



(19) 대한민국특허청(KR)

(12) 등록특허공보(B1)

(45) 공고일자 2015년02월06일

(11) 등록번호 10-1489887

(24) 등록일자 2015년01월29일

(51) 국제특허분류(Int. Cl.)

G06F 21/74 (2013.01) G06F 9/48 (2006.01)

G06F 11/30 (2006.01)

(21) 출원번호 10-2014-0008504

(22) 출원일자 2014년01월23일

심사청구일자 2014년01월23일

(56) 선행기술조사문헌

KR1020050049170 A

KR1020050086673 A

KR1020070071588 A

(73) 특허권자

고려대학교 산학협력단

서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)

(72) 발명자

김세원

서울특별시 성북구 안암로 145 고려대학교 자연계 캠퍼스 우정정보통신관 308호

박혜정

서울특별시 성북구 안암로 145 고려대학교 자연계 캠퍼스 우정정보통신관 308호

(뒷면에 계속)

(74) 대리인

특허법인엠에이피에스

전체 청구항 수 : 총 6 항

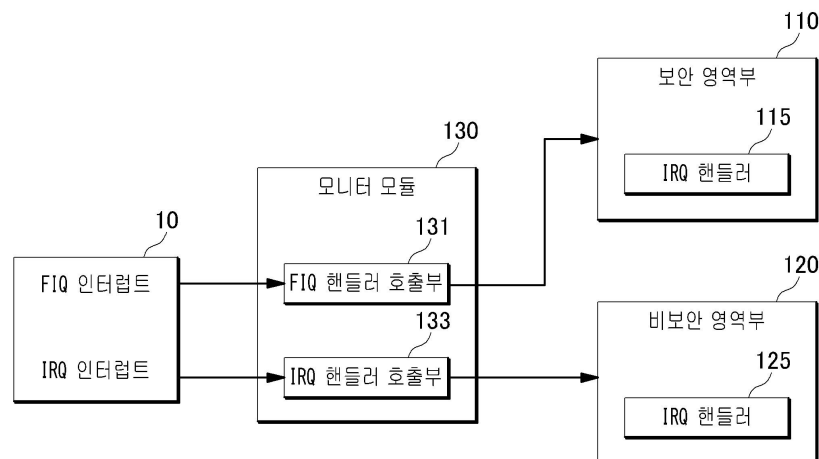
심사관 : 구본재

(54) 발명의 명칭 인터럽트 처리 시스템 및 방법

(57) 요약

본 발명에 따른 ARM 프로세서에서의 인터럽트 처리 시스템은 FIQ(Fast Interrupt reQuest) 인터럽트를 처리하는 IRQ(Interrupt ReQuest) 핸들러를 포함하는 보안 영역부, IRQ 인터럽트를 처리하는 IRQ 핸들러를 포함하는 비보안 영역부 및 FIQ 핸들러 호출부 및 IRQ 핸들러 호출부를 포함하며, 상기 보안 영역부가 예외처리 중인지 여부를 판별하는 모니터 모듈을 포함하되, 상기 FIQ 핸들러 호출부는 외부 하드웨어 장치로부터 전달받은 상기 FIQ 인터럽트를 처리하기 위하여 상기 보안 영역부의 IRQ 핸들러를 호출하고, 상기 IRQ 핸들러는 상기 외부 하드웨어 장치로부터 전달받은 상기 IRQ 인터럽트를 처리하기 위하여 상기 비보안 영역부의 IRQ 핸들러를 호출하며, 상기 모니터 모듈은 상기 보안 영역부가 예외처리 중인 것으로 판별한 경우, 상기 FIQ 핸들러 호출부는 상기 FIQ 인터럽트를 비활성화시키고, 상기 예외처리가 종료된 후 상기 FIQ 인터럽트를 다시 활성화시킨다.

대표도 - 도2



(72) 발명자

유성배

서울특별시 성북구 안암로 145 고려대학교 자연계
캠퍼스 우정정보통신관 308호

유혁

서울특별시 강남구 광평로39길 3, 801동 101호(수
서동, 동익아파트)

특허청구의 범위

청구항 1

ARM 프로세서에서의 인터럽트 처리 시스템에 있어서,
 FIQ(Fast Interrupt reQuest) 인터럽트를 처리하는 IRQ(Interrupt ReQuest) 핸들러를 포함하는 보안 영역부,
 IRQ 인터럽트를 처리하는 IRQ 핸들러를 포함하는 비보안 영역부 및
 FIQ 핸들러 호출부 및 IRQ 핸들러 호출부를 포함하며, 상기 보안 영역부가 예외처리 중인지 여부를 판별하는 모니터 모듈을 포함하되,
 상기 FIQ 핸들러 호출부는 외부 하드웨어 장치로부터 전달받은 상기 FIQ 인터럽트를 처리하기 위하여 상기 보안 영역부의 IRQ 핸들러를 호출하고,
 상기 IRQ 핸들러는 상기 외부 하드웨어 장치로부터 전달받은 상기 IRQ 인터럽트를 처리하기 위하여 상기 비보안 영역부의 IRQ 핸들러를 호출하며,
 상기 모니터 모듈은 상기 보안 영역부가 예외처리 중인 것으로 판별한 경우, 상기 FIQ 핸들러 호출부는 상기 FIQ 인터럽트를 비활성화시키고, 상기 예외처리가 종료된 후 상기 FIQ 인터럽트를 다시 활성화시키는 것인 인터럽트 처리 시스템.

청구항 2

제 1 항에 있어서,
 상기 FIQ 핸들러 호출부는,
 상기 모니터 모듈이 상기 보안 영역부가 예외처리 중이 아닌 것으로 판별한 경우, 상기 FIQ 인터럽트를 상기 보안 영역부의 IRQ 핸들러를 통해 처리하도록 호출하는 것인 인터럽트 처리 시스템.

청구항 3

제 1 항에 있어서,
 상기 보안 영역부 및 비보안 영역부는 서로 독립적으로 운영되는 운영체제에 의하여 관리되며, 상기 운영체제는 리눅스인 것인 인터럽트 처리 시스템.

청구항 4

ARM 프로세서의 인터럽트를 처리하기 위한 방법에 있어서,
 모니터 모듈의 FIQ(Fast Interrupt reQuest) 핸들러 호출부가 외부 하드웨어 장치로부터 상기 FIQ 인터럽트를 전달받는 단계,
 상기 모니터 모듈이, IRQ(Interrupt ReQuest) 핸들러를 각각 포함하는 보안 영역부 및 비보안 영역부로 구분된 상기 ARM 프로세서에서 예외처리 중인지 여부를 판별하는 단계,
 상기 ARM 프로세서가 예외처리 중인 것으로 판별된 경우, 상기 FIQ 핸들러 호출부가 상기 FIQ 인터럽트를 비활성화시키는 단계,
 상기 예외처리가 종료된 후, 상기 FIQ 핸들러 호출부가 상기 비활성화된 FIQ 인터럽트를 활성화시키는 단계,
 상기 FIQ 핸들러 호출부가 상기 보안 영역부의 IRQ 핸들러를 호출하는 단계 및
 상기 보안 영역부의 IRQ 핸들러가 상기 FIQ 인터럽트를 처리하는 단계를 포함하되,
 상기 보안 영역부의 IRQ 핸들러는 상기 FIQ 인터럽트를 처리하고, 상기 비보안 영역부의 IRQ 핸들러는 IRQ 인터럽트를 처리하는 것인 인터럽트 처리 방법.

청구항 5

제 4 항에 있어서,

상기 ARM 프로세서가 예외처리 중인지 여부를 판별하는 단계는,

상기 ARM 프로세서가 예외처리 중이 아닌 것으로 판별된 경우, 상기 FIQ 핸들러 호출부는 상기 FIQ 인터럽트를 상기 보안 영역부의 IRQ 핸들러를 통해 처리하도록 호출하는 것인 인터럽트 처리 방법.

청구항 6

제 4 항에 있어서,

상기 보안 영역부 및 비보안 영역부는 서로 독립적으로 운영되는 운영체제에 의하여 관리되며, 상기 운영체제는 리눅스인 것인 인터럽트 처리 방법.

명세서

기술 분야

[0001] 본 발명은 인터럽트 처리 시스템 및 방법에 관한 것이다.

배경 기술

[0002] ARM 프로세서의 SoC(System On Chip) 기술 중 하나로서, 하드웨어 레벨의 보안을 향상시키기 위한 ARM TrustZone 기술이 있다. 이는 CPU를 보안 영역(Secure World)과 비보안 영역(Normal World)으로 나누어 제공하며, 각 영역에서 사용되는 메모리와 장치를 분할하여 사용할 수 있도록 한다.

[0003] 이와 같이 ARM TrustZone 기술은 보안을 향상시키기 위한 기술이지만, CPU를 비롯한 메모리와 장치를 2개의 영역으로 분할하여 사용하기 때문에, 2개의 가상 머신이 필요한 환경에서 임베디드 장치의 가상화를 위한 방법으로 활용되어 왔다.

[0004] 한편, ARM 프로세서는 실행 중에 발생할 수 있는 예외를 RESET, UNDEFINED INSTRUCTION, SWI, PREFETCH ABORT, DATA ABORT, IRQ, FIQ인 총 7개로 정의하고 있다. 이때, ARM 프로세서는 FIQ와 IRQ로 구분되는 2개의 인터럽트 우선 순위 모델을 제공할 수 있다. ARM 프로세서는 위와 같은 예외가 발생하면 실행 모드를 변경하여 예외를 처리한 후 본래 실행하던 모드로 복귀하게 된다.

[0005] 구체적으로, SoC에 연결되는 모든 장치는 인터럽트를 발생할 수 있으며, 이는 FIQ 또는 IRQ 중 하나의 형태로 인터럽트를 프로세서에 전달되도록 설정할 수 있다. 특히, ARM TrustZone 가상화 환경에서 보안 영역은 FIQ 인터럽트를, 비보안 영역은 IRQ 인터럽트를 전달하도록 함으로써, 두 영역에서 사용하는 장치에 대한 독립적인 인터럽트 처리 방법을 제공할 수 있다.

[0006] 이하에서는, 도 1을 참조하여 종래 기술에 따른 인터럽트 처리 방법을 설명하도록 한다.

[0007] 도 1은 종래 기술에 따른 인터럽트 처리 방법을 도시한 도면이다.

[0008] 종래 기술에 따른 ARM TrustZone은 두 개의 영역인 보안 영역부와 비보안 영역부에서 실행하는 운영체제에게 인터럽트를 제공하기 위하여, 인터럽트를 FIQ와 IRQ로 구분하여 각 영역에 제공할 수 있다. 이때, FIQ 인터럽트는 보안 영역부로 전달되고, IRQ 인터럽트는 비보안 영역부로 전달된다.

[0009] ARM 프로세서로 포팅된 리눅스는 모든 장치의 인터럽트를 IRQ 인터럽트 형식으로 받도록 되어있다. 따라서, ARM 프로세서로 포팅된 리눅스를 보안 영역부의 운영체제에서 실행하기 위하여, FIQ로 들어오는 인터럽트를 처리할 수 있도록 수정해야 한다. 하지만, 현재의 ARM 리눅스는 인터럽트를 IRQ로 받고 IRQ 모드에서 처리하기 때문에 FIQ로 들어오는 인터럽트를 ARM 리눅스의 IRQ 핸들러가 처리할 수 있도록 해야 한다.

[0010] 도 1을 참조하면, 하드웨어에서 FIQ 인터럽트가 발생하여 모니터 모듈로 전달되면(S110), 모니터 모듈은 이를 처리하기 위하여 보안 영역부의 IRQ 핸들러를 호출한다(S111). 그리고 보안 영역부의 IRQ 핸들러를 통하여 FIQ 인터럽트를 처리하게 된다(S113). 이와 같이, 종래 기술에 따른 인터럽트 처리 시스템은 보안 영역부가 예외처리 중이 아닌 경우, FIQ 인터럽트가 발생하더라도 오류가 발생하지 않는다.

[0011] 다만, 하드웨어에서 예외 처리를 요구하는 동작이 발생하여(S120), 보안 영역부가 예외처리 중에(S121) FIQ 인터럽트가 발생하면(S130), 모니터 모듈은 FIQ 인터럽트를 IRQ 핸들러로 전달한다(S131). 이와 같이 보안 영역부

가 예외처리 중에 있는 경우 FIQ 인터럽트가 발생하면, “BAD Mode in Interrupt” 오류가 발생하게 된다. 이는 ARM 프로세서가 예외를 처리하는 동안 다른 예외가 발생하여 처리하는 것을 금지하고 있기 때문이다. 하지만, IRQ로 인터럽트를 받고 이를 처리하는 경우에는 예외발생 시 하드웨어적으로 동작하는 매커니즘 상의 차이에 따라 위와 같은 문제가 발생하지 않는다.

[0012] 아래의 [표 1]은 각 예외처리 시 하드웨어적으로 FIQ 인터럽트 또는 IRQ 인터럽트의 활성화 및 비활성화 관계를 설명한 것이다.

표 1

예외처리	FIQ	IRQ
Reset	비활성화	비활성화
Undefined Instruction	활성화	비활성화
SWI	활성화	비활성화
Prefetch Abort	활성화	비활성화
Data Abort	활성화	비활성화
IRQ	활성화	비활성화
FIQ	비활성화	비활성화

[0014] 위 [표 1]을 참조하면, IRQ 인터럽트는 모든 예외처리 시 비활성화로 설정되므로, 예외처리 시 IRQ 인터럽트로 인한 문제는 발생하지 않는다. 이와 달리, FIQ 인터럽트는 RESET과 FIQ 예외 처리에만 비활성화되기 때문에, 나머지 예외 처리시 FIQ 인터럽트가 발생할 경우 “BAD Mode in Interrupt” 오류가 발생하는 문제가 생기게 된다.

[0015] 이와 관련하여 한국등록특허 제10-2010-0052477호(발명의 명칭: 임베디드 운영체제 커널의 인터럽트 처리 모드 결정 방법)에는 인터럽트 처리 모드를 복수 개로 하여 인터럽트 서비스를 수행할 수 있도록 한 임베디드 운영체제 커널의 인터럽트 처리 모드 결정 방법이 개시되어 있다.

[0016] 또한, 한국공개특허 제10-2013-0012126호(발명의 명칭: 예외 이벤트 처리 장치 및 방법)에는 예외 이벤트를 처리하기 위한 구성 요소로서 처리 회로, 예외 복귀 레지스터, 스택 포인터 레지스터, 스택 포인터 선택회로 등을 포함하는 데이터 처리 장치가 개시되어 있다.

발명의 내용

해결하려는 과제

[0017] 본 발명은 전술한 종래 기술의 문제점을 해결하기 위한 것으로서, 본 발명의 일부 실시예는 모니터 모듈의 FIQ 핸들러를 수정하여, ARM 프로세서의 보안 영역부에서 예외처리 중에 FIQ 인터럽트가 발생한 경우에도 오류 발생 없이 FIQ 인터럽트를 처리할 수 있도록 하는 인터럽트 처리 시스템 및 방법을 제공하는 것을 그 목적으로 한다.

과제의 해결 수단

[0018] 상술한 기술적 과제를 달성하기 위한 기술적 수단으로서, 본 발명의 제 1 측면에 따른 ARM 프로세서에서의 인터럽트 처리 시스템은 FIQ(Fast Interrupt reQuest) 인터럽트를 처리하는 IRQ(Interrupt ReQuest) 핸들러를 포함하는 보안 영역부, IRQ 인터럽트를 처리하는 IRQ 핸들러를 포함하는 비보안 영역부 및 FIQ 핸들러 호출부 및 IRQ 핸들러 호출부를 포함하며, 상기 보안 영역부가 예외처리 중인지 여부를 판별하는 모니터 모듈을 포함하되, 상기 FIQ 핸들러 호출부는 외부 하드웨어 장치로부터 전달받은 상기 FIQ 인터럽트를 처리하기 위하여 상기 보안 영역부의 IRQ 핸들러를 호출하고, 상기 IRQ 핸들러는 상기 외부 하드웨어 장치로부터 전달받은 상기 IRQ 인터럽트를 처리하기 위하여 상기 비보안 영역부의 IRQ 핸들러를 호출하며, 상기 모니터 모듈은 상기 보안 영역부가 예외처리 중인 것으로 판별한 경우, 상기 FIQ 핸들러 호출부는 상기 FIQ 인터럽트를 비활성화시키고, 상기 예외처리가 종료된 후 상기 FIQ 인터럽트를 다시 활성화시킨다.

[0019] 또한, 본 발명의 제 2 측면에 따른 ARM 프로세서의 가상화 환경에서 FIQ(Fast Interrupt reQuest) 인터럽트를 처리하기 위한 방법은 모니터 모듈의 FIQ 핸들러 호출부가 외부 하드웨어 장치로부터 상기 FIQ 인터럽트를 전달받는 단계, 상기 모니터 모듈이, IRQ(Interrupt ReQuest) 핸들러를 각각 포함하는 보안 영역부 및 비보안 영역부로 구분된 상기 ARM 프로세서에서 예외처리 중인지 여부를 판별하는 단계, 상기 ARM 프로세서가 예외처리 중

인 것으로 판별된 경우, 상기 FIQ 핸들러 호출부가 상기 FIQ 인터럽트를 비활성화시키는 단계, 상기 예외처리가 종료된 후, 상기 FIQ 핸들러 호출부가 상기 비활성화된 FIQ 인터럽트를 활성화시키는 단계, 상기 FIQ 핸들러 호출부가 상기 보안 영역부의 IRQ 핸들러를 호출하는 단계 및 상기 보안 영역부의 IRQ 핸들러가 상기 FIQ 인터럽트를 처리하는 단계를 포함하되, 상기 보안 영역부의 IRQ 핸들러는 상기 FIQ 인터럽트를 처리하고, 상기 비보안 영역부의 IRQ 핸들러는 IRQ 인터럽트를 처리한다.

발명의 효과

- [0020] 전술한 본 발명의 과제 해결 수단의 어느 실시예에 의하면, ARM TrustZone 가상화를 이용하여 보안 영역부의 IRQ 핸들러를 통해 FIQ 인터럽트를 처리할 수 있다. 즉, 보안 영역부와 비보안 영역부 모두에서 리눅스와 같은 범용 운영체제를 동작시키기 위해 필요한 FIQ 인터럽트 처리시 발생하는 문제를 해결할 수 있다.
- [0021] 이에 따라, ARM 프로세서의 보안 영역부에서 예외처리 중에 FIQ 인터럽트를 전달받더라도 오류 발생 없이 FIQ 인터럽트를 처리할 수 있다.
- [0022] 또한, 리눅스의 IRQ 핸들러를 수정하지 않고 보안 영역부의 FIQ 인터럽트를 처리할 수 있다.

도면의 간단한 설명

- [0023] 도 1은 종래 기술에 따른 인터럽트 처리 방법을 도시한 도면이다.
- 도 2는 본 발명의 일 실시예에 따른 ARM 프로세서에서의 인터럽트 처리 시스템의 블록도이다.
- 도 3은 종래 기술 및 본 발명에 따른 FIQ 핸들러 호출부에 적용된 알고리즘의 일 예시를 도시한 도면이다.
- 도 4는 본 발명의 일 실시예에 따른 ARM 프로세서에서의 인터럽트 처리 방법의 순서도이다.

발명을 실시하기 위한 구체적인 내용

- [0024] 아래에서는 첨부한 도면을 참조하여 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자가 용이하게 실시할 수 있도록 본 발명의 실시예를 상세히 설명한다. 그러나 본 발명은 여러 가지 상이한 형태로 구현될 수 있으며 여기에서 설명하는 실시예에 한정되지 않는다. 그리고 도면에서 본 발명을 명확하게 설명하기 위해서 설명과 관계없는 부분은 생략하였으며, 명세서 전체를 통하여 유사한 부분에 대해서는 유사한 도면 부호를 붙였다.
- [0025] 명세서 전체에서, 어떤 부분이 다른 부분과 "연결"되어 있다고 할 때, 이는 "직접적으로 연결"되어 있는 경우뿐 아니라, 그 중간에 다른 소자를 사이에 두고 "전기적으로 연결"되어 있는 경우도 포함한다. 또한 어떤 부분이 어떤 구성요소를 "포함"한다고 할 때, 이는 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라 다른 구성요소를 더 포함할 수 있는 것을 의미한다.
- [0026] 도 2는 본 발명의 일 실시예에 따른 ARM 프로세서에서의 인터럽트 처리 시스템(100)의 블록도이다.
- [0027] 본 발명의 일 실시예에 따른 인터럽트 처리 시스템(100)은 보안 영역부(110), 비보안 영역부(120) 및 모니터 모듈(130)을 포함한다.
- [0028] 참고로, 본 발명의 실시예에 따른 도 2에 도시된 구성 요소들은 소프트웨어 또는 FPGA(Field Programmable Gate Array) 또는 ASIC(Application Specific Integrated Circuit)와 같은 하드웨어 구성 요소를 의미하며, 소정의 역할들을 수행한다.
- [0029] 그렇지만 '구성 요소들'은 소프트웨어 또는 하드웨어에 한정되는 의미는 아니며, 각 구성 요소는 어드레싱할 수 있는 저장 매체에 있도록 구성될 수도 있고 하나 또는 그 이상의 프로세서들을 재생시키도록 구성될 수도 있다.
- [0030] 따라서, 일 예로서 구성 요소는 소프트웨어 구성 요소들, 객체지향 소프트웨어 구성 요소들, 클래스 구성 요소들 및 테스크 구성 요소들과 같은 구성 요소들과, 프로세스들, 함수들, 속성들, 프로시저들, 서브루틴들, 프로그램 코드의 세그먼트들, 드라이버들, 펌웨어, 마이크로 코드, 회로, 데이터, 데이터베이스, 데이터 구조들, 테이블들, 어레이들 및 변수들을 포함한다.
- [0031] 구성 요소들과 해당 구성 요소들 안에서 제공되는 기능은 더 작은 수의 구성 요소들로 결합되거나 추가적인 구성 요소들로 더 분리될 수 있다.
- [0032] 보안 영역부(110)는 FIQ(Fast Interrupt reQuest) 인터럽트를 처리하는 IRQ(Interrupt ReQuest) 핸들러(115)를

포함한다. 그리고 비보안 영역부(120)는 IRQ 인터럽트를 처리하는 IRQ 핸들러(125)를 포함한다.

- [0033] 이때, 보안 영역부(110) 및 비보안 영역부(120)는 서로 독립적으로 운영되는 운영체제에 의하여 관리될 수 있으며, 이때 적용된 운영체제는 리눅스일 수 있다. 즉, ARM 프로세서의 TrustZone은 하나의 물리 프로세서 코어를 보안 영역부(110)와 비보안 영역부(120)로 구분하여 운영할 수 있으며, ARM TrustZone을 부팅시킬 경우, 2회의 부팅을 통해 각 영역이 부팅된다. ARM TrustZone의 각 영역은 CPU의 레지스터 및 페이지 테이블 관련 레지스터가 분리되어 있고, 인터럽트 또한 보안 영역부(110) 및 비보안 영역부(120)로 각각 나누어 발생시킬 수 있다. 이와 같이, 보안 영역부(110) 및 비보안 영역부(120)는 각 영역마다 각각 독립적으로 운영될 수 있으며, 별도의 운영체제에 의하여 관리될 수 있다.
- [0034] 모니터 모듈(130)은 보안 영역부(110)와 비보안 영역부(120)의 중간 역할을 수행하며, 각 영역의 스위칭, 인터럽트 처리를 담당한다. 모니터 모듈(130)은 장치 인터럽트가 발생하면 모니터 벡터라는 특정 주소 영역에 설정된 인터럽트 핸들러로 프로세서를 실행 흐름에 옮기고, 인터럽트 핸들러에서 FIQ 인터럽트 및 IRQ 인터럽트를 처리한다. 뿐만 아니라, 각 영역의 스위칭을 통해 호출된 SMC(Secure Monitor Call) 명령어 역시 모니터 모듈(130)이 처리하며, 이때, 각 영역의 스위칭을 위한 프로세서 컨텍스트 정보의 저장과 복구를 담당한다.
- [0035] 이와 같은 모니터 모듈(130)은 FIQ 핸들러 호출부(131) 및 IRQ 핸들러 호출부(133)를 포함한다. 이때, FIQ 핸들러 호출부(131)는 외부 하드웨어 장치(10)로부터 전달받은 FIQ 인터럽트를 처리하기 위하여 보안 영역부(110)의 IRQ 핸들러(115)를 호출한다. 그리고 IRQ 핸들러 호출부(133)는 외부 하드웨어 장치(10)로부터 전달받은 IRQ 인터럽트를 처리하기 위하여 비보안 영역부(120)의 IRQ 핸들러(125)를 호출한다.
- [0036] 한편, 모니터 모듈(130)은 보안 영역부(110)가 예외처리 중인지 여부를 판별한다. 이때, 모니터 모듈(130)이 보안 영역부(110)가 예외처리 중인 것으로 판별한 경우, FIQ 핸들러 호출부(131)는 FIQ 인터럽트를 비활성화시키고, 예외처리가 종료된 후 FIQ 인터럽트를 다시 활성화시킨다.
- [0037] 즉, 모니터 모듈(130)은 보안 영역부(110)의 IRQ 핸들러(115)를 호출하기 전에 현재 실행 중인 프로세서의 보안 영역부(110)가 예외를 처리하는 중인지 아닌지 여부를 판별한다. 프로세서의 보안 영역부(110)가 예외 처리 중에 FIQ 인터럽트가 발생하면, 강제로 FIQ 인터럽트를 비활성화시키고, 현재 발생한 FIQ 인터럽트를 잠시 대기시킨다. 그 다음 수행중인 예외처리 단계를 종료시킨 후, 대기시킨 FIQ 인터럽트를 다시 활성화시켜 처리한다.
- [0038] 이와 같이, 모니터 모듈(130)은 FIQ 인터럽트를 처리하기 위하여 보안 영역부(110)의 IRQ 핸들러(115)를 호출하기 전에, 보안 영역부(110)가 예외처리 중인지 여부를 판별함으로써 “BAD Mode in Interrupt” 오류 발생을 방지할 수 있다. 이에 따라, 보안 영역부(110)의 IRQ 핸들러(115)를 수정하지 않고 보안 영역부(110)로 하여금 FIQ 인터럽트를 처리할 수 있다.
- [0039] 이와 달리, 모니터 모듈(130)이 보안 영역부(110)가 예외처리 중이 아닌 것으로 판별한 경우, FIQ 핸들러 호출부(131)는 FIQ 인터럽트를 보안 영역부(110)의 IRQ 핸들러(115)를 통해 처리하도록 호출할 수 있다.
- [0040] 이하에서는 도 3을 참조하여 모니터 모듈(130)의 FIQ 핸들러 호출부(131)에 적용된 알고리즘의 일 예시를 설명하도록 한다.
- [0041] 도 3은 종래 기술 및 본 발명에 따른 FIQ 핸들러 호출부에 적용된 알고리즘의 일 예시를 도시한 도면이다.
- [0042] 종래 기술에 따른 FIQ 핸들러 호출부에 적용된 알고리즘의 경우, 먼저 S11단계에서 FIQ 인터럽트를 모니터 모듈의 FIQ 핸들러 호출부가 전달받으면, S13 단계에서 FIQ 핸들러 호출부는 보안 영역부의 IRQ 핸들러로 처리하기 위하여 FIQ 인터럽트를 전달한다.
- [0043] 이때, 종래 기술에 따른 FIQ 핸들러 호출부는 보안 영역부가 예외처리 중인지 여부를 판별하는 단계가 없으며, 이에 따라 예외 처리 중에 FIQ 인터럽트를 전달받을 경우 “BAD Mode in Interrupt” 오류가 발생하는 문제가 있었다.
- [0044] 이와 달리, 본 발명에 따른 FIQ 핸들러 호출부(131)에 적용된 알고리즘의 경우, S21단계에서 보안 영역부(110)가 예외처리 중인지 여부를 판별한다. 이때, 보안 영역부(110)가 예외처리 중인 것으로 판별된 경우, S23 단계에서 FIQ 인터럽트를 비활성화시킨다. 즉, 예외처리가 종료될 때까지 FIQ 인터럽트를 잠시 대기시킨다. 다음으로, S25 단계를 통해 다시 예외처리 수행 단계로 복귀하여 예외처리를 수행한다.
- [0045] 이와 같이, 본 발명에 따른 인터럽트 처리 시스템(100)에 포함된 FIQ 핸들러 호출부(131)는 FIQ 인터럽트가 발생할 경우, 보안 영역부(110)가 예외처리 중인지 여부를 판별하고, 예외처리가 종료될 때까지 FIQ 인터럽트를

잠시 대기시킴으로써, 종래 기술에 따른 FIQ 핸들러 호출부에서 발생하던 오류 문제를 해소할 수 있다.

[0046] 도 4는 본 발명의 일 실시예에 따른 인터럽트 처리 시스템(100)을 이용하여 FIQ 인터럽트를 처리하는 방법을 도시한 순서도이다.

[0047] 본 발명에 따른 인터럽트 처리 방법은 하드웨어 장치(10)로부터 FIQ 인터럽트가 발생하면(S210), ARM 프로세서의 모니터 모듈(130)이 이를 전달받는다.

[0048] 다음으로, 모니터 모듈(130)이 IRQ 핸들러(115, 125)를 각각 포함하는 보안 영역부(110) 및 비보안 영역부(120)로 구분된 ARM 프로세서에서 예외처리 중인지 여부를 판별한다(S220). 즉, FIQ 인터럽트를 보안 영역부(110)의 IRQ 핸들러(115)로 전달하기 전에, 하드웨어 장치(10)로부터 예외가 발생하여 보안 영역부(110)에서 예외를 처리하고 있는 중인지 여부를 먼저 판별한다.

[0049] 이때, ARM 프로세서가 예외처리 중인 것으로 판별된 경우, FIQ 핸들러 호출부(131)가 FIQ 인터럽트를 비활성화시키고(S230), FIQ 인터럽트를 잠시 대기시킨다. 이에 따라, ARM 프로세서는 수행 중이던 예외처리를 먼저 처리한다.

[0050] 다음으로, ARM 프로세서에서 수행 중인 예외처리가 종료되면(S240), FIQ 핸들러가 비활성화된 FIQ 인터럽트를 활성화시킨다(S250). 그리고 FIQ 핸들러 호출부(131)가 보안 영역부(110)의 IRQ 핸들러(115)를 호출하고(S255), 이에 따라 보안 영역부(110)의 IRQ 핸들러가 FIQ 인터럽트를 처리한다(S260).

[0051] 이와 같이, 본 발명에 따른 인터럽트 처리 방법은 FIQ 인터럽트를 보안 영역부(110)의 IRQ 핸들러(115)로 전달하기 전에 ARM 프로세서에서 예외처리 중인지 여부를 먼저 판별함으로써 “BAD Mode in Interrupt” 오류 발생 하던 기존의 문제점을 해결할 수 있다.

[0052] 이와 달리, ARM 프로세서가 예외처리 중인 것이 아닌 경우로 판별된 경우, 모니터 모듈(130)은 FIQ 인터럽트를 처리하기 위하여 FIQ 핸들러 호출부(131)를 통하여 보안 영역부(110)의 IRQ 핸들러(115)로 FIQ 인터럽트를 전달한다(S255). 이에 따라, 보안 영역부(110)의 IRQ 핸들러(115)는 FIQ 인터럽트를 처리하게 된다(S260).

[0053] 한편, 모니터 모듈(130)에 포함된 보안 영역부(110)의 IRQ 핸들러(115)는 FIQ 인터럽트를 처리하고, 비보안 영역부(120)의 IRQ 핸들러(125)는 IRQ 인터럽트를 처리한다. 이때, 보안 영역부(110) 및 비보안 영역부(120)는 서로 독립적으로 운영되는 운영체제에 의하여 관리되며, 이때 적용된 운영체제는 리눅스일 수 있다. 한편, 보안 영역부(110) 및 비보안 영역부(120)에 대하여는 도 2에서 구체적으로 설명하였으므로, 이하에서는 생략하도록 한다.

[0054] 본 발명의 일 실시예는 컴퓨터에 의해 실행되는 프로그램 모듈과 같은 컴퓨터에 의해 실행가능한 명령어를 포함하는 기록 매체의 형태로도 구현될 수 있다. 컴퓨터 판독 가능 매체는 컴퓨터에 의해 액세스될 수 있는 임의의 가용 매체일 수 있고, 휘발성 및 비휘발성 매체, 분리형 및 비분리형 매체를 모두 포함한다. 또한, 컴퓨터 판독 가능 매체는 컴퓨터 저장 매체 및 통신 매체를 모두 포함할 수 있다. 컴퓨터 저장 매체는 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 기타 데이터와 같은 정보의 저장을 위한 임의의 방법 또는 기술로 구현된 휘발성 및 비휘발성, 분리형 및 비분리형 매체를 모두 포함한다. 통신 매체는 전형적으로 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈, 또는 반송파와 같은 변조된 데이터 신호의 기타 데이터, 또는 기타 전송 매커니즘을 포함하며, 임의의 정보 전달 매체를 포함한다.

[0055] 전술한 본 발명의 설명은 예시를 위한 것이며, 본 발명이 속하는 기술분야의 통상의 지식을 가진 자는 본 발명의 기술적 사상이나 필수적인 특징을 변경하지 않고서 다른 구체적인 형태로 쉽게 변형이 가능하다는 것을 이해할 수 있을 것이다. 그러므로 이상에서 기술한 실시예들은 모든 면에서 예시적인 것이며 한정적이 아닌 것으로 이해해야만 한다. 예를 들어, 단일형으로 설명되어 있는 각 구성 요소는 분산되어 실시될 수도 있으며, 마찬가지로 분산된 것으로 설명되어 있는 구성 요소들도 결합된 형태로 실시될 수 있다.

[0056] 본 발명의 범위는 상기 상세한 설명보다는 후술하는 특허청구범위에 의하여 나타내어지며, 특허청구범위의 의미 및 범위 그리고 그 균등 개념으로부터 도출되는 모든 변경 또는 변형된 형태가 본 발명의 범위에 포함되는 것으로 해석되어야 한다.

부호의 설명

[0057] 10: 하드웨어 장치

100: 인터럽트 처리 시스템

110: 보안 영역부

115, 125: IRQ 핸들러

120: 비보안 영역부

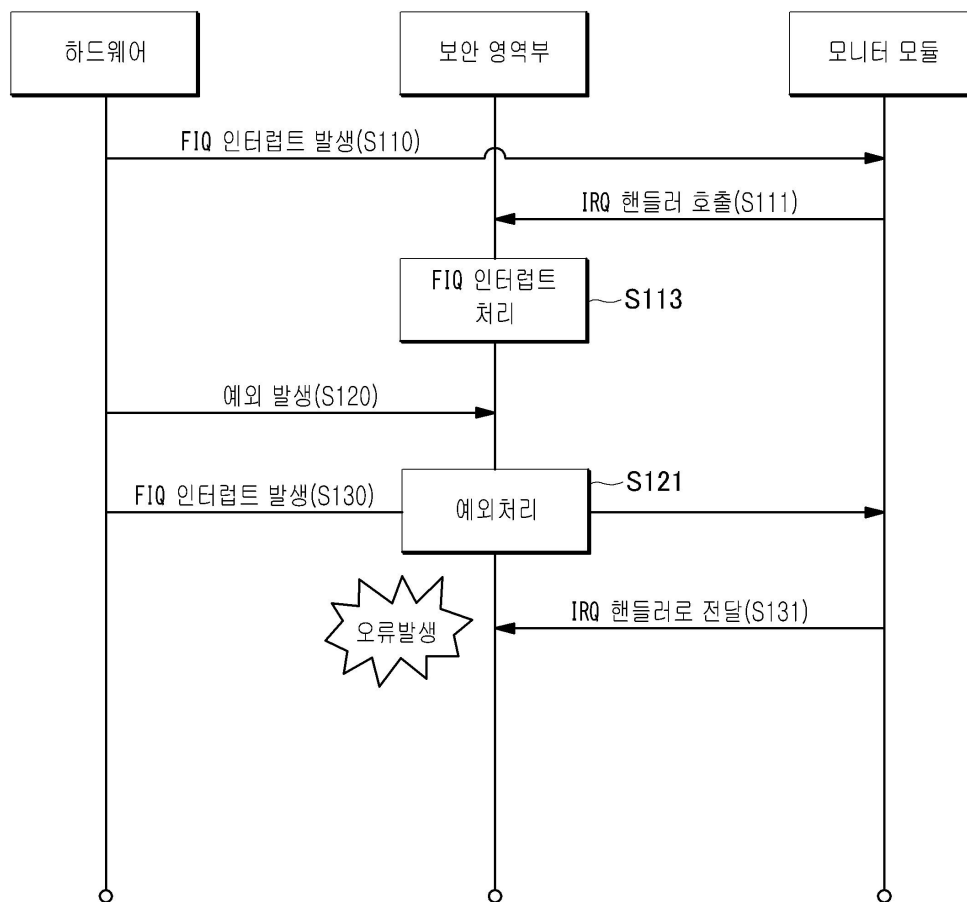
130: 모니터 모듈

131: FIQ 핸들러 호출부

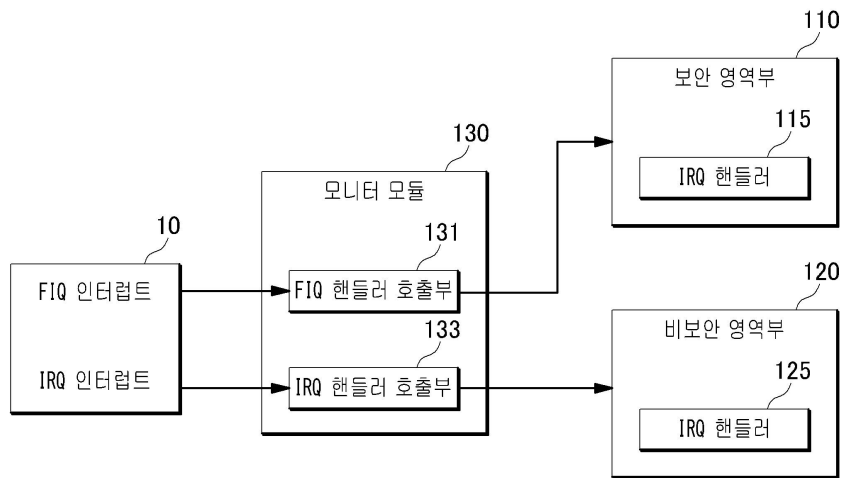
133: IRQ 핸들러 호출부

도면

도면1



도면2



도면3

기존 FIQ 핸들러 호출부

```

trusted fiq:
  ldr sp, =_smm_stack
  ldr sp, [sp]
  stmfd sp!, {r0-r1}
  mov r0, lr
  mrs r1, spsr
  cps #CPSR_MODE_IRQ
  mov lr, r0
  msr spsr, r1
  cps #CPSR_MODE_MONITOR
  ldmfd sp!, {r0-r1}
  b fiq_exit
  
```

s11

s13

본 발명에 따른 FIQ 핸들러 호출부

```

trusted fiq:
  ldr sp, =_smm_stack
  ldr sp, [sp]
  and r0, r0, #0x0f
  cmp r0, #0      @ exception from usr
  beq 1f
  cmp r0, #3      @ exception from svc
  beq 1f
  orr r0, r0, #CPSR_BIT_6_F
  msr spsr, r0
  sub lr, lr, #4
  movs pc, lr
1:
  stmfd sp!, {r0-r1}
  ....
  b fiq_exit
  
```

s21

s23

s25

도면4

