(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2021년08월05일
(11) 등록번호 10-2285138
(24) 등록일자 2021년07월28일

(51) 국제특허분류(Int. Cl.)
*G06F 30/00* (2020.01)
(52) CPC특허분류
*G06F 30/20* (2020.01)
*G06F 30/30* (2020.01)
(21) 출원번호 10-2016-7001384
(22) 출원일자(국제) 2014년07월25일
심사청구일자 2019년07월25일
(85) 번역문제출일자 2016년01월18일
(65) 공개번호 10-2016-0033695
(43) 공개일자 2016년03월28일
(86) 국제출원번호 PCT/US2014/048190
(87) 국제공개번호 WO 2015/013609
국제공개일자 2015년01월29일
(30) 우선권주장
13/951,098 2013년07월25일 미국(US)

(56) 선행기술조사문헌
"Netrace: Dependency-Tracking Traces for Efficient Network-on-Chip Experimentation" The University of Texas a t Austin, Department o f Computer Science, May 2011
"Evaluation of Multiple-Valued Packet Multiplexing Scheme for Network-on-Chip Architecture" In: Proceedings of the 36th International Symposium on Multiple-Valued Logic, 2006
"Scalable connection-based flow control scheme for application-specific network-on-chip" The Journal of China Universities of Posts and Telecommunications Vol. 18 No. 6(pp. 98-105), December 2011
US20110191774 A1

(73) 특허권자
**넷스피드 시스템즈**
미국, 캘리포니아 95134, 산 호세, 빌딩 11, 실리 애비뉴 2670
(72) 발명자
**쿠말, 살리에쉬**
미국, 캘리포니아 95134, 산 호세, 빌딩 11, 실리 애비뉴 2670, 넷스피드 시스템즈 사내
**파탄카르, 아미트**
미국, 캘리포니아 95134, 산 호세, 빌딩 11, 실리 애비뉴 2670, 넷스피드 시스템즈 사내
**노리쥬, 에릭**
미국, 캘리포니아 95134, 산 호세, 빌딩 11, 실리 애비뉴 2670, 넷스피드 시스템즈 사내
(74) 대리인
**양영준, 김연송, 백만기**

전체 청구항 수 : 총 18 항 심사관 : 박승철

(54) 발명의 명칭 **네트워크 온 칩에서의 시스템 레벨 시뮬레이션**

## (57) 요 약

전체 SoC 거동을 정확하게 흉내 내는 것에 의해, 여러 메시지 트랜잭션 기반의 네트워크 온 칩(NoC) 상호연결 아키텍처 내의 SoC IP 코어들의 성능 시뮬레이션이 개시된다. 예시적인 구현들은 다른 레이트/시간/간격들로 여러 트랜잭션의 수행에 기초하여 NoC 거동을 평가 및 검출하는 시뮬레이션을 수반하며, 각 트랜잭션은 하나 이상의 메시지를 포함하고, 각 메시지는 소스 에이전트와 목적지 에이전트를 포함한다. 각 메시지는 또한, NoC 상호연결에서 성능 보고서를 생성하기 위해 실시간 시나리오를 시뮬레이션하기 위해, 시뮬레이터에 의해 트랜잭션의 수행을 표시하도록 구성될 수 있는 여러 파라미터를 예를 들어 레이트, 크기, 값, 지연을 포함할 수 있다.

## 대 표 도

(52) CPC특허분류
*G06F 30/33* (2020.01)

# 명 세 서

## 청구범위

### 청구항 1

프로세스를 실행하기 위한 인스트럭션들을 저장하는 비일시적 컴퓨터 판독가능한 저장 매체로서, 상기 인스트럭션들은:

컴퓨터 상에서, 다수의 트랜잭션을 사용하여 네트워크 온 칩(NoC) 상호연결의 시뮬레이션을 수행하는 것 - 상기 다수의 트랜잭션 각각은 하나 이상의 메시지의 시퀀스를 포함하고, 상기 하나 이상의 메시지 각각은 소스 에이전트 및 목적지 에이전트 중 적어도 하나에 대한 표시를 포함함 -; 및

상기 하나 이상의 메시지의 시퀀스에서의 제1 메시지 목적지 노드에 기초하여 상기 다수의 트랜잭션 중의 트랜잭션들에 대한 상기 하나 이상의 메시지의 시퀀스에서 후속 메시지들을 생성하는 것 - 상기 후속 메시지들은 상기 하나 이상의 메시지의 각각의 목적지 노드에서 생성됨 -

을 포함하는, 비일시적 컴퓨터 판독가능한 저장 매체.

### 청구항 2

청구항 1에 있어서,

상기 하나 이상의 메시지 각각은 레이트, 우선순위, 값, 메시지 데이터 크기, 지연, 그리고 간격 중 적어도 하나에 대한 표시를 포함하고,

메시지들은, 상기 레이트, 상기 우선순위, 상기 값, 상기 메시지 데이터 크기, 상기 지연, 그리고 상기 간격 중 적어도 하나에 기초하여 상기 시뮬레이션을 수행하는데 사용되는, 비일시적 컴퓨터 판독가능한 저장 매체.

### 청구항 3

청구항 2에 있어서,

상기 하나 이상의 메시지 각각은 레이트에 대한 표시를 포함하고, 상기 레이트는 상기 시뮬레이션 중에 각 메시지의 선택의 확률의 표시를 포함하는, 비일시적 컴퓨터 판독가능한 저장 매체.

### 청구항 4

청구항 1에 있어서,

상기 인스트럭션들은 각 에이전트로부터 유래한 상기 다수의 트랜잭션 시퀀스 중의 트랜잭션 시퀀스들에 기초하여 각 에이전트를 위한 트레이스 파일을 생성하는 것을 더 포함하며,

상기 트레이스 파일은 상기 다수의 트랜잭션 시퀀스 중의 트랜잭션 시퀀스들에서의 상기 하나 이상의 메시지의 서브세트(subset)를 포함하는, 비일시적 컴퓨터 판독가능한 저장 매체.

### 청구항 5

청구항 1에 있어서,

상기 인스트럭션들은 상기 시뮬레이션에 기초하여 NoC 상호연결의 성능 보고서를 생성하는 것을 더 포함하는, 비일시적 컴퓨터 판독가능한 저장 매체.

### 청구항 6

프로세스를 실행하기 위한 인스트럭션들을 저장하는 비일시적 컴퓨터 판독가능한 저장 매체로서, 상기 인스트럭션들은:

다수의 트랜잭션을 사용하여 네트워크 온 칩(NoC) 상호연결의 시뮬레이션을 수행하는 것을 포함하고, 상기 다수의 트랜잭션 각각은 하나 이상의 메시지의 시퀀스를 포함하고, 상기 하나 이상의 메시지 각각은 소스 에이전트

및 목적지 에이전트 중 적어도 하나에 대한 표시를 포함하고,

상기 하나 이상의 메시지 각각은 우선순위, 값, 메시지 데이터 크기, 지연, 그리고 간격 중 적어도 하나 및 레이트에 대한 표시를 포함하고,

상기 하나 이상의 메시지는, 상기 레이트, 상기 우선순위, 상기 값, 상기 메시지 데이터 크기, 상기 지연, 그리고 상기 간격 중 적어도 하나에 기초하여 상기 시뮬레이션을 수행하는데 사용되고,

상기 레이트는 상기 시뮬레이션 중에 각 메시지의 선택의 확률의 표시를 포함하는, 비일시적 컴퓨터 판독가능한 저장 매체.

**청구항 7**

청구항 6에 있어서,

상기 인스트럭션들은 각 에이전트로부터 유래한 상기 다수의 트랜잭션 시퀀스 중의 트랜잭션 시퀀스들에 기초하여 각 에이전트를 위한 트레이스 파일을 생성하는 것을 더 포함하며,

상기 트레이스 파일은 상기 다수의 트랜잭션 시퀀스 각각을 위한 하나 이상의 메시지의 표시를 포함하는, 비일시적 컴퓨터 판독가능한 저장 매체.

**청구항 8**

청구항 6에 있어서,

상기 인스트럭션들은 제1 메시지 목적지 노드에 기초하여 상기 다수의 트랜잭션 시퀀스 중의 트랜잭션 시퀀스들을 위한 후속 메시지들을 생성하는 것을 더 포함하는, 비일시적 컴퓨터 판독가능한 저장 매체.

**청구항 9**

청구항 6에 있어서,

상기 인스트럭션들은 상기 시뮬레이션에 기초하여 NoC 상호연결의 성능 보고서를 생성하는 것을 더 포함하는, 비일시적 컴퓨터 판독가능한 저장 매체.

**청구항 10**

컴퓨터에 의해 각 단계가 수행되는 방법에 있어서,

상기 방법은 다수의 트랜잭션을 사용하여 네트워크 온 칩(NoC) 상호연결의 시뮬레이션을 수행하는 단계를 포함하고, 상기 다수의 트랜잭션 각각은 하나 이상의 메시지의 시퀀스를 포함하고, 상기 하나 이상의 메시지 각각은 소스 에이전트 및 목적지 에이전트 중 적어도 하나에 대한 표시를 포함하고,

상기 하나 이상의 메시지 각각은 우선순위, 값, 메시지 데이터 크기, 지연, 그리고 간격 중 적어도 하나 및 레이트에 대한 표시를 포함하고,

상기 하나 이상의 메시지는, 상기 레이트, 상기 우선순위, 상기 값, 상기 메시지 데이터 크기, 상기 지연, 그리고 상기 간격 중 적어도 하나에 기초하여 상기 시뮬레이션을 수행하는데 사용되고,

상기 레이트는 상기 시뮬레이션 중에 각 메시지의 선택의 확률의 표시를 포함하는, 방법.

**청구항 11**

청구항 10에 있어서,

각 에이전트로부터 유래한 상기 다수의 트랜잭션 시퀀스 중의 트랜잭션 시퀀스들에 기초하여 각 에이전트를 위한 트레이스 파일을 생성하는 단계를 더 포함하고,

상기 트레이스 파일은 상기 다수의 트랜잭션 시퀀스 각각을 위한 하나 이상의 메시지의 표시를 포함하는, 방법.

**청구항 12**

청구항 10에 있어서,

제1 메시지 목적지 노드에 기초하여 상기 다수의 트랜잭션 시퀀스 중의 트랜잭션 시퀀스들을 위한 후속 메시지들을 생성하는 단계를 더 포함하는, 방법.

**청구항 13**

청구항 10에 있어서,

상기 시뮬레이션에 기초하여 NoC 상호연결의 성능 보고서를 생성하는 단계를 더 포함하는 방법.

**청구항 14**

컴퓨터에 의해 각 단계가 수행되는 방법에 있어서,

상기 방법은 다수의 트랜잭션을 사용하여 네트워크 온 칩(NoC) 상호연결의 시뮬레이션을 수행하는 단계 - 상기 다수의 트랜잭션 각각은 하나 이상의 메시지의 시퀀스를 포함하고, 상기 하나 이상의 메시지 각각은 소스 에이전트 및 목적지 에이전트 중 적어도 하나에 대한 표시를 포함함 -; 및

상기 하나 이상의 메시지의 시퀀스에서의 제1 메시지 목적지 노드에 기초하여 상기 다수의 트랜잭션 중의 트랜잭션들에 대한 상기 하나 이상의 메시지의 시퀀스에서 후속 메시지들을 생성하는 단계 - 상기 후속 메시지들은 상기 하나 이상의 메시지의 각각의 목적지 노드에서 생성됨 -

를 포함하는 방법.

**청구항 15**

청구항 14에 있어서,

상기 하나 이상의 메시지 각각은 레이트, 우선순위, 값, 메시지 데이터 크기, 지연, 그리고 간격 중 적어도 하나에 대한 표시를 포함하고,

메시지들은, 상기 레이트, 상기 우선순위, 상기 값, 상기 메시지 데이터 크기, 상기 지연, 그리고 상기 간격 중 적어도 하나에 기초하여 상기 시뮬레이션을 수행하는데 사용되는, 방법.

**청구항 16**

청구항 14에 있어서,

상기 하나 이상의 메시지 각각은 레이트에 대한 표시를 포함하고, 상기 레이트는 상기 시뮬레이션 중에 각 메시지의 선택의 확률의 표시를 포함하는, 방법.

**청구항 17**

청구항 14에 있어서,

각 에이전트로부터 유래한 상기 다수의 트랜잭션 시퀀스 중의 트랜잭션 시퀀스들에 기초하여 각 에이전트를 위한 트레이스 파일을 생성하는 단계를 더 포함하며,

상기 트레이스 파일은 상기 다수의 트랜잭션 시퀀스 중의 트랜잭션 시퀀스들에서의 하나 이상의 메시지의 서브세트를 포함하는, 방법.

**청구항 18**

청구항 14에 있어서,

상기 시뮬레이션에 기초하여 NoC 상호연결의 성능 보고서를 생성하는 단계를 더 포함하는 방법.

## *발명의 설명*

### *기 술 분 야*

[0001] 본 명세서에 서술된 방법 및 예시적인 구현들은 상호연결 아키텍처에 대한 것으로서, 더 상세하게는 네트워크 온 칩(NoC)에서 NoC IP 코어들의 멀티-홉 트랜잭션 성능 시뮬레이션의 수행에 대한 것이다.

## 배 경 기 술

[0002] 집적도, 시스템 복잡도, 그리고 트랜지스터 크기 축소의 증가로 인해서, 칩의 부품(component) 개수가 급격하게 증가하고 있다. 칩 멀티-프로세서(CMP)들은 많은 수의 동종 프로세서 코어(processor core), 메모리 그리고 I/O 서브시스템을 포함하지만, 시스템-온-칩(SoC)들 복합체는 다양한 부품, 예를 들어, 프로세서 코어, DSP, 하드웨어 가속기, 메모리 그리고 I/O를 포함할 수 있다. 두 시스템 모두에서 온-칩 상호연결(interconnection)은, 다양한 부품 사이의 고성능 통신을 제공하는데 중요한 역할을 한다. 전통적인 버스 및 크로스바(crossbar) 기반 상호연결의 적응성(scalability) 제한 때문에, 네트워크-온-칩(NoC)이 많은 개수의 부품을 칩 상에서 상호연결하는 패러다임으로 출현하였다. 네트워크-온-칩은 점대점(point-to-point) 물리적 링크를 사용하여 서로 상호연결된 다수의 라우팅 노드로 구성된 전역 공유 통신 인프라스트럭처이다.

[0003] 메시지는 소스에서 주입되고 여러 중간 노드 및 물리적 링크를 통해 소스 노드로부터 목적지로 라우팅 된다. 목적지 노드는 이어서 메시지를 뽑아내어 목적지에 제공한다. 이하에서, 용어 '부품', '블록'(block), 호스트 또는 '코어'는 네트워크 온 칩을 사용하여 상호연결된  다양한 시스템 부품을 가리키기 위해서 상호 교환되어 사용될 것이다. 용어 '라우터' 및 '노드'는 또한 상호 교환되어 사용될 것이다. 일반성을 잃지 않고, 여러 개의 상호연결된 부품을 갖는 시스템은 그 자체로 '멀티-코어 시스템'(multi-core system)으로 언급될 것이다.

[0004] 시스템 네트워크를 형성하기 위해서 라우터들이 서로 연결되는 가능한 통신망구성(topology)이 여러 가지 있다. 양방향성 링(bi-directional ring)(도 1a), 이차원 그물망(2-D mesh)(도 1b), 및 2-D 토러스(Torus)(도 1c)는 관련 기술 분야에서 통신망구성의 예이다. 그물망 및 토러스는 또한 2.5-D(2.5차원) 또는 3-D(3차원) 조직으로 확장될 수 있다. 도 1d는 3-D 그물망 NoC를 도시하는바, 적층된 3층의 3x3 2D 그물망 NoC이 도시되어 있다. NoC 라우터들은 추가로 두 개의 포트를 구비하는데, 하나는 상위 계층의 라우터에 연결되고 다른 하나는 하위 계층의 라우터에 연결된다. 도시된 예에서 중간 계층의 라우터(111)는 두 포트를 구비하는데, 하나는 최상층의 라우터에, 다른 하나는 바닥층의 라우터에 연결된다. 라우터(110) 및 라우터(112)는 각각 바닥층 및 최상층에 위치하고 따라서 이들은 각각 하나의 상위 계층을 향하는 포트(113)와 하위 계층을 향하는 포트(114)를 구비한다.

[0005] 패킷은 다양한 부품 사이의 상호통신을 위한 메시지 전달 단위이다. 라우팅은 패킷이 소스로부터 목적지로 전달되는 네트워크의 라우터 및 물리적 링크 세트로 구성된 경로를 확인함을 포함한다. 부품들은 하나 또는 여러 라우터의 하나 또는 여러 포트에 연결된다; 이 같은 각 포트는 고유 ID를 가지고 있다. 패킷은, 목적지 부품으로 라우팅 되기 위해서, 중간 라우터가 사용하게 되는 목적지의 라우터 및 포트 ID를 소지한다.

[0006] 라우팅 기술의 예는 결정론적 라우팅을 포함하는데 이는 모든 패킷에 대해서 A로부터 B로 동일한 경로를 선택함을 포함한다. 이 같은 라우팅 형태는 네트워크의 상태에 대해 독립적이며, 기반 네트워크(underlying network)에 존재할 수 있는 경로 다이버시티(path diversity)들에 대해 균형을 부과하지(load) 않는다. 하지만, 이 같은 결정론적 라우팅은 하드웨어로 구현하기가 간단할 수 있고 패킷 순서를 유지하고, 쉽사리 네트워크 레벨에서의 교착상태가 없도록 할 수 있다. 최단 경로 라우팅은 소스로부터 목적지로의 홉 수를 감소시키기 때문에 지연을 최소화할 수 있다. 이 같은 이유로 인해, 최단 경로는 또한 두 부품의 통신에서 최저 전력 소모 경로일 수 있다. 차원-순서(dimension-order) 라우팅은 2-D, 2.5-D, 및 3-D 그물망 네트워크에서 결정론적 최단 경로 라우팅의 한 형태이다. 이 라우팅 방안에서, 메시지들은, 최종 목적지에 도달하기 전까지, 특정 순서로 각 좌표를 따라 라우팅 된다. 예를 들면 3-D 그물망 네트워크의 경우, 메시지는, 목적지 라우터의 X-좌표와 동일한 X-좌표를 갖는 라우터에 도달하기까지, 먼저 X 차원을 따라 라우팅 될 것이다. 다음으로, 메시지는 방향을 바꾸어 Y 차원을 따라 라우팅 되고, 최종 목적지 라우터에 도달할 때까지 마지막으로 방향을 바꾸어 Z 차원을 따라 이동할 것이다. 차원 순서 라우팅은 최소의 방향 전환일 수 있고 최단 경로 라우팅일 수 있다.

[0007] 도 2a는 2차원 그물망에서 XY 라우팅의 예를 도시한다. 더 상세하게, 도 2a는 노드 '34'로부터 노드 '00'로의 XY 라우팅을 도시한다. 도 2a의 예에서, 각 부품은 라우터 하나의 단지 한 개의 포트에 연결된다. 패킷은, 노드의 X 좌표가 목적지 노드의 X 좌표와 동일한 좌표를 갖는 노드 '04'에 달할 때까지, 먼저 X축을 따라 라우팅 된다. 패킷은 이어서 목적지 노드에 도달할 때까지 Y축을 따라 라우팅 된다.

[0008] 하나 또는 그 이상의 라우터 또는 하나 또는 그 이상의 링크가 없는 이종 그물망 망구성에서, 차원 순서 라우팅은 약간의 소스 및 목적지 노드 사이에서 실현 가능하지 않을 수 있고, 대안 경로들이 채택되어야 할 수 있다. 대안 경로들은 최단 경로 또는 최소 방향 전환이 아닐 수 있다.

[0009] 테이블을 사용하는 라우팅 및 소스 라우팅은 NoC에서 사용되는 다른 라우팅 옵션이다. 적응 라우팅은 네트워크의 상태에 기초하여 네트워크상의 두 지점 사이의 경로를 동적으로 변경한다. 이 같은 형태의 라우팅은 분석 및

구현에 있어서 복잡할 수 있다.

[0010] NoC은 여러 개의 물리적 네트워크를 포함할 수 있다. 각 물리적 네트워크에 대해서, 다른 형태의 메시지들이 다른 가상 네트워크들을 통해 전달되는 여러 개의 가상 네트워크가 존재할 수 있다. 이 경우, 각 물리적 링크 또는 채널에서, 여러 개의 가상 채널이 있다; 각 가상 채널은 두 끝 지점 모두에 전용 버퍼를 구비할 수 있다. 임의의 주어진 클록 사이클에서, 단지 하나의 가상 채널이 데이터를 물리적 채널에 전달할 수 있다.

[0011] NoC 상호연결은 웜홀(wormhole) 라우팅을 채용할 수 있다. 웜홀 라우팅에서는 큰 메시지 또는 패킷이 플릿(filt)(또한 흐름 제어 디지트(digit) 언급된다)으로 알려진 작은 조각으로 쪼개진다. 제1 플릿은 헤더 플릿으로 페이로드 데이터와 함께 중요 메시지 레벨 정보(key message level info)와 패킷의 라우트(rout)에 대한 정보를 보유하며, 이 메시지와 관련된 모든 후속 플릿에 대한 라우팅 양식을 마련한다. 옵션으로서, 0 또는 보다 많은 본체 플릿이 헤드 플릿을 뒤따르며, 본체 플릿은 나머지 데이터 페이로드를 보유한다. 최후 플릿은 꼬리 플릿으로서 마지막 데이터 페이로드를 보유할 뿐만 아니라 메시지에 대한 연결을 폐쇄하기 위해 몇몇 부기(bookkeeping)를 수행한다. 웜홀 흐름 제어에서, 가상 채널이 종종 구현된다.

[0012] 물리적 채널은 가상 채널(VC)이라고 불리는 복수 개의 독립적인 논리 채널로 시간 구획(time slice)된다. VC은 패킷을 라우팅하기 위해 여러 개의 독립적인 경로를 제공하지만, 물리적인 채널에서 시간-다중화된다. 가상 채널은 채널 상에서 패킷의 플릿들의 처리를 조정하는데 필요한 상태를 간직한다. 최소한도로, 이 상태는, 라우트의 다음 홉을 위한 현재 노드의 출력 채널과 가상 채널의 상태(유휴 상태, 소스를 기다리는 상태, 또는 활성화 상태)를 확인(식별)한다. 가상 채널은 또한, 현재 노드 상에 버퍼 되는(buffered on) 패킷의 플릿들에 대한 포인터와 다음 노드에서 가용한 플릿 버퍼들의 개수를 포함한다.

[0013] 용어 "웜홀"은 채널 상에서 메시지가 전달되는 방식에 관련된다; 다음 라우터의 출력 포트가 아주 짧아 수신된 데이터가 전체 메시지가 도달하기 전에 헤드 플릿에 전달될 수 있다. 이는, 라우터를 헤드 플릿이 도착하자마자 빠르게 설정하도록 하고 이어서 대화의 나머지로부터 옵트 아웃(opt out) 하도록 한다. 메시지가 플릿 단위로 전달되기 때문에, 메시지는 다른 라우터들에서 그 경로를 따라서 여러 개의 플릿 버퍼를 점유할 수 있고, 이는 벌레(worm) 모양 이미지를 생성한다.

[0014] 다양한 말단 점들 사이의 통신량에 기초하여 그리고 다양한 메시지들을 위해 사용되는 라우트 및 물리적 네트워크에 기초하여, NoC 상호연결의 다른 물리적 채널들이 다른 레벨의 부하(적재) 및 적체(load and congestion)를 경험할 것이다. NoC 상호연결의 다양한 물리적 채널들의 용량은 채널 폭(물리적 배선의 개수) 및 동작하는 클록 주파수에 의해 결정된다. NoC의 다양한 채널들은 다른 클록 주파수들에서 작동할 수 있고, 다양한 채널들은 당해 채널의 대역폭 조건에 따라 다른 폭(width)을 가질 수 있다. 채널의 대역폭 조건은 채널을 이동하는(traverse) 흐름(flow)에 의해 결정된다. 다양한 NoC 채널을 이동하는 흐름은 다양한 흐름에 의해 점유되는 라우트에 의해 영향을 받는다. 그물망 또는 토러스 NoC에서, 소스 및 목적지 노드 사이의 홉의 길이 또는 개수와 동일한 라우팅 경로가 여러 개 존재할 수 있다. 예를 들어 도 2b에서, 노드 34 및 노드 00 사이의 표준 XY 라우트에 더해서, 추가로 가용한 라우트 예를 들어 XY 라우트 203 또는 소스에서 목적지로의 1회보다 많은 횟수의 방향 전환을 하는 여러 번의 방향 전환 라우트 202가 존재할 수 있다.

[0015] 다양한 트래픽 서행(traffic slow)에 대한 정적으로 할당된 라우트의 NoC에서, 다양한 채널에서의 적재는 다양한 흐름에 대해 지능적으로 라우트를 선택함으로써 제어될 수 있다. 트래픽 량이 엄청나고 실질적인 경로 다이버시티가 존재할 때, 모든 NoC 채널의 적재가 거의 균등하게 균형이 되도록 라우트가 선택될 수 있고 이로써 한 점의 교착상태를 피할 수 있다. 일단 라우트가 선택되면, NoC 채널 폭은 채널 상의 흐름의 대역폭 수요에 기반하여 결정될 수 있다. 불행히도, 채널 폭은 타이밍 또는 배선 적체(wiring congestion) 같은 물리적인 하드웨어 디자인 제한으로 인해 임의대로 커질 수 없다. 최대 채널 폭이 제한될 수 있고 이에 따라 임의의 한 NoC 채널의 최대 대역폭에 제한을 하가게 된다.

[0016] 덧붙여, 메시지가 짧다면, 물리적으로 더 넓은 채널이 더 높은 대역폭을 달성하는데 도움이 되지 않을 수 있다. 예를 들어, 패킷이 64-비트 폭의 단일 플릿 패킷이라면, 채널이 아무리 넓더라도, 그 채널은 모든 패킷이 비슷하다면 데이터 사이클당 단지 64-비트만을 전달할 것이다. 따라서, NoC에서 채널 폭은 또한 메시지 크기에 의해서도 제한된다. 최대 NoC 채널 폭에 대한 이 같은 제한들로 인해서, 채널이 라우트의 균형에도 충분한 대역폭을 가지지 않을 수 있다.

[0017] 전술한 대역폭 관련 문제를 해결하기 위해서, 여러 개의 병렬의 물리적 NoC가 사용될 수 있다. 각 NoC는 층으로 불릴 수 있고 따라서 다층 NoC 구조를 형성한다. 호스트는 메시지를 Noc 층에 주입한다; 메시지는 이어서 NoC층

상의 목적지로 라우팅 되는데, 메시지는 NoC 층에서 호스트로 전달된다. 따라서 각 층은 서로에 대해서 어느 정도 독립적으로 동작하고, 층들 사이의 상호작용은 단지 메시지의 주입 및 추출 시에만 일어난다. 도 3a는 두 층의 NoC를 도시한다. 여기서 NoC 두 층은 좌측 및 우측에 서로 인접하게 도시되었으며, 좌측 및 우측 도면 모두에 복제된 NoC에 호스트들이 연결된다. 호스트는 이 예에서 두 개의 라우터 - R1으로 표시된 제1 층의 라우터와 R2로 표시된 제2 층의 라우터- 에 연결된다. 이 예에서, 다층 NoC는 3D NoC와 다르다. 즉, 다층이 하나의 실리콘 다이에 존재하고 동일한 실리콘 다이 상의 호스트들 사이의 통신의 높은 대역폭 수요를 충족하는데 사용된다. 메시지는 한 층에서 다른 층으로 가지 않는다. 명확성을 위해서, 본 출원은, NoC 들이 서로에 대해서 수직으로 도시되는 3D NoC와 구별하기 위해, 다층 NoC를 수평으로 좌측 및 우측에 도시하는 것을 사용할 것이다.

[0018] 도 3b에서, 도시된 바와 같이 호스트는 각 층의 라우터 R1 및 R2에 연결된다. 각 라우터 R1, R2는 방향 포트(directional port)(301)를 사용하여 자신이 속한 층에서 다른 라우터들에 연결되고, 추출 포트(302)를 사용하여 호스트에 연결된다. 브리지 로직(303)이 나가는 메시지를 위한 NoC 층을 결정하기 위해서 호스트와 두 NoC 층 사이에 위치하고, 호스트로부터 NoC층으로 메시지를 보내고, 또한 NoC 층들로부터의 오는 메시지들의 중재 및 선택(multiplexing)을 수행하고 호스트로 전달한다.

[0019] 다층 NoC에서, 필요한 층의 개수는 여러 요인 예를 들어 시스템의 모든 트래픽 양의 전체 대역폭 조건, 다양한 흐름에 사용되는 라우터들, 메시지 크기 분포, 최대 채널 폭 등에 의존할 수 있다. NoC 상호연결에서 NoC 층의 개수가 디자인으로 정해지면, 다른 메시지들 및 트래픽 흐름은 다른 NoC 층들에서 라우팅 될 것이다. 덧붙여, 다른 층들이 라우터, 채널 및 연결의 개수에서 다른 망구성을 갖도록 NoC 상호연결을 디자인할 수 있다. 다른 층의 채널은 채널을 이동하는 흐름 및 그 대역폭 조건에 기하여 다른 폭을 가질 수 있다.

[0020] NoC 상호연결에서, 트래픽 프로파일이 균일하지 않고 어느 정도의 이종성(예를 들어, 어떤 호스트는 다른 호스트들에 비해 더 자주 통신함)이 있다면, 상호연결 성능은 NoC 망구성에 의존할 것이다. 즉 서로에 대해서 다양한 호스트가 망구성에서 어디에 놓이는지 그리고 어떤 라우터에 연결되는지에 상당히 의존할 것이다. 예를 들어, 두 호스트가 빈번히 통신하고 높은 대역폭을 필요로 할 경우, 이들은 서로 인접하게 배치되어야 한다. 이는 통신의 지연을 줄일 것이고, 그래서 전체 평균 지연을 줄일 뿐만 아니라 그 통신의 높은 대역폭이 제공되어야만 하는 라우터 노드 및 링크의 수를 줄인다. 서로 중첩하지 않고 모든 호스트가 2D 평면의 NoC 망구성에 들어맞아야 하기 때문에, 서로 인접한 두 호스트를 이동하게 되면 다른 호스트들이 멀리 떨어지게 된다. 따라서, 바른 이해 상충이 이루어져야 하고, 비용 및 성능 메트릭스(performance metrics)가 최적화되도록 쌍 단위 메트릭스과 모든 호스트 사이의 지연 조건을 검사한 후에 호스트들이 배치되어야 한다. 비용 및 성능 메트릭스는 모든 통신하는 호스트 사이의 라우터 홉의 개수로 표시되는 평균 구조 지연, 또는 모든 쌍의 호스트 사이의 대역폭의 합 및 홉의 개수로 표시되는 거리, 또는 이들의 조합을 포함할 수 있다. 이 최적화 문제는 비-결정론적 다항식-시간 난해(NP-hard)가 될 것으로 알려져 있고, 경험 기반 접근법들이 종종 사용된다. 시스템에서 호스트들은 형태 및 크기가 다양하고 이는 호스트들을 2D 평면의 NoC 망구성에 배치하고, 호스트들을 여백이 거의 없이 최적으로 패키지하고 호스트들의 중첩을 피하는데 추가의 어려움을 부가한다.

[0021] 여러 NoC 구성 예를 들어 라우터, 브리지, 채널을 평가하기 위한 모델을 사용하여 NoC 성능 시뮬레이션이 종종 수행된다. 이후, 시뮬레이션 에이전트 또는 간단히 에이전트로도 알려진 부품/코어에 대한 대표 모델(트래픽을 송신 및 수신함)이 사용되어 트래픽 자극을 생성한다. 에이전트 모델들은, 다양한 레이트(rate)로 다양한 크기 및 값의 메시지를 생성함으로써 그리고 다른 다양한 에이전트들이 목적지가 되는 데이터 간 다양한 간격을 가지게 함으로써, 실제 에이전트/부품의 거동을 흉내 낸다. 메시지의 생성 레이트, 간격, 값 그리고 목적지는, 다른 조건들 하에서 성능 거동을 포획(capture)하는 에이전트의 통계 성질에 기반하여, 다양할 수 있다. 이 시뮬레이션들이 진행되는 동안, 메시지를 전달할 수 있은 모든 에이전트는 독립적으로 메시지를 생성하며 생성된 메시지들은 목적지 에이전트가 수신한다. 복잡한 시스템에서, 에이전트가 수신한 메시지가 다른 메시지의 전달을 낳는 거동이 있을 수 있다. 더욱이, 에이전트가 메시지를 수신할 때, 다양한 목적지들로 보내질 메시지들이 생성될 수 있는 메시지들의 형태가 다양할 수 있다. 예를 들어, 캐시 컨트롤러가 CPU로부터 메시지를 수신하고 캐시 히트(cache hit)가 있다면 CPU에 반응 메시지를 생성하고, 미스(miss)가 있다면 메모리에 재충전(refill) 메시지를 생성할 수 있다. 따라서, CPU에서 캐시로, 메모리로 그리고 백(back)으로의 다양한 메시지 시퀀스들이 있다.

[0022] 하지만, 종래 기술의 NoC 성능 시뮬레이터는 다자-점(multipoint) 상호 의존(inter-dependent) 메시지 시퀀스 발생 거동을 모방하지 않고 대신에 단지 점대점(point-to-point) 메시지를 독립적으로 생성함으로써 NoC를 시뮬레이션하고 있다; 즉 위 예에서와 같이, CPU로부터 캐시로의 요청 메시지, 캐시로부터 CPU로의 반응 메시지, 캐시로부터 메모리로의 충전 메시지 등이 각 소스 에이전트에서 목적지 에이전트로 독립적으로 생성되고, 세 개의

독립적인 메시지 흐름을 생성한다. 이 흐름들의 속도, 간격 및 속성은 에이전트의 정확한 거동을 흉내 내도록 제어된다. 이 같은 자극이 비록 다른 메시지 흐름의 대역폭 특성을 정확하게 포획할 수 있지만, 실제 시스템에 있을 것이기 때문에 다른 흐름들은 동기화되지 않을 것이다. 예를 들어, 독립적으로 생성되기 때문에, 요청 메시지 전에 충전 메시지가 생성될 수 있다.

[0023] 에이전트에서 요청이 결정론적으로 반응을 생성할 경우 요청 메시지에 대한 반응 메시지를 생성하는 시도를 하는 NoC 시뮬레이터가 관련기술에 거의 없다. 하지만, 메시지를 수신한 후 에이전트가 에이전트의 상태, 네트워크의 상태, 전송 프로토콜 및 그 당시 사용되고 있는 방향에 기초한 메시지의 다양한 형태 및 수신된 메시지 중에서 메시지를 생성하는 경우가 있다. 이 같은 경우에, 에이전트의 정확한 모델이 사용되지 않는다면 반응 메시지를 생성하는 것은 사소한 문제가 아니다. 에이전트의 정확한 모델은 성능 시뮬레이션을 느리게 할 수 있고 전체 시스템 모델은 복잡하고 구축이 지루할 수 있다. 따라서 대부분의 NoC 시뮬레이터는 그 같은 디자인을 지원하지 않으며 홉 쌍들 사이의 단지 점대점 메시지 흐름 발생 및 시뮬레이션에 의존한다.

## *발명의 내용*

### *해결하려는 과제*

[0024] 본 발명은 네트워크 온 칩에서의 시스템 레벨 시뮬레이션을 제공한다.

### *과제의 해결 수단*

[0025] 본 발명은 완전한 SoC 거동을 흉내 냄으로써, 네트워크 온 칩(NoC) 상호 연결 아키텍처 내에서 SoC IP의 성능 시뮬레이션에 기반을 둔 여러-메시지 트랜잭션을 수행하는 것에 관련된 것이다. 본 발명의 시스템은 여러 트랜잭션에 기반하여 NoC 거동을 평가 및 검출하기 위해 시뮬레이션을 구현함(implement)을 포함하며, 여기서, 각 트랜잭션은 하나 이상의 메시지를 포함한다. 각 메시지는 또한 이후부터 홉으로도 언급될 수 있으며, 소스 에이전트 및 목적지 에이전트를 포함할 수 있고, 소스 에이전트 및 목적지 에이전트는 실제 시나리오 부품 및 IP 코어에 대응한다. 시스템 트래픽이 전형적으로 여러 에이전트에 걸쳐서 많은 메시지를 포함하고 있기 때문에, 둘 이상의 에이전트에 걸쳐 있는(span across) 그 같은 메시지들의 시퀀스(sequence)는 트랜잭션(transaction)으로 언급될 수 있고, 그 결과 여러 트랜잭션을 포함하는 시뮬레이션 환경을 야기하며 이는 메시지 흐름의 가능한 시나리오들을 입증한다. 동작에 있어서, 그 목적지 에이전트에 도착하게 되면 트랜잭션의 전임 메시지(former message)는 후임 메시지(latter message)를 생성한다. 트랜잭션이 서로 중첩될 수 있기 때문에, 본 발명의 시스템은 메시지들의 방향성 그래프(directed graph)의 형성을 허락한다.

[0026] 본 출원의 양상들은 여러 트랜잭션 시퀀스들을 사용하여 NoC에 대한 시뮬레이션 수행을 가능케 하는 방법을 포함할 수 있으며, 여기서 각 트랜잭션 시퀀스는 여러 에이전트에 걸쳐서 하나 이상의 메시지를 포함한다. 트랜잭션 시퀀스의 각 메시지는 다른 특성을 예를 들어 레이트(rate), 우선순위, 값, 데이터 크기 및 지연을 가질 수 있다. 더욱이, 각 트랜잭션은 또한 데이터 간 다른 간격들(different inter-data intervals)로 확산하고 다른 속성/특성이 있는 메시지들을 포함할 수 있다. 다른 양상에 따르면, 본 발명의 방법은 여러 엔트리(multiple entries)를 갖는 트레이스 파일(trace file)의 생성을 포함하며, 각 엔트리는 실시예에서 주어진 트랜잭션의 시작 메시지(starting message)의 표시(indication)를 대변(represent)하고 트랜잭션 시퀀스를 개시(initiate)하는데 사용된다. 여러 다른 엔트리 및/또는 조건이 또한 트랜잭션들의 개시 및 실행을 돕기 위해 트레이스 파일에 정의될 수 있다. 다양한 통계학적 그리고 확률적 모델이 그래프 이동 결정(graph traversal decision)을 판단하는데 사용될 수 있다. 트레이스 파일의 엔트리들에 기초하여, 트랜잭션 시퀀스 그래프에서 다른 경로들이 취해질 수 있고 대응하는 메시지들이 생성되고 시뮬레이션 될 수 있다. 그래프 이동 결정은 다양한 에이전트들의 거동에 의존하여 그리고/또는 SoC 아키텍처의 조건/상태에 기초하여 행해질 수 있다. 더욱이, 트레이스 파일들은 다양한 메시지 시퀀스들이 개시되는 레이트를 반영하는 대기 간격(wait interval)들을 포함할 수 있고, 에이전트 모델들 또한 다양한 시퀀스의 메시지들이 소스 에이전트에서 생성되고 목적지 에이전트에서 소비되는 레이트를 제어하는 레이트 설명(서)를 포함할 수 있다.

[0027] 본 출원의 양상들은 프로세스를 실행하기 위한 명령(instruction)들을 저장하는 컴퓨터로 판독가능한 저장 매체를 포함할 수 있다. 명령들은 여러 트랜잭션 시퀀스를 사용하는 것에 의해 NoC의 시뮬레이션의 수행을 포함하고, 여기서 각 트랜잭션 시퀀스는 여러 에이전트에 걸쳐서 하나 이상의 메시지를 포함한다. 병렬적으로 다수의 트랜잭션이 수행되면서 여러 트랜잭션이 중첩 메시지들을 가질 수 있으며, 이는 NoC의 전체 트랜잭션 시퀀스를 대변하는 트랜잭션 시퀀스 그래프를 통해 대변될 수 있다.

[0028] 본 출원의 양상들은 시스템을 포함할 수 있고, 이 시스템은 주어진 NoC에 대해 트랜잭션 시퀀스들을 수신하도록 구성된 트랜잭션 시퀀스 입력 모듈, 각 트랜잭션 시퀀스에서 메시지들에 근거하여 시퀀스 그래프를 생성하도록 구성된 트랜잭션 시퀀스 그래프 생성 모듈, 각 시퀀스에 대한 시작 메시지를 표시하고 하나 이상의 트랜잭션 시퀀스의 실행 사이의 대기 간격의 구성을 허락하는 트레이스 파일을 생성하도록 구성된 트레이스 파일 생성 모듈, 그리고 트레이스 파일에 기초하여 하나 이상의 트랜잭션 시퀀스의 시뮬레이션을 수행하고 NoC 성능 보고서를 생성하도록 구성된 트랜잭션 시뮬레이션 모듈을 포함한다.

## *발명의 효과*

[0029] 본 발명의 제안된 시스템 및 방법은 NoC 성능이 SoC에서 정확하게 특성화될 수 있는 정도로 SoC IP 코어들 또는 부품들의 거동을 정확하게 모방하기 위한 기술 및 메커니즘을 구현한다.

## *도면의 간단한 설명*

[0030] 도 1a 내지 도 1d는 각각 양방향 링, 2차원 그물망, 2차원 토러스 및 3차원 그물망 NoC 통신망구성을 도시한다.

도 2a는 종래의 2차원 그물망에서 XY 라우팅의 예를 도시한다.

도 2b는 소스 노드 및 목적지 노드 사이의 서로 다른 세 개의 라우트를 도시한다.

도 3a는 종래의 두 층 NoC 상호연결의 예를 도시한다.

도 3b는 종래의 호스트 및 여러 NoC 층 사이의 브리지 로직을 도시한다.

도 4a는 트랜잭션 시퀀스들 및 각 트랜잭션 시퀀스에 포함된 메시지(들)을 도시한다.

도 4b는 트랜잭션의 각 메시지와 연관된 가중치를 갖는 트랜잭션 시퀀스 그래프의 예를 도시한다.

도 5는 하나 이상의 메시지가 트랜잭션들에 걸쳐서 중첩하는, 복수의 메시지를 갖는 에이전트 간 여러-지점 트랜잭션을 보여주는 에이전트들 그리고/또는 에이전트 인터페이스들 세트의 예시를 도시한다.

도 6은 수신한 트랜잭션 시퀀스들 및 거기에서 생성된 트레이스 파일들에 기초하여 시뮬레이션을 수행하는 진행 다이어그램을 도시한다.

도 7은 에이전트를 위한 트레이스 파일을 생성하기 위한 진행 다이어그램을 도시한다.

도 8은 새로운 트래픽에 기초하여 새로운 트랜잭션 시퀀스들을 추가 및 실행하기 위한 진행 다이어그램을 도시한다.

도 9는 여기에 설명된 예시적인 구현들이 구현될 수 있는 컴퓨터/서버 블록 다이어그램을 도시한다.

## *발명을 실시하기 위한 구체적인 내용*

[0031] 이하의 상세한 설명은 본 출원의 도면들 및 예시적인 구현들에 대한 상세한 설명을 제공한다. 도면들 사이의 중복 구성들에 대한 설명 및 참조번호는 명확성을 위해 생략된다. 본 섹션에서 사용된 용어들은 예로서 제공된 것으로서 제한적으로 사용된 것은 아니다. 예를 들어, "자동적"이라는 용어의 사용은, 완전 자동적인 구현이거나 본 출원의 구현들을 실행하는데 통상의 지식을 가진 자의 원하는 구현에 따라서, 구현의 어떤 양상들에 대한 사용자 또는 관리자 제어를 수반하는 반-자동적인 구현을 포함한다.

[0032] SoC 응용 트래픽 프로파일을 특성화하고 벤치마킹하기 위한 NoC 상호연결 성능 시뮬레이션은 NoC 상호연결 모델에서 트래픽 생성에 대한 지원을 요구할 수 있다. NoC에 연결된 다양한 SoC 에이전트/부품에 의해 생성되고 소비된 트래픽이 정확하게 응용의 트래픽 특성을 모방하는 것이 바람직하기만, 이 목적으로 모든 SoC 에이전트를 정확하게 모델화하는 것은 시뮬레이션 레이트 측면에서 엄청나게 복잡하고 느릴 수 있다. 제안된 시스템 및 방법은 NoC 성능이 SoC에서 정확하게 특성화될 수 있는 정도로 SoC IP 코어들 또는 부품들의 거동을 정확하게 모방하기 위한 기술 및 메커니즘을 구현한다.

[0033] 예시적인 구현에 따르면, NoC 성능 특성화를 위해서, SoC의 NoC 트래픽 프로파일은 에이전트들의 인터페이스들의 다양한 쌍 사이의 메시지들의 시퀀스를 사용하여 기술될 수 있다. 전체 메시지 체인은 트랜잭션으로 언급될 수 있고, 각 메시지는 또한 홉(hop)으로 언급될 수 있다. 각 홉은 예를 들어 비트 수로 표시되는 메시지 데이터 폭 및 길이, 평균 및 피크 데이터 율 그리고 간헐성(burstiness) 레벨로 표시되는 대역폭 설명(서)(bandwidth specification), 지연 제한(latency constraint), 또는 메시지가 NoC에서 취해야할 순서 또는 라우트에 대한

다른 제한 같은 특성을 가질 수 있다. 트랜잭션에서 메시지의 특성으로 인해, 각 트랜잭션은, 다른 응용가능한 속성들 중에서, 우선순위, 레이트 같은 하나 이상의 속성과 연관될 수 있다. 여러 트랜잭션(multiple transactions)은 중첩 메시지들을 가질 수 있고 따라서 두 트랜잭션은 이동에 있어서 공통의 메시지들을 가질 수 있다. 도 4a는 도시된 바와 같이 서로 상호 통신하는 네 개의 에이전트를 가지는 시스템의 예를 도시하며(에이전트 인터페이스는 미도시), 에이전트들 사이의 메시지는 동그라미 안에 놓인 숫자로 표시된 에지(edge)로 도시되어 있다. 예를 들어, 소스 에이전트 hp1에서 목적지 에이전트 hp2로 발송된 메시지는 메시지 1로 언급될 수 있다. 비슷하게, 소스 에이전트 hp2에서 목적지 에이전트 hp1로 발송된 메시지는 메시지 2로 언급될 수 있다.

[0034] 도 4a는 또한 메시지 그래프를 통한 여러 트랜잭션을 도시한다. 이를 나타내기 위해서, 사용자는 별개의 트랜잭션으로 그래프의 각 경로를 개별적으로 제공할 수 있다. 도 4a는 총 5개의 트랜잭션을 도시하며, 각각은 에이전트 hp1과 에이전트 hp2 사이의 메시지 1로부터 출발하고, 모든 트랜잭션에서 이 홉을 공통으로 한다. 예를 들어, 트랜잭션 1은 hp1에서 hp2로 그리고 다시 hp2에서 hp1으로 전송되는 메시지를 가리킨다. 마찬가지로, 트랜잭션 4는 각각 메시지 1, 3, 5, 6, 4 및 2를 통해 hp1에서 hp2로, hp2에서 hp3로, hp3에서 hp4로, hp4에서 hp3로, hp3에서 hp2로, 그리고 hp2에서 hp1으로 전송되는 메시지를 대변한다. 각 트랜잭션의 각 메시지는 레이트로 그리고, 얼마나 자주 이 홉이 실제 SoC 시스템에서 취해지는지를 나타내는 다른 메시지 속성으로 표식이 될 수 있다. 예를 들어, (hp1에서 hp2로의) 메시지 1의 레이트는 다른 메시지들에 비해 높을 수 있는데, 왜냐하면 메시지 1이 여러 트랜잭션으로 인해 공통으로 사용되기 때문이다. 제안된 NoC 성능 시뮬레이션 메커니즘은 따라서, NoC 성능이 에이전트의 정확한 기능적인 거동을 시뮬레이션하지 않고 공정하게 특성화되는 정도로 완전한 여러-홉 트랜잭션 설명(서) 및 시뮬레이션을 허락한다.

[0035] 예시적인 일 구현에 따르면, 트랜잭션 시퀀스들의 그래프는 메시지들이 수신 또는 전송되는 에이전트들 그리고/또는 인터페이스들을 나타내도록 구성될 수 있다. 예를 들어 도 4a의 경우, 도 4b에 도시된 것 같이 트랜잭션 시퀀스 그래프가 생성될 수 있고, 이때 각 에지는 소스 에이전트에서 목적지 에이전트로 보내지는 메시지를 대변한다. 도시된 바와 같이, 위쪽 행의 노드들은 메시지 시퀀스들의 앞으로 나아가는 전송을 가리키고, 아래 행의 노드들은 뒤쪽의 에이전트들이 수신하는 메시지들을 가리킨다. 도 4b에 도시된 것 같은 트랜잭션 시퀀스 그래프는 예시적인 것으로서, SoC 아키텍처의 여러 에이전트에서의 트랜잭션 메시지 흐름의 시퀀스를 보여주기 위해 다른 그래프 도는 표현이 가능하다. 일반적으로, 그래프는 트랜잭션(들)에 수반되는 에이전트 그리고/또는 에이전트 인터페이스를 그리고 특정 메시지에 있어서 소스 에이전트 및 목적지 에이전트를 보여주는 에지들/화살표에 의해 표시되는 메시지들을 포함할 수 있다. 에이전트로 들어오는 화살표에 의해 표시된 도착하는 메시지는 그 에이전트로부터 나가는 화살표에 의해 표시된 떠나는 메시지가 될 수 있다. 그래프들은 또한 메시지가 소속된 트랜잭션을 포함하도록 그리고 묘사하도록 구성될 수 있다. 일 실시예에서, 트리(tree) 레이아웃 기반 그래프가 또한 에이전트들과 그들이 트랜잭션 하는 메시지들을 대변하도록 생성될 수 있고 이 경우 하나 이상의 트랜잭션/메시지를 위해 출발 지점으로 작용하는 에이전트는 루트 노드로 작용한다.

[0036] 예시적인 구현에 따르면, 도 4b에 도시된 바와 같이, 주어진 트랜잭션의 각 메시지의 레이트 및 메시지 속성 설명(서) 메시지/에지에 기초하여 가중치가 메시지/가장자리에 할당될 수 있다. 이 경우 메시지가 여러 트랜잭션에 존재할 경우(예를 들어 도 4a의 hp1에서 hp2에서와 같이), 모든 트랜잭션에서 메시지의 레이트가 그래프의 메시지의 순 레이트를 결정하기 위해 추가될 있고 또는 평균 혹은 최대값 같이 대안 함수가 추가될 수 있다. 트랜잭션 시퀀스 그래프의 하나 이상의 메시지에 대한 레이트 할당은, 얼마나 자주 시뮬레이션이 관심 있는 메시지 또는 홉/에지를 사용해야 하는지를 도시한다. 왜냐하면, 실재 시나리오 경험을 만들고 그에 따라 NoC 성능의 거동을 평가하기 위해서, 실재 SoC 아키텍처에서 더 자주 사용될 메시지 상에서 더 많은 시뮬레이션을 가동하는 것이 중요하기 때문이다. 예시적인 일 구현에 따르면, 일단 메시지들/에지들의 레이트들이 알려지면, 도 4b에 도시된 바와 같이 그들은 가중치로 정규화될 수 있다. 정규화를 하게 되면, 임의의 노드에서 들어오는 모든 가장자리 또는 임의의 노드로 나가나 모든 가장자리의 가중치는 총 1이 될 수 있다. 예시적인 가중치 할당이 도 4b에 도시되어 있다.

[0037] 예시적인 일 구현에 따르면, 가중치들이 트랜잭션 시퀀스 그래프에 할당되면, NoC 성능 시뮬레이터는 시뮬레이션 중에 그래프에서 취할 경로를 확률론적으로 결정하도록 이 가중치들을 사용하고 그에 따라 적절한 메시지를 생성하도록 구성된다. 예를 들어, 도 4a를 참조하면, hp2에서 hp3로의 메시지가 0.25의 가중치를 가지며 hp2에서 hp4로의 메시지가 0.5의 가중치를 가질 경우, 시뮬레이션 아키텍처는 hp2에서 hp4로의 메시지 흐름의 개수가 hp2에서 hp3로의 메시지 흐름의 개수의 2배가 되도록 트랜잭션 시퀀스들 및 메시지들을 구현할 수 있다. 다른 경우, 임의의 주어진 에이전트에서 모든 나가는 가장자리들 중에서 가중치에 기초해서 하나가 선택되도록 가중

치들이 메시지 선택에 포함될 수 있고 대응하는 메시지가 메시지의 다른 측에 의해 표시된 에이전트가 목적지인 NoC 안으로 주입된다. 일단 이 메시지가 목적지 에이전트에 도착하게 되면, 대응하는 노드의 나가는 에지들이 검사되고 다음 메시지가 생성된다. 다음 메시지가 더 이상 나가는 에지들이 없는 노드에 도착할 때, 트랜잭션은 완료된 것으로 여겨진다. 따라서 이 구현에서, 큰 가중치들(실제 SoC 시스템에서 그 같은 메시지의 사용의 레이트 또는 빈도를 반영함)을 갖는 메시지들이 더 자주 뒤따르고(is followed) 따라서 SoC 트래픽의 대역폭 특성을 정확하게 흉내를 낸다. 더욱이, 트랜잭션의 후반부에 있는 메시지들은 그 목적지 에이전트에 전달되어 당해 목적지 에이전트에 의해 소모되는 이전의 메시지로 인해 생성되기 때문에, SoC에 대한 정확한 메시지 특성화가 모방이 될 수 있다.

[0038] 예시적인 일 구현에 따르면, 정규화 이후에 노드로부터 나가는 모든 에지들의 가중치가 1이 아닌 경우에 그래프가 구성될 수 있고 이때 트랜잭션 시퀀스는 노드로부터 나가는 에지들의 가중치의 총합에 기초하여 확률론적으로 노드에서 중단되거나 계속 진행할 수 있다. 이는 어떤 트랜잭션은 어떤 에이전트에서 끝나지만 다른 트랜잭션 시퀀스는 당해 어떤 에이전트에 도착하고 다른 에이전트들로 계속할 때 발생한다. 예를 들어, 동일한 레이트의 두 트랜잭션 hp1=>hp2 와, hp1=>hp2=>hp3 이 있을 경우, 에이전트 hp2에서 도착하는 메시지는 끝이 나고 제1 트랜잭션을 형성하고 또는 계속 진행하여 제2 트랜잭션을 형성할 수 있다. 이 경우, hp2에서 나가는 에지는 가중치 0.5를 가질 것이며 이는 동일한 확률로 트랜잭션이 hp2에서 끝나거나 또는 hp3로 계속 진행할 것이라는 것을 가리킨다.

[0039] 예시적인 일 구현에 따르면, 본 발명의 시뮬레이터는 SoC 시스템에 대해서 정의되었었고 또는 정의된 메시지 파라미터들과 속성들을 예를 들어 크기, 값, 레이트, 데이터 간 간격 및 소스-목적지 정보를 포함하는 모든 트랜잭션 시퀀스들에 접근할 수 있어, 일단 개시되면, 트랜잭션 시퀀스들은 자동으로 완성이 된다. 하지만, 각 트랜잭션이 메시지를 NoC에 주입하기 위해 트랜잭션의 제1 메시지로 개시될 필요가 있기 때문에, 하나 이상의 트랜잭션의 제1 메시지의 세부사항 및 속성을 상세하게 하는 트레이스 파일이 생성될 수 있다. 트레이스 파일들은 전체 트랜잭션의 제1 메시지를 서술하도록 구성될 수 있다. 몇몇 트랜잭션들이 시작하는 각 에이전트의 인터페이스에 대해 트레이스 파일들이 사용될 수 있다. 트레이스 파일은, 에이전트의 전송 거동을 포착하기 위해 대기 간격으로 서로 떨어진(spaced apart) 메시지들의 시퀀스 형태로 이 에이전트에 의해 개시되는 트랜잭션들의 제1 메시지 정보를 포함할 수 있다. 호스트 hp1에 대한 단순한 트레이스 파일의 예가 아래에 설명된다. 제1 워드는 메시지의 목적지를 가리키고 그 뒤에 메시지의 특성이 뒤따른다. 메시지들은 대기 간격들에 의해 서로 떨어진다.

[0040] *hp2 [including properties and attributes of the first message]*

[0041] *wait 10*

[0042] *hp3 [including properties and attributes of the first message]*

[0043] *wait 20*

[0044] *hp2 [including properties and attributes of the first message]*

[0045] *hp4 [including properties and attributes of the first message]*

[0046] *wait 20*

[0047] 상술한 hp1을 위한 트레이스 파일은 hp2로의 두 개의 트랜잭션, hp3으로의 하나의 트랜잭션 그리고 hp4로의 하나의 트랜잭션을 나타낸다. hp2로의 두 개의 트랜잭션은 동일한 트랜잭션이거나 hp2로의 동일한 메시지를 갖는 다른 트랜잭션들일 수 있다. 더욱이, 상술한 트레이스 파일은 hp2로의 첫 번째 트랜잭션과 hp3으로의 트랜잭션 사이에 10 유닛의 대기 시간(wait time)을 나타내고, hp2로의 두 번째 트랜잭션과 hp4로의 트랜잭션 사이에는 대기 시간이 없다는 것을 나타낸다. 주어진 트랜잭션에서 제1 메시지의 특성/속성 및 대시 시간은 모니터 되는 NoC 상호연결의 원하는 거동에 기초하여 항상 수정될 수 있다는 것을 이해해야 한다. 더욱이, 트랜잭션의 제1 메시지 및 그 이후의 메시지들의 특성이 트레이스 파일에 포함되는 것을 피하기 위해 사전에 정의될 수 있다. 더욱이, 아래에서 더 상세히 설명되는 것 같이, 트랜잭션 시퀀스의 단지 제1 메시지 대신에, 트레이스 파일은 또한 그 레이트/값/크기/다른 속성과 더불어 모든 메시지 시퀀스를 포함하도록 구성될 수 있다.

[0048] 예시적인 다른 구현에서, 다른 제1 메시지들로 여러 트랜잭션이 시작할 때, 어떠한 트랜잭션의 제1 메시지도 무작위로 사용되어 트랜잭션을 개시할 수 있다. 다른 실시예에서, 심지어 모든 메시지 시퀀스들의 실행이 수행되도록 각 트랜잭션이 균등하게 선택될 수 있다. 또는, 트랜잭션들은 또한 특성들과 결부될 수 있는 그 특성은 상

대적으로 낮은 우선순위를 갖는 트랜잭션들에 비해서 높은 우선순위를 갖는 트랜잭션들이 더 자주 실행되도록 각 메시지의 가중치로부터 계산될 수 있다. 임의의 다른 알고리즘이 또한 시퀀스들이 공통의 메시지를 가질 때 트랜잭션 시퀀스들을 운용 및 선택하기 위해 사용될 수 있다.

[0049] 도 5는 다수의 메시지를 갖는 에이전트 간 여러-지점 트랜잭션들을 도시하며, 여기서 하나 이상의 메시지는 트랜잭션들에서 중첩된다. 도 5의 예는 에이전트 A-I를 도시하며, 서로 동작상 결합할 수 있고 또는 결합하지 않을 수 있다. 이 예는 또한 7개의 트랜잭션 즉 T1~T7을 도시한다. 예를 들어, T1은 에이전트 A에서 에이전트 B를 거쳐 에이전트 H로 이동한다. 마찬가지로, 트랜잭션 시퀀스 T5는 에이전트 B에서 에이전트 C 및 에이전트 D를 거쳐 에이전트 E로 이동한다. 전술한 트랜잭션 시퀀스들에 기초하여, 트레이스 파일은 따라서 하나 이상의 트랜잭션이 개시되는 에이전트 A, B, C 및 D에 대해서 생성될 수 있다.

[0050] 전술한 바와 같이, 각 트랜잭션의 각 메시지는 가중치(정규화된 레이트)와 결부되고, 그것은 실제 SoC 구현에서 관심이 있는 메시지가 사용되는 빈도(또는 레이트)를 강조할 수 있다. 가중치는 또한 여러 옵션이 주어진 목적지 노드에 존재할 경우 활용될 트랜잭션과 라우트를 정의한다. 예를 들어 시뮬레이션 중에 에이전트 A에서 에이전트 B로의 메시지가 이동한다면, 시뮬레이터를 위해 다음의 가능한 옵션은 H로 가는 것(트랜잭션 T1), I로 가는 것(트랜잭션 T3), 또는 C로 가는 것(트랜잭션 T3 또는 T4)을 포함한다. 이 같은 경우, 모든 트랜잭션은 무작위로 또는 확률론적으로 선택되어 선택에서 균등을 유지하거나 또는 각 메시지(B에서 H, B에서 I, B에서 C)의 가중치는 계산되고 만약 B에서 C로의 메시지의 가중치가 0.5이고, B에서 I로의 메시지의 가중치가 0.25이고 B에서 H로의 메시지의 가중치가 0.25라면 B에서 네 개의 메시지에서 두 개는 0.5로 가고, 하나는 I로 가고 하나는 H로 가도록 가장 높은 레이트를 갖는 메시지가 선택되고 따라서 메시지의 가중치에 근거하여 비율(ratio)을 유지한다. 비슷하게, 에이전트 C에서 에이전트 D로 가는 공통의 메시지의 경우, 다음 목적지 에이전트로서 E(T5)를 선택할지 F(T6)을 선택할지 G(T7)을 선택할지 D(T4) 자체에서 중단할지에 대한 선택이 이루어질 수 있다. 이 같은 각 옵션의 가중치가 동일할 경우, 임의의 무작위 메시지 시퀀스가 선택되거나 트랜잭션이 실제로 에이전트 D 자체에서 중단되어 트랜잭션 4가 완료될 수 있다. 하지만, 이는 예시적인 구현으로서 임의의 다른 모드 및 메커니즘이 적용되어 트랜잭션 및 앞으로 나아가는 메시지(onward message)를 선택할 수 있다.

[0051] 예시적인 일 구현에 따르면, 제안된 시뮬레이터 디자인은 매 에이전트에서 트레이스 파일을 읽고, 대응하는 목적지 에이전트들로 향하는 제1 메시지를 생성하고, 이 메시지들을 NoC 상호연결에 주입할 수 있다. 제1 메시지는 다음 목적지 라우터를 표시할 뿐만 아니라 그것이 관련된 트랜잭션 시퀀스를 확인한다. 시뮬레이터는 또한 각 홉 메시지가 그 목적지에 도달할 때 자동으로 남아있는 홉 메시지들을 판단할 수 있다. 메시지에서 일치하는(match) 여러 트랜잭션이 있을 경우(즉 에이전트 인터페이스에서 만나고 다음 메시지가 동일할 경우), 트랜잭션의 메시지들 중 하나가 트랜잭션들 또는 트랜잭션들의 메시지들의 레이트에 기초하여 확률론적으로 선택될 수 있다. 예를 들어, 도 4a에서, 메시지 1이 트레이스 파일에 나타날 때, 그것은 모든 다섯 개의 트랜잭션에 맞을 것이다. 다섯 개의 트랜잭션의 레이트가 0.3, 0.3, 0.1, 0.2 및 0.2인 것으로 가정할 경우, 트레이스 파일에서 메시지 1은 이와 같은 확률로 다섯 개의 트랜잭션 중 하나로 전개된다. 트랜잭션 3이 무작위로 선택된다면, hp2가 목적지가 되는 메시지 1이 hp1에서 생성될 것이다. hp2에 이 메시지가 도착할 때, 이 메시지는 목적지가 h3이 되는 메시지 3을 hp2에서 생성할 것이고, 이 같은 방식으로 hp1이 목적지가 되는 마지막 메시지 1이 hp2에서 생성될 때까지 메시지가 생성될 것이다.

[0052] 다른 예시적인 구현에 따르면, 사용자에 대한 추가의 제어를 위해서, 트레이스 파일 엔트리들은 제1 메시지 설명(서)의 일부로서 비트 수와 같은 몇몇 메시지 속성을 포함할 수 있다. 제공된다면, 그 속성이 트레이스 파일의 상세하게 기술된 속성과 일치하는 트랜잭션들만이 트래픽 생성에서 고려된다.

[0053] 도 6은 예시적인 구현에 따른 시뮬레이션을 수행하기 위한 흐름을 보여주는 블록 다이어그램(600)을 도시한다. 블록(601)은 NoC 상호연결의 성능을 다루고 특성화하도록 구성된 트랜잭션 시퀀스들의 세트를 수신함을 도시하며 이로부터 시뮬레이션이 수행된다. 각 트랜잭션 시퀀스는 다른 또는 동일한 특성/속성을 가지는 하나 이상의 메시지를 포함할 수 있으며, 하나 이상의 트랜잭션의 메시지는 동일한 소스 에이전트 및 목적지 에이전트를 가질 수 있다. 블록(602)은 여러 에이전트와 그들 사이에서 그들이 처리하는 메시지 사이의 상호연결을 도시하는 트랜잭션 시퀀스 그래프의 생성을 도시한다. 이 시퀀스 그래프는 따라서 메시지들이 트랜잭션들에서 공통인 정도를 주기적으로 대변하고 표시할 수 있다. 그래프는 각 메시지의 가중치(정규화된 레이트)를 더 표시할 수 있는바, 제1 메시지의 가중치가 높을수록 시뮬레이션 중에 제1 메시지의 사용 확률이 높다. 가중치는 또한 시뮬레이션 가동 중에 관심이 있는 메시지 그리고/또는 그것이 관계된 트랜잭션이 포함될 횟수를 표시할 수 있다.

[0054] 블록(603)은 하나 이상의 트레이스 파일을 생성함을 도시하는바, 각 트레이스 파일은 트랜잭션 시퀀스 그래프의

생성에서 동시에 혹은 순차적으로 생성될 수 있다. 트랜잭션 시퀀스 그래프는 단지 설명의 목적으로 제시한 것이며 생략될 수 있다. 트레이스 파일은 한편 하나 이상의 에이전트에 대해 생성될 수 있고, 그 같은 에이전트의 각각을 위해서 관련된 에이전트에 의해 개시된 트랜잭션들의 제1 메시지를 표시하는 것을 도울 수 있다. 트레이스 파일들은 또한 다른 방식으로 구성될 수 있다. 예를 들어 트레이스 파일들은 시뮬레이션 중에 실행되어야 하는 트랜잭션들을 정의하는 사전조건을 설정하는데 사용될 수 있다. 트랜잭션의 단지 제1 메시지만을 포함시키는 대신에, 트레이스 파일들은 각 메시지 그리고/또는 트랜잭션의 특성 및 속성을 정의함과 더불어 트랜잭션의 다른 메시지들을 포함할 수 있다. 트레이스 파일들은 또한 수동으로 편집될 수 있고, 여러 트랜잭션의 실행 사이에 지연을 가질 수 있다. 블록(604)에서, 시뮬레이션이 여러 에이전트의 트레이스 파일들의 엔트리들에 기초하여 하나 이상의 트랜잭션을 사용함으로써 수행된다. 시뮬레이션은 NoC 상호연결의 특징적인 성능을 나타낼 수 있는 성능 보고서를 생성하는데 일조를 할 수 있다.

[0055] 제안된 시뮬레이터 디자인은 또한 트레이스 파일 생성 프로세스를 쉽게 하기 위해서 SoC의 트래픽 트랜잭션들의 설명(서)에 기초하여 자동으로 트레이스 파일들을 생성할 수 있다. 이어 사용자들은 필요에 따라 SoC 에이전트 거동을 더 정확하게 반영하기 위해 이 파일들을 편집할 수 있다. 갱신된 트레이스 파일들은 새로운 시뮬레이션이 시작할 때마다 매번 다시 로딩 될 수 있다. 생성된 트레이스 파일들은 설명(서)에 상세하게 설명된 트랜잭션 게이트를 설명할 수 있다. 간헐성 특성, 데이터 비트 수 등과 같은 추가 정보뿐만 아니라 트래픽 설명(서)에서 평균 또는 피크 레이트가 사용될 수 있다. 트레이스 파일을 생성하는 데 사용되는 알고리즘의 예시적인 구현이 아래에서 설명된다.

[0056] 에이전트 I로부터 N개의 트랜잭션이 시작한다고 가정한다. 즉, 에이전트 I가 N 트랜잭션들의 제1 메시지를 개시한다. N번째($N^{th}$) 트랜잭션의 레이트를 사이클당 메시지들 유닛으로 $R_n$ 이라 한다. 레이트들에 기초하여 제1 홉 메시지들을 포함하는 리스트가 구성될 수 있다. 이 리스트에서, 트랜잭션 I의 제1 메시지는 $R_1$ x C 회 만큼 리스트 될 수 있고, 제2 메시지는 $R_2$x C 회 만큼 리스트 될 수 있고, 이 같은 방식으로 메시지들이 리스트 될 수 있으며, 높은 레이트를 갖는 트랜잭션들이 더 자주 존재하는 상황으로 이어진다. 예를 들어, 에이전트 I가 서로 다른 다섯 개의 목적지 에이전트 이름하여 D1, D2, D3, D4 및 D5로 5개의 트랜잭션을 개시한다고 하자. 이 구현에서, D1을 사용한 트랜잭션 시퀀스가 가장 높은 레이트를 가지며 D4를 사용한 트랜잭션 시퀀스가 가장 낮은 레이트를 가진다고 할 경우에, D1을 사용한 트랜잭션은 가장 높은 반복될 확률을 가질 것이며, 따라서 C와 곱해질 때 D1을 사용한 트랜잭션이 사용될 횟수가 나온다. 알고리즘의 구현을 위해서, C는 전역 상수로서 N 정수들의 최소 공배수일 수 있고 여기서 i번째($i^{th}$) 정수는 (int)1/Ri, 1/Ri의 정수 부분이다. 최소 공배수는 부동 소수점(floating point) R1를 다양한 홉들의 예(instance)의 수의 정수 카운트로 전환할 때의 반올림 오차(rounding error)를 최소화할 것이다. 그 뒤에, 다양한 트랜잭션들의 레이트를 반영하기 위해 다양한 트랜잭션들의 제1 홉들의 리스트가 간격(delay interval)을 삽입하는 것에 의해 정규화될 수 있다. 지연 간격을 결정하기 위해서, 모든 트랜잭션의 레이트들이 $R = \sum Ri$ 로 함께 가산될 수 있다. R이 1보다 클 경우, 지연 간격은 필요치 않다. 하지만, R이 1보다 작을 경우, 리스트의 메시지들은 1/R 지연 사이클에 의해 떨어질 필요가 있다. 반올림 오차를 줄이기 위해서, M/R 단일 사이클 지연 간격이 리스트에 삽입될 수 있고, 여기서 M은 리스트의 메시지 엔트리의 총 개수이다. 지연 간격은 동일한 개수의 메시지 엔트리로 떨어진 리스트에 균일하게 삽입될 수 있고 또는 무작위일 수 있다.

[0057] 도 7은 트레이스 파일의 생성을 위한 예시적인 흐름 다이어그램(700)을 도시한다. 701에서 트레이스 파일의 창조/생성 대상이 되는 에이전트가 확인된다. 702에서, 확인된 에이전트로부터 시작하는 트랜잭션들이 검출된다. 703에서, 검출된 트랜잭션 각각의 레이트가, 트랜잭션이 반복될 횟수를 평가하기 위해서, 탐색되며(retrieved), 여기서 레이트는 실제 SoC 구현 중에 트랜잭션 시퀀스를 선택하는 확률을 의미한다. 704에서, 각 트랜잭션 시퀀스가 실행되는 횟수(number of times)가, 각 트랜잭션이 수행되는 횟수 계산을 촉진하는 곱셈 인자(multiplicaton factor)와 각 트랜잭션의 레이트에 기초하여 결정된다. 705에서, 704에서 계산된 각 트랜잭션이 수행되는 횟수에 기초하여, 시뮬레이션이 레이트 또는 발생 확률에 기초하여 각 트랜잭션을 가동할 수 있도록, 각 트랜잭션을 위한 여러 엔트리가 트레이스 파일에 만들어진다. 706에서, 트랜잭션들의 레이트들 총합이 계산된다. 707에서, 트랜잭션들의 레이트들 총합이 1보다 큰지 여부가 결정된다. 708에서 트랜잭션들의 레이트들 총합이 1(N)보다 크지 않으면 총합이 적어도 1과 같도록 적절한 지연이 트랜잭션들 사이에 도입된다. 그렇지 않으면, 트랜잭션들의 레이트들 총합이 1(N)이거나 1보다 크면, 지연은 도입되지 않는다.

[0058] 각 트랜잭션의 제1 메시지의 표시와 함께, 트레이스 파일이 제1 메시지의 특성 그리고/또는 속성을 표시할 수

있고, 필요에 따라 이후 메시지들을 표시할 수 있다. 전술한 트레이스 파일 생성 모드는 예시적인 구현이며, 다른 포맷 및 모드가 트레이스 파일의 생성 또는 시뮬레이션의 개시를 위해 사용될 수 있다.

[0059] 대안적인 시뮬레이터 디자인은 각 트랜잭션을 위해 여러 메시지를 서술하는 트레이스 파일들을 활용할 수 있다. 이 경우, 여러 메시지는 시스템에서 트랜잭션의 접두사(prefix)를 형성할 것이고, 정의된 접두사와 부합(match)하는 트랜잭션들만이 시뮬레이션을 위해 사용될 것이다. 도 4a에 도시된 예를 고려한다. hp1를 위한 트레이스 파일이 hp2의 엔트리를 포함한다면, 다섯 개 트랜잭션 모두는 부합할 것이고 전술한 확률/레이트/가중치 기반 모델에 기초한 시뮬레이션 중에 어떠한 트랜잭션도 선택될 수 있다. 하지만, hp1을 위한 트레이스 파일이 "hp2 및 hp3"의 엔트리를 포함할 경우, 마지막 네 개의 트랜잭션만이 이 엔트리에 부합할 것이며, 이 트랜잭션들 중 임의의 트랜잭션이 무작위로 또는 트랜잭션 그리고/또는 메시지의 가중치에 기초하여 선택될 것이다. 그래프에서 어떠한 경로가 취해질지를 정확하게 결정하기 위해서, 확률론적 모델이 그 같은 디자인에서 조정될 수 있다. 이 디자인에서, 완전한 트랜잭션 또는 트레이스 파일에서 트랜잭션들의 접두사가 서술될 수 있으며, 단지 하나의 트랜잭션이 부합할 것이므로 시뮬레이션은 결정론적일 수 있다. 더욱이, 중간 에이전트들이 또한 트레이스 파일에서 언급될 수 있는바, 시뮬레이션 프로세스 중에 그 같은 에이전트들을 통해 이동하는 트랜잭션들이 수행되도록 중간 에이전트들이 언급될 수 있다. 요약하면, 트레이스 파일은 하나 이상의 트랜잭션의 실행을 위한 임의의 주어진 기준을 설정할 수 있고, 또한 하나 이상의 트랜잭션 사이의 지연과 함께 각 트랜잭션이 수행되는 횟수를 정의할 수 있다.

[0060] 예시적인 시뮬레이터 디자인은 사용자가 NoC 디자인의 기초가 되는 원래 설명(서)와는 다른 트래픽 트랜잭션 설명(서)에 기초한 트레이스 파일들을 생성하는 것을 가능하게 할 수 있다. 이것은 표준 트래픽과 다른 트래픽을 위한 NoC의 시뮬레이션을 수행하는데 유용하다. 새로운 트랜잭션 설명(서)의 정의를 촉진하기 위해 원래 설명(서)는 텍스트 파일로 쓰일 수 있고, 이 텍스트 파일은 사용자가 설명(서)에 수정을 하기 위해 편집될 수 있다. 사용자는 홉의 다른 파라미터들뿐만 아니라 다양한 트랜잭션의 다양한 메시지의 레이트들을 수정할 수 있다. 사용자는 하지만 디자인된 NoC에서 지원되지 않을 수 있기 때문에, 새로운 트랜잭션을 추가하는 것이 가능하지 않을 수 있다. 편집된 트랜잭션 설명(서)는 로딩 될 수 있고 트레이트 파일들은 전술한 알고리즘을 사용하여 이 설명(서)에 기초하여 자동으로 생성될 수 있다. 새로운 트레이스 파일이 일단 생성되면, 이 트레이스 파일은 새로운 트래픽 설명(서)의 거동을 반영할 수 있고 NoC 성능 시뮬레이션을 위한 시뮬레이터에 의해 사용될 수 있다. 사용자는 여전히 트래픽 자극(traffic stimulus)에서 추가의 고객화(customization)를 위해 트레이스 파일을 편집할 수 있고 또는 원래 시뮬레이션을 위해 원래 트래픽 설명(서)로 되돌아 갈 수 있다. 이 같은 시뮬레이터의 예시적인 구현이 도 8에 나타나 있다.

[0061] 도 8은 새로운 트래픽 및 가능한 트랜잭션 시퀀스들이 수동으로 사용자에 의해 추가되거나 자동으로 추가되는 흐름 다이어그램(800)을 도시한다. 801에서, 새로운 트래픽이 원래 트래픽 트랜잭션 프로파일에 추가되며, 여기서 새로운 트래픽은 장차 SoC 통신의 일부를 형성할 수 있는 에이전트들/부품들/IP 코어들 사이의 새로운 트랜잭션들을 나타낼 수 있다. 802에서, 원래 트래픽 트랜잭션 프로파일이 새로운 트래픽 정보에 기초하여 갱신된다. 803에서, 갱신된 트래픽 트랜잭션 프로파일에 기초하여, 새로운 트랜잭션 시퀀스들이 생성된다. 804에서, 트랜잭션 시퀀스들의 특성이 수정될 수 있고, 여기서 이 수정은 각 트랜잭션 시퀀스의 메시지들의 속성들 중에서도 레이트, 크기, 값, 간격 같은 속성의 변경을 포함한다. 트랜잭션 시퀀스들 및 거기의 메시지들은 또한 시뮬레이션 구동을 고객화 하기 위해 어느 때라도 사용자에 의해 수정 그리고/또는 변경될 수 있다. 805에서, 새로운 트래픽 시퀀스들 및 그 속성들에 기초하여 트래픽 파일이 생성되고 또는 재-생성된다. 이 같은 트래픽 파일들은 또한 트랜잭션들 사이의 지연, 각 트랜잭션/메시지의 파라미터들 같은 인자들을 변경시키기 위해 미리 정의된 기준에 근거하여 또는 수동으로 개정될 수 있다. 806에서, 갱신된 혹은 새로 생성된 트레이트 파일은 이어서 엔트리들에 기초하여 트랜잭션 시퀀스들의 실행을 개시하기 위해 그리고 시뮬레이션을 수행하기 위해 사용될 수 있다. 807에서, 성능 보고서가 NoC 상호연결의 성능을 분석하는데 일조를 하기 위해서 각 시뮬레이션 구동 혹은 조합에 의한 시뮬레이션 구동들 이후에 생성될 수 있다.

[0062] 성능 보고서는 평균 지연, 시스템의 처리량, 쌍 단위 지연(pair-wise latency), 다양한 소스 및 목적지 쌍 사이의 처리량, 지연 분포, 최대값 및 최소값 등을 포함할 수 있다. 성능 보고서는 또한 다양한 소스 및 목적지 쌍에 의해 수신된 처리량의 비율을 포함할 수 있으며 이는 시스템에서 다양한 호스트에 의해 수신된 상대적인 대역폭을 가리킨다. 하지만, 예시적인 구현은 여기에 한정되는 것은 아니며 본 예시적인 구현의 발명 개념으로부터 벗어나지 않고 다른 성능 보고 정보로 대체되거나 다른 성능 보고 정보가 추가될 수 있다.

[0063] 도 9는 예시적인 구현이 구현될 수 있는 예시적인 컴퓨터 시스템(900)을 도시한다. 컴퓨터 시스템(900)은 I/O 유닛(935), 저장수단(storage)(960) 및 통상의 기술자에게 잘 알려진 하나 이상의 유닛을 수행하도록 동작하는

프로세서(910)를 포함할 수 있는 서버(905)를 포함한다. 용어 "컴퓨터 판독가능한 매체"는 실행을 위해 프로세서(910)에 인스트럭션들(instructions)을 제공하는데 참여하는 임의의 매체를 가리키며 이것은 여기에 한정되는 것은 아니며 광 디스크, 자기 디스크, 읽기전용메모리, 랜덤액세스 메모리, 고체 상태 장치 및 드라이브, 또는 다른 형태의 전자적 정보를 저장하기에 적절한 유형 매체, 또는 반송파를 포함할 수 있는 컴퓨터 판독가능한 신호 매체 같은 컴퓨터 판독가능한 저장 매체 형태일 수 있다. I/O 유닛은 키보드, 마우스, 터치 장치 또는 음성 명령 같은 입력 장치를 사용할 수 있는 사용자 인터페이스(940) 및 조작자 인터페이스(945)로부터의 입력을 처리한다.

[0064] 서버(905)는 또한 외부 저장수단(950)에 연결될 있고, 외부 저장수단(950)은 제거가능한 저장수단 예를 들어 휴대용 하드 디스크, 광 매체(CD 또는 DVD), 디스크 매체 또는 컴퓨터가 실행 코드를 판독하는 임의 다른 매체를 포함할 수 있다. 서버는 또한 사용자로부터의 추가 정보를 요청할 뿐만 아니라 사용자에게 출력 데이터 및 다른 정보를 표시하는 디스플레이 같은 출력 장치(955)에 연결될 수 있다. 서버(905)로부터 사용자 인터페이스(940), 조작자 인터페이스(945), 외부 저장수단(950), 그리고 출력 장치(955)로의 연결은 무선 프로토콜을 통해서 예를 들어 802.11 표준, 블루투스®, 또는 셀룰러 프로토콜을 통해서 또는 물리적인 전송 매체를 통해서 예를 들어 케이블 또는 광섬유를 통해서 이루어질 수 있다. 출력 장치(955)는 따라서 사용자와 상호작용하기 위한 입력 장치로서도 작용할 수 있다.

[0065] 프로세서(910)는 하나 이상의 모듈을 수행할 수 있다. 트랜잭션 시퀀스 입력 모듈(911)은 주어진 NoC를 위해 트랜잭션 시퀀스들을 수신하도록 구성될 수 있다. 각 트랜잭션 시퀀스는 소스 에이전트에서 목적지 에이전트로 가도록 구성된 하나 이상의 메시지를 포함할 수 있다. 트랜잭션 시퀀스 그래프 생성 모듈(912)은 각 트랜잭션 시퀀스에 관여하는 에이전트들 및 메시지들을 기초로 시퀀스 그래프를 생성하도록 구성될 수 있다. 트레이스 파일 생성 모듈(913)은, 다른 메시지 속성들 및 각 메시지 사이의 대기 시간의 구성을 허락함과 함께 하나 이상의 트랜잭션 시퀀스를 위한 제1 메시지를 위한 에이전트를 나타내거나 시작 메시지를 나타내는 트레이스 파일을 생성하도록 구성될 수 있다. 트랜잭션 시뮬레이션 모듈(914)은 생성된 트레이스 파일에 기초하여 하나 이상의 트랜잭션 시퀀스의 시뮬레이션을 실행하도록 구성될 수 있다.

[0066] 트랜잭션 시퀀스 입력 모듈(911), 트랜잭션 시퀀스 그래프 생성 모듈(912), 트레이스 파일 생성 모듈(913), 및 트랜잭션 시뮬레이션 모듈(914)은 원하는 구현에 따라 다양한 방식으로 서로 상호작용할 수 있다. 예를 들어, 트랜잭션 시퀀스 그래프 생성 모듈(912)은 입력 모듈(911)로부터 트랜잭션 시퀀스들을 취하고, 각 트랜잭션 시퀀스를 위해, 트랜잭션 시퀀스의 각 메시지에 관여하는 에이전트들을 확인하고, 그 메시지들과 함께 모든 에이전트를 나타내는 시퀀스 그래프를 생성한다. 모듈(912)은 또한 가중치를 트랜잭션의 각 메시지와 연결하도록 구성될 수 있다. 더욱이, 트레이스 파일 생성 모듈(913)은 입력 모듈(911)로부터 입력을 취하고 포함이 될 각 트랜잭션의 제1 메시지들의 리스트 생성을 포함하기 위해 각 에이전트를 위해 트레이스 파일을 생성하는 것을 도울 수 있다. 트레이스 파일 생성 모듈(913)은 또한, 트랜잭션이 수행될 수 있는 레이트의 구성을 가능하게 하도록, 파일에 대기 시간을 포함시킴과 아울러 트레이스 파일에 각 메시지의 특성의 추가를 위해 트랜잭션 시퀀스 그래프 생성 모듈(912)과 동작에 있어 결합할 수 있다.

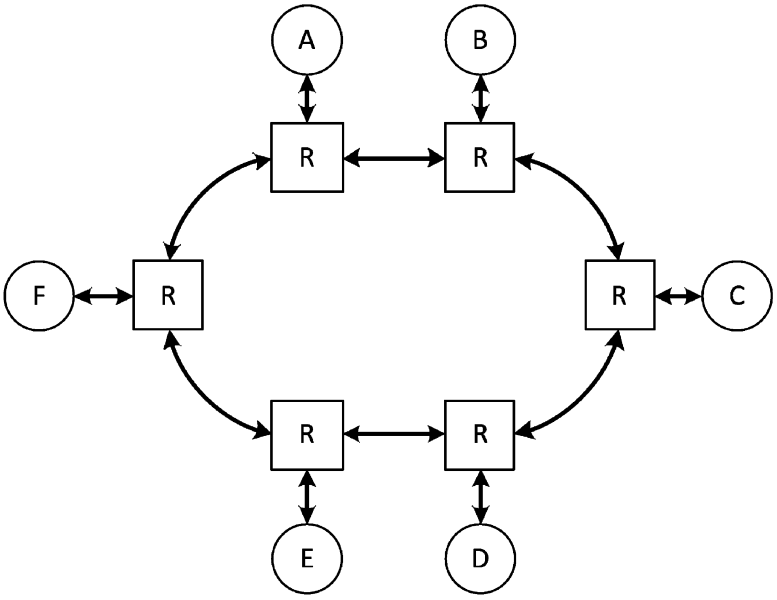
[0067] 더욱이, 상세한 설명의 몇몇 부분이 알고리즘과, 컴퓨터 내의 동작의 심벌 표현 관점으로 제시되었다. 이 알고리즘 서술 및 심벌 표현은 통상의 기술자에게 혁신의 정수를 가장 효과적으로 전달하기 위해 데이터 프로세싱 기술 분야에서 통상의 기술자에 의해 사용되는 수단이다. 알고리즘은 원하는 최종 상태 또는 결과로 이어지는 일련의 정의된 단계들이다. 예시적인 구현들에서, 수행된 단계들은 유형의 결과를 달성하기 위해 유형의 양(tangible quantity)의 물리적인 조작을 필요로 한다.

[0068] 더욱이, 본 출원의 다른 구현들은 여기에 개시된 예시적인 구현들의 실행과 명세서에 대한 고찰로부터 통상의 기술자에게 자명할 것이다. 설명된 예시적인 구현들의 다양한 구성들 그리고/또는 양상들은 단독으로 또는 임의의 조합으로 사용될 수 있다. 명세서 및 예시들은 예시적인 것으로 간주 되어야 하고 본 출원의 진정한 범위 및 정신은 다음의 특허청구범위 청구항들에 의해 표현된 것이다.

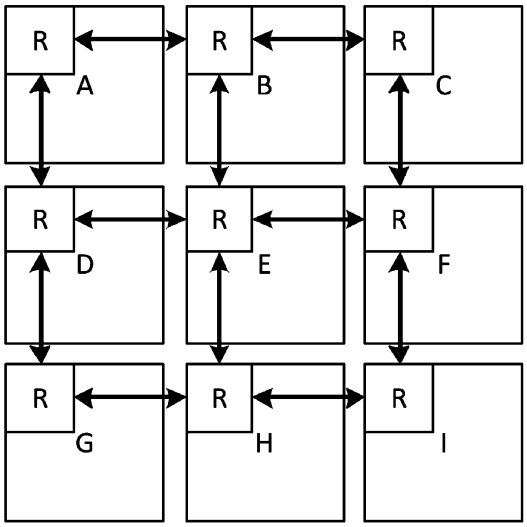
[0069]

## 도면

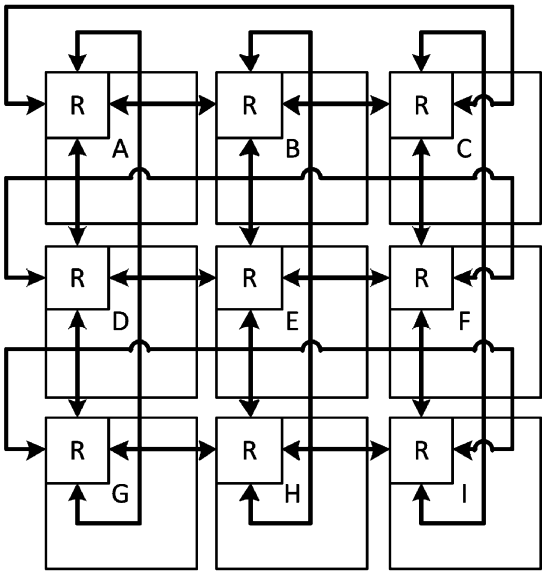
### 도면1a



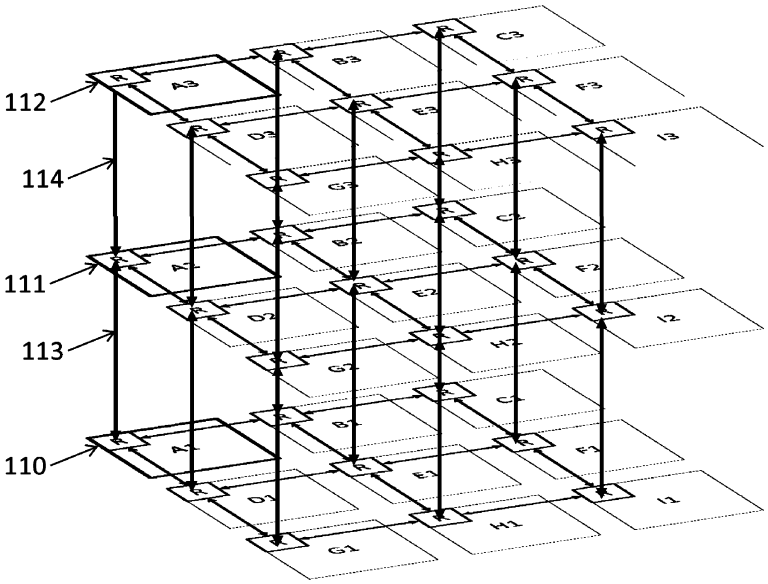

### 도면1b



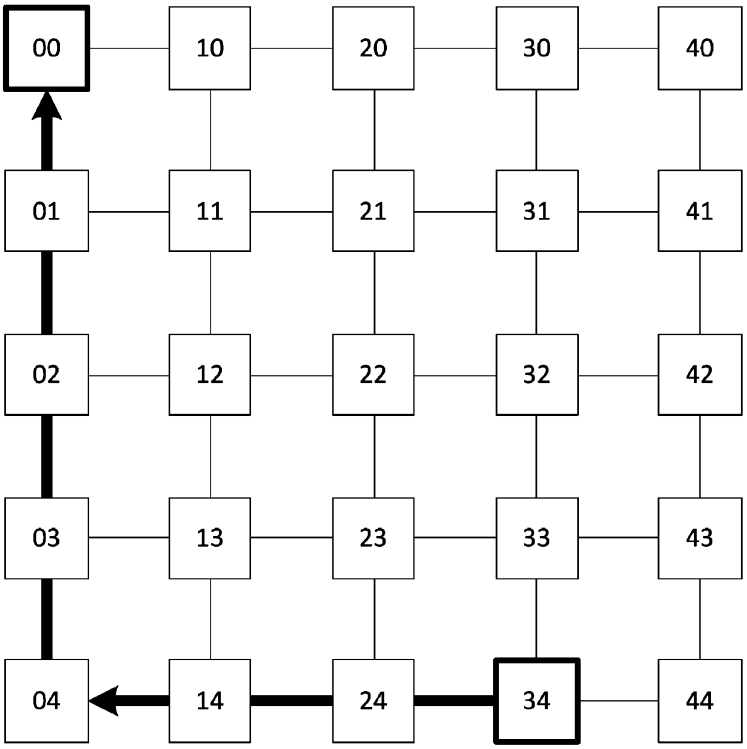
도면1c



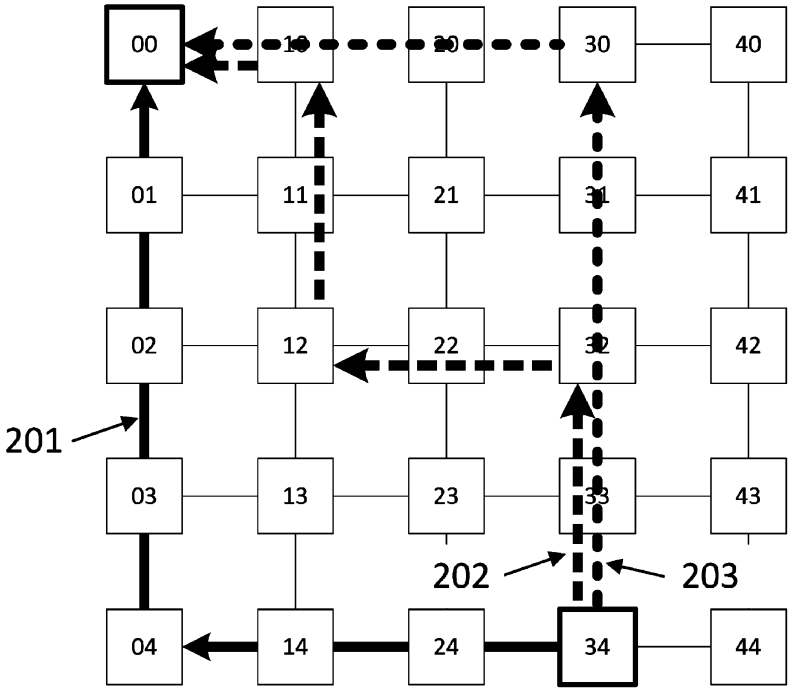

도면1d



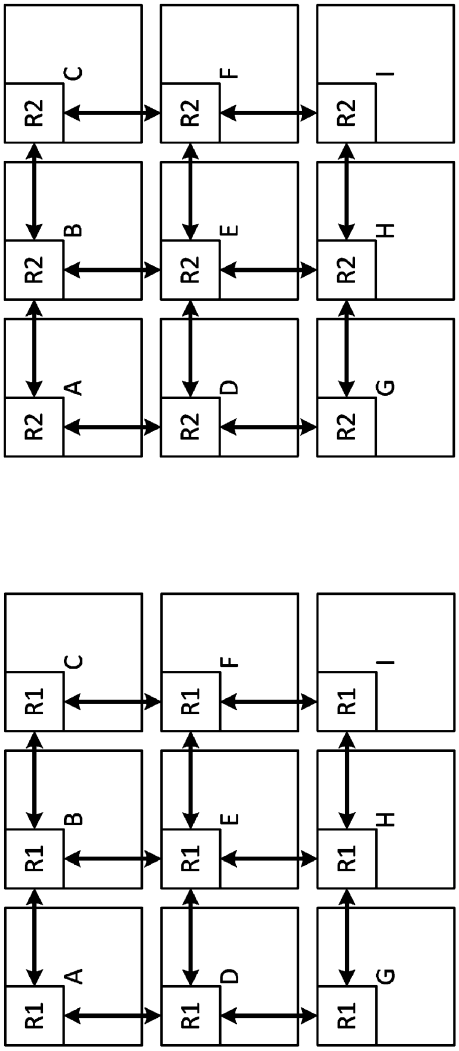
도면2a


00
10
20
30
40
01
11
21
31
41
02
12
22
32
42
03
13
23
33
43
04
14
24
34
44

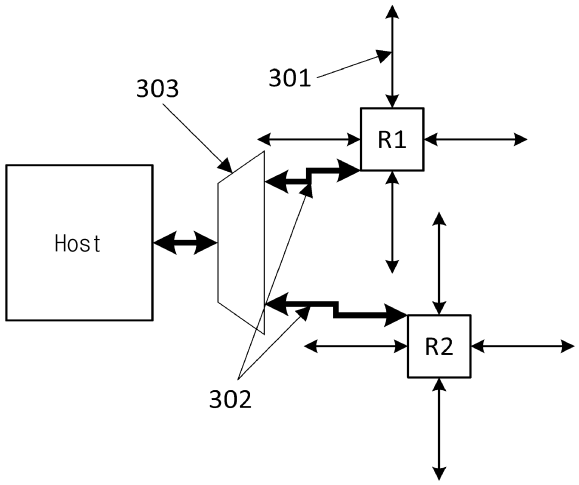
도면2b


00
30
40
01
11
21
41
02
12
22
42
201
03
13
23
43
202
203
04
14
24
34
44

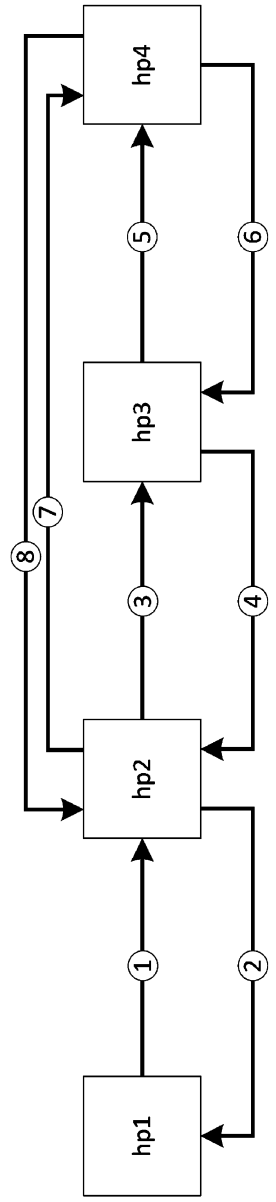
도면3a




도면3b

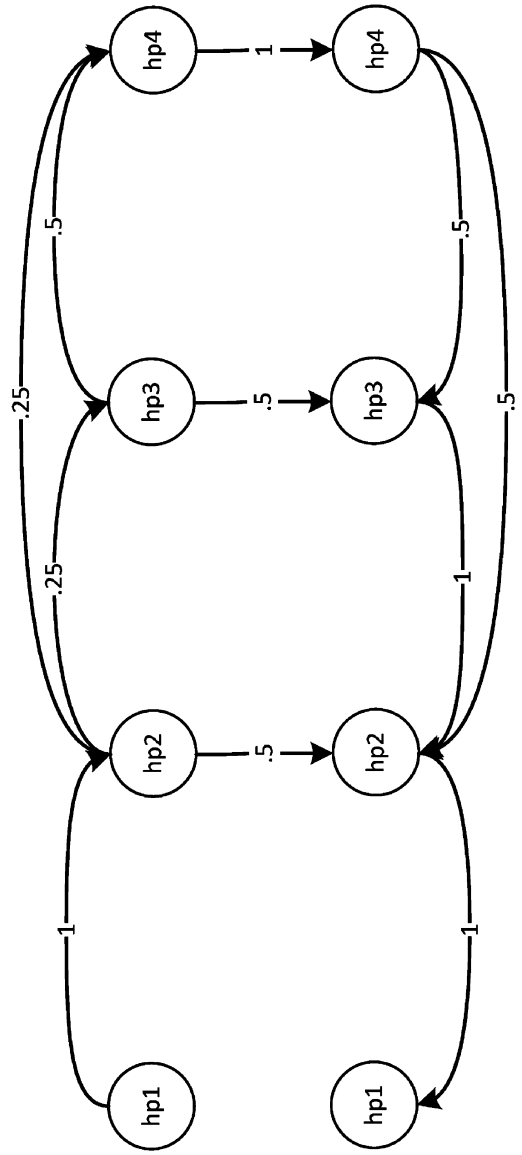


도면4a

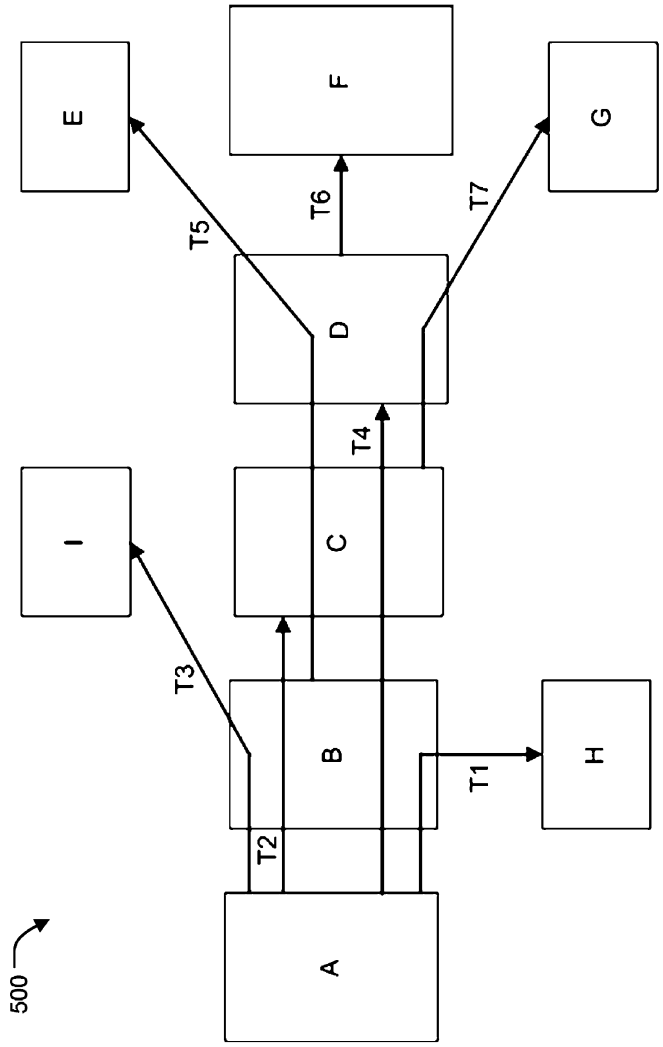



트랜잭션 1 hp1 -> hp2 -> hp1(1,2)
트랜잭션 2 hp1 -> hp2 -> hp3 -> hp2 -> hp1(1, 3, 4, 2)
트랜잭션 3 hp1 -> hp2 -> hp3 -> hp2 -> hp4 -> hp2 -> hp1(1, 3, 4, 7, 8, 2)
트랜잭션 4 hp1 -> hp2 -> hp3 -> hp4 -> hp3 -> hp2 -> hp1(1, 3, 5, 6, 4, 2)
트랜잭션 5 hp1 -> hp2 -> hp3 -> hp4 -> hp3 -> hp2 -> hp4 -> hp2 -> hp1(1, 3, 5, 6, 4, 7, 8, 2)

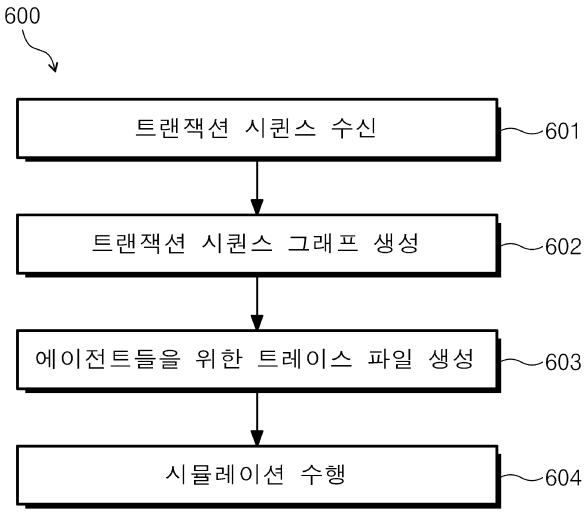
도면4b



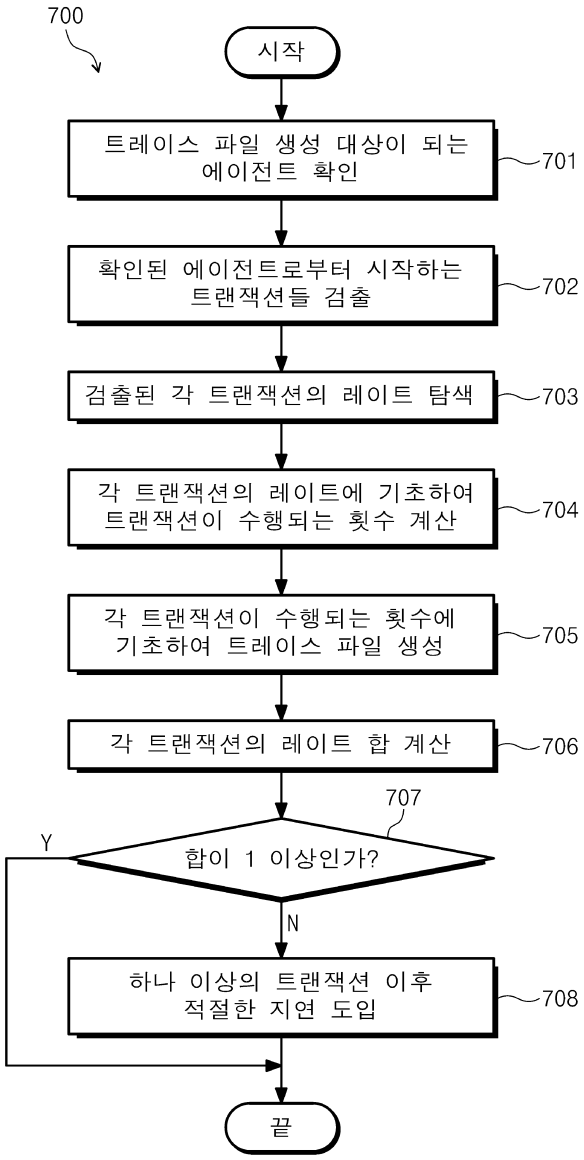
도면5


E
F
G
T5
T6
T7
D
T4
I
C
T3
B
H
T1
T2
A
500

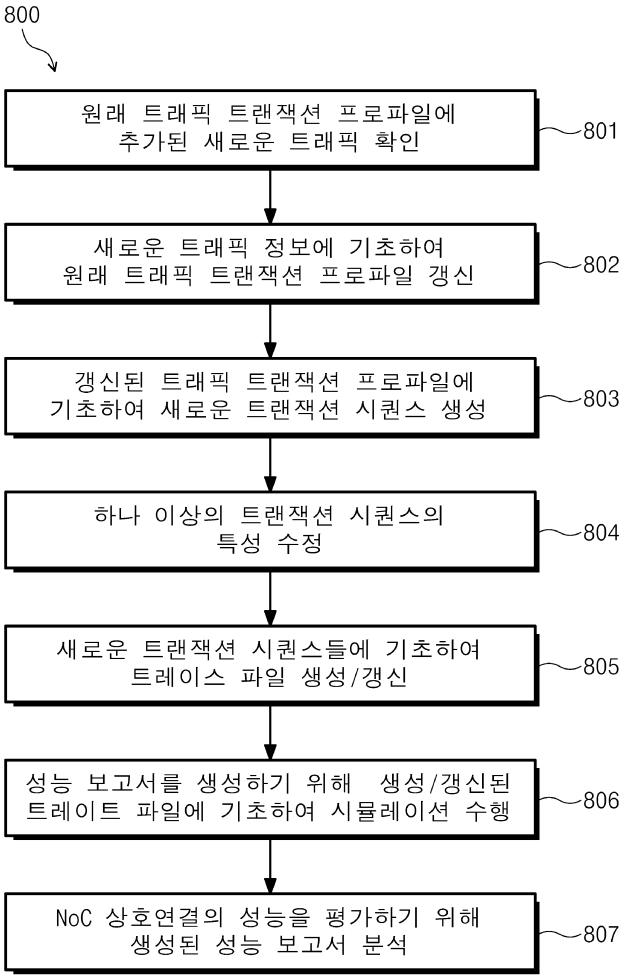
도면6


600
트랜잭션 시퀀스 수신
601
트랜잭션 시퀀스 그래프 생성
602
에이전트들을 위한 트레이스 파일 생성
603
시뮬레이션 수행
604

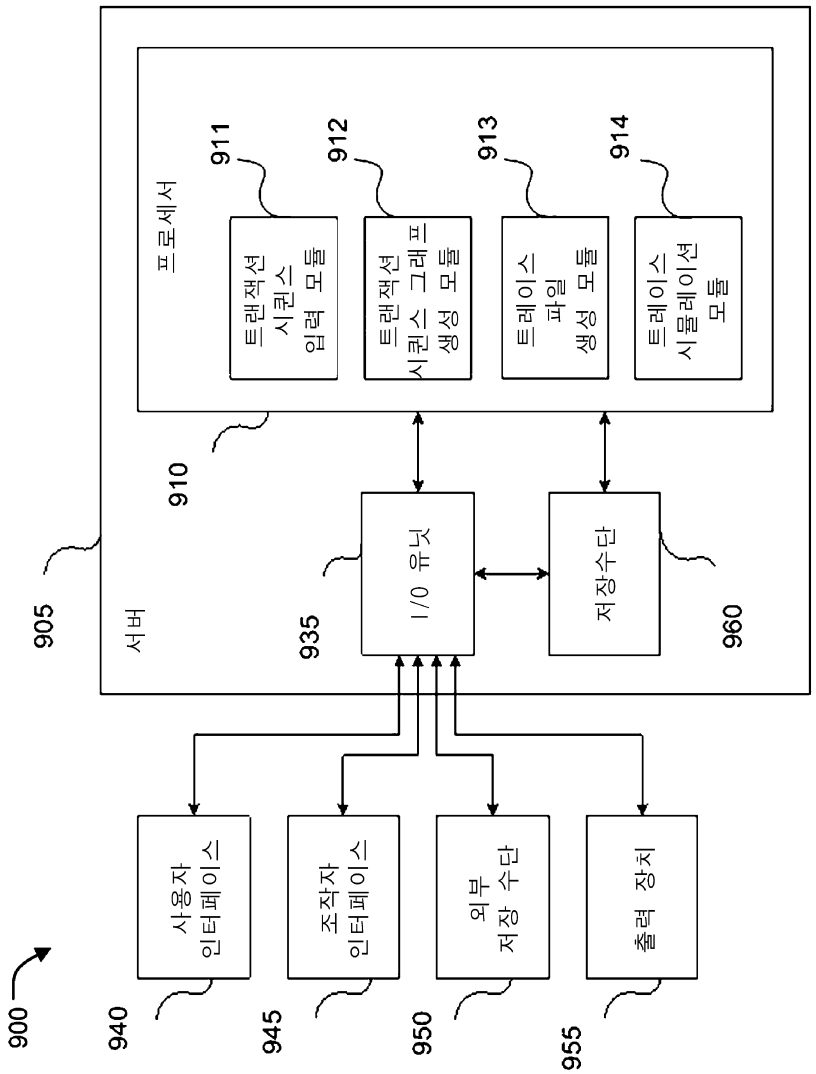
도면7


700
시작
트레이스 파일 생성 대상이 되는 에이전트 확인
701
확인된 에이전트로부터 시작하는 트랜잭션들 검출
702
검출된 각 트랜잭션의 레이트 탐색
703
각 트랜잭션의 레이트에 기초하여 트랜잭션이 수행되는 횟수 계산
704
각 트랜잭션이 수행되는 횟수에 기초하여 트레이스 파일 생성
705
각 트랜잭션의 레이트 합 계산
706
707
합이 1 이상인가?
Y
N
하나 이상의 트랜잭션 이후 적절한 지연 도입
708
끝

도면8


800
원래 트래픽 트랜잭션 프로파일에 추가된 새로운 트래픽 확인
801
새로운 트래픽 정보에 기초하여 원래 트래픽 트랜잭션 프로파일 갱신
802
갱신된 트래픽 트랜잭션 프로파일에 기초하여 새로운 트랜잭션 시퀀스 생성
803
하나 이상의 트랜잭션 시퀀스의 특성 수정
804
새로운 트랜잭션 시퀀스들에 기초하여 트레이스 파일 생성/갱신
805
성능 보고서를 생성하기 위해 생성/갱신된 트레이트 파일에 기초하여 시뮬레이션 수행
806
NoC 상호연결의 성능을 평가하기 위해 생성된 성능 보고서 분석
807

도면9


900
905
서버
910
프로세서
911
트랜잭션 시퀀스 입력 모듈
912
트랜잭션 시퀀스 그래프 생성 모듈
913
트레이스 파일 생성 모듈
914
트레이스 시뮬레이션 모듈
935
I/O 유닛
960
저장수단
940
사용자 인터페이스
945
조작자 인터페이스
950
외부 저장 수단
955
출력 장치