사용하도록 구성된다. 몇몇 실시예들에서, 온도 유닛(320)은 결정 유닛(220)으로 하여금 코어(112)가 최대 온도 임계값을 초과하였다고 결정하는 것에 응답하여 논리적-코어 재맵핑을 수행하게 하도록 구성될 수 있다. 일 실시예에서, 온도 유닛(320)은 코어(112)가 임계 온도를 초과하는지 여부, 및 그렇다면, 코어(112)가 상기 온도를 얼마나 오래 초과했는지를 결정하도록 구성될 수 있다. 상기 코어(112)가 특정된 기간 동안 상기 온도를 초과한 후, 온도 유닛(320)은 그 후 논리적-코어 재맵핑을 야기하도록 구성될 수 있다. 일 실시예에서, 온도 유닛(320)은 코어들(112)로부터 수집된 온도들 및 상기 온도들에서 소비된 시간에 기초하여 각각의 코어(112)에 대한 작업 부하 점수를 계산하도록 구성될 수 있다. 온도 유닛(320)은 상기 점수들이 임계값을 초과한다면 그 후 재맵핑을 야기하도록 구성될 수 있다. 몇몇 실시예들에서, 이러한 점수들은 재맵핑이 수행되어야 할 때를 결정하기 위해 동작-상태 유닛(310)에 의해 생성된 점수들과 조합될 수 있다.

[0049] 일 실시예에서, 온도 유닛(320)은 또한 논리적-코어 맵핑(332)을 결정할 때 결정 유닛(220)을 보조하기 위해 프로세싱된 정보(322)를 사용하도록 구성된다. 일 실시예에서, 온도 유닛(320)은 결정 유닛(220)으로 하여금 상위 온도들에서 동작하는 코어들(112)로부터 하위 온도들에서 동작하는 코어들(112)로 상기 논리적-코어들을 재맵핑하게 하도록 구성된다. 일 실시예에서, 온도 유닛(320)은 결정 유닛(220)으로 하여금 논리적 코어들을 보다 짧은 기간들 동안 유사한 온도들에서 동작하는 코어들(122)로 재맵핑하기 전에 논리적 코어들을 보다 긴 기간들 동안 보다 높은 온도들에서 동작하는 코어들(112)로 재맵핑하게 하도록 구성될 수 있다.

[0050] 이제 도 4로 돌아가면, 논리적 코어들을 동적으로 재맵핑하기 위한 방법(400)의 흐름도가 묘사된다. 방법(400)은 프로세서(110)와 같은, 다중-코어 프로세서에 의해 수행될 수 있는 방법의 일 실시예이다. 다양한 실시예들에서, 방법(400)은 후속 재맵핑들을 수행하기 위해 다수 회 수행될 수 있다. 몇몇 인스턴스들에서, 방법(400)을 수행하는 것은 프로세서 코어들 가운데 작업 부하들의 보다 양호한 분산을 생성할 수 있으며, 이것은 결국 프로세서의 수명을 향상시킬 수 있다.

[0051] 단계(410)에서, 프로세서(110)는 논리적 코어들의 물리적 코어들로의 맵핑을 저장한다. 상기 논의된 바와 같이, 다양한 실시예들에서, 이러한 맵핑은 태스크들을 성능을 위한 물리적 코어들(예로서, 코어들(112))에 할당하기 위해 운영 체제, 다른 코어들, I/O 디바이스들 등에 의해 사용될 수 있다. 일 실시예에서, 프로세서(110)는 할당 유닛(예로서, 할당 유닛(114))의 한 세트의 레지스터들(예로서, 레지스터들(230))에서의 맵핑을 저장한다. 다른 실시예들에서, 프로세서(110)는 메모리(예로서, 메모리(120)), BIOS(예로서, BIOS(130)) 등 내에서와 같이 어딘가에서의 맵핑을 저장한다. 몇몇 인스턴스들에서, 상기 저장된 맵핑은 예를 들면, 컴퓨터 시스템(예로서, 시스템(100))의 부트 프로세스 동안 생성된 초기 맵핑일 수 있다. 다른 인스턴스들에서, 상기 저장된 맵핑은 이전에 수행된 재맵핑 동안 생성된 맵핑일 수 있다.

[0052] 단계(420)에서, 프로세서(110)(예로서, 할당 유닛(114)을 사용하여)는 부트 프로세스에 이어 논리적 코어들을 물리적 코어들로 재맵핑한다. 이러한 재맵핑은 그 후 다양한 실시예들에서, 후속 태스크들을 성능을 위한 물리적 코어들에 할당하기 위해 사용될 수 있다. 상기 논의된 바와 같이, 프로세서(110)(예로서, 결정 유닛(220)을 사용하여)는 다양한 조건들 중 임의의 것에 기초하여 상기 재맵핑을 수행할 때를 식별할 수 있다. 일 실시예에서, 프로세서(110)는 규칙적인 간격들로 재맵핑을 수행할 수 있다. 몇몇 실시예들에서, 프로세서(110)는 물리적 코어들의 하나 이상의 동작 상태들에 기초하여 재맵핑을 수행하도록 결정할 수 있다. 예를 들면, 일 실시예에서, 프로세서(110)는 하나 이상의 코어들이 유휴 상태에 들어가는 것에 응답하여 재맵핑을 수행하도록 결정할 수 있다. 몇몇 실시예들에서, 프로세서(110)는 상기 코어들로부터 수집된 온도 정보에 기초하여 재맵핑을 수행하도록 결정할 수 있다.

[0053] 상기 논의된 바와 같이, 프로세서(110)는 다양한 적절한 기준들 중 임의의 것에 기초하여 상기 재맵핑을 결정할 수 있다. 일 실시예에서, 프로세서(110)는 상기 물리적 코어들의 미리 결정된 순서에 기초하여 논리적 코어들을 물리적 코어들에 재맵핑할 수 있다. 예를 들면, 논리적 코어(L0)가 물리적 코어(P0)에 할당되고 물리적 코어(P1)가 P0 후 순서에서 다음이면, 논리적 코어(L0)는 그 후 코어(P1)에 재맵핑될 수 있다. 몇몇 실시예들에서, 프로세서(110)는 상기 코어들의 동작 상태들에 기초하여 재맵핑을 결정할 수 있다. 예를 들면, 일 실시예에서, 프로세서(110)는 상위 동작 상태들에서의 물리적 코어들(예로서, 성능 상태(P0)에서의 물리적 코어)로부터 하위 동작 상태들에서의 물리적 코어들(예로서, 성능 상태(P1)에서의 물리적 코어)로 논리적 코어들을 재맵핑할 수 있다. 몇몇 실시예들에서, 프로세서(110)는 상기 물리적 코어들로부터 측정된 온도에 기초하여 상기 재맵핑을 결정할 수 있다. 예를 들면, 일 실시예에서, 프로세서(110)는 보다 높은 온도에서 동작하는 물리적 코어들로부

터 보다 낮은 온도들에서 동작하는 물리적 코어들로 논리적 코어들을 재맵핑할 수 있다.

[0054] 다양한 실시예들에서, 프로세서(110)는 상기 재맵핑에 응답하여 물리적 코어들 사이에서의 실행 상태들을 전달한다. 상기 논의된 바와 같이, 몇몇 실시예들에서, 프로세서(110)는 상기 실행 상태를 메모리에 저장하고 그 후 상기 제 2 코어 상에서 상기 실행 상태를 재로딩함으로써 제 1 코어로부터 제 2 코어로 실행 상태를 전달한다. 일 실시예에서, 프로세서(110)는 상기 제 1 코어로 하여금 상기 코어가 그것의 실행 상태를 메모리에 저장하는 유휴 상태에 들어가게 함으로써 상기 저장을 수행한다. 프로세서(110)는 상기 제 2 코어로 하여금 유휴 상태를 빠져나오고 상기 실행 상태를 재로딩하게 함으로써 상기 재로딩을 수행할 수 있다.

[0055] 대표적인 컴퓨터 시스템

[0056] 이제 도 5로 가면, 프로세서(110)를 포함할 수 있는 대표적인 컴퓨터 시스템(500)의 일 실시예가 묘사된다. 컴퓨터 시스템(500)은 상호연결(560)을 통해 시스템 메모리(520) 및 I/O 인터페이스(들)(540)에 결합되는 프로세서 서브시스템(580)을 포함한다. I/O 인터페이스(들)(540)는 하나 이상의 I/O 디바이스들(550)에 결합된다. 컴퓨터 시스템(500)은 이에 제한되지 않지만, 서버 시스템, 개인용 컴퓨터 시스템, 데스크탑 컴퓨터, 랩탑 또는 노트북 컴퓨터, 메인프레임 컴퓨터 시스템, 핸드헬드 컴퓨터, 워크스테이션, 네트워크 컴퓨터, 이동 전화기, 페이저, 또는 개인용 데이터 보조기(PDA)와 같은 소비자 디바이스를 포함한 다양한 유형들의 디바이스들 중 임의의 것일 수 있다. 컴퓨터 시스템(500)은 또한 저장 디바이스들, 스위치들, 모뎀들, 라우터들 등과 같은 임의의 유형의 네트워킹된 주변 디바이스일 수 있다. 단일 컴퓨터 시스템(500)이 편리함을 위해 도시되지만, 시스템(500)은 또한 함께 동작하는 둘 이상의 컴퓨터 시스템들로서 구현될 수 있다.

[0057] 프로세서 서브시스템(580)은 하나 이상의 프로세서들 또는 프로세싱 유닛들을 포함할 수 있다. 예를 들면, 프로세서 서브시스템(580)은 하나 이상의 리소스 제어 프로세싱 요소들(520)에 결합되는 하나 이상의 프로세싱 유닛들(그 각각은 다수의 프로세싱 요소들 또는 코어들을 가질 수 있다)을 포함할 수 있다. 컴퓨터 시스템(500)의 다양한 실시예들에서, 프로세서 서브시스템(580)의 다수의 인스턴스들은 상호연결(560)에 결합될 수 있다. 다양한 실시예들에서, 프로세서 서브시스템(580)(또는 580 내에서의 각각의 프로세서 유닛 또는 프로세싱 요소)은 캐시 또는 다른 형태의 탑재된 메모리를 포함할 수 있다. 일 실시예에서, 프로세서 서브시스템(580)은 상술된 프로세서(110)를 포함할 수 있다.

[0058] 시스템 메모리(520)는 프로세서 서브시스템(580)에 의해 사용가능하다. 시스템 메모리(520)는 하드 디스크 저장 장치, 플로피 디스크 저장 장치, 착탈 가능한 디스크 저장 장치, 플래시 메모리, 랜덤 액세스 메모리(RAM - 정적 RAM(SRAM), EDO(extended data out) RAM, 동기식 동적 RAM(SDRAM), 이중 데이터 레이트(DDR) SRAM, RAMBUS RAM 등), 판독 전용 메모리(ROM - 프로그램가능한 ROM(PROM), 전기적으로 삭제가능한 프로그램가능한 ROM(EEPROM) 등) 등과 같이, 상이한 물리적 메모리 미디어를 사용하여 구현될 수 있다. 컴퓨터 시스템(500)에서의 메모리는 메모리(520)와 같은 1차 저장 장치에 제한되지 않는다. 오히려, 컴퓨터 시스템(500)은 또한 프로세서 서브시스템(580)에서의 캐시 메모리 및 I/O 디바이스들(550) 상에서의 2차 저장 장치(예로서, 하드 드라이브, 저장 어레이 등)와 같은 다른 형태들의 저장 장치를 포함할 수 있다. 몇몇 실시예들에서, 이들 다른 형태들의 저장 장치는 또한 프로세서 서브시스템(580)에 의해 실행가능한 프로그램 명령들을 저장할 수 있다.

[0059] I/O 인터페이스들(540)은 다양한 실시예들에 따라, 다른 디바이스들에 결합하고 그와 통신하도록 구성된 다양한 유형들의 인터페이스들 중 임의의 것일 수 있다. 일 실시예에서, I/O 인터페이스(540)는 전면에서 하나 이상의 후면 버스들로의 브리지 칩(예로서, 사우스브리지(Southbridge))이다. I/O 인터페이스들(540)은 하나 이상의 대응하는 버스들 또는 다른 인터페이스들을 통해 하나 이상의 I/O 디바이스들(550)에 결합될 수 있다. I/O 디바이스들의 예들은 저장 디바이스들(하드 드라이브, 광 드라이브, 착탈 가능한 플래시 드라이브, 저장 어레이, SAN, 또는 그것들의 연관된 제어기), 네트워크 인터페이스 디바이스들(예로서, 로컬 또는 광역 네트워크로의) 또는 다른 디바이스들(예로서, 그래픽스, 사용자 인터페이스 디바이스들 등)을 포함한다. 일 실시예에서, 컴퓨터 시스템(500)은 네트워크 인터페이스 디바이스를 통해 네트워크에 결합된다.

[0060] 컴퓨터 시스템들(예로서, 컴퓨터 시스템(500))에 의해 실행되는 프로그램 명령들은 다양한 형태들의 컴퓨터 판독가능한 저장 미디어 상에 저장될 수 있다. 일반적으로 말하면, 컴퓨터 판독가능한 저장 매체는 상기 컴퓨터에 명령들 및/또는 데이터를 제공하기 위해 컴퓨터에 의해 판독가능한 임의의 비-일시적/유형의 저장 미디어를 포함할 수 있다. 예를 들면, 컴퓨터 판독가능한 저장 매체는 자기 또는 광 미디어, 예로서 디스크(고정되거나 또는 착탈 가능한), 테이프, CD-ROM, 또는 DVD-ROM, CD-R, CD-RW, DVD-R, DVD-RW, 또는 블루-레이와 같은 저장 미디어를 포함할 수 있다. 저장 미디어는 범용 시리얼 버스(USB) 인터페이스 등과 같은 주변 인터페이스를 통해 액세스 가능한 RAM(예로서, 동기식 동적 RAM(SDRAM), 이중 데이터 레이트(DDR, DDR2, DDR3 등) SDRAM, 저-전력

DDR(LPDDR2 등) SDRAM, 램버스 DRAM(RDRAM), 정적 RAM(SRAM) 등), ROM, 플래시 메모리, 비-휘발성 메모리(예로서, 플래시 메모리)와 같은 휘발성 또는 비-휘발성 메모리 미디어를 더 포함할 수 있다. 저장 미디어는 마이크로 전자기계 시스템들(MEMS), 뿐만 아니라 네트워크 및/또는 무선 링크와 같은 통신 매체를 통해 액세스 가능한 저장 미디어를 포함할 수 있다.

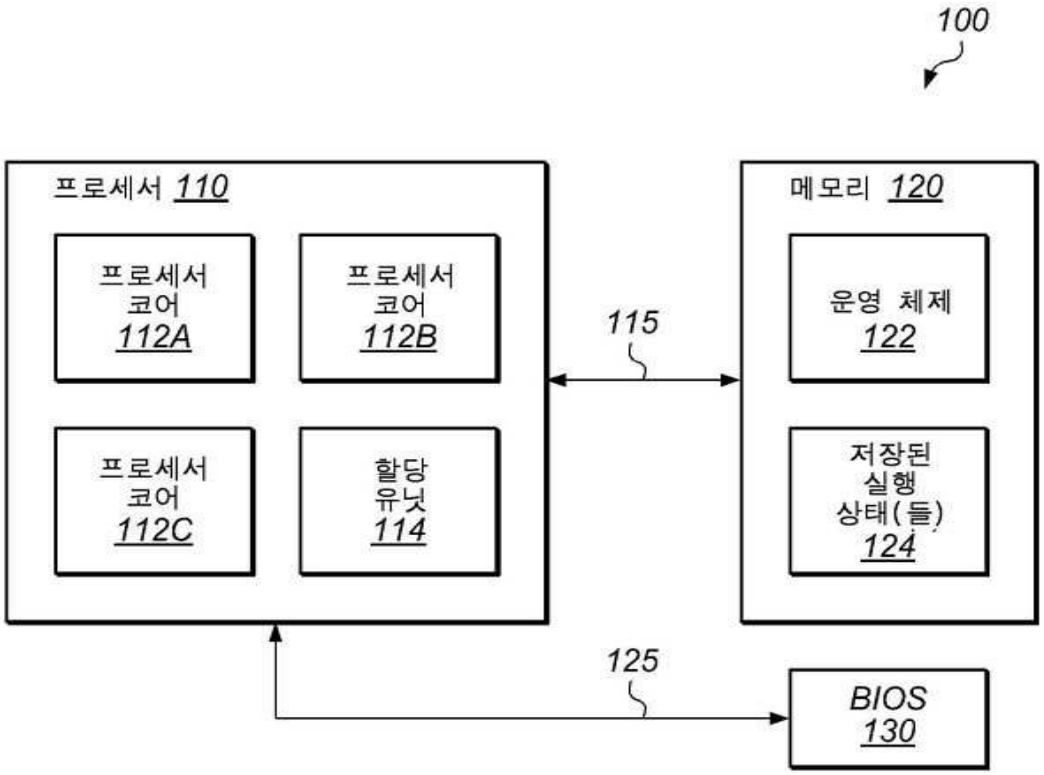
[0061] 몇몇 실시예들에서, 컴퓨터-판독가능한 저장 매체는 프로그램에 의해 판독된 명령들을 저장하기 위해 사용될 수 있으며 상술된 프로세서(110)를 위한 하드웨어를 제작하기 위해 직접 또는 간접적으로 사용될 수 있다. 예를 들면, 상기 명령들은 베릴로그(Verilog) 또는 VHDL과 같은 고 레벨 설계 언어(HDL)에서의 하드웨어 기능성의 행동-레벨 또는 레지스터-전달 레벨(RTL) 설명을 기술하는 하나 이상의 데이터 구조들을 개괄할 수 있다. 상기 설명은 합성 툴에 의해 판독될 수 있으며, 이것은 넷리스트(netlist)를 생성하기 위해 상기 설명을 합성할 수 있다. 상기 넷리스트는 한 세트의 게이트들(예로서, 합성 라이브러리에 정의된)을 포함할 수 있으며, 이것은 프로세서(110)의 기능을 나타낸다. 상기 넷리스트는 그 후 마스크들에 적용될 기하학적 형상들을 기술하는 데이터 세트를 생성하기 위해 위치되고 라우팅될 수 있다. 상기 마스크들은 그 후 프로세서(110)에 대응하는 반도체 회로 또는 회로들을 생성하기 위해 다양한 반도체 제작 단계들에서 사용될 수 있다.

[0062] 특정 실시예들이 상기 설명되었지만, 이들 실시예들은 본 개시의 범위를 제한하고자 의도되지 않으며, 심지어 여기에서 단일 실시예가 특정한 특징에 대하여 설명된다. 본 개시에 제공된 특징들의 예들은 달리 서술되지 않는다면 제한적이기보다는 예시적인 것으로 의도된다. 상기 설명은 본 개시의 이득을 가진 이 기술분야의 숙련자에게 명백할 바와 같이 이러한 대안들, 변경들, 및 등가물들을 커버하도록 의도된다.

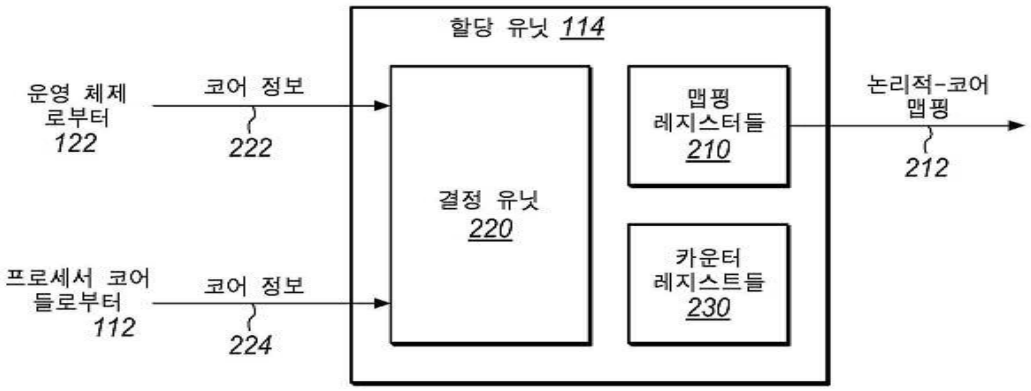
[0063] 본 개시의 범위는, 그것이 여기에서 처리된 문제들 중 임의의 것 또는 모두를 완화시키는지 여부에 상관없이, 임의의 특징 또는 여기에 개시된 특징들의 조합(명시적으로 또는 암시적으로), 또는 그것들의 임의의 일반화를 포함한다. 따라서, 새로운 청구항들은 특징들의 임의의 이러한 조합에 본 출원(또는 그것에 대한 우선권을 주장하는 출원)의 심사 동안 만들어질 수 있다. 특히, 첨부된 청구항들을 참조하여, 종속 청구항들로부터의 특징들은 독립 청구항들의 것들과 조합될 수 있으며 각각의 독립 청구항들로부터의 특징들은 임의의 적절한 방식으로 조합될 수 있으며 단지 첨부된 청구항들에 열거된 특정 조합들로 조합되는 것은 아니다.

도면

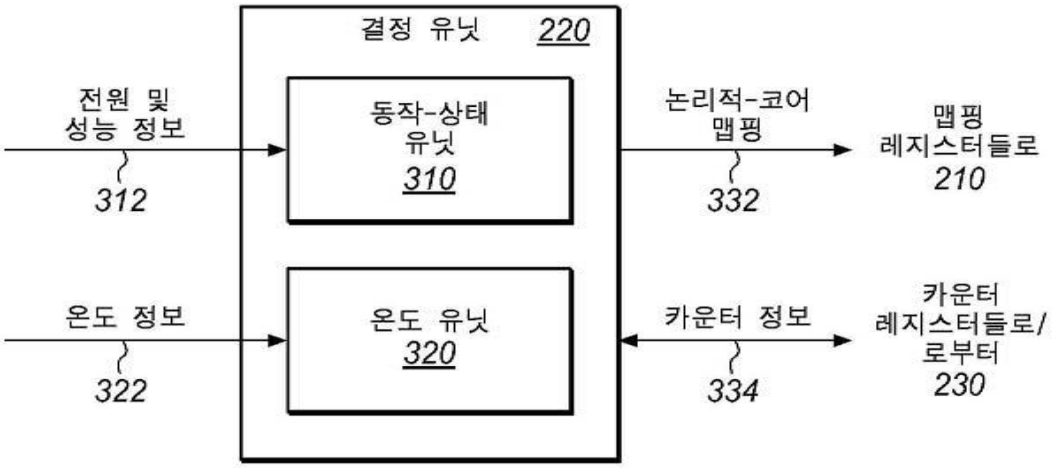
도면1



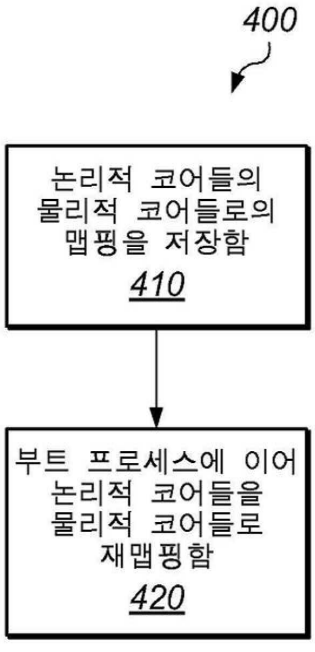
도면2


할당 유닛 114
운영 체제
로부터
122
코어 정보
222
결정 유닛
220
맵핑
레지스터들
210
논리적-코어
맵핑
212
프로세서 코어
들로부터
112
코어 정보
224
카운터
레지스트들
230

도면3


결정 유닛 220
전원 및
성능 정보
312
동작-상태
유닛
310
논리적-코어
맵핑
332
맵핑
레지스터들로
210
온도 정보
322
온도 유닛
320
카운터 정보
334
카운터
레지스터들로/
로부터
230

도면4


400
논리적 코어들의
물리적 코어들로의
맵핑을 저장함
410
부트 프로세스에 이어
논리적 코어들을
물리적 코어들로
재맵핑함
420

도면5