(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2017-0076767
(43) 공개일자 2017년07월04일

(51) 국제특허분류(Int. Cl.)
*G06F 1/32* (2006.01) *G06F 9/50* (2006.01)
(52) CPC특허분류
*G06F 1/3293* (2013.01)
*G06F 1/3275* (2013.01)
(21) 출원번호 10-2017-7014597
(22) 출원일자(국제) 2015년10월09일
심사청구일자 2017년05월29일
(85) 번역문제출일자 2017년05월29일
(86) 국제출원번호 PCT/US2015/054998
(87) 국제공개번호 WO 2016/081090
국제공개일자 2016년05월26일
(30) 우선권주장
14/548,912 2014년11월20일 미국(US)

(71) 출원인
**애플 인크.**
미합중국 95014 캘리포니아 쿠퍼티노 인피니트 루프 1
(72) 발명자
**윌리엄슨, 데이비드 제이.**
미국 95014 캘리포니아주 쿠퍼티노 메일 스탑 241-이엔지 인피니트 루프 1
**윌리엄스 3세, 제라드 알.**
미국 95014 캘리포니아주 쿠퍼티노 메일 스탑 74-3피에이 인피니트 루프 1
*(뒷면에 계속)*
(74) 대리인
**장덕순, 백만기**

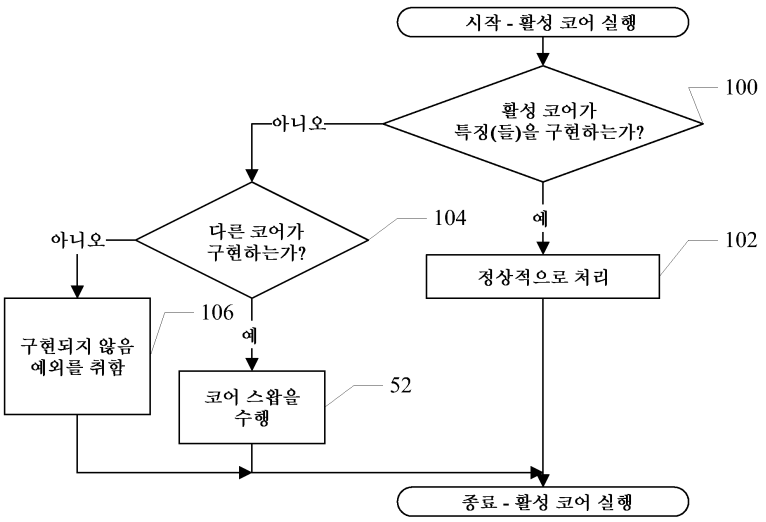
전체 청구항 수 : 총 15 항

(54) 발명의 명칭 **명령어 세트 아키텍처의 상이한 부분들을 구현하는 다수의 비유사 프로세서 코어들을 포함하는 프로세서**

## (57) 요 약

일 실시예에서, 집적회로는 하나 이상의 프로세서를 포함할 수 있다. 각각의 프로세서는 다수의 프로세서 코어들을 포함할 수 있으며, 각각의 코어는 상이한 설계/구현 및 성능 레벨을 갖는다. 예를 들어, 코어는 고성능을 위해 구현될 수 있고, 다른 코어는 더 낮은 최대 성능에서 구현될 수 있지만, 효율을 위해 최적화될 수 있다. 또한, 일부 실시예들에서, 프로세서에 의해 구현되는 명령어 세트 아키텍처의 일부 특징(feature)들은 프로세서를 구성하는 코어들 중 단지 하나에서 구현될 수 있다. 그러한 특징이 상이한 코어가 활성인 동안에 코드 시퀀스에 의해 호출되는 경우, 프로세서는 코어들을, 그 특징을 구현하는 코어로 스와핑할 수 있다. 대안적으로, 예외가 취해질 수 있으며, 특징을 식별하고 대응하는 코어를 활성화시키기 위해 예외 핸들러(exception handler)가 실행될 수 있다.

*대 표 도* - 도5



(52) CPC특허분류
***G06F 1/3287*** (2013.01)
***G06F 9/5094*** (2013.01)
*Y02B 60/1225* (2013.01)
*Y02B 60/1282* (2013.01)
*Y02B 60/32* (2013.01)
(72) 발명자
**하디지 주니어, 제임스 엔.**
미국 95014 캘리포니아주 쿠퍼티노 메일 스탑 241-이엔지 인피니트 루프 1
**루소, 리차드 에프.**
미국 95014 캘리포니아주 쿠퍼티노 메일 스탑 23-2씨에이치피 인피니트 루프 1

# 명 세 서

## 청구범위

### 청구항 1

프로세서 장치로서,

복수의 프로세서 코어 -

상기 프로세서 코어들은 상기 프로세서 장치에 의해 채용된 명령어 세트 아키텍처의 적어도 일부를 구현하고;

상기 명령어 세트 아키텍처의 적어도 제1 특징(feature)은 상기 복수의 프로세서 코어 중 제1 프로세서 코어에서 구현되지 않고;

상기 제1 특징은 상기 복수의 프로세서 코어 중 제2 프로세서 코어에서 구현되고;

주어진 시점에서 상기 복수의 프로세서 코어 중 최대 하나가 활성임 -; 및

상기 복수의 프로세서 코어에 결합된 프로세서 전력 관리자를 포함하며, 상기 프로세서 전력 관리자는,

상기 제1 프로세서 코어에서 실행되고 있는 코드가 상기 명령어 세트 아키텍처의 상기 제1 특징을 사용함을 검출하고;

상기 코드가 상기 제1 특징을 사용함을 검출하는 것에 응답하여 상기 제2 프로세서 코어를 활성화시키고 상기 제1 프로세서 코어를 비활성화시키도록 구성되는, 프로세서 장치.

### 청구항 2

제1항에 있어서,

상기 프로세서 전력 관리자는 복수의 프로세서 상태로 프로그래밍 가능하며, 상기 복수의 프로세서 상태 각각은 상기 복수의 프로세서 코어 중 하나에 매핑되고;

상기 프로세서 전력 관리자는, 상기 프로세서 장치가 현재 프로세서 상태로부터 요청된 프로세서 상태(requested processor state)로 프로그래밍되는 것에 응답하여, 상기 제2 프로세서 코어로부터, 상기 요청된 프로세서 상태가 매핑되는 상기 제1 프로세서 코어로의 프로세서 컨텍스트(processor context)의 이송(transfer)을 야기하도록 구성되고;

상기 프로세서 전력 관리자는, 상기 장치에 의해 실행되고 있는 상기 코드가 상기 제1 특징을 사용하는 것에 응답하여, 상기 이송을 방지하고 상기 제2 프로세서 코어를 활성으로 계속 유지하도록 구성되는, 프로세서 장치.

### 청구항 3

제1항 또는 제2항에 있어서,

상기 제1 프로세서 코어는, 상기 코드의 실행 동안 상기 제1 특징의 상기 사용을 검출하고 상기 사용을 검출하는 것에 응답하여 예외를 시그널링하도록 구성되고;

상기 프로세서 전력 관리자는, 상기 예외에 응답하여 상기 제2 프로세서 코어를 활성화시키고 상기 제1 프로세서 코어를 비활성화시키도록 구성되는, 프로세서 장치.

### 청구항 4

제3항에 있어서, 상기 프로세서 전력 관리자는, 상기 예외에 응답하여 상기 제1 프로세서 코어로부터 상기 제2 프로세서 코어로의 프로세서 컨텍스트의 이송을 야기하도록 구성되는, 프로세서 장치.

### 청구항 5

제3항 또는 제4항에 있어서, 상기 프로세서 전력 관리자는 상기 프로세서 장치에 의해 실행가능한 복수의 명령어를 저장하는 비일시적 컴퓨터 액세스가능 저장 매체를 포함하는, 프로세서 장치.

**청구항 6**

제3항 내지 제5항 중 어느 한 항에 있어서,

상기 제1 프로세서 코어는 상기 명령어 세트 아키텍처의 제2 특징이 상기 복수의 코어 중 어떠한 코어에 의해서도 구현되지 않음을 검출하도록 구성되고;

상기 제1 프로세서 코어는 상기 제2 특징의 사용을 검출하는 것에 응답하여 상이한 예외를 시그널링하도록 구성되는, 프로세서 장치.

**청구항 7**

제1항 내지 제6항 중 어느 한 항에 있어서, 상기 프로세서 전력 관리자는,

상기 제2 프로세서 코어를 전력 온(power on)시키고;

상기 제2 프로세서 코어로 상기 프로세서 컨텍스트를 이송하는 것에 응답하여 상기 제1 프로세서 코어를 전력 차단(power down)하도록 구성되는, 프로세서 장치.

**청구항 8**

제1항 내지 제7항 중 어느 한 항에 있어서, 상기 제1 특징은 벡터 명령어 세트를 포함하는, 프로세서 장치.

**청구항 9**

제1항 내지 제8항 중 어느 한 항에 있어서, 상기 제1 특징은 술어 벡터 명령어 세트(predicated vector instruction set)를 포함하는, 프로세서 장치.

**청구항 10**

제1항 내지 제9항 중 어느 한 항에 있어서, 상기 명령어 세트 아키텍처는 제1 피연산자 유형에 대한 복수의 피연산자 크기를 특정하고, 상기 제1 특징은 상기 복수의 피연산자 크기 중 제1 피연산자 크기를 포함하는, 프로세서 장치.

**청구항 11**

제10항에 있어서, 상기 복수의 피연산자 크기 중 가장 큰 피연산자 크기는 상기 제1 프로세서 코어에 의해 구현되는 유일한 피연산자 크기인, 프로세서 장치.

**청구항 12**

제1항 내지 제11항 중 어느 한 항에 있어서,

상기 제1 프로세서 코어는 상기 명령어 세트 아키텍처의 서브세트를 구현하고;

상기 제2 프로세서 코어는 상기 명령어 세트 아키텍처의 전체를 구현하는, 프로세서 장치.

**청구항 13**

방법으로서,

명령어 세트 아키텍처로 코딩되는 제1 코드가, 프로세서를 형성하는 복수의 프로세서 코어 중 제1 프로세서 코어 상에서 구현되지 않는 특징을 포함한다고 결정하는 단계 - 상기 특징은 상기 복수의 코어 중 제2 프로세서 코어 상에서 구현되고, 주어진 시점에서 상기 복수의 프로세서 코어 중 최대 하나가 활성임 -;

상기 결정하는 단계에 응답하여 상기 제2 프로세서 코어를 활성화시키는 단계;

상기 결정하는 단계에 응답하여 상기 제1 프로세서 코어를 비활성화시키는 단계; 및

상기 제2 프로세서 코어 상에서 상기 제1 코드를 실행하는 단계를 포함하는, 방법.

**청구항 14**

제13항에 있어서,

상기 결정하는 단계에 응답하여 상기 제1 프로세서 코어로부터 상기 제2 프로세서 코어로 프로세서 컨텍스트를 이송하는 단계를 추가로 포함하는, 방법.

**청구항 15**

제13항 또는 제14항에 있어서,

상기 결정하는 단계에 응답하여 상기 제2 프로세서 코어를 전력 온시키는 단계; 및

상기 이송하는 단계 후에 상기 제1 프로세서 코어를 전력 오프(power off)시키는 단계를 추가로 포함하는, 방법.

## *발명의 설명*

### *기 술 분 야*

[0001] 본 명세서에 기술된 실시예들은 프로세서들에 관한 것이며, 보다 상세하게는 프로세서를 형성하는 다수의 프로세서 코어들에 관한 것이다.

### *배 경 기 술*

[0002] 소정 량의 사용자 기능성을 제공하는 소프트웨어를 실행하는 다양한 프로세서들이 전자 시스템들 내에 포함된다. 프로세서들은 시스템 내의 중앙 처리 장치(CPU)들뿐만 아니라, 그래픽, 미디어 처리 등과 같은 특정 태스크들에 전용되는 특수 목적 프로세서들을 포함할 수 있다. 일반적으로, 프로세서들은 다수의 동작점들(공급 전압 크기 및 클록 주파수의 설정들)에서 동작하도록 설계된다. 더 낮은 동작점들은 더 적은 전력을 소모하지만, 또한 더 높은 동작점들에 비해 제한된 성능을 제공한다. 일부 작업부하(workload)들의 경우, 제한된 성능은 충분하며 더 낮은 동작점들이 사용될 수 있다. 다른 작업부하들의 경우, 충분한 성능을 제공하기 위해 더 높은 동작점들이 필요하다.

[0003] 일부 시스템들에서, 매우 다양한 작업부하들이 경험된다. 가장 요구가 많은 작업부하들에 의해 필요한 성능을 제공할 수 있는 프로세스를 설계하면서, 또한 많은 빈번하게-실행되는 작업부하들에 대한 충분한 성능을 제공할 최저의 가능한 동작점을 지원하는 것이, 과제가 되었다. 높은 동작점들에서 동작하는 프로세서들은 단지, 회로가 올바르게 기능하는 것을 중지하기 전에 공급 전압의 특정 레벨로의 감소를 지원할 수 있다. 절충이 이루어져야 하며, 전형적으로 최저 동작점은, 원하는 하이 엔드(high end) 동작점을 설계가 충족시킬 수 있을 때까지 증가된다. 하이 엔드 동작점들이 계속해서 증가함에 따라, 점점 더 많은 작업부하들이 최저 동작점에서 실행가능하다(그리고 많은 작업부하들이 훨씬 더 낮은 동작점들에서 실행될 수 있다). 그러한 작업부하들에 대해 전력은 불필요하게 소비되는데, 이는 배터리와 같은 제한된 에너지원에서 빈번하게 동작하는 모바일 시스템들에 있어서 중요한 인자일 수 있다.

## *발명의 내용*

[0004] 일 실시예에서, 집적회로는 하나 이상의 프로세서를 포함할 수 있다. 각각의 프로세서는 다수의 프로세서 코어들을 포함할 수 있으며, 각각의 코어는 상이한 설계/구현 및 성능 레벨을 갖는다. 예를 들어, 코어는 고성능에 대해 구현될 수 있지만, 그것이 올바르게 동작하는 더 높은 최소 전압을 가질 수 있다. 다른 코어는 더 낮은 최대 성능에서 구현될 수 있지만, 효율을 위해 최적화될 수 있고 더 낮은 최소 전압에서 올바르게 동작할 수 있다. 또한, 일부 실시예들에서, 프로세서에 의해 채용되는 명령어 세트 아키텍처의 일부 특징(feature)들은 프로세서를 구성하는 코어들 중 단지 하나에서 구현될 수 있다(또는 적어도 하나의 코어를 배제하는, 코어들의 서브세트에 의해 구현될 수 있다). 그러한 특징이 상이한 코어가 활성인 동안에 코드 시퀀스에 의해 호출되는 경우, 프로세서는 코어들을, 그 특징을 구현하는 코어들 중 하나로 스와핑(swapping)할 수 있다. 대안적으로, 예외가 취해질 수 있으며, 특징을 식별하고 대응하는 코어를 활성화시키기 위해 예외 핸들러(exception handler)가 실행될 수 있다.

[0005] 일부 실시예들에서, 특정 특징들을 하나의 코어로 또는 적어도 모든 코어들보다 적은 코어들로 제한하는 것은, 동일한 명령어 유형들을 처리하는 코어들 내의 중복 회로를 제거함으로써 영역 효율적인 구현을 제공할 수 있다. 예를 들어, 고성능 코드에서만 사용될 것같은 특징들은 고성능 코어에서만 구현될 수 있으며, 이는 그

코어가 고성능 코드를 실행할 가능성이 가장 높기 때문이다. 사용될 가능성이 낮은 특징들(예를 들어, 하위 호환성을 위해 제공되었지만 더 새로운 코드에 의해 사용되지 않는 특징들)은 하나의 코어에서 구현될 수 있으며, 따라서 영역 관점에서 효율적으로 지원될 수 있다.

[0006] 프로세서는 다수의 프로세서 상태(PState)들을 지원할 수 있다. 각각의 PState는 동작점(예를 들어, 공급 전압 크기와 클록 주파수의 조합)을 특정할 수 있고, 각각의 PState는 프로세서 코어들 중 하나에 매핑될 수 있다. 동작 동안, 코어들 중 하나가 활성이며, 그 코어에는 현재 PState가 매핑된다. 새로운 PState가 선택되어 상이한 코어에 매핑되는 경우, 프로세서는 프로세서 상태를 새로-선택된 코어로 자동으로 컨텍스트 전환할 수 있고 그 코어 상에서 실행을 시작할 수 있다. 일 실시예에서, 프로세서는 새로-선택된 코어가 현재 작업부하에 의해 사용 중인 특징들을 지원하는지 여부를 검출할 수 있고, 지원되지 않는다면 교정 동작을 취할 수 있다.

## 도면의 간단한 설명

[0007] 하기의 상세한 설명은 첨부 도면들을 참조하며, 이제 도면들이 간단히 설명된다.

도 1은 프로세서 클러스터의 일 실시예의 블록도이다.

도 2는 일 실시예에 대해 도 1에 도시된 바와 같은 PCore 및 ECore에 대한 효율 대 성능을 예시하는 그래프이다.

도 3은 프로세서 상태들을 변경하기 위한 프로세서 전력 관리 유닛의 일 실시예의 동작을 예시하는 흐름도이다.

도 4는 코어들을 스와핑하기 위한 프로세서 전력 관리 유닛의 일 실시예의 동작을 예시하는 흐름도이다.

도 5는 명령어들의 실행 동안 활성 코어의 일 실시예에 대한 동작을 예시하는 흐름도이다.

도 6은 명령어들의 실행 동안 활성 코어의 다른 실시예에 대한 동작을 예시하는 흐름도이다.

도 7은 컴퓨터 액세스가능 저장 매체의 블록도이다.

도 8은 코어 스왑을 위한 컨텍스트 전환 하드웨어의 일 실시예의 블록도이다.

도 9는 도 1에 도시된 프로세서 클러스터의 일 실시예를 포함하는 시스템온칩(SOC)의 일 실시예의 블록도이다.

도 10은 시스템의 일 실시예의 블록도이다.

본 개시내용에 기술된 실시예들은 다양한 수정들 및 대안적인 형태들을 허용하지만, 본 개시내용의 특정 실시예들이 도면들에 예로서 도시되고, 본 명세서에서 상세히 기술될 것이다. 그러나, 그에 대한 도면들 및 상세한 설명은 실시예들을 개시된 특정 형태로 제한하는 것으로 의도되는 것이 아니라, 그와는 반대로, 의도는 첨부된 청구범위의 사상 및 범주 내에 속한 모든 수정들, 등가물들 및 대안들을 커버하기 위한 것임을 이해하여야 한다. 본 명세서에서 사용된 표제들은 단지 구성상의 목적을 위한 것일 뿐, 설명의 범주를 제한하기 위해 사용되는 것으로 의도되지 않는다. 본 출원 전반에 걸쳐 사용되는 바와 같이, "일 수 있다(may)"라는 단어는 의무적인 의미(즉, "이어야만 한다(must)"를 의미)라기보다 오히려 허용의 의미(즉, "~에 대해 가능성을 갖는다"는 의미)로 사용된다. 유사하게, "포함하다(include, includes)" 및 "포함하는(including)"이라는 단어는, 포함하지만 이로 제한되지 않음을 의미한다.

다양한 유닛들, 회로들 또는 기타 컴포넌트들이 태스크 또는 태스크들을 수행하도록 "구성되는 것"으로 설명될 수 있다. 그러한 맥락에서, "~하도록 구성된"은 동작 동안에 태스크 또는 태스크들을 수행하는 "회로를 갖는"을 일반적으로 의미하는 구조의 광의의 설명이다. 이와 같이, 유닛/회로/컴포넌트는 유닛/회로/컴포넌트가 현재 온(on) 상태가 아닐 시에도 태스크를 수행하도록 구성될 수 있다. 일반적으로, "~하도록 구성된"에 대응하는 구조를 형성하는 회로는 동작을 구현하기 위하여 실행가능한 프로그램 명령어들을 저장하는 하드웨어 회로들 및/또는 메모리를 포함할 수 있다. 메모리는 정적 또는 동적 랜덤 액세스 메모리와 같은 휘발성 메모리 및/또는 비휘발성 메모리, 예를 들어 광 또는 자기 디스크 저장장치, 플래시 메모리, 프로그래밍 가능한 판독 전용 메모리(programmable read-only memory) 등을 포함할 수 있다. 유사하게, 다양한 유닛/회로/컴포넌트들은 설명의 편의를 위해 태스크 또는 태스크들을 수행하는 것으로 설명될 수 있다. 그러한 설명은 "~하도록 구성된"이라는 문구를 포함하는 것으로 해석되어야 한다. 하나 이상의 태스크를 수행하도록 구성된 유닛/회로/컴포넌트를 언급하는 것은 그 유닛/회로/컴포넌트에 대해 35 U.S.C. § 112 (f)항의 해석을 적용하지 않고자 명확히 의도된다.

본 명세서는 "일 실시예" 또는 "실시예"에 대한 참조를 포함한다. 본 명세서에서 명시적으로 부인하지 않는다

면, 임의의 특징들의 조합을 포함하는 실시예들이 일반적으로 고려되더라도, "일 실시예에서" 또는 "실시예에서"라는 문구가 나타난다고 해서 반드시 동일한 실시예를 지칭하지는 않는다. 특정 특징들, 구조들 또는 특성들이 본 개시내용과 일관성을 유지하는 임의의 적합한 방식으로 조합될 수 있다.

## *발명을 실시하기 위한 구체적인 내용*

[0008] 도 1은 프로세서 클러스터(30)의 일 실시예의 블록도이다. 도시된 실시예에서, 다수의 프로세서들(32A 내지 32n) 및 레벨 2(L2) 캐시(34)가 포함된다. 프로세서들(32A 내지 32n)은 L2 캐시(34)에 결합되며, 이것은 또한 클러스터(30)를 포함하는 시스템의 다른 요소들과 통신하기 위해 결합된다. 도시된 실시예에서, L2 캐시(34)는 프로세서들(32A 내지 32n)에 대한 PState를 저장하는 PState 레지스터(38)를 포함하는 프로세서 전력 관리자(36)를 포함한다. 다양한 실시예들에서, 각각의 프로세서(32A 내지 32n)가 자신의 독립적인 PState를 가질 수 있거나, 프로세서들(32A 내지 32n)의 그룹들이 PState를 공유할 수 있거나, 또는 클러스터(30)가 프로세서들(32A 내지 32n)에 대한 공유 PState를 가질 수 있다. 프로세서(32A)는 적어도 2개의 프로세서 코어인, 성능 코어(performance core; PCore)(40) 및 효율적 코어(efficient core; ECore)(42)를 포함하는 것으로, 도 1에서 더 상세히 도시되어 있다. 다른 실시예들은 추가 코어들을 포함할 수 있다. 각각의 코어(40, 42)는 각각의 전력 스위치들(44, 46)을 통해 전력 공급 레일($V_P$)에 결합된다. 따라서, 각각의 코어(40, 42)는 독립적으로 전력 공급되거나(powered up) 차단될(powered down) 수 있다. 프로세서(32n)와 같은 다른 프로세서들은 프로세서(32A)와 유사할 수 있다.

[0009] 각각의 프로세서(32A 내지 32n)는, 시스템에서 실행되는 소프트웨어가 실행할 코드를 할당할 수 있는 엔티티일 수 있다. 예를 들어, 소프트웨어는 시스템 내의 하드웨어를 제어하는 운영체제(OS)의 일부일 수 있다. 소프트웨어는 실행될 코드를 스케줄링하는 스레드(thread) 또는 태스크 스케줄러일 수 있다. OS는 또한 실행되고 있는 코드의 성능 요구들에 기초하여 프로세서들(32A 내지 32n)에 PState를 할당할 수 있다. OS는 코드의 거동을 추적하여 PState들을 결정할 수 있거나, 각 스레드/태스크에 대한 정보를 정적으로 기록하여 PState를 선택하는 것 등을 수행할 수 있거나, 또는 이들의 임의의 조합일 수 있다. 추가로 또는 대안적으로, PState는 시스템 내의 다른 조건들(열 제한들, 이용가능한 배터리 전력 등)에 의해 영향을 받을 수 있다.

[0010] PCore(40) 및 ECore(42)는 상이한 설계들, 프로세서들(32A 내지 32n)에 의해 채용된 명령어 세트 아키텍처(ISA)의 상이한 구현들일 수 있다. 다른 방식으로 보면, PCore(40) 및 ECore(42)는 상이한 마이크로아키텍처들을 구현할 수 있다. PCore(40)는, 전력 보존을 덜-강조된 설계 목표로 하면서, 성능을 최대화하려고 시도하는 적극적인 설계(aggressive design)일 수 있다. PCore(40) 내의 회로는 적극적일 수 있으며, 이는 PCore(40)가 동작할 수 있는 최소 공급 전압이 PState들 중 일부에서 요구될 수 있는 것만큼 낮아지는 것을 방지할 수 있다. 반면에, ECore(42)는 더 보존적인 설계(conservative design)를 구현할 수 있으며, 따라서 PCore(40)보다 낮은 최소 전압들에서 올바르게 동작할 수 있다. ECore(42)의 성능은 주어진 동작점에서 PCore(40)보다 낮을 수 있으며, 전력 보존은 ECore(42)에 대한 더 많이-강조된 목표가 될 수 있다. ECore(42)가 차지하는 반도체 영역은 또한 PCore(40)의 반도체 영역보다 작을 수 있다.

[0011] 보다 상세하게는, 일 실시예에서, ECore(42) 및/또는 PCore(40)는 프로세서들(32A 내지 32n)에 의해 채용된 ISA의 서브세트를 구현할 수 있으며, 여기서 ISA의 하나 이상의 특징은 서브세트에 포함되지 않는다. 일 실시예에서, PCore(40)는 ISA의 전체를 구현할 수 있고 ECore(42)는 서브세트를 구현할 수 있다. 다른 실시예에서, PCore(40) 및 ECore(42)는 각각 상이한 서브세트를 구현할 수 있다. 서브세트들은 부분적으로 중첩할 수 있다(예를 들어, 정수 명령어들과 같은 일반적으로 사용되는 명령어들은 각각의 서브세트의 일부일 수 있다).

[0012] 다양한 실시예들은 다양한 방식들로 특정 서브세트로부터 배제되는 특징들을 선택할 수 있다. ECore(42)의 경우, 드물게 사용되고 (예를 들어, 차지하는 반도체 기판 영역, 전력 소모 등의 관점에서) 구현하는 데 비용이 많이 드는 특징은 배제될 수 있다. 예를 들어, 일 실시예에서, ISA는 주어진 피연산자 유형의 다수의 피연산자 크기들을 정의할 수 있다. 일 실시예에서, 피연산자 유형은 정수일 수 있고, 크기들은 32-비트 및 64-비트를 포함할 수 있다. 현대의 코드는 64-비트 정수 코드를 향하는 추세로 보인다. 반면에, 64-비트 및 32-비트 정수 피연산자 크기들 둘 모두를 지원하는 하드웨어는 영역-소모적일 수 있고 타이밍을 가중시킬 수 있으며, 이는 더 높은 전력 구현을 초래할 수 있다. 따라서, ECore(42)는 64-비트 정수 하드웨어를 구현할 수 있고 32-비트 정수 코드에 대한 지원을 배제할 수 있다. 다른 ISA 특징들이 유사하게 배제될 수 있다. 예를 들어, ISA들은 종종 피연산자들의 벡터에 대한 단일-명령어, 다중 데이터(single-instruction, multiple data; SIMD) 처리를 수행하는 벡터 명령어 세트들을 포함한다. 벡터 구현들은 높은 전력 및/또는 영역-소모적일 수 있다. 보다 최근의 ISA 도입은 루프 벡터화를 용이하게 하기 위한 술어 벡터 명령어 세트(predicated vector instruction

set)이다. 이러한 특징은 또한 ECore(42)로부터 제거될 수 있다. 일반적으로, ISA 특징은 명령어, 명령어들의 세트, 피연산자 유형 또는 크기, 모드 등을 포함할 수 있다.

[0013] 각각의 가능한 PState는 프로세서(32A 내지 32n)에 대한 동작점을 특정할 수 있다. 예를 들어, 동작점은 프로세서(32A 내지 32n)에서의 $V_P$에 대한 공급 전압 크기 및 클록들에 대한 클록 주파수를 포함할 수 있다. 다른 실시예들은 다른 방식들로 동작점을 정의할 수 있지만, 일반적으로 동작점은 프로세서의 성능 및 전력 소모를 나타낼 수 있다. 일 실시예에서, PState는 공급 전압 크기 및 클록 주파수로서 직접 사용되는 한 쌍의 값일 수 있다. 다른 실시예들에서, PState는 공급 전압 크기 및 클록 주파수를 획득하는 데 사용되는 값일 수 있다(예를 들어, 값들의 표로의 인덱스).

[0014] 도 1에 도시된 바와 같이, 프로세서(32A)는 PCore(40) 및 ECore(42)를 포함한다. 프로세서(32A)에 의해 지원되는 각각의 PState는 코어들(40, 42) 중 하나에 매핑된다. 각각의 코어(40, 42)는 그것에 매핑된 하나 초과의 PState를 가질 수 있다.

[0015] 프로세서(32A 내지 32n)에 의해 실행되고 있는 코드가 변하고/하거나 다른 시스템 고려사항들이 PState의 변경을 보증함에 따라, PState 레지스터(38)는 (예를 들어, OS에 의해) 업데이트될 수 있다. PState가 코어들(40, 42) 중 하나("활성 코어")에 매핑된 현재 PState로부터 코어들(40, 42) 중 다른 하나("목표 코어")에 매핑된 새로운 PState로 변경되는 경우, 클러스터(30)는 프로세서(32A)의 프로세서 컨텍스트를 활성 코어로부터 목표 코어로 자동으로 하드웨어적으로 이송(transfer)할 수 있다. 목표 코어는 PState가 변경될 때 전력 오프되어 있을 수 있다. 컨텍스트를 이송하는 프로세스는 목표 코어를 전력 온(power on)시키고, 목표 코어를 리셋 및 초기화하고, 프로세서 컨텍스트를 이송하고, 활성 코어를 전력 오프(power off)시키는 것(목표 코어를 활성 코어로 만드는 것)을 포함할 수 있다. 목표 코어(현재 활성 코어) 상에서 실행이 계속될 수 있다. 따라서, 코어들 사이의 전환은 소프트웨어에 보이지 않을(invisible) 수 있다. 사실상, 소프트웨어는 프로세서(32A 내지 32n) 내에 다수의 코어들이 있다는 것을 "인식"하지 못할 수도 있다.

[0016] 도 1에 도시된 예는 프로세서(32A) 내에 2개의 코어를 포함하지만, 다른 실시예들은 2개 초과의 코어를 포함할 수 있다. 하나의 코어가 최저 PState들에서 동작하는 가장 효율적 코어일 수 있으며, 다른 코어들은, 또 다른 코어가 프로세서 내의 다수의 코어들 중 최고 성능 코어일 때까지, 성능/효율 스펙트럼을 따라 다른 지점들에 대해 최적화될 수 있다. 임의의 수의 코어들이 다양한 실시예들에서 사용될 수 있다.

[0017] 일반적으로, 프로세서는 정의된 명령어 세트 아키텍처(ISA)를 구현하도록 구성된 임의의 회로일 수 있다. x86 아키텍처(APX로도 알려짐), ARM 아키텍처, MIPS 아키텍처, 파워 PC(PowerPC; 현재는 단순히 Power) 등과 같은 다양한 ISA들이 존재하며, 다양한 실시예들에서 사용될 수 있다. 전술한 다중 코어 접근법을 포함하는 다양한 마이크로아키텍처 기술들이 프로세서에 의해 채용될 수 있다. 각각의 코어는 또한 다양한 마이크로아키텍처 기술들을 구현할 수 있다. 일반적으로, 마이크로아키텍처는 ISA를 구현하는 데 사용되는 실행 유닛들 및 다른 회로의 구조를 지칭할 수 있다. 예들은 순차형(in-order) 대 비순차형(out-of-order) 실행, 추측형 실행, 분기 예측, 수퍼스칼라, 수퍼파이프라인형 등을 포함할 수 있다. 실시예들은 다양한 다른 기술들에 부가하여 마이크로코딩 기술들을 구현할 수 있다.

[0018] 프로세서들(32A 내지 32n) 및/또는 프로세서 컴플렉스(30)는 시스템 내의 임의의 프로세서들로서 사용될 수 있다. 예를 들어, 프로세서들은, OS를 실행하여 시스템 내의 다른 하드웨어를 제어하고 실행될 애플리케이션 코드를 스케줄링하는 중앙 처리 장치(CPU)들일 수 있다. CPU는 또한 애플리케이션 코드를 실행할 수 있다. 프로세서들은 그래픽 조작을 위해 최적화된 그래픽 처리 유닛(GPU), 신호 처리를 위해 최적화된 디지털 신호 프로세서(DSP), 다양한 주변 컴포넌트들에서의 소프트웨어 실행을 수행하는 임베디드 프로세서 등과 같은 특수 목적 프로세서들일 수 있다.

[0019] 일 실시예에서, 프로세서(32A 내지 32n)를 형성하는 코어들(40, 42) 중 최대 하나는 프로세서 컨텍스트가 이송되는 시간들을 제외하고는 실행 동안에 전력 온될 수 있다. 주어진 프로세서(32A 내지 32n)는 완전히 오프될 수 있다(모든 코어들이 전력 차단됨). 프로세서 전력 관리자(36)는 전력 스위치들(44, 46)을 사용하여 프로세서 코어들의 전력 온/공급 및 프로세서 코어들의 전력 오프/차단을 제어하도록 구성될 수 있다.

[0020] 일부 실시예들에서, 코어들(40 및/또는 42)은 수정된 데이터(즉, 예를 들어, 실행되고 있는 프로세서 코드에서의 저장에 응답하여 캐시에 기입된 데이터이지만, 아직 메모리에 기입되지 않아서 메모리 내의 데이터는 더 이상 올바른 데이터가 아님)를 저장할 수 있는 데이터 캐시들을 구현할 수 있다. 프로세서 컨텍스트를 이송하는 것에 부가하여, 수정된 데이터는 데이터 캐시로부터 플러시(flush)될 수 있다. 특히, 데이터는 L2 캐시(34)로

플러시될 수 있지만, L2 캐시(34)의 정상 동작이 데이터를 축출되게 하지 않는 한 L2 캐시(34)에 저장된 채로 유지될 수 있다. 일단 새로운-활성 코어가 실행되고 있으면, 수정된 데이터는 L2 캐시(34)에서 히트(hit)일 수 있고 상대적으로 낮은 레이턴시를 가지면서 새로운-활성 코어의 캐시 내로 이동될 수 있다.

[0021] 프로세서 전력 관리자(36)는 프로세서 클러스터(30) 내의 PState 전이(transition)들을 관리하도록 구성될 수 있다. 프로세서 전력 관리자(36)는 공급 전압 크기 전이들을 시스템 레벨 전력 관리자에 전달하거나, 또는 시스템에 전압들을 공급하는 전력 관리 유닛(PMU)에 직접 전달하도록 구성될 수 있다. 프로세서 전력 관리자(36)는 위상 고정 루프(PLL) 등과 같은 클록 생성 하드웨어(도 1에 도시되지 않음)와 상호작용하도록 구성될 수 있다.

[0022] 프로세서 컨텍스트는 일반적으로 임의의 소프트웨어-가시적(software-visible) 프로세서 상태를 포함할 수 있다. 상태는 전형적으로 ISA 내에 정의된 다양한 명령어들의 피연산자들로서 액세스가능한 레지스터들 내에 저장될 수 있다. 상태는 다양한 유형들(정수, 부동 소수점, 벡터 등)의 피연산자 레지스터들과 같은 구조화된 레지스터들을 포함할 수 있다. 레지스터들은 또한 상태 레지스터, 프로세서 모드 레지스터 등과 같은 프로세서 제어 레지스터들을 포함할 수 있다. 레지스터들은 또한 특정 유닛에 대한 특정 콘텐츠를 포함하도록 정의된 특수 목적 레지스터들을 포함할 수 있다. 레지스터들은 모델 특정 레지스터들을 추가로 포함할 수 있으며, 그 존재는 구조적으로 특정될 수 있지만, 그 콘텐츠는 구현예에 따라 달라질 수 있다.

[0023] L2 캐시(34)는 임의의 용량 및 구성을 가질 수 있다. L2 캐시(34)는 프로세서들(32A 내지 32n) 내의 캐시들을 포함하거나, 그 캐시들을 제외하거나, 또는 포함하지 않을 수도 있다.

[0024] 도 2는 코어들(40, 42)의 일 실시예에 대한 효율 대 성능을 도시하는 그래프이다. 점선 곡선은 ECore(42)에 대응하고 실선은 PCore(40)에 대응한다. 효율은 수직축 상에, 성능은 수평축 상에 그래프로 나타난다. 효율은 다양한 방식들로 측정될 수 있다(예를 들어, 성능/와트). 성능은 스페크인트(Specint), 스페크Flt(SpecFlt), 드라이스톤(Dhrystone) 등과 같은 다양한 벤치마크 프로그램들을 사용하여 측정될 수 있다. 프로세서(32A)에 대한 다양한 PState들이 도 2에서 곡선들을 따라 도시된다. PCore가 성능에 최적화되어 있기 때문에 더 높은 성능에 대응하는 PState들은 PCore 곡선 상에 있으며, 여기서 더 낮은 성능/더 높은 에너지 보존에 대응하는 PState들은 ECore 곡선 상에 있으며, 이것은 더 낮은 성능 레벨들에서 더 효율적이지만 더 높은 성능 레벨들에서는 성능이 더 낮다.

[0025] 따라서, 도 2의 예에서, PState 1, 2 및 3은 ECore(42)에 매핑되고 PState 4, 5, 6 및 7은 PCore(40)에 매핑된다. 임의의 수의 PState들이 지원될 수 있고 임의의 수는 다양한 코어들에 매핑될 수 있다. 다른 실시예에서, 연속적인 PState 설정들이 지원될 수 있다. 그러한 실시예에서, 도 2에서 곡선들이 교차하는 브레이크오버 포인트(break over point)는 코어 전환이 발생할 수 있는 위치에서 정의될 수 있다.

[0026] 도 3은 PState 레지스터(38)에 기입된 새로운 PState에 응답하는 프로세서 전력 관리자(36)의 동작의 일 실시예를 도시하는 흐름도이다. 블록들이 이해의 용이함을 위해 특정 순서로 도시되었지만, 다른 순서들도 사용될 수 있다. 블록들은 프로세서 전력 관리자(36) 내의 조합 로직에서 병렬로 수행될 수 있다. 블록들, 블록들의 조합들, 및/또는 흐름도 전체는 다수의 클록 사이클들에 걸쳐 파이프라인화될 수 있다. 프로세서 전력 상태 관리자(36)는 도 3에 도시된 동작을 구현하도록 구성될 수 있다.

[0027] 활성 코어는 코드를 현재 실행하고 있는 코어(40/42)일 수 있다. 일부 실시예들에서, 활성 코어는 현재 PState에서 정상 상태 동작 동안 전력 온되는 유일한 코어일 수 있다. 새로운 PState가 활성 코어에 매핑되지 않는 경우(결정 블록(50), "아니오" 레그(leg)), 프로세서 전력 상태 관리자(36)는 작업부하 및 목표 코어에 관한 임의의 정보를 고려하여 목표 코어가 작업부하를 지원하는지 여부를 결정할 수 있다(결정 블록(64)). 예를 들어, 일부 코드는 플래시 메모리와 같은 비휘발성 메모리에 저장될 때 디스크립터들을 포함할 수 있으며, 디스크립터들은 어떤 ISA 특징들이 코드에 의해 사용되는지를 나타낼 수 있다. 프로세서 전력 상태 관리자(36)는 디스크립터들로부터 코드에 의해 사용되는 특징들을 결정할 수 있다. 대안적으로, 프로세서들(32A 내지 32n)은 모든 코어들보다 적은 코어들에 의해 구현되는 ISA 특징들을 추적할 수 있다. 추적된 상태는 목표 코어가 현재 사용중인 특징들을 지원하는지 여부를 결정하는 데 사용될 수 있다.

[0028] 실행되고 있는 코드가 목표 코어 상에 구현되지 않는 특징들을 사용하는 경우(결정 블록(64), "아니오" 레그), 프로세서 전력 상태 관리자(36)는 상태 변경을 수행하지 않을 수 있다. 일 실시예에서, 프로세서 전력 상태 관리자(36)는 상태 변경의 부재를 레지스터 또는 다른 소프트웨어-판독가능 위치에 기록할 수 있어서 소프트웨어가 상태 변경이 발생하지 않았다고 결정하게 할 수 있다. 상태 변경들이 수행되지 않을 때 다른 표시들(예를

들어, 인터럽트 또는 다른 시그널링 메커니즘)이 또한 사용될 수 있다. 일부 실시예들에서, 상태 변경을 방지하려는 시도는 수행되지 않을 수 있고, 결정 블록(64)은 생략될 수 있다. 대신에, 미지원되는 특징들이 코드가 목표 코어 상에서 실행되는 동안에 검출될 수 있다. 실행되고 있는 코드가 목표 코어 상에 구현되는 특징들만을 사용하는 경우(결정 블록(64), "예" 레그), 프로세서 전력 상태 관리자(36)는 새로운 PState가 매핑되는 코어로의 "코어 스왑"을 수행할 수 있다(블록(52)).

[0029] 새로운 PState가 활성 코어에 매핑되는 경우(결정 블록(50), "예" 레그), 활성 코어는 활성으로 유지될 수 있고, PState가 변경되는 동안에 실행이 계속될 수 있다. 새로운 PState가 현재 PState로부터의 증가인 경우(결정 블록(54), "예" 레그), 증가된 주파수를 지원하기 위해 공급 전압 크기가 먼저 증가될 수 있다. 따라서, 프로세서 전력 상태 관리자(36)는 전압 증가를 요청하고(블록(56)), 전압 증가가 완료되기를 기다릴 수 있다(결정 블록(58), "예" 레그). 프로세서 전력 상태 관리자(36)는 특정된 기간 동안 기다림으로써 전압 증가가 완료되었음을 결정할 수 있거나, 또는 전압 증가가 완료되는 때를 나타내는 통신을 수신할 수 있다. 일 실시예에서, 프로세서 전력 상태 관리자(36)는 전압 증가 요청을 다른 전력 관리자(예를 들어, 일 실시예에서, 도 9에 도시된 SOC 레벨 전력 관리자)에 전송할 수 있거나, 또는 전압을 공급하는 PMU에 직접 전압 요청을 전송할 수 있다. 전압 증가가 완료되면, 프로세서 전력 관리자(36)는 클록의 주파수를 증가시킬 수 있다(블록(60)). 반면에, 새로운 PState가 현재 PState로부터의 감소인 경우, 현재 공급 전압은 새로운(더 낮은) 주파수를 지원할 수 있다. 이와 같이(결정 블록(54), "아니오" 레그), 프로세서 전력 관리자(36)는 전압 변경이 완료되기를 기다리지 않고 클록 주파수를 업데이트하고 새로운 공급 전압을 요청할 수 있다(블록(62)).

[0030] 도 4는 코어 스왑을 수행하기 위한(도 3으로부터의 블록(52)) 프로세서 전력 관리자(36)의 동작의 일 실시예를 도시하는 흐름도이다. 블록들이 이해의 용이함을 위해 특정 순서로 도시되었지만, 다른 순서들도 사용될 수 있다. 블록들은 프로세서 전력 관리자(36) 내의 조합 로직에서 병렬로 수행될 수 있다. 블록들, 블록들의 조합들, 및/또는 흐름도 전체는 다수의 클록 사이클들에 걸쳐 파이프라인화될 수 있다. 프로세서 전력 상태 관리자(36)는 도 4에 도시된 동작을 구현하도록 구성될 수 있다.

[0031] 프로세서 전력 관리자(36)는 활성 코어를 "안전한" PState로 전이시킬 수 있다(블록(70)). 안전한 PState는 활성 코어 및 목표 코어 둘 모두가 올바르게 동작하는 상태일 수 있다. 이 컨텍스트에서, 목표 코어는 새로운 PState가 매핑되는 코어일 수 있다. 2개 초과의 코어가 있는 실시예들에서, 안전한 PState는 어떤 코어들이 활성 및 목표 코어들인지에 따라 상이할 수 있다. 안전한 PState는 PState 레지스터(36)에서 선택가능한 PState일 필요는 없다. 즉, 공급 전압과 주파수의 조합은, 코어들에 매핑되는 지원된 조합들 중 하나가 아닐 수 있다. 예를 들어, PCore는 안전한 PState에서의 공급 전압 크기가 주어질 때 더 높은 주파수에서 실행될 수 있다. 그러나, ECore는 주어진 공급 전압 크기로 더 높은 주파수에서 실행되지 못할 수 있다. 따라서, 안전한 PState는 현재 공급 전압 크기이지만 더 낮은 클록 주파수를 포함할 수 있다. 대안적으로, 목표 코어는 현재 공급 전압을 지원하지 않을 수 있으며, 안전한 PState는 상이한 공급 전압 크기 및 클록 주파수를 포함할 수 있다. 안전한 PState로 전이시키는 것은 도 3의 블록들(54, 56, 58, 60, 62)과 유사할 수 있다.

[0032] 일부 실시예들에서, 도 3 및 도 4의 동작은 하드웨어 회로로 구현될 수 있다. 다른 실시예들에서, 동작은 컴퓨터 액세스가능 저장 매체 상에 저장되고 프로세서들(32A 내지 32n)에 의해 실행되는 소프트웨어와 하드웨어의 조합으로 구현되거나, 또는 전적으로 소프트웨어로 구현될 수 있다.

[0033] 프로세서 전력 관리자(36)는 목표 코어에 전력을 공급할 수 있다(블록(72)). 예를 들어, 도 1의 실시예에서, 프로세서 전력 관리자(36)는 목표 코어로의 전력 스위치들을 닫아서, 목표 코어로 전력이 흐르게 할 수 있다. 목표 코어는 전력이 안정화된 후에 리셋될 수 있다. 일부 실시예들에서, 목표 코어는 리셋이 완료된 후에 초기화될 수 있다. 리셋(및 적용가능한 경우 초기화)이 완료되면(결정 블록(74), "예" 레그), 프로세서 전력 관리자(36)는 활성 코어로부터 목표 코어로 프로세서 컨텍스트의 이송을 개시할 수 있다(블록(76)). 일 실시예에서, 코어들은 프로세서 컨텍스트를 전송/수신하도록 구성된 회로를 포함할 수 있다. 다른 실시예에서, 회로는 프로세서 전력 관리자(36) 내에 있을 수 있다. 이전에 언급한 바와 같이, 코어들은 또한 컨텍스트 이송 동안 캐시들을 플러시하도록 구성될 수 있다. 컨텍스트 이송이 완료되면(결정 블록(78), "예" 레그), 프로세서 전력 관리자는 (이전) 활성 코어를 전력 차단할 수 있고 목표 코어는 활성 코어가 될 수 있다(블록(80)). 전력 차단하는 것은 예를 들어, 이전 활성 코어로의 전력 스위치들을 여는 것에 의해 수행될 수 있다. 프로세서 전력 관리자(36)는 활성 코어를 새로운 PState로 전이시킬 수 있다(블록(82)). 새로운 PState로 전이시키는 것은 도 3의 블록들(54, 56, 58, 60, 62)과 유사할 수 있다.

[0034] 도 5는 코드 실행 동안 주어진 프로세서(32A 내지 32n)(및 보다 상세하게는 활성 코어(40/42))의 동작의 일 실

시예를 도시하는 흐름도이다. 블록들이 이해의 용이함을 위해 특정 순서로 도시되었지만, 다른 순서들도 사용될 수 있다. 블록들은 프로세서(32A 내지 32n) 내의 조합 로직에서 병렬로 수행될 수 있다. 블록들, 블록들의 조합들, 및/또는 흐름도 전체는 다수의 클록 사이클들에 걸쳐 파이프라인화될 수 있다.

[0035] 코드 내의 각각의 명령어는 ISA 특징일 수 있고/있거나 하나 이상의 ISA 특징을 이용할 수 있다. 주어진 명령어에 사용된 ISA 특징들이 활성 코어에 의해 구현되는 경우(결정 블록(100), "예" 가지), 명령어는 정상적으로 처리될 수 있다(블록(102)). 반면에, 적어도 하나의 특징이 활성 코어에 의해 구현되지 않지만(결정 블록(100), "아니오" 레그) 다른 코어가 그 특징을 구현하는 경우(결정 블록(104), "예" 레그), 특징들을 구현하는 코어로 코어 스왑이 수행된다(블록(52)). 코어들 중 어느 것도 특징을 구현하지 않는 경우(결정 블록들(100, 104), "아니오" 레그들), "구현되지 않음(not implemented)" 예외가 취해져 소프트웨어가 오류를 핸들링하도록 할 수 있다(블록(106)).

[0036] 일반적으로, 도 5(및 이하에 논의되는 도 6)에 도시된 동작은 그것이 프로세서 파이프라인을 통해 처리될 때 각각의 명령어에 대해 수행될 수 있다. 다양한 특징들이 상이한 상태들에서 검출될 수 있다. 따라서, 도 5 및 도 6의 흐름도들은 실행되고 있는 코드 시퀀스 내의 각 명령어에 대해 병렬로 프로세서들(32A 내지 32n)에 의해 구현될 수 있다.

[0037] 도 6은 코드 실행 동안 주어진 프로세서(32A 내지 32n)(및 보다 상세하게는 활성 코어(40/42))의 동작의 다른 실시예이다. 도 5의 실시예와 유사하게, 도 6의 실시예는, 활성 코어가 코드에 의해 사용되는 ISA 특징들을 구현하는지 여부를 결정하고(결정 블록(100)), 만약 그렇다면 코드를 정상적으로 처리하고(블록(102)), 다른 코어가 특징을 구현하는지 여부를 결정하며(결정 블록(104)), 임의의 코어 상에서 구현되지 않는다면 구현되지 않음 예외를 취할 수 있다(블록(106)). 그러나, 이 실시예에서, 다른 코어가 특징들을 구현하는 경우(결정 블록(104)), 코어 스왑 예외가 취해질 수 있다(블록(108)). 코어 스왑 예외는 구현되지 않음 예외 및 코어들(40/42)에 의해 구현되는 다른 예외들과 상이할 수 있다. 코어 스왑 예외는 프로세서로 하여금, 이전에 언급된 코어 스왑(52)을 수행할 수 있는 코어 스왑 예외 핸들러를 실행하게 할 수 있다. 유사하게, 코어 스왑 예외 핸들러는, 일부 실시예들에서 코어 스왑(52)이 수행되는 다른 시간들에서 사용될 수 있다.

[0038] 도 7은 컴퓨터 액세스 가능 저장 매체(200)의 일 실시예의 블록도이다. 일반적으로 말하면, 컴퓨터 액세스가능 저장 매체는 명령어들 및/또는 데이터를 컴퓨터에 제공하기 위하여 사용하는 동안 컴퓨터에 의해 액세스가능한 임의의 저장 매체를 포함할 수 있다. 예를 들어, 컴퓨터 액세스가능 저장 매체는, 자기 또는 광학 매체, 예를 들어, 디스크(고정식 또는 탈착가능), 테이프, CD-ROM, DVD-ROM, CD-R, CD-RW, DVD-R, DVD-RW, 또는 블루레이(Blu-Ray)와 같은 저장 매체를 포함할 수 있다. 저장 매체는, RAM(예를 들어, 동기식 동적 RAM(synchronous dynamic RAM; SDRAM), 램버스 DRAM(RDRAM), 정적 RAM(SRAM) 등), ROM, 또는 플래시 메모리와 같은 휘발성 또는 비휘발성 메모리 매체를 추가로 포함할 수 있다. 저장 매체는 저장 매체가 명령어들/데이터를 제공하는 컴퓨터 내에 물리적으로 포함될 수 있다. 대안적으로, 저장 매체는 컴퓨터에 접속될 수 있다. 예를 들어, 저장 매체는 네트워크 부착형 저장장치와 같이, 네트워크 또는 무선 링크를 통해 컴퓨터에 접속될 수 있다. 저장 매체는 범용 시리얼 버스(USB)와 같은 주변장치 인터페이스를 통해 접속될 수 있다. 일반적으로, 컴퓨터 액세스가능 저장 매체(200)는 비일시적 방식으로 데이터를 저장할 수 있고, 본 맥락에서 비일시적이란 신호 상에서 명령어들/데이터를 전송하지 않는 것을 지칭할 수 있다. 예를 들어, 비일시적 저장장치는 휘발성(그리고 전력 차단에 응답하여 저장된 명령어들/데이터를 잃을 수 있음) 또는 비휘발성일 수 있다.

[0039] 도 7의 컴퓨터 액세스가능 저장 매체(200)는 코어 스왑 예외 핸들러(202)를 형성하는 코드를 저장할 수 있다. 코어 스왑 예외 핸들러(202)는, 프로세서(32A 내지 32n)에 의해 실행될 때, 코어 스왑 예외 핸들러에 대해 전술한 동작(예를 들어, 도 6의 블록(108) 및 도 4의 블록들)을 구현하는 명령어들을 포함할 수 있다. 캐리어 매체는 유선 또는 무선 전송과 같은 전송 매체뿐만 아니라 컴퓨터 액세스가능 저장 매체를 포함할 수 있다.

[0040] 도 8은 프로세서(32A)의 일 실시예의 보다 상세한 블록도이다. 도시된 실시예에서, PCore(40) 및 ECore(42)는 컨텍스트 상태 기계(90)의 인스턴스들(즉, 도 8의 90A 및 90B)을 포함하는 것으로 도시된다. 코어들(40, 42) 내의 상태 기계(90)의 구현들은 상이할 수 있지만, 그것들은 논리적으로 유사한 방식으로 동작할 수 있다. 일반적으로, 활성 코어 내의 상태 기계(90)는 레지스터 상태가 활성 코어에 의해 컨텍스트 버퍼(92)로 출력되게 할 수 있으며, 컨텍스트 버퍼(92)에는 상태 기계(90)가 결합된다. 상태 내의 레지스터들의 순서는 고정되어서, 수신하는 상태 기계는 단순히 데이터를 판독하고 그것을 올바른 레지스터들에 기입할 수 있다. 다른 구현예에서, 그 순서는 임의적일 수 있고 각각의 레지스터는 식별자를 할당받을 수 있으며, 식별자는 레지스터 콘텐츠와 함께 컨텍스트 버퍼(92)에 기입될 수 있으며 수신 코어 내에서 올바른 레지스터를 기입하기 위해 수신 상태 기

계에 의해 사용될 수 있다.

[0041] 상태 기계는 다양한 방식으로 구현될 수 있다: 고정된 기능 회로(예를 들어, 유한 상태 기계), 프로세서에 의해 실행되는 마이크로코드, 프로세서 전력 관리자(36)에서(예를 들어, 다양한 레지스터들을 이송하기 위해 코어들에 명령들을 전송하는 것) 등. 또한, 활성 프로세서 내의 상태 기계(90)는 전술한 바와 같이, 데이터 캐시(들)를 L2 캐시(34)로 플러시할 수 있다.

[0042] 컨텍스트 버퍼(92)는 컨텍스트 상태를 하나의 코어로부터 다른 코어로 캡처하기 위한 선입선출 버퍼(FIFO)일 수 있다. 컨텍스트 버퍼(92)는 신축성(elasticity)을 제공하고, 클록 도메인 크로싱(clock domain crossing)들을 처리하는 것 등을 수행할 수 있다. 일 실시예에서, 컨텍스트 버퍼(92)는 프로세서 전력 관리자(36)의 일부일 수 있으며, 따라서 도 8에서 점선으로 도시된다. 상태 기계들(90)은 또한 다른 실시예에서 프로세서 전력 관리자(36)에서 구현될 수 있다. 이러한 실시예들에서, 프로세서 전력 관리자(36)는 코어들(40, 42) 내의 레지스터 상태에의 액세스를 가질 수 있거나, 또는 레지스터 판독들/기입들을 수행하여 레지스터 상태들의 전송을 수행하기 위해 명령어들이 실행되게 할 수 있다.

[0043] 도 9는 메모리(12)에 결합된 SOC(10)의 일 실시예의 블록도이다. 이름에서 암시되는 바와 같이, SOC(10)의 컴포넌트들은 집적회로 "칩"처럼 단일의 반도체 기판 상에 집적될 수 있다. 일부 실시예들에서, 컴포넌트들은 시스템 내의 둘 이상의 별개의 칩 상에 구현될 수 있다. 그러나, SOC(10)는 본 명세서에서 예로서 사용될 것이다. 예시된 실시예에서, SOC(10)의 컴포넌트들은 중앙 처리 장치(CPU) 컴플렉스(14)(이는 도 1에 도시된 프로세서 클러스터(30)에 의해 구현될 수 있음), 주변 컴포넌트들(18A, 18B)(더 간단하게, "주변장치들"(18)), 메모리 제어기(22), SOC 전력 관리자(PMGR)(16), 및 통신 패브릭(communication fabric)(27)을 포함한다. 컴포넌트들(14, 16, 18A, 18B, 22)은 모두 통신 패브릭(27)에 결합될 수 있다. 메모리 제어기(22)는 사용 중에 메모리(12)에 결합될 수 있다.

[0044] 메모리 제어기(22)는 일반적으로, SOC(10)의 다른 컴포넌트들로부터 메모리 동작들을 수신하고, 메모리(12)에 액세스하여 메모리 동작들을 완료하기 위한 회로를 포함할 수 있다. 메모리 제어기(22)는 임의의 유형의 메모리(12)에 액세스하도록 구성될 수 있다. 예를 들어, 메모리(12)는 SRAM(static random access memory), DRAM(dynamic RAM), 예컨대 더블 데이터 레이트(DDR, DDR2, DDR3, DDR4 등) DRAM을 포함하는 SDRAM(synchronous DRAM)일 수 있다. 저전력/모바일 버전들의 DDR DRAM(예를 들어, LPDDR, mDDR 등)이 지원될 수 있다. 메모리 제어기(22)는, 동작들을 순서화하고(그리고 잠재적으로 재순서화하고) 동작들을 메모리(12)에 제시하는, 메모리 동작들을 위한 큐(queue)들을 포함할 수 있다. 메모리 제어기(22)는 메모리로의 기입을 기다리는 기입 데이터 및 메모리 동작의 소스로의 복귀를 기다리는 판독 데이터를 저장하는 데이터 버퍼들을 추가로 포함할 수 있다. 일부 실시예들에서, 메모리 제어기(22)는 최근에 액세스된 메모리 데이터를 저장하는 메모리 캐시를 포함할 수 있다. SOC 구현예들에서, 예를 들어, 메모리 캐시는, 곧 다시 액세스될 것으로 예상되는 경우에 메모리(12)로부터의 데이터의 재-액세스를 피함으로써, SOC에서의 전력 소모를 감소시킬 수 있다. 일부 경우들에서, 메모리 캐시는 또한, 소정의 컴포넌트들만을 보조하는 전용 캐시(private cache)들 그 예로서 프로세서들 내의 L2 캐시 또는 캐시들과 상반되는, 시스템 캐시로서 지칭될 수 있다. 추가적으로, 일부 실시예들에서, 시스템 캐시는 메모리 제어기(22) 내에 위치될 필요가 없다.

[0045] 주변장치들(18A, 18B)은 SOC(10)에 포함된 추가 하드웨어 기능성의 임의의 세트일 수 있다. 예를 들어, 주변장치들(18A, 18B)은 비디오 주변장치들, 예컨대 카메라 또는 다른 이미지 센서로부터의 이미지 캡처 데이터를 처리하도록 구성된 이미지 신호 프로세서, 하나 이상의 디스플레이 디바이스 상에 비디오 데이터를 표시하도록 구성된 디스플레이 제어기들, 그래픽 처리 유닛(GPU)들, 비디오 인코더/디코더들, 스케일러(scaler)들, 로테이터(rotator)들, 블렌더(blender)들 등을 포함할 수 있다. 주변장치들은 오디오 주변장치들, 예컨대 마이크로폰, 스피커, 마이크로폰 및 스피커에 대한 인터페이스, 오디오 프로세서, 디지털 신호 프로세서, 믹서 등을 포함할 수 있다. 주변장치들은 SOC(10)의 외부에 있는 다양한 인터페이스들(예를 들어 주변장치(18B))에 대한 인터페이스 제어기들을 포함할 수 있고, 인터페이스들에는 USB(Universal Serial Bus), PCIe(PCI Express)를 포함하는 PCI(peripheral component interconnect), 직렬 및 병렬 포트 등이 포함된다. 주변장치들은 네트워킹 주변장치들, 예컨대 MAC(media access controller)들을 포함할 수 있다. 하드웨어의 임의의 세트가 포함될 수 있다.

[0046] 통신 패브릭(27)은 SOC(10)의 컴포넌트들 간의 통신을 위한 임의의 통신 상호접속부 및 프로토콜일 수 있다. 통신 패브릭(27)은 공유 버스 구성, 크로스 바(cross bar) 구성, 및 브릿지를 이용한 계층적 버스를 포함하는 버스-기반일 수 있다. 또한, 통신 패브릭(27)은 패킷-기반일 수 있고, 브릿지를 이용한 계층, 크로스 바, 포인

트-투-포인트(point-to-point), 또는 다른 상호접속부일 수 있다.

[0047] SOC PMGR(16)은 시스템 내의 PMU로부터 요청된 공급 전압 크기들을 제어하도록 구성될 수 있다. SOC(10)를 위해 PMU에 의해 생성된 다수의 공급 전압들이 있을 수 있다. 예를 들어, $V_P$ 전압은 CPU 컴플렉스(14) 내의 프로세서들(32A 내지 32n)에 대해 생성될 수 있고, $V_{SOC}$ 전압은 SOC(10) 내의 다른 컴포넌트들에 대해 생성될 수 있다. 일 실시예에서, $V_{SOC}$는 메모리 제어기(22), 주변장치들(18), SOC PMGR(16), 및 SOC(10)의 다른 컴포넌트들을 보조할 수 있고, 전력 도메인들에 기초하여 전력 게이팅이 채용될 수 있다. 일부 실시예들에서, SOC(10)의 나머지에 대한 다수의 공급 전압들이 있을 수 있다. 일부 실시예들에서, CPU 컴플렉스(14) 및/또는 SOC(10) 내의 다양한 메모리 어레이들에 대한 메모리 공급 전압이 또한 있을 수 있다. 메모리 공급 전압은 로직 회로에 공급되는 전압(예를 들어, $V_P$ 또는 $V_{SOC}$)과 함께 사용될 수 있으며, 이는 강력한 메모리 동작을 보장하기 위해 요구되는 것보다 더 낮은 전압 크기를 가질 수 있다. SOC PMGR(16)은 직접적인 소프트웨어 제어를 받을 수 있고/있거나(예를 들어, 소프트웨어가 컴포넌트들의 전력 공급 및/또는 전력 차단을 직접 요청할 수 있음), SOC(10)를 모니터링하고 다양한 컴포넌트들이 언제 전력 공급되거나 전력 차단될지를 결정하도록 구성될 수 있다. CPU 컴플렉스(14)에 대해, $V_P$에 대한 전압 요청들이 SOC PMGR(16)에 제공될 수 있으며, 이것은 요청들을 PMU로 전달하여 공급 전압 크기들의 변경을 초래할 수 있다.

[0048] 일반적으로, 컴포넌트는 전력 온되거나 전력 오프되는 것으로 지칭될 수 있다. 컴포넌트는, 공급 전압을 수신하고 있어서 설계된 대로 동작할 수 있는 경우, 전력 온될 수 있다. 컴포넌트가 전력 오프되는 경우, 그것은 공급 전압을 수신하고 있지 않으며 동작중이 아니다. 컴포넌트는 또한 그것이 전력 온되는 경우 전력이 공급되는 것으로 지칭될 수 있고, 그것이 전력 오프되는 경우에는 전력 차단된 것으로 지칭될 수 있다. 컴포넌트에 전력을 공급하는 것은 전력 오프된 컴포넌트에 공급 전압을 공급하는 것을 지칭할 수 있고, 컴포넌트를 전력 차단하는 것은 컴포넌트로의 공급 전압의 공급을 종료하는 것을 지칭할 수 있다. 유사하게, 임의의 서브컴포넌트 및/또는 SOC(10)는 전체로서 전력 공급/차단되는 것 등으로 지칭될 수 있다. 컴포넌트는, SOC(10) 내의 특정된 기능을 제공하고 SOC(10)의 나머지로의 특정 인터페이스를 갖는 미리정의된 회로 블록일 수 있다. 따라서, 주변장치들(18A, 18B), CPU 컴플렉스(14), 메모리 제어기(22), 및 SOC PMGR(16) 각각은 컴포넌트의 예들일 수 있다.

[0049] SOC(10)의 컴포넌트들의 개수(및 CPU 컴플렉스(14) 내에서와 같은, 도 1에 도시된 컴포넌트들에 대한 서브컴포넌트들의 개수)는 실시예마다 다를 수 있다는 것에 유의한다. 도 1에 도시된 개수보다 많거나 적은 각각의 컴포넌트/서브컴포넌트가 있을 수 있다.

[0050] 다음으로 도 10을 참조하면, 시스템(150)의 일 실시예의 블록도가 도시된다. 예시되는 실시예에서, 시스템(150)은 하나 이상의 주변장치(154) 및 외부 메모리(12)에 결합된 SOC(10)의 적어도 하나의 인스턴스를 포함한다. SOC(10)에 공급 전압들을 공급할 뿐만 아니라, 메모리(12) 및/또는 주변장치들(154)에 하나 이상의 공급 전압을 공급하는 PMU(156)가 제공된다. 일부 실시예들에서, SOC(10)의 둘 이상의 인스턴스가 포함될 수 있다(그리고 둘 이상의 메모리(12) 또한 포함될 수 있다).

[0051] PMU(156)는 일반적으로 공급 전압들을 생성하고, SOC(10), 메모리(12), 다양한 오프-칩 주변장치 컴포넌트들(154), 예컨대 디스플레이 디바이스들, 이미지 센서들, 사용자 인터페이스 디바이스들 등과 같은 시스템의 다른 컴포넌트들로 그러한 공급 전압들을 제공하는 회로를 포함할 수 있다. 따라서, PMU(156)는 프로그래밍 가능한 전압 레귤레이터들, 전압 요청들을 수신하기 위해 SOC(10)에 그리고 보다 상세하게는 SOC PMGR(16)에 인터페이스하는 로직 등을 포함할 수 있다.

[0052] 주변장치들(154)은 시스템(150)의 유형에 따라, 임의의 원하는 회로를 포함할 수 있다. 예를 들어, 일 실시예에서, 시스템(150)은 모바일 디바이스(예컨대, 개인용 휴대 단말기(PDA), 스마트 폰 등)일 수 있으며, 주변장치들(154)은 wifi, 블루투스, 셀룰러, 글로벌 포지셔닝 시스템 등과 같은 다양한 유형의 무선 통신용 디바이스들을 포함할 수 있다. 또한, 주변장치들(154)은 RAM 저장장치, 솔리드 스테이트 저장장치 또는 디스크 저장장치를 포함한, 추가 저장장치를 포함할 수 있다. 주변장치들(154)은 터치 디스플레이 스크린 또는 멀티터치 디스플레이 스크린을 포함하는 디스플레이 스크린, 키보드 또는 다른 입력 디바이스들, 마이크로폰, 스피커 등과 같은 사용자 인터페이스 디바이스들을 포함할 수 있다. 다른 실시예들에서, 시스템(150)은 임의의 유형의 컴퓨팅 시스템(예를 들어, 데스크톱 개인용 컴퓨터, 랩톱, 워크스테이션, 넷톱 등)일 수 있다.

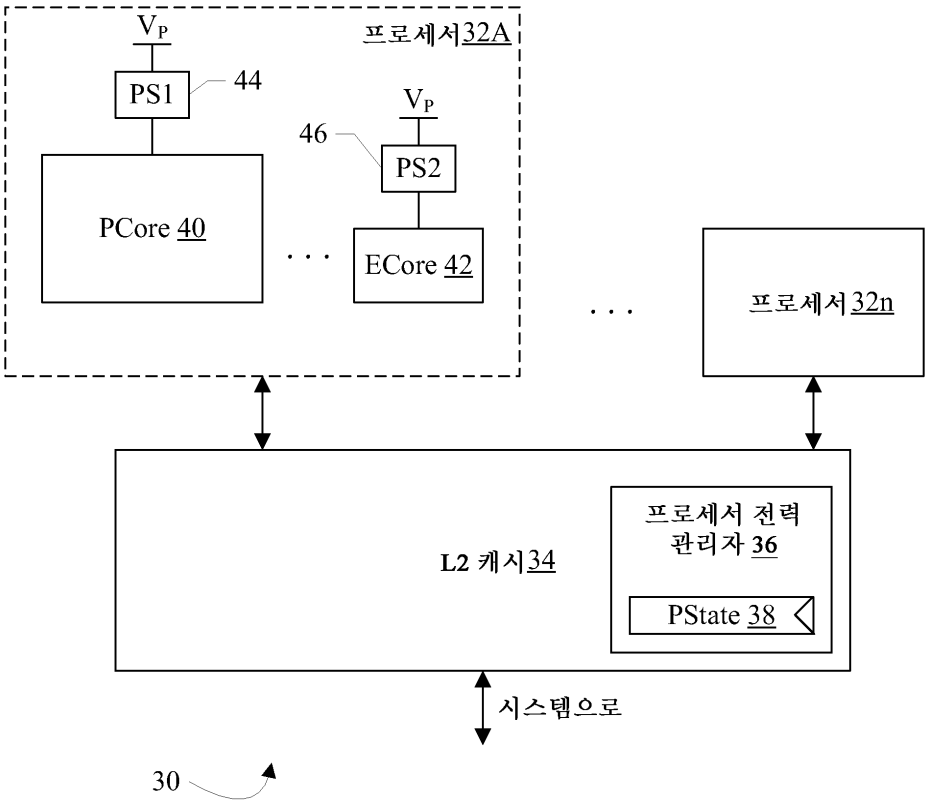
[0053] 외부 메모리(12)는 임의의 유형의 메모리를 포함할 수 있다. 예를 들어, 외부 메모리(12)는 SRAM, 동적 RAM(DRAM), 예컨대 동기 DRAM(SDRAM), 더블 데이터 레이트(DDR, DDR2, DDR3 등) SDRAM, 램버스 DRAM, 저전력

버전들의 DDR DRAM(예컨대, LPDDR, mDDR 등) 등일 수 있다. 외부 메모리(12)는 메모리 디바이스들이 장착되는 하나 이상의 메모리 모듈, 예컨대 단일 인라인 메모리 모듈(single inline memory module; SIMM), 듀얼 인라인 메모리 모듈(dual inline memory module; DIMM) 등을 포함할 수 있다. 대안적으로, 외부 메모리(12)는 칩-온-칩 또는 패키지-온-패키지 구현으로 SOC(10) 상에 장착되는 하나 이상의 메모리 디바이스를 포함할 수 있다.

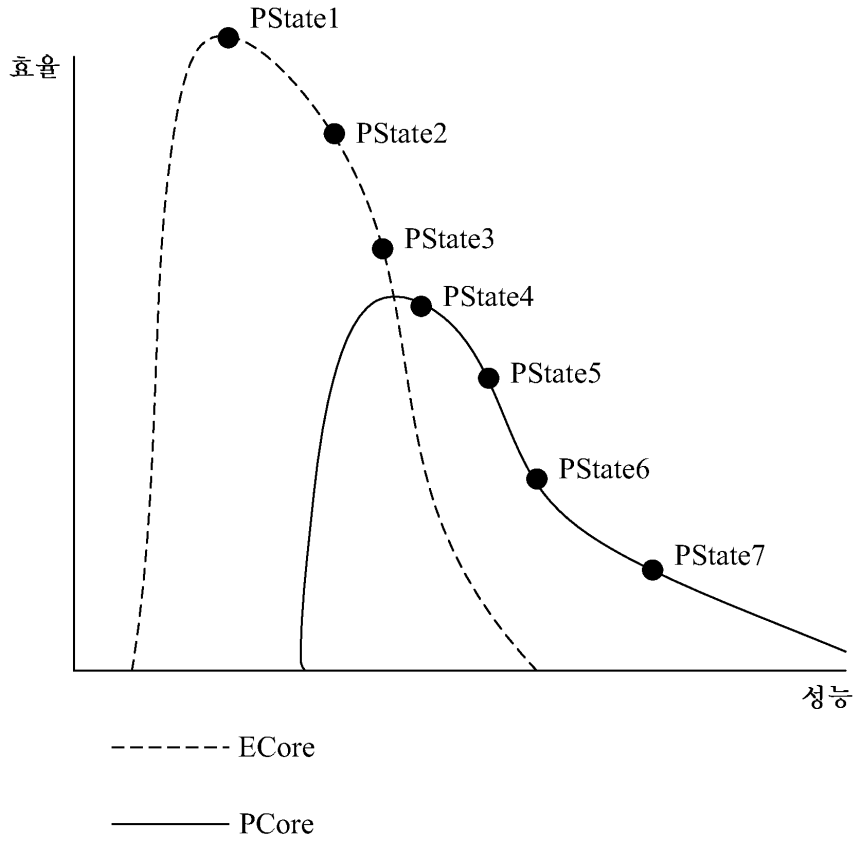
[0054] 상기의 개시내용이 완전히 이해된다면, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 있어서 다수의 변형들 및 수정들이 명백해질 것이다. 하기의 청구범위는 모든 그러한 변형들 및 수정들을 포괄하는 것으로 해석되는 것으로 의도된다.

## 도면

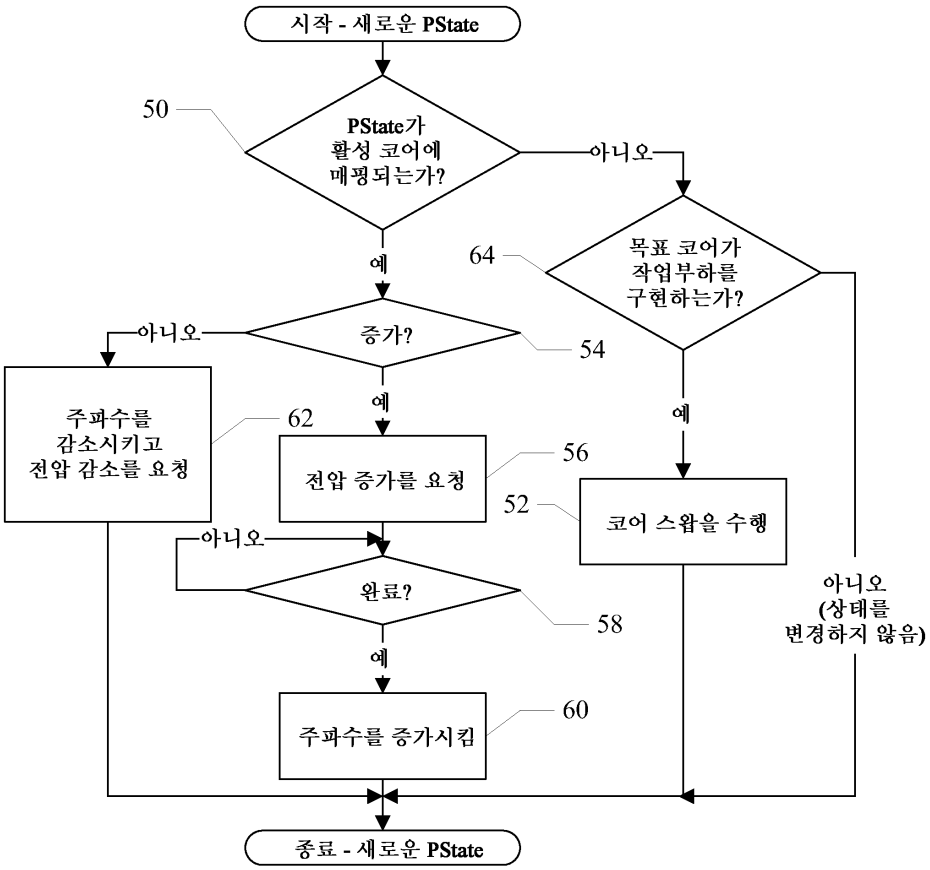
### 도면1



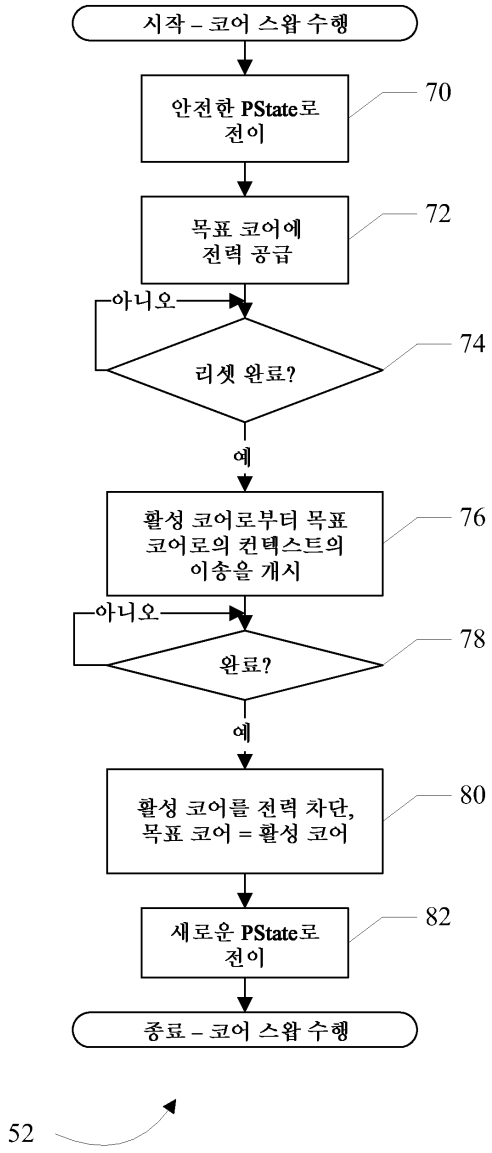
도면2


PState1
PState2
PState3
PState4
PState5
PState6
PState7
효율
성능
ECore
PCore

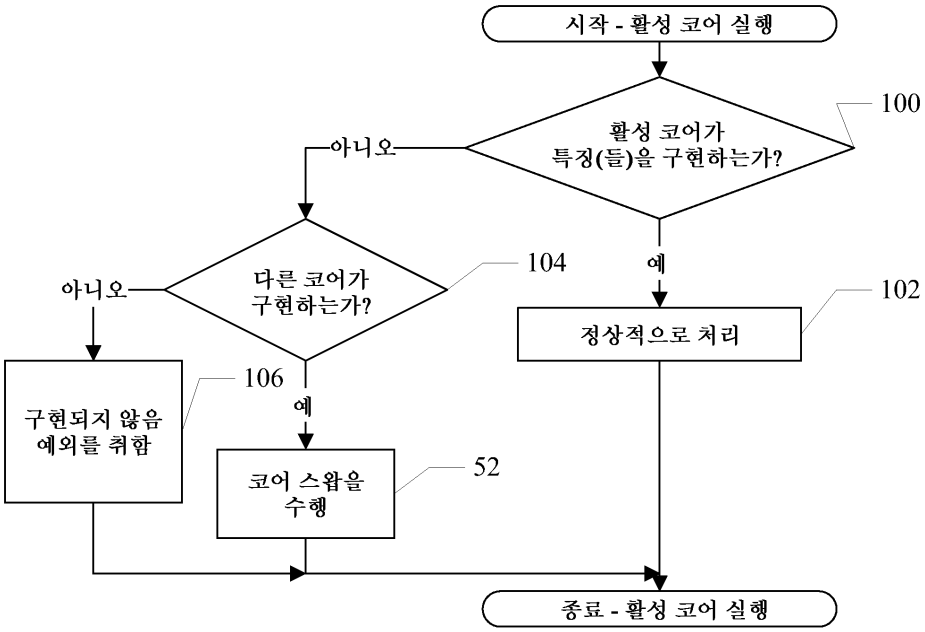
도면3


시작 - 새로운 PState
50
PState가 활성 코어에 매핑되는가?
아니오
예
64
목표 코어가 작업부하를 구현하는가?
증가?
54
아니오
예
주파수를 감소시키고 전압 감소를 요청
62
전압 증가를 요청
56
52
코어 스왑을 수행
아니오
완료?
58
예
아니오 (상태를 변경하지 않음)
주파수를 증가시킴
60
종료 - 새로운 PState

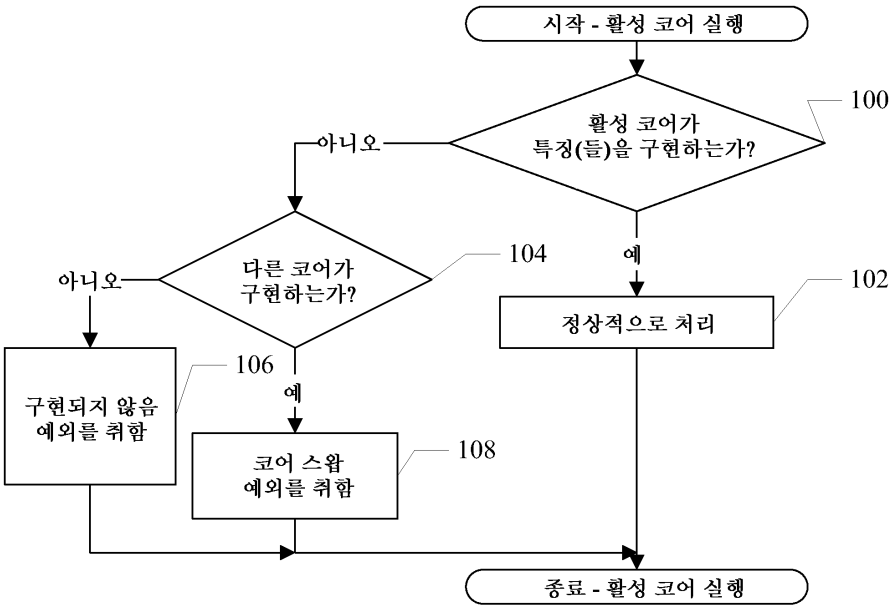
도면4


시작 – 코어 스왑 수행
안전한 PState로 전이
70
목표 코어에 전력 공급
72
아니오
리셋 완료?
74
예
활성 코어로부터 목표 코어로의 컨텍스트의 이송을 개시
76
아니오
완료?
78
예
활성 코어를 전력 차단, 목표 코어 = 활성 코어
80
새로운 PState로 전이
82
종료 – 코어 스왑 수행
52

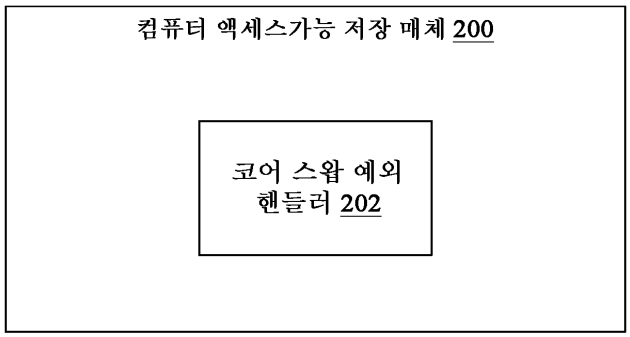
도면5


시작 - 활성 코어 실행
활성 코어가
특징(들)을 구현하는가?
100
아니오
다른 코어가
구현하는가?
104
예
아니오
정상적으로 처리
102
구현되지 않음
예외를 취함
106
예
코어 스왑을
수행
52
종료 - 활성 코어 실행

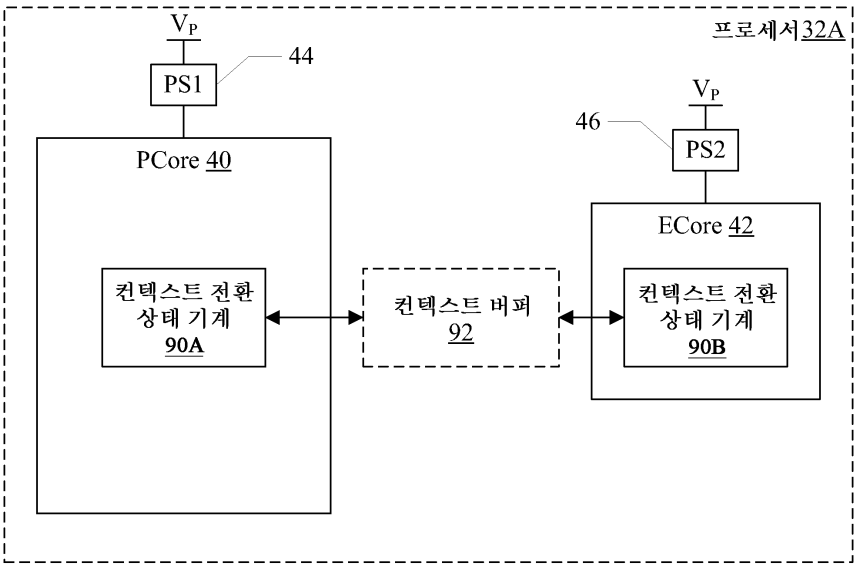
도면6


시작 - 활성 코어 실행
활성 코어가
특징(들)을 구현하는가?
100
아니오
다른 코어가
구현하는가?
104
예
아니오
정상적으로 처리
102
구현되지 않음
예외를 취함
106
예
코어 스왑
예외를 취함
108
종료 - 활성 코어 실행

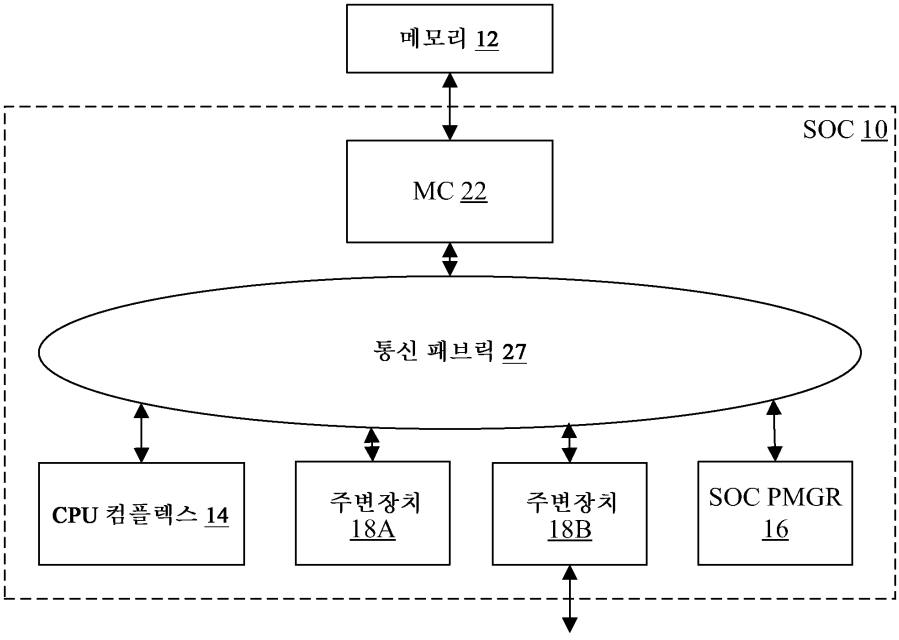
도면7


컴퓨터 액세스가능 저장 매체 200
코어 스왑 예외 핸들러 202

도면8


프로세서 32A
VP
PS1
44
PCore 40
컨텍스트 전환 상태 기계 90A
컨텍스트 버퍼 92
VP
46
PS2
ECore 42
컨텍스트 전환 상태 기계 90B

도면9


메모리 12
SOC 10
MC 22
통신 패브릭 27
CPU 컴플렉스 14
주변장치 18A
주변장치 18B
SOC PMGR 16

도면10