(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년12월06일
(11) 등록번호 10-1678133
(24) 등록일자 2016년11월15일

(51) 국제특허분류(Int. Cl.)
G06F 9/46 (2006.01) *G06F 9/30* (2006.01)
(21) 출원번호 10-2012-7025000
(22) 출원일자(국제) 2011년02월22일
심사청구일자 2016년02월19일
(85) 번역문제출일자 2012년09월24일
(65) 공개번호 10-2012-0130004
(43) 공개일자 2012년11월28일
(86) 국제출원번호 PCT/US2011/025778
(87) 국제공개번호 WO 2011/106333
국제공개일자 2011년09월01일
(30) 우선권주장
12/711,851 2010년02월24일 미국(US)

(56) 선행기술조사문헌
Moravan 외 7명. 'Supporting nested transactional memory in logTM.' Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems, pp.359-370.
US20060294326 A1
US20090276599 A1
KR1020080076981 A

(73) 특허권자
어드밴스드 마이크로 디바이시즈, 인코포레이티드
미국 캘리포니아 94088-3453 서니베일 피.오.박스 3453 원 에이엠디 플레이스
(72) 발명자
정 재웅
미국 워싱톤 98004 벨레뷰 #263 11000 엔이 10번 스트리트
크리스티 데이비드 에스.
미국 텍사스 78739 오스틴 니드햄 레인 6201
(뒷면에 계속)
(74) 대리인
박장원

전체 청구항 수 : 총 10 항 심사관 : 유진태

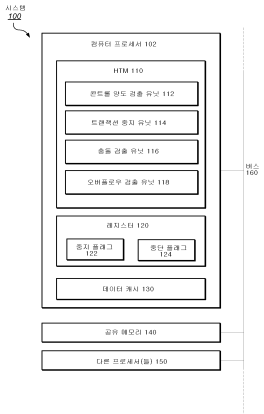
(54) 발명의 명칭 **하드웨어 트랜잭션 메모리에서의 자동 중지 및 재개**

(57) 요 약

컴퓨터 프로세서를 위한 장치 및 방법이 개시되는바, 상기 컴퓨터 프로세서는 다수의 프로세싱 코어들에 의해서 공유되는 메모리에 액세스하도록 구성되며 그리고 복수의 메모리 액세스 동작들을 트랜잭션 모드에서 하나의 원자적 트랜잭션으로서 실행하도록 구성되며 그리고 암시적인 중지 조건(가령, 프로그램 콘트롤 양도)을 결정하는

(뒷면에 계속)

대 표 도 - 도1



것에 응답하여 트랜잭셔널 모드를 중지하도록 구성된다. 트랜잭션을 실행하는 것의 일부로서, 상기 프로세서는 추론적 메모리 액세스 동작들에 의해서 액세스되는 데이터를 추론적 데이터라고 마킹한다. 암시적인 중지 조건을 결정하는 것에 응답하여(실행 중인 쓰레드에서의 콘트롤 양도를 검출하는 것을 포함하여), 프로세서는 트랜잭셔널 실행 모드를 중지하며, 이는 중지 플래그를 세팅하는 것과 그리고 추론적 데이터에 대한 마킹을 중지하는 것을 포함한다. 만일, 프로세서가 나중에 재개 조건(예컨대, 콘트롤 양도로부터의 반환에 대응하는 리턴 콘트롤 양도)을 검출한다면, 프로세서는 추론적 데이터에 대한 마킹을 재개하도록 구성된다.

(72) 발명자

호흐무쓰 마이클 피.

독일 드레스덴 01099 베티나스트라세 12

디에스텔호르스트 스테판

독일 드레스덴 01159 코너트플라츠 18

폴랙 마틴

독일 드레스덴 01099 프리스니츠스트라세 35

명 세 서

청구범위

청구항 1

하드웨어 트랜잭션 메모리에서의 자동 중지 및 재개를 위한 장치로서,

추론적 메모리 액세스 동작들에 의해서 액세스되는 데이터를 추론적 데이터라고 마킹하는 것을 포함하여, 트랜잭셔널 모드에서 복수의 상기 추론적 메모리 액세스 동작들을 하나의 원자적 트랜잭션(single atomic transaction)으로서 실행하는 프로세서를 포함하며, 그리고 상기 프로세서는,

상기 프로세서에 의해서 실행 중인 쓰레드(thread)에서의 콘트롤 양도(control transfer) 및 상기 프로세서가 특정한 특권 레벨에서 실행 중이라는 것 중 적어도 하나를 검출함에 의해서, 암시적인 중지 조건(implicit suspension condition)을 결정하는 검출 유닛과; 그리고

상기 암시적인 중지 조건에 대한 표시(indication)를 수신하는 것에 응답하여 상기 트랜잭셔널 모드를 중지하는 트랜잭션 중지 유닛을 포함하며,

상기 트랜잭셔널 모드의 중지에 응답하여, 상기 트랜잭셔널 모드가 중지되는 동안에 실행되는 메모리 동작들에 의해서 상기 데이터가 액세스되는 때에 상기 프로세서는 데이터를 추론적이라고 마킹하는 것을 중지하며,

상기 프로세서는 상기 하나의 원자적 트랜잭션이 중지되는 동안 상기 하나의 원자적 트랜잭션이 실패하였음을 검출하며, 그리고 상기 프로세서는 상기 추론적 데이터를 폐기하는 것을 포함하여, 상기 실패한 하나의 원자적 트랜잭션을 중단(abort)시키는 것을 특징으로 하는 하드웨어 트랜잭션 메모리에서의 자동 중지 및 재개를 위한 장치.

청구항 2

제1항에 있어서,

상기 프로세서는 또한, 상기 하나의 원자적 트랜잭션에 대한 상기 트랜잭셔널 모드의 재개 조건(resumption condition)을 검출한 이후에만, 상기 실패한 하나의 원자적 트랜잭션을 중단하는 것을 특징으로 하는 하드웨어 트랜잭션 메모리에서의 자동 중지 및 재개를 위한 장치.

청구항 3

제2항에 있어서,

상기 트랜잭셔널 모드의 중지는 상기 프로세서가 중지 플래그(suspend flag)를 세팅하는 것을 포함하며, 그리고 상기 하나의 원자적 트랜잭션이 실패하였음을 검출하는 것에 응답하여 상기 프로세서는 또한 중단 플래그(abort flag)를 세팅하는 것을 특징으로 하는 하드웨어 트랜잭션 메모리에서의 자동 중지 및 재개를 위한 장치.

청구항 4

제3항에 있어서,

상기 프로세서는 상기 중지 플래그와 상기 중단 플래그를 실행 콘텍스트(execution context)의 일부로서 저장하며, 저장된 실행 콘텍스트는 상기 프로세서에 의해서 수행되는 콘텍스트 스위치의 이벤트에서 상기 프로세서의 상태를 복원하는데 이용될 수 있는 것을 특징으로 하는 하드웨어 트랜잭션 메모리에서의 자동 중지 및 재개를 위한 장치.

청구항 5

하드웨어 트랜잭션 메모리에서의 자동 중지 및 재개를 위한 장치로서,

추론적 메모리 액세스 동작들에 의해서 액세스되는 데이터를 추론적 데이터라고 마킹하는 것을 포함하여, 트랜잭셔널 모드에서 복수의 상기 추론적 메모리 액세스 동작들을 하나의 원자적 트랜잭션으로서 실행하는 프로세서

를 포함하며, 그리고 상기 프로세서는,

상기 프로세서에 의해서 실행 중인 쓰레드에서의 콘트롤 양도 및 상기 프로세서가 특정한 특권 레벨에서 실행 중이라는 것 중 적어도 하나를 검출함에 의해서, 암시적인 중지 조건을 결정하는 검출 유닛과; 그리고

상기 암시적인 중지 조건에 대한 표시를 수신하는 것에 응답하여 상기 트랜잭셔널 모드를 중지하는 트랜잭션 중지 유닛을 포함하며,

상기 트랜잭셔널 모드의 중지에 응답하여, 상기 트랜잭셔널 모드가 중지되는 동안에 실행되는 메모리 동작들에 의해서 상기 데이터가 액세스되는 때에 상기 프로세서는 데이터를 추론적이라고 마킹하는 것을 중지하며,

상기 콘트롤 양도로부터의 반환에 대응하는 리턴 콘트롤 양도를 검출하는 것을 포함하여, 상기 검출 유닛은 재개 조건을 결정하며, 그리고 상기 재개 조건의 결정 이후에 상기 프로세서는 데이터를 추론적이라고 마킹하는 것을 재개하는 것을 특징으로 하는 하드웨어 트랜잭션 메모리에서의 자동 중지 및 재개를 위한 장치.

청구항 6

제5항에 있어서,

상기 트랜잭셔널 모드가 중지된 동안에 상기 하나의 원자적 트랜잭션이 실패했다라고 결정하는 것에 응답하여 상기 하나의 원자적 트랜잭션을 중단시킴으로써, 상기 트랜잭션 중지 유닛은 상기 재개 조건의 결정에 응답하며, 상기 중단시키는 것은 상기 추론적 데이터의 폐기를 포함하는 것을 특징으로 하는 하드웨어 트랜잭션 메모리에서의 자동 중지 및 재개를 위한 장치.

청구항 7

제6항에 있어서,

상기 하나의 원자적 트랜잭션이 재개되면, 상기 하나의 원자적 트랜잭션의 중지 이전 및 이후 둘다에서 마킹된 추론적 데이터를 비-추론적이라고 마킹하는 것을 포함하여, 상기 프로세서는 상기 하나의 원자적 트랜잭션을 커미트(commit)하는 것을 특징으로 하는 하드웨어 트랜잭션 메모리에서의 자동 중지 및 재개를 위한 장치.

청구항 8

하드웨어 트랜잭션 메모리에서의 자동 중지 및 재개를 위한 방법으로서,

프로세서에서, 추론적 메모리 액세스 동작들에 의해서 액세스되는 데이터를 추론적 데이터라고 마킹하는 것을 포함하여, 트랜잭셔널 모드에서 복수의 상기 추론적 메모리 액세스 동작들을 하나의 원자적 트랜잭션으로서 실행하는 단계;

상기 프로세서에서, 상기 프로세서에 의해서 실행 중인 쓰레드에서의 콘트롤 양도 및 상기 프로세서가 특정한 특권 레벨에서 실행 중이라는 것 중 적어도 하나를 검출함에 의해서, 암시적인 중지 조건을 결정하는 단계;

상기 프로세서에서, 상기 암시적인 중지 조건에 대한 표시를 수신하는 것에 응답하여 상기 트랜잭셔널 모드를 중지하는 단계, 상기 트랜잭셔널 모드의 중지에 응답하여, 상기 트랜잭셔널 모드가 중지되는 동안에 실행되는 메모리 동작들에 의해서 상기 데이터가 액세스되는 때에 데이터를 추론적이라고 마킹하는 것이 중지되며;

상기 프로세서에서, 상기 콘트롤 양도로부터의 반환에 대응하는 리턴 콘트롤 양도를 검출함으로써 재개 조건을 결정하는 단계; 및

상기 프로세서에서, 상기 재개 조건의 결정 이후에 데이터를 추론적이라고 마킹하는 것을 재개하는 단계

를 포함하는 것을 특징으로 하는 하드웨어 트랜잭션 메모리에서의 자동 중지 및 재개를 위한 방법.

청구항 9

제8항에 있어서,

상기 프로세서에서, 상기 재개 조건의 결정에 응답하여 그리고 상기 트랜잭셔널 모드가 중지된 동안에 상기 하나의 원자적 트랜잭션이 실패했다라고 결정하는 것에 응답하여 상기 하나의 원자적 트랜잭션을 중단시키는 단계를 더 포함하며,

상기 중단시키는 단계는 상기 추론적 데이터를 폐기하는 단계를 포함하는 것을 특징으로 하는 하드웨어 트랜잭션 메모리에서의 자동 중지 및 재개를 위한 방법.

청구항 10

제9항에 있어서,

상기 하나의 원자적 트랜잭션이 재개됨에 응답하여, 상기 프로세서에서 상기 하나의 원자적 트랜잭션을 커미트(commit)하는 단계를 더 포함하며,

상기 커미트하는 단계는 상기 하나의 원자적 트랜잭션의 중지 이전 및 이후 둘다에서 마킹된 추론적 데이터를 비-추론적이라고 마킹하는 단계를 포함하는 것을 특징으로 하는 하드웨어 트랜잭션 메모리에서의 자동 중지 및 재개를 위한 방법.

청구항 11

삭제

청구항 12

삭제

청구항 13

삭제

청구항 14

삭제

청구항 15

삭제

청구항 16

삭제

청구항 17

삭제

청구항 18

삭제

청구항 19

삭제

청구항 20

삭제

청구항 21

삭제

발명의 설명

기 술 분 야

[0001] 하드웨어 트랜잭셔널 메모리(Hardware Transactional Memory : HTM)는 병렬 프로그래밍을 지원하기 위한 컴퓨터

아키텍처의 일 매커니즘이다. HTM을 이용하면, 프로그래머는 명령들의 그룹을 트랜잭션(transaction)이라고 간단하게 선언할 수 있으며 그리고 HTM 시스템은 트랜잭션인 명령들이 원자적이며 격리된 방식(atomic and isolated way)으로 실행되는 것을 보장한다. 원자성(atomicity)은, 트랜잭션의 모든 명령들이 다른 모든 동시 실행 쓰레드들에 대하여 하나의 원자 블록(single atomic block)으로서 실행되는 것을 의미한다. 격리(isolation)는, 트랜잭션이 완료될 때까지 시스템의 나머지 부분에 대하여 트랜잭션의 그 어떤 중간 결과도 노출되지 않는 것을 의미한다. HTM 시스템은 트랜잭션들이 병렬로 구동(run)되게 할 수 있다(이들 트랜잭션들이 충돌하지 않는 한). 2개의 트랜잭션들은 이들 2개의 트랜잭션들 둘다가 동일한 메모리 영역을 액세스하고 그리고 이들 2개의 트랜잭션들 중 어느 하나가 그 메모리 영역에 기입하는 경우에 충돌할 수 있다.

배 경 기 술

[0002] 기존의 많은 HTM 설계들은, 가령, 시스템 콜, 예외(exception), 인터럽트(interrupt), 시그널(signal) 혹은 다른 이벤트들에 응답하여 운영 시스템(operation system)에게 양도하는 것과 같은 프로그램 콘트롤 양도(program control transfer)의 존재하에서, 정확성 문제 및/또는 보안 문제(correctness and/or security issue)를 나타낸다. 전통적으로, 이러한 프로그램 콘트롤 양도는 HTM 시스템에 대하여 투명하게(transparently) 발생된다. 만일, 이러한 양도가 트랜잭셔널 실행(transactional execution) 동안에 발생된다면, 트랜잭션의 일부로서 운영 시스템 코드가 실행될 수도 있다. 이러한 것은 정확성 및/또는 보안성에 관련된 문제들을 야기할 수도 있다. 전통적인 몇몇 시스템들에서는, 운영 시스템에서의 실행 동안에 트랜잭션이 중단(abort)된다면, 예기치 못한 부작용이 발생할 수도 있다.

[0003] 하나의 일례로서, TCP/IP 네트워크 디바이스 드라이버를 고려하자. 네트워크로부터 패킷을 수신하는 경우, 디바이스 드라이버 코드를 실행하도록 인터럽트가 트리거링된다. 어플리케이션 트랜잭션을 구동하는 동안에 프로세서가 인터럽트를 수신한다고 가정하자. 프로그램 콘트롤은 디바이스 드라이버로 양도될 것이며 그리고 통상적인 HTM 시스템 하에서는, 상기 드라이버는 트랜잭션의 일부로서 구동될 것인바, 이는 통상적인 HTM 설계의 경우 트랜잭션의 시작과 끝 사이에 있는 모든 명령들을 트랜잭션의 일부라고 단순하게 간주하기 때문이다.

발명의 내용

해결하려는 과제

[0004] 이러한 거동은 적어도 2개의 문제점을 야기할 수 있다. 첫번째로, 디바이스 드라이버 코드가 실행되는 동안에 트랜잭션이 중단(abort)된다면, 시스템 오류가 야기될 수도 있다. 예를 들어, 메모리-매핑된(memory-mapped) 레지스터에 기입하여 네트워크 인터페이스 하드웨어를 관리하도록 디바이스 드라이버가 구성되어 있으며, 그리고 레지스터 값들을 변경하는 도중에 중단(abort)된다면, 상기 인터페이스 하드웨어는 모순된 상태(inconsistent state)에 남아있을 수 있으며, 이는 네트워크 시스템의 완결성(integrity)를 깨뜨릴 수 있다. 또한, 상기 일례에서, 트랜잭션의 일부로서 디바이스 드라이버가 실행되는 동안에 혹은 이후에 트랜잭션을 중단시키는 것은, 네트워크 연결로 하여금 패킷을 상실하게 할 수도 있다. 예를 들면, 드라이버 코드가 패킷을 프로세싱한 이후, 상기 드라이버는 패킷 송신자에게 확인 메시지(acknowledge message)를 전송한다. 이 시점에서, 송신자는 패킷을 재전송할 필요가 없기 때문에 상기 패킷을 삭제할 수도 있다. 하지만, 트랜잭션이 나중에 중단된다면, 수신기측에 패킷을 저장하는데 이용되었던 메모리 기입 동작이 롤 백(roll back)될 수도 있고 그리고 상기 패킷은 영원히 손실될 수 있으며, 이는 TCP/IP 시스템의 신뢰성 있는 통신 개런티(guarantee)를 위반할 수 있다.

[0005] 두번째로, 보안 문제는, 트랜잭션의 일부로서 운영 시스템 코드 쪽으로로 투명하게 점프하는 것의 다른 부작용들 중 하나이다. 현대적인 많은 프로세서들은, 운영 시스템과 어플리케이션 코드 실행을 분리하기 위한 보안 피처들(security features)을 지원한다. 예를 들면, x86 아키텍처는 시스템 콜들의 경계(boundary)에서 코드 세그먼트 선택기를 스위칭함에 의해서, 운영 시스템과 어플리케이션이 서로 다른 코드 세그먼트들 및 특권 레벨들(privilege levels)(가령, 사용자-레벨 vs. 커널-레벨 특권)을 이용하는 것을 허용한다. 하지만, 통상적인 HTM 시스템은 종종, 코드 세그먼트 선택기 등과 같은 프로세서 상태의 소정 부분들을 체크 및 관리하지 못한다. 사용자-레벨 코드에서 발원된 트랜잭션이 운영 시스템에 대한 시스템 콜을 실행하는 경우, 특권 레벨이 상승된다. 만일, 이러한 트랜잭션이 중단된다면, 통상적인 HTM 시스템은 특권 레벨을 하위 레벨로 복원시키도록 구성되지 않을 수도 있다. 따라서, 보안 누설이 야기될 수도 있다. 악의적인 프로그램들은 트랜잭션이 중단된 이후에 운영 시스템 특권 레벨을 획득하기 위해서 이러한 보안 허점을 이용할 수도 있다.

과제의 해결 수단

[0006] 다수의 프로세싱 코어들에 의해서 공유되는 메모리에 액세스하도록 구성되며 그리고 트랜잭션 모드에서 하나의 원자적 트랜잭션으로서 메모리 액세스 동작들을 실행하도록 구성된 컴퓨터 프로세서를 위한 장치 및 방법이 개시된다. 이러한 장치는 암시적인 중지 조건(implicit suspension condition)을 결정하는 것에 응답하여 트랜잭셔널 모드를 중지하도록 구성된다. 트랜잭션을 실행하는 것의 일부로서, 상기 프로세서는 추론적 메모리 액세스 동작들에 의해서 액세스되는 데이터를 추론적 데이터라고 마킹한다. 암시적인 중지 조건을 결정하는 것에 응답하여(가령, 운영 시스템쪽으로의 양도와 같은, 실행 중인 쓰레드에서의 콘트롤 양도를 검출하는 것을 포함하여), 프로세서는 트랜잭셔널 실행 모드를 중지하며, 이는 중지 플래그를 세팅하는 것과 그리고 추론적 데이터에 대한 마킹을 중지하는 것을 포함한다. 다음으로, 프로세서가 재개 조건을 검출한다면(예컨대, 콘트롤 양도로부터의 반환에 대응하는 리턴 콘트롤 양도를 검출함에 의해서), 프로세서는 추론적 데이터에 대한 마킹을 재개함으로써 트랜잭셔널 실행 모드를 재개한다.

[0007] 몇몇 실시예에서 프로세서는 또한, 트랜잭션이 중지되는 동안에 원자적 트랜잭션이 실패하였음을 검출하도록 구성될 수 있다. 몇몇 실시예에서는, 트랜잭션을 재개하고자 하는 시도가 이루어지는 때에 중단 플래그를 세팅하고 그리고 상기 중단 플래그를 체크함에 의해서, 상기 프로세서는 중지된 트랜잭션이 실패하였음을 검출하는 것에 응답할 수 있다. 만일 중단 플래그가 세팅될 것임을 상기 체크가 판별하면, 프로세서는 트랜잭션을 중단한다. 몇몇 실시예에서, 상기 프로세서는, 실행 콘텍스트(execution context)의 일부로서 중단 플래그 및/또는 중지 플래그의 값들을 세이브/리스토어(save/restore)하도록 구성될 수도 있다.

도면의 간단한 설명

[0008] 도1은 본 발명의 실시예들에 따라 중지(suspend)/재개(resume) 능력을 구비한 HTM 매커니즘을 구현하도록 구성된 컴퓨터 시스템을 예시한 블록도이다.

도2는 본 발명의 실시예들에 따라 메모리 트랜잭션을 실행하는 동안에 트랜잭셔널 실행 모드를 중지시키기 위한 방법을 예시한 순서도이다.

도3은 트랜잭션 동안에 트랜잭셔널 실행 모드를 중지시키는 것을 포함하여 HTM을 이용한 원자적 트랜잭션을 실행하기 위한 방법을 예시한 순서도이다.

도4는 본 발명의 실시예들에 따라 중지된 트랜잭셔널 모드 동안에 검출된 중단 조건(abort condition)을 핸들링하기 위한 방법을 예시한 순서도이다.

도5는 본 발명의 실시예들에 따라 자동화된 중지/재개 능력들을 구비한 HTM을 구현하기 위한 특정 방법을 예시한다.

도6은 그 동안에 중지(suspension)가 수행될 수도 있는 콘트롤 양도의 다양한 유형들을 예시한 테이블이다.

도7은 본 발명의 실시예들에 따라 전술한 바와 같은 자동화된 중지/재개 기능을 구비한 하드웨어 트랜잭셔널 메모리를 구현하도록 구성된 컴퓨터 시스템을 예시한다.

발명을 실시하기 위한 구체적인 내용

[0009] 본 명세서에 사용된 임의의 표제들(headings)은 오직 예시적인 목적이며 상세한 설명 혹은 청구범위를 제한하고자 의도된 것이 아니다. 본 명세서에서 사용된 바와 같이, "할 수 있다(may)" 라는 용어는 강제하는 의미(즉, 해야한다는 의미)가 아니라 허용하는 의미(즉, ~ 할 가능성을 갖는다는 의미)로 이용된다. 이와 유사하게, "포함한다(include, including, includes)" 라는 용어는, 포함하지만 이에 한정되지 않음을 의미한다.

[0010] 통상적인 HTM 설계들은 프로그램 콘트롤 양도(program control transfer)가 있는 경우 정확성 및/또는 보안 문제를 나타낸다. 본 명세서에서 사용되는 바와 같이, "콘트롤 양도(control transfer)" 라는 용어는 프로그램 플로우(program flow)에서의 변화(change)를 시그널링하거나, 야기하거나 혹은 이를 나타내는 명령(instruction) 혹은 이벤트를 지칭한다(예컨대, 소정의 실행 소프트웨어 프로그램으로부터 소정의 관리 프로그램(supervisory program), 혹은 에러(또는, 인터럽트) 핸들러로의 변화 및 그 반대의 경우). 소정의 실시예들에서, 관리 프로그램은 운영 시스템(operating system)이 될 수 있다. 다양한 실시예에서, 서로 다른 운영 시스템들(예컨대, Linux, WindowsTM, 등등)이 이용될 수 있으며, 이들 운영 시스템들 각각은, 컴퓨터의 기본 기능들(예컨대, 작업 스케줄링, 어플리케이션 실행, 그리고 주변장치 제어 등등)을 지원하는 소프트웨어로 구현될 수 있다. 몇몇 실시예에서, 운영 시스템은 기본 기능을 구현하기 위한 커널(kernel)을 포함할 수 있다. 다양한 실시예에서, 운영 시스템의 커널은 모노리식(monolithic)이 될 수도 있으며 혹은 마이크로-커널이 될 수도 있다.

[0011] 몇몇 실시예에서, 하나 이상의 운영 시스템들은 가상(virtual)이 될 수도 있다(예컨대, 운영 시스템은 가상 머신 내의 "게스트(guest)" OS 가 될 수도 있다). 이러한 실시예에서, 콘트롤 양도는 프로그램과 가상 운영 시스템 사이의 프로그램 콘트롤을 가상 운영 시스템을 디플로이(deploy)하는데 이용되는 하이퍼바이저 (hypervisor)에게 양도하거나, 혹은 또 다른 런타임(runtime) 시스템에게 양도한다. 비록, 본 발명의 실시예에서는 운영 시스템으로/운영 시스템으로부터(to/from) 콘트롤을 양도하지만, 가상 운영 시스템, 하이퍼바이저, 및/또는 임의의 다른 런타임 시스템으로/으로부터 양도하는 것을 포함하도록 본 발명의 실시예들이 수정될 수도 있음을 유의해야 한다. 이와 같이, 다르게 표현되지 않는 한, "콘트롤 양도"는 소정의 실행 코드와 이 실행 코드가 구동되는 운영 시스템 간의 프로그램 플로우에서의 변화만으로 한정되지 않는다

[0012] 다른 환경들에서는, 가령, 운영 시스템쪽으로 시스템 콜을 실행하는 프로그램, 소프트웨어 및/또는 하드웨어 예외(exception), 하드웨어 인터럽트, 소프트웨어 시그널, 혹은 상이한 이벤트들과 같은 다른 이벤트들을 검출하는 것에 응답하여 콘트롤 양도가 수행될 수 있다. 어플리케이션으로부터 운영 시스템으로의 각각의 콘트롤 양도는, 대응하는 리턴 콘트롤 양도(return control transfer)을 가질 수 있는데, 리턴 콘트롤 양도는 최초(original)(suspending) 콘트롤 양도가 발생되었던 프로그램으로 콘트롤을 반환(return)한다. 예를 들어, 만일 프로그램이 운영 시스템쪽으로 콘트롤을 양도하기 위하여 시스템 콜을 인보크(invoke)하면, 운영 시스템은 소정의 기능을 실행함에 의해서 응답할 수 있으며 이후, 시스템 콜 인보케이션(invocation) 직후에 인보킹 프로그램 내의 포인트로 콘트롤을 반환할 수 있다

[0013] 다양한 실시예에서, 하드웨어 트랜잭셔널 메모리 시스템(HTM)는 하나의 원자적 트랜잭션으로서 복수의 메모리 액세스 동작들을 실행하기 위한 기능을 제공할 수 있다. 몇몇 실시예에서, 프로그램은 트랜잭셔널 실행 모드를 개시하기 위한 개시 명령(initiating instruction)과 그리고 트랜잭셔널 실행 모드를 종료하기 위한 후속 종결 명령(subsequent terminating instructions)을 실행할 수 있다. HTM은, 트랜잭셔널 모드에서(즉, 개시 명령과 종결 명령 사이) 어플리케이션에 의해서 실행되는 메모리 동작들의 전부 혹은 일부가, 시스템의 다른 쓰레드들(threads)/프로세서들의 실행에 대하여 원자적으로(atomically) 수행되는 것을 보장하도록 구성될 수도 있다

[0014] 원자적 실행(atomic execution)을 보장하기 위하여, HTM은, 트랜잭션이 성공적으로 커미트(commit)될 때까지 트랜잭션의 메모리 액세스 동작들을 추론적(speculative)이라고 취급하도록 구성될 수 있으며 그리고 중단 조건(abort condition)을 모니터링하도록 구성될 수 있다. 예를 들어, HTM은 트랜잭셔널 실행 모드 동안에 추론적 명령들에 의해서 기입되거나 혹은 판독되는 데이터를 추론적이라고 마킹할 수 있으며(mark as speculative), 그리고 트랜잭션이 성공적으로 완료되면(즉, 그 어떤 중단 조건도 검출되지 않으면), HTM은 상기 데이터를 비-추론적(non-speculative)이라고 마킹할 수 있거나 및/또는 그렇지 않으면 이를 공유 메모리에 통합시킬 수 있다. 다른 한편으로, 만일 중단 조건이 검출되면, HTM은, 추론적 데이터를 탈락/무효화(drop/invalidate)시키도록 구성될 수도 있으며 및/또는 공유 메모리에 대한 추론적 데이터 기입들의 효과들을 롤 백(roll back)시키도록 구성될 수도 있다.

[0015] 다양한 실시예들에 따르면, 하드웨어 매커니즘(예컨대, 프로세서)은 액티브 트랜잭션 내에서 프로그램 콘트롤 양도를 검출하는 것을 포함하여, 중지 조건(suspend condition)을 자동으로 검출하도록 구성될 수 있으며 그리고 트랜잭셔널 실행 모드를 중지함에 의해서 이에 응답하도록 구성될 수 있다. 몇몇 실시예에서, HTM은 대응하는 리턴 콘트롤 양도를 후속으로 검출하도록 구성될 수 있으며 그리고 적절하다면, 트랜잭셔널 실행 모드를 재개하도록 구성될 수 있다.

[0016] 몇몇 실시예에서는, 트랜잭셔널 실행 모드가 중지되는 동안, 프로세서는 메모리 동작들에 의해 액세스되는 데이터를 추론적이라고 마킹하지 않는다. 따라서, 이러한 실시예에서는, 만일 트랜잭션이 중단된다면, 트랜잭셔널 모드가 중지되었던 동안에 실행된 메모리 액세스 동작들은 롤 백되지 않을 것이다.

[0017] 몇몇 실시예에서는, 트랜잭셔널 실행이 중지되는 동안에 중단 조건이 검출된다면, HTM은 트랜잭셔널 모드가 재개된 이후가 될 때까지 트랜잭션을 중단시키는 것을 지연하도록 구성될 수 있다. 예를 들면, 중단 조건(예컨대, 데이터 충돌에 의해서 야기되는)을 검출하는 것에 응답하여, HTM은 중단 플래그(abort flag)를 세트할 수 있으며 그리고 중지된 트랜잭셔널 모드에서 실행을 계속할 수 있다. 트랜잭셔널 실행 모드가 재개되면, HTM은 중단 플래그를 체크할 수 있으며 그리고 상기 플래그가 세트되면 상기 중단(abort)을 수행할 수 있다.

[0018] 다양한 실시예들에서, 트랜잭셔널 실행 모드의 중지/재개는 실행 소프트웨어에 대하여 투명한 방식으로 HTM에 의해서 수행될 수 있다. 즉, 자동으로 중지/재개 조건들을 검출하고 그리고 트랜잭셔널 실행 모드를 중지/재개시키도록 HTM이 구성되기 때문에, HTM에게 명령하여 트랜잭셔널 실행 모드를 적절한 시간들에서 중지 혹은 재개시키기 위한 특별한 명령들을 실행하도록 소프트웨어(예컨대, 프로그램 및/또는 운영 시스템)가 수정될 필요는

없다.

[0019] 도1은 본 발명의 실시예들에 따라 중지/재개 능력을 구비한 HTM 매커니즘을 구현하도록 구성된 컴퓨터 시스템을 예시한 블록도이다. 도1에서, 시스템(100)(다른 실시예에서 시스템 100은 트랜잭셔널 메모리 시스템으로 지칭될 수도 있다)은, 컴퓨터 프로세서(102), 공유 메모리(140), 그리고 다른 프로세서들(150)을 포함한다. 이들 구성요소들은 버스(150)에 의해서 연결될 수 있으며, 버스(150)는 브로드캐스트(broadcast) 및/또는 포인트-투-포인트를 포함하는 다양한 유형의 상호연결(interconnect)이 될 수 있다. 다양한 실시예들에서, 상기 다른 프로세서들(150)은 컴퓨터 프로세서(102)와 메모리(140)를 공유하는 하나 이상의 프로세서들을 포함한다. 몇몇 실시예에서, 프로세서들(150) 각각은 컴퓨터 프로세서(102)로 구성될 수도 있다.

[0020] 예시된 실시예에 따르면, 컴퓨터 프로세서(102)는 전술한 바와 같은 중지/재개 능력을 구비한 하드웨어 트랜잭셔널 메모리 기능을 구현하기 위한 HTM(110)을 포함한다. 다양한 실시예에서, HTM(110)은 서로 다른 방식으로 추론적 데이터를 마킹할 수 있다. 예를 들어, 데이터 캐시(130) 등과 같은 데이터 캐시 내에 추론적 데이터를 저장 및 마킹하는 것을 포함하여, HTM(110)은 캐시-기반의 추론적 데이터 버퍼를 이용할 수 있다. 다른 실시예에서, 추론적 데이터 버퍼들은 가령, 로드(load), 스토어(store) 혹은 로드/스토어 큐(queue)와 같은 다른 구조로 구현될 수도 있다. 다른 실시예에서, 각각의 프로세서는 다수개의, 협력성 추론적 데이터 버퍼들(cooperating speculative data buffers)을 구비할 수 있다(예컨대, 데이터 캐시 130 내에 그리고 로드/스토어 큐 내에).

[0021] 시스템(100)에서, HTM(110)은 콘트롤 양도 검출 유닛(112), 트랜잭션 중지 유닛(114), 충돌 검출 유닛(116), 그리고 오버플로우 검출 유닛(118) 등을 포함하는 다양한 하드웨어 유닛들을 포함한다. 다양한 실시예에서, 이들 구성요소들 중 서로 다른 것들은 전체적으로 혹은 부분적으로 결합될 수 있으며 혹은 더 작은 서브컴포넌트들로 쪼개질 수도 있다.

[0022] 몇몇 실시예에서, 콘트롤 양도 검출 유닛(112)은 중지 조건을 결정하도록 구성될 수 있다. 소정 실시예에서, 이러한 중지 조건은 "암시적인(implicit)" 중지 조건이 될 수도 있는데, 이는 명시적인 명령(예컨대, "pause" 혹은 "suspend")의 존재 말고도 다른 기준들의 소정 세트에 기초하여 중지 조건이 결정될 수 있음을 의미한다. 몇몇 실시예에서, 상기 유닛(112)은, 프로세서(102)에 의해 실행중인 쓰레드에서의 콘트롤 양도를 검출하는 것에 부분적으로 기초하여 중지 조건이 존재함을 결정하도록 구성될 수도 있다. 예를 들어, 콘트롤 양도 검출 유닛은, 시스템 콜 인보케이션 혹은 하드웨어 인터럽트(예컨대, I/O 인터럽트)에 의해서 야기되는 콘트롤 양도를 검출할 수 있다.

[0023] 다양한 실시예에서, 중지 조건을 검출하는 것은, 프로세서(120)가 현재 트랜잭셔널 실행 모드에 있는지를 확인하는 것 및/또는 프로세서가 상위 특권 레벨(가령, 커널 레벨)이 아니라 하위 특권 레벨(가령, 어플리케이션-레벨)에서 실행 중임을 확인하는 것을 포함할 수 있다. 예를 들어, 몇몇 시스템에서는, 현재 특권 레벨(current privilege level : CPL)이 레지스터(가령, 레지스터 120)에 저장되며 그리고 다양한 특권 레벨들을 나타내는 값을 보유할 수 있다. 이러한 시스템에서는, 상위 특권 레벨에서, 상기 시스템은 보호받고 있는 리소스들(가령, 다양한 메모리 영역들, I/O 포트들, 및/또는 특별한 명령들)에 대한 실행 리소스의 액세스를 허용한다. 예를 들면, x86 아키텍처의 경우, 특권이 가장 높은 0에서부터 특권이 가장 낮은 3까지의 4개의 특권 레벨들이 존재한다. 이러한 시스템에서, 레벨 0은 커널/관리(kernel/executive)를 위해 이용되며 그리고 레벨 3은 어플리케이션 프로그램들을 위해 이용된다. 몇몇 시스템에서는, 소프트웨어가 상승된 특권 레벨에 있는 명령들을 실행할 필요가 있는 경우(가령, 운영 시스템 코드를 실행하는 경우), 이 소프트웨어는 특권 레벨을 높이기 위한(예컨대, 레벨 3에서 레벨 0으로) 하나 이상의 명령들을 수행할 수도 있으며 그리고 상승된 특권 레벨이 더 이상 필요 없는 경우 특권 레벨을 낮추기 위한(예컨대, 레벨 0에서 레벨 3으로) 하나 이상의 명령들을 수행할 수도 있다. 몇몇 시스템에서, 운영 시스템 코드는 엔트리 및 엑시트(entry and exit) 각각에 대한 특권 레벨을 높이거나 혹은 낮출 수 있다.

[0024] 몇몇 실시예에서, 콘트롤 양도 검출 유닛(112)은, a) 운영 시스템으로의 콘트롤 양도, b) 프로세서가 트랜잭셔널 모드에서 실행중임, 그리고 c) 현재의 특권 레벨이 상승되지 않음(예컨대, 레벨 3) 을 검출하는 것에 응답하여 중지 조건을 검출하도록 구성될 수 있다. 중지 조건을 검출하는 것에 응답하여, 콘트롤 양도 검출 유닛(112)은 트랜잭션 중지 유닛(114)에게 중지 조건을 통지할 수 있다. 이에 응답하여, 트랜잭션 중지 유닛(114)은 가령, 레지스터(120)에 중지 플래그(122)를 세팅하는 것과 같이, 트랜잭셔널 실행 모드를 중지함으로써 이에 응답하도록 구성될 수 있다. 전술한 바와 같이, 트랜잭셔널 모드가 중지되는 동안, 몇몇 실시예에서 프로세서는, 메모리 동작들에 의해서 액세스되는 데이터를 추론적이라고 마킹하는 것을 중지하도록 구성된다. 하지만, 트랜잭

셔널 모드가 중지되기 전에 추론적이라고 마킹된 데이터는, 추론적 마크(speculative mark)와 함께 유지된다. 따라서, 소정 실시예에서, 중단(abort)이 발생하면, 중지 동안에 발생된 메모리 값들에 대한 수정들(modifications)은 롤 백되지 않을 것이다. 반면에, 트랜잭셔널 모드가 중지되기 이전에 혹은 이후에 발생된 수정들은 롤 백될 것이다.

[0025] 몇몇 실시예에서, 콘트롤 양도 검출 유닛(112)은 또한, 중지 조건을 결정하기 위해 검출되었던 콘트롤 양도에 대응하는 리턴 콘트롤 양도를 검출하는 것을 포함하여, 재개 조건(resumption condition)을 결정하도록 구성될 수도 있다. 예를 들어, 만일 중지 조건을 결정하기 위해 검출되었던 콘트롤 양도가 운영 시스템에 대한 시스템 콜이었다면, 리턴 트랜스퍼 콜은 시스템 콜로부터의 리턴이 될 수 있다.

[0026] 몇몇 상황에서는, 중지 조건을 최초로(originally) 야기하는 콘트롤 양도(본 명세서에서 "서스펜딩" 양도("suspending" transfer)라고 지칭되기도 함) 다음에는, 서스펜딩 콘트롤 양도로부터의 리턴이 수행되기 이전에 하나 이상의 후속 콘트롤 양도가 뒤따를 수도 있다. 이러한 경우, 비록 다수개의 리턴 콘트롤 양도들이 수행될 수도 있지만, 상기 서스펜딩 콘트롤 양도에 대응하는 리턴 콘트롤 양도가 수행될 때까지 트랜잭셔널 모드가 재개되지 않을 것이다.

[0027] 몇몇 실시예에서, 콘트롤 양도 검출 유닛(112)은, 트랜잭셔널 실행 모드를 재개하도록 트랜잭션 중지 유닛에게 통지함으로써, 재개 조건을 검출하는 것에 응답하도록 구성될 수 있다. 몇몇 실시예에서, 트랜잭셔널 모드를 재개하는 것은, 다음에 설명되는 바와 같이 중지 플래그(suspend flag)(122)를 언세팅하고(unsetting) 및/또는 중단 플래그(abort flag)(124)를 체크하는 것을 포함할 수 있다.

[0028] 예시적인 실시예에 따르면, HTM(110)은 예컨대, 충돌 검출 유닛(116)과 같은 다양한 다른 매커니즘들 및/또는 유닛들을 포함할 수 있다. 다양한 실시예에서, 충돌 검출 유닛(116)은 서로 다른 종류의 중단 조건들을 검출하도록 구성될 수 있다. 예를 들어, 충돌 검출 유닛은 가령, MESI 혹은 MOESI 와 같은 캐시 일관성 프로토콜(cache coherency protocol)의 일부로서 다른 프로세서(150)로부터 수신된 캐시 일관성 메시지들(프로브들)을 모니터링하도록 구성될 수 있다. 만일, 하나 이상의 이러한 프로브들이 데이터 충돌을 나타낸다면, 충돌 검출 유닛(116)은 중단 조건이 존재하며 그리고 트랜잭션 시도가 중단될 수도 있다고 결정한다.

[0029] 전술한 바와 같이, 몇몇 실시예에서, HTM은 트랜잭셔널 실행 모드가 재개될 때까지 중지된 트랜잭셔널 모드에서 실행중인 트랜잭션 시도를 중단시키는 것을 지연시키도록 구성될 수 있다. 이러한 실시예에서는, 중지된 트랜잭셔널 모드 동안 중단 조건을 검출하는 것에 응답하여, 상기 충돌 검출 유닛은 트랜잭션을 그 즉시(right away) 중단 및/또는 롤 백시키는 것이 아니라, 가령 124와 같은 중단 플래그를 세팅하도록(즉, 기결정된 값을 적절한 저장 위치에 저장함) 구성될 수 있다. 트랜잭셔널 실행 모드가 재개되는 때(예컨대, 운영 시스템 커널로부터 리턴하면), HTM은 중단 플래그(124)가 세팅되었음을 검출할 수 있으며 그리고 트랜잭션 시도를 중단시킴에 의해서 이에 응답할 수 있다.

[0030] 다양한 다른 하드웨어 유닛들은 다른 중단 조건들을 검출할 수 있다. 예를 들어, 오버플로우 검출 유닛(118)은 추론적 버퍼 오버플로우 조건(speculative buffer overflow condition) 형식의 중단 조건을 검출하도록 구성될 수 있다. 트랜잭션을 실행하는 것이, 새로운 추론적 데이터를 버퍼링하기에 불충분한 용량을 갖는 추론적 버퍼(예를 들면, 데이터 캐시 130로 구현됨)에 데이터의 소정 부분이 버퍼링될 것을 요구하는 경우, 버퍼 오버플로우 조건이 검출될 수 있다. 본 명세서에서 사용되는 바와 같이, "추론적 버퍼 오버플로우" 는, 오버플로우 조건이 검출됨을 지칭하며, 추론적 데이터가 버퍼로부터 실제로 퇴거(evict)되는 상황을 지칭하는 것이 아니다.

[0031] 데이터 충돌의 경우에서와 같이, 중지된 트랜잭셔널 모드에서 프로세서가 동작하는 동안 오버플로우 검출 유닛(118)이 오버플로우 조건을 검출한다면, 이것은 중단 플래그(124)를 세팅할 수 있으며, 이는 트랜잭셔널 실행 모드가 재개되는 때에 HTM(110)으로 하여금 트랜잭션 시도를 중단/롤백시키게 할 수 있다.

[0032] 도2는 본 발명의 실시예들에 따라 메모리 트랜잭션을 실행하는 동안에 트랜잭셔널 실행 모드를 중지시키기 위한 방법을 예시한 순서도이다. 방법(200)은, 메모리 트랜잭션을 이용하도록 설정된 컴퓨터 프로그램을 실행하는 동안에 HTM(가령, 도1의 HTM 110)에 의해서 실행될 수 있다.

[0033] 예시된 실시예에 따르면, 상기 방법(200)은 210에서와 같이 프로세서가 트랜잭셔널 모드에서 실행을 개시하는 때에 시작한다. 몇몇 실시예에서, 이는 트랜잭션 실행을 개시하기 위한 하나 이상의 구조화된(architected) 명령들을 실행하는 소프트웨어 프로그램에 응답하여 수행될 수도 있다. 210에서와 같이 트랜잭셔널 실행 모드를 개시하는 것은, 가령, 시스템(100)의 레지스터(120)에 하나 이상의 플래그들을 세팅하는 것을 포함할 수 있다.

[0034] 프로세서가 트랜잭셔널 모드에서 실행 중이면, 이것은 추론적 메모리 액세스 동작에 의해서 액세스되는 데이터

를 추론적이라고 마킹하도록 구성될 수 있다(220). 몇몇 실시예에서는, 트랜잭셔널 모드에서 실행되는 모든 메모리 액세스 동작은 추론적이라고 간주될 수 있으며, 따라서 이러한 각각의 메모리 액세스 동작에 의해서 액세스되는 데이터는 추론적이라고 마킹될 수 있다. 다른 실시예에서, 트랜잭션 내의 메모리 액세스 동작들의 서브세트는, 추론적이라고 명시적으로 지명될 수도 있다. 이러한 소정 실시예에서는, 오직 이들 명시적으로 추론적인 명령들 각각에 의해서 액세스되는 데이터만이 추론적이라고 마킹될 수 있다(220).

[0035] 다양한 실시예에서, 데이터를 추론적이라고 마킹하는 프로세스는, HTM에 의해서 이용되는 추론적 버퍼의 유형에 의존할 수도 있다. 예를 들면, 데이터 캐시 내에 추론적 데이터를 버퍼링하는 HTM들의 경우, 데이터를 추론적이라고 마킹하는 것은, 데이터의 복사본을 데이터 캐시에 저장하고 그리고 추론적 데이터를 포함하고 있는 캐시의 캐시 블록을 하나 이상의 추론적 비트를 이용하여 마킹하는 것을 포함할 수 있다. 추론적 비트를 마킹하는 것은 가령, 로드/스토어 큐와 같은 다른 구조들에 의해서 구현되는 버퍼들 내에 추론적 데이터를 마킹하는데에도 또한, 이용될 수 있다.

[0036] 예시적인 실시예에 따르면, 트랜잭셔널 실행 동안의 소정 포인트에서, HTM(혹은 가령, 콘트롤 양도 검출 유닛 112과 같은 HTM의 소정의 구성요소)은 230에서와 같이 콘트롤 양도를 검출하는 것을 포함하여 중지 조건을 결정한다. 몇몇 실시예에서, 중지 조건은, 프로그램 콘트롤이 운영 시스템 커널로 지금 곧 양도될 예정임을 나타내는 것이 될 수도 있다. 예를 들면, 몇몇 실시예에서는, 시스템 콜이 발생되며 그리고 프로세서의 현재 특권 레벨이 소정의 어플리케이션 레벨(예컨대, 이러한 소정의 특권 레벨은 레벨 3(사용자 혹은 어플리케이션 레벨)이 될 수 있음)에 대응한다는 것에 응답하여 중지 조건이 검출될 수 있다. 또 다른 일례에서는, 하드웨어 인터럽트를 검출하고 그리고 현재 특권 레벨(CPL)이 어플리케이션 특권 레벨(가령, 레벨 3)을 나타내는 것에 응답하여 중지 조건이 검출될 수 있다. 따라서, 이러한 실시예에서, 트랜잭셔널 실행 동안에 콘트롤 양도가 검출되면, 콘트롤 양도가 어플리케이션으로부터 운영 시스템 커널로의 양도인지를 결정하도록, HTM은 CPL을 체크할 수 있다. 만일, 그렇다면, 중지 조건이 존재하는 것으로 결정될 수 있다.

[0037] 상기 방법(200)에서와 같이, 중지 조건을 검출하는 것에 응답하여, HTM은 트랜잭셔널 실행을 중지할 수 있다(240). 트랜잭셔널 실행을 중지하는 것은, 중지 플래그(예컨대, 중지 플래그 122, 이는 소정 저장 위치에 저장된 값을 나타낸다)를 세팅하는 것을 포함할 수 있으며, 중지 플래그는 트랜잭셔널 실행 모드가 중지됨을 HTM에게 알려줄 수 있다.

[0038] 트랜잭셔널 실행 모드가 중지되면, 프로세서는 250에서와 같이, 액세스된 데이터를 추론적이라고 마킹함이 없이 메모리 동작들을 실행할 수 있다. 즉, 콘트롤이 운영 시스템쪽으로 양도되면, 트랜잭셔널 실행 모드가 중지되며 그리고 운영 시스템에 의해서 실행되는 메모리 동작들은 추론적이라고 취급되지 않는바, 따라서 중단 이벤트의 경우에도 롤백되지 않는다. 전술한 바와 같이, 몇몇 실시예에서는, 중단 조건이 검출된 경우라 하더라도 프로세서는 중지 모드(suspended mode) 동안 트랜잭션 시도를 중단하지 않을 수도 있다. 이러한 실시예에서, 프로세서는 중단 플래그(소정의 저장 위치에 저장된 값)를 세팅하고 그리고 트랜잭셔널 실행 모드가 재개되면 트랜잭션을 중단(롤백을 포함하여)시킴으로써, 중단 조건을 검출하는 것에 응답할 수 있다.

[0039] 도3은 트랜잭션 동안에 트랜잭셔널 실행 모드를 중지시키는 것을 포함하여 HTM을 이용한 원자적 트랜잭션을 실행하기 위한 방법을 예시한 순서도이다. 305에서 트랜잭셔널 실행 모드를 개시함으로써, 방법(300)이 시작된다. 앞서 설명한 바와 같이, 프로세서는 트랜잭션 개시 명령을 실행하는 것에 응답하여 상기 단계를 수행할 수도 있다.

[0040] 상기 방법(300)에서, HTM이 중지 조건을 검출한다면(단계 310에서 '예' 화살표로 표시됨), HTM은 가령, 중지 플래그를 셋팅함에 의해서 트랜잭셔널 모드를 중지시킬 수 있다(315).

[0041] 단계 310에서 중지 조건을 검출하는 것은 도2의 단계 230에서 중지 조건을 검출하는 것과 유사할 수 있으며, 그리고 액티브(즉, 중지되지 않음) 트랜잭셔널 모드 동안에 운영 시스템쪽으로의 콘트롤 양도를 검출하는 것을 포함할 수 있다. 운영 시스템쪽으로의 콘트롤 양도를 검출하는 것에 응답하여, HTM은 사용된 콘트롤 양도의 유형을 또한 기록할 수 있는데 이는, 적절한 리턴 콘트롤 양도가 나중에 식별될 수 있게 하기 위한 것이다. 중지 조건이 검출되지 않는다면(단계 310에서 '아니오' 화살표로 표시됨), 트랜잭션은 중지되지 않는다.

[0042] 상기 방법(300)에 따르면, 이후 프로세서는 다음번 메모리 액세스 명령을 실행한다(단계 320). 트랜잭셔널 실행 모드가 중지되지 않는다면(단계 325에서 '아니오' 화살표로 표시됨), HTM은 메모리 동작에 의해서 액세스되는 임의의 추론적 데이터를 추론적이라고 마킹한다(단계 330). 예를 들어, 상기 동작이 판독 동작이라면, HTM은 그 값이 판독되었던 메모리 위치가 트랜잭션의 판독 세트의 일부임을 알려줄 수 있다. 따라서, 추론적 메모리 위치

에 저장된 값을 다른 프로세서가 변경하였음을 나타내는 무효화 프로브(invalidating probe)가 수신되면, 프로세서는 중단 조건을 검출한다.

[0043] 전술한 바와 같이, 몇몇 실시예에서는, 액티브 트랜잭션에 있는 모든 메모리 동작이 추론적이라고 간주될 수 있다. 반면에, 다른 실시예에서는 소정의 데이터 및/또는 메모리 동작들은 추론적이라고 명시적으로 식별될 수 있는 반면에 다른 것들은 그렇지 않다. 이러한 실시예에서, 오직 비-추론적인 데이터만을 액세스하는 메모리 동작은 임의의 데이터가 추론적이라고 마킹되게 야기하지 않을 것이다. 예를 들어, 이러한 일실시예에서는, 단계 325에서의 '아니오' 출력이 메모리 동작 및/또는 추론적이라고 명시적으로 마킹되었던 메모리 동작에 의해서 액세스되는 데이터에 추가로 좌우되도록 상기 방법(300)이 증대될 수 있다.

[0044] 추론적 데이터 버퍼를 구현하기 위하여 캐시가 이용되는 실시예들에서, 판독 데이터에 대응하는 캐시 블록은, 트랜잭셔널 실행 모드에서 판독되었음을 나타내는 "추론적 플래그(speculative flag)"를 이용하여 마킹될 수 있다. 몇몇 실시예들에서, 각각의 캐시 블록은, 메모리 영역에 대해서 수행되었던 추론적 메모리 액세스 동작(있다면)의 유형을 나타내는 "판독 추론적 플래그(read speculative flag)" 와 "기입 추론적 플래그(write speculative flag)" 을 포함하거나 혹은 이들에 관련될 수 있다. 다양한 실시예들에서, 이들 플래그들은 결합되거나 혹은 분리될 수도 있으며, 그리고 가령, 데이터 캐시, 로드/스토어 큐, 및/또는 기타 다른 것들에 의해서 구현되는 바와 같은, 서로 다른 추론적 버퍼들에서 이용될 수도 있다.

[0045] 예시된 실시예에 따라, 만일 트랜잭션이 중단된다면(단계 325의 "예" 화살표), 추론적 데이터에 대한 마킹(단계 330에서와 같은)이 HTM에 의해서 수행되지 않는다. 대신에, HTM은 재개 조건이 존재하는지의 여부를 결정할 수 있다(단계 335). 몇몇 실시예에서, 재재 조건은, 운영 시스템으로부터 어플리케이션으로의 리턴 콘트롤 양도를 검출하는 것에 응답하여 검출될 수 있다. 몇몇 실시예에서, 이러한 리턴 콘트롤 양도를 검출하는 것은, 실행중인 콘트롤 양도(혹은, 곧바로 실행될 예정인 콘트롤 양도)가 운영 시스템으로의 최초(original)(suspending) 콘트롤 양도에 대응하는 반환(return)인지를 결정하는 것을 포함한다. 또한, 재개 조건을 검출하는 것은, 프로세서가 가령 레벨 0과 같은 특권 레벨(예컨대, 커널-레벨)에서 실행중임을 나타내는 CPL에 추가로 의존할 수도 있다.

[0046] 방법(300)에서, 만일 재개 조건이 검출되면(단계 335의 "예" 화살표), 트랜잭셔널 실행 모드가 재개될 수 있다(단계 340). 트랜잭셔널 실행 모드를 재개하는 것은, 중지 플래그를 언세팅하는 것을 포함할 수 있으며, 따라서 추론적 데이터에 대한 마킹이 적절히 재개되어야 함을 HTM에게 알려줄 수 있다. 도4를 참조하여 다음에 설명되는 바와 같이, 트랜잭셔널 모드를 재개하는 것은 또한, 중지 동안에 중단 조건이 검출되었는지의 여부를 결정하는 것을 포함할 수 있다(예컨대, 중단 플래그를 체크함에 의해서).

[0047] 도3에 예시된 실시예에서, 만일 트랜잭션이 더 많은 명령들을 포함한다면(단계 345의 "예" 화살표), 프로세서는 상기 방법(300)을 반복할 수 있다(단계 345에서 단계 310으로의 피드백 루프로 표현됨). 그렇지 않다면, 단계 345의 "아니오" 화살표로 표현되는 바와 같이, 트랜잭션이 커미트(commit)될 수 있다(단계 350). 다양한 실시예에서, 트랜잭션을 커미트하는 것은 서로 다른 단계들을 포함할 수 있다. 예를 들면, 트랜잭션을 커미트하는 것의 일부로서, 추론적이라고 마킹된 데이터는 비-추론적이라고 마킹될 수도 있다. 몇몇 실시예에서 트랜잭션을 커미트하는 것은, 추론적으로 기입된 데이터를 공유 메모리에 기입하는 것 혹은 이게 아니면 추론적으로 기입된 값들을 시스템의 다른 프로세서들과 공유하는 것을 포함할 수 있다. 비록, 상기 방법(300)은 단계 310에서의 중지 조건의 검출과 단계 335에서의 재개 조건의 검출 사이에서 적어도 하나의 메모리 액세스 명령을 실행하는 것을 포함하고 있지만, 해당 기술분야의 당업자라면 중지된 트랜잭션의 모든 인스턴스들에 대해서 반드시 이럴 필요는 없다는 점을 능히 이해할 것이다.

[0048] 도4는 본 발명의 실시예들에 따라 중지된 트랜잭셔널 모드 동안에 검출된 중단 조건(abort condition)을 핸들링하기 위한 방법을 예시한 순서도이다. 앞의 경우와 같이, 단계 405에서 트랜잭셔널 실행 모드를 개시함에 의해서 방법이 시작되며 그리고 중지 조건을 검출하는 것에 응답하여 트랜잭셔널 실행 모드를 중지시킨다(단계 410). 트랜잭셔널 실행 모드가 중지되면, 프로세서는 비-트랜잭셔널 모드(non-transactional mode)에서 운영 시스템 코드를 실행하기 시작할 것이다(즉, 액세스된 데이터를 추론적이라고 마킹함이 없이).

[0049] 방법(400)에 따르면, HTM은 트랜잭션 시도가 실패임을 나타내는 중단 조건을 검출할 수 있다(단계 415). 다양한 실시예에서, 서로 다른 환경들은 전술한 바와 같은 다른 프로세서와의 데이터 충돌 혹은 추론적 버퍼 오버플로우 조건의 검출 등을 포함하는 중단 조건을 야기할 수 있다. 예를 들면, 상기 프로세서에 의해서 추론적이라고 마킹된 메모리 영역을 다른 프로세서가 액세스하고, 그리고 상기 프로세서들 중 어느 하나가 상기 메모리 영역에 저장된 데이터를 변경하는 경우, 데이터 충돌이 발생할 수 있다. 가령, 추론적 데이터 버퍼(가령, 데이터 캐

시)가 트랜잭션 동안에 액세스되는 모든 추론적 데이터를 버퍼링하기에 충분치 못한 용량을 갖는 경우에서와 같이, 추론적 버퍼 오버플로우 조건이 검출되는 것에 응답하여, 중단 조건의 다른 일례가 검출될 수 있다. 너무 많은 추론적 데이터 및/또는 비-추론적 데이터가 동시에 버퍼링되기 때문에, 상기 버퍼는 충분치 못한 용량을 가질 수 있다.

[0050] 통상적인 시스템에서, HTM은 중단 조건이 검출되면 트랜잭션을 즉시 중단시킬 것이다. 하지만, 이러한 것은 운영 시스템 소프트웨어의 다양한 스트럭처들을 불일치 상태(inconsistent state)로 남겨놓는다. 대신에, 본 발명의 다양한 실시예에 따른 HTM은, 트랜잭션이 재개될 때까지, 중지된(suspended) 트랜잭션을 중단(abort)시키는 것을 지연시킬 수 있다. 예를 들어, 도4의 방법(400)에서, 단계 415에서 중단 조건을 검출하는 것에 응답하여 HTM은 중단 플래그를 세팅하며(단계 420), 그리고 중지된 트랜잭셔널 모드에서 실행을 계속한다(단계 425).

[0051] 중지된 트랜잭셔널 모드에서 운영 시스템에서의 실행이 계속되는 동안, 다양한 추가적인 콘트롤 양도들이 수행될 수 있다. 후속 포인트에서(at a later point), HTM은 서스펜딩 콘트롤 양도에 매칭되는 리턴 콘트롤 양도를 검출하고 그리고 콘트롤을 어플리케이션 코드로 반환하는 것에 응답하여 재개 조건을 검출할 수 있다(단계 430).

[0052] 예시된 실시예에 따르면, 어플리케이션 코드로 반환하고 그리고 트랜잭셔널 실행을 재개한 이후에, HTM은 중단 플래그가 세팅되는지의 여부를 결정할 수 있다(단계 435). 중단 플래그가 세팅되지 않는다면, 트랜잭션은 중지된 실행 모드 동안에 실패하지 않았으며 따라서 HTM은 중지된 플래그를 클리어할 수 있고 그리고 트랜잭셔널 모드에서 실행을 계속할 수 있다. 하지만, 상기 방법(400)에서, 중단 플래그가 단계 420에서 세팅되었다면, HTM은 단계 435에서 중단 플래그가 세팅되었다고 결정할 것이며 그리고 이에 응답하여, 트랜잭션을 중단시킨다(단계 440).

[0053] 전술한 바와 같이, 단계 440에서 트랜잭션을 중단하는 것은, 추론적 데이터를 무효(invalid)라고 마킹하는 것을 포함할 수 있다. 이러한 것은 본 명세서에서 추론적 데이터를 드롭(drop)하는 것으로 지칭될 수 있다. 추론적 데이터가 드롭되면, 이는 다른 프로세서들에 의한 액세스를 위한 공유 메모리에 저장되지 않는다. 예를 들어, 추론적 버퍼 내의 추론적 데이터가 무효라고 마킹되면, 프로세서는 미래의(future) 명령 실행에 대해서 상기 데이터에 의존하지 않을 뿐만 아니라, 이 데이터를 공유 메모리로 포워딩하지도 않는다. 또한, 무효화된 데이터와 시스템 내의 다른 프로세서들을 조화시키기 위한 미래의 그 어떤 트랜잭셔널 중단들 혹은 캐시 일관성 액션들도 취해질 필요가 없다.

[0054] 도5는 본 발명의 실시예들에 따라 자동화된 중지/재개 능력들을 구비한 HTM을 구현하기 위한 특정 방법을 예시한다. 단계 500에서 트랜잭셔널 실행 모드를 개시함에 의해서 방법(502)이 시작된다. 실행 동안, HTM은 운영 시스템 커널로의 콘트롤 양도를 검출한다(단계 505). 단계 510의 "예" 화살표로 표시되는 바와 같이, 중지된 트랜잭셔널 모드에서 프로세서가 실행중이라면, 트랜잭셔널 실행 모드를 중지시키는 것에 관하여 그 어떤 추가적인 액션도 취해질 필요가 없다. 하지만, 프로세서가 현재 중지된 실행 모드에 있지 않다면, HTM은 현재의 특권 레벨을 체크할 수 있다(단계 515). 단계 515의 "아니오"로 표시되는 바와 같이, 특권 레벨이 상승된 레벨(예컨대, 커널-레벨 0, 혹은 몇몇 실시예에서는 어플리케이션 레벨이 아닌 레벨)이라면, 트랜잭셔널 실행 모드를 중지시키는 것에 관하여 그 어떤 후속 액션들도 취해질 필요가 없다.

[0055] 예시된 실시예에 따르면, 만일 프로세서가 중지된 트랜잭셔널 모드에 있지 않고(단계 510에서 "아니오" 로 표시됨) 그리고 현재 특권 레벨(CPL)이 가령 어플리케이션-레벨 3과 같은 하위 특권 레벨이라면(단계 515에서 "예" 로 표시됨), HTM은 콘트롤 양도 이벤트의 표시를 기록할 수 있으며(단계 520) 그리고 트랜잭셔널 실행 모드를 중지시킬 수 있다(단계 525).

[0056] 방법(502)에서, 트랜잭셔널 실행 모드가 단계 525에서 중지된 이후, HTM은 또 다른 콘트롤 양도를 검출할 수 있다(단계 530). 단계 530에서 검출된 이러한 후속 콘트롤 양도가 단계 505의 서스펜딩 콘트롤 양도에 대응하는 리턴 콘트롤 양도라면(단계 535에서 "예" 화살표로 표시됨), 프로세서는 실행을 어플리케이션 모드로 반환할 수 있으며 그리고 HTM은 트랜잭셔널 실행 모드를 재개(즉, 언-서스펜드(un-suspend))할 수 있다(단계 540). 하지만, 콘트롤 양도가 단계 505의 서스펜딩 콘트롤 양도에 대응하는 리턴 콘트롤 양도가 아니라면, HTM은 대응하는 리턴 콘트롤 양도가 발견될 때까지 트랜잭셔널 실행 모드를 계속할 수 있다(단계 535로부터 단계 530으로의 피드백 루프로 표시됨). 몇몇 경우에 있어서, 소정의 트랜잭션은 최종적으로 커미트되기 전에, 여러 번 중지 및/또는 재개될 수 있다(예컨대, 540으로부터 505으로의 선택적인 피드백 루프로 표현되는 바와 같이).

[0057] 다양한 실시예들에서는, 서스펜딩 양도와 그것의 대응 리턴 콘트롤 양도 사이에서 임의 개수의 콘트롤 양도가

실행될 수 있다. 하지만, 몇몇 실시예에서는, 서스펜딩 양도에 대응하는 리턴 콘트롤 양도가 검출되는 경우에만(단계 535의 "예" 화살표), HTM이 트랜잭셔널 모드를 재개할 것이다.

[0058] 도6은 그 동안에 중지가 실행될 수도 있는 콘트롤 양도의 다양한 유형들을 예시한 테이블이다. 예시된 콘트롤 양도의 유형들은, 파 콘트롤 양도(far control transfer), 시스템 콜들(system calls), 및 예외/인터럽트들을 포함한다. 예시된 각각의 중지 이벤트들에 의해서 양도의 각각의 유형이 검출될 수 있으며 그리고 예시된 각각의 재개 이벤트들에 의해서 양도로부터의 반환이 검출될 수 있다. 예를 들어, SYSENTER 명령을 검출함에 의해서 운영 시스템 안으로의 시스템 콜이 검출될 수 있으며 그리고 이에 매칭되는 리턴 콘트롤 양도는 SYSEXIT 명령에 의해서 표시될 것이다.

[0059] 도7은 본 발명의 실시예들에 따라 전술한 바와 같은 자동화된 중지/재개 기능을 구비한 하드웨어 트랜잭셔널 메모리를 구현하도록 구성된 컴퓨터 시스템을 예시한다. 컴퓨터 시스템(700)은 퍼스널 컴퓨터 시스템, 데스트탑 컴퓨터, 랩탑 혹은 노트북 컴퓨터, 메인프레임 컴퓨터 시스템, 휴대용 컴퓨터, 워크스테이션, 네트워크 컴퓨터, 컨슈머 디바이스, 어플리케이션 서버, 저장 디바이스, 주변 디바이스들(가령, 스위치, 모뎀, 라우터, 등등) 혹은 일반적인 임의 유형의 컴퓨팅 디바이스를 포함하는 임의의 유형의 디바이스가 될 수 있지만, 이에 한정되는 것은 아니다.

[0060] 컴퓨터 시스템(700)은 하나 이상의 프로세서들(750)을 포함할 수 있으며, 이들 각각은 다수개의 코어들을 포함할 수 있으며, 이들 중 임의의 것은 단일 혹은 다중 쓰레드화될 수 있다. 본 명세서 서술된 바와 같이, 각각의 프로세서는 다양한 플래그들 및 오퍼랜드들을 저장하는데 이용될 수 있는 레지스터들(752)을 포함할 수 있다. 프로세서(750)는, 액세스된 데이터를 캐시하도록 구성되고 및/또는 추론적 데이터 버퍼를 구현하도록 구성된 데이터 캐시(754)를 더 포함할 수 있다. 예를 들어, 데이터 캐시(754)는 각각의 캐시 블록에 관련되며 그리고 추론적으로 판독 및/또는 기입된 데이터를 나타내는데 이용될 수 있는 하나 이상의 비트들을 포함할 수 있다. 프로세서(750)는 콘트롤 양도들을 검출하는 것을 포함하여 중지/재개 조건들을 검출하기 위한 콘트롤 양도 검출 유닛(756)을 더 포함할 수 있다. 도7에 따르면, 프로세서(750)는 또한, 트랜잭셔널 실행 모드를 전술한 바와 같이 중지/재개하기 위한 트랜잭션 중지 유닛(758)을 포함할 수 있다.

[0061] 예시된 실시예에서, 컴퓨터 시스템(700)은 가령, 오프-칩 L2 혹은 L3 캐시와 같은 하나 이상의 오프-칩 캐시(760)를 또한 포함할 수 있다. 몇몇 실시예에서, L2 및/또는 L3 캐시는 온-칩으로 구현될 수도 있다. 또한, 컴퓨터 시스템(700)은 영속적인 저장 디바이스(770)(예컨대, 광 저장소자, 마그네틱 저장소자, 하드 드라이브, 테이프 드라이브, 솔리드 스테이트 메모리(solid state memory), 기타 등등)를 포함할 수 있으며, 이는 772와 같은 파일 시스템에 의해서 조직화된 데이터를 저장할 수 있다. 컴퓨터 시스템(700)은 또한, 임의 개수의 네트워크를 통하여 데이터를 전송 및 수신하기 위한 하나 이상의 네트워크 인터페이스들(가령, 780)을 포함할 수 있다. 컴퓨터 시스템(700)은 또한, 하나 이상의 메모리들(710)(예컨대, 하나 이상의 캐시들, SRAM, DRAM, RDRAM, EDO RAM, DDR 7 RAM, SDRAM, 램버스 RAM, EEPROM 등등)을 포함할 수 있다. 다양한 실시예들은 도7에 예시되지 않은 더 적은 구성요소들 혹은 추가 구성요소들(예컨대, 비디오 카드, 오디오 카드, 추가적인 네트워크 인터페이스, 주변 디바이스들, ATM 인터페이스와 같은 네트워크 인터페이스, 이더넷 인터페이스, 프레임 릴레이 인터페이스(Frame Relay interface) 등등)을 포함할 수 있다.

[0062] 하나 이상의 프로세서들(750), 저장 디바이스(들)(740), 오프-칩 캐시(760), 영속적인 저장 디바이스들(770), 네트워크 인터페이스(780) 및 시스템 메모리들(710)은 시스템 상호연결(740)을 통하여 접속될 수 있다. 하나 이상의 시스템 메모리들(710)은, 프로그램 명령들(720), 다양한 데이터 구조들 및 변수들(730)을 내포할 수 있다. 프로그램 명령들(720)은 프로그램 네이티브 바이너리(platform native binary), Java™ 바이트-코드와 같은 임의의 인터프리티드 언어(interpreted language) 혹은 가령, C/C++, 포트란 등등의 임의의 다른 언어들 혹은 이들의 임의의 조합으로 인코딩될 수 있다.

[0063] 프로그램 명령들(720)은 하나 이상의 싱글 및/또는 다중 쓰레드된 프로그램들(722)을 구현하도록 실행가능한 프로그램 명령들을 포함할 수 있는바, 이는 전술한 바와 같이 가령, 개시 및 커미트 명령들을 실행함에 의해서 원자적(atomic) 메모리 트랜잭션들을 이용할 수 있다.

[0064] 예시된 실시예에 따르면, 메모리(710)는 가령, 윈도우즈(Windows™) 및/또는 리눅스와 같은 운영 시스템(724)을 구현하도록 실행가능한 프로그램 명령들을 포함할 수 있다. 몇몇 실시예에서, 운영 시스템(724)은 상승된 특권 레벨에서 다양한 기능들을 실행하도록 구성된 커널을 포함할 수 있다. 예를 들어, 운영 시스템(724)은 어플리케이션들(722)이 인보크할 수도 있는 어플리케이션 프로그래밍 인터페이스(API) 정의 시스템 콜을 노출(expose)시

킬 수 있다. 이러한 시스템 콜들을 인보크하는 것에 응답하여, 프로세서(750)는 어플리케이션(722)에 의해서 실행중인 액티브 트랜잭션을 어플리케이션들(722) 및 운영 시스템(724)에 대해 실질적으로 투명한 방식으로 중지시킬 수 있다.

[0065] 어플리케이션들(722) 등과 같은 소프트웨어 프로그램들은 컴퓨터 프로그램 제품 혹은 소프트웨어로 제공될 수도 있는바, 이는 명령들이 저장되어 있는 컴퓨터-판독가능한 저장 매체를 포함할 수 있으며, 이는 본 발명의 다양한 실시예들에 따른 프로세스를 수행하도록 컴퓨터 시스템(혹은 다른 전자 디바이스들)을 프로그래밍하는데 이용될 수 있다. 컴퓨터 판독가능한 저장 매체는 머신(예컨대, 컴퓨터)에 의해서 판독가능한 형태(예컨대, 소프트웨어, 프로세싱 어플리케이션)로 정보를 저장하기 위한 임의의 매커니즘을 포함할 수 있다. 일시적이지 않은(non-transitory) 머신-판독가능한 저장 매체는 자기 저장 매체(예컨대, 플로피 디스크), 광 저장 매체(예컨대, CD-ROM), 랜덤 액세스 메모리(RAM), 소거 및 프로그램가능한 메모리(예컨대, EPROM 및 EEPROM), 플래시 메모리, 혹은 프로그램 명령들을 저장하기에 적절한 다른 유형의 매체들을 포함할 수 있으나, 이에 한정되는 것은 아니다. 또한, 프로그램 명령들은, 광(optical), 음향(acoustical) 혹은 다른 형태의 전파된 신호들(예컨대, 캐리어 웨이브들, 적외선 신호들, 디지털 신호들 등등)을 이용하여 통신될 수 있다.

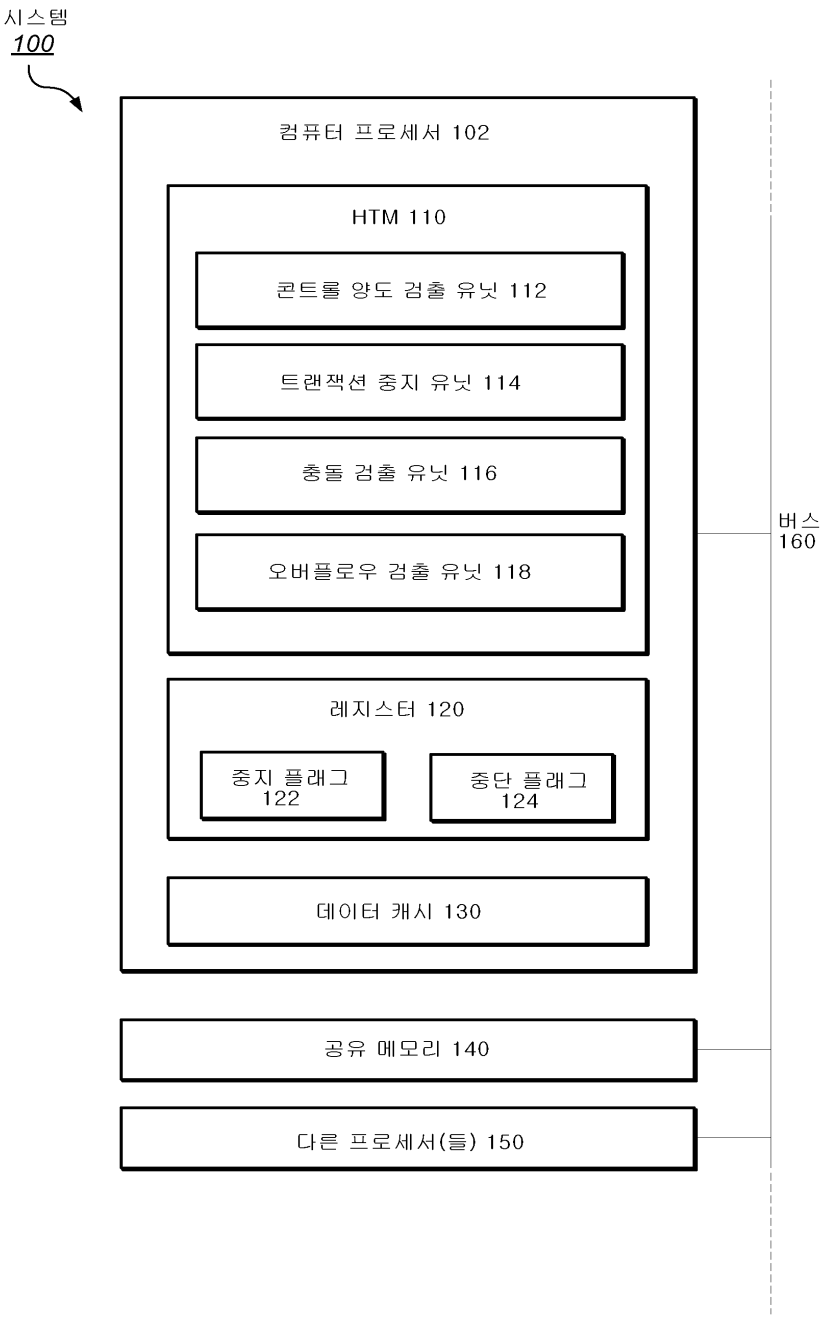
[0066] 일실시예에서, 전술한 바와 같은 컴퓨터-판독가능한 저장 매체는 프로그램에 의해서 판독된 명령들을 저장하는데 이용될 수 있으며 그리고 시스템 프로세서(750)를 포함하는 하드웨어를 제작하는데 직접적으로 혹은 간접적으로 이용될 수 있다. 예를 들어, 상기 명령들은 하드웨어 기능들의 거동-레벨(behavioral-level) 혹은 레지스터-트랜스퍼 레벨(register-transfer level:RTL) 서술을 Verilog 혹은 VHDL 등과 같은 고레벨 설계 언어(HDL)로 서술하는 하나 이상의 데이터 스트럭처들을 서술할 수 있다. 이러한 서술(description)은 합성 툴에 의해서 판독될 수 있으며, 합성 툴(synthesis tool)은 네트리스트(netlist)를 생성하도록 상기 서술을 합성할 수 있다. 네트리스트는 게이트들의 세트(예컨대, 합성 라이브러리(synthesis library)에서 정의되는)를 포함할 수 있으며, 이는 프로세서(750)의 기능들을 나타낸다. 이후 네트리스트는, 적용될 마스크들의 기하학적 형상들을 서술하는 데이터 세트를 생성하도록 배치 및 라우팅될 수 있다. 이후 상기 마스크가 다양한 반도체 제조 단계들에서 이용되어, 프로세서(750)에 대응하는 반도체 회로 혹은 회로들을 제작할 수 있다. 대안적으로는, 캐리어 매체(300) 상의 데이터베이스가 네트리스트(합성 라이브가 있거나 혹은 없거나) 혹은 데이터 세트가 될 수 있다.

[0067] 본 개시내용의 범위는 본 명세서에서 설명된 문제점들의 일부 혹은 전부를 완화시키는지의 여부에 상관없이, 본 명세서에 개시된(명시적으로 혹은 암시적으로) 임의의 피처 혹은 피처들의 조합 혹은 이들의 임의의 일반화를 포함한다. 따라서, 본 출원(혹은 우선권을 주장하는 출원)에 대한 절차가 진행되는 동안에 이러한 피처들의 임의의 조합에 대한 새로운 청구항이 만들어질 수도 있다. 특히, 첨부된 청구범위에 대해서, 종속항들로부터의 피처들은 독립항의 피처들과 조합될 수 있으며 그리고 각각의 독립항으로부터 피처들은 임의의 적절한 방식으로 그리고 첨부된 청구항들에 열거된 특정 조합 이외의 방식으로 조합될 수 있다.

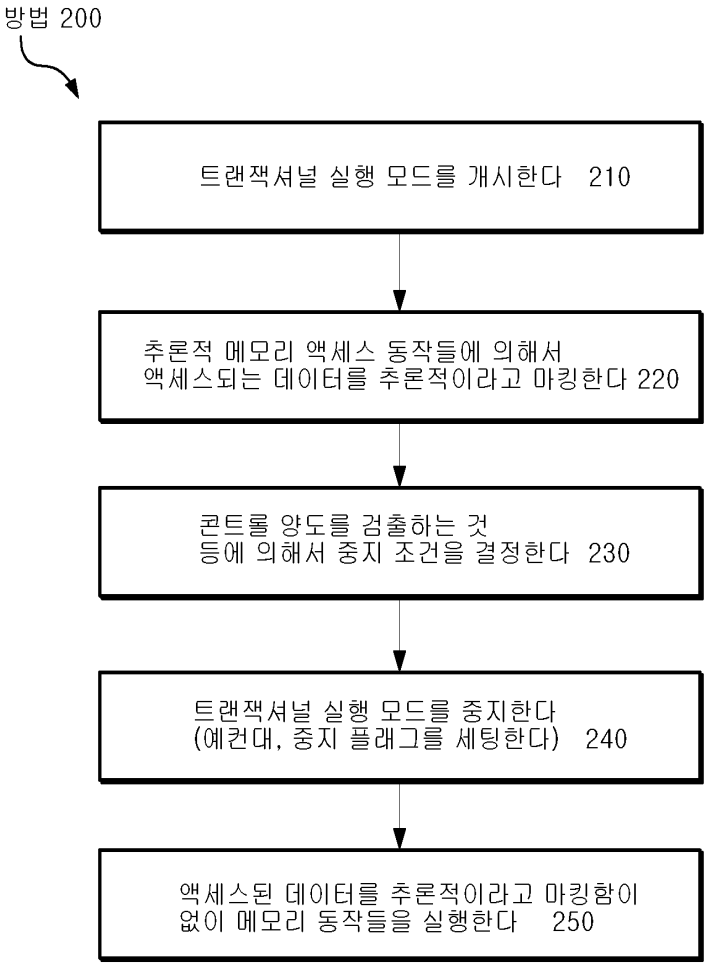
[0068] 비록, 전술한 실시예들이 상당히 상세하게 서술되었지만, 앞선 개시 내용이 완전히 이해된다면 다양한 변형예들 및 수정예들이 해당 기술분야의 당업자에게 명백할 것이다. 다음의 청구범위는 이러한 모든 변형예들과 수정예들을 포괄하도록 해석되어야만 한다.

도면

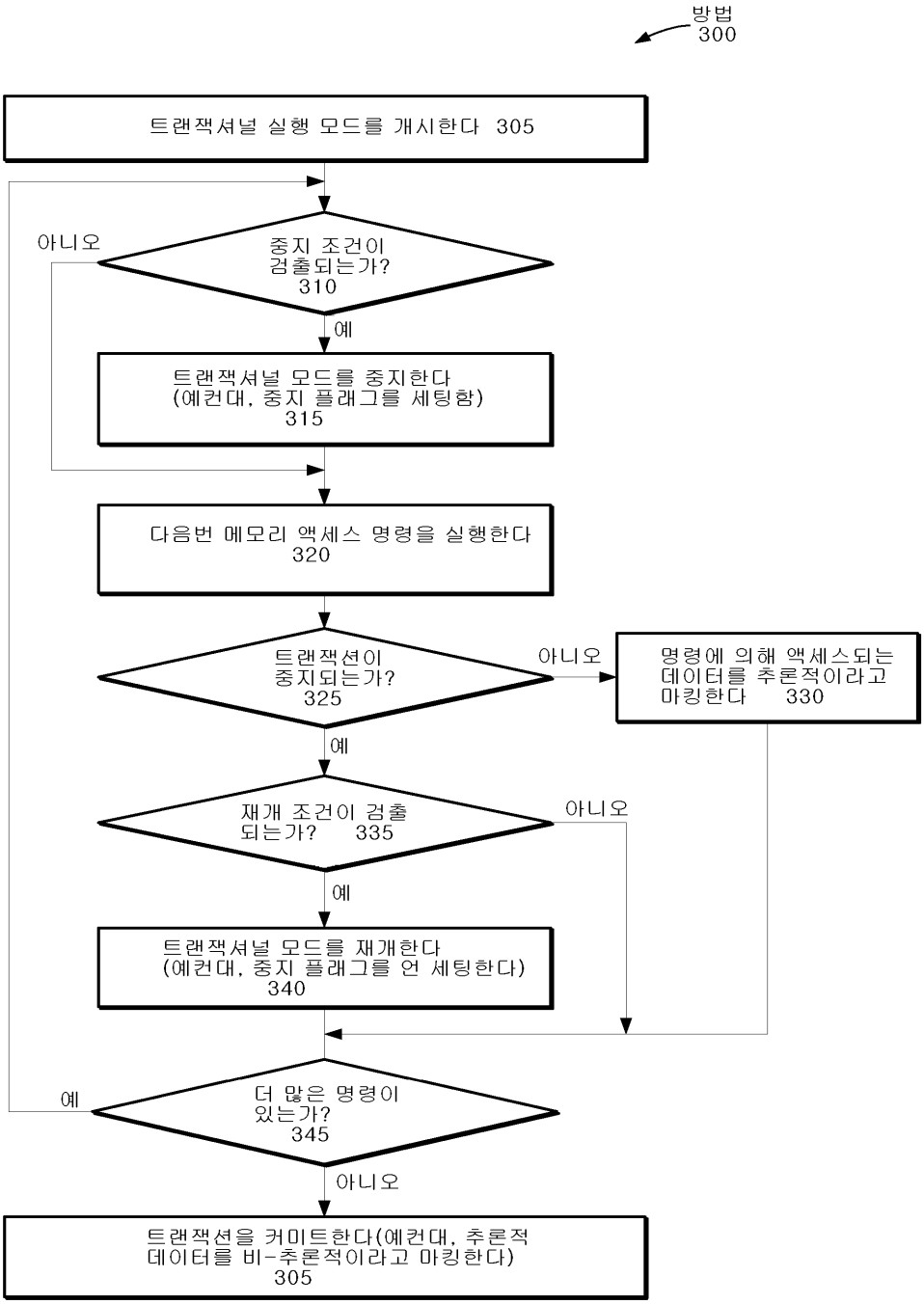
도면1


시스템
100
컴퓨터 프로세서 102
HTM 110
콘트롤 양도 검출 유닛 112
트랜잭션 중지 유닛 114
충돌 검출 유닛 116
오버플로우 검출 유닛 118
레지스터 120
중지 플래그
122
중단 플래그
124
데이터 캐시 130
버스
160
공유 메모리 140
다른 프로세서(들) 150

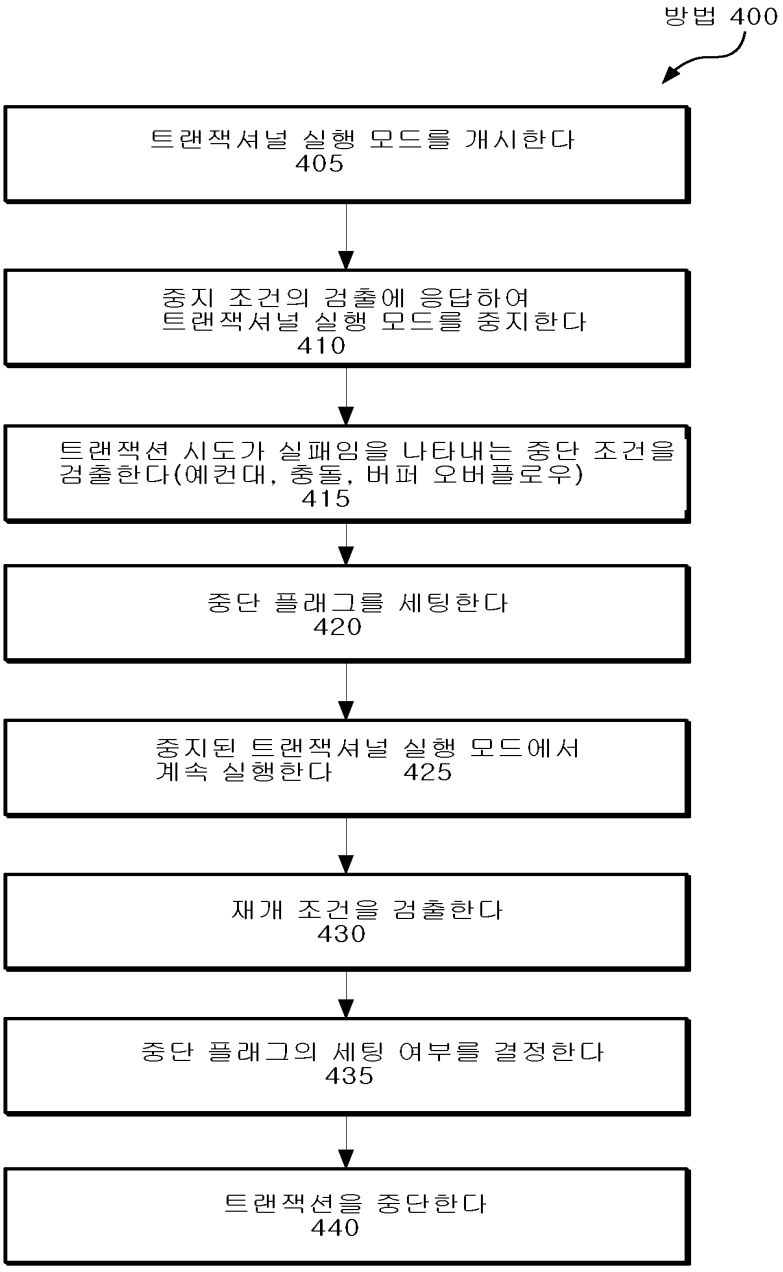
도면2


방법 200
트랜잭셔널 실행 모드를 개시한다 210
추론적 메모리 액세스 동작들에 의해서
액세스되는 데이터를 추론적이라고 마킹한다 220
콘트롤 양도를 검출하는 것
등에 의해서 중지 조건을 결정한다 230
트랜잭셔널 실행 모드를 중지한다
(예컨대, 중지 플래그를 세팅한다) 240
액세스된 데이터를 추론적이라고 마킹함이
없이 메모리 동작들을 실행한다 250

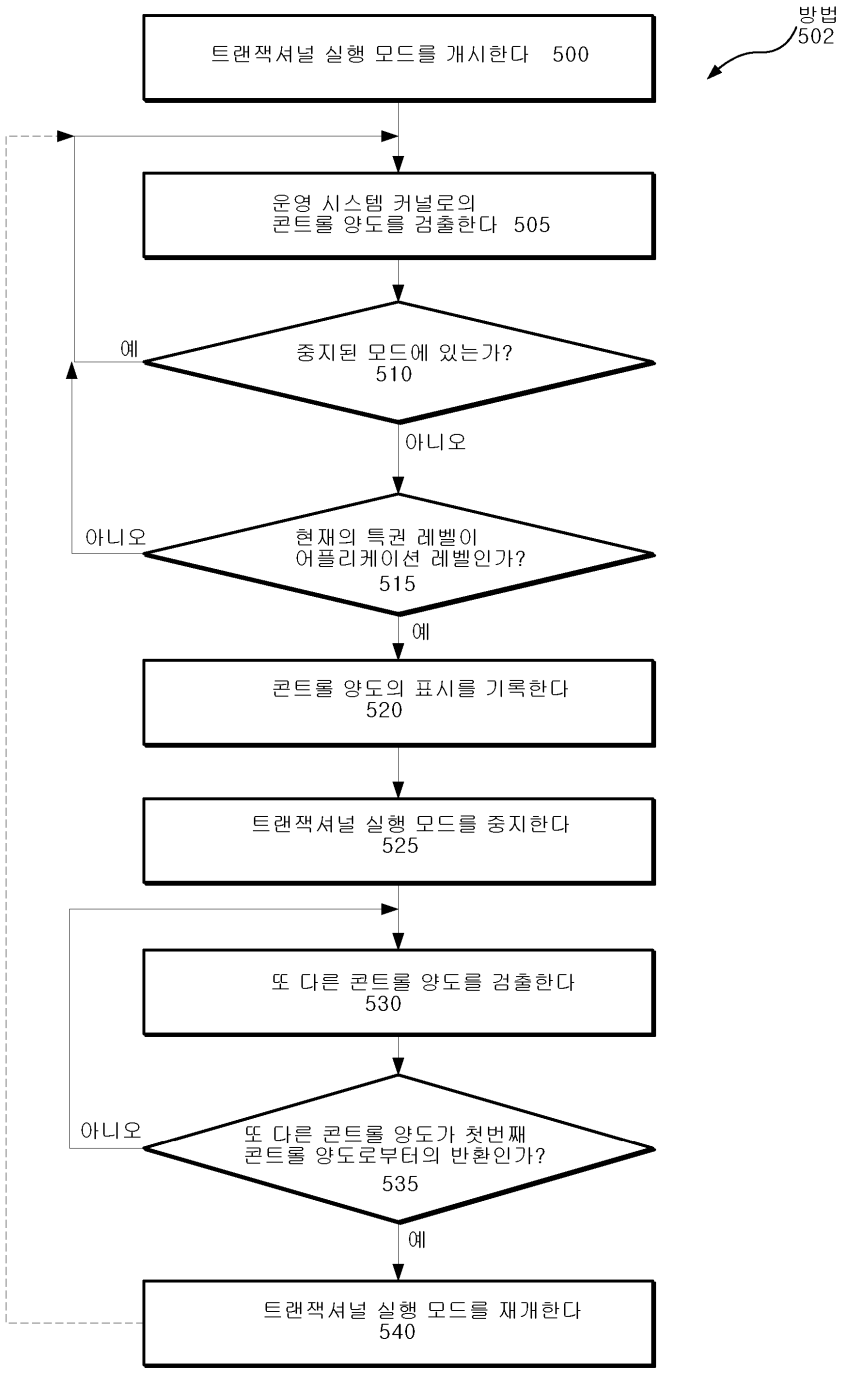
도면3


방법
300
트랜잭셔널 실행 모드를 개시한다 305
중지 조건이
검출되는가?
310
아니오
예
트랜잭셔널 모드를 중지한다
(예컨대, 중지 플래그를 세팅함)
315
다음번 메모리 액세스 명령을 실행한다
320
트랜잭션이
중지되는가?
325
아니오
명령에 의해 액세스되는
데이터를 추론적이라고
마킹한다 330
예
재개 조건이 검출
되는가? 335
아니오
예
트랜잭셔널 모드를 재개한다
(예컨대, 중지 플래그를 언 세팅한다)
340
더 많은 명령이
있는가?
345
예
아니오
트랜잭션을 커미트한다(예컨대, 추론적
데이터를 비-추론적이라고 마킹한다)
305

도면4


방법 400
트랜잭셔널 실행 모드를 개시한다
405
중지 조건의 검출에 응답하여
트랜잭셔널 실행 모드를 중지한다
410
트랜잭션 시도가 실패임을 나타내는 중단 조건을
검출한다(예컨대, 충돌, 버퍼 오버플로우)
415
중단 플래그를 세팅한다
420
중지된 트랜잭셔널 실행 모드에서
계속 실행한다
425
재개 조건을 검출한다
430
중단 플래그의 세팅 여부를 결정한다
435
트랜잭션을 중단한다
440

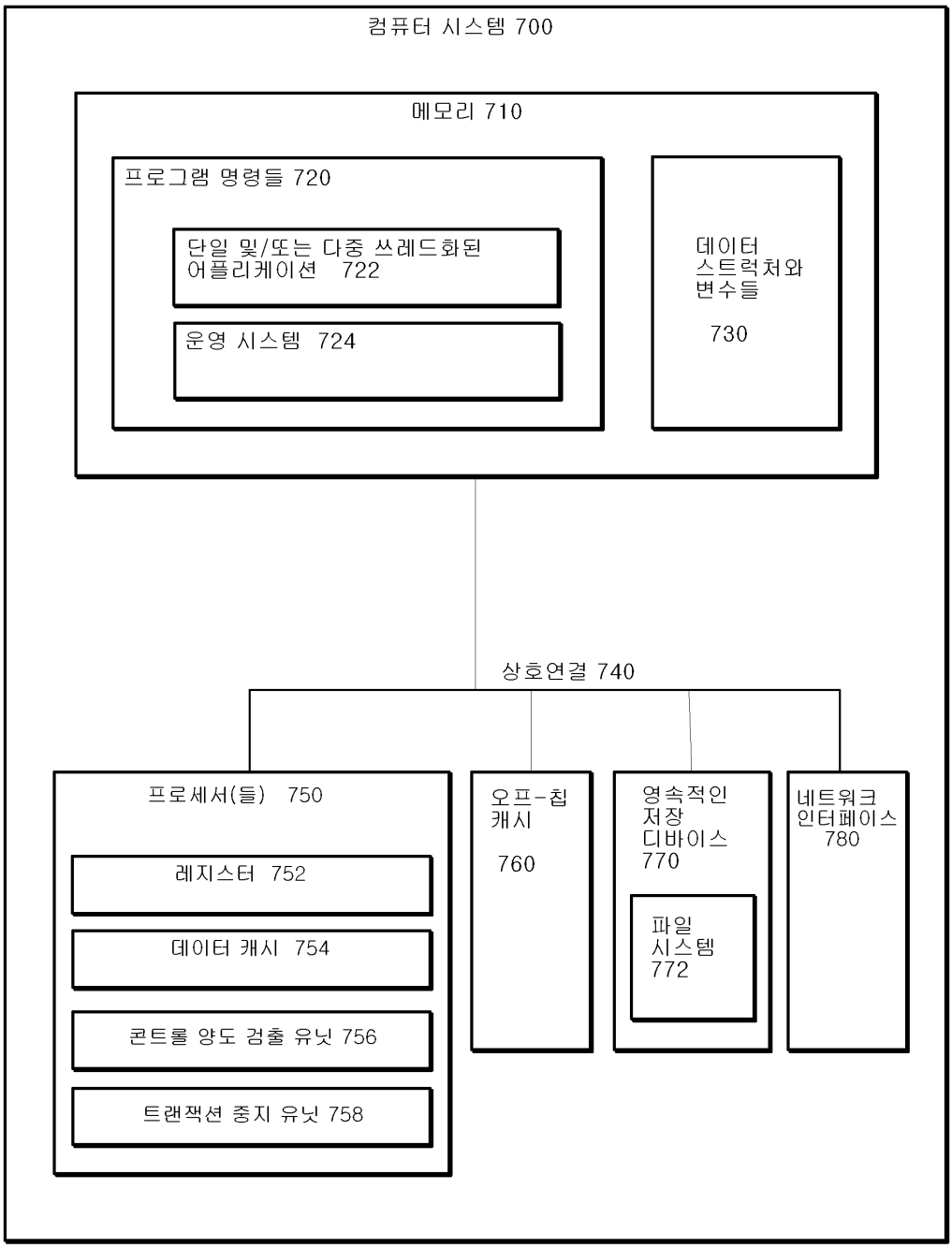
도면5



도면6

양도의 유형	중지 이벤트들	재개 이벤트들
Far control transfer	CALL	RET
System call	SYSCALL, SYSENTER	SYSRET, SYSEXIT
Exception/Interrupt	INT, INTn, exception, interrupt	IRET, RSM

도면7



【심사관 직권보정사항】

【직권보정 1】

【보정항목】 명세서

【보정세부항목】 식별번호 [0053]

【변경전】

드롭(droppomg)

【변경후】

드롭(drop)