(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2010년12월20일
(11) 등록번호 10-1002911
(24) 등록일자 2010년12월14일

(51) Int. Cl.
*G06F 9/46* (2006.01)
(21) 출원번호 10-2005-7006201
(22) 출원일자(국제출원일자) 2003년10월10일
심사청구일자 2008년09월10일
(85) 번역문제출일자 2005년04월09일
(65) 공개번호 10-2005-0050126
(43) 공개일자 2005년05월27일
(86) 국제출원번호 PCT/US2003/032322
(87) 국제공개번호 WO 2004/036354
국제공개일자 2004년04월29일
(30) 우선권주장
10/404,175 2003년04월01일 미국(US)
60/418,455 2002년10월15일 미국(US)
(56) 선행기술조사문헌
US20020103847 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
**아스펜 액퀴지션 코포레이션**
미국 캘리포니아 샌디에고 모어하우스 드라이브 5775 (우:92121)
(72) 발명자
**호케넥, 에르뎀**
미국, 뉴욕 10598, 요크타운 하이츠, 페어뷰 코트 3426
**모우드길, 마얀**
미국, 뉴욕 10607, 화이트 플레인즈, 주니퍼 힐 로드 143
**도워드, 션 엠.**
미국, 뉴저지 08836, 마틴스빌, 월드론 드라이브 3
(74) 대리인
**강명구**

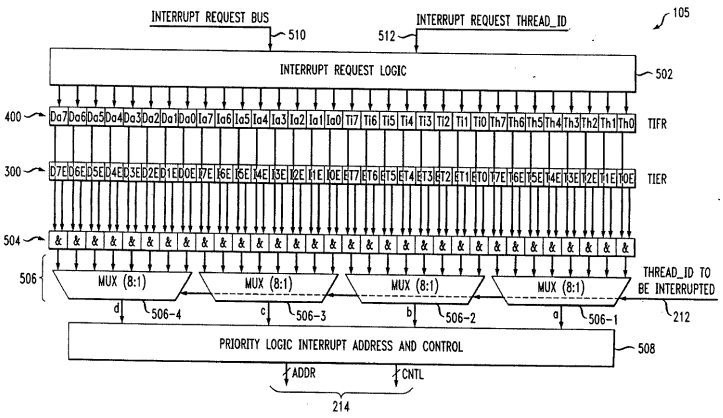
전체 청구항 수 : 총 19 항 심사관 : 임영희

# (54) 멀티스레디드 프로세서에서 고속 크로스-스레드 인터럽트를위한 방법 및 장치

## (57) 요 약

본 발명의 멀티스레디드 프로세서는 요청 스레드로부터 목적지 스레드로 향하는 크로스-스레드 인터럽트의 프로세싱을 위한 인터럽트 컨트롤러를 포함한다. 상기 인터럽트 컨트롤러는 실시예에서, 상기 크로스-스레드 인터럽트를 상기 목적지 스레드로 전달하기 위한 요청을 수신하고, 상기 크로스-스레드의 목적지 스레드가 크로스-스레드 인터럽트의 수신이 가능한지를 결정하며, 그리고 상기 목적지 스레드가 크로스-스레드 인터럽트의 수신이 가능할 경우, 상기 목적지 스레드로 상기 크로스-스레드 인터럽트의 전달을 제어하기 위해 스레드 식별기를 이용한다. 상기 요청 스레드는 상기 멀티스레디드 프로세서의 플래그 레지스터에서 상응하는 인터럽트 보류 비트를 설정함으로써 상기 크로스-스레드 인터럽트를 상기 목적지 스레드로 전달하는 것을 요청한다. 만일 상응하는 이네이블 비트가 상기 멀티스레디드 프로세서의 이네이블 레지스터 내에 설정된다면, 상기 목적지 스레드는 크로스-스레드 인터럽트의 수신이 가능하다. 상기 플래그 및 이네이블 레지스터는 상기 인터럽트 컨트롤러 내에 구현될 수 있다.

## 대 표 도



## *특허청구의 범위*

**청구항 1**

멀티스레디드 프로세서에서 요청 스레드로부터 목적지 스레드로 향하는 크로스-스레드 인터럽트를 프로세싱하는 방법에 있어서, 상기 크로스-스레드 인터럽트 프로세싱 방법은

- 인터럽트 컨트롤러에서, 상기 요청 스레드로부터 상기 목적지 스레드로의 크로스-스레드 인터럽트의 전달에 대한 요청을 수신하는 단계,

- 상기 인터럽트 컨트롤러에서, 상기 크로스-스레드 인터럽트의 목적지 스레드가 상기 요청 스레드로부터 크로스-스레드 인터럽트의 수신을 위해 활성화되어 있는지 여부를 판단하는 단계, 그리고

- 상기 목적지 스레드가 크로스-스레드 인터럽트의 수신에 대해 활성화된 경우, 상기 인터럽트 컨트롤러가 상기 요청 스레드로부터 상기 목적지 스레드로 크로스-스레드 인터럽트가 전달되도록 제어하는 단계를 포함하되,

상기 요청 스레드는 단일 크로스-스레드 인터럽트 레지스터의 제 1 비트를 설정(set)함으로써, 상기 인터럽트 컨트롤러에 상기 목적지 스레드로의 크로스-스레드 인터럽트의 전달을 요청하며,

상기 크로스-스레드 인터럽트 레지스터는 상기 요청 스레드와 상기 목적지 스레드를 포함하는 다수의 스레드에 의해 공유되어, 상기 목적지 스레드가 상기 공유되는 크로스-스레드 인터럽트 레지스터의 제 2 비트를 설정(set)함으로써, 상기 인터럽트 컨트롤러에, 상기 목적지 스레드로부터 상기 요청 스레드로의 추가적인 크로스-스레드 인터럽트의 전달을 요청하도록 동작가능한 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 방법.

**청구항 2**

제 1 항에 있어서, 이때 상기 요청 스레드는 상기 멀티스레디드 프로세서의 플래그 레지스터에서 상응하는 인터럽트 보류 비트를 설정함으로써 상기 목적지 스레드로 상기 크로스-스레드 인터럽트의 전달을 요청하는 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 방법.

**청구항 3**

제 2 항에 있어서, 이때 상기 인터럽트 보류 비트는 상기 크로스-스레드 인터럽트가 상기 목적지 스레드로 전달될 때 자동적으로 소거되는 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 방법.

**청구항 4**

제 1 항에 있어서, 이때 상기 목적지 스레드는 상응하는 이네이블 비트가 상기 멀티스레디드 프로세서의 이네이블 레지스터 내에 설정될 때 크로스-스레드 인터럽트의 수신이 가능한 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 방법.

**청구항 5**

제 4 항에 있어서, 이때 상기 이네이블 비트는 상기 크로스-스레드 인터럽트가 상기 목적지 스레드에 전달될 때 자동적으로 소거됨으로써, 상기 목적지 스레드에 의하여 다중 인터럽트의 동시 수신을 예방하는 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 방법.

**청구항 6**

제 5 항에 있어서, 이때 상기 이네이블 비트는 상기 크로스-스레드 인터럽트가 상기 목적지 스레드에 의해 제공될 때 자동적으로 소거되는 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 방법.

**청구항 7**

제 1 항에 있어서, 이때 상기 크로스-스레드 인터럽트에 대한 요청은 수신되고, 상기 크로스-스레드 인터럽트는 상기 멀티스레디드 프로세서의 명령어 실행 사이클보다 작은 지속시간을 갖는 주기 내에서 상기 목적지 스레드로 계속 전달되는것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 방법.

**청구항 8**

삭제

**청구항 9**

제 1 항에 있어서, 이때 상기 인터럽트 컨트롤러는 이네이블 레지스터, 플래그 레지스터 및 로직 회로를 포함하는 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 방법.

**청구항 10**

제 9 항에 있어서, 이때 상기 이네이블 레지스터는 N개 이상의 부분을 포함하고, 각 부분은 상기 멀티스레디드 프로세서의 N 개의 스레드 중 하나에 상응하는 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 방법.

**청구항 11**

제 10 항에 있어서, 이때 상기 각각의 스레드는 상기 이네이블 레지스터의 상응하는 부분에만 액세스가 허용되고, 다른 스레드에 상응하는 상기 이네이블 레지스터의 N-1 개의 부분에는 액세스가 허용되지 않는 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 방법.

**청구항 12**

제 9 항에 있어서, 이때 상기 플래그 레지스터는 N개 이상의 부분을 포함하고, 각각은 상기 멀티스레디드 프로세서의 N개의 스레드 중 하나에 상응하는 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 방법.

**청구항 13**

제 12 항에 있어서, 이때 상기 스레드 각각은 상기 플래그 레지스터의 N개의 부분 각각에 액세스가 허용되는 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 방법.

**청구항 14**

멀티스레디드 프로세서에서 요청 스레드로부터 목적지 스레드로 향하는 크로스-스레드 인터럽트를 프로세싱하는 장치에 있어서, 상기 크로스-스레드 인터럽트 프로세싱 장치는

- 로직 회로를 포함하는 인터럽트 컨트롤러로서, 이때 상기 인터럽트 컨트롤러는 (ⅰ)상기 요청 스레드로부터 상기 크로스-스레드 인터럽트를 상기 목적지 스레드로 전달하기 위한 요청을 수신하고, (ii)상기 크로스-스레드의 목적지 스레드가 상기 요청 스레드로부터 크로스-스레드 인터럽트를 수신하기 위해 활성화되었는지 여부를 결정하며, 그리고 (iii)상기 목적지 스레드가 크로스-스레드 인터럽트를 수신할 수 있는 경우, 상기 요청 스레드로부터 상기 목적지 스레드로 상기 크로스-스레드 인터럽트를 전달하도록 제어하는 상기 인터럽트 컨트롤러를 포함하며,

상기 요청 스레드는, 단일 크로스-스레드 인터럽트 레지스터의 제 1 비트를 설정함으로써, 상기 인터럽트 컨트롤러에 목적지 스레드로의 크로스-스레드 인터럽트의 전달을 요청하며,

상기 크로스-스레드 인터럽트 레지스터는 요청 스레드와 목적지 스레드를 포함하는 다수의 스레드에 의해 공유되어, 상기 목적지 스레드는 상기 공유되는 크로스-스레드 인터럽트 레지스터의 제 2 비트를 설정함으로써, 상기 인터럽트 컨트롤러에 상기 목적지 스레드로부터 상기 요청 스레드로의 추가적인 크로스-스레드 인터럽트의 전달을 요청할 수 있는 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 장치.

**청구항 15**

제 14 항에 있어서, 이때 상기 인터럽트 컨트롤러는 이네이블 레지스터 및 플래그 레지스터를 추가로 포함하고, 이때 상기 요청 스레드는 상기 플래그 레지스터에서 상응하는 인터럽트 보류 비트를 설정함으로써 상기 목적지 스레드로 상기 크로스-스레드 인터럽트의 전달을 요청하며, 상기 목적지 스레드는 만일 상응하는 이네이블 비트가 상기 이네이블 레지스터 내에 설정될 경우 크로스-스레드 인터럽트의 수신이 가능한 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 장치.

**청구항 16**

제 15 항에 있어서, 이때 상기 로직 회로는

인터럽트 요청을 처리하기 위한 인터럽트 요청 로직,

다수의 로직 게이트로서, 이때 각각의 로직 게이트는 입력으로서, 상기 이네이블 레지스터 및 플래그 레지스터로부터 각각 상응하는 비트를 수신하는 상기 다수의 로직 게이트, 및

멀티플렉싱 회로로서, 이때 상기 멀티플렉싱 회로는 상기 스레드 식별기를 바탕으로 상기 크로스-스레드 인터럽트의 생성을 제어함에 있어서, 로직 게이트의 출력들 중에서 하나이상의 출력을 선택하는 상기 멀티플렉싱 회로

를 포함하는 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 장치.

**청구항 17**

제 16 항에 있어서, 이때 상기 로직 회로는 타이머 인터럽트, 명령어 어드레스 인터럽트 및 데이터 어드레스 인터럽트 중 하나이상을 포함하는 인터럽트의 하나이상의 추가 형태에 비해 상기 크로스-스레드 인터럽트를 우선시키는 우선 순위 로직을 추가로 포함하는 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 장치.

**청구항 18**

제 17 항에 있어서, 이때 상기 우선순위 로직은 상기 크로스-스레드 인터럽트에 상기 추가 인터럽트 형태보다 높은 우선순위 또는 낮은 우선순위를 제공하도록 구성되는 것을 특징으로 하는 크로스-스레드 인터럽트 프로세싱 장치.

**청구항 19**

멀티스레디드 프로세서에 있어서, 상기 멀티스레디드 프로세서는

- 멀티스레디드 프로세서에서 요청 스레드로부터 목적지 스레드로 향하는 크로스-스레드 인터럽트를 처리하기 위한 인터럽트 컨트롤러를 포함하되,

상기 인터럽트 컨트롤러는 (ⅰ)상기 요청 스레드로부터, 상기 크로스-스레드 인터럽트를 상기 목적지 스레드로 전달하기 위한 요청을 수신하고, (ⅱ)상기 크로스-스레드의 목적지 스레드가 상기 요청 스레드로부터 크로스-스레드 인터럽트를 수신하도록 활성화되었는지 여부를 결정하며, 그리고 (ⅲ)상기 목적지 스레드가 크로스-스레드 인터럽트의 수신이 가능할 경우, 상기 요청 스레드로부터 상기 목적지 스레드로 상기 크로스-스레드 인터럽트가 전달되도록 제어하고,

상기 요청 스레드는, 단일 크로스-스레드 인터럽트 레지스터의 제 1 비트를 설정함으로써, 상기 인터럽트 컨트롤러에 목적지 스레드로의 크로스-스레드 인터럽트의 전달을 요청하며,

상기 크로스-스레드 인터럽트 레지스터는 요청 스레드와 목적지 스레드를 포함하는 다수의 스레드에 의해 공유되어, 상기 목적지 스레드가 상기 공유되는 크로스-스레드 인터럽트 레지스터의 제 2 비트를 설정함으로써, 상기 인터럽트 컨트롤러에 상기 목적지 스레드로부터 상기 요청 스레드로의 추가적인 크로스-스레드 인터럽트의 전달을 요청하는 것을 특징으로 하는 멀티스레디드 프로세서.

**청구항 20**

멀티스레디드 프로세서에서 요청 스레드에서 목적지 스레드로 향하는 크로스-스레드 인터럽트를 프로세싱하기 위해 사용되는 프로그램이 기록된 컴퓨터 판독가능형 기록 매체에 있어서, 상기 프로그램은,

- 인터럽트 컨트롤러에서, 상기 요청 스레드로부터 상기 목적지 스레드로의 크로스-스레드 인터럽트의 전달에 대한 요청을 수신하는 단계,

- 상기 인터럽트 컨트롤러에서, 상기 크로스-스레드 인터럽트의 목적지 스레드가 상기 요청 스레드로부터 상기 크로스-스레드 인터럽트의 수신을 위해 활성화되어 있는가의 여부를 판단하는 단계, 그리고

- 상기 목적지 스레드가 크로스-스레드 인터럽트의 수신에 대해 활성화된 경우, 상기 인터럽트 컨트롤러는 상기 요청 스레드로부터 목적지 스레드로 크로스-스레드 인터럽트를 전달하도록 제어하는 단계

를 실행하며, 상기 요청 스레드는 단일 크로스-스레드 인터럽트 레지스터의 제 1 비트를 설정함으로써, 상기 인터럽트 컨트롤러에 목적지 스레드로의 크로스-스레드 인터럽트의 전달을 요청하며,

상기 크로스-스레드 인터럽트 레지스터는 요청 스레드와 목적지 스레드를 포함하는 다수의 스레드에 의해 공유되어, 상기 목적지 스레드가 상기 공유되는 크로스-스레드 인터럽트 레지스터의 제 2 비트를 설정함으로써, 상기 인터럽트 컨트롤러에 상기 목적지 스레드로부터 상기 요청 스레드로의 추가적인 크로스-스레드 인터럽트의 전달을 요청하는 것을 특징으로 하는 멀티스레디드 프로세서에서 요청 스레드에서 목적지 스레드로 향하는 크로스-스레드 인터럽트를 프로세싱하기 위해 사용되는 프로그램이 기록된 컴퓨터 판독가능형 기록 매체.

## 명 세 서

### 기 술 분 야

[0001] 본 발명은 디지털 데이터 프로세서의 분야에 관한 것으로, 특히 멀티스레디드 프로세서에서 사용하기 위한 인터럽트 기술에 관한 것이다.

### 배 경 기 술

[0002] 컴퓨터 시스템은 전통적으로 프로세서에 예상하지 못하거나 드문 상태를 알리기 위해 인터럽트 메커니즘을 사용하였다. 예를 들어, 무효한 연산, 네트워크 패킷의 도달, 및 디스크 액세스의 종료 등은 일반적인 컴퓨터 상에서 인터럽트를 일으킨다. 프로세서의 일반적인 동작에 포함되지 않은 특별한 하드웨어는 상태를 검출하고 프로세서에 신호를 보낸다. 인터럽트를 수신할 때, 상기 프로세서는 현재의 작업을 중단하고, 상태를 조정하기 위해 필요한 단계를 수행하며, 이후 정상적인 실행으로 복원한다. 이러한 인터럽트는 드문 경우에 대하여 계속적으로 체크하거나 폴링(polling)할 필요가 없이, 컴퓨터가 상기 드문 경우에 빠르게 응답하도록 한다.

[0003] 멀티-프로세서 시스템은 일반적으로 하나의 프로세서가 또 다른 프로세서를 인터럽트하게 하는 인터-프로세서(inter-processor) 인터럽트를 지원한다. 인터-프로세서 인터럽트는 서로 다른 프로세서의 동작을 동기화하는데 사용되고 또한 시스템이나 실행중인 프로그램에서의 상태 변화를 프로세서에 알려주는데 사용된다. 예를 들어, 다중 프로세서에서 실행중인 프로그램의 상태가 변할 때, 이러한 변화는 상기 변화를 검출한 프로세서상에 반영되어야 하고 또한 상기 프로그램을 실행하는데 관련된 모든 프로세서에 반영되어야 한다. 추가적으로, 인터-프로세서 인터럽트는 프로세서에 인터-프로세서 메시지가 도달하였음을 알려주는데 사용될 수 있다.

[0004] 인터-프로세서 인터럽트의 전달 및 수신은 비교적 빠르지만, 여전히 많은 프로세서 클럭 사이클이 걸린다. 인터럽트를 송수신하는데 사용된 인터럽트 컨트롤러는 흔히 상기 프로세서와 분리되고, 조작을 위해 멀티-사이클 동작을 필요로 한다. 또한, 하나의 인터럽트 컨트롤러로부터 또 다른 인터럽트 컨트롤러로 인터럽트를 실제로 송신하는 것은 가령, 프로세서가 일반적으로 종래의 인터럽트 컨트롤러보다 빠른 클럭 속도로 실행하기 때문에 수개의 프로세서 클럭 사이클을 필요로 할 수 있다.

[0005] 멀티스레디드 프로세서는 다수의 개별 명령어 시퀀스 또는 스레드의 동시 실행을 지원하는 프로세서이다. 프로세싱 스레드는 단일 스레드보다는 상기 프로세서에서 보다 많은 연산 능력을 사용하도록 한다. 따라서 멀티-프로세서 시스템에서보다 많은 동기화 및 통신에 이르게 되고, 이로써 크로스-스레드(cross-thread) 인터럽트(즉, 하나의 스레드에서 또 다른 스레드로의 인터럽트)와 같은 통신 메커니즘의 비용에 높은 비중을 두게 된다.

[0006] "하이퍼-스레딩(hyper-threading)" 기술을 이용하는 인텔 프로세서와 같은 현재의 멀티스레디드 프로세서는 앞서 설명된 인터-프로세서 인터럽트를 처리하는데 사용된 것을 모방한 기술을 이용하여 크로스-스레드 인터럽트를 처리한다. 결과적으로, 크로스-스레드 인터럽트를 처리하기 위한 종래의 기술은 매우 느리고, 프로세싱 자원에서 상당한 오버헤드를 필요로 할 수 있다. 예를 들어, 하나의 스레드가 종래의 기술을 이용하여 또 다른 인터럽트를 처리하는데 상당한 수의 명령어 실행 사이클이 필요하다.

[0007] 전술한 바와 같이, 멀티스레디드 프로세서에서는 향상된 크로스-스레드 인터럽트 프로세싱 기술의 필요성이 존재한다.

### 발명의 상세한 설명

[0008] 본 발명은 멀티스레디드 프로세서에서 크로스-스레드 인터럽트를 처리하기 위한 향상된 기술을 제공한다.

[0009] 본 발명의 한 태양에서, 멀티스레디드 프로세서는 인터럽트 컨트롤러를 포함함으로써, 요청 스레드로부터 목적지(destination) 스레드로 향하는 크로스-스레드 인터럽트를 처리하게 된다.

[0010] 도시된 실시예에서 인터럽트 컨트롤러는 상기 크로스-스레드 인터럽트를 목적지 스레드로 전달하기 위한 요청을

수신하고, 상기 크로스-스레드 인터럽트의 목적지 스레드가 크로스-스레드 인터럽트를 수신할 수 있는지를 결정하며, 그리고 만일 상기 목적지 스레드가 크로스-스레드 인터럽트를 수신할 수 있다면 상기 크로스-스레드 인터럽트를 상기 목적지 스레드로 전달하는 것을 제어하도록 스레드 식별기를 이용한다.

[0011] 상기 요청 스레드는 상기 멀티스레디드 프로세서의 플래그 레지스터(flag register)에서 상응하는 인터럽트 보류(pending) 비트를 설정함으로써 상기 크로스-스레드 인터럽트를 상기 목적지 스레드로 전달하는 것을 요청한다. 상기 인터럽트 보류 비트는 상기 크로스-스레드 인터럽트가 상기 목적지 스레드로 전달될 때 자동적으로 소거될 수 있다.

[0012] 만일 상응하는 이네이블(enable) 비트가 상기 멀티스레디드 프로세서의 이네이블 레지스터 내에 설정된다면, 상기 목적지 스레드는 크로스-스레드 인터럽트의 수신이 가능하다. 상기 이네이블 비트는 상기 크로스-스레드 인터럽트가 상기 목적지 스레드로 전달될 때 자동적으로 소거될 수 있어서, 상기 목적지 스레드에 의한 다중 인터럽트의 동시 수신을 예방할 수 있고, 이후 상기 목적지-스레드에 의한 다른 인터럽트의 수신을 리-이네이블(re-enable)하도록 상기 크로스-스레드 인터럽트가 제공된 후 재설정하게 된다.

[0013] 도시된 실시예에서 상기 이네이블 레지스터 및 플래그 레지스터는 각각 적어도 N개의 부분을 포함하고, 상기 N개 부분의 각각은 상기 멀티스레디드 프로세서의 N개의 스레드 중 하나에 상응한다. 상기 N개 스레드 각각은 상기 이네이블 레지스터의 상응하는 부분에만 액세스가 허용되며, 다른 스레드에 상응하는 이네이블 레지스터의 N-1개의 부분에는 액세스가 허용되지 않는다.

[0014] 본 발명의 또 다른 태양에 따르면, 인터럽트 컨트롤러 내의 로직 회로는 인터럽트 요청을 처리하기 위한 인터럽트 요청 로직, 다수의 로직 게이트, 및 멀티플렉싱 회로를 포함할 수 있고, 이때 상기 다수의 로직 게이트들의 각각은 입력으로서 상기 각각의 이네이블 레지스터 및 상기 플래그 레지스터로부터 상응하는 비트를 수신한다. 상기 멀티플렉싱 회로는 부분적으로 스레드 식별기를 바탕으로, 상기 다수의 로직 게이트의 출력들 중에서, 상기 크로스-스레드 인터럽트의 생성을 제어하기 위하여 하나이상의 출력을 선택한다.

[0015] 상기 인터럽트 컨트롤러의 로직 회로는 가령, 타이머 인터럽트, 명령어 어드레스 인터럽트, 외부적으로-생성된 인터럽트 등과 같은 다른 형태의 인터럽트에 비하여 상기 크로스-스레드 인터럽트를 우선하는 우선 순위 로직을 추가로 포함할 수 있다.

[0016] 본 발명은 멀티스레디드 프로세서에서 크로스-스레드 인터럽트를 전달하고 제공하도록 낮은-레이턴시(latency), 낮은-오버헤드 메커니즘을 제공하는 것이 선호된다. 예를 들어, 앞서 언급된 실시예에서, 주어진 스레드는 단일 명령어 실행 사이클 내에서 또 다른 스레드를 인터럽트할 수 있다.

## 실 시 예

[0023] 본 발명은 멀티스레디드 프로세서에서 주 메모리, 멀티스레디드 캐시 메모리, 및 멀티스레디드 데이터 메모리가 관련되는 것으로 도시될 것이다. 그러나, 본 발명은 특별한 멀티스레디드 프로세서 및 실시예의 메모리 구조를 반드시 사용할 필요는 없으며, 크로스-스레드 인터럽트의 향상된 프로세싱을 제공하는 것이 선호되는 멀티스레디드 프로세서 인터럽트 프로세싱 용도에 사용하기에 적합하다.

[0024] 본 발명에 따른 크로스-스레드 인터럽트 프로세싱 기술을 구현하는 예시적 프로세싱 시스템(100)은 도 1 및 2와 결합하여 설명될 것이다.

[0025] 도 1의 프로세싱 시스템(100)은 주 메모리(104)에 연결된 멀티스레디드 프로세서(102)를 포함한다. 상기 멀티스레디드 프로세서(102)는 인터럽트 컨트롤러(105), 멀티스레디드 캐시 메모리(110) 및 멀티스레디드 데이터 메모리(112)를 포함한다.

[0026] 본 발명은 한 태양에서 멀티스레디드 프로세서(102)에서 크로스-스레드 인터럽트를 처리하는 향상된 기술을 제공한다. 실시예에서, 이러한 기술은 인터럽트 컨트롤러(105) 내에서 주로 구현된다. 상기 인터럽터 컨트롤러(105)는 이 실시예에서 도 1과 같이 상기 멀티스레디드 프로세서(102) 내에 통합되는 것이 선호된다.

[0027] 본 발명의 기술에 따른 상기 인터럽트 컨트롤러(105)의 구조가 크로스-스레드 인터럽트의 효율적 처리를 목적으로 하지만, 또한 당업자에 공지된 종래의 기술을 이용하여 데이터 또는 명령어 어드레스 인터럽트, 타이머 인터럽트, 외부적으로-생성된 인터럽트 등과 같은 다른 형태의 인터럽트를 처리하도록 구성될 수 있다. 상기 외부적으로-생성된 인터럽트의 예로는 상기 멀티스레디드 프로세서(102)의 외부에 있는 특별한 디바이스나 상기 프로세서의 특별한 입력 핀에 관련된 인터럽트를 들 수 있다.

[0028] 인터럽트 컨트롤러(105)는 따라서 본원에서 설명되는 특별한 인터럽트 컨트롤러 요소 이외에 다양한 종래의 요소를 포함할 수 있다.

[0029] 도 2는 상기 멀티스레디드 프로세서(102)의 가능한 하나의 구현예를 상세히 보여주고 있다. 이 실시예에서, 상기 멀티스레디드 프로세서(102)는 인터럽트 컨트롤러(105), 멀티스레디드 캐시 메모리(110), 데이터 메모리(112), 캐시 컨트롤러(114), 명령어 디코더(116), 레지스터 파일(118), 및 산술 연산 유닛(ALU)(120)을 포함한다. 상기 멀티스레디드 캐시 메모리(110)는 또한 멀티스레디드 캐시로 일컬어진다.

[0030] 도 1 및 2에 도시된 특별한 배열은 도해의 목적으로 단순화되어 있고, 도시되지 않은 추가 요소 및 다른 대안 요소는 당업자에게 분명하기 때문에 포함될 수 있다.

[0031] 상기 인터럽트 컨트롤러(105)는 로직 회로(202) 및 인터럽트 컨트롤러 레지스터(204)를 포함한다. 실시예에서 상기 레지스터(204)는 도 3과 4와 결합하여 각각 아래에서 설명될 이네이블 및 플래그 레지스터를 포함한다. 상기 로직 회로(202) 및 상기 레지스터(204)와의 상호작용은 도 5와 결합하여 아래에 상세히 설명될 것이다.

[0032] 상기 인터럽트 컨트롤러(105)는 시스템 버스(206)를 통해 상기 멀티스레디드 프로세서(102)의 다른 소자에 연결될 수 있다.

[0033] 실시예에서 브랜치(branch) 컨트롤러(210)는 상기 인터럽트 컨트롤러(105)와 관련된다. 상기 브랜치 컨트롤러(210)는 캐시 컨트롤러(114) 및 인터럽트 컨트롤러(105)에 연결되고, 또한 버스(206)나 다른 도면에 표시되지 않은 적절한 연결을 통해 상기 멀티스레디드 프로세서(102)의 하나이상의 다른 소자와 통신할 수 있다. 도면에서 분리된 소자로 표시되어 있지만, 상기 브랜치 컨트롤러(210)의 기능은 전체적으로 혹은 부분적으로 상기 인터럽트 컨트롤러(105)나 상기 멀티스레디드 프로세서(102)의 다른 소자 내에 구현될 수 있다. 상기 브랜치 컨트롤러(210) 또는 관련된 인터럽트 컨트롤러(105)는 또한 종래의 스레드 실행 컨트롤러와 관련된 기능과 같은 다른 기능을 구현할 수 있다.

[0034] 멀티스레디드 캐시(110)는 다수의 스레드 캐시(110-1, 110-2,...110-N)를 포함하고, 이때 N은 상기 멀티스레디드 프로세서(102)에 의해 지원된 스레드의 수를 표시한다. 따라서 각각의 스레드는 상기 멀티스레디드 캐시(110) 내에 각각의 스레드와 관련된 상응하는 스레드 캐시를 갖는다. 유사하게, 상기 데이터 메모리(112)는 N개의 데이터 메모리, 즉 112-1, 112-2,...112-N을 포함한다.

[0035] 상기 멀티스레디드 캐시(110) 내의 각 스레드 캐시는 하나이상의 메모리 위치 세트를 갖는 메모리 어레이를 포함할 수 있다. 주어진 스레드 캐시는 관련된 스레드 식별기를 저장하기 위한 스레드 식별기 레지스터를 추가로 포함할 수 있다.

[0036] 멀티스레디드 캐시(110)는 상기 캐시 컨트롤러(114)를 통해 주 메모리(104)와 인터페이스한다. 캐시 컨트롤러(114)는 주 메모리(104)로부터의 적절한 명령어가 상기 멀티스레디드 캐시(110)로 로딩되는 것을 보장한다. 상기 캐시 컨트롤러(114)는 이 실시예에서, 각각의 스레드 캐시 110-1, 110-2,...110-N과 관련된 논리회로 또는 다른 프로세싱 요소와 결합하여 동작하면서, 어소시에이티브 매핑(associative mapping), 다이렉트(direct) 매핑, 또는 세트-어소시에이티브(set-associative) 매핑 등과 같은 어드레스 매핑 기술의 적어도 일부분을 구현한다. 본 발명과 결합하여 사용하기에 적합한 세트-어소시에이티브 매핑 기술은 미국 특허 출원 10/161,774 및 10/161,874에 설명되어 있으며, 둘다 2002년 6월 4일에 출원되었고 본원에서 참조로 인용된다.

[0037] 일반적으로, 상기 멀티스레디드 캐시(110)는 상기 멀티스레디드 프로세서(102)에 의해 실행될 명령어를 저장하는데 사용되고, 반면 데이터 메모리(112)는 상기 명령어에 의해 동작되는 데이터를 저장한다. 명령어는 종래의 방식으로 명령어의 실행을 제어함에 있어서, 레지스터 파일(118) 및 ALU(120)와 결합하여 동작하는 명령어 디코더(116)에 의해 상기 멀티스레디드 캐시(110)로부터 인출된다. 116, 118, 및 120과 같은 멀티스레디드 프로세서 요소의 동작은 당분야에 공지되어 있기 때문에 본원에서는 추가로 상세히 설명되지 않는다.

[0038] 상기 데이터 메모리(112)는 주 메모리(104)에 직접 연결되어 있지만, 도면에서는 나타나지 않는다.

[0039] 메모리(104, 110, 및 112) 중 하나이상은 다중 뱅크 또는 다른 지정된 부분을 포함하도록 각각 구성될 수 있다. 실시예에서, 각각의 뱅크는 하나이상의 메모리 모듈, 또는 단일 메모리의 특정 부분으로 구성되는 것으로 보일 수 있다.

[0040] 멀티스레디드 프로세서와 관련된 이러한 메모리 및 다른 메모리들의 스레드-방식 뱅킹 기술은 2002년 10월 11일 출원된 미국 특허 출원 10/269,247 "Method and Apparatus for Thread-Based Memory Access in a

Multithreaded Processor"에 설명되어 있고, 본원에서 참조로 인용된다.

[0041] 본원에서 사용되는 용어 "메모리"는 내부 메모리나 외부 메모리, 캐시 메모리, 데이터 메모리, 또는 다른 데이터 저장 요소의 배열을 포함하도록 의도된다. 본 발명은 특정 메모리 형태, 구조 또는 용도에 제한되지 않는다. 그러나, 메모리들은 일반적으로 프로세서에서 도 2의 레지스터 파일(118)을 포함하는 것과 같은 레지스터와는 별개의 것으로 이해된다.

[0042] 레지스터 파일에의 스레드-방식 액세스에 대한 기술은 2002년 10월 11일 출원된 미국 특허 출원 10/269,373 "Method and Apparatus for Thread-Based Memory Access in a Multithreaded Processor"에 설명되어 있고, 본원에서 참조로 인용된다.

[0043] 본 발명은 도 2에 도시된 특별한 멀티스레디드 프로세서 구조를 필요로 하지는 않는다. 본 발명은 다양한 다른 멀티스레디드 프로세서 구조에서 구현될 수 있다.

[0044] 도 2에 도시되어 있고 본 발명과 결합하여 사용하기에 적합한 형태의 멀티스레디드 프로세서의 보다 많은 특별한 예는 2002년 10월 11일에 출원되었고 본원에서 참조로 인용되는 미국 특허 출원 10/269,372 "Multithreaded Processor With Efficient Processing For Convergence Device Applications"에 설명되어 있다. 미국 특허 출원 번호 10/269,372에 설명된 멀티스레디드 프로세서의 실시예는 RISC-방식 제어 코드, 디지털 신호 프로세서(DSP) 코드, 자바 코드 및 네트워크 프로세싱 코드를 실행할 수 있다. 상기 프로세서는 SIMD(single instruction multiple data) 벡터 유닛, 리덕션(reduction) 유닛, 및 LIW(long instruction word) 복합 명령어 실행을 포함한다.

[0045] 상기 멀티스레디드 프로세서(102)는 토큰 트리거 방식 스레딩으로 일컬어지는 스레딩 접근 또는 다른 적절한 스레딩 기술을 이용하도록 구성될 수 있다. 파이프라이닝은 또한 이러한 스레딩과 결합하여 사용되는 것이 선호된다. 본 발명에서 사용하기 적합한 스레딩 및 파이프라이닝 기술의 예는 2002년 10월 11일에 출원되었고 본원에서 참조로 인용되는 미국 특허 출원 10/269,245 "Method and Apparatus for Token Triggered Multithreading"에 설명되어 있다.

[0046] 멀티스레디드 프로세서에서, 주어진 스레드는 하드웨어 및 소프트웨어에 의해 보여질 수 있다. 주어진 스레드와 관련된 특별한 프로세서 하드웨어는 흔히 하드웨어 스레드 유닛 또는 단순히 "콘텍스트"로 일컬어진다. 본원에서 사용되는 용어 "thread"는 소프트웨어 또는 하드웨어 스레드 또는 둘다를 포함하도록 의도된다.

[0047] 상기 브랜치 컨트롤러(210)와 상기 인터럽트 컨트롤러(105) 및 멀티스레디드 프로세서(102)의 다른 소자와의 상호작용은 이제 상세히 설명될 것이다. 일반적으로, 상기 브랜치 컨트롤러(210)는 각 스레드가 어떤 명령어를 실행 중인지를 규정하는 정보를 유지하고, 그리고 디코딩 및 후속 실행을 위해 어떤 명령어가 상기 명령어 디코더(116)로 보내어져야 하는지에 관하여 상기 스레드 캐시의 방향을 설정한다. 상기 브랜치 컨트롤러(210)는 따라서 각각의 스레드에 대하여 분리된 프로그램 카운터를 유지할 것이며, 또한 각각의 일반적인 명령어, 브랜치 명령어, 인터럽트 등이 이루어진 후에 상기 프로그램 카운터를 업데이트하는 메커니즘을 유지한다.

[0048] 도 2에는, 상기 브랜치 컨트롤러(210)와 상기 인터럽트 컨트롤러(105) 사이에 두 개의 연결(212 및 214)이 있다. 이러한 연결의 보다 상세한 예는 도 5에 도시되어 있으며, 아래에서 더욱 상세히 설명될 것이다. 상기 브랜치 컨트롤러(210)는 상기 연결(212)을 이용하여 "실행할 다음(next to execute) 스레드"에 대한 인터럽트가 있는지에 관하여 상기 인터럽트 컨트롤러(105)에 조회를 한다. 상기 인터럽트 컨트롤러(105)는 상기 조회에 대한 응답을 연결(214)을 통해 상기 브랜치 컨트롤러(210)에 되돌린다. 도 5에서, 연결(212)은 인터럽트될 특정 스레드를 식별하도록 구체화되는 반면, 연결(214)은 특정 인터럽트가 있을 경우 이 특정 인터럽트와 관련된 어드레스 및 제어 출력을 포함한다.

[0049] 앞서 언급된 바와 같이, 본 발명은 한 태양에서 상기 멀티스레디드 프로세서(102)에서 사용되는 향상된 크로스-스레드 인터럽트 프로세싱 기술을 제공한다.

[0050] 본 발명에서, 상기 멀티스레디드 프로세서(102)와 같은 멀티스레디드 프로세서는 하나의 스레드가 단일 명령어 실행 사이클에 있는 또 다른 스레드를 인터럽트하도록 구성된다. 아래에서 설명될 것이지만, 상기 인터럽트 컨트롤러(105)를 상기 멀티스레디드 프로세서(102)로 통합하고, 인터럽트 컨트롤러 레지스터(204)를 상기 프로세싱 스레드로부터 빠르게 액세스 가능하게 함으로써, 전달에 필요한 시간, 서비스, 또는 다른 프로세스 크로스-스레드 인터럽트는 매우 감소된다.

[0051] 도 3 및 4는 본 발명에 따른 인터럽트 컨트롤러(105)와 관련된 예시적 이네이블 및 플래그 레지스터를 각각 보

여준다. 도 3 및 4의 레지스터는 도 2에 도시된 레지스터(204)의 특별한 예로 보여질 수 있다. 아래에서 상술될 것이지만, 이러한 예시적인 인터럽트 컨트롤러 레지스터는 어떤 인터럽트가 어떤 주어진 시간에 어떤 스레드로 전달될 것인가를 제어하고, 그리고 단일 명령어 실행 사이클에서 판독되거나 수정될 수 있다. 앞서 언급된 바와 같이, 이러한 레지스터는 상기 멀티스레디드 프로세서(102)와 통합되는 것이 선호된다. 예를 들어, 이러한 인터럽트 컨트롤러 레지스터는 소위 "특별한 목적의" 레지스터로서, 상기 인터럽트 컨트롤러(105) 내에 통합되거나 혹은 상기 멀티스레디드 프로세서 내에 통합될 수 있다. 그러나, 다른 형태의 레지스터가 또한 본 발명의 기술을 구현하는데 사용될 수 있다.

[0052] 도 3에서 5와 결합하여 설명된 실시예에서, 스레드의 수 N이 8이 되도록 가정될 것이다. 물론, 본 발명은 이러한 특별한 수의 스레드를 필요로 하지는 않는다. 설명된 배열은 당업자에 분명하도록 다른 수의 스레드를 수용할 수 있는 방식으로 변경될 수 있다.

[0053] 도 3에는, TIER(thread interrupt enable register)(300)이 도시되어 있다. 상기 TIER은 각 스레드에 하나씩 8개의 4-비트 레지스터로 보여질 수 있지만, 이 실시예에서 상기 TIER은 단일 32-비트 레지스터를 이용하여 구현되어 있고 비트 0, 8, 16 및 24는 Thread1과 관련되고, 비트 1, 9, 17 및 25는 Thread1와 관련된다.

[0054] 상기 인터럽트 컨트롤러는 각각의 스레드가 상기 TIER의 상응하는 부분에 액세스할 수 있도록 구성된다. 상기 TIER은 상기 8개의 스레드 각각에 대하여, 데이터 어드레스(DADR) 인터럽트 이네이블 비트, 명령어 어드레스(IADR) 인터럽트 이네이블 비트, 타이머 인터럽트 이네이블 비트, 및 마스터 인터럽트 이네이블 비트를 포함한다. 상기 스레드들 중 주어진 하나에 대하여, 상기 마스터 인터럽트 이네이블 비트는 설정시 상기 스레드가 모든 인터럽트 소스로부터 인터럽트를 수신하도록 하고, 또한 소거시 상기 스레드가 모든 인터럽트 소스로부터 인터럽트를 수신하는 것을 예방한다. 설명의 단순화를 위해 본원에서 사용되는 상기 용어 "설정(set)" 및 소거(cleared)"는 비록 다른 가정이 사용될 수 있지만 각각 로직 0 및 로직 1 레벨을 의미한다.

[0055] 상기 TIER은 크로스-스레드 인터럽트를 이네이블하기 위한 특정 비트를 필요로 하지는 않는데, 그 이유는 도시된 실시예에서 상기 인터럽트들이 상기 멀티스레디드 프로세서(102) 상에서 실행 중인 소프트웨어에 의해 적어도 부분적으로 생성될 수 있기 때문이다.

[0056] 일반적으로, 상기 TIER은 주어진 스레드가 다양한 소스로부터의 상기 스레드에 의해 인터럽트의 수신을 가능하게 한다. 이 실시예에서, 상기 마스터 이네이블 비트를 상기 TIER의 상응하는 부분 내에 설정함으로써, 주어진 스레드는 다른 스레드로부터 크로스-스레드를 포함하여 모든 소스로부터 인터럽트들을 수신할 수 있도록 한다. 다른 배열은 본 발명에 따라 크로스-스레드 인터럽트를 이네이블하는데 사용될 수 있다.

[0057] 도 4는 이 실시예에서 8개의 스레드들 사이에서 공유되는 TIFR(thread interrupt flag register)(400)를 보여준다. 상기 TIFR은 각각의 스레드가 전체 레지스터에 액세스할 수 있는 스레드들 사이에서 공유된다. 상기 TIFR은 상응하는 스레드에만 액세스 가능한 개별 부분을 포함하는 상기 TIER과 대조적이다.

[0058] 상기 TIFR은 8개 스레드 각각에 대하여 본원에서 "인터럽트 보류(interrupt pending)"로도 일컬어지는 플래그 비트 세트를 포함한다. 특히, 상기 TIFR은 주어진 스레드에 대하여 총 네 개의 인터럽트 보류 비트를 포함하고, 상기 인터럽트 보류 비트는 데이터 어드레스(DADR) 인터럽트, 명령어 어드레스(IADR) 인터럽트, 타이머 인터럽트 및 크로스-스레드(XTHR) 인터럽트 각각에 대하여 분리되어 있다. 이 실시예에서 상기 TIFR은 또한 단일 32-비트 레지스터를 이용하여 구현되고, 비트 0, 8, 16, 및 24는 Thread1과 관련되며, 비트 1, 9, 17 및 25는 Thread2와 관련된다. 상기 TIFR에서, 주어진 인터럽트 보류 비트는 설정시 상응하는 인터럽트가 보류 중임을 표시하고, 소거시 상기 상응하는 인터럽트가 보류 중이지 않음을 나타낸다. 일반적으로, 주어진 스레드는 상기 TIFR 내의 상응하는 XTHR 인터럽트 보류 비트를 설정함으로써 또 다른 스레드에 대한 크로스-스레드 인터럽트를 설정한다.

[0059] 앞서 언급한 바와 같이, 본 발명의 상기 TIER 및 TIFR 레지스터는 도면에 주어진 특정 구조를 필요로 하지는 않는다. 예를 들어, 도 3과 4에 단일 레지스터로 도시되어 있지만, 상기 TIER 및 TIFR의 하나 또는 둘다는 각각 다중 레지스터, 가령 스레드 당 하나의 레지스터, 한 쌍의 스레드 당 하나의 레지스터 등으로 구현될 수 있다. 상기 TIER 및 TIFR의 수많은 다른 대안적 구조는 당분야에서 분명할 것이다.

[0060] 도 5는 본 발명의 실시예에서, 인터럽트 컨트롤러(105)를 상세히 보여주고 있다. 상기 인터럽트 컨트롤러(105)는 TIER(300) 및 TIFR(400)를 포함하고, 각각 앞서 설명된 방식으로 32-비트 레지스터로서 구현된다. 상기 인터럽트 컨트롤러(105)는 인터럽트 요청 로직(502)을 포함하는 로직 회로, 상기 실시예에서 서른 두 개의 2-입력 AND 게이트를 포함하는 로직 게이트(504), 멀티플렉싱 회로(506), 및 인터럽트 어드레스 및 컨트롤 출력을 생성

하기 위한 우선 순위(priority) 로직(508)을 추가로 포함한다.

[0061] 인터럽트 요청 로직(502)은 인터럽트 요청 버스(510)에 연결되고, 상기 인터럽트 요청 버스(510)는 도 2의 시스템 버스(206)의 적어도 한 부분을 나타낼 수 있다. 상기 인터럽트 요청 로직(502)은 또한 라인(512)을 통해 입력으로서 인터럽트 요청 스레드 식별기를 수신한다. 용어 "스레드 식별기"는 또한 Thread_ID로 표시된다. 상기 스레드들 중 주어진 하나에 대한 스레드 식별기는 앞서 언급된 상기 스레드에 대한 스레드 식별기 레지스터에 저장될 수 있다. 상기 실시예에서, N=8이고, 상기 스레드 식별기는 3-비트 식별기가 될 수 있다.

[0062] 일반적으로, 입력(510 및 512)은 상기 TIFR내의 어떤 비트가 설정되어야 하는가를 결정하는데 사용된다. 특히, 입력(512)을 통해 제공된 상기 인터럽트 요청 스레드 식별기는 요청된 목적지 스레드에 대하여 플래그 레지스터의 부분을 선택하도록 상기 인터럽트 컨트롤러(105)에 의해 사용되고, 그리고 상기 인터럽트 요청 버스(510)는 요청된 인터럽트의 특정 형태를 구체화하도록 정보를 제공한다.

[0063] 본원에서 사용되는 용어 "스레드 식별기"는 특정 스레드 또는 멀티스레디드 프로세서에서의 다중 스레드 세트를 식별하기에 적합한 정보를 포함하도록 의도된다. 실시예에 의해서 및 제한없이, 상기 스레드 식별기는 멀티스레디드 프로세서에서 스레드 카운터의 출력에 상응할 수 있다. 특히, 주어진 멀티스레디드 프로세서는 다중 스레드가 지정된 순서로 가령, 라운드 로빈 순서로 처리되도록 구성될 수 있고, 이때 상기 스레드 카운터 출력은 실행될 특정 스레드를 식별하는데 사용된다. 상기 실시예에서, 라운드 로빈 순서로 처리되는 총 8개의 스레드가 있을 수 있으며, 각각의 스레드는 3-비트 식별기에 의해 식별됨으로써, 3-비트 카운터의 출력은 처리 중인 특정 스레드를 식별하는데 사용될 수 있다. 다른 실시예는 상기 스레드 식별기의 넌-카운터(non-counter) 구현예를 이용할 수 있다. 본 발명에서 사용하기에 적합한 다양한 서로 다른 스레드 식별기 구조는 당업자에게 분명할 것이다.

[0064] 상기 멀티스레디드 프로세서(102)의 다양한 스레드는 인터럽트 요청 로직(502)을 통해 상기 TIER 및 TIFR에 액세스할 수 있다. 또 다른 스레드의 크로스-스레드 인터럽트를 요청하는 주어진 스레드는 인터럽트될 상기 스레드의 상기 인터럽트 요청 스레드 식별기를 입력(512)을 통해 상기 인터럽트 요청 로직(502)으로 공급한다. 크로스-스레드 인터럽트에 의해 인터럽트될 스레드는 또한 본원에서 목적지 스레드로 일컬어진다.

[0065] 상기 로직 게이트 세트(504)의 상기 각각의 2-입력 AND 게이트는 한 쌍의 비트를 입력으로서 하나는 TIER로부터 그리고 하나는 TIFR로부터 수신한다. 예를 들어, 상기 2-입력 AND 게이트의 좌측은 상기 TIER 및 TIFR의 서른 두 번째 비트(즉, 상기 TIER 및 TIFR의 DADR 부분 내의 비트 31로서 식별된 비트)를 입력으로서 수신한다. 상기 TIER 및 TIFR의 다른 비트들은 유사하게 상기 2-입력 AND 게이트에 쌍으로 인가된다.

[0066] 상기 AND 게이트의 출력은 멀티플렉싱 회로(506)의 입력에 인가된다. 특히, 네 개의 8-대-1 멀티플렉서(506-1, 506-2, 506-3 및 506-4) 각각은 상기 로직 게이트 세트(504) 내의 서른 두 개의 2-입력 AND 게이트 중 여덟 개의 출력을 입력으로서 수신한다. 상기 멀티플렉서(506-1, 506-2, 506-3 및 506-4)에 의해 수신된 입력은 각각 상기 TIER 및 TIFR의 비트 0-7, 비트 8-15, 비트 16-23 및 비트 24-31과 관련된다.

[0067] 상기 멀티플렉서 각각은 또한, 앞선 도 2와 결합하여 설명된 방식으로 상기 연결(212)을 통해 상기 브랜치 컨트롤러(210)로부터 상기 인터럽트 컨트롤러(105)로 공급되는 것처럼, 실행할 다음 스레드의 스레드 식별기를 선택 신호 입력으로서 수신한다. 상기 스레드 식별기는 실행할 다음 스레드가 어떤 보류 인터럽트를 갖는지에 관하여 앞서-언급된 조회와 결합하여 제공된다. 상기 인가된 "실행할 다음" 스레드 식별기를 바탕으로, 각각의 멀티플렉서는 여덟 개의 입력 중 선택된 하나를 상기 우선 순위 로직(508)으로 출력한다. 상기 우선 순위 로직(508)은 상기 입력을 이용하여 상응하는 인터럽트에 대하여 적절한 어드레스 및 제어 출력을 생성한다.

[0068] 아래에서 상술될 것이지만, 만일 "실행할 다음" 스레드에 대하여 보류 인터럽트가 있다면, 앞서-언급된 조회에 대한 응답으로 상기 인터럽트 컨트롤러(105)는 상기 마스터 인터럽트 이네이블 비트가 특별한 목적지 스레드에 대하여 설정되어 있는지를 체크한다. 이후, 만일 상기 마스터 인터럽트 이네이블 비트가 설정된다면, 상기 인터럽트 컨트롤러(105)는 적절한 제어 출력 및 상응하는 인터럽트 핸들러(handler) 어드레스를 상기 연결(214)을 통해 상기 브랜치 컨트롤러에 전달한다.

[0069] 도 6은 인터럽트 컨트롤러(105)에서 크로스-스레드의 처리와 관련된 예시적 동작을 보여주는 흐름도(600)이다. 크로스-스레드 인터럽트를 전달하기 위해서, 요청 스레드는 단계(602)에서 상기 TIFR내의 목적지 스레드에 상응하는 XTHR 비트를 설정한다. "실행할 다음" 스레드의 스레드 식별기를 바탕으로, 상기 멀티플렉서(506-1, 506-2, 506-3 및 506-4)는 상응하는 2-입력 AND 게이트의 출력을 상기 우선 순위 로직(508)으로 전달한다.

[0070] 만일 상기 목적지 스레드가 상기 TIER 내에 설정된 마스터 이네이블 비트를 가진다면, 상응하는 AND 게이트의

출력은 로직 하이 레벨에 있을 것이고, 이러한 배열은 상기 멀티플렉서의 동작과 결합하여 상기 우선 순위 로직(508)에 알려주는 메커니즘을 제공함으로써, 단계(604)에 표시된 바와 같이 다음 사용가능한 기회에서 상기 목적지 스레드로 전달하기 위한 크로스-스레드 인터럽트를 생성하게 된다.

[0071] 실시예에서, 상기 다음 사용가능한 기회는 상기 목적지 스레드가 명령어를 실행하게 되는 다음 시간을 발생시킬 것이다. 앞서 언급된 바와 같이, 상기 목적지 스레드는 또한 마스터 이네이블 비트 설정을 가져야 한다. 단계(606)에서, 상기 인터럽트는 아래에서 상술되겠지만, 상기 우선 순위 로직(58)으로부터 어드레스 및 제어 정보의 형태로 상기 목적지 스레드로 전달된다.

[0072] 인터럽트가 상기 목적지 스레드로 전달된 후, 마스터 이네이블 비트는 단계(608)에서 표시된 바와 같이 자동으로 소거된다. 이는 상기 목적지 스레드에 의하여 다수의 인터럽트의 동시 수신을 예방한다. 상기 인터럽트는 이후 단계(610)에서 표시된 대로, 상기 목적지 스레드에 의해 제공된다. 상기 인터럽트가 제공된 후에, 상기 목적지 스레드의 마스터 이네이블 비트는 단계(612)에 표시된 대로, 상기 목적지 스레드에 대한 인터럽트를 리-이네이블하도록 자동적으로 설정된다. 상기 프로세스는 또 다른 크로스-스레드 인터럽트를 처리하도록 단계(602)로 되돌아간다.

[0073] 도 6의 흐름도에서 상기 마스터 이네이블 비트의 자동 소거 및 설정은 본 발명의 요건 보다 일반적으로 생각된다. 대안적 실시예에서, 이러한 기능은 다른 기술을 이용하여 구현될 수 있었다. 예를 들어, 상기 마스터 이네이블 비트는 적절한 소프트웨어 명령어에 의해 분명히 설정되거나 소거될 수 있다.

[0074] 도면에 도시되지는 않았지만, 상기 TIFR 내의 상기 XTHR 인터럽트 보류 비트를 자동적으로 소거하기 위한 동작은 단계(606) 이후의 프로세서에 포함될 수 있지만, 상기 비트는 인터럽트가 제공될 때 리-이네이블되지 않아야 된다.

[0075] 앞서 표시된 바와 같이, 상기 우선 순위 로직(508)은 상기 멀티플렉싱 회로(506)의 출력을 바탕으로한 어드레스 및 제어 정보를 생성한다. 이 실시예에서, 크로스-스레드, 타이머, 명령어 어드레스 및 데이터 어드레스 인터럽트의 우선 순위는 a, b, c 및 d 순서로 고정되며, 이때 a, b, c 및 d는 상기 멀티플렉서(506-1, 506-2, 506-3 및 506-4)의 각각의 출력을 나타낸다. 상기 네 개의 입력 세트 a, b, c 및 d에 대하여, 상기 우선 순위 로직(508)은 출력으로서 인터럽트 어드레스 및 멀티-비트 컨트롤 신호를 생성한다. 상기 어드레스 출력은 상기 스레드에 대하여 새로운 실행 어드레스인 상기 인터럽트 핸들러에 대한 실행 어드레스를 제공한다. 상기 멀티-비트 제어 신호는 가령, 5-비트 신호가 될 수 있으며, 이때 상기 5-비트 신호에서 최상위 비트는 인터럽트가 취해졌는지를 나타내고 나머지 비트는 어떤 인터럽트가 취해지는가를 나타낸다.

[0076] 보다 특별한 예로서, 상기 어드레스 및 제어 신호는 입력 a, b, c 및 d를 바탕으로 아래의 값 상에서 취할 수 있고, 이때 상기 어드레스는 16진수이고 제어 신호는 이진수이다:

[0077] Xthr_I0=a; XThr_address : 0x200

[0078] cntl : 10001

[0079] Xthr_I1=(not a and b); Timer_address : 0x280

[0080] cntl : 10010

[0081] Xthr_I2=(not a and not b and c); IADR_address : 0x300

[0082] cntl : 10100

[0083] Xthr_I3=(not a and not b and not c and d); DADR_address : 0x380

[0084] cntl : 11000

[0085] Otherwise Address : don't care

[0086] cntl : 00000

[0087] 앞선 예에서, Xthr_I0, Xthr_I1, Xthr_I2, 및 Xthr_I3은 우선 순위 로직 입력 a, b, c 및 d에 대한 서로 다른 조합을 표시하며, 이때 값의 존재는 상기 값이 로직 하이 레벨에 있음을 나타내고, 상기 값 앞의 용어 "not"은 상기 값이 로직 로우 레벨에 있음을 나타낸다.

[0088] 앞서 설명된 고정된 우선 순위 배열은 본 발명의 필수요건은 아니다. 또 다른 우선 순위의 형태, 가령 랜덤 우

선 순위 또는 프로그래머블 우선 순위 등은 직접적인 방식으로 제공될 수 있다.

[0089] 앞서 언급된 바와 같이, 본 발명의 크로스-스레드 인터럽트 기술은 종래의 기술에 비해 많은 장점을 제공한다. 예를 들어, 본 발명은 주어진 스레드가 단일 명령어 실행 사이클 내에 또 다른 스레드를 인터럽트하도록 한다. 본 발명은 따라서 멀티스레디드 프로세서에서 크로스-헤드 인터럽트를 전달하고 제공하기 위한 로우-레이턴시, 로우-오버헤드 메커니즘을 제공한다.

## 도면의 간단한 설명

[0017] 도 1은 본 발명이 구현되는 예시적 프로세싱 시스템의 블록도이다.

[0018] 도 2는 도 1의 프로세싱 시스템의 멀티스레디드 프로세서를 보다 상세히 설명한 블록도이다.

[0019] 도 3은 본 발명의 기술에 따른 도 2의 멀티스레디드 프로세서의 인터럽트 컨트롤러에서 구현될 수 있는 TIER(thread interrupt enable register)를 보여준다.

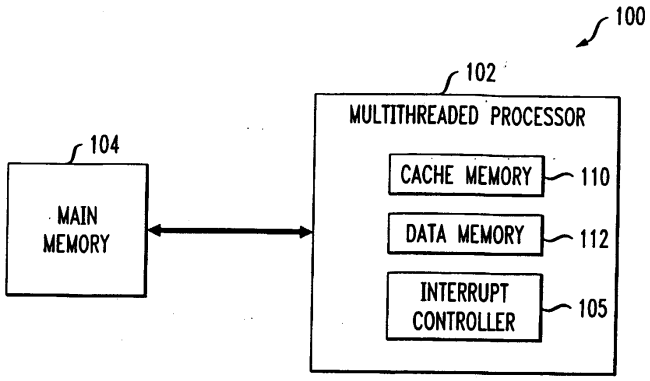
[0020] 도 4는 본 발명의 기술에 따른 도 2의 멀티스레디드 프로세서의 인터럽트 컨트롤러에서 구현될 수 있는 TIER을 보여준다.

[0021] 도 5는 본 발명에 따라 구성된 인터럽트 컨트롤러의 실시예를 보여준다.

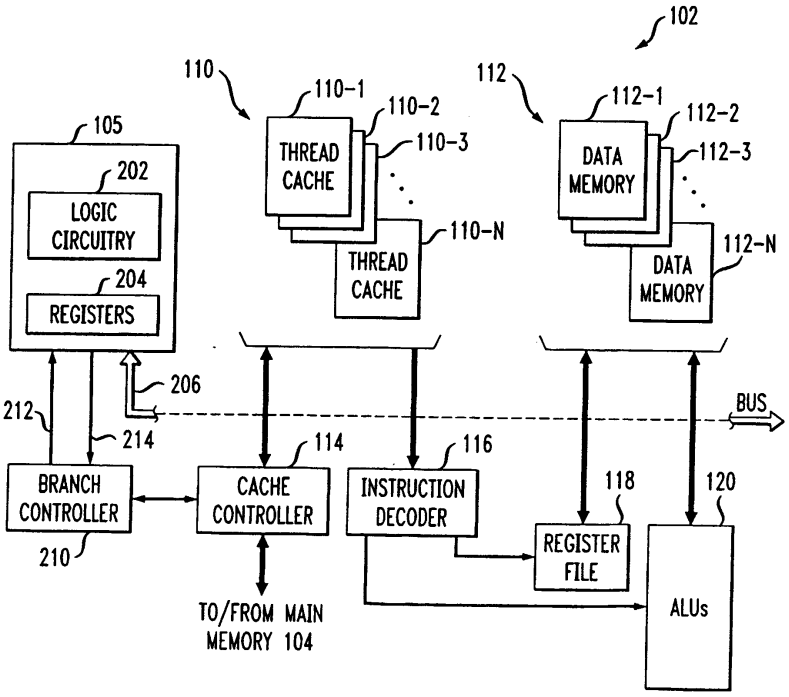
[0022] 도 6은 본 발명의 기술을 이용한 크로스-스레드 인터럽트의 프로세싱을 보여주는 흐름도이다.

## 도면

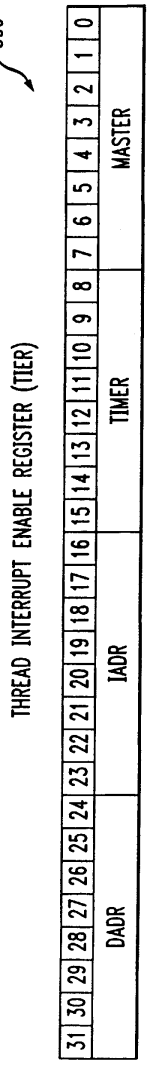
### 도면1



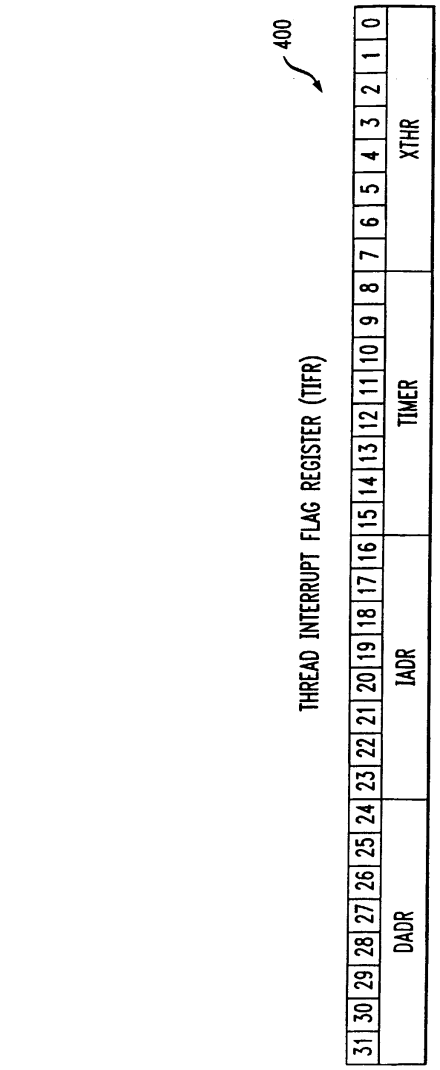
도면2


102
110
110-1
110-2
110-3
110-N
THREAD CACHE
THREAD CACHE
112
112-1
112-2
112-3
112-N
DATA MEMORY
DATA MEMORY
105
202
LOGIC CIRCUITRY
204
REGISTERS
206
212
214
BUS
BRANCH CONTROLLER
210
CACHE CONTROLLER
114
INSTRUCTION DECODER
116
118
120
REGISTER FILE
ALUs
TO/FROM MAIN MEMORY 104

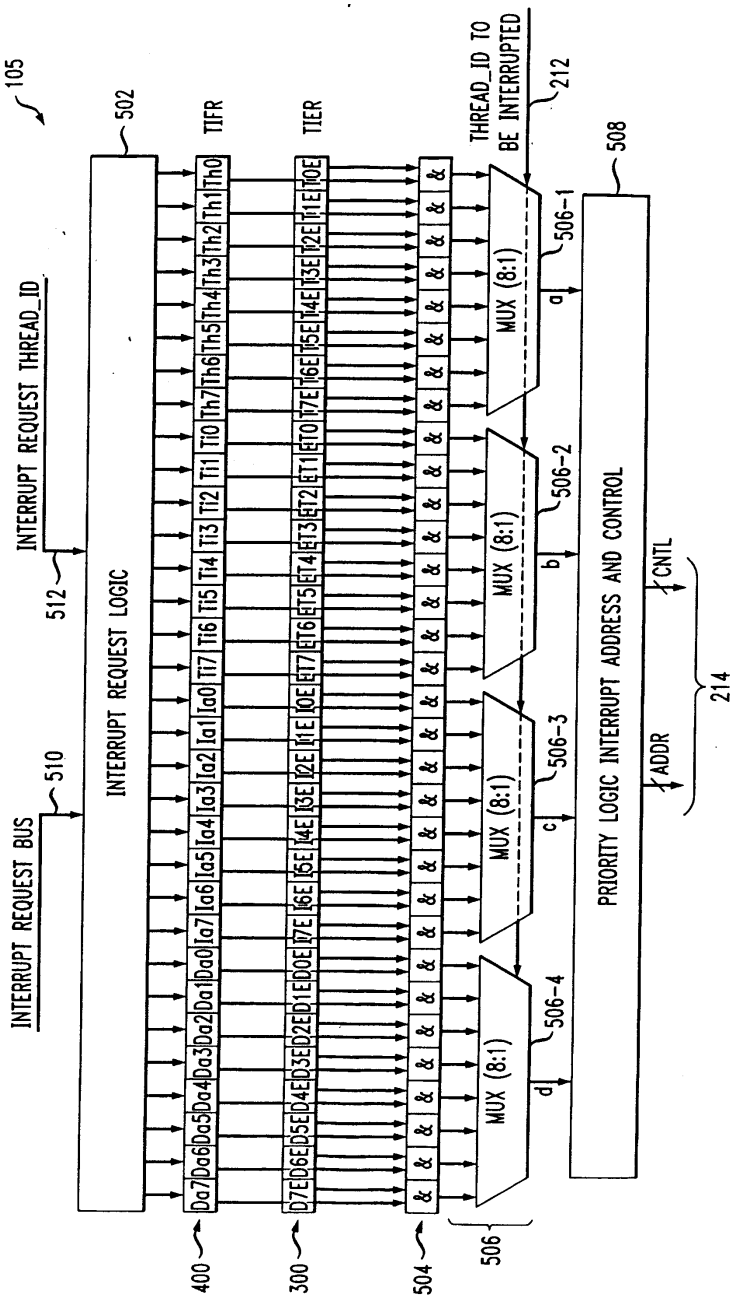
도면3


300
THREAD INTERRUPT ENABLE REGISTER (TIER)
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
DADR
IADR
TIMER
MASTER

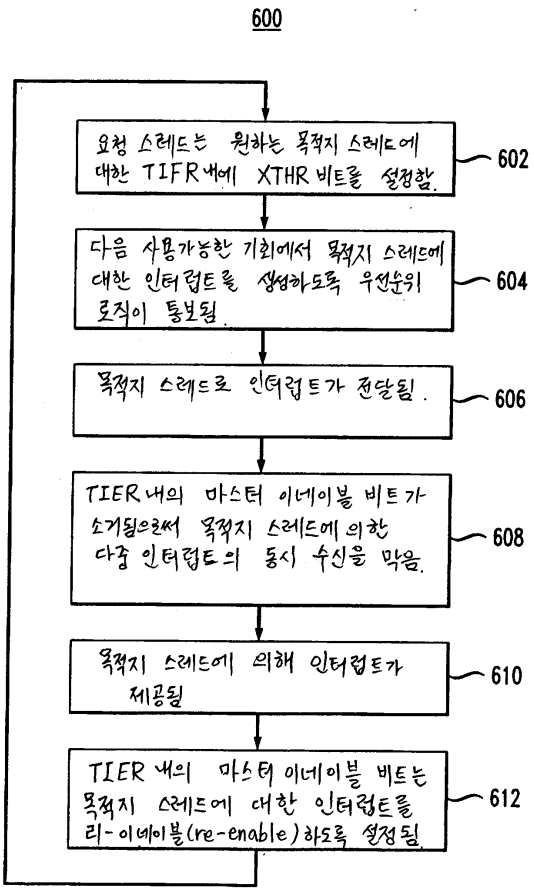
도면4


400
THREAD INTERRUPT FLAG REGISTER (TIFR)
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
DADR
IADR
TIMER
XTHR

도면5


105
502
TIFR
TIER
THREAD_ID TO BE INTERRUPTED
212
INTERRUPT REQUEST THREAD_ID
512
INTERRUPT REQUEST LOGIC
INTERRUPT REQUEST BUS
510
MUX (8:1)
506-1
MUX (8:1)
506-2
MUX (8:1)
506-3
MUX (8:1)
506-4
506
508
PRIORITY LOGIC INTERRUPT ADDRESS AND CONTROL
CNTL
ADDR
214
400
300
504

도면6


600
요청 스레드는 원하는 목적지 스레드에 대한 TIFR 내에 XTHR 비트를 설정함.
602
다음 사용가능한 기회에서 목적지 스레드에 대한 인터럽트를 생성하도록 우선순위 로직이 통보됨.
604
목적지 스레드로 인터럽트가 전달됨.
606
TIER 내의 마스터 이네이블 비트가 소거됨으로써 목적지 스레드에 의한 다중 인터럽트의 동시 수신을 막음.
608
목적지 스레드에 의해 인터럽트가 제공됨
610
TIER 내의 마스터 이네이블 비트는 목적지 스레드에 대한 인터럽트를 리-이네이블(re-enable)하도록 설정됨
612