(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0073376
(43) 공개일자 2025년05월27일

(51) 국제특허분류(Int. Cl.)
G06F 9/48 (2018.01) *G06F 9/38* (2024.01)
(52) CPC특허분류
G06F 9/4831 (2013.01)
G06F 9/3861 (2013.01)
(21) 출원번호 10-2025-7013386
(22) 출원일자(국제) 2023년08월01일
심사청구일자 없음
(85) 번역문제출일자 2025년04월23일
(86) 국제출원번호 PCT/GB2023/052035
(87) 국제공개번호 WO 2024/069120
국제공개일자 2024년04월04일
(30) 우선권주장
2214155.0 2022년09월28일 영국(GB)

(71) 출원인
에이알엠 리미티드
영국 캠브리지 씨비1 9엔제이 체리힌톤 풀번로드 110
(72) 발명자
윌리암스, 마이클 존
영국 씨비1 9엔제이 캠브리지시어 캠브리지 풀본로드 110 에이알엠 리미티드 씨/오
(74) 대리인
남호현

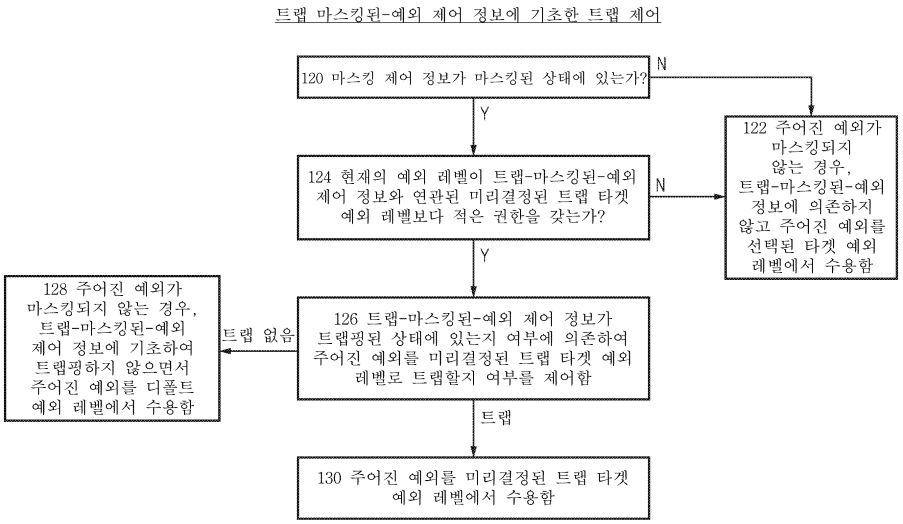
전체 청구항 수 : 총 19 항

(54) 발명의 명칭 **예외 제어**

(57) 요 약

예외 제어 회로부(40)는 적어도 하나의 레지스터(14)에 저장된 제어 정보에 의존하여 프로세싱 회로부(4)에 의한 예외의 수용을 제어하고, 제어 정보는 마스킹된 상태 또는 마스킹되지 않은 상태로 설정 가능한 마스킹 제어 정보; 및 트랩핑되지 않은 상태 또는 트랩핑된 상태로 설정 가능한 트랩-마스킹된-예외 제어 정보를 포함한다. 마스킹 가능 클래스의 예외 중 주어진 예외에 응답하여, 마스킹 제어 정보가 상기 마스킹된 상태에 있고 현재의 예외 레벨이 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖는 적어도 하나의 시나리오에서, 예외 제어 회로부는 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있는지 여부에 의존하여 주어진 예외를 미리결정된 트랩 타겟 예외 레벨로 트랩할지 여부를 제어한다. 마스킹 제어 정보가 마스킹되지 않은 상태에 있는 경우에, 주어진 예외를 핸들링하기 위한 타겟 예외 레벨은 트랩-마스킹된-예외 제어 정보에 의존하지 않고 선택된다.

대 표 도 - 도5



(52) CPC특허분류
G06F 2209/481 (2013.01)

명 세 서

청구범위

청구항 1

장치로서,

프로세싱 회로부에 의해 지원되는 복수의 예외 레벨 중 하나에서 데이터 프로세싱을 수행하기 위한 상기 프로세싱 회로부; 및

적어도 하나의 레지스터에 저장된 제어 정보에 의존하여, 상기 프로세싱 회로부에 의한 예외의 수용을 제어하기 위한 예외 제어 회로부를 포함하고, 상기 제어 정보는,

마스킹된 상태와 마스킹되지 않은 상태 중 하나로 설정 가능한 마스킹 제어 정보; 및

소프트웨어에 의해 트랩핑되지 않은 상태와 트랩핑된 상태 중 하나로 설정 가능한 트랩-마스킹된-예외 제어 정보를 적어도 포함하고;

여기서

상기 마스킹 제어 정보가 상기 마스킹된 상태에 있는 것에 기초하여 마스킹 가능한 마스킹 가능 클래스의 예외 중 주어진 예외에 응답하여, 상기 예외 제어 회로부는,

상기 마스킹 제어 정보가 상기 마스킹된 상태에 있는지 여부를 결정하고;

상기 마스킹 제어 정보가 상기 마스킹된 상태에 있고 현재의 예외 레벨이 상기 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖는 경우에 상기 주어진 예외가 발생하는 적어도 하나의 시나리오에서, 상기 트랩-마스킹된-예외 제어 정보가 상기 트랩핑된 상태에 있는지 여부에 의존하여 상기 주어진 예외를 상기 미리결정된 트랩 타겟 예외 레벨로 트랩할지 여부를 제어하고;

상기 마스킹 제어 정보가 상기 마스킹되지 않은 상태에 있고 상기 주어진 예외가 마스킹되지 않는 것으로 결정되는 경우에, 상기 주어진 예외를 핸들링하기 위한 타겟 예외 레벨을 선택하도록 - 상기 타겟 예외 레벨이 상기 트랩-마스킹된-예외 제어 정보에 의존하지 않고 선택됨 - 구성되는, 장치.

청구항 2

제1항에 있어서,

상기 제어 정보는 복수의 미리결정된 트랩 타겟 예외 레벨 중 각자의 것과 각각 연관된 트랩-마스킹된-예외 제어 정보의 복수의 항목을 포함하고,

상기 예외 제어 회로부는, 상기 마스킹 제어 정보가 상기 마스킹된 상태에 있고 상기 현재의 예외 레벨이 상기 복수의 미리결정된 트랩 타겟 예외 레벨 중 주어진 것보다 더 적은 권한을 갖는 경우에 상기 주어진 예외가 발생하는 적어도 하나의 시나리오에서, 상기 복수의 미리결정된 트랩 타겟 예외 레벨 중 해당 주어진 것과 연관된 트랩-마스킹된-예외 제어 정보의 대응하는 항목이 상기 트랩핑된 상태에 있는지 여부에 의존하여, 상기 주어진 예외가 상기 복수의 미리결정된 트랩 타겟 예외 레벨 중 상기 주어진 것으로 트랩핑되는지 여부를 제어하도록 구성되는, 장치.

청구항 3

제2항에 있어서, 상기 마스킹 제어 정보가 상기 마스킹된 상태에 있고, 트랩-마스킹된-예외 제어 정보의 적어도 2개의 항목이 상기 트랩핑된 상태에 있고, 트랩-마스킹된-예외 제어 정보의 상기 적어도 2개의 항목 각각에 대해 상기 미리결정된 트랩 타겟 예외 레벨로 트랩핑하기 위한 하나 이상의 조건이 만족되는 경우에 상기 주어진 예외가 발생하는 적어도 하나의 시나리오에서:

상기 예외 제어 회로부는 상기 주어진 예외를, 트랩-마스킹된-예외 제어 정보의 선택된 항목과 연관된 상기 미리결정된 트랩 타겟 예외 레벨로 트랩하도록 구성되고;

트랩-마스킹된-예외 제어 정보의 상기 선택된 항목은, 상기 현재의 예외 레벨과 상기 연관된 미리결정된 트랩 타겟 예외 레벨 사이에서 권한의 차이가 가장 작은 트랩-마스킹된-예외 제어 정보의 상기 적어도 2개의 항목 중 하나를 포함하는, 장치.

청구항 4

제1항 내지 제3항 중 어느 한 항에 있어서, 상기 트랩-마스킹된-예외 제어 정보가 상기 트랩핑되지 않은 상태에 있는 경우에, 상기 예외 제어 회로부는 상기 마스킹 제어 정보가 상기 마스킹된 상태에 있는지 여부에 적어도 의존하여 상기 주어진 예외가 마스킹되는지 여부를 결정하도록 구성되는, 장치.

청구항 5

제4항에 있어서, 상기 예외 제어 회로부는 또한, 상기 트랩-마스킹된-예외 제어 정보가 상기 트랩핑되지 않은 상태에 있는 경우 상기 예외 레벨이 수용되면, 상기 현재의 예외 레벨이 상기 주어진 예외가 수용될 것인 예외 레벨보다 더 많은 권한을 갖는지 여부에 의존하여 상기 주어진 예외가 마스킹될지 여부를 결정하도록 구성되는, 장치.

청구항 6

제4항 또는 제5항에 있어서,

상기 제어 정보는, 상기 마스킹 가능 클래스의 예외가 마스킹 가능한지 여부를 표시하기 위해 비-마스킹 가능 상태와 마스킹 가능 상태 중 하나로 설정 가능한 비-마스킹 가능-예외-제어 정보를 포함하고;

상기 비-마스킹 가능-예외-제어 정보가 상기 비-마스킹 가능 상태에 있는 적어도 하나의 시나리오에서, 상기 예외 제어 회로부는, 상기 마스킹 제어 정보가 상기 마스킹된 상태로 설정되는 경우에도, 상기 주어진 예외가 마스킹되지 않는 것으로 결정하도록 구성되는, 장치.

청구항 7

제6항에 있어서, 상기 현재의 예외 레벨이 최소 권한을 갖는 예외 레벨이고, 상기 비-마스킹 가능-예외 제어 정보가 상기 비-마스킹 가능 상태에 있고, 상기 주어진 예외가 마스킹되지 않고 상기 트랩-마스킹된-예외 제어 정보가 상기 트랩핑되지 않은 상태에 있는 경우에 상기 주어진 예외를 수용하기 위해 활성화되는 다음 최소 권한을 갖는 예외 레벨로 상기 주어진 예외가 수용될 것인 경우에 상기 주어진 예외가 발생하는 시나리오에서, 상기 예외 제어 회로부는,

상기 트랩-마스킹된-예외 제어 정보가 상기 트랩핑된 상태에 있는 경우에도 상기 미리결정된 트랩 타겟 예외 레벨로 트랩핑하지 않으면서, 상기 다음 최소 권한을 갖는 예외 레벨에서 상기 주어진 예외의 수용을 제어하도록 구성되는, 장치.

청구항 8

제7항에 있어서, 상기 제어 정보는 상이한 예외 레벨과 연관된 비-마스킹 가능-예외 제어 정보의 복수의 항목을 포함하고;

상기 예외 제어 회로부는 상기 현재의 예외 레벨에 기초하여 선택된 비-마스킹 가능-예외-제어 정보의 상기 복수의 항목 중 하나에 의존하여 상기 주어진 예외가 마스킹되는지 여부를 결정하도록 구성되는, 장치.

청구항 9

제1항 내지 제8항 중 어느 한 항에 있어서, 상기 제어 정보는 소프트웨어에 의해 트랩핑된 상태와 트랩핑되지 않은 상태 중 하나로 설정 가능한 트랩 제어 정보의 적어도 하나의 추가 항목을 포함하고;

상기 예외 제어 회로부는, 트랩 제어 정보의 상기 추가 항목이 상기 트랩핑된 상태에 있는지 여부에 의존하여, 적어도 상기 마스킹 가능 클래스의 예외를 특정 예외 레벨로 트랩할지 여부를 제어하도록 구성되고;

트랩 제어 정보의 상기 적어도 하나의 추가 항목은 상기 트랩-마스킹된-예외 제어 정보보다 우선하고, 그에 따라 트랩 제어 정보의 상기 추가 항목이 상기 트랩핑된 상태에 있는 경우에, 상기 예외 제어 회로부는, 상기 트랩-마스킹된-예외 제어 정보가 상기 트랩핑된 상태에 있는지 여부에 관계없이, 트랩 제어 정보의 상기 추가 항

목에 기초하여 상기 예외를 상기 특정 예외 레벨로 트랩하도록 구성되는, 장치.

청구항 10

제1항 내지 제9항 중 어느 한 항에 있어서, 상기 마스킹 가능 클래스의 예외는 정확한 서비스로부터의 편차를 표시하는 비동기 에러 예외를 포함하는, 장치.

청구항 11

제1항 내지 제10항 중 어느 한 항에 있어서, 상기 마스킹 제어 정보를 상기 마스킹된 상태로 설정하는 것에 기초하여 마스킹될 수 없는 비-마스킹 가능 클래스의 예외 중 비-마스킹 가능 예외에 응답하여, 상기 예외 제어 회로부는,

상기 마스킹 제어 정보가 상기 마스킹된 상태에 있고, 현재의 예외 레벨이 상기 트랩-마스킹된-예외 제어 정보와 연관된 상기 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖고 상기 현재의 예외 레벨이 예외를 핸들링하기 위한 타겟 예외 레벨일 수 있는 예외 레벨인 경우에 상기 주어진 예외가 발생하는 적어도 하나의 시나리오에서, 상기 트랩-마스킹된-예외 제어 정보가 상기 트랩핑된 상태에 있는지 여부에 의존하여 상기 비-마스킹 가능 예외를 상기 미리결정된 트랩 타겟 예외 레벨로 트랩할지 여부를 제어하도록 구성되는, 장치.

청구항 12

제11항에 있어서, 상기 비-마스킹 가능 클래스의 예외는 정확한 서비스로부터의 편차를 표시하는 동기 외부 중단 예외를 포함하는, 장치.

청구항 13

제1항 내지 제12항 중 어느 한 항에 있어서, 하나 이상의 예외 신드롬 레지스터의 복수의 세트를 포함하고, 하나 이상의 예외 신드롬 레지스터의 각각의 세트는 상기 예외 레벨 중 각자의 것에서 수용된 예외에 대한 예외 신드롬 정보를 저장하는, 장치.

청구항 14

제1항 내지 제13항 중 어느 한 항에 있어서, 상기 트랩-마스킹된-예외 제어 정보와 연관된 상기 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖는 예외 레벨에서 실행되는 상기 트랩-마스킹된-예외 제어 정보에 대한 업데이트를 요청하는 명령어에 응답하여, 상기 프로세싱 회로부는 상기 트랩-마스킹된-예외 제어 정보에 대한 상기 요청된 업데이트를 거부하거나 트랩하도록 구성되는, 장치.

청구항 15

제1항 내지 제14항 중 어느 한 항의 장치의 제조를 위한, 컴퓨터-판독 가능 코드.

청구항 16

제15항의 컴퓨터-판독 가능 코드를 저장하는, 저장 매체.

청구항 17

방법으로서,

상기 프로세싱 회로부에 의해 지원되는 복수의 예외 레벨 중 하나에서 데이터 프로세싱을 수행하는 단계; 및

적어도 하나의 레지스터에 저장된 제어 정보에 의존하여, 상기 프로세싱 회로부에 의한 예외의 수용을 제어하는 단계를 포함하고, 상기 제어 정보는,

마스킹된 상태와 마스킹되지 않은 상태 중 하나로 설정 가능한 마스킹 제어 정보; 및

소프트웨어에 의해 트랩핑되지 않은 상태와 트랩핑된 상태 중 하나로 설정 가능한 트랩-마스킹된-예외 제어 정보를 적어도 포함하고;

여기서

상기 마스킹 제어 정보를 상기 마스킹된 상태로 설정하는 것에 기초하여 마스킹 가능한 마스킹 가능 클래스의

예외 중 주어진 예외에 응답하여, 상기 방법은,

상기 마스킹 제어 정보가 상기 마스킹된 상태에 있는지 여부를 결정하는 단계;

상기 마스킹 제어 정보가 상기 마스킹된 상태에 있고 현재의 예외 레벨이 상기 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖는 경우에 상기 주어진 예외가 발생하는 적어도 하나의 시나리오에서, 상기 트랩-마스킹된-예외 제어 정보가 상기 트랩핑된 상태에 있는지 여부에 의존하여 상기 주어진 예외를 상기 미리결정된 트랩 타겟 예외 레벨로 트랩할지 여부를 제어하는 단계; 및

상기 마스킹 제어 정보가 상기 마스킹되지 않은 상태에 있고 상기 주어진 예외가 마스킹되지 않는 것으로 결정되는 경우에, 상기 주어진 예외를 핸들링하기 위한 타겟 예외 레벨을 선택하는 단계 - 상기 타겟 예외 레벨이 상기 트랩-마스킹된-예외 제어 정보에 의존하지 않고 선택됨 - 를 포함하는, 방법.

청구항 18

호스트 데이터 프로세싱 장치에 의해 실행될 때, 타겟 코드를 실행시키기 위한 명령어 실행 환경을 제공하기 위해 상기 호스트 데이터 프로세싱 장치를 제어하는 명령어를 포함하는 컴퓨터 프로그램으로서, 상기 컴퓨터 프로그램은,

복수의 예외 레벨 중 하나에서 상기 타겟 코드의 명령어의 실행을 시뮬레이션하기 위한 프로세싱 프로그램 로직; 및

상기 타겟 코드에 의해 사용되는 타겟 명령어 세트 아키텍처의 적어도 하나의 레지스터를 나타내는 상기 호스트 데이터 프로세싱 장치의 적어도 하나의 저장 위치에 저장된 제어 정보에 의존하여, 예외의 수용을 시뮬레이션하기 위한 예외 제어 프로그램 로직을 포함하고, 상기 제어 정보는,

마스킹된 상태와 마스킹되지 않은 상태 중 하나로 설정 가능한 마스킹 제어 정보; 및

소프트웨어에 의해 트랩핑되지 않은 상태와 트랩핑된 상태 중 하나로 설정 가능한 트랩-마스킹된-예외 제어 정보를 적어도 포함하고;

여기서

상기 마스킹 제어 정보를 상기 마스킹된 상태로 설정하는 것에 기초하여 마스킹 가능한 마스킹 가능 클래스의 예외 중 주어진 예외에 응답하여, 상기 예외 제어 프로그램 로직은,

상기 마스킹 제어 정보가 상기 마스킹된 상태에 있는지 여부를 결정하고;

상기 마스킹 제어 정보가 상기 마스킹된 상태에 있고 현재의 예외 레벨이 상기 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖는 경우에 상기 주어진 예외가 발생하는 적어도 하나의 시나리오에서, 상기 트랩-마스킹된-예외 제어 정보가 상기 트랩핑된 상태에 있는지 여부에 의존하여 상기 주어진 예외를 상기 미리결정된 트랩 타겟 예외 레벨로 트랩할지 여부를 제어하고;

상기 마스킹 제어 정보가 상기 마스킹되지 않은 상태에 있고 상기 주어진 예외가 마스킹되지 않는 것으로 결정되는 경우에, 상기 주어진 예외를 핸들링하기 위한 타겟 예외 레벨을 선택하도록 - 상기 타겟 예외 레벨이 상기 트랩-마스킹된-예외 제어 정보에 의존하지 않고 선택됨 - 구성되는, 컴퓨터 프로그램.

청구항 19

제18항의 컴퓨터 프로그램을 저장하는, 저장 매체.

발명의 설명

기 술 분 야

[0001] 본 기법은 데이터 프로세싱의 분야에 관한 것이다. 더욱 상세하게는, 예외 핸들링에 관한 것이다.

배 경 기 술

[0002] 데이터 프로세싱을 수행하는 동안, 프로세싱 회로부는 예외를 마주할 수 있으며, 예외는 프로그램 실행의 통상 흐름이 인터럽트되는 것을 필요로 하고, 그에 따라 프로세싱이 예외 핸들러로 스위칭하여 예외를 야기한 이벤트

를 처리할 수 있는 이벤트이다. 예를 들어, 예외는 여러 이유로, 예컨대 정의되지 않은 명령어의 실행, 가상-물리 어드레스 변환 맵핑이 정의되지 않거나 메모리 액세스 권한 체크가 실패한 메모리 액세스로 인해 발생하는 어드레스 장애, 또는 메모리 시스템 에러 또는 외부 디바이스 또는 주변장치에 의해 시그널링된 인터럽트 이벤트의 검출과 같은 외부 이벤트의 발생으로 인해 일어날 수 있다.

발명의 내용

[0003] 적어도 일부 예는 장치를 제공하고, 장치는, 프로세싱 회로부에 의해 지원되는 복수의 예외 레벨 중 하나에서 데이터 프로세싱을 수행하기 위한 상기 프로세싱 회로부; 및 적어도 하나의 레지스터에 저장된 제어 정보에 의존하여, 프로세싱 회로부에 의한 예외의 수용을 제어하는 예외 제어 회로부를 포함하고, 제어 정보는 마스킹된 상태와 마스킹되지 않은 상태 중 하나로 설정 가능한 마스킹 제어 정보; 및 소프트웨어에 의해 트랩핑되지 않은 상태와 트랩핑된 상태 중 하나로 설정 가능한 트랩-마스킹된-예외 제어 정보를 적어도 포함하고; 여기서, 마스킹 제어 정보를 마스킹된 상태로 설정하는 것에 기초하여 마스킹 가능한 마스킹 가능 클래스의 예외 중 주어진 예외에 응답하여, 예외 제어 회로부는 마스킹 제어 정보가 마스킹된 상태에 있는지 여부를 결정하고; 마스킹 제어 정보가 마스킹된 상태에 있고 현재의 예외 레벨이 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖는 경우에 주어진 예외가 발생하는 적어도 하나의 시나리오에서, 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있는지 여부에 의존하여 주어진 예외를 미리결정된 트랩 타겟 예외 레벨로 트랩할지 여부를 제어하고; 마스킹 제어 정보가 마스킹되지 않은 상태에 있고 주어진 예외가 마스킹되지 않는 것으로 결정되는 경우에, 주어진 예외를 핸들링하기 위한 타겟 예외 레벨을 선택하도록 - 타겟 예외 레벨이 트랩-마스킹된-예외 제어 정보에 의존하지 않고 선택됨 - 구성된다.

[0004] 적어도 일부 예는 상술한 장치의 제조를 위한 컴퓨터-판독 가능 코드를 제공한다. 컴퓨터-판독 가능 코드는 저장 매체 상에 저장될 수 있다. 저장 매체는 일시적 또는 비일시적 저장 매체일 수 있다.

[0005] 적어도 일부 예는 방법을 제공하고, 방법은 프로세싱 회로부에 의해 지원되는 복수의 예외 레벨 중 하나에서 데이터 프로세싱을 수행하는 단계; 및 적어도 하나의 레지스터에 저장된 제어 정보에 의존하여, 프로세싱 회로부에 의한 예외의 수용을 제어하는 단계를 포함하고, 제어 정보는 마스킹된 상태와 마스킹되지 않은 상태 중 하나로 설정 가능한 마스킹 제어 정보; 및 소프트웨어에 의해 트랩핑되지 않은 상태와 트랩핑된 상태 중 하나로 설정 가능한 트랩-마스킹된-예외 제어 정보를 적어도 포함하고; 여기서, 마스킹 제어 정보를 마스킹된 상태로 설정하는 것에 기초하여 마스킹 가능한 마스킹 가능 클래스의 예외 중 주어진 예외에 응답하여, 상기 방법은 마스킹 제어 정보가 마스킹된 상태에 있는지 여부를 결정하는 단계; 마스킹 제어 정보가 마스킹된 상태에 있고 현재의 예외 레벨이 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖는 경우에 주어진 예외가 발생하는 적어도 하나의 시나리오에서, 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있는지 여부에 의존하여 주어진 예외를 미리결정된 트랩 타겟 예외 레벨로 트랩할지 여부를 제어하는 단계; 마스킹 제어 정보가 마스킹되지 않은 상태에 있고 주어진 예외가 마스킹되지 않는 것으로 결정되는 경우에, 주어진 예외를 핸들링하기 위한 타겟 예외 레벨을 선택하는 단계 - 타겟 예외 레벨이 트랩-마스킹된-예외 제어 정보에 의존하지 않고 선택됨 - 를 포함한다.

[0006] 적어도 일부 예는, 컴퓨터 프로그램으로서, 호스트 데이터 프로세싱 장치에 의해 실행될 때, 타겟 코드를 실행시키기 위한 명령어 실행 환경을 제공하기 위해 호스트 데이터 프로세싱 장치를 제어하는 명령어를 포함하는 컴퓨터 프로그램을 제공하고, 컴퓨터 프로그램은, 복수의 예외 레벨 중 하나에서 타겟 코드의 명령어의 실행을 시뮬레이션하기 위한 프로세싱 프로그램 로직; 및 타겟 코드에 의해 사용되는 타겟 명령어 세트 아키텍처의 적어도 하나의 레지스터를 나타내는 호스트 데이터 프로세싱 장치의 적어도 하나의 저장 위치에 저장된 제어 정보에 의존하여, 예외의 수용을 시뮬레이션하기 위한 예외 제어 프로그램 로직을 포함하고, 제어 정보는 소프트웨어에 의해 마스킹된 상태와 마스킹되지 않은 상태 중 하나로 설정 가능한 마스킹 제어 정보; 및 소프트웨어에 의해 트랩핑되지 않은 상태와 트랩핑된 상태 중 하나로 설정 가능한 트랩-마스킹된-예외 제어 정보를 적어도 포함하고; 여기서, 소프트웨어가 마스킹 제어 정보를 마스킹된 상태로 설정하는 것에 기초하여 마스킹 가능한 마스킹 가능 클래스의 예외 중 주어진 예외에 응답하여, 예외 제어 프로그램 로직은 마스킹 제어 정보가 마스킹된 상태에 있는지 여부를 결정하고; 마스킹 제어 정보가 마스킹된 상태에 있고 현재의 예외 레벨이 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖는 경우에 주어진 예외가 발생하는 적어도 하나의 시나리오에서, 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있는지 여부에 의존하여 주어진 예외를 미리결정된 트랩 타겟 예외 레벨로 트랩할지 여부를 제어하고; 마스킹 제어 정보가 마스킹되지 않은 상태에 있고 주어진 예외가 마스킹되지 않는 것으로 결정되는 경우에, 주어진 예외를 핸들링하기 위한 타겟 예외

레벨을 선택하도록 - 타겟 예외 레벨이 트랩-마스킹된-예외 제어 정보에 의존하지 않고 선택됨 - 구성된다.

[0007] 컴퓨터 프로그램은 저장 매체 상에 저장될 수 있다. 저장 매체는 일시적 또는 비일시적 저장 매체일 수 있다.

[0008] 본 기법의 추가의 양태, 특징 및 이점은 첨부 도면과 관련하여 읽혀질 예의 하기의 설명으로부터 명백해질 것이다.

도면의 간단한 설명

[0009] 도 1은 데이터 프로세싱 장치의 일례를 개략적으로 예시한다.

도 2는 프로세싱 회로부가 데이터 프로세싱을 수행할 수 있는 다수의 예외 레벨을 예시한다.

도 3은 레지스터에 저장된 제어 정보를 예시한다.

도 4는 마스킹 가능 클래스의 예외 중 주어진 예외가 마스킹되어야 하는지 여부의 결정을 예시하는 흐름 다이어그램이다.

도 5는, 마스킹 제어 정보가 마스킹된 상태에 있고 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있는지 여부에 의존하여, 주어진 예외를 미리결정된 트랩 타겟 예외 레벨로 트랩할지 여부의 결정을 예시하는 흐름 다이어그램이다.

도 6은 트랩-마스킹된-예외 제어 정보에 대한 업데이트의 제어를 예시하는 흐름 다이어그램이다.

도 7은 비-마스킹 가능 클래스의 예외 중 예외가 트랩-마스킹된-예외 제어 정보에 기초하여 미리결정된 트랩 타겟 예외 레벨로 트랩핑되는지 여부의 제어를 예시하는 흐름 다이어그램이다.

도 8a 내지 도 8o는 마스킹 가능 클래스의 예외 중 주어진 예외가 마스킹되어야 하는지 여부, 및 마스킹되지 않으면, 어느 예외 레벨이 주어진 예외가 수용되는 타겟 예외 레벨인지를 결정하는 구체적인 예를 예시한다.

도 9a 내지 도 9i는 비-마스킹 가능 예외 클래스의 예외 중 예외에 대한 타겟 예외 레벨을 결정하는 구체적인 예를 예시한다.

도 10a 및 도 10b는 마스킹 가능 클래스의 주어진 예외가 마스킹되는지 여부, 및 마스킹되지 않으면, 주어진 예외가 발생한 시점의 현재의 예외 레벨 및 제어 정보에 의존하여 주어진 예외가 어느 예외 레벨에 수용되는지를 예시하는 테이블이다.

도 11은 주어진 예외가 발생한 시점의 현재의 예외 레벨 및 제어 정보에 의존하여 비-마스킹 가능 클래스에 대한 타겟 예외 레벨의 선택을 예시하는 테이블이다.

도 12는 시뮬레이션 예를 예시한다.

발명을 실시하기 위한 구체적인 내용

[0010] 장치는 다수의 예외 레벨 중 하나에서 데이터 프로세싱을 수행하기 위한 프로세싱 회로부, 및 적어도 하나의 레지스터에 저장된 제어 정보에 의존하여, 프로세싱 회로부에 의한 예외의 수용을 제어하기 위한 예외 제어 회로부를 갖는다. 제어 정보는 예외가 마스킹 또는 수용되는지 여부에 영향을 미치고, 수용되면, 어떤 예외 레벨이 예외가 수용되는 타겟 예외 레벨인지에 영향을 미친다.

[0011] 제어 정보는 마스킹된 상태와 마스킹되지 않은 상태 중 하나로 설정 가능한 마스킹 제어 정보를 포함한다. 마스킹 제어 정보는, 소정의 이벤트에 응답하여 프로세싱 회로부 또는 예외 제어 회로부의 하드웨어에 의해 자동적으로 마스킹된 상태로 설정될 수 있고/있거나(예를 들어, 마스킹 제어 정보는 예외의 수용에 응답하여 마스킹된 상태로 설정될 수 있음), 주어진 예외 레벨에서 실행되는 소프트웨어에 의해 마스킹 가능 클래스의 예외가 마스킹되는지 여부를 제어하여, 예외가 주어진 예외 레벨에서 수용되는 것을 방지할 수 있다. 예외를 수용할 때 하드웨어에서 설정되면, 소프트웨어는 마스킹 제어 정보가 마스킹되지 않은 상태로 클리어되는 타이밍을 제어할 수 있다. 예를 들어, 소프트웨어가 마스킹 제어 정보를 원하는 바에 따라 마스킹되거나 마스킹되지 않은 상태로 설정하는 것을 트리거할 수 있게 하기 위해, 명령어는 명령어 세트 아키텍처에서 지원될 수 있다. 소프트웨어에 의해 마스킹된 상태로 설정되면, 소프트웨어는 일정 시간 동안 예외의 수용을 비활성하고자 할 때에 임의의 이유로 마스킹 제어 정보를 마스킹된 상태로 설정하도록 선택할 수 있다. 마스킹 제어 정보가 하드웨어 또는 소프트웨어에서 설정되는지에 관계없이, 예외를 마스킹하는 하나의 이유는 예외가 수용된다면 덮어쓰기될

수 있는, 소프트웨어에 의해 현재 수행되고 있는 프로세싱과 연관된 상태 정보를 보존하기 위한 것일 수 있다.

[0012] 그러나, 프로세싱 회로부 상에서 실행되는 다른 소프트웨어(예를 들어, 하이퍼바이저 또는 보안 모니터와 같은 감독 소프트웨어)는, 적은 권한을 갖는 예외 레벨에서 동작할 때 프로세싱 회로부가 (하드웨어에서 또는 소프트웨어의 제어 하에서) 마스킹 제어 정보를 마스킹된 상태로 설정했는지에 관계없이, 마스킹 가능 클래스의 예외를 신속하게 처리되어야 하는 중요한 예외로 간주할 수 있다. 예를 들어, 마스킹 가능 클래스의 예외가 오류의 검출을 나타내면, 마스킹 제어 정보에 기초한 마스킹으로 인해 예외의 핸들링을 지연시키는 것은 오류가 전파되고 그에 따라 문제가 더 악화되어 위험을 초래할 수 있다는 우려가 있을 수 있다(예를 들어, 메모리에 저장된 잘못된 값이 다른 동작에 의해 프로세싱되어 오류를 메모리의 다른 위치로 전파시킬 수 있다). 마스킹 가능 클래스의 예외가 신속하게 처리되는 것을 보장하기 위한 하나의 접근법은, 마스킹 제어 정보가 마스킹된 상태에 있더라도 마스킹 제어 정보에 기초한 마스킹을 오버라이드하도록, 더 많은 권한을 갖는 예외 레벨에서의 소프트웨어가 마스킹 가능 클래스의 예외가 해당 더 많은 권한을 갖는 예외 레벨에서 수용되어야 하는 것을 표시하게 설정할 수 있는 제어를 제공하는 것일 수 있다. 그러나, 이러한 접근법이 갖는 문제점은, 이러한 제어가 설정될 경우에, 마스킹 가능 클래스의 예외의 핸들링이 예외를 핸들링하기 위한 통상의 디폴트 레벨로부터 항상 벗어나게 되어, 소프트웨어 개발의 복잡성을 증가시킬 수 있고, 이는, 상이한 예외 레벨에서 실행되는 소프트웨어를 개발하는 상이한 소프트웨어 개발자 사이에서 더 많은 협력을 필요로 할 수 있기 때문이다.

[0013] 이하에 논의되는 예에서, 제어 정보는 소프트웨어에 의해 트랩핑되지 않은 상태와 트랩핑된 상태 중 하나로 설정 가능한 트랩-마스킹된-예외 제어 정보를 포함한다. 마스킹 제어 정보를 마스킹된 상태로 설정하는 것에 기초하여 마스킹 가능한 마스킹 가능 클래스의 예외 중 주어진 예외에 응답하여, 예외 제어 회로부는 마스킹 제어 정보가 마스킹된 상태에 있는지 여부를 결정하고; 마스킹 제어 정보가 마스킹된 상태에 있고 현재의 예외 레벨이 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖는 경우에 주어진 예외가 발생하는 적어도 하나의 시나리오에서, 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있는지 여부에 의존하여 주어진 예외를 미리결정된 트랩 타겟 예외 레벨로 트랩할지 여부를 제어하고; 마스킹 제어 정보가 마스킹되지 않은 상태에 있고 주어진 예외가 마스킹되지 않는 것으로 결정되는 경우에, 주어진 예외를 핸들링하기 위한 타겟 예외 레벨을 선택하도록 - 타겟 예외 레벨이 트랩-마스킹된-예외 제어 정보에 의존하지 않고 선택됨 - 구성된다.

[0014] 따라서, 트랩-마스킹된-예외 제어 정보는, 마스킹 제어 정보가 마스킹된 상태로 설정된 경우에, 예외를 핸들링하는 데 지연을 감소시키기 위해, 예외가 미리결정된 트랩 타겟 예외 레벨로 트랩핑될 수 있도록 예외 핸들링 회로부를 구성하기 위해 사용될 수 있는 아키텍처 제어 옵션을 제공한다. 그러나, 트랩-마스킹된-예외 제어 정보에 기초한 이러한 트랩핑은 마스킹 제어 정보가 마스킹된 상태로 설정되는 것을 조건으로 한다. 마스킹 제어 정보가 마스킹되지 않은 상태에 있고 주어진 예외가 마스킹되지 않은 경우에, 예외는, 트랩-마스킹된-예외 제어 정보에 의존하지 않고 선택된 타겟 예외 레벨에서 수용될 수 있다. 실제로, 마스킹 제어 정보는 전형적으로 실행 시간의 비교적 짧은 부분 동안만 마스킹된 상태로 설정되기 때문에, 이는, 마스킹 가능 클래스의 예외가 통상적으로 예외를 수용할 것인 디폴트 레벨보다 더 많은 권한을 갖는 예외 레벨로 트랩핑되어야 하는 경우의 수를 크게 감소시킬 수 있다. (마스킹 제어 정보의 상태에 관계없이) 마스킹 가능 클래스의 모든 예외가 더 많은 권한을 갖는 예외 레벨로 트랩핑되어야 함을 표시하는 제어 옵션과 비교하여, 이는, (더 많은 권한을 갖는 코드가 전형적으로 대부분의 예외를 핸들링하는 "펌웨어 우선" 예외 핸들링 대신) 더 적은 권한을 갖는 소프트웨어가 대부분의 경우의 마스킹 가능 클래스의 예외를 핸들링할 수 있는 "커널 우선" 예외 핸들링을 활성화하여 더 적은 권한을 갖는 예외 레벨에서 더 많은 부분의 예외가 핸들링될 수 있게 한다. 커널 우선 예외 핸들링은 펌웨어 우선 예외 핸들링에 비해 소프트웨어 개발을 단순화할 수 있기 때문에 종종 선호된다. 따라서, 트랩-마스킹된-예외 제어 정보를 지원하는 명령어 세트 아키텍처를 지원하는 프로세서는, 소프트웨어 개발자가 소프트웨어를 개발하기가 더 간단하기 때문에, 트랩-마스킹된-예외 제어 정보를 지원하지 않는 프로세서에 비해 개선된다.

[0015] 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖는 예외 레벨에서 실행되는 트랩-마스킹된-예외 제어 정보에 대한 업데이트를 요청하는 명령어에 응답하여, 프로세싱 회로부는 트랩-마스킹된-예외 제어 정보에 대한 요청된 업데이트를 거부하거나 트랩할 수 있다. 따라서, 트랩-마스킹된-예외 제어 정보는 예외가 트랩핑될 더 많은 권한을 갖는 예외 레벨에서 실행되는 소프트웨어에 의해 설정되는 것으로 제한될 수 있다(또는 해당 예외 레벨보다 훨씬 많은 권한을 갖는 예외 레벨에 의해 설정될 수 있다). 따라서, 트랩-마스킹된-예외 제어 정보는, 더 많은 권한을 갖는 코드가, 마스킹 제어 정보가 마스킹된 상태에 있는 것에 기초하여 마스킹 가능 클래스의 예외를 마스킹하기 위한 프리퍼런스(더 적은 권한을 갖는 예외 레벨

에서의 동작과 연관됨)를 오버라이드할 수 있게 하는 아키텍처 제어 옵션으로 보여질 수 있다.

[0016] 일부 예에서, 마스킹 제어 정보에 의존한 트랩이 특정 예외 레벨로 수용되는지 여부를 제어하는 트랩-마스킹된-예외 제어 정보의 단일 항목만 있을 수 있다.

[0017] 그러나, 복수의 미리결정된 트랩 타겟 예외 레벨 중 각자의 것과 각각 연관된, 트랩-마스킹된-예외 제어 정보의 다수의 항목을 제공하는 것이 유용할 수 있다. 예외 제어 회로부는, 마스킹 제어 정보가 마스킹된 상태에 있고 현재의 예외 레벨이 복수의 미리결정된 트랩 타겟 예외 레벨 중 주어진 것보다 더 적은 권한을 갖는 경우에 주어진 예외가 발생하는 적어도 하나의 시나리오에서, 복수의 미리결정된 트랩 타겟 예외 레벨 중 해당 주어진 것과 연관된 트랩-마스킹된-예외 제어 정보의 대응하는 항목이 트랩핑된 상태에 있는지 여부에 의존하여, 주어진 예외가 복수의 미리결정된 트랩 타겟 예외 레벨 중 주어진 것에 트랩핑되는지 여부를 제어할 수 있다. 상이한 예외 레벨에서 더 많은 권한을 갖는 소프트웨어에 의해 설정될 수 있는 트랩-마스킹된-예외 제어 정보의 다수의 항목을 제공함으로써, 이는 다수의 더 많은 권한을 갖는 소프트웨어(예를 들어, 하이퍼바이저 및 보안 모니터 코드)에게, 마스킹 제어 정보가 마스킹된 상태로 설정된 경우에 예외가 마스킹되지 않고 트랩핑되어야 하는지 여부에 대한 별개의 제어를 제공한다. 이는, 상이한 소프트웨어 제공자가 마스킹 가능 클래스의 예외가 핸들링되는 방식에 영향을 주는 더 많은 유연성을 제공할 수 있다.

[0018] 마스킹 제어 정보가 마스킹된 상태에 있고, 트랩-마스킹된-예외 제어 정보의 적어도 2개의 항목이 트랩핑된 상태에 있고, 트랩-마스킹된-예외 제어 정보의 상기 적어도 2개의 항목 각각에 대해 미리결정된 트랩 타겟 예외 레벨로 트랩핑하기 위한 하나 이상의 조건이 만족되는 경우에 주어진 예외가 발생하는 적어도 하나의 시나리오에서: 예외 제어 회로부는 주어진 예외를 트랩-마스킹된-예외 제어 정보의 선택된 항목과 연관된 미리결정된 트랩 타겟 예외 레벨로 트랩하도록 구성되고; 트랩-마스킹된-예외 제어 정보의 선택된 항목은 현재의 예외 레벨과 연관된 미리결정된 트랩 타겟 예외 레벨 사이에서 권한의 차이가 가장 작은 트랩-마스킹된-예외 제어 정보의 상기 적어도 2개의 항목 중 하나를 포함한다. 따라서, 트랩-마스킹된-예외 제어 정보의 하나 초과의 항목이 (예를 들어, 상이한 예외 레벨에서의 소프트웨어에 의해) 트랩핑된 상태로 설정되었을 경우에, 예외가 트랩-마스킹된-예외 제어 정보에 기초하여 트랩핑되면, 예외는, 트랩핑된 상태로 설정된 트랩-마스킹된-예외 제어 정보의 해당 항목과 연관되는 연관된 미리결정된 트랩 타겟 예외 레벨 중 최소 권한을 갖는 레벨로 수용된다. 이는 직관적이지 않게 보일 수 있는데, 이는, 상이한 예외 레벨에서 소프트웨어에 의해 설정된 상충되는 제어가 있으면, 일반적으로는 더 많은 권한을 갖는 예외 레벨에서 설정된 제어가 우선해야 한다고 가정할 것인 다른 예외 핸들링 제어와 상이하기 때문이다. 그러나, 더 많은 권한을 갖는 예외 레벨이 또한 트랩-마스킹된-예외 제어 정보를 트랩핑된 상태로 설정한 경우에도 더 적은 권한을 갖는 예외 레벨과 연관된 트랩-마스킹된-예외 제어 정보의 항목이 우선할 수 있게 하는 것은 유용할 수 있으며, 이는, 예외를 수용하는 데 이용 가능한 최소 권한을 갖는 예외 레벨에서 예외를 수용하는 것이 소프트웨어 개발을 단순화할 수 있기 때문이다.

[0019] 트랩-마스킹된-예외 제어 정보가 트랩핑되지 않은 상태에 있는 경우에, 예외 제어 회로부는 마스킹 제어 정보가 마스킹된 상태에 있는지 여부에 적어도 의존하여 주어진 예외가 마스킹되는지 여부를 결정할 수 있다. 주어진 예외가 마스킹되는지 여부는 또한 다른 요인에 의존할 수 있다. 예를 들어, 트랩이 트랩-마스킹된-예외 제어 정보에 기초하여 수용되는 경우에, 마스킹 제어 정보가 마스킹된 상태임에도 불구하고 예외는 마스킹되지 않는다.

[0020] 또한, 예외를 마스킹하는 다른 이유도 있을 수 있는데, 이는 예를 들어 예외가 발생한 시점의 현재의 권한 레벨에 기초할 수 있다. 예를 들어, 예외 제어 회로부는, 트랩-마스킹된-예외 제어 정보가 트랩핑되지 않은 상태에 있는 경우 예외 레벨이 수용되면, 현재의 예외 레벨이 주어진 예외가 수용될 것인 예외 레벨보다 더 많은 권한을 갖는지 여부에 의존하여 주어진 예외가 마스킹될지 여부를 결정할 수 있다. 이는, 일반적으로 예외를 핸들링할 예외 레벨보다 현재 많은 권한을 갖는 예외 레벨에 있으면(트랩-마스킹된-예외 제어 정보에 기초한 트랩핑을 고려하지 않음), 현재의 프로세싱이 예외의 핸들링보다 더 중요하다고 간주될 수 있고, 따라서 예외는, 마스킹 제어 정보가 마스킹된 상태에 있는지 여부에 관계없이, 암묵적으로 마스킹된다고 간주될 수 있다고 가정할 수 있기 때문에 유용하다.

[0021] 또한, 마스킹 제어 정보가 마스킹된 상태에 있더라도, 이것이, 트랩-마스킹된-예외 제어 정보에 기초한 트랩의 부재 하에 예외가 마스킹된다는 것을 반드시 항상 의미하는 것은 아니다. 예를 들어, 이하에서 더 상세히 설명되는 바와 같이, 소프트웨어가, 적어도 일부 시나리오에서, 마스킹 가능 클래스의 예외가 비-마스킹 가능한 것처럼 취급되어야 한다는 것을 표현할 수 있게 하는 아키텍처 제어 옵션을 제공하는 것도 가능하고, 그에 따라 주어진 예외는 마스킹 제어 정보가 마스킹된 상태에 있더라도 마스킹되지 않을 수 있다.

[0022] 따라서, 일반적으로, 트랩-마스킹된-예외 제어 정보가 트랩핑되지 않은 상태에 있는 경우에, 예외 제어 회로부는 마스킹 제어 정보가 마스킹된 상태에 있는지 여부에 적어도 의존하여 주어진 예외가 마스킹되는지 여부를 결정하도록 구성되지만, 일부 구현에서, 예외가 실제로 마스킹되는지 여부의 결정은 또한 여러 다른 파라미터를 고려할 수 있음이 이해될 것이다.

[0023] 일부 예에서, 제어 정보는 마스킹 가능 클래스의 예외가 마스킹 가능한지 여부를 표시하기 위해 비-마스킹 가능 상태와 마스킹 가능 상태 중 하나로 설정 가능한 비-마스킹 가능-예외-제어 정보를 포함하고; 비-마스킹 가능-예외-제어 정보가 비-마스킹 가능 상태에 있는 적어도 하나의 시나리오에서, 예외 제어 회로부는, 마스킹 제어 정보가 마스킹된 상태로 설정되는 경우에도, 주어진 예외가 마스킹되지 않는 것으로 결정하도록 구성된다. 이러한 제어는, 마스킹 제어 정보가 마스킹된 상태로 설정되었는지 여부에 관계없이, 더 많은 권한을 갖는 소프트웨어가 마스킹 가능 클래스의 예외가 비-마스킹 가능한 것으로 취급되어야 하는 프리퍼런스를 표현할 수 있게 하여, 이들이 더 신속하게 처리되는 것을 보장하는 데 유용할 수 있다.

[0024] 현재의 예외 레벨이 최소 권한을 갖는 예외 레벨이고, 비-마스킹 가능-예외 제어 정보가 비-마스킹 가능 상태에 있고, 주어진 예외가 마스킹되지 않고 트랩-마스킹된-예외 제어 정보가 트랩핑되지 않은 상태에 있는 경우에 주어진 예외를 수용하기 위해 활성화되는 다음 최소 권한을 갖는 예외 레벨로 주어진 예외가 수용될 것인 경우에 주어진 예외가 발생하는 시나리오에서, 예외 제어 회로부는, 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있는 경우에도 미리결정된 트랩 타겟 예외 레벨로 트랩핑하지 않으면서, 다음 최소 권한을 갖는 예외 레벨에서 주어진 예외의 수용을 제어하도록 구성된다. 따라서, 현재의 예외 레벨이 최소 권한을 갖는 레벨인 경우에, 비-마스킹 가능 상태에 있는 비-마스킹 가능-예외 제어 정보는 또한 트랩-마스킹된-예외 제어 정보에 기초하여 임의의 트랩핑을 오버라이드한다. 주어진 예외를 수용하기 위해 활성화되는 다음 가장 최소 권한을 갖는 예외 레벨은 다른 제어 정보에 기초하여 달라질 수 있음에 유의한다. 예를 들어, 시스템이 게스트 운영 체제 방식(하이퍼바이저가 게스트 운영 체제를 관리하지만 운영 체제 기능 자체를 수행하지는 않는 경우) 또는 호스트 운영 체제 방식(하이퍼바이저가 애플리케이션도 관리하고 운영 체제 기능을 수행하는 경우)에 있는지 여부를 특정하는 제어 정보는, 게스트 운영 체제 방식에서 게스트 운영 체제에 대해 사용되는 예외 레벨이 호스트 운영 체제 방식에서 비활성화되는지 여부를 제어할 수 있으므로, 활성화되는 다음 최소 권한을 갖는 예외 레벨은 게스트 운영 체제 방식에서 커널 레벨 예외 레벨(게스트 운영 체제가 실행됨)일 수 있고, 호스트 운영 체제 방식에서 하이퍼바이저 레벨 예외 레벨(하이퍼바이저/호스트 운영 체제가 실행됨)일 수 있다.

[0025] 현재의 예외 레벨이 최소 권한을 갖는 예외 레벨이 아닌 경우에, 비-마스킹 가능 상태에 있는 비-마스킹 가능-예외 제어 정보는 마스킹 자체를 오버라이드할 수 있지만, 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있는 것 및 마스킹 제어 정보가 마스킹된 상태에 있는 것에 기초하여 필요한 것으로 결정된 어떠한 트랩도 오버라이드하지 않는다. 이는, 마스킹 제어 정보를 마스킹된 상태로 설정하는 것은 예외를 수용할 때 오버라이드될 수도 있는 일부 상태를 보존하기 위해 예외를 수용하지 않는 (기본 예외 레벨에서 프로세싱의 현재 상태가) 프리퍼런스가 있음을 표현하는 것이기 때문이며, 따라서 실제로 예외가 마스킹되지 않았을 경우라도 미리결정된 트랩 타겟 예외 레벨로의 트랩을 트리거하여 예외를 더 높은 레벨에서 수용하는 것이 여전히 바람직할 수 있다.

[0026] 일부 예에서, 제어 정보는 상이한 예외 레벨과 연관된 비-마스킹 가능-예외-제어 정보의 복수의 항목을 포함하고; 예외 제어 회로부는 현재의 예외 레벨에 기초하여 선택된 비-마스킹 가능-예외-제어 정보의 복수의 항목 중 하나에 의존하여 주어진 예외가 마스킹되는지 여부를 결정한다. 이는, 상이한 예외 레벨에서 실행되는 소프트웨어의 상이한 제공자가 마스킹 가능 클래스의 예외의 마스킹을 허용할지 여부에 대한 독립적인 프리퍼런스를 표현하는 유연성을 제공할 수 있다.

[0027] 따라서, 미리결정된 트랩 타겟 예외 레벨로의 트랩을 수용할지 여부의 정확한 결정은, 마스킹 제어 정보 및 트랩-마스킹된-예외 제어 정보 이외에도, (일부 구현에서) 현재의 예외 레벨 및 (적어도 하나의 예외 레벨에서) 비-마스킹 가능-예외 제어 정보의 현재 상태와 같은 다수의 요인에 의존할 수 있다는 것이 이해될 것이다.

[0028] 일부 구현에서, 트랩-마스킹된-예외 제어 정보를 오버라이드하는 다른 (선택적) 제어도 있을 수 있으며, 트랩을 수용하기 위해 위에 설명된 조건이 충족되는 경우에도(마스킹 제어 정보가 마스킹된 상태에 있고, 그에 따라 현재의 예외 레벨이 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖고, 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있음), 미리결정된 트랩 타겟 예외 레벨로의 트랩은 여전히 수용되지 않을 수 있다(예를 들어, 다른 제어가 마스킹 가능 클래스의 예외를 상이한 예외 레벨로 트랩하도록 특정하였기 때문임).

[0029] 예를 들어, 제어 정보는 소프트웨어에 의해 트랩핑된 상태와 트랩핑되지 않은 상태 중 하나로 설정 가능한 트랩

제어 정보의 적어도 하나의 추가 항목을 포함할 수 있고; 예외 제어 회로부는 트랩 제어 정보의 추가 항목이 트랩핑된 상태에 있는지 여부에 의존하여 적어도 마스킹 가능 클래스의 예외를 특정 예외 레벨로 트랩할지 여부를 제어할 수 있고; 트랩 제어 정보의 적어도 하나의 추가 항목은 트랩-마스킹된-예외 제어 정보보다 우선할 수 있고, 그에 따라 트랩 제어 정보의 추가 항목이 트랩핑된 상태에 있는 경우에, 예외 제어 회로부는, 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있는지 여부에 관계없이, 트랩 제어 정보의 추가 항목에 기초하여 예외를 특정 예외 레벨로 트랩하도록 구성된다. 예를 들어, 제어 정보의 이러한 추가 항목은 비교를 위해 위에서 설명된 제어와 유사할 수 있으며, 이는, 마스킹 제어 정보가 마스킹된 상태에 있는지 여부에 관계없이 마스킹 가능 클래스의 예외가 소정의 더 많은 권한을 갖는 예외 레벨로 트랩핑될 수 있음을 의미한다. 따라서, 트랩-마스킹된-예외 제어 정보의 사용은, 트랩 제어 정보의 적어도 하나의 추가 항목이 트랩핑되지 않은 상태에 있는 경우와는 관련될 수 있지만, 트랩 제어 정보의 적어도 하나의 추가 항목이 트랩핑된 상태에 있는 경우와는 관련되지 않을 수 있다.

[0030] 일부 경우에서, 트랩 제어 정보의 추가 항목 중 적어도 하나는 마스킹 가능 클래스의 예외에 특정될 수 있고, 다른 클래스의 예외는 트랩 제어 정보의 해당 항목에 의해 영향을 받지 않을 수 있다.

[0031] 또한, 트랩 제어 정보의 추가 항목 중 적어도 하나가, 마스킹 가능 클래스의 예외 및 적어도 하나의 다른 클래스의 예외 둘 모두의 핸들링에 영향을 미치는 보다 일반적인 제어가 될 수 있다.

[0032] 따라서, 어떤 추가적인 아키텍처 제어 설정이 지원되고 어떻게 해당 아키텍처 제어 설정을 정의하는 정보가 설정되었는지에 의존하여, 트랩-마스킹된-예외 제어 정보가 사용될 수 있는 많은 방식이 있다. 일반적으로, 마스킹 제어 정보가 마스킹된 상태에 있고, 현재의 예외 레벨이 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨보다 낮은 권한을 갖는 경우에 주어진 예외가 발생하는 적어도 하나의 시나리오에서, 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있는지 여부에 의존하여 미리결정된 트랩 타겟 예외 레벨로 트랩을 제어하지만, 마스킹 제어 정보가 마스킹되지 않은 상태에 있는 경우에, 마스킹 가능 예외가 마스킹되지 않은 경우 타겟 예외 레벨의 선택에 영향을 주지 않는 트랩-마스킹된-예외 제어 정보를 제공하는 것이 유용하다. 그러나, 예외가 발생할 수 있는 모든 시나리오에서 트랩-마스킹된-예외 제어 정보가 반드시 고려되어야 하는 것은 아니다.

[0033] 마스킹 가능 클래스의 예외는, 선택적으로 마스킹 아웃될 수 있지만 때로는 보다 신속하게 핸들링될 필요가 있을 수 있는 임의의 클래스의 예외일 수 있다.

[0034] 그러나, 위에서 논의된 기법은, 마스킹 가능 클래스의 예외가 정확한 서비스로부터의 편차를 표시하는 비동기 시스템 에러 예외를 포함하는 경우에 특히 유용할 수 있다. 에러 예외는, 예를 들어, 장치 자체(또는 외부 장치)가 정확한 기능 사양을 준수하지 않은 경우, 정확한 서비스로부터의 편차를 표시한다. 이는, 하드웨어 시스템 자체가 정확하게 기능하고 있지만 소프트웨어 구성 정보가 잘못 설정된 경우(예를 들어, 정의되지 않은 명령어가 제공되었거나, 페이지 테이블 매핑이 메모리에서 액세스된 어드레스에 대해 특정되지 않았거나, 메모리 액세스 권한이 위반된 경우(예를 들어, 메모리의 읽기 전용 영역에 대한 기록)), 다른 장애 원인(예컨대, 메모리 관리 장애 또는 정의되지 않은 명령어 에러)과는 상이할 수 있다. 많은 경우에서, 에러 예외는 메모리 시스템 액세스 후에 발생할 수 있지만, 다른 이유(예를 들어, 프로세싱 회로부의 레지스터에서의 오류)로도 발생할 수 있다. 비동기 예외는 실행되는 명령어 스트림의 특정 지점에 귀속될 수 없는 예외이다. 예를 들어, 비동기 예외의 경우에, 명령어 스트림의 특정 지점에 귀속될 수 있는 동기 예외와는 달리, 어떤 명령어가 에러에 의해 영향을 받지 않는 오랜 명령어인지 및 에러에 의해 잠재적으로 영향을 받는 최근 명령어인지를 명확하게 결정하지 못할 수도 있다. 비동기 시스템 에러 예외는 여러 원인을 가질 수 있지만, 비동기 시스템 에러의 일반적인 예는 캐시 라인으로부터 외부 메모리로의 더티 데이터의 후기록(writeback)에 의해 트리거되는 데이터 중단일 수 있다.

[0035] 이러한 에러는, 비동기 시스템 에러 예외의 핸들링을 지연시키면 에러 전파가 더 큰 문제를 야기할 위험성이 있을 수 있기 때문에, 비교적 신속하게 처리되는 것이 바람직할 수 있다. 그러나, 예외를 수용할 때 중요한 상태 정보가 덮어쓰기되는 것을 방지하기 위해, 소프트웨어 또는 하드웨어가, 에러 예외가 마스킹 제어 정보를 사용하여 마스킹되어야 한다는 것을 특정하는 능력을 지원하는 것이 또한 바람직할 수 있다. 트랩-마스킹된-예외 제어 정보의 제공은, 비동기 시스템 에러 예외가 마스킹되지 않는 대부분의 경우에, 예외를 커널 우선(kernel-first)으로 처리할 수 있지만, 마스킹 제어 정보가 마스킹된 상태로 설정된 경우에 예외의 펌웨어 우선 핸들링을 허용하는 안전망이 있는 타협을 허용한다.

[0036] 예외 제어 회로부는 또한 마스킹 제어 정보를 마스킹된 상태로 설정하는 것에 기초하여 마스킹될 수 없는 비-마

스킹 가능 클래스의 예외를 지원할 수 있다. 이러한 클래스의 예외는 마스킹 제어 정보에 기초하여 마스킹될 수 없기 때문에, 이러한 마스킹이 예외를 핸들링하는 데 지연을 야기하는 위에서 논의된 문제가 일어나지 않으므로, 트랩-마스킹된-예외 제어 정보에 기초하여 임의의 트랩을 트리거할 필요가 없을 것으로 예상할 수 있다. 그러나, 발명자는 비-마스킹 가능 클래스의 예외가 실제로 마스킹되지 않을 것이더라도, 마스킹된 상태로의 마스킹 제어 정보의 설정은 예외를 수용할 때, 평소라면 덮어쓰기될 수 있는 중요한 프로세서 상태를 보존하기 위해 예외가 수용되지 않는 프리퍼런스를 시그널링할 수 있다는 것을 인식했다. 따라서, 비-마스킹 가능 클래스의 예외가 마스킹 가능하지 않더라도, 마스킹 제어 정보가 마스킹된 상태에 있는 경우에, 미리결정된 트랩 타겟 예외 레벨로의 트랩이 수용되는지 여부를 제어하기 위해 트랩-마스킹된-예외 제어 정보를 사용하는 것이 여전히 유용할 수 있다. 따라서, 마스킹 제어 정보가 마스킹된 상태에 있고 현재의 예외 레벨이 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖고 현재의 예외 레벨이 예외를 핸들링하기 위한 타겟 예외 레벨일 수 있는 예외 레벨인 경우에 주어진 예외가 발생하는 적어도 하나의 시나리오에서, 예외 제어 회로부는 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있는지 여부에 의존하여 비-마스킹 가능 예외를 미리결정된 트랩 타겟 예외 레벨로 트랩할지 여부를 제어할 수 있다. 이는, 마스킹 제어 정보가 마스킹된 상태에 있는 시점에서 예외가 발생하는 경우에 덮어쓰기로 인해 중요한 상태가 손실될 가능성을 줄이는데 도움이 된다.

[0037] 현재의 예외 레벨이 예외를 핸들링하기 위한 타겟 예외 레벨(예를 들어, 최소 권한을 갖는 예외 레벨)일 수 없는 예외 레벨인 경우라면, 트랩-마스킹된-예외 제어 정보에 기초하여 비-마스킹 가능 예외의 트랩을 트리거할 필요가 없다.

[0038] 비-마스킹 가능 클래스의 예외는, 예를 들어, 정확한 서비스로부터의 편차를 표시하는 동기 외부 중단 예외를 포함할 수 있다. 다시 말하면, 정확한 서비스로부터의 편차는 비동기 시스템 에러 예외에 대해 위에서 설명된 예시적인 이벤트 중 임의의 것에 의해 야기될 수 있지만, 비동기 예외와 대조적으로, 동기 예외는 실행되는 명령어 스트림의 특정 지점에 귀속될 수 있는 것이다. 예를 들어, 동기 예외는 (비동기 에러로 취급될 가능성이 더 높은 캐시 후기록에 대해 검출된 에러가 아니라) 특정된 명령어의 실행 후에 발행된 요구 메모리 액세스 후에 메모리 시스템에서 검출된 에러(예컨대, 실패한 에러 정정 코드 체크)일 수 있으며, 따라서 이는 명령어 스트림의 특정 지점에 귀속될 수 있다.

[0039] 장치는 하나 이상의 예외 신드롬 레지스터의 복수의 세트를 포함하고, 하나 이상의 예외 신드롬 레지스터의 각각의 세트는 상기 예외 레벨의 각자의 것에서 수용된 예외에 대한 예외 신드롬 정보를 저장할 수 있다. 소프트웨어가 임의의 소프트웨어 특정 이유로 마스킹 제어 정보를 마스킹된 상태로 설정하도록 선택할 수 있지만, 마스킹 제어 정보를 마스킹된 상태로 설정하는 일반적인 이유는 이전 예외가 프로세싱되어 있고, 후속 예외가 해당 예외 레벨로 수용된 경우, 이전 예외에 대한 정보가 현재의 예외 레벨에 대한 관련 신드롬 레지스터에 덮어쓰기되는 것은 바람직하지 않기 때문이다. 따라서, 미리결정된 트랩 타겟 예외 레벨로의 트랩을 수용하는 것은, 예외가 신속하게 수용되는 것을 여전히 허용하면서, 현재의 예외 레벨과 연관된 하나 이상의 신드롬 레지스터의 세트의 정보를 보호하는데 도움이 될 수 있다. 따라서, 위에서 설명된 기법은, 상이한 예외 레벨에서 핸들링되는 예외와 연관된 예외 정보의 동시 등록 저장을 허용하는 특정 예외 레벨에 대한 예외 신드롬 레지스터의 뱅크된 세트가 있는 아키텍처에서 특히 유용할 수 있다.

[0040] 위에서 논의된 기법은 위에서 논의된 바와 같이 프로세싱 회로부 및 예외 제어 회로부를 제공하는 하드웨어 회로 특징부를 갖는 하드웨어 프로세서에서 구현될 수 있다. 하드웨어 장치는 위에서 설명된 바와 같이 제어 정보를 저장하기 위해 하드웨어에 구비된 레지스터를 가질 수 있다.

[0041] 그러나, 다른 예에서, 기법은 타겟 코드를 실행하기 위한 명령어 실행 환경을 제공하도록 호스트 데이터 프로세싱 장치를 제어하기 위한 프로그램 내에 구현될 수 있다. 시뮬레이션을 제공하는 컴퓨터 프로그램은 저장 매체 상에 저장될 수 있다. 저장 매체는 일시적 저장 매체 또는 비일시적 저장 매체일 수 있다.

[0042] 그러한 시뮬레이션 컴퓨터 프로그램은, 타겟 명령어 세트 아키텍처에 따라 개발된 타겟 프로그램이, 타겟 명령어 세트 아키텍처를 자체적으로 지원하지 않는 호스트 데이터 프로세싱 장치 상에서 실행될 수 있게 하는 데 유용할 수 있다. 이는, 코드가 개발된 아키텍처를 지원하지 않는 새로운 디바이스 상에서 레거시 코드가 실행될 수 있게 하거나, 새로운 명령어 세트 아키텍처를 지원하는 임의의 하드웨어 디바이스가 아직 이용 가능하기 전에 향후 출시될 새로운 명령어 세트 아키텍처를 위해 개발 중인 타겟 프로그램을 테스트하는 것과 같은 다양한 이유로 유용할 수 있다. 하드웨어 구현 디바이스가 위에서 논의된 특징부를 제공하는 하드웨어 회로를 가질 수 있는 한편, 소프트웨어 시뮬레이션은 대응하는 하드웨어의 기능성을 에뮬레이션하는 프로그램 로직 및 데이터

구조와 같은 대응하는 소프트웨어 특징부를 제공할 수 있고, 그에 따라 타겟 프로그램은, 타겟 명령어 세트 아키텍처를 본래적으로 지원하는 대응하는 하드웨어 디바이스에 의해 타겟 프로그램이 실행된 것처럼 동일한 결과를 제공하기 위해 시뮬레이션 프로그램 상에서 실행될 수 있다.

[0043] 따라서, 컴퓨터 프로그램은 위에서 논의된 프로세싱 회로부 및 예외 제어 회로부와 유사한 방식으로 기능하는 프로세싱 프로그램 로직 및 예외 제어 프로그램 로직을 가질 수 있다. 예외 제어 프로그램 로직은 타겟 코드에 의해 사용되는 타겟 명령어 세트 아키텍처의 적어도 하나의 레지스터를 나타내는 호스트 데이터 프로세싱 장치의 적어도 하나의 저장 위치에 저장된 제어 정보에 기초하여 예외 핸들링을 제어할 수 있다. 마스킹 가능 클래스의 시뮬레이션된 예외는 위에서 논의된 방식으로 핸들링되며, 미리결정된 트랩 타겟 예외 레벨로의 트랩은, 마스킹 제어 정보가 마스킹된 상태에 있는 경우에, 트랩-마스킹된-예외 제어 정보에 기초하여 제어될 수 있다. 따라서, 시뮬레이션 프로그램 상에서 실행되는 타겟 코드(상이한 시뮬레이션된 예외 레벨에서 실행되는 소프트웨어의 각각의 부분을 포함할 수 있음)는 앞서 논의된 것과 동일한 아키텍처 제어 옵션을 지원하는 하드웨어 장치 상에서 실행되는 것과 동일한 효과를 볼 수 있다.

[0044] 도 1은 데이터 프로세싱 장치(2)의 일례를 개략적으로 도시한다. 장치(2)는 명령어에 응답하여 데이터 프로세싱을 수행하기 위한 프로세싱 회로부를 제공하는 프로세싱 파이프라인(4)을 갖는다. 프로세싱 파이프라인(4)은 명령어 캐시(8)로부터 실행될 명령어를 페칭하기 위한 페치 스테이지(6)를 갖는다. 디코드 스테이지(10)는 페칭된 명령어를 디코딩하여, 명령어에 의해 표현되는 동작을 수행하기 위해 파이프라인(4)의 후속 스테이지를 제어하기 위한 제어 신호를 생성한다. 발행 스테이지(12)는 디코딩된 명령어를 수신하고 이들을 큐잉하는 한편, 레지스터(14)에서 임의의 필요한 피연산자가 이용 가능할 때까지 대기한다. 필요한 피연산자가 이용 가능한 것으로 결정되면, 명령어는, 명령어에 응답하여 프로세싱 동작을 수행하기 위한 다수의 실행 유닛(18, 20, 22, 24)을 포함하는 실행 스테이지(16)로 전달된다. 프로세싱 동작은 레지스터(14)로부터 판독된 피연산자를 사용하며, 프로세싱 파이프라인(4)의 후기록 스테이지(26)에 의해 레지스터(14)에 후기록될 수 있는 프로세싱 결과를 생성할 수 있다. 다른 예는 파이프라인 스테이지의 상이한 배열을 가질 수 있다(예를 들어, 명령어에 의해 특정된 아키텍처 레지스터를 하드웨어에 구비된 물리적 레지스터에 매핑하기 위한 레지스터 리네임 스테이지가 포함될 수 있다).

[0045] 이러한 예에서, 실행 스테이지(16)는, 실행 유닛으로서, 정수 피연산자에 대해 산술 또는 논리 연산을 수행하기 위한 산술/논리 유닛(ALU)(18), 입력 피연산자와 출력 피연산자 중 적어도 하나가 부동 소수점 표현으로 표현되는 부동 소수점 연산을 수행하기 위한 부동 소수점 유닛(20), 비순차적 프로그램 플로우를 트리거하기 위한 분기 명령어의 결과를 평가하기 위한 분기 유닛(22), 및 메모리에 대한 액세스를 제어하기 위한 로드/저장 유닛(24)을 포함한다. 로드/저장 유닛(24)은 메모리 시스템으로부터 레지스터(14)로 데이터를 로드하도록 로드/저장 유닛을 제어하는 로드 명령어를 실행하거나, 레지스터(14)로부터 메모리 시스템으로 데이터를 저장하도록 로드/저장 유닛(24)을 제어하는 명령어를 저장할 수 있다. 도 1의 예에 도시된 특정 세트의 실행 유닛(18 내지 24)은 단지 하나의 가능한 구현일 뿐이고 다른 구현은 상이한 세트의 실행 유닛을 가질 수 있다는 것이 이해될 것이다. 예를 들어, 도 1에 도시된 하나 이상의 유형의 실행 유닛이 생략될 수 있거나, 이러한 유형 중 하나가 중복되고, 그에 따라 동일한 유형의 여러 실행 유닛이 구비될 수 있다. 또한, 다른 유형의 실행 유닛이 도 1에 도시되지 않은 다른 클래스의 프로세싱 동작을 핸들링하는 데 전용되게 구비될 수 있다.

[0046] 로드/저장 유닛(24)에 의해 수행되는 로드/저장 동작에 대해, 로드/저장 명령어에 의해 특정된 가상 어드레스는, 메모리 시스템 내의 페이지 테이블 구조로부터 획득된 페이지 테이블 데이터에 정의된 어드레스 맵핑에 기초하여 메모리 관리 유닛(28)에 의해 물리적 어드레스로 변환될 수 있다. 변환 색인 버퍼(TLB)(30)가, 메모리 관리 유닛(28)에 의한 고속 액세스를 위해 페이지 테이블 구조로부터의 페이지 테이블 엔트리를 캐싱하기 위해 구비될 수 있다.

[0047] 이러한 예에서, 메모리 시스템은 레벨 1 명령어 캐시(8), 레벨 1 데이터 캐시(32), 명령어 또는 데이터를 위해 사용되는 공유 레벨 2 캐시(34), 및 메인 메모리(36)를 포함한다. 그러나, 이는 가능한 캐시 계층의 단지 하나의 예일뿐이고, 다른 예는 더 많은 수의 캐시 레벨을 가질 수 있거나 명령어 캐싱과 데이터 캐싱 사이에 상이한 관계를 가질 수 있다는 것이 이해될 것이다.

[0048] 예외 제어 회로부(40)는 프로세싱 장치(2)에 대한 예외 핸들링을 제어하기 위해 제공된다. 예외 신호(42)는 여러 예외 조건 중 하나가 발생된 경우를 표시하기 위해 예외 제어 회로부에 의해 수신될 수 있다. 예외는 여러 유형의 예외를 포함할 수 있다. 예를 들어, 디코드 스테이지(10)는, 장치(2)에 의해 지원되지 않는 정의되지 않은 명령어 인코딩에 대응하는 페칭된 명령어와 마주하는 경우 정의되지 않은 명령어 예외를 생성할 수 있다.

또한, MMU(28)는, 로드/저장 명령어가 페이지 테이블 구조에서 정의된 어드레스 맵핑을 갖지 않거나, 정의된 어드레스 맵핑을 갖지만 대응하는 어드레스 맵핑이 로드/저장 명령어를 발행한 소프트웨어 프로세스에 의해 충족되지 않는 액세스 권한을 특정하는 타겟 어드레스를 특정하는 경우, 어드레스 장애 예외를 생성할 수 있다. 또한, 예를 들어, 사용자가 프로세싱 장치(2)가 포함된 디바이스 상의 버튼을 눌렀다는 것을 표시하는 메시지 또는 외측 네트워크와 통신하는 네트워크 제어기가 메시지를 수신했다는 것을 표시하는 메시지와 같은 외부 디바이스 또는 주변장치로부터의 메시지를 수신하는 것으로 인해 외부 인터럽트가 발생했는지 여부를 표시하는 예외 신호(42)가 수신될 수 있다. 메모리 시스템은 또한, 입자 충격, 이온화 방사선 또는 전자 컴포넌트 에이징 효과로 인한 랜덤 비트 플립, 메모리 시스템 버스 상의 부정확한 제어 설정으로 인한 버스 에러, 또는 요청된 메모리 시스템 동작이 정확하게 수행될 수 없음을 의미하는 다른 조건과 같은 이벤트에 의해 야기되는 저장된 데이터에서의 에러를 검출하기 위해 사용되는 패리티 코드 또는 에러 정정 코드의 체크 실패와 같은 메모리 시스템 에러가 검출되는 경우 예외 신호를 생성할 수 있다. 많은 상이한 유형의 예외가 정의될 수 있고 상기 예는 포괄적이지 않다는 것이 이해될 것이다.

[0049] 예외가 발생하는 경우, 예외 제어 회로부는 레지스터(14)의 레지스터 상태를 사용하여 예외를 수용할지 여부를 결정할 수 있고, 예외가 수용되어야 한다면, 프로세싱 회로부가 예외를 수용하는 경우에 어떤 동작 상태에 있어야 하는지를 결정할 수 있다. 예외가 수용되면, 예외 제어 회로부(40)는 파이프라인에 의해 현재 수행되고 있는 프로세싱을 인터럽트하고 파이프라인을 제어하여 예외 핸들러를 실행하는 것으로 전환하고, 이는 발생한 예외 유형에 의해 표시된 이벤트에 응답하기 위한 소프트웨어 명령어를 제공할 수 있다.

[0050] 도 2는, 명령어를 실행하는 경우 프로세싱 회로부(4)가 동작할 수 있는 상이한 실행 상태(예외 레벨로도 지칭됨)를 예시하는 다이어그램이다. 이 예에서는, 4개의 예외 레벨(EL0, EL1, EL2, EL3)이 있는데, 여기서 예외 레벨(EL0)은 최소 권한을 갖는 예외 레벨이고 예외 레벨(EL3)은 최대 권한을 갖는 예외 레벨이다. 일반적으로, 더 많은 권한을 갖는 예외 레벨에서 실행하는 경우, 프로세싱 회로부는 더 낮은, 더 적은 권한을 갖는 예외 레벨이 액세스할 수 없는 일부 메모리 위치 또는 레지스터(14)에 대한 액세스를 가질 수 있다.

[0051] 이 예에서, 예외 레벨(EL0)은 예외 레벨(EL1)에서 실행되는 대응하는 운영 체제 또는 가상 기계에 의해 관리되는 애플리케이션을 실행하기 위한 것이다. 다수의 가상 기계가 동일한 물리적 플랫폼 상에서 공존하는 경우, EL2에서 동작되는 하이퍼바이저가 제공되어, 각각의 가상 기계를 관리할 수 있다. 도 2는 하이퍼바이저가 가상 머신을 관리하고 가상 머신이 애플리케이션을 관리하는 예를 도시하지만, 하이퍼바이저가 EL0에서 애플리케이션을 직접 관리하는 것이 또한 가능하다(이 경우에 예외 레벨 EL1은 비활성화될 수 있음).

[0052] 필수적인 것은 아니지만, 일부 구현은 프로세싱 회로부에 대해 별개의 하드웨어로 분할된 보안 및 비보안 동작 도메인을 구현할 수 있다. 데이터 프로세싱 시스템(2)은, 보안 도메인에서 동작하는 소프트웨어 프로세스와 연관되는 데이터 및 코드가 비보안 도메인에서 동작하는 프로세스에 의한 액세스로부터 격리되는 것을 보장하기 위해, 프로세서 및 메모리 시스템 내에 구현되는 하드웨어 특징을 가질 수 있다. 예를 들어, 영국 케임브리지 소재의 Arm® Limited에 의해 제공되는 TrustZone® 아키텍처와 같은 하드웨어 아키텍처가 사용될 수 있다. 대안적으로, 다른 하드웨어 강제 보안 구획화 아키텍처가 사용될 수 있다. 보안 애플리케이션(신뢰되는 서비스들)은 보안 도메인의 예외 레벨(EL0)에서 동작될 수 있고 보안(신뢰되는) 운영 체제 또는 가상 기계는 보안 도메인의 예외 레벨(EL1)에서 동작될 수 있다. 일부 구현에서, 보안 상태에서 EL2에 대한 지원이 없으며 하이퍼바이저는 오로지 비보안 EL2에서만 실행될 수 있다. 다른 구현에서, 도 2의 별표에 의해 표시되는 바와 같이 보안 EL2에서 실행되는 보안 하이퍼바이저에 대한 지원이 있을 수 있다. 일부 예에서, 예외 레벨(EL3)에서 실행되는, 비보안 도메인과 보안 도메인 사이의 전이를 관리하기 위한 보안 모니터 프로그램이 제공될 수 있다. 다른 구현은, 보안 모니터 프로그램이 필요로 되지 않을 수 있도록, 보안 도메인 사이의 전이를 하드웨어에서 단속할 수 있다.

[0053] 도 3은 레지스터(14)의 서브세트를 도시한다. 레지스터는 예외 신드롬 레지스터를 포함하며, 이는 각자의 예외 레벨(EL1, EL2 및 EL3)에서 수용된 예외에 대한 예외 신드롬 정보를 저장하기 위해 하나 이상의 예외 신드롬 레지스터의 별도의 세트를 제공하도록 뱅크된다(최소 권한을 갖는 예외 레벨(EL0)은 예외 레벨을 수용할 수 없기 때문에 예외 신드롬 레지스터를 필요로 하지 않는다). 예를 들어, 수용된 예외와 연관된 신드롬 정보는 예외의 원인에 대한 정보(예를 들어, 장애 유형), 예외와 연관된 어드레스에 대한 정보, 또는 예외 핸들러가 예외를 처리하는 방법을 결정할 수 있게 하는 데 유용한 임의의 다른 정보를 포함할 수 있다.

[0054] 도 3은 또한 예외 제어 회로부(40)가 예외의 핸들링을 제어하기 위해 사용할 수 있는 제어 정보의 서브세트를 도시한다. 도 3은 프로세서에 구비된 레지스터 모두를 도시하는 것은 아니며, 해당 레지스터에 구비된 제어 상

태 모두를 도시하는 것은 아님이 이해될 것이다. 또한, 상이한 레지스터에 걸친 제어 정보의 특정 레이아웃은 달라질 수 있고, 동일한 정보가 상이한 형태로 표현될 수 있다는 것이 이해될 것이다. 예를 들어, 상이한 레지스터에서 도 3에 도시된 정보는 동일한 레지스터에 저장될 수 있거나, 도 3에서 동일한 레지스터에 도시된 정보는 상이한 레지스터에 저장될 수 있다. 따라서, 도시된 특정 레지스터 레이아웃은 필수적인 것은 아니며, 다른 레지스터 레이아웃을 사용하여 제어 옵션의 동일한 기능성이 제공될 수 있다. 표기법 XXX_ELx.YYY는 레지스터 XXX 내에 구비된 필드 YYY를 의미하고, 해당 레지스터가 기록될 수 있는 최소 권한을 갖는 예외 레벨이 예외 레벨 ELx임을 표시한다. 레지스터 접미사에 표시된 예외 레벨 ELx보다 더 적은 권한을 갖는 예외 레벨에서 시도되는 경우, 레지스터에 대한 기록이 거부되거나 더 높은 예외 레벨로 트랩핑된다.

[0055] 이러한 특정 예에서, 제어 정보는 다음을 포함한다:

[0056] • PSTATE.EL: 현재의 예외 레벨의 표시;

[0057] • PSTATE.A: 마스킹된 상태(이 예에서는, PSTATE.A = 0b1)와 마스킹되지 않은 상태(이 예에서는, PSTATE.A = 0b0) 중 하나로 설정 가능한 마스킹 제어 정보. 마스킹된 상태에 있는 경우에, 이는 PSTATE.A가 마스킹되지 않은 상태로 클리어될 때까지 예외가 수용되는 것을 방지하도록 현재의 예외 레벨을 타겟팅하는 비동기 예외(비동기 메모리 시스템 에러 예외 - SError 예외를 포함함)가 마스킹되는 프리퍼런스를 표현하지만, 이러한 프리퍼런스는 이하에 나타나는 바와 같이 다른 제어 상태 옵션에 의해 오버라이드될 수 있다. 예를 들어, 비동기 시스템 에러 예외를 마스킹하면, 다른 예외가 이전 예외와 동일한 예외 레벨에서 수용되면, 평소라면 덮어쓰기될 것인 예외 신드롬 레지스터에 저장된 (이전 예외에 대한) 정보를 보호하는 데 유용할 수 있다. 도 3에 도시되지 않았지만, PSTATE 레지스터(들)는 또한 비동기 시스템 에러 예외 이외의 다른 클래스의 예외를 마스킹하기 위한 추가 마스크 필드를 포함할 수 있다. 그러나, 아래에서 논의되는 예는 비동기 시스템 에러(SError) 예외를 마스킹 가능 클래스의 예외의 구체적인 예로 지칭할 것이다. 비동기 시스템 에러는 실행된 명령어 스트림의 특정 지점에 귀속될 수 없는 에러(시스템 하드웨어의 정확한 기능으로부터의 편차)가 검출되는 경우 시그널링되는 예외이다.

[0058] • SCR_EL3.EA: 추가 트랩 제어의 예이며, 0b1로 설정되는 경우, 소정의 클래스의 예외(예를 들어, 동기 외부 중단 및 비동기 시스템 에러(SError) 예외)가, SCR_EL3.EA = 0b0인 경우, 평소라면 핸들링될 것인 예외 레벨 대신, EL3으로 라우팅되게 한다.

[0059] • HCR_EL2.TEA: 추가 트랩 제어의 다른 예이며, 0b1로 설정되는 경우, EL0, EL1 또는 EL2에서 발생하는 외부 중단 예외가, HCR_EL2.TGE 또는 다른 제어가 EL2 또는 EL3으로의 라우팅을 필요로 하지 않으면 HCR_EL2.TEA = 0b0일 경우, 평소라면 핸들링될 것인 예외 레벨 대신, (SCR_EL3.EA에 기초하여 EL3으로 라우팅되지 않는 경우) EL2로 라우팅되게 한다.

[0060] • HCR_EL2.TGE: 추가 트랩 제어의 다른 예이며, EL2로 "일반 예외를 트랩할지" 여부를 표시하고, 0b0으로 설정되는 경우 EL0에서의 실행에 대한 영향이 없는 반면, 0b1 예외로 설정되는 경우에는 일반적으로 EL0으로부터 EL1로 라우팅될 것인 예외가 대신 EL2로 라우팅된다. 이것은 EL1이 예외를 수용하는 데 이용 가능하게 되는 것을 효과적으로 비활성화하며, 소프트웨어는 EL1에 도달할 수 없다. HCR_EL2.E2H가 또한 0b1이면, EL1은 완전히 비활성화되고 EL0은 호스트 운영 체제에서 실행된다. HCR_EL2.E2H가 0b0이면, 예외가 EL1로 수용되지 않지만 EL1 제어는 여전히 영향을 미칠 수 있다.

[0061] • HCR_EL2.AMO: 추가 트랩 제어의 다른 예이며, 물리적 시스템 에러(SError) 예외가 EL2로 라우팅되어야 하는지 여부를 표시한다. HCR_EL2.AMO = 0b1인 경우에, SError 예외는 SCR_EL3.EA에 기초하여 EL3으로 라우팅되지 않는 한 EL2로 수용된다. 이것은, HCR_EL2.AMO가 0b0인 경우에(HCR_EL2.TGE 또는 다른 제어가 EL2로의 라우팅을 필요로 하지 않는 경우에) 필요로 되지 않는다.

[0062] • HCR_EL2.E2H: 호스트 운영 체제(유형 2 하이퍼바이저)를 실행하기 위한 구성이 EL2에서 지원되는지 여부를 표시하며, EL0과 같은 더 낮은 레벨에서의 예외 핸들링에 영향을 줄 수 있다(유형 2 하이퍼바이저가 지원되는 경우, 평소라면 EL1에서 핸들링될 것인 일부 예외가 대신 EL2에서 핸들링될 수 있다). HCR_EL2.E2H = 0b1인 경우, EL2에서 호스트 운영 체제를 지원하는 기능이 활성화된다. HCR_EL2.E2H = 0b0 인 경우, EL2에서 호스트 운영 체제를 지원하는 기능이 비활성화된다.

[0063] • SCR_EL3.NMEA, SCTLR2_EL2.NMEA, SCTLR2_EL1.NMEA: 예외 레벨 EL3, EL2 및 EL1과 각자 연관된 비-마스킹 가능-예외 제어 정보의 각자의 항목이며, 해당 예외 레벨(또는 더 많은 권한을 갖는 예외 레벨)에서 실행되는 소프트웨어에 의해 설정 가능하여, PSTATE.A를 사용하는 SError 예외가 마스킹 가능한지 여부를 제어하여 예외가 해당 예외 레벨에서 수용되는 것을 방지한다. 이들은 비-마스킹 가능 상태(예를 들어, 0b1)와 마스킹 가능 상태(예를 들어, 0b0) 중 하나로 각각 설정될 수 있다. 비-마스킹 가능 상태에 있는 경우, PSTATE.A의 값에 관계없이 대응하는 예외 레벨에서 Serror 예외가 수용될 수 있다. 마스킹 가능 상태에 있는 경우, PSTATE.A가 마스킹된 상태(예를 들어, 0b1)로 설정되는 것에 기초하여 Serror 예외가 대응하는 예외 레벨에서 수용되는 것이 마스킹될 수 있다. 이러한 제어가 Serror 예외의 핸들링에 어떻게 영향을 주는지에 대한 자세한 내용은 아래에 제시된다.

[0064] • SCR_EL3.TMEA, HCRX_EL2.TMEA: EL3 및 EL2와 각자 연관된 트랩-마스킹된-예외 제어 정보의 각자의 항목이며, PSTATE.A가 마스킹된 상태에 있고(0b1) 현재의 예외 레벨이 대응하는 예외 레벨(EL3 또는 EL2)보다 더 적은 권한을 갖는 경우에, 비동기 Serror 예외 또는 동기 외부 중단 예외가 대응하는 예외 레벨(EL3 또는 EL2)로 트랩핑되거나, 해당 트랩이 발생하지 않는 트랩핑되지 않은 상태(예를 들어, 0b0)로 트랩핑되는 것을 제어하기 위해 트랩핑된 상태(예를 들어, 0b1)로 설정될 수 있다. TMEA 제어는, 보다 일반적인 유형의 추가 트랩 제어(SCR_EL3.EA, HCR_EL2.TEA, HCR_EL2.TGE 및 HCR_EL2.AMO) 중 하나에 의해 야기되는 EL2 또는 EL3으로의 임의의 트랩에 의해 오버라이드되지만, 이러한 추가 트랩 제어와는 달리, TMEA 제어는 마스킹 제어 정보가 마스킹된 상태(PSTATE.A = 0b1)에 있는 경우에 특정되고, 마스킹 제어 정보가 마스킹되지 않은 상태(PSTATE.A = 0b0)에 있는 경우에 타겟 예외 레벨의 선택에 영향을 미치지 않는다.

[0065] 따라서, PSTATE.A 플래그는 비동기 예외로부터 신드롬 레지스터를 보호하기 위해 사용될 수 있다. 플래그는, 예외가 수용되는 경우에(즉, CPU가 신드롬 레지스터를 채우게 되는 경우에) 프로세싱 회로부의 하드웨어에 의해(소프트웨어 개입 없이 자동으로) 설정되고, PSTATE.A 플래그가 설정 또는 클리어될 것임을 표시하는 명령어 세트 아키텍처의 명령어를 실행하는 소프트웨어에 의해 설정/클리어될 수도 있다. 대부분의 경우에, 통상 사용에서, 플래그는 클리어이다. 플래그가 설정되는 경우, 비동기 예외가 마스킹되고, 즉 수용되지 않는다. 이는 비동기 에러 예외에 대한 문제인데, 에러 예외를 신속하게 수용하지 못하면 에러 전파를 초래할 수 있기 때문이다. 또한, 동기 예외는 결코 마스킹되지 않는 비-마스킹 가능 예외이며, 이는, (예기치 않은) 동기 에러 예외가 신드롬 레지스터를 덮어쓸 수 있음을 의미하며, 이는 에러 복구에도 문제가 된다.

[0066] 이러한 문제에 대한 하나의 가능한 해결책은 에러 예외를 더 높은 예외 레벨로 전달(라우팅)하는 것이며, 각각의 예외 레벨은 자신의 신드롬 레지스터 세트를 갖는다. 이는 펌웨어 우선 에러 핸들링이라고 알려져 있다. 이러한 접근법이 갖는 문제점은 항상 EL1에서 운영 체제로부터 에러 핸들링 빼앗고, 운영 체제에 오류에 대해 알리고 이를 복구하는 것을 가능하게 하기 위해 복잡한 소프트웨어 구성을 필요로 한다는 것이다. 또한 EL2 또는 EL3에서 동작하는 펌웨어가 종종 그렇듯이 시스템 관리자에게는 불투명한 경우에 오류의 근본 원인을 식별하는 것을 더 어렵게 한다.

[0067] 본원에 설명된 예는 새로운 예외 라우팅 제어인 TMEA(트랩-마스킹된-예외 제어 정보)를 도입하며, 이는 EL2 또는 EL3과 같은 더 높은 예외 레벨에서의 소프트웨어의 제어를 받는다. PSTATE.A가 클리어인 경우, 에러 예외가 더 높은 예외 레벨로 라우팅하지 않고 정상적으로 핸들링할 것인 디폴트 예외 레벨로 정상적으로 수용된다(다른 제어 SCR_EL3.EA, HCR_EL2.TEA, HCR_EL2.TGE 및 HCR_EL2.AMO 중 하나에 기초하여 EL2 또는 EL3으로 트랩핑하지 않는 것으로 가정함). PSTATE.A가 설정되고 (더 상위 예외 레벨에서) TMEA가 설정되면, 에러 예외(동기와 비동기 둘 모두)는 TMEA 플래그를 설정하는 상위 예외 레벨로 수용된다. Arm 아키텍처의 EL1의 경우, 2개의 더 높은 예외 레벨(EL2 및 EL3)이 있으며, 위에 나타난 바와 같이 둘 모두는 설정될 수 있는 TMEA 플래그를 갖는다. SCR_EL3.TMEA, HCRX_EL2.TMEA 둘 모두가 트랩핑된 상태(0b1)로 설정되고 PSTATE.A가 설정된 경우에, 예외는 이러한 더 높은 예외 레벨 중 가장 가까운 레벨로(예를 들어, EL0 또는 EL1에 있는 경우에 EL2로) 수용된다.

[0068] 이러한 방식으로, PSTATE.A는 예상치 못한 에러 예외로부터 신드롬 레지스터를 보호할 수 있지만, 더 높은 예외 레벨에 의해 설정된 TMEA 제어는 "안전망"을 제공하여, PSTATE.A가 마스킹을 표시하는 0b1로 설정된 드문 경우에, 예외가 여전히 더 높은 예외 레벨로 신속하게 수용될 수 있기 때문에 에러가 전파되지 않는 것을 보장한다. 하지만, 대부분의 경우에, PSTATE.A가 0b0으로 설정될 것이므로 에러 예외는 EL1에서 운영 체제에 의해 직접 핸들링될 수 있다.

[0069] 또한, 운영 체제 또는 더 높은 예외 레벨은 또한 PSTATE.A에 관계없이 비동기 예외가 비-마스킹 가능해야 한다

는 것을 표시하기 위해 NMEA 제어 비트를 (TMEA와 직교하게) 설정할 수 있다. NMEA 제어는 모든 예외-핸들링 예외 레벨 EL1, EL2, EL3에 대해 제공된다. 이하의 예는 이러한 제어가 다른 제어와 상호 작용하는 방법을 나타낸다.

[0070] 도 4는 마스킹 가능 클래스의 예외(예를 들어, 비동기 시스템 에러 예외) 중 주어진 예외가 수용되는 것을 방지하도록 마스킹된 것으로 간주되어야 하는지 여부의 결정을 예시하는 흐름 다이어그램이다. 단계 100에서, 예외 제어 회로부(40)는, 트랩-마스크-예외 제어 정보(즉, EL2 및 EL3 TMEA 제어)가 트랩핑되지 않은 상태에 있는 경우에, 마스킹 가능 클래스의 주어진 예외가 수용되는 경우, 수용될 것인 예외 레벨보다 현재의 예외 레벨이 더 많은 권한을 갖는지 여부를 결정한다. 또한, 단계 102에서, 예외 제어 회로부(40)는 마스킹 제어 정보(PSTATE.A)가 마스킹된 상태에 있는지 여부를 결정한다.

[0071] 현재의 예외 레벨이, 수용되는 경우, 예외를 핸들링할 것인 디폴트 예외 레벨보다 더 많은 권한을 갖거나(TMEA에 기초한 임의의 트랩핑을 무시함) 마스킹 제어 정보(PSTATE.A)가 마스킹된 상태에 있으면, 예외는 잠재적으로 마스킹되지만, 비-마스킹 가능-예외 제어 정보(앞서 언급한 NMEA 제어)에 기초하여 오버라이드될 수 있다. 단계 104에서, 예외 제어 회로부(40)는, 현재의 예외 레벨에 기초하여 선택된 비-마스킹 가능-예외 제어 정보(적용 가능한 NMEA 제어)가 비-마스킹 가능 상태(예를 들어, 0b1)로 설정되는지 여부를 결정한다. 비-마스킹 가능-예외 제어 정보가 비-마스킹 가능 상태에 있지 않으면(즉, 마스킹 가능 상태, 예를 들어 0b0에 있으면), 단계 100 또는 102에서 예외가 마스킹되어야 한다는 결정은 오버라이드되지 않고, 단계 106에서 예외 제어 회로부는 (예를 들어, 도 5와 관련하여 더 상세히 논의되는 바와 같이 TMEA 제어에 기초하여) 마스킹됨에도 불구하고 예외를 트랩할 어떠한 이유가 있는지 여부를 결정한다. 예외를 트랩할 이유가 없으면, 단계 107에서 주어진 예외는 마스킹되는 것으로 결정되고, 그에 따라 예외는 수용되지 않는다.

[0072] 비-마스킹 가능-예외 제어 정보가 비-마스킹 가능 상태(예를 들어, 0b1)에 있거나, 단계 106에서 예외를 트랩할 이유가 있다고 결정되면, 단계 108에서, 단계 100 및 단계 102 중 하나의 결과가 Y임에도 불구하고, 주어진 예외는 마스킹되지 않는 것으로 결정되고, 그에 따라 (NMEA 제어가 마스킹된 상태(masked status)의 초기 결정을 오버라이드했거나, TMEA 제어 또는 다른 제어에 기초하여 수용되는 트랩이 있음에 따라) 예외가 여전히 수용된다.

[0073] 일반적으로, 단계 104에서 사용하기 위한 적용 가능한 NMEA 제어는 현재의 예외 레벨에 기초하여 선택되지만, 일부 예에서 다른 제어 정보가 또한 관련될 수 있다. 예를 들어, 아래의 도 8e에 더 상세히 도시된 바와 같이, 적용 가능한 NMEA 제어는 현재의 예외 레벨이 EL3인 경우 SCR_EL3.NMEA이고, 현재의 예외 레벨이 EL2이거나 제어 정보가 프로세싱 회로부가 호스트 운영 체제 방식으로 현재 동작하고 있음을 표시하는 경우 SCTLR2_EL2.NMEA이며, SCTLR2_EL1이다. 현재의 예외 레벨이 EL1 또는 EL0이고 프로세싱 회로부가 호스트 운영 체제 방식으로 현재 동작하고 있지 않은 경우, NMEA(HCR_EL2의 E2H 및 TGE 필드와 현재의 예외 레벨에 기초하여, 프로세싱 회로부가 호스트 운영 체제 방식으로 동작하고 있는지 여부를 결정하는 방법의 예에 대해서는 도 8b 참조).

[0074] 주어진 예외가, 수용되는 경우, TMEA 제어가 트랩핑되지 않은 상태에 있고(단계 100에서 N) 마스킹 제어 정보가 마스킹된 상태에 있지 않은 경우에(PSTATE.A = 0b0) 수용될 것인 디폴트 예외 레벨보다 현재의 예외 레벨이 많은 권한을 갖지 않으면, 단계 108에서 주어진 예외는 마스킹되지 않고 예외가 수용되는 것으로 결정된다.

[0075] 도 5는 트랩-마스킹된-예외 제어 정보(TMEA 제어)에 기초하여 마스킹 가능 클래스의 예외 중 주어진 예외를 트랩할지 여부를 결정하기 위한 단계를 예시하는 흐름 다이어그램이다. 단계 120에서, 예외 제어 회로부는 마스킹 제어 정보가 마스킹된 상태(예를 들어, PSTATE.A = 0b1)에 있는지 여부를 결정한다. 만약 그렇지 않다면, 단계 122에서, 주어진 예외가 도 4에 따라 마스킹되는 것으로 결정되지 않았다면, 주어진 예외는 트랩-마스킹된-예외 제어 정보에 의존하지 않고 선택된 타겟 예외 레벨에서 수용된다.

[0076] 단계 124에서, 예외 제어 회로부는 현재의 예외 레벨이 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖는지 여부를 결정한다(예를 들어, TMEA 제어의 EL2 또는 EL3 버전이 사용되는 경우 EL0 또는 EL1에서, 또는 TMEA 제어의 EL3 버전을 사용하는 경우 EL2에서). 현재의 예외 레벨이 주어진 TMEA 제어와 연관된 예외 레벨 EL2 또는 EL3보다 낮은 권한을 갖지 않으면, 해당 TMEA 제어는 관련이 없다. 현재의 예외 레벨에서 예외의 핸들링을 결정하기 위해 적용 가능한 관련 TMEA 제어가 없으면, 단계 122에서 주어진 예외는, 마스킹되지 않은 경우, 트랩-마스킹된-예외 제어 정보에 의존하지 않고 선택된 타겟 예외 레벨에서 수용된다.

[0077] 마스킹 제어 정보가 마스킹된 상태(단계 120에서 Y)에 있고 현재의 예외 레벨이 트랩-마스킹된-예외 제어 정보

의 적어도 하나의 항목과 연관된 미리결정된 트랩 타겟 예외 레벨(EL2 또는 EL3)보다 더 적은 권한을 가지면, 단계 126에서, 예외 제어 회로부(40)는 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있는지 여부에 의존하여 주어진 예외를 미리결정된 트랩 타겟 예외 레벨로 트랩할지 여부를 제어한다. 트랩이 필요한 것으로 결정되지 않으면, 단계 128에서, 주어진 예외가 마스킹되지 않은 경우, 주어진 예외는 트랩-마스킹된-예외 제어 정보에 기초하여 트랩핑하지 않으면서 디폴트 예외 레벨에서 수용된다. 이러한 디폴트 예외 레벨은 다른 제어 상태에 기초하여 달라질 수 있다(예를 들어, 앞서 언급된 EA, TEA, AMO, E2H, TGE 제어에 기초함 - TMEA에 기초한 트랩이 없는 경우에 이들이 타겟 예외 레벨에 어떻게 영향을 줄 수 있는지에 대해 아래의 더 상세한 예를 참조한다). 트랩이 트랩-마스킹된-예외 제어 정보에 기초하여 필요한 것으로 결정되면, 단계 130에서, 주어진 예외는 트랩핑된 상태에 있는 트랩-마스킹된-예외 제어 정보의 항목과 연관된 미리결정된 트랩 타겟 예외 레벨로 트랩핑된다. 트랩핑된 상태의 트랩-마스킹된-예외 제어 정보의 하나 초과의 관련 항목이 있으면, 트랩은, (현재의 예외 레벨과 상이하지만) 현재의 예외 레벨과 연관된 미리결정된 트랩 타겟 예외 레벨 사이에서 권한의 차이가 가장 작은, 트랩핑된 상태(TMEA = 0b1)의 트랩-마스킹된-예외 제어 정보의 항목 중 하나와 연관된 예외 레벨로 수용된다. 예를 들어, 현재의 예외 레벨이 EL1이고 EL2와 EL3 TMEA 제어 둘 모두가 1로 설정되는 경우, 트랩은 EL2로 수용될 것인 한편, 현재의 예외 레벨이 EL2이면 EL2 및 EL3의 TMEA 제어 둘 모두가 설정되더라도 트랩은 EL3으로 수용된다.

[0078] 단계 126에서, TMEA 제어에 기초하여 트랩할지 여부의 제어는 어떤 다른 제어가 구현되는지에 의존하여 달라질 수 있다. 일반적으로, 트랩핑된 상태의 관련된 트랩-마스킹된-예외 제어 정보(TMEA 제어)가 없는 경우, 트랩은 수용되지 않는다. 그러나, 트랩핑된 상태의 트랩-마스킹된-예외 제어 정보의 관련 항목이 있는 경우에, 트랩이 발생할지 여부는 또한 다른 파라미터에 의존할 수 있다. 예를 들어, 비-마스킹 가능-예외 제어-정보(NMEA 제어)가 지원되는 경우, 일부 시나리오에서, 비-마스킹 가능 상태로 설정되는 NMEA 제어는 PSTATE.A에 기초하여 임의의 마스킹을 오버라이드하는 것 외에도 TMEA 제어에 기초하여 트랩을 오버라이드할 수도 있다. 예를 들어, 현재의 예외 레벨이 최소 권한을 갖는 예외 레벨(EL0)이고, 비-마스킹 가능-예외 제어 정보가 비-마스킹 가능 상태(NMEA 제어 = 0b1)에 있고, 주어진 예외가 마스킹되지 않고 트랩-마스킹된-예외 제어 정보가 트랩핑되지 않은 상태에 있는 경우에 주어진 예외를 수용하기 위해 활성화되는 다음 최소 권한을 갖는 예외 레벨로 주어진 예외가 수용될 것인 경우에 주어진 예외가 발생하는 시나리오에서(예를 들어, 다음 최소 권한을 갖는 예외 레벨은 게스트 운영 체제 방식에서 EL1이거나 호스트 운영 체제 방식에서 EL2일 수 있음), 예외 제어 회로부는, 트랩-마스킹된-예외 제어 정보(TMEA)가 트랩핑된 상태에 있는 경우에도 미리결정된 트랩 타겟 예외 레벨로 트랩핑하지 않으면서, 다음 최소 권한을 갖는 예외 레벨에서 주어진 예외가 수용되어야 하는 것으로 결정할 수 있다. 또한, 관련 TMEA 제어가 트랩핑된 상태로 설정됨에도 불구하고, 단계 130의 트랩이 발생하지 않을 수 있는 다른 시나리오는, 앞서 언급된 EA, AMO, TGE 제어가 (마스킹 정보 PSTATE의 값 또는 TMEA 제어의 값에 관계없이) 예외가 EL3 또는 EL2로 라우팅되어야 하는 것으로 특정했을 때일 수 있다. 따라서, 일부 예에서, TMEA에 기초하여 트랩할지 여부에 대한 결정은 TMEA 제어의 상태 자체를 단순히 체크하는 것보다 더 복잡할 수 있다. 특정 결정 흐름은 어떤 다른 제어 옵션이 특정 실시형태에서 구현되는지에 의존할 수 있다.

[0079] 그럼에도 불구하고, 일반적으로, PSTATE.A 플래그가 마스킹된 상태로 설정되는 적어도 하나의 시나리오에서, PSTATE.A 플래그가 마스킹되지 않은 상태에 있는 경우에 모든 예외가 더 높은 레벨로 수용되는 것을 불필요하게 필요로 하지 않으면서, 2개의 경쟁 요건이 충족될 수 있게 하기 위해(예외가 신속하게 처리되고, 더 적은 권한을 갖는 예외 레벨과 연관된 신드롬 정보가 보호됨), 더 많은 권한을 갖는 예외 레벨로의 트랩이 수용되는 것을 사용자가 정의할 수 있게 하는 TMEA 제어를 제공하는 것이 유용하다.

[0080] 도 6은 트랩-마스킹된-예외 제어 정보(TMEA 제어 정보)에 대한 업데이트가 허용되는지 여부의 제어를 예시하는 흐름 다이어그램이다. 단계 150에서, 트랩-마스킹된-예외 제어 정보에 대한 업데이트를 요청하는 명령어가 실행된다. 단계 152에서, 프로세싱 회로부(4)는 현재의 예외 레벨이, 업데이트되도록 요청되는 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨(HCRX_EL2.TMEA의 경우에 EL2, SCR_EL3.TMEA의 경우에 EL3)보다 더 적은 권한을 갖는지 여부를 결정한다. 만약 그렇다면, 단계 154에서 트랩-마스킹된-예외 제어 정보를 업데이트하려는 시도는 완전히 거부되거나, 더 많은 권한을 갖는 예외 레벨로 트랩핑되고, 그에 따라 더 많은 권한을 갖는 예외 레벨에서 소프트웨어가 요청된 업데이트를 허용할지 여부를 결정할 수 있다. 현재의 예외 레벨이 미리결정된 트랩 타겟 예외 레벨 또는 미리결정된 트랩 타겟 예외 레벨보다 더 많은 권한을 갖는 예외 레벨이면, 단계 156에서 트랩-마스킹된-예외 제어 정보에 대한 요청된 업데이트가 현재의 예외 레벨에서 허용된다(다른 이유로 트랩-마스킹된-예외 제어 정보에 대한 이러한 기록이 트랩핑되게 할 것인 임의의 다른 제어가 구현되어 있지 않은 한). 예를 들어, EL3에서의 소프트웨어는 HCRX_EL2에 대한 액세스를 트랩하도록 제어를

설정하여, EL2로부터의 EL2 TMEA 제어에 대한 기록이 EL3으로 트랩되게 할 수 있다.

[0081] 도 7은 비-마스킹 가능 예외, 예를 들어, 동기 외부 중단 예외의 클래스의 핸들링을 예시하는 흐름 다이어그램이다. 단계 160에서, 이러한 클래스의 비-마스킹 가능 예외가 검출된다. 단계 162에서, 예외 제어 회로부는 레지스터(14)의 제어 정보가, 현재의 예외 레벨이 트랩-마스킹된-예외 제어 정보의 주어진 항목과 연관된 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖고, 현재의 예외 레벨이 예외를 핸들링하기 위한 타겟 예외 레벨일 수 있는 시나리오를 표시하는지 여부를 결정한다(예를 들어, EL0은 타겟 예외 레벨일 수 없을 수 있다). 만약 그렇지 않다면, 단계 164에서, TMEA 제어에 기초한 트랩이 필요하지 않고, 비-마스킹 가능 클래스의 예외가 현재의 예외 레벨에 기초하여 및/또는 다른 제어 정보(TMEA 제어 이외)에 기초하여 선택된 타겟 예외 레벨에서 수용된다.

[0082] 단계 162에서, 현재의 예외 레벨이 트랩-마스킹된-예외 제어 정보의 주어진 항목과 연관된 미리결정된 트랩 타겟 예외 레벨보다 더 적은 권한을 갖고, 현재의 예외 레벨이 예외를 핸들링하기 위한 타겟 예외 레벨일 수 있는 것으로 결정되면(그리고 선택적으로, 일부 예에서, 예를 들어 TMEA에 의해 표시된 트랩에 대해 상이한 거동을 필요로 할 수 있는 SCR_EL3.EA 또는 HCR_EL2.TEA 제어에 의존하여, 적어도 하나의 다른 조건이 만족되면), TMEA 제어(트랩-마스킹된-예외 제어 정보)에 기초한 트랩이 가능하지만, 다른 파라미터에 의존한다.

[0083] 단계 166에서, 예외 제어 회로부는 마스킹 제어 정보가 마스킹된 상태(예를 들어, PSTATE.A = 0b1)에 있는지 여부를 결정한다. 만약 그렇지 않다면, 단계 168에서, TMEA에 기초한 트랩이 필요하지 않고, 미-마스킹 가능 예외는 현재의 예외 레벨 및/또는 다른 제어 정보에 기초하여 선택된 타겟 예외 레벨에서 수용된다.

[0084] 단계 170에서, 예외 제어 회로부는 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있는지 여부를 결정한다(예를 들어, 관련 TMEA 제어가 0b1로 설정됨).

[0085] 마스킹 제어 정보가 마스킹된 상태(단계 166에서 Y)에 있고 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태(단계 170에서 Y)에 있으면, 트랩은 트랩핑된 상태로 설정된 TMEA 제어와 연관된 미리결정된 트랩 타겟 예외 레벨로 수용된다(다시 말하면, 두 TMEA 제어가 설정되고 현재의 예외 레벨이 EL1이면, 트랩은 현재의 예외 레벨에 가장 근접한 권한 레벨인 EL2로 수용된다). 마스킹 제어 정보가 마스킹되지 않은 상태(PSTATE = 0b0)에 있거나 트랩-마스킹된-예외 제어 정보가 트랩핑되지 않은 상태(모든 TMEA 제어가 0b0임)에 있는 경우 TMEA에 기초한 트랩은 수용되지 않지만(단계 168), 이는 EA, TEA 제어에 기초한 것과 같은 다른 이유로 더 높은 레벨로의 트랩을 반드시 배제하지는 않는다.

[0086] 따라서, PSTATE.A 플래그가 비-마스킹 가능 예외의 마스킹을 야기하지 않지만, 그럼에도 불구하고 TMEA 제어에 기초하여 예외가 더 많은 권한을 갖는 레벨로 트랩핑되는지 여부를 제어하기 위해 고려될 수 있다. 이는, PSTATE.A 플래그가 설정되는 경우에, 신드롬 정보를 보존하려는 프리퍼런스가 있어서, 실제로 임의의 마스킹이 없었더라도 높은 예외 레벨로 트랩핑하면 이러한 요구를 만족하는데 도움이 되기 때문에 유용하다. 그럼에도 불구하고, PSTATE.A에 대한 조건부로 이러한 트랩을 제조하면 커널 우선 예외 핸들링의 사용을 증가시키고 펌웨어가 예외를 핸들링해야 할 필요가 있을 경우의 수를 줄여 소프트웨어 개발을 단순화하는 데 도움이 된다.

[0087] 결과가 여러 조건의 체크에 의존하는 흐름 다이어그램의 일부에서, 그러한 조건은 임의의 순서로 또는 일부 경우에서 적어도 부분적으로 병렬로 체크될 수 있어서, 흐름 다이어그램에 도시된 특정 순서의 단계는 이것이 행해질 수 있는 방법의 단지 하나의 예일뿐이라는 것이 이해될 것이다.

[0088] 앞서 설명한 제어 정보의 다양한 항목을 사용하여 예외 핸들링이 제어될 수 있는 방법의 더 상세한 예가 이제 설명된다.

[0089] 도 8a 내지 도 8o는 메모리 시스템 액세스 후에 에러가 검출되었음을 표시하는 비동기 시스템 에러(SError) 예외의 핸들링을 예시하며, 여기서 에러가 명령어 스트림의 특정 지점에 귀속될 수 없으므로, 어떤 명령어가 에러에 의해 영향을 받을 수 있는지 알 수 없다. 이것은 위에서 설명된 마스킹 가능 클래스의 예외의 예이다. 이러한 도면은 전체적으로, 예외 시점에서의 현재의 예외 레벨 및 도 3에 도시된 제어 정보에 기초하여, 예외가 마스킹되어야 하는지 여부, 및 마스킹되지 않는 경우, 어느 예외 레벨이 예외를 수용해야 하는지를 결정하기 위한 프로세스를 예시한다.

[0090] 도 8a는 SError 예외가 더 낮은 예외 레벨로부터 라우팅되어 EL3에서 수용될지 여부를 표시하는 값 route_to_el3의 결정을 예시한다. 현재의 예외 레벨이 EL3이 아니고(단계 1000에서 N) SCR_EL3.EA = 0b1인(단계 1002에서 Y) 경우, route_to_el3은 TRUE이고(단계 1006), 이는 예외가 EL3으로 라우팅되어야 함을 표시한다. 현재의 예외 레벨이 EL3이거나(단계 1000에서 Y) SCR_EL3.EA = 0b0이면(단계 1002에서 N),

route_to_el3은 FALSE이다(단계 1004).

[0091] 도 8b는 프로세싱이 현재의 호스트 운영 체제 방식(유형 2 하이퍼바이저)에 있는지 여부를 표시하는 값 IsInHost()의 결정을 예시한다. IsInHost()는 게스트 운영 체제 방식에 있는 경우 FALSE이고 호스트 운영 체제 방식에 있는 경우 TRUE이다. 현재의 예외 레벨이 EL2도 아니고(단계 1008에서 N) EL0도 아니라면(단계 1014에서 N), IsInHost()는 FALSE이다(단계 1018). IsInHost()는 또한, 현재의 예외 레벨이 EL2이지만(단계 1008에서 Y) HCR_EL2.E2H가 0b0인 경우(단계 1010에서 N), 또는 현재의 예외 레벨이 EL0이고(단계 1014에서 Y) HCR_EL2.E2H와 HCR_EL2.TGE 중 하나 또는 둘 모두가 0b0인 경우(단계 1016에서 N), FALSE이다(단계 1018). IsInHost()는, 현재의 예외 레벨이 EL2이고(단계 1008에서 Y) HCR_EL2.E2H = 0b1이거나(단계 1010에서 Y), 현재의 예외 레벨이 EL0이고(단계 1014에서 Y) HCR_EL2.E2H와 HCR_EL2.TGE 둘 모두가 0b1로 설정되는 경우(단계 1016에서 Y), TRUE이다(단계 1012).

[0092] 도 8c는 SError 예외가 더 낮은 예외 레벨 EL0 또는 EL1로부터 라우팅되어 EL2에서 수용될지 여부를 표시하는 값 route_to_el2의 결정을 예시한다. route_to_el3이 TRUE이면(단계 1020에서 N), route_to_el2는 FALSE이다(단계 1032). route_to_el3이 FALSE이고(단계 1020에서 Y) 현재의 예외 레벨이 EL2 또는 EL3이면(단계 1022 및 1030 둘 모두에서 N), route_to_el2는 FALSE이다(단계 1032).

[0093] route_to_el3이 FALSE이고(단계 1020에서 Y) 현재의 예외 레벨이 EL1이면(단계 1022에서 Y),

[0094] • 단계 1026에서, HCR_EL2.AMO가 0b1인 경우, route_to_el2는 TRUE이고(단계 1024에서 Y)

[0095] • 단계 1028에서, HCR_EL2.AMO가 0b0인 경우, route_to_el2는 FALSE이다(단계 1024에서 N).

[0096] route_to_el3이 FALSE이고(단계 1020에서 Y), 현재의 예외 레벨이 EL0이고(단계 1022에서 N, 단계 1030에서 Y), 도 8b의 프로세스에서 결정된 바와 같이 IsInHost()가 TRUE이거나(단계 1034에서 N) HCR_EL2.E2H와 HCR_EL2.TGE 둘 모두가 0b0으로 설정되는 경우(단계 1038에서 Y), route_to_el2는 FALSE이다(단계 1036).

[0097] route_to_el3이 FALSE이고(단계 1020에서 Y), 현재의 예외 레벨이 EL0이고(단계 1022에서 N, 단계 1030에서 Y), IsInHost()가 도 8b의 프로세스에서 FALSE로 결정되고(단계 1034에서 Y) HCR_EL2.E2H와 HCR_EL2.TGE 중 하나 또는 둘 모두가 0b1로 설정되되는 경우(단계 1038에서 N), route_to_el2는 TRUE이다(단계 1040).

[0098] 도 8d는 SError 예외가 마스킹되어야 하는지 여부의 초기 결정을 나타내는 값 soft_masked의 결정을 예시한다(도 8e 및 도 8f에서 아래에 언급되는 바와 같이, 이러한 초기 결정이 비-마스킹 가능 예외 정보(NMEA 제어)에 기초하여 오버라이드되는 것이 가능하다). soft_masked의 결정은 현재의 예외 레벨에 의존한다(단계 1042).

[0099] 현재의 예외 레벨이 EL3이면, SCR_EL3.EA = 0b0(단계 1044에서 Y) 또는 PSTATE.A = 0b1인 경우(단계 1046에서 Y) soft_masked는 TRUE이고(단계 1050), SCR_EL3.EA = 0b1(단계 1044에서 N)이고 PSTATE.A = 0b0이면(단계 1046에서 N) soft_masked는 FALSE이다(단계 1048).

[0100] 현재의 예외 레벨이 EL2이면,

[0101] • 도 8a에서 결정된 바와 같이 route_to_el3이 TRUE인 경우(단계 1052에서 N) soft_masked는 FALSE이다(단계 1054).

[0102] • route_to_el3이 FALSE이면(단계 1052에서 Y),

[0103] o HCR_EL2.TGE 및 HCR_EL2.AMO 둘 모두가 0인 경우(단계 1056에서 Y) 또는 PSTATE.A = 1인 경우(단계 1058에서 Y), soft_masked는 TRUE이다(단계 1060).

[0104] o HCR_EL2.TGE 및 HCR_EL2.AMO 중 하나 또는 둘 모두가 0b1이고(단계 1056에서 N) PSTATE.A = 0인 경우(단계 1058에서 N), soft_masked는 FALSE이다(단계 1062).

[0105] 현재의 예외 레벨이 EL0 또는 EL1이면,

[0106] • 도 8a에서 결정된 바와 같이 route_to_el3이 TRUE이거나(단계 1064에서 N) 도 8c에서 결정된 바와 같이 route_to_el2가 TRUE인 경우(단계 1066에서 N), soft_masked는 FALSE이다(단계 1070).

[0107] • route_to_el3과 route_to_el2 둘 모두가 FALSE이면, soft_masked는 PSTATE.A = 1인 경우(단계 1068에서 Y)

TRUE이고(단계 1072), PSTATE.A = 0인 경우(단계 1068에서 N) FALSE이다(단계 1070).

[0108] EA, TGE, AMO 제어가 SError 예외를 더 높은 레벨 중 하나로 트랩하도록 설정되지 않은 경우 단계 1044, 1056이 통과되기 때문에, 단계 1044, 1056의 효과는, 현재의 예외 레벨(단계 1050의 경우에 EL3, 또는 단계 1060의 경우에 EL2)이 예외 레벨을 수용하기 위해 지정된 예외 레벨보다 더 많은 권한을 갖는 경우(TMEA 트랩 제어를 무시함) SError 예외가 단계 1050 또는 1060에서 마스킹되는 것으로 간주될 수 있다는 것에 유의한다.

[0109] 도 8e는 도 8d에서 결정된 마스킹된 상태가 오버라이드되어야 하는지 여부를 결정하는 데 사용할 비-마스킹 가능-예외 제어 정보(NMEA 제어)의 각자의 항목 중 어느 것의 선택을 예시한다. 이는 현재의 예외 레벨과 도 8b에서 결정된 IsInHost() 상태에 의존한다. 현재의 예외 레벨이 EL3이면(단계 1076에서 Y), 사용할 NMEA 제어는 SCR_EL3.NMEA이다(단계 1078). 현재의 예외 레벨이 EL2이거나(단계 1080에서 Y) IsInHost()가 TRUE이면(단계 1084에서 Y), 사용할 NMEA 제어는 SCTLR2_EL2.NMEA이다(단계 1082). 현재의 예외 레벨이 EL1 또는 EL0이고(단계 1080에서 N) IsInHost()가 FALSE이면(단계 1084에서 N), 선택되는 NMEA 제어는 SCTLR2_EL1.NMEA이다(단계 1086).

[0110] 도 8f는 SError 예외의 마스킹을 방지하기 위해 도 8d의 초기 마스킹된 결정이 오버라이드되어야 하는지 여부를 결정하기 위해 선택된 NMEA 제어의 적용을 예시한다. soft_masked가 도 8d의 프로세스에서 FALSE로 이미 결정되었다면(단계 1098에서 N), 선택된 NMEA 제어의 값에 관계없이 FALSE로 유지될 것이다(단계 1100). soft_masked가 TRUE였고(단계 1098에서 Y) 선택된 NMEA 제어가 비-마스킹 가능 상태 0b1에 있으면(단계 1102에서 N), soft_masked는 FALSE가 되고(단계 1100), 그에 따라 이전의 마스킹된 상태가 NMEA 제어에 의해 제공된 비-마스킹 가능 표시에 기초하여 오버라이드되었다. soft_masked가 TRUE이고(단계 1098에서 Y) 선택된 NMEA 제어가 마스킹 가능 상태 0b0에 있으면(단계 1102에서 Y), soft_masked는 TRUE로 유지되므로(단계 1104), SError 예외는 마스킹되어 예외가 수용되는 것을 방지할 것이다(하지만 예외는 보류로 남아 있으므로, PSTATE.A 플래그가 나중에 마스킹되지 않은 상태 0b0으로 클리어될 경우에 여전히 추후에 수용될 수 있다).

[0111] 도 8g는 TMEA 제어의 EL2 버전인 HCRX_EL2.TMEA에 기초하여 EL2로의 트랩이 수용되어야 하는지 여부를 표시하는 신호 route_soft_masked_to_el2의 결정을 예시한다. 이것은, EL1 세트의 신드롬 레지스터에 예외 신드롬 정보를 보존하면서 SError 예외가 EL2에서 더 신속히 핸들링될 수 있는 것을 보장하기 위해 PSTATE.A = 1인 경우 수용되는 트랩이다.

[0112] 단계 1122에서, 도 8a에서 결정된 route_to_el3이 TRUE이거나(단계 1108에서 N) 또는 외부 디버거가 EL3 미만의 예외 레벨에서 예외가 수용되는 것을 비활성화하는 경우(단계 1110에서 Y), 파라미터 route_soft_masked_to_el2는 FALSE로 결정된다(즉, TMEA에 기초한 트랩이 발생하지 않는다). 이러한 조건 중 어느 것도 충족되지 않는다고 가정하면(단계 1108에서 Y 및 단계 1110에서 N),

[0113] • HCRX_EL2.TMEA가 트랩핑되지 않은 상태에 있으면(0b0 - 단계 1112에서 N), route_soft_masked_to_el2는 FALSE이고(단계 1122), 이는, 이 경우에 EL2 소프트웨어는, PSTATE.A에 의존하여 EL2로의 트랩이 필요함을 표시하도록 trap-masked-exception 제어 정보를 설정하지 않았기 때문이다.

[0114] • HCRX_EL2.TMEA가 트랩핑된 상태에 있으면(0b1 - 단계 1112에서 Y), TMEA로 인한 EL2로의 트랩이 가능하지만, 이것이 발생할지 여부는 다른 파라미터에도 의존한다:

[0115] o 현재의 예외 레벨이 EL1이고(단계 1114에서 Y) PSTATE.A가 마스킹된 상태에 있으면(0b1 - 단계 1116에서 Y), route_soft_masked_to_el2는 TRUE이다(단계 1118).

[0116] o 현재의 예외 레벨이 EL1이지만(단계 1114에서 Y) PSTATE.A가 마스킹되지 않은 상태에 있으면(0b0 - 단계 1116에서 N), route_soft_masked_to_el2는 FALSE이다(단계 1120).

[0117] o 현재의 예외 레벨이 EL0이고(단계 1124에서 Y), 마스킹된 상태(평가되는 임의의 NMEA 오버라이드 이후)가 TRUE이고(단계 1126에서 Y), 도 8b에서 결정된 IsInHost()가 FALSE이면(단계 1128에서 Y), route_soft_masked_to_el2는 TRUE이다(단계 1130). 단계 1126에서 용어 "soft_masked"는 (임의의 NMEA 오버라이드 이전에) 도 8d에서 결정된 초기 마스킹된 상태가 아니라, 도 8f의 프로세스에서 NMEA에 기초하여 결정된 최종 마스킹된 상태를 지칭하는 것에 유의한다.

[0118] o 현재의 예외 레벨이 EL0이거나(단계 1124에서 Y), (도 8d에서 결정된 초기 마스킹된 상태가 아닌) 도 8f의 프로세스에서 NMEA 오버라이드 이후에 결정된 바와 같이 마스킹된 상태가 FALSE이거나(단계 1126에서 N) 도 8b에

서 결정된 IsInHost()가 TRUE이면(단계 1128에서 N), route_soft_masked_to_el2는 FALSE이다(단계 1122).

[0119] o 현재의 예외 레벨이 EL2 또는 EL3이면(단계 1124에서 N), route_soft_masked_to_el2는 FALSE이다(단계 1122).

[0120] 도 8h는 TMEA 제어의 EL3 버전인 SCR_EL3.TMEA에 기초하여 EL3으로의 트랩이 수용되어야 하는지 여부를 표시하는 신호 route_soft_masked_to_el3의 결정을 예시한다. 다시 말하면, 수용되는 경우, 이러한 트랩은, PSTATE.A = 1임을 고려하면, EL1 또는 EL2 세트의 신드롬 레지스터에 예외 신드롬 정보를 여전히 보존할 수 있으면서 SError 예외가 EL3에서 더 신속하게 핸들링될 수 있도록 하는 것이 의도된다.

[0121] EL3에서의 예외의 핸들링이 외부 디버거에 의해 비활성화되면(단계 1136에서 Y), route_soft_masked_to_el3은 FALSE이다(단계 1134).

[0122] SCR_EL3.TMEA가 트랩핑되지 않은 상태에 있으면(0b0 - 단계 1138에서 N), route_soft_masked_to_el3은 FALSE이고(단계 1134), 이는, 이 경우에 EL3 소프트웨어는 PSTATE.A = 1인 경우 더 낮은 레벨에서 SError 예외를 마스킹하는 것을 회피하기 위한 임의의 프리퍼런스를 표현하지 않았기 때문이다.

[0123] 외부 디버거에 의한 EL3에서의 예외의 비활성화가 없고(단계 1136에서 N) SCR_EL3.TMEA가 트랩핑된 상태에 있으면(0b1 - 단계 1138에서 Y), TMEA로 인한 EL3으로의 트랩이 가능하지만, 이것이 발생할지 여부는 다른 파라미터에도 의존한다:

[0124] • route_to_el2 또는 route_soft_masked_to_el2 중 어느 것이 TRUE이면(단계 1140 또는 단계 1142에서 N),

[0125] o route_soft_masked_to_el3은, 현재의 예외 레벨이 EL3보다 더 적은 권한을 갖고(단계 1150에서 N) EL3보다 더 적은 권한을 갖는 레벨에서의 예외의 수용이 외부 디버거에 의해 현재 비활성화되는 경우(단계 1152에서 Y), TRUE이고(단계 1154); 및

[0126] o route_soft_masked_to_el3은, 현재의 예외 레벨이 EL3이거나(단계 1150에서 Y) EL3보다 더 적은 권한을 갖는 레벨에서의 예외의 수용이 외부 디버거에 의해 현재 비활성화되지 않는 경우(단계 1152에서 N), FALSE이다(단계 1156).

[0127] 따라서, 예외가 EL0-EL2에서 수용되는 것을 비활성화하는 외부 디버거의 부재 하에, 도 8g에서 결정된 바와 같은 EL2로의 TMEA 트랩은 EL3으로의 TMEA 트랩보다 우선할 것이다 - 단계 1142로 인해, SCR_EL3.TMEA와 HCRX_EL2.TMEA 둘 모두가 트랩핑된 상태로 설정되는 경우, 트랩은, HCRX_EL2.TMEA가 관련있는 경우 SError가 발생할 수 있는 EL1/EL0 레벨에 가장 가까운 예외 레벨(권한의 최소 차이)인 EL2로 수용될 것이다.

[0128] • route_to_el2 및 route_soft_masked_to_el2 둘 모두가 FALS인 경우(단계 1140 및 1142에서 Y), route_soft_masked_to_el3은 다음 중 어느 하나인 경우 TRUE이다(단계 1146):

[0129] o 현재의 예외 레벨이 EL2 또는 EL1이고 PSTATE.A가 마스킹된 상태에 있거나(0b1 - 단계 1144에서 Y),

[0130] o 현재의 예외 레벨이 EL2 또는 EL0이고 마스킹된 상태가 TRUE이다(단계 1148에서 Y). 다시 말하면, 단계 1148에서 soft_masked는 (도 8d에서 결정된 초기 마스킹된 상태가 아닌) 도 8f의 프로세스에서 NMEA 오버라이드 이후의 마스킹된 상태를 지칭한다.

[0131] • route_soft_masked_to_el3은 또한, route_to_el2 및 route_soft_masked_to_el2 둘 모두가 FALSE이고(단계 1140 및 1142에서 Y), 단계 1144, 1148에서 테스트된 조건 중 어느 것도 충족되지 않고, 현재의 예외 레벨이 EL3보다 더 적은 권한을 갖고(단계 1150에서 N), EL3보다 더 적은 권한을 갖는 레벨에서의 예외의 수용이 외부 디버거에 의해 현재 비활성화되는 경우(단계 1152에서 Y), TRUE일 수 있다(단계 1154).

[0132] 도 8g의 단계 1116과 단계 1126 사이 및 도 8h의 단계 1144와 단계 1148 사이의 차이에 유의한다. 현재의 예외 레벨이 EL1일 때 발생하는 예외의 경우에, TMEA 트랩은, 도 8f에서 결정된 실제 마스킹 상태가 NMEA 제어를 사용하여 오버라이드되었는지 여부에 관계없이, PSTATE.A가 1인 것에 의존한다. 이는, EL1에서, PSTATE.A의 마스킹된 상태(0b1)로의 설정은 EL1에서의 프로세싱의 현재의 상태가 EL1 신드롬 레지스터에서 보호할 일부 중요한 신드롬 정보가 있는 것을 표시하므로, 마스킹이 NMEA 제어를 사용하여 오버라이드되었더라도, EL1 신드롬 레지스터의 신드롬 정보를 보호하기 위해 EL2 또는 EL3으로의 TMEA 트랩을 수행하는 것이 여전히 바람직하기 때문이다. 대조적으로, 예외 레벨 EL0의 경우에, TMEA 트랩은 (NMEA 제어를 사용하여 오버라이드된 후) 도 8f에서 결정된 실제의 마스킹 상태에 의존한다. 따라서, NMEA 제어가 1로 설정되는 것은 TMEA 트랩이 EL0으로부터 EL2

또는 EL3으로 수용되는 것을 방지할 것이며, 대신 다른 제어(예를 들어, EA, AMO, TGE 제어)가 우선하지 않는다면 비마스킹된 예외가 그것의 디폴트 예외 레벨(예를 들어, EL1)로 수용될 것이다. EL0에 대한 이러한 거동은 유용한데, 이는, EL0 예외 레벨(예외를 핸들링하기 위한 타겟 예외 레벨이 될 수 없음)에서 보호할 예외 신드롬 정보가 없으므로, 관련 TMEA 제어가 설정되더라도 EL2 또는 EL3으로 트랩할 필요가 없기 때문이고, 이는, EL1 신드롬 정보가 캡처된 이후 EL0으로의 이전의 예외 리턴이 이미 있었기 때문에 어떠한 EL1 신드롬 정보도 더 이상 필요하지 않다는 것이 가정될 수 있기 때문이다. 따라서, 관련 TMEA 제어가 트랩핑된 상태(0b1)로 설정되더라도, NMEA 제어가 마스킹 발생을 방지한 경우에는 EL1의 신드롬 정보를 보호하기 위해 EL2 또는 EL3으로 트랩할 필요가 없다.

[0133] 도 8i는 예외를 평소라면 수용할 것인 디폴트 예외 레벨(예를 들어, 게스트 운영 체제 환경의 EL1, 또는 호스트 운영 체제 환경의 EL2)로부터 벗어나 SError 예외를 라우팅할 어떠한 이유가 있는지 여부를 표시하는 "routed_away" 표시의 결정을 도시한다. route_away 표시는, route_to_el2, route_soft_masked_to_el2, route_to_el3 또는 route_soft_masked_to_el3 중 임의의 것이 도 8c, 도 8g, 도 8a 및 도 8h에 각자 도시된 프로세스에서 결정된 바와 같이 TRUE인 경우(단계 1158, 1162, 1164, 1166에서 Y), TRUE이다(단계 1160). route_to_el2, route_soft_masked_to_el2, route_to_el3 및 route_soft_masked_to_el3이 모두 FALSE이면(단계 1166에서 N), routed_away는 FALSE이다(단계 1168).

[0134] 도 8j는 예외가 실제 마스킹(수용되는 것이 방지)되는지 여부를 표시하는 최종 마스킹된 상태의 결정을 예시한다. 도 8f에서 임의의 NMEA 오버라이드 이후에 결정된 바와 같은 soft_masked가 TRUE이고(단계 1170에서 Y) 도 8i에서 결정된 바와 같은 routed_away가 FALSE이면(단계 1174에서 Y), masked는 TRUE이고(단계 1176), 이는 예외가 수용되는 것이 방지될 것임을 표시한다. soft_masked(임의의 NMEA 오버라이드 이후)가 FALSE이거나(단계 1170에서 N) routed_away가 TRUE이면(단계 1174에서 N), 예외가 수용될 것이므로 masked는 FALSE이다(단계 1172).

[0135] 도 8k는 예외가 EL3에서 수용되어야 하는지 여부의 결정을 예시한다. 현재의 예외 레벨이 EL3이고(단계 1178에서 Y) 예외가 마스킹되지 않으면(단계 1180에서 Y), take_in_el3은 TRUE이고(단계 1184), 그렇지 않으면 take_in_el3은 FALSE이다(단계 1182).

[0136] 도 8l은 예외가 EL2에서 수용되어야 하는지 여부의 결정을 예시한다(take_in_el2_0). 현재의 예외 레벨이 EL2이거나(단계 1186에서 Y) 도 8b에서와 같이 결정된 IsInHost()가 TRUE이고(단계 1188에서 Y), 도 8i에서 결정된 바와 같은 routed_away가 FALSE이고(단계 1192에서 Y), 예외가 도 8j에서와 같이 마스킹되지 않는 것으로 결정되면(단계 1194에서 Y), take_in_el2_0은 TRUE이다(단계 1196). 그렇지 않으면, take_in_el2_0은 FALSE이다(단계 1190).

[0137] 도 8m은 EL1에서 예외가 수용되어야 하는지 여부의 결정을 예시한다(take_in_el1_0). 현재의 예외 레벨이 EL1이거나(단계 1198에서 Y) 현재의 예외 레벨이 EL0이고 IsInHost()가 FALSE이고(단계 1200에서 Y), routed_away가 FALSE이고(단계 1204에서 Y), 예외가 마스킹되지 않는 것으로 결정되면(단계 1206에서 Y), take_in_el1_0은 TRUE이다(단계 1208). 그렇지 않으면, take_in_el1_0은 FALSE이다(단계 1202).

[0138] 도 8n은 예외를 수용하기 위한 타겟 예외 레벨의 선택을 예시한다. 예외가 도 8j에 따라 마스킹되는 것으로 결정되면(단계 1210에서 N), 예외가 수용되지 않을 것이므로 타겟 예외 레벨은 알 수 없다.

[0139] "masked" 상태가 FALSE이고(단계 1210에서 Y) take_in_el3(도 8k), route_to_el3(도 8a) 및 route_soft_masked_to_el3(도 8h) 중 임의의 것이 TRUE이면(단계 1214에서 Y), 타겟 예외 레벨은 EL3이다(단계 1216).

[0140] masked가 FALSE이고(단계 1210에서 Y), 예외가 EL3에서 수용되지 않고(단계 1214에서 N), take_in_el2_0(도 8l), route_to_el2(도 8c) 및 route_soft_masked_to_el2(도 8g) 중 임의의 것이 TRUE이면(단계 1218에서 Y), 타겟 예외 레벨은 EL2이다(단계 1220).

[0141] masked가 FALSE이고(단계 1210에서 Y), 예외가 EL3 또는 EL2에서 수용되지 않으면(단계 1214 및 1218에서 N), 타겟 예외 레벨은 EL1이다(단계 1224). 일부 경우에, 타겟 예외 레벨이 EL1이라고 결정하기 전에 "take_in_el1_0"(도 8m에 따라 결정됨)이 TRUE인지 여부의 추가적인 체크가 있을 수 있음에 유의한다. 그러나, 실제로, 단계 1210, 1214, 1218의 결과가 모두 N이고 "take_in_el1_0"이 FALSE인 상태는 아래의 의사코드에서 언급되는 바와 같이 도달 가능하지 않아야 하므로, "take_in_el1_0"이 TRUE인지의 체크는 중복될 수 있다.

[0142] 도 8o는 도 8n에서 결정된 타겟 예외 레벨에서 외부 디버거가 예외를 비활성화하는 것에 기초하여 예외를 마스

킹하는 추가 능력을 예시한다. 외부 디버거가 타겟 예외 레벨에서 예외를 비활성화하는 경우(단계 1228에서 Y), masked는 TRUE가 되고 타겟 예외 레벨을 알 수 없게 된다(단계 1232). 그렇지 않으면, 마스킹된 상태 및 타겟 예외 레벨은 도 8j 및 8n에 따라 각자 결정된 바와 같이 유지된다.

[0143] 도 9a 내지 도 9i는 동기 외부 중단 예외의 핸들링을 예시하며, 이는 (MMU 장애 이외에) 메모리 시스템 액세스 후 외부 메모리 시스템 에러가 검출되었음을 표시하고, 여기서 에러는 실행 중인 명령어 스트림의 특정 지점에 귀속될 수 있다. 이것은 위에서 설명된 비-마스킹 가능 클래스의 예외의 예이다. 이들 도면은 전체적으로 예외 시의 현재의 예외 레벨과 도 3에 도시된 제어 정보에 기초하여 어느 예외 레벨이 예외를 수용해야 할지를 결정하기 위한 프로세스를 예시한다. 이러한 유형의 예외는 마스킹될 수 없다. 도 9a 내지 도 9i의 "IsInHost()"에 대한 참조의 경우에, 이는 비동기 SError 예외에 대해 도 8b에 도시된 것과 동일한 방식으로 결정되므로, 여기서는 이를 다시 반복하지 않는다.

[0144] 도 9a는 동기 외부 중단 예외가 더 낮은 예외 레벨로부터 라우팅되어 EL3에서 수용될지 여부를 표시하는 값 route_to_el3의 결정을 예시한다. 현재의 예외 레벨이 EL3이 아니고(단계 1234에서 N) SCR_EL3.EA = 0b1이면(단계 1236에서 Y), route_to_el3은 TRUE이고(단계 1240), 이는 예외가 EL3으로 라우팅되어야 함을 표시한다. 현재의 예외 레벨이 EL3이거나(단계 1234에서 Y) SCR_EL3.EA = 0b0이면(단계 1236에서 N), route_to_el3은 FALSE이다(단계 1238).

[0145] 도 9b는, 동기 외부 중단이 더 낮은 예외 레벨로부터 라우팅되어 EL2에서 수용될지 여부를 표시하는 값 route_to_el2의 결정을 예시한다. "tea_bit"에 대한 참조는 HCR_EL2.TEA 제어의 값을 지칭한다(단계 1254). 도 9a에서 결정된 바와 같은 route_to_el3이 TRUE이면(단계 1256에서 N), route_to_el2는 FALSE이다(단계 1258).

[0146] route_to_el3이 FALSE이고(단계 1256에서 Y) 현재의 예외 레벨이 EL1이면(단계 1260에서 Y),

[0147] • tea_bit가 1이거나 장애가 EL2에서 핸들될 특정 유형을 가지면(단계 1262에서 Y), route_to_el2는 TRUE이다(단계 1264). 예를 들어, EL2에서 핸들링될 특정 유형의 장애는 메모리에 대한 액세스를 통해 에뮬레이션되는 EL2에서 실행되는 소프트웨어에 의해 관리되는 시스템 레지스터에 대한 액세스와 관련된 장애를 포함할 수 있거나, 가상 어드레스를 중간 어드레스에 매핑하기 위해 운영 체제에 의해 정의된 제1 단계 페이지 테이블 및 중간 어드레스를 물리적 어드레스에 매핑하기 위해 하이퍼바이저에 의해 정의된 제2 단계 페이지 테이블에 기초하여 어드레스 변환이 제어되는 2 단계 어드레스 변환 방식에서 MMU에 의해 검출된 제2 단계 어드레스 변환 장애일 수 있다.

[0148] • tea_bit가 0이고 장애가 EL2에서 핸들링될 유형이 아니면(단계 1262에서 N), route_to_el2는 FALSE이다(단계 1266).

[0149] route_to_el3이 FALSE이고(단계 1256에서 Y) 현재의 예외 레벨이 EL0이면(단계 1268에서 Y),

[0150] • route_to_el2는, IsInHost()(도 8b에 따라 결정됨)가 FALSE(단계 1270에서 Y)이고, tea_bit가 1이거나, HCR_EL2.TGE 제어가 1이거나 장애가 EL2에서 핸들링될 유형 중 하나를 갖는 경우(단계 1274에서 Y), TRUE이다(단계 1276).

[0151] • route_to_el2는, IsInHost()가 TRUE이거나(단계 1270에서 N), 이하의 모두가 충족되는 경우, FALSE이다(단계 1272): tea_bit = 0b0, HCR_EL2.TGE = 0b0, 및 장애가 EL2에서 핸들링될 유형이 아님(단계 1274에서 N).

[0152] 현재의 예외 레벨이 EL3 또는 EL2이면, route_to_el2는 FALSE이다(단계 1278(단계 1268에서 N 이후)).

[0153] 도 9c는 TMEA 제어의 EL2 버전인 HCRX_EL2.TMEA에 기초하여 EL2로의 트랩이 수용되어야 하는지 여부를 표시하는 신호 route_soft_masked_to_el2의 결정을 예시한다. 이는 EL1 세트의 신드롬 레지스터에 예외 신드롬 정보를 보존할 수 있게 하면서 동기 외부 중단 예외가 EL2에서 더 신속히 핸들링될 수 있는 것을 보장하기 위해 PSTATE.A = 1인 경우 수용되는 트랩이다.

[0154] 파라미터 route_soft_masked_to_el2는, 다음 중 임의의 하나 이상이 적용되는 경우, FALSE로 결정된다(단계 1282):

[0155] • route_to_el3(도 9a에 따라 결정된 바와 같음)이 TRUE임(단계 1280에서 N);

[0156] • 현재의 예외 레벨이 EL1이 아닌 임의의 예외 레벨임(단계 1284에서 N);

[0157] • PSTATE.A = 0(단계 1286에서 N);

[0158] • HCRX_EL2.TMEA = 0(단계 1288에서 N).

[0159] 파라미터 route_soft_masked_to_el2는, 다음의 조건 모두가 충족되는 경우, TRUE로 결정된다(단계 1282):

[0160] • route_to_el3이 FALSE임(단계 1280에서 Y);

[0161] • 현재의 예외 레벨이 EL1임(단계 1284에서 Y);

[0162] • PSTATE.A = 1(단계 1286에서 Y);

[0163] • HCRX_EL2.TMEA = 1(단계 1288에서 Y).

[0164] 따라서, HCRX_EL2.TMEA가 1로 설정되는 것(트랩핑된 상태)은, PSTATE.A가 마스킹된 상태로 설정되는 경우, 예외가 EL1에서 발생하면, 동기 외부 중단 예외가 EL2로 트랩하게 한다. 이러한 트랩은, PSTATE.A가 0인 경우(마스킹되지 않은 상태), 수용되지 않는다.

[0165] 도 9d는 TMEA 제어의 EL3 버전인 SCR_EL3.TMEA에 기초하여 EL3으로의 트랩이 수용되어야 하는지 여부를 표시하는 신호 route_soft_masked_to_el3의 결정을 예시한다.

[0166] 파라미터 route_soft_masked_to_el3은, 다음 중 임의의 하나 이상이 적용되는 경우, FALSE로 결정된다(단계 1294):

[0167] • route_to_el2(도 9b에 따라 결정된 바와 같음)가 TRUE임(단계 1292에서 N);

[0168] • route_soft_masked_to_el2(도 9c에 따라 결정된 바와 같음)가 TRUE임(단계 1296에서 N);

[0169] • 현재의 예외 레벨이 EL1 또는 EL2가 아닌 임의의 예외 레벨임(단계 1298에서 N).

[0170] • PSTATE.A = 0(step 1300에서 N).

[0171] • SCR_EL3.TMEA = 0(단계 1302에서 N).

[0172] 파라미터 route_soft_masked_to_el3은, 다음의 조건 모두가 충족되는 경우, TRUE로 결정된다(단계 1304):

[0173] • route_to_el2가 FALSE임(단계 1292에서 Y);

[0174] • route_soft_masked_to_el2가 FALSE임(단계 1296에서 N);

[0175] • 현재의 예외 레벨이 EL1 또는 EL2임(단계 1298에서 Y);

[0176] • PSTATE.A = 1(단계 1300에서 Y);

[0177] • SCR_EL3.TMEA = 1(단계 1302에서 Y).

[0178] 따라서, SCR_EL3.TMEA가 1로 설정되는 것(트랩핑된 상태)은, PSTATE.A가 1(마스킹된 상태)로 설정되는 경우, HCRX_EL2.TMEA 또는 HCR_EL2.TEA에 기초하여 EL2로 트랩핑되지 않고 EL1 또는 EL2에서 발생하면, 동기 외부 중단 예외가 EL3으로 트랩하게 한다. 이러한 트랩은, PSTATE.A가 0인 경우(마스킹되지 않은 상태), 수용되지 않는다.

[0179] 도 9e는 예외를 평소라면 수용할 것인 디폴트 예외 레벨(예를 들어, EL1 또는 EL2)로부터 벗어나 동기 외부 중단 예외를 라우팅할 어떠한 이유가 있는지 여부를 표시하는 "routed_away" 표시의 결정을 도시한다. route_away 표시는, route_to_el2, route_soft_masked_to_el2, route_to_el3 또는 route_soft_masked_to_el3 중 임의의 것이 도 9b, 9c, 9a 및 9d에 각자 도시된 프로세스에서 결정된 바와 같이 TRUE인 경우(단계 1306, 1310, 1312, 1314에서 Y), TRUE이다(단계 1308). route_to_el2, route_soft_masked_to_el2, route_to_el3 및

route_soft_masked_to_el3이 모두 FALSE이면(단계 1306, 1310, 1312, 1314에서 N), routed_away는 FALSE이다(단계 1316).

[0180] 도 9f는 EL3에서 예외가 수용되어야 하는지 여부의 결정을 예시한다(take_in_el3). 현재의 예외 레벨이 EL3이면(단계 1318에서 Y), take_in_el3은 TRUE이고(단계 1322), 그렇지 않으면 take_in_el3은 FALSE이다(단계 1320).

[0181] 도 9g는 EL2에서 예외가 수용되어야 하는지 여부의 결정을 예시한다(take_in_el2_0). 현재의 예외 레벨이 EL2이거나(단계 1324에서 Y) 도 8b에서와 같이 결정된 IsInHost()가 TRUE이고(단계 1326에서 Y), 도 9e에서 결정된 바와 같이 routed_away가 FALSE이면(단계 1330에서 Y), take_in_el2_0은 TRUE이다(단계 1332). 그렇지 않으면, take_in_el2_0은 FALSE이다(단계 1328).

[0182] 도 9h는 EL1에서 예외가 수용되어야 하는지 여부의 결정을 예시한다(take_in_el1_0). 현재의 예외 레벨이 EL1이거나(단계 1334에서 Y) 현재의 예외 레벨이 EL0이고 IsInHost()가 FALSE이고(단계 1336에서 Y), routed_away가 FALSE이면(단계 1340에서 Y), take_in_el1_0은 TRUE이다(단계 1342). 그렇지 않으면, take_in_el1_0은 FALSE이다(단계 1338).

[0183] 도 9i는 예외를 수용하기 위한 타겟 예외 레벨 선택을 예시한다. take_in_el3(도 9f), route_to_el3(도 9a) 및 route_soft_masked_to_el3(도 9d) 중 임의의 것이 TRUE이면(단계 1344에서 Y), 타겟 예외 레벨은 EL3이다(단계 1346).

[0184] 예외가 EL3에서 수용되지 않고(단계 1344에서 N), take_in_el2_0(도 9g), route_to_el2(도 9b) 및 route_soft_masked_to_el2(도 9c) 중 임의의 것이 TRUE이면(단계 1348에서 Y), 타겟 예외 레벨은 EL2이다(단계 1350).

[0185] 예외가 EL3 또는 EL2에서 수용되지 않으면(단계 1348에서 N), 타겟 예외 레벨은 EL1이다(단계 1354). 일부 경우에, 타겟 예외 레벨이 EL1이라고 결정하기 전에 "take_in_el1_0"(도 9h에 따라 결정됨)이 TRUE인지 여부의 추가적인 체크가 있을 수 있음에 유의한다. 그러나, 실제로, 단계 1344 및 1348의 결과가 N이고 "take_in_el1_0"이 FALSE인 상태는 아래의 의사코드에서 언급되는 바와 같이 도달 가능하지 않아야 하므로, "take_in_el1_0"이 TRUE인지의 체크는 중복될 수 있다.

[0186] 위에 나타난 흐름 다이어그램의 경우에, 단계를 다른 순서로 수행하거나 일부 단계를 병렬로 수행함으로써 동일한 논리적 기능이 달성될 수 있음이 이해될 것이다. 또한, 모든 단계가 주어진 실시형태에 필수적인 것은 아니다. 일부 구현에서, 소정의 유형의 예외 라우팅 제어가 지원되지 않을 수 있으며 아래의 의사코드에 나타난 바와 같이, 소정의 선택적인 특징이 지원되는지 여부에 대한 일부 추가적인 테스트가 있을 수 있다(예를 들어, "???Enabled()" 또는 "Have???()" 형식의 변수의 테스트를 참조하며, 이것은 소정의 선택적인 특징이 활성화되고 구현되는지 여부를 표시하는 변수이다). 의사코드에 나타난 AND 또는 부울 관계는 이러한 선택적인 특징 중 하나가 지원되지 않는 경우 무슨 일이 일어날지를 보여준다. 대부분의 경우에, 예외 라우팅 제어가 지원되지 않는 경우, 대응하는 제어 값은 0b0인 것처럼 취급된다.

[0187] 도 8a 내지 도 8o의 기능을 구현하기 위한 예시적인 의사코드가 아래에 나타난다:

```
// Returns a tuple of whether SError exception can be taken and, if
// so, the target Exception level

(boolean, bits(2)) AArch64.PhysicalSErrorTarget()

boolean route_to_el3;
boolean route_to_el2;

// The exception is explicitly routed to EL3
if PSTATE.EL != EL3 then
    route_to_el3 = (HaveEL(EL3) && SCR_EL3.EA == '1');
else
    route_to_el3 = FALSE;

// The exception is explicitly routed to EL2
if !route_to_el3 && EL2Enabled() && PSTATE.EL == EL1 then
    route_to_el2 = (HCR_EL2.AMO == '1');
elsif !route_to_el3 && EL2Enabled() && PSTATE.EL == EL0 then
    route_to_el2 = (!IsInHost() && HCR_EL2.<TGE,AMO> != '00');
else
    route_to_el2 = FALSE;
// The exception is "masked" by software or hardware
boolean soft_masked;
case PSTATE.EL of
    when EL3
        soft_masked = (SCR_EL3.EA == '0' || PSTATE.A == '1');
    when EL2
        soft_masked = (!route_to_el3 &&
                       (HCR_EL2.<TGE,AMO> == '00' ||
                            PSTATE.A == '1'));
    when EL1, EL0
        soft_masked = (!route_to_el3 && !route_to_el2
                                     && PSTATE.A == '1');

// The exception is "disabled" by external debug in the current
Security state
boolean intdis = ExternalDebugInterruptDisabled(EL1);

// When FEAT_DoubleFault or FEAT_DoubleFault2 is implemented,
// the mask might be overridden
```

[0188]

```
if HaveDoubleFault2Ext() then
    bit nmea_bit;
    if PSTATE.EL == EL3 then
        nmea_bit = SCR_EL3.NMEA;
    elsif PSTATE.EL == EL2 || IsInHost() then
        nmea_bit = if IsSCTLR2EL2Enabled() then SCTLR2_EL2.NMEA
                   else '0';
    else
        nmea_bit = if IsSCTLR2EL1Enabled() then SCTLR1_EL2.NMEA
                   else '0';
    soft_masked = soft_masked && (nmea_bit == '0');

elsif HaveDoubleFaultExt() && PSTATE.EL == EL3 then
      bit nmea_bit = SCR_EL3.NMEA AND SCR_EL3.EA;
      soft_masked = soft_masked && (nmea_bit == '0');

boolean route_soft_masked_to_el3;
boolean route_soft_masked_to_el2;
if HaveDoubleFault2Ext() then
    // The soft_masked exception is routed to EL2
    route_soft_masked_to_el2 =
                    (EL2Enabled() && !route_to_el3 && !intdis &&
                    ((PSTATE.EL == EL1 && PSTATE.A == '1') ||
                    (PSTATE.EL == EL0 && soft_masked && !IsInHost()))
                      && (IsHCRXEL2Enabled() && HCRX_EL2.TMEA == '1'));

    // The soft_masked exception is routed to EL3
    route_soft_masked_to_el3 =
              (HaveEL(EL3) && !ExternalDebugInterruptDisabled(EL3) &&
                     ((!(route_to_el2 || route_soft_masked_to_el2) &&
                      ((PSTATE.EL IN {EL2, EL1} && PSTATE.A == '1') ||
                        (PSTATE.EL IN {EL2, EL0} && soft_masked))) ||
                        (PSTATE.EL != EL3 && intdis)) &&
                        SCR_EL3.TMEA == '1');
else
    route_soft_masked_to_el2 = FALSE;
    route_soft_masked_to_el3 = FALSE;

// The exception is routed away from the current exception regime
routed_away = (route_to_el2 || route_soft_masked_to_el2 ||
               route_to_el3 || route_soft_masked_to_el3);

// The exception is actually "masked" - "disabled" will be factored in
// later
masked = soft_masked && !routed_away;

// The exception is taken at EL3
take_in_el3 = PSTATE.EL == EL3 && !masked;

// The exception is taken at EL2 or from the Host EL0 to EL2
take_in_el2_0 = ((PSTATE.EL == EL2 || IsInHost()) &&
                    !routed_away && !masked);

// The exception is taken at EL1 or from the non-Host EL0 to EL1
take_in_el1_0 = ((PSTATE.EL == EL1 ||
                  (PSTATE.EL == EL0 && !IsInHost())) &&
                 !routed_away && !masked);
```

[0189]

```
bits(2) target_el;
if masked then
    target_el = bits(2) UNKNOWN;
elsif take_in_el3 || route_to_el3 || route_soft_masked_to_el3 then
    target_el = EL3;
elsif take_in_el2_0 || route_to_el2 || route_soft_masked_to_el2 then
    target_el = EL2;
elsif take_in_el1_0 then
    target_el = EL1;
else
    Unreachable();

// The exception is "disabled" by external debug for the actual target
if ExternalDebugInterruptDisabled(target_el) then
    masked = TRUE;
    target_el = bits(2) UNKNOWN;
return (masked, target_el);
```

[0190]

[0191] 함수 IsInHost()의 예시적인 의사코드는 다음과 같다:

```
// IsInHost()
// ==========
boolean IsInHost(bits(2) el)
    if !HaveVirtHostExt() || ELUsingAArch32(EL2) then
        return FALSE;
    case PSTATE.EL of
        when EL3
            return FALSE;
        when EL2
            return EL2Enabled() && HCR_EL2.E2H == '1';
        when EL1
            return FALSE;
        when EL0
            return EL2Enabled() && HCR_EL2.<E2H,TGE> == '11';
        otherwise
            Unreachable();
```

[0192]

[0193] 도 10a 및 도 10b는, 제어 정보의 값의 각각의 조합에 대해, 예외 레벨 EL0, EL1, EL2, EL3 중 임의의 것에서 발생하는 비동기 시스템 에러(SError) 예외가 수용될 것인지 또는 마스킹될 것인지 여부, 및 수용되면, 어떤 예외 레벨이 예외가 수용되는 타겟 예외 레벨일 것인지를 표시하는 요약 테이블을 도시한다.

[0194] 테이블의 좌측에 있는 각각의 열은 레이블이 지정된 예외 레벨에 대한 시스템 레지스터 중 하나에서의 비트를 지칭하거나, 다음과 같은 PSTATE.A 플래그를 지칭한다("x"는 대응하는 제어가 0b0인지 0b1인지 여부가 중요하지 않음을 표시함):

[0195] • EA EL3은 SCR_EL3.EA를 의미한다

[0196] • TMEA EL3은 SCR_EL3.TMEA를 의미한다

[0197] • NMEA EL3은 SCR_EL3.NMEA를 의미한다

[0198] • TGE EL2는 HCR_EL2.TGE를 의미한다

[0199] • E2H EL2는 HCR_EL2.E2H를 의미한다

[0200] • AMO EL2는 HCR_EL2.AMO를 의미한다

[0201] • NMEA EL2는 SCTLR2_EL2.NMEA를 의미한다

[0202] • TMEA EL2는 HCRX_EL2.TMEA를 의미한다

[0203] • NMEA EL1은 SCTLR2_EL1.NMEA를 의미한다

[0204] • A PSTATE는 PSTATE.A를 의미한다.

[0205] 테이블의 우측에 있는 각각의 열은 현재의 예외 레벨이 각자 EL3, EL2, EL1, EL0인 경우, 대응하는 세트의 제어 정보 값에 대해 SError 예외가 발생하는 경우, 결과를 나타낸다. 오른쪽 열의 각각의 셀은 다음 결과 중 하나를 나타낸다.

[0206] • ELx는 에러 예외가 ELx로 수용됨을 의미한다.

[0207] • B는 에러 예외가 PSTATE.A에 의해 마스킹되기 때문에 수용되지 않고 보류로 남아 있음을 의미한다.

[0208] • C는 에러 예외가 암시적으로 마스킹되기 때문에(현재의 예외 레벨이 예외를 수용해야 하는 예외 레벨보다 더 많은 권한을 가짐으로 인함) 수용되지 않고 보류로 남아 있음을 의미한다.

[0209] • n/a: 이러한 구성에서는 예외 레벨이 액세스 가능하지 않음을 의미한다.

[0210] 이러한 값은 위의 의사코드 및 도 8a 내지 도 8o에 도시된 함수에 따라 결정된다.

[0211] TMEA 및 PSTATE.A 이외의 임의의 제어 옵션은 선택적이며, 생략되는 경우 0b0으로 설정된 것과 동일한 방식으로 취급될 수 있다.

[0212] 도 10a 및 도 10b에서 유의해야 할 예시적인 라인은 다음을 포함한다:

[0213] • 라인 1: EL1 및 EL0에서의 에러 예외는 디폴트 예외 레벨 EL1로 수용되고 EL2 및 EL3에서는 디폴트 예외 레벨 EL1보다 더 많은 권한을 갖는 예외 레벨에 있기 때문에 암시적으로 마스킹된다. 트랩 제어가 설정되지 않았다.

[0214] • 라인 2: PSTATE.A를 1로 설정하는 것은 에러 예외가 EL1 및 EL0에서 마스킹되게 한다.

[0215] • 라인 3: NMEA EL1을 1로 설정하는 것은 PSTATE.A가 1이더라도 에러 예외의 마스킹을 방지한다.

[0216] • 라인 4 및 5: TMEA EL2를 1로 설정하는 것은 PSTATE.A가 1인 경우 에러 예외가 EL1 또는 EL0으로부터 EL2로 트랩핑되게 하지만, PSTATE.A가 0이면 거동은 여전히 라인 1에서와 같다.

[0217] • 라인 6: TMEA EL2 및 NMEA EL1 둘 모두가 1이고 PSTATE.A가 1인 경우에, NMEA = 1에 의해 야기되는 비-마스킹 가능 상태는 EL0에서 에러 예외가 발생하는 경우, EL2로의 트랩을 오버라이드하지만, EL1에서 에러 예외가 발생하면 그렇지 않고, 그에 따라 TMEA EL2, NMEA EL1 및 PSTATE.A가 모두 1이지만 다른 트랩 제어가 설정되지 않을 경우에, EL0에서 발생하는 에러 예외는 EL1에서 수용되지만 EL1에서 발생하는 에러 예외는 EL2에서 수용된다.

[0218] • 라인 7: NMEA EL2를 1로 설정하는 것은 라인 1 내지 6에서와 같이 암시적으로 마스킹되지 않고 EL2에서 발생하더라도 에러 예외가 EL2로 수용됨을 의미한다.

[0219] • 라인 13 내지 19 및 라인 63 내지 68: TMEA 제어는 EA EL3, TGE EL2 및 AMO EL2 제어 중 임의의 것이 설정되는 경우 영향을 미치지 않고, 이는, 이러한 경우에 EL2 또는 EL3으로의 에러 예외의 명시적 라우팅이 있기 때문이다.

[0220] • 라인 41: TMEA EL3과 TMEA EL2 둘 모두가 설정되고 PSTATE.A가 설정되면, EL0 및 EL1로부터의 TMEA 기반 트랩이 EL2, 즉 현재의 예외 레벨에 권한이 가장 가까운 TMEA 연관 예외 레벨로 수용되고, EL2로부터의 TMEA 기반 트랩이 EL3으로 수용된다. 라인 42는 NMEA EL1이 1이면 이것이 EL0으로부터의 TMEA 기반 트랩을 오버라이드하여, 대신 예외가 EL1에서 수용되도록 하는 방법을 나타낸다(라인 6과 유사).

[0221] 도 9a 내지 도 9i의 기능을 구현하기 위한 예시적인 의사코드가 아래에 나타난다:

```
// AArch64.SyncExternalAbortTarget()
// =================================
// Returns the target Exception level for a Synchronous External Data
// or Instruction Abort

bits(2) AArch64.SyncExternalAbortTarget(FaultRecord fault)

    boolean route_to_el3;
    boolean route_to_el2;

    // The exception is explicitly routed to EL3
    if PSTATE.EL != EL3 then
        route_to_el3 = (HaveEL(EL3) && SCR_EL3.EA == '1');
    else
        route_to_el3 = FALSE;

    // The exception is explicitly routed to EL2
    bit tea_bit = (if HaveRASExt() then HCR_EL2.TEA else '0');
    if !route_to_el3 && EL2Enabled() && PSTATE.EL == EL1 then
        route_to_el2 = (tea_bit == '1' ||
                        fault.acctype == AccType_NV2REGISTER ||
                        IsSecondStage(fault));
    elsif !route_to_el3 && EL2Enabled() && PSTATE.EL == EL0 then
        route_to_el2 = (!IsInHost() && (HCR_EL2.TGE == '1' ||
           tea_bit =='1' || IsSecondStage(fault)));
    else
        route_to_el2 = FALSE;

    boolean route_soft_masked_to_el3;
    boolean route_soft_masked_to_el2;
    if HaveDoubleFault2Ext() then
        // The soft masked exception is routed to EL2
        route_soft_masked_to_el2 = (EL2Enabled() && !route_to_el3 &&
                             (PSTATE.EL == EL1 && PSTATE.A == '1')
                         && (IsHCRXEL2Enabled() && HCRX_EL2.TMEA == '1'));

        // The soft masked exception is routed to EL3
        route_soft_masked_to_el3 =
                              (HaveEL(EL3) &&
                         !(route_to_el2 || route_soft_masked_to_el2) &&
```

[0222]

```
                    (PSTATE.EL IN {EL2, EL1} && PSTATE.A == '1') &&
                            SCR_EL3.TMEA == '1');
   else
       route_soft_masked_to_el2 = FALSE;
       route_soft_masked_to_el3 = FALSE;

   // The exception is routed away from the current exception regime
   routed_away = (route_to_el2 || route_soft_masked_to_el2 ||
                  route_to_el3 || route_soft_masked_to_el3);

   // The exception is taken at EL3
   take_in_el3 = PSTATE.EL == EL3;

   // The exception is taken at EL2 or in the Host EL0
take_in_el2_0 = ((PSTATE.EL == EL2 || IsInHost()) && !routed_away);

   // The exception is taken at EL1 or in the non-Host EL0
take_in_el1_0 = ((PSTATE.EL == EL1 ||
                    (PSTATE.EL == EL0 && !IsInHost()))
                    &&  !routed_away);

   bits(2) target_el;
   if take_in_el3 || route_to_el3 || route_soft_masked_to_el3 then
       target_el = EL3;
   elsif take_in_el2_0 || route_to_el2 || route_soft_masked_to_el2
      then
        target_el = EL2;
   elsif take_in_el1_0 then
       target_el = EL1;
   else
       Unreachable();

   return target_el;
```

[0223]

[0224] 도 11은 제어 정보의 값의 각각의 조합에 대해, 어느 예외 레벨이 예외 레벨(EL0, EL1, EL2, EL3) 중 임의의 것에서 발생하는 (비-마스킹 가능) 동기 외부 중단 예외에 대한 타겟 예외 레벨인지를 표시하는 요약 테이블을 도시한다. 열 TEA EL2는 HCR_EL2.TEA의 값을 지칭하며, 다른 열은 도 10a 및 도 10b와 동일한 의미를 갖는다. TMEA EL2가 1인 경우, PSTATE.A가 1일 때, EL1(EL0이 아님)로부터 EL2로의 트랩이 수용되지만, PSTATE.A가 0이면, EL1 및 EL0 둘 모두로부터 EL1에서 예외가 수용됨을 나타내는 라인 2 및 3에 유의한다.

[0225] 본원에 설명된 개념은 설명된 개념을 구현하는 장치의 제조를 위한 컴퓨터-판독 가능 코드로 구현될 수 있다. 예를 들어, 컴퓨터-판독 가능 코드는 개념을 구현하는 장치를 포함하는 집적 회로를 제조하기 위해 전자 설계 자동화(electronic design automation; EDA) 스테이지를 포함하는 반도체 설계 및 제조 프로세스의 하나 이상의 스테이지에서 사용될 수 있다. 위의 컴퓨터-판독 가능 코드는 추가적으로 또는 대안적으로 본원에 설명된 개념을 구현하는 장치의 정의, 모델링, 시뮬레이션, 검증 및/또는 테스트를 가능하게 할 수 있다.

[0226] 예를 들어, 본원에 설명된 개념을 구현하는 장치의 제조를 위한 컴퓨터-판독 가능 코드는 개념의 하드웨어 기술 언어(hardware description language; HDL) 표현을 정의하는 코드로 구현될 수 있다. 예를 들어, 코드는 개념을 구현하는 장치를 정의하기 위한 하나 이상의 논리 회로의 레지스터-전송-레벨(register-transfer-level; RTL) 개요(abstraction)를 정의할 수 있다. 코드는 FIRRTL과 같은 중간 표현뿐만 아니라 Verilog, SystemVerilog, Chisel, 또는 VHDL(Very High-Speed Integrated Circuit Hardware Description Language)로 장치를 구현하는 하나 이상의 논리 회로의 HDL 표현을 정의할 수 있다. 컴퓨터-판독 가능 코드는 SystemC 및 SystemVerilog와 같은 시스템 레벨 모델링 언어를 사용하여 개념을 구현하는 정의 또는 개념의 시뮬레이션, 기능적 및/또는 형식적 검증, 및 테스트를 가능하게 하기 위해 컴퓨터에 의해 해석될 수 있는 개념의 다른 거동 표현을 제공할 수 있다.

[0227] 추가적으로 또는 대안적으로, 컴퓨터-판독 가능 코드는 GDSII와 같은 표현을 포함하는 하나 이상의 넷리스트 또는 집적 회로 레이아웃 정의와 같이 본원에 설명된 개념을 구현하는 집적 회로 컴포넌트의 하위 레벨 설명을 정의할 수 있다. 집적 회로 컴포넌트의 하나 이상의 넷리스트 또는 다른 컴퓨터-판독 가능 표현은 본 발명을 구현하는 장치의 제조에 사용하기 위한 정의를 생성하기 위해 하나 이상의 논리 합성 프로세스를 RTL 표현에 적용함으로써 생성될 수 있다. 대안적으로 또는 추가적으로, 하나 이상의 논리 합성 프로세스는 설명된 개념을 구

현하도록 FPGA를 구성하기 위해 필드 프로그래밍 가능 게이트 어레이(FPGA) 내에 로딩될 비트스트림을 컴퓨터-판독 가능 코드로부터 생성할 수 있다. FPGA는 집적 회로에서 제조하기 전에 개념의 검증 및 테스트 목적으로 배치되거나 FPGA가 제품에 직접적으로 배치될 수 있다.

[0228] 컴퓨터-판독 가능 코드는, 예를 들어, 본 발명을 구현하는 장치를 제조하기 위해 반도체 설계 및 제조 프로세스에서 사용될 RTL 표현, 넷리스트 표현, 또는 또 다른 컴퓨터-판독 가능 정의 중 하나 이상의 혼합을 포함하는, 장치의 제조를 위한 코드 표현의 혼합을 포함할 수 있다. 대안적으로 또는 추가적으로, 개념은 반도체 설계 및 장치를 제조하기 위한 제조 프로세스에서 사용될 컴퓨터-판독 가능 정의와, 일단 제조되면 정의된 장치에 의해 실행될 명령어를 정의하는 컴퓨터-판독 가능 코드의 조합으로 정의될 수 있다.

[0229] 그러한 컴퓨터-판독 가능 코드는 임의의 알려진 일시적 컴퓨터-판독 가능 매체(예컨대, 네트워크를 거치는 코드의 유선 또는 무선 송신) 또는 반도체, 자기 디스크 또는 광학 디스크와 같은 비일시적 컴퓨터-판독 가능 매체에 배치될 수 있다. 컴퓨터-판독 가능 코드를 사용하여 제조된 집적 회로는 중앙 프로세싱 유닛, 그래픽 프로세싱 유닛, 신경 프로세싱 유닛, 디지털 신호 프로세서, 또는 개별적으로 또는 집합적으로 개념을 구현하는 다른 컴포넌트 중 하나 이상과 같은 컴포넌트를 포함할 수 있다.

[0230] 도 12는 사용될 수 있는 시뮬레이터 구현을 예시한다. 앞서 설명한 실시형태는 관련 기법을 지원하는 특정 프로세싱 하드웨어를 동작시키기 위한 장치 및 방법의 관점에서 본 발명을 구현하지만, 본원에 설명된 실시형태에 따라 컴퓨터 프로그램의 사용을 통해 구현되는 명령어 실행 환경을 제공하는 것이 또한 가능하다. 이러한 컴퓨터 프로그램은, 이들이 하드웨어 아키텍처의 소프트웨어 기반 구현을 제공하는 한, 종종 시뮬레이터로 지칭된다. 다양한 시뮬레이터 컴퓨터 프로그램은 에뮬레이터, 가상 머신, 모델, 및 동적 이진 변환기를 비롯한, 이진 변환기를 포함한다.전형적으로, 시뮬레이터 구현은 호스트 프로세서(330) 상에서 실행될 수 있으며, 선택적으로 호스트 운영 체제(320)를 실행하여, 시뮬레이터 프로그램(310)을 지원할 수 있다. 일부 배열에서, 하드웨어와 제공된 명령어 실행 환경 사이에 다수의 시뮬레이션 계층이 있을 수 있고/있거나, 동일한 호스트 프로세서 상에 제공된 다수의 별개의 명령어 실행 환경이 있을 수 있다.역사적으로, 합리적인 속도로 실행되는 시뮬레이터 구현을 제공하기 위해 강력한 프로세서가 요구되었지만, 그러한 접근법은 특정 상황에서, 예컨대 호환성 또는 재사용을 이유로 다른 프로세서에 기본적인(native) 코드를 실행하기를 바라는 경우, 정당화될 수 있다.예를 들어, 시뮬레이터 구현은 호스트 프로세서 하드웨어에 의해 지원되지 않는 추가적인 기능성을 갖는 명령어 실행 환경을 제공하거나, 상이한 하드웨어 아키텍처와 전형적으로 관련되는 명령어 실행 환경을 제공할 수 있다. 시뮬레이션의 개요는 문헌["Some Efficient Architecture Simulation Techniques", Robert Bedichek, Winter 1990 USENIX Conference, Pages 53 - 63]에 제공되어 있다.

[0231] 실시형태가 특정 하드웨어 작제 또는 특징을 참조하여 전술된 경우, 시뮬레이션된 실시형태에서 동등한 기능성이 적합한 소프트웨어 작제 또는 특징에 의해 제공될 수 있다. 예를 들어, 특정 회로부는 시뮬레이션된 실시형태에 컴퓨터 프로그램 로직으로서 구현될 수 있다. 마찬가지로, 메모리 하드웨어, 예컨대 레지스터 또는 캐시는 시뮬레이션된 실시형태에서 소프트웨어 데이터 구조로서 구현될 수 있다. 앞서 설명된 실시형태에서 언급된 하드웨어 요소 중 하나 이상이 호스트 하드웨어(예를 들어, 호스트 프로세서(330)) 상에 존재하는 배열에서, 일부 시뮬레이션된 실시형태는 적합한 경우 호스트 하드웨어를 사용할 수 있다.

[0232] 시뮬레이터 프로그램(310)은 컴퓨터-판독 가능 저장 매체(이는 비일시적 매체일 수 있음) 상에 저장될 수 있고, 타겟 코드(300)(이는 애플리케이션, 운영 체제, 및 하이퍼바이저를 포함할 수 있음)에 프로그램 인터페이스(명령어 실행 환경)를 제공하는데, 이는 시뮬레이터 프로그램(310)에 의해 모델링되는 하드웨어 아키텍처의 애플리케이션 프로그램 인터페이스와 동일하다. 따라서, 타겟 코드(300)의 프로그램 명령어는 시뮬레이터 프로그램(310)을 사용하는 명령어 실행 환경 내로부터 실행될 수 있고, 그에 따라 위에서 논의된 장치(2)의 하드웨어 특징을 실제로 갖지 않는 호스트 컴퓨터(330)가 이러한 특징을 에뮬레이팅할 수 있다. 따라서, 시뮬레이터 프로그램(310)은 타겟 프로세싱 회로부의 존재를 시뮬레이션한다.

[0233] 시뮬레이터 코드(310)는 명령어 디코딩 프로그램 로직(312), 프로세싱 프로그램 로직(314) 및 예외 제어 프로그램 로직(318)을 포함하며, 이들은 위에서 논의된 명령어 디코더(10), 프로세싱 회로부(4) 및 예외 제어 회로부(40)에 대한 기능성에 대응한다. 또한, 레지스터 에뮬레이션 데이터 구조(316)는, 시뮬레이터 코드(310)에 의해 시뮬레이션되는 타겟 프로세서의 명령어 세트 아키텍처에 제공된 레지스터(14)를 에뮬레이션하기 위해 시뮬레이터 코드(310)에 의해 유지될 수 있다. 예를 들어, 레지스터 에뮬레이션 데이터 구조(316)는 호스트 데이터 프로세싱 장치의 메모리에 저장될 수 있고, 도 3에 도시된 바와 같이 제어 정보를 저장하기 위한 레지스터를 포함하는 레지스터(14)에 대응하는 데이터를 저장하는 저장 위치를 포함할 수 있다. 예외 제어 프로그램 로직

(318)은, 위에서 설명된 예외 제어 회로부(40)에 의한 예외에 대한 응답에 대응하는 방식으로 시뮬레이터 코드(310)에 의해 표현된 타겟 프로세서를 타겟팅하는 시뮬레이션된 예외에 응답할 수 있으며, 이는, 예외가 마스킹되는지 여부를 제어하는 것 및 트랩-마스킹된-예외 제어 정보 및 위에서 논의된 바와 같은 다른 제어 정보에 기초하여 트랩핑하는 것을 제어하는 것을 포함한다(시뮬레이션에서, 예외 라우팅 제어 정보는 레지스터 에뮬레이팅 데이터 구조(316)의 일부로서 유지될 수 있음). 따라서, 시뮬레이터 코드(310)는 앞선 예에서 설명된 프로세싱 장치(2)의 물리적인 하드웨어 구현에서 달성되는 것과 유사한 이점을 제공할 수 있다.

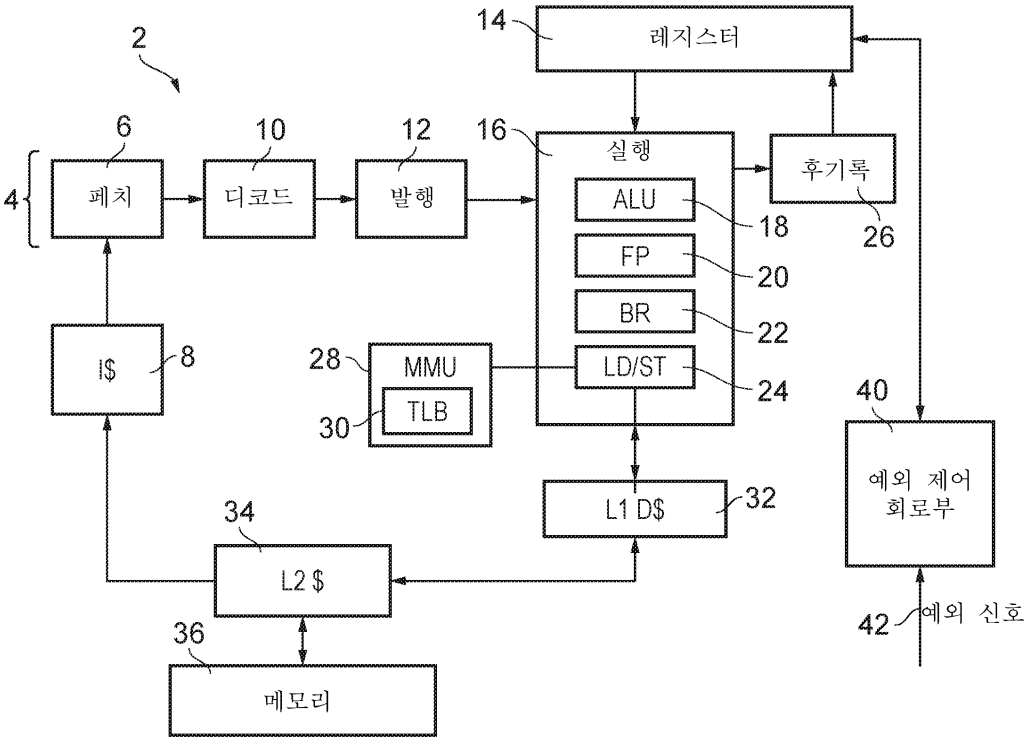
[0234] 본 출원에서, "...하도록 구성된"이라는 말은 장치의 요소가 정의된 동작을 수행할 수 있는 구성을 갖는다는 것을 의미하는 데 사용된다. 이러한 맥락에서, "구성"은 하드웨어 또는 소프트웨어의 상호연결(interconnection)의 배열 또는 방식을 의미한다. 예를 들어, 장치는 정의된 동작을 제공하는 전용 하드웨어를 가질 수 있거나, 프로세서 또는 다른 프로세싱 디바이스는 기능을 수행하도록 프로그래밍될 수 있다."하도록 구성된"은, 장치 요소가, 정의된 동작을 제공하기 위해 임의의 방식으로 변경될 필요가 있음을 암시하지는 않는다.

[0235] 본 출원에서, "중 적어도 하나"라는 어구가 앞에 오는 특징의 목록은 그러한 특징 중 임의의 하나 이상이 개별적으로 또는 조합하여 제공될 수 있음을 의미한다. 예를 들어, "[A], [B] 및 [C] 중 적어도 하나"는 다음 옵션 중 임의의 것을 포괄한다: A 단독(B 또는 C 제외), B 단독(A 또는 C 제외), C 단독(A 또는 B 제외), A와 B의 조합(C 제외), A와 C의 조합(B 제외), B와 C의 조합(A 제외) 또는 A, B 및 C의 조합.

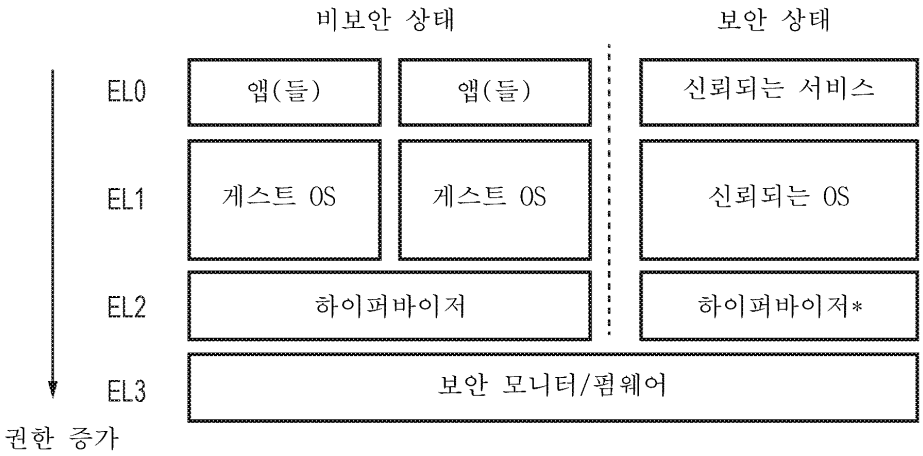
[0236] 본 발명의 예시적인 실시형태가 첨부 도면을 참조하여 본원에서 상세히 설명되었지만, 본 발명은 그러한 정확한 실시형태로 제한되지 않으며, 첨부된 청구범위에 의해 정의된 바와 같은 본 발명의 범위로부터 벗어남이 없이 다양한 변경 및 수정이 당업자에 의해 이루어질 수 있다는 것이 이해되어야 한다.

도면

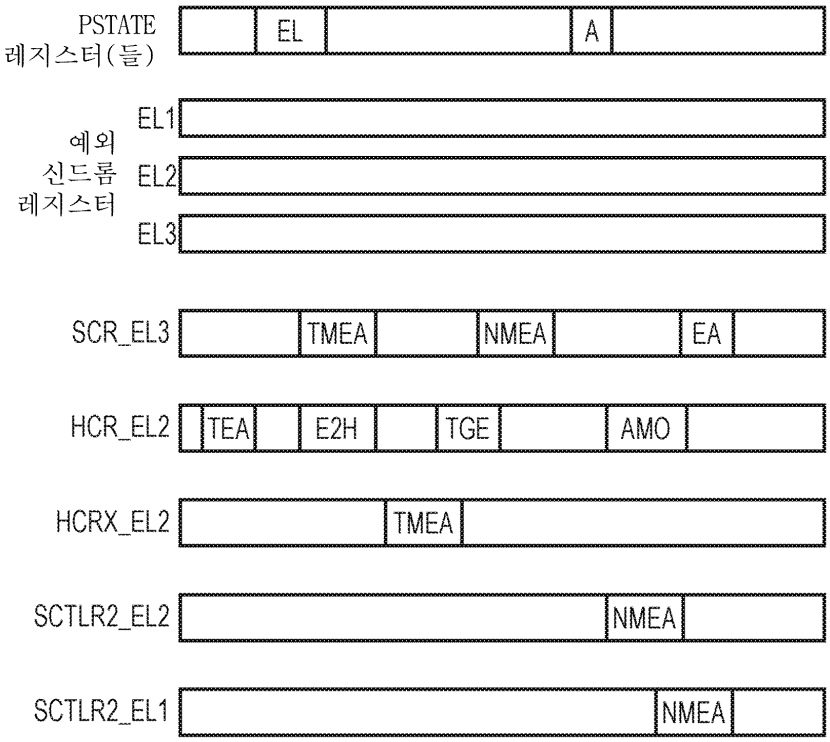
도면1



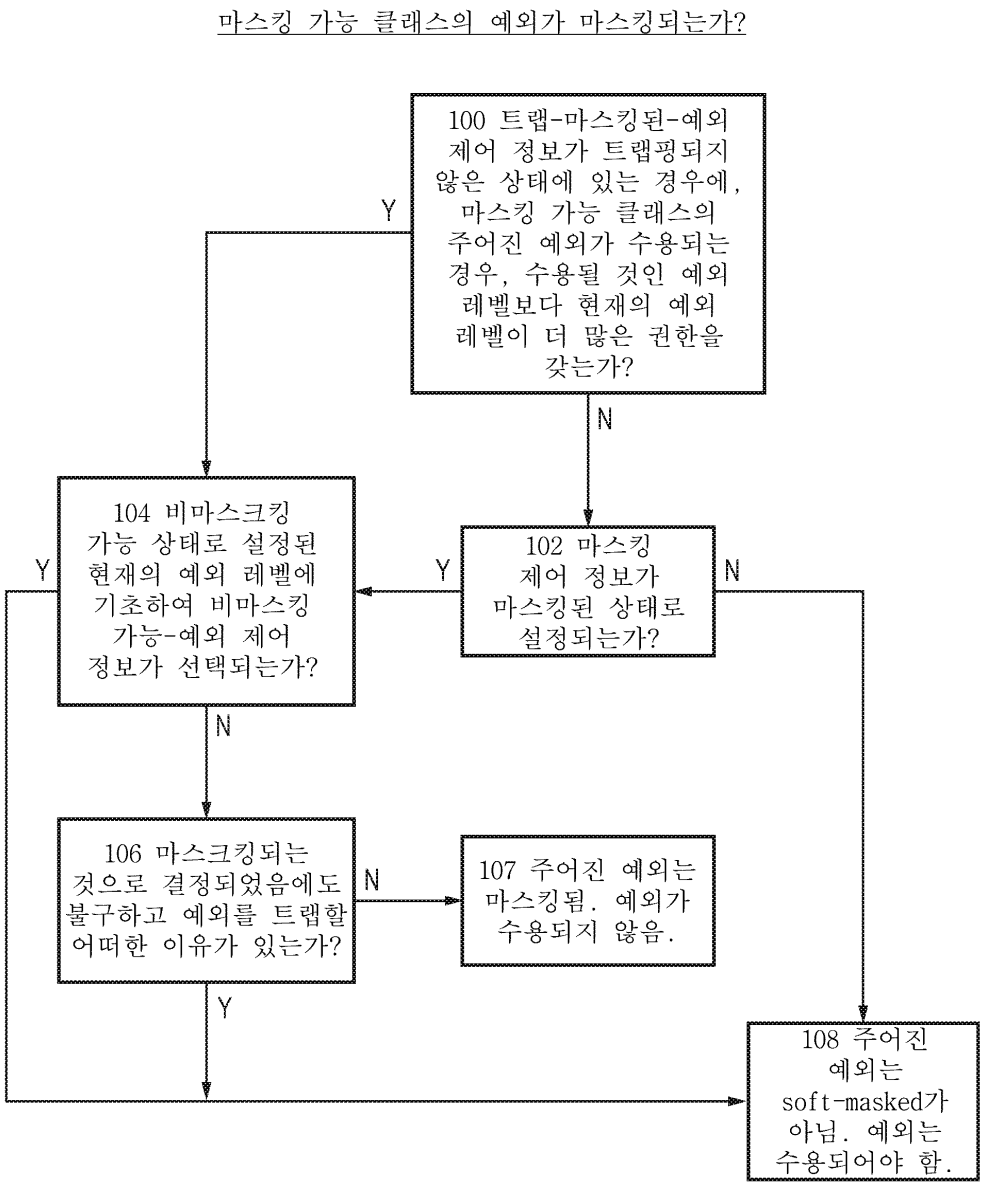
도면2


비보안 상태
보안 상태
EL0
앱(들)
앱(들)
신뢰되는 서비스
EL1
게스트 OS
게스트 OS
신뢰되는 OS
EL2
하이퍼바이저
하이퍼바이저*
EL3
보안 모니터/펌웨어
권한 증가

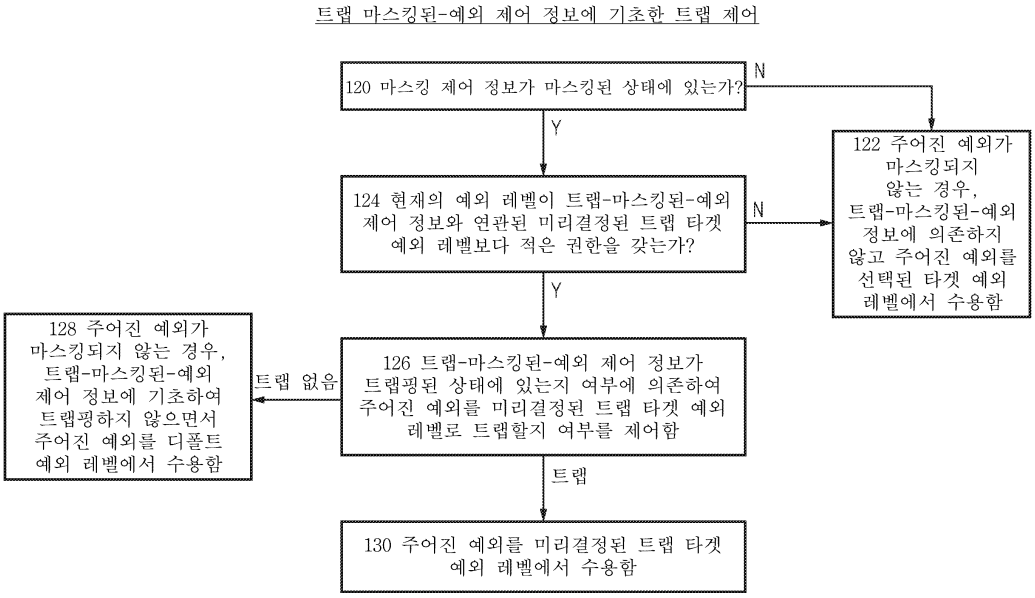
도면3


PSTATE 레지스터(들)
EL
A
예외 신드롬 레지스터
EL1
EL2
EL3
SCR_EL3
TMEA
NMEA
EA
HCR_EL2
TEA
E2H
TGE
AMO
HCRX_EL2
TMEA
SCTLR2_EL2
NMEA
SCTLR2_EL1
NMEA

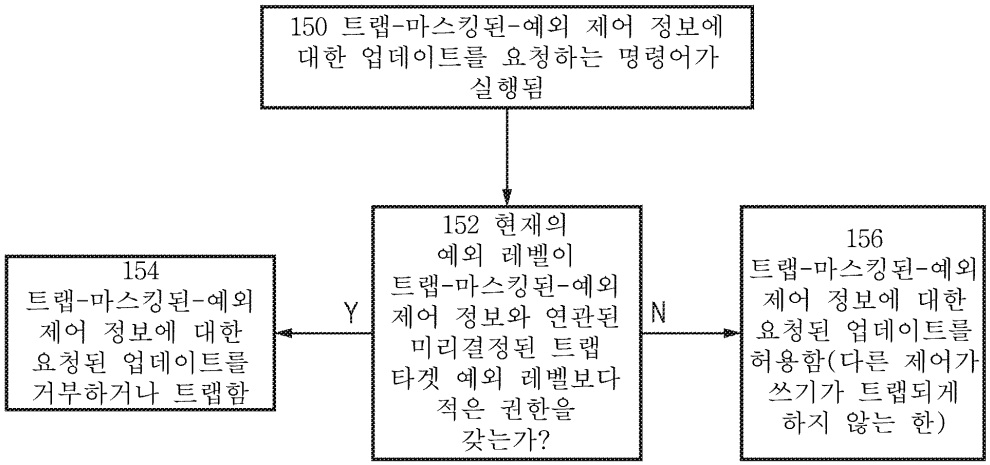
도면4


마스킹 가능 클래스의 예외가 마스킹되는가?
100 트랩-마스킹된-예외 제어 정보가 트랩핑되지 않은 상태에 있는 경우에, 마스킹 가능 클래스의 주어진 예외가 수용되는 경우, 수용될 것인 예외 레벨보다 현재의 예외 레벨이 더 많은 권한을 갖는가?
Y
N
102 마스킹 제어 정보가 마스킹된 상태로 설정되는가?
Y
N
104 비마스크킹 가능 상태로 설정된 현재의 예외 레벨에 기초하여 비마스킹 가능-예외 제어 정보가 선택되는가?
Y
N
106 마스크킹되는 것으로 결정되었음에도 불구하고 예외를 트랩할 어떠한 이유가 있는가?
N
107 주어진 예외는 마스킹됨. 예외가 수용되지 않음.
Y
108 주어진 예외는 soft-masked가 아님. 예외는 수용되어야 함.

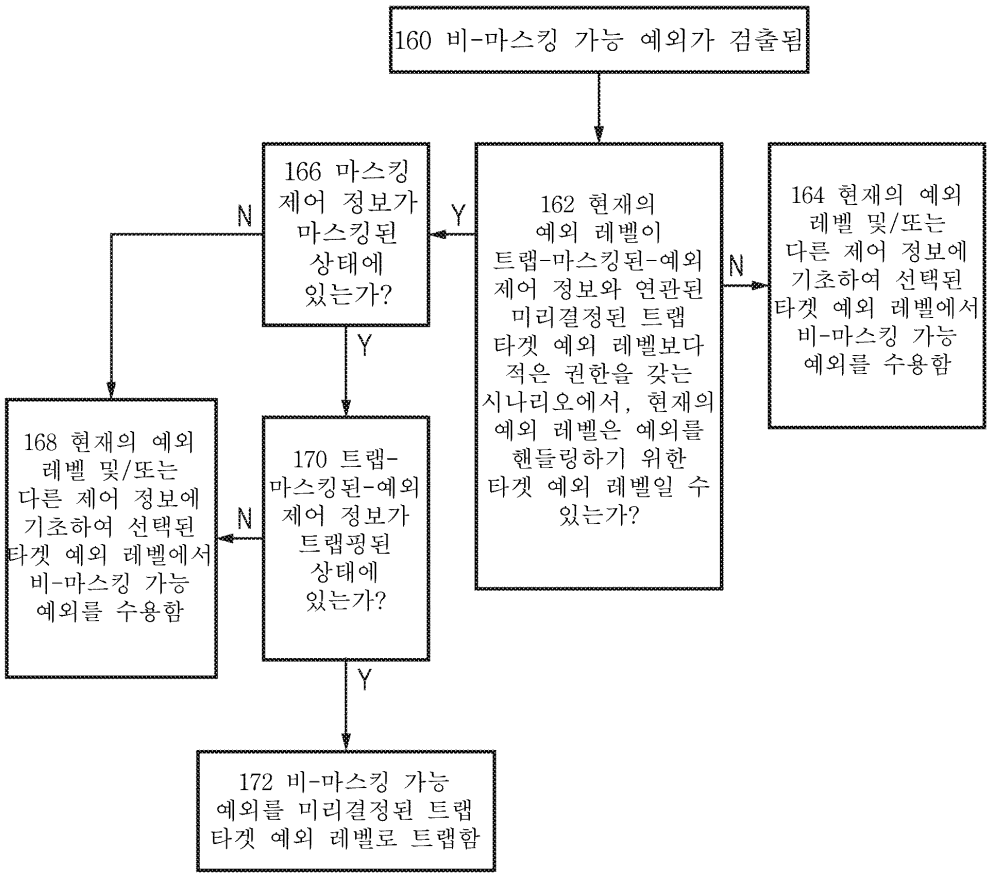
도면5


트랩 마스킹된-예외 제어 정보에 기초한 트랩 제어
120 마스킹 제어 정보가 마스킹된 상태에 있는가?
N
Y
122 주어진 예외가 마스킹되지 않는 경우, 트랩-마스킹된-예외 정보에 의존하지 않고 주어진 예외를 선택된 타겟 예외 레벨에서 수용함
124 현재의 예외 레벨이 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨보다 적은 권한을 갖는가?
N
Y
128 주어진 예외가 마스킹되지 않는 경우, 트랩-마스킹된-예외 제어 정보에 기초하여 트랩핑하지 않으면서 주어진 예외를 디폴트 예외 레벨에서 수용함
트랩 없음
126 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있는지 여부에 의존하여 주어진 예외를 미리결정된 트랩 타겟 예외 레벨로 트랩할지 여부를 제어함
트랩
130 주어진 예외를 미리결정된 트랩 타겟 예외 레벨에서 수용함

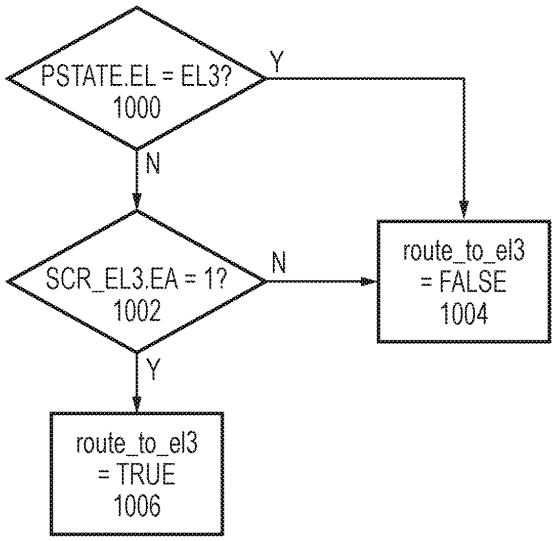
도면6


150 트랩-마스킹된-예외 제어 정보에 대한 업데이트를 요청하는 명령어가 실행됨
152 현재의 예외 레벨이 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨보다 적은 권한을 갖는가?
Y
154 트랩-마스킹된-예외 제어 정보에 대한 요청된 업데이트를 거부하거나 트랩함
N
156 트랩-마스킹된-예외 제어 정보에 대한 요청된 업데이트를 허용함(다른 제어가 쓰기가 트랩되게 하지 않는 한)

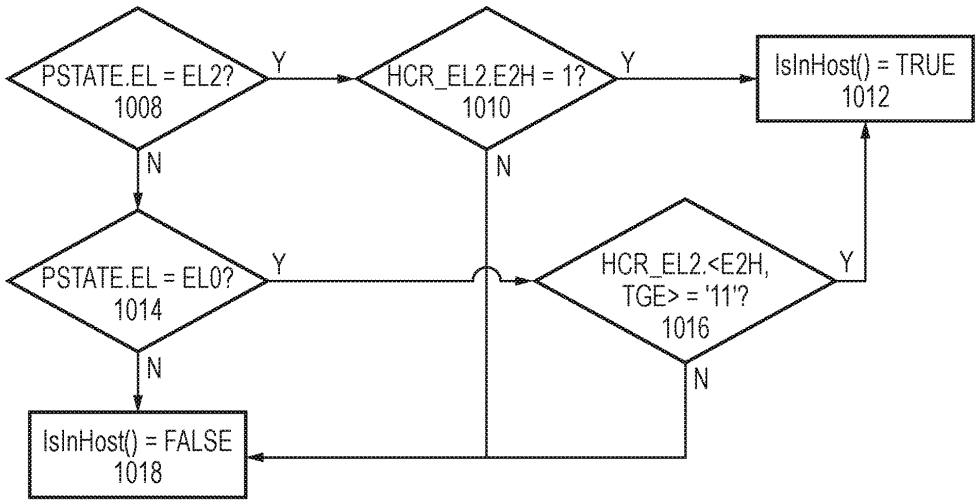
도면7


160 비-마스킹 가능 예외가 검출됨
162 현재의 예외 레벨이 트랩-마스킹된-예외 제어 정보와 연관된 미리결정된 트랩 타겟 예외 레벨보다 적은 권한을 갖는 시나리오에서, 현재의 예외 레벨은 예외를 핸들링하기 위한 타겟 예외 레벨일 수 있는가?
Y
N
164 현재의 예외 레벨 및/또는 다른 제어 정보에 기초하여 선택된 타겟 예외 레벨에서 비-마스킹 가능 예외를 수용함
166 마스킹 제어 정보가 마스킹된 상태에 있는가?
Y
N
168 현재의 예외 레벨 및/또는 다른 제어 정보에 기초하여 선택된 타겟 예외 레벨에서 비-마스킹 가능 예외를 수용함
170 트랩-마스킹된-예외 제어 정보가 트랩핑된 상태에 있는가?
Y
N
172 비-마스킹 가능 예외를 미리결정된 트랩 타겟 예외 레벨로 트랩함

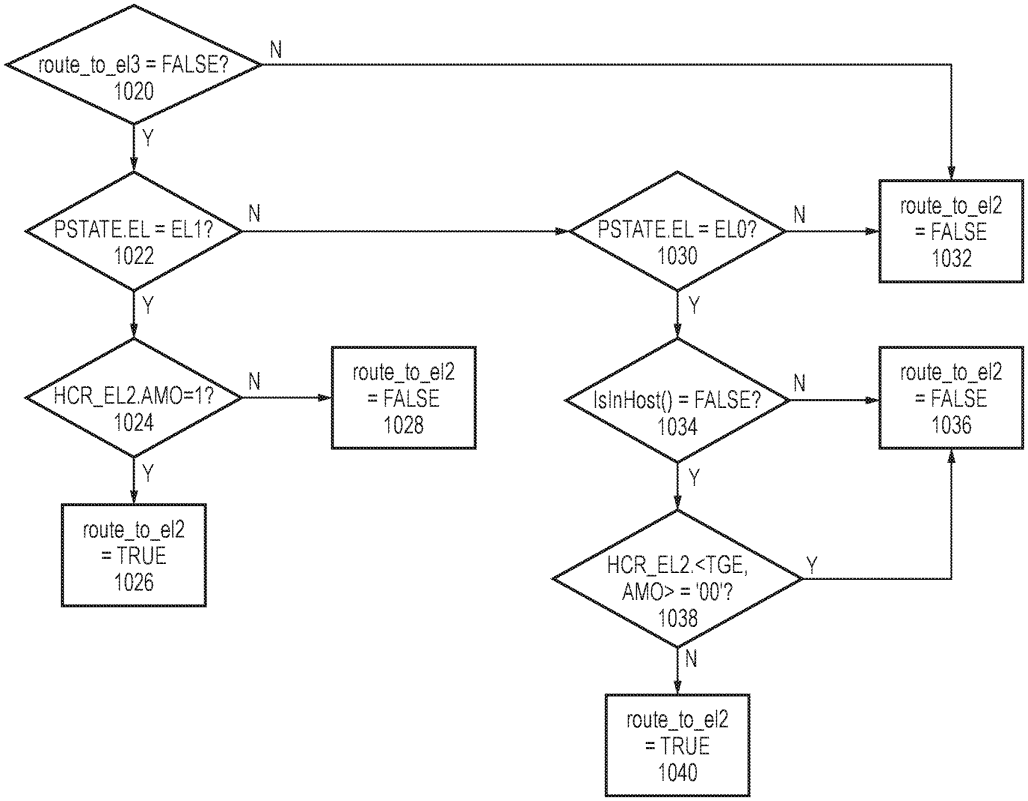
도면8a


PSTATE.EL = EL3?
1000
Y
N
SCR_EL3.EA = 1?
1002
N
Y
route_to_el3
= FALSE
1004
route_to_el3
= TRUE
1006

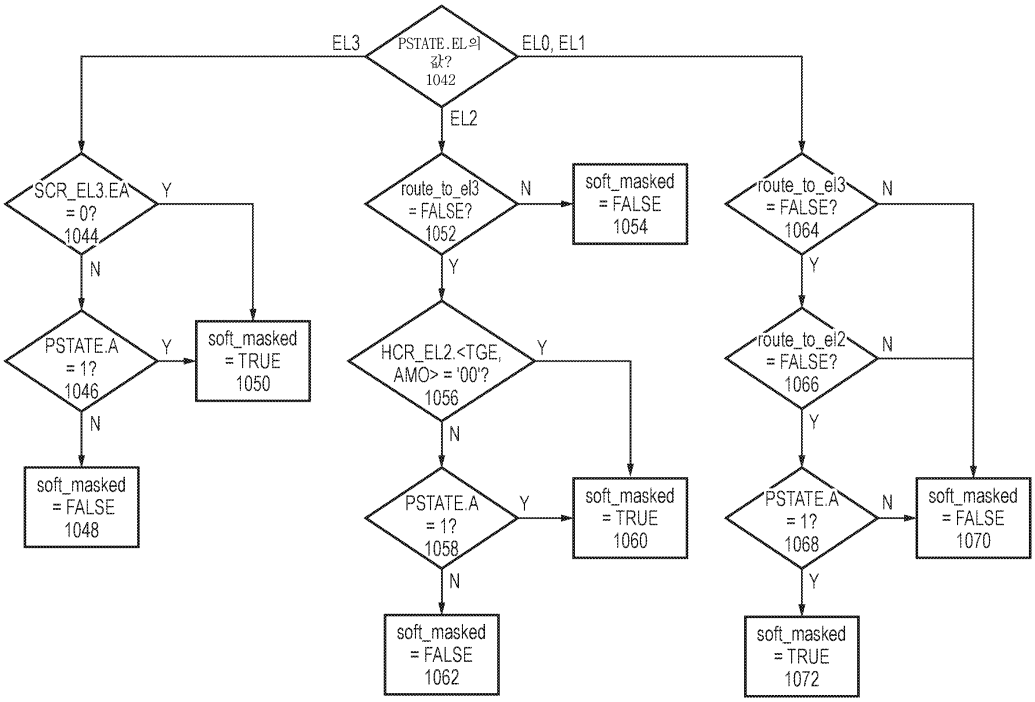
도면8b


PSTATE.EL = EL2?
1008
Y
HCR_EL2.E2H = 1?
1010
Y
IsInHost() = TRUE
1012
N
N
PSTATE.EL = EL0?
1014
Y
HCR_EL2.<E2H, TGE> = '11'?
1016
Y
N
N
IsInHost() = FALSE
1018

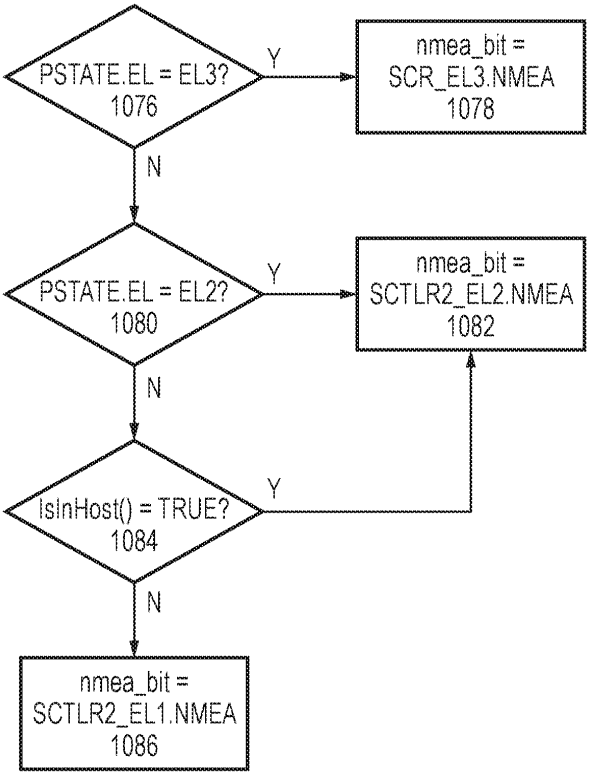
도면8c


route_to_el3 = FALSE?
1020
N
Y
PSTATE.EL = EL1?
1022
N
PSTATE.EL = EL0?
1030
N
route_to_el2 = FALSE
1032
Y
Y
HCR_EL2.AMO=1?
1024
N
route_to_el2 = FALSE
1028
IsInHost() = FALSE?
1034
N
route_to_el2 = FALSE
1036
Y
Y
route_to_el2 = TRUE
1026
HCR_EL2.<TGE, AMO> = '00'?
1038
Y
N
route_to_el2 = TRUE
1040

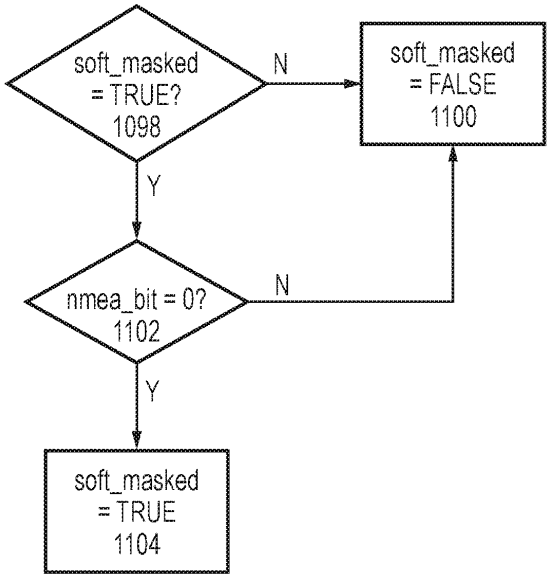
도면8d


PSTATE.EL의 값? 1042
EL3
EL2
EL0, EL1
SCR_EL3.EA = 0? 1044
Y
N
PSTATE.A = 1? 1046
Y
N
soft_masked = TRUE 1050
soft_masked = FALSE 1048
route_to_el3 = FALSE? 1052
N
Y
soft_masked = FALSE 1054
HCR_EL2.<TGE, AMO> = '00'? 1056
Y
N
PSTATE.A = 1? 1058
Y
N
soft_masked = TRUE 1060
soft_masked = FALSE 1062
route_to_el3 = FALSE? 1064
N
Y
route_to_el2 = FALSE? 1066
N
Y
PSTATE.A = 1? 1068
N
Y
soft_masked = FALSE 1070
soft_masked = TRUE 1072

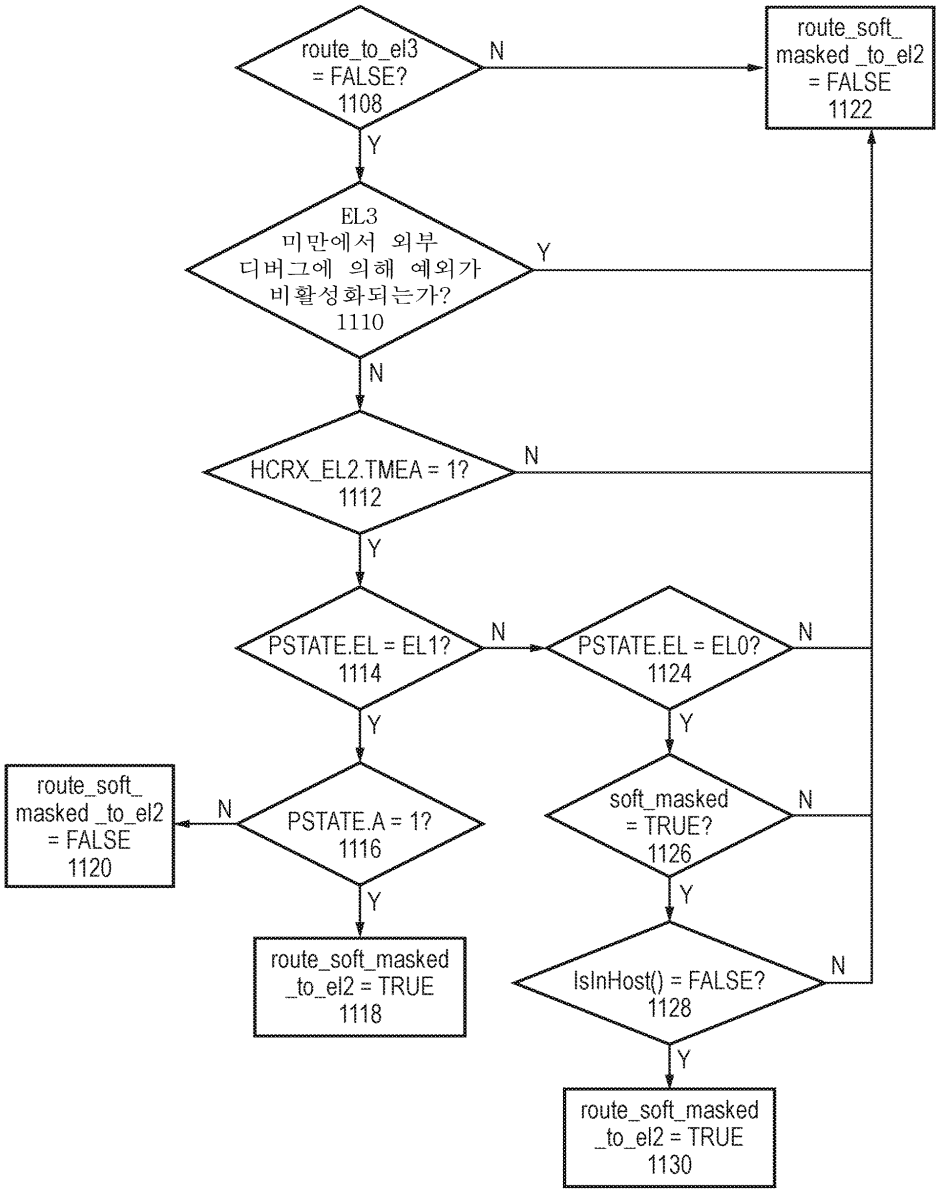
도면8e


PSTATE.EL = EL3? 1076
Y
N
nmea_bit = SCR_EL3.NMEA 1078
PSTATE.EL = EL2? 1080
Y
N
nmea_bit = SCTLR2_EL2.NMEA 1082
IsInHost() = TRUE? 1084
Y
N
nmea_bit = SCTLR2_EL1.NMEA 1086

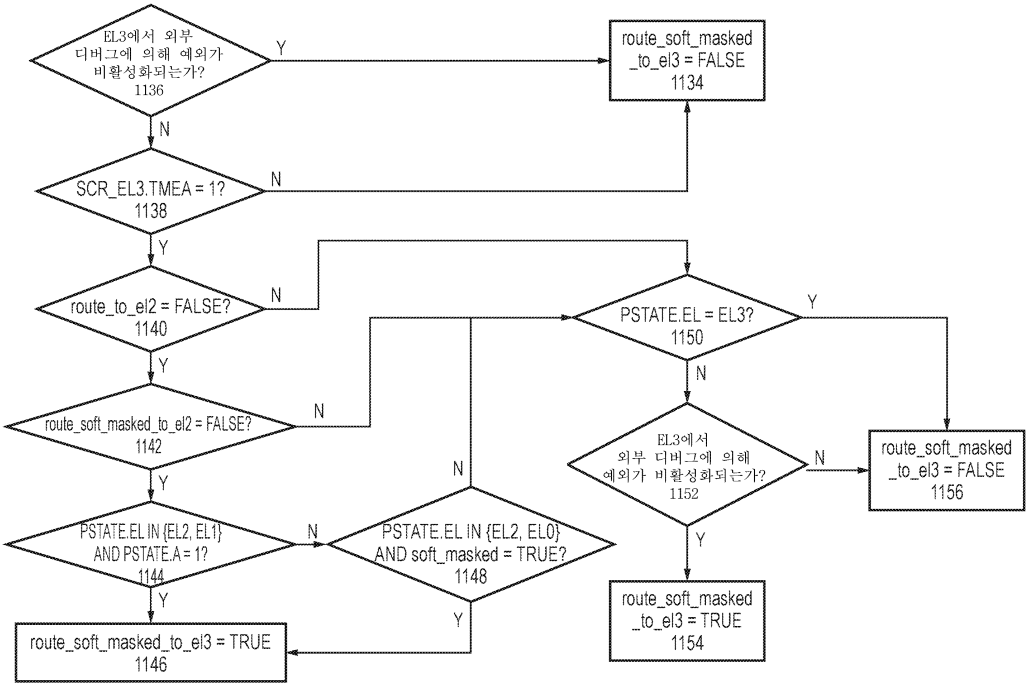
도면8f


soft_masked
= TRUE?
1098
N
soft_masked
= FALSE
1100
Y
nmea_bit = 0?
1102
N
Y
soft_masked
= TRUE
1104

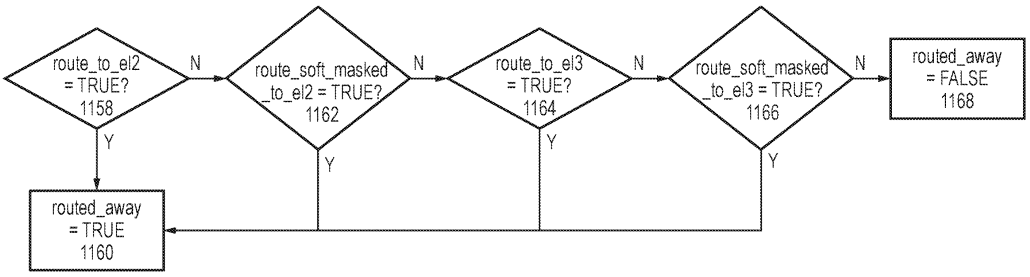
도면8g


route_to_el3 = FALSE? 1108
N
route_soft_ masked _to_el2 = FALSE 1122
Y
EL3 미만에서 외부 디버그에 의해 예외가 비활성화되는가? 1110
Y
N
HCRX_EL2.TMEA = 1? 1112
N
Y
PSTATE.EL = EL1? 1114
N
PSTATE.EL = EL0? 1124
N
Y
Y
route_soft_ masked _to_el2 = FALSE 1120
N
PSTATE.A = 1? 1116
soft_masked = TRUE? 1126
N
Y
Y
route_soft_masked _to_el2 = TRUE 1118
IsInHost() = FALSE? 1128
N
Y
route_soft_masked _to_el2 = TRUE 1130

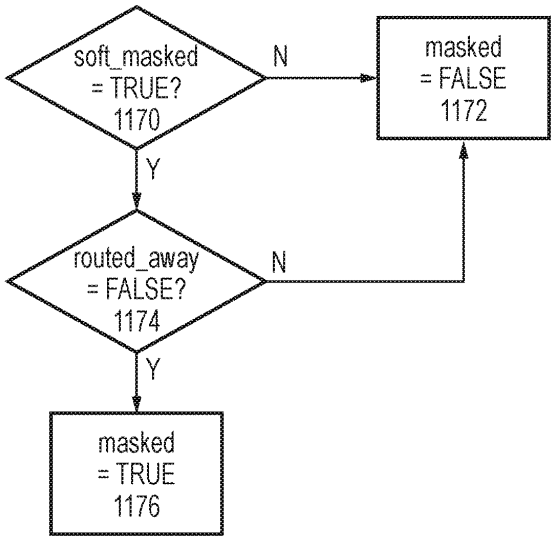
도면8h



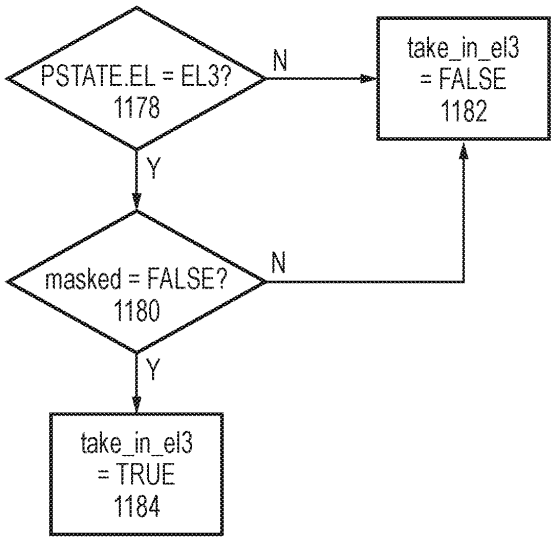
도면8i



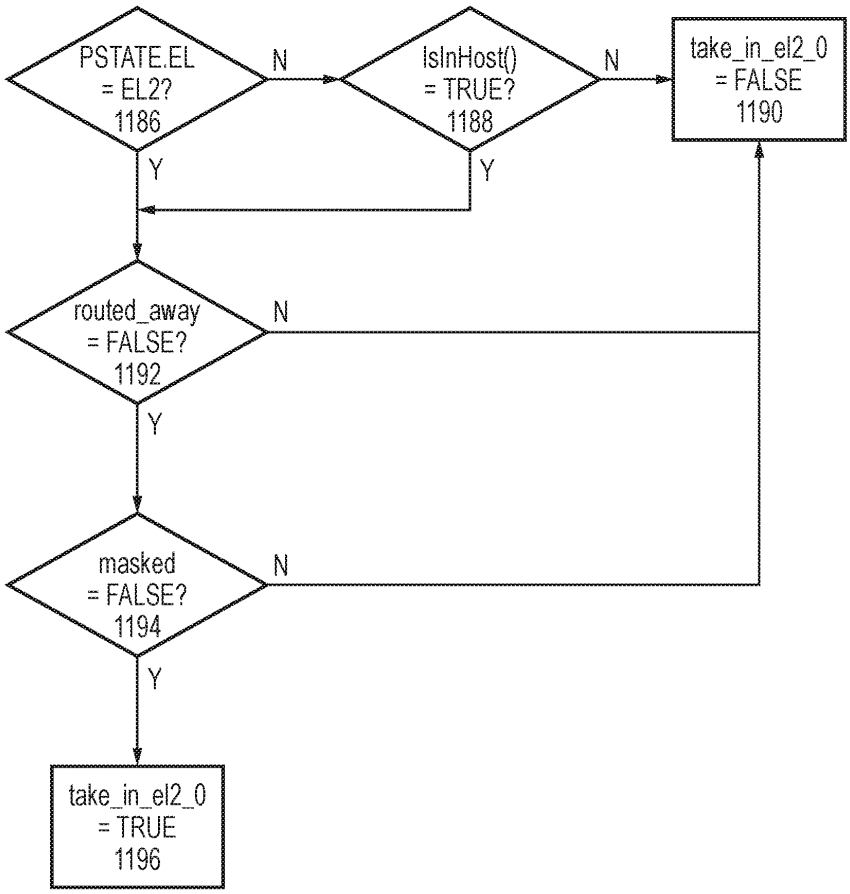
도면8j



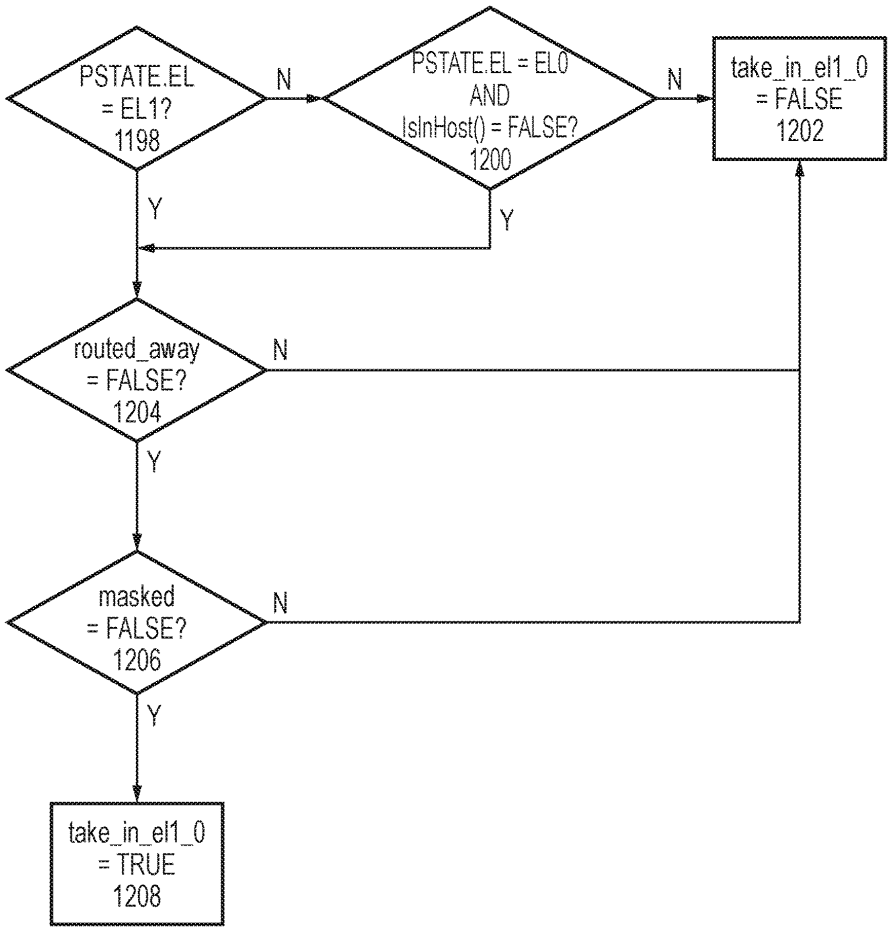
도면8k


PSTATE.EL = EL3?
1178
N
take_in_el3
= FALSE
1182
Y
masked = FALSE?
1180
N
Y
take_in_el3
= TRUE
1184

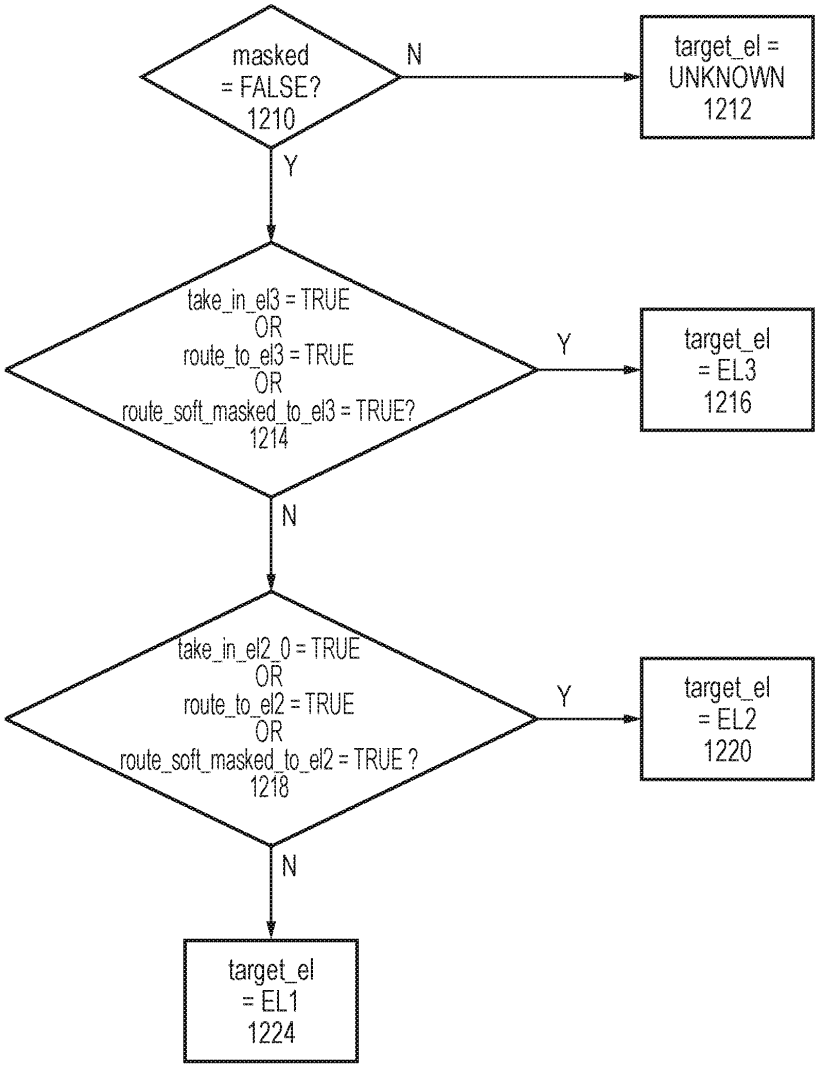
도면81


PSTATE.EL
= EL2?
1186
N
IsInHost()
= TRUE?
1188
N
take_in_el2_0
= FALSE
1190
Y
Y
routed_away
= FALSE?
1192
N
Y
masked
= FALSE?
1194
N
Y
take_in_el2_0
= TRUE
1196

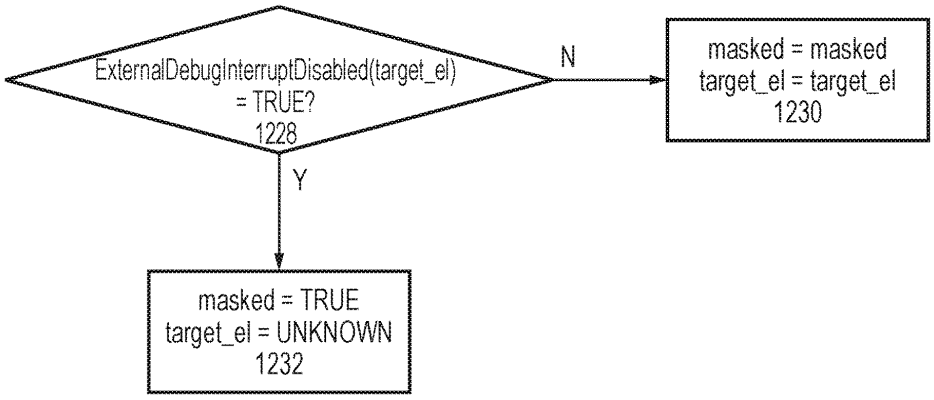
도면8m


PSTATE.EL
= EL1?
1198
N
PSTATE.EL = EL0
AND
IsInHost() = FALSE?
1200
N
take_in_el1_0
= FALSE
1202
Y
Y
routed_away
= FALSE?
1204
N
Y
masked
= FALSE?
1206
N
Y
take_in_el1_0
= TRUE
1208

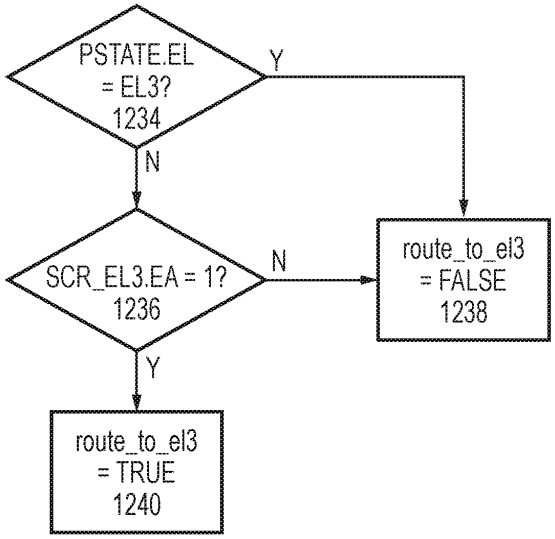
도면8n


masked
= FALSE?
1210
N
target_el =
UNKNOWN
1212
Y
take_in_el3 = TRUE
OR
route_to_el3 = TRUE
OR
route_soft_masked_to_el3 = TRUE?
1214
Y
target_el
= EL3
1216
N
take_in_el2_0 = TRUE
OR
route_to_el2 = TRUE
OR
route_soft_masked_to_el2 = TRUE ?
1218
Y
target_el
= EL2
1220
N
target_el
= EL1
1224

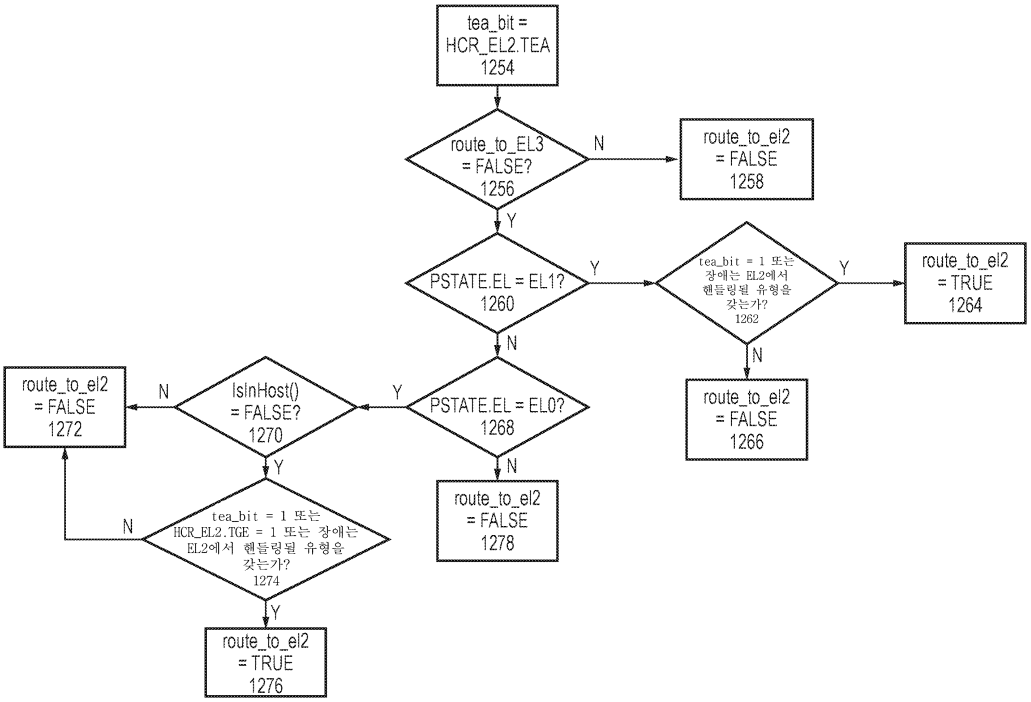
도면8o


ExternalDebugInterruptDisabled(target_el)
= TRUE?
1228
N
masked = masked
target_el = target_el
1230
Y
masked = TRUE
target_el = UNKNOWN
1232

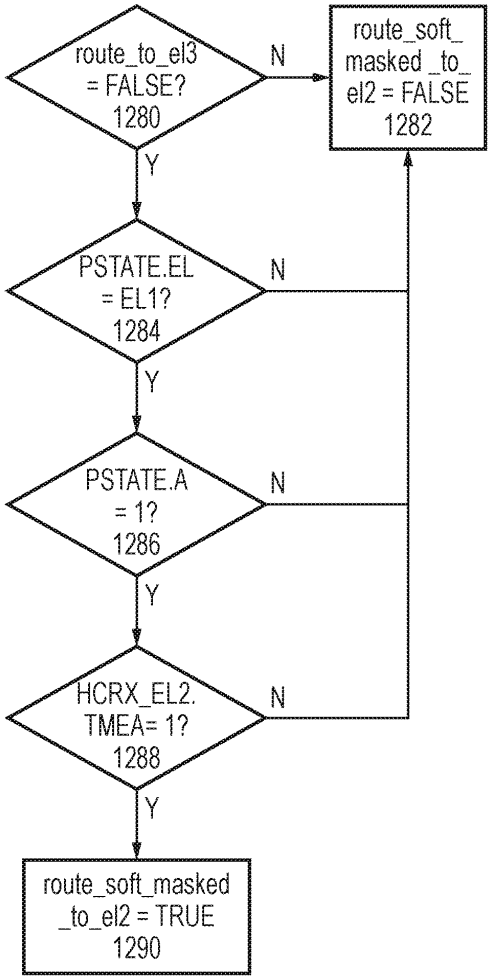
도면9a


PSTATE.EL
= EL3?
1234
Y
N
SCR_EL3.EA = 1?
1236
N
route_to_el3
= FALSE
1238
Y
route_to_el3
= TRUE
1240

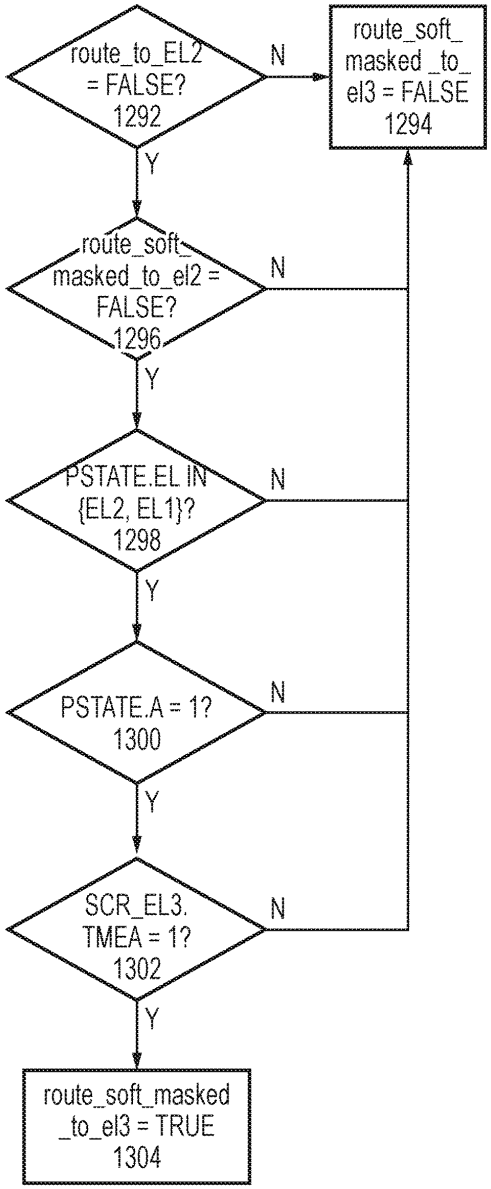
도면9b


tea_bit =
HCR_EL2.TEA
1254
route_to_EL3
= FALSE?
1256
N
route_to_el2
= FALSE
1258
Y
PSTATE.EL = EL1?
1260
Y
tea_bit = 1 또는
장애는 EL2에서
핸들링될 유형을
갖는가?
1262
Y
route_to_el2
= TRUE
1264
N
route_to_el2
= FALSE
1266
N
PSTATE.EL = EL0?
1268
Y
IsInHost()
= FALSE?
1270
N
route_to_el2
= FALSE
1272
N
tea_bit = 1 또는
HCR_EL2.TGE = 1 또는 장애는
EL2에서 핸들링될 유형을
갖는가?
1274
Y
route_to_el2
= TRUE
1276
Y
route_to_el2
= FALSE
1278
N

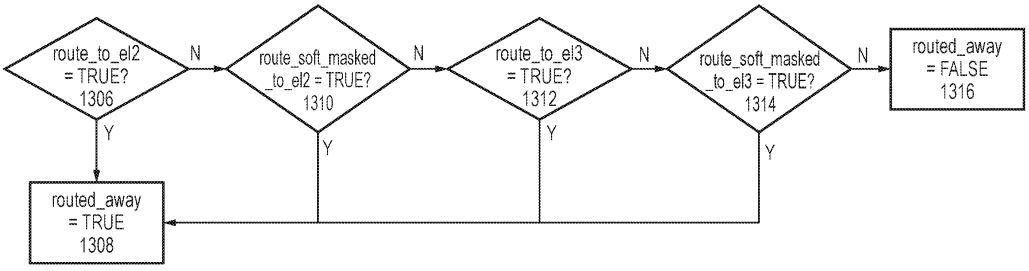
도면9c


route_to_el3 = FALSE? 1280
N
route_soft_ masked _to_ el2 = FALSE 1282
Y
PSTATE.EL = EL1? 1284
N
Y
PSTATE.A = 1? 1286
N
Y
HCRX_EL2. TMEA= 1? 1288
N
Y
route_soft_masked _to_el2 = TRUE 1290

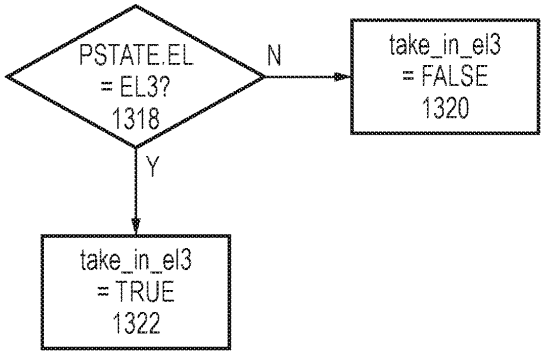
도면9d


route_to_EL2 = FALSE? 1292
N
route_soft_masked _to_el3 = FALSE 1294
Y
route_soft_masked_to_el2 = FALSE? 1296
N
Y
PSTATE.EL IN {EL2, EL1}? 1298
N
Y
PSTATE.A = 1? 1300
N
Y
SCR_EL3. TMEA = 1? 1302
N
Y
route_soft_masked _to_el3 = TRUE 1304

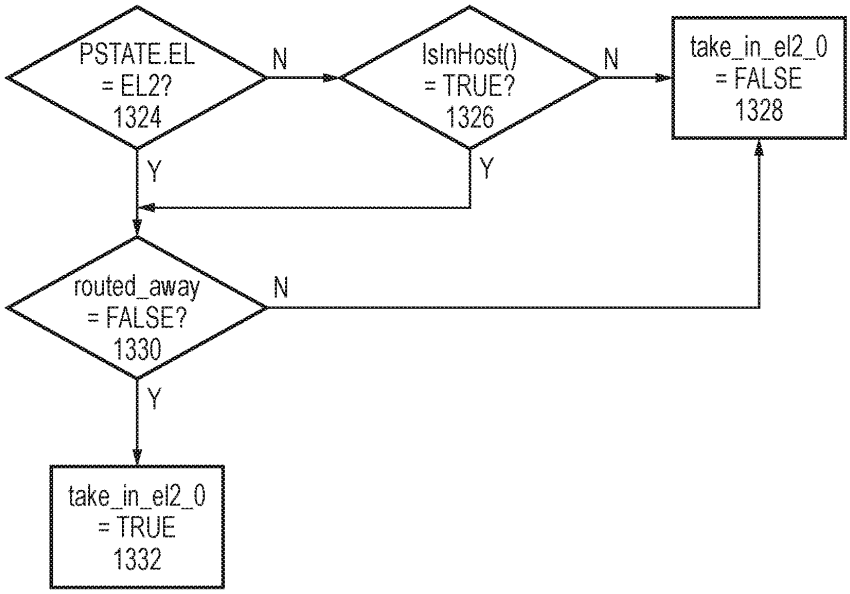
도면9e


route_to_el2 = TRUE? 1306
N
route_soft_masked _to_el2 = TRUE? 1310
N
route_to_el3 = TRUE? 1312
N
route_soft_masked _to_el3 = TRUE? 1314
N
routed_away = FALSE 1316
Y
Y
Y
Y
routed_away = TRUE 1308

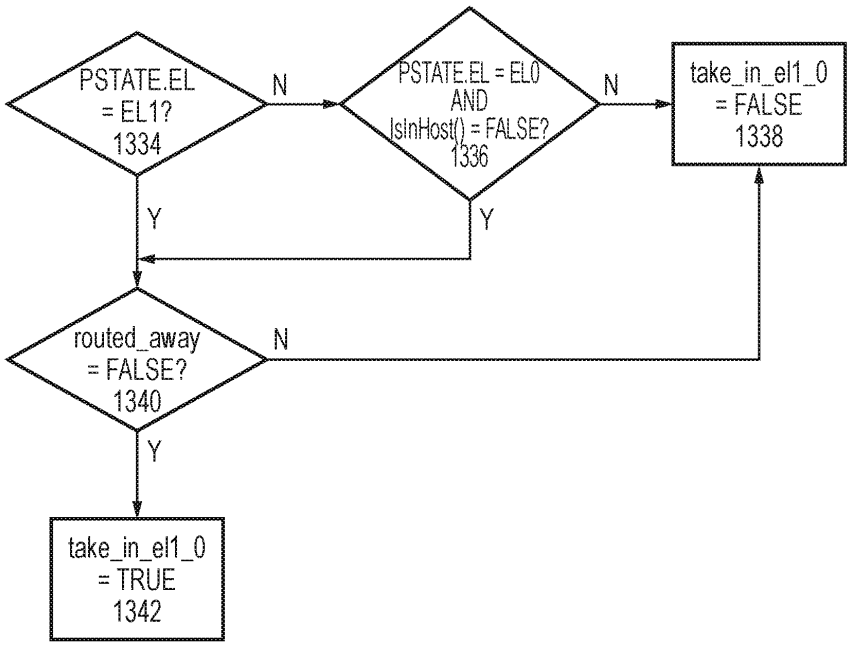
도면9f



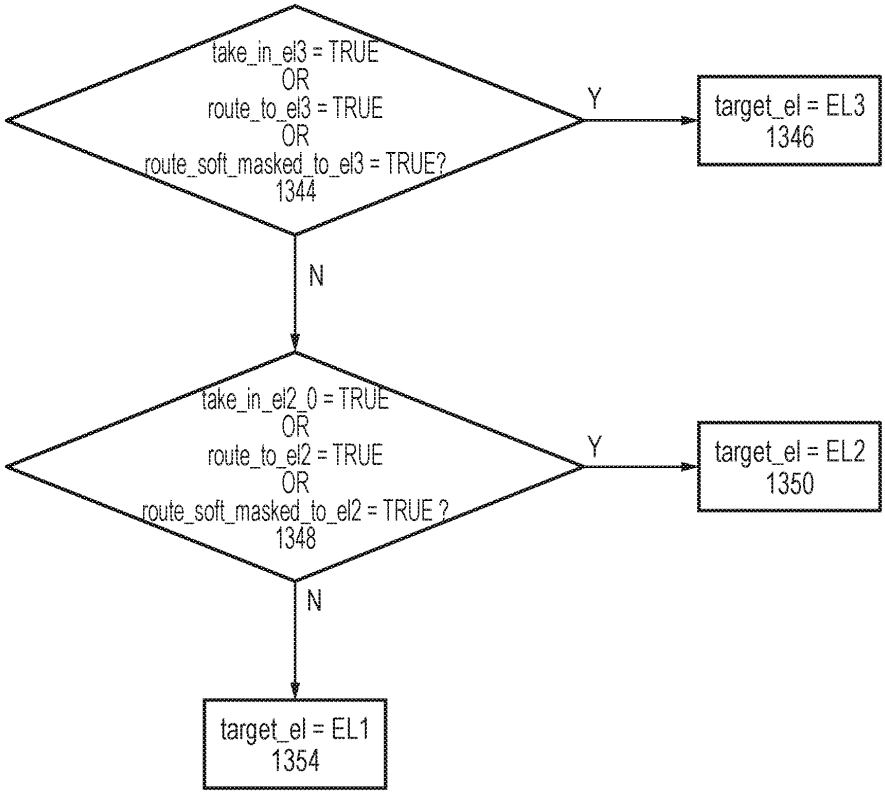
도면9g



도면9h



도면9i


take_in_el3 = TRUE
OR
route_to_el3 = TRUE
OR
route_soft_masked_to_el3 = TRUE?
1344
Y
target_el = EL3
1346
N
take_in_el2_0 = TRUE
OR
route_to_el2 = TRUE
OR
route_soft_masked_to_el2 = TRUE ?
1348
Y
target_el = EL2
1350
N
target_el = EL1
1354

도면10a

	EA EL3	TMEA EL3	NMEA EL3	TGE EL2	E2H EL2	AMO EL2	NMEA EL2	TMEA EL2	NMEA EL1	A PSTATE	EL3	EL2	EL1	EL0
1	0b0	0b0	0b0	0b0	X	0b0	0b0	0b0	0b0	0b0	C	C	EL1	EL1
2	0b0	0b0	0b0	0b0	X	0b0	0b0	0b0	0b0	0b1	C	C	B	B
3	0b0	0b0	0b0	0b0	X	0b0	0b0	0b0	0b1	X	C	C	EL1	EL1
4	0b0	0b0	0b0	0b0	X	0b0	0b0	0b1	X	0b0	C	C	EL1	EL1
5	0b0	0b0	0b0	0b0	X	0b0	0b0	0b1	0b0	0b1	C	C	EL2	EL2
6	0b0	0b0	0b0	0b0	X	0b0	0b0	0b1	0b1	0b1	C	C	EL2	EL1
7	0b0	0b0	0b0	0b0	X	0b0	0b1	0b0	0b0	0b0	C	EL2	EL1	EL1
8	0b0	0b0	0b0	0b0	X	0b0	0b1	0b0	0b0	0b1	C	EL2	B	B
9	0b0	0b0	0b0	0b0	X	0b0	0b1	0b0	0b1	X	C	EL2	EL1	EL1
10	0b0	0b0	0b0	0b0	X	0b0	0b1	0b1	X	0b0	C	EL2	EL1	EL1
11	0b0	0b0	0b0	0b0	X	0b0	0b1	0b1	0b0	0b1	C	EL2	EL2	EL2
12	0b0	0b0	0b0	0b0	X	0b0	0b1	0b1	0b1	0b1	C	EL2	EL2	EL1
13	0b0	0b0	0b0	0b0	X	0b1	0b0	X	X	0b0	C	EL2	EL2	EL2
14	0b0	0b0	0b0	0b0	X	0b1	0b0	X	X	0b1	C	B	EL2	EL2
15	0b0	0b0	0b0	0b0	X	0b1	0b1	X	X	X	C	EL2	EL2	EL2
16	0b0	0b0	0b0	0b1	X	X	0b0	X	X	0b0	C	EL2	n/a	EL2
17	0b0	0b0	0b0	0b1	X	X	0b1	X	X	X	C	EL2	n/a	EL2
18	0b0	0b0	0b0	0b1	0b0	X	0b0	X	X	0b1	C	B	n/a	EL2
19	0b0	0b0	0b0	0b1	0b1	X	0b0	X	X	0b1	C	B	n/a	B
20	0b0	0b0	0b1	0b0	X	0b0	0b0	0b0	0b0	0b0	EL3	C	EL1	EL1
21	0b0	0b0	0b1	0b0	X	0b0	0b0	0b0	0b0	0b1	EL3	C	B	B
22	0b0	0b0	0b1	0b0	X	0b0	0b0	0b0	0b1	X	EL3	C	EL1	EL1
23	0b0	0b0	0b1	0b0	X	0b0	0b0	0b1	X	0b0	EL3	C	EL1	EL1
24	0b0	0b0	0b1	0b0	X	0b0	0b0	0b1	0b0	0b1	EL3	C	EL2	EL2
25	0b0	0b0	0b1	0b0	X	0b0	0b0	0b1	0b1	0b1	EL3	C	EL2	EL1
26	0b0	0b0	0b1	0b0	X	0b0	0b1	0b0	0b0	0b0	EL3	EL2	EL1	EL1
27	0b0	0b0	0b1	0b0	X	0b0	0b1	0b0	0b0	0b1	EL3	EL2	B	B
28	0b0	0b0	0b1	0b0	X	0b0	0b1	0b0	0b1	X	EL3	EL2	EL1	EL1
29	0b0	0b0	0b1	0b0	X	0b0	0b1	0b1	X	0b0	EL3	EL2	EL1	EL1
30	0b0	0b0	0b1	0b0	X	0b0	0b1	0b1	0b0	0b1	EL3	EL2	EL2	EL2
31	0b0	0b0	0b1	0b0	X	0b0	0b1	0b1	0b1	0b1	EL3	EL2	EL2	EL1
32	0b0	0b0	0b1	0b0	X	0b1	0b0	X	X	0b0	EL3	EL2	EL2	EL2
33	0b0	0b0	0b1	0b0	X	0b1	0b0	X	X	0b1	EL3	B	EL2	EL2
34	0b0	0b0	0b1	0b0	X	0b1	0b1	X	X	X	EL3	EL2	EL2	EL2
35	0b0	0b0	0b1	0b1	X	X	0b0	X	X	0b0	EL3	EL2	n/a	EL2
36	0b0	0b0	0b1	0b1	X	X	0b1	X	X	X	EL3	EL2	n/a	EL2

도면10b

	EA EL3	TMEA EL3	NMEA EL3	TGE EL2	E2H EL2	AMO EL2	NMEA EL2	TMEA EL2	NMEA EL1	A PSTATE	EL3	EL2	EL1	EL0
37	0b0	0b0	0b1	0b1	0b0	X	0b0	X	X	0b1	EL3	B	n/a	EL2
38	0b0	0b0	0b1	0b1	0b1	X	0b0	X	X	0b1	EL3	B	n/a	B
39	0b0	0b1	0b0	0b0	X	0b0	X	0b0	0b0	0b1	C	EL3	EL3	EL3
40	0b0	0b1	0b0	0b0	X	0b0	X	0b0	0b1	0b1	C	EL3	EL3	EL1
41	0b0	0b1	0b0	0b0	X	0b0	X	0b1	0b0	0b1	C	EL3	EL2	EL2
42	0b0	0b1	0b0	0b0	X	0b0	X	0b1	0b1	0b1	C	EL3	EL2	EL1
43	0b0	0b1	0b0	0b0	X	0b0	0b0	X	X	0b0	C	EL3	EL1	EL1
44	0b0	0b1	0b0	0b0	X	0b0	0b1	X	X	0b0	C	EL2	EL1	EL1
45	0b0	0b1	0b0	0b0	X	0b1	X	X	X	0b0	C	EL2	EL2	EL2
46	0b0	0b1	0b0	0b0	X	0b1	X	X	X	0b1	C	EL3	EL2	EL2
47	0b0	0b1	0b0	0b1	X	X	X	X	X	0b0	C	EL2	n/a	EL2
48	0b0	0b1	0b0	0b1	0b0	X	X	X	X	0b1	C	EL3	n/a	EL2
49	0b0	0b1	0b0	0b1	0b1	X	0b0	X	X	0b1	C	EL3	n/a	EL3
50	0b0	0b1	0b0	0b1	0b1	X	0b1	X	X	0b1	C	EL3	n/a	EL2
51	0b0	0b1	0b1	0b0	X	0b0	X	0b0	0b0	0b1	EL3	EL3	EL3	EL3
52	0b0	0b1	0b1	0b0	X	0b0	X	0b0	0b1	0b1	EL3	EL3	EL3	EL1
53	0b0	0b1	0b1	0b0	X	0b0	X	0b1	0b0	0b1	EL3	EL3	EL2	EL2
54	0b0	0b1	0b1	0b0	X	0b0	X	0b1	0b1	0b1	EL3	EL3	EL2	EL1
55	0b0	0b1	0b1	0b0	X	0b0	0b0	X	X	0b0	EL3	EL3	EL1	EL1
56	0b0	0b1	0b1	0b0	X	0b0	0b1	X	X	0b0	EL3	EL2	EL1	EL1
57	0b0	0b1	0b1	0b0	X	0b1	X	X	X	0b0	EL3	EL2	EL2	EL2
58	0b0	0b1	0b1	0b0	X	0b1	X	X	X	0b1	EL3	EL3	EL2	EL2
59	0b0	0b1	0b1	0b1	X	X	X	X	X	0b0	EL3	EL2	n/a	EL2
60	0b0	0b1	0b1	0b1	0b0	X	X	X	X	0b1	EL3	EL3	n/a	EL2
61	0b0	0b1	0b1	0b1	0b1	X	0b0	X	X	0b1	EL3	EL3	n/a	EL3
62	0b0	0b1	0b1	0b1	0b1	X	0b1	X	X	0b1	EL3	EL3	n/a	EL2
63	0b1	X	0b0	0b0	X	X	X	X	X	0b0	EL3	EL3	EL3	EL3
64	0b1	X	0b0	0b0	X	X	X	X	X	0b1	B	EL3	EL3	EL3
65	0b1	X	0b0	0b1	X	X	X	X	X	0b0	EL3	EL3	n/a	EL3
66	0b1	X	0b0	0b1	X	X	X	X	X	0b1	B	EL3	n/a	EL3
67	0b1	X	0b1	0b0	X	X	X	X	X	X	EL3	EL3	EL3	EL3
68	0b1	X	0b1	0b1	X	X	X	X	X	X	EL3	EL3	n/a	EL3

도면11

	EA EL3	TMEA EL3	TGE EL2	TEA EL2	TMEA EL2	A PSTATE	EL3	EL2	EL1	EL0
1	0b0	0b0	0b0	0b0	0b0	X	EL3	EL2	EL1	EL1
2	0b0	0b0	0b0	0b0	0b1	0b0	EL3	EL2	EL1	EL1
3	0b0	0b0	0b0	0b0	0b1	0b1	EL3	EL2	EL2	EL1
4	0b0	0b0	0b0	0b1	X	X	EL3	EL2	EL2	EL2
5	0b0	0b0	0b1	X	X	X	EL3	EL2	n/a	EL2
6	0b0	0b1	0b0	0b0	X	0b0	EL3	EL2	EL1	EL1
7	0b0	0b1	0b0	0b0	0b0	0b1	EL3	EL3	EL3	EL1
8	0b0	0b1	0b0	0b0	0b1	0b1	EL3	EL3	EL2	EL1
9	0b0	0b1	0b0	0b1	X	0b0	EL3	EL2	EL2	EL2
10	0b0	0b1	0b0	0b1	X	0b1	EL3	EL3	EL2	EL2
11	0b0	0b1	0b1	X	X	0b0	EL3	EL2	n/a	EL2
12	0b0	0b1	0b1	X	X	0b1	EL3	EL3	n/a	EL2
13	0b1	X	0b0	X	X	X	EL3	EL3	EL3	EL3
14	0b1	X	0b1	X	X	X	EL3	EL3	n/a	EL3

도면12

