(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2024년07월01일
(11) 등록번호 10-2679562
(24) 등록일자 2024년06월25일

(51) 국제특허분류(Int. Cl.)
G06F 1/24 (2017.01) *G06F 1/3209* (2019.01)
G06F 1/3234 (2019.01) *G06F 1/324* (2019.01)
G06F 1/3296 (2019.01) *G06F 11/14* (2006.01)
G06F 15/78 (2006.01) *G06F 9/54* (2018.01)
G06N 5/04 (2023.01) *G09C 1/00* (2006.01)
H04L 9/08 (2006.01)
(52) CPC특허분류
G06F 1/24 (2018.05)
G06F 1/3209 (2019.01)
(21) 출원번호 10-2020-7026420
(22) 출원일자(국제) 2019년03월28일
심사청구일자 2021년12월30일
(85) 번역문제출일자 2020년09월14일
(65) 공개번호 10-2020-0139673
(43) 공개일자 2020년12월14일
(86) 국제출원번호 PCT/US2019/024586
(87) 국제공개번호 WO 2019/191431
국제공개일자 2019년10월03일
(30) 우선권주장
62/650,589 2018년03월30일 미국(US)
62/691,117 2018년06월28일 미국(US)
(56) 선행기술조사문헌
KR1020140079417 A*
(뒷면에 계속)
전체 청구항 수 : 총 52 항

(73) 특허권자
구글 엘엘씨
미국 캘리포니아 마운틴 뷰 엠피시어터 파크웨이 1600 (우:94043)
(72) 발명자
드세이 샤일렌드라
인도, 아메다바드 380058, 니어 에스피 링 로드 크로싱스, 샬린 레지던시 52
피어스 마크
미국, 캘리포니아 94131, 샌 프란시스코, 28쓰 스트리트 372
(뒷면에 계속)
(74) 대리인
박장원

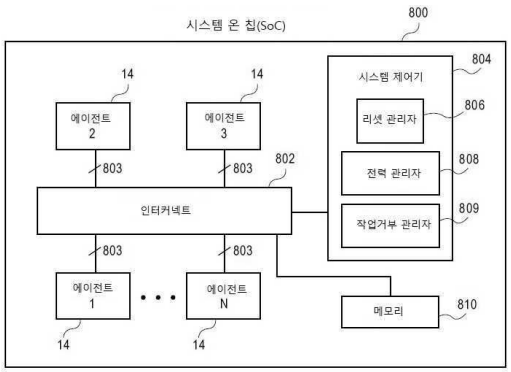
심사관 : 신현상

(54) 발명의 명칭 **시스템 온 칩(SoC) 에이전트 리셋 및 전력 관리를 위한 프로토콜 레벨 제어**

(57) 요 약

시스템 온 칩(SoC) 상의 IP 에이전트의 리셋 및 전력 관리를 일관되게 구현하기 위한 시스템. IP 에이전트가 리셋을 겪을 때, 링크를 통해 인터커넥트와 각각의 IP 에이전트 간 개별 협상이 발생한다. 각각의 IP 에이전트가 다른 IP 에이전트의 타이밍에 독립적으로 자신의 고유 시간 스케줄에 따라 리셋으로부터 벗어날 수 있다. 인터커넥트가 리셋 전, 또는 파워-다운 모드일 때, 또는 오작동 중일 때 동작 불능인 임의의 IP 에이전트에 대한 프록시로서 구성될 수 있다.

대 표 도 - 도8



(52) CPC특허분류
G06F 1/324 (2019.01)
G06F 1/3243 (2013.01)
G06F 1/3296 (2019.01)
G06F 11/1402 (2013.01)
G06F 15/7807 (2013.01)
G06F 9/542 (2013.01)
G06N 5/043 (2023.01)
G09C 1/00 (2013.01)
H04L 9/085 (2013.01)

(72) 발명자
야인 아밋
인도, 아메다바드 380008, 마니나가 이스트, 고르카 투바, 니어 비제이 파크 소사이어티, 쉬레 랑 하이츠, 에이105
바트 루툴
인도, 아메다바드 380061, 가틀로디야, 알.씨. 테크니컬 칼리지 로드, 포람 파크 소사이어티 30
파텔 제이민
인도, 아메다바드 382418, 바스트랄 로드, 마하데브나가, 아제이 비브하그-5, 아르티 소사이어티, 아/89

(56) 선행기술조사문헌
KR1020160086421 A*
US20150026494 A1*
US20160283428 A1*
US20130262918 A1*
KR1020160066188 A*
*는 심사관에 의하여 인용된 문헌

명 세 서

청구범위

청구항 1

시스템 온 칩(SoC)으로서,

인터커넥트,

SoC 상에 제공되며 상기 인터커넥트와 통신하도록 배열된 IP 에이전트, 및

IP 에이전트의 리셋을 개시하기 위한 리셋 관리자(reset manager) - 리셋은 IP 에이전트가 트랜잭션 준비 상태가 되게 하는 인터커넥트와 IP 에이전트 간 협상을 포함함 - 를 포함하고,

상기 협상에서 인터커넥트와 IP 에이전트는 그 인터커넥트와 IP 에이전트를 연결하는 링크를 통해 클럭 사이클당 전송될 수 있는 데이터의 양에 관한 정보를 교환하는, SoC.

청구항 2

제1항에 있어서, 협상은 인터커넥트가 IP 에이전트에 대한 적어도 하나의 질의를 생성하는 것을 포함하며, 적어도 하나의 질의는 IP 에이전트의 "어웨이크(awake)" 상태를 요청하는, SoC.

청구항 3

제1항에 있어서, 협상은 인터커넥트가 IP 에이전트의 "어웨이크" 상태를 각각 요청하는 IP 에이전트에 대한 복수의 질의를 생성하는 것을 포함하며, 복수의 질의는 각각 주기적 간격으로 전송되는, SoC.

청구항 4

제1항에 있어서, 협상은 IP 에이전트의 "어웨이크" 상태를 요청하는 인터커넥트로부터의 질의에 응답하여 IP 에이전트가 긍정 응답을 생성하는 것을 포함하며, 긍정 응답은 IP 에이전트가 깨어 나고(awaken) 트랜잭션 준비 상태인 때 생성되는, SoC.

청구항 5

제1항에 있어서, 협상은 인터커넥트와 IP 에이전트에서 각각 이용 가능한 크레딧의 교환을 포함하는, SoC.

청구항 6

제1항에 있어서, 협상은 인터커넥트와 IP 에이전트가 보안 정보를 공유하는 것을 포함하는, SoC.

청구항 7

제1항에 있어서, 협상은 인터커넥트가 인터커넥트와 IP 에이전트를 연결하는 링크와 연관된 복수의 가상 채널을 공유하는 것을 포함하는, SoC.

청구항 8

제1항에 있어서, 협상이 완료된 후, IP 에이전트의 가용성이 시스템 제어기를 포함하는 인터커넥트에 연결된 요소에 알려지는, SoC.

청구항 9

제1항에 있어서, 인터커넥트와 IP 에이전트 사이에 연결된 링크를 더 포함하며, 협상은 링크를 통해 인터커넥트와 IP 에이전트 간에 실시되는, SoC.

청구항 10

제1항에 있어서, 복수의 가상 채널은 링크와 연관되는, SoC.

청구항 11

제1항에 있어서, 협상은 인터커넥트와 IP 에이전트 간 링크를 통해 전송되는 링크-레벨 정보를 사용하여 수행되는, SoC.

청구항 12

제11항에 있어서, 링크-레벨 정보는 인터커넥트와 IP 간 링크를 통해 전송되는 하나 이상의 제어 패킷으로 전달되는, SoC.

청구항 13

제12항에 있어서, 하나 이상의 제어 패킷은 인터커넥트와 IP 에이전트 간 링크를 통해 전송되고, 다른 목적지로 전달되지 않는, SoC.

청구항 14

제1항에 있어서, 리셋 관리자는 리셋 트리거 이벤트에 응답하여 IP 에이전트의 리셋을 개시하는, SoC.

청구항 15

제14항에 있어서, 리셋 트리거 이벤트는

SoC의 파워 업,

IP 에이전트가 오작동함, 및

IP 에이전트가 저전력 모드에서 벗어남

중 하나를 포함하는, SoC.

청구항 16

제1항에 있어서, 인터커넥트와 IP 에이전트 간 링크를 휴지(quiescent) 상태로 만들기 위한 휴지 관리자를 더 포함하는, SoC.

청구항 17

제16항에 있어서, 휴지 관리자는 IP 에이전트에 의해 생성된 모든 미결 트랜잭션이 완료된 후 링크를 휴지 상태로 만들도록 구성되는, SoC.

청구항 18

제17항에 있어서, 휴지 관리자는 미결 트랜잭션이 완료되기를 기다리는 동안 IP 에이전트가 새로운 트랜잭션을 생성하는 것을 방지하도록 구성되는, SoC.

청구항 19

제17항에 있어서, 휴지 관리자는 리셋 관리자가 IP 에이전트의 리셋을 개시하기 전에 링크를 휴지 상태로 만들도록 구성되는, SoC.

청구항 20

시스템 온 칩(SoC)으로서,

인터커넥트,

복수의 IP 에이전트, 및

복수의 IP 에이전트의 리셋을 구현하도록 배열된 리셋 관리자 - 복수의 IP 에이전트의 각각의 에이전트의 리셋은 인터커넥트와 각각의 IP 에이전트 각각 간 개별 협상을 포함하며, 개별 협상으로 인해 각 IP 에이전트는 각각 트랜잭션 준비 상태가 됨 - 를 포함하고,

상기 개별 협상에서 인터커넥트와 복수의 IP 에이전트 각각은 인터커넥트와 IP 에이전트를 연결하는 링크를 통해 클록 사이클 단위로 전송될 수 있는 데이터의 양에 관한 정보를 교환하는, SoC.

청구항 21

제20항에 있어서, 복수의 IP 에이전트의 각각의 IP 에이전트는 자신 고유의 리셋 루틴을 이용하는, SoC.

청구항 22

제20항에 있어서, 복수의 IP 에이전트의 각각은 자신 고유의 시간 스케줄에 따라 트랜잭션 준비 상태가 되는, SoC.

청구항 23

제20항에 있어서, 복수의 IP 에이전트 각각은 다른 IP 에이전트와 독립적으로 트랜잭션 준비 상태가 되고 모든 IP 에이전트가 동시에 트랜잭션 준비 상태가 될 필요는 없는, SoC.

청구항 24

제20항에 있어서, 복수의 IP 기능 각각에 대한 개별 협상은 IP 에이전트가 상이한지 여부와 관계없이 동일한, SoC.

청구항 25

제20항에 있어서, 인터커넥트는 적어도 하나의 IP 에이전트가 트랜잭션 준비 상태가 되기 전에 복수의 IP 에이전트 중 적어도 하나를 대리하여 프록시로서 동작하도록 더 구성되는, SoC.

청구항 26

제25항에 있어서, 인터커넥트는 적어도 하나의 IP 에이전트로 트랜잭션을 전송하려 시도하는 소스에 예외 메시지를 생성함으로써, 트랜잭션 준비 상태가 되기 전에 적어도 하나의 IP 에이전트에 대한 프록시로서 동작하는, SoC.

청구항 27

시스템 온 칩(SoC)으로서,

복수의 IP 에이전트, 및

복수의 IP 에이전트와 통신하는 인터커넥트 - 인터커넥트는 적어도 하나의 IP 에이전트가 트랜잭션 준비 상태가 아닐 때 복수의 IP 에이전트 중 적어도 하나의 IP 에이전트를 대리하는 프록시로서 동작하도록 배열됨 - 를 포함하고,

복수의 IP 에이전트 각각은 다른 IP 에이전트가 동시에 트랜잭션 준비 상태가 될 필요 없이 링크를 통해 인터커넥트와 각 IP 에이전트 사이에 수행되는 협상에 따라 독립적으로 트랜잭션 준비 상태가 되는, SoC.

청구항 28

제27항에 있어서, 적어도 하나의 IP 에이전트는 리셋이 진행될 때 트랜잭션 준비 상태가 아닌, SoC.

청구항 29

제27항에 있어서, 적어도 하나의 IP 에이전트는 파워 다운 모드에 있는 동안 트랜잭션 준비 상태가 아닌, SoC.

청구항 30

제27항에 있어서, 적어도 하나의 IP 에이전트는 오작동할 때 트랜잭션 준비 상태가 아닌, SoC.

청구항 31

제27항에 있어서, 인터커넥트는,

적어도 하나의 IP 기능이 트랜잭션 준비 상태가 아닐 때를 확인하고,

트랜잭션 준비 상태가 아닌 동안 소스가 트랜잭션을 적어도 하나의 IP 에이전트로 전송하는지 여부를 확인하며,

적어도 하나의 IP 에이전트가 이용 가능하지 않음을 소스로 통지하는 예외 메시지를 소스에게 전송함으로써,

적어도 하나의 IP 에이전트를 대리하는 프록시로서 동작하는, SoC.

청구항 32

제27항에 있어서, 소스는 예외 메시지에 응답하여 전송되었으나 목적지로 전달되지 않는 트랜잭션에 대해 자신 고유의 응답 절차를 거치는, SoC.

청구항 33

제27항에 있어서, 복수의 IP 에이전트의 리셋을 각각 개시하기 위한 리셋 관리자를 더 포함하는, SoC.

청구항 34

제27항에 있어서, 상기 인터커넥트는 복수의 IP 에이전트의 각각의 IP 에이전트가 각자 리셋될 때 복수의 IP 에이전트의 각각의 IP 에이전트와의 개별 협상에 연루되도록 더 구성되는, SoC.

청구항 35

제27항에 있어서, 복수의 IP 에이전트 중 둘 이상의 IP 에이전트의 리셋 후, 둘 이상의 IP 에이전트 각각이 개별적으로 리셋되도록 배열되고 다른 IP 에이전트 각각에 독립적으로 트랜잭션 준비 상태가 되는, SoC.

청구항 36

삭제

청구항 37

제27항에 있어서, 시스템 제어기를 더 포함하며, 적어도 하나의 IP 에이전트가 트랜잭션 준비 상태가 될 때 적어도 하나의 IP 에이전트가 시스템 제어기에 의해 가시적이 되는, SoC.

청구항 38

시스템 온 칩으로서,

복수의 IP 에이전트,

인터커넥트,

복수의 IP 에이전트를 인터커넥트와 연결하는 하나 이상의 링크, 및

복수의 IP 에이전트의 적어도 하나를 선택적으로 저전력 상태로 만듦으로써 SoC에 의한 전력 소비를 적어도 부분적으로 관리하기 위한 전력 관리자를 포함하며,

인터커넥트는 (a) 저전력 상태에 있을 때 및 (b) 트랜잭션을 처리할 수 없을 때 적어도 하나의 IP 에이전트를 대리하는 프록시로서 동작하고, 인터커넥트는 처리되지 않은 트랜잭션의 소스로 예외 메시지를 전송하여 해당 트랜잭션의 소스가 적어도 하나의 IP 에이전트로부터 응답을 수신하지 못해 발생하는 끊김(hang up) 상황을 방지함으로써 프록시로서 동작하는, SoC.

청구항 39

제38항에 있어서, 적어도 하나의 IP 에이전트는 저전력 상태에 들어갈 때 자신 고유의 저전력 시퀀스를 구현하는, SoC.

청구항 40

제38항에 있어서, 저전력 상태는 적어도 하나의 IP 에이전트가 트랜잭션 준비 상태를 유지하는 저전력 모드를 포함하는, SoC.

청구항 41

제40항에 있어서, 인터커넥트는 적어도 하나의 IP 에이전트가 저전력 상태에 있고 트랜잭션 준비 상태를 유지할 때 프록시로서 동작하지 않지만, 적어도 하나의 IP 에이전트를 동작하기 위한 낮아진 클록 주파수 및 전압 중 적어도 하나를 수용하도록 하나 이상의 링크 중 하나에 대한 세팅을 조절하는, SoC.

청구항 42

제38항에 있어서, 저전력 상태는 적어도 하나의 IP 에이전트가 더 이상 트랜잭션 준비 상태가 아닌 저전력 모드를 포함하는, SoC.

청구항 43

제42항에 있어서, 적어도 하나의 IP 에이전트는 저전력 모드에 있고 더 이상 트랜잭션 준비 상태가 아닌 때 상태 정보를 유지하는, SoC.

청구항 44

제42항에 있어서, 적어도 하나의 IP 에이전트는 저전력 모드에 있고 더 이상 트랜잭션 준비 상태가 아닌 때 상태 정보를 유지하지 않는, SoC.

청구항 45

제38항에 있어서, 저전력 상태는 적어도 하나의 IP 에이전트로 최소 전력이 공급되는 파워 오프 모드를 포함하는, SoC.

청구항 46

제38항에 있어서, 저전력 상태는 적어도 하나의 IP 에이전트에 대한 동작 클록 주파수를 감소시킴으로써 구현되는, SoC.

청구항 47

제38항에 있어서, 저전력 상태는 적어도 하나의 IP 에이전트에 의해 사용되는 전압을 감소시킴으로써 구현되는, SoC.

청구항 48

제38항에 있어서, 저전력 상태는

적어도 하나의 IP 에이전트에 대한 동작 클록 주파수를 감소시키고,

동작 클록 주파수를 감소시킨 후 적어도 하나의 IP 에이전트에 의해 사용되는 공급 전압을 감소시킴으로써 구현되는, SoC.

청구항 49

시스템 온 칩(SoC)으로서,

SoC 상의 복수의 IP 에이전트 간의 트랜잭션 트래픽을 처리하기 위한 인터커넥트 - 인터커넥트는 포트를 포함하고, 복수의 IP 에이전트는 제1 IP 에이전트가 정상 동작을 재개할 수 있도록 하기 위해 웨이크-업 시퀀스를 개시하도록 구성된 제1 IP 에이전트를 포함함 - 와, 상기 웨이크-업 시퀀스는,

제1 IP 에이전트가 동작 불능 상태이고 링크가 휴지(quiescent) 상태인 동안, 제1 IP 에이전트가 웨이크-업 트리거 이벤트에 응답하여 링크를 통해 인터커넥트로 웨이크-업 통신을 전송하는 것,

인터커넥트가 동작 불능 상태에 있는 제1 IP 에이전트에 대한 웨이크-업 통신을 기다리는 것, 및

인터커넥트가 웨이크-업 통신을 시스템 제어기로 전달함으로써 웨이크-업 통신이 수신될 때 시스템 제어기에 이를 통지하는 것을 포함하고;

제1 IP 에이전트와 연관된 클럭을 차단하거나, 또는

제1 IP 에이전트에 대한 전원 공급을 감소시키는 것 중,

적어도 하나를 수행함으로써 제1 IP 에이전트를 동작 불능 상태로 만들도록 구성된 전력 관리자와; 그리고

(a) 제1 IP 에이전트에게 트랜잭션을 생성하는 것을 중단하도록 명령하고,

(b) 미결 트랜잭션이 완료되기를 기다리고 제1 IP 에이전트에 의해 생성된 트랜잭션이 완료되었다는 결정에 응답하여 링크를 휴지 상태로 만들고,

(c) 제1 IP 에이전트가 동작 불능 상태에 들어가기 위한 시퀀스를 개시함으로써,

링크를 휴지 상태로 만들도록 구성된 휴지 관리자를 포함하는, SoC.

청구항 50

제49항에 있어서, 동작 불능 상태에 있을 때 인터커넥트를 제1 IP 에이전트를 대리하는 프록시로서 구성하는 것을 더 포함하고, 상기 인터커넥트는,

새로운 트랜잭션이 개시작되는 것을 방지하거나, 또는

미결 트랜잭션이 완료되기를 기다리는 것 중,

하나 이상에 의해 제1 IP 에이전트가 이미 발행한 트랜잭션을 인터커넥트가 비운(drains) 후에 프록시로서 동작하는, SoC.

청구항 51

제50항에 있어서, 인터커넥트는,

제1 IP 에이전트에 의해 추가 트랜잭션이 생성되는 것을 허용하지 않고,

동작 불능 상태에 있을 때 소스가 제1 IP 에이전트로 트랜잭션을 전송하는지 여부를 확인하고,

제1 IP 에이전트를 타킷으로 한 트랜잭션에 응답함으로써, 제1 IP 에이전트를 대리하는 프록시로서 동작하는, SoC.

청구항 52

삭제

청구항 53

제49항에 있어서, 웨이크-업 트리거 이벤트는

(a) 제1 IP 에이전트가 SoC 외부의 소스로부터의 통신을 수신하는 것,

(b) 지정 시간이 만료된 것,

(c) 시스템 제어기에 의해 명령될 때, 및

(d) 유효 트랜잭션이 동작 불능 상태인 제1 IP 에이전트를 타깃으로 할 때

중 하나를 포함하는, SoC.

청구항 54

삭제

청구항 55

삭제

청구항 56

제49항에 있어서, 웨이크-업 시퀀스는,

인터커넥트에 의해 전달되는 웨이크-업 통신을 수신하는 것에 응답하여 시스템 제어기가 웨이크-업 커맨드를 동

작 불능 상태인 제1 IP 에이전트로 전송하는 것, 및

제1 IP 에이전트가 시스템 제어기로부터 웨이크-업 커맨드를 수신하는 것에 응답하여 자신의 웨이크-업 시퀀스를 개시하는 것을 포함하는, SoC.

청구항 57

삭제

발명의 설명

기 술 분 야

[0001] **관련 출원의 교차-참조**

[0002] 본 발명은 2018년 03월 30일에 출원된 미국 가특허 출원 번호 62/650,589(PRTIP001P) 및 2018년 06월 18일에 출원된 가특허 출원 번호 62/691,117(PRTIP002P)의 의 우선권을 주장한다. 상기에서 나열된 우선권 출원 각각은 모든 목적으로 그 전체가 본 명세서에서 참조로서 포함된다.

[0003] **발명의 분야**

[0004] 본 출원은 시스템 온 칩(SoC: System on a Chip)과 관련되며, 더 구체적으로 SoC 상 리셋 및/또는 전력 관리 기능을 일관되게 구현함으로써, 특히 SoC의 군에 걸쳐 SoC의 더 균일한 시스템 소프트웨어 뷰를 제공하는 시스템 및 방법과 관련된다.

배 경 기 술

[0005] 시스템 온 칩("SoC")이 종종 IP 에이전트로 지칭되는 복수의 서브-시스템을 포함하는 집적 회로이다. 일반적으로 IP 에이전트는 특정 기능을 구현 또는 수행하도록 설계된 회로의 "재사용 가능" 블록이다. IP 에이전트를 이용함으로써, 복합 SoC를 개발하는 시간 및 비용이 상당히 감소될 수 있다.

[0006] SoC는 일반적으로 시스템 제어기 및 인터커넥트, 가령, 버스 또는 네트워크 온 칩(NoC)을 포함한다. 시스템 제어기는 시스템 소프트웨어를 운영하고 SoC의 전체 동작을 관리하도록 제공된다. 다양한 IP 에이전트는 하나 이상의 링크를 통해 인터커넥트로 연결되고 상기 인터커넥트를 통해 서로 통신한다.

[0007] SoC 개발자는 일반적으로, 복수의 벤더로부터의 개별 IP 에이전트를 이용한다. 각각의 IP 에이전트는 리셋을 위해 자신 고유의 절차를 정상적으로 구현할 것이다. 시스템 제어기 및 SoC 상의 인터커넥트의 관점에서, 이는 몇몇 이유로 문제가 된다.

[0008] 일반적인 SoC가 보통 인터커넥트에 연결된 복수의 IP 에이전트를 가질 것이다. 리셋되면, 각각의 IP 에이전트가 고유의 리셋 절차 때문에 각각 사용될 때 상이한 시간에서 리셋 상태로부터 벗어날 가능성이 높을 것이다. 각각의 IP 에이전트가 리셋으로부터 벗어나는 상이한 시간이 상당한 문제를 일으킬 수 있다. 출발지(source, 소스) IP 에이전트가 아직 리셋 중인 도착지 IP 에이전트에 대한 트랜잭션을 생성하는 경우, (1) 도착지 IP 에이전트가 요청을 처리할 수 없고 (2) 출발지 IP 에이전트가 결코 응답을 수신하지 않는다. 따라서, 전체 시스템은 멈출 수 있으며(hang up), 시스템 단위 리셋이 필요할 가능성이 있다.

[0009] 행업을 방지하기 위한 한 가지 알려진 방식이 인터커넥트와 각각의 IP 에이전트 사이 각각의 링크 중간에 회로를 설계하고 배치하는 것이다. 이 회로의 목적은 동일한 클록 사이클 동안 인터커넥트로 연결된 모든 IP 에이전트가 리셋으로부터 나감을 보장하는 것이다. 그러나 이러한 방식은 몇 가지 이유로 단점을 가진다:

[0010] 1. 중간 회로의 설계는 SoC의 개발을 종종 지연시킬 시간과 노력을 필요로 한다.

[0011] 2. 일반적으로, 하나의 SoC에서 다른 SoC로의 중간 회로가 상이한 설계 팀에 의해 개발된다. 따라서, 중간 회로는 일반적으로 SoC 간에 상이하거나, 심지어 동일한 SoC 상의 상이한 서브-시스템 간에 상이하다.

[0012] 3. 회로의 복잡도는 일반적으로 주어진 인터커넥트에 연결될 수 있는 IP 에이전트의 수가 제한적임을 의미한다. 이 제한이 미치는 실제 영향은 주어진 개수의 IP 에이전트를 수용하는 데 더 많은 인터커넥트 레벨이 필요하다는 것이다. 따라서 SoC의 전체 복잡도가 증가한다.

[0013] 때때로, IP 에이전트가 오작동한다. 예를 들어, IP 에이전트가 스퓨리어스 트랜잭션을 인터커넥트 상으로 주입

할 수 있고, 수신된 트랜잭션에 응답하지 못할 수 있으며, 예외 메시지를 생성할 수 있다. 특정 상황에서, 오작동 IP 에이전트가 리셋될 필요가 있을 수 있다. 현재 SoC 인터커넥트 표준에 따르면, 어떠한 표준화된 IP 에이전트 리셋 메커니즘도 없다. 전체 SoC가 리셋되어야 하거나, 중재 회로가 IP의 필요한 절연, 리셋, 및 시스템으로의 재-도입 등을 수행하도록 설계될 필요가 있다.

[0014] 전력 관리는 또한 특정 현재 SoC 인터커넥트 표준에 의해 해결되지 않는다. 예를 들어, AMBA(Advanced Microcontroller Bus Architecture) 프로토콜은, 전력 관리를 해결하지 않으며 IP 에이전트를 의도적으로 끄기 위한 어떠한 방법도 제공하지 않는다. 이 능력을 제공하기 위해, 예를 들어,전력 관리를 핸들링하기 위한 링크 상의 추가 중간 회로를 개발함으로써, 전력 관리 기능은 일반적으로 칩 단위로 SoC로 맞춤 설계될 필요가 있다.

[0015] 복수의 SoC를 제공하는 많은 회사가 시스템 소프트웨어의 특정 양을 유사한 디바이스들 간에 공유하여, 출시까지의 시간을 감소시킬 것이다. 그러나 유사한 SoC의 경우라도, 소프트웨어는 일반적으로, 심지어, IP 에이전트가 동일할 수 있는 상황에서도, 하나의 디바이스에서 다른 디바이스로 단순히 포팅될 수 없다. 리셋 및/또는 전력 관리를 위해 사용되는 임의의 중간 회로의 사소한 차이가 존재하는 경우, 시스템 소프트웨어는 각각의 디바이스에 대해 수정되고 디버깅될 필요가 있을 수 있다.

[0016] 따라서 다량의 SoC를 개발하는 회사는 (1) 각각의 디바이스에 대해 리셋 및 가능하다면 전력 관리를 구현하기 위한 커스텀화된 회로를 개발하는 것 및 (2) 각각의 디바이스에 대해 시스템 소프트웨어를 수정하고 디버깅하는 것을 해결해야 한다. 복수의 디바이스 간 이러한 일은 비싸고, 복잡하며, 시간 소모적이어서, 제품을 시장으로 내보내는 것을 저해한다.

[0017] 따라서 커스텀화에 대한 필요성을 제거하여, 복수의 SoC 간 일관된 시스템 소프트웨어 관점을 야기하는 SoC 상의 IP 에이전트의 리셋 및 전력 관리를 지속적으로 구현하기 위한 시스템이 필요하다.

발명의 내용

해결하려는 과제

과제의 해결 수단

[0018] 따라서 커스텀화에 대한 필요성을 제거하여, 복수의 SoC 간 일관된 시스템 소프트웨어 관점을 야기하는 SoC 상의 IP 에이전트의 리셋 및 전력 관리를 지속적으로 구현하기 위한 시스템이 개시된다.

[0019] 하나의 실시예에서, 시스템은 하나 이상의 IP 에이전트, 인터커넥트 및 IP 에이전트와 인터커넥트 간 하나 이상의 링크를 각각 포함한다. IP 에이전트가 리셋을 겪을 때, 링크를 통한 인터커넥트와 각각의 IP 에이전트 간 개별적인 협상이 발생한다. 개별 협상이 있으면, 각각의 IP 에이전트는 다른 IP 에이전트의 타이밍에 무관하게, 자신 고유의 시간 스케줄로 리셋으로부터 벗어날 수 있다. 리셋으로부터 벗어나면, 각각의 IP 에이전트는 "트랜잭션 준비(transaction ready)" 상태가 되며 인터커넥트로 도입되어, 상기 인터커넥트에 연결된 타 요소, 가령, 시스템 제어기에게 보이게 될 수 있다.

[0020] 또 다른 실시예에서, 인터커넥트는 동작 불능인 임의의 IP 에이전트에 대한 프록시로서 구성될 수 있다. 이 특징은, (1) 트랜잭션 준비 상태 전, (2) 오작동 시 및/또는 (3) 전력 다운 상태에서 작동 불능일 때 IP 에이전트가 트랜잭션 타깃인 때 발생할 수 있는 시스템 전체 행업을 방지하기 때문에 유리하다. 프록시로서 동작하는 인터커넥트의 경우, 타깃 IP 에이전트로부터의 응답을 무기한 대기하는 출발지에 의해 야기되는 행업을 방지하는 예외 메시지가 트랜잭션을 전송하는 출발지로 전송될 수 있다.

[0021] 또 다른 실시예에서, 인터커넥트가 IP 에이전트에 대한 프록시로서 동작하도록 배열할 수 있음으로써 (1) IP 에이전트가 개별적으로 리셋되고, (2) IP 에이전트가 전력 절약 상태가 될 수 있다. 다양한 실시예에서, 전력 절약 상태는 상태 정보와 함께 복수의 모드, 가령, 저전력, 동작 모드, 상태 정보가 유지되거나 유지되지 않는 저전력 동작 불능 모드, 또는 전력 오프 모드 중 하나를 포함할 수 있다.

[0022] 본 발명의 많은 문제를 해결한다. (1) 동일한 시간/클록 사이클 동안 리셋으로부터 벗어난 각각의 IP 에이전트를 관리하고 (2) IP 에이전트의 전력을 관리하기 위한 커스텀 회로를 생성할 필요성을 없앤다. 대신, 본 발명은 바람직하게도 이들 기능 중 둘 모두에 대한 균일 구현예를 제공하며, 이는 복수의 SoC 간 일관된 시스템 소프트웨어를 야기한다. 일관된 소프트웨어 뷰를 갖고, SoC의 패밀리 간 커스텀 설계 및 소프트웨어 수정 중 다수가

제거되어, 개별 비용을 절약하고, 복잡도를 감소시키며, 더 빨른 출시 시간을 제공할 수 있다.

도면의 간단한 설명

[0023] 본 출원 및 이의 이점이 첨부된 도면과 함께 이하의 기재를 참조하여 더 잘 이해될 수 있다.

도 1은 비-제한적 실시예에 따른 시스템 온 칩(SoC)에 대한 공유 인터커넥트의 블록도이다.

도 2는 비-제한적 실시예에 따른 트랜잭션의 예시적 패킷이다.

도 3a는 제1 비-제한적 실시예에 따른 중재 요소를 도시하는 논리도이다.

도 3b는 제2 비-제한적 실시예에 따른 중재 요소를 도시하는 논리도이다.

도 4는 비-제한적 실시에에 따른 공유 인터커넥트의 가상 채널을 통한 트랜잭션의 부분을 중재 및 전송하기 위한 동작 단계를 도시한다.

도 5는 비-제한 실시예에 따른 공유 인터커넥트의 가상 채널을 통한 상이한 트랜잭션의 부분의 전송을 인터리빙하는 제1 예시이다.

도 6은 비-제한 실시예에 따른 공유 인터커넥트의 가상 채널을 통한 상이한 트랜잭션의 부분의 전송을 인터리빙하는 제2 예시이다.

도 7은 본 발명의 비-제한 실시예에 따른 두 개의 방향으로 트래픽을 핸들링하기 위한 두 개의 공유 인터커넥트의 블록도이다.

도 8은 본 발명의 비-제한적 실시예에 따른 리셋, 전력 관리 및 작업거부(quiesce) 기능을 갖는 SoC의 블록도이다.

도 9는 본 발명의 비제한적 실시예에 따른 IP 에이전트 리셋 시퀀스를 보여주는 흐름도이다.

도 10은 본 발명의 비-제한적 실시예에 따르는 오작동 IP 에이전트의 리셋 시퀀스를 보여주는 흐름도이다.

도 11은 본 발명의 비제한적 실시예에 따르는 IP 에이전트에 대한 파워 다운/업 시퀀스를 도시하는 흐름도이다.

도 12는 본 발명의 비-제한적 실시예에 따르는 IP 에이전트에 대한 파워 다운/업 시퀀스를 도시하는 흐름도이다.

도 13은 본 발명의 또 다른 비-제한적 실시예에 따르는 IP 에이전트에 대한 파워 다운/업 시퀀스를 도시하는 흐름도이다.

도 14는 링크를 작업거부 상태로 두기 위한 단계를 도시하는 흐름도이다.

도 15a-15d는 IP 에이전트에 대한 다양한 "웨이크-업" 시퀀스를 도시하는 흐름도이다.

도면에서, 때때로 유사한 도면 부호가 유사한 구조 요소를 지정하는 데 사용된다.

발명을 실시하기 위한 구체적인 내용

[0024] 본 출원이 지금부터 첨부된 도면에서 도시된 몇 가지 비-제한적 실시예를 참조하여 상세히 기재될 것이다. 이하에서, 다수의 특정 세부사항이 본 개시의 완전한 이해를 제공하기 위해 제공된다. 그러나 해당 분야의 통상의 기술자라면, 본 개시가 이들 특정 세부사항 중 일부 또는 전부 없이 실시될 수 있음을 자명하게 알 것이다. 또 다른 경우, 공지된 프로세스 단계 및/또는 구조는 본 개시를 불필요하게 모호하게 하지 않도록 상세히 기재되지 않았다.

[0025] 오늘날 개발 중인 집적 회로 중 다수가 극히 복잡하다. 따라서, 많은 칩 설계자가 단일 실리콘 피스 상에서 복수의 서브-시스템 또는 IP 에이전트를 인터커넥트하는 시스템 온 칩, 즉, "SoC" 방식을 이용했다. SoC는 다양한 방식, 가령, 소비자 디바이스(가령, 핸드헬드, 모바일 폰, 태블릿 컴퓨터, 랩톱 및 데스크톱 컴퓨터, 미디어 프로세싱 등), 가상 또는 증강 현실(가령, 로봇, 자율 주행 차량, 항공 등), 의료 기구(가령, 영상 장치 등), 산업, 가정용 자동화, 산업(가령, 스마트 어플라이언스, 가정용 감시 등) 및 데이터 센터 애플리케이션(가령, 네트워크 스위치, 연결된 저장 디바이스 등)에 대해 이용 가능하거나 개발 중이다.

[0026] 본 출원은 일반적으로 공유 자원으로의 액세스를 중재하기 위한 중재 시스템 및 방법에 관련된다. 이러한 공유

자원은, 예를 들어, 복수의 경쟁자들 간 공유되는 버스 인터커넥트, 메모리 자원, 처리 자원, 또는 단지 임의의 그 밖의 다른 자원일 수 있다. 예시로서, 이하에서 상세히 기재될 공유 자원은 시스템 온 칩, 즉, "SoC" 상의 복수의 서브-시스템에 의해 공유되는 인터커넥트이다.

[0027] SoC의 경우, 이하에서 상세히 기재될 바와 같이, 복수의 서브-시스템이 트랜잭션(transaction)의 형태로 서로 트래픽을 교환하고, 공유 자원은 물리 인터커넥트이며, 다양한 트랜잭션, 또는 이의 일부분이 공유 인터커넥트와 연관된 다양한 가상 채널을 통해 전송되고, 복수의 상이한 중재 스킴 및/또는 우선순위 중 하나가 서브-기능들 간 트랜잭션의 전송을 위해 공유 인터커넥트로의 액세스를 중재하는 데 사용될 수 있다.

[0028] **트랜잭션 클래스**

[0029] SoC를 위해 사용되는 상기에서 언급된 공유 인터커넥트 내에서, 적어도 세 가지 유형 또는 클래스의 트랜잭션, 가령, 포스트(P), 비-포스트(NP) 및 완료(C)가 존재한다. 각각에 대한 간단한 정의가 이하의 표 1에서 제공된다.

표 1

[0030]

트랜잭션 클래스	설명
프스트(P)	응답이 필요 없는 트랜잭션(가령, 쓰기 동작)
비-포스트(NP)	도착지 에이전트로부터의 응답 트랜잭션이 필요한 트랜잭션(가령, 읽기 또는 쓰기 동작)
완료(C)	비-포스트 트랜잭션에 대한 응답 트랜잭션

[0031] 포스트 트랜잭션, 가령, 쓰기는 응답 트랜잭션을 요구하지 않는다. 출발지가 지정 도착지로 데이터를 쓰면, 트랜잭션이 종료된다. 비-포스트 트랜잭션, 가령, 읽기 또는 쓰기의 경우, 응답이 필요하다. 그러나 응답은 개별 완료 트랜잭션으로 두 갈래로 나뉜다. 읽기 경우로 다시 말하면, 제1 트랜잭션이 읽기 동작을 위해 사용되고, 별도의, 그러나 관련된 완료 트랜잭션이 읽힌 콘텐츠를 반환하기 위해 사용된다. 비-포스트 쓰기의 경우, 제1 트랜잭션이 쓰기를 위해 사용되고, 제2의 관련 완료 트랜잭션이 쓰기가 완료된 후 확인을 위해 필요하다.

[0032] 트랜잭션은, 유형과 무관하게, 하나 이상의 패킷에 의해 나타내어질 수 있다. 일부 환경에서, 트랜잭션은 단일 패킷에 의해 표현될 수 있다. 또 다른 환경에서, 복수의 패킷이 전체 트랜잭션을 나타내기 위해 필요할 수 있다.

[0033] 비트(beat)가 클록 사이클마다 공유 인터터넥트를 통해 전송될 수 있는 데이터의 양이다. 예를 들어, 공유 인터커넥트가 물리적으로 128 비트 폭이라면, 128 비트가 각각의 비트 또는 클록 사이클마다 전송될 수 있다.

[0034] 일부 상황에서, 트랜잭션이 전송의 복수의 부분으로 분할될 필요가 있을 수 있다. 512비트(64 바이트)이 페이로드를 갖는 단일 패킷을 갖는 트랜잭션을 고려할 수 있다. 공유 인터커넥트가 128 비트 폭(16바이트)에 불과하다면, 트랜잭션은 4개의 부분(가령, 4 x 128 = 512)으로 세그먼트되어 4개의 클록 사이클 또는 비트를 통해 전송될 필요가 있다. 다른 한편으로 트랜잭션이 128 비트 폭 이하인 단일 패킷에 불과하면, 전체 트랜잭션이 하나의 클록 사이클 또는 비트로 전송될 수 있다. 동일한 트랜재션이 추가 패킷을 포함하도록 발생하는 경우, 추가 클록 사이클 또는 비트가 필요할 수 있다.

[0035] 따라서 트랜잭션의 "부분"이라는 용어는 주어진 클록 사이클 또는 비트 동안 공유 인터커넥트를 통해 전송될 수 있는 데이터의 양이다. 부분의 크기는 공유 인터커넥트의 물리적 폭에 따라 달라질 수 있다. 예를 들어, 공유 인터커넥트가 물리적으로 64 데이터 비트 폭인 경우, 하나의 사이클 또는 비트 동안 전송될 수 있는 최대 비트 수는 64 비트이다. 주어진 트랜잭션이 64 비트 이하의 페이로드를 갖는 경우, 전체 트랜잭션이 단일 부분으로 공유 인터커넥트를 통해 전송될 수 있다. 반면에 페이로드가 더 큰 경우 패킷은 복수의 부분에서 공유 인터커넥트를 통해 전송되어야 한다. 128, 256 또는 512 비트의 페이로드가 있는 트랜잭션은 각각 2개, 4개 및 8개 부분이 필요로 한다. 따라서, "부분" 또는 "부분들"이라는 용어는 임의의 주어진 클록 사이클 또는 비트 동안 공유 인터커넥트를 통해 전송될 수 있는 트랜잭션의 일부 또는 전체를 의미하는 것으로 광범위하게 해석되어야 한다.

[0036] **스트림**

[0037] 스트림은 가상 채널과 트랜잭션 클래스의 쌍으로 정의된다. 예를 들어, 4개의 가상 채널(가령, VC0, VC1, VC2 및 VC3)과 3개의 트랜잭션 클래스(P, NP, C)가 있는 경우, 최대 12개의 서로 다른 가능한 스트림이 존재한다. 가상 채널과 트랜잭션 클래스의 다양한 조합은 아래 표 II에 자세히 설명되어 있다.

표 2

[0038]

스트림 번호	VC/TC 조합
1	VC0/NP
2	VC0/P
3	VC0/C
4	VC1/NP
5	VC1/P
6	VC1/C
7	VC2/NP
8	VC2/P
9	VC2/C
10	VC3/NP
11	VC3/P
12	VC3/C

[0039] 위에서 논의된 트랜잭션 클래스의 수는 단지 예시 일 뿐이며 제한하는 것으로 해석되어서는 안된다는 점에 유의해야 한다. 이와 달리, 임의의 수의 가상 채널 및/또는 트랜잭션 클래스가 사용될 수 있다.

[0040] **공유 인터커넥트의 가상 채널을 통한 중재**

[0041] 도 1을 참조하면, 중재 시스템(10)의 블록도가 도시된다. 비-제한적 실시예에서, 중재 시스템은 거래를 업스트림 서브-기능(14)(즉, IP_4, IP_5 및 IP_6)으로 전송하려고 시도하는 공유 인터커넥트(12)에 대한 다수의 서브-기능(14)(즉, IP_1, IP_2 및 IP_3)에 의한 액세스를 중재하는 데 사용된다.

[0042] 공유 인터커넥트(12)는 N 데이터 비트 폭이고 M 제어 비트를 포함하는 물리 인터커넥트이다. 공유 인터커넥트(12)은 또한 단방향인데, 이는 출발지(즉, IP_1, IP2 및 IP3)에서 도착지(즉, IP_4, IP_5 및 IP_6)로의 트래픽만 핸들링하는 것을 의미한다.

[0043] 다양한 대안에서, N 데이터 비트의 수는 임의의 정수일 수 있지만, 일반적으로 숫자 2의 급수(예를 들어, 2^1, 2^2, 2^3,2^4, 2^5, 2^6, 2^7 ,2^8 2^9 등) 또는(2, 4, 6, 8, 16, 32, 64, 128, 256 등) 비트 폭이다. 대부분의 실제 적용예에서 N 비트의 수는 32, 64, 128, 256 또는 심지어 512이다. 그러나 이러한 폭은 단지 예시일 뿐이며 어떤 방식으로든 제한하는 것으로 해석되어서는 안 된다.

[0044] 제어 비트 M의 수는 또한 변할 수 있고 임의의 수일 수 있다.

[0045] 이후에 "가상 채널" 또는 "VC"로 지칭되는 하나 이상의 논리 채널(도시되지 않음)이 공유 인터커넥트(12)와 연관된다. 각각의 가상 채널은 독립적이다. 각각의 가상 채널은 복수의 독립 스트림과 연관될 수 있다. 가상 채널의 수는 매우 다양할 수 있다. 예를 들어, 최대 32개 이상의 가상 채널이 정의되거나 공유 인터커넥트(12)와 연관될 수 있다.

[0046] 다양한 대안적인 실시예에서, 각각의 가상 채널은 상이한 우선 순위를 할당받을 수 있다. 하나 이상의 가상 채널(들)에는 더 높은 우선 순위가 할당될 수 있는 반면, 하나 이상의 다른 가상 채널(들)에는 더 낮은 우선 순위가 할당될 수 있다. 높은 우선 순위 채널일수록 낮은 우선순위 가상 채널에 비해 공유 인터커넥트(12)로의 액세스를 부여 받거나 중재된다. 다른 실시예에서, 각각의 가상 채널에는 동일한 우선 순위가 주어질 수 있으며, 이 경우 공유 인터커넥트(12)로의 액세스를 부여 받거나 중재될 때 다른 가상 채널에 비해 하나의 가상 채널도 선호되지 않는다. 또 다른 실시예에서, 가상 채널 중 하나 이상에 할당되는 우선순위가 또한 동적으로 변할 수 있다. 예를 들어, 제1 환경 세트에서 모든 가상 채널에 동일한 우선순위가 할당될 수 있지만, 제2 환경 세트에서는 특정 가상 채널(들)이 다른 가상 채널(들)보다 더 높은 우선 순위를 할당 받을 수 있다. 따라서 상황이 변함에 따라, 가상 채널 간에 사용되는 우선순위화 스킴이 현재 작동 조건을 가장 잘 충족하도록 변경될 수 있다.

[0047] 각각의 서브-시스템(14)은 일반적으로 IP 코어 또는 에이전트로 일반적으로 지칭되는 "재사용 가능한"회로 또는 로직의 블록이다. 대부분의 IP 에이전트는 특정 기능, 가령, 주변 장치, 가령, 이더넷 포트, 디스플레이 드라이버, SDRAM 인터페이스, USB 포트 등을 위한 제어기를 수행하도록 설계된다. 이러한 IP 에이전트는 일반적으로, 집적 회로(IC), 가령, ASIC(Application Specific Integrated Circuit) 또는 FPGA(Field Programmable Gate

Array) 상에서 제공되는 복합 시스템의 전체 설계 내에서 필요한 서브-시스템 기능을 제공하는 "빌딩 블록"으로 사용된다. 사용 가능한 IP 에이전트의 라이브러리를 사용함으로써, 칩 설계자는 더 복잡한 집적 회로의 설계에서 다양한 논리 기능을 함께 쉽게 "볼트(bolt)"하여, 설계 시간을 감소시키고 개발 비용을 절약할 수 있다.서브-시스템 에이전트(14)가 전용 IP 코어의 관점에서 위에서 설명되었지만, 이는 필수 요건이 아님을 이해해야 한다. 반대로, 서브-시스템(14)은 단일 포트(20)에 연결되거나 공유되는 IP 기능의 집합일 수도 있다. 따라서 "에이전트"라는 용어는, 서브-시스템이 단일 기능 또는 복수 기능을 수행하는지에 무관하게, 포트(20)에 연결된 임의의 유형의 서브-시스템으로 광범위하게 해석되어야 한다.

[0048] 한 쌍의 스위치(16 및 18)는 각각 전용 액세스 포트(20)를 통해 각각의 서브-시스템 에이전트(14)와 공유 인터커넥트(12) 사이에 액세스를 제공한다. 아래와 같은 예시적 실시 형태가 제공된다:

[0049] (1) 서브-시스템 에이전트(IP_1, IP_2 및 IP_3)은 각각 액세스 포트 0, 포트 1 및 포트 2를 통해 스위치(16)와 연결된다.

[0050] (2) 서브-시스템 에이전트(IP_4, IP_5 및 IP_6)는 각각 포트 3, 포트 4 및 포트 5를 통해 스위치(18)와 연결된다.

[0051] (3) 또한, 인터커넥트(12)를 통한 액세스 포트(22)는 스위치(16)에 대한 서브-시스템 에이전트(IP_4, IP_5 및 IP_6) 액세스를 집합적으로 제공한다.

[0052] 스위치(16 및 18)는 다중화 및 역-다중화 기능을 수행한다. 스위치(16)는 서브-시스템 에이전트(IP_1, IP_2 및/또는 IP_3)에 의해 생성된 업스트림 트래픽을 선택하고 공유 인터커넥트(12)를 통해 트래픽 다운스트림을 전송한다. 스위치(18)에서, 역-다중화 작업이 수행되고 트래픽이 타깃 서브-시스템 에이전트(즉, IP_4, IP_5, 또는 IP_6)로 제공된다.

[0053] 각각의 액세스 포트(20)는 고유한 포트 식별자(ID)를 가지며 각각의 서브-시스템 에이전트(14)에게 스위치(16 또는 18)에 대한 전용 액세스를 제공한다. 예를 들어, 서브-시스템 에이전트(IP_1, IP_2 및 IP_3)는 각각 액세스 포트 포트 0, 포트 1 및 포트 2를 액세스하도록 할당된다. 마찬가지로 서브-시스템 에이전트(IP_4, IP_5 및 IP_6) 각각에 액세스 포트 포트 3, 포트 4 및 포트 5가 할당된다.

[0054] 스위치(16, 18)로/로부터의 수신 및 송신 지점을 제공하는 것 외에도, 고유한 포트 ID(20)는 서브-시스템 에이전트(14) 간의 트래픽을 어드레싱하는 데 사용된다. 각각의 포트(20)는 시스템 메모리(24) 내 할당된 어드레싱 가능한 공간의 특정 양을 가진다.

[0055] 특정 비-제한적 실시예에서, 액세스 포트(20)의 전부 또는 일부는 또한 "전역" 포트 식별자 및 그들의 고유 포트 ID를 할당 받을 수 있다. 트랜잭션 및 그 밖의 다른 트래픽은 전역 포트 식별자에게 할당된 액세스 포트 전체 또는 일부로 전송될 수 있다. 따라서, 전역 식별자를 사용하면, 트랜잭션 및 그 밖의 다른 트래픽이 액세스 포트(20)의 전부 또는 일부에 널리 전파되거나 브로드캐스팅 될 수 있어서, 고유 식별자를 사용하여 각각의 액세스 포트(20)를 개별적으로 어드레싱할 필요가 없어진다.

[0056] 스위치(16)는 또한 중재 요소(26), ARL(Address Resolution Logic)(28) 및 어드레스 분해 LUT(Look Up Table)(30)를 포함한다.

[0057] 동작 중에 서브-시스템 에이전트(IP_1, IP_2 및 IP_3)는 트랜잭션을 생성한다. 각각의 트랜잭션이 생성되면, 발신 서브-시스템 에이전트(14)에 의해 패킷화되고 패킷화된 트랜잭션은 대응하는 포트(20)를 통해 로컬 스위치(16)로 주입된다. 예를 들어, IP_1, IP_2 및 IP_3에 의해 발생된 트랜잭션의 일부가 포트 0, 포트 1 및 포트 2를 통해 각각 스위치(16)로 제공된다.

[0058] 포트(20)는 각각 인터커넥트 채널(12)과 연관된 가상 채널 각각에 대한 복수의 선입 선출 버퍼(도시되지 않음)를 포함한다. 비제한적 실시예에서, 4개의 가상 채널이 존재한다. 이 경우, 각각의 포트(20)는 4개의 버퍼, 즉, 각각의 가상 채널에 대해 하나씩을 포함한다. 다시 말해, 포트(20)에 포함된 가상 채널 및 버퍼의 수는 변할 수 있으며 4개로 제한되지 않는다는 것을 이해해야 한다. 반대로, 가상 채널과 버퍼의 수는 4개보다 많거나 적을 수 있다.

[0059] 주어진 트랜잭션이 2개(또는 그 이상)의 부분으로 표현되는 경우, 그 부분은 동일한 버퍼에 유지된다. 예를 들어, 인터커넥트(12)가 128 데이터 비트 폭이고 트랜잭션이 512 비트 페이로드를 포함하는 패킷으로 표현되는 경

우 트랜잭션은 4개의 클록 사이클 또는 비트를 통해 전송되는 4개의 부분으로 세그먼팅되어야 한다. 한편, 트랜잭션이 64 비트의 페이로드를 갖는 단일 패킷으로 표현될 수 있는 경우, 단일 부분은 하나의 클록 사이클 또는 비트로 전송될 수 있다. 주어진 트랜잭션의 모든 부분을 동일한 버퍼에 유지함으로써 가상 채널은 논리적으로 독립적으로 유지된다. 즉, 특정 트랜잭션과 연관된 모든 트래픽이 항상 동일한 가상 채널을 통해 스트림으로서 전송되고 복수의 가상 채널을 통해 분기되지 않는다.

[0060] 중재 요소(26)는 다양한 액세스 포트(20)에 의해 유지되는 트랜잭션의 경쟁 버퍼링 부분들 간 중재를 담당한다. 비-제한적 실시예에서, 중재 요소(26)는, 복수의 경쟁 트랜잭션이 이용 가능하다고 가정할 때, 매 클록 사이클마다 중재를 수행한다. 사이클별 중재 위너가 서브-시스템(IP_1, IP_2 및 IP_3) 중 하나로부터의, 인터커넥트(12)에 대한 액세스를 승인 받고 이를 통해 전송되는 트랜잭션의 부분을 산출한다.

[0061] 트랜잭션을 생성할 때, 출발지 서브-시스템(IP1, IP2 및 IP3)은 일반적으로 가능한 도착지 서브-시스템 에이전트(IP_4, IP_5 및 IP_6)에 대한 어드레스 공간 내 어드레스를 알지만, 트랜잭션을 이들의 도착지로 라우팅하는 데 필요한 정보(가령, 포트 ID(20 및/또는 22))를 알지 못한다. 하나의 실시예에서, 로컬 ARL(Address Resolution Logic)(28)이 알려진 도착지 어드레스를 필요한 라우팅 정보로 분해하는 데 사용된다. 즉, 출발지 서브-에이전트(14)는 시스템 메모리(24) 내 주어진 주소에 액세스하기를 원한다는 것을 단순히 알 수 있다. 따라서 ARL(28)은 LUT(30)를 액세스하고 특정된 어드레스에 대응한 최종 도착지로의 전달 경로를 따라 포트(들)(20/22)의 어드레스 조회를 수행한다. 포트(20/22)가 알려지면 이 정보는 트랜잭션의 패킷의 도착지 필드에 삽입된다. 따라서, 패킷(들)은 전달 경로를 따라 포트(20/22)로 전달된다. 일반적으로, 전달 경로를 따르는 다운스트림 노드는 필요한 전달 정보가 이미 알려져 있고 패킷의 도착지 필드에 포함되어 있기 때문에 추가 조회를 수행할 필요가 없다. 이하에서 더 상세히 기재될 바와 같이, SBR(Source Based Routing)이라고 지칭되는 그 밖의 다른 유형의 트랜잭션을 사용할 경우, 출발지 IP 에이전트가 도착지 포트 어드레스를 안다. 그 결과, ARL(28)에 의해 수행되는 조회는 일반적으로 수행 될 필요가 없다.

[0062] 대안적인 실시예에서, 인터커넥트 내의 모든 노드가 ARL(28) 및 LUT(30)를 필요로 하는 것은 아니다. 이러한 요소를 갖지 않는 노드의 경우, 필요한 라우팅 정보가 없는 트랜잭션이 디폴트 노드로 전달될 수 있다. 디폴트 노드에서 ARL(28) 및 LUT(30)가 액세스되고 그런 다음 필요한 라우팅 정보가 트랜잭션의 패킷(들)의 헤더에 삽입될 수 있다. 디폴트 노드는 일반적으로 ARL(28) 및 LUT(30)가 없는 노드의 업스트림이다. 그러나 이는 필수 사항이 아니다. 하나 또는 복수의 디폴트 노드는 SoC의 어느 곳에나 위치할 수 있다. 특정 노드에서 ARL(28) 및 LUT(30)을 제거함으로써 이들의 복잡도가 감소될 수 있다.

[0063] ARL(28)은 또한 "오더링 포인트"라고도 지칭될 수도 있는데, 이는 트랜잭션의 위닝 부분에 대한 전달 도착지를 디코딩하는 것 외에, 각각의 가상 채널 내에서의 트랜잭션의 위닝 부분에 대한시퀀스 오더를 정의하기 때문이다. 각각의 중재가 해결될 때 ARL(28)이 어드레스 포트 조회를 수행하는 데 사용되는지 여부에 관계없이 트랜잭션의 위닝 부분이 각각의 가상 채널에 대해 제공되는 선입 선출 큐로 삽입된다. 그런 다음 트랜잭션의 위닝 부분이 버퍼에서 인터커넥트(12)를 통한 전송을 위해 자신의 차례를 기다린다.

[0064] ARL(28)은 또한 "업스트림" 및 다운스트림" 트래픽을 정의하는 데 사용된다. 즉, 스위치(16)(즉, IP_1, IP_2 및 IP_3)와 연관된 IP 에이전트(14)에 의해 생성된 임의의 트랜잭션은 ARL(28)에 대해 업스트림으로 간주된다. ARL(28)을 포스트하는(즉, IP_4, IP_5 및 IP_6으로 전송된) 모든 트랜잭션은 다운스트림 트래픽으로 간주된다.

[0065] 스위치(16)(즉, IP_1, IP_2 및 IP_3)와 연관된 IP 에이전트(14)는 직접 또는 간접적으로 서로 통신하고 트랜잭션을 전송할 수 있다. 종종 SBR(Source Based Routing)이라고 지칭되는 직접 통신을 통해, IP 에이전트(14)는 피어-투-피어 모델에서 트랜잭션을 서로에게 전송할 수 있다. 이 모델을 사용하면, 출발지 IP 에이전트는 이의 피어 IP 에이전트(14)의 고유 포트 ID를 알아서, LUT(30)에 액세스하기 위해 ARL(28)을 사용할 필요성이 없어진다. 또는, 스위치(16)와 연관된 IP 에이전트 간의 트랜잭션이 ARL(28)을 이용해 라우팅될 수 있다. 앞서 기재된 것과 유사하게, 이 모델을 이용할 때, 출발지 IP 에이전트만 도착지 IP 에이전트(14)의 어드레스만 알며, 라우팅을 위해 필요한 정보는 알지 못한다. 그런 다음 ARL(28)은 LUT(30)를 액세스하고, 대응하는 포트 ID를 찾는 데 사용되며, 그런 다음 이 ID는 트랜잭션의 패킷(들)의 도착지 필드에 삽입된다.

[0066] **패킷 포맷**

[0067] IP 에이전트(14)는 인터커넥트(12)와 연관된 가상 채널을 통해 트랜잭션을 생성하고 처리한다. 각각의 트랜잭션은 일반적으로 하나 이상의 패킷으로 구성된다. 일반적으로 각각의 패킷은 고정 헤더 크기 및 포맷을 가진다.

일부 경우, 각각의 패킷은 고정된 크기의 페이로드를 가질 수 있다. 다른 경우에, 패킷 페이로드는 크기가 너무 크거나, 너무 작거나, 페이로드가 전혀 없는 경우에도 다양 할 수 있다.

[0068] 도 2를 참조하면, 예시적인 패킷(32)이 도시된다. 패킷(32)은 헤더(34) 및 페이로드(36)를 포함한다. 이 특정 실시예에서, 헤더(34)는 크기가 16 바이트이다. 이 크기는 예시적이며 더 큰 크기(예를 들어, 더 많은 바이트) 또는 더 작은 크기(예를 들어, 더 적은 바이트) 패킷이 사용될 수 있음을 이해해야 한다. 또한 패킷(32)의 헤더(34)는 반드시 모두 동일한 크기일 필요는 없다는 것을 이해해야 한다. 대안적인 실시예에서, SoC에서 패킷 헤더의 크기는 가변적일 수 있다.

[0069] 헤더(34)는 다수의 필드, 가령, 도착지 식별자(DST_ID), 출발지 식별자(SRC_ID), 페이로드 크기 지시자(PLD_SZ), 예약 필드(RSVD), 커맨드 필드(CMD), TAG 필드, 상태(STS), 트랜잭션 ID 필드(TAG), 어드레스 또는 ADDR 필드, USDR/컴팩트 페이로드 필드, 트랜잭션 클래스 또는 TC 필드, 포맷 FMT 필드, 및 바이트 활성화(BE) 필드를 포함한다. 헤더(34)의 다양한 필드는 아래 표 III에서 간략하게 설명된다.

표 3

[0070]

필드의 명칭	설명
DST	타깃 IP 에이전트(14)에 대해 대응하는 포트 ID 특정
SRC	전송 IP 에이전트(14)에 대해 포트 ID를 특정
PLD_SZ	패킷의 페이로드의 크기를 특정
CMD	패킷이 포함하는 트랜잭션/커맨드의 유형을 특정 예시적 커맨드가 증분 읽기(Incremental Read), 증분 쓰기(Incremental Write), 컴팩트 읽기(Compact Read), 컴팩트 쓰기(Compact Write), FIFO에 쓰기, 파괴적 증분 읽기, 둘러 싸기 등을 포함할 수 있다.
TAG	패킷에 대한 트랜잭션 ID 특정 트랜잭션 ID가 비-포스트 트랜잭션 및 이들의 대응하는 완료 트랜잭션을 매칭하는 데 사용된다. TAG 필드 내 매칭되는 트랜잭션 ID를 포함하는 완료 트랜잭션이 수신될 때, 대응하는 비-포스트 읽기 또는 쓰기 동작이 완료된다.
ADDR	요청의 시스템 메모리(24)에서 물리 어드레스를 특정
USRD/컴팩트 페이로드	패킷의 페이로드가 충분히 작은 경우, 페이로드가 아니라 헤더 내 이 필드에서 전송될 수 있다. 이 필드는 또한 출발지와 도착지 IP 포트 간 개인 또는 보안 메시지를 전송하는 데 사용될 수 있다.
STS	이 필드는 완료 패킷의 경우에만 유효하다. 대응하는 비-포스트 트랜잭션의 상태를 특정(즉, 성공적인 완료 또는 실패 트랜잭션).
RSVD	이는 출발지와 도착지 IP 포트 간 개인 또는 보안 메시지를 또한 전송하는 데 사용될 수 있는 예약 필드이다.
FMT	복수의 헤드 포맷이 정의되고 사용되는 경우 포맷 규격을 특정
BE	바이트 활성화는 페이로드 중 어느 바이트가 유효한지를 가리킴.

[0071] 페이로드(36)는 패킷의 콘텐츠를 포함한다. 페이로드의 크기는 달라질 수 있다. 일부 경우, 페이로드가 클 수 있다. 또 다른 경우에는, 작을 수 있다. 또 다른 경우, 콘텐츠가 매우 작거나, "컴팩트"인 경우, 헤더(34)의 USRD 필드에서 전송될 수 있다.

[0072] 트랜잭션의 유형이 종종 트랜잭션을 나타내는 데 사용되는 패킷이 페이로드를 갖는지 여부를 가리킬 것이다. 예를 들어, 포스트 또는 비-포스트 읽기의 경우, 패킷은 액세스될 위치 어드레스를 지정할 것지만, 일반적으로 어떠한 페이로드도 갖지 않을 것이다. 그러나 관련 완료 트랜잭션에 대한 패킷은 읽힌 콘텐츠를 포함하는 페이로드(들)을 포함할 것이다. 포스트 및 비-포스트 쓰기 트랜잭션의 경우, 패킷(들)은 도착지로 써질 데이터를 포함하는 페이로드를 포함할 것이다. 쓰기의 비-포스트 버전을 이용해, 완료 트랜잭션에 대한 패킷이 일반적으로 페이로드를 정의하지 않을 것이다. 그러나 일부 상황에서는 완료 트랜잭션이 페이로드를 정의할 것이다.

[0073] 예시적인 패킷 및 상기의 설명은 패킷에 포함될 수 있는 많은 기본 필드 중 다수를 다룬다. 추가 필드가 삭제되거나 추가될 수 있음이 이해되어야 한다. 예를 들어, 개인 시그널링 필드가 사용되어 출발지와 도착지가 개인 메시지를 공유할 수 있다.

[0074] **중재**

[0075] 도 3a를 참조하면, PCI(Peripheral Component Interconnect) 오더링을 갖는 중재 요소(26)에 의해 수행되는 중재 논리를 나타내는 논리 다이어그램이 도시된다.

[0076] PCI 오더링의 경우, 각각의 포트(20)는 각각의 가상 채널 및 트랜잭션 클래스(P, NP 및 C) 조합에 대해 개별 버퍼를 포함한다. 예를 들어, 4개의 가상 채널(VC0, VC1, VC2 및 VC3)이 있는 경우, 포트 0, 포트 1 및 포트 2 각각은 12개의 선입 선출 버퍼를 가진다. 즉, 각각의 포트 20에 대해 각각의 트랜잭션 클래스(P, NP 및 C) 및 가상 채널(VC0, VC1, VC2 및 VC30) 조합에 대해 버퍼가 제공된다.

[0077] 각각의 IP 에이전트(14)(가령, IP_1, IP_2 및 IP_3)가 트랜잭션을 생성함에 따라, 결과적 패킷이, 대응하는 포트(Port 0, Port 1 및 Port 2)에서, 트랜잭션 유형에 기초하여, 적절한 버퍼에 배치된다. 예를 들어, IP_1에 의해 생성된 포스트(P), 비-포스트(NP) 및 완료(C) 트랜잭션은 각각 포트 0에서 할당된 가상 채널에 대해 포스트, 비-포스트 및 완료 버퍼에 배치된다. IP_2 및 IP_3에 의해 생성된 트랜잭션은 유사한 방식으로 포트 1 및 포트 2에서 할당된 가상 채널에 대해 포스트, 비-포스트 및 완료 버퍼에 유사하게 배치된다.

[0078] 주어진 트랜잭션이 복수의 패킷에 의해 표현되는 경우, 트랜잭션의 모든 패킷이 동일한 버퍼로 삽입된다. 결과적으로 트랜잭션의 모든 패킷은 결국 동일한 가상 채널을 통해 전송된다. 이 정책을 사용하면, 가상 채널이 독립적으로 유지되는데, 이는 상이한 가상 채널이 동일한 트랜잭션과 연관된 복수의 패킷의 전송을 위해 사용되지 않음을 의미한다.

[0079] 각각의 포트(20) 내에서, 패킷은 다수의 상이한 방식으로 주어진 가상 채널에 할당될 수 있다. 예를 들어, 할당은 임의적일 수 있다. 또는, 할당이 각각의 가상 채널에 대한 작업 부하 및 미해결 트래픽의 양에 기초할 수 있다. 하나의 채널이 매우 바쁘고 다른 채널은 그렇지 않은 경우, 포트(20)는 로드 밸런싱을 하고 새로 생성된 트랜잭션 트래픽을 과소-사용되는 가상 채널로 할당하려 시도할 것이다. 결과적으로 라우팅 효율성이 향상된다. 또 다른 대안에서, 트랜잭션 트래픽은 긴급성, 보안 또는 이 둘의 조합에 기초하여 특정 가상 채널에 할당될 수 있다. 특정 가상 채널에 다른 가상 채널보다 높은 우선 순위 및/또는 보안이 부여되는 경우, 높은 우선순위 및/또는 보안 트래픽이 더 높은 우선 순위 가상 채널에 할당된다. 또 다른 실시예에서, 포트(20)는 하드-코딩(hard-code)될 수 있는데, 이는 포트(20)가 단지 하나의 가상 채널을 갖고 상기 포트(20)에 의해 생성된 모든 트래픽이 하나의 가상 채널을 통해 전송된다는 것을 의미한다. 또 다른 실시예에서, 할당은 도착지 포트(20)에 도달하도록 선택된 경로에 기초할 수 있다.

[0080] 또 다른 실시예에서, 가상 채널의 할당은 출발지 IP 에이전트(14)에 의해 단독으로 또는 대응하는 포트(20)와 협력하여 구현될 수 있다. 예를 들어, 출발지 IP 에이전트(14)는 주어진 트랜잭션의 패킷이 특정 가상 채널로 할당될 것을 요청하는 대응하는 포트(20)로 제어 신호를 생성할 수 있다. IP 에이전트(14)는 또한 앞서 논의된 바와 같이 모든 가상 채널, 보안, 긴급성 등에 걸쳐 균형 잡힌 사용에 기초하여 임의적이고 하드 코딩된 할당 결정을 내릴 수 있다.

[0081] 중재 위너를 선택함에 있어서, 중재 요소(26)는 사이클별로 복수의 중재 단계를 수행한다. 이들 중재 단계는 다음을 포함한다:

[0082] (1) 포트 선택,

[0083] (2) 가상 채널 선택, 및

[0084] (3) 트랜잭션 클래스 선택.

[0085] 상기 순서(1), (2) 및 (3)은 고정되어 있지 않다. 이와 달리, 위의 세 단계는 임의의 순서로 완료될 수 있다. 어느 순서가 사용되는지에 무관하게, 단일 중재 위너가 각각의 사이클에서 선택된다. 그런 다음, 위닝 트랜잭션이 인터커넥트(12)와 연관된 대응하는 가상 채널을 통해 전송된다.

[0086] 중재 요소(26)에 의해 수행되는 각각의 중재 (1), (2) 및 (3)에 대해, 다수의 중재 스킴 또는 규칙 세트가 사용될 수 있다. 이러한 중재 스킴은 엄격한 또는 절대 우선순위, 4개의 가상 채널 각각에 특정 퍼센티지의 트랜잭션 트래픽이 할당되는 가중화된 우선 순위 또는 트랜잭션이 지정된 시퀀스 순서로 가상 채널에 할당되는 라운드-로빈 스킴을 포함할 수 있다. 추가 실시예에서, 그 밖의 다른 우선순위화 스킴이 사용될 수 있다. 또한, 중재 요소(26)는 때때로 상이한 중재 스킴들을 동적으로 스위칭하거나 및/또는 각각의 (1), (2) 및 (3) 중재에 대해 동일하거나 상이한 중재 스킴을 사용할 수 있다.

[0087] 선택적 실시예에서, 주어진 중재 사이클 동안 고려되는 미해결 트랜잭션(들)에 의해 정의되는 도착지 포트(20)의 사용성이 고려된다. 도착지 포트(20) 내 버퍼가 주어진 트랜잭션을 처리하기 위해 이용 가능한 자원을 갖지 않는 경우, 대응하는 가상 채널이 이용 가능하지 않다. 결과적으로, 관심 트랜잭션은 중재에서 경쟁하지 않고,

오히려, 타깃 자원이 이용 가능해질 때 후속 중재 사이클까지 기다린다. 다른 한편으로, 타깃 자원(들)이 이용 가능할 때, 인터커넥트(12)로의 액세스를 위해 대응하는 거래가 중재되고 경쟁된다.

[0088] 도착지 포트(20)의 가용성은 앞서 언급한 복수의 중재 단계 (1), (2) 및 (3)과 관련하여 상이한 시간에 체크될 수 있다. 예를 들어, 중재 사이클 전에(즉, 단계 (1), (2) 및 (3) 중 하나를 완료하기 전에) 가용성 체크가 수행될 수 있다. 결과적으로, 이용 가능한 도착지 자원을 정의하는 트랜잭션만 후속 중재 중에 고려된다. 대안으로, 가용성 체크는 구현 순서에 관계없이 세 가지 중재 단계 (1), (2) 및 (3) 중 임의의 하나 중간에 수행될 수 있다.

[0089] 중재 프로세스에서 이르게 또는 늦게 도착지 자원 가용성 체크를 수행하는 이점과 단점이 있다. 체크를 일찍 수행함으로써, 트랜잭션의 가능한 경쟁 부분의 도착지가 이용 가능하지 않은 경우 이들이 경쟁으로부터 제거될 가능성이 있다. 그러나 가용성의 이른 통지는 시스템 자원에 상당한 양의 오버헤드를 발생시킬 수 있다. 따라서 환경에 따라, 주어진 중재 사이클에서 가용성 체크를 나중에 수행하는 것이 더 실용적일 수 있다.

[0090] 트랜잭션 클래스의 선택을 포함하는 중재 단계를 위해, 다수의 규칙이 N, NP 및 C의 경쟁 부분 간 중재를 위해 정의된다. 이들 규칙은 다음과 같다:

[0091] **포스트(P) 트랜잭션의 경우:**

[0092] - 포스트 트랜잭션 부분이 또 다른 포스트 트랜잭션 부분을 전달하지 않을 수 있다.

[0093] - 포스트 트랜잭션 부분은 데드록 상태(deadlock)를 피하기 위해 비-포스트 트랜잭션 부분을 전달할 수 있어야 한다.

[0094] - 포스트 트랜잭션 부분은 둘 모두가 강력한 순서 모드에 있는 경우 완료를 전달할 수 있어야 한다. 즉, 강력한 모드에서 트랜잭션은 규칙에 따라 엄격하게 수행되어야 하며 규칙은 완화될 수 없다.

[0095] - 포스트 요청은 완료를 전달하도록 허가되지만, 임의의 트랜잭션 부분이 이의 완화된 순서(RO) 비트 세트를 갖는 경우, 통과는 의무가 아니다. 완화된 순서를 사용하면 일반적으로 규칙을 따르지만 예외가 있을 수 있다.

[0096] **비-포스트(NP) 트랜잭션의 경우:**

[0097] - 비-포스트 트랜잭션 부분이 포스트 트랜잭션 부분을 전달하지 않아야 한다.

[0098] - 비-포스트 트랜잭션 부분이 또 다른 비-포스트 트랜잭션 부분을 전달하지 않아야 한다.

[0099] 둘 모두가 강력한 순서 모드에 있는 경우 비-포스트 트랜잭션 부분이 완료를 전달하지 않아야 한다.

[0100] 비-포스트 트랜잭션 부분이 완료를 전달하도록 허용되지만, 어떠한 트랜잭션 부분도 자신의 RO 비트 세트를 갖지 않는 경우 의무는 아니다.

[0101] **완료(C) 트랜잭션의 경우:**

[0102] - 둘 모두 강력한 순서 모드에 있는 경우 완료는 포스트 트랜잭션 부분을 전달하지 않아야 한다.

[0103] - 완료는 포스트 트랜잭션 부분을 전달하도록 허용되지만, 어떠한 트랜잭션 부분도 자신의 RO 비트 세트를 갖지 않는 경우, 이는 의무가 아니다.

[0104] - 둘 모두 강력한 순서 모드에 있는 경우 완료가 비-포스트 트랜잭션 부분을 전달하지 않아야 한다.

[0105] - 완료는 비-포스트 트랜잭션 부분을 전달하도록 허용되지만, 어떠한 트랜잭션 부분도 자신의 RO 비트 세트를 갖지 않는 경우 이는 의무는 아니다.

[0106] 완료는 타 완료를 전달하도록 허용되지 않는다.

[0107] 아래의 표 IV는 PCI 순서화 규칙의 요약을 제공한다. 어떠한 (a) 및 (b) 옵션도 갖지 않는 박스에서, 엄격한 순서화 규칙을 따를 필요가 있다. (a) 및 (b) 옵션이 있는 표의 박스에서 RO 비트가 각각 재설정 또는 설정되었는지에 따라 엄격한 순서(a) 또는 완화된 순서(b) 규칙이 적용될 수 있다. 다양한 대안적인 실시예에서, RO 비트는 패킷 레벨에서 전역적으로 또는 개별적으로 설정되거나 재설정될 수 있다.

표 4

[0108]

행 전달 열?	포스트 요청 (열 2)	비-포스트 요청 (열 3)	완료 (열 4)
포스트 요청 행 A	아니오	예	(a) 예 (b) 예/아니오
비-포스트 요청 행 B	아니오	아니오	(a) 아니오 (b) 예/아니오
완료 행 C	(a) 아니오 (b) 예/아니오	(a) 예 (b) 예/아니오	아니오

[0109] 중재 요소(26)는 경쟁하는 포트(20), 가상 채널 및 트랜잭션 클래스 각각 사이에서 특별한 순서 없이 중재를 수행함으로써 최종 위닝 트랜잭션 부분을 선택한다. 사이클당 위닝 부분이 공유 인터커넥트(12)에의 액세스를 얻고 대응하는 가상 채널을 통해 전송된다.

[0110] 도 3b를 참조하면, 디바이스 순서화를 이용하는 중재 요소(26)에 의해 수행되는 중재 로직을 도시하는 논리도가 도시된다. 중재 프로세스, 및 가능한 이용 가능한 도착지 자원의 고려가 두 가지 차이점을 제외하고, 상기에서 기재된 것과 본질적으로 동일하다.

[0111] 첫째, 디바이스 순서화의 경우, 단 2개의 트랜잭션 클래스만 정의되며, (a) 모든 요청에 대한 응답이 필요한 비-포스트 읽기 또는 쓰기 트랜잭션 및 (b) 필요한 응답을 정의한 완료 트랜잭션이 그것이다. 트랜잭션 클래스가 두 개뿐이므로 각각의 포트(20) 내 가상 채널당 두 개의 버퍼만 있다. 예를 들어 4개의 가상 채널(VC0, VC1, VC2 및 VC3)이 있는 경우, 각각의 포트(20)(가령, 포트 0, 포트 1 및 포트 2)는 총 8개의 버퍼를 가진다.

[0112] 둘째, 디바이스 순서화를 위한 트랜잭션을 선택하는 규칙도 PCI 순서화와 상이하다. 디바이스 순서화의 경우 오버 클래스보다 한 클래스를 선택하는 데 적용되는 엄격한 규칙이 없다. 반대로 어느 트랜잭션 클래스라도 임의로 선택될 수 있다. 그러나 일반적인 프랙티스는 완료 트랜잭션이 해결될 때까지 가용하지 않을 수 있는 자원을 비우기 위해 선호되는 완료 트랜잭션을 필요로 한다.

[0113] 그렇지 않으면, 디바이스 순서에 대한 중재 프로세스가 앞서 기재된 바와 동일하다. 즉, 각각의 중재 사이클에 대해, 중재 위너를 선택하기 위해 중재 단계 (1), (2) 및 (3)이 특정 순서로 수행된다. 트랜잭션 클래스 중재가 수행될 때 PCI 순서 규칙이 아닌 디바이스 순서가 사용된다. 또한, 도착지 자원 및/또는 가상 채널의 가용성이 또한 중재 단계 (1), (2) 및 (3)의 이전에 또는 중간에 고려될 수도 있다.

[0114] **동작 흐름도**

[0115] 앞서 언급한 바와 같이, 앞서 기재된 중재 스킴은 임의의 공유 자원으로의 액세스를 공유하기 위해 사용될 수 있으며 단지 공유 인터커넥트에 대한 것으로 제한되지 않는다. 이러한 그 밖의 다른 공유 자원은 ARL(28), 처리 자원, 메모리 자원, 가령 LUT(30)를 포함하거나, 액세스를 위해 경쟁하는 복수의 파티들 간에 공유되는 그 밖의 다른 임의의 자원을 포함할 수 있다.

[0116] 도 4를 참조하면, 공유 자원으로의 액세스를 중재하기 위한 동작 단계를 도시하는 흐름도(40)가 도시된다.

[0117] 단계(42)에서, 다양한 출발지 서브-시스템 에이전트(14)가 트랜잭션을 생성한다. 트랜잭션은 세 가지 클래스, 가령, 포스트(P), 비-포스트(NP), 및 완료(C) 중 임의의 것일 수 있다.

[0118] 단계(44)에서, 출발지 서브-시스템 에이전트(14)에 의해 생성된 각각의 트랜잭션이 패킷화된다. 앞서 언급 한 바와 같이, 주어진 트랜잭션의 패킷화는 하나 또는 복수의 패킷을 도출할 수 있다. 패킷은 또한 크기가 다를 수 있으며, 일부 패킷은 큰 페이로드를 가지며 또 다른 패킷은 작은 페이로드를 갖거나 갖지 않는다. 트랜잭션이 인터커넥트(12)의 폭보다 작은 데이터 페이로드(36)를 갖는 단일 패킷으로 표현되는 상황에서, 트랜잭션은 단일 부분으로 표현될 수 있다. 트랜잭션이 복수의 패킷으로 표현되거나 공유 자원의 액세스 폭보다 큰 데이터 페이로드(36)를 갖는 단일 패킷으로 표현되는 상황에서, 트랜잭션을 표현하기 위해 복수의 부분이 필요하다.

[0119] 단계(46)에서, 각각의 서브-시스템 에이전트(14)에 의해 생성된 패킷화된 트랜잭션의 일부(들)가 대응하는 포트(20)를 통해 로컬 스위치(16)로 주입된다. 포트(20) 내에서 각각의 트랜잭션의 패킷은 가상 채널에 할당된다. 앞서 언급했듯이, 할당은 모든 가상 채널, 보안, 긴급성 등의 균형 잡힌 사용을 기반으로 임의적으로 하드 코딩될 수 있다.

[0120] 단계(48)에서, 각각의 서브-시스템 에이전트(14)에 의해 생성된 패킷화된 트랜잭션의 일부(들)는 두 트랜잭션 클래스 모두에 의해 그리고 할당된 가상 채널(가령, VC0, VC1, VC2 및 VC3)에 의해 적절한 선입 선출 버퍼에 저장된다. 앞서 언급한 바와 같이, 가상 채널은 다수의 상이한 우선순위 스킴, 가령, 엄격한 또는 절대 우선순위, 라운드 로빈, 가중화된 우선 순위, 최소 최근 서비스 등 중 하나에 의해 할당될 수 있다. 주어진 트랜잭션이 복수의 부분을 갖는 경우, 각각의 부분은 동일한 버퍼에 저장될 것이다. 결과적으로, 주어진 트랜잭션의 복수의 부분은 인터커넥트(12)와 연관된 동일한 가상 채널을 통해 전송된다. 트랜잭션 부분이 주입됨에 따라, 각각의 버퍼에서 콘텐츠 아이템의 수를 추적하기 위한 대응하는 카운터가 감분된다. 특정 버퍼가 채워지면 이의 카운터가 0으로 감분하는데, 이는 버퍼가 더 이상 추가 콘텐츠를 받을 수 없음을 의미한다.

[0121] 단계(50, 52 및 54)에서, 제1, 제2 및 제 3 레벨 중재가 수행된다. 앞서 언급한 바와 같이, 포트(20), 가상 채널 및 트랜잭션 클래스의 선택은 임의의 순서로 수행될 수 있다.

[0122] 요소(56)는 제1, 제2 및 제 3 레벨의 중재를 수행하는 데 사용되는 규칙을 유지하는 데 사용될 수 있다. 각각의 경우에, 요소(56)는 각각의 중재 레벨을 해결하는데 필요에 따라 사용된다. 예를 들어, 요소(56)는 PCI 및/또는 디바이스 순서화 규칙을 유지할 수 있다. 요소(56)는 또한 복수의 우선순위 스킴, 가령, 엄격하거나 절대적인 우선순위, 가중화된 우선순위, 라운드 로빈 등을 구현하기 위한 규칙, 및 주어진 중재 사이클에서 어느 것이 사용될지를 결정하기 위한 논리 또는 지능을 유지할 수 있다.

[0123] 단계(58)에서, 중재의 위너가 결정된다. 단계(60)에서, 위닝 부분은 공유 자원을 액세스하기 위해 사용되는 버퍼에 배치되고 버퍼와 연관된 카운터가 감분된다.

[0124] 단계(62)에서, 위닝 부분이 버퍼에 더는 없기 때문에 위닝 부분과 연관된 버퍼가 증분된다.

[0125] 단계(64)에서, 위닝 부분은 공유 자원에 대한 액세스 권한을 얻는다. 액세스가 완료되면 공유 자원에 대한 버퍼가 증분된다.

[0126] 단계(42 내지 64)는 각각 연속적인 클록 사이클 동안 연속적으로 반복된다. 상이한 위닝 부분으로서, 각각이 공유 자원에 대한 액세스 권한을 획득한다.

[0127] **인터리빙 - 예시 1**

[0128] 트랜잭션은 여러 모드 중 하나에서 인터커넥트(12)를 통해 전송될 수 있다.

[0129] "헤더 인-라인(header in-line)"이라고 하는 하나의 모드에서, 각각 개별 부분 또는 비트에서 트랜잭션의 패킷(들)(32)의 헤더(34)가 항상 페이로드(36)에 앞서 먼저 전송된다. 인터커넥트(12)의 데이터 비트 수 N과 관련하여 헤더(34) 및/또는 페이로드(36)의 상대 크기에 따라, 헤더 인-라인 모드는 인터커넥트(12) 상에서 이용 가능한 비트를 낭비하거나 낭비하지 않을 수 있다. 예를 들어, 512 비트 폭(N = 512)인 인터커넥트(12) 및 128 비트인 헤더 및 256 비트의 페이로드를 갖는 패킷을 고려할 수 있다. 이 시나리오에서, 헤더의 128 비트는 첫 번째 부분 또는 비트로 전송되는 반면, 인터커넥트(12)의 대역폭의 나머지 384 비트가 사용되지 않는다. 두 번째 부분 또는 비트에서, 페이로드(36)의 256 비트가 전송되는 반면, 인터커넥트(12)의 나머지 256 비트는 사용되지 않는다. 이 예에서는 두 비트 동안 인터커넥트의 대역폭의 상당 퍼센티지가 사용되지 않는다. 반면에 트랜잭션의 패킷의 대부분이 인터커넥트와 같거나 큰 크기를 갖는 경우, 낭비되는 대역폭의 정도가 감소되거나 가능한 제거된다. 예를 들어 384 또는 512 비트인 헤더 및/또는 페이로드를 사용하면, 낭비되는 양이 크게 감소되거나(가령, 384 비트) 함께 제거된다(가령, 512 비트).

[0130] "측대역 상의 헤더"라고 하는 또 다른 모드에서, 패킷의 헤더(34)는 데이터의 "측대역에서" 전송되는데, 이는 제어 비트 M을 사용하는 반면, 페이로드(36)가 인터커넥트(12)의 N개의 데이터 비트를 통해 전송되는 것을 의미한다. 측대역 모드 상의 헤더 모드의 경우, 패킷(32)의 페이로드(36)의 비트의 수 또는 크기가 주어진 인터커넥트(12)를 통해 패킷을 전송하는 데 피룡한 비트의 수를 결정한다. 예를 들어, 64, 128, 256 또는 512 비트의 페이로드(36)를 갖는 패킷(32) 및 128 데이터 비트(N = 128)를 갖는 인터커넥트(12)의 경우, 패킷은 각각 1, 1, 2 및 4 비트를 필요로 한다. 각각의 비트(들)의 전송과 함께, 헤더 정보는 인터커넥트(12)의 N 데이터 비트를 통해 페이로드의 데이터와 함께 또는 "측면에서" 제어 비트 M을 통해 전송된다.

[0131] 또 다른 모드에서, 패킷(32)의 헤더(34)가 페이로드에 맞춰 전송되지만, 헤더(34) 및 페이로드(36)가 개별 부분 또는 비트로 전송되어야 할 필요는 없다. 패킷(32)이 128 비트인 헤더(34)와 128 비트인 페이로드(36)를 갖는 경우 총 크기는 256 비트(128 + 128)이다. 인터커넥트(12)의 N 데이터 비트가 64, 128, 256 또는 512 비트 폭인 경우, 256 비트의 패킷이 각각 4, 2, 1 및 1 비트로 전송된다. 또 다른 예에서, 패킷(32)은 128 비트의 헤더

와 256 비트의 페이로드(36), 즉 384 비트(128 + 256)의 총 패킷 크기를 가진다. 64, 128, 256 또는 512 폭의 N 데이터 비트의 동일한 인터커넥트(12) 폭의 경우, 패킷은 각각 6, 3, 2 또는 1 비트로 전송된다. 이 모드는 항상 적어도 앞서 기재된 헤더 인-라인 모드만큼 효율적 또는 더 효율적일 것이다.

[0132] 도 5를 참조하면, 복수의 가상 채널을 통한 상이한 트랜잭션의 부분의 인터리빙의 제1 예시가 도시된다. 이 예에서는 단순성을 위해 단 두 개의 트랜잭션만 정의된다. 두 트랜잭션은 이 예에서 128 데이터 비트 폭(N = 128)인 공유 인터커넥트(12)에 대한 액세스를 위해 경쟁하고 있다. 두 트랜잭션의 세부 사항은 다음과 같다:

[0133] (1) 시점 T1에서 생성되었으며 가상 채널 VC2에 할당된 트랜잭션 1 (T1). T1의 크기는 T1A, T1B, T1C 및 T1D로 지정된 4 비트이다.

[0134] (2) (시점 T1 이후의) 시점 T2에서 생성되며 가상 채널 VC0로 할당된 트랜잭션 2(T2) T2의 크기는 단일 부분 또는 비트이다.

[0135] 이 예에서, VC0는 절대적 또는 엄격한 우선 순위가 할당 받는다. 복수의 사이클 과정에서, 두 트랜잭션 T1 및 T2의 일부는, 도 5에 도시된 바와 같이, 두 개의 트랜잭션(T1 및 T2)이 공유 인터커넥트를 통해 전송된다.

[0136] 사이클 1: T1의 비트 T1A는 유일한 가용한 트랜잭션이기 때문에, VC2를 통해 전송된다.

[0137] 사이클 2: T1의 비트 T1B 및 T2의 단일 부분이 인터커넥트(12)로의 액세스를 위해 경쟁 중이다. VC0가 엄격한 우선순위를 갖기 때문에 자동으로 T2가 위닝한다. 따라서 T2의 비트는 VC0을 통해 전송된다.

[0138] 사이클 3: 경쟁하는 트랜잭션이 없기 때문에, T1의 비트 T1B가 VC2를 통해 전송된다.

[0139] 사이클 4: 경쟁 거래가 없기 때문에 T1의 비트 T1C가 VC2를 통해 전송된다.

[0140] 사이클 5 : 경쟁 거래가 없기 때문에 T1의 비트 T1D가 VC2를 통해 전송된다.

[0141] 이 예시는 (1) 절대적 우선순위를 갖는 가상 채널의 경우, 다른 트래픽이 이전에 대기 중인지 여부와 무관하게, 공유 인터커넥트(12)로의 액세스가 트래픽이 이용 가능해질 때마다 즉시 수여되고, (2) 상이한 트랜잭션의 위닝 부분들 또는 비트들이 인터리빙되고 인커넥트(12)와 연관된 상이한 가상 채널을 통해 전송되는 것을 예시한다. 이 예에서, 가상 채널 VC0는 절대적인 우선 순위를 부여 받았다. 절대적 또는 엄격한 우선순위 스킴의 경우, 가상 채널 중 임의의 것이 가장 높은 우선순위를 할당 받을 수 있다.

[0142] **인터리빙- 예시 2**

[0143] 도 6을 참조하면, 복수의 가상 채널을 통한 상이한 트랜잭션의 일부분의 인터리빙의 제2 예시가 도시된다.

[0144] 이 예에서, 인터커넥트(12)로의 액세스에 대한 우선순위 스킴이 가중화되며, 각각 VC0에 시간의 액세스(40%)가 수여되고 VC1-VC3 각각에 시간의 액세스(20%)가 수여된다. 또한, 인터커넥트가 128 비트 폭이다.

[0145] 또한 이 예에서, T1, T2, T3 및 T4의 4개의 경쟁 트랜잭션이 존재한다.

[0146] - T1이 VC0에 할당되고 4개의 부분 또는 비트 T1A, T1B, T1C 및 T1D를 포함한다.

[0147] - T2가 VC1에 할당되고 2개의 부분 또는 비트 T2A 및 T2B를 포함한다.

[0148] - T3은 VC2에 할당되고 2개의 부분 또는 비트 T3A 및 T3B를 포함한다.

[0149] - T4는 VC3에 할당되며 2개의 부분 또는 비트 T4A 및 T4B를 포함한다.

[0150] 이 예에서는 우선순위 스킴이 가중화된다. 결과적으로 각각의 가상 채널은 이의 가중치 비율에 따라 위닝할 것이다. 즉, 10개의 사이클 동안 VC0는 4회 위닝하고 VC1, VC2 및 VC3는 각각 2회 승리할 것이다. 예를 들어, 도 6에 도시된 바와 같이:

[0151] - T1의 4개의 부분 또는 비트 T1A, T1B, T1C 및 T1D는 10회 사이클 중 4회 사이클(40%)(즉, 사이클 1, 4 7 및 10) 동안 VC0를 통해 전송된다.

[0152] - T2의 T2A 및 T2B의 2개의 부분 또는 비트가 10회 사이클 중 2회 사이클(즉, 사이클 2 및 사이클 6)에서 VC1을 통해 전송된다.

[0153] - T3의 T3A 및 T3B의 2개의 부분 또는 비트는 10회 사이클 중 2회 사이클(20%)(즉, 사이클 5 및 사이클 9)에서 VC2를 통해 전송된다.

[0154] - T4의 T4A와 T4B의 2개의 부분 또는 비트는 10회 사이클 중 2회 사이클(20%)(즉, 사이클 3 및 사이클 8) 동안 전송된다.

[0155] 따라서 이 예는 다음을 예시한다: (1) 지정 비율을 기초로 각각의 가상 채널에 인터커넥트(12)로의 액세스가 수여되는 가중화된 우선순위 스킴 및 (2) 상이한 트랜잭션의 위닝 부분이 인터리빙되고 인터커넥트(12)와 연관된 상이한 가상 채널을 통해 전송되는 또 다른 예시.

[0156] 이 가중화된 예시의 경우, 가중화된 비율에 따라, 다양한 가상 채널로 트랜잭션의 일부분을 할당하기 위한 충분한 트래픽이 존재함이 이해되어야 한다. 반면에 트래픽 양이 불충분한 경우 가중화된 비율이 엄격하게 적용되거나 엄격하지 않게 적용될 수 있다. 예를 들어, 가상 채널 VC3 상에 많은 트래픽이 존재하고 나머지 가상 채널 VC0, VC1 및 VC2 상에 트래픽이 없는 것으로 제한되는 경우, 가중화 비율이 엄격하게 적용되면 VC3가 트래픽의 전부 또는 대부분을 전송할 것이다. 그러나 결과적으로, 트랜잭션의 일부가 매 클록 사이클 또는 비트마다 전송되지 않을 수 있기 때문에 인터커넥트(12)가 충분히 활용되지 않을 수 있다. 반면에 가중화 비율이 엄격하게 적용되지 않으면 인터커넥트의 활용도를 높이기 위해 트랜잭션 트래픽을 재 할당할 수 있다(가령, 트래픽이 더 많은 사이클 또는 비트를 통해 전송 됨).

[0157] 상기 두 가지 예는 앞서 기재된 전송 모드 중 어느 것이 사용되는지에 무관하게 적용 가능하다. 트랜잭션(들)이 부분 또는 비트로 분할되면, 이들은 인터리빙되어 본 명세서에 정의된 중재 스킴 중 임의의 것을 이용해 공유 인터커넥트(12)를 통해 전송될 수 있다.

[0158] 앞서 기재된 중재 스킴은 단지 몇 가지 예만 나타낸다. 또 다른 예에서, 낮은 지터, 가중화, 엄격, 라운드-로빈 또는 그 밖의 다른 임의의 유사한 중재 스킴이 사용될 수 있다. 따라서 본 명세서에 나열되거나 기재된 중재 스킴은 예시로서 간주되어야 하고 어떠한 방식으로도 한정이 아니다.

[0159] **복수의 동시 중재**

[0160] 지금까지 단순함을 위해 단일 중재만 기재하였다. 그러나 SoC와 같은 실제 응용 프로그램에서는 복수의 중재가 동시에 발생할 수 있음을 이해해야 한다.

[0161] 도 7을 참조하면, 스위치(16, 18) 간 두 개의 방향으로 트래픽을 처리하기 위한 두 개의 공유 인터커넥트(12, 12Z)의 블록도가 도시되어있다. 앞서 기재된 바와 같이, 스위치(16)는 공유 인터커넥트(12)를 통해 출발지 서브-기능(14)(즉, IP_1, IP_2 및 IP_3)에서 도착지 서브-기능(14)(즉, I_{P4}, I_{P5} 및 I_{P6})으로 트랜잭션 트래픽을 전달하는 역할을 한다. 반대 방향으로 트랜잭션 트래픽을 처리하기 위해, 스위치(18)는 중재 요소(26Z) 및 선택적으로 ARL(28Z)을 포함한다. 작동 중에 요소(26Z) 및 ARL(28Z)은 앞서 설명된 것을 보완하여 동작하는, 즉, 출발지 IP 에이전트(14)(즉, IP_4, IP_5 및 IP_6)에 의해 생성된 트랜잭션 트래픽이 중재되고 공유 인터커넥트(12Z)를 통해 도착지 IP 에이전트(즉, IP_1, IP_2 및 IP_3)로 전송된다. 대안으로, 중재는 ARL(28Z) 없이 수행 될 수 있는데, 즉, 중재는 단순히 경쟁 포트(20)(가령, 포트 3, 포트 3 또는 포트 5) 중에서 결정하고, 부분의 최종 도착지와 무관하게, 위닝 포트와 연관된 트랜잭션 부분이 인터커넥트(12)를 통해 전송된다. 요소(12Z, 26Z 및 28Z)가 앞서 기재된 바 있기 때문에, 간결성을 위해 본 명세서에서는 상세한 설명을 제공하지 않는다.

[0162] SoC에, 복수의 레벨의 서브-기능(14) 및 복수의 공유 인터커넥트(12)가 존재할 수 있다. 각각에 대해, 앞서 기재된 중재 스킴이 다양한 서브-기능들 간 인터커넥트(12)를 통해 전송되는 트랜잭션들을 중재하는 데 사용될 수 있다.

[0163] **IP 에이전트 리셋 및 전력 관리**

[0164] 도 8을 참조하면, 리셋 및 전력 관리 기능을 갖는 SoC(800)의 블록도가 도시된다. SoC(800)는 인터커넥트(802), 복수의 IP 에이전트(14)(가령, 에이전트 1 내지 에이전트 N), IP 에이전트(14)를 인터커넥트(802)로 연결 또는 결합하는 하나 이상의 링크(803), 및 시스템 제어기(804)를 포함한다. 도시되지는 않지만, 각각의 IP 에이전트(14)가 리셋 입력 명령을 수신하기 위한 하나 이상의 전용 "하드-와이어" 입력을 더 포함할 수 있다. 이러한 명령은 다수의 출발지, 가령, SoC, 시스템 제어기(804), 또는 또 다른 IP 에이전트(14) 등으로부터 올 수 있다.

[0165] 다양한 실시예에서, IP 에이전트(14)는 개별적이고 다양한 상이한 기능을 구현할 수 있다.

[0166] 인터커넥트(802)는 다양한 유형의 인터커넥트, 가령, 네트워크 온 칩(NoC), 버스, 스위칭 네트워크 등일 수 있다.

[0167] 다양한 실시예에서, 링크(803)는 각각의 IP 에이전트(14)와 인터커넥트(802) 간 전용 링크 또는 버스일 수 있다. 대안으로, 인터커넥트(802)로의 액세스가 하나의 링크(803)를 이용해 복수의 IP 에이전트(14) 간에 공유될 수 있고 중재 스킴이 사용되어 경쟁 IP 에이전트(14) 중에서 선택하도록 사용된다. 또 다른 실시예에서, 다수의 가상 채널이 하나 이상의 링크(803), 가령, 앞서 기재된 공유 링크와 연관된 가상 채널과 연관될 수 있다.

[0168] 시스템 제어기(804) 및 관리자(806, 808 및 809)는 다수의 방식으로 구현될 수 있다. 예를 들어, CPU 또는 마이크로제어기로서, 프로그램 가능 로직으로서, SoC(800) 상의 모든 또는 대부분의 시스템 제어 기능을 핸들링하기 위한 복합 상태 머신, 소수 예외 상황을 핸들링하기 위한 단순 상태 머신으로서, 또는 이들의 조합으로서 구현될 수 있다. 시스템 제어기(804)는 도시된 바와 같이 SoC(800) 상에 위치하거나 SoC(800)로부터 떨어져서 위치할 수 있다(도시되지 않음). 상태 머신이 사용되는 경우, 상태들 및 상태들 간 변환이 SoC(800)로 하드-코딩되는 것이 일반적이다.

[0169] 또 다른 실시예에서, 리셋, 파워 및/또는 작업거부(quiescent, 휴지) 관리자(806, 808 및 809) 중 하나 이상이 도시된 바와 같이 시스템 제어기(804) 내에서 중앙 집중화될 수 있다. 대안으로, 각각의 관리자(806, 808 및/또는 809)는 SoC(800) 상의 다양한 위치를 통해 분산되어 있거나 심지어 SoC를 벗어나 있다. 각각의 리셋 관리자(806), 전력 관리자(808) 및 작업거부 관리자(809)는 소프트웨어, 하드웨어, 프로그램 가능 로직, 상태 머신 또는 그 밖의 다른 임의의 적합한 수단으로 구현될 수 있다.

[0170] 리셋 관리자(806)는 SoC(800) 상의 다양한 IP 에이전트(14)의 리셋으로부터 벗어남을 조직화된 방식으로 관리한다. IP 에이전트(14)의 리셋이 많은 상황에서 필수 또는 바람직할 수 있다. 예를 들어, SoC(800)에 제공되는 전력의 제거 또는 차단 또는 SoC(800)의 시스템 단위 리셋 후에 "콜드 리셋"이 발생한다. 대안으로, (콜드 리셋(cold reset)과 유사하게) IP 에이전트(14) 중 하나, 그룹, 또는 전부가 리셋될 때 "웜 리셋(warm reset)"이 발생하지만, SoC(800)로부터 전력이 제거되거나 차단되지 않는다. 웜 리셋은 SoC(800) 상에서 또는 외부에서 기원하는 시그널링을 통해 구현될 수 있다. 리셋이 개시되는 방식과 무관하게, 리셋 관리자(806)가 IP 에이전트(14) 또는 IP 에이전트(14)들의 리셋으로부터의 벗어남을 조직화된 방식으로 관리한다.

[0171] IP 에이전트(14)가 어떤 이유로 오작동하는 경우, 이는 재설정될 필요가 있을 수 있다. 오작동하는 IP 에이전트(14)의 예시가 IP 에이전트(14)가 응답하지 않거나, 에러 상태이거나, 잘못된 트랜잭션을 생성하고 있는 상황을 포함한다. 또 다른 예에서, IP 에이전트(14)는 저전력 상태, 가령, 이하에서 기재된 복수의 전력 절약 모드 중 하나를 빠져 나올 때 리셋 동작을 겪을 필요가 있을 수 있다.

[0172] 전력 관리자(808)는 다양한 IP 에이전트(14)를 저전력 상태, 일반적으로 복수의 전력 절약 모드 중 하나로 두는 프로세스를 관리한다. 모드에 따라, 전력 관리자(808)는 필요에 따라 IP 에이전트(14)를 리셋하기 위해 리셋 관리자(806)와 협업하여 동작할 수 있다.

[0173] 작업거부 관리자(quiesce manager)가 시스템 제어기(804), 리셋 관리자(806), 전력 관리자(808) 및 인터커넥트(802)와 협업하여 동작하여, (1) 동작 또는 오작동 IP 에이전트(14)를 IP 에이전트가 동작 불능인 리셋 또는 전력 절약 모드로 전환, (2) 인터커넥트와 IP 에이전트(802) 간 링크(803)를 작업거부 상태로 두고, (3) 동작 불능인 동안 인터커넥트에게 IP 에이전트에 대한 프록시로서 동작하도록 지시할 수 있다.

[0174] 메모리(810)는 휘발성 및 비휘발성 유형의 메모리 모두를 포함할 수 있다. 덧붙여, 메모리(810)는 SoC(800) 상에 중앙집중되거나 시스템 제어기(804), 인터커넥트(802), 링크(803), 및 관리자(806, 808 및/또는 809) 중 임의의 것 간에 널리 분산될 수 있다. 또 다른 실시예에서, 메모리(810)의 부분 또는 전부가 SoC(800)에서 벗어나 제공될 수 있다.

[0175] 일반적으로 메모리(810)의 휘발성 부분이, 시스템 제어기(804), 관리자(806, 808, 809), 인터커넥트(802), IP 에이전트(14) 등에 의해 생성된 현재 데이터가 저장되는 시스템 메모리에 대해 사용된다. 이러한 메모리는 다양한 캐시, SRAM, DRAM 등을 포함할 수 있다.

[0176] 메모리(810)의 비-휘발성 또는 영속 부분이 SoC(800)에 대한 "부트-업" 코드를 저장하기 위해 사용되는 것이 일반적이다. 부트 코드에 의해, 관리자(806, 808, 809), 인터커넥트(802) 및 IP 에이전트(14)를 포함하는 시스템 제어기(803) 각각이, 전력이 공급된 후 동작을 개시하는 데 필요할 때 자신의 운영 시스템 및/또는 그 밖의 다른 시스템 소프트웨어를 로딩할 수 있다. 일반적으로 리부팅 프로세스는 완료될 때, 각각의 IP 에이전트(14)를 포함하는 전체 시스템이 자신의 보통 동작을 수행하게 하는 다수의 셀프-테스트를 포함한다. 비-휘발성 또는 영속 부분이 NVRAM(non-volatile random-access memory), EEPROM (electrically erasable programmable read only memory), 하드 드라이브, CD ROM 등을 이용해 구현될 수 있다.

[0177] 리셋 관리자(806)는 임의의 IP 에이전트(14)의 리셋으로부터의 벗어남을 조직화된 방식으로 조율한다. 본 명세서에 언급된 바와 같이, 주어진 IP 에이전트(14)의 리셋이 임의의 이유로 발생할 수 있는데, 가령, (1) 전체 SoC(800)가 외부 리셋, 재-시작 커맨드 또는 파워-온 이벤트 후 리셋으로부터 벗어날 때, 또는 (2) 오작동으로 인해, 파워 다운 또는 수면 모드 후에 SoC(800)의 동작 동안 개별 IP 에이전트(14)가 리셋할 때 발생할 수 있다. 이유와 무관하게, 내부 리셋 시퀀스가 완료되면 주어진 IP 에이전트(14)는 인터커넥트(802)로 도입될 준비가 된다. 리셋으로부터 벗어나면, 링크(803)를 통해 인터커넥트(802) 상에서 IP 에이전트(14)와 이의 IP 포트(20) 간 협상이 조율된다.

[0178] 도 9를 참조하면, IP 에이전트(14)와 인터커넥트(802) 간 예시적 IP 에이전트 리셋 협상 시퀀스를 나타내는 흐름도가 도시되어 있다.

[0179] 초기 단계(902)에서, IP 에이전트(14)가 리셋으로부터 벗어나고 인터커넥트(802)로 도입될 준비가 됐는지 여부에 대한 결정이 이뤄진다. 벗어남이 발생할 때, IP 에이전트(14)를 인터커넥트(802)로 재도입하기 위한 후속 단계(904 내지 912)가 뒤 따른다.

[0180] 단계(904)에서, 인터커넥트(802)가 주기적 간격으로 IP 에이전트(14)에 대한 질의를 생성한다. 각각의 질의에 대해, 인터커넥트(802)는 IP 에이전트(14)에게 "어웨이크(awake)"(즉, 트랜잭션이 준비되었는지, 즉, 수신된 트랜잭션을 전송 또는 처리할 수 있는지) 여부를 묻는다.

[0181] 결정(906)에서, 인터커넥트가 IP 에이전트(14)로부터의 질의에 대한 긍정 응답을 수신했는지 여부를 결정한다. 그렇지 않은 경우, 인터커넥트(802)는 질의를 계속 전송한다. 그런 경우, IP 에이전트(14)가 자신의 리셋 루틴을 부분적으로 완료했고 협상의 다음 단계에 준비가 됐음을 인터커넥트(802)에게 나타낸다.

[0182] 단계(908)에서, 인터커넥트(802) 및 IP 에이전트(14)가 자신들의 크레딧 정보를 교환함으로써 자신들의 협상을 계속한다. 인터커넥트(802) 및 IP 에이전트(14) 각각이 이용 가능한 수의 비트를 서로 교환한다(즉, 클록 사이클 단위로 링크(803)를 통해 전송될 수 있는 데이터의 양). 링크(803)의 반대 측부 상의 각각의 파트너는, 교환 후, 상대방이 이 협상의 결과로서 갖는 이용 가능한 수의 크레딧을 안다.

[0183] 선택사항적 단계(910)에서, 인터커넥트(802) 및 IP 에이전트(14)가 그 밖의 다른 유용한 정보, 가령, 보안 자격증명, 인터커넥트(802)와 IP 에이전트(14)를 연결하는 링크(803)와 연관될 수 있는 협의된 가상 채널 수 등을 교환함으로서 협상을 계속한다.

[0184] 마지막 단계(912)에서, 협상이 완료될 때, IP 에이전트(14)는 "트랜잭션 준비됨"으로 선언된다. 다시 말해, IP 에이전트는 인터커넥트(802)로부터 수신된 들어오는 트랜잭션을 처리하거나 인터커넥트(802)를 통해 또 다른 도착지로의 나가는 트랜잭션을 전송할 준비가 된다. IP 에이전트(14)가 트랜잭션 준비된 상태이면, 인터커넥트(802), 시스템 제어기(804), 및 중간 회로, 로직, 또는 그 밖의 다른 요소를 통해 직접 또는 간접적으로 인터커넥트(802)에 연결되거나 그 밖의 다른 방식으로 연결된 그 밖의 다른 임의의 요소에게 가시적이 된다.

[0185] 리셋 관리자(806)는 오작동 IP 에이전트(14)의 리셋을 조화시킨다. SoC(800)의 동작 동안, IP 에이전트(14)는 오작동할 수 있다(가령, 응답하지 않음, 에러 상태로 들어감, 트랜잭션을 잘못 생성함, 또는 그 밖의 다른 방식으로 오작동). 예를 들어, IP 에이전트는 수신된 트랜잭션을 처리하지 못할 수 있다. 따라서, 트랜잭션을 전송한 기원 IP 에이전트가 응답을 대기하는 것을 행업할 수 있다. 문제의 심각도에 따라, 행업은 기원 IP 에이전트(14), 도착지 IP 에이전트(14)로만 한정되거나 최악 시나리오에서 그 밖의 다른 부분, 심지어 전체 SoC(800)가 부정적인 영향을 받을 수 있다. 따라서, 특정 환경에서, 오작동 IP 에이전트가 문제를 해결하도록 리셋될 필요가 있을 수 있다.

[0186] 도 10을 참조하면, 오작동 IP 에이전트에 대한 리셋 시퀀스를 보여주는 흐름도(1000)가 도시된다.

[0187] 단계(1002)에서, 전송된 트랜잭션을 생성 및/또는 수신된 트랜잭션을 처리함으로써 SoC(800) 상의 다양한 IP 에이전트(14)가 정규 동작한다.

[0188] 결정 단계(1004)에서, 시스템 제어기(804)는 IP 에이전트의 동작을 모니터링한다. 어떠한 문제도 검출되지 않는 경우, IP 에이전트(14)는 자신의 정규 동작을 계속한다. 다른 한편 IP 에이전트가 어떠한 이유로든 오작동하는 경우 리셋 관리자(806)는 이를 오작동 IP 에이전트(14)로 플래그한다.

[0189] 단계(1005)에서, 시스템 제어기(804) 및 인터커넥트(802)는 SoC(800)의 나머지가 추가 문제 없이 동작하는 데 도움이 되는 다수의 프로세스를 개시하도록 더 협업한다. 이들 추가 프로세스는 다음을 포함할 수 있다:

[0190] 1. 시스템 제어기(804)는 인터커넥트(802)가 오작동 IP 에이전트(14)에 의해 생성되는 임의의 추가 트랜잭션을 허용하지 않도록 요청한다.

[0191] 2. 오작동 IP 에이전트(14)를 타깃으로 하는 미결 트랜잭션을 추적한다.

[0192] 3. 인터커넥트(802)는 프록시로서 기능하고 리셋 협상을 하는 동안 오작동 IP 에이전트(14)를 타깃으로 삼는 임의의 트랜잭션에 응답할 수 있다. 예를 들어, 인터커넥트(802)는 처리되지 않은 트랜잭션에 응답하여 예외 메시지를 생성할 수 있다. 프록시로서 동작함으로써, 트랜잭션의 발신자가 오작동 IP 에이전트(14)로부터 응답을 수신하지 않기 때문에 전체 시스템이 행업되는 것을 포함해 훨씬 더 큰 시스템 단위 문제가 피해진다. 다양한 실시예에서, 예외 메시지가 여러 유형일 수 있는데, 가령, IP 에이전트(14)가 이용 가능하지 않거나, IP 에이전트가 저전력 모드에 있거나 그 밖의 다른 경우일 수 있다. 일반적으로, 다양한 유형의 예외 메시지가 사용될 수 있으며, 각각이 발생한 조건 또는 에러를 나타낸다.

[0193] 단계(1006)에서, 리셋 관리자(806)가 오작동 IP 에이전트(14)에 대한 리셋 명령을 생성한다.

[0194] 단계(1007)에서, 리셋될 IP 에이전트(14)와 인터커넥트(802) 간 링크(803)가 작업거부 상태에 놓인다. 이 프로세스는 도 14와 관련하여 더 기재된다.

[0195] 단계(1008)에서, 오작동 IP 에이전트(14)는 인터커넥트(802)를 통해 수신되거나 전용 리셋 와이어를 통해 수신될 수 있는 명령에 응답하여 자신의 리셋 루틴을 개시한다. 이 프로세스는 도 9와 관련하여 앞서 기재된 바와 같이 IP 에이전트(14)가 이의 고유 리셋 프로토콜 또는 루틴을 실행시키는 것 및 (2) 인터커넥트(802)와 협상하는 것을 포함한다.

[0196] 결정 단계(1012)에서, IP 에이전트(14)의 리셋 협상이 완료되는지 여부가 결정된다. 완료될 때, 제어가 단계(1002)로 복귀되고 IP 에이전트(14) 및 SoC(800)의 동작이 일반적으로 재개된다. 앞서 언급된 바와 같이, 리셋 IP 에이전트(14)가 리셋으로부터 벗어난 후 인터커넥트(802) 및 시스템 제어기에게 가시적이 되며 트랜잭션 준비가 된다. 마지막으로, 단계(1014)에서, 이제 리셋 IP 에이전트(14)와 인터커넥트(802) 간 링크(803)가 작업거부 모드를 빠져 나온다. 이 포인트에서, 인터커넥트(802)는 더는 IP 에이전트(14)에 대한 프록시로서 동작할 필요가 없다.

[0197] 전력 관리자(808)는 IP 에이전트(14)를 복수의 파워 다운 모드 중 하나로 둠으로써, IP 에이전트(14)를 저전력 상태로 지능적으로 선택적으로 놓는다. IP 에이전트(14)를 파워 다운 또는 파워 다운 모드로 두는 것이 다양한 이유로 수행될 수 있다.

[0198] 예를 들어, SoC(800)가 배터리 파워 디바이스에서 사용되는 경우, 전력 관리자(808)가 IP 에이전트를 파워 다운 모드로 둬서 제한된 배터리 전력을 보존할 수 있다. 대안으로, 비-배터리 파워 디바이스에서도, 전력 관리자(808)는 비-핵심 IP 에이전트(14)를 저전력 모드로 만들어 과열을 방지할 수 있다. 이들은 전력 관리를 구현하기 위한 가능한 이유이다. 또 다른 이유가 사용되고 있지 않을 때 하나 이상의 IP 에이전트(14)를 파워 다운 모드로 두는 것을 포함할 수 있다. 다양한 대안 실시예에서, 파워 다운 모드는 다음을 포함한다:

[0199] 1. 저전력 모드, 동작: 하나의 대안으로, 경우에 따라 IP 에이전트(14)에 대한 클록 주파수가 느려진다. 대안으로, 경우에 따라 공급 전압이 감소될 수 있다. 또 다른 실시예에서, 전력 소비를 더 감소시키도록 경우에 따라 클록 주파수와 공급 전압 모두 감소될 수 있다. 경우에 따라 클록 주파수 및/또는 공급 전압을 감소시키는 것이 적용될 때만 이뤄지는데, 즉, 모든 IP 에이전트(14)가 감소된 클록 주파수, 감소된 공급 전압, 또는 둘 모두에서 동작할 수 있는 것은 아님이 이해되어야 한다. 또 다른 실시예에서, IP 에이전트(14)가 저전력, 동작 모드를 가진다고 가정할 때 클록 및/또는 공급 전압을 감소시키기 위한 커맨드는, 적용될 때, 시스템 제어기(804) 또는 IP 에이전트(14)로부터 얻어질 수 있다.

[0200] IP 에이전트가 기능 상태를 유지하기 때문에, 인터커넥트(802)는 이 모드에서 상당한 역할을 하지 않을 수 있는데, 이는 IP 에이전트(14)가 응답 자체를 생성할 수 있기 때문에 IP 에이전트(14)에 대한 프록시로서 역할하고 들어오는 트랜잭션에 대한 응답을 생성할 필요가 없음을 의미한다. 그러나 시스템 제어기(804) 및/또는 인터커넥트(802)가 더 낮은 클록 주파수에서 동작하는 동안 이의 성능 능력이 감소될 수 있기 때문에 IP 에이전트(14)에 대한 링크(803) 설정을 재구성할 수 있다. 변경될 가능성이 있는 설정이 IP 에이전트(14)에 대한 중재 설정 또는 허용된 미결 트랜잭션의 카운트의 감소를 포함한다. IP 에이전트가 이 저전력 모드를 빠져 나갈 때, 우선 (감소된 경우) 전압이 증가되고 그 후 (감소된 경우) 클록 주파수가 증가되며 링크(803) 설정의 임의의 변경이 (재구성되는 경우) 정규 동작 모드로 다시 알려진다.

[0201] 2. 저전력, 동작 불능 모드, 유지된 상태 정보: 이 모드에서, 클록이 셧 오프되며 전력 공급이 감소되지만, 완전히 꺼지지는 않을 수 있다. 따라서, IP 에이전트(14) 내 메모리에 유지되는 상태 정보가 유지된다. 이 모드에 들어가기 전에, 인터커넥트(802)는 새로운 트랜잭션이 개시되지 못하게 하거나 미결 트랜잭션이 완료되기를 기다림으로써 IP 에이전트가 이미 발행한 트랜잭션을 "드레인(drain)"한다. 모든 트랜잭션이 드레인되면, 앞서 기재된 바와 같이, 오작동 IP 에이전트(14)의 리셋과 관련하여 인터커넥트(802)가 프록시로서 동작하고 유사한 프로세스 (1), (2) 및 (3)를 수행할 수 있다. IP 에이전트가 정규 상태로 복귀하고 이 모드를 빠져나갈 때, 우선 전압이 증가하고 클록 주파수의 증가가 뒤 따른다.

[0202] 3. 저전력, 동작 불능 모드 - 상태 정보가 보유되지 않음: 이 모드는, 전력이 IP 에이전트에 포함되는 상태 정보가 소실되는 포인트로 감소되는 것을 제외하고, 직전에 설명된 모드 2와 유사하다. 인터커넥트(802)는 앞서 기재된 바와 같이 이 모드에서 프록시로서 동작한다. 전력이 다시 공급될 때, IP 에이전트는 도 9와 관련하여 앞서 기재된 바와 유사하게 리셋 협상 프로세스를 거칠 필요가 있다.

[0203] 4. 파워 오프 모드: 이 모드에서, 클록이 꺼지고 전력이 완전히 제거된다. 인터커넥트(802)는 앞서 기재된 바와 같이 프록시로서 동작한다. 파워 업 되면, 공급 전압이 우선 램프 업(ramp up)되고, 그 후 도 9와 관련하여 앞서 기재된 바와 같이 리셋 협상 프로세스가 뒤 따른다.

[0204] 도 11은 IP 에이전트(14)를 저전력, 동작 모드로 두고 여기서 빠져 나가는 시퀀스를 도시하는 흐름도(1100)이다.

[0205] 초기 단계(1102)에서, SoC(800) 상의 IP 에이전트(14)가 자신의 정규 모드에서 동작한다, 즉, 표준 클록 주파수 및 전압이 사용된다.

[0206] 결정 단계(1104)에서, SoC(800) 내 조건이 시스템 제어기(804)에 의해 모니터링된다. 동작 조건이 비교적 정상이거나 IP 에이전트(14)의 파워 다운을 트리거하는 어떠한 이벤트도 발생하지 않는 경우, SoC 및 IP 에이전트(14)가 단계(1102)마다 자신의 정규 모드에서 계속 동작한다. 그러나 트리거 조건이 충족되면(가령, 감소된 배터리 공급량, 과열 등), 전력 관리자(808)가 IP 에이전트(14)를 저전력 동작 모드로 두도록 선택할 수 있다.

[0207] 선택적 단계(1106)에서, 인터커넥트(802)는 링크(803)를 재구성하도록 선택할 수 있다. 재구성은 IP 에이전트(14)에 대한 중재 설정을 변경하거나 가능한 미결 트랜잭션의 카운트를 감소시켜, 저전력 모드에서 동작할 때 IP 에이전트의 더 낮은 처리 능력을 고려하는 것을 포함할 수 있다.

[0208] 단계(1108)에서, 적용될 때 IP 에이전트(14)의 동작 클록 주파수가 감소된다. 클록 주파수가 감소되면, IP 에이전트는 전력을 덜 소비한다.

[0209] 단계(1110)에서, 적용될 때 IP 에이전트에 공급되는 전압이 감소된다. 전압을 감소시킴으로써, 추가 전력 절약이 실현될 수 있다.

[0210] 클록 주파수 및/또는 전압이 감소되면, IP 에이전트(14)가 동작 가능한 상태로 유지된다. 따라서, 표준 클록 주파수 및/또는 공급 전압에서 동작할 때 속도는 느릴 수 있더라도 트랜잭션을 처리할 수 있다. 선택적 실시예에서, 인터커넥트(802)가 앞서 기재된 바와 같이 프록시로서 동작하거나 저전력 모드에서 IP 에이전트(14)의 느린 수행 속도를 고려 및 지원하도록 조절되거나 재구성될 수 있다. 이들 대안이 선택사항이기 때문에, 반드시 구현되어야 하는 것은 아니다.

[0211] 결정 단계(1112)에서, 결정이 정규 동작을 재개하도록 이뤄질 때까지 IP 에이전트(14)가 저전력 모드에서 동작한다. 어느 경우라도, IP 에이전트(14)가 정규 동작을 재개하기 위한 시퀀스를 겪는다.

[0212] 선택 단계(1114)에서, 경우에 따라(즉, 전압이 이전에 감소되었다면) 전압이 표준 동작 전압까지 증가한다.

[0213] 단계(1116)에서, 경우에 따라(즉, 클록이 이전에 감소됐다고 가정되면) 클록 주파수가 증가한다. 단계(1117)에서, IP 에이전트가 정규 동작으로 복귀한다.

[0214] 마지막으로, 동작 단계(1118)에서, 인터커넥트가 임의의 재구성된 인터커넥트 셋팅을 정규로 복귀시킨다. 이 포인트에서, 단계(1102)에서 제공되는 바와 같이, IP 에이전트는 정규 동작을 재개할 준비가 된다.

[0215] 도 12를 참조하면, 저전력, 동작 불능, 상태 정보 유지 모드에서 IP 에이전트(14)를 파워 다운/업하기 위한 시퀀스를 도시하는 흐름도(1200)가 도시된다.

[0216] 단계(1202)에서, IP 에이전트(14)가 자신의 정규 모드로 동작한다.

[0217] 단계(1204)에서, IP 에이전트(14)를 저전력, 동작 불능, 상태 정보 유지 모드에서 IP 에이전트(14)를 동작시키기 위한 결정이 이뤄진다.

[0218] 단계(1206)에서, 링크(803)가 작업거부 상태에 놓이고 인터커넥트(802)가 IP 에이전트(14)에 대한 프록시로서 동작하도록 구성된다. 일반적으로 이는 (1) 임의의 새로운 트랜잭션이 IP 에이전트(14)에 의해 생성되지 못하게 하는 것, (2) 임의의 미결 트랜잭션이 완료되기를 기다리는 것 그리고 그런 다음 (3) IP 에이전트(14)를 타깃으로 하는 임의의 트랜잭션에 응답함으로써 프로시로서 동작하는 것을 포함한다. 예를 들어, 인터커넥트(802)는 예외 메시지를 처리되지 않은 트랜잭션의 출발지로 전송하여, 트랜잭션의 전송자가 IP 에이전트(14)로부터의 응답을 수신하지 않았기 때문에 행업 상황이 발생하는 것을 방지할 수 있다.

[0219] 단계(1208)에서, 적용 가능할 때 IP 에이전트(14)의 클록 주파수가 감소된다.

[0220] 단계(1210)에서, 적용 가능할 때 IP 에이전트(14)의 동작 전압이 감소된다. 그러나 IP 에이전트(14) 내 메모리 또는 저장 요소가 자신의 상태 정보를 유지하도록 전압이 적절하게 유지된다.

[0221] 결정 단계(1212)에서, IP 에이전트(14)는 정규 동작을 재개하라는 결정이 이뤄질 때까지 저전력 상태로 유지된다. 시스템 제어기(804), SoC 외부 이벤트(가령, 센서로부터 수신된 신호, 외부 소스로부터 수신된 신호 등), 타이머, IP 에이전트 자체 또는 또 다른 IP 에이전트가 웨이크-업을 모두 트리거할 수 있다. 결정이 이뤄질 때, IP 에이전트는 보통 동작을 재개하기 위한 시퀀스를 경험한다.

[0222] 단계(1214 및 1216)에서, 적용 가능한 경우 IP 에이전트(14)로 제공되는 전압 및 클록 주파수 각각이 증가한다. 상태 정보가 유지되어 있기 때문에, IP 에이전트(14)는 단계(127)에서 정규 동작을 재개한다.

[0223] 단계(1218)에서, 링크(803)가 작업거부 모드를 빠져 나오고 IP 에이전트가 트랜잭션 준비 상태가 되며 인터커넥트(802)는 더 길게 프록시로서 동작해야 한다고 통지 받는다.

[0224] 도 13을 참조하면, 저전력, 동작 불능 모드에 대한 시퀀스를 도시하는 흐름도(1300)가 도시된다. 이 시퀀스에서, 단계(1202, 1206 및 1212)가 도 12를 참조하여 앞서 기재된 것과 동일하다. 따라서, 이들 단계에 대한 언급이 본 명세서에서 반복되지 않는다.

[0225] 단계(1302)에서, IP 에이전트(14)를 파워 다운하기 위한 결정이 이뤄진다. 그 후, 인터커넥트가 프록시로서 구성되고(단계(1206)) IP 에이전트(14)에 대한 클록이 완벽하게 꺼지거나(적용 가능한 경우) 및/또는 전압이 단계(1304)에서 상태 정보가 소실되는 포인트까지 상당히 감소된다(적용 가능한 경우). 상태 정보 없이, 단계(1212)마다 정규 동작을 재개하기 위한 결정이 이뤄질 때, 단계(1306)에서 전압이 램프 업되고(적용 가능한 경우) 클록이 켜진다(적용 가능한 경우). 그런 다음, 도 9와 관련하여 앞서 기재된 바와 같이 IP 에이전트(14)는 리셋 동작을 겪는다. 리셋이 완료되면, IP 에이전트(14)가 트랜잭션 준비 상태가 된다. 그런 다음 시스템은 단계(1310)에서 작업거부 모드를 종료하기 위한 링크를 기다린다. 종료가 발생하면, IP 에이전트가 인터커넥트(802) 상에서 가시적이다. 그런 다음, 단계(1312)에서, 인터커넥트(802)는 더 이상 IP 에이전트(14)에 대한 프록시로서 동작한다.

[0226] 마지막으로, 파워 오프 모드의 경우, 단순히 감소되는 것과 달리 파워가 완벽히 꺼지는 점을 제외하고, 시퀀스는 도 13과 동일하다. 이와 달리, 파워 오프 모드 시퀀스가 동일하다. 이 모드에서, IP 에이전트(14)는 사실상 전력을 소비하지 않고, 동작 불능이며, 인터커넥트(802)는 IP 에이전트를 대신하여 프로시로서 동작할 수 있다.

[0227] 도 14를 참조하면, 링크(803)를 작업거부 상태로 두기 위한 단계를 나타내는 흐름도(1400)가 도시된다.

[0228] 초기 단계(1402)에서, 시스템 제어기(804)는 IP 에이전트(14)가 동작 불능 전력 절약 모드 중 하나가 되어야 한다는 결정을 내린다.

[0229] 단계(1404)에서, IP 에이전트(14)는 트랜잭션을 생성하기를 중단하도록 명령 받는다.

[0230] 결정 단계(1406)에서, 시스템은 모든 미결 트랜잭션이 완료되는지 여부를 결정한다. 모든 미결 비-포스트 트랜잭션의 경우, 완료 트랜잭션이 수신되어야 한다(즉, 읽기 트랜잭션의 경우, 액세스된 데이터가 반환되어야 하며, 비-포스트 쓰기 트랜잭션의 경우, 수신확인이 수신되어야 한다). 포스트 트랜잭션의 경우, 어떠한 응답 트랜잭션도 필요하지 않다. 따라서 IP 에이전트에 의해 전송되면 포스트 트랜잭션은 "완료"로 간주된다.

[0231] 단계(1408)에서, 모든 미결 트랜잭션이 완료될 때 링크(803)가 작업거부 상태가 된다. 그런 다음, 인터커넥트(802)는 IP 에이전트(14)에 대한 프록시로서 구성된다.

[0232] 단계(1410)에서, IP 에이전트는 리셋 또는 바람직한 동작 불능 저전력 모드가 된다.

[0233] 도 15a-15d는 IP 에이전트 "웨이크-업" 시퀀스에 대한 흐름도를 도시한다.

[0234] 도 15a를 참조하면, 에이전트-개시 "웨이크-업" 시퀀스를 도시하는 흐름도(1500)가 나타난다. 이 실시예에서, IP 에이전트에 의해 웨이크 업 시퀀스가 개시되지만, 시스템 제어기(804)를 통해 구현된다.

[0235] 단계(1502)에서, 동작 불능 상태의 IP 에이전트(14)는 웨이크-업 트리거 이벤트를 검출한다. IP 에이전트가 파워 다운 또는 "오프"될 수 있지만, 웨이크-업 트리거가 발생할 때를 검출하기 위한 능력을 유지하는 관점에서 적어도 부분적으로 작동가능함을 유지할 수 있다. 웨이크-업 트리거가 상이한 유형의 이벤트를 포함할 수 있다. 예를 들어, 이는 IP 에이전트(14)가 지정 시간 주기 후에 웨이크-업하게 하는 내부 타이머이거나, SoC(800) 외부의 이벤트, 가령, IP 에이전트(14)와 통신하기를 원하는 또 다른 디바이스일 수 있다. 단계(1504)에서, IP 에이전트가 인터커넥트(802)까지의 링크(803)를 통해 "웨이크-업" 통신을 전송한다. 다시 말해, 링크가 대응하는 IP 에이전트(14)가 동작 불능 상태에 있는 작업거부 상태에 있더라도, 웨이크-업 신호를 인터커넥트(802)로 전송할 수 있다.

[0236] 단계(1506)에서, 인터커넥트(802)는 동작 불능 IP 에이전트로부터 웨이크-업 신호에 대해 "청취"하도록 구성된다. 신호가 검출되는 경우, 인터커넥트(802)는 시스템 제어기(804)에게 통지한다.

[0237] 단계(1508)에서, 시스템 제어기(804)는 IP 에이전트(14)에 대한 인터커넥트(802)를 통해 커맨드를 전송하여 자신의 웨이크-업 시퀀스를 개시할 수 있다.

[0238] 단계(1510)에서, IP 에이전트가 커맨드에 응답하여 자신의 웨이크-업 시퀀스를 개시한다.

[0239] 앞서 기재된 실시예의 경우, IP 에이전트(14)가 시스템 제어기에게 웨이크-업 시퀀스를 개시하도록 요청한다. 시스템 제어기로부터의 웨이크-업 커맨드에 응답하여, IP 에이전트가 자신 고유의 시퀀스를 개시한다. 따라서 시스템 제어기는 동작 불능 상태로부터 벗어나고 인터커넥트(802) 상에서 가시적이 될 때 IP 에이전트의 상태를 인지한다.

[0240] 도 15b는 시스템 제어기(804)가 IP 에이전트(14)의 웨이크0업을 개시할 때 시퀀스를 도시한다. 이 시퀀스의 경우, 단계(1508)에서 시스템 제어기(804)가 IP 에이전트로 웨이크 업 커맨드를 전송하고, 이에 응답하여 단계(1510)에서 IP 에이전트가 자신의 웨이크 업 시퀀스를 개시한다. 이 실시예의 변형(도시되지 않음)에서, 웨이크 업이 시스템 제어기(804)를 통해 SoC(800)로부터 개시될 수 있다. 시스템 제어기(804)가 커맨드를 수신할 때, 상기 기재된 프로세스가 개시된다.

[0241] 도 15c는 IP 에이전트(14)에 대한 웨이크 업 커맨드가 SoC(800)로부터 기원하고 시스템 제어기(804)를 통해 구현될 때의 시퀀스를 도시한다. 이 시퀀스를 이용할 때, 단계(1512)에서 시스템 제어기(804)는 커맨드를 수신한다. 이에 응답하여, 단계(1508)에서 시스템 제어기는 웨이크 업 커맨드를 IP 에이전트로 전송하고 이에 응답하여 단계(1510)에서 IP 에이전트는 자신 고유의 웨이크 업 시퀀스를 개시한다. SoC(800)으로부터의 직접 웨이크 업을 이용하는 경우, 커맨드는 이의 하드-와이어 입력을 통해 IP 에이전트(14)로 직접 제공된다. 이에 응답하여, IP 에이전트는 자신 고유의 웨이크 업 시퀀스를 개시한다.

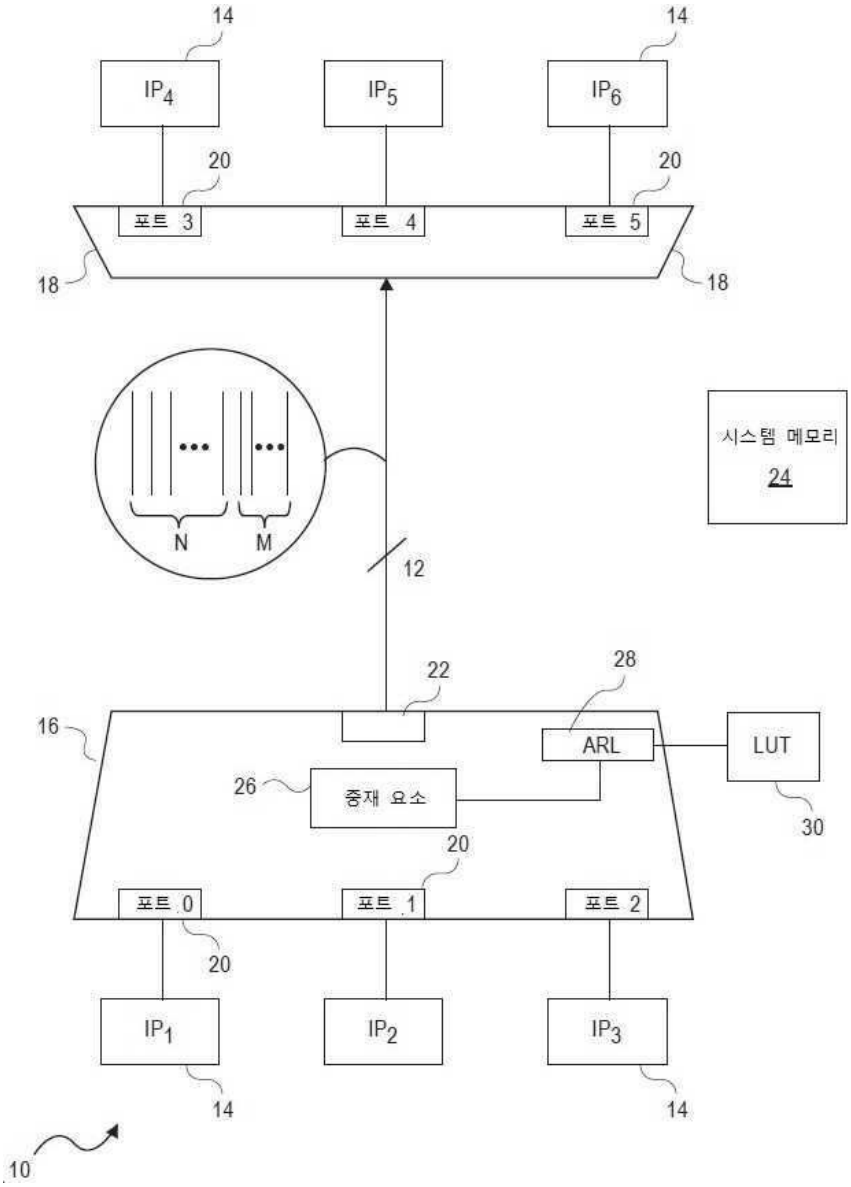
[0242] 도 15d를 참조하면, IP 에이전트에 의해 개시 및 구현되는 웨이크 업 시퀀스를 도시하는 흐름도(1520)가 도시된다. 이 실시예에서, 웨이크 업 조건, 가령, 앞서 언급된 것 중 임의의 것이 단계(1522)에서 발생한다. 이에 응답하여, 단계(1524)에서 IP 에이전트는 자신 고유의 웨이크 업 시퀀스를 개시한다. 단계(1526)에서, 웨이크 업 시퀀스가 완료된다. 그 후, 단계(1528)에서, IP 에이전트가, 직접 또는 인터커넥트(802)를 통해, 인터커넥트(802) 및 시스템 제어기(804)에게, 이의 웨이크 업된 상태를 통지한다.

[0243] 상기 예시에서, 단일 IP 에이전트를 상기 기재된 저전력 모드로 전환하기 위한 시퀀스가 단순성을 위해 기재되었다. 실제 실시예에서, SoC 상의 복수의 IP 에이전트(14)가 동시에 파워 다운될 수 있다. 둘 이상이 동시에 또는 거의 동시에 파워 다운되는 경우, 모드와 무관하게, 각각은 상기 기재된 시퀀스 중 하나를 독립적으로 경험한다.

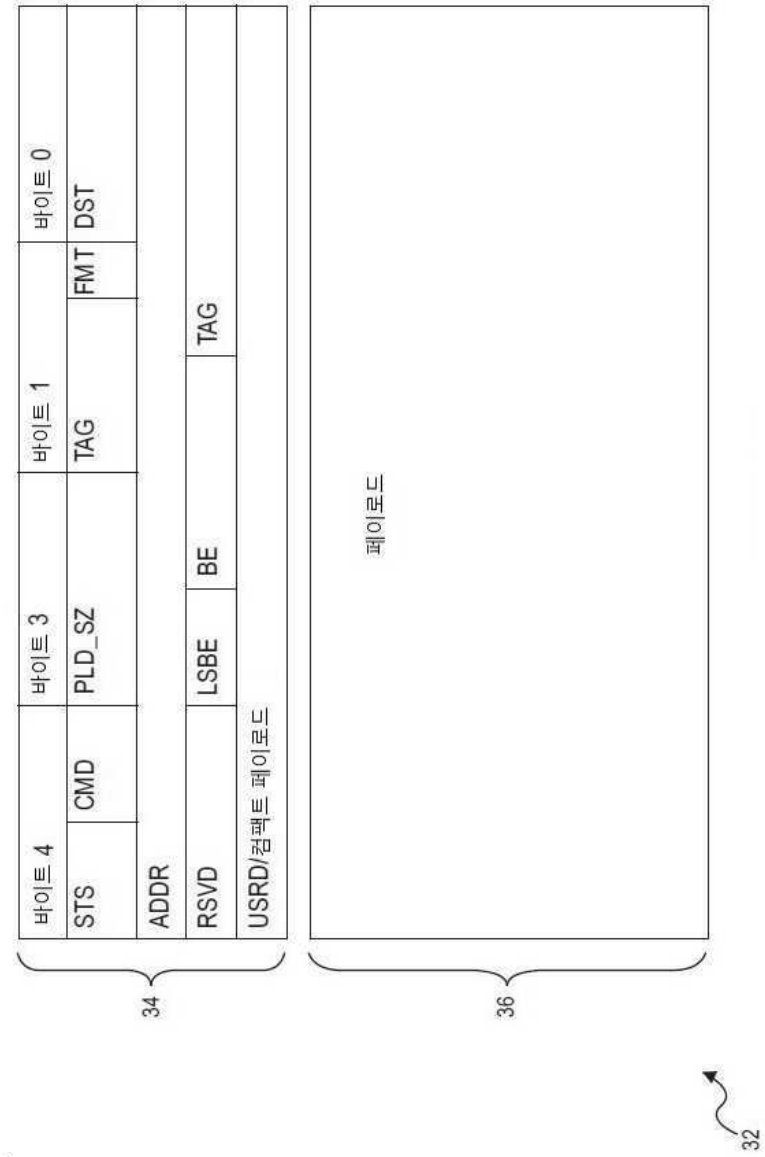
[0244] 비록 몇몇 실시예만이 상세하게 설명되었지만, 본 출원은 여기에 제공된 개시 내용의 사상 또는 범위를 벗어나지 않고 많은 다른 형태로 구현 될 수 있음을 이해해야 한다. 따라서, 본 실시예는 제한적이지 않고 예시적인 것으로 간주되어야 하며 여기에 제공된 세부 사항에 제한되지 않고 첨부 된 청구 범위의 범위 및 등가물 내에서 수정 될 수 있다.

도면

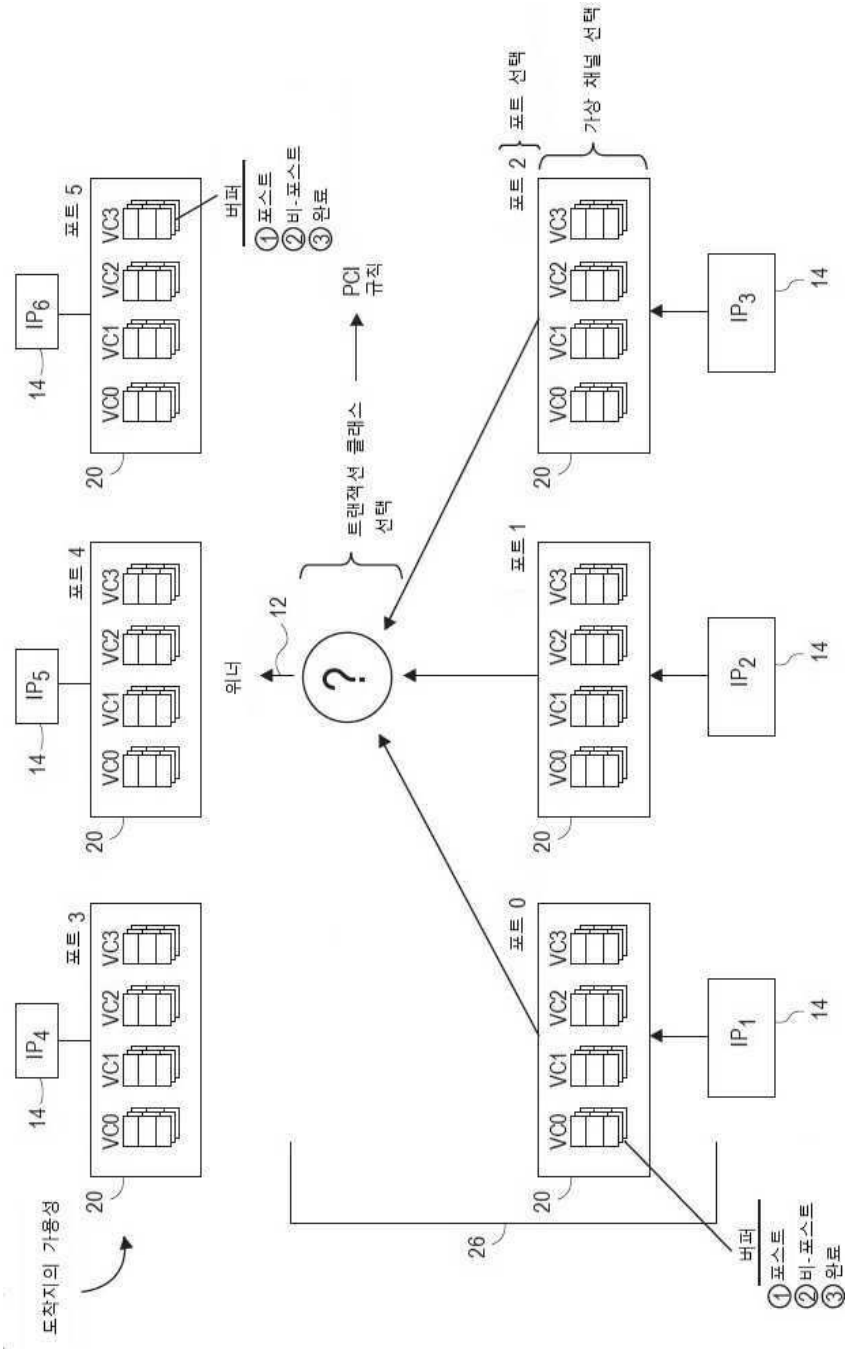
도면1


14
IP4
IP5
14
IP6
20
20
포트 3
포트 4
포트 5
18
18
N
M
12
시스템 메모리
24
22
28
16
ARL
LUT
26
중재 요소
30
20
포트 0
포트 1
포트 2
20
IP1
IP2
IP3
14
14
10

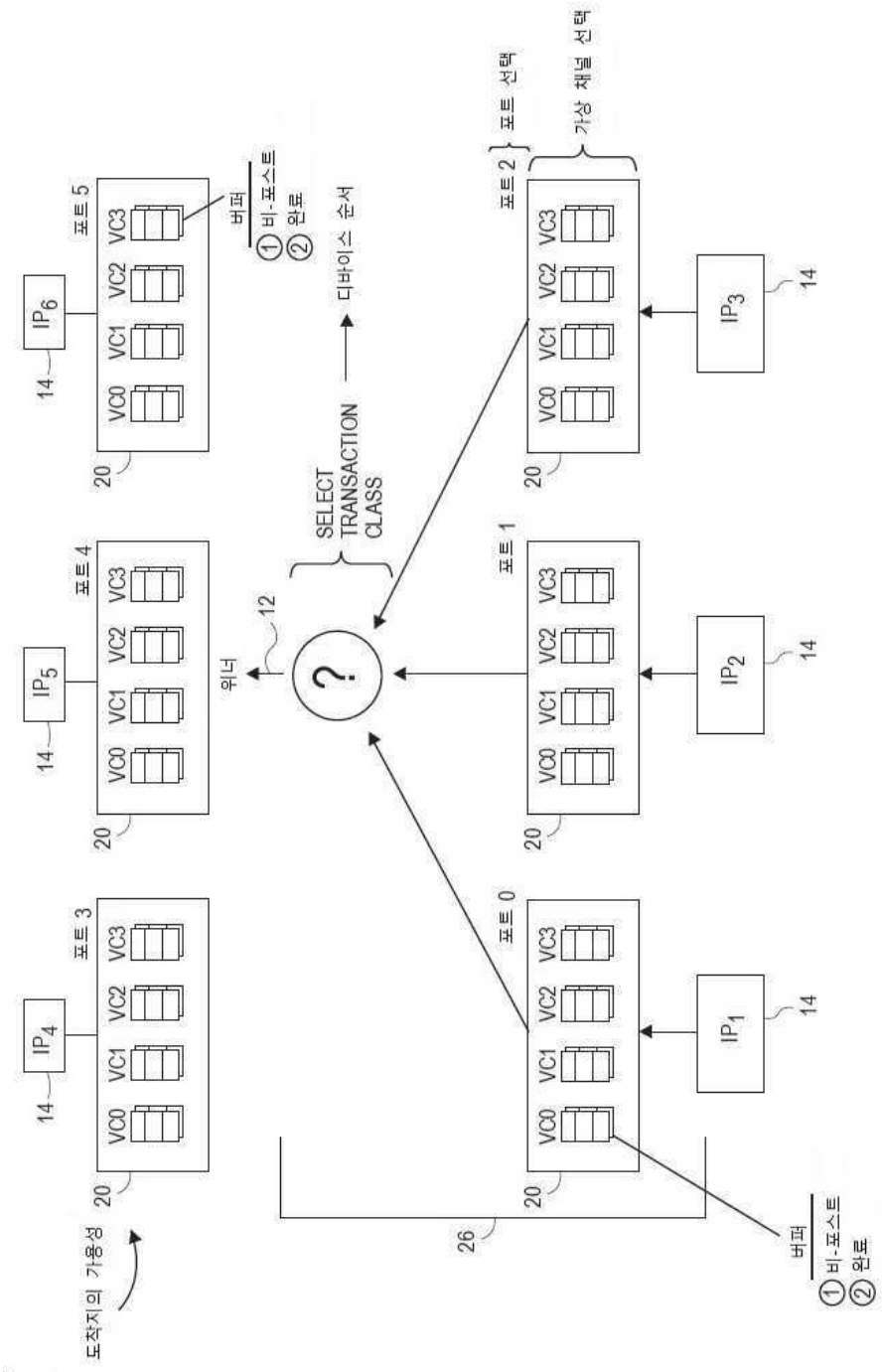
도면2


바이트 4
바이트 3
바이트 1
바이트 0
STS
CMD
PLD_SZ
TAG
FMT
DST
ADDR
RSVD
LSBE
BE
TAG
USRD/컴팩트 페이로드
페이로드
34
36
32

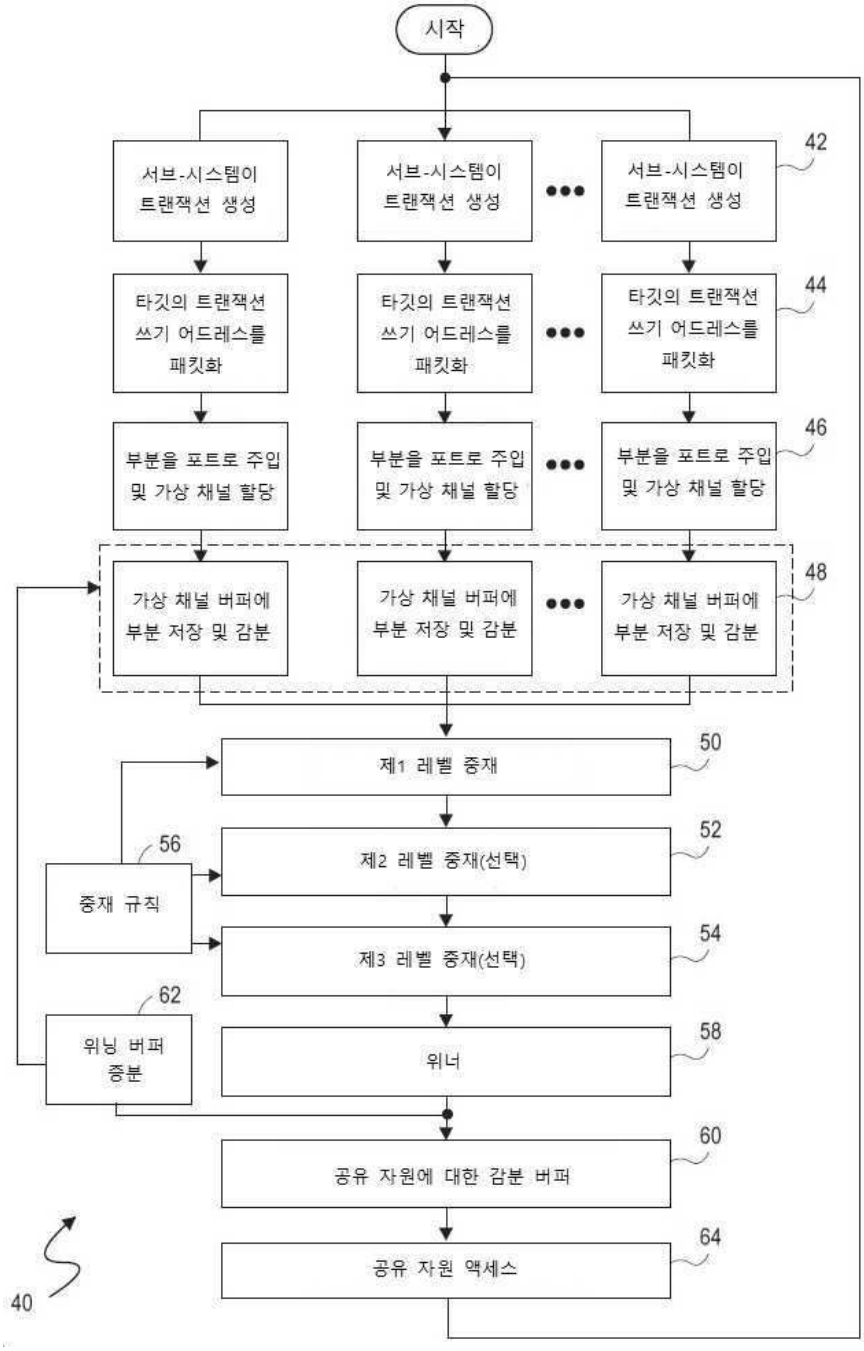
도면3a


포트 5
포트 4
포트 3
포트 2
포트 1
포트 0
VC3
VC2
VC1
VC0
IP6
IP5
IP4
IP3
IP2
IP1
14
20
버퍼
①포스트
②비-포스트
③완료
PCI
규칙
트랜잭션 클래스
선택
포트 선택
가상 채널 선택
12
위너
?
26
도착지의 가용성

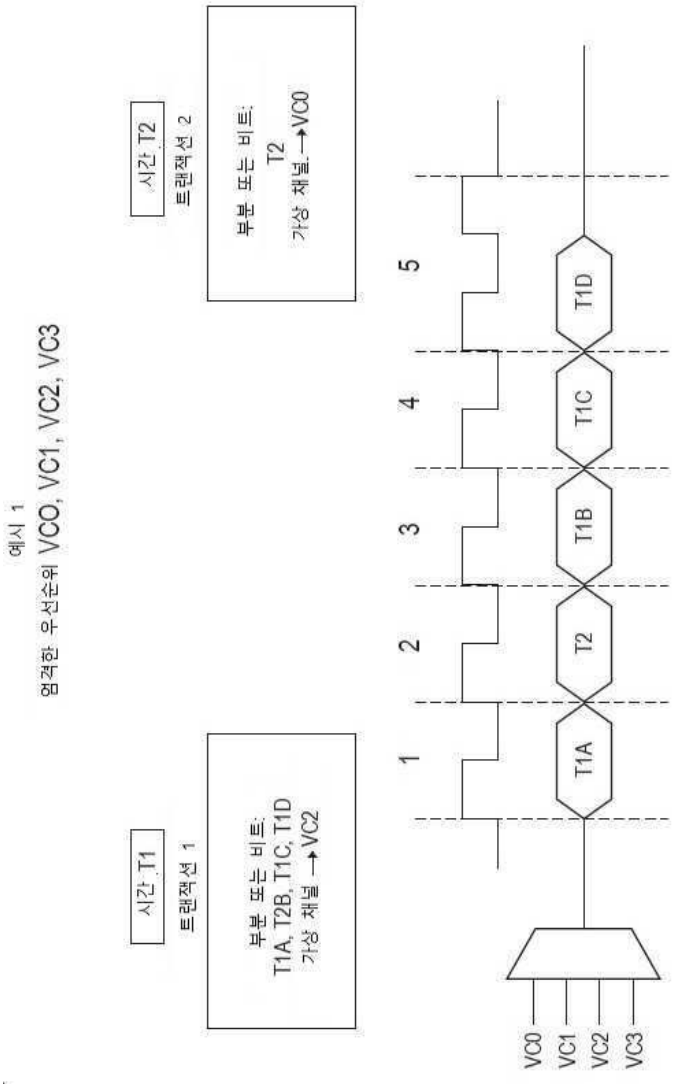
도면3b


포트 5
포트 4
포트 3
포트 2
포트 1
포트 0
VC3
VC2
VC1
VC0
IP6
IP5
IP4
IP3
IP2
IP1
14
20
12
26
위너
?
SELECT TRANSACTION CLASS
디바이스 순서
포트 선택
가상 채널 선택
버퍼
① 비-포스트
② 완료
도착지의 가용성

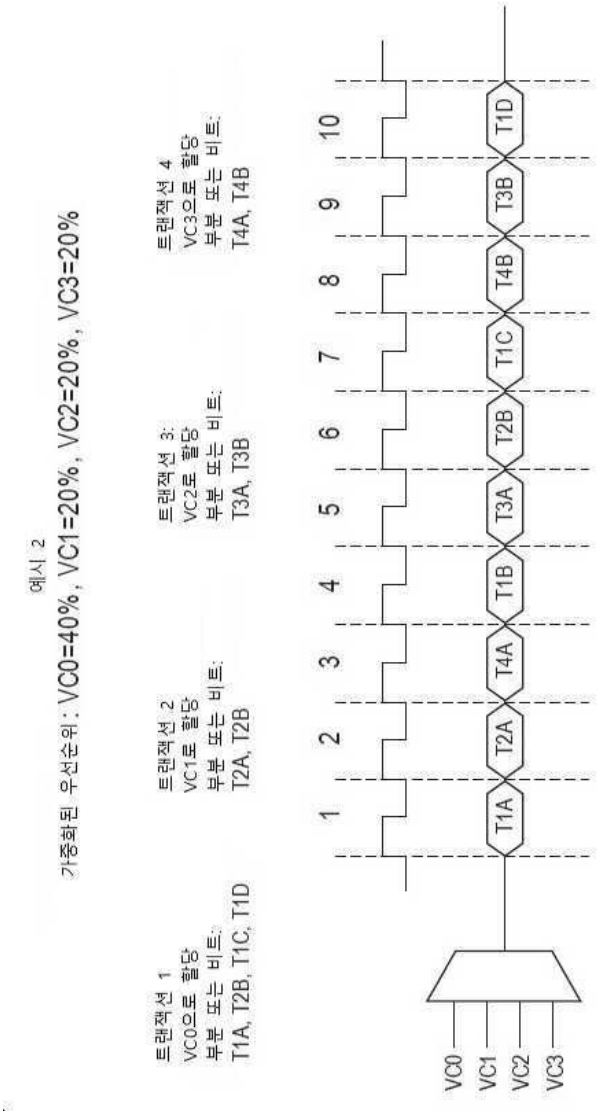
도면4


시작
서브-시스템이 트랜잭션 생성
서브-시스템이 트랜잭션 생성
서브-시스템이 트랜잭션 생성
42
타깃의 트랜잭션 쓰기 어드레스를 패킷화
타깃의 트랜잭션 쓰기 어드레스를 패킷화
타깃의 트랜잭션 쓰기 어드레스를 패킷화
44
부분을 포트로 주입 및 가상 채널 할당
부분을 포트로 주입 및 가상 채널 할당
부분을 포트로 주입 및 가상 채널 할당
46
가상 채널 버퍼에 부분 저장 및 감분
가상 채널 버퍼에 부분 저장 및 감분
가상 채널 버퍼에 부분 저장 및 감분
48
제1 레벨 중재
50
제2 레벨 중재(선택)
52
56
중재 규칙
제3 레벨 중재(선택)
54
62
위닝 버퍼 증분
위너
58
공유 자원에 대한 감분 버퍼
60
공유 자원 액세스
64
40

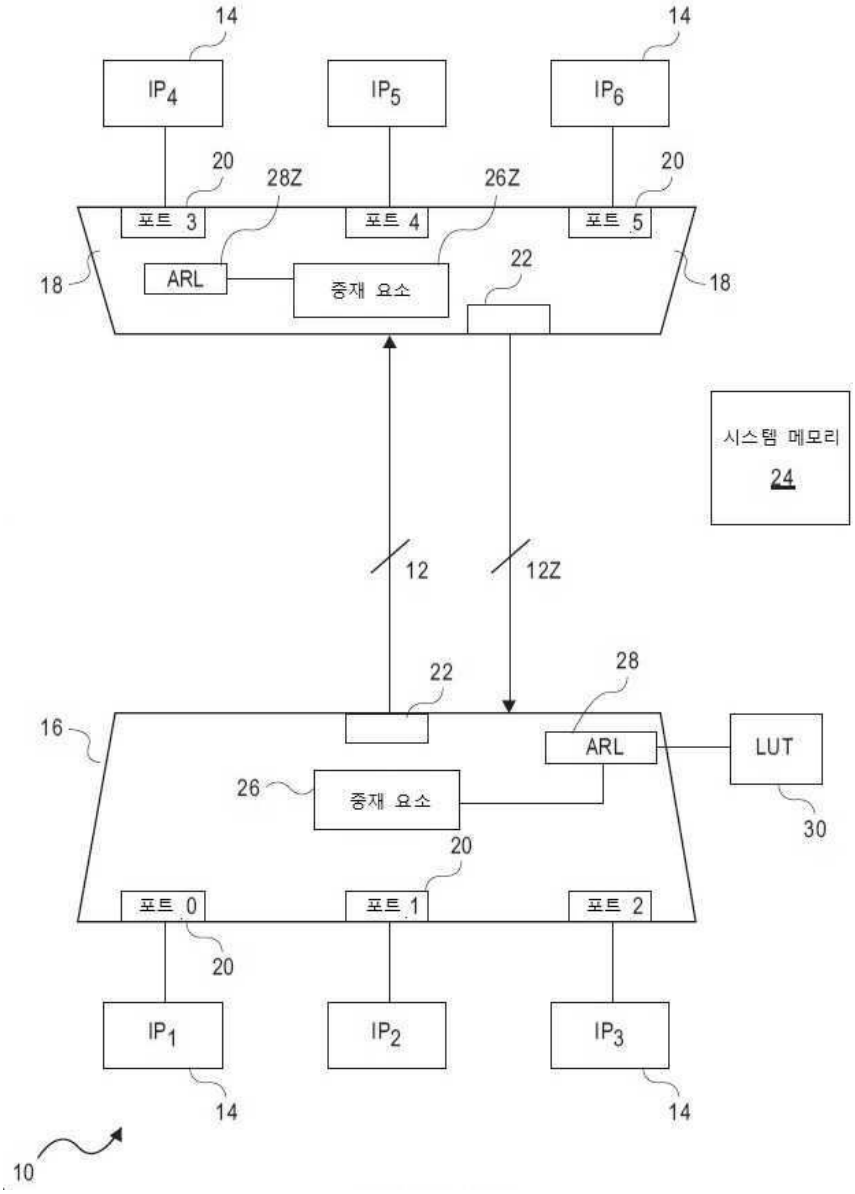
도면5


예시 1
엄격한 우선순위 VC0, VC1, VC2, VC3
시간 T1
트랜잭션 1
부분 또는 비트:
T1A, T2B, T1C, T1D
가상 채널 → VC2
시간 T2
트랜잭션 2
부분 또는 비트:
T2
가상 채널 → VC0
1
2
3
4
5
T1A
T2
T1B
T1C
T1D
VC0
VC1
VC2
VC3

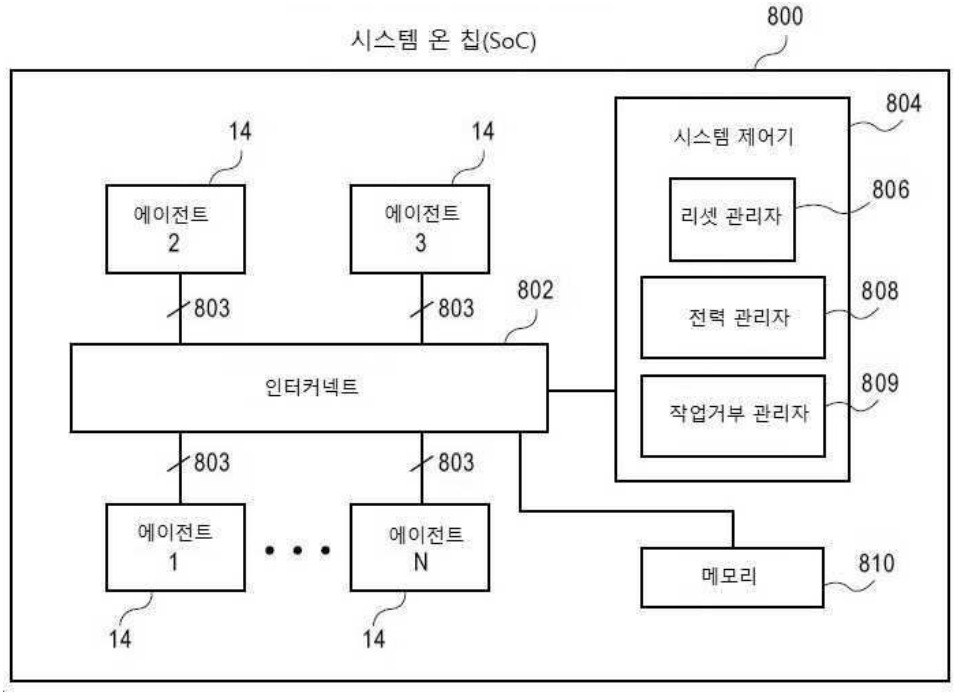
도면6


예시 2
가중화된 우선순위 : VC0=40%, VC1=20%, VC2=20%, VC3=20%
트랜잭션 1
VC0으로 할당
부분 또는 비트:
T1A, T2B, T1C, T1D
트랜잭션 2
VC1로 할당
부분 또는 비트:
T2A, T2B
트랜잭션 3:
VC2로 할당
부분 또는 비트:
T3A, T3B
트랜잭션 4
VC3으로 할당
부분 또는 비트:
T4A, T4B
1
2
3
4
5
6
7
8
9
10
T1A
T2A
T4A
T1B
T3A
T2B
T1C
T4B
T3B
T1D
VC0
VC1
VC2
VC3

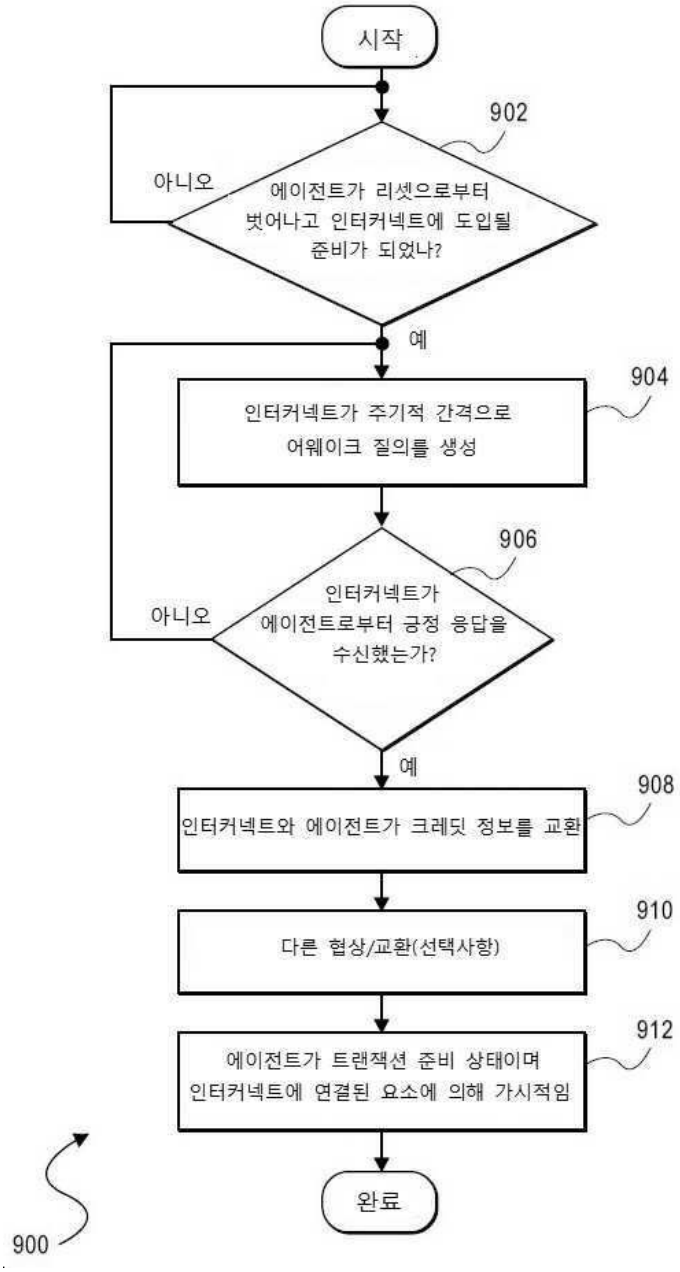
도면7


14
IP4
IP5
14
IP6
20
28Z
26Z
20
포트 3
포트 4
포트 5
18
ARL
중재 요소
22
18
시스템 메모리
24
12
12Z
22
28
16
ARL
LUT
26
중재 요소
30
20
포트 0
포트 1
포트 2
20
IP1
IP2
IP3
14
14
10

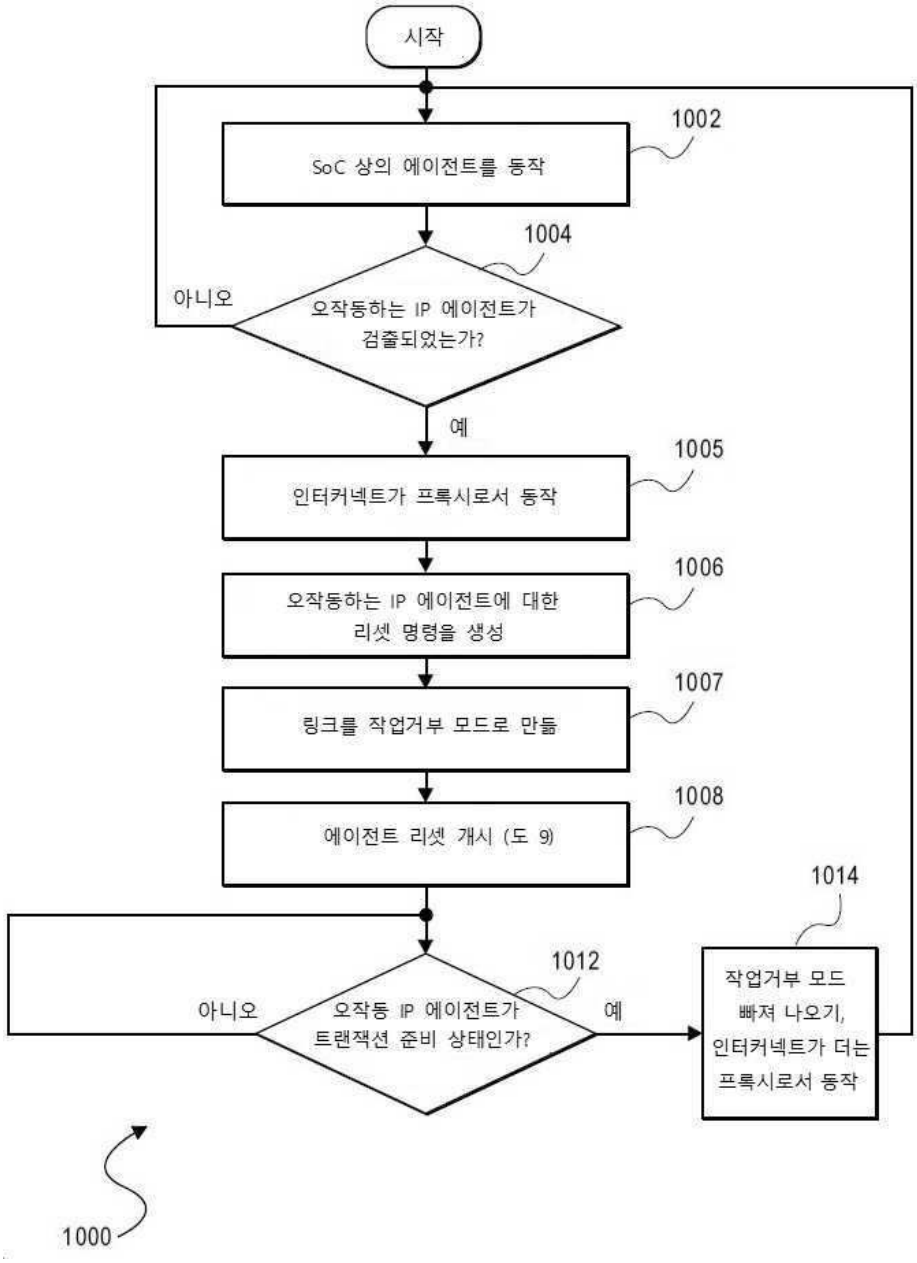
도면8


시스템 온 칩(SoC)
800
804
시스템 제어기
806
리셋 관리자
808
전력 관리자
809
작업거부 관리자
810
메모리
802
인터커넥트
14
에이전트 2
에이전트 3
에이전트 1
에이전트 N
803

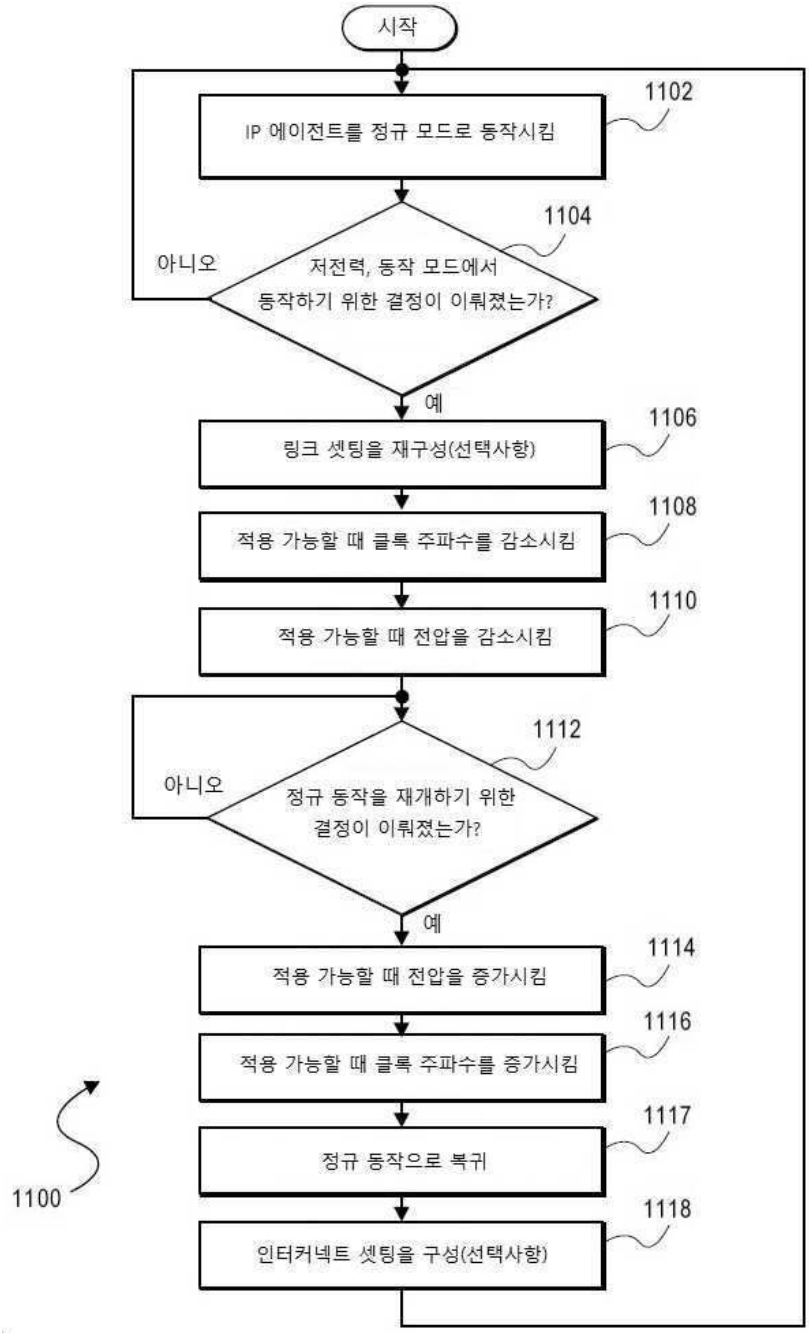
도면9


시작
902
아니오
에이전트가 리셋으로부터 벗어나고 인터커넥트에 도입될 준비가 되었나?
예
904
인터커넥트가 주기적 간격으로 어웨이크 질의를 생성
906
아니오
인터커넥트가 에이전트로부터 긍정 응답을 수신했는가?
예
908
인터커넥트와 에이전트가 크레딧 정보를 교환
910
다른 협상/교환(선택사항)
912
에이전트가 트랜잭션 준비 상태이며 인터커넥트에 연결된 요소에 의해 가시적임
완료
900

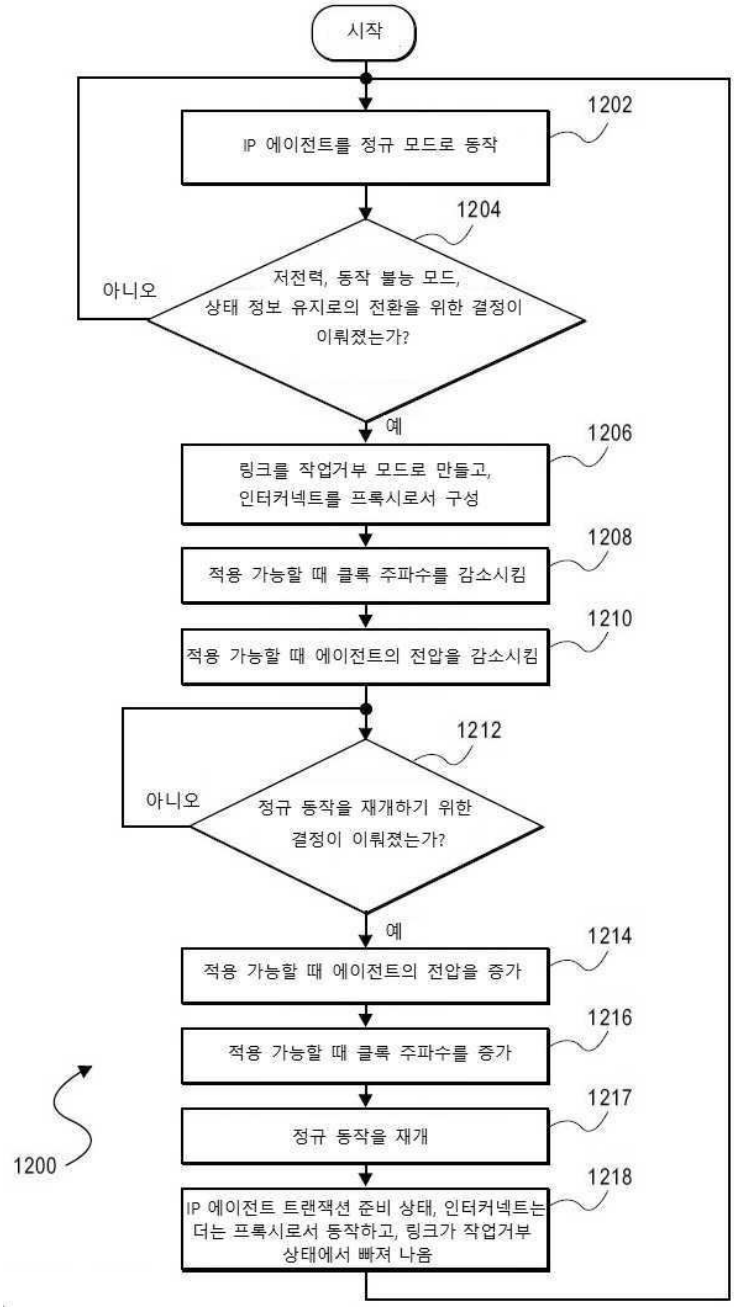
도면10


시작
SoC 상의 에이전트를 동작
1002
오작동하는 IP 에이전트가 검출되었는가?
1004
아니오
예
인터커넥트가 프록시로서 동작
1005
오작동하는 IP 에이전트에 대한 리셋 명령을 생성
1006
링크를 작업거부 모드로 만듦
1007
에이전트 리셋 개시 (도 9)
1008
오작동 IP 에이전트가 트랜잭션 준비 상태인가?
1012
아니오
예
작업거부 모드 빠져 나오기, 인터커넥트가 더는 프록시로서 동작
1014
1000

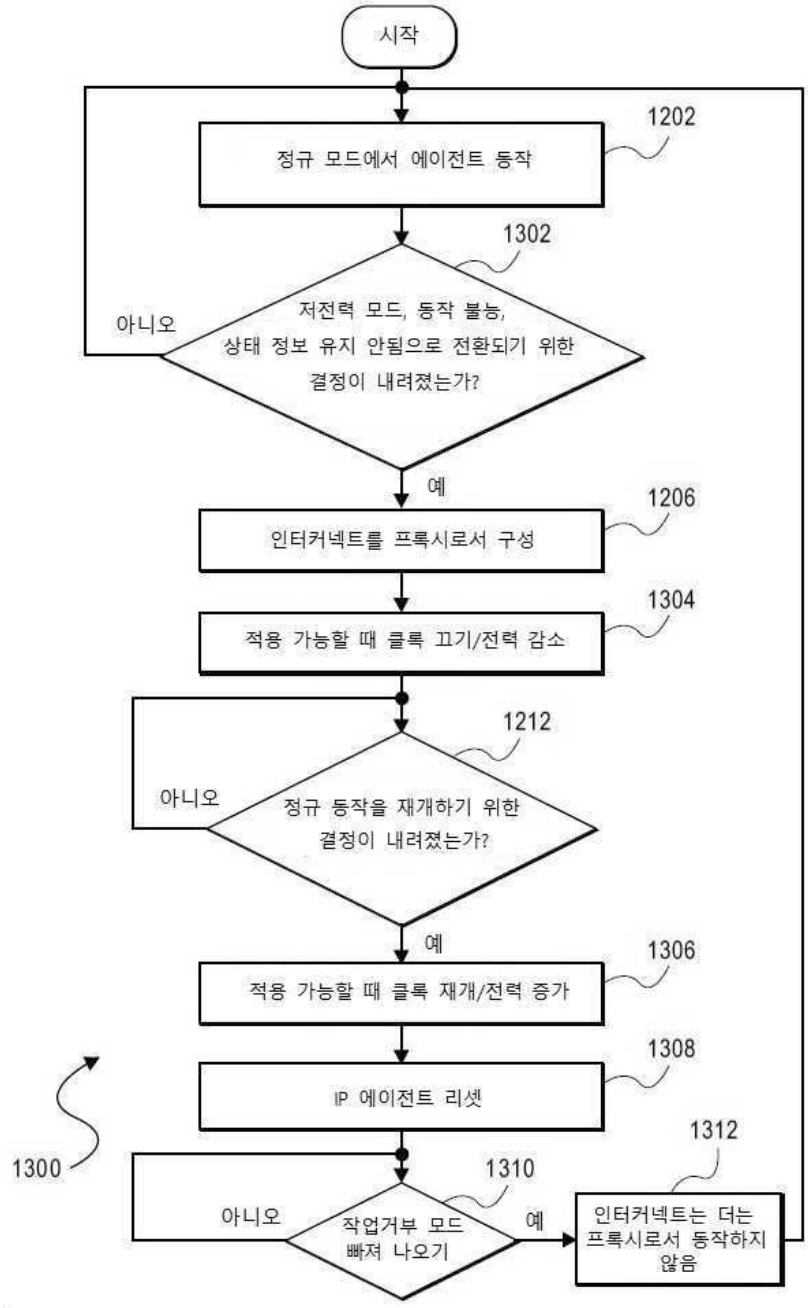
도면11


시작
IP 에이전트를 정규 모드로 동작시킴
1102
저전력, 동작 모드에서 동작하기 위한 결정이 이뤄졌는가?
1104
아니오
예
링크 셋팅을 재구성(선택사항)
1106
적용 가능할 때 클록 주파수를 감소시킴
1108
적용 가능할 때 전압을 감소시킴
1110
정규 동작을 재개하기 위한 결정이 이뤄졌는가?
1112
아니오
예
적용 가능할 때 전압을 증가시킴
1114
적용 가능할 때 클록 주파수를 증가시킴
1116
정규 동작으로 복귀
1117
인터커넥트 셋팅을 구성(선택사항)
1118
1100

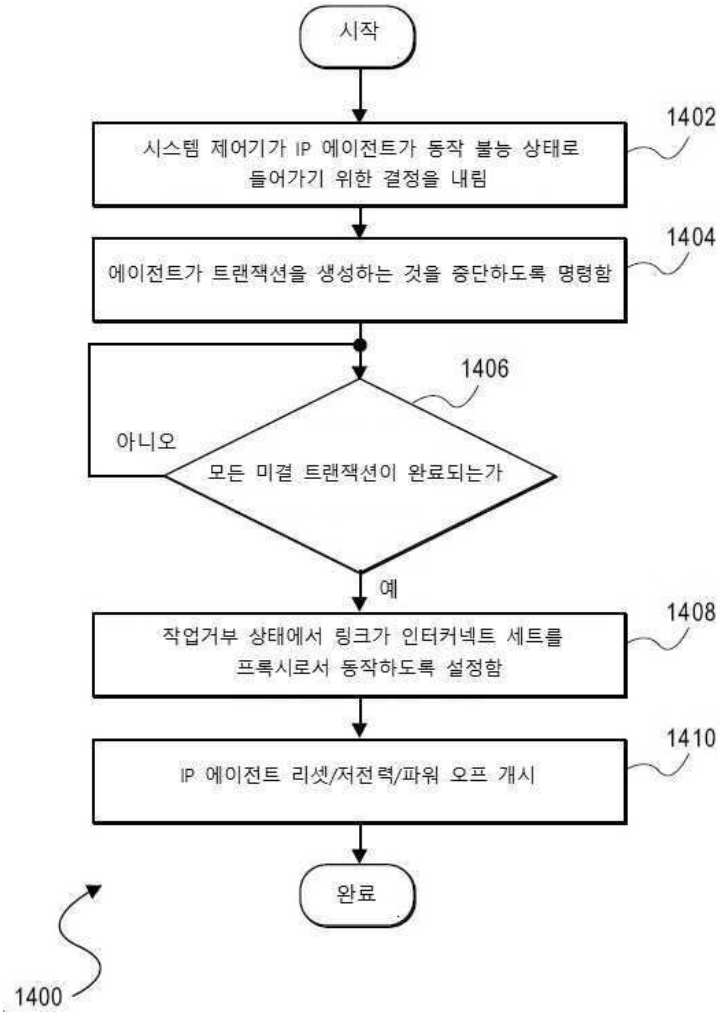
도면12


시작
1202
IP 에이전트를 정규 모드로 동작
1204
저전력, 동작 불능 모드,
상태 정보 유지로의 전환을 위한 결정이
이뤄졌는가?
아니오
예
1206
링크를 작업거부 모드로 만들고,
인터커넥트를 프록시로서 구성
1208
적용 가능할 때 클록 주파수를 감소시킴
1210
적용 가능할 때 에이전트의 전압을 감소시킴
1212
정규 동작을 재개하기 위한
결정이 이뤄졌는가?
아니오
예
1214
적용 가능할 때 에이전트의 전압을 증가
1216
적용 가능할 때 클록 주파수를 증가
1217
정규 동작을 재개
1218
IP 에이전트 트랜잭션 준비 상태, 인터커넥트는
더는 프록시로서 동작하고, 링크가 작업거부
상태에서 빠져 나옴
1200

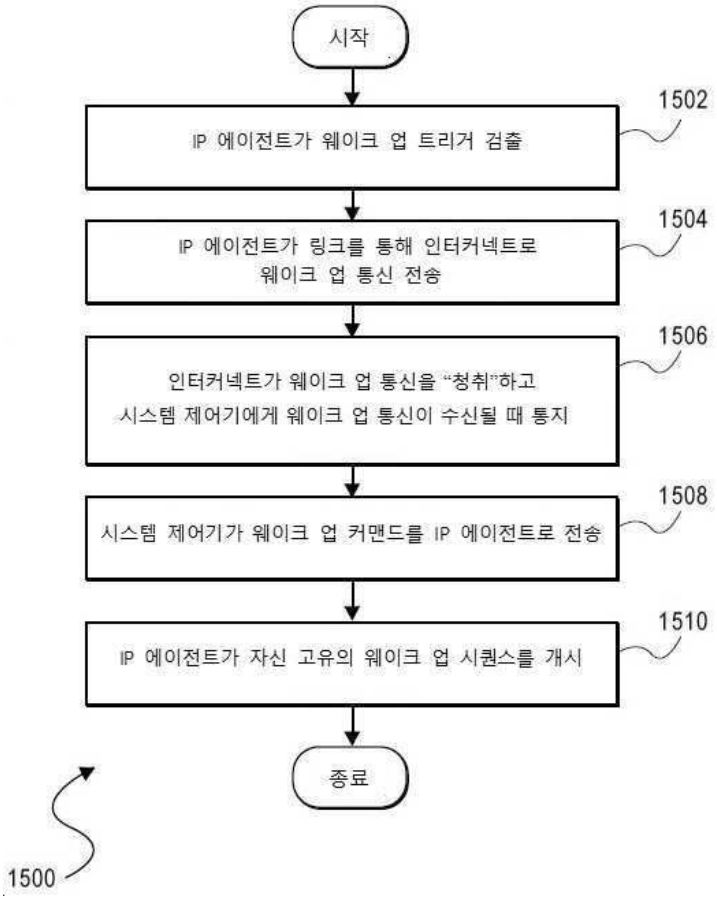
도면13


시작
1202
정규 모드에서 에이전트 동작
1302
저전력 모드, 동작 불능,
상태 정보 유지 안됨으로 전환되기 위한
결정이 내려졌는가?
아니오
예
1206
인터커넥트를 프록시로서 구성
1304
적용 가능할 때 클록 끄기/전력 감소
1212
정규 동작을 재개하기 위한
결정이 내려졌는가?
아니오
예
1306
적용 가능할 때 클록 재개/전력 증가
1308
IP 에이전트 리셋
1300
1310
작업거부 모드
빠져 나오기
아니오
예
1312
인터커넥트는 더는
프록시로서 동작하지
않음

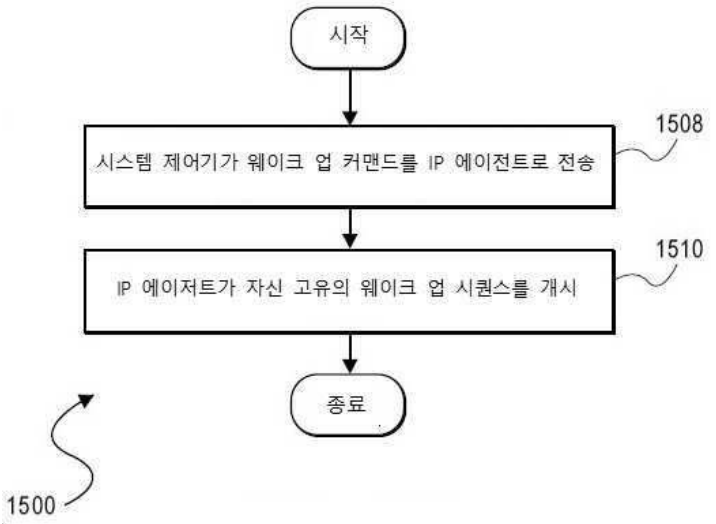
도면14


시작
시스템 제어기가 IP 에이전트가 동작 불능 상태로 들어가기 위한 결정을 내림
1402
에이전트가 트랜잭션을 생성하는 것을 중단하도록 명령함
1404
1406
아니오
모든 미결 트랜잭션이 완료되는가
예
작업거부 상태에서 링크가 인터커넥트 세트를 프록시로서 동작하도록 설정함
1408
IP 에이전트 리셋/저전력/파워 오프 개시
1410
완료
1400

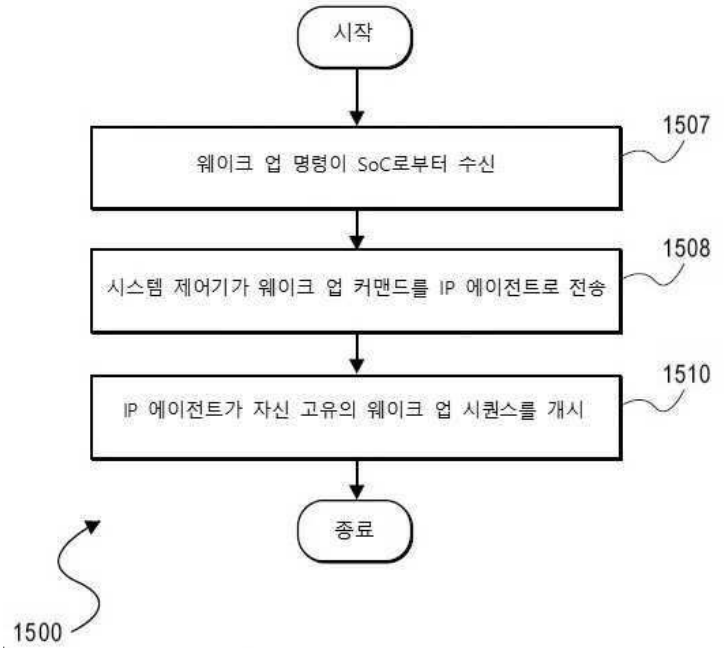
도면15a


시작
IP 에이전트가 웨이크 업 트리거 검출
1502
IP 에이전트가 링크를 통해 인터커넥트로
웨이크 업 통신 전송
1504
인터커넥트가 웨이크 업 통신을 "청취"하고
시스템 제어기에게 웨이크 업 통신이 수신될 때 통지
1506
시스템 제어기가 웨이크 업 커맨드를 IP 에이전트로 전송
1508
IP 에이전트가 자신 고유의 웨이크 업 시퀀스를 개시
1510
종료
1500

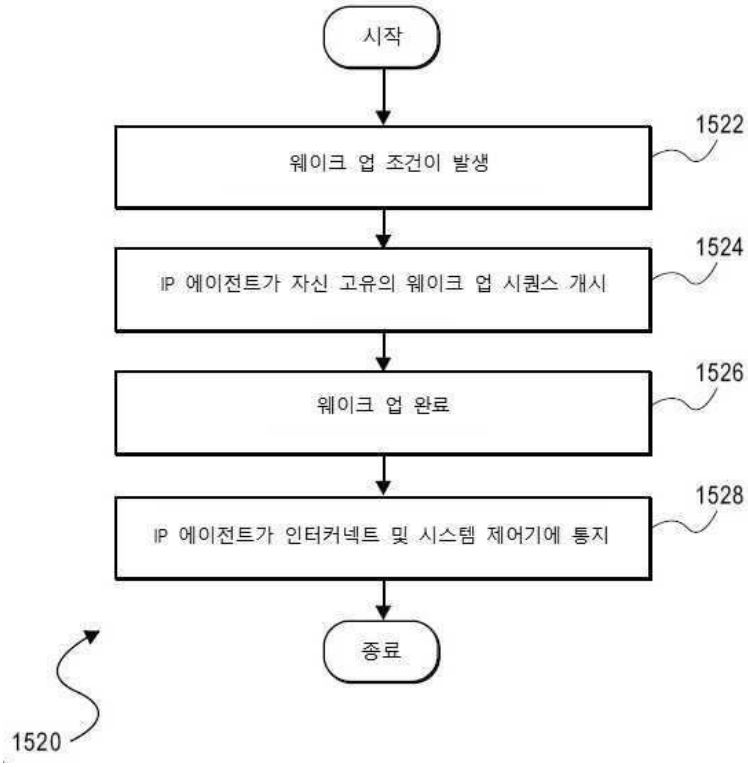
도면15b


시작
시스템 제어기가 웨이크 업 커맨드를 IP 에이전트로 전송
1508
IP 에이저트가 자신 고유의 웨이크 업 시퀀스를 개시
1510
종료
1500

도면15c


시작
웨이크 업 명령이 SoC로부터 수신
1507
시스템 제어기가 웨이크 업 커맨드를 IP 에이전트로 전송
1508
IP 에이전트가 자신 고유의 웨이크 업 시퀀스를 개시
1510
종료
1500

도면15d


시작
웨이크 업 조건이 발생
1522
IP 에이전트가 자신 고유의 웨이크 업 시퀀스 개시
1524
웨이크 업 완료
1526
IP 에이전트가 인터커넥트 및 시스템 제어기에 통지
1528
종료
1520