(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년04월08일
(11) 등록번호 10-1610838
(24) 등록일자 2016년04월04일

(51) 국제특허분류(Int. Cl.)
G06F 9/48 (2006.01) *G06F 13/24* (2006.01)
(21) 출원번호 10-2011-7019983
(22) 출원일자(국제) 2010년01월26일
심사청구일자 2014년10월17일
(85) 번역문제출일자 2011년08월26일
(65) 공개번호 10-2011-0113756
(43) 공개일자 2011년10월18일
(86) 국제출원번호 PCT/US2010/022111
(87) 국제공개번호 WO 2010/085804
국제공개일자 2010년07월29일
(30) 우선권주장
12/611,595 2009년11월03일 미국(US)
(뒷면에 계속)
(56) 선행기술조사문헌
US7209994 B1
US20040117532 A1
US20080162762 A1
JP2007183951 A

(73) 특허권자
어드밴스드 마이크로 디바이시즈, 인코포레이티드
미국 캘리포니아 94088-3453 서니베일 피.오.박스 3453 원 에이엠디 플레이스
(72) 발명자
세레브린 벤자민 씨.
미국 캘리포니아 94086 서니베일 마호가니 레인 771
맥컬리 도날드 더블유.
미국 텍사스 78734 레이크웨이 에지워터 코브 102
(뒷면에 계속)
(74) 대리인
박장원

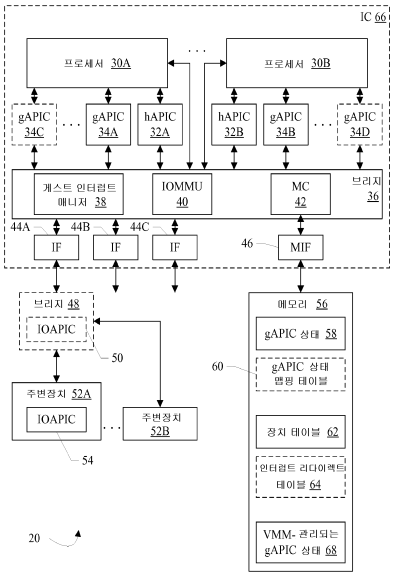
전체 청구항 수 : 총 28 항 심사관 : 유진태

(54) 발명의 명칭 **인터럽트 가상화를 돕기 위한 각 프로세서에 대한 게스트 인터럽트 제어기들**

(57) 요 약

일 실시예에서, 시스템은 프로세서와, 상기 프로세서에 결합된 제 1 인터럽트 제어기와, 그리고 상기 프로세서에 결합된 제 2 인터럽트 제어기를 포함한다. 상기 제 1 인터럽트 제어기는, 상기 시스템 내의 호스트를 목표로 하고 있는 제 1 인터럽트를 전달하는 제 1 인터럽트 메시지를 수신하는 것에 응답하여, 상기 프로세서에 상기 인터럽트에 대해 신호하도록 구성된다. 상기 제 2 인터럽트 제어기는, 상기 프로세서 상에서 실행가능하고 상기 호스트에 의해 제어되는 게스트를 목표로 하는 제 2 인터럽트를 전달하는 제 2 인터럽트 메시지를 수신하는 것에 응답하여, 상기 프로세서에 인터럽트에 대해 신호하도록 구성된다.

대 표 도 - 도2



(72) 발명자

위더하이른 존 에프.

미국 캘리포니아 95125 산호세 루비노 드라이브 #115 3110

쿠퍼 엘리자베스 엠.

미국 캘리포니아 95033 로스가토스 파브레 리쥐 로드 18764

험멜 마크 디.

미국 매사추세츠 02038 프랭클린 스튜어트 스트리트 68

(30) 우선권주장

12/611,607 2009년11월03일 미국(US)

12/611,622 2009년11월03일 미국(US)

61/147,269 2009년01월26일 미국(US)

명 세 서

청구범위

청구항 1

인터럽트들을 제어하는 방법으로서,

프로세서에 전용되는 제 1 인터럽트 제어기에서 제 1 인터럽트 메시지를 수신하는 단계 -상기 제 1 인터럽트 메시지는 가상 머신 매니저를 목표로 하는 제 1 인터럽트를 통신(communicate)하고, 상기 가상 머신 매니저는 상기 프로세서 상에서 실행가능하며- ;

상기 프로세서에 전용되는 제 2 인터럽트 제어기에서 제 2 인터럽트 메시지를 수신하는 단계 -상기 제 2 인터럽트 메시지는 게스트를 목표로 하는 제 2 인터럽트를 통신하고, 상기 게스트는 상기 가상 머신 매니저에 의해서 제어되고 상기 프로세서 상에서 실행가능하며- ;

상기 제 1 인터럽트 메시지를 수신하는 것에 응답하여, 상기 제 1 인터럽트 제어기가 상기 제 1 인터럽트를 상기 프로세서로 전달(deliver)하는 단계; 그리고

상기 제 2 인터럽트 메시지를 수신하는 것에 응답하여, 상기 제 2 인터럽트 제어기가 상기 제 2 인터럽트를 상기 프로세서로 전달하는 단계

를 포함하는 것을 특징으로 하는 인터럽트들을 제어하는 방법.

청구항 2

청구항 2은(는) 설정등록료 납부시 포기되었습니다.

제1항에 있어서,

상기 제 1 인터럽트를 상기 프로세서로 전달하는 단계 이전에, 상기 제 1 인터럽트 제어기 내의 다른 인터럽트들에 비하여 상기 제 1 인터럽트가 더 높은 우선순위를 갖는다라고 상기 제 1 인터럽트 제어기가 결정하는 단계 -상기 제 1 인터럽트를 상기 프로세서로 전달하는 상기 제 1 인터럽트 제어기는 상기 제 1 인터럽트 메시지를 수신하는 것에 응답할 뿐만 아니라 상기 결정에 응답하며- ; 그리고

상기 제 2 인터럽트를 상기 프로세서로 전달하는 단계 이전에, 상기 제 2 인터럽트 제어기 내의 다른 인터럽트들에 비하여 상기 제 2 인터럽트가 더 높은 우선순위를 갖는다라고 상기 제 2 인터럽트 제어기가 결정하는 단계

를 더 포함하며,

상기 제 2 인터럽트를 상기 프로세서로 전달하는 상기 제 2 인터럽트 제어기는 상기 제 2 인터럽트 메시지를 수신하는 것에 응답할 뿐만 아니라 상기 결정에 응답하는 것을 특징으로 하는 인터럽트들을 제어하는 방법.

청구항 3

청구항 3은(는) 설정등록료 납부시 포기되었습니다.

제1항에 있어서,

상기 제 1 인터럽트를 상기 프로세서로 전달하는 단계 이후에, 상기 제 1 인터럽트에 응답하여 가상 머신 모니터를 인터럽트하는 것을 포함하여, 상기 프로세서가 상기 가상 머신 모니터를 실행하는 단계를 더 포함하는 것을 특징으로 하는 인터럽트들을 제어하는 방법.

청구항 4

청구항 4은(는) 설정등록료 납부시 포기되었습니다.

제3항에 있어서,

상기 프로세서가 상기 가상 머신 모니터를 실행하는 단계는,

상기 제 2 인터럽트를 차단(masking)하는 단계를 더 포함하는 것을 특징으로 하는 인터럽트들을 제어하는 방법.

청구항 5

청구항 5은(는) 설정등록료 납부시 포기되었습니다.

제1항에 있어서,

상기 제 1 인터럽트를 상기 프로세서로 전달하는 단계 이후에, 상기 제 1 인터럽트에 응답하여 상기 게스트를 퇴장(exiting)시키는 것을 포함하여, 상기 프로세서가 상기 게스트를 실행하는 단계를 더 포함하는 것을 특징으로 하는 인터럽트들을 제어하는 방법.

청구항 6

청구항 6은(는) 설정등록료 납부시 포기되었습니다.

제5항에 있어서,

상기 프로세서가 상기 게스트를 실행하는 단계는,

상기 제 2 인터럽트에 응답하여 상기 게스트를 인터럽트하고 그리고 상기 게스트와 관련된 가상 머신 내에 상기 제 2 인터럽트를 서비스하는 단계

를 더 포함하는 것을 특징으로 하는 인터럽트들을 제어하는 방법.

청구항 7

제1항에 있어서,

상기 제 2 인터럽트 메시지를 수신하는 단계는,

상기 제 2 인터럽트 메시지 내에 포함된 게스트 식별자와 상기 제 2 인터럽트 제어기에 저장된 제 2 게스트 식별자가 매칭하는지를 검출하는 단계 -상기 게스트 식별자는 상기 프로세서를 포함하고 있는 시스템에서 실행가능한 복수의 게스트들 중에서 상기 게스트를 식별하며- ;

상기 제 2 인터럽트 메시지 내에 포함된 목적지 식별자와 상기 제 2 인터럽트 제어기에 저장된 제 2 목적지 식별자가 매칭하는지를 검출하는 단계; 및

상기 게스트 식별자가 제 2 게스트 식별자와 매칭되고 그리고 상기 목적지 식별자가 상기 제 2 목적지 식별자와 매칭됨을 검출하는 것에 응답하여, 상기 제 2 인터럽트 제어기에서 상기 제 2 인터럽트를 수락하는 단계

를 포함하는 것을 특징으로 하는 인터럽트들을 제어하는 방법.

청구항 8

제7항에 있어서,

상기 제 1 인터럽트 메시지를 수신하는 단계는,

상기 제 1 인터럽트 메시지 내에 포함된 제 3 목적지 식별자와 상기 제 1 인터럽트 제어기에 저장된 제 4 목적지 식별자가 매칭하는지를 검출하는 단계; 및

게스트 식별자 없이 상기 매칭을 검출하는 것에 응답하여, 상기 제 1 인터럽트 제어기에서 상기 제 2 인터럽트를 수락하는 단계

를 포함하는 것을 특징으로 하는 인터럽트들을 제어하는 방법.

청구항 9

컴퓨터 시스템으로서,

프로세서;

상기 프로세서에 접속되며, 그리고 상기 시스템 내의 호스트를 목표로 하는 제 1 인터럽트를 통신하는 제 1 인터럽트 메시지에 응답하여 인터럽트를 위해 상기 프로세서에게 시그널링하도록 된 제 1 인터럽트 제어기; 및

상기 프로세서에 접속되며, 그리고 상기 호스트에 의해서 제어되며 상기 프로세서 상에서 실행가능한 게스트를 목표로 하는 제 2 인터럽트를 통신하는 제 2 인터럽트 메시지에 응답하여 인터럽트를 위해 상기 프로세서에게 시그널링하도록 된 제 2 인터럽트 제어기

를 포함하며,

상기 제 1 인터럽트 제어기는 상기 프로세서에 전용이며 상기 제 2 인터럽트 제어기는 상기 프로세서에 전용인 것을 특징으로 하는 컴퓨터 시스템.

청구항 10

제9항에 있어서,

상기 프로세서는, 사용중에 상기 호스트가 상기 프로세서 상에서 실행되는 시간 동안 상기 제 1 인터럽트 제어기로부터의 인터럽트를 위한 상기 시그널링에 응답하여 상기 제 1 인터럽트를 서비스하기 위해 상기 호스트를 인터럽트하도록 된 것을 특징으로 하는 컴퓨터 시스템.

청구항 11

제10항에 있어서,

상기 호스트는 상기 게스트에 관련된 가상 머신을 제어하는 가상 머신 매니저인 것을 특징으로 하는 컴퓨터 시스템.

청구항 12

제10항에 있어서,

상기 프로세서는, 사용중에 상기 게스트가 상기 프로세서 상에서 실행되는 시간 동안 상기 제 1 인터럽트 제어기로부터의 인터럽트를 위한 상기 시그널링에 응답하여 상기 호스트에서 상기 제 1 인터럽트를 서비스하도록 상기 게스트를 퇴장시키도록 된 것을 특징으로 하는 컴퓨터 시스템.

청구항 13

제9항에 있어서,

상기 프로세서는, 상기 게스트를 인터럽트하고 그리고 사용중에 상기 게스트가 실행되는 시간 동안 상기 제 2 인터럽트 제어기로부터의 인터럽트를 위한 상기 시그널링에 응답하여 상기 게스트에 관련된 가상 머신 내의 상기 제 2 인터럽트를 서비스하도록 된 것을 특징으로 하는 컴퓨터 시스템.

청구항 14

청구항 14은(는) 설정등록료 납부시 포기되었습니다.

제13항에 있어서,

상기 프로세서는,

사용중에 상기 호스트가 실행되는 시간 동안 상기 제 2 인터럽트 제어기로부터의 인터럽트를 위한 상기 시그널링에 응답하여 상기 게스트가 실행될 때까지 상기 제 2 인터럽트 제어기에 의해서 요청되는 인터럽트를 차단하도록 된 것을 특징으로 하는 컴퓨터 시스템.

청구항 15

청구항 15은(는) 설정등록료 납부시 포기되었습니다.

제10항에 있어서,

제 2 프로세서;

상기 제 2 프로세서에 접속되며, 그리고 상기 호스트를 목표로 하는 제 3 인터럽트를 통신하는 제 3 인터럽트 메시지에 응답하여 인터럽트를 위해 상기 제 2 프로세서에게 시그널링하도록 된 제 3 인터럽트 제어기; 및

상기 프로세서에 접속되며, 그리고 상기 호스트에 의해서 제어되며 상기 제 2 프로세서 상에서 실행가능한 제 2 게스트를 목표로 하는 제 4 인터럽트를 통신하는 제 4 인터럽트 메시지에 응답하여 인터럽트를 위해 상기 제 2 프로세서에게 시그널링하도록 된 제 4 인터럽트 제어기

를 더 포함하는 것을 특징으로 하는 컴퓨터 시스템.

청구항 16

인터럽트 제어기로서,

상기 인터럽트 제어기가 전용인 프로세서 상에서 실행가능한 게스트를 식별하는 게스트 식별자를 저장하도록 된 게스트 식별자 레지스터;

목적지 식별자를 저장하도록 된 적어도 하나의 추가 레지스터;

상기 게스트 식별자 레지스터 및 상기 적어도 하나의 추가 레지스터에 접속되는 제어 회로

를 포함하며,

상기 제어 회로는 상기 프로세서를 위한 인터럽트의 표시(indication)를 수신하며, 상기 표시는 제 1 게스트 식별자와 제 1 목적지 식별자를 포함하며, 그리고 상기 게스트 식별자가 상기 제 1 게스트 식별자와 매칭되는 것 그리고 상기 목적지 식별자가 상기 제 1 목적지 식별자와 매칭되는 것에 응답하여, 상기 제어 회로는 상기 프로세서를 대신하여 인터럽트를 수락하도록 된 것을 특징으로 하는 인터럽트 제어기.

청구항 17

제16항에 있어서,

상기 적어도 하나의 추가 레지스터는,

논리적(logical) 목적지 식별자를 저장하도록 된 제 1 레지스터; 및

물리적(physical) 목적지 식별자를 저장하도록 된 제 2 레지스터

를 포함하는 것을 특징으로 하는 인터럽트 제어기.

청구항 18

제17항에 있어서,

상기 제어 회로는 상기 제 1 목적지 식별자와 상기 물리적 목적지 식별자의 비교에 응답하여 상기 제 1 목적지 식별자와 상기 물리적 목적지 식별자 사이의 매칭을 검출하며, 그리고 상기 물리적 목적지 식별자가 상기 제 1 목적지 식별자와 매칭되지 않는 경우라 하여도, 상기 제어 회로는 또한 상기 인터럽트가 브로드캐스트로서 표시됨을 검출하며 그리고 상기 인터럽트가 브로드캐스트로서 표시됨을 검출하는 것에 응답하여 상기 프로세서를 대신하여 상기 인터럽트를 수락하는 것을 특징으로 하는 인터럽트 제어기.

청구항 19

제17항에 있어서,

상기 논리적 목적지 식별자는 클러스터 식별자와 클러스터의 멤버들을 식별하는 비트 벡터를 포함하며, 그리고

상기 제어 회로는, 상기 제 1 목적지 식별자의 대응 부분과 상기 클러스터 식별자와의 비교에 응답하여 그리고 상기 논리적 목적지 식별자의 상기 비트 벡터 내의 세트 비트가 상기 제 1 목적지 식별자의 대응 비트 벡터 부분에서도 세트되는 것에 또한 응답하여, 상기 제 1 목적지 식별자와 상기 논리적 목적지 식별자 사이의 매칭을 검출하도록 된 것을 특징으로 하는 인터럽트 제어기.

청구항 20

인터럽트 제어 방법으로서,

입력/출력 메모리 관리 유닛(IOMMU)에서 인터럽트 메시지를 수신하는 단계 -상기 인터럽트 메시지는 인터럽트를 통신하고, 상기 인터럽트는 시스템 내의 복수의 게스트들 중에서 제 1 게스트에 할당되는 주변 장치에 의해서

비롯되며- ;

상기 인터럽트 메시지를 상기 제 1 게스트에 맵핑시키기 위하여, 메모리 내의 하나 이상의 제 1 데이터 구조들을 상기 IOMMU에 의해서 액세스하는 단계 -상기 하나 이상의 제 1 데이터 구조들은 다른 메모리 위치에 대한 포인터를 또한 포함하고- ;

상기 포인터에 응답하여 메모리 내의 하나 이상의 제 2 데이터 구조들을 게스트 인터럽트 매니저에 의해서 로케이팅(locating)하는 단계 -상기 하나 이상의 제 2 데이터 구조들은 상기 제 1 게스트에 대한 인터럽트 제어기 상태를 포함하며- ;

상기 인터럽트를 상기 하나 이상의 제 2 데이터 구조들에 기록하고, 상기 제 1 게스트가 실행 중인 시간에 상기 인터럽트가 상기 제 1 게스트로 전달되는 것을 허용하는 단계; 그리고

상기 인터럽트를 기록하는 것에 부가하여, 상기 인터럽트 메시지가 수신되는 시간에 상기 제 1 게스트가 실행중이라고 결정하는 것에 응답하여 상기 인터럽트를 제 1 인터럽트 제어기로 통신하는 단계

를 포함하며,

상기 제 1 인터럽트 제어기는 상기 제 1 게스트에 할당되며 그리고 상기 제 1 게스트 내의 인터럽트에 대한 목적지인 것을 특징으로 하는 인터럽트 제어 방법.

청구항 21

청구항 21은(는) 설정등록료 납부시 포기되었습니다.

제20항에 있어서,

상기 시스템은 게스트들에 할당가능한 복수의 인터럽트 제어기들을 포함하고, 상기 제 1 인터럽트 제어기는 상기 복수의 인터럽트 제어기들 중 하나이며, 그리고 상기 통신하는 단계는 상기 인터럽트 메시지를 상기 복수의 인터럽트 제어기들 각각에 전송하는 단계를 포함하는 것을 특징으로 하는 인터럽트 제어 방법.

청구항 22

청구항 22은(는) 설정등록료 납부시 포기되었습니다.

제21항에 있어서,

상기 복수의 인터럽트 제어기들 각각은 상기 인터럽트 제어기가 할당되는 게스트를 식별하는 게스트 식별자를 포함하며, 그리고 상기 인터럽트를 통신하는 단계는 상기 인터럽트와 함께 상기 제 1 게스트를 식별하는 제 1 게스트 식별자를 통신하는 것을 포함하며, 상기 인터럽트 제어 방법은,

상기 인터럽트와 함께 상기 제 1 게스트를 식별하는 상기 제 1 게스트 식별자를 통신한 후에, 각각의 인터럽트 제어기가 상기 게스트 식별자와 상기 제 1 게스트 식별자를 비교하는 단계; 및

상기 제 1 인터럽트 제어기 내의 상기 게스트 식별자가 상기 제 1 게스트 식별자와 매칭되는 것에 응답하여 상기 제 1 인터럽트 제어기가 상기 인터럽트를 기록하는 단계

를 더 포함하는 것을 특징으로 하는 인터럽트 제어 방법.

청구항 23

청구항 23은(는) 설정등록료 납부시 포기되었습니다.

제22항에 있어서,

상기 복수의 인터럽트 제어기들 각각은 그 인터럽트 제어기가 할당되는 게스트 내의 적어도 하나의 목적지 식별자를 포함하고, 그리고 상기 인터럽트를 통신하는 단계는 상기 인터럽트와 관련된 제 1 목적지 식별자를 통신하는 것을 포함하며, 상기 인터럽트 제어 방법은,

상기 인터럽트와 관련된 상기 제 1 목적지 식별자를 통신한 후에, 각각의 인터럽트 제어기가 상기 목적지 식별자와 상기 제 1 목적지 식별자를 비교하는 단계; 및

상기 제 1 인터럽트 제어기 내의 상기 목적지 식별자가 상기 제 1 목적지 식별자와 매칭되는 것에 응답하여 상

기 제 1 인터럽트 제어기가 상기 인터럽트를 기록하는 단계

를 더 포함하는 것을 특징으로 하는 인터럽트 제어 방법.

청구항 24

제20항에 있어서,

상기 인터럽트 제어기 상태는 인터럽트 제어기 내의 인터럽트 요청 레지스터의 상태를 포함하고, 상기 인터럽트는 인터럽트 벡터를 포함하고, 상기 인터럽트 요청 레지스터는 상기 벡터에 관련된 비트 위치를 포함하며,

상기 인터럽트를 기록하는 것은 상기 하나 이상의 제 2 데이터 구조들 내의 상기 비트 위치에 상기 비트를 세트시키는 것을 포함하는 것을 특징으로 하는 인터럽트 제어 방법.

청구항 25

청구항 25은(는) 설정등록료 납부시 포기되었습니다.

제24항에 있어서,

상기 비트를 세트시키는 것은 원자적(atomic)인 것을 특징으로 하는 인터럽트 제어 방법.

청구항 26

청구항 26은(는) 설정등록료 납부시 포기되었습니다.

제24항에 있어서,

상기 비트를 세트시키는 것은 상기 세트된 비트를 상기 비트 위치에 원자적으로 OR시키는 것을 포함하는 것을 특징으로 하는 인터럽트 제어 방법.

청구항 27

청구항 27은(는) 설정등록료 납부시 포기되었습니다.

제24항에 있어서,

상기 비트를 세트시키는 것은, 비교(compare) 및 스와프(swap) 동작을 이용하여 수행되는 것을 특징으로 하는 인터럽트 제어 방법.

청구항 28

컴퓨터 시스템으로서,

메모리 시스템;

상기 컴퓨터 시스템 상에서 실행가능한 게스트를 목표로 하는 인터럽트에 대응하는 인터럽트 메시지를 수신하도록 된 게스트 인터럽트 매니저 -상기 게스트 인터럽트 매니저는, 상기 인터럽트 메시지가 수신되는 때에 상기 게스트가 상기 컴퓨터 시스템에서 활성화되지 않는 경우에도 상기 인터럽트가 상기 게스트로 전달되는 것을 보장하기 위하여, 상기 메모리 시스템의 하나 이상의 제 1 데이터 구조들 내에 상기 인터럽트를 기록하며- ;

상기 인터럽트를 개시하도록 된 주변 장치;

상기 주변 장치로부터 상기 인터럽트를 수신하는 입력/출력 메모리 관리 유닛(IOMMU) -상기 IOMMU는 상기 메모리 시스템 내의 하나 이상의 제 2 데이터 구조들의 데이터에 응답하여 상기 인터럽트를 상기 게스트에 관련시키고, 상기 IOMMU는 상기 인터럽트가 상기 게스트에 관련되는 것에 응답하여 상기 게스트를 식별하는 게스트 식별자를 포함하고 있는 상기 인터럽트 메시지를 전송하도록 되며, 상기 하나 이상의 제 2 데이터 구조는 또한 다른 메모리 위치에 대한 포인터를 포함하고, 상기 게스트 인터럽트 매니저는 상기 포인터에 응답하여 상기 메모리의 하나 이상의 제 1 데이터 구조들을 로케이팅(locate)하며- ; 및

상기 게스트에 할당가능한 인터럽트 제어기

를 포함하며,

상기 인터럽트 메시지가 수신되는 때에 상기 게스트가 실행중이라는 것에 응답하여 상기 인터럽트를 상기 인터럽트 제어기 내에 캡춰하도록, 상기 게스트 인터럽트 매니저는 상기 인터럽트 메시지를 상기 인터럽트 제어기로 포워딩하는 것을 특징으로 하는 컴퓨터 시스템.

청구항 29

제28항에 있어서,

상기 인터럽트 제어기는 상기 게스트의 게스트 식별자를 저장하며, 상기 인터럽트 제어기는 수신된 인터럽트 메시지로부터의 게스트 식별자와 상기 인터럽트 제어기에 저장된 게스트 식별자를 비교하며, 그리고 상기 인터럽트 제어기는 상기 게스트 식별자들의 비교에서 매칭이 발생하는 것에 응답하여 상기 인터럽트를 수락하도록 된 것을 특징으로 하는 컴퓨터 시스템.

청구항 30

제28항에 있어서,

상기 하나 이상의 제 1 데이터 구조들은 복수의 엔트리들을 포함하고, 각각의 엔트리는 주어진 게스트에 대한 상기 인터럽트 제어기의 상태의 적어도 일부를 저장하며, 상기 각각의 엔트리의 상기 상태는 인터럽트 요청 레지스터의 상태를 포함하고, 그리고 상기 게스트 인터럽트 매니저는 상기 인터럽트를 기록하도록 상기 인터럽트 요청 레지스터의 상태를 업데이트하도록 된 것을 특징으로 하는 컴퓨터 시스템.

청구항 31

청구항 31은(는) 설정등록료 납부시 포기되었습니다.

제30항에 있어서,

상기 인터럽트 요청 레지스터는 상기 인터럽트 제어기에 의해서 지원되는 각각의 인터럽트 벡터에 대응하는 비트를 포함하고, 상기 인터럽트 메시지는 상기 인터럽트의 상기 인터럽트 벡터를 포함하며, 그리고 상기 게스트 인터럽트 매니저는 상기 인터럽트 메시지로부터의 인터럽트 벡터에 대응하는 비트를 업데이트하도록 된 것을 특징으로 하는 컴퓨터 시스템.

청구항 32

청구항 32은(는) 설정등록료 납부시 포기되었습니다.

제31항에 있어서,

상기 업데이트는 원자적(atomic)인 것을 특징으로 하는 컴퓨터 시스템.

청구항 33

청구항 33은(는) 설정등록료 납부시 포기되었습니다.

제32항에 있어서,

상기 업데이트는 원자적 OR(atomic OR)인 것을 특징으로 하는 컴퓨터 시스템.

청구항 34

복수의 명령어들을 저장하는 컴퓨터 액세스가능한 저장 매체로서, 상기 명령어들은 실행되는 때에,

제 1 게스트를 위한 인터럽트를 수락하도록 인터럽트 제어기를 프로그래밍하고;

상기 인터럽트 제어기를 프로그래밍한 이후에, 상기 제 1 게스트에 대응하는 인터럽트 상태를 캡춰하도록 게스트 인터럽트 매니저에 의해서 유지되는 데이터 구조를 판독하고 -상기 데이터 구조는 복수의 엔트리들을 포함하고, 각각의 엔트리는 주어진 게스트에 대한 인터럽트 제어기의 상태의 적어도 일부를 저장하며, 상기 상태는 인터럽트 요청 레지스터의 상태를 포함하고, 상기 복수의 엔트리들 중 제 1 엔트리는 상기 제 1 게스트에 대응하며- ; 그리고

상기 데이터 구조로부터 판독된 상기 인터럽트 상태로 상기 인터럽트 제어기의 상태를 업데이트하며,

상기 인터럽트 제어기는 상기 인터럽트 요청 레지스터를 포함하고 그리고 상기 인터럽트 요청 레지스터는 상기 데이터 구조 내의 상기 제 1 엔트리로부터의 상기 상태로 업데이트되는 것을 특징으로 하는 컴퓨터 액세스가능한 저장 매체.

청구항 35

청구항 35은(는) 설정등록료 납부시 포기되었습니다.

제34항에 있어서,

실행되는 때에 상기 인터럽트 제어기의 상태를 업데이트하는 상기 복수의 명령어들은,

실행되는 때에, 상기 데이터 구조로부터 판독된 인터럽트 요청 비트들의 세트를 상기 인터럽트 제어기의 인터럽트 요청 레지스터 내에 OR 시키는 명령어들을 포함하는 것을 특징으로 하는 컴퓨터 액세스가능한 저장 매체.

청구항 36

제34항에 있어서,

상기 복수의 명령어들은, 실행되는 때에,

상기 제 1 게스트를 위한 인터럽트들을 수락하는 것을 중단하도록 상기 인터럽트 제어기를 프로그래밍하며;

상기 인터럽트 제어기로부터 인터럽트 제어기 상태를 판독하며; 그리고

상기 인터럽트 제어기 상태를 상기 데이터 구조에 기입하는 것

을 특징으로 하는 컴퓨터 액세스가능한 저장 매체.

청구항 37

인터럽트 제어 방법으로서,

복수의 게스트들이 실행가능한 시스템의 메모리에 데이터 구조를 저장하는 단계 -상기 데이터 구조는 상기 복수의 게스트들 각각의 각 목적지에 대한 인터럽트 요청 상태를 적어도 저장하며, 그리고 상기 인터럽트 요청 상태는 상기 복수의 게스트들 중 해당 게스트의 해당 인터럽트 제어기에서 어떤 인터럽트들이 요청되었는지를 식별하며- ;

상기 복수의 게스트들 중 제 1 게스트의 제 1 목적지를 목표로 하는 인터럽트 메시지를 수신하는 단계; 그리고

상기 데이터 구조 내에서 상기 인터럽트 메시지에서 식별된 인터럽트를 상기 제 1 목적지 및 제 1 게스트에 대한 인터럽트 요청 상태에 기록하도록 상기 제 1 목적지 및 제 1 게스트에 대응하는 상기 인터럽트 요청 상태를 업데이트하는 단계

를 포함하며,

각각의 인터럽트는 관련된 전달 모드를 가지며, 상기 데이터 구조는 제 1 전달 모드에 대응하는 제 1 섹션과 제 2 전달 모드에 대응하는 제 2 섹션을 포함하며, 각각의 목적지 및 게스트에 대한 상기 인터럽트 요청 상태는 상기 제 1 섹션 및 제 2 섹션 각각에 저장된 데이터를 포함하며, 상기 인터럽트 요청 상태를 업데이트하는 단계는, 상기 관련된 전달 모드에 응답하여 상기 인터럽트에 대한 표시를 상기 제 1 섹션 및 제 2 섹션 중 선택된 섹션에 기입하는 단계를 포함하는 것을 특징으로 하는 인터럽트 제어 방법.

청구항 38

청구항 38은(는) 설정등록료 납부시 포기되었습니다.

제37항에 있어서,

각각의 인터럽트는 관련된 인터럽트 벡터를 가지며, 상기 인터럽트 요청 상태는 비트 벡터를 포함하고 상기 비트 벡터는 각각의 인터럽트 벡터에 대한 비트를 가지며, 상기 제 1 섹션에 저장된 데이터는 각각의 인터럽트 벡터에 대해 상기 비트를 갖는 제 1 비트 벡터를 포함하고, 상기 제 2 섹션에 저장된 데이터는 각각의 인터럽트 벡터에 대해 상기 비트를 갖는 제 2 비트 벡터를 포함하고, 그리고 상기 제 1 목적지 및 상기 제 1 게스트에 대응하는 인터럽트 요청 상태는 상기 제 1 비트 벡터와 상기 제 2 비트 벡터의 논리 OR(logical OR)를 포함하는

것을 특징으로 하는 인터럽트 제어 방법.

청구항 39

청구항 39은(는) 설정등록료 납부시 포기되었습니다.

제38항에 있어서,

상기 인터럽트에 대한 표시를 기입하는 단계는,

상기 인터럽트 메시지 내의 인터럽트 벡터에 대응하는 상기 비트를 상기 관련된 전달 모드에 응답하여 상기 제 1 비트 벡터 혹은 상기 제 2 비트 벡터 중 하나에 세트하는 단계를 포함하는 것을 특징으로 하는 인터럽트 제어 방법.

청구항 40

청구항 40은(는) 설정등록료 납부시 포기되었습니다.

제38항에 있어서,

상기 제 1 전달 모드는 물리적(physical)이며, 상기 제 1 섹션은 로우들(rows)을 갖는 제 1 테이블을 포함하고, 상기 로우들은 물리적 전달 모드 인터럽트들에 대한 목적지들을 식별하는 물리적 식별자들에 대응하며, 그리고 상기 제 1 비트 벡터는 상기 제 1 테이블의 하나의 로우에 저장되는 것을 특징으로 하는 인터럽트 제어 방법.

청구항 41

청구항 41은(는) 설정등록료 납부시 포기되었습니다.

제40항에 있어서,

상기 제 2 전달 모드는 논리적(logical)이며, 논리적 전달 모드 인터럽트에 대한 하나 이상의 목적지들을 식별하는 논리적 식별자는 클러스터 식별자와 클러스터의 멤버들을 나타내는 식별 비트 벡터를 포함하고, 상기 제 2 섹션은 각각의 클러스터에 관련된 서브섹션들을 갖는 제 2 테이블을 포함하고, 상기 서브섹션의 로우들은 상기 클러스터의 멤버들에 대응하며, 그리고 상기 제 2 비트 벡터는 상기 제 2 테이블의 하나의 로우에 저장되는 것을 특징으로 하는 인터럽트 제어 방법.

청구항 42

청구항 42은(는) 설정등록료 납부시 포기되었습니다.

제38항에 있어서,

상기 제 1 전달 모드는 물리적(physical)이며, 상기 제 1 섹션은 인터럽트 벡터들에 대응하는 로우들을 갖는 제 1 테이블을 포함하고, 각각의 로우의 각각의 비트는 물리적 전달 모드 인터럽트들에 대한 목적지를 식별하는 다른 물리적 식별자에 대응하며, 그리고 상기 제 1 비트 벡터는 상기 제 1 테이블의 하나의 컬럼(column)에 저장되는 것을 특징으로 하는 인터럽트 제어 방법.

청구항 43

청구항 43은(는) 설정등록료 납부시 포기되었습니다.

제42항에 있어서,

상기 제 2 전달 모드는 논리적(logical)이며, 논리적 전달 모드 인터럽트에 대한 하나 이상의 목적지들을 식별하는 논리적 식별자는 클러스터 식별자와 클러스터의 멤버들을 나타내는 식별 비트 벡터를 포함하고, 상기 제 2 섹션은 각각의 클러스터에 관련된 서브섹션들을 갖는 제 2 테이블을 포함하고, 각각의 로우의 각각의 비트는 상기 클러스터의 다른 멤버에 대응하며, 그리고 상기 제 2 비트 벡터는 상기 제 2 테이블의 하나의 서브섹션의 하나의 컬럼에 저장되는 것을 특징으로 하는 인터럽트 제어 방법.

청구항 44

청구항 44은(는) 설정등록료 납부시 포기되었습니다.

제43항에 있어서,

상기 제 1 섹션의 각각의 로우 및 상기 제 2 섹션의 각 서브섹션의 각각의 로우는 서로 다른 인터럽트 벡터에 대응하는 것을 특징으로 하는 인터럽트 제어 방법.

청구항 45

청구항 45은(는) 설정등록료 납부시 포기되었습니다.

제37항에 있어서,

상기 인터럽트 메시지를 수신하기 전에, 상기 인터럽트 메시지로부터의 데이터를 상기 제 1 게스트 내의 상기 제 1 목적지에 대응하는 인터럽트 요청 상태를 저장하는 상기 데이터 구조 내의 엔트리에 맵핑시키는 상태 맵핑 데이터 구조를, 상기 시스템의 메모리에 저장하는 단계; 및

상기 데이터 구조를 업데이트하기 전에 상기 인터럽트 메시지 내의 데이터가 상기 엔트리를 로케이팅하는 것에 응답하여 상기 상태 맵핑 데이터 구조에 액세스하는 단계를 더 포함하고,

상기 데이터 구조에서 상기 인터럽트 요청 상태를 업데이트하는 단계는 상기 엔트리 내의 인터럽트 요청 상태를 업데이트하는 단계를 포함하는 것을 특징으로 하는 인터럽트 제어 방법.

청구항 46

컴퓨터 시스템으로서,

데이터 구조를 저장하도록 된 메모리 시스템 -상기 데이터 구조는 상기 컴퓨터 시스템 상에서 실행가능한 복수의 게스트들 각각의 각 목적지에 대한 인터럽트 요청 상태를 적어도 저장하며, 그리고 상기 인터럽트 요청 상태는 상기 복수의 게스트들 중 해당 게스트의 해당 인터럽트 제어기에서 어떤 인터럽트들이 요청되었는지를 식별하며- ; 및

상기 복수의 게스트들 중 제 1 게스트의 제 1 목적지를 목표로 하는 인터럽트 메시지를 수신하도록 된 게스트 인터럽트 매니저

를 포함하며,

상기 게스트 인터럽트 매니저는 상기 인터럽트 메시지에서 식별된 인터럽트를 상기 제 1 목적지 및 제 1 게스트에 대한 상기 인터럽트 요청 상태에 기록하도록, 상기 제 1 목적지 및 제 1 게스트에 대응하는 상기 데이터 구조 내의 상기 인터럽트 요청 상태를 업데이트하며, 각각의 인터럽트는 관련된 전달 모드를 가지며, 상기 데이터 구조는 제 1 전달 모드에 대응하는 제 1 섹션과 제 2 전달 모드에 대응하는 제 2 섹션을 포함하며, 각각의 목적지 및 게스트에 대한 상기 인터럽트 요청 상태는 상기 제 1 섹션 및 제 2 섹션 각각에 저장된 데이터를 포함하며, 상기 게스트 인터럽트 매니저는 상기 관련된 전달 모드에 응답하여 상기 인터럽트에 대한 표시를 상기 제 1 섹션 및 제 2 섹션 중 선택된 섹션에 기입함으로써, 상기 인터럽트 요청 상태를 업데이트하는 것을 특징으로 하는 컴퓨터 시스템.

청구항 47

청구항 47은(는) 설정등록료 납부시 포기되었습니다.

제46항에 있어서,

각각의 인터럽트는 관련된 인터럽트 벡터를 가지며, 상기 인터럽트 요청 상태는 비트 벡터를 포함하고 상기 비트 벡터는 각각의 인터럽트 벡터에 대한 비트를 가지며, 상기 제 1 섹션 내의 데이터는 각각의 인터럽트 벡터에 대해 상기 비트를 갖는 제 1 비트 벡터를 포함하고, 상기 제 2 섹션 내의 데이터는 각각의 인터럽트 벡터에 대해 상기 비트를 갖는 제 2 비트 벡터를 포함하고, 그리고 상기 제 1 목적지 및 상기 제 1 게스트에 대응하는 인터럽트 요청 상태는 상기 제 1 비트 벡터와 상기 제 2 비트 벡터의 논리 OR(logical OR)를 포함하는 것을 특징으로 하는 컴퓨터 시스템.

청구항 48

제47항에 있어서,

상기 게스트 인터럽트 매니저는 상기 인터럽트 메시지 내의 인터럽트 벡터에 대응하는 상기 비트를 상기 전달 모드에 응답하여 상기 제 1 비트 벡터 혹은 상기 제 2 비트 벡터 중 하나에 세트하도록 된 것을 특징으로 하는 컴퓨터 시스템.

청구항 49

제47항에 있어서,

상기 제 1 전달 모드는 물리적(physical)이며, 상기 제 1 섹션은 로우들(rows)을 갖는 제 1 테이블을 포함하고, 상기 로우들은 물리적 전달 모드 인터럽트들에 대한 목적지들을 식별하는 물리적 식별자들에 대응하며, 그리고 상기 제 1 비트 벡터는 상기 제 1 테이블의 하나의 로우에 존재하는 것을 특징으로 하는 컴퓨터 시스템.

청구항 50

제49항에 있어서,

상기 제 2 전달 모드는 논리적(logical)이며, 논리적 전달 모드 인터럽트에 대한 하나 이상의 목적지들을 식별하는 논리적 식별자는 클러스터 식별자와 클러스터의 멤버들을 나타내는 식별 비트 벡터를 포함하고, 상기 제 2 섹션은 각각의 클러스터에 관련된 서브섹션들을 갖는 제 2 테이블을 포함하고, 상기 서브섹션의 로우들은 상기 클러스터의 멤버들에 대응하며, 그리고 상기 제 2 비트 벡터는 상기 제 2 테이블의 하나의 로우에 존재하는 것을 특징으로 하는 컴퓨터 시스템.

청구항 51

제47항에 있어서,

상기 제 1 전달 모드는 물리적(physical)이며, 상기 제 1 섹션은 인터럽트 벡터들에 대응하는 로우들을 갖는 제 1 테이블을 포함하고, 각각의 로우의 각각의 비트는 물리적 전달 모드 인터럽트들에 대한 목적지를 식별하는 다른 물리적 식별자에 대응하며, 그리고 상기 제 1 비트 벡터는 상기 제 1 테이블의 하나의 컬럼(column)에 존재하는 것을 특징으로 하는 컴퓨터 시스템.

청구항 52

제51항에 있어서,

상기 제 2 전달 모드는 논리적(logical)이며, 논리적 전달 모드 인터럽트에 대한 하나 이상의 목적지들을 식별하는 논리적 식별자는 클러스터 식별자와 클러스터의 멤버들을 나타내는 식별 비트 벡터를 포함하고, 상기 제 2 섹션은 각각의 클러스터에 관련된 서브섹션들을 갖는 제 2 테이블을 포함하고, 각각의 로우의 각각의 비트는 상기 클러스터의 다른 멤버에 대응하며, 그리고 상기 제 2 비트 벡터는 상기 제 2 테이블의 하나의 서브섹션의 하나의 컬럼에 존재하는 것을 특징으로 하는 컴퓨터 시스템.

청구항 53

제52항에 있어서,

상기 제 1 섹션의 각각의 로우 및 상기 제 2 섹션의 각 서브섹션의 각각의 로우는 서로 다른 인터럽트 벡터에 대응하는 것을 특징으로 하는 컴퓨터 시스템.

청구항 54

제46항에 있어서,

상기 메모리 시스템은 또한, 상기 인터럽트 메시지로부터의 데이터를, 상기 제 1 게스트 내의 상기 제 1 목적지에 대응하는 인터럽트 요청 상태를 저장하는 상기 데이터 구조 내의 엔트리에 맵핑시키는 상태 맵핑 데이터 구조를 저장하며,

상기 게스트 인터럽트 매니저는 상기 인터럽트 메시지 내의 데이터가 상기 엔트리를 로케이팅하는 것에 응답하여 상기 상태 맵핑 데이터 구조에 액세스하도록 구성되고,

상기 게스트 인터럽트 매니저는 상기 엔트리 내의 상기 인터럽트 요청 상태를 업데이트함으로써 상기 데이터 구

조에서 상기 인터럽트 요청 상태를 업데이트하는 것을 특징으로 하는 컴퓨터 시스템.

발명의 설명

기 술 분 야

[0001] 본 발명은 프로세서들 및 가상화(virtualization)에 관한 것으로서, 보다 특정하게는 가상 머신 게스트들에 인터럽트들을 전달하는 것에 관한 것이다.

배 경 기 술

[0002] 가상화는 컴퓨터 시스템들에서 다양한 다른 목적들로 이용되어 왔다. 예를 들어, 가상화는, 가상 머신을 제어하는 가상 머신 매니저(VMM)에 의해 그렇게 하도록 먼저 허가를 받지 않으면서, 특권 소프트웨어가 물리적인 머신 상태의 적어도 일부를 직접 액세스 및/또는 변경하는 것을 막기 위해 "컨테이너" 내에서 특권 소프트웨어를 실행시키는 데에 이용된다. 이러한 컨테이너는 "버기(buggy)" 또는 악의성(malicious) 소프트웨어가 물리적인 머신 상에서 문제들을 야기하는 것을 막는다. 또한, 가상화는 2개 이상의 특권 프로그램들이 동일한 물리적인 머신 상에서 동시에 실행될 수 있도록 하는 데에 이용될 수 있다. 물리적인 머신에 대한 액세스가 제어되기 때문에, 이러한 특권 프로그램들이 서로 충돌하는 것을 막을 수 있다. 특권 프로그램들은 운영 체제를 포함할 수 있으며, 그리고 소프트웨어가 실행되고 있는 하드웨어 전체를 제어하는 것으로 기대되는 다른 소프트웨어를 또한 포함할 수 있다. 다른 예에서, 가상화는 특권 프로그램에 의해 기대되는 하드웨어와 다른 하드웨어 상에서 특권 프로그램을 실행시키는 데에 이용될 수 있다.

[0003] 일반적으로, 프로세서 또는 컴퓨터 시스템의 가상화는 가상 머신(상기 언급한 컨테이너)에 대한 액세스를 갖는 하나 이상의 특권 프로그램들을 제공하는 것을 포함할 수 있지만, 물리적인 머신의 제어는 VMM에 의해 유지된다. 가상 머신은 프로세서(또는 프로세서들) 메모리, 및 특권 프로그램이 자신이 실행되고 있는 머신 내에서 찾을 것으로 기대되는 다양한 주변 장치들을 포함할 수 있다. 가상 머신은 VMM이 가상 머신에 할당하는 하드웨어에 할당하는 하드웨어에 의해 적어도 일시적으로 구현될 수 있으며, 그리고/또는 소프트웨어 내에서 에뮬레이트(emulate)될 수도 있다. 여기에서, 각 특권 프로그램 (및 일부 경우들에서는, 운영 체제 상에서 실행되는 애플리케이션들과 같은 관련 소프트웨어)은 게스트로서 지칭될 수 있다. 가상화는, VMM 및 그 가상 머신들이 실행되는 물리적인 머신 내에서의 어떠한 특정의 하드웨어 가상화 지원없이, 소프트웨어(예를 들어, 상기 언급한 VMM) 내에서 구현될 수 있다. 하지만, 얼마간의 하드웨어 지원이 제공된다면, 가상화는 단순화될 수 있으며, 그리고/또는 더 높은 성능을 달성할 수 있다.

발명의 내용

해결하려는 과제

[0004] 가상화와 관련하여 발생할 수 있는 하나의 문제는 인터럽트 전달의 레이턴시이다. 상기 설명한 바와 같이, 주변 장치들이 가상 머신에 의해 이용될 수 있도록 할당되어, (가상 머신에서 가상 주변 장치로서 이용된다). 이러한 주변장치들은 가상 머신 내에서 소프트웨어에 의해 처리되어야 하는 인터럽트들을 발생시킬 수 있다. 비-가상화된 환경들에서, 인터럽트 처리 레이턴시는 비교적 짧을 수 있다. 가상화된 환경들에서, 인터럽트들은 일반적으로 VMM에 의해 인터셉트되어 이 VMM에 의해 처리되고, 어떠한 종류의 소프트웨어 메커니즘을 이용하여 VMM에 의해 목표 가상 머신에 전달될 수 있다. 하지만, 인터럽트 처리 레이턴시가 상당히 더 커질 수 있다(예를 들어, 약 100배 더 길어질 수 있다).

과제의 해결 수단

[0005] 일 실시예에서, 시스템은 프로세서와, 프로세서에 결합되는 제 1 인터럽트 제어기와, 그리고 프로세서에 결합되는 제 2 인터럽트 제어기를 포함한다. 제 1 인터럽트 제어기는, 시스템 내의 호스트를 목표로 하는 제 1 인터럽트를 전달하는 제 1 인터럽트 메시지를 수신하는 것에 응답하여, 프로세서에 인터럽트를 신호하도록 구성된다. 제 2 인터럽트 제어기는, 호스트에 의해 제어되고 프로세서 상에서 실행가능한 게스트를 목표로 하는 제 2 인터럽트를 전달하는 제 2 인터럽트 메시지를 수신하는 것에 응답하여, 프로세서에 인터럽트를 신호하도록 구성된다.

도면의 간단한 설명

[0006] 하기의 상세한 설명은 첨부 도면들을 참조하며, 이에 첨부 도면들에 대해 간단히 설명한다.

도 1은 가상화를 구현하는 컴퓨터 시스템의 일 실시예의 블록도이다.

도 2는 도 1에 도시된 호스트 하드웨어의 일 실시예의 블록도이다.

도 3은 게스트에 전달되는 인터럽트의 일 실시예를 도시하는 블록도이다.

도 4는 게스트 APIC(advanced programmable interrupt controller)의 일 실시예를 도시하는 블록도이다.

도 5는 게스트 APIC 상태 데이터 구조에서의 게스트 APIC 상태 엔트리의 일 실시예를 도시하는 블록도이다.

도 6은 게스트를 목표로 하는 인터럽트를 수신하는 것에 응답하여, 도 2에 나타낸 게스트 인터럽트 매니저의 일 실시예의 동작을 도시하는 흐름도이다.

도 7은 인터럽트 메시지를 수신하는 것에 응답하여 게스트 APIC의 일 실시예의 동작을 도시하는 흐름도이다.

도 8은 하나의 게스트로부터 다른 게스트로 게스트 APIC 상태를 변경시키기 위한 가상 머신 모니터의 일 실시예의 동작을 도시하는 흐름도이다.

도 9는 게스트 APIC 상태 엔트리 내에 인터럽트 상태를 배열하는 일 실시예를 도시하는 블록도이다.

도 10은 인터럽트에 대한 게스트 APIC 상태 엔트리의 위치를 찾는 일 실시예를 도시하는 블록도이다.

도 11은 인터럽트에 대한 게스트 APIC 상태 엔트리의 위치를 찾는 다른 실시예를 도시하는 블록도이다.

도 12는 인터럽트에 대한 게스트 APIC 상태 엔트리의 위치를 찾는 또 다른 실시예를 도시하는 블록도이다.

도 13은 인터럽트에 대한 게스트 APIC 상태 엔트리의 위치를 찾는 또 다른 실시예를 도시하는 블록도이다.

도 14는 도 1에 도시된 호스트 하드웨어의 다른 실시예의 블록도이다.

도 15는 VMM의 일 실시예를 저장하는 컴퓨터 액세스가능한 저장 매체의 일 실시예의 블록도이다.

본 발명은 다양한 변경들 및 대안적인 형태들이 가능하지만, 특정의 실시예들이 도면들에 예시적으로 제시되어, 여기에서 상세히 설명될 것이다. 하지만, 이해될 사항으로서, 도면들 및 상세한 설명은 본 발명을 개시된 특정의 형태로 제한하는 것으로 의도되지 않으며, 그렇다기 보다는, 첨부된 청구범위에 의해 정의되는본 발명의 정신 및 범위 내에 있는 모든 변경들, 등가물들 및 대안들을 포괄하는 것으로서 의도된다. 여기에서 이용되는 표제(heading)들은 단지 구성의 목적을 위한 것으로서, 설명의 범위를 제한하기 위해 이용되는 것이 아니다. 본 명세서에서 전체적으로 이용되는 용어 "수도 있다(may)"는, 강제적인 의미(즉, "해야 한다(must)")가 아닌, 허가의 의미(즉, 가능성을 갖는 의미)로 이용된다. 유사하게, 단어들 "포함한다", "포함하는" 및 "포함하고 있는"은 포함하고 있음을 의미하지만, 오직 이것으로만 제한되지 않는다.

다양한 유닛들, 회로들 또는 다른 컴포넌트들이 어떠한 작업 또는 작업들을 수행하도록 "구성되는 것"으로서 설명된다. 이러한 환경에서, "구성되는 것"은 동작하는 동안 작업 또는 작업들을 수행하는 "회로를 가지고 있음"을 일반적으로 의미하는 구조의 광범위한 기재이다. 이 때문에, 유닛/회로/컴포넌트는 이러한 유닛/회로/컴포넌트가 현재 온(on) 상태가 아닐 때에도 작업을 수행하도록 구성될 수 있다. 일반적으로, "~하도록 구성된"에 해당하는 구조를 형성하는 회로는 동작을 구현하기 위한 하드웨어 회로들을 포함할 수 있다. 유사하게, 다양한 유닛들/회로들/컴포넌트들은 설명에 있어서의 편의를 위해 작업 또는 작업들을 수행하는 것으로서 설명될 수 있다. 이러한 설명은 "~하도록 구성된"의 구를 포함하는 것으로서 해석되어야 한다. 하나 이상의 작업들을 수행하도록 구성된 유닛/회로/컴포넌트를 기재하는 것이 명백하게 이러한 유닛/회로/컴포넌트에 대해 35 U.S.C. § 112 패러그래프 6에 의해 해석되도록 의도되지는 않는다.

발명을 실시하기 위한 구체적인 내용

[0007] 일 실시예에서, 컴퓨터 시스템은 적어도 하나의 호스트 인터럽트 제어기 및 적어도 하나의 게스트 인터럽트 제어기를 포함한다. 호스트 인터럽트 제어기는 호스트(예를 들어, 가상화된 환경에서, 가상 머신 매니저, 즉 VMM)에 의해 서비스되는 인터럽트들을 관리할 수 있다. 이러한 인터럽트들은, 예를 들어 (컴퓨터 시스템 상에서 실행되는 게스트에게 할당되지 않은) 컴퓨터 시스템 내의 장치들로부터의 인터럽트들, VMM이 게스트에게 노출하기를 원하지 않는 시스템 레벨 인터럽트들 등을 포함할 수 있다. 게스트 인터럽트 제어기는 게스트에 의해 서비스되는 인터럽트들을 관리할 수 있다. 이러한 인터럽트들은, 예를 들어 게스트의 가상 머신에 대해 장치의 기능

을 제공하기 위해 게스트에게 할당되는 장치에 의해 발행되는 인터럽트들을 포함할 수 있다.

[0008] 시스템 내의 하드웨어는 호스트 인터럽트 제어기 또는 게스트 인터럽트 제어기에게 인터럽트를 전송할 수 있다. 대안적으로, 인터럽트는 인터럽트 제어기들 모두에게 전송될 수 있으며, 인터럽트가 호스트 인터럽트 또는 게스트 인터럽트이냐에 기초하여, 인터럽트 제어기들중 하나가 그 인터럽트를 수락할 수 있다. 각 인터럽트 제어기는 프로세서에 결합될 수 있으며, 이 프로세서와 통신하여 인터럽트를 전달할 수 있다. 몇몇 실시예들에서, 게스트에게 인터럽트들을 전달하기 위해 게스트 인터럽트 제어기를 제공하는 것은, 호스트 인터럽트 레이턴시와 거의 같은 레이턴시를 가지면서 게스트 인터럽트 전달을 제공할 수 있다.

[0009] 일 실시예에서, 게스트 인터럽트 제어기는 호스트 인터럽트 제어기의 복사본(duplicate)이 될 수 있다. 즉, 게스트 인터럽트 제어기는 호스트 인터럽트 제어기와 동일한 하드웨어를 포함할 수 있다. 다른 실시예에서, 게스트 인터럽트 제어기는 호스트 인터럽트 제어기의 단지 일부분 만을 복사할 수 있다. 게스트 인터럽트 제어기에 의해 복사되지 않는 부분들은 VMM 인터셉트 및 VMM에 의해 에뮬레이션에 의해 가상화될 수 있다.

[0010] 가상화 개요

[0011] 도 1은 가상화를 구현하는 컴퓨터 시스템(5)의 일 실시예의 블록도이다. 도 1의 실시예에는, 다수의 게스트들(10A-10N)이 나타나 있다. 게스트(10A)는 게스트 운영 체제(OS)(12) 및 이러한 게스트 OS(12) 상에서 실행되는 하나 이상의 애플리케이션들(14A-14N)을 포함한다. 게스트(10N)는 특권 코드(privileged code)(16)를 포함한다. 게스트들(10A-10N)은 가상 머신 매니저(VMM)(18)에 의해 관리된다. VMM(18) 및 게스트들(10A-10N)은 호스트 하드웨어(20) 상에서 실행되며, 호스트 하드웨어(20)는 컴퓨터 시스템(5) 내에 포함되는 물리적인 하드웨어를 포함할 수 있다. 일 실시예에서, VMM(18)은 가상 머신 제어 블록들(VMCBs)(22)의 세트를 유지할 수 있다. 각 게스트(10A-10N)에 대해 하나의 VMCB(22)가 있을 수 있다. 도 1에서는 설명을 위해 VMCB들(22)이 VMM(18)의 일부인 것으로서 나타내었지만, VMCB들(22)은 메모리 내에 그리고/또는 호스트 하드웨어(20) 내의 디스크 드라이브들과 같은 비휘발성 매체에 저장될 수 있다.

[0012] 일반적으로, 호스트 하드웨어(20)는 컴퓨터 시스템(5) 내에 포함되는 모든 하드웨어를 포함한다. 다양한 실시예들에서, 호스트 하드웨어(20)는 하나 이상의 프로세서들, 메모리, 주변 장치들, 및 이러한 컴포넌트들을 결합시키는 데에 이용되는 다른 회로를 포함할 수 있다. 구체적으로, 호스트 하드웨어(20)는 하나 이상의 호스트 인터럽트 제어기들, 하나 이상의 게스트 인터럽트 제어기들, 그리고/또는 하나 이상의 게스트 인터럽트 매니저들을 포함할 수 있다. 예를 들어, 개인용 컴퓨터(PC)-스타일 시스템들은 프로세서들을 결합시키는 노쓰브리지(Northbridge), 메모리, 및 AGP(advanced graphic port) 인터페이스를 이용하는 그래픽 장치를 포함할 수 있다. 또한, 노쓰브리지는 PCI(peripheral component interface) 버스와 같은 주변 버스에 결합될 수 있으며, 이러한 PCI 버스에는 다양한 주변 컴포넌트들이 직접적으로 또는 간접적으로 결합될 수 있다. 또한, 사우쓰브리지(Southbridge)가 포함되는 바, 이는 PCI 버스에 결합되어, 레거시 기능을 제공하고 그리고/또는 레거시 하드웨어에 결합된다. 다양한 구현들에서, 게스트 인터럽트 매니저는 노쓰브리지, 사우쓰브리지, 또는 인터페이스들 중 하나 상의 디바이스에서 구현될 수 있다. 호스트 및 게이트 인터럽트 제어기들은 각 프로세서에 대해 구현되거나, 또는 프로세서들의 그룹 사이에서 공유될 수 있다. 다른 실시예들에서는, 다른 회로를 이용하여 다양한 하드웨어 컴포넌트들을 연결할 수 있다. 예를 들어, HyperTransport™ (HT) 링크들을 이용하여 노드들을 연결할 수 있으며, 이러한 노드들 각각은 하나 이상의 프로세서들, 호스트브리지 및 메모리 제어기를 포함할 수 있다. 각 노드는 또한 노쓰브리지를 포함할 수 있으며, 이러한 노쓰브리지는 게스트 인터럽트 매니저 그리고/또는 호스트 및 게스트 인터럽트 제어기들을 포함할 수 있다. 대안적으로, 호스트 브리지는 게스트 인터럽트 매니저 그리고/또는 호스트 및 게스트 인터럽트 제어기들을 포함할 수 있다. 호스트 브리지는 데이지 체인(daisy chain) 방식으로 HT 링크들을 통해 주변 장치들에 연결하는 데에 이용될 수 있다. 요구되는 임의의 회로/호스트 하드웨어 구조가 이용될 수 있다.

[0013] VMM(18)은 게스트들(10A-10N) 각각에 대한 가상화를 제공하도록 구성될 수 있으며, 호스트 하드웨어(20)에 대한 게스트들(10A-10N)의 액세스를 제어할 수 있다. 또한, VMM(18)은 호스트 하드웨어(20) 상에서의 실행을 위해 게스트들(10A-10N)을 스케쥴링하는 것을 담당할 수 있다. VMM(18)은 가상화를 위해 호스트 하드웨어(20)에 제공되는 하드웨어 지원을 이용하도록 구성될 수 있다. 예를 들어, 프로세서들은, 이벤트들을 인터셉트하고, 핸들링을 위해 게스트를 VMM(18)로 빠져나가게 하는 하드웨어를 포함하여, 가상화를 위한 하드웨어 지원을 제공할 수 있다. 게스트 인터럽트 매니저 그리고/또는 게스트 인터럽트 제어기들 역시 가상화를 지원하기 위해 제공되는 하드웨어가 될 수 있다.

[0014] 몇몇 실시예들에서, VMM(18)은 "얇은(thin)" 독립형 프로그램으로서 구현될 수 있는데, 이러한 프로그램은 호스

트 하드웨어(20) 상에서 실행되며, 게스트들(10A-10N)에 대한 가상화를 제공한다. 이러한 VMM 구현은 종종 "하이퍼바이저(hypervisor)"로서 지칭될 수 있다. 다른 실시예들에서, VMM(18)은 호스트 OS에 통합되거나, 또는 이러한 호스트 OS 상에서 실행될 수 있다. 이러한 실시예들에서, VMM(18)은 호스트 OS 내의 임의의 드라이버들, 시스템 BIOS에 의해 제공되는 플랫폼 시스템 관리 모드(SMM) 코드 등을 포함하는 호스트 OS에 의존할 수 있다. 따라서, 호스트 OS 컴포넌트들 (및 플랫폼 SMM 코드와 같은 다양한 하위-레벨 컴포넌트들)은 호스트 하드웨어(20) 상에서 직접 실행되며, VMM(18)에 의해 가상화되지 않는다. 일 실시예에서, VMM(18) 및 (포함되는 경우) 호스트 OS가 함께 호스트로서 지칭될 수 있다. 일반적으로, 호스트는 이용되는 동안 호스트 하드웨어(20)에 의해 직접적으로 제어되는 임의의 코드를 포함할 수 있다. 예를 들어, 호스트는 VMM(18)이거나, 또는 호스트 OS와 함께 VMM(18)이거나, 또는 (예를 들어, 비가상화된 환경에서는) 호스트 OS 단독이 될 수 있다.

[0015] 다양한 실시예들에서, VMM(18)은 전 가상화(full virtualization), 반 가상화(paravirtualization), 또는 양쪽 모두를 지원할 수 있다. 또한, 몇몇 실시예들에서, VMM(18)은 반 가상화된 게스트들과 전 가상화된 게스트들을 동시에 실행시킬 수 있다.

[0016] 전 가상화를 이용하게 되면, 게스트들(10A-10N)은 가상화가 일어나고 있다는 것을 인식하지 못한다. 각 게스트(10A-10N)는 자신의 가상 머신 내에 인접하는 제로 기반 메모리(contiguous zero based memory)를 가질 수 있으며, VMM(18)은 쉐도우 페이지 테이블들(shadow page tables) 또는 내포형 페이지 테이블들(nested page tables)을 이용하여, 호스트 물리 어드레스 공간에 대한 액세스를 제어할 수 있다. 쉐도우 페이지 테이블들은 게스트 가상 어드레스들로부터 호스트 물리 어드레스들로 리맵핑(remapping)(실제로는, 게스트(10A-10N) 내의 메모리 관리 소프트웨어에 의해 할당되는 게스트 "물리 어드레스"를 호스트 물리 어드레스들로 리맵핑함)하는 반면, 내포형 페이지 테이블들은 게스트 물리 어드레스를 입력으로서 수신하여, 호스트 물리 어드레스에 맵핑한다. 각 게스트(10A-10N)에 대해 쉐도우 페이지 테이블들 또는 내포형 페이지 테이블들을 이용함으로써, VMM(18)은 게스트들이 호스트 하드웨어(20) 내의 다른 게스트들의 물리적인 메모리를 액세스하지 않도록 보장한다.

[0017] 반 가상화를 이용하게 되면, 게스트들(10A-10N)은 적어도 부분적으로 VM을 인식(VM-aware)하게 된다. 이러한 게스트들(10A-10N)은 VMM(18)과 메모리 페이지들에 대해 협상할 수 있으며, 이에 따라 호스트 물리 어드레스들로의 게스트 물리 어드레스들의 리맵핑이 요구되지 않을 수도 있다. 일 실시예에서, 반 가상화시, 게스트들(10A-10N)은 호스트 하드웨어(20) 내의 주변 장치들과 직접 상호작용하도록 허가될 수 있다. 임의의 소정의 시간에, 주변 장치는 게스트 또는 게스트들(10A-10N)에 의해 "소유(own)"될 수 있다. 예를 들어, 일 구현에 있어서, 주변 장치는 그 주변 장치를 현재 소유하고 있는 하나 이상의 게스트들(10A-10N)을 갖는 보호 도메인(protection domain) 내로 맵핑될 수 있다. 주변 장치를 소유하는 게스트들 만이 그 주변 장치와 직접 상호작용할 수 있다. 또한, 보호 도메인 내의 장치들이 다른 보호 도메인 내의 게스트에 할당된 페이지들을 판독/기록하는 것을 막기 위한 보호 메커니즘이 있다.

[0018] 이전에 설명한 바와 같이, VMM(18)은 각 게스트(10A-10N)에 대해 VMCB(22)를 유지할 수 있다. 일반적으로, VMCB(22)는 해당하는 게스트(10A-10N)에 대해 VMM(18)에 의해 할당되는 저장 영역에 저장된 데이터 구조를 포함할 수 있다. 일 실시예에서, VMCB(22)는 메모리의 페이지를 포함하지만, 다른 실시예들은 더 크거나 또는 더 작은 메모리 영역들을 이용하고 그리고/또는 비휘발성 저장소와 같은 다른 매체 상의 저장소를 이용할 수 있다. 일 실시예에서, VMCB(22)는 게스트의 프로세서 상태를 포함할 수 있는 바, 이는 게스트가 실행하도록 스케쥴될 때에는 호스트 하드웨어(20) 내의 프로세서에 적재되며, 그리고 (스케쥴된 시간이 끝남으로 인해, 또는 게스트를 빠져나오게 하기 위해 프로세서가 검출하는 하나 이상의 인터셉트들로 인해) 게스트가 빠져나올 때에는 VMCB(22)에 다시 저장될 수 있다. 몇몇 실시예들에서는, 프로세서 상태의 단지 일부 만이 VMCB(22)에 해당하는 게스트에 대한 제어를 전달하는 명령("가상 머신 런(VMRN)" 명령)을 통해 적재되며, 요구되는 다른 상태는 VMRUN 명령을 실행하기 전에 VMM(18)에 의해 적재될 수 있다. 유사하게, 이러한 실시예들에서는, 프로세서 상태의 단지 일부 만이 게스트가 빠져나갈 때에 프로세서에 의해 VMCB(22)에 저장될 수 있으며, VMM(18)은 요구되는 임의의 부가적인 상태를 저장하는 것을 담당할 수 있다. 다른 실시예들에서, VMCB(22)는 프로세서 상태가 저장되는 다른 메모리 영역에 대한 포인터를 포함할 수 있다. 또한, 일 실시예에서는, 2개 이상의 퇴장 메커니즘(exit mechanism)이 정의될 수 있다. 일 실시예에서, 저장되는 상태의 양 및 적재되는 상태의 위치는 어떤 퇴장 메커니즘이 선택되는 지에 의존하여 달라질 수 있다.

[0019] 일 실시예에서, VMM(18)은 또한 VMM(18)에 해당하는 프로세서 상태를 저장하도록 할당되는 메모리 영역을 가질 수 있다. VMRUN이 실행될 때, VMM(18)에 해당하는 프로세서 상태가 이러한 영역에 세이브(save)될 수 있다. 게스트가 VMM(18)로 빠져나갈 때, 이러한 영역으로부터의 프로세서 상태는 이 영역으로부터 재적재(reload)되어, VMM(18)이 실행을 계속할 수 있게 한다. 예를 들어, 하나의 구현에 있어서, 프로세서는 VMM(18) 세이브 영역의

어드레스를 저장하기 위해 레지스터(예를 들어, 모델 특정의 레지스터, 즉 MSR)를 구현할 수 있다.

[0020] 부가적으로, VMCB(22)는 게스트에 대해 인에이블되는 인터셉트 이벤트들을 식별하는 인터셉트 구성(intercept configuration), 및 인에이블된 인터셉트 이벤트가 검출되는 경우 게스트를 빠져나오게 하기 위한 메커니즘을 포함할 수 있다. 일 실시예에서, 인터셉트 구성은 인터셉트 표시들의 세트를 포함할 수 있는 바, 프로세서가 지원하는 각 인터셉트 이벤트에 대해 하나의 표시를 갖는다. 이러한 인터셉트 표시는 프로세서가 해당하는 이벤트를 인터셉트할 것인 지의 여부(또는, 다른 점에서 보면, 인터셉트가 인에이블되는 지의 여부)를 표시할 수 있다. 여기에서 이용되는 바와 같이, 이벤트는 그 이벤트가 게스트 내에서 일어나는 경우 그 게스트 내에서 "인터셉트"되며, 프로세서는 그 이벤트의 처리에 대해 그 게스트를 빠져나오게 한다. 일 실시예에서, 인터셉트 구성은 제 2 세트의 표시들을 포함할 수 있는데, 이러한 표시들은 2개의 퇴장 메커니즘들 중에서 어느 메커니즘이 이용되는 지를 나타낸다. 다른 실시예들은 2개 이상의 퇴장 메커니즘들을 정의할 수 있다. 다른 실시예들에서, 인터셉트 구성은 인터셉트 이벤트 마다 하나씩 인터셉트 표시들의 제 1 세트(이들은 제 1 퇴장 메커니즘이 이벤트에 대해 이용되어야 하는 지를 나타낸다); 및 인터셉트 이벤트 마다 하나씩 인터셉트 표시들의 제 2 세트(이들은 제 2 퇴장 메커니즘이 이벤트에 대해 이용되어야 하는 지를 나타낸다)를 포함한다.

[0021] 일반적으로, 퇴장 메커니즘은, (일반적으로 재시작가능한 방식으로) 게스트 실행을 빠져나오고 다른 코드의 실행을 시작하기 위해 프로세서에 의해 수행되는 동작들을 정의할 수 있다. 일 실시예에서, 하나의 퇴장 메커니즘은 프로세서 상태의 작은 양을 세이브하고, 미니바이저(Minivisor)에 대해 상태를 적재하는 것을 포함할 수 있다. 이러한 미니바이저는 게스트의 물리적인 어드레스 공간 내에서 실행될 수 있으며, 비교적 간단한 인터셉트 처리를 수행할 수 있다. 다른 퇴장 메커니즘이 VMM에 빠져나갈 수 있으며, 이에 의해 프로세서 상태의 더 많은 양을 세이브하고, VMM의 프로세서 상태를 적재한다. 따라서, 인터셉트 이벤트들은 이벤트에 따라 다른 명령 코드에 의해 처리될 수 있다. 또한, 몇몇 실시예들에서는, 비교적 간단한 인터셉트 처리가 "보다 경량(lighter weight)"의 퇴장 메커니즘(이는 수행하는 데에 시간이 덜 걸린다)을 통해 처리됨으로써, 성능을 개선할 수 있다. "보다 중량(heavier weight)"의 메커니즘을 이용하여 빠져나온 후에, 보다 복잡한 처리가 VMM에서 수행될 수 있다. 따라서, 본 실시예에서, VMM(18)은 게스트(10A-10N)가 내부적으로 처리하는 것을 VMM(18)이 원하지 않는 이벤트들을 인터셉트하도록 프로세서를 구성할 수 있으며, 또한 퇴장 메커니즘을 이용하도록 프로세서를 구성할 수 있다. 이벤트들은 명령들(즉, 명령을 실행하는 대신 그 명령을 인터셉트한다), 인터럽트들, 예외들, 그리고/또는 게스트 실행 동안 일어날 수 있는 요구되는 어떠한 다른 이벤트들을 포함할 수 있다.

[0022] 일 실시예에서, VMCB(22)는 이 VMCB(22)를 적재할 때에 프로세서로 하여금 어떠한 행동들을 수행하게 하는 다른 제어 비트들을 더 포함할 수 있다. 예를 들어, 이러한 제어 비트들은 프로세서 내에서 TLB를 플러싱(flushing)하기 위한 명령들을 포함할 수 있다. 다른 제어 비트들은 게스트에 대한 실행 환경(예를 들어, 인터럽트 핸들링 모드들, 게스트에 대한 어드레스 공간 식별자 등)을 특정할 수 있다. 또 다른 제어 비트들은 왜 게스트가 퇴장하는 지 등을 기술하는 퇴장 코드(exit code)를 전달하는 데에 이용될 수 있다.

[0023] 일반적으로, "게스트"는 컴퓨터 시스템(5) 내에서의 실행을 위해 가상화되는임의의 하나 이상의 소프트웨어 프로그램들을 포함할 수 있다. 게스트는 특권 모드에서 실행되는 적어도 어떠한 코드를 포함할 수 있으며, 이에 따라 자신이 실행되고 있는 컴퓨터 시스템 상에서의 전체 제어를 갖는 것으로 기대된다. 이전에 설명한 바와 같이, 게스트(10A)는, 게스트가 게스트 OS(12)를 포함할 수 있는 하나의 예이다. 게스트 OS(12)는, 마이크로소프트사(워싱턴 레드몬드 소재)로부터 입수가능한 Windosw OS들중 임의의 것, 리눅스, IMB사(뉴욕 아멍크 소재)로부터의 AIX와 같은 임의의 UNIX-타입 운영 체제, 선마이크로시스템스사(캘리포니아 산타클라라 소재)로부터의 솔라리스, 휴렛팩커드사(캘리포니아 팔로 알토 소재)로부터의 HP-UX 등과 같은 임의의 OS가 될 수 있다. 게스트(10N)는 비(non)-OS 특권 코드(16)를 포함하는 게스트의 하나의 예이다.

[0024] 주목할 사항으로서, 10N 등과 같이 기호 "N"이 참조 부호들에서 이용되면, 이는 일반적으로 그 참조 부호를 가지고 있는 요소들의 임의의 번호(예를 들어, 하나의 게스트를 포함하는, 게스트들(10A-10N)의 임의의 번호)를 나타내는 것으로 의도된다. 또한, 기호 "N"을 이용하는 다른 참조 부호들(예를 들어, 10N 및 14N)은, 달리 나타내지 않는 한, 다른 요소들이 같은 수로 제공됨을 나타내는 것은 아니다(예를 들어, 게스트들(10A-10N)의 수는 애플리케이션들(14A-14B)의 수와 다를 수 있다).

[0025] 호스트 하드웨어

[0026] 이제, 도 2를 참조하면, 호스트 하드웨어(20)의 일 실시예를 도시하는 블록도가 나타나있다. 도시된 실시예에서, 호스트 하드웨어(20)는 다수의 프로세서들(30A-30B), 각각의 호스트 진보된 프로그램가능 인터럽트 제어기들(Advanced Programmable Interrupt Controllers)(hAPICs)(34A-34B), 각각의 게스트

APICs(gAPICs)(34A-34B), 선택적인 부가의 gAPICs(34C-34D), 브리지(36)(이는 게스트 인터럽트 매니저(38), 입/출력(I/O) 메모리 관리 유닛(IOMMU)(40) 및 메모리 제어기(42)를 포함한다), 다수의 인터페이스 회로들(IF)(44A-44C), 메모리 인터페이스 회로(MIF)(46), IOAPIC(50)를 포함할 수 있는 선택적인 브리지(48), 주변장치들(52A-52B)(이들중 일부는 IOAPIC(54)와 같은 IOAPIC를 포함할 수 있다), 및 메모리(56)를 포함한다. 도 2에 도시된 바와 같이, 프로세서들(30A-30B)은 브리지(36)에 결합되며, 그리고 각각의 hAPICs(32A-32B) 및 gAPICs(34A-34B)에 결합된다. hAPICs(32A-32B) 및 gAPICs(34A-34B)는 브리지(36)에 결합되며, 이 브리지(36)는 인터페이스 회로들(44A-44C) 및 메모리 인터페이스 회로(46)에 결합된다. 메모리 인터페이스 회로(46)는 메모리(56)에 결합되고, 인터페이스 회로(44A)는 브리지(48)에 결합되며, 그리고 브리지(48)는 주변장치들(52A-52B)에 결합된다.

[0027] 도시된 실시예에서, 각 프로세서(30A-30B)는 관련된 hAPIC(32A-32B) 및 적어도 하나의 관련된 gAPIC(34A-34D)를 갖는다. 이러한 실시예에서, 인터럽트들은 인텔(캘리포니아 산타 클라라 소재)에 의해 규정되는 APIC 사양에 따라 호스트 하드웨어(20) 내에서 통신될 수 있다. 이 사양에서, 각 프로세서는 관련된 로컬 APIC를 가지며, 이 로컬 APIC는 프로세서 자체, 다른 프로세서들, 내부 APIC 인터럽트 소스들, 및 주변장치들과 관련된 IOAPIC들 모두로부터 인터럽트들을 수신한다. 이러한 로컬 APIC는 계류중인(pending) 인터럽트들의 우선순위를 매기고, 프로세서 내에서 진행되고 있는 다른 인터럽트들 보다 더 높은 우선순위를 가지면, 그리고/또는 프로세서의 현재 작업 보다 높은 우선순위를 가지면, 프로세서에 인터럽트를 전송한다.

[0028] 도 2의 실시예에서, hAPIC(32A-32B)는 프로세서의 호스트 인터럽트들(즉, 호스트에 의해 처리될 인터럽트들)에 대한 로컬 APIC가 될 수 있으며, 그리고 gAPIC(36A-36D)는 프로세서의 게스트 인터럽트들(즉, 각각의 프로세서(30A-30B) 상에서 활동중인 게스트에 의해 처리될 인터럽트들)에 대한 로컬 APIC가 될 수 있다. 게스트가 프로세서 상에서 활동적이 될 수 있는 경우는, 게스트가 그 프로세서 상에서 현재 실행되고 있는 경우(예를 들어, VMRUN 명령이 그 게스트에 대하여 프로세서 상에서 실행되었고, 게스트 퇴장이 아직 일어나지 않은 경우), 또는 게스트가 퇴장하고 VMM(18)이 실행되고 있지만, 그 게스트가 그 프로세서 상에서 다시 실행될 것으로 기대되는 경우이다.

[0029] VMM(18)이 프로세서(30A-30B) 상에서의 게스트를 스케쥴링할 때, 이 VMM(18)은 그 게스트에 해당하는 gAPIC 상태를 갖는 그 프로세서(30A-30B)의 gAPIC(34A-34D)를 적재할 수 있다. 구체적으로, 소정의 게스트는 다수의 가상 CPU들(vCPUs)을 가질 수 있다. VMM(18)은 프로세서(30A-30B) 상에서 실행하기 위해 게스트의 vCPU를 스케쥴링하고, 그 게스트의 가상 머신 내의 상기 vCPU에 대한 인터럽트 상태를 갖는 gAPIC(34A-34D)를 적재할 수 있다. 부가적으로, 이러한 게스트 (및 vCPU)를 목표로 하고 있으며 이 게스트가 활동중인 동안 시그널링되는 임의의 인터럽트들은 gAPIC(34A-34D)에 의해 캡춰될 수 있다. 상기 설명한 바와 같이, 이러한 gAPIC(34A-34D)는 APIC 사양에 따라 게스트를 인터럽트할 수 있다.

[0030] 소정의 프로세서(30A-30B)에 대한 hAPIC(32A-32B) 및 gAPIC(34A-34D)는 그 프로세서에 대한 임의의 인터페이스를 가질 수 있다. 예를 들어, 로컬 APIC들과 이들 각각의 프로세서들 간에 이용되는 임의의 인터페이스가 이용될 수 있다. 각 APIC는 인터럽트가 서비스를 위해 전달되고 있음을 프로세서에게 독립적으로 신호하도록 구성될 수 있다. 만일 프로세서가 게스트를 실행시키고 있고, 게스트 인터럽트가 신호되는 경우, 그 프로세서는 게스트 코드를 인터럽트하고, 그 게스트의 가상 머신 내에서 정확한 인터럽트 핸들러의 실행을 시작하도록 구성될 수 있다. 따라서, 일 실시예에서, 게스트 인터럽트는 호스트 내의 인터럽트의 전달과 유사한 레이턴시를 가지며 전달될 수 있다. 만일 프로세서가 게스트를 실행시키고 있고, hAPIC가 인터럽트를 신호하는 경우, 프로세서는 호스트 인터럽트를 처리하기 위해 VMM(18)으로 게스트를 퇴장시키도록 구성될 수 있다. 만일 프로세서가 게스트를 실행시키고 있지 않으면, gAPIC에 의해 신호되는 인터럽트는 그 게스트가 다시 실행될 때 까지 차단될 수 있다. 만일 프로세서가 게스트를 실행시키고 있지 않고, hAPIC가 인터럽트를 신호하는 경우, 그 프로세서는 호스트 실행을 인터럽트시키고, 호스트 인터럽트 핸들러로 분기(branch)하도록 구성될 수 있다.

[0031] 일 실시예에서는, 하나 이상의 gAPIC(34A-34D)가 프로세서(30A-30B) 마다 포함될 수 있다. 각 gAPIC(34A-34D)는 다른 게스트/vCPU에 해당하는 APIC 상태를 저장할 수 있다. 이러한 실시예에서, 각 gAPIC(34A-34D)는 프로세서에 대해 게스트 인터럽트를 신호할 때 어떤 게스트가 해당되는 지를 식별하도록 구성될 수 있다(또는, 프로세서(30A-30B)가 내부 레지스터들을 가질 수 있는데, 이러한 내부 레지스터들은 어떤 게스트가 현재 각 gAPIC(34A-34D)에 할당되어 있는 지를 식별한다). VMM(18)이 실행하고 있지 않을 때에 게스트 인터럽트를 차단하는 것과 유사하게, 다른 게스트가 현재 실행되고 있으면, 프로세서는 게스트 인터럽트를 차단할 수 있다. 대안적으로, 각 gAPIC(34A-34D)는 해당하는 게스트가 스케쥴될 때에 VMM(18)에 의해 활동적인 것으로 설정될 수 있는 활동 표시(active indication)를 포함할 수 있으며, 그리고 gAPIC(34A-34D)는 해당하는 게스트가 활동적일

때에만 자신의 게스트 인터럽트를 신호하도록 구성될 수 있다. 프로세서(30A-30B) 마다 1개 이상의 gAPIC(34A-34D)를 포함하게 되면, 다수의 게스트들이 시간의 경과에 따라 프로세서 상에서 실행하도록 스케쥴될 때에 gAPIC 상태 변동의 양을 줄일 수 있다. 예를 들어, 프로세서(30A-30B) 마다 N개의 gAPIC들(34A-34D)이 있는 경우(여기서, N은 0 보다 큰 정수이다), gAPIC 상태가 게스트들중 임의의 것에 대해 세이브될 것이 요구되기 전에, N개까지의 다른 게스트들이 실행을 위해 스케쥴링될 수 있다. 프로세서(30A-30B) 마다 1개 보다 많은 gAPIC(34A-34D)를 구현하는 몇몇 실시예들에서, 인터럽트 메시지들이 적절히 수락되고 그리고/또는 로그(log)되도록 보장하기 위해, gAPIC들(34A-34D)은 부가적인 상태를 포함할 수 있다. 예를 들어, gAPIC들(34A-34D)은 "현재 실행(currently running)" 표시를 포함할 수 있는데, 이러한 현재 실행 표시는 (VMM 실행에 대해 정지(suspension)되는 것과 대조적으로, 또는 다른 가상 머신이 실행되고 있는 동안), 해당하는 가상 머신이 해당하는 프로세서(30A-30B) 상에서 현재 실행되고 있는 지를 식별한다. 만일 현재 실행 표시가 가상 머신이 실행 중임을 나타내면, gAPIC는 인터럽트 메시지를 수락할 수 있다. 만일 현재 실행 표시가 가상 머신이 실행중이 아님을 나타내면, gAPIC는 비수락(not-accepted) 인터럽트를 신호할 수 있다. 대안적으로, gAPIC는 이 gAPIC가 비수락 인터럽트를 신호할 것인 지의 여부를 나타내는 부가적인 표시를 포함할 수 있다. 이러한 실시예에서, gAPIC는 현재 실행 표시가 현재 실행되고 있지 않음을 표시하고, 비수락 표시가 gAPIC가 비수락 인터럽트를 신호하는 경우, 비수락 인터럽트를 신호할 수 있다. 이러한 기능은 실행되고 있지 않는 게스트에 대해 인터럽트가 수신되는 것을 검출하는 데에 이용될 수 있으며, 이는 그 인터럽트에 의해 목표되는 게스트를 스케쥴링하는 데에 이용될 수 있다.

[0032] gAPIC들(34A-34D)은 hAPIC들(32A-32D)에 포함되는 하드웨어의 적어도 일부를 포함할 수도 있고, 모든 하드웨어를 포함할 수도 있다(예를 들어, hAPIC들(32A-32D)의 복사본이 될 수 있다). gAPIC(34A-34D)가 어느 게스트에 할당되는 지를 식별하기 위해, gAPIC들(34A-34D)은, APIC 상태 뿐 아니라, 게스트 식별자(ID)에 의해 프로그램 가능하다. 만일 게스트가 다수의 vCPU들을 포함한다면, 물리적인 APIC ID 및 논리적인 APIC ID가 그 게스트 내의 vCPU를 식별할 수 있다. 일 실시예에서, 게스트 ID는 주변 장치들에 대해 IOMMU(40)에 의해 지원되는 도메인 ID와 동일할 수 있다. 대안적으로, 게스트 ID는 개별적으로 관리되는 자원이 될 수 있다. 어느 경우이든, VMM(18)은 게스트들에 게스트 ID들을 할당할 수 있으며, gAPIC들(34A-34D)이 각 게스트에 대해 적절히 프로그램되도록 보장할 수 있다. 본원에서 vCPU 그리고/또는 gAPIC 그리고/또는 그 쌍은 게스트 내의 인터럽트의 목적지로서 보다 간결하게 지칭될 수 있다. 궁극적으로, 이러한 목적지는 인터럽트를 서비스하는 vCPU가 될 수 있지만, 해당하는 gAPIC 역시 목적지로서 보여질 수 있는데, 왜냐하면 이것은 해당하는 프로세서와 관련되며, 인터럽트를 기록하기 때문이다.

[0033] gAPIC들(34A-34D) 및 hAPIC들(32A-32B)은 브리지(36)에 결합되어, 인터럽트들을 수신한다. 임의의 인터페이스를 이용하여, gAPIC들(34A-34D) 및 hAPIC들(32A-32B)에 인터럽트들을 전송할 수 있다. 예를 들어, APIC 인터럽트 전송을 위해 구현되는 임의의 인터페이스가 이용될 수 있다. 일 실시예에서는, 프로세서들(30A-30B)로/로부터 다른 동작들(프로세서들(30A-30B)에 의해 개시되는 메모리 판독/기록 동작들, 캐시 코히런시 유지를 위한 프로브 등)을 전달하는 데에 이용되는 것과 동일한 통신 메커니즘이 인터럽트 메시지들을 전송하는 데에 이용될 수 있다. 다른 측면에서 보면, gAPIC들(34A-34D) 및 hAPIC들(32A-32D)의 결합이 브리지(36)로의 프로세서들(30A-30B)의 결합과 공유될 수 있다. 대안적으로, 가령 gAPIC들(34A-34D) 및 hAPIC들(32A-32D)이 APIC "3 와이어 인터페이스"를 이용한다면, 프로세서들(30A-30B)은 브리지(36)에 대한 개별적인 경로를 가질 수 있다. 인터럽트 메시지는, 전송되고 있는 인터럽트 및 그 인터럽트의 목적지를 식별하는 임의의 인터페이스 상에서의 임의의 통신이 될 수 있다. 예를 들어, 인터럽트들은 관련된 인터럽트 벡터들을 가질 수 있으며, 인터럽트 벡터는 인터럽트 메시지의 일부가 될 수 있다. 이러한 인터럽트 메시지는 또한 게스트 ID 및 목적지 ID(예를 들어, 논리적 또는 물리적 APIC ID)를 포함할 수 있다.

[0034] hAPIC들(32A-32B)은 논리적 APIC들과 유사할 수 있다. 예를 들어, hAPIC들(32A-32B)은 게스트 식별을 위한 부가적인 하드웨어를 포함하지 않을 수도 있는데, 왜냐하면 hAPIC들은 호스트 인터럽트들에 대해 이용되기 때문이다. 대안적으로, hAPIC들(32A-32B)는 부가적인 하드웨어를 포함할 수 있지만, 이러한 부가적인 하드웨어는 hAPIC들(32A-32B)이 호스트 인터럽트들을 위한 것임을 나타내도록 프로그램될 수 있다. 브리지(36)에 의해 hAPIC들(32A-32B) 및 gAPIC들(34A-34D)에 전송되는 인터럽트 메시지들은 호스트 인터럽트들과 대비되는 게스트 인터럽트들을 식별할 수 있으며, 게스트 인터럽트들에 대한 게스트 ID를 포함하거나 (또는, 호스트 인터럽트를 나타내기 위해, 0 제로 또는 모두 이진 1과 같은, 지정된 게스트 ID를 이용할 수 있다). (호스트 인터럽트의 물리적 APIC ID 또는 논리적 APIC ID가 해당 hAPIC ID와 일치하는 경우) hAPIC들(32A-32B)은 호스트 인터럽트들로서 식별되는 인터럽트들을 수락하도록 구성될 수 있으며, 그리고 (게스트 ID가 일치하고, 게스트 인터럽트의 물리적 APIC ID 또는 논리적 APIC ID가 해당 gAPIC ID와 일치하는 경우) gAPIC들(34A-34D)은 자신들의 각각의

게스트들에 대한 게스트 인터럽트들을 수락하도록 구성될 수 있다.

[0035] gAPIC들(34A-34D)이 활동적인 게스트들을 관리할 수 있기는 하지만, 일부 게스트들은 활동적이 아니고 (그리고/또는 게스트 인터럽트들에 의해 목표로 되지 않는 비활동적인 vCPU들을 가질 수도 있다). 일 실시예에서, 게스트 인터럽트 매니저(38)는 비활동적인 게스트들에 대한 게스트 인터럽트 상태를 유지하도록, 그리고 활동적인 게스트들에 대한 gAPIC들이 자신들의 인터럽트들을 확실하게 수신하도록 구성될 수 있다.

[0036] 특히, 일 실시예에서, 게스트 인터럽트 매니저(38)는 분배 인터럽트 전달 방식(distributed interrupt delivery scheme)을 이용할 수 있는 바, 여기서 게스트 인터럽트 매니저(38)는 브리지(36)에서 수신되는 각 게스트 인터럽트를 기록하도록 구성될 수 있으며, 그리고 각 gAPIC들(34A-34D)에 게스트 인터럽트를 전송하도록 구성될 수 있다. gAPIC들(34A-34D)이 인터럽트를 수락하면, 게스트 인터럽트에 의해 목표되는 게스트가 활동적이 된다. 어떠한 gAPIC(34A-34D)도 인터럽트를 수락하지 않는 다면, 게스트 인터럽트에 의해 목표되는 게스트는 비활동적이 된다.

[0037] 도시된 실시예에서, 게스트 인터럽트 매니저(38)는 메모리(56) 내의 gAPIC 상태 데이터 구조(58)에서 시스템(5)에서 정의되는 게스트들에 대한 gAPIC 상태를 유지하도록 구성될 수 있다. gAPIC 상태 데이터 구조(58)는 시스템에서 정의되는 각 gAPIC에 대한 gAPIC 상태 엔트리(예를 들어, 시스템 내의 각 게스트(10A-10N)의 각 vCPU에 대한 하나의 엔트리)를 포함할 수 있다. gAPIC가 시스템 내의 활동 게스트 또는 비활동 게스트중 어느 하나와 관련되는 경우, 이러한 gAPIC가 시스템에서 정의될 수 있다. 이에 따라, 게스트 인터럽트를 수신하는 것에 응답하여, 게스트 인터럽트 매니저(38)는 그 인터럽트에 의해 목표되는 게스트/vCPU에 대해 gAPIC 상태 데이터 구조(58) 내의 gAPIC 상태를 업데이트하도록 구성될 수 있다. 일 실시예에서, 게스트 인터럽트 매니저(38)는 게스트가 활동적인지의 여부에 상관없이 gAPIC 상태를 업데이트하도록 구성될 수 있다. 1개 이상의 타겟을 갖는 멀티캐스트 및 브로드캐스트 인터럽트들에 대해, 게스트 인터럽트 매니저(38)는 gAPIC 상태 데이터 구조(58) 내의 gAPIC 상태를 업데이트하도록 구성될 수 있다. 대안적으로, 게스트 인터럽트 매니저(38)는 이러한 다수의 목적지 인터럽트들에 대해 VMM(18)에 의존하도록 구성될 수 있다. 이러한 실시예들에서, 게스트 인터럽트 매니저(38)는 VMM(18)을 액세스할 수 있는 메모리 위치 내의 인터럽트를 로그하도록 구성될 수 있으며, VMM(18)에게 메시지를 처리할 것을 신호하도록 구성될 수 있다.

[0038] 몇몇 실시예들에서, 게스트 인터럽트 매니저(38)는 게스트 ID 및/또는 게스트 인터럽트 메시지 내의 다른 정보에 바로 응답하여, gAPIC 상태 데이터 구조(58) 내의 gAPIC 상태 엔트리의 위치를 찾도록 구성될 수 있다. 다른 실시예들에서, gAPIC 상태 데이터 구조(58)에 유연성을 제공하고, 및/또는 메모리 공간을 보존하기 위해, 게스트 인터럽트 매니저(38)는 gAPIC 상태 맵핑 테이블들(60)을 이용하여, gAPIC 상태 데이터 구조(58) 내의 gAPIC 상태 엔트리의 위치를 찾도록 구성될 수 있다. gAPIC 상태 데이터 구조(58) 및 (몇몇 실시예들에 대한) 맵핑 테이블들(60)의 다양한 실시예들이 도 10 내지 13에 도시되어 있으며, 하기에서 보다 상세히 설명된다. 이에 따라, 게스트 인터럽트에 응답하여, 게스트 인터럽트 매니저(38)는 gAPIC 상태 맵핑 테이블들(60)을 이용하여 gAPIC 상태 엔트리의 위치를 찾고, 그 인터럽트를 기록하기 위해 gAPIC 상태 엔트리를 업데이트하도록 구성될 수 있다.

[0039] 일 실시예에서, gAPIC 상태 데이터 구조(58)는 gAPIC 상태의 서브세트를 저장할 수 있다. 이러한 서브세트는 하드웨어(20)(예를 들어, IOMMU(40)과 함께, 게스트 인터럽트 매니저(38))에 의해 추적되는 gAPIC 상태가 될 수 있다. 보다 특정하게, 이러한 서브세트는, 해당 게스트가 비활동적인 동안 달라질 수 있는 gAPIC 상태의 일부가 될 수 있다. 예를 들어, 일 실시예에서, 주변장치(52A-52B)는 해당 게스트가 비활동적인 동안 인터럽트를 신호할 수 있으며, 이에 의해 gAPIC가 해당 인터럽트 요청을 캡춰하게 된다. 인터럽트 요청들은 gAPIC 상태 데이터 구조(58)에서 추적될 수 있다. 다른 gAPIC 상태는 어떤 인터럽트들이 프로세서에 의해 서비스되고 있는지, 프로세서의 작업 우선순위 등을 추적할 수 있다. 이러한 값들은 게스트가 활동적일 때에만 달라질 수 있다. 일 실시예에서, 게스트가 비활동적일 때에 달라지지 않는 gAPIC 상태는, 도 2에서 VMM-관리되는 gAPIC 상태 데이터 구조(68)로서 도시된 하나 이상의 다른 데이터 구조들을 이용하여, VMM(18)에 의해 추적될 수 있다. VMM(18)은 VMM-관리되는 상태(68)와 gAPIC들(34A-34D) 간의 상태를 시스템 내에서 게스트들을 활성화하고 비활성화는 것의 일부로서 전송할 수 있다.

[0040] 도시된 실시예에서 gAPIC 상태 맵핑 테이블들(60) 및 gAPIC 상태 데이터 구조(58)는 메모리(56)에 저장되는 것으로 나타나 있지만, 이들중 하나 또는 양쪽 모두의 일부분은 게스트 인터럽트 매니저(38) 및/또는 브리지(36)를 액세스할 수 있는 캐시에 의해 캐시될 수 있다. 부가적으로 또는 대안적으로, 하나 이상의 gAPIC 상태 엔트리들에 대한 전용 메모리가 브리지(36)에서 구현될 수 있다. 이러한 전용 메모리는 gAPIC들(34A-34D) 내로 그리

고 gAPIC들(34A-34D)로부터 신속하게 스위치될 수 있는 "고속(fast)" gAPIC 상태들의 세트를 저장할 수 있다. 다른 gAPIC 상태들은 메모리(56) 내에서 보다 느리게 액세스가능하다. 몇몇 실시예들에서, 고속 gAPIC 상태 스위치들은 게스트 인터럽트 매니저(38)에 의해 핸들링될 수 있으며, 보다 느린 gAPIC 상태 스위치들은 VMM(18)에 의해 핸들링될 수 있다.

[0041] APIC 인터럽트 메커니즘에서, 각 프로세서는 (자신의 로컬 APIC를 통해) 물리적 APIC ID 및 논리적 APIC ID를 가질 수 있다. 물리적 APIC ID는 APIC ID 레지스터에 저장된다. 물리적 APIC ID는 물리적인 전달 모드 인터럽트에 의해 표시되는 물리적 APIC ID와 일대일로 매치된다. 논리적 APIC ID는 로컬 APIC 내에 논리적 목적지 레지스터로서 저장된다. 논리적 APIC ID는 클러스터 ID 및 로컬 APIC ID를 가지며, 여기서 로컬 APIC ID는 원핫 벡터(one-hot vector)이다. 논리적인 전달 모드 인터럽트들은, 클러스터 내의 하나 이상의 로컬 APIC들을 전달하기 위해, 이러한 원핫 벡터 내에 임의 세트의 비트들을 포함할 수 있다. 이에 따라, 논리적 APIC ID를 매치시키는 것은, 클러스터 ID를 비교하고, 로컬 APIC 내의 원핫 비트 벡터의 세트 비트와 동일한 위치에서 로컬 APIC ID 벡터 내의 세트 비트를 검출하는 것을 포함할 수 있다. 다른 측면에서 보면, 논리적 전달 모드 인터럽트의 로컬 APIC ID 벡터는 로컬 APIC의 로컬 APIC ID 벡터와 논리적으로 AND될 수 있으며, 그리고 결과가 넌제로이고, 클러스터 ID가 매치하면, 로컬 APIC는 논리적 인터럽트의 타겟이 된다. 논리적 APIC ID는 여기에서 논리적 ID로서 보다 간결하게 지칭될 수 있고, 유사하게 물리적 APIC ID는 여기에서 물리적 ID로서 보다 간결하게 지칭될 수 있다. 인터럽트와 관련된 소정의 (논리적 또는 물리적) ID는 인터럽트의 목적지 ID로서 지칭될 수도 있다. 인터럽트에 대한 해당하는 전달 모드는 그 인터럽트의 목적지 ID를 식별할 수 있다.

[0042] 또한, gAPIC들(34A-34D)은 물리적 및 논리적 전달 모드들을 모두 지원할 수 있다. 상기 강조한 바와 같이 모드에 따라 인터럽트 메시지 내의 APIC ID를 매칭하는 것에 부가하여, gAPIC들(34A-34D)은 gAPIC 내의 게스트 ID에 대해 인터럽트 메시지 내의 게스트 ID를 매치시킬 수 있다.

[0043] IOMMU(40)는 I/O-개시되는 메모리 동작들(예를 들어, 주변장치들(52A-52B)로부터, 또는 주변장치들(52A-52B) 대신 DMA 제어기들에 의해 비롯되는 메모리 판독/기록 동작들)에 대해 가상 대 물리 어드레스 맵핑을 수행하도록 구성될 수 있다. 변환 동작의 일부로서, IOMMU(40)는 장치 테이블(62) 및 선택적으로 인터럽트 리다이렉트 테이블(64)을 액세스하도록 구성될 수 있다. 장치 테이블(62)은 각 주변장치(52A-52B)에 대한 엔트리들을 포함할 수 있으며, (그리고 주변장치들이 결합되는 주변장치 인터페이스 상에서 1개 이상의 식별자를 포함하는 주변장치에 대한 다수의 엔트리들을 포함할 수도 있다). 장치 테이블(62)은 메모리 판독/기록 동작들(미도시)의 메모리 어드레스들을 변환하기 위한 I/O 페이지 테이블들에 대한 페이지 테이블 포인터를 포함할 수 있으며, 인터럽트 리다이렉트 테이블(64)에 대한 포인터를 포함할 수도 있다. 몇몇 실시예들에서, 장치 테이블(62)은 게스트에 할당되는 주변장치들에 대한 게스트 ID를 저장할 수 있다. 일 실시예에서, 이러한 게스트 ID는 IOMMU(40)에서 장치 액세스 보호를 위해 이용되는 도메인 ID와 같을 수 있다. 대안적으로, 이러한 게스트 ID는 개별적으로 할당될 수도 있다. 일 실시예에서, 장치 테이블(62)은 또한 (이용되는 경우) gAPIC 상태 맵핑 테이브들(60)에 대한 포인터, 또는 gAPIC 상태 데이터 구조(58)에 대한 포인터를 저장할 수 있다. 다른 실시예에서는, 게스트 ID 및/또는 테이블(60)/데이터 구조(58)에 대한 포인터가 인터럽트 리다이렉트 테이블(64)에 저장될 수 있다. 이러한 인터럽트 리다이렉트 테이블(64)은 자신의 최초 목적지로부터의 인터럽트 및/또는 인터럽트 벡터를 새로운 목적지 및/또는 인터럽트 벡터로 리다이렉트시키는 데에 이용될 수 있다. 본 개시의 나머지 부분에서의 간단함을 위해, 게스트 ID가 장치 테이블(62)로부터의 도메인 ID 이고, 맵핑 테이블들(60) 및/또는 gAPIC 상태 데이터 구조(58)에 대한 포인터가 장치 테이블(62)에 저장되는 실시예가 이용될 것이다. 하지만, 본 개시의 나머지 부분에서의 실시예들은 일반적으로 상기 설명한 바와 같이 변경될 수 있다.

[0044] 다른 실시예들에서는, 게스트 인터럽트 매니저(38)가 제공되지 않을 수도 있다. 이러한 구성은, 예를 들어 게스트들이 하나의 프로세서(30A-30B)로부터 다른 프로세서로 이동할 때, VMM(18)이 장치 테이블(62) 및/또는 인터럽트 리다이렉트 테이블(64)을 업데이트하는 경우, 그리고 (메모리(56) 내의 gAPIC 상태 데이터 구조(58)를 업데이트하고, 및/또는 요구되는 경우, 인터럽트를 서비스하기 위해) 프로세서(30A-30B)가 비활동 게스트들을 대신하여 인터럽트들을 수신하는 데에 전용되는 경우에 가능하다.

[0045] 메모리 제어기(42)는 프로세서들(30A-30B)에 의해 발행되는 메모리 동작들(예를 들어, 명령 페치들, 적재/저장 데이터 액세스들, 변환을 위한 프로세서 페이지 테이블 액세스들 등), (예를 들어, gAPIC 상태 데이터 구조(58) 및/또는 gAPIC 상태 맵핑 테이블들(60)을 판독/업데이트하기 위해) 게스트 인터럽트 매니저(38), (예를 들어, I/O 페이지 테이블들, 장치 테이블(62) 및 인터럽트 리다이렉트 테이블(64)을 액세스하기 위해) IOMMU(40)로부터의 메모리 동작들, 및 (일부 실시예들에서는) 인터페이스 회로들(44A-44C)로부터 수신되는 메모리 동작들을 수신하도록 결합될 수 있다. 메모리 제어기(42)는 메모리 동작들을 배열하고, 이러한 메모리 동작들을 수행하기

위해 메모리(56)와 통신하도록 구성될 수 있다. 메모리 인터페이스 회로(46)는 메모리(56)에 대한 물리 레벨 액세스들을 수행할 수 있다.

[0046] 메모리(56)는 임의 타입의 메모리를 포함할 수 있다. 예를 들어, 메모리(56)는 동기 DRAM (SDRAM), 더블 데이터 레이트 (DDR, DDR2, DDR3 등) SDRAM, RAMBUS DRAM 과 같은 동적 랜덤 액세스 메모리(DRAM), 정적 RAM 등을 포함할 수 있다. 메모리(56)는, SIMMs(singel inline memory modules), DIMMs(dual inline memory modules) 등과 같은, 다수의 메모리 칩들을 포함하는 하나 이상의 메모리 모듈을 포함할 수 있다.

[0047] 본 실시예에서, 게스트 인터럽트 매니저(38), IOMMU(40) 및 메모리 제어기(42)를 포함하는 것에 부가하여, 브리지(36)는 또한 프로세서들(30A-30B), hAPIC들(32A-32B), gAPIC들(34A-34B), 및 인터페이스 회로들(44A-44B)에 결합된 장치들 간에 통신하기 위한 다른 통신 기능을 포함할 수 있다. 예를 들어, 도시된 실시예에서는, 다른 브리지(48)가 인터페이스 회로(44A)에 결합될 수 있으며, 인터페이스 회로(44A)에 의해 이용되는 프로토콜들과 주변장치들(52A-52B)에 의해 이용되는 프로토콜들 간의 통신들을 브리지하도록 구성될 수 있다. 일 실시예에서, 인터페이스 회로들(44A-44C)은, 예를 들어 상기 설명한 바와 같은 HT 인터페이스를 구현할 수 있으며, 그리고 브리지(48)는 HT로부터 PCIe(PCI Express) 인터페이스와 같은 다른 인터페이스로 브리지할 수 있다. 이러한 실시예에서, 주변장치들(52A-52B)은 PCIe 장치들이 될 수 있다. 브리지(48)는 또한 다른 인터페이스들로 브리지하도록 구성되거나, 또는 다른 브리지가 브리지(48)에 결합되어, 다른 인터페이스들로 브리지할 수 있다. 임의의 주변 인터페이스 또는 인터페이스들이 이용될 수 있다. 또한, 주변장치들(52A-52B)은 HT 인터페이스에 직접 결합되도록 구성되는 HT 주변장치들이 될 수 있다. 이러한 주변장치들은 브리지(48)를 요구하지 않을 수도 있다.

[0048] 일 실시예에서, 브리지(48) 및/또는 주변장치들(52A-52B)중 하나 이상은 IOAPIC들(도 2의 50 및 54)을 포함할 수도 있다. 이러한 IOAPIC들은 주변장치들로부터 인터럽트 요청들을 수신하고, (gAPIC들(34A-34D)로의 전송 및/또는 메모리 내의 gAPIC 상태 데이터 구조(58)에서의 기록을 위해) 이러한 인터럽트 요청들을 hAPIC들(32A-32B) 및 게스트 인터럽트 매니저(38)에 전송하기 위해 인터럽트 메시지들을 형성하는 것을 담당한다.

[0049] 상기 설명한 바와 같이, 일 실시예에서, 인터페이스 회로들(44A-44C)은 HT 인터페이스 상에서 통신하도록 구성될 수 있다. 인터페이스 회로들(44A-44C)은 HT를 이용하여 주변 장치들/브리지들과 통신하도록 구성될 수 있다. 부가적으로, 몇몇 실시예들에서, 인터페이스 회로들(44A-44C)은 프로세서들, hAPIC들, gAPIC들 등을 갖는 다른 노드들에 결합되도록 구성될 수 있다. 이러한 실시예들에서, 브리지(36)는 이전에 설명한 회로 이외에 코히런스 관리 회로를 포함할 수 있다.

[0050] 프로세서들(30A-30B)은 임의의 명령 세트 아키텍쳐를 구현할 수 있으며, 이러한 명령 세트 아키텍쳐에서 정의되는 명령들을 실행하도록 구성될 수 있다. 프로세서들(30A-30B)은 슈퍼파이프라인드, 슈퍼스칼라, 및/또는 그 결합들과 같은 임의의 마이크로 아키텍쳐; 순서에 따른(in-order) 또는 순서를 벗어난(out-of-order) 실행; 추론적 실행(speculative execution) 등을 포함할 수 있다. 프로세서들(30A-30B)은 요구에 따라 마이크로코딩 기술들을 실시할 수도 있고, 실시하지 않을 수도 있다.

[0051] 주변장치들(52A-52B)은 임의의 타입의 주변 장치를 포함할 수 있다. 주변장치들(52A-52B)은 자기, 고체, 또는 광 디스크 드라이브들, 비휘발성 메모리 디바이스들(플래시 메모리 등)과 같은 저장 디바이스들을 포함할 수 있다. 주변장치들(52A-52B)은 사용자 I/O 장치들(키보드, 마우스, 디스플레이, 음성 입력 등)과 같은 I/O 장치들, 네트워킹 장치들, 외부 인터페이스 장치들(범용 직렬 버스(USB) 또는 파이어와이어(firewire)) 등을 포함할 수 있다.

[0052] 도시된 실시예에서, 프로세서들(30A-30B), 브리지(36), hAPIC들(32A-32B), gAPICS(34A-34D), 인터페이스 회로들(44A-44C), 및 메모리 인터페이스 회로(46)은 집적 회로(66)로서 단일 반도체 기판 상에 집적될 수 있다. 다른 실시예들은 요구에 따라 다른 양의 집적 및 개별적인 회로를 구현할 수 있다. 주목할 사항으로서, 도 2에는 프로세서들, hAPIC들, gAPIC들, 인터페이스 회로들, 주변장치들, 브리지 등과 같은 많은 수의 컴포넌트들을 도시하였지만, 다른 실시예들은 요구에 따라 하나 이상의 각 컴포넌트의 임의 수를 구현할 수 있다.

[0053] 다른 실시예들에서, IOMMU(40) 및 게스트 인터럽트 매니저(38)의 위치는 달라질 수 있다. 예를 들어, 하나 또는 양쪽 모두가 브리지에 있거나, 주변장치들(52A-52B)에 있거나, 또는 브리지에 결합된 다른 브리지 등에 있을 수 있다.

[0054] 도시된 실시예에서, 도 2에 도시한 바와 같이, 각 gAPIC들(34A-34D) 및 hAPIC(32A-32B)은 특정의 프로세서(30A-30B)와 관련된다. 따라서, 이러한 실시예에서, 소정의 인터럽트 제어기는 해당 프로세서(30A-30B)에 전용된다. 보다 특정하게는, 도 2에서, hAPIC(32A) 및 gAPIC들(34A 및 34C)은 프로세서(30A)에 전용되고,

hAPIC(32B) 및 gAPIC들(34B 및 34D)은 프로세서(30B)에 전용된다. 인터럽트 제어기는 임의의 방식으로 해당 프로세서에 인터럽트를 신호할 수 있다. 일반적으로, 이러한 시그널링은 인터럽트가 필요함을 나타낼 수 있다. 이러한 시그널링이 인터럽트 벡터를 포함하거나, 또는 인터럽트가 전달된 후 실행되는 소프트웨어에 의해 이러한 인터럽트 벡터가 판독될 수 있다. 일 실시예에서, 인터럽트를 전달하는 것은, 프로세서에게 시그널링하고, 프로세서가 인터럽트를 수락하는 것을 말한다. 인터럽트를 서비스하는 것은, 인터럽팅 장치에 의해 요구되는 동작들을 수행하기 위해 인터럽트 벡터와 관련된 인터럽트 서비스 루틴을 실행하는 것을 말한다.

[0055] 이제, 도 3을 참조하면, 일 실시예에 대한, 주변장치로부터 gAPIC로의 인터럽트의 진행을 도시하는 블록도가 나타나있다. 다른 프로세서들로부터의 인터럽트들(프로세서간 인터럽트들, 또는 IPI들)은 게스트 인터럽트 매니저(38)에도 전송될 수 있으며, 도 3과 유사하게 전방으로 그 지점으로부터 핸들링될 수 있다. 대안적으로, IPI를 개시하는 프로세서로부터 IPI를 수신하는 gAPIC는 (수신 게스트가 비활동적인 경우, 이 수신 게스트에 대한 gAPIC 상태를 업데이트하기 위해) 게스트 인터럽트 매니저(38)에게 업데이트를 전송할 수 있으며, 또한 다른 gAPIC들에게 (게스트 ID를 포함하는) IPI를 전송할 수 있다.

[0056] 도시된 실시예에서, 주변장치(52A)는 인터럽트가 요구됨을 결정한다. 주변장치(52A) 내의 IOAPIC(54)(도 2 참조)는 주변장치(52A)의 동작에 응답하여 인터럽트 메시지를 발생시킬 수 있다. 구체적으로, IOAPIC(54)는 (예를 들어, 주변장치(52A)에 의해 요구되는 서비스, 주변장치(52A)가 다수의 기능들을 구현하는 경우 인터럽트를 시그널링하는 특정 기능 등에 기초하여), 요구되는 인터럽트에 해당하는 인터럽트 벡터를 발생시킬 수 있다. 인터럽트 벡터는 인터럽트 통신의 일부가 될 수 있으며, 그리고 인터럽트 소스들을 식별하고, 인터럽트들의 우선순위를 매기는 등을 행하기 위해 소프트웨어에 의해 이용될 수 있다. 몇몇 경우들에서, 인터럽트 벡터는 IOMMU(40)에 의해 리맵핑될 수 있으며, 이에 따라 이러한 인터럽트 벡터는 도 3에서 "최초 벡터"로서 도시된다. 주변장치(52A)는 IOMMU(40)에 인터럽트 메시지를 전송할 수 있다(화살표 A). 이러한 실시예에서, 인터럽트는, 예를 들어 PCIe 사양에서 정의되는 바와 같이, MSI(message-signalled interrupt)의 형태로 전송될 수 있다. 다른 실시예들은 임의의 원하는 방식으로 인터럽트를 전송할 수 있다. 일반적으로, 이러한 전송은 인터럽트, 그 전달 모드(예를 들어, 논리적 또는 물리적), 및 인터럽트의 목적지 ID (DestID)를 식별할 수 있다.

[0057] IOMMU(40)는 MSI를 수신할 수 있다. 이 MSI는 주변장치의 식별자를 포함한다. 예를 들어, PCI 프로그래밍 모델을 구현하는 인터페이스들은 버스 번호 및 이 버스 상에서의 디바이스 번호에 의해 각 디바이스를 식별할 수 있으며, (이에 의해, 다수의 PCI 인터페이스들이 계층적 및/또는 병렬 방식으로 시스템 내에 존재할 수 있게 된다). 장치들은 다수의 "기능들"을 가질 수 있는데, 이들은 물리적인 장치 상에서의 개별적인 가상 장치들, 또는 그 장치 상에서의 동작들의 분할이 될 수 있다. 식별자는 또한 기능 번호를 포함할 수 있다. 따라서, 이러한 실시예에서, 식별자는 BDF(Bus-Device-Function)로서 지칭될 수 있다. IOMMU(40)는 BDF를 이용하여 장치 테이블(62) 내에 인덱스될 수 있으며(화살표 B), 그리고 주변장치(52A)에 해당하는 장치 테이블 엔트리를 식별할 수 있다. 몇몇 실시예들에서, 이러한 엔트리는 게스트 ID, 및 gAPIC 상태 맵핑 테이블들(60) 또는 gAPIC 상태 데이터 구조(58)에 대한 포인터를 포함할 수 있다(화살표 C). 이러한 실시예에서, 장치 테이브 엔트리는 또한 인터럽트 리다이렉트 테이블 포인터(IRTP)를 포함할 수 있는데, 이는 장치에 해당하는 인터럽트 리다이렉트 테이블(64)을 식별할 수 있다(화살표 C1). 인터럽트 리다이렉트 테이블(64)은 최초 인터럽트 벡터에 의해 인덱스될 수 있으며, 그리고 인터럽트에 대한 출력 벡터 및 목적지 ID (DestID, 예를 들어 논리적 또는 물리적 APIC ID)를 제공할 수 있다.

[0058] 도 3은 MSI가 벡터(42), 게스트 ID(99)에 리맵핑되는 예를 도시한다. 이러한리맵핑은 게스트 ID를 부가하는 것을 포함할 수 있으며, 또한 인터럽트 리다이렉트 테이블(64)이 이용되는 경우, 벡터가 변경될 수 있다. 그렇지 않으면, MSI로부터의 최초 인터럽트 벡터가 인터럽트 메시지 내에 제공된다. 인터럽트 벡터(42) 및 게스트 ID(99)의 특정 예가 이용되는 도 3에서의 포인트들은 [] 로서 도시된다.

[0059] IOMMU(40)는 게스트 ID(본 예에서는, 99)를 포함하는 인터럽트 메시지를 게스트 인터럽트 매니저(38)에 전송할 수 있다. 이러한 인터럽트 메시지는 또한 인터럽트 벡터(예를 들어, 본 예에서는 42)를 포함한다. 이러한 인터럽트 메시지는 또한 gAPIC 상태 맵핑 테이블들(60) 또는 gAPIC 상태 데이터 구조(58)에 대한 포인터를 포함할 수 있다(화살표 D).

[0060] gAPIC 상태 맵핑 테이블들(60)을 구현하는 실시예들에서, 게스트 인터럽트 매니저(38)는 포인터와, 그리고 가능하게는 게스트 ID 및/또는 목적지 ID와 같은 다른 정보를 이용하여, gAPIC 상태 맵핑 테이블들(60) 내에서 gAPIC 상태 포인터의 위치를 찾으며(화살표 E1, 그리고 화살표 E2로 나타낸 바와 같이, 게스트 인터럽트 매니저(38)에 대한 복귀 포인터(returning pointer)가 나타나있다). gAPIC 상태 포인터는 gAPIC 상태 데이터 구조

(58) 내에서 gAPIC 상태 엔트리를 식별할 수 있으며, 그리고 게스트 인터럽트 매니저(38)는 이러한 gAPIC 상태 엔트리를 이용하여, gAPIC 상태 데이터 구조(58) 내에서 gAPIC 상태 업데이트를 수행할 수 있다(화살표 E). 이러한 예에서, gAPIC 상태 업데이트는 벡터(42)에 해당하는 인터럽트 요청 레지스터 내에 비트를 설정할 수 있다. 인터럽트 요청 레지스터(IRR)는 도 4와 관련하여 하기에서 보다 상세히 설명된다.

[0061] 일 실시예에서, gAPIC 상태(58)에 대한 업데이트는 원자적(atomic)이 될 수 있다. 예를 들어, 게스트 인터럽트 매니저(38)는 원자 OR 트랜잭션(atomic OR transaction)을 발생시킬 수 있는데, 이러한 원자 OR 트랜잭션은 세트되는 인터럽트 요청 비트를 gAPIC 상태 엔트리 내의 인터럽트 요청 레지스터의 현재 상태로 원자적으로 OR시킨다. 원자적 동작은, 동작이 다수의 단계들로서 구현된다고 할지라도, 하나의 단위(unit)로서 효과적으로 수행되는 동작이 될 수 있다. 원자적으로 업데이트되는 위치를 액세스하고자 하는 관찰자는 원자적 업데이트 이전에, 또는 원자적 업데이트 이후에 값을 수신하지만, 중간 값을 수신하지 못할 수도 있다. 원자적으로 업데이트되는 위치를 업데이트하고자 하는 관찰자는 원자 동작 이전에, 또는 원자 동작이 완료된 후에 (하지만, 원자 동작 동안에는 아니다), 자신의 업데이트를 수행한다. 비록 본 실시예가 원자 OR을 구현하기는 하지만, 다른 실시예들은 보다 일반적인 원자 업데이트 동작을 구현할 수 있다. 예를 들어, 원자 업데이트는 변경되지 않아야 하는 타겟의 비트들을 식별하는 AND 마스크 및 어떤 비트들이 OR될 것인 지를 식별하는 OR 마스크를 포함할 수 있다. 다른 구현들이 또한 가능하다. 예를 들어, 비교(compare) 및 스와프(swap) 구현이 이용될 수 있는데, 여기에서는 메모리 위치로부터 최초 값이 판독되고, OR되는 새로운 값과 최초 값에 대해 비교 및 스와프 동작이 수행된다. 만일 비교가 실패하면, 프로세스가 반복될 수 있다(새로운 최초 값을 읽고, 비교 및 스와프를 수행한다). 요구에 따라, 백오프(backoff) 및/또는 타임아웃(timeout) 메커니즘들을 이용하여, 루프를 끊어지게 할 수 있다.

[0062] 또한, 게스트 인터럽트 매니저(38)는, 인터럽트 벡터, 게스트 ID 및 목적지 ID를 포함하는 인터럽트 메시지를 gAPIC들(34A-34D)에 방송할 수 있다(화살표 F). gAPIC들중 하나(도 3에서 gAPIC(34A))는 게스트 ID 99, 및 목적지 ID와 매치하는 논리적 또는 물리적 APIC ID를 가질 수 있으며, 이에 따라 gAPIC(34A)가 그 인터럽트 메시지에 응답하는 바, 자신이 인터럽트 메시지를 수신했음을 나타내는 수신 확인(Ack)을 한다(화살표 G). 또한, 본 예에서, gAPIC(34A)는 자신의 인터럽트 요청 레지스터를 업데이트하여, 벡터(42)에 해당하는 비트를 설정할 수 있다. 만일 인터럽트가 임의의 진행중인 인터럽트(만약 있다면) 및/또는 프로세서의 작업 우선순위 보다 더 높은 우선순위를 갖는 다면, gAPIC(34A)는 또한 프로세서(34A)에 인터럽트를 신호할 수 있다. 다른 gAPIC들(34B-34D)은 방송 인터럽트 메시지에 응답할 수 있지만, 수락했다는 수신 확인을 하지 않을 수도 있는데, 왜냐하면 이들은 인터럽트의 타겟이 아니기 때문이다(화살표 H). 논리적 인터럽트들에 있어서, 논리적 인터럽트가 다수의 타겟들을 식별한다면, 1개 이상의 수신확인을 할 수 있다.

[0063] 상기 메커니즘을 이용하게 되면, 게스트 인터럽트 매니저(38)는 어느 gAPIC(34A-34D)가 어느 게스트에 할당되는 지를 "인식"할 필요가 없다. 어느 gAPIC(34A-34D)가 어느 게스트에 할당되는 지를 게스트 인터럽트 매니저(38) 추정하고, 목표로 되는 gAPIC들에만 인터럽트를 전송하는 다른 실시예들이 고려될 수 있다. 게스트 인터럽트 매니저(38)는 gAPIC들을 자동으로 추적하거나, 또는 aAPIC가 다른 게스트에게 재할당될 때 마다 VMM(18)에 의해 프로그램될 수 있다. 이러한 실시예에서, 게스트 인터럽트 매니저(38)는 목표로 되는 gAPIC들에만 인터럽트 메시지를 전송할 수 있다.

[0064] hAPIC들(32A-32B)로의 인터럽트의 전송은 통상의 APIC 방식으로 수행될 수 있다. 구체적으로, 일 실시예에서, 인터럽트는 게스트 인터럽트 매니저(38)에 의해 동작되지 않을 수 있지만, 다른 면들에서는 도 3의 동작과 유사하다.

[0065] 주목할 사항으로서, 비록 여기에서는 게스트 인터럽트 매니저(38)가 블록으로서 도시되었지만, 이러한 게스트 인터럽트 매니저(38)를 구현하는 회로는 분배될 수 있다. 예를 들어, 일 실시예에서, 포인터를 수신하고, gAPIC 상태 맵핑 테이블(60)을 선택적으로 처리하고, gAPIC 상태 데이터 구조(58)에 대한 업데이트를 발생시키는 게스트 인터럽트 매니저(38)의 일부가 IOMMU(40)에 포함될 수 있으며, 이에 따라 IOMMU(40)는 gAPIC들(34A-34D)에 전송될 인터럽트 메시지 및 gAPIC 상태 데이터 구조(58)에 대한 원자 OR을 전송한다. 하나 이상의 물리적 위치들에서의 게스트 인터럽트 매니저(38)의 임의의 구현이 이용될 수 있다.

[0066] 이제, 도 4를 참조하면, gAPIC(34A)이 일 실시예가 도시되어 있다. 다른 gAPIC들(34B-34D)도 유사하다. 도 4의 실시예에서, gAPIC(34A)는 인터럽트 요청 레지스터(IRR)(70), 인터럽트 서비스 레지스터(ISR)(72), 트리거 모드 레지스터(TMR)(74), 작업 우선순위 레지스터(TPR)(76), 제어 유닛(78), 물리적 ID 레지스터(80), 논리적 ID 레지스터(82), 게스트 ID 레지스터(84), 및 선택적으로 기타 APIC 상태(86)를 포함한다. 제어 유닛(78)은

IRR(70), ISR(72), TMR(74), TPR(76), 물리적 ID 레지스터(80), 논리적 ID 레지스터(82), 게스트 ID 레지스터(84) 및 기타 APIC 상태(86)에 결합된다. 또한, 제어 유닛(78)은 인터럽트들을 수신하기 위해 게스트 인터럽트 매니저(38)와 통신하도록 결합되고, 프로세서 인터페이스에 결합되어, 프로세서(30A)와 통신한다.

[0067] 게스트 인터럽트 매니저(38)로부터 인터럽트 메시지를 수신하는 것에 응답하여, 인터럽트가 gAPIC(34A)에 해당하는 게스트를 목표로 하고 있다면, 제어 유닛(78)은 IRR(70)에 인터럽트를 기록하도록 구성될 수 있다. IRR 내의 인터럽트 요청의 위치는 인터럽트 벡터에 해당한다. IRR은 "고정된(fixed)" 인터럽트들을 추적할 수 있다. 다른 인터럽트 타입들은 NMI(non-maskable interrupt), SMI(system management interrupt), extINT(legacy external interrupt) 등을 포함할 수 있다. 이러한 인터럽트들은 기타 APIC 상태(86)의 일부로서 핸들링될 수 있다.

[0068] 일 실시예에서, 인터럽트 메시지는 또한 각 인터럽트(레벨 또는 에지)에 대한 트리거 모드를 포함할 수 있다. TMR(74)은 어떤 트리거 모드가 인터럽트에 적용되는 지의 표시를 저장할 수 있다. 예를 들어, 에지 트리거되는 인터럽트들(edge triggered interrupts)은 TMR(74)에서 이진 0에 의해 표현될 수 있고, 트리거되는 레벨은 이진 1에 의해 표현될 수 있다. 다른 실시예들에서는, 에지 트리거되는 인터럽트들 만이 gAPIC(34A)에서 지원될 수 있으며, TMR(74) (및 gAPIC 상태 데이터 구조(58) 내의 그 카피)는 제거될 수 있다. 다른 실시예에서, TMR(74)은 VMM(18)이 가상 레벨 감지 인터럽트들을 로그할 수 있도록 다른 용도로 될 수 있다. VMM(18)은 TMR(74) 내에 다양한 비트들을 설정하여, 해당하는 인터럽트 벡터에 대해 프로세서(30A)에 의해 인터럽트의 끝이 시그널링되면, 프로세서(30A)는 VMM(18)으로 빠져나감을 나타낼 수 있다.

[0069] 고정된 인터럽트들에 대해, gAPIC(34A)는 인터럽트 요청들 및 서비스되는 인터럽트들의 우선순위를 정하여, 인터럽트 요청이 프로세서에 전달되어야 하는 지를 결정하도록 구성될 수 있다. 일반적으로, 최고 우선순위의 인터럽트 요청이 최고 우선순위의 서비스되는 인터럽트 보다 더 높은 우선순위를 갖는 다면(인터럽트에 해당하는 인터럽트 핸들러를 실행하기 위해 프로세서가 자신의 소프트웨어 실행을 인터럽트하는 경우, 인터럽트가 서비스된다), 제어 유닛(78)은 요청되는 인터럽트를 프로세서(30A)에 전달하도록 구성될 수 있다. 또한, TPR(76)은 프로세서(30A)에 의해 수락되고 있는 인터럽트의 최소 우선순위 레벨을 확립하도록 소프트웨어에 의해 프로그램될 수 있다. 제어 유닛(78)은 가장 높은 우선순위의 인터럽트 요청을 전달하도록 구성될 수 있는데, 이는 이러한 가장 높은 우선순위의 인터럽트 요청이 가장 높은 우선순위의 서비스되고 있는 인터럽트 보다 높은 우선순위를 가지는 경우, 그리고 이러한 가장 높은 우선순위의 인터럽트 요청이 TPR(76)에 표시되는 우선순위 보다 높은 경우에 그러하다.

[0070] 프로세서(30A)가 인터럽트를 받으면, 그 프로세서는 gAPIC(34A)에 대한 인터럽트 수신확인 커맨드(interrupt acknowledge command)로 응답할 수 있다. 제어 유닛(78)은 IRR(70)로부터 가장 높은 우선순위 인터럽트 요청을 제거하고, ISR(72) 내에 그 인터럽트를 서비스되는 것으로서 로그한다. ISR 내의 인터럽트에 해당하는 인서비스 표시(in-service indication)의 위치는 그 인터럽트의 인터럽트 벡터에 해당한다. 프로세서(30A)는 인터럽트를 서비스하기 위해 인터럽트 서비스 루틴(또는 루틴들)을 실행시킬 수 있다. 이러한 인터럽트 서비스 루틴은 gAPIC(34A)에 대한 EOI(end of interrupt) 커맨드에 의해 종료됨으로써, 인터럽트 서비스가 완료되었음을 신호할 수 있다. 이러한 EOI 커맨드에 응답하여, 제어 유닛(78)은 ISR(72)로부터 가장 높은 우선순위의 서비스되는 인터럽트를 제거하도록 구성될 수 있다.

[0071] IRR(70), ISR(72) 및 TMR(74) 각각은 gAPIC(34A)에 의해 지원되는 각 인터럽트 벡터에 해당하는 위치를 포함한다. 도시된 실시예에서는, 벡터 0 내지 255가 지원된다. 인터럽트 벡터 번호는 또한, 다른 인터럽트들에 대한 자신의 상대적인 우선순위를 나타낼 수 있다(예를 들어, 더 높은 벡터 번호들은 더 낮은 벡터 번호들 보다 더 높은 우선순위를 가지며, 다른 실시예들에서는, 그 반대의 경우도 가능하다). 각 인터럽트 벡터에 대해, IRR(70)는 인터럽트가 그 인터럽트 벡터에서 요청되는 지를 표시하는 인터럽트 요청 비트를 저장한다. 예를 들어, 이러한 표시는, 세트되면 요청을 나타내고, 클리어(clear)되면 어떠한 요청도 없음을 나타내는 비트가 될 수 있다. 유사하게, 각 인터럽트 벡터에 대해, ISR(72)은 그 인터럽트 벡터에 대해 인터럽트가 서비스되는 지를 표시하는 인서비스 비트(in-service bit)를 저장할 수 있다(예를 들어, 세트될 때에는 인서비스 인터럽트를 나타내고, 클리어될 때에는 어떠한 인서비스 인터럽트도 없음을 나타낸다). 각 인터럽트 벡터에 대해, TMR(74)은 트리거 모드를 저장한다. IRR(70), ISR(72) 및 TMR(74) 각각에 대해, 레지스터 내의 비트 위치는 인터럽트에 해당하는 인터럽트 벡터 번호와 같다.

[0072] 도시된 실시예에서, 계류중인 인터럽트 요청이 프로세서에 전달될 것인 지를 결정하기 위해, 인터럽트들은 우선순위 레벨들이 할당되는 그룹들로 논리적으로 배열된다. 예를 들어, 인터럽트 벡터들 0 내지 15에는 우선순위

레벨 0이 할당되고, 인터럽트 벡터들 16 내지 31에는 우선순위 레벨 1이 할당되는 방식으로, 인터럽트 벡터들 244 내지 255에는 우선순위 레벨 15가 할당된다. 이러한 실시예에서, 우선순위 레벨 번호들이 증가하는 것은 더 높은 우선순위 레벨을 나타낸다. 제어 유닛(78)은 요청 우선순위 레벨을 계산하는데, 이는 IRR(70) 내에서 적어도 하나의 인터럽트 요청이 계류중인 가장 높은 우선순위 레벨이다. 제어 유닛(78)은 또한 인서비스 우선순위 레벨을 계산할 수 있는데, 이는 ISR(72) 내에서 적어도 하나의 인터럽트가 인서비스로서 표시되는 가장 높은 우선순위 레벨이다. 제어 유닛(78)은, 요청 우선순위 레벨이 인서비스 우선순위 레벨을 넘고, TPR(76) 내에 표시된 우선순위 레벨을 또한 넘을 경우, 인터럽트를 전달할 수 있다. 주목할 사항으로서, 도시된 실시예에서는 256개의 인터럽트 벡터들이 16개의 우선순위 레벨 그룹들로 지원되지만, 다른 실시예들에서는, 이 보다 더 많거나 적은 인터럽트 벡터들 및/또는 더 많거나 적은 우선순위 레벨 그룹들이 지원될 수 있다.

[0073] 물리적 ID 레지스터(80) 및 논리적 ID 레지스터(82)는 gAPIC(34A)에 할당된 물리적 APIC ID 및 논리적 APIC ID를 각각 저장할 수 있다. 게스트 ID 레지스터(84)는 gAPIC(34A)에 할당된 게스트 ID를 저장할 수 있다. 이에 따라, 제어 유닛(78)은 게스트 인터럽트 매니저(38)로부터 인터럽트를 수락하도록 구성될 수 있는데, 이는 그 인터럽트의 게스트 ID가 게스트 ID 레지스터(84) 내의 게스트 ID와 일치하며, 그리고 인터럽트가 물리적이고 그 인터럽트의 APIC ID가 물리적 ID 레지스터(80) 내의 물리적 ID와 일치하거나, 또는 인터럽트가 논리적이고 그 인터럽트의 APIC ID가 논리적 ID 레지스터(82) 내의 논리적 ID와 일치하는 경우에 이루어진다.

[0074] 다른 APIC 상태(86)는 내부적으로 발생된 인터럽트들, 타이머들, 및 로컬 벡터 테이블 등을 포함할 수 있다. 다양한 실시예들에서, 이러한 다른 APIC 상태(86)중 일부 또는 전부는 gAPIC(34A)에 포함되거나, 또는 VMM(18)에 대한 인터셉트 및 그 상태의 VMM(18) 에뮬레이션을 가지며 가상화될 수 있다.

[0075] hAPIC들(32A-32B)은, 이들이 게스트 ID 레지스터를 포함하지 않는 것을 제외하고, gAPIC(34A)와 유사하다. 대안적으로, hAPIC들(32A-32B) 및 gAPIC들(34A-34D)은 (gAPIC들(34A-34D)이 모든 APIC 상태를 구현하는 경우) 동일한 하드웨어의 인스턴스들(instances)이 될 수 있으며, 게스트 ID 레지스터는 게스트 ID가 유효한지를 나타내는 인에이블 비트(enable bit)를 포함하거나, 또는 게스트 ID 레지스터는 hAPIC를 나타내도록 0으로 프로그램될 수 있다.

[0076] 이제, 도 5를 참조하면, gAPIC 상태 엔트리(90)의 일 실시예 및 VMM-관리되는 gAPIC 상태 엔트리(92)의 일 실시예의 블록도가 도시되어 있다. 도 5의 도시는 상태의 논리도가 될 수 있다. 몇몇 실시예들에 대해, 메모리 내에서의 상태의 실제 배열은 도 9, 12 또는 13에 도시한 바와 같이 달라질 수 있다.

[0077] 일반적으로, gAPIC 상태 엔트리(90)는 적어도 gAPIC 상태를 포함할 수 있는데, 이러한 gAPIC 상태는 그 gAPIC 상태에 해당하는 게스트가 활동적이지 않는 동안 변경될 수 있다. 본 실싱에서, 주변 장치는 IRR 상태를 변경했을 수도 있는 게스트에게 인터럽트를 신호할 수 있다. 하지만, ISR 상태는 게스트의 vCPU가 인터럽트를 수락할 때에만 변경될 수 있으며, 이는 게스트가 활동적이지 않을 때에는 일어나지 않는다. 유사하게, TPR은 vCPU에 의해 변경되며, 이에 따라 게스트가 활동적이지 않은 동안에는 변경되지 않는다. VMM(18)은 VMM-관리되는 gAPIC 상태 엔트리(92) 내에서의 이러한 상태의 복구 및 세이브를 관리할 수 있다.

[0078] 이에 따라, 도 4와 유사한 gAPIC(34A)의 실시예에 있어서, gAPIC 상태 엔트리(90)는 IRR(70)의 상태를 포함할 수 있다. VMM-관리되는 gAPIC 상태 엔트리(92)는 ISR(72), TMR(74) 및 TPR(76)의 상태, 및 다양한 다른 APIC 상태(86)를 포함할 수 있다. 또한, VMM-관리되는 gAPIC 상태 엔트리(92)는 게스트 ID와 논리 및 물리 ID들을 저장하거나, 또는 이들은 엔트리(92)를 선택함에 있어서 고유할 수 있다(즉, VMM(18)은 이러한 값들을 이용하여, 데이터 구조(68)로부터 엔트리(92)를 선택할 수 있다).

[0079] 다음으로, 도 6은 게스트에 대해 IOMMU(40)로부터 인터럽트 메시지를 수신하는 것에 응답하여, 게스트 인터럽트 매니저(38)의 일 실시예의 동작을 도시하는 흐름도이다. 이해의 용이를 위해, 블록들을 특정 순서로 나타내기는 했지만, 다른 순서들도 이용될 수 있다. 블록들은 게스트 인터럽트 매니저(38)의 결합 논리에서 동시에 수행될 수 있다. 블록들, 블록들의 결합들, 및/또는 흐름도는 다수의 클럭 주기들에 대해 파이프라인될 수 있다. 일반적으로, 게스트 인터럽트 매니저(38)는 도 6에 도시된 동작을 실시하도록 구성될 수 있다.

[0080] 몇몇 실시예들에서, 인터럽트 메시지의 처리는 인터럽트가 논리적인지 또는 물리적인지에 의존하여 (즉, 그 인터럽트의 전달 모드가 논리적인지 또는 물리적인지에 의존하여) 달라질 수 있다. 예를 들어, 도 11의 실시예에서는, 논리적 인터럽트들 및 물리적 인터럽트들에 대해 다른 테이블들이 읽혀진다. 도 12 및 도 13에서, 논리적 및 물리적 테이블들은 메모리 내에서 인접할 수 있지만, 논리적 인터럽트에 대한 논리 테이블의 위치를 찾기 위해 베이스 어드레스 포인터에 오프셋이 부가될 수 있으며, 물리적 인터럽트에 대해서는 어떠한 오프셋도 부가될

필요가 없다. 이에 따라, 게스트 인터럽트 매니저(38)는 인터럽트가 논리적인지 또는 물리적인지를 결정하도록 구성될 수 있다(결정 블록 100). 다른 실시예들은 전달 모드에 기초하여 달라지지 않으며, 결정 블록(100)을 없앨 수 있고, (그리고, 하기 설명되는 방송 또는 더 많은 목적지들에 대한 체크는 양자를 위한 체크 내에 병합될 수 있다).

[0081] 인터럽트가 논리적이면(결정 블록(100)에서 "예"이면), 게스트 인터럽트 매니저(38)는 논리적인 인터럽트로부터 gAPIC 상태 데이터 구조(58) 내의 해당하는 gAPIC 상태 엔트리(90)로의 맵핑을 결정하도록 구성된다(블록 102). 다양한 실시예들은 도 10-13에 도시한 바와 같이 다른 맵핑들을 구현할 수 있으며, 이에 따라 이러한 결정은 달라질 수 있다. 게스트 인터럽트 매니저(38)는 gAPIC 상태 엔트리(90)에 표현되는 IRR 내의 인터럽트 벡터에 해당하는 비트를 세트시키도록 구성된다(블록 104). 논리적인 인터럽트들은 다수의 목적지들을 가질 수 있다(예를 들어, 클러스터 내의 목적지는 1개 이상의 세트된 비트를 가질 수 있는 비트 벡터이다). 만일 논리적인 인터럽트가 더 많은 목적지들을 포함한다면(결정 블록(106)에서, "예"이면), 게스트 인터럽트 매니저(38)는 각각의 부가적인 목적지에 대해 블록들(102 및 104)을 반복하도록 구성될 수 있다. 대안적으로, 도 12의 실시예에서, 논리적인 목적지 비트 벡터는 하나의 동작으로 gAPIC 상태 엔트리에 기록될 수 있는 바, 이에 대해서는 하기에서 보다 상세히 설명된다. 게스트 인터럽트 매니저(38)는 gAPIC들(34A-34D)에 인터럽트 메시지를 전송하도록 구성될 수 있다(블록 108).

[0082] 인터럽트가 물리적이면(결정 블록(100)에서, "아니오"이면), 게스트 인터럽트 매니저(38)는 물리적인 인터럽트로부터 해당하는 gAPIC 상태 엔트리(90)로의 맵핑을 결정하도록 구성될 수 있다(블록 110). 다양한 실시예들은 도 10-13에 도시한 바와 같이 다른 맵핑들을 구현할 수 있으며, 이에 따라 이러한 결정은 달라질 수 있다. 게스트 인터럽트 매니저(38)는 gAPIC 상태 엔트리(90)에 표현되는 IRR 내의 인터럽트 벡터에 해당하는 비트를 세트시키도록 구성된다(블록 112). 물리적인 인터럽트들은 방송되거나 또는 단일 목적지를 가질 수도 있다. 만일 물리적인 인터럽트가 방송된다면(결정 블록(114)에서, "예"이면), 게스트 인터럽트 매니저(38)는 게스트의 가상 머신 내의 각 목적지(예를 들어, 각 vCPU)에 대해 블록들(110 및 112)을 반복하도록 구성될 수 있다. 대안적으로, 도 12의 실시예에서, 방송은 하나의 동작으로 gAPIC 상태 엔트리에 기록될 수 있는바, 이에 대해서는 하기에서 보다 상세히 설명된다. 게스트 인터럽트 매니저(38)는 gAPIC들(34A-34D)에 인터럽트 메시지를 전송하도록 구성될 수 있다(블록 108).

[0083] gAPIC 상태 엔트리(90)에 표현되는 IRR 내의 비트의 설정은 원자 OR 동작으로 수행될 수 있는데, 여기서 세트되는 비트는 메모리 위치 내의 다른 IRR 비트들에 OR된다. 이러한 원자 OR 동작의 실제 구현은, 로크(lock)된 판독/변경/기록 동작으로부터, 하나의 동작으로서 OR을 수행하도록 정의된 특별 목적의 회로까지 달라질 수 있다. 상기 설명한 바와 같이, 다른 실시예들에서는, 비교 및 스와프 동작이 수행될 수 있다.

[0084] 다른 실시예에서, 하나 이상의 목적지를 갖는 논리적인 인터럽트들 및 방송되는 물리적인 인터럽트들은, VMM(18)(예를 들어, 이벤트 큐)에 액세스가능한 데이터 구조에 인터럽트를 로깅함으로써, 게스트 인터럽트 매니저(38)에 의해 핸들링될 수 있다. 또한, 게스트 인터럽트 매니저(38)는 VMM(18)에게 이벤트를 통지하기 위해 VMM(18)에게 신호하도록 구성될 수 있다(예를 들어, 프로세서들(30A-30B)중 하나 상의 가상 머신으로부터 빠져나오게 한다). 대안적으로, 게스트 인터럽트 매니저(38)는 VMM(18)에게 단지 주기적으로 (예를 들어, 매 N 밀리초에 한번씩 및/또는 이벤트 큐에서의 높은 워터마크(watermark)에서) 신호할 수 있으며, 그리고 VMM(18) 역시 이벤트 큐를 주기적으로 체크하여, 시그널링이 지원할 수 있는 것 보다 더 신속하게 임의의 이벤트들을 서비스할 수 있게 된다. 일 실시예에서, 이벤트 큐는 게스트 인터럽트 관리자(38) 대신 IOMMU(40)에 의해 관리될 수 있다.

[0085] 다음으로, 도 7은 게스트 인터럽트 매니저(38)로부터 인터럽트 메시지를 수신하는 것에 응답하여, gAPIC(34A-34D)의 실시예의 동작을 도시하는 흐름도이다. 이해의 용이를 위해, 블록들을 특정 순서로 나타내기는 했지만, 다른 순서들도 이용될 수 있다. 블록들은 gAPIC의 결합 논리에서 동시에 수행될 수 있다. 블록들, 블록들의 결합들, 및/또는 흐름도는 다수의 클럭 주기들에 대해 파이프라인될 수 있다. 일반적으로, gAPIC는 도 7에 도시된 동작을 실시하도록 구성될 수 있다.

[0086] 일 실시예에서, gAPIC는 (게스트 ID 레지스터(84)(도 4 참조)의) 자신의 게스트 ID를 0으로 설정함으로써 비활성화(deactivate)된다. 따라서, 인터럽트 메시지를 수신하는 것에 응답하여, gAPIC의 게스트 ID가 0 이면(결정 블록(120)에서, "예"이면), gAPIC는 비 활동적이 되며, 인터럽트를 처리하지 않는다. 다른 실시예들은 다른 방식들(예를 들어, 레지스터 내의 액티브 비트)로 gAPIC를 비활성화시킬 수 있으며, 결정 블록(120)은 gAPIC 활동적/비활동적을 체크하도록 그에 따라 변경될 수 있다.

[0087] 만일 gAPIC의 게스트 ID가 0이 아니면, 이 gAPIC는 이러한 게스트 ID와 수신된 인터럽트의 게스트 ID를 비교할 뿐 아니라, 수신된 목적지 ID와 각각 레지스터들(80 및 82) 내의 논리적 ID 및 물리적 ID를 비교한다(도 4 참조). 만일 gAPIC의 게스트 ID가 수신된 게스트 ID와 일치하지 않으면(결정 블록(122)에서, "아니오"이면), gAPIC는 현재 다른 게스트에 할당되어 있으며, 이에 따라 gAPIC는 인터럽트에 의해 목표로 되지 않는다. gAPIC는 인터럽트의 비 수신확인(non-ACK)에 응답하도록 구성될 수 있다. 이러한 비 수신확인은, gAPIC가 인터럽트를 수신했지만, 이 인터럽트가 해당 프로세서를 목표로 하지 않기 때문에, 수락하지 않았음을 나타낼 수 있다. 유사하게, gAPIC의 게스트 ID가 수신된 게스트 ID와 일치하지만, 인터럽트가 논리적이고, gAPIC의 논리적인 ID와 일치하지 않거나, 또는 인터럽트가 물리적인 단일 목적지이고, gAPIC의 물리적인 ID와 일치하지 않으면(결정 블록들(126 및 128)에서, "아니오"이면), gAPIC는 인터럽트의 비 수신확인에 응답하도록 구성될 수 있다(블록 124).

[0088] 일반적으로, 논리적인 인터럽트를 매칭시키는 것은, 논리적인 ID들의 클러스터 ID 부분이 같은 지를 비교하는 것과, 그리고 gAPIC의 논리적인 ID 레지스터 내의 세트된 비트가 인터럽트로부터 수신된 논리적인 ID의 목적지 부분에도 세트되어 있음을 결정하는 것을 포함한다. 또한, 하나 이상의 논리적인 목적지가 있다면, 인터럽트의 논리적인 ID의 목적지 부분 내의 다른 비트들이 세트될 수 있다. 게스트 ID가 매치하는 한, 방송 물리 인터럽트가 매치로서 처리될 수 있는 것을 제외하고, 물리적인 ID들이 같은 지에 대해 비교될 수 있다.

[0089] 만일 인터럽트가 논리적이고, 논리적인 ID와 매치하거나(결정 블록(126)에서, "예"이면), 또는 인터럽트가 물리적이고, 물리적인 ID와 매치하거나 또는 방송된다면(결정 블록(128)에서, "예"이면), gAPIC는 게스트 인터럽트 매니저(38)에 응답하도록 구성될 수 있는 바, gAPIC가 해당 프로세서(30A-30B)에 제시하기 위한 인터럽트를 수신하고 있음을 나타내는 수신 확인을 한다(블록 130). 또한, gAPIC는 IRR 레지스터(70)를 업데이트하도록 구성됨으로써, 인터럽트 메시지 내의 인터럽트 벡터에 해당하는 인터럽트 요청 비트를 설정한다(블록 132). gAPIC는 임의의 인서비스 인터럽트들 및/또는 태그스 우선순위 레비스터에 대해 인터럽트의 우선순위를 재평가(reevaluate)하도록 구성될 수 있으며, 그리고 이러한 재평가에 기초하여 프로세서에 인터럽트를 신호하도록 구성될 수 있다(블록 136). 즉, gAPIC는, 인터럽트의 우선순위가 인서비스 인터럽트 보다 높은 우선순위를 갖고, 작업 우선순위 레지스터 보다 높은 우선순위를 갖는 경우, 인터럽트를 신호하도록 구성될 수 있다.

[0090] 이제, 도 8을 참조하면, gAPIC 상태를 하나의 게스트로부터 다른 게스트로 변경하기 위한 VMM(18)의 일 실시예의 동작을 도시하는 흐름도가 나타나있다. 즉, 도 8의 흐름도는 gAPCI(34A-34D)를 하나의 게스트/vCPU로부터 다른 게스트, 또는 동일한 게스트 내의 다른 vCPU로 재할당하는 것을 나타낸다. 이해의 용이를 위해, 블록들을 특정 순서로 나타내기는 했지만, 다른 순서들도 이용될 수 있다. 일반적으로, VMM(18)은, 시스템(5) 상에서 실행될 때, 도 8에 도시된 동작을 실시하는 명령들을 포함할 수 있다.

[0091] VMM(18)은 gAPIC 상태 데이터 구조(58) 내에서의 "오래된 게스트(old guest)"(gAPIC로부터 비활성화되고 있는 게스트)에 해당하는 gAPIC 상태 엔트리(90)의 위치를 결정할 수 있다(블록 140). gAPIC 상태 엔트리(90) 내의 데이터는 "스테일(stale)"한데, 왜냐하면 이것은 gAPIC에 의해 변경되었기 때문이다. 예를 들어, IRR 비트는 해당 프로세서에 인터럽트를 전달하는 것에 응답하여 리셋되었을 수 있다. 이에 따라, VMM(18)은 gAPIC 상태 엔트리(90) 내의 IRR을 0이 되게 한다(블록 142). VMM(18)은 게스트 ID 레지스터(84), 논리적 ID 레지스터(82) 및 물리적 ID 레지스터(80)를 클리어할 수 있다(블록 144). 이러한 행동은 gAPIC가 어떠한 부가적인 인터럽트들을 수락하는 것을 멈추게 할 수 있는데, 왜냐하면 게스트 ID, 논리적 ID 및 물리적 ID가 어떠한 인터럽트 메시지들과도 매치되지 않을 것이기 때문이다. 레지스터들(80-84)이 클리어된 후(블록 144), 그리고 IRR 상태가 gAPIC 상태 엔트리(90)에 기록되기 전에, 인터럽트가 전송되는 것이 가능하다. 이에 따라, 인터럽트 상태의 손실을 막기 위해, VMM(18)은 IRR(70)로부터의 IRR 상태를 gAPIC 상태 엔트리(90) 내로 자동으로 OR시킬 수 있다(블록 146). 또한, VMM(18)은 오래된 게스트와 관련된 VMM-관리되는 gAPIC 상태 엔트리(92)에 다른 gAPIC 상태를 기록할 수 있다(블록 148).

[0092] VMM(18)은 오래된 게스트의 인터럽트 상태를 제거하기 위해, gAPIC의 IRR, ISR 및 TMR 레지스터들(70, 72 및 74)을 클리어할 수 있다(블록 150). VMM(18)은 gAPIC에 할당되고 있는 게스트에 대한 새로운 게스트 ID, 논리적 ID 및 물리적 ID를, 게스트 ID 레지스터(84), 논리적 ID 레지스터(82) 및 물리적 ID 레지스터(80)에 각각 기록할 수 있다(블록 152). 일단 블록(152)이 수행되면, gAPIC는 게스트에 대한 인터럽트들을 수락하는 것을 시작할 수 있다. VMM(18)은 "새로운 게스트"(gAPIC에서 활성화되고 있는 게스트에 대한 gAPIC 상태 엔트리(90)를 결정할 수 있으며(블록 154), 그리고 gAPIC 상태 엔트리(90)로부터 IRR 상태를 판독할 수 있다(블록 156). 레지스터들(80-84)의 프로그래밍은 gAPIC로 하여금 인터럽트들을 수락하게 할 수 있기 때문에, VMM(18)이 엔트리를 판독한 후 gAPIC 상태 엔트리(90)에 기록되었던 IRR 내의 인터럽트를 gAPIC가 수락할 수 있다. 이에 따라, VMM(18)

은 IRR 레지스터(70) 내에 IRR 상태를 자동으로 OR시킬 수 있다. 즉, gAPIC는 IRR 레지스터 상에서의 원자 OR 동작을 지원할 수 있다(블록 158). VMM(18)은 새로운 게스트에 대해 VMM-관리되는 gAPIC 상태 엔트리(92)로부터 다른 상태를 판독하고(블록 160), 이러한 상태를 gAPIC에 기록할 수 있다(블록 162). 주목할 사항으로서, 블록들(160 및 162)은 또한 블록(150) 이후의 어떠한 다른 지점에서도 수행될 수 있다.

[0093] 블록들(140-148)은 일반적으로 gAPIC로부터 게스트를 비활성화시키기 위한 동작들을 나타낼 수 있는 반면, 블록들(150-162)은 일반적으로 gAPIC 내에서 게스트를 활성화시키기 위한 동작들을 나타낼 수 있다. 이에 따라, 도 8에서 수평 점선으로 나타낸 바와 같이, 만일 VMM(18)이 gAPIC에서 게스트를 비활성화시키는 것 만을 원한다면, 수평 점선 윗쪽의 블록들이 수행될 수 있다. 만일 VMM(18)이 gAPIC에서 게스트를 활성화시키는 것 만을 원한다면, 수평 점선 아랫쪽의 블록들이 수행될 수 있다.

[0094] 도 9는 일 실시예에 따른, gAPIC 상태 엔트리(90)내의 gAPIC 상태의 하나의 예시적인 배열(170)을 도시하는 블록도이다. 도 9의 블록도에서, IRR의 각 비트는 다른 바이트에 저장된다. 예를 들어, 도 9의 IRR 비트 0, 또는 IRR0 은 메모리 내의 연속적인 바이트들의 세트의 바이트 0에 저장되고; IRR1 은 바이트 1에 저장되고; ... IRR255 는 바이트 255에 저장된다. 도시된 실시예에서는, IRR 비트가 바이트의 비트 0에 저장되기는 하지만, 임의의 비트 위치가 이용될 수 있다. 도시된 실시예에서, 바이트 내의 다른 비트들은 DC(don't care)이다. 개별적인 바이트(메모리 액세스의 최소 단위)에 각 비트를 저장함으로써, 각 비트는 다른 비트들에 영향을 주지 않으면서 개별적으로 기록될 수 있다. 따라서, 비트는 바이트에 대한 기록에 의해 세트될 수 있는 바, 이는 원자 동작이다. 바이트 내의 IRR 비트 위치에 세트되는 비트를 기록하고, 나머지 바이트들을 업데이트하지 않음으로써, IRR 비트의 원자 OR가 그 결과가 될 수 있다. 다른 실시예들에서, 원자 OR는 다른 방식들로 달성될 수 있으며, IRR 상태의 비트들은 다른 방식들로 저장될 수 있다.

[0095] 도 10은 gAPIC 상태 엔트리(90)의 일 실시예의 블록도이다. 도시된 실시예에는, gAPIC 상태 맵핑 테이블들(60)의 일 실시예 뿐 아니라, 장치 테이블(62) 및 인터럽트 리다이렉트 테이블(64)이 나타나있다. 본 실시예에서, 인터럽트를 전송한 주변장치의 BDF는 장치 테이블(62) 내로의 인덱스로서 이용되며, 그리고 엔트리는 BDF가 할당되는 게스트에 대한 게스트 ID를 포함할 수 있다. 부가적으로, 이러한 예에서, 엔트리는 인터럽트 리다이렉트 테이블 포인터(IRTP)를 포함할 수 있는데, 이는 인터럽트 리다이렉트 테이블(64)의 베이스를 지시한다. 인터럽트 리다이렉트 테이블(64) 내로의 인덱스는 인터럽트에 대한 인터럽트 식별자이다. 이러한 인터럽트 식별자는 인터럽트 벡터를 포함할 수 있으며, 물리적 또는 논리적인, 인터럽트의 전달 모드(Delmode)를 또한 포함할 수 있다. 선택된 엔트리는 새로운 벡터 및 목적지 ID (DestID)를 포함할 수 있다. 인터럽트 리다이렉트 테이블(64)을 이용하지 않는 실시예들에서, 주변장치에 의해 제공되는 목적지 ID 및 인터럽트 벡터는 gAPIC 상태 맵핑 테이블(60)을 인덱스하는 데에 직접적으로 이용될 수 있다.

[0096] gAPIC 상태 맵핑 테이블(60)은 gAPIC 상태 맵핑 테이블 베이스 어드레스에 의해 메모리 내에 위치될 수 있다. 다양한 실시예들에서, 베이스 어드레스는 모든 게스트들에 대해 동일하거나, 게스트 마다 특정되거나, 또는 장치 테이블(62) 내에 저장될 수도 있다. 도 10에서, 베이스 어드레스는 계층적인 테이블들의 세트의 가장 높은 레벨(L3)을 식별하는바, 이러한 테이블은 더 낮은 레벨의 테이블들(예를 들어, L2, 및 L2를 지시하지 않는 L3으로부터의 포인터들에 의해 표시되는 유사한 테이블들)에 대한 포인터들을 저장할 수 있다. L2 테이블들은 훨씬 더 낮은 레벨의 테이블들(L1)에 대한 포인터들을 저장할 수 있는데, 이는 gAPIC 상태 데이터 구조(58) 내의 gAPIC 상태 엔트리(90)에 대한 포인터들을 포함할 수 있다. 다른 실시예들은, 도 10에 나타낸 3 레벨 보다 더 많거나 적은 레벨을 포함하여, 계층 구조 내에서 임의 수의 레벨들을 이용할 수 있다.

[0097] gAPIC 상태 맵핑 테이블(60) 내의 각 레벨(L3-L1)로의 인덱스는 장치 테이블(62)로부터 게스트 ID를 연관시킴으로써 형성되는 값의 일부, 주변장치로부터 또는 인터럽트 리다이렉트 테이블(64)로부터의 인터럽트 벡터, 및 주변장치로부터 또는 인터럽트 리다이렉트 테이블(64)로부터의 목적지 ID가 될 수 있다. 레벨들(L3-L1)에 대한 인덱스들은 연관된 값의 모든 비트들을 소모할 수 있으며, 이에 따라 게스트 ID, 벡터 및 목적지 ID의 각각의 결합은 gAPIC 상태 맵핑 테이블(60) 내에서 그 자신 만의 고유한 포인터를 가질 수 있다. 몇몇 포인터들은 동일한 gAPIC 상태 엔트리(90)를 지시할 수도 있지만, (예를 들어, 일 실시예에서는, 동일한 gAPIC의 논리적 및 물리적 ID들이 동일한 gAPIC 상태(90)에 대한 포인터들을 가질 수도 있다).

[0098] 도 11은 gAPIC 상태 엔트리(90)의 위치를 찾는 다른 실시예의 블록도이다. 도시된 실시예에서는, gAPIC 상태 맵핑 테이블들(60)의 일 실시예 뿐 아니라, 장치 테이블(62) 및 인터럽트 리다이렉트 테이블(64)이 나타나있다. 본 실시예에서, 인터럽트를 전송한 주변장치의 BDF는 장치 테이블(62) 내로의 인덱스로서 이용되며, 그리고 엔트리는 BDF가 할당되는 게스트에 대한 게스트 ID를 포함할 수 있다. 부가적으로, 이러한 예에서, 엔트리는 인터

럽트 리다이렉트 테이블 포인터(IRTP)를 포함하는데, 이는 인터럽트 리다이렉트 테이블(64)의 베이스를 지시한다. 장치 테이블(62)은 gAPIC 상태 맵핑 테이블들(60) 내의 테이블들에 대한 하나 이상의 포인터들을 더 포함할 수 있다. 구체적으로, 게스트 물리 테이블에 대한 포인터 및 게스트 논리 테이블에 대한 다른 포인터가 저장될 수 있다. 게스트 물리 테이블은 gAPIC 상태 엔트리들(90)에 물리적인 목적지 ID들을 맵핑할 수 있다. 즉, 게스트 물리 테이블은 목적지 ID에 의해 인덱스될 수 있으며, gAPIC 상태 엔트리들(90)에 대한 포인터들을 저장할 수 있다. 유사하게, 게스트 논리 테이블은 gAPIC 상태 엔트리들(90)에 논리적인 목적지 ID들을 맵핑할 수 있다.

[0099] 인터럽트 리다이렉트 테이블(64) 내로의 인덱스는 인터럽트에 대한 인터럽트 식별자이다. 이러한 인터럽트 식별자는 인터럽트 벡터를 포함할 수 있으며, 물리적 또는 논리적인, 인터럽트의 전달 모드(Delmode)를 또한 포함할 수 있다. 선택된 엔트리는 새로운 벡터 및 목적지 ID (DestID)를 포함할 수 있다. 인터럽트 리다이렉트 테이블(64)을 이용하지 않는 실시예들에서, 주변장치에 의해 제공되는 목적지 ID 및 인터럽트 벡터는 gAPIC 상태 맵핑 테이블(60)을 인덱스하는 데에 직접적으로 이용될 수 있다.

[0100] 도 12는 gAPIC 상태 엔트리(90)의 위치를 찾는 다른 실시예의 블록도이다. 이러한 실시예에서는, 어떠한 gAPIC 상태 맵핑 테이블(60)도 없다. 도 10-11의 실시예와 유사하게, 인터럽트를 전송한 주변장치의 BDF는 장치 테이블(62) 내로의 인덱스로서 이용되며, 그리고 엔트리는 BDF가 할당되는 게스트에 대한 게스트 ID를 포함하고, 인터럽트 리다이렉트 테이블(64)의 베이스를 지시하는 인터럽트 리다이렉트 테이블 포인터(IRTP)를 선택적으로 포함한다. 장치 테이블(62)은 gAPIC 상태 데이터 구조(58) 내의 테이블의 베이스에 대한 적어도 하나의 포인터(Ptr)를 더 포함할 수 있다. 도시된 실시예에서, 이러한 테이블은 게스트 물리 부분(section: 또는, '섹션' 이라함)(180) 및 게스트 논리 부분(182)을 포함한다. 도면에서의 명확성을 위해, 도 12에서 이러한 부분들(180 및 182)은 이들 사이에 공간을 가지며 도시되었지만, 이들은 메모리 내에서 인접할 수도 있다. 즉, 게스트 물리 부분(180)의 꼭대기가 게스트 논리 부분(182)의 바닥에 인접할 수 있다. 장치 테이블(62) 엔트리는 게스트 논리 부분(182)의 꼭대기를 나타내는 논리 리미트(logical limit)(LLim) 필드를 더 포함할 수 있다. 다른 실시예들에서, 게스트 물리 부분(180) 및 게스트 논리 부분(182)은 인접하지 않을 수도 있으며, 그리고 개별적인 포인터들이 장치 테이블(62) 엔트리에 저장됨으로써, 게스트 물리 부분(180) 및 게스트 논리 부분(182)을 각각 나타낼 수 있다.

[0101] 도 12의 실시예에서, 게스트 물리 부분(180)은 (주변장치로부터의, 또는 인터럽트 리다이렉트 테이블(64)로부터의) 인터럽트 벡터에 의해 인덱스될 수 있다. 게스트 물리 부분(180) 내의 각 엔트리는 게스트 물리 머신에서 지원되는 목적지 ID들(예를 들어, 도 12에서, 0 내지 63의 번호가 붙은 64개까지의 목적지들)에 해당하는 비트 벡터를 포함할 수 있다. 물리적인 인터럽트에 응답하여, 게스트 인터럽트 매니저(38)는 목적지 ID에 해당하는 인터럽트 벡터에 대한 엔트리에 비트를 세트시키도록 구성될 수 있다. 방송 인터럽트에 대해, 게스트 인터럽트 매니저(38)는, 가상 머신 내의 vCPU들의 수 까지, 인터럽트 벡터에 해당하는 엔트리에 각 비트를 세트시키도록 구성될 수 있다.

[0102] 게스트 논리 부분(182)은 벡터 및 논리적인 ID의 클러스터 부분에 의해 인덱스될 수 있다. 이러한 클러스터 부분은 인덱스의 최상위 비트들이 될 수 있으며, 이에 따라 게스트 논리 부분(182)은 각 논리적인 클러스터(도 12의 클러스터 0 내지 클러스터 N)에 해당하는 클러스터 부분들로 분할된다. 각 클러스터 내에서, 엔트리들은 인터럽트 벡터에 의해 배열되며, 각 엔트리는 논리적인 ID들의 벡터 부분에 해당하는 비트 벡터를 저장한다. 도시된 실시예에서는, 하나의 클러스터 내에 16개까지의 목적지들이 포함될 수 있다(예를 들어, 논리적인 ID의 비트 벡터 부분은 16 비트가 될 수 있다). 논리적인 인터럽트에 응답하여, 게스트 인터럽트 매니저(38)는 인터럽트 벡터에 해당하는 엔트리의 콘텐츠들과 논리적인 ID의 비트 벡터 부분을 논리적으로 OR시키도록 구성될 수 있다.

[0103] 이에 따라, 도 12의 실시예는 gAPIC 상태 데이터 구조(58)에 단일 업데이트로 다수의 목적지들을 갖는 논리적인 인터럽트들 및 방송되는 물리적인 인터럽트들을 기록하는 것을 지원할 수 있다. gAPIC에 대한 gAPIC 상태 엔트리는, gAPIC의 물리적인 ID에 해당하는 게스트 물리 부분(180)의 컬럼(column)(이는 gAPIC의 논리적인 ID에 의해 표시되는 클러스터로부터의 컬럼과 논리적으로 OR된다)과, 그리고 gAPIC의 논리적인 ID의 비트 벡터 부분 내의 세트된 비트에 의해 식별되는 컬럼을 포함할 수 있다. gAPIC 내의 게스트를 비활성화시키는 것에 응답하여 gAPIC 상태 데이터 구조(58)를 업데이트하는 것은, 그 게스트에 해당하는 컬럼들중 하나를 0이 되게 하고, 나머지 컬럼에 IRR을 기록하는 것을 포함할 수 있다.

[0104] 도 13은 gAPIC 상태 엔트리(90)의 위치를 찾는 다른 실시예이다. 도 13의 실시예는, 게스트 물리 부분(180) 및 게스트 논리 부분(182) 내에서의 데이터의 배열이 다른 것을 제외하고, 도 12의 실시예와 유사하다. 각 엔트리는 IRR에 해당하며, 이에 따라 각 인터럽트 벡터에 대해 하나의 비트를 포함한다. 게스트 물리 부분(180)은 인

터럽트의 물리적인 ID에 의해 인덱스되며, 게스트 논리 부분(182)은 인터럽트의 논리적인 ID에 의해 인덱스된다. 인터럽트 벡터에 해당하는 IRR 비트는 인터럽트의 전달 모드에 의존하여 논리 부분(182) 또는 물리 부분(180) 내에 세트된다. 게스트/vCPU에 대한 gAPIC는 그 게스트/vCPU에 할당된 물리적인 ID에 해당하는 게스트 물리 부분(180)으로부터의 로우(row)와, 그 게스트/vCPU에 할당된 논리적인 ID에 해당하는 게스트 논리 부분(182)으로부터의 로우의 OR 이다.

[0105] 도 14는 호스트 하드웨어(20)의 다른 실시예의 블록도를 도시한다. 도시된 실시예에서는, 도 2의 집적 회로(66)와 유사하게, 2개의 집적 회로들(66A-66B)이 포함된다. 따라서, 나타낸 바와 같이, 각 집적 회로는, 집적 회로(66A) 내의 34A-34D 및 집적 회로(66B) 내의 34E-34G와 같은 gAPIC들을 포함할 수 있다. 각 집적 회로(66A-66B)는 각각의 게스트 인터럽트 매니저(28A-28B) 및 IOMMU(도 14에는 미도시)를 포함할 수 있다. 이러한 집적 회로들(66A-66B) 중 적어도 하나는 메모리(56A-56B)에 결합되며, 그리고 선택적으로는, 집적 회로들(66A-66B) 모두가 메모리들을 포함할 수도 있다. 도시된 실시예에서, 집적 회로들(66A-66B)은 인터페이스 회로들(44C 및 44D)을 통해 결합된다. 다른 실시예들에서는, 2개 보다 많은 집적 회로들(66A-66B)이 제공될 수 있으며, 다양한 집적 회로들은 임의의 요구되는 방식으로 서로 연결될 수 있다.

[0106] 일 실시예에서는, 각각의 게스트 인터럽트 매니저(28A-28B)가 인에이블되어, 동일한 집적 회로 내의 gAPIC들(34A-34C)를 목표로 하는 인터럽트 메시지들을 관리할 수 있다. 따라서, 이러한 게스트 인터럽트 매니저(28A-28B)는 게스트 인터럽트 전달에 대한 스케일러블 솔루션(scalable solution)을 제공할 수 있다. 게스트 인터럽트 매니저(28A-28B)에 의해 이용되는 데이터 구조들은 하나의 메모리(예를 들어, 메모리(58A))에 저장되거나, 또는 각 게스트 인터럽트 매니저(28A-28B)가 그 자신의 메모리(58A-58B) 내에 그 자신의 데이터 구조들을 가질 수 있다. 데이터 구조들에 대한 액세스에 관하여 약간의 논쟁(contention)이 있을 수도 있지만, 많은 경우들에서, (집적 회로들(66A-66B) 중 하나 내의 프로세서 상에서 실행되고 있는) 특정의 게스트에 주변장치가 할당되며, 이에 따라 논쟁의 양은 비교적 작을 수 있다.

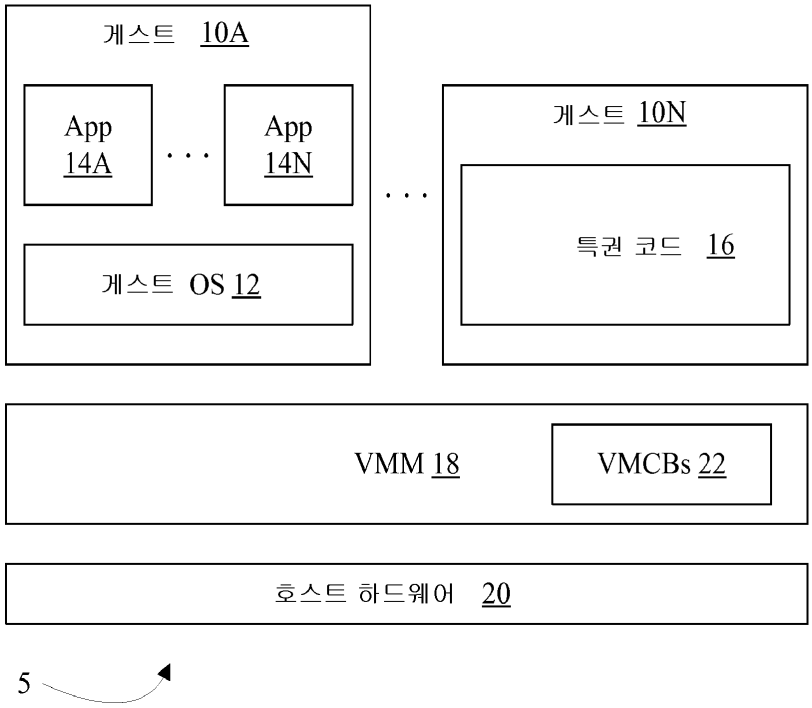
[0107] 다른 실시예들에서는, 게스트 인터럽트 매니저들(28A-28B) 중 하나가 인에이블되어, 시스템 내의 각 gAPIC(34A-34G)에 대한 게스트 인터럽트 전달을 수행할 수 있다. 이러한 실시예는 집적 회로들(66A-66B) 간의 상호 연결에 대하여 더 많은 인터럽트-관련 트래픽을 겪을 수 있지만, 게스트 인터럽트 관리를 위한 중심점의 개념적인 단순화를 제공할 수 있다.

[0108] 도 15는 컴퓨터 액세스가능한 저장 매체(200)의 일 실시예의 블록도이다. 대체로, 컴퓨터 액세스가능한 저장 매체는 컴퓨터에 명령들 및/또는 데이터를 제공하기 위해 이용하는 동안 컴퓨터에 의해 액세스될 수 있는 임의의 저장 매체를 포함할 수 있다. 예를 들어, 컴퓨터 액세스가능한 저장 매체는 자기 또는 광 매체와 같은 저장 매체, 예를 들어 (고정된 또는 착탈가능한) 디스크, 테이프, CD-ROM, 또는 DVD-ROM, CD-R, CD-RW, DVD-R, DVD-RW, 및/또는 블루레이 디스크(Blue-Ray discs)를 포함할 수 있다. 저장 매체는, RAM(예를 들어, SDRAM, RDRAM (램버스 DRAM), 정적 RAM (SRAM) 등), ROM, 플래시 메모리, 범용 직렬 버스(USB) 인터페이스 또는 임의의 다른 인터페이스와 같은 주변장치 인터페이스를 통해 액세스가능한 비휘발성 메모리(예를 들어, 플래시 메모리) 등과 같은, 휘발성 또는 비휘발성 메모리 매체를 더 포함할 수 있다. 저장 매체는, 네트워크 및/또는 무선 링크와 같은 통신 매체를 통해 액세스가능한 저장 매체 뿐 아니라, MEMS(microelectromechanical systems)를 포함할 수 있다. 도 15의 컴퓨터 액세스가능한 저장 매체(200)는 VMM(18)을 저장할 수 있는데, 이 VMM(18)은 도 8의 흐름도 및/또는 VMM(18)에 할당되는 임의의 다른 기능을 구현할 수 있다. 일반적으로, 컴퓨터 액세스가능한 저장 매체(20)는, 실행될 때, 도 8에 나타낸 흐름도의 일부 또는 전부를 구현하는 명령들의 임의의 세트를 저장할 수 있다. 캐리어 매체는 유선 또는 무선 전송과 같은 전송 매체 뿐 아니라, 컴퓨터 액세스가능한 저장 매체를 포함할 수 있다.

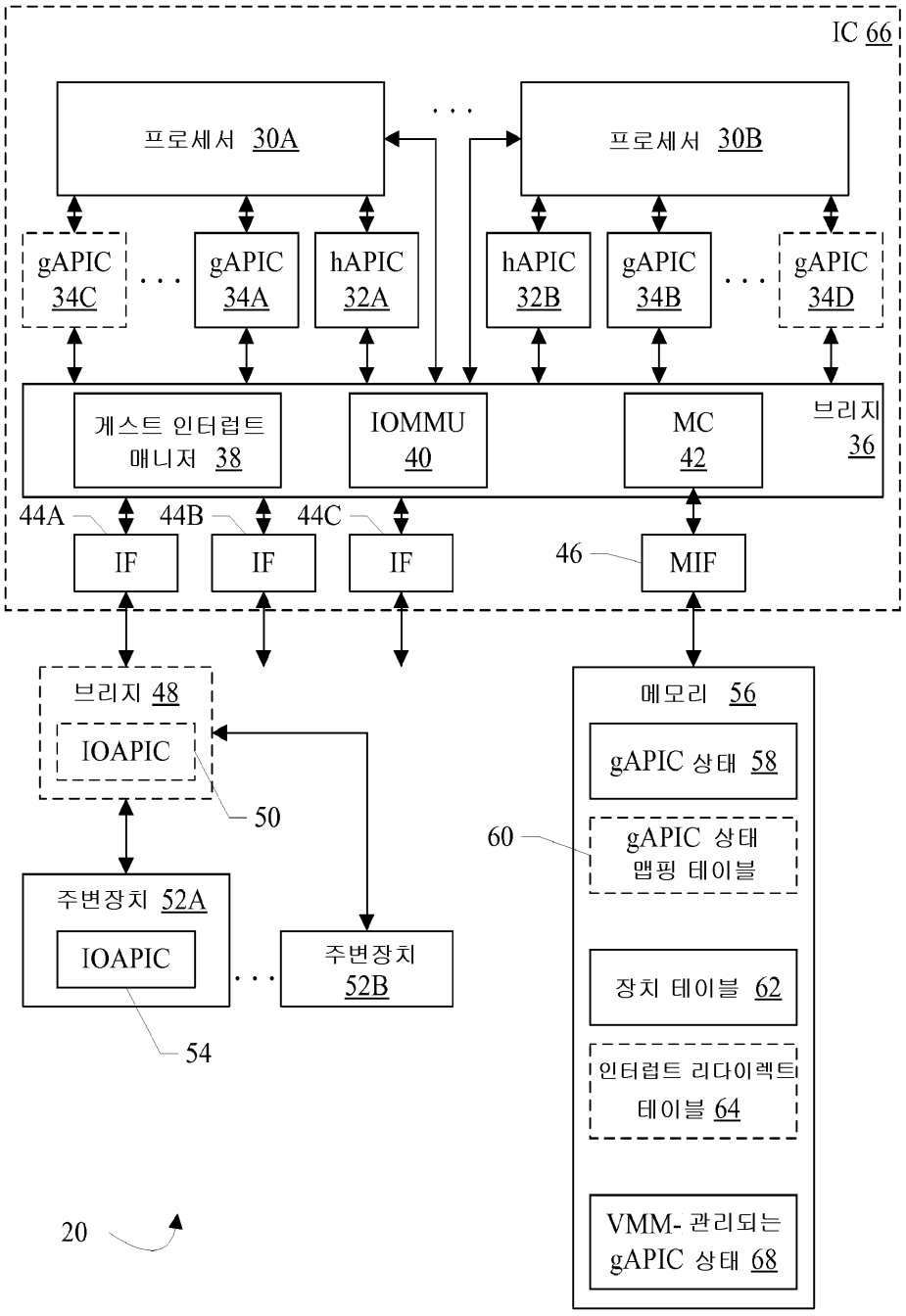
[0109] 상기 개시를 완전히 이해하게 되면, 많은 수정들 및 변형들이 당업자에게 자명할 것이다. 하기의 청구범위는 이러한 모든 수정들 및 변형들을 포괄하는 것으로서 해석되어야 한다.

도면

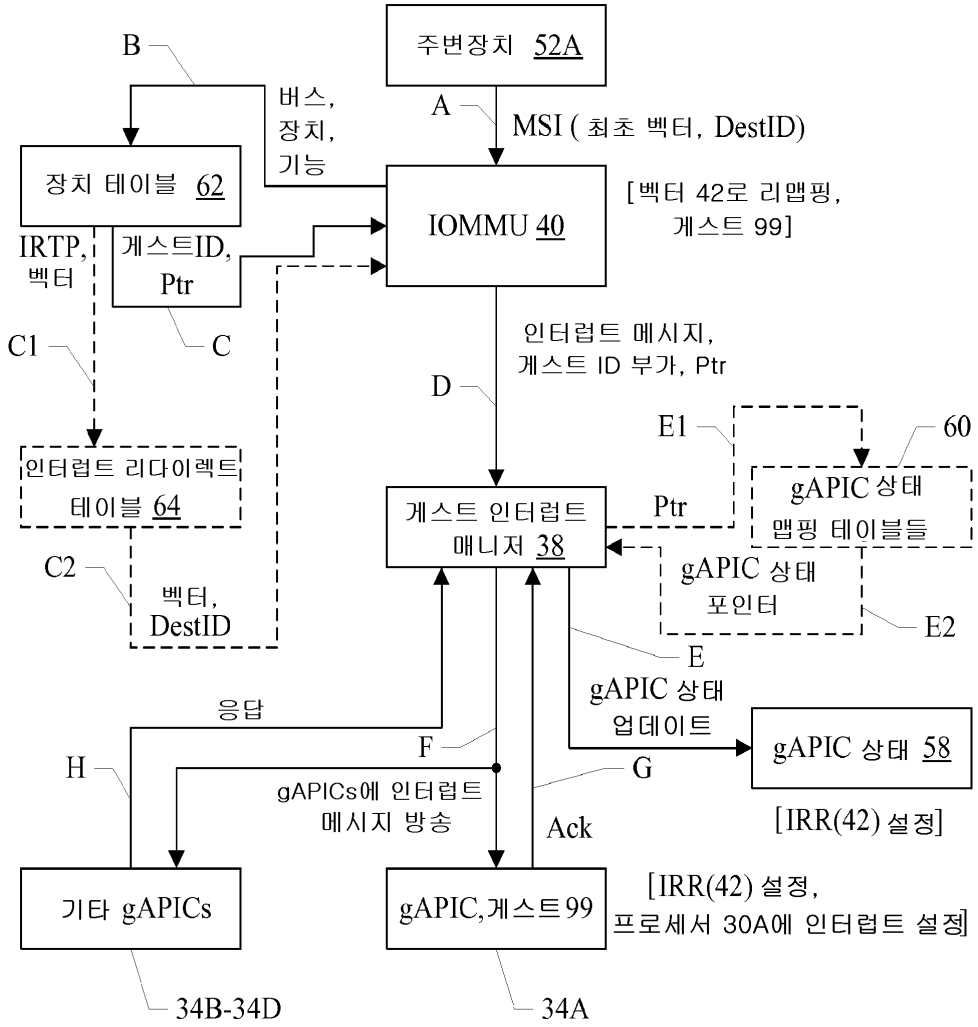
도면1


게스트 10A
App 14A
App 14N
게스트 OS 12
게스트 10N
특권 코드 16
VMM 18
VMCBs 22
호스트 하드웨어 20
5

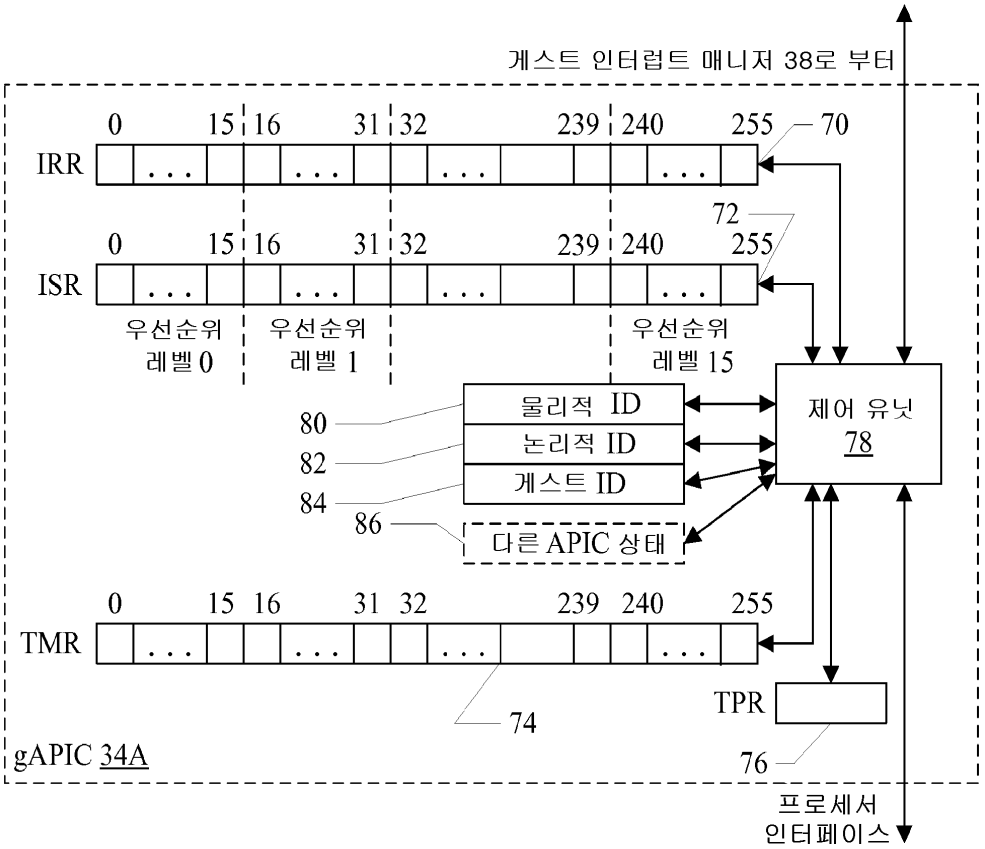
도면2


IC 66
프로세서 30A
프로세서 30B
gAPIC 34C
gAPIC 34A
hAPIC 32A
hAPIC 32B
gAPIC 34B
gAPIC 34D
게스트 인터럽트 매니저 38
IOMMU 40
MC 42
브리지 36
44A
44B
44C
IF
IF
IF
46
MIF
브리지 48
IOAPIC
50
주변장치 52A
IOAPIC
54
주변장치 52B
메모리 56
gAPIC 상태 58
60
gAPIC 상태 맵핑 테이블
장치 테이블 62
인터럽트 리다이렉트 테이블 64
VMM-관리되는 gAPIC 상태 68
20

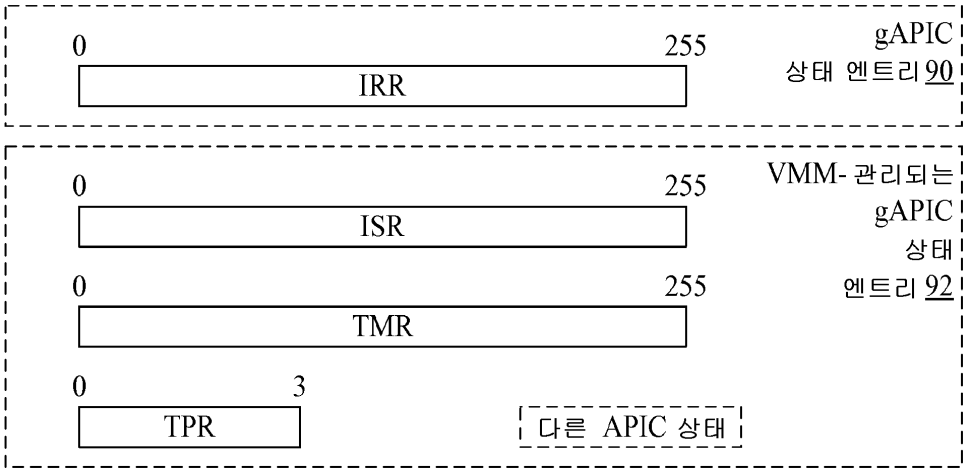
도면3


주변장치 52A
B
버스,
장치,
기능
A
MSI (최초 벡터, DestID)
장치 테이블 62
IOMMU 40
[벡터 42로 리맵핑,
게스트 99]
IRTP,
벡터
게스트ID,
Ptr
C
C1
인터럽트 메시지,
게스트 ID 부가, Ptr
D
E1
60
gAPIC 상태
맵핑 테이블들
인터럽트 리다이렉트
테이블 64
게스트 인터럽트
매니저 38
Ptr
C2
벡터,
DestID
gAPIC 상태
포인터
E2
E
gAPIC 상태
업데이트
응답
F
gAPIC 상태 58
H
G
gAPICs에 인터럽트
메시지 방송
Ack
[IRR(42) 설정]
기타 gAPICs
gAPIC,게스트 99
[IRR(42) 설정,
프로세서 30A에 인터럽트 설정]
34B-34D
34A

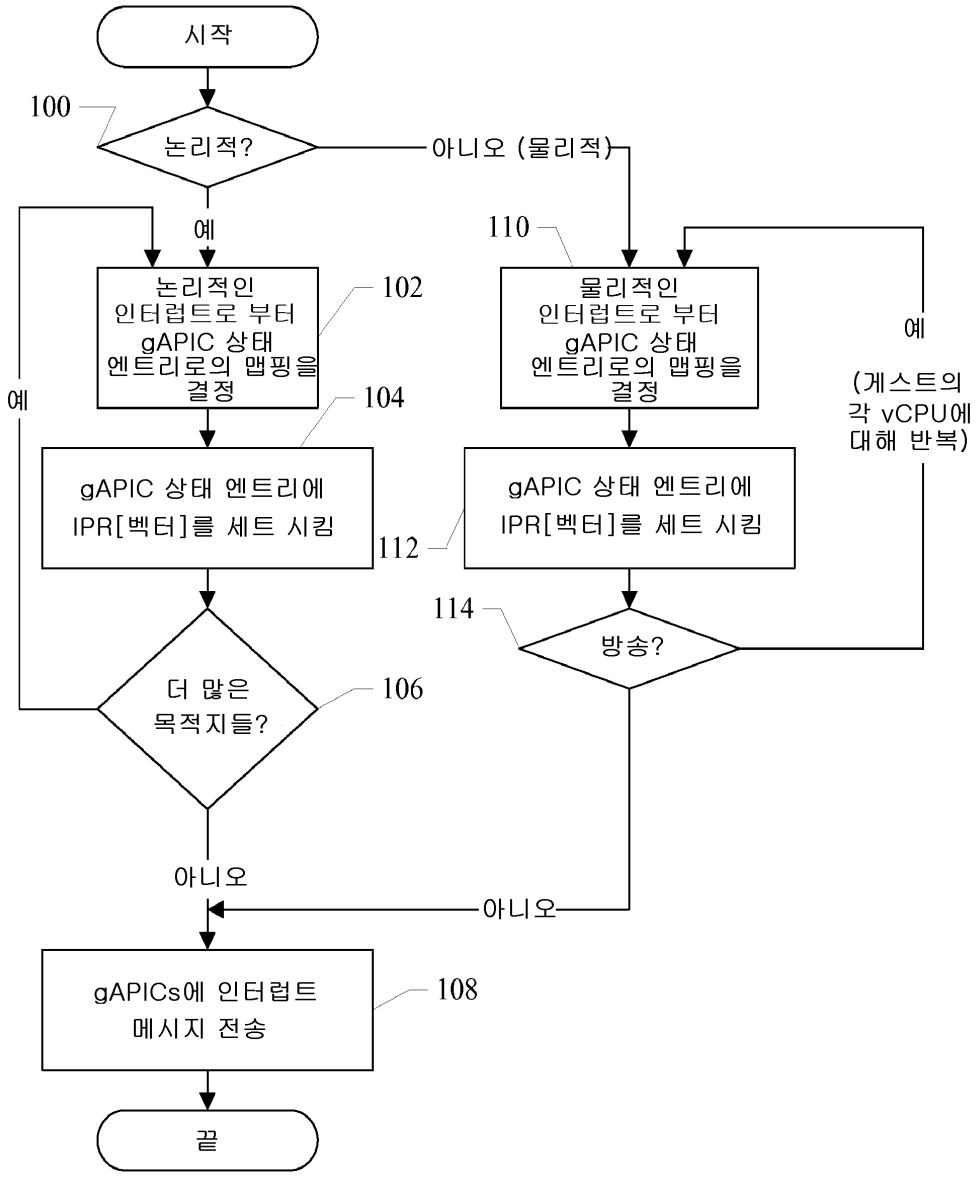
도면4


게스트 인터럽트 매니저 38로 부터
0
15
16
31
32
239
240
255
70
IRR
72
ISR
우선순위 레벨 0
우선순위 레벨 1
우선순위 레벨 15
물리적 ID
논리적 ID
게스트 ID
다른 APIC 상태
80
82
84
86
제어 유닛 78
TMR
74
TPR
76
gAPIC 34A
프로세서 인터페이스

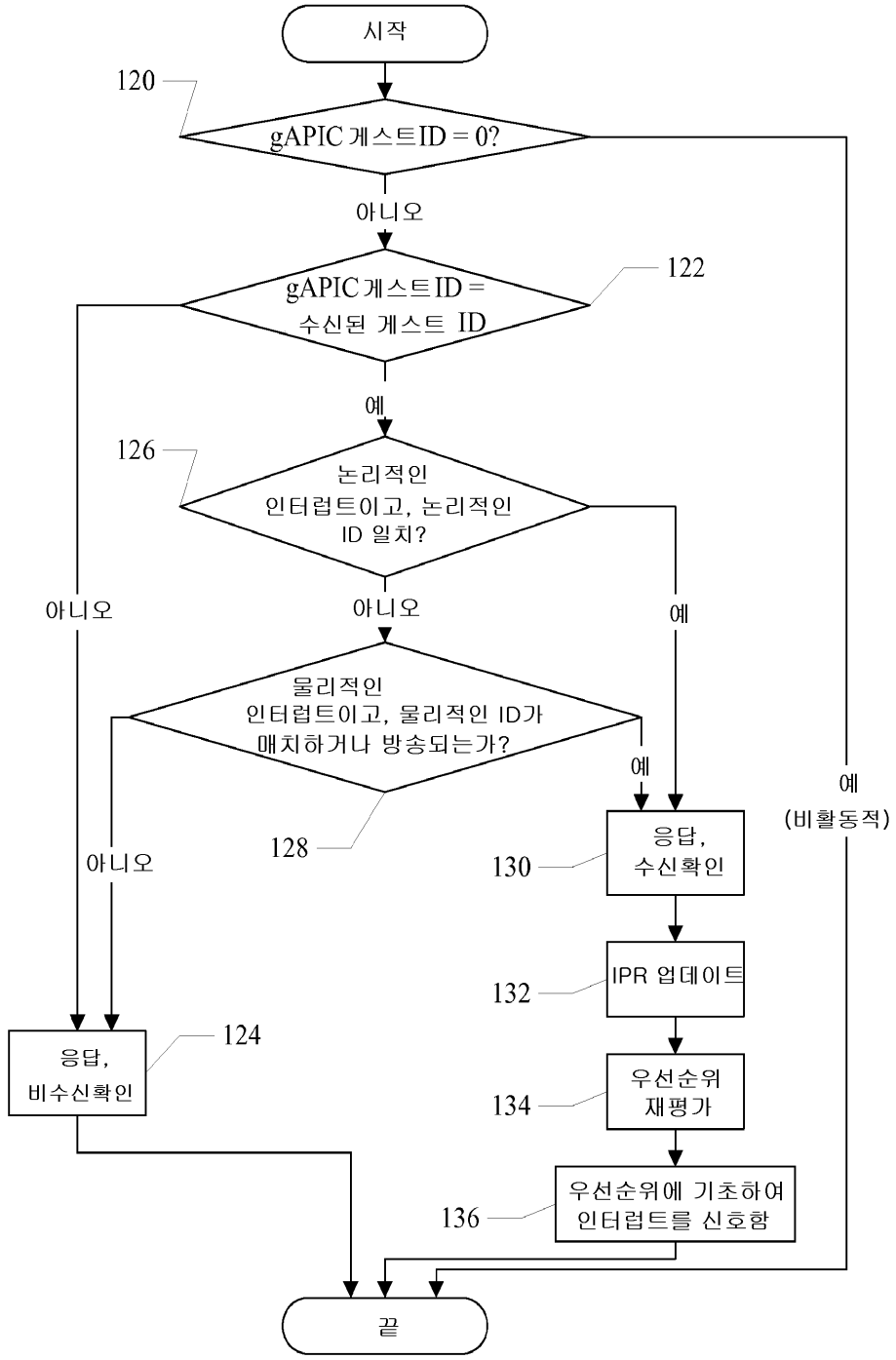
도면5


0
255
IRR
gAPIC 상태 엔트리 90
ISR
TMR
VMM-관리되는 gAPIC 상태 엔트리 92
3
TPR
다른 APIC 상태

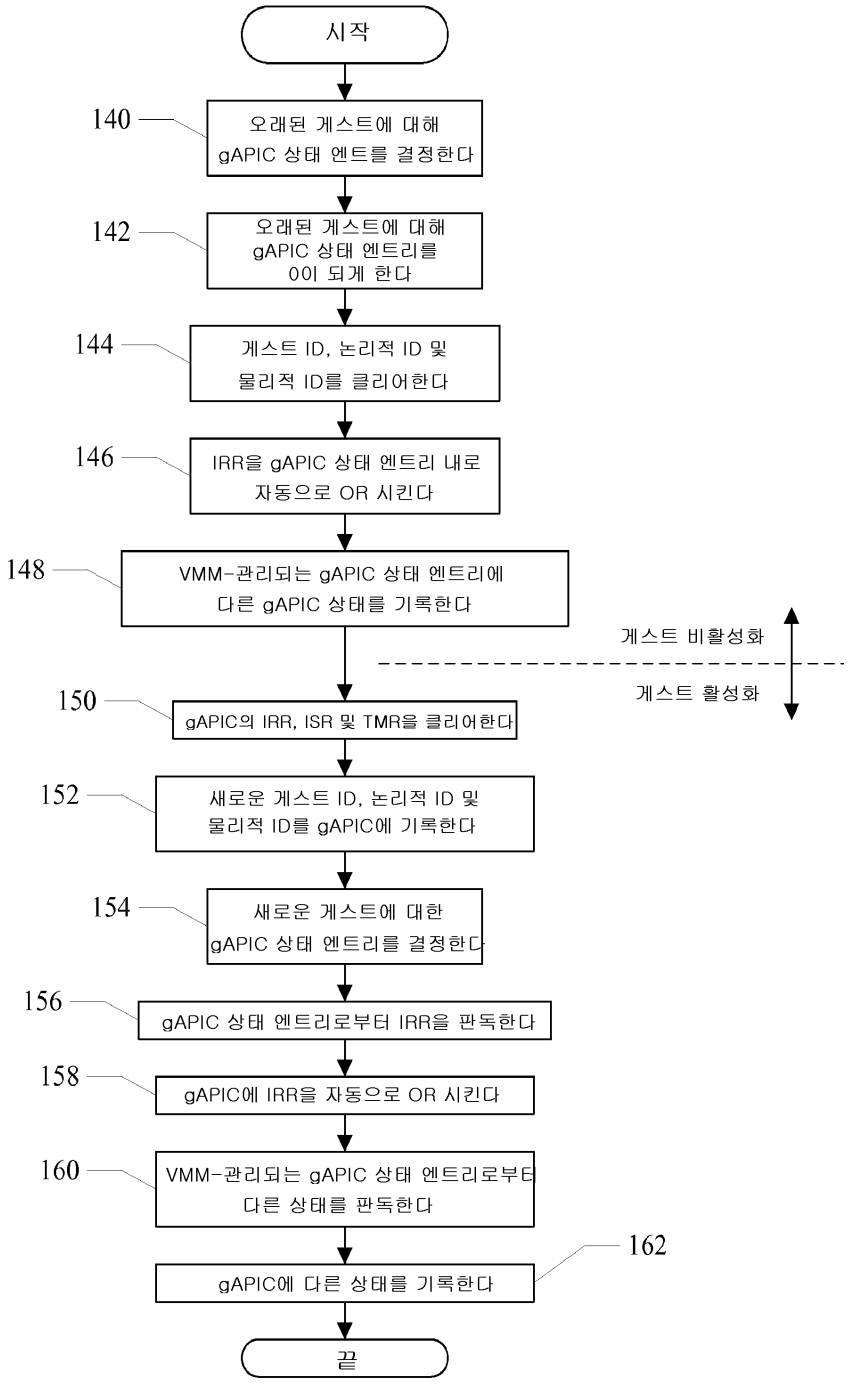
도면6


시작
100
논리적?
아니오 (물리적)
예
110
논리적인
인터럽트로 부터
gAPIC 상태
엔트리로의 맵핑을
결정
102
물리적인
인터럽트로 부터
gAPIC 상태
엔트리로의 맵핑을
결정
예
(게스트의
각 vCPU에
대해 반복)
예
104
gAPIC 상태 엔트리에
IPR[벡터]를 세트 시킴
gAPIC 상태 엔트리에
IPR[벡터]를 세트 시킴
112
114
방송?
더 많은
목적지들?
106
아니오
아니오
gAPICs에 인터럽트
메시지 전송
108
끝

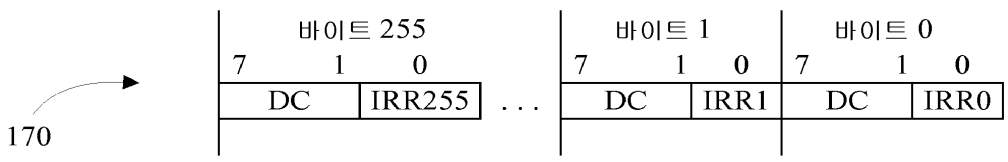
도면7


시작
120
gAPIC 게스트ID = 0?
아니오
gAPIC 게스트ID =
수신된 게스트 ID
122
예
126
논리적인
인터럽트이고, 논리적인
ID 일치?
아니오
아니오
예
물리적인
인터럽트이고, 물리적인 ID가
매치하거나 방송되는가?
예
128
아니오
130
응답,
수신확인
132
IPR 업데이트
134
우선순위
재평가
136
우선순위에 기초하여
인터럽트를 신호함
예
(비활동적)
124
응답,
비수신확인
끝

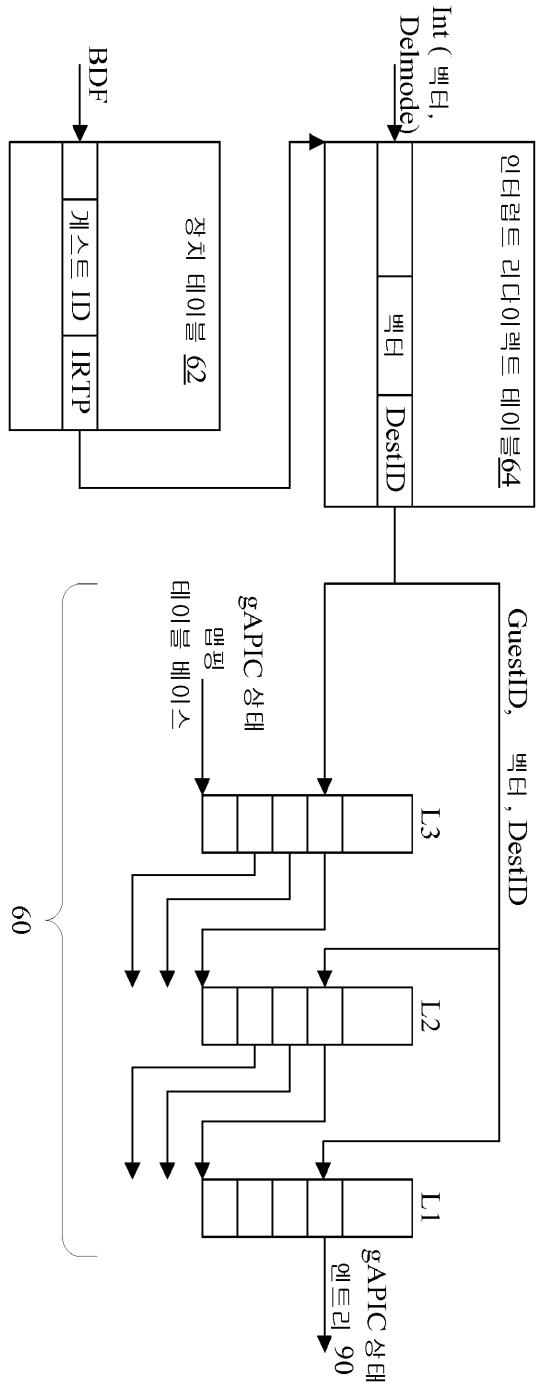
도면8


시작
140
오래된 게스트에 대해 gAPIC 상태 엔트를 결정한다
142
오래된 게스트에 대해 gAPIC 상태 엔트리를 0이 되게 한다
144
게스트 ID, 논리적 ID 및 물리적 ID를 클리어한다
146
IRR을 gAPIC 상태 엔트리 내로 자동으로 OR 시킨다
148
VMM-관리되는 gAPIC 상태 엔트리에 다른 gAPIC 상태를 기록한다
게스트 비활성화
게스트 활성화
150
gAPIC의 IRR, ISR 및 TMR을 클리어한다
152
새로운 게스트 ID, 논리적 ID 및 물리적 ID를 gAPIC에 기록한다
154
새로운 게스트에 대한 gAPIC 상태 엔트리를 결정한다
156
gAPIC 상태 엔트리로부터 IRR을 판독한다
158
gAPIC에 IRR을 자동으로 OR 시킨다
160
VMM-관리되는 gAPIC 상태 엔트리로부터 다른 상태를 판독한다
162
gAPIC에 다른 상태를 기록한다
끝

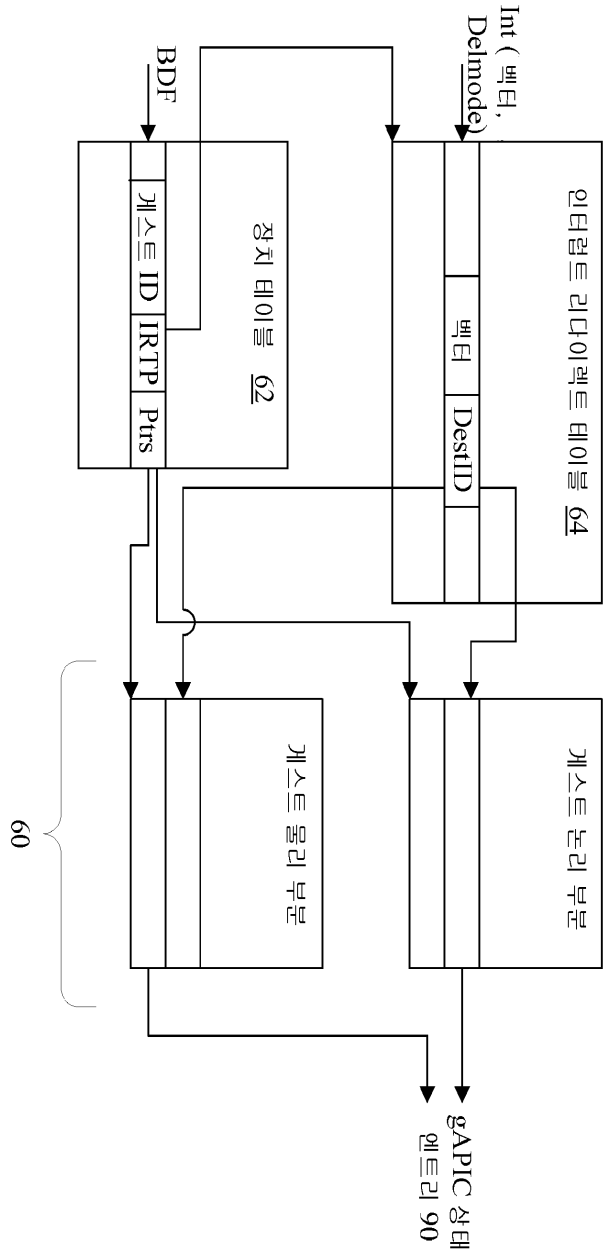
도면9


170
바이트 255
7 1 0
DC
IRR255
. . .
바이트 1
7 1 0
DC
IRR1
바이트 0
7 1 0
DC
IRR0

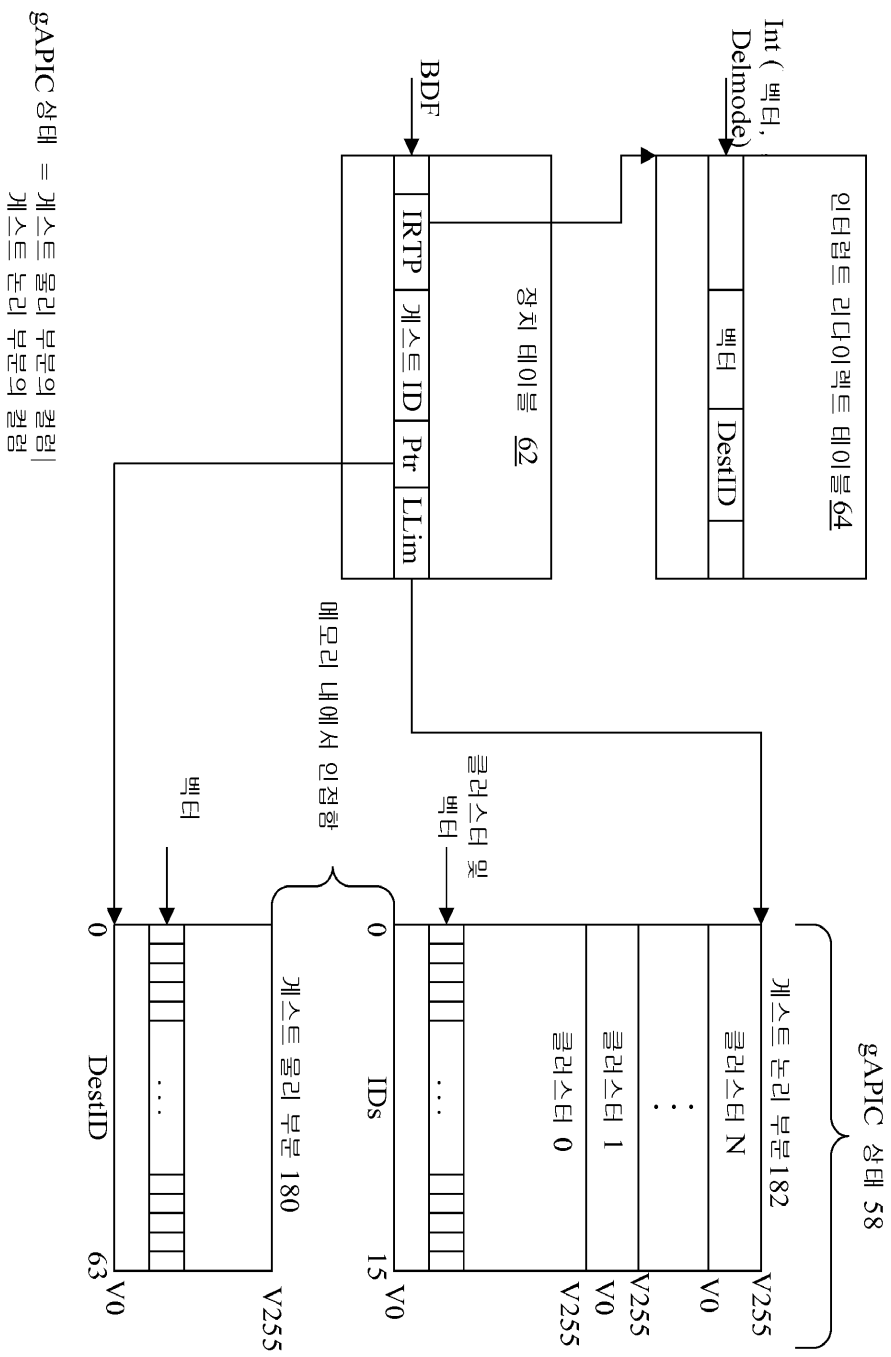
도면10


Int (벡터, Delmode)
BDF
인터럽트 리다이렉트 테이블64
벡터
DestID
장치 테이블 62
게스트 ID
IRTP
GuestID, 벡터, DestID
gAPIC 상태 맵핑 테이블 베이스
L3
L2
L1
gAPIC 상태 엔트리 90
60

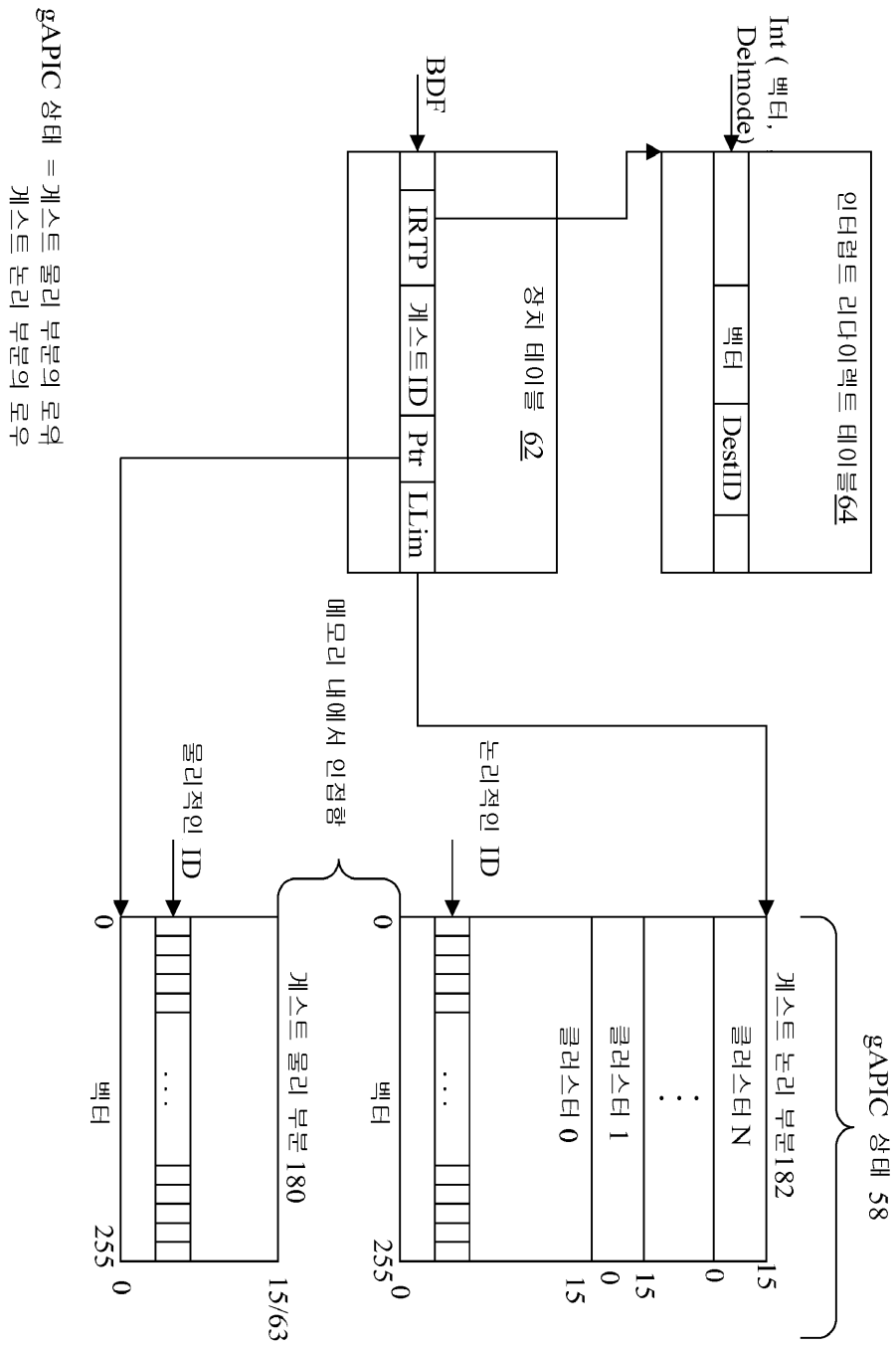
도면11


Int (벡터, Delmode)
BDF
인터럽트 리다이렉트 테이블 64
벡터
DestID
장치 테이블 62
게스트 ID
IRTP
Ptrs
게스트 논리 부분
게스트 물리 부분
60
gAPIC 상태 엔트리 90

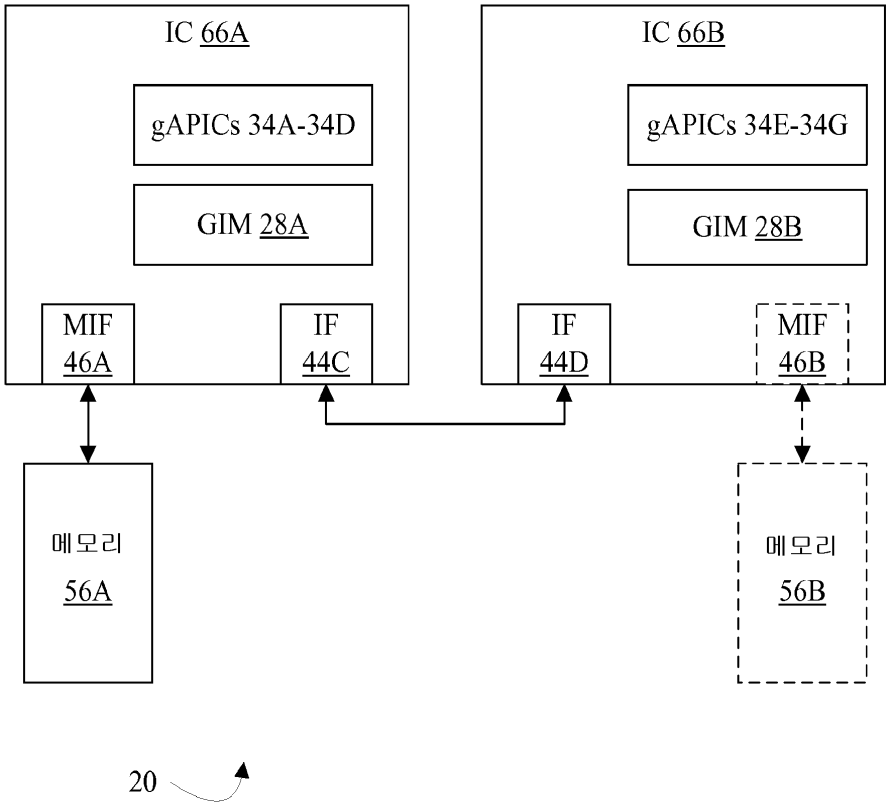
도면12


gAPIC 상태 = 게스트 물리 부분의 컬럼 | 게스트 논리 부분의 컬럼
Int (벡터, DelMode)
BDF
IRTP
게스트 ID
Ptr
LLim
장치 테이블 62
인터럽트 리다이렉트 테이블 64
벡터
DestID
메모리 내에서 인접함
클러스터 및 벡터
벡터
0
DestID
63
V0
게스트 물리 부분 180
V255
0
IDs
15
V0
클러스터 0
V255
클러스터 1
V0
클러스터 N
V255
게스트 논리 부분182
gAPIC 상태 58

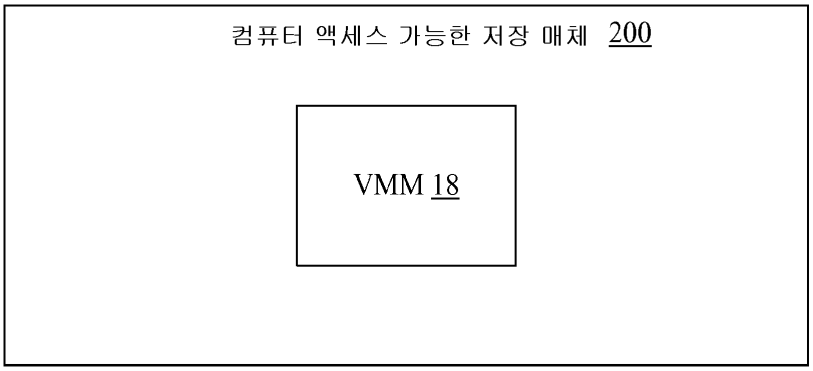
도면13


gAPIC 상태 = 게스트 물리 부분의 모음
게스트 논리 부분의 모음
BDF
Int (벡터, Delmode)
인터럽트 리다이렉트 테이블64
벡터
DestID
장치 테이블 62
IRTP
게스트ID
Ptr
LLim
메모리 내에서 인접함
논리적인 ID
물리적인 ID
gAPIC 상태 58
게스트 논리 부분182
클러스터 N
클러스터 1
클러스터 0
벡터
게스트 물리 부분 180
0
255
15
63

도면14


IC 66A
gAPICs 34A-34D
GIM 28A
MIF 46A
IF 44C
IC 66B
gAPICs 34E-34G
GIM 28B
IF 44D
MIF 46B
메모리 56A
메모리 56B
20

도면15


컴퓨터 액세스 가능한 저장 매체 200
VMM 18