(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2015년06월16일
(11) 등록번호 10-1529018
(24) 등록일자 2015년06월09일

(51) 국제특허분류(Int. Cl.)
G06F 1/32 (2006.01)
(21) 출원번호 10-2014-7010486
(22) 출원일자(국제) 2012년08월31일
심사청구일자 2014년04월18일
(85) 번역문제출일자 2014년04월18일
(65) 공개번호 10-2014-0079420
(43) 공개일자 2014년06월26일
(86) 국제출원번호 PCT/US2012/053352
(87) 국제공개번호 WO 2013/043352
국제공개일자 2013년03월28일
(30) 우선권주장
13/312,678 2011년12월06일 미국(US)
61/536,207 2011년09월19일 미국(US)
(56) 선행기술조사문헌
KR1020120117015 A
US20110173474 A1
US20060026447 A1
US20060053326 A1

(73) 특허권자
퀄컴 인코포레이티드
미국 92121-1714 캘리포니아주 샌 디에고 모어하우스 드라이브 5775
(72) 발명자
울머 트레이시 에이
미국 92121 캘리포니아주 샌디에고 모어하우스 드라이브 5775
프란츠 앤드류 제이
미국 92121 캘리포니아주 샌디에고 모어하우스 드라이브 5775
(뒷면에 계속)
(74) 대리인
특허법인코리아나

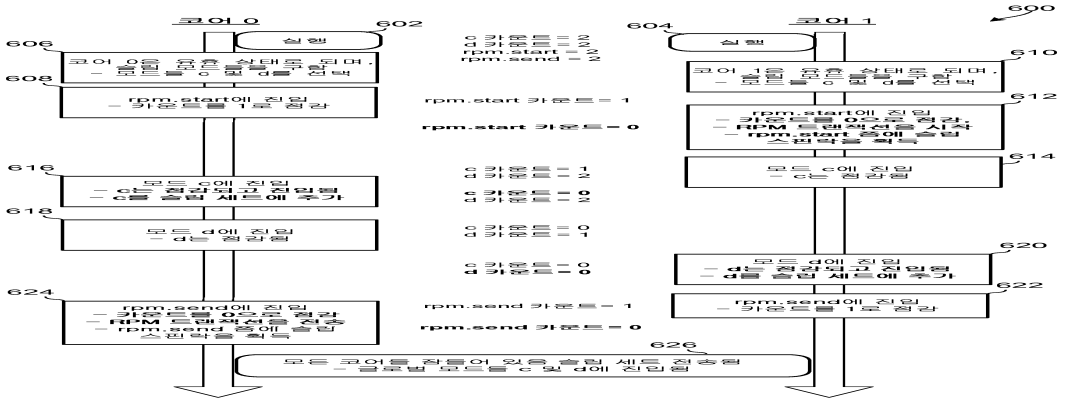
전체 청구항 수 : 총 76 항 심사관 : 김곤희

(54) 발명의 명칭 **멀티코어 컴퓨팅 디바이스들에 대한 다이나믹 슬립**

(57) 요 약

양태들은 특정 디바이스의 허용가능한 시스템 레이턴시들, 다이나믹 동작 조건들 (예컨대, 온도), 예상된 유휴 시간, 및 고유 전기 특성들에 의존하여 저전력 모드로 선택된 자원들을 배치시키는 것에 의해 최고의 시스템 전력 절약을 제공하는 저전력 구성을 결정하도록 멀티-코어 프로세서 또는 시스템 온 칩을 인에이블시킨다. 그 코어들/프로세싱 유닛들의 각각은 대칭 방식으로 처리되고, 각각의 코어는, 복잡한 핸드셰이킹 또는 시그널링 동작들을 수행하는 일 없이, 그것의 동작 상태를 다른 코어들과는 독립적으로 선택할 수도 있다.

대 표 도



(72) 발명자

가르가쉬 노먼 에스

미국 92121 캘리포니아주 샌디에고 모어하우스 드라이브 5775

아벨 마이클

미국 92121 캘리포니아주 샌디에고 모어하우스 드라이브 5775

명 세 서

청구범위

청구항 1

2개 이상의 실행 환경들에 의해 공유되는 자원들을 관리하기 위한 자원 전력 관리자 (resource power manager; RPM) 프로세서를 포함하는 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법으로서,

상기 멀티코어 컴퓨팅 디바이스의 하나를 초과하는 코어에 의해 공유되는 각각의 저전력 자원에 대한 참조 카운트를 메모리에 유지하는 단계;

스레드들을 실행하는 코어들과 유휴 스레드들을 실행하는 코어들 사이에서 공유되는 변수를 액세스하는 것에 의해 어떤 코어들이 슬립 상태에 있는지를 결정하는 단계로서, 공유된 상기 변수는 유휴 스레드의 진입 함수의 수행시 각각의 코어에 의해 증가되고 상기 유휴 스레드의 종료 함수의 수행시 각각의 코어에 의해 감소되는, 상기 결정하는 단계;

메모리 로케이션에 연관된 상기 저전력 자원의 각각의 코어의 사용량 및 각각의 코어의 전력 상태에 기초하여 상기 메모리에 저장된 값을 독립적으로 조정하는 단계; 및

상기 메모리 로케이션에서의 값을 이용하여 시스템 저전력 구성 (configuration) 에 진입하는 때를 결정하는 단계를 포함하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 2

제 1 항에 있어서,

상기 멀티코어 컴퓨팅 디바이스의 각각의 코어는 상기 멀티코어 컴퓨팅 디바이스에서의 하나 이상의 다른 코어들의 운영 체제와는 상이한 운영 체제 하에서 동작하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 3

제 1 항에 있어서,

상기 멀티코어 컴퓨팅 디바이스의 적어도 하나의 코어는 상기 코어를 적어도 하나의 다른 코어의 동작들에 바인딩하는 적어도 하나의 자원을 공유하고, 각각의 코어는 다른 코어들의 각각과는 독립적으로 저전력 모드들에 진입하고 종료하도록 구성되는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 4

제 1 항에 있어서,

플래그 비트 설정에 기초하여 저전력 모드에 배치될 수도 있는 자원들을 식별하는 단계;

레이턴시 요건을 코어 단위 (per-core) 기반 또는 글로벌 기반으로 등록하는 단계;

등록된 레이턴시 요건들로부터 가장 엄격한 레이턴시 요건을 선택하는 단계;

상기 멀티코어 컴퓨팅 디바이스 상에서, 상기 코어 상에 존재하는 레이턴시 제한들에 기초하여, 선택된 상기 가장 엄격한 레이턴시 요건을 초과하는 조합된 레이턴시 요건을 가지는, 저전력 자원 모드들의 임의의 조합, 또는 임의의 저전력 자원 모드를 제거하기 위해 저전력 모드에 배치될 수도 있는 각각의 자원에 대한 저전력 모드들을 평가하는 단계;

잠재적 전력 절약량을 극대화하고 현재 코어에 대한 선택된 최악의 레이턴시 요건 이하의 총 레이턴시 요건을 갖는 저전력 자원 모드들의 조합을 선택하는 단계; 및

식별된 상기 자원들의 각각에 대한 선택된 상기 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 선택된 상기 조합에 진입하는 단계를 더 포함하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 5

제 4 항에 있어서,

상기 식별된 자원들의 각각에 대한 상기 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 상기 선택된 조합에 진입하는 단계는, 공유된 모드들에 대해, 참조 카운트 값이 0과 동일한 경우에 상기 진입 함수를 실행하는 단계를 포함하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 6

제 4 항에 있어서,

상기 현재 코어에 대한 예상된 유휴 시간에 대해, 현재 온도에서의 단위 시간당 잠재적 전력 절약량 곱하기 예상된 유휴 시간에 기초하여, 각각의 평가된 저전력 자원 모드의 잠재적 전력 절약량을 결정하는 단계를 더 포함하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 7

제 1 항에 있어서,

자원을 자동 무효 (auto-invalidate) 라고 마킹하는 단계; 및

상기 RPM 프로세서에서, 마킹된 상기 자원에 연관된 슬립 세트를 무효화하는 단계를 더 포함하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 8

제 1 항에 있어서,

이전의 트랜잭션으로부터의 자원들이 그것들의 슬립 세트에 진입하지 않도록 상기 이전의 트랜잭션을 무효화하는 단계를 더 포함하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 9

제 1 항에 있어서,

상기 멀티코어 컴퓨팅 디바이스에서의 각각의 코어는, 당해 코어의 상기 전력 상태를 제어하고 상기 코어가 그것의 저전력 모드에 진입하는 경우에 상기 RPM 프로세서와 핸드셰이크하는 서브시스템 전력 관리 (subsystem power management; SPM) 하드웨어 블록을 가지며,

상기 방법은,

RPM 슬립 드라이버에서, 스핀락 (spinlock) 이 현재 홀딩되고 있다는 것을 검출하는 단계;

상기 스핀락이 현재 홀딩되고 있다는 검출에 응답하여 트랜잭션을 무시하고 슬립 세트 트랜잭션을 전송하지 않는 단계; 및

각각의 코어의 상기 SPM 하드웨어 블록이 상기 RPM과의 핸드셰이크를 수행함에도 불구하고, 상기 RPM 프로세서에 대해 상기 슬립 세트를 적용하지 않는 단계를 더 포함하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 10

제 1 항에 있어서,

RPM 드라이버에서, 액티브 콘텍스트에서의 스핀락을 수신하는 단계;

상기 멀티코어 컴퓨팅 디바이스의 제 1 코어의 슬립 모드에 진입하는 단계;

RPM 확인응답 인터럽트를 대기하는 동안에 상기 멀티코어 컴퓨팅 디바이스의 제 0 코어에 대한 유휴 프로세스를 개시하는 단계;

상기 제 0 코어에 대한 슬립 모드들을 구하고 (solving) 상기 제 0 코어가 진입할 글로벌 저전력 모드를 선택하는 단계;

상기 제 0 코어에 대한 상기 글로벌 저전력 모드에 진입하는 단계;

상기 제 0 코어에 대한 상기 RPM 확인응답 인터럽트를 수신하는 단계; 및

상기 스핀락을 해제하는 단계를 더 포함하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 11

제 1 항에 있어서,

제 0 코어에 대한 RPM 메시지의 전송을 개시하는 단계;

RPM 드라이버에서, 운영 체제 잠금을 수신하는 단계;

상기 멀티코어 컴퓨팅 디바이스의 제 1 코어에 대한 슬립 모드에 진입하고 RPM 트랜잭션을 만드는 단계;

상기 제 1 코어에 대한 슬립 세트의 전송을 개시하는 단계;

상기 제 1 코어로부터 상기 RPM 드라이버에 대한 슬립 세트 트랜잭션 요청을 수신하고 상기 운영 체제 잠금이 홀딩되는지를 결정하기 위해 검사하는 단계; 및

상기 운영 체제 잠금이 홀딩된다고 결정되면 슬립 세트 트랜잭션 요청을 무시하고 인터럽트 신호를 상기 제 1 코어로 전송하는 단계를 더 포함하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 12

제 1 항에 있어서,

다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 단계를 더 포함하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 13

제 12 항에 있어서,

다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 상기 단계는, RPM 시작 동작 동안에 그리고 RPM 정지 동작 동안에 상기 스핀락을 홀딩하는 단계를 포함하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 14

제 12 항에 있어서,

상기 멀티코어 컴퓨팅 디바이스에서의 각각의 코어는 슬립 세트의 서로소 부분들 (disjoint parts) 에 기록하고, 다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 상기 단계는 자원들을 상기 슬립 세트에 추가하는 경우에 상기 스핀락을 해제시키는 단계를 포함하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 15

메모리; 및

상기 메모리에 연결된 하나 이상의 프로세서들을 포함하며,

상기 하나 이상의 프로세서들은 2개 이상의 실행 환경들에 의해 공유되는 자원들을 관리하기 위한 자원 전력 관리자 (resource power manager; RPM) 프로세서를 포함하고, 상기 하나 이상의 프로세서들은 멀티코어 컴퓨팅 디바이스가:

상기 멀티코어 컴퓨팅 디바이스의 하나를 초과하는 코어에 의해 공유되는 각각의 저전력 자원에 대한 참조 카운트를 메모리에 유지하는 동작;

스레드들을 실행하는 코어들과 유휴 스레드들을 실행하는 코어들 사이에서 공유되는 변수를 액세스하는 것에 의해 어떤 코어들이 슬립 상태에 있는지를 결정하는 동작으로서, 공유된 상기 변수는 유휴 스레드의 진입 함수의 수행시 각각의 코어에 의해 증가되고 상기 유휴 스레드의 종료 함수의 수행시 각각의 코어에 의해 감소되는, 상기 결정하는 동작;

메모리 로케이션에 연관된 상기 저전력 자원의 각각의 코어의 사용량 및 각각의 코어의 전력 상태에 기초하여 상기 메모리에 저장된 값을 독립적으로 조정하는 동작; 및

상기 메모리 로케이션에서의 값을 이용하여 시스템 저전력 구성에 진입하는 때를 결정하는 동작을 포함하는 동작들을 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 16

제 15 항에 있어서,

상기 하나 이상의 프로세서들은 상기 멀티코어 컴퓨팅 디바이스의 각각의 코어가 상기 멀티코어 컴퓨팅 디바이스에서의 하나 이상의 다른 코어들의 운영 체제와는 상이한 운영 체제 하에서 동작하도록 하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 17

제 15 항에 있어서,

상기 하나 이상의 프로세서들은 상기 멀티코어 컴퓨팅 디바이스의 적어도 하나의 코어가 상기 코어를 적어도 하나의 다른 코어의 동작들에 바인딩하는 적어도 하나의 자원을 공유하고, 각각의 코어가 다른 코어들의 각각과는 독립적으로 저전력 모드들에 진입하고 종료하도록 구성되도록 하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 18

제 15 항에 있어서,

상기 하나 이상의 프로세서들은,

플래그 비트 설정에 기초하여 저전력 모드에 배치될 수도 있는 자원들을 식별하는 동작;

레이턴시 요건을 코어 단위 기반 또는 글로벌 기반으로 등록하는 동작;

등록된 레이턴시 요건들로부터 가장 엄격한 레이턴시 요건을 선택하는 동작;

상기 멀티코어 컴퓨팅 디바이스 상에서, 상기 코어 상에 존재하는 레이턴시 제한들에 기초하여, 선택된 상기 가장 엄격한 레이턴시 요건을 초과하는 조합된 레이턴시 요건을 가지는, 저전력 자원 모드들의 임의의 조합, 또는 임의의 저전력 자원 모드를 제거하기 위해 저전력 모드에 배치될 수도 있는 각각의 자원에 대한 저전력 모드들을 평가하는 동작;

잠재적 전력 절약량을 극대화하고 현재 코어에 대한 선택된 최악의 레이턴시 요건 이하의 총 레이턴시 요건을 갖는 저전력 자원 모드들의 조합을 선택하는 동작; 및

식별된 상기 자원들의 각각에 대한 선택된 상기 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 선택된 상기 조합에 진입하는 동작을 더 포함하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 19

제 18 항에 있어서,

상기 하나 이상의 프로세서들은 상기 식별된 자원들의 각각에 대한 상기 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 상기 선택된 조합에 진입하는 동작이, 공유된 모드들에 대해, 참조 카운트 값이 0과 동일한 경우에 상기 진입 함수를 실행하는 동작을 포함하도록 하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 20

제 18 항에 있어서,

상기 하나 이상의 프로세서들은,

상기 현재 코어에 대한 예상된 유휴 시간에 대해, 현재 온도에서의 단위 시간당 잠재적 전력 절약량 곱하기 예상된 유휴 시간에 기초하여, 각각의 평가된 저전력 자원 모드의 잠재적 전력 절약량을 결정하는 동작을 더 포함하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 21

제 15 항에 있어서,

상기 하나 이상의 프로세서들은,

자원을 자동 무효라고 마킹하는 동작; 및

상기 RPM 프로세서에서, 마킹된 상기 자원에 연관된 슬립 세트를 무효화하는 동작을 더 포함하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 22

제 15 항에 있어서,

상기 하나 이상의 프로세서들은,

이전의 트랜잭션으로부터의 자원들이 그것들의 슬립 세트에 진입하지 않도록 상기 이전의 트랜잭션을 무효화하는 동작을 더 포함하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 23

제 15 항에 있어서,

상기 멀티코어 컴퓨팅 디바이스는 코어의 전력 상태를 제어하고 상기 코어가 그것의 저전력 모드에 진입하는 경우에 상기 RPM 프로세서와 핸드셰이크하도록 구성된 서브시스템 전력 관리 (SPM) 하드웨어 블록을 더 포함하며; 및

상기 하나 이상의 프로세서들은,

RPM 슬립 드라이버에서, 스핀락이 현재 홀딩되고 있다는 것을 검출하는 동작;

상기 스핀락이 현재 홀딩되고 있다는 검출에 응답하여 트랜잭션을 무시하고 슬립 세트 트랜잭션을 전송하지 않는 동작; 및

각각의 코어의 상기 SPM 하드웨어 블록이 상기 RPM과의 핸드셰이크를 수행함에도 불구하고, 상기 RPM 프로세서에 대해 상기 슬립 세트를 적용하지 않는 동작을 더 포함하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 24

제 15 항에 있어서,

상기 하나 이상의 프로세서들은,

RPM 드라이버에서, 액티브 콘텍스트에서의 스핀락을 수신하는 동작;

상기 멀티코어 컴퓨팅 디바이스의 제 1 코어의 슬립 모드에 진입하는 동작;

RPM 확인응답 인터럽트를 대기하는 동안에 상기 멀티코어 컴퓨팅 디바이스의 제 0 코어에 대한 유휴 프로세스를 개시하는 동작;

상기 제 0 코어에 대한 슬립 모드들을 구하고 상기 제 0 코어가 진입할 글로벌 저전력 모드를 선택하는 동작;

상기 제 0 코어에 대한 상기 글로벌 저전력 모드에 진입하는 동작;

상기 제 0 코어에 대한 상기 RPM 확인응답 인터럽트를 수신하는 동작; 및

상기 스핀락을 해제하는 동작을 더 포함하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 상기 멀티코어 컴퓨팅 디바이스.

청구항 25

제 15 항에 있어서,

상기 하나 이상의 프로세서들은,

제 0 코어에 대한 RPM 메시지의 전송을 개시하는 동작;

RPM 드라이버에서, 운영 체제 잠금을 수신하는 동작;

상기 멀티코어 컴퓨팅 디바이스의 제 1 코어에 대한 슬립 모드에 진입하고 RPM 트랜잭션을 만드는 동작;

상기 제 1 코어에 대한 슬립 세트의 전송을 개시하는 동작;

상기 제 1 코어로부터 상기 RPM 드라이버에 대한 슬립 세트 트랜잭션 요청을 수신하고 상기 운영 체제 잠금이 홀딩되는지를 결정하기 위해 검사하는 동작; 및

상기 운영 체제 잠금이 홀딩된다고 결정되면 슬립 세트 트랜잭션 요청을 무시하고 인터럽트 신호를 상기 제 1 코어로 전송하는 동작을 더 포함하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 26

제 15 항에 있어서,

상기 하나 이상의 프로세서들은 다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 동작을 더 포함하는 동작들을 수행하기 위한 프로세서 실행가능 명령들로 상기 프로세서가 구성되도록 하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 27

제 26 항에 있어서,

상기 하나 이상의 프로세서들은 다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 상기 동작이, RPM 시작 동작 동안에 그리고 RPM 정지 동작 동안에 상기 스핀락을 홀딩하는 동작을 포함하도록 하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 28

제 26 항에 있어서,

상기 하나 이상의 프로세서들은 상기 멀티코어 컴퓨팅 디바이스에서의 각각의 코어가 슬립 세트의 서로소 부분들에 기록하도록 하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되고, 다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 상기 동작은, 자원들을 상기 슬립 세트에 추가하는 경우에 상기 스핀락을 해제시키는 동작을 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 29

프로세서 실행가능 소프트웨어 명령들이 저장된 비일시적 컴퓨터 판독가능 저장 매체로서,

저장된 상기 프로세서 실행가능 소프트웨어 명령들은 프로세서로 하여금, 2개 이상의 실행 환경들에 의해 공유

되는 자원들을 관리하기 위한 자원 전력 관리자 (resource power manager; RPM) 프로세서를 포함하는 멀티코어 컴퓨팅 디바이스에서 전력을 보존하기 위한 동작들을 수행하게 하도록 구성되며,

상기 동작들은,

상기 멀티코어 컴퓨팅 디바이스의 하나를 초과하는 코어에 의해 공유되는 각각의 저전력 자원에 대한 참조 카운트를 메모리에 유지하는 동작;

스레드들을 실행하는 코어들과 유휴 스레드들을 실행하는 코어들 사이에서 공유되는 변수를 액세스하는 것에 의해 어떤 코어들이 슬립 상태에 있는지를 결정하는 동작으로서, 공유된 상기 변수는 유휴 스레드의 진입 함수의 수행시 각각의 코어에 의해 증가되고 상기 유휴 스레드의 종료 함수의 수행시 각각의 코어에 의해 감소되는, 상기 결정하는 동작;

메모리 로케이션에 연관된 상기 저전력 자원의 각각의 코어의 사용량 및 각각의 코어의 전력 상태에 기초하여 상기 메모리에 저장된 값을 독립적으로 조정하는 동작; 및

상기 메모리 로케이션에서의 값을 이용하여 시스템 저전력 구성에 진입하는 때를 결정하는 동작을 포함하는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 30

제 29 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 프로세서로 하여금, 상기 멀티코어 컴퓨팅 디바이스의 각각의 코어가 상기 멀티코어 컴퓨팅 디바이스에서의 하나 이상의 다른 코어들의 운영 체제와는 상이한 운영 체제 하에서 동작하게 하는 동작들을 수행하게 하도록 구성되는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 31

제 29 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 프로세서로 하여금, 상기 멀티코어 컴퓨팅 디바이스의 적어도 하나의 코어가 상기 코어를 적어도 하나의 다른 코어의 동작들에 바인딩하는 적어도 하나의 자원을 공유하고, 각각의 코어는 다른 코어들의 각각과는 독립적으로 저전력 모드들에 진입하고 종료하도록 구성되게 하는 동작들을 수행하게 하도록 구성되는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 32

제 29 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 프로세서로 하여금,

플래그 비트 설정에 기초하여 저전력 모드에 배치될 수도 있는 자원들을 식별하는 동작;

레이턴시 요건을 코어 단위 기반 또는 글로벌 기반으로 등록하는 동작;

등록된 레이턴시 요건들로부터 가장 엄격한 레이턴시 요건을 선택하는 동작;

상기 멀티코어 컴퓨팅 디바이스 상에서, 상기 코어 상에 존재하는 레이턴시 제한들에 기초하여, 선택된 상기 가장 엄격한 레이턴시 요건을 초과하는 조합된 레이턴시 요건을 가지는, 저전력 자원 모드들의 임의의 조합, 또는 임의의 저전력 자원 모드를 제거하기 위해 저전력 모드에 배치될 수도 있는 각각의 자원에 대한 저전력 모드들을 평가하는 동작;

잠재적 전력 절약량을 극대화하고 현재 코어에 대한 선택된 최악의 레이턴시 요건 이하의 총 레이턴시 요건을 갖는 저전력 자원 모드들의 조합을 선택하는 동작; 및

식별된 상기 자원들의 각각에 대한 선택된 상기 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 선택된 상기 조합에 진입하는 동작을 더 포함하는 동작들을 수행하게 하도록 구성되는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 33

제 32 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 프로세서로 하여금, 상기 식별된 자원들의 각각에 대한 상기 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 상기 선택된 조합에 진입하는 동작이, 공유된 모드들에 대해, 참조 카운트 값이 0과 동일한 경우에 상기 진입 함수를 실행하는 동작을 포함하게 하는 동작들을 수행하게 하도록 구성되는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 34

제 32 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 프로세서로 하여금, 상기 현재 코어에 대한 예상된 유휴 시간에 대해, 현재 온도에서의 단위 시간당 잠재적 전력 절약량 곱하기 예상된 유휴 시간에 기초하여, 각각의 평가된 저전력 자원 모드의 잠재적 전력 절약량을 결정하는 동작을 더 포함하는 동작들을 수행하게 하도록 구성되는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 35

제 29 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 상기 프로세서로 하여금,

자원을 자동 무효라고 마킹하는 동작; 및

상기 RPM 프로세서에서, 마킹된 상기 자원에 연관된 슬립 세트를 무효화하는 동작을 더 포함하는 동작들을 수행하게 하도록 구성되는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 36

제 29 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 상기 프로세서로 하여금,

이전의 트랜잭션으로부터의 자원들이 그것들의 슬립 세트에 진입하지 않도록 상기 이전의 트랜잭션을 무효화하는 동작을 더 포함하는 동작들을 수행하게 하도록 구성되는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 37

제 29 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 프로세서로 하여금, 상기 멀티코어 컴퓨팅 디바이스에서의 각각의 코어가, 당해 코어의 상기 전력 상태를 제어하고 상기 코어가 그것의 저전력 모드에 진입하는 경우에 상기 RPM 프로세서와 핸드셰이크하는 서브시스템 전력 관리 (SPM) 하드웨어 블록을 가지게 하는 동작들을 수행하게 하도록 구성되며,

상기 동작들은,

RPM 슬립 드라이버에서, 스핀락이 현재 홀딩되고 있다는 것을 검출하는 동작;

상기 스핀락이 현재 홀딩되고 있다는 검출에 응답하여 트랜잭션을 무시하고 슬립 세트 트랜잭션을 전송하지 않는 동작; 및

각각의 코어의 상기 SPM 하드웨어 블록이 상기 RPM과의 핸드셰이크를 수행함에도 불구하고, 상기 RPM 프로세서에 대해 상기 슬립 세트를 적용하지 않는 동작을 더 포함하는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 38

제 29 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 상기 프로세서로 하여금,

RPM 드라이버에서, 액티브 콘텍스트에서의 스핀락을 수신하는 동작;

상기 멀티코어 컴퓨팅 디바이스의 제 1 코어의 슬립 모드에 진입하는 동작;

RPM 확인응답 인터럽트를 대기하는 동안에 상기 멀티코어 컴퓨팅 디바이스의 제 0 코어에 대한 유휴 프로세스를 개시하는 동작;

상기 제 0 코어에 대한 슬립 모드들을 구하고 상기 제 0 코어가 진입할 글로벌 저전력 모드를 선택하는 동작;

상기 제 0 코어에 대한 상기 글로벌 저전력 모드에 진입하는 동작;

상기 제 0 코어에 대한 상기 RPM 확인응답 인터럽트를 수신하는 동작; 및

상기 스핀락을 해제하는 동작을 더 포함하는 동작들을 수행하게 하도록 구성되는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 39

제 29 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 상기 프로세서로 하여금,

제 0 코어에 대한 RPM 메시지의 전송을 개시하는 동작;

RPM 드라이버에서, 운영 체제 잠금을 수신하는 동작;

상기 멀티코어 컴퓨팅 디바이스의 제 1 코어에 대한 슬립 모드에 진입하고 RPM 트랜잭션을 만드는 동작;

상기 제 1 코어에 대한 슬립 세트의 전송을 개시하는 동작;

상기 제 1 코어로부터 상기 RPM 드라이버에 대한 슬립 세트 트랜잭션 요청을 수신하고 상기 운영 체제 잠금이 홀딩되는지를 결정하기 위해 검사하는 동작; 및

상기 운영 체제 잠금이 홀딩된다고 결정되면 슬립 세트 트랜잭션 요청을 무시하고 인터럽트 신호를 상기 제 1 코어로 전송하는 동작을 더 포함하는 동작들을 수행하게 하도록 구성되는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 40

제 29 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 프로세서로 하여금, 다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 동작을 더 포함하는 동작들을 수행하게 하도록 구성되는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 41

제 40 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 프로세서로 하여금, 다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 상기 동작은, RPM 시작 동작 동안에 그리고 RPM 정지 동작 동안에 상기 스핀락을 홀딩하는 동작을 포함하게 하는 동작들을 수행하게 하도록 구성되는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 42

제 40 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 프로세서로 하여금, 상기 멀티코어 컴퓨팅 디바이스에서의 각각의 코어가 슬립 세트의 서로소 부분들에 기록하도록 하는 동작들을 수행하게 하도록 구성되고, 다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 상기 동작은 자원들을 상기 슬립 세트에 추가하는 경우에 상기 스핀락을 해제시키는 동작을 포함하는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 43

자원 전력 관리자 (resource power manager; RPM) 프로세서를 통해 멀티코어 컴퓨팅 디바이스에서의 2개 이상의 실행 환경들에 의해 공유되는 자원들을 관리하는 수단;

상기 멀티코어 컴퓨팅 디바이스의 하나를 초과하는 코어에 의해 공유되는 각각의 저전력 자원에 대한 참조 카운트를 메모리에 유지하는 수단;

스레드들을 실행하는 코어들과 유휴 스레드들을 실행하는 코어들 사이에서 공유되는 변수를 액세스하는 것에 의해 어떤 코어들이 슬립 상태에 있는지를 결정하는 수단으로서, 공유된 상기 변수는 유휴 스레드의 진입 함수의 수행시 각각의 코어에 의해 증가되고 상기 유휴 스레드의 종료 함수의 수행시 각각의 코어에 의해 감소되는, 상기 결정하는 수단;

메모리 로케이션에 연관된 상기 저전력 자원의 각각의 코어의 사용량 및 각각의 코어의 전력 상태에 기초하여 상기 메모리에 저장된 값을 독립적으로 조정하는 수단; 및

상기 메모리 로케이션에서의 값을 이용하여 시스템 저전력 구성에 진입하는 때를 결정하는 수단을 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 44

제 43 항에 있어서,

상기 멀티코어 컴퓨팅 디바이스의 각각의 코어를 상기 멀티코어 컴퓨팅 디바이스에서의 하나 이상의 다른 코어들의 운영 체제와는 상이한 운영 체제 하에서 동작시키는 수단을 더 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 45

제 43 항에 있어서,

상기 멀티코어 컴퓨팅 디바이스의 적어도 하나의 코어는 상기 코어를 적어도 하나의 다른 코어의 동작들에 바인딩하는 적어도 하나의 자원을 공유하도록, 그리고 각각의 코어가 다른 코어들의 각각과는 독립적으로 저전력 모드들에 진입하고 종료하게 구성되도록 상기 멀티코어 컴퓨팅 디바이스를 구성하는 수단을 더 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 46

제 43 항에 있어서,

플래그 비트 설정에 기초하여 저전력 모드에 배치될 수도 있는 자원들을 식별하는 수단;

레이턴시 요건을 코어 단위 기반 또는 글로벌 기반으로 등록하는 수단;

등록된 레이턴시 요건들로부터 가장 엄격한 레이턴시 요건을 선택하는 수단;

상기 멀티코어 컴퓨팅 디바이스 상에서, 상기 코어 상에 존재하는 레이턴시 제한들에 기초하여, 선택된 상기 가장 엄격한 레이턴시 요건을 초과하는 조합된 레이턴시 요건을 가지는, 저전력 자원 모드들의 임의의 조합, 또는 임의의 저전력 자원 모드를 제거하기 위해 저전력 모드에 배치될 수도 있는 각각의 자원에 대한 저전력 모드들을 평가하는 수단;

잠재적 전력 절약량을 극대화하고 현재 코어에 대한 선택된 최악의 레이턴시 요건 이하의 총 레이턴시 요건을 갖는 저전력 자원 모드들의 조합을 선택하는 수단; 및

식별된 상기 자원들의 각각에 대한 선택된 상기 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 선택된 상기 조합에 진입하는 수단을 더 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 47

제 46 항에 있어서,

상기 식별된 자원들의 각각에 대한 상기 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 상기 선택된 조합에 진입하는 수단은, 공유된 모드들에 대해, 참조 카운트 값이 0과 동일한 경우에 상기 진입 함수를 실행하는 수단을 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 48

제 46 항에 있어서,

상기 현재 코어에 대한 예상된 유휴 시간에 대해, 현재 온도에서의 단위 시간당 잠재적 전력 절약량 곱하기 예상된 유휴 시간에 기초하여, 각각의 평가된 저전력 자원 모드의 잠재적 전력 절약량을 결정하는 수단을 더 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 49

제 43 항에 있어서,

자원을 자동 무효라고 마킹하는 수단; 및

상기 RPM 프로세서에서, 마킹된 상기 자원에 연관된 슬립 세트를 무효화하는 수단을 더 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 50

제 43 항에 있어서,

이전의 트랜잭션으로부터의 자원들이 그것들의 슬립 세트에 진입하지 않도록 상기 이전의 트랜잭션을 무효화하는 수단을 더 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 51

제 43 항에 있어서,

상기 멀티코어 컴퓨팅 디바이스에서의 각각의 코어는, 당해 코어의 상기 전력 상태를 제어하고 상기 코어가 그것의 저전력 모드에 진입하는 경우에 상기 RPM 프로세서와 핸드셰이크하는 서브시스템 전력 관리 (SPM) 하드웨어 블록을 가지며,

상기 멀티코어 컴퓨팅 디바이스는,

RPM 슬립 드라이버에서, 스핀락이 현재 홀딩되고 있다는 것을 검출하는 수단;

상기 스핀락이 현재 홀딩되고 있다는 검출에 응답하여 트랜잭션을 무시하고 슬립 세트 트랜잭션을 전송하지 않는 수단; 및

각각의 코어의 상기 SPM 하드웨어 블록이 상기 RPM과의 핸드셰이크를 수행함에도 불구하고, 상기 RPM 프로세서에 대해 상기 슬립 세트를 적용하지 않는 수단을 더 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 52

제 43 항에 있어서,

자원 전력 관리자 (RPM) 드라이버에서, 액티브 콘텍스트에서의 스핀락을 수신하는 수단;

상기 멀티코어 컴퓨팅 디바이스의 제 1 코어의 슬립 모드에 진입하는 수단;

RPM 확인응답 인터럽트를 대기하는 동안에 상기 멀티코어 컴퓨팅 디바이스의 제 0 코어에 대한 유휴 프로세스를 개시하는 수단;

상기 제 0 코어에 대한 슬립 모드들을 구하고 상기 제 0 코어가 진입할 글로벌 저전력 모드를 선택하는 수단;

상기 제 0 코어에 대한 상기 글로벌 저전력 모드에 진입하는 수단;

상기 제 0 코어에 대한 상기 RPM 확인응답 인터럽트를 수신하는 수단; 및

상기 스핀락을 해제시키는 수단을 더 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 53

제 43 항에 있어서,

제 0 코어에 대한 자원 전력 관리자 (RPM) 메시지의 전송을 개시하는 수단;

자원 전력 관리자 (RPM) 드라이버에서, 운영 체제 잠금을 수신하는 수단;

상기 멀티코어 컴퓨팅 디바이스의 제 1 코어에 대한 슬립 모드에 진입하고 RPM 트랜잭션을 만드는 수단;

상기 제 1 코어에 대한 슬립 세트의 전송을 개시하는 수단;

상기 제 1 코어로부터 상기 RPM 드라이버에 대한 슬립 세트 트랜잭션 요청을 수신하고 상기 운영 체제 잠금이 홀딩되는지를 결정하기 위해 검사하는 수단; 및

상기 운영 체제 잠금이 홀딩된다고 결정되면 슬립 세트 트랜잭션 요청을 무시하고 인터럽트 신호를 상기 제 1 코어로 전송하는 수단을 더 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 54

제 43 항에 있어서,

다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 수단을 더 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 55

제 54 항에 있어서,

다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 상기 수단은, RPM 시작 동작 동안에 그리고 RPM 정지 동작 동안에 상기 스핀락을 홀딩하는 수단을 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 56

제 54 항에 있어서,

상기 멀티코어 컴퓨팅 디바이스에서의 각각의 코어가 슬립 세트의 서로소 부분들에 기록되도록 상기 코어들을 구성하는 수단을 더 포함하고, 다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 상기 수단은 자원들을 상기 슬립 세트에 추가하는 경우에 상기 스핀락을 해제시키는 수단을 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 57

멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법으로서,

상기 멀티코어 컴퓨팅 디바이스의 하나를 초과하는 코어에 의해 공유되는 각각의 저전력 자원에 대한 참조 카운트를 메모리에 유지하는 단계;

메모리 로케이션에 연관된 상기 저전력 자원의 각각의 코어의 사용량 및 각각의 코어의 전력 상태에 기초하여 상기 메모리에 저장된 값을 독립적으로 조정하는 단계;

상기 메모리 로케이션에서의 값을 이용하여 시스템 저전력 구성에 진입하는 때를 결정하는 단계; 및

플래그 비트 설정에 기초하여 저전력 모드에 배치될 수도 있는 자원들을 식별하는 단계;

레이턴시 요건을 코어 단위 기반 또는 글로벌 기반으로 등록하는 단계;

등록된 레이턴시 요건들로부터 가장 엄격한 레이턴시 요건을 선택하는 단계;

상기 코어 상에 존재하는 레이턴시 제한들에 기초하여, 선택된 상기 가장 엄격한 레이턴시 요건을 초과하는 조합된 레이턴시 요건을 가지는, 저전력 자원 모드들의 임의의 조합, 또는 임의의 저전력 자원 모드를 제거하기 위해 저전력 모드에 배치될 수도 있는 각각의 자원에 대한 저전력 모드들을 평가하는 단계;

잠재적 전력 절약량을 극대화하고 현재 코어에 대한 선택된 최악의 레이턴시 요건 이하의 총 레이턴시 요건을 갖는 저전력 자원 모드들의 조합을 선택하는 단계; 및

식별된 상기 자원들의 각각에 대한 선택된 상기 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실

행하는 것에 의해 저전력 자원 모드들의 선택된 상기 조합에 진입하는 단계

에 의해 상기 시스템 저전력 구성에 진입하는 단계를 포함하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 58

제 57 항에 있어서,

상기 멀티코어 컴퓨팅 디바이스의 각각의 코어는 상기 멀티코어 컴퓨팅 디바이스에서의 하나 이상의 다른 코어들의 운영 체제와는 상이한 운영 체제 하에서 동작하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 59

제 57 항에 있어서,

상기 멀티코어 컴퓨팅 디바이스의 적어도 하나의 코어는 상기 코어를 적어도 하나의 다른 코어의 동작들에 바인딩하는 적어도 하나의 자원을 공유하고, 각각의 코어는 다른 코어들의 각각과는 독립적으로 저전력 모드들에 진입하고 종료하도록 구성되는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 60

제 57 항에 있어서,

상기 식별된 자원들의 각각에 대한 상기 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 상기 선택된 조합에 진입하는 단계는, 공유된 모드들에 대해, 참조 카운트 값이 0과 동일한 경우에 상기 진입 함수를 실행하는 단계를 포함하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 61

제 57 항에 있어서,

상기 현재 코어에 대한 예상된 유휴 시간에 대해, 현재 온도에서의 단위 시간당 잠재적 전력 절약량 곱하기 예상된 유휴 시간에 기초하여, 각각의 평가된 저전력 자원 모드의 잠재적 전력 절약량을 결정하는 단계를 더 포함하는, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법.

청구항 62

메모리; 및

상기 메모리에 연결된 하나 이상의 프로세서들을 포함하며,

상기 하나 이상의 프로세서들은, 멀티코어 컴퓨팅 디바이스가,

상기 멀티코어 컴퓨팅 디바이스의 하나를 초과하는 코어에 의해 공유되는 각각의 저전력 자원에 대한 참조 카운트를 메모리에 유지하는 동작;

메모리 로케이션에 연관된 상기 저전력 자원의 각각의 코어의 사용량 및 각각의 코어의 전력 상태에 기초하여 상기 메모리에 저장된 값을 독립적으로 조정하는 동작;

상기 메모리 로케이션에서의 값을 이용하여 시스템 저전력 구성에 진입하는 때를 결정하는 동작; 및

플래그 비트 설정에 기초하여 저전력 모드에 배치될 수도 있는 자원들을 식별하는 것;

레이턴시 요건을 코어 단위 기반 또는 글로벌 기반으로 등록하는 것;

등록된 레이턴시 요건들로부터 가장 엄격한 레이턴시 요건을 선택하는 것;

상기 멀티코어 컴퓨팅 디바이스 상에서, 상기 코어 상에 존재하는 레이턴시 제한들에 기초하여, 선택된 상기 가장 엄격한 레이턴시 요건을 초과하는 조합된 레이턴시 요건을 가지는, 저전력 자원 모드들의 임의의 조합, 또는 임의의 저전력 자원 모드를 제거하기 위해 저전력 모드에 배치될 수도 있는 각각의 자원에 대한 저전력 모드들을 평가하는 것;

잠재적 전력 절약량을 극대화하고 현재 코어에 대한 선택된 최악의 레이턴시 요건 이하의 총 레이턴시 요건을 갖는 저전력 자원 모드들의 조합을 선택하는 것; 및

식별된 상기 자원들의 각각에 대한 선택된 상기 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 선택된 상기 조합에 진입하는 것

에 의해 상기 시스템 저전력 구성에 진입하는 동작을 포함하는 동작들을 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 63

제 62 항에 있어서,

상기 하나 이상의 프로세서들은 상기 멀티코어 컴퓨팅 디바이스의 각각의 코어가 상기 멀티코어 컴퓨팅 디바이스에서의 하나 이상의 다른 코어들의 운영 체제와는 상이한 운영 체제 하에서 동작하도록 하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 64

제 62 항에 있어서,

상기 하나 이상의 프로세서들은 상기 멀티코어 컴퓨팅 디바이스의 적어도 하나의 코어가 상기 코어를 적어도 하나의 다른 코어의 동작들에 바인딩하는 적어도 하나의 자원을 공유하고, 각각의 코어가 다른 코어들의 각각과는 독립적으로 저전력 모드들에 진입하고 종료하도록 구성되도록 하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 65

제 62 항에 있어서,

상기 하나 이상의 프로세서들은 상기 식별된 자원들의 각각에 대한 상기 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 상기 선택된 조합에 진입하는 동작이, 공유된 모드들에 대해, 참조 카운트 값이 0과 동일한 경우에 상기 진입 함수를 실행하는 동작을 포함하도록 하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 66

제 62 항에 있어서,

상기 하나 이상의 프로세서들은,

상기 현재 코어에 대한 예상된 유휴 시간에 대해, 현재 온도에서의 단위 시간당 잠재적 전력 절약량 곱하기 예상된 유휴 시간에 기초하여, 각각의 평가된 저전력 자원 모드의 잠재적 전력 절약량을 결정하는 동작을 더 포함하는 동작들을 상기 멀티코어 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성되는, 멀티코어 컴퓨팅 디바이스.

청구항 67

프로세서 실행가능 소프트웨어 명령들이 저장된 비일시적 컴퓨터 판독가능 저장 매체로서,

저장된 상기 프로세서 실행가능 소프트웨어 명령들은 프로세서로 하여금, 멀티코어 컴퓨팅 디바이스에서 전력을 보존하기 위한 동작들을 수행하게 하도록 구성되며,

상기 동작들은,

상기 멀티코어 컴퓨팅 디바이스의 하나를 초과하는 코어에 의해 공유되는 각각의 저전력 자원에 대한 참조 카운트를 메모리에 유지하는 동작;

메모리 로케이션에 연관된 상기 저전력 자원의 각각의 코어의 사용량 및 각각의 코어의 전력 상태에 기초하여 상기 메모리에 저장된 값을 독립적으로 조정하는 동작;

상기 메모리 로케이션에서의 값을 이용하여 시스템 저전력 구성에 진입하는 때를 결정하는 동작; 및

플래그 비트 설정에 기초하여 저전력 모드에 배치될 수도 있는 자원들을 식별하는 것;

레이턴시 요건을 코어 단위 기반 또는 글로벌 기반으로 등록하는 것;

등록된 레이턴시 요건들로부터 가장 엄격한 레이턴시 요건을 선택하는 동작;

상기 멀티코어 컴퓨팅 디바이스 상에서, 상기 코어 상에 존재하는 레이턴시 제한들에 기초하여, 선택된 상기 가장 엄격한 레이턴시 요건을 초과하는 조합된 레이턴시 요건을 가지는, 저전력 자원 모드들의 임의의 조합, 또는 임의의 저전력 자원 모드를 제거하기 위해 저전력 모드에 배치될 수도 있는 각각의 자원에 대한 저전력 모드들을 평가하는 것;

잠재적 전력 절약량을 극대화하고 현재 코어에 대한 선택된 최악의 레이턴시 요건 이하의 총 레이턴시 요건을 갖는 저전력 자원 모드들의 조합을 선택하는 것; 및

식별된 상기 자원들의 각각에 대한 선택된 상기 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 선택된 상기 조합에 진입하는 것

에 의해 상기 시스템 저전력 구성에 진입하는 동작을 포함하는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 68

제 67 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 프로세서로 하여금, 상기 멀티코어 컴퓨팅 디바이스의 각각의 코어가 상기 멀티코어 컴퓨팅 디바이스에서의 하나 이상의 다른 코어들의 운영 체제와는 상이한 운영 체제 하에서 동작하게 하는 동작들을 수행하게 하도록 구성되는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 69

제 67 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 프로세서로 하여금, 상기 멀티코어 컴퓨팅 디바이스의 적어도 하나의 코어가 상기 코어를 적어도 하나의 다른 코어의 동작들에 바인딩하는 적어도 하나의 자원을 공유하고, 각각의 코어가 다른 코어들의 각각과는 독립적으로 저전력 모드들에 진입하고 종료하도록 구성되게 하는 동작들을 수행하게 하도록 구성되는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 70

제 67 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 프로세서로 하여금, 상기 식별된 자원들의 각각에 대한 상기 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 상기 선택된 조합에 진입하는 동작이, 공유된 모드들에 대해, 참조 카운트 값이 0과 동일한 경우에 상기 진입 함수를 실행하는 동작을 포함하게 하는 동작들을 수행하게 하도록 구성되는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 71

제 67 항에 있어서,

상기 저장된 프로세서 실행가능 소프트웨어 명령들은 프로세서로 하여금, 상기 현재 코어에 대한 예상된 유휴 시간에 대해, 현재 온도에서의 단위 시간당 잠재적 전력 절약량 곱하기 예상된 유휴 시간에 기초하여, 각각의 평가된 저전력 자원 모드의 잠재적 전력 절약량을 결정하는 동작을 더 포함하는 동작들을 수행하게 하도록 구성되는, 비일시적 컴퓨터 판독가능 저장 매체.

청구항 72

멀티코어 컴퓨팅 디바이스의 하나를 초과하는 코어에 의해 공유되는 각각의 저전력 자원에 대한 참조 카운트를 메모리에 유지하는 수단;

메모리 로케이션에 연관된 상기 저전력 자원의 각각의 코어의 사용량 및 각각의 코어의 전력 상태에 기초하여

상기 메모리에 저장된 값을 독립적으로 조정하는 수단;

상기 메모리 로케이션에서의 값을 이용하여 시스템 저전력 구성에 진입하는 때를 결정하는 수단;

플래그 비트 설정에 기초하여 저전력 모드에 배치될 수도 있는 자원들을 식별하는 수단;

레이턴시 요건을 코어 단위 기반 또는 글로벌 기반으로 등록하는 수단;

등록된 레이턴시 요건들로부터 가장 엄격한 레이턴시 요건을 선택하는 수단;

상기 멀티코어 컴퓨팅 디바이스 상에서, 상기 코어 상에 존재하는 레이턴시 제한들에 기초하여, 선택된 상기 가장 엄격한 레이턴시 요건을 초과하는 조합된 레이턴시 요건을 가지는, 저전력 자원 모드들의 임의의 조합, 또는 임의의 저전력 자원 모드를 제거하기 위해 저전력 모드에 배치될 수도 있는 각각의 자원에 대한 저전력 모드들을 평가하는 수단;

잠재적 전력 절약량을 극대화하고 현재 코어에 대한 선택된 최악의 레이턴시 요건 이하의 총 레이턴시 요건을 갖는 저전력 자원 모드들의 조합을 선택하는 수단; 및

식별된 상기 자원들의 각각에 대한 선택된 상기 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 선택된 상기 조합에 진입하는 수단을 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 73

제 72 항에 있어서,

상기 멀티코어 컴퓨팅 디바이스의 각각의 코어를 상기 멀티코어 컴퓨팅 디바이스에서의 하나 이상의 다른 코어들의 운영 체제와는 상이한 운영 체제 하에서 동작시키는 수단을 더 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 74

제 72 항에 있어서,

상기 멀티코어 컴퓨팅 디바이스의 적어도 하나의 코어가 상기 코어를 적어도 하나의 다른 코어의 동작들에 바인딩하는 적어도 하나의 자원을 공유하도록, 그리고 각각의 코어가 다른 코어들의 각각과는 독립적으로 저전력 모드들에 진입하고 종료하게 구성되도록 상기 멀티코어 컴퓨팅 디바이스를 구성하는 수단을 더 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 75

제 72 항에 있어서,

상기 식별된 자원들의 각각에 대한 상기 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 상기 선택된 조합에 진입하는 수단은, 공유된 모드들에 대해, 참조 카운트 값이 0과 동일한 경우에 상기 진입 함수를 실행하는 수단을 포함하는, 멀티코어 컴퓨팅 디바이스.

청구항 76

제 72 항에 있어서,

상기 현재 코어에 대한 예상된 유휴 시간에 대해, 현재 온도에서의 단위 시간당 잠재적 전력 절약량 곱하기 예상된 유휴 시간에 기초하여, 각각의 평가된 저전력 자원 모드의 잠재적 전력 절약량을 결정하는 수단을 더 포함하는, 멀티코어 컴퓨팅 디바이스.

발명의 설명

기 술 분 야

[0001] 관련 출원들

[0002] 본 출원은 2011년 9월 19일자로 출원된 발명의 명칭이 "Dynamic Sleep For Multicore Computing Devices"인 미국 특허 가출원 제61/536,207호를 우선권 주장하며, 그 전체 내용은 참조로 본원에 통합된다.

배 경 기 술

[0003] 셀룰러 및 무선 통신 기술들은 지난 몇 년간 폭발적인 성장을 보였다. 이 성장은 양호한 통신들, 하드웨어, 큰 네트워크들, 및 더 신뢰성 있는 프로토콜들에 의해 가열되었다. 무선 서비스 제공자들은 이제 그들의 고객들에게 특징들 및 서비스들의 줄곧 확장하는 어레이를 제공하고, 사용자들에게 정보, 자원들, 및 통신들에 대한 전례 없는 액세스 레벨들을 제공할 수 있다. 이들 서비스 향상들과 보조를 맞추기 위해, 모바일 전자 디바이스들 (예컨대, 셀룰러 폰들, 테블릿들, 랩톱들 등) 은 그 어느 때보다 더 강력하고 복잡해지고 있다. 예를 들어, 모바일 전자 디바이스들은 이제 일반적으로 시스템-온-칩들 (SoCs) 및/또는 단일 기판 상에 내장된 다수의 마이크로프로세서 코어들을 구비하여, 모바일 디바이스 사용자들이 그들의 모바일 디바이스들 상에서 복잡하고 전력 집약적인 소프트웨어 애플리케이션들을 실행하는 것을 허용한다. 그 결과, 모바일 디바이스의 배터리 수명 및 소비 전력 특성들은 모바일 디바이스들의 소비자들에게 더욱 중요 고려사항들이 되고 있다.

발명의 내용

과제의 해결 수단

[0004] 다양한 양태들은 멀티코어 컴퓨팅 디바이스에서 전력을 보존하는 방법들을 제공하며, 그 방법들은 멀티코어 컴퓨팅 디바이스의 하나를 초과하는 코어들에 의해 공유되는 각각의 저전력 자원에 대한 참조 카운트를 메모리에 유지하는 단계, 메모리 로케이션에 연관된 저전력 자원의 각각의 코어의 사용량 및 각각의 코어의 전력 상태에 기초하여 메모리에 저장된 값을 독립적으로 조정하는 단계, 및 메모리 로케이션에서의 값을 이용하여 시스템 저전력 구성에 진입하는 때를 결정하는 단계를 포함한다. 양태의 방법들은, 플래그 비트 설정에 기초하여 저전력 모드에 배치될 수도 있는 자원들을 식별하는 단계, 레이턴시 요건을 코어 단위 기반 또는 글로벌 기반으로 등록하는 단계, 등록된 레이턴시 요건들로부터 가장 엄격한 레이턴시 요건을 선택하는 단계, 컴퓨팅 디바이스 상에서, 코어 상에 존재하는 레이턴시 제한들에 기초하여, 임의의 저전력 자원 모드, 또는 선택된 가장 엄격한 레이턴시 허용오차를 초과하는 조합된 레이턴시 요건을 가지는 저전력 자원 모드들의 임의의 조합을 제거하기 위해 저전력 모드에 배치될 수도 있는 각각의 자원에 대한 저전력 모드들을 평가하는 단계, 잠재적 전력 절약량을 극대화하고 현재 코어에 대한 선택된 최악의 레이턴시 요건 이하의 총 레이턴시 요건을 갖는 저전력 자원 모드들의 조합을 선택하는 단계, 및 식별된 자원들의 각각에 대한 상기 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 상기 선택된 조합에 진입하는 단계를 더 포함할 수도 있다. 일 양태에서, 식별된 자원들의 각각에 대한 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 선택된 조합에 진입하는 단계는, 공유된 모드들에 대해, 참조 카운트 값이 0과 동일한 경우에 진입 함수를 실행하는 단계를 포함할 수도 있다. 일 양태에서, 그 방법은 현재 코어에 대한 예상된 유휴 시간에 대해, 현재 온도에서의 단위 시간당 잠재적 전력 절약량 곱하기 예상된 유휴 시간에 기초하여, 각각의 평가된 저전력 자원 모드의 잠재적 전력 절약량을 결정하는 단계를 더 포함할 수도 있다. 일 양태에서, 멀티코어 컴퓨팅 디바이스의 각각의 코어는 멀티코어 컴퓨팅 디바이스에서의 하나 이상의 다른 코어들의 운영 체제와는 상이한 운영 체제 하에서 동작할 수도 있다. 일 양태에서, 멀티코어 컴퓨팅 디바이스의 적어도 하나의 코어는 그 코어를 적어도 하나의 다른 코어의 동작들에 바인딩하는 적어도 하나의 자원을 공유하고, 각각의 코어는 다른 코어들의 각각과는 독립적으로 저전력 모드들에 진입하고 종료하도록 구성된다. 일 양태에서, 멀티코어 컴퓨팅 디바이스는 둘 이상의 실행 환경들에 의해 공유된 자원들을 관리하는 자원 전력 관리자 (resource power manager; RPM) 프로세서를 구비할 수도 있다. 일 양태에서, 그 방법은 스레드들을 실행하는 코어들 및 유휴 스레드들을 실행하는 코어들 간에 공유된 변수에 액세스하는 것에 의해 어떤 코어들이 슬립 상태에 있는지를 결정하는 단계를 포함할 수도 있으며, 공유된 변수는 유휴 스레드의 진입 함수의 수행시 각각의 코어에 의해 점증되고 유휴 스레드의 종료 함수의 수행시 각각의 코어에 의해 점감된다. 일 양태에서, 그 방법은 다른 코어가 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 단계를 더 포함할 수도 있다. 일 양태에서, 다른 코어가 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 단계는, RPM 시작 동작 동안에 그리고 RPM 정지 동작 동안에 상기 스핀락을 홀딩하는 단계를 포함할 수도 있다. 일 양태에서, 멀티코어 컴퓨팅 디바이스에서의 각각의 코어는 슬립 세트의 서로소 부분들 (disjoint parts) 에 기록하고, 다른 코어가 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 단계는 자원들을 슬립 세트에 추가하는 경우에 스핀락을 해제시키는 단계를 포함할 수도 있다.

일 양태에서, 그 방법은, 자원을 자동 무효라고 마킹하는 단계, 및 RPM 프로세서에서, 마킹된 자원에 연관된 슬립 세트를 무효화하는 단계를 포함할 수도 있다. 일 양태에서, 그 방법은, 이전의 트랜잭션으로부터의 자

원들이 그것들의 슬립 세트에 바람직하지 않게 진입하지 않도록 이전의 트랜잭션을 무효화하는 단계를 포함할 수도 있다. 일 양태에서, 멀티코어 컴퓨팅 디바이스에서의 각각의 코어는, 그 코어가 그것의 저전력 모드에 진입하는 경우에 당해 코어의 전력 상태 및 RPM 프로세서와의 핸드셰이크들을 제어하는 서브시스템 전력 관리 (SPM) 하드웨어 블록을 가질 수도 있고, 그 방법은, RPM 슬립 드라이버에서, 스핀락 (spinlock) 이 현재 홀딩되고 있다는 것을 검출하는 단계, 그 스핀락이 현재 홀딩되고 있다는 검출에 응답하여 트랜잭션을 무시하고 슬립 세트 트랜잭션을 전송하지 않는 단계, 및 각각의 코어의 SPM 하드웨어 블록이 RPM과의 핸드셰이크를 수행함에도 불구하고, RPM 프로세서에 대해 상기 슬립 세트를 적용하지 않는 단계를 더 포함할 수도 있다. 일 양태에서, 그 방법은, RPM 드라이버에서, 액티브 콘텍스트에서의 스핀락을 수신하는 단계, 멀티코어 컴퓨팅 디바이스의 제 1 코어에 대한 슬립 모드에 진입하는 단계, RPM 확인응답 인터럽트를 대기하는 동안에 멀티코어 컴퓨팅 디바이스의 제 0 코어에 대한 유휴 프로세스를 개시하는 단계, 제 0 코어에 대한 슬립 모드들을 구하고 (solving) 제 0 코어가 진입할 글로벌 저전력 모드를 선택하는 단계, 제 0 코어에 대한 글로벌 저전력 모드에 진입하는 단계, 제 0 코어에 대한 RPM 확인응답 인터럽트를 수신하는 단계, 및 스핀락을 해제하는 단계를 포함할 수도 있다. 일 양태에서, 그 방법은, 제 0 코어에 대한 RPM 메시지의 전송을 개시하는 단계, RPM 드라이버에서, 운영 체제 잠금을 수신하는 단계, 멀티코어 컴퓨팅 디바이스의 제 1 코어의 슬립 모드에 진입하고 RPM 트랜잭션을 만드는 단계, 제 1 코어에 대한 슬립 세트의 전송을 개시하는 단계, 제 1 코어로부터 RPM 드라이버에 대한 슬립 세트 트랜잭션 요청을 수신하고 운영 체제 잠금이 홀딩되는지를 결정하기 위해 검사하는 단계, 및 운영 체제 잠금이 홀딩된다고 결정되면 슬립 세트 트랜잭션 요청을 무시하고 인터럽트 신호를 제 1 코어로 전송하는 단계를 포함할 수도 있다.

[0005] 추가의 양태들은, 메모리, 및 메모리에 연결된 하나 이상의 프로세서들을 구비할 수도 있는 컴퓨팅 디바이스를 구비하며, 하나 이상의 프로세서들은 컴퓨팅 디바이스가 멀티코어 컴퓨팅 디바이스의 하나를 초과하는 코어들에 의해 공유되는 각각의 저전력 자원에 대한 참조 카운트를 메모리에 유지하는 단계, 메모리 로케이션에 연관된 저전력 자원의 각각의 코어의 사용량 및 각각의 코어의 전력 상태에 기초하여 메모리에 저장된 값을 독립적으로 조정하는 단계, 및 메모리 로케이션에서의 값을 이용하여 시스템 저전력 구성에 진입하는 때를 결정하는 단계를 포함하는 동작들을 수행하게 하는 프로세서 실행가능 명령들로 구성된다. 일 양태에서, 하나 이상의 프로세서들은, 플래그 비트 설정에 기초하여 저전력 모드에 배치될 수도 있는 자원들을 식별하는 단계, 레이턴시 요건을 코어 단위 기반 또는 글로벌 기반으로 등록하는 단계, 등록된 레이턴시 요건들로부터 가장 엄격한 레이턴시 요건을 선택하는 단계, 컴퓨팅 디바이스 상에서, 코어 상에 존재하는 레이턴시 제한들에 기초하여, 임의의 저전력 자원 모드, 또는 상기 선택된 가장 엄격한 레이턴시 허용오차를 초과하는 조합된 레이턴시 요건을 가지는 저전력 자원 모드들의 임의의 조합을 제거하기 위해 저전력 모드에 배치될 수도 있는 각각의 자원에 대한 저전력 모드들을 평가하는 단계, 잠재적 전력 절약량을 극대화하고 현재 코어에 대한 선택된 최악의 레이턴시 요건 이하의 총 레이턴시 요건을 갖는 저전력 자원 모드들의 조합을 선택하는 단계, 및 식별된 자원들의 각각에 대한 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 선택된 조합에 진입하는 단계를 더 포함할 수도 있는 동작들을 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성될 수도 있다. 일 양태에서, 하나 이상의 프로세서들은, 식별된 자원들의 각각에 대한 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 선택된 조합에 진입하는 단계가, 공유된 모드들에 대해, 참조 카운트 값이 0과 동일한 경우에 진입 함수를 실행하는 단계를 포함하도록 하는 동작들을 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성될 수도 있다. 일 양태에서, 하나 이상의 프로세서들은, 현재 코어에 대한 예상된 유휴 시간에 대해, 현재 온도에서의 단위 시간당 잠재적 전력 절약량 곱하기 예상된 유휴 시간에 기초하여, 각각의 평가된 저전력 자원 모드의 잠재적 전력 절약량을 결정하는 단계를 더 포함할 수도 있는 동작들을 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성될 수도 있다. 일 양태에서, 하나 이상의 프로세서들은, 멀티코어 컴퓨팅 디바이스의 각각의 코어가 멀티코어 컴퓨팅 디바이스에서의 하나 이상의 다른 코어들의 운영 체제와는 상이한 운영 체제 하에서 동작하도록 하는 동작들을 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성될 수도 있다. 일 양태에서, 하나 이상의 프로세서들은, 멀티코어 컴퓨팅 디바이스의 적어도 하나의 코어가 그 코어를 적어도 하나의 다른 코어의 동작들에 바인딩하는 적어도 하나의 자원을 공유하고, 각각의 코어는 다른 코어들의 각각과는 독립적으로 저전력 모드들에 진입하고 종료하도록 구성되도록 하는 동작들을 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성될 수도 있다. 일 양태에서, 하나 이상의 프로세서들은, 멀티코어 컴퓨팅 디바이스가 둘 이상의 실행 환경들에 의해 공유된 자원들을 관리하는 자원 전력 관리자 (RPM) 프로세서를 포함할 수도 있도록 하는 동작들을 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성될 수도 있다. 일 양태에서, 하나 이상의 프로세서들은, 스레드들을 실행하는 코어들 및 유휴 스레드들을 실행하는 코

어들 간에 공유된 변수에 액세스하는 것에 의해 어떤 코어들이 슬립 상태에 있는지를 결정하는 단계로서, 공유된 변수는 유휴 스레드의 진입 함수의 수행시 각각의 코어에 의해 점증되고 유휴 스레드의 종료 함수의 수행시 각각의 코어에 의해 점감되는, 결정하는 단계를 더 포함할 수도 있는 동작들을 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령로 구성될 수도 있다. 일 양태에서, 하나 이상의 프로세서들은, 다른 코어가 RPM 트랜잭션을 전송하는 동안에 하나의 코어가 자원 전력 관리자 (RPM) 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 단계를 더 포함할 수도 있는 동작들을 수행하기 위한 프로세서 실행가능 명령들로 프로세서가 구성되도록 하는 동작들을 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성될 수도 있다. 일 양태에서, 하나 이상의 프로세서들은, 다른 코어가 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 단계는, RPM 시작 동작 동안에 그리고 RPM 정지 동작 동안에 스핀락을 홀딩하는 단계를 포함할 수도 있도록 하는 동작들을 상기 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령로 구성될 수도 있다. 일 양태에서, 하나 이상의 프로세서들은 멀티코어 컴퓨팅 디바이스에서의 각각의 코어가 슬립 세트의 서로소 부분들에 기록하도록 하는 동작들을 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성될 수도 있고, 다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 단계는 자원들을 슬립 세트에 추가하는 경우에 스핀락을 해제시키는 단계를 포함할 수도 있다. 일 양태에서, 하나 이상의 프로세서들은, 자원을 자동 무효라고 마킹하는 단계, 및 RPM 프로세서에서, 마킹된 자원에 연관된 슬립 세트를 무효화하는 단계를 더 포함할 수도 있는 동작들을 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성될 수도 있다. 일 양태에서, 하나 이상의 프로세서들은, 이전의 트랜잭션으로부터의 자원들이 그것들의 슬립 세트에 바람직하지 않게 진입하지 않도록 이전의 트랜잭션을 무효화하는 단계를 더 포함할 수도 있는 동작들을 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성될 수도 있다. 일 양태에서, 하나 이상의 프로세서들은, 멀티코어 컴퓨팅 디바이스에서의 각각의 코어는, 그 코어가 그것의 저전력 모드에 진입하는 경우에 당해 코어의 전력 상태 및 RPM 프로세서와의 핸드셰이크들을 제어하는 서브시스템 전력 관리 (SPM) 하드웨어 블록을 구비하도록 하는 동작들을 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성될 수도 있고, 그 방법은, RPM 슬립 드라이버에서, 스핀락 (spinlock) 이 현재 홀딩되고 있다는 것을 검출하는 단계, 그 스핀락이 현재 홀딩되고 있다는 검출에 응답하여 트랜잭션을 무시하고 슬립 세트 트랜잭션을 전송하지 않는 단계, 및 각각의 코어의 SPM 하드웨어 블록이 RPM과의 핸드셰이크를 수행함에도 불구하고, RPM 프로세서에 대해 상기 슬립 세트를 적용하지 않는 단계를 더 포함할 수도 있다. 일 양태에서, 하나 이상의 프로세서들은, RPM 드라이버에서, 액티브 콘텍스트에서의 스핀락을 수신하는 단계, 멀티코어 컴퓨팅 디바이스의 제 1 코어의 슬립 모드에 진입하는 단계, RPM 확인응답 인터럽트를 대기하는 동안에 멀티코어 컴퓨팅 디바이스의 제 0 코어에 대한 유휴 프로세스를 개시하는 단계, 상기 제 0 코어에 대한 슬립 모드들을 구하고 상기 제 0 코어가 진입할 글로벌 저전력 모드를 선택하는 단계, 제 0 코어에 대한 글로벌 저전력 모드에 진입하는 단계, 제 0 코어에 대한 RPM 확인응답 인터럽트를 수신하는 단계, 및 스핀락을 해제하는 단계를 더 포함할 수도 있는 동작들을 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성될 수도 있다. 일 양태에서, 하나 이상의 프로세서들은, 제 0 코어에 대한 RPM 메시지의 전송을 개시하는 단계, RPM 드라이버에서, 운영 체제 잠금을 수신하는 단계, 멀티코어 컴퓨팅 디바이스의 제 1 코어에 대한 슬립 모드에 진입하고, RPM 트랜잭션을 만드는 단계, 제 1 코어에 대한 슬립 세트의 전송을 개시하는 단계, 제 1 코어로부터 RPM 드라이버에 대한 슬립 세트 트랜잭션 요청을 수신하고 운영 체제 잠금이 홀딩되는지를 결정하기 위해 검사하는 단계, 및 운영 체제 잠금이 홀딩된다고 결정되면 슬립 세트 트랜잭션 요청을 무시하고 인터럽트 신호를 제 1 코어로 전송하는 단계를 더 포함할 수도 있는 동작들을 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성될 수도 있다.

[0006] 추가의 양태들은, 프로세서 실행가능 명령들이 저장된 비일시적 프로세서 실행가능 저장 매체를 포함하며, 그 프로세서 실행가능 명령들은 서버로 하여금, 멀티코어 컴퓨팅 디바이스의 하나를 초과하는 코어들에 의해 공유되는 각각의 저전력 자원에 대한 참조 카운트를 메모리에 유지하는 단계, 메모리 로케이션에 연관된 저전력 자원의 각각의 코어의 사용량 및 각각의 코어의 전력 상태에 기초하여 메모리에 저장된 값을 독립적으로 조정하는 단계, 및 메모리 로케이션에서의 값을 이용하여 시스템 저전력 구성에 진입하는 때를 결정하는 단계를 포함할 수도 있는 동작들을 수행하게 구성된다. 일 양태에서, 하나 이상의 프로세서들은, 플래그 비트 설정에 기초하여 저전력 모드에 배치될 수도 있는 자원들을 식별하는 단계, 레이턴시 요건을 코어 단위 기반 또는 글로벌 기반으로 등록하는 단계, 등록된 레이턴시 요건들로부터 가장 엄격한 레이턴시 요건을 선택하는 단계, 컴퓨팅 디바이스 상에서, 코어 상에 존재하는 레이턴시 제한들에 기초하여, 임의의 저전력 자원 모드, 또는 선택된 가장 엄격한 레이턴시 허용오차를 초과하는 조합된 레이턴시 요건을 가지는 저전력 자원 모드들의 임의의 조합을 제거하기 위해 저전력 모드에 배치될 수도 있는 각각의 자원에 대한 저전력 모드들을 평가하는 단계, 잠재적 전

력 절약량을 극대화하고 현재 코어에 대한 선택된 최악의 레이턴시 요건 이하의 총 레이턴시 요건을 갖는 저전력 자원 모드들의 조합을 선택하는 단계, 및 식별된 자원들의 각각에 대한 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 선택된 조합에 진입하는 단계를 더 포함할 수도 있는 동작들을 컴퓨팅 디바이스가 수행하게 하는 프로세서 실행가능 명령들로 구성될 수도 있다.

일 양태에서, 저장된 프로세서 실행가능 소프트웨어 명령들은, 식별된 자원들의 각각에 대한 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 선택된 조합에 진입하는 단계는, 공유된 모드들에 대해, 참조 카운트 값이 0과 동일한 경우에 진입 함수를 실행하는 단계를 포함할 수도 있도록 하는 동작들을 프로세서가 수행하게 하도록 구성될 수도 있다.

일 양태에서, 저장된 프로세서 실행가능 소프트웨어 명령들은, 현재 코어에 대한 예상된 유휴 시간에 대해, 현재 온도에서의 단위 시간당 잠재적 전력 절약량 곱하기 예상된 유휴 시간에 기초하여, 각각의 평가된 저전력 자원 모드의 잠재적 전력 절약량을 결정하는 단계를 더 포함할 수도 있는 동작들을 프로세서가 수행하게 하도록 구성될 수도 있다.

일 양태에서, 저장된 프로세서 실행가능 소프트웨어 명령들은, 멀티코어 컴퓨팅 디바이스의 각각의 코어가 멀티코어 컴퓨팅 디바이스에서의 하나 이상의 다른 코어들의 운영 체제와는 상이한 운영 체제 하에서 동작하도록 하는 동작들을 프로세서가 수행하게 하도록 구성될 수도 있다.

일 양태에서, 저장된 프로세서 실행가능 소프트웨어 명령들은, 멀티코어 컴퓨팅 디바이스의 적어도 하나의 코어는 그 코어를 적어도 하나의 다른 코어의 동작들에 바인딩하는 적어도 하나의 자원을 공유하고, 각각의 코어는 다른 코어들의 각각과는 독립적으로 저전력 모드들에 진입하고 종료하도록 하는 동작들을 프로세서가 수행하게 하도록 구성될 수도 있다.

일 양태에서, 멀티코어 컴퓨팅 디바이스는 둘 이상의 실행 환경들에 의해 공유된 자원들을 관리하는 RPM 프로세서를 구비할 수도 있다.

일 양태에서, 저장된 프로세서 실행가능 소프트웨어 명령들은, 스레드들을 실행하는 코어들 및 유휴 스레드들을 실행하는 코어들 간에 공유된 변수에 액세스하는 것에 의해 어떤 코어들이 슬립 상태에 있는지를 결정하는 단계로서, 공유된 변수는 유휴 스레드의 진입 함수의 수행시 각각의 코어에 의해 점증되고 유휴 스레드의 종료 함수의 수행시 각각의 코어에 의해 점감되는, 결정하는 단계를 더 포함할 수도 있는 동작들을 프로세서가 수행하게 하도록 구성될 수도 있다.

일 양태에서, 저장된 프로세서 실행가능 소프트웨어 명령들은, 다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 단계를 더 포함할 수도 있는 동작들을 프로세서가 수행하게 하도록 구성될 수도 있다.

일 양태에서, 저장된 프로세서 실행가능 소프트웨어 명령들은, 다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 단계가, RPM 시작 동작 동안에 그리고 RPM 정지 동작 동안에 상기 스핀락을 홀딩하는 단계를 포함할 수도 있도록 하는 동작들을 프로세서가 수행하게 하도록 구성될 수도 있다.

일 양태에서, 저장된 프로세서 실행가능 소프트웨어 명령들은, 멀티코어 컴퓨팅 디바이스에서의 각각의 코어가 슬립 세트의 서로소 부분들에 기록하도록 하는 동작들을 프로세서가 수행하게 하도록 구성될 수도 있고, 다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 단계는 자원들을 슬립 세트에 추가하는 경우에 스핀락을 해제시키는 단계를 포함할 수도 있다.

일 양태에서, 저장된 프로세서 실행가능 소프트웨어 명령들은, 자원을 자동 무효라고 마킹하는 단계, 및 RPM 프로세서에서, 마킹된 자원에 연관된 슬립 세트를 무효화하는 단계를 추가로 포함할 수도 있는 동작들을 프로세서가 수행하게 하도록 구성될 수도 있다.

일 양태에서, 저장된 프로세서 실행가능 소프트웨어 명령들은, 이전의 트랜잭션으로부터의 자원들이 그것들의 슬립 세트에 바람직하지 않게 진입하지 않도록 이전의 트랜잭션을 무효화하는 단계를 더 포함할 수도 있는 동작들을 프로세서가 수행하게 하도록 구성될 수도 있다.

일 양태에서, 저장된 프로세서 실행가능 소프트웨어 명령들은, 멀티코어 컴퓨팅 디바이스에서의 각각의 코어가 그것의 저전력 모드에 진입하는 경우에 당해 코어의 전력 상태 및 RPM 프로세서와의 핸드셰이크들을 제어하는 서브시스템 전력 관리 (SPM) 하드웨어 블록을 그 코어가 가질 수도 있게 하는 동작들을 수행하게 하도록 구성되며, 그 방법은, RPM 슬립 드라이버에서, 스핀락 (spinlock) 이 현재 홀딩되고 있다는 것을 검출하는 단계, 그 스핀락이 현재 홀딩되고 있다는 검출에 응답하여 트랜잭션을 무시하고 슬립 세트 트랜잭션을 전송하지 않는 단계, 및 각각의 코어의 SPM 하드웨어 블록이 RPM과의 핸드셰이크를 수행함에도 불구하고, RPM 프로세서에 대해 상기 슬립 세트를 적용하지 않는 단계를 더 포함할 수도 있다.

일 양태에서, 저장된 프로세서 실행가능 소프트웨어 명령들은, RPM 드라이버에서, 액티브 콘텍스트에서의 스핀락을 수신하는 단계, 멀티코어 컴퓨팅 디바이스의 제 1 코어의 슬립 모드에 진입하는 단계, RPM 확인응답 인터럽트를 대기하는 동안에 멀티코어 컴퓨팅 디바이스의 제 0 코어에 대한 유휴 프로세스를 개시하는 단계, 제 0 코어에 대한 슬립 모드들을 구하고 제 0 코어가 진입할 글로벌 저전력 모드를 선택하는 단계, 제 0 코어에 대한 글로벌 저전력 모드에 진입하는 단계, 제 0 코어에 대한 RPM 확인응답 인터럽트를 수신하는 단계, 및 스핀락을 해제하는 단계를 더 포함할 수도 있는 동작들을 프로세서가 수행하게 하도록 구성될 수도 있다.

일 양태에서, 저장된 프로세서 실행가능 소프트웨어 명령들은, 제 0 코어에 대한 RPM 메시지의 전송을 개시하는 단계, RPM 드라이버에서, 운영 체제 잠금을 수신하

는 단계, 멀티코어 컴퓨팅 디바이스의 제 1 코어의 슬립 모드에 진입하는 단계, RPM 트랜잭션을 만드는 단계, 제 1 코어에 대한 슬립 세트의 전송을 개시하는 단계, 제 1 코어로부터 RPM 드라이버에 대한 슬립 세트 트랜잭션 요청을 수신하고 운영 체제 잠금이 홀딩되는지를 결정하기 위해 검사하는 단계, 및 운영 체제 잠금이 홀딩된다고 결정되면 슬립 세트 트랜잭션 요청을 무시하고 인터럽트 신호를 제 1 코어로 전송하는 단계를 더 포함할 수도 있는 동작들을 프로세서가 수행하게 하도록 구성될 수도 있다.

[0007] 추가의 양태들은, 멀티코어 컴퓨팅 디바이스의 하나를 초과하는 코어들에 의해 공유되는 각각의 저전력 자원에 대한 참조 카운트를 메모리에 유지하는 수단, 메모리 로케이션에 연관된 저전력 자원의 각각의 코어의 사용량 및 각각의 코어의 전력 상태에 기초하여 메모리에 저장된 값을 독립적으로 조정하는 수단, 및 메모리 로케이션에서의 값을 이용하여 시스템 저전력 구성에 진입하는 때를 결정하는 수단을 갖는 컴퓨팅 디바이스를 구비한다.

일 양태에서, 그 컴퓨팅 디바이스는, 플래그 비트 설정에 기초하여 저전력 모드에 배치될 수도 있는 자원들을 식별하는 수단, 레이턴시 요건을 코어 단위 기반 또는 글로벌 기반으로 등록하는 수단, 등록된 레이턴시 요건들로부터 가장 엄격한 레이턴시 요건을 선택하는 수단, 컴퓨팅 디바이스 상에서, 코어 상에 존재하는 레이턴시 제한들에 기초하여, 임의의 저전력 자원 모드, 또는 선택된 가장 엄격한 레이턴시 허용오차를 초과하는 조합된 레이턴시 요건을 가지는 저전력 자원 모드들의 임의의 조합을 제거하기 위해 저전력 모드에 배치될 수도 있는 각각의 자원에 대한 저전력 모드들을 평가하는 수단, 잠재적 전력 절약량을 극대화하고 현재 코어에 대한 선택된 최악의 레이턴시 요건 이하의 총 레이턴시 요건을 갖는 저전력 자원 모드들의 조합을 선택하는 수단, 및 식별된 자원들의 각각에 대한 상기 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 상기 선택된 조합에 진입하는 수단을 더 포함할 수도 있다. 일 양태에서, 그 컴퓨팅 디바이스는, 식별된 자원들의 각각에 대한 선택된 저전력 모드들의 각각의 저전력 모드의 진입 함수를 실행하는 것에 의해 저전력 자원 모드들의 선택된 조합에 진입하는 수단을 포함할 수도 있으며, 그 수단은 공유된 모드들에 대해, 참조 카운트 값이 0과 동일한 경우에 진입 함수를 실행하는 수단을 포함할 수도 있다. 일 양태에서, 그 컴퓨팅 디바이스는, 현재 코어에 대한 예상된 유휴 시간에 대해, 현재 온도에서의 단위 시간당 잠재적 전력 절약량 곱하기 예상된 유휴 시간에 기초하여, 각각의 평가된 저전력 자원 모드의 잠재적 전력 절약량을 결정하는 수단을 더 포함할 수도 있다. 일 양태에서, 그 컴퓨팅 디바이스는, 멀티코어 컴퓨팅 디바이스에서의 하나 이상의 다른 코어들의 운영 체제와는 상이한 운영 체제 하에서 멀티코어 컴퓨팅 디바이스의 각각의 코어를 동작시키는 수단을 구비할 수도 있다. 일 양태에서, 그 컴퓨팅 디바이스는, 멀티코어 컴퓨팅 디바이스의 적어도 하나의 코어는 상기 코어를 적어도 하나의 다른 코어의 동작들에 바인딩하는 적어도 하나의 자원을 공유하도록, 그리고 각각의 코어는 다른 코어들의 각각과는 독립적으로 저전력 모드들에 진입하고 종료하게 구성되도록 컴퓨팅 디바이스를 구성하는 수단을 더 포함할 수도 있다. 일 양태에서, 그 컴퓨팅 디바이스는, 멀티코어 컴퓨팅 디바이스에서 둘 이상의 실행 환경들에 의해 공유된 자원들을 관리하는 RPM 프로세서를 구비하는 수단을 포함할 수도 있다. 일 양태에서 그 컴퓨팅 디바이스는, 스레드들을 실행하는 코어들 및 유휴 스레드들을 실행하는 코어들 간에 공유된 변수에 액세스하는 것에 의해 어떤 코어들이 슬립 상태에 있는지를 결정하는 수단을 포함할 수도 있으며, 공유된 변수는 유휴 스레드의 진입 함수의 수행시 각각의 코어에 의해 점증되고 유휴 스레드의 종료 함수의 수행시 각각의 코어에 의해 점감된다. 일 양태에서 그 컴퓨팅 디바이스는, 다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 수단을 구비할 수도 있다. 일 양태에서, 다른 코어가 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 수단은, RPM 시작 동작 동안에 그리고 RPM 정지 동작 동안에 상기 스핀락을 홀딩하는 수단을 구비할 수도 있다. 일 양태에서, 멀티코어 컴퓨팅 디바이스에서의 각각의 코어는 슬립 세트의 서로소 부분들에 기록하고, 다른 코어가 다른 RPM 트랜잭션을 전송하는 동안 하나의 코어가 RPM 트랜잭션을 시작하는 것을 막기 위해 스핀락을 이용하는 수단은 자원들을 슬립 세트에 추가하는 경우에 스핀락을 해제시키는 수단을 구비할 수도 있다. 일 양태에서, 그 컴퓨팅 디바이스는, 자원을 자동 무효라고 마킹하는 수단, 및 RPM 프로세서에서, 마킹된 자원에 연관된 슬립 세트를 무효화하는 수단을 구비할 수도 있다. 일 양태에서, 그 컴퓨팅 디바이스는, 이전의 트랜잭션으로부터의 자원들이 그것들의 슬립 세트에 바람직하지 않게 진입하지 않도록 이전의 트랜잭션을 무효화하는 수단을 포함할 수도 있다. 일 양태에서, 멀티코어 컴퓨팅 디바이스에서의 각각의 코어는, 그 코어가 그것의 저전력 모드에 진입하는 경우에 당해 코어의 전력 상태 및 RPM 프로세서와의 핸드셰이크들을 제어하는 서브시스템 전력 관리 (SPM) 하드웨어 블록을 가질 수도 있고, 그 디바이스는, RPM 슬립 드라이버에서, 스핀락이 현재 홀딩되고 있다는 것을 검출하는 수단, 그 스핀락이 현재 홀딩되고 있다는 검출에 응답하여 트랜잭션을 무시하고 슬립 세트 트랜잭션을 전송하지 않는 수단, 및 각각의 코어의 SPM 하드웨어 블록이 RPM과의 핸드셰이크를 수행함에도 불구하고, RPM 프로세서에 대해 상기 슬립 세트를 적용하지 않는 수단을 더 포함할 수도 있다. 일 양태에서, 그 컴퓨팅 디바이스는, RPM 드

라이버에서, 액티브 콘텍스트에서의 스핀락을 수신하는 수단, 멀티코어 컴퓨팅 디바이스의 제 1 코어의 슬립 모드에 진입하는 수단, RPM 확인응답 인터럽트를 대기하는 동안에 멀티코어 컴퓨팅 디바이스의 제 0 코어에 대한 유휴 프로세스를 개시하는 수단, 제 0 코어에 대한 슬립 모드들을 구하고 제 0 코어가 진입할 글로벌 저전력 모드를 선택하는 수단, 제 0 코어에 대한 글로벌 저전력 모드에 진입하는 수단, 제 0 코어에 대한 RPM 확인응답 인터럽트를 수신하는 수단, 및 스핀락을 해제시키는 수단을 구비할 수도 있다. 일 양태에서, 그 컴퓨팅 디바이스는, 제 0 코어에 대한 RPM 메시지의 전송을 개시하는 수단, RPM 드라이버에서, 운영 체제 잠금을 수신하는 수단, 멀티코어 컴퓨팅 디바이스의 제 1 코어에 대한 슬립 모드에 진입하고 RPM 트랜잭션을 만드는 수단, 제 1 코어에 대한 슬립 세트의 전송을 개시하는 수단, 제 1 코어로부터 RPM 드라이버에 대한 슬립 세트 트랜잭션 요청을 수신하고 운영 체제 잠금이 홀딩되는지를 결정하기 위해 검사하는 수단, 및 운영 체제 잠금이 홀딩된다고 결정되면 슬립 세트 트랜잭션 요청을 무시하고 인터럽트 신호를 상기 제 1 코어로 전송하는 수단을 구비할 수도 있다.

도면의 간단한 설명

[0008] 본원에 통합되고 본 명세서의 일부를 구성하는 첨부 도면들은, 본 발명의 예시적인 양태들을 도시하고, 위에서 주어진 전반적인 설명 및 아래에 주어지는 상세한 설명과 함께 본 발명의 특징들을 설명하도록 기능한다.

도 1은 일 양태에서 2 개의 저전력 모드들 중 하나에 진입하기 위해 프로그래밍 노드에 의해 제어되는 자원의 다이어그램이다.

도 2는 다양한 양태들을 구현하기에 적합한 일 예의 시스템 온 칩의 아키텍처 도면이다.

도 3은 다양한 양태들을 구현하기에 적합한 일 예의 멀티코어 프로세서의 아키텍처 도면이다.

도 4는 다양한 양태들을 구현하도록 구성된 멀티코어 프로세서에서 다양한 글로벌 및 로컬 컴포넌트들을 도시하는 아키텍처 도면이다.

도 5 내지 도 9는 멀티코어 슬립을 수행하고 저전력 모드들의 최적의 세트를 선택하는 양태의 방법들의 프로세스 흐름도들이다.

도 10은 일 양태에서 사용하기에 적합한 모바일 디바이스의 컴포넌트 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0009] 다양한 양태들이 첨부 도면들을 참조하여 상세히 설명될 것이다. 가능한 곳이라면 어디서든, 동일한 참조 번호들이 도면들 전체를 통해서 동일하거나 유사한 부분들을 참조하는 데 사용될 것이다. 특정한 예들 및 구현예들에 대한 언급들은 예시의 목적을 위한 것이고, 발명 또는 청구항들의 범위를 제한하려는 의도는 아니다.

[0010] 단어 "예시적"은 본원에서는 "예, 사례, 또는 예시로서 역할을 한다"는 의미로 사용된다. "예시적인" 것으로서 본 명세서에서 설명된 어떤 구현예라도 다른 구현예들보다 바람직하거나 유익하다고 생각할 필요는 없다.

[0011] 용어들 "모바일 디바이스" 및 "컴퓨팅 디바이스"는 셀룰러 전화기들, 랩탑 컴퓨터들, 개인휴대 정보단말들(PDAs), 팜-톱 컴퓨터들, 무선 전자 메일 수신기들 (예컨대, Blackberry® 및 Treo® 디바이스들), 멀티미디어 인터넷 가능 셀룰러 전화기들 (예컨대, Blackberry Storm®), 위성 위치확인 시스템 (GPS) 수신기들, 무선 게이밍 제어기들, 및 프로그래밍가능 프로세서를 구비하고 전력 보존 방법들이 유익하게 되는 배터리 전력 하에서 동작하는 유사한 개인용 전자 디바이스들 중 임의의 하나 또는 모두를 지칭하기 위해 본원에서 상호교환적으로 사용된다.

[0012] 용어 "시스템 온 칩" (system on chip; SOC) 은 단일 기판 상에 통합된 다수의 자원들 및 프로세서들을 포함하는 단일 집적회로 (IC) 칩을 지칭하기 위해 본원에서 사용된다. 단일 SOC는 디지털, 아날로그, 혼합된 신호, 및 라디오-주파수 기능들을 위한 회로를 포함할 수도 있다. 단일 SOC는 또한 임의의 수의 범용 및/또는 특화된 프로세서들 (DSP, 모뎀 프로세서들, 비디오 프로세서들 등), 메모리 블록들 (예컨대, ROM, RAM, 플래시 등), 및 자원들 (예컨대, 타이머들, 전압 조정기들, 발진기들 등) 을 포함할 수도 있다. SOC들은 또한 통합된 자원들 및 프로세서들을 제어하기 위한, 뿐만 아니라 주변 디바이스들을 제어하기 위한 소프트웨어를 포함할 수도 있다.

[0013] 용어 "멀티코어 프로세서"는 프로그램 명령들을 읽고 실행하도록 구성된 둘 이상의 독립 프로세싱 코어들 (예컨

대, CPU 코어들) 을 포함하는 단일 집적회로 (IC) 칩 또는 칩 패키지를 지칭하기 위해 본원에서 사용된다. SOC는 다수의 멀티코어 프로세서들을 포함할 수도 있고, SOC에서의 각각의 프로세서는 코어라고 지칭될 수도 있다.

[0014] 용어 "자원"은 매우 다양한 회로들 (예컨대, 포트들, 클록들, 버스들, 발진기들 등), 컴포넌트들 (예컨대, 메모리), 신호들 (예컨대, 클록 신호들), 및 컴퓨팅 디바이스 상에서 실행하고 있는 프로세서들 및 클라이언트들을 지원하는데 이용되는 전압들 (예컨대, 전압 레일들) 중 임의의 것을 지칭하기 위해 본원에서 사용된다. 모바일 컴퓨팅 디바이스는 통상 다수의 자원들, 이를테면 수정 발진기, 전압 레일들, 하나 이상의 메모리 유닛들, 통신 버스들 등을 포함할 수도 있다.

[0015] 멀티코어 프로세서 시스템들 및 SOC들에서의 자원들은 코어들의 특정 코어에 국소적이거나, 서브세트 간에 공유되거나, 또는 모든 코어들 간에 공유될 수도 있다. 본원에서 사용되는 바와 같이, 용어 "글로벌 자원"은 디바이스, 칩, 프로세서, 그룹 등의 모든 코어들 간에 공유되는 자원을 지칭한다.

[0016] 명료함을 위해, 다양한 양태의 방법들이 노드 전력 아키텍처 (node power architecture; NPA) 및 관련된 기술용어를 이용하여 설명된다. 그러나, 본원에서의 NPA 노드들에 관련된 예들과 NPA 노드 또는 NPA 방법들에 대한 다른 언급들은 예시 목적만을 위한 것이라는 것이 이해되어야 한다. 그러므로 청구항들의 범위는 청구항들에서 구체적으로 그렇게 언급되지 않는 한 NPA 노드 또는 NPA 프로세스를 요구하는 것으로서 해석되지 않아야 한다.

[0017] 배터리 수명을 최대화하기 위해, 모바일 디바이스들은, 가능할 때마다, 이를테면 프로세서가 유휴 상태로 있는 경우에, 하나 이상의 디바이스 자원들을 저전력 상태로 배치하도록 구성될 수도 있다. 디바이스 자원을 저전력 상태로 배치하는 것은 통상, 프로세서가 태스크들을 능동적으로 프로세싱하지 않을 때마다 자원의 동작들을 줄이거나 또는 전원을 끊는 것으로 구성된다. 특정한 자원들은 적절한 동작을 위해 필수적이고 및/또는 다른 컴포넌트들에 의해 사용되고, 프로세서가 유휴인 경우에 턴 오프될 수 없거나 또는 저전력 상태로 배치될 수 없다. 명료함을 위해, 프로세서가 태스크들을 프로세싱하고 있지 않고 및/또는 유휴 상태로 있는 경우에 턴 오프 될 수 있거나 또는 하나 이상의 저전력 상태들로 배치될 수 있는 자원들은 본원에서 저전력 자원 (low power resource; LPR) 들이라고 지칭된다. 다수의 LPR들은 주어진 유휴 상태에서 활성화, 인에이블, 비활성화, 및/또는 디스에이블될 수 있다. 각각의 LPR은 다수의 저전력 모드들 (유휴, 액티브-슬립, 슬립, 깊은-슬립 등) 을 가질 수도 있는데, 그것들은 본원에서 저전력 자원 모드 (low power resource mode; LPRM) 들이라고 지칭된다. 각각의 자원은 주어진 상태에 대한 LPRM들의 최적의 세트를 선택함에 있어서 프로세서에 의해 사용되는 정보를 포함하는 LPRM 특성 데이터를 정의할 수도 있다. 이 정보는 자원에 대해 이용가능한 각각의 LPRM, 시간의 함수로서 또는 단위 시간당 각각의 LPRM에 대한 잠재적 전력 절약량, 각각의 LPRM에 대한 레이턴시 특성들 (즉, 저전력 모드 종료 시간), 잠재적 전력 절약량에 대한 온도 효과, 의존도들 (즉, 클라이언트들 이외의 다른 자원들과의 상호의존도들), 및 각각의 자원에 대한 최적의 LPRM을 선택하는 것에 관계가 있을 수도 있는 다른 정보의 리스팅을 포함할 수도 있다.

[0018] 전체 "시스템 저전력 구성"은 현재의 운영 상태 및 조건들에 기초하여 저전력 모드에 진입하기 위해 이용가능한 자원들의 각각에 대해 선택된 저전력 모드들의 세트이다.

[0019] 저전력 모드들을 동적으로 컴퓨팅하고 선택하는 방법들 및 시스템들은 2010년 12월 10일자로 출원된 발명의 명칭이 "Dynamic Low Power Mode Implementation For Computing Devices"인 미국 특허 출원 제12/965,008호에 개시되어 있으며, 그 전체 내용은 참조로 본원에 통합된다. 요약하면, 디스에이블될 수 있는 자원들 뿐만 아니라 디바이스 저전력 구성 동안에 남아 있어야 하는 자원들을 식별하는 것에 의해 디바이스가 하나 이상의 저전력 자원 모드들 (LPRMs) 을 선택할 수도 있다. 디바이스 프로세서는, 그 프로세서가 유휴 상태에 진입하는 시간에 자원의 저전력 자원 모드들 중 어떤 것들이 유효한지를 결정하며, 현재 디바이스 조건들이 주어지면 예상된 전력 절약량에 의해 유효 저전력 자원 모드들을 랭킹화하며, 유효 저전력 자원 모드들 중 어떤 것들이 레이턴시 요건들에 부합하면서 가장 큰 전력 절약량을 제공하는지를 결정하고, 각각의 자원이 진입할 특정 저전력 자원 모드를 선택하는 것에 의해, (저전력 자원 모드들의 세트로 이루어지는) 최적의 시스템 저전력 구성을 컴퓨팅할 수도 있다.

[0020] 자원들은 저전력 상태에 진입하기 위해 그것들이 이용가능하게 되는 때를 저전력 모드 마스크 ("LPM 마스크") 내의 플래그 비트를 인에이블하는 것에 의해 나타낼 수도 있다. 프로세서가 시스템 저전력 모드에 진입할 준비가 된 경우, 저전력 태스크는 저전력 상태에 배치될 수 있는 자원들을 식별하기 위해 LPM 마스크에 액세스하고, 주어진 다이나믹 시스템 상태 (예컨대, 현재의 액티브 클라이언트들, 요구된 레이턴시, 예상된 유휴

시간, 및 온도) 에 기초하여 그들 자원들에 대해 진입할 적절한 LPRM들을 결정할 수도 있다.

[0021] 일 양태에서, 프로세서가 유휴 상태에 진입할 수 있는 경우, 저전력 태스크는 다양한 자원들에 대해 어떤 저전력 모드들이 진입되어야 하는지를 결정하기 위해 "솔버 (solver)" 프로세스를 실행할 수도 있다. 이러한 경우들에서, 상이한 자원들에 대한 저전력 모드들 및 그들 저전력 모드들의 특성들은 유휴 상태에 진입하는 시간일 때에 평가될 슬립 태스크에 의한 사용을 위해 수집될 것을 필요로 할 수도 있다. 이는 노드 전력 아키텍처 (NPA) 에서 "/sleep/lpr" NPA 노드를 통해 구현될 수도 있다.

[0022] "/sleep/lpr"에 대한 요청들은 저전력 자원 모드들을 인에이블시키는 비트마스크들의 형태로 이루어질 수도 있다. 개발자들은 "/sleep/lpr" NPA 노드에 저전력 자원 모드 (및 그것의 자원 저전력 모드 비트마스크들) 를 등록하는 sleep_define_lpr() 함수를 통해 저전력 자원 모드를 등록시킬 수도 있다. "/sleep/lpr" NPA 노드는 인에이블/디스에이블 시키는 것에 관심있는 저전력 자원 모드들을 나타내는 비트마스크에 대해 임의의 시간에 쿼리될 수도 있다. NPA 자원들은 또한 올바른 비트마스크로 "/sleep/lpr"에 요청하는 것에 의해 그것들의저전력 자원 모드들이 유휴 시간에 인에이블되어야 한다고 요청할 수 있다. 슬립 솔버는 그 다음에 유휴 시간에 저전력 자원 모드들의 목록 및 어떤 것들이 현재 인에이블되어 있는지의 마스크에 대해 "/sleep/lpr"을 쿼리할 수 있다.

[0023] 위에서 논의된 바와 같이, 프로세서가 유휴 상태에 진입할 수 있는 경우, 저전력 태스크는 다양한 자원들에 대해 어떤 저전력 자원 모드들이 진입되어야 하는지를 결정하기 위해 "솔버" 프로세스를 실행할 수도 있다. 이것의 일 예는 노드 전력 아키텍처 (NPA) (2) 내의 프로세스 (즉, 노드) 가, 자원 (수정 발진기 (4)) 에 대해, 이용가능한 저전력 모드들 (6, 8) 중 어떤 것에 진입될 수 있는지를 결정하는 방법을 예시하는 도 1에 도시되어 있다. 도 1에 도시된 예에서, 수정 발진기 자원 (4) 은 2 개의 대안적 저전력 자원 모드들 (LPRMs), 즉 게이트식 동작 상태 (6) 및 완전 셧다운 (8) 을 가진다. 선택된 LPRM들에는 자원을 전력 절약 상태로 배치하는 "진입 (enter)" 함수를 호출하는 것에 의해 진입될 수도 있다. 프로세서가 유휴인 동안, 프로세서는 인터럽트 대기 (wait for interrupt; WFI) 프로세스 및/또는 유휴 프로세스를 수행할 수도 있다. 그 프로세서 및 선택된 자원들은 기상 이벤트가 발생하기까지 이 상태로 남아 있을 수도 있다. 기상 이벤트가 발생하는 경우, "종료 (exit)" 함수가 각각의 선택된 자원에 대해 그 자원을 소망의 동작 상태 (예컨대, 정상 또는 최대 (full) 전력 상태) 로 반환하기 위해 호출될 수도 있다.

[0024] 저전력 모드들을 동적으로 컴퓨팅하고 선택하는 위에서 언급된 프로세스가 전통적인 컴퓨팅 디바이스들에 효과적인 반면, 현대의 모바일 디바이스들은 더욱더 복잡해지고 디바이스 프로세서들에 의해 사용되거나 또는 관리되는 자원들은 매년 증가하고 있다. 예를 들어, 많은 모바일 컴퓨팅 디바이스들은 이제 다수의 수정 발진기들, 다수의 전압 레일들, 및 다수의 메모리 로케이션들을 구비하며, 그것들의 각각은 독립적으로 제어될 수 있다. 결과적으로, 모바일 디바이스 프로세서가 전력을 절약하기 위하여 턴 오프하거나 또는 저전력 모드에 배치할 수도 있는 많은 상이한 자원들이 있다. 더구나, 현대의 모바일 디바이스들은, 다양한 디바이스 자원들을 이용하고 상이한 태스크들을 수행하는 다수의 프로세서들/코어를 가질 수도 있 (고 이에 따라 동시에 유휴가 되지 않을 수도 있) 다. 다수의 프로세서들/코어들의 존재는 복잡도의 부가적인 레벨을 저전력 모드들의 관리에 추가하고, 실행 제어의 정도는 저전력 모드에 배치될 자원들을 선택하는 경우에 연습되어야만 한다.

[0025] 일반적으로, 다양한 저전력 모드들에서 어떤 자원들을 배치할지를 선택하는 것은 "슬립 문제"로서 알려져 있다. 예를 들어, 전원을 내리고 자원을 그것의 요구된 동작 상태로 반환시키는데 필요한 전력이 프로세서 유휴 상태의 추정된 지속기간 동안 자원을 저전력 모드로 가지는 것에 의해 절약된 전력보다 크면, 모바일 디바이스의 배터리 수명은 프로세서가 유휴일 때마다 단순히 자원을 턴 오프하고 및/또는 그 자원을 저전력 상태로 배치하는 것에 의해 최대화되지 않을 수도 있다. 따라서, 어떤 자원들이 턴 오프되어야 하고 및/또는 저전력 모드들로 배치되어야 하는지 (및 어떤 저전력 모드가 그 자원에 대해 선택되어야 하는지) 를 결정하는 것은 통상, 프로세서 상태, 다른 디바이스 프로세서들의 상태들, 및 다른 팩터들, 이를테면 레이턴시, 전력 절약 잠재력, 소비 전력, 및 자원들/프로세서들의 상호의존도들의 분석을 필요로 한다.

[0026] 도 2는 다양한 양태들을 구현하기 위해 사용될 수도 있는 일 예의 시스템-온-칩 (SOC) (200) 아키텍처를 도시하는 아키텍처 도면이다. SOC (200) 는 다수의 이종 프로세서들, 이를테면 디지털 신호 프로세서 (DSP) (202), 모뎀 프로세서 (204), 그래픽스 프로세서 (206), 및 애플리케이션 프로세서 (208) 를 구비할 수도 있고, 그것들의 각각은 하나 이상의 코어들을 구비할 수도 있다. SOC는 또한 그 프로세서들 중 하나 이상에 접속된 하나 이상의 코프로세서들 (예컨대, 벡터 코-프로세서) (210) 을 구비할 수도 있다. 각각의 프로세서는 하나 이상의 코어들을 구비할 수도 있고, 각각의 프로세서/코어는 다른 프로세서들/코어들과는 독립적인 동작들

을 수행할 수도 있다. 예를 들어, SOC (200) 는 UNIX 기반 운영 체제 (예컨대, FreeBSD, LINUX, OS X 등) 를 실행하는 프로세서와 윈도우즈 기반 운영 체제 (예컨대, 마이크로소프트 윈도우즈 7) 를 실행하는 프로세서를 구비할 수도 있다.

[0027] SOC (200) 는 또한 센서 데이터, 아날로그-디지털 변환들, 무선 데이터 송신들을 관리하고, 다른 특수 동작들, 이를테면 게임들 및 영화들에 대한 인코딩된 오디오 신호들을 프로세싱하는 아날로그 회로 및 커스텀 회로 (214) 를 구비할 수도 있다. 그 SOC는 시스템 컴포넌트들 및 자원들 (216), 이를테면 전압 조정기들, 발진기들, 위상 동기 루프들, 주변 브리지들, 데이터 제어기들, 메모리 제어기들, 시스템 제어기들, 액세스 포트들, 타이머들과, 프로세서들, 및 컴퓨팅 디바이스 상에서 실행하고 있는 클라이언트들을 지원하는 데 사용되는 다른 유사한 컴포넌트들을 더 구비할 수도 있다.

[0028] 시스템 컴포넌트들 (216) 및 커스텀 회로 (214) 는 주변 디바이스들, 이를테면 카메라들, 전자 디스플레이들, 무선 통신 유닛들, 외부 메모리 칩들 등과 인터페이싱하는 회로를 구비할 수도 있다. 프로세서들 (202, 204, 206, 208) 은 하나 이상의 메모리 엘리먼트들 (212), 시스템 컴포넌트들, 및 자원들 (216) 및 커스텀 회로 (214) 에 상호접속/버스 모듈을 통해 상호접속될 수도 있으며, 그 상호접속/버스 모듈은 재구성가능 로직 게이트들의 어레이를 구비하고 및/또는 버스 아키텍처 (예컨대, CoreConnect, AMBA 등) 를 구현할 수도 있다.

[0029] 도 3은 다양한 양태들을 구현하기 위해 사용될 수도 있는 일 예의 멀티코어 프로세서 아키텍처를 도시하는 아키텍처 도면이다. 위에서 언급했듯이, 멀티코어 프로세서 (302) 는 (예컨대, 단일 기판, 다이, 통합된 칩 등 상에서) 가까이 근접한 둘 이상의 독립 프로세싱 코어들 (304, 306) 을 포함할 수도 있다. 프로세서들/코어들의 근접은 신호들이 칩 밖으로 이동해야 한다면 가능한 것보다 훨씬 높은 주파수/클록-레이트에서 메모리가 동작하는 것을 허용한다. 더구나, 코어들의 근접은 온 칩 메모리 및 자원들의 공유, 뿐만 아니라 코어들 간의 더 조정된 협력을 허용한다.

[0030] 멀티코어 프로세서 (302) 는 레벨 1 (L1) 캐시들 (312, 314) 및 레벨 2 (L2) 캐시 (316) 를 포함하는 멀티-레벨 캐시를 구비할 수도 있다. 멀티코어 프로세서 (302) 는 또한 버스/상호접속 인터페이스 (318), 메인 메모리 (320), 및 입력/출력 모듈 (322) 을 구비할 수도 있다. L2 캐시 (316) 는 L1 캐시들 (312, 314) 보다 클 (및 느릴) 수도 있지만, 메인 메모리 유닛 (320) 보다 작을 (및 실질적으로 빠를) 수도 있다. 각각의 프로세싱 코어 (304, 306) 는 L1 캐시 (312, 314) 에 대한 사적 액세스 및 L2 캐시 (316) 에 대한 공유된 액세스를 가지는 프로세싱 유닛 (308, 310) 을 구비할 수도 있다. L1 및 L2 캐시들 (312, 314) 은 프로세싱 유닛들 (308, 310) 에 의해 빈번하게 액세스되는 데이터를 저장하는 데 사용될 수도 있다. 메인 메모리 (320) 는 프로세싱 코어들 (304, 306) 에 의해 액세스되고 있는 큰 파일들 및 데이터 단위들을 저장하는 데 사용될 수도 있다.

[0031] 멀티코어 프로세서 (302) 는 프로세싱 코어들 (304, 306) 이 L1 캐시를 먼저, L2 캐시를 그 다음에 쿼리하고, 정보가 그 캐시들에 저장되어있지 않으면 메인 메모리에 쿼리하는 순서로 메모리로부터 데이터를 찾도록 구성될 수도 있다. 정보가 캐시들 또는 메인 메모리 (320) 에 저장되어 있지 않으면, 멀티코어 프로세서 (302) 는 외부 메모리/하드 디스크 (324) 로부터 정보를 찾을 수도 있다.

[0032] 프로세싱 코어들 (304, 306) 은 서로 동일할 수도 있고, 이종일 수도 있고 및/또는 상이한 기능들을 구현할 수도 있다. 따라서, 프로세싱 코어들 (304, 306) 은 (예컨대, 상이한 운영 체제들을 실행할 수도 있는) 운영 체제 관점 또는 (예컨대, 상이한 명령 세트들/아키텍처들을 구현할 수도 있는) 하드웨어 관점 중 어느 하나에서, 대칭적일 필요는 없다.

[0033] 프로세싱 코어들 (304, 306) 은 서로 버스/상호접속 (318) 을 통해 통신할 수도 있다. 각각의 프로세싱 코어 (304, 306) 는 일부 자원들에 대해 단독 제어 (exclusive control) 를 하고 다른 자원들을 다른 코어들과 공유할 수도 있다. 단일 칩 상의 다수의 코어들의 포함과, 코어들 간의 메모리 및 자원들의 공유는, 슬립 문제 및 최적의 저전력 자원 구성의 선택에 대한 솔루션들을 복잡하게 하는 다수의 전력 및 온도 관리 문제들을 일으킨다.

[0034] 위에서 논의된 바와 같이, 배터리 수명을 보존하기 위해, 각각의 자원은 그 자원이 배치될 수 있는 저전력 모드들 (LPRMs) 의 세트를 가질 수도 있다. SOC들 및 멀티코어 프로세서 시스템들에서, 프로세싱 유닛들/코어들의 각각은 상이한 운영 체제 하에서 동작할 수도 있고 코어들은 그 코어들이 저전력/슬립 모드에 진입하고 종료할 것으로 예상되는 (즉, 각각의 코어가 슬립 모드에 독립적으로 진입하고 종료하는데 필요할 수도 있는) 상이한 시간들, 작업부하 (workload) 들, 및 제약들의 상이한 세트들을 가질 수도 있다. 프로세싱 유닛들/코어

들의 각각은 또한 그것들을 다른 코어들의 동작들/상태들에 바인딩하는 자원들을 공유할 수도 있다. 예를 들어, 멀티-코어 프로세서/SOC는, 이론적으로는 저전력/슬립 모드에 독립적으로 진입할 수 있지만 모든 코어들이 저전력 모드에 있지 않는 한 셧 다운될 수 없는 루트 수정 발진기의 동작들에 바인딩되는 4 개의 코어들을 가질 수도 있다. 이는 코어들이 저전력 모드에 진입 및/또는 종료하기 전에 그 코어들 사이의 많은 명시적 핸드셰이킹, 잠금, 또는 직접 시그널링을 필요로 한다. 이들 팩터들은 SOC들 및/또는 멀티코어 프로세서들을 구비하는 시스템들에서 최적의 저전력 자원 구성의 선택을 곤란하게 한다.

[0035] 기존의 멀티코어/SOC 슬립 구현예들은 잘 확장되지 않고 최적의 전반적인 저전력 자원 모드 구성의 선택을 허용하지 않는다. 위에서 언급했듯이, 코어들/프로세싱 유닛들은 시스템이 글로벌 저전력 모드에 진입할 수 있기 전에 복잡한 잠금을 수행하고 동작들을 동기화해야 하고, 코어들/프로세싱 유닛들의 수가 (8, 16, 32 등으로) 증가함에 따라 상당한 병목들이 발생한다. 더구나, 멀티-코어 프로세서 시스템에서 프로세서/코어 유휴 상태들을 관리하는 기존의 솔루션들은 일반적으로는 사용되는 운영 체제의 유형에 의존한다. 예를 들어, 일부 운영 체제들 (예컨대, LINUX 등) 은, (예컨대, 모든 코어들이 슬립 모드에 있는) 최종 슬립을 하기 원하는 경우 모든 동작들이 제로 코어까지 라우팅 다운되도록, 슬립 프로세스 위에 존재하고 (슬립 프로세스 외부에서) 다른 코어들/자원들을 명시적으로 턴 오프시키는 프로세스를 통해 멀티-코어 슬립 동작들을 제어하도록 구성될 수도 있다. 이는 사실상 (제로 코어가 모든 다른 코어들의 역할을 하는) 단일 코어 솔루션에 유사한 방식으로 멀티-코어 시스템들이 자원들/다수의 코어들을 관리하는 것을 허용한다.

[0036] 다른 운영 체제들 (예컨대, 마이크로소프트 윈도우즈 등) 은, 상이한 코어들이 슬립 모드에 진입할 준비를 한 경우, 그들은 차단하고 마지막 코어가 정지하기를 대기하도록 구성될 수도 있다. 이 솔루션은, 모든 글로벌 자원들 및 코어들이 유휴가 되기까지, 코어들이 낮은 전력 상태에서 그러나 최저 시스템 전력 상태가 아닌 상태에서 실행하게 한다.

[0037] 다양한 양태들은 제로 코어가 다른 코어들을 모니터링할 것을 필요로 하지 않으며, 슬립 프로세스 위에 상주하는 프로세스가 코어들 (예컨대, LINUX 등) 을 모니터링하는 것을 필요로 하지 않고, 모든 글로벌 자원들이 유휴가 되기까지 코어들이 차단하고 대기하는 것을 필요로 하지 않는다. 기존의 솔루션들과는 달리, 다양한 양태들은 멀티코어 프로세서/SOC에서의 모든 코어들/프로세싱 유닛들이 대칭 방식으로 처리되는 것을 허용한다.
각각의 코어는 참조 카운트 (reference count) 를 모니터링하는 것에 의해 다른 코어들과는 독립적으로 그것의 동작 상태를 선택할 수도 있다.

[0038] 위에서 언급했듯이, SOC/멀티코어 프로세서에서, 각각의 프로세서/코어는 상이한 운영 체제 하에서 동작될 수도 있고, 각각의 코어/프로세서는 다른 코어들의 동작들/상태들에 자신들을 바인딩하는 자원들을 공유하며, 동시에, 그것들이 저전력/슬립 모드에 진입하고 종료할 것으로 예상되는 (즉, 각각의 코어가 슬립 모드에 독립적으로 진입하고 종료하는데 필요할 수도 있는) 상이한 시간들, 작업부하들, 및 제약들의 상이한 세트들을 가질 수도 있다. 따라서, 다이나믹 슬립 동작들이 멀티코어 환경에서 수행되는 경우, 특수한 고려사항들이 다양한 코어들 간에 데이터를 공유하고 공유된 자원들을 그것들의 저전력 모드들로 두는 것을 조정하기 위하여 취해져야만 하며, 이는 통상 코어들이 저전력 모드로 진입 및/또는 종료하기 전에 그 코어들 사이에 명시적 핸드셰이킹, 잠금, 또는 직접 시그널링을 필요로 한다. 다양한 양태들은 코어들이 저-전력 모드들을 선택하는 방법을 개정하며, 디스에이블될 수 있는 자원들을 식별하고, 디바이스 저-전력 구성 동안에 남아 있어야 하는 자원들을 식별한다. 다양한 양태들은 또한, 시스템 저전력 모드에 들어가기 전에 코어들 간에 임의의 명시적 핸드셰이킹, 잠금 또는 직접 시그널링을 요구하지 않고도, 이산 자원들이 신중히 핸들링되는 것과 글로벌 자원들이 코어들에 걸쳐 공유되는 것을 허용하는 방식으로, 제약들을 국소적으로 (즉, 특정 코어에 대함) 또는 전역적으로 (즉, 모든 코어들에 대함) 중 어느 하나로 설정하는 것에 의해 멀티코어 시스템들이 이들 동작들을 효율적으로 수행하는 것을 허용한다.

[0039] 도 4는 다양한 양태들에 따라 구현되는 멀티코어 프로세서 시스템 (400) 에서의 예의 흐름들을 도시하는 컴포넌트 흐름도이다. 멀티코어 프로세서 시스템 (400) 은 둘 이상의 독립 프로세싱 코어들 (404, 406) 을 구비할 수도 있으며, 그것들의 각각은 다른 코어들과는 독립적으로 슬립/저전력 모드에 배치될 수도 있다. 멀티코어 프로세서 시스템 (400) 은 또한 코어들 (404, 406) 중 하나 이상에 대해, 코어들이 슬립 상태/저전력 모드에서 깨어나게 할 수도 있는 인터럽트들을 발사 (fire) 하도록 구성된 인터럽트 모듈 (416) 을 포함할 수도 있다.

[0040] 멀티코어 프로세서 시스템 (400) 은 또한 다수의 자원들 (438, 440, 452) 을 포함할 수도 있다. 일부 자원들 (예컨대, 로컬 자원 (438, 440)) 은 코어 단위 (per-core) 기반으로 제어될 수도 있고 각각의 코어에 대해 로컬로 간주된다 (즉, 코어 레일의 전력 붕괴). 다른 자원들 (예컨대, 공유 자원 (454)) 은 하나 이상의

코어들에 의해 공유될 수도 있고 모든 코어들 사이에서 그것들의 LPRM들의 진입 전에 조정을 필요로 할 수도 있다.

[0041] 각각의 프로세서 코어 (404, 406) 는 코어 타이머 (412, 414) 및 서브시스템 전력 관리 (SPM) 하드웨어 블록 (408, 410) 을 가질 수도 있다. SPM 하드웨어 블록들 (408, 410) 은 코어가 프로세싱 동작들을 빠르게 재시작하는 능력을 유지하면서도 저전력 모드에 진입할 수도 있도록 그것들의 개별 코어들의 전력 상태들을 제어할 수도 있다.

[0042] 각각의 코어 (404, 406) 는 다른 코어들과는 독립적으로 실행하는 유휴 스레드 (418, 420) 를 실행할 수도 있다. 유휴 스레드들 (418, 420) 은 그것들의 개별 코어 타이머들 (412, 414) 및 서브시스템 전력 관리 (SPM) 하드웨어 블록들 (408, 410) 과 통신할 수도 있다. 각각의 유휴 스레드 (418, 420) 는 또한 그것들의 개별 코어들 (404, 406) 의 로컬 레이턴시 모듈 (430, 432), 로컬 기상 모듈 (426, 428) 및/또는 로컬 파킹된 모듈 (422, 424) 과, 그리고 코어들 간에 공유되는 글로벌 기상 모듈 (436) 및 글로벌 레이턴시 모듈 (434) 과 통신할 수도 있다. 일 양태에서, 레이턴시 모듈들 (430, 432, 434), 기상 모듈들 (426, 428, 436), 및 파킹된 모듈들 (422, 424) 은 노드 전력 아키텍처 (NPA) 구현예에서 노드들일 수도 있다.

[0043] 클라이언트 애플리케이션들은 특정 코어가 레이턴시 제약을 지키게 하기 위해 로컬 레이턴시 노드들 (430, 432) 에 등록할 수도 있다. 클라이언트 애플리케이션들은 그것들의 인터럽트가 어떤 코어에 발사될 것인지를 알지 못하면 글로벌 레이턴시 모듈 (434) 에 등록할 수도 있다. 일 양태에서, 인터럽트 레이턴시는 인터럽트의 발사 대 인터럽트 서비스 루틴 (ISR) 이 실행되는 시간 사이의 시간량으로서 계산될 수도 있다.

[0044] 일 양태에서, 멀티코어 프로세서 시스템 (400) 은 시스템에서 하나를 초과하는 실행 환경 (또는 "마스터") 들에 의해 공유된 자원들을 관리하는 자원 전력 관리자 (RPM) (450) 프로세서를 구비할 수도 있다.

[0045] 일 양태에서, 자원 전력 관리자 (RPM) (450) 는 슬립 요청들을 지키기 전에 각각의 코어의 서브시스템 전력 관리 (SPM) 하드웨어 블록들 (408, 410) 로부터의 핸드셰이크들을 기다릴 수도 있다.

[0046] 기상 모듈들 (426, 428, 436) 은 타이머 서브시스템을 통해 스케줄링되지 않은 예상된 기상 인터럽트들에 대한 입력을 슬립 서브시스템에 제공하기 위해 클라이언트 애플리케이션들을 인에이블시키도록 구성될 수도 있다. 이는, 인터럽트가 시스템에 발사되어 기상시킬 것으로 예상되는 경우의 지식으로 슬립 프로세스가 저전력 모드들의 선택을 최적화하는 것을 허용할 수도 있다. 클라이언트 애플리케이션들은 특정 코어에 대한 기상 이벤트를 나타내기 위해 로컬 기상 모듈들 (426, 428) 에 등록할 수도 있다. 클라이언트 애플리케이션들은 그것들의 인터럽트가 어떤 코어에 발사될 것인지를 알지 못하면 글로벌 기상 모듈 (436) 에 등록할 수도 있다.

[0047] 각각의 코어 (404, 406) 는 로컬 및/또는 글로벌 기상 입력들 및/또는 레이턴시 제약들을 지킬 지의 여부를 독립적으로 결정하도록 구성될 수도 있는데, 이들은 코어들이 파킹된 모듈 (422, 424) 자원들의 존재 및/또는 상태에 기초하여 결정할 수도 있다.

[0048] 일 양태에서, 파킹된 모듈들 (422, 424) 은 코어가 "액티브"인지 또는 "파킹됨"인지를 운영 체제 (또는 MP-DCVS 서브시스템) 가 나타내는 것을 허용하는, 각각의 개별 코어에 대한 노드 전력 아키텍처 (NPA) 자원을 포함할 수도 있다.

[0049] 액티브 상태는 운영 체제가 코어에 대한 스레드들을 능동적으로 스케줄링한다는 것과, 로컬 및 글로벌 양쪽 모두의 레이턴시 및 기상 제약들이 지켜져야 한다는 것을 나타낼 수도 있다.

[0050] 파킹된 상태는 운영 체제가 코어에 대한 스레드들을 능동적으로 스케줄링하지 않는다는 것과, 로컬 레이턴시 및 기상 제한들만이 지켜진다 (즉, 글로벌 레이턴시 및 기상 제약들은 무시될 수도 있다) 는 것을 나타낼 수도 있다.

[0051] 일 양태에서, 멀티코어 프로세서 시스템 (400) 은, 코어가 액티브 상태에 있는 경우 운영 체제는 양쪽 모두의 로컬 및 글로벌 레이턴시들 및/또는 기상 제한들을 지키고, 코어가 파킹된 상태에 있는 경우 운영 체제는 로컬 레이턴시 및 기상 제한들만을 지키도록 구성될 수도 있다.

[0052] 일 양태에서, 운영 체제가 모든 인터럽트들을 개개의 코어 (예컨대, 코어 0) 에 라우팅하도록 구성될 수도 있고, 멀티코어 프로세서 시스템은 개개의 코어 (즉, 코어 0) 만이 글로벌 기상/레이턴시 제약들을 지키도록 구성될 수도 있다. 이러한 양태들에서, 글로벌 기상/레이턴시 모듈은 코어의 (코어 0의) 로컬 기상/레이턴시 모듈 (예컨대, 기상 모듈 (426), 레이턴시 모듈 (430)) 로 별칭될 수도 있다.

[0053] 일 양태에서, 코어에서의 파킹된 모듈 (예컨대, 모듈 (422, 424)) 의 부재 (absense) 는 그 코어가 액티브 상태로 동작하게 한다.

[0054] 일 양태에서, 파킹된 모듈들 (422, 424) 은 디폴트로 액티브 상태에 있을 수도 있다.

[0055] 일 양태에서, 파킹된 모듈들 (422, 424) 은, 운영 체제가 코어에 대한 스레드들을 능동적으로 스케줄링하고 있지 않지만 코어가 지켜야 하는 등록된 레이턴시 및 기상 제약들에 대응하는 인터럽트들이 코어에 라우팅될 수도 있다는 것을 나타내는 제 3 상태 (예컨대, 유휴 상태) 를 식별하도록 구성될 수도 있다. 예를 들어, 코어 상에서 실행하는 운영 체제는 그 코어가 파킹된 (즉, 그 코어에 대한 스레드들을 스케줄링하고 있지 않은) 경우에 그 코어에 라우팅되는 인터럽트들을 가질 수 있도록 "파킹된 상태"를 정의할 수도 있다. 이러한 경우들에서, 인터럽트 서브시스템 (예컨대, 인터럽트 모듈 (416)) 은 코어가 유휴 상태에 있다는 것 및 코어가 글로벌 레이턴시 및 기상 제약들을 지켜야 한다는 것을 파킹된 모듈들 (422, 424) 에 통지하도록 구성될 수도 있다.

[0056] 다양한 양태들에서, 직접 쿼리되는 타이머 서브시스템에 연관되는 알려진 기상 이벤트가 있을 수도 있다. 다양한 양태들은 또한 기상 이벤트가 특정한 시간량 후에 발생할 것이라고 여기게 하는 "힌트"를 클라이언트들이 주는 것을 허용하는 깨어있는 (wake) 자원을 포함할 수도 있다.

[0057] 다양한 양태들에서, 시스템은 자원들 및/또는 코어들이 얼마나 오래 주어진 슬립 상태로 남아있을 가능성이 있는지를 예측 및/또는 제어하는 메커니즘들을 포함할 수도 있다. 이들 양태들에서, 프로세서는 자원들이 얼마나 오래 잠들어 있을 것으로 예상할 수 있는지를 제어하는 특정한 이벤트들을 연기할 수도 있다. 다양한 양태들에서, 자원들이 강제로 기상되는 하드 (hard) 기상 포인트가 있을 수도 있다. 다양한 양태들에서, 시스템은 예상되는 기상 시간 프레임을 결정하기 위해 자원들로부터의 "힌트들"을 이용할 수도 있다.

[0058] 다양한 양태들은 가장 효율적인 기상 시간을 추정하기 위해 프로세서에 의해 사용될 수도 있는 예측 알고리즘들을 구현할 수도 있다.

[0059] 다양한 양태들은 하드 한계 (limit), 힌트들, 및 학습 메커니즘들의 조합을 이용하여 예상되는 기상 시간을 결정할 수도 있다.

[0060] 다양한 양태들은, 결정된 예상된 기상 시간까지 자원들이 저전력 자원 모드들로 배치되면, 예상된 기상 시간을 이용하여 얼마나 많은 전력이 절약될 것인지를 결정할 수도 있다. 이러한 예상된 전력 절약량은 그 다음에 임박한 슬립 사이클에 대해 시스템에서 저전력 모드 구성을 구현하기 위해 저전력 모드에 배치할 특정한 자원들을 선택하는 (즉, 저전력 자원 모드들을 선택하는) 프로세스에서 사용될 수도 있다.

[0061] 일 양태에서, 기상 모듈들 (426, 428, 436) 은 다음의 스케줄링된 기상까지 남아있는 시간량 (예컨대, "하드 기상"까지의 지속기간) 을 식별하기 위해 쿼리될 수도 있다. 글로벌 기상 모듈 (436) 은 요청을 하는 코어에 지속기간을 반환할 수도 있고, 코어-특정 기상 모듈들 (426, 428) 은 그것들이 나타내는 코어에 지속기간을 반환할 수도 있다.

[0062] 일 양태에서, 기상 노드를 통해 스케줄링된 기상 (예컨대, 하드 기상) 에 대해 쿼리하는 대신, 스케줄링된 기상 정보는 프로세싱을 위해 슬립 태스크로 전달될 수도 있다. 예를 들어, 하이레벨 운영 체제 (HLOS) 가 프로세싱을 위해 슬립 태스크에 대해 유휴인 특정 코어에 하드기상 지속기간을 전달할 수도 있다.

[0063] 일 양태에서, 슬립 서브시스템은 모든 코어들에 의해 공유된 라이브러리를 포함할 수도 있다. 다수의 코어들은, 유휴 스레드의 콘텍스트 내 또는 다른 스레드의 콘텍스트 내 중 어느 하나에서, 공유된 데이터 (예컨대, 슬립 서브시스템 라이브러리) 를 동시에 판독하고 및/또는 수정할 수도 있다.

[0064] 위에서 언급했듯이, 프로세서에 의해 사용되는 일부 자원들은 코어 단위 기반으로 제어될 수도 있고 따라서 각각의 코어에 대해 로컬인 반면, 다른 자원들은 하나 이상의 코어들에 의해 공유될 수도 있다 (즉, 코어 레일의 전력 붕괴). 자원들이 글로벌일 (예컨대, 하나를 초과하는 코어들에 의해 공유될) 수도 있으므로, 코어들 간의 조정은 코어가 슬립 모드에 진입하는 것에 관해 또는 자원들이 저전력 모드들로 진입하는 것이 허용되기 전에 요청될 수도 있다. 일 양태에서, 멀티코어 프로세서 시스템 (400) 은 코어들 간의 조정을 제공하고 각각의 코어에 대해 저전력 모드들의 선택을 지원하는 다이나믹 슬립 프레임워크를 구비할 수도 있다. 다이나믹 슬립 프레임워크는 각각의 저전력 자원 (LPR) 이 인가되는 코어(들)를 식별할 수도 있다.

[0065] 특정 코어들 및 저전력 자원들 간의 관계를 식별하는 필드가 슬립 구조 (452) 에 추가될 수도 있다. 자원이 로컬이거나, 코어들의 서브세트 간에 공유되거나, 또는 모든 코어들 (글로벌) 간에 공유될 수도 있다. 예를 들어, 자원이 하나를 초과하는 코어들에 인가되면 (즉, LPRM이 글로벌이면), 그 코어들은 자원/코어가 저전력

모드에 배치될 수도 있는지를 결정하기 위해 공유 메모리에 저장된 참조 카운트에 액세스할 수도 있다. 자원이 특정 코어에만 인가되면 (즉, LPRM이 로컬이면), 자원이 인가되는 코어는 참조 카운트에 대한 필요 없이 그 자원을 그것의 저전력 모드로 둘 수도 있다.

[0066] 특정 코어에 대한 슬립 세트를 선택하기 위하여, 다이나믹 슬립 프레임워크는 어떤 코어(들)에 특정 LPR이 인가되는지를 알게 될 수도 있다. 이는 sleep_lpr 노드 (452) 의 추가된 필드에 쿼리하여 LPR이 인가되는 특정 코어들를 식별하는 것에 의해 달성될 수도 있다. 추가된 필드는 각각의 비트가 특정 코어에 대응하는 (즉, 비트 0이 코어 0에 대응하고, 비트 1이 코어 1에 등등으로 대응하는) 다수의 비트들을 포함할 수도 있다. 비트들 중 임의의 것이 설정되면, LPR은 설정 비트들에 의해 식별된 코어들에만 인가되는 것으로서 식별될 수도 있다. 비트들이 설정되어 있지 않으면, 또는 모든 비트들이 설정되면, LPR은 모든 코어들 간에 공유된 (즉, LPR은 글로벌이라는) 것으로서 처리될 수도 있다.

[0067] 일 양태에서, 슬립 서브시스템은 슬립 서브시스템에 등록되는 컴포넌트 모드들의 유효한 조합들을 표현하는 합성된 모드들을 생성할 수도 있다. 합성된 모드들은 컴포넌트 모드들에 주어진 의존도, 레이턴시, 순서, 및 전력 절약 정보에 기초하여 생성될 수도 있다.

[0068] 일 양태에서, 각각의 코어는, LPRM들이 인에이블되고 디스에이블됨에 따라 슬립 프레임워크에 의해 즉석에서 간결해지는 인에이블된 목록을 포함할 수도 있다. 특정한 코어가 유휴 상태가 되고 인에이블된 모드들의 목록을 요청할 때마다, 슬립 레지스트리는 요청하는 코어에 대한 목록을 반환할 수도 있다. 예를 들어, 다음의 LPRM들이 시스템에 존재하면:

코어 0	코어 1	글로벌
cpu_vdd0.off	cpu_vdd1.off	cxo.shutdown pxo.shutdown

[0069]

[0070] 각각의 코어에 대한 합성된 모드들의 목록은 다음이 될 수도 있다 (의존성들이 암시됨):

코어 0	코어 1
cpu_vdd0.off + cxo.shutdown + pxo.shutdown	cpu_vdd1.off + cxo.shutdown + pxo.shutdown
cpu_vdd0.off + cxo.shutdown	cpu_vdd1.off + cxo.shutdown
cpu_vdd0.off + pxo.shutdown	cpu_vdd1.off + pxo.shutdown
cpu_vdd0.off	cpu_vdd1.off

[0071]

[0072] 합성된 모드들은 다양한 양태들의 부분으로서 요구되지 않고, 그러므로 이 합성된 모드들의 설명은 청구항들에서 구체적으로 한정되지 않는 한 청구항들의 범위를 임의의 방식으로 제한하지 않는다는 것에 주의해야 한다.

[0073] LPRM들의 각각이 인에이블되거나 또는 디스에이블됨에 따라, 슬립 레지스트리는 각각의 코어에 대한 인에이블된 목록을 간결하게 할 수도 있어서 인에이블된 목록은 그 코어에 대한 인에이블된 모드들의 현재의 목록을 항상 반영한다. 예를 들어, cpu_vdd0.off 가 코어 0에 대해 디스에이블되면, cpu_vdd1.off 에는 여전히 코어 1이 진입하고, 슬립 레지스트리는 이 모드들을 반영하기 위해 각각의 코어에 대한 인에이블된 목록을 간결하게 할 수도 있다.

[0074] 일 양태에서, 합성된 모드에서의 컴포넌트 모드들의 모두가 인에이블되는 경우, 전체 합성된 모드는 인에이블되고 슬립의 후보가 될 수도 있다.

[0075] 일 양태에서, 솔버가 진입을 위해 로컬 (비-공유된) LPRM을 선택하면, LPRM은 다른 코어들과의 조정 없이 진입될 수도 있고 로컬 LPRM들을 위한 진입 함수들은 무조건으로 실행될 수도 있다.

[0076] 도 5는 상이한 코어들의 "진입" 및 "종료" 함수들을 조정하고 시스템 저전력 구성을 식별하고 입력하는 멀티코어 프로세서 시스템에 의해 구현될 수도 있는 일 양태의 방법 (500) 을 예시한다. 그 멀티코어 프로세서 시스템은 공유된 모드들에 대한 참조 카운팅을 위해 이용되는, 저전력 자원 모드 (LPRM) 에 연관된 하나 이상의 메모리 로케이션들을 포함할 수도 있다. 위에서 논의된 바와 같이, 멀티코어 프로세서 시스템에 의해 사용되는 자원들은 로컬일 (코어 단위 기반으로 제어될) 또는 공유될 (하나를 초과하는 코어들에 의해 사용될) 수도 있다. 도 5의 도시된 예에서, LPRM a는 코어 0에 대해 로컬이며, LPRM b는 코어 1에 대해 로컬이고, LPRM들

c, d, 및 e는 모든 코어들에 의해 공유된다.

[0077] 블록 502 및 블록 504에서, 멀티코어 프로세서의 코어 0 및 코어 1은 실행하고 있고 로컬 및 글로벌 자원들을 사용하고/드라이브하고 있다. 블록 506에서, 코어 0은 유휴 상태에 진입하고 슬립 모드들을 "구하는" 솔버 프로세스의 실행을 개시할 수도 있다. 위에서 논의된 바와 같이, 프로세서가 시스템 저전력 모드에 진입할 준비가 된 경우, 이를테면 코어가 유휴 상태에 있는 경우, 저전력 태스크 (예컨대, "솔버" 태스크/프로세스) 는 저전력 모드 마스크들에 액세스하여 저전력 상태에 배치될 수 있는 자원들을 식별하고, 주어진 다이나믹 시스템 상태 (예컨대, 현재의 액티브 클라이언트들, 요청된 레이턴시, 예상된 유휴 시간, 및 온도) 에 기초하여 그들 자원들에 대해 진입할 적절한 저전력 자원 모드들을 결정할 수도 있다. 다시 말하면, 코어가 유휴 상태에 진입할 수 있는 경우, 저전력 태스크는 다양한 자원들에 대해 어떤 저전력 모드들이 진입되어야 하는지를 결정하기 위해 "솔버" 프로세스를 실행할 수도 있다. 더구나, 임의의 코어가 정지하는 (go down) 맨 처음 것이 되고 임의의 코어가 깨어나는 (come up) 마지막 것이 되는 것을 허용하기 위하여, 각각의 코어는, 국소적으로 그리고 전역적으로 모두, 그것이 진입할 수 있는 모드들의 전체 세트를 "구할 (solve)" 수도 있다. 예를 들어, 블록 506에서, 코어 0은 모드들 a, c, 및 d를 이용가능한 저전력 모드들로서 식별하고, 로컬 모드 a, 및 글로벌 모드들 c 및 d를 선택하는 동작들을 수행할 수도 있다.

[0078] 선택되는 모든 모드에 대해, 그 모드에 대한 진입 함수 및 종료 함수는 호출될 수도 있다. 일 양태에서, 코어들 간에 공유되는 모드들 (예컨대, LPRM들 c, d, 및 e) 에 대해, 진입 함수 및 종료 함수의 콘텐츠들은 그 모드를 공유하는 모든 코어가 진입을 위해 그 모드를 선택하는 경우에만 실행될 수도 있다. 이는 LPRM에 몇 번이나 진입하고 종료했는지를 표현하는 각각의 LPRM에 연관된 카운트를 유지하는 것에 의해 달성될 수도 있다. 그 카운트는 LPRM을 공유하는 실행중인 코어들의 수에서 시작할 수도 있다. 도 5의 도시된 예에서, 멀티코어 프로세서는 2 개의 코어들 (코어 0, 코어 1) 을 포함하고, 이에 따라, 글로벌 LPRM들 c, d, 및 e의 각각은 2의 초기 값 (모드 c = 2, 모드 d = 2, 모드 e = 2) 을 가진다. LPRM 진입 함수가 호출될 때마다, 카운트는 원자적으로 (atomically) 점감 (decrement) 될 수도 있다. 예를 들어, 블록 508에서, 코어 0은 글로벌 LPRM c (예컨대, LPRM c가 1로 점감함) 및 글로벌 LPRM d (예컨대, 모드 d가 1로 점감함) 에 연관된 값들을 점감할 수도 있다. 글로벌 LPRM 메모리 로케이션에 대한 카운트가 0에 도달하는 경우, 그 LPRM에 대한 진입 함수는 실행될 수도 있다. 비슷하게, 코어가 기상하는 경우, 그것은 LPRM 종료 함수를 호출하고, 카운트는 원자적으로 점증 (increment) 될 수도 있다. 종료 함수는 카운트가 0에서 1로 전이하는 경우에 콘텐츠들 (예컨대, 종료 태스크들) 의 실행을 시작할 수도 있다.

[0079] 일 양태에서, 슬립 태스크는 LPRM을 선택하는 모든 코어에 대해 진입 및 종료 함수들을 실행할 수도 있다. 진입 함수 및 종료 함수들을 실행할 지의 여부에 대한 카운트 조작 및 결정은 LPRM에 의해 제어될 수도 있으며, 일 양태에서, LPRM의 소유자는 코어들 간에 그 LPRM의 진입 및 종료를 조정할 수도 있다. LPRM은 LPRM의 요건들에 의존하여, 코어들 간을 조정하기 위해 다른 메커니즘들을 사용하거나, 또는 스핀락할 수도 있다.

[0080] 일 양태에서, 솔버 태스크가 진입을 위해 로컬 (비-공유된) LPRM을 선택하면, 멀티코어 프로세서 시스템은 그 LPRM이 다른 코어들과의 조정 없이 진입될 수 있다고 가정할 수도 있고, 로컬 LPRM들에 대한 진입 함수들은 무조건으로 실행될 수도 있다. 예를 들어, 블록 508에서, 로컬 LPRM은 코어 1과의 조정 없이 진입될 수도 있다. 그러나, 글로벌 LPRM들 c 및 d에 대응하는 메모리 로케이션들이 0이 아닌 값들 (모드 c 메모리 = 1, 모드 d 메모리 = 1) 을 포함하기 때문에 글로벌 LPRM들 c 및 d는 블록 508에서 진입되지 않는다.

[0081] 블록 510에서, 코어 1은 유휴 상태로 되어 LPRM들 b, d, 및 e를 이용가능한 저전력 모드들로서 식별하는 동작들을 수행할 수도 있다. 또한 블록 510에서, 코어 1은 그것이 입장할 수 있는 모드들의 전체 세트를 국소적으로 (예컨대, LPRM b) 및 전역적으로 (예컨대, LPRMs d 및 e) 양쪽 모두로 "구할" 수도 있다. 위에서 논의된 바와 같이, 진입될 수 있는 모드들의 전체 세트를 구하는 것 (solving) 은 임의의 코어가 정지하는 맨 처음 것이 되고 임의의 코어가 깨어나는 마지막 것이 되는 허용한다.

[0082] 블록 512에서, 코어 1은 로컬 LPRM b에 진입하며, 글로벌 LPRM d의 값을 점감 (예컨대, LPRM d가 0으로 점감) 하고 글로벌 LPRM e의 값을 점감 (예컨대, LPRM e가 1로 점감) 할 수도 있다. 위에서 언급했듯이, LPRM에 몇 번이나 진입하고 종료했는지를 나타내는 각각의 글로벌 LPRM (예컨대, LPRM들 c, d, 및 e) 에 연관된 카운트들을 유지하는 것에 의해, 멀티코어 프로세서 시스템은 그 모드를 공유하는 모든 코어가 진입을 위해 그것을 선택한 경우에만 진입 함수 및 종료 함수의 콘텐츠들이 실행되는 것을 보장할 수도 있다.

[0083] 블록 514에서, 코어 0 및 코어 1 양쪽 모두는 슬립 모드에 진입하고 슬립 세트를 전송할 수도 있다. 글로벌 모드 d에 대응하는 메모리 로케이션이 0의 값을 포함하므로, 글로벌 LPRM d는 블록 512에서 진입될 수도 있다.

[0084] 블록 516에서, 코어 0은 기상 이벤트 (예컨대, 인터럽트) 를 검출할 수도 있다. 블록 518에서, 코어 0은 로컬 LPRM a를 종료하고 글로벌 LPRM들 c 및 d에 대응하는 메모리 로케이션의 값을 점증 (예컨대, LPRM c는 2로 점증하며, LPRM d은 1로 점증) 하는 것에 의해 슬립 모드들을 종료할 수도 있다. LPRM d에 대응하는 메모리 로케이션의 값이 0의 값을 더 이상 포함하지 않으므로 (LPRM d =1), 코어 0은 또한 글로벌 LPRM d의 종료 함수의 콘텐츠들을 실행할 수도 있다.

[0085] 위에서 언급했듯이, 다양한 양태들은 하나의 코어 (예컨대, 코어 1) 에 의해 진입되고 및 다른 코어 (예컨대, 코어 0) 에 의해 종료되도록 글로벌 LPRM들을 인에이블시킨다. LPRM d는 (블록 512에서) 코어 1에 의해 진입되었고 (블록 518에서) 코어 0에 의해 종료되었다는 것에 주의해야 한다.

[0086] 블록 520에서, 코어 0은 다시 유휴 상태로 되며 (go idle), 로컬 LPRM a를 "구하고", 코어 1과의 조정 없이 로컬 LPRM a에 진입 (예컨대, 로컬 LPRM a의 진입 함수의 실행을 개시) 할 수도 있다. 블록 522에서, 코어 0은 슬립 모드로 진입하고 슬립 세트를 전송하여서 글로벌 모드들 (예컨대, 모드들 c, d, 및 e) 중 어느 것에도 진입되지 않을 수도 있다.

[0087] 블록 526에서, 기상 이벤트가 코어 1 상에서 검출될 (예컨대, 인터럽트가 코어 1 상에서 검출될) 수도 있다. 블록 528에서, 멀티코어 프로세서는 로컬 모드 b를 종료하고, 글로벌 모드들 d 및 e에 대응하는 메모리 로케이션들의 값들을 점증 (예컨대, 모드 d는 2로 점증하고, 예컨대, 모드 e는 2로 점증) 하는 것에 의해 슬립 모드들을 종료할 수도 있다.

[0088] 다양한 양태들에서, 멀티코어 프로세서 시스템은 제약들을 지키도록 구성될 수도 있다. 다시 말하면, 코어가 유휴 상태로 되고 그것의 저전력 모드들을 선택하는 경우, 유휴 프로세싱이 시스템의 나머지 부분에 대해 투명해지게 하기 위하여 충족되어야만 하는 특정한 제약들이 있다. 이들 제약들은 런 타임에서 모아지고, 코어가 유휴 상태로 되는 경우에 슬립 코드에 피드될 수도 있다. 임의의 특정 시점에, 다른 코어가 깨어 있고 그들 제약들을 수정하는 스레드를 실행하는 동안, 하나의 코어는 제약들의 특정한 세트와 함께 유휴로 갈 수도 있다.

[0089] 일부 경우들에서, 이를테면 레이턴시 버짓 (latency budget) 이 증가하거나 또는 LPRM이 인에이블되면, 제약들은 잠들어 있는 코어들이 새로운 제약들에 대해 이제 최적은 아닌 모드들의 세트에 진입하는 그런 방식으로 수정될 수도 있다. 이 경우, 잠들어 있는 코어들은 최적화로 기상될 수도 있다. 다른 경우들에서, 이를테면 레이턴시 버짓이 감소하거나 또는 LPRM이 디스에이블되면, 제약들은 잠들어 있는 코어들이 새로운 제약들을 위반하는 모드들의 세트에 진입하도록 수정될 수도 있다. 이 경우, 잠들어 있는 코어들은 올바른 시스템 거동을 획득하기 위하여 기상될 수도 있다.

[0090] 어떤 코어들이 잠들어 있는지를 결정하기 위하여, 코어가 잠들어있는지의 여부를 나타내는 각각의 코어에 대한 변수는 실행하고 있는 스레드들 및 유휴인 스레드들 사이에 공유될 수도 있다. 이 변수는 그 코어가 유휴 스레드에 진입하는 경우 그 코어에 의해 점증될 수도 있고, 유휴 스레드를 종료하는 경우에 점감될 수도 있다.

[0091] 일 양태에서, 공유된 데이터는 저장된 데이터의 유형 및/또는 그 데이터가 액세스되는 방식에 기초하여 보호될 수도 있다. 멀티코어 프로세서 시스템은 스핀락들, 운영 체제 (OS) 잠금들, 또는 데이터에 대한 액세스를 보호하고 동기화하는 프로세서 특화 원자적 명령들을 구현할 수도 있다. 멀티코어 프로세서 시스템은 또한, 공유된 데이터를 읽는 스레드들이 대기 없는 방식으로 행해지고 데이터를 수정하는 경우 동기화만이 필요하도록 동작들을 수행할 수도 있다.

[0092] 도 4를 참조하여 위에서 논의된 바와 같이, 일 양태에서, 멀티코어 프로세서 시스템은 시스템에서 하나를 초과하는 실행 환경 (또는 "마스터") 들에 의해 공유된 자원들을 관리하는 자원 전력 관리자 (RPM) 프로세서를 구비할 수도 있다. 각각의 마스터는 마스터가 저전력 모드들에 두고 싶어하는 자원들에 대한 "슬립 세트"를 생성할 수도 있다. 그 슬립 세트는 다수의 RPM-관리된 자원들, 및 각각의 자원에 대한 소망의 상태 값들을 포함하는 트랜잭션을 포함할 수도 있다. RPM에 대해, 이는 그 마스터에 대한 "투표"로 해석할 수도 있다. 일 양태에서, 모든 마스터 투표들의 수집된 상태만이 자원에 대해 지켜지는 최종 상태이다. 멀티코어 환경은 슬립 코드에 의해서만 호출되는 RPM 슬립 드라이버, 및 슬립이 실행중인 프로세스를 가질 수도 있다. 그 시스템은 또한 슬립 콘텍스트 외부로 RPM 메시지를 전송하는 프로세스들에 의해 호출되는 RPM 액티브 모드 드라이버를 구비할 수도 있다. RPM 드라이버가 멀티코어 환경에서 슬립 콘텍스트로 실행하고 있는 경우, 슬립 트랜잭션이 적절히 만들어지며, 바람직하지 않은 경우에 슬립 세트들에는 진입되지 않고, 액티브 모드 RPM 메시지들로의 조정은 준비가 되어 있는 것을 보장하는 것이 특별히 고려되어야만 한다.

[0093] RPM 슬립 드라이버를 구비할 수도 있는 멀티코어 환경에 대해 다음의 가정들이 이루어질 수도 있다: 각각의 코어는, 당해 코어의 전력 상태를 제어하고 그 코어가 그것의 저전력 모드에 진입하는 경우에 RPM과 핸드셰이크하는 SPM 하드웨어 블록을 가지며; 앱 프로세서는 전체적으로 하나의 RPM 마스터이고 (코어 당 하나의 마스터는 아님), 이에 따라 앱 프로세서에 대해 하나의 슬립 세트가 있으며; RPM은 앱 슬립 세트를 지키기 전에 각각의 코어의 SPM으로부터의 핸드셰이크들을 기다리며; RPM 슬립 드라이버는, 슬립 코드가 먼저 RPM 시작 함수를 호출하여 트랜잭션을 시작한 다음, 자원들을 슬립 세트에 추가하기 위해 후속 호출들이 이루어진 후, 트랜잭션을 전송하기 위해 RPM 전송 함수가 호출되는 그런 방식으로 사용된다.

[0094] 위에서 언급했듯이, 각각의 LPRM은 "진입" 함수를 호출하기 위해 마지막 코어에 의해 한 번만 진입될 수도 있다. 그러나, 다수의 코어들이 그것들의 진입 함수들을 동시에 호출하고 슬립 세트에 대한 최종 RPM 트랜잭션을 생성하기 위한 책임을 공유할지도 모를 가능성이 존재한다. 양쪽 모두의 코어들이 동일한 RPM 핸들을 사용하여 트랜잭션을 조작할 수도 있으므로, 그 코어들은 트랜잭션을 공유할 수도 있고, 트랜잭션은 임의의 코어에 의해 시작되거나, 수정되거나, 또는 전송될 수도 있다. 다른 코어가 RPM 트랜잭션을 전송하고 있는 동안 하나의 코어가 RPM 트랜잭션을 시작하려고 시도하는 것을 막기 위해, 슬립 스핀락이 사용될 수도 있다. 그 스핀락은 RPM이 시작하고 RPM이 전송하는 쯤에만 홀딩될 수도 있다. 그것은 자원들을 슬립 세트에 추가하는 경우에 홀딩되지 않을 수도 있는데, 각각의 코어가 슬립 세트의 부분들을 분리하기 위해 쓰고 있을 것이라고 가정될 수 있어서이다.

[0095] 도 6은 멀티코어 프로세서 시스템이 트랜잭션의 인터리브된 구축 (building) 을 조정하는 일 양태의 방법 (600) 을 예시한다. 그 멀티코어 프로세서 시스템은 상이한 저전력 자원 모드들 (LPRM) 의 각각에 연관된 하나 이상의 레지스터들/메모리 로케이션들, 뿐만 아니라 rpm.start 및 rpm.send 동작들에 연관된 레지스터들/ 메모리 로케이션들을 포함할 수도 있다. 도 6의 도시된 예에서, 글로벌 LPRM들 c 및 d는 모든 코어들 (예컨대, 코어 0 및 코어 1) 에 의해 공유된다.

[0096] 블록들 602 및 604에서, 멀티코어 프로세서의 코어 0 및 코어 1은 실행 상태에 있고 글로벌 자원들을 사용하고/드라이브하고 있다. 블록 606에서, 코어 0은 유휴 상태로 되고 슬립 모드들을 "구하는" 동작들을 수행 (예컨대, 다양한 자원들에 대해 어떤 저전력 모드들이 진입되어야 하는지를 결정하기 위해 "솔버" 태스크/프로세스를 실행) 할 수도 있다. 이 프로세스의 부분으로서, 코어 0은 모드들 c 및 d를 이용가능한 저전력 모드들로서 식별하고, 블록 606에서의 진입을 위해 글로벌 모드들 c 및 d를 선택하는 동작들을 수행할 수도 있다.

[0097] 위에서 논의된 바와 같이, 멀티코어 프로세서 시스템은 그 LPRM에 몇 번이나 진입하고 종료하였는지를 나타내는 각각의 LPRM에 연관된 카운트를 유지할 수도 있다. 그 카운트는 LPRM을 공유하는 실행중인 코어들의 수에 대응하는 값을 초기에 포함할 수도 있다. 도 6의 도시된 예에서, 2 개의 코어들 (즉, 코어 0 및 코어 1) 이 글로벌 LPRM들 c 및 d를 공유하므로, LPRM들 c 및 d에 연관된 메모리 로케이션들은 각각 2의 값 (LPRM c = 2, LPRM d =2) 으로 시작한다. 비슷하게, rpm.start 및 rpm.send 함수들에 연관된 메모리 로케이션들은 2의 값 (rpm.start =2, rpm.send=2) 을 저장할 수도 있다.

[0098] 블록 608에서, 코어 0은 rpm.start 함수에 진입하고 rpm.start 함수에 연관된 변수 및/또는 메모리 로케이션에 저장된 값을 (rpm.start = 1으로) 점감시킬 수도 있다. rpm.start 함수에 연관된 메모리 로케이션이 영이 아닌 값을 포함하므로, 코어 0은 RPM 트랜잭션의 시작을 개시하지 않는다.

[0099] 블록 610에서, 코어 1은 유휴 상태로 진입하며, 슬립 모드들을 구하고, 모드들 c 및 d을 선택할 수도 있다. 블록 612에서, 코어 1은 rpm.start 함수에 진입하고 rpm.start 함수에 연관된 메모리 로케이션에 의해 저장된 값을 (rpm.start = 0으로) 점감시킬 수도 있다. rpm.start 함수에 연관된 메모리 로케이션이 1에서 0으로 전이하는 값을 저장 (즉, 이제 0 값을 포함) 하므로, 코어 1은 RPM 트랜잭션의 시작을 개시할 수도 있다.

[0100] 블록 614에서, 코어 1은 글로벌 LPRM c에 진입하고 LPRM c에 연관된 메모리 로케이션에 의해 저장된 값을 (LPRM c = 1로) 점감시킬 수도 있다. LPRM c에 연관된 메모리 로케이션이 영이 아닌 값을 포함하므로, 코어 1은 블록 614에서 LPRM c를 슬립 세트에 추가하지 않는다.

[0101] 블록 616에서, 코어 0은 글로벌 LPRM c에 진입하고 LPRM c에 연관된 메모리 로케이션에 의해 저장된 값을 (LPRM c = 0으로) 점감시킬 수도 있다. LPRM c에 연관된 메모리 로케이션에 의해 저장된 값이 이제 0 값을 포함하므로, 코어 0은 LPRM c를 슬립 세트에 추가할 수도 있다. 다시 말하면, 블록 616에서, 코어 0은 글로벌 LPRM c를 코어 1에 의해 시작된 슬립 세트 트랜잭션에 추가한다.

[0102] 블록 618에서, 코어 0은 글로벌 LPRM d에 진입하고 LPRM d에 연관된 메모리 로케이션에 의해 저장된 값을 (LPRM

d = 1로) 점감시킬 수도 있다. LPRM d에 연관된 메모리 로케이션이 영이 아닌 값을 포함하므로, 코어 0은 블록 618에서 LPRM d를 슬립 세트에 추가하지 않는다.

[0103] 블록 620에서, 코어 1은 글로벌 LPRM d에 진입하고 LPRM d에 연관된 메모리 로케이션에 의해 저장된 값을 (LPRM d = 0으로) 점감시킬 수도 있다. LPRM d에 연관된 메모리 로케이션에 의해 저장된 값이 이제 0 값을 포함하므로, 코어 1은 블록 620에서 LPRM d를 슬립 세트에 추가할 수도 있다.

[0104] 블록 622에서, 코어 1은 rpm.send에 진입하고 rpm.send 함수에 연관된 메모리 로케이션에 의해 저장된 값을 (rpm.send = 1로) 점감시킬 수도 있다. rpm.send 함수에 연관된 메모리 로케이션이 영이 아닌 값을 포함하므로, 코어 1은 RPM 트랜잭션을 전송하지 않는다. 블록 624에서, 코어 0은 rpm.send에 진입하고 rpm.send 함수에 연관된 메모리 로케이션에 의해 저장된 값을 (rpm.send = 0으로) 점감시킬 수도 있다. rpm.send 함수에 연관된 메모리 로케이션에 의해 저장된 값이 1에서 0으로 전이 (즉, 이제 0 값을 포함) 하므로, 코어 0은 RPM 트랜잭션을 전송할 수도 있다. 따라서, 블록 624에서, 코어 0은 코어 1이 시작했던 슬립 세트 트랜잭션을 전송한다.

[0105] 블록 626에서, 양쪽 모두의 코어들은 슬립 모드에 있을 수도 있으며, 슬립 세트는 전송될 수도 있고, 글로벌 LPRM들 c 및 d는 진입될 수도 있다.

[0106] 위에서 언급했듯이, 다른 코어가 슬립 세트 트랜잭션을 만드는 중에 있을 동안 하나의 코어가 기상하고 트랜잭션 또는 저전력 모드들의 세트를 수정할 가능성이 있다. 이 경우, 기상했던 코어는 다른 코어가 그것의 트랜잭션을 전송하기 전에 새로운 트랜잭션을 시작할 수 있다. 이 일이 일어나면, RPM 드라이버는 제 1 트랜잭션을 소거하고 제 2 트랜잭션만을 지킬 수도 있다.

[0107] 도 7은 중첩 RPM 트랜잭션들을 전송하고, 중첩 RPM 전이들을 전송하는 코어들 간에 진입 및 종료 함수들을 조정하는 일 양태의 방법 (700) 을 도시한다. 도 7의 도시된 예에서, LPRM들 c 및 d는 글로벌 (예컨대, 코어 0 및 코어 1에 의해 공유됨) 이고, rpm.start 함수, rpm.send 함수 및 LPRM들 c 및 d에 연관된 메모리 로케이션들은 각각 1의 값 (LPRM c = 1, LPRM d = 1, rpm.start = 1, rpm.send = 1) 을 저장한다.

[0108] 블록 702에서, 코어 1은 실행 상태이다. 블록 704에서 코어 0은 슬립 상태에 있고 글로벌 모드들 c 및 d에 진입되었다.

[0109] 블록 706에서, 코어 1은 유휴 상태로 되며, 다양한 자원들에 대해 어떤 저전력 모드들에 진입되어야하는지를 결정하기 위해 "솔버" 프로세스를 실행하며, 모드들 c 및 d를 이용가능한 저전력 모드들로서 식별하고, 진입할 글로벌 LPRM들 c 및 d를 선택할 수도 있다. 블록 708에서, 코어 1은 rpm.start 함수에 진입하고 rpm.start 함수에 연관된 메모리 로케이션에 의해 저장된 값을 (rpm.start = 0으로) 점감시킬 수도 있다. rpm.start 함수에 연관된 메모리 로케이션에 의해 저장된 값이 1에서 0으로 전이 (즉, 이제 0 값을 포함) 하므로, 코어 1은 블록 708에서 RPM 트랜잭션의 시작을 개시할 수도 있다.

[0110] 블록 710에서, 코어 1은 LPRM들 c 및 d에 진입하고 LPRM들 c 및 d에 연관된 메모리 로케이션들에 의해 저장된 값들을 (LPRM c = 0, LPRM d = 0으로) 점감시킬 수도 있다. LPRM들 c 및 d에 연관된 메모리 로케이션들에 의해 저장된 값들이 1에서 0으로 전이하므로, 코어 1은 또한 블록 710에서 LPRM들 c 및 d를 슬립 세트에 추가할 수도 있다.

[0111] 블록 722에서, 코어 1은 다양한 다른 함수들을 실행 (예컨대, 다른 진입 함수들을 실행, 값들을 점감하는 등등을) 할 수도 있다.

[0112] 블록 712에서, 기상 이벤트가 코어 0 상에서 검출될 (예컨대, 인터럽트가 코어 0 에 의해 수신될) 수도 있다. 블록 714에서, 멀티코어 프로세서는 슬립 모드를 종료하고 rpm.start, rpm.send, 및 글로벌 LPRM들 c 및 d에 대응하는 메모리 로케이션들의 값들을 점증 (예컨대, LPRM c는 1로 점증, LPRM d는 1로 점증, rpm.start는 1로 점증, rpm.send는 2로 점증) 시킬 수도 있다.

[0113] 블록 716에서, 시스템은 글로벌 LPRM c를 디스에이블시키는 스레드를 실행할 수도 있다.

[0114] 블록 718에서, 코어 0은 유휴 상태로 진입하며, 다양한 자원들에 대해 어떤 저전력 모드들에 진입되어야하는지를 결정하기 위해 "솔버" 프로세스를 실행하며, 이용가능한 저전력 모드들을 식별하고, 진입할 글로벌 LPRM d를 선택할 수도 있다. 블록 720에서, 코어 0은 rpm.start으로 진입하며, rpm.start에 대응하는 메모리 로케이션들의 값을 (rpm.start = 0으로) 점감시키고, 새로운 RPM 트랜잭션의 시작을 개시할 수도 있다. 또한 블록 720에서, RPM 드라이버는 코어 1로부터 이전의 트랜잭션을 그만둘 수도 있다. 일 양태에서, RPM 드라이버는

새로운 RPM 트랜잭션의 시작의 검출에 응답하여 이전의 트랜잭션들을 그만두도록 구성될 수도 있다.

[0115] 블록 724에서, 코어 1은 rpm.send에 진입하고 rpm.send 함수에 연관된 메모리 로케이션에 의해 저장된 값을 (rpm.send = 1로) 점감시킨다. rpm.send 함수에 연관된 메모리 로케이션들이 영이 아닌 값을 포함하므로, 코어 1은 RPM 트랜잭션을 전송하지 않는다.

[0116] 블록 726에서, 코어 0은 글로벌 LPRM d에 진입하고 LPRM d에 연관된 메모리 로케이션들에 의해 저장된 값을 (LPRM d = 0으로) 점감시킬 수도 있다. LPRM d에 연관된 메모리 로케이션들에 의해 저장된 값이 0 값을 포함하므로, 블록 726에서, 코어 0은 또한 LPRM d를 트랜잭션에 추가할 수도 있다.

[0117] 블록 728에서, 코어 0은 rpm.send에 진입하고 rpm.send 함수에 연관된 메모리 로케이션들에 의해 저장된 값을 (rpm.send = 0으로) 점감시킬 수도 있다. rpm.send 함수 트랜잭션들에 연관된 메모리 로케이션들에 의해 저장된 값이 1에서 0으로 전이 (즉, 이제 0 값을 포함) 하므로, 코어 0은 코어 0이 시작했던 RPM 트랜잭션을 전송할 (RPM 드라이버가 코어 1에 의해 시작된 이전의 트랜잭션을 그만둘) 수도 있다. 블록 730에서, 양쪽 모두의 코어들은 슬립 모드에 있을 수도 있으며, 슬립 세트는 전송될 수도 있고, 글로벌 LPRM d는 진입될 수도 있다.

[0118] 다양한 양태들에서, 슬립 세트 트랜잭션을 전송하기 전에, 이전의 트랜잭션은 이전의 트랜잭션로부터의 자원들이 그것들의 슬립 세트에 바람직하지 않게 입력되지 않도록 무효화될 수도 있다. 멀티코어 환경에서, 이미 비행 중인 (in flight) 액티브 메시지가 있으면, 슬립 세트는 항상 전송되지 않을 수도 있다. 그러나, SPM들은 그것들의 셧다운 요청들을 RPM로 여전히 전송할 수도 있으며, 이에 RPM은 마스터를 그것의 슬립 세트로 전이시킬 것이고, 디폴트로 이전의 슬립 세트를 사용할 수도 있으며, 이는 바람직하지 않을 수도 있다. 이를 해결하기 위해, 일 양태에서, RPM은 "자동 무효화" 특징을 지원할 수도 있어서, 자원이 자동 무효라고 마킹되면, 마스터가 슬립으로부터 기상하는 경우에 RPM은 자원의 슬립 세트를 무효화할 것이다. 이는 모든 슬립 세트 트랜잭션 전에 명시적 무효화를 마스터가 전송하는 것을 방지한다. 액티브 및 슬립 콘텍스트들 사이의 RPM 통신들을 조정하기 위해, 이전의 슬립 세트는 이미 자동 무효화되어 있을 수도 있으며, 이에 어떤 슬립 세트도 이 경우에는 진입되지 않을 것이며, 이는 소망의 거동이다. RPM은 자원마다 기반, 뿐만 아니라 글로벌 기반 양쪽 모두로 자동 무효화를 지원 (즉, 항상 모든 자원들 등을 무효화) 할 수도 있다. 슬립 세트에 추가되어 있는 자원들의 성질에 기초하여 어떤 것이 바람직한 것인지를 결정하는 것은 호출하는 코드에 달려있을 수도 있다.

[0119] 다양한 양태들에서, 멀티코어 프로세서 시스템은 실효된 (stale) 슬립 세트들을 무효화하도록 구성될 수도 있다. 일 양태에서, 무효화 자원은 슬립 세트 트랜잭션을 전송하기 전에 이전의 트랜잭션을 무효화하는데 사용될 수도 있다. 슬립 세트 무효화 요청은 임의의 슬립 세트 트랜잭션에서 제 1 메시지로서 전송될 수도 있고, RPM 드라이버는 무효화 요청이 트랜잭션에서 존재한다면 먼저 실행될 것임을 보장하도록 구성될 수도 있다. 슬립 모드에 진입하는 동안 (예컨대, 슬립 세트 트랜잭션을 전송하는 동안) 에 무효화 요청을 전송하는 것은, 그렇지 않고 슬립 모드들을 종료하는 동안 무효화 요청이 별도의 메시지로서 전송되었다면 발생하는 잠재적인 경합 (race) 조건들을 제거한다.

[0120] 다양한 양태들에서, 멀티코어 프로세서 시스템은 공유된 저전력 자원들 모드들 (LPRMs) 을 인에이블 및/또는 디스에이블하도록 구성될 수도 있다. 공유된 자원에 대한 저전력 모드들이 인에이블되는 경우, 그 자원을 공유하는 코어들은 낮은 전력 모드에 진입할 수 있고, 그것들의 슬립 세트들을 재평가하도록 기상될 수도 있다.
공유된 저전력 모드가 인에이블되거나 또는 디스에이블될 때마다, 슬립 서브시스템은 변화들을 반영하게 각각의 코어에 대한 인에이블된 목록을 수정할 수도 있다.

[0121] 다양한 양태들에서, 멀티코어 프로세서 시스템은 SPM 핸드셰이킹 동작들을 수행하도록 구성될 수도 있다. 위에서 언급했듯이, 정지할 마지막 코어는 일반적으로 RPM 슬립 세트를 전송하는 것이다. 그러나, 앱 슬립 세트를 지키기 전에 각각의 코어의 SPM로부터의 핸드셰이크들을 RPM이 기다릴 수도 있으므로, 정지할 마지막 코어가 아닌 코어들은, 공유된 저전력 모드들이 진입을 위해 선택되면, RPM과의 핸드셰이크를 위해 그 코어들의 SPM 하드웨어를 프로그래밍하는 것을 여전히 필요할 수도 있다.

[0122] 다양한 양태들에서, 핸드셰이크 LPRM (예컨대, spm.handshake LPRM) 은 각각의 코어에 대해 존재할 수도 있다.
코어가 정지할 마지막 코어인지의 여부에 상관없이 각각의 코어가 로컬 및 글로벌 모드들을 구할 수 있으므로, 정지할 마지막 코어가 아닌 코어들은 여전히 핸드셰이크 LPRM에 진입할 수도 있다. RPM을 요청하는 글로벌 모드에 진입하는 것이 허용가능하다고 코어들이 결정하면, 그 코어들은 RPM과의 핸드셰이크를 위해 SPM을

프로그래밍할 수도 있다.

[0123] 다양한 양태들에서, 멀티코어 프로세서 시스템은 RPM-타이머식 (timed) 트리거를 전송하도록 구성될 수도 있다. 위에서 언급했듯이, 슬립 상태가 되는 (go sleep) 각각의 코어는 그것 소유의 타이머 하드웨어 및 그것 소유의 하드 기상 시간을 가질 수도 있다. 코어가 유휴로 갈 때마다, 그것은 선택된 LPRM들에 기초하여 "백오프 시간 (backoff time)"을 계산하고, 계산된 백오프 시간을 로컬 하드웨어에 프로그래밍할 수도 있다. 마지막 코어가 슬립으로 가면, 타이머식 트리거는 코어들 중 임의의 것에 대한 다음의 스케줄링된 기상 시간을 나타내기 위해 RPM으로 전송될 수도 있다. 타이머식 트리거는 프로세서가 기상을 예상하고 있는 때에 대한 지식을 RPM이 가지는 것을 허용하여서, 어떤 저전력 모드들에 진입되는지에 관한 결정들에 대해 RPM이 알 수 있게 된다. 타이머식 트리거는 또한 RPM이 시스템 저전력 모드들 동안에 타이머를 설정하는 것을 허용하여서, 프로세서를 정시에 기상시킬 수 있는데, 타이머 하드웨어는 시스템 저전력 모드들 동안에 기능하는 것이 보장되지 않을 수도 있어서이다.

[0124] 정지할 마지막 코어가 다음 코어가 기상할 시간을 알기 위하여, 다양한 양태들은 (백오프를 뺀 후에) 코어의 절대 기상 시간을 정의하는 각각의 코어에 대한 변수를 저장할 수도 있다. LPRM (예컨대, 핸드셰이크 LPRM 등) 은 코어에 대한 기상 시간을 계산하고 그것을 글로벌 변수로서 저장할 수도 있다. LPRM은 또한 다음의 기상까지의 최소 시간을 계산하고 그것을 타이머식 트리거로서 RPM에 전송할 수도 있다.

[0125] 올바른 타이머식 트리거를 RPM으로 전송하는 것에 더하여, 정지할 마지막 코어는 또한 RPM 기상 인터럽트가 제일 빠른 기한을 갖는 코어에 라우팅되는 것을 보장하기 위해 필요할 수도 있어서, 코어는 기상하여 그것의 타이머를 서비스할 수 있다. 제일 빠른 기한을 갖는 코어는 시스템 저전력 모드들로부터 맨 먼저 기상하도록 구성될 수도 있어서, 그것의 로컬 타이머 인터럽트를 발사하도록 트리거할 수 있다.

[0126] 각각의 코어에 대한 절대 기상 시간은 공유된 로케이션에 저장될 수도 있다. 각각의 코어는 다음의 코어의 기상까지의 최소 지속기간을 저전력 모드들을 선택하는 경우의 소프트 기상 힌트로서 이용할 수도 있다.

[0127] 다양한 양태들에서, 멀티코어 프로세서 시스템은 코어들 간에 RPM 통신들을 조정하도록 구성될 수도 있다. 마스터 프로세서 및 RPM 프로세서 사이의 통신에 대해, RPM으로부터 마스터 프로세서로의 인터럽트들은 코어 단위 기반으로는 되지 않을 수도 있다. 그러므로, RPM 인터럽트는 임의의 특정한 시간에 임의의 코어로 라우팅될 수도 있다. 이 인터럽트는 프로세서로부터 수신된 메시지들에 확인응답하기 위해, 그리고 또 통지 인터럽트들을 프로세서로 전송하기 위해 RPM에 의해 사용될 수도 있다.

[0128] 각각의 코어는 그것이 액티브인 임의의 시간에 RPM 메시지를 전송할 수도 있다. 액티브 모드들에서, 코어는 한 번에 하나의 코어만이 메시지를 전송하는 것을 보장하기 위해 RPM 드라이버에서 운영 체제 잠금 (즉, 뮤텍스 (mutex)) 을 획득할 수도 있다. 그러나 슬립 동안에, 슬립 세트를 전송하는 경우, 프로세서의 RPM 드라이버는 RPM으로부터 돌아오는 확인응답 인터럽트에 대해 폴링할 수도 있는데, 슬립 코드가 INTLOCK된 콘텍스트에서 실행될 수도 있어서이다.

[0129] 다른 코어가 RPM으로부터의 확인응답 인터럽트를 기다리는 동안 하나의 코어가 슬립 세트를 전송하려고 시도하는 것을 방지하기 위하여, RPM 드라이버가 슬립 세트 트랜잭션 요청을 수신하는 경우, 그것은 운영 체제 잠금이 이미 홀딩되어 있는지를 먼저 체크할 수도 있다. 그렇다면, RPM 드라이버는 슬립 세트를 전송하지 않는 대신, 그 요청을 무시하고 그것을 요구했던 코어를 인터럽트할 수도 있다. 인터럽트의 존재는 그 코어가 슬립을 종료하게 할 수도 있다. 슬립으로의 다음의 진입시, 슬립 세트는 재평가될 수도 있고, 슬립 코드는 적용가능하면 슬립 세트를 전송하려고 다시 시도할 수도 있다.

[0130] RPM 드라이버가 RPM 인터럽트에 대해 폴링하고 있는 경우, 그 드라이브는 그 인터럽트가 폴링하고 있는 코어에 라우팅되는 것을 보장하기 위해 인터럽트 제어기 드라이버와 인터페이싱할 수도 있다.

[0131] 하이 레벨 운영 체제 (HLOS) 에 의존하여, 슬립 코드는 액티브 모드들 동안에 RPM 트랜잭션들을 전송하고 있는 스레드들과는 상이한 프로세스에서 실행할 수도 있다. 그런 경우들에서, 슬립 트랜잭션은 메시지들을 전송하고 있는 다른 프로세스들 사이에 동기화를 보장하기 위해 스핀락을 사용할 수도 있다.

[0132] 도 8은 다른 코어가 확인응답 인터럽트를 기다리는 동안 하나의 코어가 슬립 세트를 전송하는 것이 방지되도록 코어들 간에 RPM 통신들을 조정하는 일 양태의 방법 (800) 을 도시한다. 도 8의 도시된 예에서, LPRM들 c 및 d는 글로벌 (예컨대, 코어 0 및 코어 1에 의해 공유됨) 이고, rpm.start 함수, rpm.send 함수 및 LPRM들 c 및 d에 연관된 메모리 로케이션들은 각각 1의 값 (LPRM c = 1, LPRM d = 1, rpm.start = 1, rpm.send = 1) 을

저장한다.

[0133] 블록 802에서, 코어 1은 슬립 상태에 있고 글로벌 모드들 c 및 d에 진입되었다. 블록 804에서, 코어 1은 실행 상태이다. 블록 806에서, 코어 1은 유휴 상태로 되며, 슬립 모드들을 구하고 (예컨대, "솔버" 프로세스를 실행하고) 진입을 위해 글로벌 모드들을 선택할 수도 있다. 블록 808에서, 코어 1은 RPM 트랜잭션의 시작을 개시할 수도 있다.

[0134] 블록 810에서, 기상 이벤트가 코어 0 상에서 검출될 (예컨대, 인터럽트가 코어 0에 의해 수신될) 수도 있다. 블록 812에서, 멀티코어 프로세서는 슬립 모드를 종료할 수도 있다.

[0135] 블록 814에서, 코어 1은 슬립 모드에 진입하고 RPM 트랜잭션을 만들 수도 있다.

[0136] 블록 816에서, 코어 0은 RPM 메시지를 전송하는 스레드를 실행할 수도 있다. 또한 블록 816에서, RPM 드라이버는 운영 체제 잠금을 얻을 수도 있다. 운영 체제가 코어 1에 의해 잠겨있지 않으면 스레드는 실행되지 않을 것이라는 것에 주의해야 한다. 위에서 논의된 바와 같이, 각각의 코어는 그것이 액티브인 동안 임의의 시간에 RPM 메시지를 전송할 수도 있고, 코어는 한 번에 하나의 코어만이 메시지를 전송하는 것을 보장하기 위해 RPM 드라이버에서 운영 체제 잠금을 획득할 수도 있다.

[0137] 블록 818에서, 코어 0은 메시지가 전송되는 동안에 운영 체제 잠금을 유지할 수도 있다.

[0138] 블록 820에서, 코어 1은 슬립 세트를 전송할 수도 있다. 위에서 논의된 바와 같이, 슬립 세트를 전송하는 경우, 프로세서의 RPM 드라이버는 RPM으로부터 돌아오는 확인응답 인터럽트에 대해 폴링할 수도 있는데, 슬립 코드가 INTLOCK된 콘텍스트에서 실행될 수도 있어서이다.

[0139] 또한 블록 820에서, RPM은 OS 잠금이 홀딩되는지를 알기 위해 체크하고, 그렇다고 결정할 수도 있다. 위에서 언급했듯이, 다른 코어가 RPM으로부터의 확인응답 인터럽트를 기다리는 동안 하나의 코어가 슬립 세트를 전송하려고 시도하는 것을 방지하기 위하여, RPM 드라이버가 슬립 세트 트랜잭션 요청을 수신하는 경우, 그것은 운영 체제 잠금이 이미 홀딩되어 있는지를 먼저 체크할 수도 있다. 그렇다면, RPM 드라이버는 슬립 세트를 전송하지 않고, 대신에, 그 요청을 무시하고 그것을 요구했던 코어를 인터럽트할 수도 있다. 인터럽트의 존재는 그 코어가 슬립을 종료하게 할 수도 있고, 슬립으로의 다음의 진입시, 슬립 세트는 재평가될 수도 있고, 슬립 코드는 적용가능하면 슬립 세트를 전송하려고 다시 시도할 수도 있다.

[0140] 블록 822에서, 코어 0은 확인응답 인터럽트를 수신하고 운영 체제 잠금을 해제할 수도 있다.

[0141] 블록 824에서, RPM 드라이버는 RPM 트랜잭션을 무시하고 인터럽트를 코어 1로 전송할 수도 있다.

[0142] 블록 826에서, 코어 1은 정지하려고 시도할 수도 있지만, 계류 중인 (pending) 인터럽트에 대해 즉각 기상할 수도 있다.

[0143] 따라서, 블록들 812, 816, 818 및 822에서, 코어 0은 깨어 있는 상태로 진입하고 코어 1이 그것의 슬립 세트를 전송했던 동일한 기간 동안 RPM 메시지를 전송한다.

[0144] 위에서 언급했듯이, RPM으로부터의 인터럽트들은 임의의 특정 시간에 임의의 코어로 라우팅될 수도 있으며, 그것들은 프로세서로부터 수신된 메시지들을 확인응답하고 통지 인터럽트들을 전송하기 위해 RPM에 의해 사용될 수도 있다.

[0145] 도 9는 슬립 동안에 RPM 드라이버에서 공유된 스핀락을 체크하는 일 양태의 방법 (900) 을 도시한다.. 블록 902에서, 코어 0은 실행 상태이다. 블록 904에서, 코어 1은 실행 상태이다. 블록 906에서, 코어 1은 유휴 상태로 되며, 슬립 모드들을 구하고 (예컨대, "솔버" 프로세스를 실행하고) 진입을 위해 글로벌 모드들을 선택할 수도 있다. 블록 908에서, 코어 0은 스레드를 실행하며, RPM 메시지를 전송할 수도 있다. 또한 블록 908에서, RPM 드라이버는 액티브 콘텍스트에서 공유된 스핀락을 얻을 수도 있다. 블록 910에서, 코어 1은 슬립 모드들에 진입하고 RPM 핸드셰이크 모드들에 대한 적절한 참조 카운트들을 점감할 수도 있다.

[0146] 블록 912에서, 코어 0은 RPM 확인응답 인터럽트를 기다리는 동안에 유휴로 갈 수도 있다. 블록 914에서, 코어 0은 슬립 모드들을 구하고 진입하는 글로벌 모드를 선택할 수도 있다. 블록 916에서, 코어 0은 RPM 트랜잭션을 시작할 수도 있다. 블록 918에서, 코어 0은 슬립 모드들에 진입하고 RPM 트랜잭션을 만들 수도 있다. 블록 920에서, 코어 0은 슬립 세트를 전송할 수도 있다. 또한 블록 920에서, RPM 슬립 드라이버는 공유된 스핀락이 홀딩되는지를 알기 위해 체크할 수도 있다. 도 9의 도시된 예에서, RPM 슬립 드라이버는 스핀락이 현재 홀딩되어 있음을 검출하고, 블록 922에서, 트랜잭션을 무시하고 슬립 세트는 전송되지

않는다. 일 양태에서 그 시스템은, 비록 양쪽 모두의 SPM들이 RPM과의 핸드셰이크를 수행하더라도, 어떤 슬립 세트도 이전에 언급된 자동 무효화 특징에 따라 RPM에 적용되지 않도록 구성될 수도 있다.

[0147] 블록 924에서, 코어 0은 저전력 모드에 진입할 수도 있다. 블록 926에서, 모든 코어들이 슬립 상태에 있고 어떤 슬립 세트들도 전송되지 않는다. 블록 928에서, 코어 0은 RPM 확인응답 트랜잭션을 수신하고 기상할 수도 있다. 블록 930에서, 코어 0은 확인응답을 수신하고 스핀락을 해제할 수도 있다.

[0148] 다양한 양태들은 다수의 유용한 이점들을 제공한다. 다양한 양태들은 SOC들 및 멀티코어 프로세서들 상에서 저전력 모드들의 선택 및 구현을 개선시킨다. 예를 들어, 다양한 양태는 다른 코어들을 모니터링하는 제로 코어를 필요로 하지 않으며, (다른 운영 체제들에 의해 요구된 바와 같이) 코어들을 모니터링하기 위해 슬립 프로세스 위에 상주하는 프로세스를 필요로 하지 않고, (다른 운영 체제들에 의해 요구된 바와 같이) 모든 글로벌 자원들이 유휴가 되기까지 코어들이 차단하고 대기하는 것을 필요로 하지 않는다. 다양한 양태들은 또한, 모든 코어들/프로세싱 유닛들이 대칭 방식으로 처리될 것을 허용하는 방식으로, 그리고 각각의 코어가 그것의 동작 상태를 다른 코어들과는 독립적으로 선택할 수도 있도록 SOC들 및 멀티코어 프로세서들 상에서 다이나믹 저전력 자원 모드들을 구현한다.

[0149] 전력 함수가, 만약 있다면, 어떤 자원들이 저전력 자원 모드에 배치되어야 하는지를 결정하는데 이용될 수도 있다. 그 전력 함수는 전력 절약량과 자원들을 다운시키고 동작으로 되돌리는 에너지 비용의 합의 함수일 수도 있다. 다양한 양태들은 구현된 저전력 자원 모드들의 다양한 조합들에 의해 제공된 잠재적인 시스템 저전력 구성들의 각각에 연관된 순수 전력 절약량을 결정하는데 전력 함수를 사용할 수도 있다. 다양한 양태들은 다양한 자원들을 저전력 자원 모드에 배치하고 그것들을 동작 모드로 되돌리는데 요구된 작업량에 의해 오프셋된, 계산된 시간 프레임에 걸쳐 절약된 전력량으로서 순수 전력 절약량을 계산할 수도 있다. 다양한 양태들에서, 순수 전력 절약량은 예상되는 유휴 시간 X와 기울기 M 및 오프셋 B를 갖는 간단한 선형 다항식 모듈을 이용한 함수에 의해 계산될 수도 있으며, 컴퓨터 전력 절약량은 MX+B이다.

[0150] 다양한 양태들은 구성가능한 시스템 파라미터들에 저장된 값들에 기초하여 순수 전력 절약량을 주기적으로 계산할 수도 있다.

[0151] 다양한 양태들에서, 다수의 메커니즘들이 현재의 운영 상태 및 조건들에 기초하여 저전력 자원 모드들의 최적의 세트를 식별하기 위해 저전력 자원 모드 선택 솔버에서 사용될 수도 있다. 배낭 문제 (knapsack problem)에 대한 다양한 알려진 알고리즘 또는 휴리스틱스 솔루션들이 저전력 자원 모드 선택 솔버에서 구현될 수도 있다. 이러한 방법들은 if/then/else 로직 트리 알고리즘들, 테이블 룩업 알고리즘, 및 상이한 자원들의 대안적 저전력 자원 모드들의 대안적 순열들 및 조합들을 통해 체계적으로 작동하는 비교 방법들을 수반할 수도 있다.

[0152] 다양한 양태들에 따른 다이나믹 저전력 자원 모드들의 구현예는 또한 단순히 부가적인 배터리 전력을 절약하는 것을 넘어서는 다수의 유용한 이점들을 제공한다. 하나의 이점으로서, 다양한 자원들의 저전력 자원 모드들 및 그것들의 정의들은 그 모드들을 구현하는 코드와는 대부분 독립적이다. 관련 드라이버 호출들은 "진입" 및 "종료" 함수들 내에 포함될 수도 있으며, 이에 저전력 자원 모드는 영향을 받지 않는다. 솔버 프로세스는 하드 데이터, 이를테면 각각의 저전력 자원 모드에 대한 전력 및 레이턴시 데이터, 그리고 현재의 동작 상태에서 존재하는 다이나믹 제한들 (예컨대, 레이턴시 요건들) 및 동작 조건들에 기초하여 저전력 자원 모드가 진입되고 종료되어야 하는 때를 결정하는 것을 처리할 수도 있다. 따라서, 개발자는 적합한 저전력 자원 모드들을 디바이스로 하드코딩하기 위하여 이러한 파라미터들의 조합들을 예측하기 위해 시도하지 않아도 된다.

[0153] 위에서 설명된 양태들의 다른 이점은, 컴퓨팅 디바이스가, 시스템 모드들의 미리 구성된 또는 미리 정의된 세트로부터 단일의 정의된 저전력 자원 모드를 선택하는 데 필요하지 않고, 현재의 동작 상태, 자원들, 동작 조건, 추정된 슬립 사이클, 디바이스 구성 등에 기초하여 시스템에 대한 저전력 구성을 동적으로 구현하는 저전력 자원 모드들의 조합을 선택할 수 있다는 것이다. 이는, 종래의 알려진 전력 관리 시스템들이 미리 정의된 저전력 구성들의 세트 간에 (예컨대, 모드 1, 모드 2, 또는 모드 3 중 하나를) 선택하도록 구성되는 반면, 위에서 설명된 양태들은 디바이스가 저전력 모드에 진입하기 위해 이용가능한 각각의 자원에 대해 하나 이상의 저전력 자원 모드들을 동적으로 선택하는 것을 가능하게 하여, 현재의 조건들 및 상태들에 가장 적합한 시스템 저전력 구성들을 구현하는 훨씬 더 큰 유연성을 제공하기 때문에 유익하다. 예를 들어, 특정 프로세서가 각각 저전력 자원 모드들 (A', B', 및 C') 를 갖는 3 개의 자원들 (A, B, 및 C) 을 가진다고 가정한다. 저전력 자원 모드들은 상이한 레이턴시들을 가질 수도 있으며, 이를테면 저전력 자원 모드 A'는 .4 ms의 레이턴시를 가질 수도 있으며, 저전력 자원 모드 B'는 .5 ms의 레이턴시를 가질 수도 있고, 저전력 자원 모드 C'은 .6 ms의 레이턴

시를 가질 수도 있다. 예를 들어, 클라이언트가 시스템에 1 밀리초 레이턴시 요건을 배치하면 (예컨대, 1ms가 최악의 허용가능 레이턴시가면), 자원들 (A, B, 및 C) 의 저전력 모드들은 선택된 저전력 모드들의 조합이 최악의 레이턴시 요건을 충족하는 한 서로 독립적으로 인에이블되거나 또는 디스에이블될 수 있다. 예를 들어, 프로세서가 유휴 상태로 되고 자원들 (A, B, 및 C) 이 모드 인에이블되면, 시스템은 저전력 자원 모드 A' (.4 ms의 레이턴시), 저전력 자원 모드 B' (.5 ms의 레이턴시), 저전력 자원 모드 C' (.6 ms의 레이턴시), 모드들 A 및 B (.9 ms의 레이턴시), 또는 모드들 A 및 C (1 ms의 레이턴시) 를 고를 수 있다. 따라서, 다양한 양태들에서, 솔버 태스크는 1 ms의 최악의 레이턴시 허용오차가 주어지면 최고의 전력을 절약하는 저전력 자원 모드들의 최상의 세트를 고를 수도 있다.

[0154] 덧붙여, 전형적인 전력 관리 시스템들은, 클라이언트들이 인액티브 모드 및 액티브 모드를 가지고 레이턴시 허용오차가 현재의 성능 상태에 의존하는 것을 요구한다. 위에서 설명된 다양한 양태들에서, 클라이언트 메커니즘들은 "액티브" 또는 "인액티브"보다는 "존재할" 또는 "존재하지 않을" 수도 있다. 다시 말하면, 다양한 양태들에서, 다양한 저전력 자원 모드들은, 동작 상태들 (예컨대, 액티브 또는 인액티브) 에 기초하여 선택되기보다는 가능한 상태들을 제거하기 위해 상세히 검토될 수도 있다. 게다가, 다양한 양태들은 클라이언트들이 다양한 자원들에 대한 저전력 자원 모드들을 생성, 등록, 및/또는 무시하는 것과, 다수의 가능한 시스템 저전력 구성들을 가능하게 하기 위해 저전력 자원 모드들의 조합을 동적으로 선택하는 것을 가능하게 한다. 이는 시스템 클라이언트들이 디바이스의 저전력 상태들을 추가로 제어하고 미세 튜닝하는 것을 허용한다.

[0155] 위에서 설명된 양태들의 다른 이점은 컴퓨팅 디바이스 프로세서가 시스템 클라이언트들의 다양한 동작 모드들을 알아차릴 필요가 없다는 것이다. 다양한 양태들에서, 클라이언트들은 그것들의 레이턴시 허용오차만을 직접 제출할 수도 있다. 이처럼, 프로세서는 각각의 클라이언트의 동작 상태들에 연관된 다양한 세부사항들에 관해 알 필요가 없다. 프로세서는 클라이언트들의 등록된 레이턴시 허용오차들을 알고 보고된 레이턴시 허용오차들에 기초하여 저전력 자원 모드에 진입하기 위해 저전력 자원들을 선택하는 것만 필요하다. 다양한 양태들에서, 허용오차들 및 저전력 모드들의 설정은 별개의 엔티티들에 의할 수도 있다. 예를 들어, USB 클라이언트는 레이턴시 허용오차를 설정하지만 반드시 저전력 모드를 설정하지 않을 수도 있다. 각각의 저전력 자원 모드는 레이턴시 고려와는 완전히 독립적인 임의의 주어진 슬립 사이클로 진입될 수 있는지의 여부를 나타내기 위해 시그널링 메커니즘들의 세트를 가질 수도 있다.

[0156] 추가의 이점으로서, 클라이언트들이 얼마나 오래 잠들어 있을 것으로 예상되는지를 그것들이 특정하는 것을 가능하게 하기 위해 새로운 NPA 프로그래밍 노드가 일 양태에서 제공될 수도 있다. 예를 들어, NPA 프로그래밍 노드 "/코어/프로세서/기상"은 클라이언트들이 "X" 마이크로초 보단 길지 않은 동안 (71 시간까지) 잠들어 있을 (즉, 프로세서 또는 자원들을 이용하지 않을) 것으로 예상된다는 것을 클라이언트들이 특정하는 것을 가능하게 하기 위해 제공될 수도 있다. 이러한 프로그래밍 능력은 프로세서 유휴 상태들 및 저전력 구성들과의 호환성을 위한 클라이언트 애플리케이션들의 개발을 단순화할 수도 있다.

[0157] 추가의 양태에서, 솔버 계산들의 결과들은 동일한 또는 유사한 동작 조건들 (예컨대, 동작 상태들, 온도, 및 레이턴시 제한들) 이 유휴 상태에 진입될 수도 있는 시간에 존재하는 경우에 솔버 알고리즘을 재수행하는 일 없이 최적의 저전력 구성이 재사용될 수도 있도록 메모리 내에 캐싱될 수도 있다. 이런 식으로, 프로세서는 최적의 또는 거의 최적의 전력 절약을 여전히 달성하면서도 솔버 알고리즘을 수행하는 프로세스를 건너뛰는 것에 의해 유휴 상태에 빠르게 진입할 수 있다. 추가의 양태에서, 동작 상태 및 조건들은 캐싱된 최적의 저전력 구성들이 조건들 및 상태들의 통계적으로 결정된 범위들에 링크될 수도 있도록 통계적으로 분석될 수도 있다.

[0158] 다양한 양태들과 함께 사용하기에 적합한 전형적인 모바일 디바이스들 (1000) 이 도 10에 도시된 구성요소들을 공통으로 가질 것이다. 예를 들어, 예시적인 모바일 수신기 디바이스 (1000) 는, 디스플레이 (1053), 스피커 (1059) 및 내부 메모리 (1052) 에 연결된 연결된 프로세서 (1051) 를 구비할 수도 있다. 덧붙여, 모바일 디바이스 (1000) 는 프로세서 (1051) 에 연결된 모바일 멀타미디어 수신기 (1056) 에 접속되는 전자기 방사를 전송하고 수신하기 위한 안테나 (1054) 를 가질 수도 있다. 일부 양태들에서, 모바일 멀티미디어 수신기 (1056) 는 내부 프로세서 (1058), 이를테면 수신기 (1056) 의 동작들을 제어하고 디바이스 프로세서 (1051) 와 통신하는 디지털 신호 프로세서 (DSP) 를 구비할 수도 있다. 또한 통상적으로, 모바일 디바이스들은 보통 사용자 입력들을 수신하기 위한 키 패드 (1056) 또는 미니어처 키보드 및 메뉴 선택 버튼들 또는 로커 (rocker) 스위치들 (1057) 을 구비한다.

[0159] 프로세서 (1051) 는 본원에서 설명된 다양한 양태들의 기능들을 포함한 다양한 기능들을 수행하기 위해 프로세서 실행가능 소프트웨어 명령들 (애플리케이션들) 에 의해 구성될 수 있는 임의의 프로그래밍가능 마이크로프로

세서, 마이크로컴퓨터 또는 다중 프로세서 칩 또는 칩들일 수도 있다. 또한, 다양한 양태들의 기능들은 DSP-실행가능 명령들로 구성된 수신기 (1056) 내의 DSP 프로세서 (1058) 로 구현될 수도 있다. 전형적으로, 소프트웨어 애플리케이션들과 프로세서 실행가능 명령들은 그것들이 액세스되어 프로세서 (1051) 에 로딩되기 전에 내부 메모리 (1052) 에 저장될 수도 있다. 일부 모바일 디바이스들에서, 프로세서 (1051) 는 애플리케이션 소프트웨어 명령들을 저장하기에 충분한 내부 메모리를 구비할 수도 있다. 일부 모바일 디바이스들에서, 보안 메모리는 프로세서 (1051) 에 결합된 개별 메모리 칩 내에 있을 수도 있다. 많은 모바일 디바이스들 (1050) 에서, 내부 메모리 (1052) 는 휘발성 또는 비휘발성 메모리, 이를테면 플래시 메모리, 또는 이것 양쪽 모두의 혼합체일 수도 있다. 이 설명을 위해, 메모리에 대한 종합적인 언급은 내부 메모리 (1052), 모바일 디바이스에 꽂아지는 착탈식 메모리, 및 프로세서 (1051) 자체 내의 메모리를 포함하여, 프로세서 (1051) 에 의해 액세스가능한 모든 메모리를 말한다.

[0160] 앞서의 방법 설명들 및 프로세스 흐름도들은 예시적인 예들로서만 제공되고 갖가지 양태들의 단계들이 제시된 순서로 반드시 수행되어야 함을 요구하거나 의미하도록 의도된 것은 아니다. 이 기술분야의 숙련된 자에 의해 이해될 바와 같이 앞서의 양태들에서의 단계들의 순서는 어떤 순서로도 수행될 수도 있다. 단어들 이를테면 "그 후", "그 다음", "다음에" 등은 단계들의 순서를 제한하려는 의도는 아니고; 이들 단어들은 단지 방법들의 설명을 통하여 독자를 인도하는 데 사용된다. 게다가, 단수형으로, 예를 들어, 관사 "a", "an" 또는 "the"의 사용에 해당하는 것으로 여겨질 수 있는 청구항 요소들에 대한 어떤 언급이라도, 그 요소를 단수형으로 제한하는 것으로 해석되지는 않는다.

[0161] 본원에서 개시된 양태들에 관련하여 설명되는 각종 구체적인 논리 블록들, 모듈들, 회로들, 및 알고리즘 단계들은 전자적 하드웨어, 컴퓨터 소프트웨어, 또는 이것 둘의 조합들로 구현될 수도 있다. 하드웨어 및 소프트웨어의 이러한 상호교환가능성을 명백하게 예증하기 위하여, 다양한 예시적인 컴포넌트들, 블록들, 모듈들, 회로들, 및 단계들이 대체로 그것들의 기능성의 측면에서 설명되어 있다. 이러한 기능성이 하드웨어 또는 소프트웨어 중 어느 것으로 구현되는지는 전체 시스템에 부과되는 설계 제약들 및 특정 애플리케이션에 달려있다.

당업자들은 설명된 기능을 각각의 특정한 애플리케이션에 대하여 다양한 방식으로 구현할 수도 있지만, 이러한 구현 결정은 본 발명의 범위를 벗어나도록 하는 것으로 해석되지 않아야 한다.

[0162] 본원에 개시된 양태들에 관련하여 설명된 다양한 실례의 로직들, 논리 블록들, 모듈들, 및 회로들을 구현하는 데 사용되는 하드웨어는, 범용 프로세서, 디지털 신호 프로세서 (DSP), 멀티미디어 브로드캐스트 수신기 칩 내의 DSP, 주문형 집적회로 (ASIC), 필드 프로그램가능 게이트 어레이 (FPGA) 또는 다른 프로그램가능 로직 디바이스, 개별 게이트 또는 트랜지스터 로직, 개별 하드웨어 컴포넌트들, 또는 본원에서 설명된 기능들을 수행하도록 고안된 그것들의 임의의 조합으로 구현되거나 수행될 수도 있다. 범용 프로세서는 마이크로프로세서일 수도 있지만, 대체예에서, 그 프로세서는 기존의 임의의 프로세서, 제어기, 마이크로제어기, 또는 상태 머신 (state machine) 일 수도 있다. 프로세서는 또한, 컴퓨팅 디바이스들의 조합, 예컨대, DSP 및 마이크로프로세서의 조합, 복수의 마이크로프로세서들의 조합, DSP 코어와 협력하는 하나 이상의 마이크로프로세서들의 조합, 또는 임의의 다른 이러한 구성으로 구현될 수도 있다. 다르게는, 일부 단계들 또는 방법들이 주어진 기능에 특정된 회로에 의해 수행될 수도 있다.

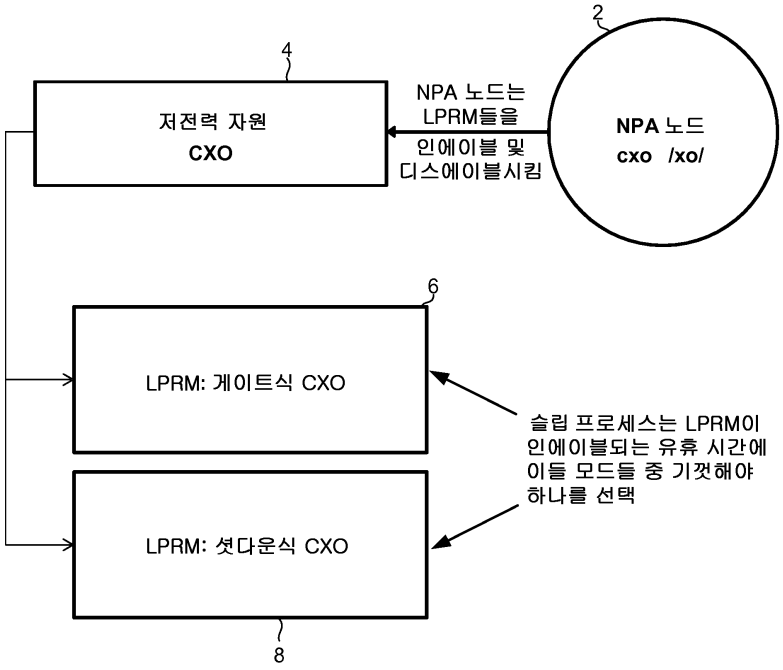
[0163] 하나 이상의 예시적인 양태들에서, 설명된 기능들은 하드웨어, 소프트웨어, 펌웨어, 또는 그것들의 임의의 조합으로 구현될 수 있다. 소프트웨어로 구현된다면, 기능들은 하나 이상의 명령들 또는 코드로서 컴퓨터 판독가능 매체 상에 저장되거나 전송될 수도 있다. 본원에 개시된 방법 또는 알고리즘의 단계들은 컴퓨터 판독가능 매체 상에 상주할 수도 있는 프로세서 실행가능 소프트웨어 모듈로 실시될 수도 있다. 컴퓨터 판독가능 매체들은 한 장소에서 다른 장소로의 컴퓨터 프로그램의 전송을 용이하게 하는 임의의 매체를 포함하는 컴퓨터 저장 매체 및 통신 매체 양쪽 모두를 포함한다. 저장 매체들은 컴퓨터에 의해 액세스될 수도 있는 임의의 이용가능한 매체들일 수도 있다. 비제한적인 예로, 이러한 컴퓨터 판독가능 매체들은 RAM, ROM, EEPROM, CD-ROM 또는 다른 광 디스크 스토리지, 자기 디스크 스토리지, 또는 다른 자기적 저장 디바이스들, 또는 소망의 프로그램 코드를 컴퓨터에 의해 액세스될 수도 있는 명령들 또는 데이터 구조들의 형태로 운반하거나 저장하는 데 사용될 수도 있는 임의의 다른 매체를 포함할 수도 있다. 또한, 임의의 접속이 컴퓨터 판독가능 매체로 적절히 칭해진다. 예를 들어, 소프트웨어가 웹사이트, 서버, 또는 다른 원격 자원으로부터 동축 케이블, 광섬유 케이블, 연선 (twisted pair), 디지털 가입자 회선 (DSL), 또는 무선 기술들 이를테면 적외선, 라디오, 및/또는 마이크로파를 이용하여 송신된다면, 동축 케이블, 광섬유 케이블, 연선, DSL, 또는 적외선, 라디오, 및 마이크로파와 같은 무선 기술들은 매체의 정의에 포함된다. 디스크 (disk 및 disc) 는 본원에서 사용되는 바와 같이, 콤팩트 디스크 (compact disc, CD), 레이저 디스크, 광 디스크, 디지털 다용도 디스크 (DVD), 플로

피 디스크 (floppy disk) 및 블루레이 디스크를 포함하는데, disk들은 보통 데이터를 자기적으로 재생하지만, disc들은 레이저들로써 광적으로 데이터를 재생한다. 상기한 것들의 조합들은 또한 컴퓨터 판독가능 매체들의 범위 내에 포함되어야 한다. 덧붙여, 방법 또는 알고리즘의 동작들은 코드들 및/또는 명령어들 중의 하나 또는 임의의 조합 또는 세트로서 컴퓨터 프로그램 제품에 통합될 수도 있는 기계 판독가능 매체 및/또는 컴퓨터-판독가능 매체 상에 존재할 수도 있다.

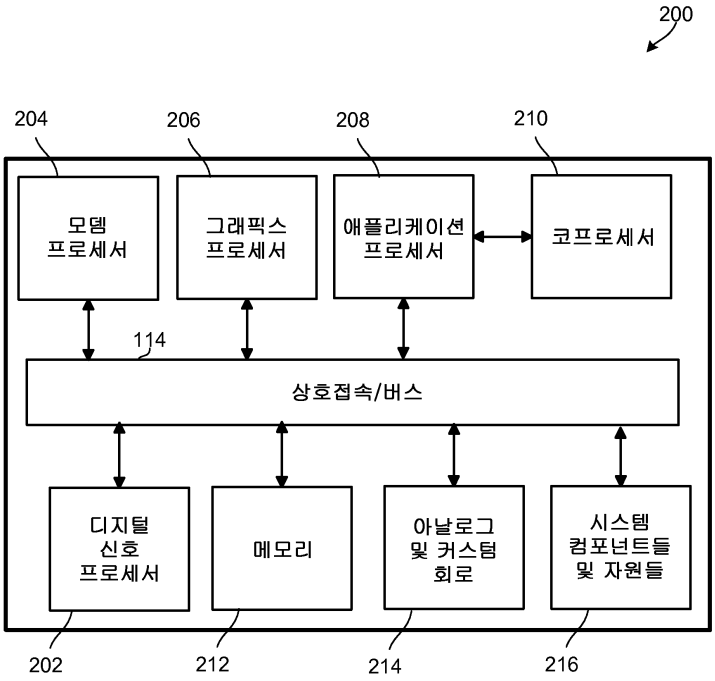
[0164] 개시된 양태들의 전술한 설명은 이 기술분야의 숙련된 임의의 사람이 본 발명을 제작하고 사용할 수 있게끔 제공된다. 이들 양태들에 대한 갖가지 변형예들은 이 기술분야의 숙련된 자들에게 쉽사리 명확하게 될 것이고, 여기에 정의된 일반 원리들은 본 발명의 사상 또는 범위로부터 벗어남 없이 다른 양태들에 적용될 수도 있다. 따라서, 본 발명은 본 명세서에서 보인 양태들로 한정할 의도는 아니며 다음의 청구항들 및 여기에 개시된 원리들 및 신규한 특징들과 일치하는 가장 넓은 범위를 부여하는 것을 의도한다.

도면

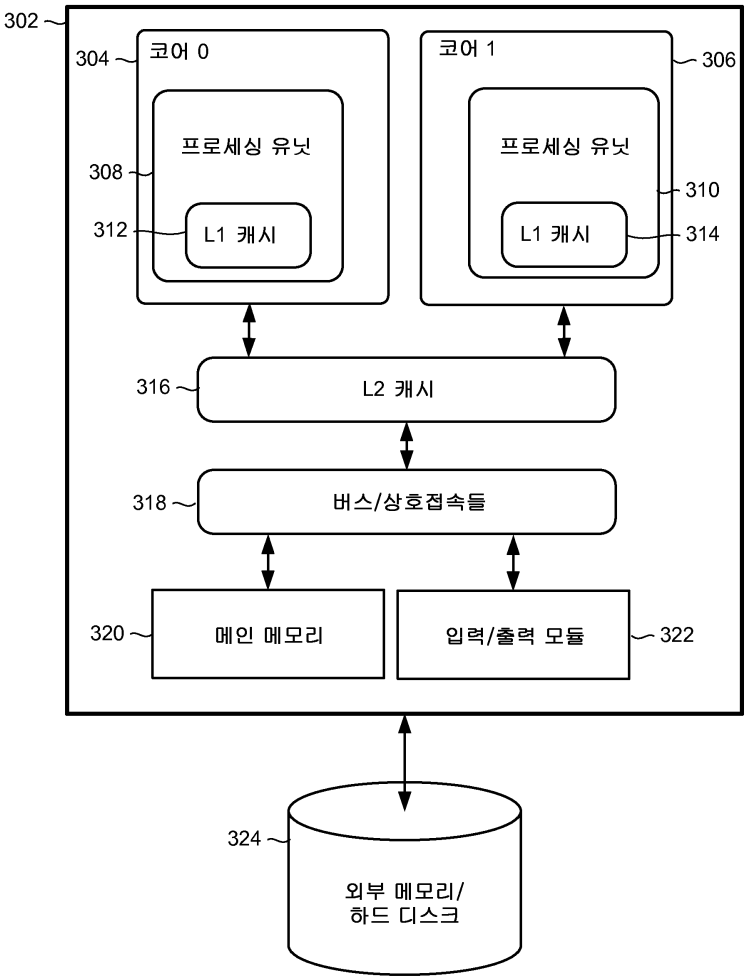
도면1



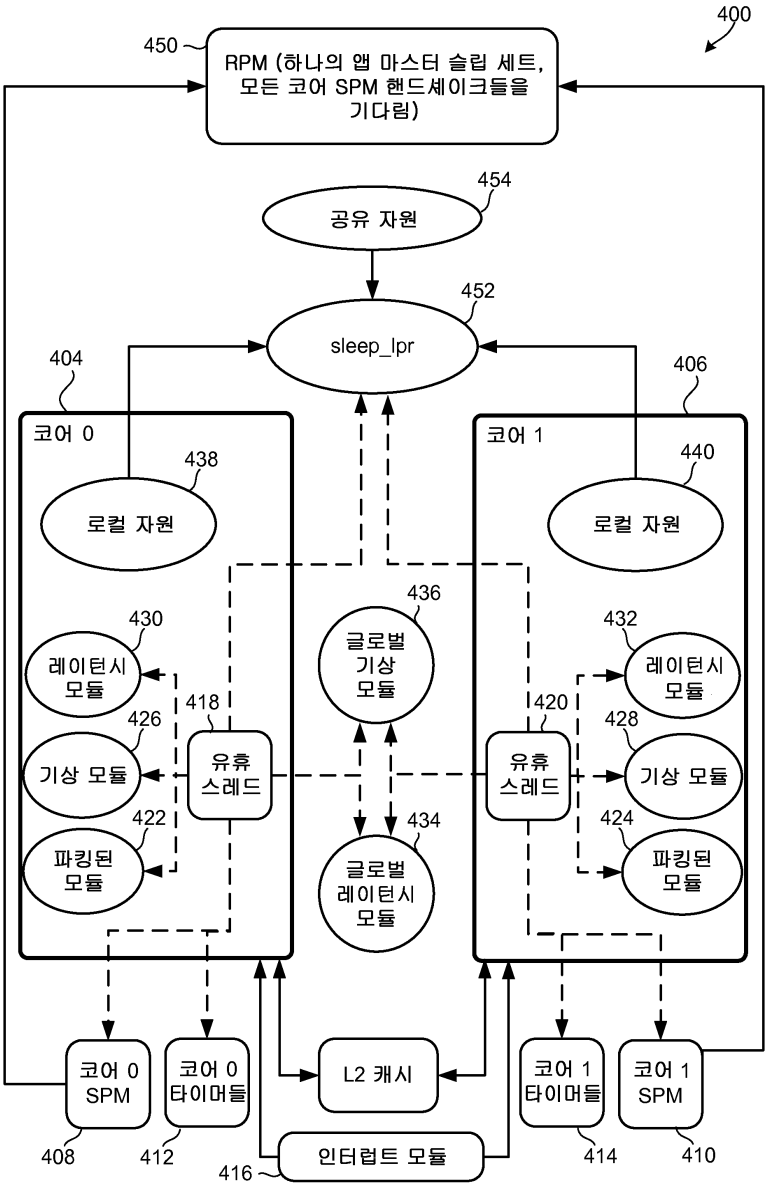
도면2


200
204
206
208
210
모뎀 프로세서
그래픽스 프로세서
애플리케이션 프로세서
코프로세서
114
상호접속/버스
디지털 신호 프로세서
메모리
아날로그 및 커스텀 회로
시스템 컴포넌트들 및 자원들
202
212
214
216

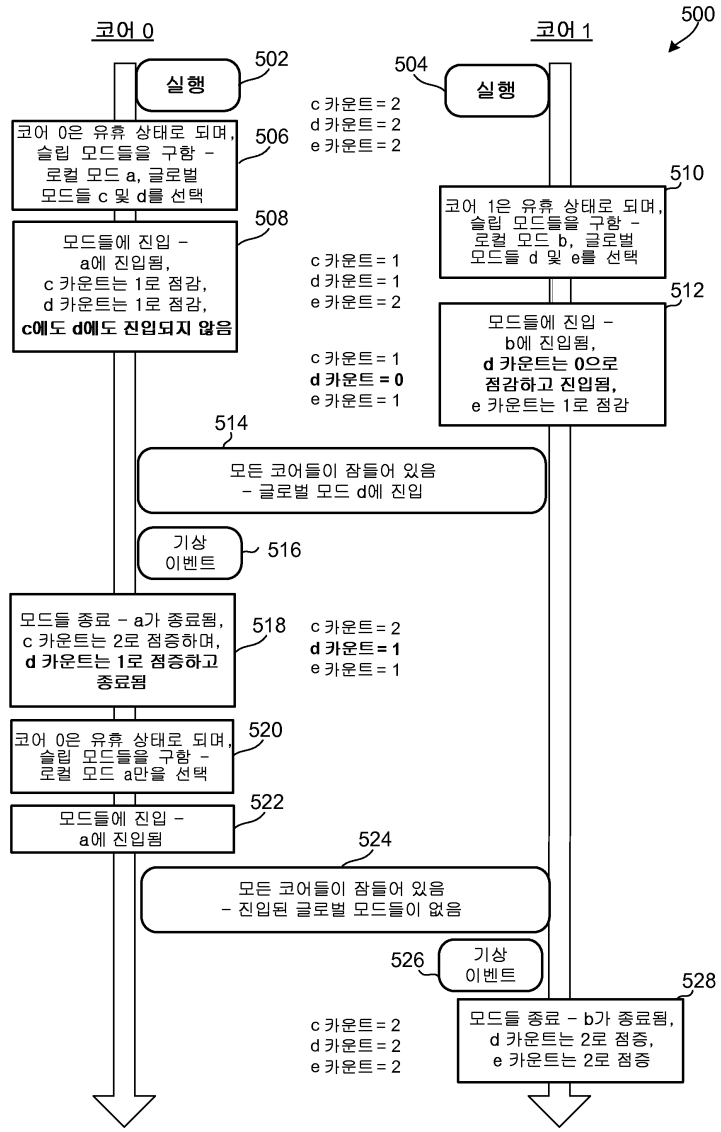
도면3


302
304
코어 0
306
코어 1
308
프로세싱 유닛
310
프로세싱 유닛
312
L1 캐시
314
L1 캐시
316
L2 캐시
318
버스/상호접속들
320
메인 메모리
입력/출력 모듈
322
324
외부 메모리/
하드 디스크

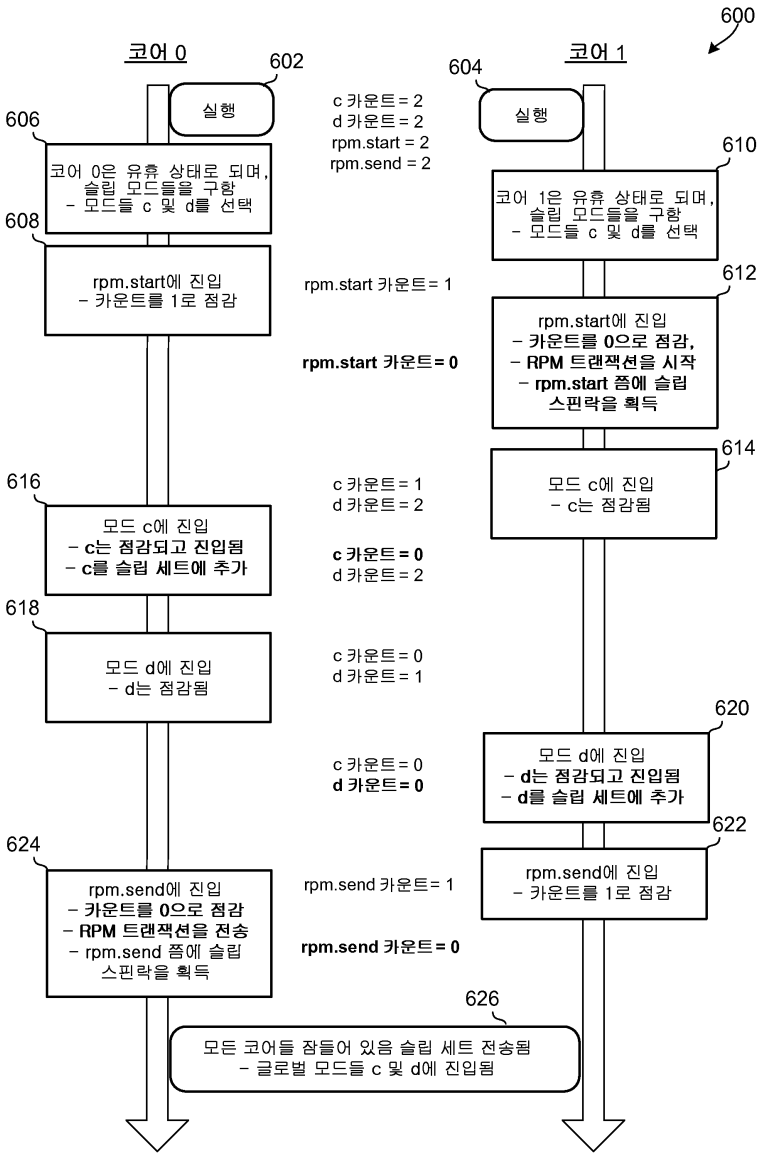
도면4


400
450
RPM (하나의 앱 마스터 슬립 세트, 모든 코어 SPM 핸드셰이크들을 기다림)
454
공유 자원
452
sleep_lpr
404
406
코어 0
코어 1
438
로컬 자원
440
로컬 자원
436
글로벌 기상 모듈
430
레이턴시 모듈
418
426
기상 모듈
유휴 스레드
422
파킹된 모듈
434
글로벌 레이턴시 모듈
420
유휴 스레드
432
레이턴시 모듈
428
기상 모듈
424
파킹된 모듈
코어 0 SPM
코어 0 타이머들
L2 캐시
코어 1 타이머들
코어 1 SPM
408
412
416
인터럽트 모듈
414
410

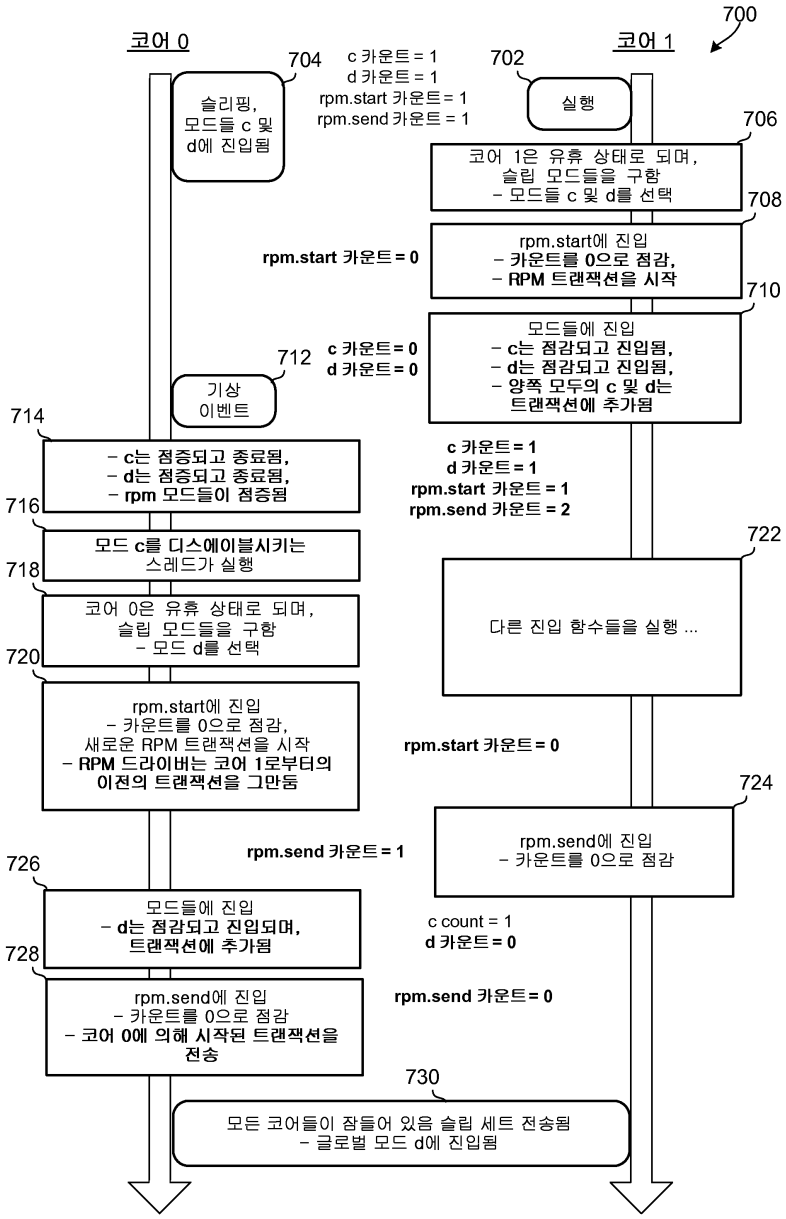
도면5


500
코어 0
코어 1
502
실행
504
실행
c 카운트 = 2
d 카운트 = 2
e 카운트 = 2
506
코어 0은 유휴 상태로 되며, 슬립 모드들을 구함 - 로컬 모드 a, 글로벌 모드들 c 및 d를 선택
510
코어 1은 유휴 상태로 되며, 슬립 모드들을 구함 - 로컬 모드 b, 글로벌 모드들 d 및 e를 선택
508
모드들에 진입 - a에 진입됨, c 카운트는 1로 점감, d 카운트는 1로 점감, c에도 d에도 진입되지 않음
c 카운트 = 1
d 카운트 = 1
e 카운트 = 2
512
모드들에 진입 - b에 진입됨, d 카운트는 0으로 점감하고 진입됨, e 카운트는 1로 점감
c 카운트 = 1
d 카운트 = 0
e 카운트 = 1
514
모든 코어들이 잠들어 있음 - 글로벌 모드 d에 진입
516
기상 이벤트
518
모드들 종료 - a가 종료됨, c 카운트는 2로 점증하며, d 카운트는 1로 점증하고 종료됨
c 카운트 = 2
d 카운트 = 1
e 카운트 = 1
520
코어 0은 유휴 상태로 되며, 슬립 모드들을 구함 - 로컬 모드 a만을 선택
522
모드들에 진입 - a에 진입됨
524
모든 코어들이 잠들어 있음 - 진입된 글로벌 모드들이 없음
526
기상 이벤트
528
모드들 종료 - b가 종료됨, d 카운트는 2로 점증, e 카운트는 2로 점증
c 카운트 = 2
d 카운트 = 2
e 카운트 = 2

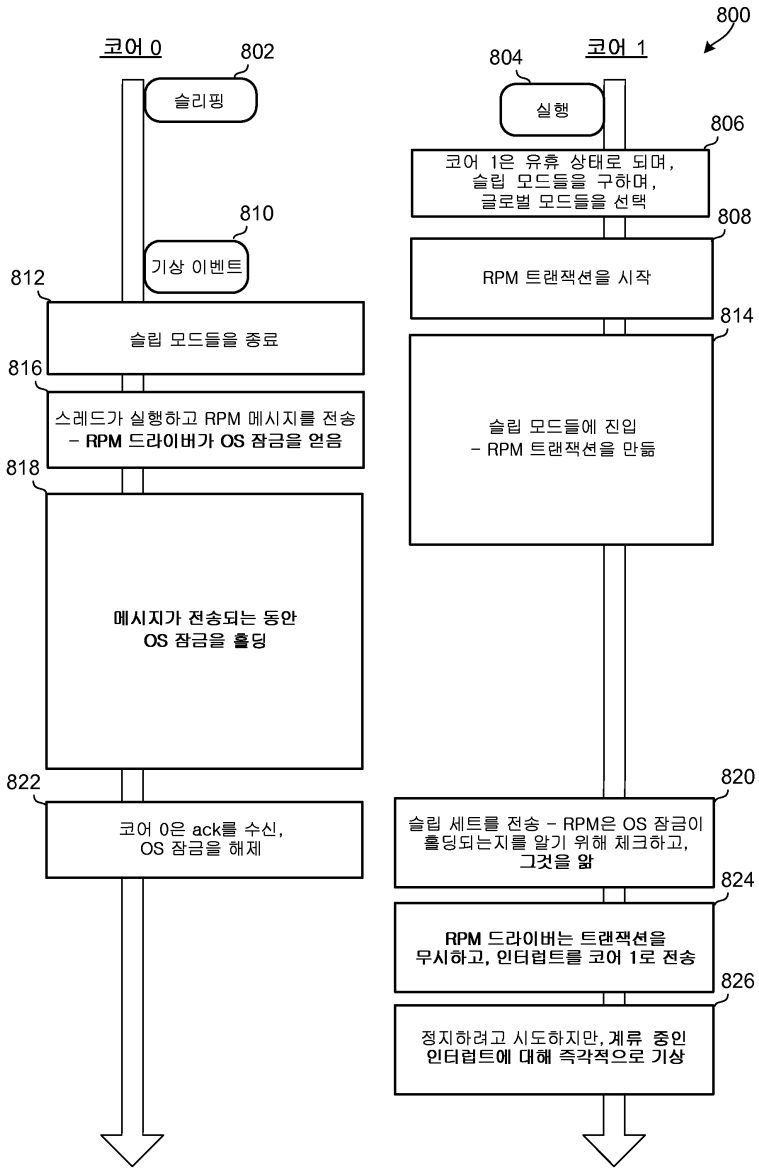
도면6


600
코어 0
코어 1
602
604
실행
실행
606
코어 0은 유휴 상태로 되며, 슬립 모드들을 구함
- 모드들 c 및 d를 선택
c 카운트 = 2
d 카운트 = 2
rpm.start = 2
rpm.send = 2
610
코어 1은 유휴 상태로 되며, 슬립 모드들을 구함
- 모드들 c 및 d를 선택
608
rpm.start에 진입
- 카운트를 1로 점감
rpm.start 카운트 = 1
612
rpm.start에 진입
- 카운트를 0으로 점감,
- RPM 트랜잭션을 시작
- rpm.start 쯤에 슬립 스핀락을 획득
rpm.start 카운트 = 0
614
모드 c에 진입
- c는 점감됨
616
모드 c에 진입
- c는 점감되고 진입됨
- c를 슬립 세트에 추가
c 카운트 = 1
d 카운트 = 2
c 카운트 = 0
d 카운트 = 2
618
모드 d에 진입
- d는 점감됨
c 카운트 = 0
d 카운트 = 1
620
모드 d에 진입
- d는 점감되고 진입됨
- d를 슬립 세트에 추가
c 카운트 = 0
d 카운트 = 0
622
rpm.send에 진입
- 카운트를 1로 점감
624
rpm.send에 진입
- 카운트를 0으로 점감
- RPM 트랜잭션을 전송
- rpm.send 쯤에 슬립 스핀락을 획득
rpm.send 카운트 = 1
rpm.send 카운트 = 0
626
모든 코어들 잠들어 있음 슬립 세트 전송됨
- 글로벌 모드들 c 및 d에 진입됨

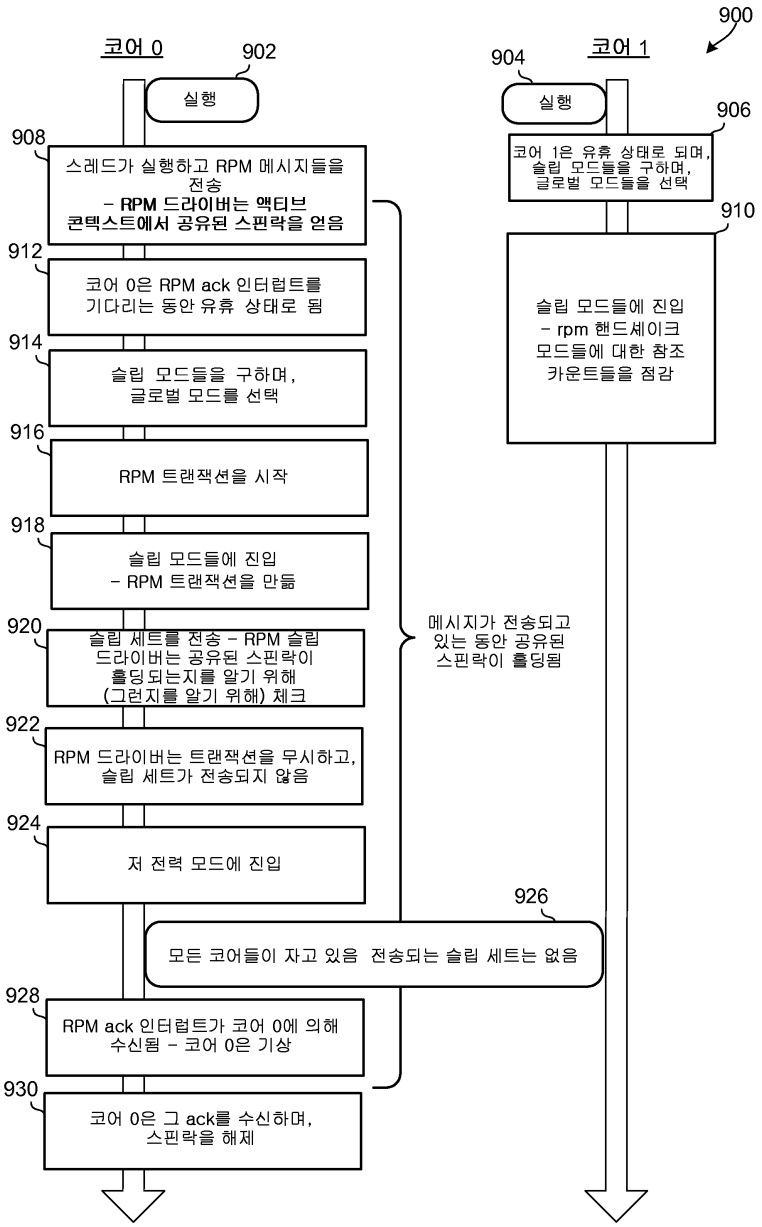
도면7


700
코어 0
코어 1
704
c 카운트 = 1
d 카운트 = 1
rpm.start 카운트 = 1
rpm.send 카운트 = 1
702
슬리핑, 모드들 c 및 d에 진입됨
실행
706
코어 1은 유휴 상태로 되며, 슬립 모드들을 구함
- 모드들 c 및 d를 선택
708
rpm.start 카운트 = 0
rpm.start에 진입
- 카운트를 0으로 점감,
- RPM 트랜잭션을 시작
710
모드들에 진입
- c는 점감되고 진입됨,
- d는 점감되고 진입됨,
- 양쪽 모두의 c 및 d는 트랜잭션에 추가됨
712
c 카운트 = 0
d 카운트 = 0
기상 이벤트
714
- c는 점증되고 종료됨,
- d는 점증되고 종료됨,
- rpm 모드들이 점증됨
c 카운트 = 1
d 카운트 = 1
rpm.start 카운트 = 1
rpm.send 카운트 = 2
716
모드 c를 디스에이블시키는 스레드가 실행
722
718
코어 0은 유휴 상태로 되며, 슬립 모드들을 구함
- 모드 d를 선택
다른 진입 함수들을 실행 ...
720
rpm.start에 진입
- 카운트를 0으로 점감,
새로운 RPM 트랜잭션을 시작
- RPM 드라이버는 코어 1로부터의 이전의 트랜잭션을 그만둠
rpm.start 카운트 = 0
724
rpm.send 카운트 = 1
rpm.send에 진입
- 카운트를 0으로 점감
726
모드들에 진입
- d는 점감되고 진입되며, 트랜잭션에 추가됨
c count = 1
d 카운트 = 0
728
rpm.send에 진입
- 카운트를 0으로 점감
- 코어 0에 의해 시작된 트랜잭션을 전송
rpm.send 카운트 = 0
730
모든 코어들이 잠들어 있음 슬립 세트 전송됨
- 글로벌 모드 d에 진입됨

도면8


800
코어 0
802
슬리핑
810
기상 이벤트
812
슬립 모드들을 종료
816
스레드가 실행하고 RPM 메시지를 전송
- RPM 드라이버가 OS 잠금을 얻음
818
메시지가 전송되는 동안
OS 잠금을 홀딩
822
코어 0은 ack를 수신,
OS 잠금을 해제
804
코어 1
실행
806
코어 1은 유휴 상태로 되며,
슬립 모드들을 구하며,
글로벌 모드들을 선택
808
RPM 트랜잭션을 시작
814
슬립 모드들에 진입
- RPM 트랜잭션을 만듦
820
슬립 세트를 전송 - RPM은 OS 잠금이
홀딩되는지를 알기 위해 체크하고,
그것을 앎
824
RPM 드라이버는 트랜잭션을
무시하고, 인터럽트를 코어 1로 전송
826
정지하려고 시도하지만, 계류 중인
인터럽트에 대해 즉각적으로 기상

도면9


900
코어 0
902
실행
908
스레드가 실행하고 RPM 메시지들을 전송
- RPM 드라이버는 액티브 콘텍스트에서 공유된 스핀락을 얻음
912
코어 0은 RPM ack 인터럽트를 기다리는 동안 유휴 상태로 됨
914
슬립 모드들을 구하며, 글로벌 모드를 선택
916
RPM 트랜잭션을 시작
918
슬립 모드들에 진입
- RPM 트랜잭션을 만듦
920
슬립 세트를 전송 - RPM 슬립 드라이버는 공유된 스핀락이 홀딩되는지를 알기 위해 (그런지를 알기 위해) 체크
922
RPM 드라이버는 트랜잭션을 무시하고, 슬립 세트가 전송되지 않음
924
저 전력 모드에 진입
926
모든 코어들이 자고 있음 전송되는 슬립 세트는 없음
928
RPM ack 인터럽트가 코어 0에 의해 수신됨 - 코어 0은 기상
930
코어 0은 그 ack를 수신하며, 스핀락을 해제
메시지가 전송되고 있는 동안 공유된 스핀락이 홀딩됨
904
코어 1
실행
906
코어 1은 유휴 상태로 되며, 슬립 모드들을 구하며, 글로벌 모드들을 선택
910
슬립 모드들에 진입
- rpm 핸드셰이크 모드들에 대한 참조 카운트들을 점감

도면10