(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl. [6]
G06F 11/28

(45) 공고일자 2002년12월28일
(11) 등록번호 10-0350568
(24) 등록일자 2002년08월16일

(21) 출원번호 10-1996-0035679
(22) 출원일자 1996년08월27일

(65) 공개번호 특1997-0012152
(43) 공개일자 1997년03월29일

(30) 우선권주장 520,943 1995년08월30일 미국(US)

(73) 특허권자 모토로라 인코포레이티드
미국, 일리노이 60196, 샤움버그, 이스트 앨공퀸 로드 1303

(72) 발명자 조셉 씨. 서셀로
미국, 아리조나 85023-5230, 피닉스, 웨스트 콘라드 드라이브 44
윌리암 에이. 홀
미국, 텍사스 78660, 풀루거빌, 더렌 레인 204

(74) 대리인 이병호

심사관 : 박성호

(54) 디버그기능을수행하기위한데이타처리시스템및방법

요약

데이터 프로세서(3)는 데이터 프로세서의 실시간 동작의 방해를 최소로하고 데이터 프로세서를 정지하지 않으면서 디버그 동작을 실행한다. 상기 데이터 프로세서는 중단점 신호가 단정될 때 상기 데이터 프로세서에 의해 실행되는 기능을 결정하기 위한 트리거 응답 값을 저장하는 제어 레지스터(40)를 구현한다. 트리거 응답 값은 중단점 신호가 단정될 때 상기 데이터 프로세서의 중앙 처리 유니트(2)가 특정 디버그 예외 처리 루틴을 실행하는 지 또는 동작을 정지하는 지를 표시한다.

대표도

도 1

명세서

도면의 간단한 설명

도 1은 본 발명의 한 실시예에 따른 데이터 처리 시스템을 도시하는 블럭도.

도 2는 도 1의 데이터 처리 시스템의 디버그 모듈의 한 부분을 도시하는 블럭도.

도 3은 도 2의 디버그 모듈의 중단점을 도시하는 블럭도.

도 4는 도 2의 디버그 모듈의 다수의 제어 레지스터를 도시하는 블럭도.

도 5는 도 2의 다수의 하드웨어 중단점 레지스터를 도시하는 블럭도.

도 6은 도 5의 다수의 하드웨어 중단점 레지스터의 중단점 레지스터를 할당하는 어드레스를 도시하는 블럭도.

도 7은 도 4의 다수의 제어 레지스터의 트리거 정의 레지스터를 도시하는 블럭도.

도 8은 도 4의 다수의 제어 레지스터의 다수의 구성 상태 레지스터를 도시하는 블럭도.

도 9는 도 2의 디버그 모듈의 다수의 중단점 출력을 도시하는 표.

도 10은 도 2의 디버그 모듈의 다수의 처리 상태 신호의 출력을 도시하는 표.

도 11은 본 발명의 한 실시예에 따른 파이프-라인 타이밍 예를 도시하는 그래픽.

도 12는 도 1의 데이터 처리 시스템의 중앙 처리 유니트의 한 부분을 도시하는 블럭도.

* 도면에 사용된 주요 부호의 설명 *

2 : 중앙 처리 유니트(CPU) 3 : 데이터 프로세서

4 : 메모리 관리 유니트(MMU)/제어기 6 : 메모리

7 : 외부 개발 시스템 8 : 시스템 버스 제어기

9 : 코어(Core) 10 : 디버그 모듈

12,13 : 슬레이브 모듈 14 : 마스터 모듈

15 : 제어 버스 20,60 : 제어 회로

25 : K-버스 30 : 직렬 인터페이스

32 : 레지스터 40 : 제어 레지스터

50 : 하드웨어 중단점 레지스터 80 : 멀티플렉서

100 : 중단점 회로 130,135,140,180 : 비교기

125 : 임시 PC 레지스터 170 : 임시 데이터 레지스터

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명의 분야

본 발명은 일반적으로 데이터 처리 시스템에 관한 것이고, 특히 데이터 처리 시스템에서의 디버그 기능을 수행하기 위한 방법 및 장치에 관한 것이다.

본 발명의 배경

데이터 처리 시스템이 설계된 대로의 동작에 실패하였을 때, 그 실패의 원인을 규정하기 위하여 다양한 분석 기술이 사용될 수 있다. 일반적으로 실패한 회로의 분리와 실패한 소프트웨어 프로그램의 정정을 돕기 위해 데이터 처리 시스템 내에서 추적 기능과 중단점 기능이 수행된다.

추적 기능은 외부 사용자가 데이터 처리 동작 실행의 중간 결과를 관측할 수 있도록 하기 위한 수단을 제공한다. 추적 기능은 일반적으로, 소프트웨어 프로그램의 각 지령 또는 소정 그룹의 지령이 데이터 처리 시스템에 의해 실행된 후, 데이터 처리 시스템 내에 포함된 선택된 레지스터 및 메모리의 상태를 제공한다. 선택된 레지스터 및 메모리의 상태를 반영함으로써 추적 기능은 외부 사용자에게 데이터 처리기 또는 데이터 처리 시스템의 내부 프로그래밍 모델에 대한 매우 상세한 정보를 제공한다. 이 정보로 많은 종류의 에러는 규정될 수 있고 결과적으로 정정될 수 있다. 중단점 기능 역시 데이터 처리 시스템 내의 에러를 갖는 소프트웨어 코드 또는 실패한 회로를 규정하고 분리하기 위한 방법을 제공한다. 사실상 중단점 기능은 사전에 프로그램된 이벤트가 발생하는 곳에서 소프트웨어 프로그램의 중단을 야기한다. 그러면 데이터는 소프트웨어 프로그램의 상태를 결정하기 위해 검색된다. 추적 기능과 같이, 중단점 기능은 외부 사용자가 선택된 레지스터 및 메모리 각각의 상태를 확인할 수 있도록 하여 데이터 처리 에러가 확인될 수 있도록 한다.

추적 기능과 중단점 기능은 현재 취득 가능한 데이터 처리 시스템에서는 집적되어 상기 설명된 분리 및 확인 기능을 제공한다. 예컨대 캘리포니아 산타 클라라의 인텔사로부터 취득 가능한 i960 Jx 마이크로프로세서는 추적 이벤트와 추적 실패를 생성하는 내부 중단점 레지스터를 갖는 집적된 회로의 마이크로프로세서이다. i960 Jx 마이크로프로세서에 있어서, 내부 중단점 레지스터는 지령 실행 어드레스 및 다양한 형태의 데이터 억세스의 어드레스 상에서의 트래핑에 기여한다. i960 Jx 마이크로프로세서의 추적 기능을 사용하기 위하여, 마이크로프로세서를 사용하는 소프트웨어는 실패 취급 절차 또는 디버깅 모니터 프로그램과의 인터페이스를 제공해야만 한다. 소프트웨어는 또한 다양한 추적 모드를 가능케 하고 추적 기능을 선택적으로 가능 또는 불가능케 하도록 수 가지의 레지스터 및 제어 비트를 연산하도록 요구된다.

유사하게 인텔사로부터 취득 가능한 80486 마이크로프로세서는 내부 중단점 레지스터를 갖고 추적 기능을 수행할 수 있는 집적된 회로의 마이크로프로세서이다. 80486 마이크로프로세서에 있어서 중단점 지령은 내부 소프트웨어 디버그 장치에 의한 사용을 위해 실행된다, 여기에서 내부 소프트웨어는 데이터 처리 시스템에서 실행 중인 소프트웨어를 단순히 칭한다. 일반 동작 중에 소프트웨어 디버그 프로그램은 실행되는 소프트웨어 코드의 형태에 따라 요구되는 모든 중단점에서 중단점 지령을 실행할 수 있다. 80486 마이크로프로세서는 또한 트랩 루틴이 실행된 후 인터럽트를 실행하는 단일 단계의 트랩 기능을 실행하기 위한 메카니즘을 제공한다.

발명이 이루고자 하는 기술적 과제

i960 Jx 마이크로프로세서와 80486 마이크로프로세서가 중단점과 추적 기능을 집적시키는 한, 마이크로프로세서의 디버그 기능은 마이크로프로세서가 정상 동작을 일시 중단하고 특정 디버그 예외 루틴으로 들어가는 것을 요구한다. 디버그 예외 루틴중 마이크로프로세서는 정상 동작 중과 같이는 기능 하지 않는다, 따라서 회로 또는 소프트웨어 내의 실패는 유사한 방법으로 발생하지 않는다. 결과적으로 실시간의 실패 조건이 실제로 일어나지 않으면, 외부 사용자는 마이크로프로세서 내에서 실패의 근본 원인을 구분하고 분리시킬 수 없을 수도 있다.

더욱이 i960 Jx 마이크로프로세서와 80486 마이크로프로세서는 둘다 중단점 및 추적 기능중 프로그램의 제어를 제공하기 위해 마이크로프로세서에 의해 실행되는 외부의 소프트웨어 디버그 프로그램을 필요로 한다. 이 같은 디버그 제어의 사용은 오버헤드 (관리 프로그램이 차지하는 시간)의 상당량을 요구하는 디버깅의 매우 강제적인 방법을 제공한다. 또한, 디버그 소프트웨어의 사용은 결함의 회로 또는 실패한 마이크로 프로세서내 순서의 진짜 반영을 제공하지 않을 수 도 있다.

따라서 실시간 추적 및 실시간 디버그 기능을 제공할 필요가 데이터 프로세서를 위하여 존재한다. 추적 및 디버그 기능이 둘 모두 특정 디버그 동작 모드가 아닌 정상 동작 중의 데이터 프로세서의 동작을 반영하는 방법으로 실행된다면, 외부 사용자는 회로 및 데이터 처리기의 소프트웨어 내의 실패를 보다 정확하게 규정하고 정정할 것이다.

발명의 구성 및 작용

양호한 실시예의 설명

본 발명은 특정 동작 모드의 필요 없이 실시간으로 중단점 및 추적 기능 둘 다를 제공할 수 있는 데이터 처리 시스템을 제공한다. 부가적으로 중단점 및 추적 기능은, 외부의 어드레스 버스 또는 외부의 데이터 버스를 통해 어드레스 및 데이터 정보를 제공하기 위해 시험되는 데이터 프로세서의 필요 없이, 데이터 처리 시스템에 의해 실행될 수 있다. 외부 버스로부터의 이 독립은 본 발명이, 크기, 형태, 프로토콜, 또는 외부 버스의 존재조차 무관하게 다양한 데이터 프로세서 시스템에 적용될 수 있는 효율적인 디버깅 메카니즘을 제공할 수 있게 한다.

부가적으로 본 발명은 추적 및 중단점 기능을 실시간으로 수행하기 위한 구조와 방법을 제공한다. 시험되는 데이터 프로세서는 전형적으로, 추적 및 중단점 기능이 실행되기 전에, 동작을 중단 또는 수정하도록 요구되지는 않는다. 그러므로 본 발명은 외부 사용자가, 데이터 프로세서가 디버깅 동작을 위한 특정 모드가 아닌 정상적으로 동작하는 동안, 실패를 분리할 수 있도록 한다. 또한 본 발명은 외부 사용자에게 실시간의 추적 또는 실시간의 디버그 동작 실행에 따른 데이터를 직접 제공한다. 추적 기능 또는 디버그 기능을 수행하기 위해 외부 소프트웨어 모니터 프로그램 및 외부 에뮬레이터를 필요로 하는 종래의 장치와는 다르게, 본 발명은 데이터 프로세서의 현재 동작을 지시하는 데이터를 외부 사용자에게 제공한다. 이 같이, 본 발명은 데이터 프로세서를 중대하게 간섭하거나 또는 해로운 영향을 미침이 없이 추적 및 디버그 기능을 제공한다. 본 발명의 동작은 후속 하여 보다 상세하게 설명된다.

다음의 논의에 있어서, 본 발명을 실현하는 데이터 처리 시스템의 한 실시예의 상호 통신 능력 및 동작이 보다 상세하게 설명된다.

본 발명의 상호 통신 능력

본 발명의 상호 통신 능력의 다음의 설명에 있어서, 데이터, 어드레스, 제어 또는 상태와 같은 하나 또는 보다 다양한 형태의 정보를 전송하기 위해 사용될 수 있는 다수의 신호 또는 도체를 참조하여 " 버스" 라는 단어가 사용된다. " 단정"

(assert) 및 " 부정" (negate)은 신호, 상태 비트 또는 유사한 장치를 각각 논리적으로 참 또는 논리적으로 부정 상태인지의 표현에 참조하여 사용된다. 논리적으로 참 상태가 논리 1 의 레벨이면, 논리적으로 부정 상태는 논리 0의 레벨이 된다. 또한 논리적으로 참 상태가 논리 0 의 레벨이면, 논리적으로 부정 상태는 논리 1의 레벨이 된다.

더욱이 숫자에 선행한 기호($)는 그 숫자가 16진수로 표현되었음을 표시한다. 숫자에 선행한 기호(%)는 그 숫자가 2진수로 표현되었음을 표시한다.

이제 도 1을 참조하여, 본 발명의 한 실시예에 따른 데이터 처리 시스템(5)을 도시한다. 데이터 처리 시스템(5)은 데이터 프로세서(3)와 외부 개발 시스템(7)을 포함한다. 데이터 프로세서(3)는 시스템 버스 제어기(8), 코어(9), 슬레이브 모듈(12), 슬레이브 모듈(13), 마스터 모듈(14) 및 디버그 모듈(10)을 포함한다. 시스템 버스 제어기(8)는 E-버스를 통해 외부 장치(도시 안됨)에 접속된다. 시스템 버스 제어기(8)는 S-버스를 통해 슬레이브 모듈(12)과 슬레이브 모들(13)에 접속된다. S-버스는 슬레이브 모듈 버스이다. 시스템 버스 제어기(8)는 M-버스(23)를 통해 코어(9)와 마스터 모듈(14)에 접속된다. M-버스(23)는 마스터 모듈 버스이다.

코어(9)는 중앙 처리 유니트(CPU)(2), 메모리 관리 유니트(MMU)/제어기(4), 및 메모리(6)를 포함한다. 중앙 처리 유니트(2), MMU/제어기(4), 메모리(6) 및 디버그 모듈(10)은 각각 K-버스(25)를 통해 서로 접속된다. 중앙 처리 유니트(2)와 메모리(6) 모두 MMU/제어기(4)에 직접 접속된다. MMU/제어기(4)는 M-버스(23)를 통해 데이터 처리 시스템의 나머지 부분에 정보를 제공한다. 중앙 처리 유니트(2)는 CPST신호를 제공하고 Stall 신호를 수신하기 위해 디버그 모듈(10)에 접속된다. CPU(2) 역시 외부 장치로부터 리세트 신호를 수신한다. 디버그 모듈(10)은 프로세서 상태(PST) 신호, 디버그 데이터(DDATA) 신호, 및 개발-직렬-데이터-아웃(DSDO) 신호를 외부 사용자에게 제공한다. 디버그 모듈(10)은 개발-직렬-데이터-인(DSDI) 신호, 개발-직렬-클럭(DSCLK), 및 중단점($\overline{BKPT}$) 신호를 데이터 처리 시스템(5)의 외부 장치(도시 안됨)로 부터 수신한다.

도 2는 디버그 모듈(10)을 매우 상세하게 도시한다. 디버그 모듈(10)은 제어 회로(20), 직렬 인터페이스(30), 레지스터(32), 다수의 제어 레지스터(40), 다수의 하드웨어 중단점 레지스터(50), 제어회로(60), FIFO(70), 다수의 멀티플렉서(80), 및 중단점 회로(100)를 포함한다.

$\overline{BKPT}$ 신호는 외부 집적된 회로 핀(도 2에 도시 안됨)을 통해 제어 회로(20)에 제공된다. 제어 회로(20)는 다수의 비트를 갖는 상태 신호를 수신하기 위해 다수의 제어 레지스터(40)에 접속된다. 제어 회로(20)는 다수의 비트를 갖는 중단점 신호를 수신하기 위해 중단점 회로(100)에 접속된다. 제어 회로(20)는 다수의 비트를 갖는 추적 신호를 제공하기 위해 제어회로(60)에 접속된다. 제어회로(20)는 제어 버스(15)를 통해 다수의 제어 레지스터(40), 다수의 하드웨어 중단점 레지스터(50), 및 레지스터(32)에 접속된다. 제어회로(20)는 다수의 중단점 제어 신호와 버스 요구 신호를 CPU(2)에 제공한다. 제어회로(20)는 또한 CPU(2)에 의해 제공된 버스 그랜트 신호를 수신한다.

다수의 제어 레지스터(40)는 레지스터(32)로 부터 멀티-비트 직렬 정보 신호를 수신한다. 다수의 제어 레지스터(40)는 다수의 비트를 갖는 구성 신호를 제어회로(60)에 제공한다. 중단점 회로(100)는 K-버스로부터 KADDR(K 어드레스)신호와 KDATA 신호를 수신한다. 다수의 하드웨어 중단점 레지스터(50) 또한 다수의 비트를 갖는 내부 데이터 신호를 레지스터(32)에 제공한다. K-버스(25) 역시 다수의 하드웨어 중단점 레지스터(50)에 제공된다. K-버스는 KADDR 신호, KDATA 신호, 및 KCONTROL 신호의 모음임을 주목해야 한다. KADDR, KDATA, 및 KCONTROL 신호 각각은 중단점 회로(100)에 입력된다.

직렬 인터페이스(30)는 제 1 및 제 2 집적회로 핀으로부터 DSDI및 DSCLK 신호를 각각 수신한다. 직렬 인터페이스(30)는 DSDO신호를 제 3의 집적회로 핀에 제공한다. 직렬 인터페이스 (30)은 레지스터(32)에 접속된다. 레지스터(32)는 멀티-비트 직렬 정보 신호를 제공하기 위해 다수의 제어 레지스터(40)와 다수의 하드웨어 중단점 레지스터(50)에 접속된다.

제어회로(60)는 Stall 신호를 CPU(2)에 제공한다. 제어회로(60) 역시 FIFO 제어 신호를 FIFO(70)에 Mux 제어 신호를 다수의 멀티플렉서(80)에 제공한다. FIFO(70)는 KADDR 및 KDATA신호를 수신하기 위해 K-버스에 접속된다. FIFO(70)는 또한 FIFO 신호를 제공하기 위해 다수의 멀티플렉서(80)에 접속된다. FIFO(70)는 또한 Full 신호를 제어회로(60)에 제공한다. 다수의 멀티플렉서(80)는 CPU(2)로부터 CPST신호를 수신한다. 다수의 멀티플렉서(80)는 DDATA와 PST신호를 제공한다.

제어회로(20)는 직렬 인터페이스(30)를 동작시키고 레지스터 (32)내에서 정보의 저장을 제어하기 위해 요구되는 신호를 제공한다. 레지스터(32)에 데이터가 로드 되면, 제어회로(20)는 이 데이터의 전송을 제어회로(40) 또는 다수의 하드웨어 중단점 레지스터(50)내의 적절한 곳으로 방향을 지정한다. 제어회로(20)에 의해 실행되는 제어 기능은 제어 버스(15)에 의해 통신된다. 로드 동작 실행의 제어에 덧붙여, 제어회로(20)는 제어회로(40)로 부터 상태 정보와 중단점 회로(100)로 부터 중단점 정보를 수신한다. 제어회로(20)는 버스 요구 신호, 중단점 제어 신호, 및 추적 신호를 생성하기 위해 상태 정보와 중단점 정보를 집합적으로 사용한다.

도 3은 중단점 회로(100)를 매우 상세하게 도시한다. 중단점 회로(100)는 어드레스 중단점 하이 레지스터(ABHR)(110), 어드레스 중단점 로우 레지스터(ABLR)(120), 비교기(130), 비교기(140), 데이터 중단점 마스크 레지스터(DBMR)(150), 데이터 중단점 레지스터(DBR)(160), 임시 데이터 레지스터(170), 비교기(180), 프로그램 카운터 중단점 마스크 레지스터(PBMR)(190), 프로그램 카운터 중단점 레지스터(PBR)(115), 임시 프로그램 카운터 레지스터 (125), 및 비교기(135)를 포함한다.

제어 버스(15)는 ABHR(110), ABLR(120), DBMR(150), DBR(160), PBMR(190), 및 PBR(115)의 각 입력에 접속된다. 직렬 정보 신호는 ABHR(110), ABLR(120), DBMR(150), 및 DBR(160)의 각 입력에 접속된다. KCONTROL 신호는 PBMR(190), PBR(115), 및 임시 PC 레지스터(125)의 각 입력에 접속된다. ABHR(110)의 제 1의 출력은 KADDR신호에 접속된다. 제어 버스(15) 역시 임시 레지스터(170 및 125)에 접속된다. 직렬 정보 신호는 비교기(130)의 제 1의 입력에 접속된다. KADDR 신호는 비교기(130)에 제 2의 입력을 제공한다. ABLR(120)은 비교기(140)에 제 1의 입력을 제공하고 KADDR 신호는 비교기(140)에 제 2의 입력을 제공한다. 비교기(130)의 출력과 비교기(140)의 출력은 둘 모두 중단점 신호 관리자에 접속된다.

DBMR(150)의 출력은 비교기(180)의 인에이블 입력에 접속된다. DBR(160)의 출력은 비교기(180)의 제 1의 입력에 접속된다. KDATA신호는 임시 데이터 레지스터(170)의 입력에 접속된다. 임시 데이터 레지스터(170)의 출력은 비교기(180)의 제 2의 입력및 내부 데이터 신호에 접속된다. 비교기(180)의 출력은 중단점 신호 관리자에 접속된다.

PBMR(190)의 출력은 비교기(135)의 인에이블 입력에 접속된다. PBR(115)의 출력은 비교기(135)의 제 1의 입력에 접속된다. KCONTROL신호는 임시 프로그램 카운터 레지스터(125)의 입력에 접속된다. 임시 PC 레지스터(135)의 출력은 비교기(135)의 제 2의 입력에 접속된다. 비교기(135)의 출력은 중단점 신호 관리자에 접속된다.

도 12는 중앙 처리 유니트(2)를 매우 자세하게 도시한다. 중앙 처리 유니트(2)의 부분은 지령 페치 파이프라인(IFP)(210)과 피연산자 실행 파이프라인(OEP)(220)을 포함한다. IFP(210)는 지령 어드레스 생성 회로(202), 지령 페치 회로(204), 및 FIFO 지령 버퍼(206)를 포함한다. OEP(220)는 피연산자 페치 회로(212)와 어드레스 생성 회로(214)를 포함한다.

지령 어드레스 생성 회로(202)의 출력은 버퍼(208)의 제 1의 입력에 접속된다. 지령 페치 회로(204)의 출력은 FIFO 지령 버퍼(206)및 피연산자 페치 회로(212)에 접속된다. FIFO 지령 버퍼(206)의 출력은 피연산자 페치 회로(212)에 접속된다. 어드레스 생성 회로(214)의 제 1의 출력은 버퍼(208)의 제 2의 입력에 접속된다. 어드레스 생성 회로(214)의 제 2의 출력은 버퍼(216)에 접속된다.

버퍼(216)의 제 1의 출력은 피연산자 페치 회로(212)와 지령 페치 회로(204)에 접속된다. 버퍼(216)의 제 2의 출력은 KDATA신호를 제공한다. KDATA 신호는 또한 버퍼(216)에 제공된다. 버퍼(208)의 출력은 KADDR신호를 제공한다.

레지스터 설명

도 4는 다수의 제어 레지스터(40)를 자세하게 도시한다. 본 발명의 한 실시예에 있어서, 다수의 제어 레지스터(40)의 부분은 트리거 한정 레지스터(TDR)및 구성/상태 레지스터(CSR)를 포함한다.

TDR은 디버그 모듈(10)의 구성을 저장하고 미리 한정된 조건하에서 실행되는 선택된 기능을 위한 제어를 제공한다. 본 발명의 현재의 실시예에 있어서, 트리거는 31 내지 16 비트가 제 2의 레벨 트리거를 한정하고 15 내지 0 비트가 제 1의 레벨 트리거를 한정하는 하나 또는 두 개의 레벨 트리거로 구성될 수 있다. 두개의 레벨 트리거로 트리거가 구성될 때, 방법상의 트리거 기능은 조건적인 구성과 유사하다. 예컨대 제 1의 조건 또는 트리거의 레벨이 만족되면, 제 2의 조건 또는 레벨은, 트리거가 데이터 처리 시스템(5)에 의해 실행될 중단점을 가능케 하기 전에, 만족해야만 한다. 다음과 같은 구문으로 작성될 수 있다.

If 1st condition is satisfied,

then if 2nd condition is satisfied

then TRIGGER.

TDR에 포함된 각 비트의 기능은 후속 하여 보다 자세하게 설명된다.

TDR은 보다 자세하게 도 7에 도시되었다. TDR의 PCI 비트는 프로그램 카운터 중단점 역비트이다. PCI비트가 " 단정" 이면, 프로그램 카운터 중단점은, 프로그램 카운터 중단점(PBR)과 프로그램 카운터 중단점 마스크(PBMR) 레지스터에 의해 한정된 범위 밖에서 시작된다. PCI비트가 " 부정" 이면, 프로그램 카운터 중단점은 PBR과 PBMR에 의해 한정된 영역 내에서 한정된다.

인에이블 프로그램 카운터 중단점(EPC) 비트는 이것이 단정일 때 프로그램 카운터 중단점, PBR, 및 PBMR에 대한 트리거의 의존을 선택적으로 가능케 한다. EPC비트가 부정이면, 프로그램 카운터 중단점은 불가능하다.

단정이면, 인에이블 어드레스 중단점 로우(EAL)비트는, 중단점 트리거로 기능 하도록 어드레스를 다수의 하드웨어 중단점 레지스터(50)의 어드레스 중단점 레지스터의 ABLR내에 저장될 수 있게 한다. 따라서 EAL비트가 단정이면, KADDR 신호를 통해 전송된 어드레스 값이 어드레스 중단점 레지스터의 ABLR내에 저장된 어드레스 값과 동일할 때, 중단점은 트리거 한다.

인에이블 어드레스 중단점 범위(EAR)비트는 중단점이 , 이 비트가 단정일 때 다수의 하드웨어 중단점 레지스터(50)의 어드레스 중단점 레지스터의 ABLR및 ABHR에 의해 한정된 총괄적인 범위로, 트리거할 수 있도록 한다. 따라서 EAR비트가 단정이면, KADDR 신호를 통해 전송된 어드레스 값이 어드레스 중단점 레지스터의 ABLR및 ABHR내에 저장된 어드레스 값에 의해 한정된 범위 내에 있을 때, 중단점은 트리거 한다.

단정이면, 인에이블 어드레스 중단점 반전(EAI)비트는, 다수의 하드웨어 중단점 레지스터(50)의 어드레스 중단점 레지스터의 ABLR및 ABHR에 의해 한정된 범위의 배타적인 어드레스 범위 또는 밖에서 중단점이 트리거할 수 있게 한다. 따라서 EAI비트가 단정일 때, KADDR신호를 통해 전송된 어드레스 값이 어드레스 중단점 레지스터의 ABLR내에 저장된 어드레스 값보다 적거나 또는 어드레스 중단점 레지스터의 ABHR내에 저장된 어드레스 값보다 클 때, 중단점은 트리거 한다.

데이터 중단점 역전(DI)비트는 다수의 데이터 중단점 비교기의 논리적인 감을 그것이 단정일 때 역전시킨다. DI 비트는 다수의 하드웨어 중단점 레지스터(50)의 DBR에 프로그램된 데이터 값과는 동일하지 않은 데이터 값의 발생에 기초한 트리거를 개발하기 위해 사용될 수 있다.

단정일 때, 상부의 상부 데이터 바이트에 대한 인에이블 데이터 중단점(EDUU)비트는 KDATA 신호 및 K-버스(25)를 통해 전송되는 데이터 값의 31 내지 24 비트에 기초한 데이터 중단점 트리거를 가능케 한다. 유사하게 단정일 때, 상부의 중간 데이터 바이트에 대한 인에이블 데이터 중단점(EDMU)비트는 KDATA 신호 및 K-버스(25)를 통해 전송되는 데이터 값의 23 내지 16 비트에 기초한 데이터 중단점 트리거를 가능케 한다. 또한 단정일 때, 하부의 중간 데이터 바이트에 대한 인에이블 데이터 중단점(EDLM)비트는 KDATA 신호 및 K-버스(25)를 통해 전송되는 데이터 값의 15 내지 8 비트에 기초한 데이터 중단점 트리거를 가능케 한다. 또한 단정일 때, 하부의 하부 데이터 바이트에 대한 인에이블 데이터 중단점(EDLL)비트는 KDATA신호 및 K-버스(25)를 통해 전송되는 데이터 값의 8 내지 0 비트에 기초한 데이터 중단점 트리거를 가능케 한다.

상부 데이터 워드에 대한 인에이블 데이터 중단점(EDWU)비트는 단정일 때 KDATA 신호 및 K-버스(25)를 통해 전송되는 데이터 값의 31 내지 비트 16에 기초한 데이터 중단점 트리거를 가능케 한다. 하부 데이터 워드에 대한 인에이블 데이터 중단점(EDWL)비트는 단정일 때 KDATA신호 및 K-버스(25)를 통해 전송되는 데이터 값의 15 내지 비트 0에 기초한 데이터 중단점 트리거를 가능케 한다.

데이터의 긴(long)워드에 대한 인에이블 데이터 중단점(EDLW) 비트는 단정일 때 KDATA 신호 및 K-버스(25)를 통해 전송되는 데이터 값의 31 내지 비트 0에 기초한 데이터 중단점 트리거를 가능케 한다. EDUU, EDUM, EDLM, EDLL, EDWU, EDWL, 및 EDLW 비트중 어느 것의 단정이라도 데이터 중단점 트리거를 가능케 함을 주목해야 한다. 일반적으로 요구되는 것과 같이, 주어진 조건은 데이터 중단점이 실제로 트리거 되기 전에 발생해야만 한다. EDUU, EDUM, EDLM, EDLL, EDWU, EDWL, 및 EDLW 비트의 각각이 부정이면, 데이터 중단점 트리거는 본 발명의 한 실시예에 있어서는 불가능하다.

덧붙여, 본 발명의 현 실시예에 있어서 데이터 중단점 트리거는 잘못 정렬 또는 비정렬된 피연산자 레퍼런스에 대해 생성될 수 있음을 주목해야 한다. 잘못 정렬 또는 비정렬된 피연산자 레퍼런스에 대한 중단점 트리거의 이러한 생성은 다음의 방법론을 사용한다. 바이트 크기의 레퍼런스에 대해, 데이터 비트의 적절한 세트는 피연산자 어드레스의 두개의 낮은 순위의 비트의 기능으로써 프로그램될 수 있을 것이다. 다음의 예를 참조.

If KADDR【1:0】=00, then KDATA【31:24】 is programmed;

If KADDR【1:0】=01, then KDATA【23:16】 is programmed;

If KADDR【1:0】=10, then KDATA【15:8】 is programmed; and

If KADDR【1:0】=11, then KDATA【7:0】 is programmed.

워드-크기의 레퍼런스에 대해, KDATA 신호를 통해 전송된 데이터 값의 31에서 16까지의 비트 와 15에서 0까지의 비트는 KADDR 신호의 비트 1의 기능으로써 프로그램될 것이다. KADDR 신호의 비트 1이 논리 '0'의 값이면, 비트 31 내지 16은 프로그램된다. KADDR 신호의 비트 1이 논리 '1'의 값이면, 비트 15 내지 0은 프로그램된다. 긴 워드 레퍼런스에 대해, DADDR 신호의 비트 31 내지 0이 프로그램된다.

인에이블 중단점 레벨(EBL) 비트는 단정일 때 중단점 트리거에 대해 전체적으로 가능하게 작용한다. EBL 비트가 단정이 아니면, 모든 중단점 트리거는 불가능하게 된다.

트리거 응답 제어(TRC) 값은 데이터 처리 시스템(5)이 종료된 트리거 조건에 응답하는 방법을 결정한다. 트리거 응답은 다수의 집적 회로 핀에 의해 제공되는 DDATA 신호를 통해 외부 사용자에게 디스플레이된다. TRC 값이 %01의 논리 값을 가질 때, 중단점 회로(100)에 의해 검출된 하드웨어 중단점 조건의 발생은 제어 회로(20)가 중단점 신호를 CPU(2)에 보낼 수 있게 한다. 디버그 인터럽트 신호가 단정이면, CPU(2)는 인터럽트 가능 시점에서 정상 실행을 중단하고 소정의 예외 처리 루틴을 실행한다.

구성 상태 레지스터(CSR)가 도 8에 매우 상세하게 도시되었다. CSR은 CPU(2), MMU/제어기(4), 메모리(6), 및 데이터 처리 시스템(5)의 디버그 모듈(10)의 동작 구성을 한정한다. 데이터 처리 시스템(5)의 구성의 한정에 추가에 있어서, CSR은 중단점 회로(100)의 상태를 지시하는 상태 정보를 포함한다.

데이터 처리 시스템이 리셋트될 때, CSR은 클리어됨을 주목해야 한다. 더욱이, 데이터 처리 시스템이 동작의 백그라운드 디버그 모드일 때, CSR은 외부 개발 시스템(도시 안됨)에 의해 판독 및 기록될 수 있다. 동작의 백그라운드 디버그 모드는 후속 하여 보다 상세하게 설명된다. CSR에 포함된 각 비트의 기능은 후속하여 보다 상세하게 설명된다.

도 7에 도시된 바와 같이, CSR은 데이터 처리 시스템(5)내에서 하드웨어 중단점 기능에 대한 판독만의 상태 정보를 한정하고 또한 공급하는 중단점 상태(상태) 필드를 포함한다. 중단점 상태 필드에 저장된 정보는 중단점 회로(100)에 의해 제공되는 중단점 트리거 정보에 따라서 다수의 제어 레지스터(40)에 의해 생성된다. 상태 필드에 저장된 정보는 다수의 제어 레지스터(40)내의 TDR의 기능이다. 상태 필드가 $0의 논리 값을 저장할 때, 어떠한 하드웨어 중단점도 가능하게 되지 않는다. 상태 필드가 $1의 논리 값을 저장할 때, 디버그 모듈(10)은 트리거될 레벨 1의 중단점을 기다린다. 이전에 설명한 바와 같이, 레벨 1의 중단점은 속성은 트리거 한정 레지스터 (TDR)의 비트(31 내지 16)에 의해 한정된다. 논리값($6)이 CSR의 상태 필드에 저장되면, 레벨 2의 중단점이 트리거된다.

CSR의 상태 필드 내에서 반영된 중단점 상태는 DDATA신호를 제공하는 다수의 외부 집적 회로 핀을 통해 외부 사용자(도 2에서 도시 안됨)에게 제공된다. 도 9를 참조. 신호가 K-버스(25)상에서 포착된 디스플레이 데이터가 아닐 때, DDATA 신호는 중단점 상태를 반영한다. 덧붙여, CSR의 내용은 어느 순간에도 직렬 인터페이스(30)를 사용하여 판독될 수 있음을 주목해야 한다. 따라서 중단점 상태는 DDATA 신호가 디스플레이 포착 데이터가 아닐 때 DDATA 신호와, 직렬 인터페이스(30)를 사용하는 READ CSR명령의 두 수단을 통해 유용하다.

CSR은 또한 백그라운드 디버그 모드로 입력 상태를 외부 개발 시스템(7)에 지시하는 4 개 비트 값을 제공한다. 4개 비트값중 제 1의 비트는 FOF 비트이다. 단정일 때, Fault-on-fault(FOF) 비트는 데이터 트리 시스템(5) CPU(2)의 돌발적인 정지가 발생했고 데이터 처리 시스템(5)은 동작의 백그라운드 디버그 모드로 진입하도록 강요된 것을 나타낸다. 동작의 백그라운드 리버그 모드는 후속 하여 보다 상세하게 설명된다. FOF 비트는 CSR의 내용 판독에 따라 클리어된다.

단정일 때, 하드웨어 중단점 트리거(TGR) 비트는 하드웨어 중단점이 데이터 처리 시스템(5)의 CPU(2) 동작을 정지시켰음을 표시한다. 단정일 때, TRG 비트는 데이터 처리 시스템(5)이 동작의 백그라운드 디버그 모드로 진입하도록 강요되었음을 표시한다. TGR 비트는 CSR 내용의 판독 또는 직렬 인터페이스(30)를 통해 직렬로 제공되는 " GO" 명령의 수신에 따라 클리어된다.

프로세서 정지(Halt) 비트는, 단정일 때 CPU(2)는 정지해야만 하는 것을 규정하는 지령을 CPU(2)가 실행했음을 표시한다. 단정일 때 정지 비트는 데이터 처리 시스템(5)이 동작의 백그라운드 디버그 모드로 진입하도록 강요된 것을 표시한다. 정지 비트는 CSR 내용의 판독 또는 직렬 인터페이스(30)를 통해 직렬로 제공되는 " GO" 명령의 수신에 따라 클리어된다.

BKPT 단정(BKPT) 비트는 $\overline{BKPT}$ 신호가 단정이었고 CPU(2)에 제공된 것을 표시한다. 단정일 때, BKPT 비트는 데이터 처리 시스템(5)이 동작의 백그라운드 디버그 모드로 진입하도록 강요된 것을 표시한다. BKPT 비트는 CSR 내용의 판독 또는 직렬 인터페이스(30)를 통해 직렬로 제공되는 " GO" 명령의 수신에 따라 클리어된다.

단정일 때, 디버그 레지스터에 대한 프로세서 기록 금지 비트(IPW)는 다수의 하드웨어 중단점 레지스터(50)와 다수의 제어 레지스터(40)에 기록 동작을 금지시킨다. IPW 비트는 외부 개발 시스템(7)과 같은 외부 개발 시스템으로부터 제공되는 명령에 의해서만이 수정될 수 있다. IPW 비트는, 외부 개발 시스템(7)에 의해 다수의 제어 레지스터(40)와 다수의 중단점 레지스터(50)에 기록된 디버그 정보의 재기록으로부터 CPU(2)를 효과적으로 로크아웃시킨다.

단정일 때, 에뮬레이터 모드에서 강제 프로세서 레퍼런스(MAP) 비트는 데이터 처리 시스템이 에뮬레이터 모드에서 동작할 때 CPU(2)가 모든 레퍼런스를 특정 어드레스 공간에 맵(map)하도록 강요한다. MAP 비트가 부정이면, 모든 에뮬레이터 모드 레퍼런스는 수퍼바이저 텍스트와 메모리(6)내의 데이터 공간에 맵된다.

본 발명은 수퍼바이저의 논리 확장 또는 특권 모드인 특별 동작 모드로써 에뮬레이터 모드를 포함한다. 동작의 이 특별 모드는 본 발명의 현재의 실시예에서 세 개의 가능한 지시자중 하나의 단정에 따라 진입된다. 에뮬레이터 모드에 한번 진입하면, 모든 정상 입력/출력 인터럽트는 무시될 수 있고, 데이터 프로세서(3)는 모든 메모리 레퍼런스를 '대체 공간'(alternate space)에 선택적으로 강요할 수 있다. MAP 비트는 데이터 프로세서(3)의 이 동작을 제어한다. 동작의 에뮬레이터 모드는 외부 개발 시스템(7)이 모든 메모리 레퍼런스를 이 대체 공간에 대행 수신(intercept)하고 일부 형태의 디버그 기능을 지원하는데 요구되는 지령과 데이터 값을 제공하는 것을 허용하기 위하여 전형적으로 사용된다. 외부 개발 시스템(7)이 상태 또는 정상 시스템 메모리를 변화시킴이 없이 데이터 프로세서(3)의 동작을 이득 제어하도록 허용하기 위해 대체 공간의 재맵핑이 제공된다. 본 발명의 현재의 실시예에 있어서, 데이터 프로세서(3)는 RTE(예외에서 복귀)와 같은 특별 지령이 수행될 때까지 에뮬레이터 모드를 유지한다. 이 때에 데이터 프로세서(3)는 에뮬레이터 모드를 빠져나가고 CPU(2)는 정상 동작 모드로 복귀한다.

본 발명의 현재의 실시예에 있어서, 에뮬레이터 모드로 진입과 이로부터의 빠져나감은 PST 신호를 통해 외부 사용자에게 신호 출력됨을 주목해야 한다. PST 신호에서 $D의 값은 에뮬레이터 모드로 진입을 표시하고 $7의 값은 에뮬레이터 모드로부터 빠져나감을 표시한다. PST 신호에 의한 에뮬레이터 동작 모드로 진입과 이로부터 빠져나감의 표시는, 이것이 하드웨어 중단점이 트리거 되는 표시를 외부 개발 시스템(7)에 제공하기 때문에, 특별히 중요하다. 따라서 직렬 인터페이스(30)를 통해 제공되는 직렬 포트는 외부 메모리 위치로부터 디버그 정보를 검색하기 위해 사용될 수 있다.

MAP 비트가 부정이면, 모든 에뮬레이터 모드 메모리 레퍼런스는 정상으로 수퍼바이저 억세스에 맵핑되어, 온-칩 메모리, 외부 메모리, 및 입력/출력 장치와 같은 시스템 자원이 레퍼런스될 수 있게 한다.

덧붙여, 본 발명의 현재의 실시예에 있어서, 에뮬레이터 동작 모드로 진입하기 위한 세 개의 방법이 실현된다. 제 1의 방법에 있어서, 트레이스 예외 상에서 강제 에뮬레이터 모드(TRC)비트가 단정이면, 소프트웨어 트레이스 예외의 발생은 데이터 프로세서(3)를 에뮬레이터 동작 모드로 진입하도록 강요한다. 이는 외부 개발 시스템(7)에 의해 제어되고 데이터 프로세서(3)상에서 실행 중인 소프트웨어 디버거(debugger)에 의해서는 제어되지 않는 단일의 지령 디버거 루틴을 생성시키기 위한 메카니즘이다.

제 2의 방법에 있어서, 강제 에뮬레이터 모드(EMU)비트가 단정이면, 데이터 프로세서(3)는 예외 프로세서의 리세트를 시작한다. 또다시, 이 방법은 외부 개발 시스템(7)이, 데이터 처리 시스템(10)이 지령 처리를 시작하기 전에, 이를 제어하는 것을 허용한다.

제 3의 방법은, 디버그 인터럽트가 트리거된 하드웨어 중단점에 따라서 단정일 때, 데이터 프로세서(3)의 에뮬레이터 모드로의 진입을 가능케 한다. TDR의 TRC 필드는 에뮬레이터 모드로 진입을 갖는 디버그 인터럽트를 생성시키기 위한 하드웨어 중단점 트리거의 사용을 프로그램하기 위해 사용될 수 있다.

디버그 데이터 제어(DDC)값은 DDATA 신호를 통해 외부 개발 시스템에 제공될 피연산자 데이터를 포착하기 위한 구성 제어 정보를 제공한다. DDC값이 %00일 때, DDATA 신호 상에는 피연산자 데이터는 디스플레이되지 않는다. DDC값이 %01일때, M-버스 (23)을 통해 통신되는 모든 기록 데이터는 포착되고 DDATA신호 상에 디스플레이된다. DDC값이 %10일 때, M-버스(23)를 통해 통신되는 모든 판독 데이터는 포착되고 DDATA 신호 상에 디스플레이된다. DDC값이 %11일 때, M-버스 (23)을 통해 통신되는 모든 기록 및 판독 데이터는 포착되고 DDATA 신호 상에 디스플레이된다. DDC값에 의해 규정되는 정보의 형태에 관계없이, DDATA 신호는 피연산자 레퍼런스 크기에 의해 한정된 바이트 수를 디스플레이한다. 피연산자가 CSR의 DDC 필드의 제어 하에서 포착되었을 때, DDATA 신호 상에 디스플레이되는 바이트 수는 실행되는 레퍼런스의 형태에 의해 결정된다. 피연산자가 바이트로써 레퍼런스 되면, 8 비트의 데이터가 디스플레이된다. 피연산자가 워드로써 레퍼런스 되면, 16 비트의 데이터가 디스플레이된다. 유사하게 피연산자가 긴 워드로써 레퍼런스 되면, 32 비트의 데이터가 디스플레이된다. DDC는 M-버스의 기록 및 판독 동작을 포착할 수 있게 한다. 그러나, 피연산자 레퍼런스 크기는 DDATA 신호 상에 실제적으로 디스플레이되는 바이트 수를 결정한다.

브랜치 타겟 바이트(BTB) 값은 DDATA 신호 상에 디스플레이될 브랜치 타겟 어드레스의 바이트 수를 한정한다. BTB 값이 %00 일 때, 브랜치 타겟 어드레스의 바이트는 DDATA 신호 상에 디스플레이되지 않는다. BTB 값이 %01 일 때, 브랜치 타겟 어드레스의 하위 두 개 바이트가 DDATA 신호 상에 디스플레이된다. BTB 값이 %10 일 때, 브랜치 타겟 어드레스의 하위 세 개 바이트가 DDATA 신호 상에 디스플레이된다. BTB 값이 %11 일 때, 브랜치 타겟 어드레스의 전체 4 개 바이트가 DDATA 신호 상에 디스플레이된다. DDATA 신호는 일반적으로 최하위에서 최상위 순서로 데이터를 출력함을 주목해야 한다. 이 순서는 포착된 피연산자와 브랜치 타겟 어드레스에 대해서도 동일하다. 더욱이, 동작 중에 제어 회로(60)의 제어 하에서 FIFO 저장 버퍼(70)는 변화 어드레스 모드를 사용하는 선택된 브랜치 동작과 관련된 이들 타겟 어드레스만을 포착할 따름이다. 변화 어드레스 모드는, 어드레스가 프로그램 카운터 관련물 또는 절대 어드레스에 의해 제공되지 않고 다른 수단에 의해 계산되는 모드이다. 이와 같은 타겟 어드레스는 전형적으로 예외 벡터, 예외로부터 복귀(TRE), 서브루틴으로부터 복귀(RTS) 지령, 및 점프(JMP)와 서브루틴으로 점프(JSR)의 형태에 관련된다.

가능 디버그 모드(END) 비트는 PST와 DDATA 신호가 단정이면, 이의 정상 동작을 불가능케 한다. 더욱이 이 비트가 가능하면, 데이터 프로세서(3)의 CPU(2)는 내부 상태 정보의 8 비트 벡터를 출력한다.

단정일 때, 비-파이프라인-모드(NPL) 비트는 CPU(2)를 비-파이프라인 동작 모드로 동작하도록 강요한다. 비-파이프라인 동작 모드에 있을 경우, 데이터 처리 시스템(3)은 둘 이상의 연속 지령 사이에 겹침이 없는 시간에 단일의 지령을 효과적으로 실행한다.

단정일 때, 무시 펜딩 인터립트(IPI) 비트는 데이터 처리 시스템(5)이 단일-지령-단계 모드에서 동작 중일 때, CPU(2)가 어떠한 펜딩 인터럽트 요구도 무시하도록 강요한다.

단일-단계-모드(SSM) 비트는 단정일 때 CPU(2)가 단일-지령 -단계 모드에서 동작하도록 강요한다. 단일-지령-단계 모드에 있는 동안 CPU(2)는 단일 지령을 실행한 후 정지한다. CPU(2)가 정지해 있는 동안, 백그라운드 디버그 모드와 관련된 명령은 모두 실행될 수 있다. GO 명령의 수신 상에, CPU(2)는 다음의 지령을 실행한 후 다시 정지한다. 이 처리는 단일-지령-단계 모드가 불가능이 될 때까지 지속된다.

저장 버퍼 불가능(SBD) 비트, K-버스 RAM 불가능(KAD) 비트, K-버스 ROM 불가능(KOD) 비트, 및 K-버스 캐쉬 불가능(KCD) 비트는 외부 개발 시스템(7)이 더 버그 목적을 위한 데이터 프로세서(3)의 정상 구성을 무시하는 것을 허용하기 위해 집합적으로 사용된다. 특히, SBD 비트 MMU/제어기(4)내에 위치하고 FIFO 저장 버퍼(70)와 혼동되지 않아야 하는 선택적인 FIFO 저장 버퍼의 사용을 불가능케 한다. KAD 비트는 메모리 블럭(6)의 사용이 통과되고 M-버스(23) 억세스의 강요를 허용한다. 메모리 블럭(6)은 RAM 모듈, ROM 모듈, 및/또는 캐쉬 모듈을 포함할 수 있다. KAD, KOD, 및 KCD 비트는 각각의 이들 모듈에 의해 제공되는 기능을 불가능케 하기 위해 각각 사용된다.

다수의 하드웨어 중단점 레지스터(50)는 도 5에 상세하게 도시되었다. 다수의 각 하드웨어 중단점 레지스터(50) 기능이 여기에서 설명된다. 다수의 각 하드웨어 중단점 레지스터(50) 기능의 상세한 설명이 후속 하여 상세하게 설명된다.

도 5에 있어서, 어드레스 중단점 레지스터(ABLR 및 ABHR)는 중단점 기능을 트리거 하기 위해 사용될 수 있는 데이터 처리 시스템(5)의 피연산자 논리 어드레스 공간 내에 영역을 한정한다. 각 ABLR 및 ABHR내에 저장된 어드레스 값은 K-버스(25)를 통해 전송되는 어드레스 값과 비교된다. 덧붙여, 다음의 논의에서 중단점 기능을 실행하기 위해 부합되어야 하는 조건의 집합은 트리거로써 참조됨을 주목해야 한다.

어드에스 속성 중단점 레지스터(AABR)는 도 6에 보다 상세하게 도시되었다. 어드레스 속성 중단점 레지스터는 중단점 기능을 위한 트리거에서 부합될 다수의 어드레스 속성과 마스크를 한정한다. ABLR 및 ABLR어드레스 값과 유사하게, AABR내에 저장된 속성값은 K-버스를 통해 전송되는 어드레스 속성 신호와 비교된다.

AABR의 하위 5 개 비트, 전송 형태(TT)값과 전송 변경자 (Modifier)(TM)값은 데이터 처리 시스템(5)이 백그라운드 디버그 모드 중에서 메모리 레퍼런스를 실행할 때 사용되는 어드레스 공간을 집합적으로 한정한다. TT 값은 전송이 수퍼바이저 피연산자, 수퍼바이저 지령, 사용자 피연산자 또는 사용자 지령인지를 표시한다.

다수의 각 하드웨어 중단점 레지스터(50)는 두 가지 기능을 함에 주목해야 한다. 데이터 처리기(3)가 백그라운드 디버그 동작 모드로 동작 중일 때, 다수의 하드웨어 중단점 레지스터(50)는 메모리 및 프로세서 레퍼런싱 동작의 실행 중에 사용된다. 데이터 프로세서(3)가 백그라운드 디버그 동작 모드에서 동작 중이 아닐 때, 다수의 하드웨어 중단점 레지스터(50)는 하드웨어 중단점을 검출하기 위해 사용된다.

다수의 하드웨어 중단점 레지스터(50)가 하드웨어 중단점을 검출하기 위해 사용될 때, AABR 레지스터는 4 개 필드 더하기 4 개의 동등한 마스크 필드를 포함한다. TT및 TM 필드는 억세스 되는 어드레스 공간을 한정한다. 표 1은 각 필드에 대한 엔코딩을 도시한다.

표 1

TT	TM	어드레스 공간
00	001	사용자 피연산자
00	010	사용자 지령
00	101	수퍼바이저 피연산자
00	110	수퍼바이저 지령
01	XXX	비 사용
10	101	에뮬레이터 모드 피연산자 (if MAP = 1)
10	110	에뮬레이터 모드 지령 (if MAP = 1)
11	000	CPU 공간
11	001	인터럽트 승인
11	01X	인터럽트 승인
11	1XX	인터럽트 승인

어드레스 공간은 간혹 동작의 특권(사용자 모드 또는 수퍼바이저 모드)레벨과 수행되는 레퍼런스(피연산자, 지령 페치)의 형태에 의해 한정됨을 주목해야 한다. 동시에, TT와 TM값은 중단점 위치를 보다 정확하게 한정하기 위해 사용될 수 있다.

AABR에서 다음값, 크기(SZ)값은 어드레스 범위에 덧붙여 K-버스를 통해 전송되는 크기 속성이 AABR의 SZ 비트에 저장된 값에 상응해야만 하는 것을 표시한다. 유사하게, 판독/기록(R) 값은 AABR내에 저장된다. R 값은 판독 또는 기록 동작에 기초한 트리거 조건이 어드레스에 기초한 비교 내에서 인가됨을 표시한다.

이전에 언급한 비트들은 다수의 하드웨어 중단점 레지스터(50)가 중단점 모니터로써 동작 중일 때 사용된다. 이들 비트는 K-버스(25) 속성 정보와 비교된다. 예컨대 사용자 모드, 워드-크기 피연산자 판독만이 트리거 되게 하는 것과 같이 다수의 중단점 레지스터(50)를 프로그램하는 것이 가능하다. 따라서 ABBR의 이들 비트는 어드레스 중단점 한정을 위한 부가적인 자격을 제공하기 위해 사용될 수 있다.

덧붙여, AABR은 중단점 한정으로부터 각 개별 비트를 마스크 -아웃하는 능력을 제공하는 속성 마스크를 포함한다.

AABR의 최상위 비트인 판독/기록 마스크(RM)비트는 어드레스에 기초한 비교에서 무시될 판독 또는 기록 동작에 기초한 트리거 조건을 표시한다. 따라서 RM 비트가 단정이면, 비교 동작 중 R 비트는 무시된다. 다음 집합의 비트인 크기 마스크(SM) 값은 어드레스에 기초한 비교에서 무시될 어드레스의 크기에 기초한 트리거 조건을 표시한다. 예컨대 SM 값의 한 비트가 단정이면, SZ 값의 상응하는 비트는 어드레스 비교 기능에서 무시된다.

전송 형태 마스크(TTM) 값은 TT값에 상응한다. TTM값에서 한 비트가 단정이면, TT 값의 상응하는 비트는 후속 하는 어드레스 비교에서 무시된다. 유사하게, 전송 변경자 마스크(TMM)값은 TM값에 상응한다. TMM값에서 한 비트가 단정이면, TM 값의 상응하는 비트는 후속 하는 어드레스 비교에서 무시된다.

다수의 하드웨어 중단점 레지스터(50)는 또한 다수의 프로그램 카운터 중단점 레지스터를 포함한다. 다수의 프로그램 카운터 중단점 레지스터는 중단점 기능을 트리거하기 위해 사용될 수 있는 데이터 처리 시스템(5)의 지령 논리 어드레스 공간 내의 영역을 한정한다. 프로그램 카운터 중단점(PBR) 값은 프로그램 카운터 중단점 마스크 (PBMR) 레지스터

내에 저장된 값에 의해 마스크될 수 있다. 본 발명의 한 실시예에 있어서, PBMR내에 저장된 상응하는 '0'값을 갖는 PBR내의 이들 비트만이 데이터 처리 시스템(5)의 프로그램 카운터 값을 갖는 비교 동작에 사용된다. 프로그램 카운터 값은 데이터 처리 시스템(5)의 CPU(2)의 프로그램 카운터 레지스터(도시 안됨)내에 저장됨을 주목해야 한다.

프로그램 카운터 중단점 레지스터에 덧붙여, 다수의 하드웨어 중단점 레지스터(50)는 다수의 데이터 중단점 레지스터를 포함한다. 다수의 데이터 중단점 레지스터는 데이터 중단점 레지스터(DBR)와 데이터 중단점 마스크 레지스터(DBMR)를 포함한다. DBR은 데이터 처리 시스템(5)이 디버그 동작을 수행할 때 트리거를 형성하기 위해 사용될 수 있는 특정 데이터 패턴을 한정하는 데이터 값을 저장한다. DBMR내에 저장된 데이터 마스크 값은 DBR내에 저장된 데이터 값을 마스크 한다. 이와 같이, 본 발명의 한 실시예에 있어서, DBMR내에 저장된 상응하는 '0'값을 갖는 DBR내에 저장된 이들 비트만이, 트리거 조건이 부합되는 지를 결정하기 위해, K-버스(25)를 통해 전송되는 데이터 신호와 비교될 수 있다.

동작 설명

후속의 논의의 목적을 위해, 디버그 지원의 일반 화제는 단일의 요구와 단일의 기능성을 갖는 세 개의 분리 영역으로 분할될 수 있다. 본 발명은 실시간 트레이스 지원, 백그라운드 디버그 동작 모드, 및 실시간 디버그 지원을 제공한다. 도 1의 데이터 처리 시스템(5)은 본 발명의 한 실시예를 도시한다.

데이터 처리 시스템(5)에 있어서, 데이터 프로세서(3)는 정보를 통신하기 위하여 외부 개발 시스템(7)에 접속된다. 외부 개발 시스템(7)은, 데이터 프로세서(3)내의 실패 회로와 에러 소프트웨어를 확인하고 분리시키기 위해 설계된 디버그 동작을 수행하기 위하여. 통신된 정보를 사용한다. 외부 개발 시스템(7)은 DSDI, DSCLK, DSDO, DDATA, PST, 및 다른 입력/출력 신호를 통해 데이터 프로세서(3)와 통신한다. DSDI, DSCLK, DSDO 신호는 외부 개발 시스템(7)과 데이터 프로세서(3) 사이에 직렬 통신 수단을 제공한다. DDATA 신호는 데이터 프로세서(3)로부터 디버그 정보를 외부 개발 시스템(7)에 제공한다. CSR의 구성에 따라서, 디버그 모듈(10)은 DDATA 신호 상에 디스플레이하기 위한 일정 피연산자와 브랜치 타겟 어드레스를 포착할 수 있다. 덧붙여, CPU(2)는 DSR내에 저장된 구성과 상관없이 DDATA상에 포착되고 디스플레이되는 지령을 실행할 수 있다. PST 신호는 내부 프로세서 상태를 외부 개발 시스템(7)에 제공한다. 따라서 프로그램 동작의 실행 중, DDATA와 PST 신호는 데이터 프로세서(3)에 의해 실행되는 현재 동작을 반영하기 위하여 집합적으로 억세스될 수 있다.

동작 중, 코어(9)는 CPU(2), MMU/제어기(4), 및 메모리(6)를 접속하기 위하여 K-버스(25)를 사용한다. 본 발명의 현재의 실시예에 있어서, K-버스(25)는 고속의 단일 주기 억세스 버스이다. 메모리(6)는 RAM, ROM, 캐쉬 블럭, 및 이들의 결합을 포함할 수 있다. 모든 다른 시스템 모듈과 주변기기는 M-버스(23)를 통해 코어(9)에 접속된다. M-버스(23)는 다중 버스 마스터중 하나에 의해 초기화된 데이터 전송을 수행하기 위해 사용되는 내부 멀티-마스터 버스이다. 시스템 버스 제어기(8)는 다중 기능을 제공한다. 시스템 버스 제어기(8)는, 외부 E-버스가 존재하면, 내부 M-버스(23)와 외부 E-버스 사이의 인터페이스를 제공한다. 부가적으로, 시스템 버스 제어기(8)는 S-버스 상의 모든 데이터 전송을 제어하기 위한 초점으로써 작용한다. S-버스는 타이머와 직렬 통신 채널과 같은 단순한 슬레이브 주변 모듈(12 및 13)을 데이터 처리 시스템(5)에 접속시키기 위해 사용된다.

본 발명에 있어서, 데이터 프로세서(3)는 데이터 전송을 위해 원가 효율적인 메카니즘을 제공하기 위해 버스 대역폭의 수 개의 층을 갖는다. 코어(9)는 수행력을 최대로 하기 위해 고속의 단일 주기의 K-버스(25)와 상호 접속된다. 전송은 이 고속의 버스에 직접 접속되지 않기 때문에, M-버스(23)는 코어(9)와 마스터 모듈(14)과 같은 임의의 내부 버스 마스터로부터 대역폭을 제공한다. 시스템 버스 제어기(8)는, 낮은 원가 상의 모든 데이터 전송을 위한 제어 기능과,

슬레이브 모듈(12및 13)과 같은 슬레이브 주변 모듈을 위한 낮은 대역폭의 S-버스를 제공하면서, 내부 M-버스(13)와 외부 E-버스 (존재하면) 사이의 접속을 제공한다. 디버그 모듈(10)은 프로세서에 의해 초기화된 모든 메모리 억세스의 비강 제적인 관측을 허용하기 위하여 K-버스(25)에 접속된다. 디버그 모듈(10)은 또한 외부 개발 시스템(7)에의 접속을 제공한다.

도 1의 디버그 모듈(10)은 이전에 설명한 세 개의 디버그 동작 각각을 지원하기 위한 논리 회로를 제공한다. 디버그 모듈(10)의 동작은 후속하여 보다 상세하게 설명된다.

실시간 트래이스 기능

본 발명은 지금까지 존재하지 않는 그리고 중요한 실시간 트래이스 기능을 제공한다. 실시간 트래이스 기능은 외부 개발 시스템(7)과 같은 외부 사용자가 동적으로 데이터 프로세서(3)의 내부 동작을 관측할 수 있도록 허용한다. 더욱이, 실시간 트래이스 기능은 이와 같은 관측을 데이터 처리 시스템(5)의 효율과 속도에 중요한 충돌 없이 시스템의 내부 동작에 제공한다. 데이터 처리 시스템(5)의 디버그 모듈(10)은 내부 동작 정보를 제공하기 위해 DDATA 및 PST신호를 통해 병렬 출력 포트를 외부 사용자에게 제공한다. DDATA신호는 피연산자 데이터를 반영하는 데이터를 제공하고 PST신호는 CPU(2)의 실행 상태를 반영하는 엔코드된 상태 정보를 제공한다. 더욱이 DDATA 신호는 또한, 외부 개발 시스템이 외부의 가시적인 어드레스 버스 또는 외부의 가시적인 데이터 버스를 요구함이 없이 프로그램의 흐름을 정확히 트래이스 하는 것을 허용하기 위하여, 포착된 지령 어드레스 프로그램 흐름 변화를 제공한다. DDATA 신호 상에 디스플레이되는 정보는 PST신호상의 상태 정보와 동기가 맞추어짐을 주목해야 한다. 따라서 외부 사용자는 데이터 프로세서(3)의 동작 지연 없이, 또한 실행 경로를 가시화 시키기 위해 특정 디버그 동작 모드의 요구 없이, 프로그램의 동적인 실행 경로를 결정하기 위해 DDATA와 PST신호를 디코드할 수 있다.

실시간 트래이스의 실행 중 데이터 처리 시스템(5)의 동작을 후속하여 보다 상세하게 설명된다. 그 논의에 앞서, PST신호의 엔코딩이 간략히 설명된다. 표는 도 10에 제공된 PST신호의 엔코딩을 도시한다.

도 10은 PST 신호의 비트(3 내지 0)를 통해 제공되는 값과 CPU(2)의 상태 사이의 관계를 도시한다. PST신호가 %0000의 값을 제공할 때, 이전의 동작은 아직 종료되지 않았고, 실행은 단순히 지속된다. %0001의 값이 제공되면, PST신호는 지령의 실행이 시작됨을 표시한다. PST 신호가 %0011의 값을 가질 때, 데이터 처리 시스템은 동작의 사용자 모드로 진입한다. %0100의 값이 PST신호에 의해 제공되면, 데이터 처리 시스템(5)은 펄스 또는 WDDATA 지령의 실행을 시작한다. 펄스 옵코드(opcode)는 단순히 이 특정 PST 엔코딩을 생성하고, WDDATA 옵코드는 피연산자가 CSR에 의해 한정된 구성에 상관없이 DDATA상에 포착되고 표시되도록 강요한다. 이것은 메모리 피연산자, 예외 스택 프레임 또는 소프트웨어 마커를 직접 DDATA 신호 상에 디스플레이하는 단순한 메카니즘을 제공한다.

%0101의 값이 PST 신호에 의해 제공되면, 데이터 처리 시스템(5)은 취해진 브랜치 동작의 실행을 시작한다. %0111의 값이 PST 신호에 의해 제공되면, 데이터 처리 시스템(5)은 RTE 지령의 실행을 시작한다. PST 신호가 %1000의 값을 제공할 때, 데이터 처리 시스템(5)은 DDATA 신호 상의 한 바이트 전송의 실행을 시작한다. PST신호가 %1001의 값을 제공할 때, 데이터 처리 시스템(5)은 DDATA 신호 상의 두개 바이트 전송의 실행을 시작한다. PST 신호가 %1010의 값을 제공할 때, 데이터 처리 시스템(5)은 DDATA 신호 상의 세 개 바이트 전송의 실행을 시작한다. PST 신호가 %1011의 값을 제공할 때, 데이터 처리 시스템(5)은 DDATA 신호 상의 네 개 바이트 전송의 실행을 시작한다. PST 신호가 %1100의 논리값을 갖으면, 데이터 처리 시스템(5)의 CPU(2)는 예외 처리 루틴을 실행한다. 유사하게, PST신호가 %1101의 논리값을 갖으면, 데이터 처리 시스템(5)의 CPU(2)는 예외를 처리한다. 이 값은 예외가 검출된 때로부터 예외 처리 루틴의 제 1의 지령이 페치될 때까지 PST 신호 상에서 단정이다. 유사하게, PST 신호가 %1101의 논리값을 갖으면, 데이터 처리 시스템(5)의 CPU(2)는 에뮬레이터 동작 모드로의 진입을 야기하는 예외 루틴을 실행한다. PST 신호가 %1110의 이면, 데이터 처리 시스템(5)의 CPU(2)는 정지되고 인터럽트를 기다린다. PST 신호

가 %1111일 때, 데이터 처리 시스템(5)의 CPU(2)는 정지한다.

이전에 지적한 바와 같이, 엔코드된 PST 및 DDATA 신호에 의해 집합적으로 제공되는 프로세서 상태 정보는 데이터 처리 시스템(5)의 동적 예외 경로를 완전히 트랙하기 위하여 외부 프로그램 모델로 사용될 수 있다.

실시간 트래이스 기능을 가능케 하기 위하여, DDATA 신호는 브랜치 타겟 지령 어드레스를 표시하도록 구성되어야만 한다. 전형적으로, 데이터 처리 시스템(5)이 이의 정상 실행을 시작하기 전에, 이 구성은 수행된다. DDATA 신호를 구성하기 위하여, 외부 개발 시스템(7)은 다수의 제어 레지스터(40)내의 CSR에 값을 기록할 수 있다. CSR 레지스터는 도 8에 도시되었다. 이 기록 기능을 수행하기 위해, 외부 개발 시스템(7)은 데이터를 CSR에 로드 하는 명령에서 DSDI 및 DSCLK 신호를 사용하여 직렬 인터페이스(30)로 이동한다. 본 발명의 한 실시예에 있어서, 명령의 제 1부분이 DSDI신호에 의해 제공되고, DSCLK 신호에 의해 클럭 된다. 명령의 추가분은 후속하여 제공된다. 명령의 모든 부분은 16 비트의 정보 패킷이 완성될 때까지 레지스터(32)에 저장된다. 본 발명의 선택적인 실시예는 본 발명의 현재의 실시예에서 요구되는 16 비트보다 더 많거나 또는 더 적은 비트를 포함할 수 있다.

여기에서 설명된 본 발명의 실시예에 있어서, CSR에 값을 기록하는 지령은 16 비트의 옵코드와 32 비트의 데이터를 갖는 48 비트 길이이다. 레지스터(32)는 16 비트 폭이다. 따라서 제 1의 16비트의 정보 패킷이 종료될 때, 이 정보가 디코드 되고 후속하여 동작이 구성되는 제어회로(20)에 이 정보가 전송된다. 제 2의 16비트의 정보 패킷이 레지스터(32)에 로드 되었을 때, 이 정보는 직렬 정보 신호를 통해 다수의 제어 레지스터에 전송된다. 제어 회로(20)로부터 제어 버스에 의해 제공된 제어 정보를 사용하면서, 직렬 정보 신호의 내용은 지정 레지스터, CSR의 상부에 로드 된다. 유사한 방법으로 제 3의 16비트의 정보 패킷이 레지스터(32)에 로드 되고 이후 다수의 제어 레지스터(40)에 전송된다. 또, 제어회로(20)의 소스인 신호를 사용하면서, 직렬 정보 신호의 내용은 CSR의 하부에 로드 된다. 이 방법에 있어서, 전체 48 비트 지령은 직렬로 디버그 모듈(10)로 이동되고, 여기에서 이는 디코드 되고 피연산자 값은 CSR에 로드 된다.

후속 하는 데이터는 DSDI 신호를 통해 외부 개발 시스템(7)에 의해 직렬로 제공된다. 이전에 설명한 바와 같이, DSDI 신호에 의해 제공된 후속 하는 데이터는 DSCLK 신호에 의해 동기가 맞추어진다. 각 데이터 비트는 16비트의 정보 패킷이 종료될 때까지 레지스터(32)에 저장된다. 제 2의 16비트의 정보 패킷이 종료될 때, 정보는 직렬 정보 신호를 통해 다수의 제어 레지스터(40)에 전송된다. 제 2의 16비트의 정보 패킷은 비트(31 내지 16)로써 CSR에 저장된다. 후속하여 부가적인 정보는 외부 개발 시스템(7)에 의해 DSDI 및 DSCLK 신호를 통해 레지스터(32)에 직렬로 제공된다. 제 3의 16비트의 정보 패킷이 종료될 때, 이 정보는 직렬 정보 신호를 통해 다수의 제어 레지스터(40)에 전송된다. 제 3의 16비트의 정보 패킷은 비트(15 내지 0)로써 CSR에 저장된다.

이전 절에서, DDATA 신호의 구성은 외부 개발 시스템(7)에 의해 CSR에 기록된 값에 의해 결정되었다. 본 발명의 다른 실시예에 있어서, DDATA 신호는, 데이터 프로세서(4)의 CPU(2)에 의해 CSR에 기록된 값에 의해, 구성될 수 있다. CPU(2)가 그 값을 CSR에 기록할 때, 그 값은 도 6에 도시된 중단점 회로(100)의 임시 레지스터(170)에 제공된다. 그 값은 K-버스(25)의 KDATA 신호를 통해 임시 레지스터(170)에 제공된다. 임시 레지스터(170)로 부터, 그 값은 레지스터(32)에 전송되고 16 비트 패킷으로 저장된다. CPU가 CSR에 기록하기 때문에, DKATA 신호를 통해 전송되는 피연산자 값은 CPU(2)로부터 K-버스(25)를 통해 제어회로(20)에 제공되는 제어 신호를 사용하여 임시 데이터 레지스터(170)에 포착된다. 따라서, 그 값은 다수의 제어 레지스터 (40)에, 외부 개발 시스템(7)으로부터 레지스터(32)에 제공되는 것과 동일하게 나타난다.

피연산자가 외부 개발 시스템(7) 또는 CPU(2)에 의해 제공된다 할지라도, 그 피연산자와 관련된 피연산자는 실시간 트래이스 기능을 수행하기 위한 DDATA 신호를 구성하기 위하여 CSR에 저장되어야만 한다. CSR에 있어서, 디버그 데이터 제어(DDC)와 브랜치 타겟 바이트(BTB) 값은, 트래이스 정보를 외부 개발 시스템(7)에 제공하는 DDATA 신호를 정확히 구성하기 위해, 기록되어야만 한다. 이전에 언급한 바와 같이, 디버그 데이터 제어(DDC)값은 DDATA신호를 통해 외부 개발 시스템에 제공될 피연산자 데이터를 포착하기 위한 구성 제어 정보를 제공한다. DDC 값이 %00 일 때, 피연산자 데이터는 DDATA 신호 상에 디스플레이되지 않는다. DDC 값이 %01 일 때, M-버스(23)를 통해 통신되는 모든 기록 데이터는 포착되고 DDATA 신호 상에 디스플레이된다. DS 값이 %10 일 때, M-버스(23)를 통해 통신되는 모든 판독 데이터는 포착되고 DDATA 신호 상에 디스플레이하다. DDC 값이 %11 일때, M-버스(23)를 통해 통신되는 모든 판독 및 기록 데이터는 포착되고 DDATA 신호 상에 디스플레이된다. DDC 값에 의해 규정되는 정보의 형태에 관계없이, DATA 신호는 피연산자 레퍼런스 크기에 의해 한정되는 바이트의 수를 디스플레이한다. 예컨대 바이트 디스플레이는 8 비트 길이이고, 워드 디스플레이는 16 비트 길이이고, 또한 긴 디스플레이는 32 비트 길이이다. 피연산자가 CSR의 DDC 필드의 제어하에서 포착될 때, DDATA 신호 상에 디스플레이되는 바이트 수는 실행되는 레퍼런스의 형태에 의해 결정된다. 피연산자가 바이트로써 레퍼런스 되면, 8 비트의 데이터가 디스플레이된다. 피연산자가 워드로써 레퍼런스 되면, 16 비트의 데이터가 디스플레이된다. 유사하게, 피연산자가 긴 워드로써 레퍼런스 되면, 32 비트의 데이터가 디스플레이된다. DDC는 M-버스(23) 판독 및 기록 동작을 포착할 능력을 제공한다. 그러나, 피연산자 레퍼런스 크기는 DDATA 신호 상에 실제로 디스플레이되는 바이트 수를 결정한다.

브랜치 타겟 바이트(BTB)값은 DDATA 신호 상에 디스플레이될 브랜치 타겟 어드레스의 바이트 수를 한정한다. BTB 값이 %00 일 때, 브랜치 타겟 어드레스의 바이트는 DDATA 신호 상에 디스플레이되지 않는다. BTB 값이 %01 일 때, 브랜치 타겟 어드레스의 하위 두 바이트가 DDATA 신호 상에 디스플레이된다. BTB 값이 %10 일 때, 브랜치 타겟 어드레스의 하위 세 바이트가 DDATA 신호 상에 디스플레이된다. BTB 값이 %11 일 때, 브랜치 타겟 어드레스의 전체 네 바이트가 DDATA 신호 상에 디스플레이된다. DDATA 신호는 일반적으로 최하위에서 최상위의 순서로 데이터를 출력함을 주목해야 한다. 더욱이, 동작 중에, 제어 회로(60)의 제어 하에서 FIFO 저장 버퍼(70)는 변화 어드레싱 모드를 사용하는 취해진 브랜치 동작과 관련된 타겟 어드레스만을 포착한다. 변화 어드레스 모드는, 어드레스가 프로그램 카운터 관련물 또는 절대 어드레스에 의해 제공되지 않고 다른 수단에 의해 계산되는 모드이다. 이와 같은 타겟 어드레스는 전형적으로 예외 벡터, 예외로부터 복귀(TRE), 서브 루틴으로부터 복귀(RTS) 지령, 및 점프(JMP)와 서브루틴으로 점프(JSR)의 형태에 관련된다.

실시간 트래이스 기능을 수행하기 위해 요구되는 정보를 제공하기 위한 DDATA 신호의 구성을 위하여 CSR이 기록된 후, CPU(2)는 다수의 타겟 어드레스를 K-버스(25)에 의해 통신되는 DADDR 신호를 통해 FIFO(First-in First-out) 저장 버퍼(70)에 제공한다. 덧붙여, CPU(2)는 다수의 피연산자 데이터 값을 K-버스(25)에 의해 통신되는 KDATA 신호를 통해 FIFO 저장 버퍼(70)에 제공한다. 다수의 타겟 어드레스와 다수의 피연산자 데이터 값은 제어 회로(60)에 의해 제공되는 FIFO 제어 신호에 따라서 FIFO저장 버퍼(70)에 저장된다.

제어 회로(60)는 제어회로(20)에 의해 제공되는 트래이스 신호와 다수의 제어 레지스터(40)의 CSR에 의해 제공되는 구성 신호에 따라서 FIFO 제어 신호를 출력한다. FIFO 저장 버퍼(70)는 거기에 저장된 KADDR 및 KDATA 정보를 먼저 입력된 것을 먼저 출력하는 순서로 출력한다. FIFO 저장 버퍼(70)는 어드레스와 피연산자 정보를 저장하기 위한 다수의 위치를 갖기 때문에, FIFO 저장 버퍼(70)는 데이터 프로세서(3)의 정상 동작 중에 억세스 되는 피연산자 및 지령 어드레스의 값을 반영하는 각 통신 정보로써 KADDR 및 KDATA 신호를 수신할 수 있다. 즉, FIFO 저장 버퍼(70)에 제공되고 이에 저장되는 KADDR 및 KDATA 신호를 통해 전송되는 값은 실시간에서 데이터 처리 시스템(5)의 현재의 동작을 반영한다.

FIFO 저장 버퍼(70)내에 모든 저장 위치가 DDATA 신호 상에 출력되어야만 하는 타당한 정보를 포함할 때, FIFO 저장 버퍼(70)는 CPU(2)의 동작 속도에만 영향을 미친다. 모든 저장 위치가 채워져 있을(풀(full)일) 때, FIFO 저장 버퍼(70)는 풀(Full) 신호를 단정으로 한다. 풀 신호가 단정일 때, 제어 회로(60)는 스톨(Stall)신호를 단정으로 한다. FIFO 저장 버퍼(70)내의 최소한 한 저장 위치가 다음값을 저장하기 위해 유용하게 될 때까지 코어(9)의 동작을 스톨시키기 위해 스톨 신호는 CPU(2)에 제공된다. 모든 다른 동작에 있어서, DDATA 신호 상의 어드레스와 피연산자 정보의 예비는 데이터 프로세서(3)의 실행 속도에 중요한 영향을 미치지는 않는다.

FIFO 저장 버퍼(70)는 FIFO 신호를 통해 어드레스와 피연산자 정보를 다수의 멀티플렉서(80)에 제공한다. CPU(2) 또한 CPU 프로세서 상태(CPST) 신호를 다수의 멀티플렉서(80)에 제공한다. CPST 신호는 데이터 프로세서(3)에 의해 현재 실행되고 있는 동작의 형태를 지시하기 위한 프로세서 상태 정보를 제공한다. 예컨대 CPST 신호는, 지령의 실행이 시작될 때, 지령의 실행이 연속되어야 할 때, 데이터 프로세서(3)가 동작의 선택 모드로 진입할 때, 사전에 선택된 브랜치 지령이 실행될 때, 및 데이터 프로세서(3)의 동작이 정지할 때를 지시할 수 있다. CPST 신호는 프로세서 동작의 어느 형태라도 지시할 수 있고, 여기에 나열된 예에 국한되지 않음을 주목해야 한다.

FIFO 및 CPST 신호가 다수의 멀티플렉서(80)에 제공될 때, 다수의 멀티플렉서의 출력은 Mux 제어 신호에 의해 제어된다. Mux 제어 신호는 다수의 멀티플렉서(80)가 동기에 맞춰 DDATA 와 PST 신호를 제공할 수 있도록 한다. DDATA 와 PST 신호는 데이터 프로세서(3)(여기에는 도시 안됨)의 클럭의 상승 에지와 동기가 맞추어지고, 따라서 PST 신호를 통해 제공되는 상태는 주어진 어느 버스 주기의 처리와 관련되지 않을 수 있음을 주목해야 한다.

더욱이, Mux 제어 시호에 의해 가능하게 되었을 때, PST신호는 CPU(2)의 상태를 지시하는 마커를 제공한다. 예컨대, PST 신호에 의해 $C, $D, $E 또는 $F의 상태 값이 제공되었을 때, PST신호는 멀티 주기 모드 또는 특정 동작이 데이터 프로세서(3)에 의해 실행되고 있음을 나타낸다. 엔코드된 PST신호의 추가의 정보를 위해선 도 10을 참조하라. PST신호가 특정 동작의 멀티-주기 모드가 데이터 프로세서(3)에 의해 실행되고 있음을 반영할 때, PST신호는 이 모드가 빠져나가거나 또는 동작이 종료될 때까지 이들 상태를 유지한다.

PST신호가 상태값($0 내지 $7)을 출력할 때, PST 신호는 각 타이밍 주기상에 갱신되는 정보를 규정한다.

PST신호가 상태값($8, $9, $A 또는 $B)을 출력할 때, PST 신호는 DDATA신호를 통해 외부적으로 제공된 값을 한정하기 위해 사용된다. 이 동작에 있어서, PST 신호는 DDATA 신호 상의 어느 값이 요구되는 어드레스와 데이터 정보를 제공하는지를 지시하는 마커로써 사용된다. PST신호는 DDATA 신호 상에 실제 데이터가 디스플레이되기 한 타이밍 주기 전에 이들 상태값으로 엔코드된다. 따라서 외부 개발 시스템은 데이터 프로세서(3)의 동작을 정확하게 관측한다.

다음의 예는 실시간 트래이스 동작에서 DDATA 와 PST신호의 사용을 보다 상세하게 도시한다. 이 예는 도 11에 도시되었다. 도 11은 점프 지령 타겟 어드레스의 하위 16 비트가 DDATA신호 상에 디스플레이되는 비직접 점프(JMP) 지령의 실행을 도시한다. 점프 지령이 실행될 때, 지령의 피연산자에 의해 지정되는 효과적인 어드레스에서 프로그램 실행은 지속된다. 현재의 실시예에 있어서, 효과적인 어드레스 값은 레지스터(A0)에 의해 지정된 메모리 위치에 위치한다.

다음의 실시예에 있어서, 적절한 포맷으로 데이터를 제공하기 위한 DDATA 신호를 정확히 구성하기 위해 CSR은 기록되었다고 가정한다. 도 11에 도시된 예에 있어서, JMP(A0)지령은 도 11에서 'LAST' 로 기록된 이전의 지령을 뒤따른다. JMP 지령은 타겟 어드레스를 비직접 적으로 지정함을 주목해야 한다. 예컨대 CPU(2)에 저장된 레지스터(A0)의 내용은 바람직한 타겟 지령 어드레스를 한정한다. 더욱이, 본 발명의 명세서에서 자세하게 도시되지는 않았지만, A0 레지스터는 CPU(2)내에 위치할 수 있다. 타겟 지령 어드레스는 A0 레지스터의 내용에 의존하고, 어드레스가 PC 관련물 또는 절대 어드레스에는 의존하지 않기 때문에, 이와 같은 비직접 어드레스의 사용을 변화 어드레스 모드로 참조된다.

다음 예의 실행 중, DSOC 및 AGEX는 데이터 프로세서(3)의 내부 파이프라인에서의 위치를 표시한다. DSOC, AGEX, IAG 및 IC를 설명하기 위해, CPU(2)의 내부 구조가 간단히 설명되어야만 한다.

데이터 프로세서(3)의 CPU(2)내에서, 두개의 독립적이고 접속되지 않은 파이프라인이 실현된다. 제 1의 파이프라인은 지령을 프리페칭하기 위한 지령 페치 파이프라인(IFP)(도시 안됨)이고, 제 2의 파이프라인은 지령의 실제 실행을 디코딩하고 수행하기 위한 피연산자 실행 파이프라인(OEP)이다. FIFO 지령 버퍼(도시 안됨)는 두개의 파이프라인 사이에 단락 시키는 메카니즘으로써 기여한다.

지령 페치 파이프라인은 두개의 단계를 포함한다. 제 1의 단계는 다음의 프리페치 어드레스가 계산되는 지령 어드레스 생성(IAG) 단계이다. 제 2의 단계는 바람직한 지령이 메모리로부터 페치 되는 지령 페치 주기이다. 피연산자 실행 파이프라인 역시 두 개의 단계를 포함하고, 여기에서 각 단계는 단일 지령의 실행 중 두 기능을 위해 사용될 수 있다. 제 1의 단계는 피연산자 페치 주기(OC)를 갖는 디코드 및 선택(DS) 상태이다. 제 2의 단계는 EXecute(EX) 단계를 갖는 어드레스 생성(AG) 단계이다.

두 레지스터 사이에서 정보가 통신되는 지령에 대하여, 두 개의 OEP 단계는 각각 한번씩 억세스 된다. 레지스터 피연산자는 OC 단계에서 억세스 되고, 그 후 지령 실행은 EX 단계에서 실제적으로 실행된다. 따라서 OEP 동작은 OC로부터 EX로 진행한다.

메모리 위치와 레지스터 사이에서 정보가 통신되는 지령에 대해서, 각 파이프라인 단계는 두 번 억세스 된다. 제 1의 것은, 피연산자 어드레스에 대한 요소가 DS 단계에서 선택된다. 제 2의 것은 이들 각 요소가 AG 단계에서 피연산자 어드레스를 형성하기 위해 합해진다. 제 3의 것은, 메모리 피연산자가 OC단계에서 다른 임의의 레지스터 피연산자에 따라 억세스 된다. 제 4의 것은, 실제 지령 실행이 EX 단계에서 수행된다. 지령의 이와 같은 형태에 대해, OEP 동작은 DS에서 AG, OC, 및 EX의 순서로 이루어진다.

지령의 다른 형태에 대해, 각 파이프라인 상태 내의 두 동작은 동시에 수행된다. 지령의 이러한 형태에 대해서, OEP동작은 DS 및 OC 에서 AG및 EX로 이루어진다.

이 예에 있어서, 레지스터(A0)의 내용은 '타겟'의 지령 어드레스를 한정한다고 가정한다. JMP지령이 피연산자 실행 파이프라인의 AGEX 단계를 점유함으로써, 지령 타겟 어드레스는 '타겟'으로 계산된다. 이 흐름-변화 동작은 지령 페치 파이프라인이 프리페치 어드레스의 현재의 줄기를 버리고 '타겟' 어드레스에서 시작하는 새로운 줄기를 한정하도록 한다. 본 발명의 이 실시예에 있어서, 이 흐름-변화 JMP 지령은 두 개의 파이프라인이, 새로운 지령 줄기가 성립되도록 야기하기 위해, AGEX 및 IAG 단계에서 접속되도록 한다. 흐름-변화가 한번 발생하면, 두 개의 파이프라인은, 새로운 어드레스 줄기에서 순차적으로 프리페치를 계속하면서, 지령 페치 파이프라인을 갖는 동작의 정상 단락 모드로 복귀한다. 따라서 타겟, 타겟+4, 및 타겟+8 의 어드레스는 후속하여 페치 된다.

JMP 지령이 그 피연산자 실행 파이프라인 단계를 종료하면, CPU(2)는 내부 CPST 신호 상에 '취해진 브랜치'값을 내보낸다. 따라서 내부 CPST 신호는 $5의 값을 갖는다. 본 발명의 이 실시예에 있어서, 실제 PST 및 DDATA 신호는 내부 상대로부터 한 타이밍 주기 지연된다. CPU(2)가 새로운 어드레스에서 지령을 프리페치함으로써, FIFO 저장 버퍼(70)는 지령 페치 파이프라인의 IC 주기의 끝에서 지령 어드레스 타겟을 포착한다. 다음 타이밍 주기 상에서, 다수의 멀티플렉서(80)는 DDATA 신호 상에 디스플레이될 두 바이트의 정보를 나타내는 PST 신호에 '마커(Maker)'를 삽입한다. 따라서 PST 신호는 $9의 값을 갖는다. 다수의 멀티플렉서(80)의 동작은 제어 회로(60)에 의해 제어된다. PST 신

호 상의 마커의 출현에 뒤따라, 다수의 멀티플렉서(80)는 최하위에서 비트(3 내지 0)에서 비트(15 내지 12)까지 조금씩 시작하는 포착한 지령 어드레스 타겟의 하위 16비트의 디스플레이를 진행한다.

내부 PST 값(도 2의 CPST신호)은 제 1의 타이밍 주기에서 $5 값을 전송한다. 도 10에 도시된 바와 같이, PST 값이 $5 값을 가질 때, 데이터 프로세서는 취해진 브랜치 지령의 실행을 시작한다. 이것은 레지스터(A0)에 의해 지정된 어드레스에 프로그램 흐름을 직접 전송하는 JMP 지령에 상응한다. 도 11에 도시된 제 1의 타이밍 주기 중, 내부 DDATA 신호(도 2의 FIFO 신호)는 $0 값을 전송한다. 도 9에 도시된 바와 같이, DDATA 값이 $0 값을 저장할 때, 데이터 프로세서(3)내에서 중단점은 가능하지 않게 된다.

제 2의 타이밍 주기 중, JMP 지령에 대한 타겟 어드레스는 내부 DDATA 값을 통해 제공되는 정보의 다음 두개의 바이트에서 지정되는 것을 지시하기 위하여, 내부 PST 값은 $9 값을 전송한다. 이 시점에, 내부 DDATA 신호는 $0 값의 전송을 계속한다. 이전에 언급한 바와 같이, 내부 PST 값이 $8, $9, $A, 또는 $B의 상태값을 전송할 때, 내부 PST신호는 DDATA 신호를 통해 외부에서 제공되는 값을 한정하기 위하여 사용된다. 이 동작에 있어서, 내부 PST신호는, 내부 DDATA 신호상의 어느 값이 요구되는 타겟 어드레스 정보를 제공하는 지를 표시하기 위해, '마커'로써 사용된다. 내부 PST신호는 실제 데이터가 내부 DDATA 신호 상에 디스플레이되기 한 타이밍 주기 전에 이들 상태값으로 엔코드된다.

제 3의 타이밍 주기 중, 내부 PST 값은 지령 실행이 계속되는 것을 표시하기 위해 $0 값을 전송한다. 부가적으로, 제 3의 타이밍 주기 중, 내부 DDATA 신호는 타겟 어드레스 값의 비트(3 내지 0)를 제공한다.

제 4의 타이밍 주기 중, CPST 값은 타겟 어드레스에 위치한 제 1의 지령에 의해 한정된 값을 표시한다. 이 때에, 내부 DDATA 신호는 타겟 어드레스 값의 비트(7 내지 4)를 전송한다. 후속 적으로, 제 5 및 제 6의 타이밍 주기에 있어서, 내부 DDATA 신호는 각각 타겟 어드레스 값의 비트(11 내지 8)와 비트(15 내지 12)를 제공한다. 이 방법에 있어서, 외부 개발 시스템(7)은 변화 어드레스를 갖는 브랜치 동작의 현재의 기록과 실시간에서 비직접 적으로 결정된 어드레스 위치 둘 다를 관측할 수 있다.

다수의 멀티플렉서(80)는 내부 PST 및 내부 DDATA 신호를 각각 PST 및 DDATA 신호로써 외부 개발 시스템(7)에 제공한다. PST 및 DDATA 신호는 집적된 회로 핀(자세히 도시 안됨)을 통해 제공된다. 덧붙여, 집적된 회로 핀을 통해 제공된 PST 및 DDATA 신호는, 내부 PST 및 내부 DDATA 신호를 통해 정보가 전송된 한 타이밍 주기후, 각각 정보를 통신함을 주목해야 한다.

이와 같은 실시간 관측을 허용함으로써, 외부 개발 시스템(7)은 실패를 보다 빠르고 보다 정확하게 확인하고 분리시킬 수 있다. 더욱이, 실시간 트래이스 관측이 외부 개발 시스템(7)에 제고되는 동안, 데이터 프로세서(3)의 CPU(2)는 지령의 실행을 계속하고, 통상적으로 DDATA 및 PST 신호의 예비에 의해 영향을 받지 않는다.

더욱이, 상기의 실시예가 JMP 지령을 실행하기 위한 트래이스 기능의 사용을 설명하는 동안, 본 발명은 데이터 프로세서(3)의 동적 실행 경로를 완벽하게 추적하기 위해 사용될 수 있다. JMP 지령과 함께 도시된 바와 같이, 동적 실행 경로의 추적은 동작 흐름의 어떠한 변화로도 복잡해진다. 예컨대, 일부 브랜치 지령은, 타겟 지령 어드레스의 계산이 PC(프로그램 카운터)관련물 또는 절대 어드레스에 의한 것이 아니고 프로그램 가시화 레지스터의 사용을 결부시키는, 변화 어드레스를 사용하여 실현될 수 있다.

변화 어드레스를 사용하는 지령을 위하여, 본 발명은 규정된 씨퀀스에서 DDATA 및 PST 신호를 제공하여 타겟 어드레스가 정확하게 결정되고 외부 개발 시스템(7)에 제공되도록 한다. 우선, 취해진 브랜치 지령의 실행이 다수의 집적된 회로 핀을 통해 전송되는 PST 신호에 의해 확인된다. 도 10의 엔코딩에서 도시된 바와 같이, PST 신호는 취해진 브랜치 지령이 PST 신호가 $5의 값을 가질 때 실행됨을 나타낸다. 다음엔, 타겟 어드레스가 DDATA 신호 상에 디스플레이되어야 함을 나타내기 위하여, PST 신호가 사용될 수 있다. PST 신호의 엔코딩은 DDATA 신호를 통해 디스플레이될 바이트의 수를 확인한다. 그 후, 타겟 어드레스는 DDATA 신호를 사용하는 후속의 타이밍 주기 중에 선택적으로 유용하게 된다. DDATA 신호 상에 디스플레이되는 타겟 어드레스의 비트 수는 외부 사용자에 의해 제어되는 구성 가능한 파라메터이다.

본 발명은 동적으로 실행되고 데이터 프로세서(3)의 CPU(2)의 수행에 중요한 충돌을 일으키지 않는 단일의 트래이스 기능을 제공한다.

본 발명에 의해 실현되는 실시간 트래이스 기능에 있어서, PST 신호에 의해 한정되는 실행된 지령 줄기와 DDATA 신호에 의해 제공되는 포착된 데이터의 유용성 사이에의 엄격한 동기화에 주목해야 한다. 여기에서 설명된 본 발명의 실시예에 있어서, DDATA 신호는 적절한 지령과 쉽게 관련될 수 있는 것과 같이 엄격한 순서가 존재한다. 이 순서는 다음의 예에서 상당히 상세하게 설명된다. 다음의 코드는 다음의 예에서 실행된다고 가정한다.

```
지령 어드레스      지    령

00001316           movq  #1,d0

00001318           mov.l d0,(-4,a6)

0000131c           pea   (-68,a6)

00001320           pea   (-36,a6)

00001324           bsr.w Func2




0000115c           mov.l d7,-(a7)
```

상기 예에 있어서, CSR이 브랜치 타겟 어드레스 정보의 두 개 바이트를 디스플레이하도록 구성되면, 모든 피연산자 M-버스(23) 억세스를 생성하면, 또한 판독 및 기록 피연산자가 포착되면, PST 신호와 DDATA 신호 상에 결과의 출력은 다음과 같이 나타난다.

PST	DDATA	설 명
$1	----	1316에서 지령 시작
$1	----	1318에서 지령 시작
$b	----	지령에 필요한 피연산자 기록
at		1318
0	[3:0]	1318에서의 지령을 위한 기록 피연산자로 부터의 비트
0	[7:4]	1318에서의 지령을 위한 기록 피연산자로 부터의 비트
0	[11:4]	1318에서의 지령을 위한 기록 피연산자로 부터의 비트
0	[15:12]	1318에서의 지령을 위한 기록 피연산자로 부터의 비트
0	[19:16]	1318에서의 지령을 위한 기록 피연산자로

부터의 비트

0 [23:20] 1318에서의 지령을 위한 기록 피연산자로 부터의 비트

0 [27:24] 1318에서의 지령을 위한 기록 피연산자로 부터의 비트

0 [31.28] 1318에서의 지령을 위한 기록 피연산자로 부터의 비트

$1 ---- 131c에서 지령 시작

$b ---- 지령에 필요한 피연산자 기록

at 131c

0 [3:0] 131c에서의 지령을 위한 기록 피연산자로 부터의 비트

0 [7:4] 131c에서의 지령을 위한 기록 피연산자로 부터의 비트

0 [11.4] 131c에서의 지령을 위한 기록 피연산자로 부터의 비트

0 [15:12] 131c에서의 지령을 위한 기록 피연산자로 부터의 비트

0 [19:16] 131c에서의 지령을 위한 기록 피연산자로 부터의 비트

0	[23:20]	131c에서의 지령을 위한 기록 피연산자로 부터의 비트
0	[27:24]	131c에서의 지령을 위한 기록 피연산자로 부터의 비트
0	[31:28]	131c에서의 지령을 위한 기록 피연산자로 부터의 비트
$1	----	1320에서 지령 시작
$b	----	지령에 필요한 피연산자 기록
at		1320
0	[3:0]	1320에서의 지령을 위한 기록 피연산자로 부터의 비트
0	[7:4]	1320에서의 지령을 위한 기록 피연산자로 부터의 비트
0	[11:4]	1320에서의 지령을 위한 기록 피연산자로 부터의 비트
0	[15:12]	1320에서의 지령을 위한 기록 피연산자로 부터의 비트
0	[19:16]	1320에서의 지령을 위한 기록 피연산자로 부터의 비트
0	[23:20]	1320에서의 지령을 위한 기록 피연산자로

		부터의 비트
0	[27:24]	1320에서의 지령을 위한 기록 피연산자로
		부터의 비트
0	[31:28]	1320에서의 지령을 위한 기록 피연산자로
		부터의 비트
$5	----	취해진 브랜치 지령의 시작
at		1324
$9	----	1324에서 지령의 위한 타겟 어드레스
0	$c	타겟 어드레스로부터 비트(115c)
0	$5	타겟 어드레스로부터 비트(115c)
0	$1	타겟 어드레스로부터 비트(115c)
0	$1	타겟 어드레스로부터 비트(115c)
$b	----	1324에서 지령을 위한 기록 피연산자
0	$8	1324에서 지령을 위한 복귀 어드레스로
		부터 비트
0	$2	1324에서 지령을 위한 복귀 어드레스로
		부터 비트
0	$3	1324에서 지령을 위한 복귀 어드레스로
		부터 비트
0	$1	1324에서 지령을 위한 복귀 어드레스로

		부터 비트
0	$0	1324에서 지령을 위한 복귀 어드레스로
		부터 비트
0	$0	1324에서 지령을 위한 복귀 어드레스로
		부터 비트
0	$0	1324에서 지령을 위한 복귀 어드레스로
		부터 비트
0	$0	1324에서 지령을 위한 복귀 어드레스로
		부터 비트
$1	----	115c에서 지령 시작

위에 제공된 예에 있어서, PST 및 DDATA 신호는 동기가 맞추어진 상태로 유지되어 DDATA 신호 상에 디스플레이되는 주어진 피연산자는 주어진 지령과 관련될 수 있게 항상 보장될 수 있음을 주목해야 한다. 설명을 명확하게 할 목적으로, 상기 예는 DDATA 신호와 0이 아닌 PST값 사이의 오버랩을 도시하지 않았다. 그러나, 이것을 발생할 수 있지만, PST 와 DDATA 신호 사이의 관계는 깨어질 수 없다.

실시간 디버그 지원

실시간 트래이스 기능에 덧붙여, 본 발명은 실시간 디버그 기능을 실행하기 위한 회로와 방법을 제공한다. 많은 데이터 프로세서, 특히 잘 알려진 시스템에 있어서, 데이터 프로세서는, 시스템 구속, 실패 회로를 확인할 수 없는 능력, 또는 데이터 프로세서가 정상 모드에서 더 이상 동작할 수 없을 때의 프로그램에 기인하여, 디버그 동작 중 정지되지 않을 수도 있다. 본 발명은 데이터 프로세서를 정시킴이 없이 데이터 프로세서의 동작을 실시간에서 최소로 강제시킴에 의한 디버그 동작을 실행한다.

중단점 동작은 전형적으로 프로그래밍 이벤트를 확인하기 위하여 디버그 동작 중 사용된다. 프로그래밍 이벤트가 발생할 때, 중단점 신호는 단정이 되고 데이터는 소프트웨어 프로그램의 상태를 결정하기 위해 검색된다. 트래이스 기능과 유사하게, 중단점 동작은 외부 사용자가 다수의 선택된 레지스터 각각의 상태와 메모리 위치를 알아낼 수 있도록 허용하여 데이터 처리 에러가 확인될 수 있게 한다.

본 발명은 이와 같은 중단점 동작을 수행하기 위하여 다수의 하드웨어 중단점 레지스터(50)와 중단점 회로(100)를 제공한다. 본 발명에 있어서, 프로그램 카운터 또는 프로그램 카운터 마스크에 기초한 중단점 동작, 피연산자 어드레스 범위, 및 데이터 값 또는 데이터 마스크 값은 모두 실현된다. 부가적으로, 본 발명에서 실현되는 중단점 동작은, 정확한 트리거 반응이 다수의 제어 레지스터(40)의 트리거 한정 레지스터(TDR)를 통해 프로그램될 수 있는, 하나 또는 두 개 레벨의 트리거로 구성될 수 있다. 본 발명을 사용하는 중단점 동작의 실행은 후속하여 보다 상세하게 설명된다.

다수의 하드웨어 중단점 레지스터(50) 각각은 외부 개발 시스템(7) 또는 데이터 프로세서(3)의 CPU(2)에 의해 기록될 수 있다. 도 3에 도시된 바와 같이, 어드레스 중단점 레지스터(ABLR 및 ABHR)는 중단점 기능을 트리거하기 위해 사용될 수 있는 데이터 처리 시스템(5)의 피연산자 논리 어드레스 공간 내의 영역을 한정한다. 도 6에 도시된 어드레스 속성 중단점 레지스터(AABR)는 중단점 기능을 위한 트리거 내에서 부합될 다수의 어드레스 속성 및 마스크를 한정한다. ABLR 및 ABLR 어드레스 값과 같이, AABR내에 저장된 속성값은 K-버스(25)를 통해 전송되는 어드레스 속성 신호와 비교된다.

다수의 하드웨어 중단점 레지스터(50) 역시 다수의 프로그램 카운터 중단점 레지스터를 포함한다. 다수의 프로그램 카운터 중단점 레지스터는 중단점 기능을 트리거하기 위해 사용될 수 있는 데이터 처리 시스템(5)의 지령 논리 어드레스 공간 내의 영역을 한정한다. 프로그램 카운터 중단점(PBR) 값은 프로그램 카운터 중단점 마스크(PBMR)레지스터 내에 저장된 값에 의해 마스크될 수 있다. 본 발명의 한 실시예에서, PBMR 내에 저장된 상응하는 '0'값을 갖는 PBR내의 이들 비트만이 데이터 처리 시스템(5)의 프로그램 카운터와의 비교 동작에 사용된다. 프로그램 카운터 값은 데이터 처리 시스템(5)의 CPU(2)의 프로그램 카운터 레지스터(도시 안됨)내에 저장됨을 주목해야 한다.

프로그램 카운터 중단점 레지스터에 있어서, 다수의 하드웨어 중단점 레지스터(50)는 다수의 데이터 중단점 레지스터를 포함한다. 다수의 데이터 중단점 레지스터는 데이터 중단점 레지스터(DBR)와 데이터 중단점 마스크 레지스터(DBMR)를 포함한다. DBR은 데이터 처리 시스템(5)이 디버그 동작을 수행할 때 트리거를 형성하기 위하여 사용될 수 있는 특정 데이터 패턴을 한정하는 데이터 값을 저장한다. DBMR내에 저장된 데이터 마스크 값은 DBR내에 저장된 데이터 값을 마스크 한다. 따라서, 본 발명의 한 실시예에 있어서, DBMR 내에 저장된 상응하는 '0'값을 갖는 DBR내의 이들 비트만이 , 트리거 조건이 부합된다면, K-버스(25)를 통해 전송되는 데이터 신호와의 비교 동작에 사용될 수 있다.

다수의 하드웨어 중단점 레지스터(50) 각각은 다수의 제어 레지스터(40)의 TDR를 사용하는 트리거에 따라서 다르게 구성될 수 있다. TDR에 의해 표시된 구성과 상관없이, 트리거된 중단점의 표시는, DDATA 신호가 포착한 피연산자와 어드레스를 디스플레이하지 않을 때는 언제나, DDATA 신호 상에 제공된다. DDATA 신호의 출력은 도 9에 도시되었다.

도 9는 DDATA 신호의 비트(3 내지 0)를 통해 제공되는 값과 중단점 상태 사이의 관계를 도시한다. CSR의 비트(32 내지 28)는 DDATA 신호의 비트(3 내지 0)에 상응함을 주목해야 한다.

DDATA 신호가 $0의 값을 가질 때, 중단점은 가능하지 않게 된다. DDATA 신호가 $1의 논리값을 제공할 때, 디버그 모듈(10)은 트리거될 레벨 1의 중단점을 기다린다. 이전에 설명한 바와 같이, 레벨 1의 중단점의 속성은 트리거 한정 레지스터(TDR)의 비트(15 내지 0)에 의해 한정된다. DDATA 신호에 의해 $2의 논리값이 제공되면, 레벨 1의 중단점이 트리거린다. DDATA 신호가 $5의 논리값을 제공할 때, 디버그 모듈(10)은 트리거될 레벨 2의 중단점을 기다린다. 이전에 설명한 바와 같이, 레벨 2의 중단점의 속성은 트리거 한정 레지스터(TDR)의 비트(31 내지 16)에 의해 한정된다. DDATA신호에 의해 $6의 논리값이 제공되면, 레벨 2의 중단점이 트리거된다.

중단점 동작의 상태는 또한 CSR의 비트(31 내지 28)에 저장됨을 주목해야 한다.

디버그 동작의 실행에 우선하여, 중단점 값은 다수의 하드웨어 중단점 레지스터(50)중의 적절한 하나에 저장되어야만 한다. 상기 중단점 값은 직렬 인터페이스(30)를 통한 외부 개발 시스템(7) 또는 K-버스(25)를 통한 CPU(2)에 의해 제공될 수 있다.

외부 장치로부터 이 기록 기능을 수행하기 위하여, 외부 개발 시스템(7)은 다수의 하드웨어 중단점 레지스터(50)중의 사전에 선택된 하나에 데이터를 로드하는 명령 내에서 DSDI 및 DSCLK 신호를 사용하여 직렬 인터페이스(30)로 이동한다. 본 발명의 한 실시예에 있어서, 명령의 제 1부는 DSDI 신호에 의해 제공되고 DSCLK 신호에 의해 동기가 맞춰진다. 명령의 부가적인 부분은 후속하여 제공된다. 명령의 모든 부분은 정보의 16 비트 패킷이 종료될 때까지 레지스터(32)에 저장된다. 본 발명의 선택적인 실시예는 본 발명의 현재의 실시예에서 요구되는 16 비트 보다 많거나 적게 포함할 수 있음을 주목해야 한다.

본 발명의 다른 실시예에 있어서, 중단점 값은 다수의 중단점 레지스터(50)중 데이터 프로세서(3)의 CPU(2)에 의해 사전에 선택된 하나에 기록된 값에 의해 결정될 수 있다. CPU(2)가 다수의 중단점 레지스터(50)중 사전에 선택된 하나에 값을 기록할 때, 그 값은 도 3에 도시된 중단점 회로(100)의 임시 레지스터(170)에 제공된다. 그 값은 K-버스(25)를 통해 임시 레지스터(170)에 제공된다. 임시 레지스터(170)로 부터 그 값은 16 비트 패킷으로 레지스터(32)에 전송되고 저장된다. 이전에 설명한 바와 같이, CPU가 CSR에 기록하기 때문에, KDATA 신호를 통해 전송되는 피연산자 값은, CPU(2)에 의해 제어 버스(15)를 통해 제어 회로(20)에 제공되는 제어 신호를 사용하면서, 임시 데이터 레지스터(170)에 포착된다. 따라서 중단점 값은 외부 개발 시스템(7)으로 부터 레지스터(32)에 제공되는 것과 동일하게 다수의 레지스터(40)에 나타난다.

중단점 값이 다수의 중단점 레지스터(50)중의 사전에 선택된 하나에 한번 기록되면, CPU(2)는 정상 동작 모드에서 프로그래밍 코드를 실행한다. 그 프로그래밍 코드의 실행 중, 어드레스 정보는 K-버스(25)의 KADDR 신호를 통해 통신되고, 데이터 정보는 K-버스(25)의 KDATA 신호를 통해 통신된다. 부가적으로 CPU(2) 내의 프로그램 카운터 정보는 프로그램 카운터 중단점 비교 동작의 목적을 위해 임시 PC 레지스터(125)내에 저장된 값을 계속하여 갱신한다.

어드레스, 데이터 및 프로그램 카운터 정보가 하드웨어 회로(100)에 전송됨으로써, 비교기(130,140,180, 및 135)는 정보를 수신하고 선택적으로 데이터 프로세서(3)의 수행에 악 영향을 미침이 없이 선택적으로 비교 동작을 수행한다. 더욱이, 다수의 제어 값은 다수의 제어 레지스터(50)로부터 제어 버스(15)를 통해 하드웨어 회로(100)에 제공된다. 다수의 제어 값은 하드웨어 회로(100)를 사용하여 트리거 동작을 선택적으로 구성하는 TDR의 이전에 설명한 값을 포함한다.

예컨대, 외부 개발 시스템(7)이 어드레스가 어드레스의 특정 범위 내에 있을 때 중단점이 단정이어야 함을 결정할 때, 외부 개발 시스템(7)은 높은 어드레스 값을 직렬 인터페이스(30)를 사용하여 도 3의 ABHR(110)에, 낮은 어드레스 값을 도 3의 ABLR(120)에 기록해야만 한다. 그러나, 비교기(130과 140)가 K-버스(25)의 KADDR 신호에 의해 제공된 어드레스가 ABLR(120)및 ABHR(110)내에 저장된 어드레스에 의해 한정된 범위 내에 있는지의 여부를 결정하기 전에, 다수의 제어 레지스터(40)의 TDR은 제어 버스(15)를 통해 적절한 제어를 제공해야만 한다.

이 예에 있어서, 적절한 제어는 단정이된 가능 어드레스 범위(EAR)비트 와 단정이된 가능 어드레스 중단점 반전(EAI)비트를 포함한다. 이전에 설명한 바와 같이, EAR비트는 중단점이 그 비트가 단정일 때 ABLR(120)및 ABHR(110)에 의해 한정된 총괄적인 범위에서 트리거를 가능케 한다. 또한, 단정일 때, EAI 비트는 중단점이 ABLR(120)및 ABHR(110)에 의해 한정된 범위의 배타적인 어드레스 범위 또는 밖에서 트리거가 가능토록 한다.

EAR 및 EAI 비트의 논리값에 따라서, 비교기(130 및 140)는 KADDR 신호를 통해 전송되는 어드레스 값이 ABHR(110)및 ABLR(120)에 관련된 값보다 큰 지 또는 적은 지를 판단할 것이다. EAR 비트가 단정이고 KADDR 신호를 통해 전송되는 어드레스 값이 ABHR(110)및 ABLR(120)내의 어드레스 값에 의해 한정된 범위 내에 있으면, 비교기(1

30 및 140)는 중단점 신호를 단정으로 하여 사전에 선택된 트리거 포인트가 억세스 되는 것을 표시한다. 중단점 신호가 단정일 때, 제어 회로(20)는 멀티-콘덕터 중단점 제어 신호를 통해 코어(9)의 CPU(2)에 전송되는 BKPT(중단점) 신호, 디버그 인터럽트 신호, 및 Go 신호 중 하나를 단정으로 한다.

중단점 신호가 단정일 때, 중단점 트리거의 발생은 TDR내에서 한정된 것과 같은 반응을 생성한다. 데이터 처리기(3)가 정지하면, BKPT 신호의 단정은 CPU(2)가 백그라운드 디버그 모드에서 실행을 정지하는 것을 강요한다. 백그라운드 디버그 모드는 후속하여 보다 자세하게 논리된다. 백그라운드 디버그 모드로 진입하기 위하여, TDR의 TRC 비트는 %01과 동일해야 한다. TRC 비트는 TDR의 비트(31 및 30)에 상응함을 주목해야 한다. 백그라운드 디버그 모드로 진입하고 프로그램 카운트 값에 기초한 중단점이 단정일 때, CPU(2)는 타겟이 된 지령이 실행되기 전에 데이터 프로세서(3)의 실행을 정지한다. 어드레스와 데이터 비교에 기초한 중단점을 위하여, CPU(2)는 정지 전에 수 개의 부가적인 지령을 실행한다. 이 형태의 중단점 실행은 데이터 프로세서(3)가 트리거에 따라서 정지하므로, 비실시간 디버그 동작을 위해 사용되어야만 한다.

그러나, 데이터 프로세서(3)가 정지하지 않으며, 제어 회로(20)는 디버그 인터럽트 신호를 단정으로 하여 특별 디버그 인터럽트가 CPU(2)에 제공되어야 함을 표시하게 한다. 특별 디버그 인터럽트가 CPU(2)에 제공되어야만 할 때, TDR의 TRC 비트는 %10의 논리값으로 설정된다. TRC 비트가 %10으로 설정되었을 때, 중단점 트리거는 제어 회로(20)에 의해 CPU(2)에 대한 특정 디버그 인터럽트로 변환된다. 특정 디버그 인터럽트는 CPU(2)가 인터럽트에 대한 샘플을 취할 때까지 지연된다. CPU(2)는 지령당 한번만 샘플을 취한다. 특정 디버그 인터럽트가 단정일 때, 프로그램 카운터 값에 기초한 중단점은 타겟이 된 지령이 실행되기 전에 발생한다. 프로그램 카운터 비교 동작이, CPU(2)가 인터럽트가 지연 중인 지를 결정하기 위해 샘플을 취하는 것과 동시에 가능이 되기 때문에, 이것은 가능하다. 어드레스와 데이터 비교에 기초한 중단점을 위하여, CPU(2)는 인터럽트 동작을 실행하기 전에 수 개의 부가적인 지령을 실행할 수 있다.

특정 디버그 인터럽트가 CPU(2)에 의해 인식될 때, CPU(2)는 소프트웨어 프로그램의 실행을 무산시키고 예외 처리 루틴을 초기화시킨다. 예외 처리 루틴의 초기화에 있어서, CPU(2)는 에뮬레이터 동작 모드로 진입한다. CPU(2)가 에뮬레이터 동작 모드에서 동작 중일 때, 모든 인터럽트는 무시됨을 주목해야 한다. 다수의 제어 레지스터(40)의 CSR내의 MAP비트의 상태에 따라서, 에뮬레이터 동작 모드는 모든 메모리 억세스를 다수의 하드웨어 중단점 레지스터(50)의 AABR 레지스터의 TT 및 TM값에 의해 지시되는 특별히 맵핑된 어드레스 공간으로 강요할 수 있다. 이 예에 있어서, TT값은 $2와 동일하고 TM 값은 $5(피연산자) 또는 $6(지령)과 동일하다.

표준 8 바이트의 예외 스택이 생성된 후, CPU(2)는 단일의 예외 벡터를 페치 한다. CPU(2)는 단일의 예외 벡터에 포함된 지령 어드레스에서 실행을 계속한다. CPU(2)내의 디버그 인터럽트 핸들러는 수퍼바이저 지령 집합을 사용하여 다수의 필요한 저장 동작을 수행하도록 프로그램될 수 있다. 예로써, 디버그 인터럽트 핸들러는 모든 프로그램 가시화 레지스터 및 임의의 특정 메모리 위치의 상태를 예약된 메모리 공간에 저장할 수 있다. 예약된 메모리 공간은 메모리(6)내에 위치할 수 있고, 또는 데이터 처리 시스템(5)내의 다른 메모리에 위치할 수 있다. 요구되는 동작이 종료되면, 예외로부터 복귀(TRE) 지령이 실행되고 CPU(2)는 에뮬레이터 동작 모드로부터 빠져나간다. CPU(2)가 디버그 인터럽트 신호를 수신하고 에뮬레이터 동작 모드로 진입할 때, PST신호는 이 조건을 나타내는 $D의 값을 갖는다. PST 신호는 디버그 인터럽트가 처리되는 시간으로부터 디버그 인터럽트 예외 처리 루틴의 제 1의 지령이 페치될 때까지 이 값을 갖는다. 부가적으로 PST 신호는 CPU(2)가 에뮬레이터 동작 모드로부터 빠져나왔음을 나타내기 위하여 $7의 값을 갖는다. 더욱이, 디버그 인터럽트 핸들러가 디버그 예외 처리 루틴의 실행을 한번 종료하면, 외부 개발 시스템(7)은 메모리(6)내의 예약된 메모리 위치 또는 또는 예약된 메모리 위치를 포함하는 데이터 처리 시스템(5)내의 다른 메모리를 억세스할 수 있다.

디버그 인터럽트는 데이터 프로세서(3)의 정상 동작에서 가벼운 강제가 허용되는 실시간 디버그 동작을 위해 사용될 수 있다. 그러나 데이터 프로세서(3)를 정지시키는 것은 선택 사항이 아니다. 데이터 프로세서의 에뮬레이터 모드에 또는 이로부터 입력및 출력은 PST신호 상의 단일의 값에 의해 시작됨을 주목해야 한다.

부가적으로 범위가 아닌 단일의 어드레스, 데이터 값 또는 데이터 값의 마스크, 및 프로그램 카운터 값 또는 프로그램 카운터 값의 마스크에 기초한 다른 비교는 본 발명에서 실현될 수 있음을 주목해야 한다. 상기 예에서 설명된 어드레스 범위 비교에 대해서, 데이터 값, 테이터 값의 마스크, 프로그램 카운터 값, 및 프로그램 카운터 값의 마스크에 기초한 비교는 제어 버스(15)를 통해 적절한 제어 값을 제공하는 TDR을 필요로 한다.

멀티-레벨 트리거 구문

본 발명에 의해 수행되는 실시간 디버그 동작은 두 레벨의 트리거 값을 제공하기 위하여 확장될 수 있다. 본 발명의 현재의 실시예에 있어서, 비트(31 내지 16)는 제 2의 레벨 트리거를 한정하고, 비트(15 내지 0)는 제 1의 레벨 트리거를 한정한다. 트리거가 두 개 레벨의 트리거로 구성될 때, 트리거는 " if-then-if" 의 조건 구문과 유사한 방법으로 기능한다. 예컨대 트리거의 제 1의 조건 또는 레벨이 만족하면, 제 2의 조건 또는 레벨이, 데이터 처리 시스템(5)에 의해 실행될 중단점 기능을 가능하게 하기 전에 만족해야만 한다. 이와 같은 조건적인 트리거 구문은 외부 개발 시스템(7)이 증가된 유연성 및 정확도를 갖고 기능 하도록 허용한다. 본 발명의 이 실시예에 있어서, 트리거 구문의 두 레벨만이 실현된다. 그러나 트리거 구문의 부가적인 레벨이 실현될 수 있다.

두 레벨 트리거 동작의 예로써, 외부 개발 시스템(7)은, $DEADBEEF의 값과 동일한 프로그램 카운터가 뒤따르는 어드레스 $0000FACE에 $FEED 가 아닌 값의 사용자 모드 피연산자 워드 기록으로써 트리거를 한정하기를 원한다고 가정한다. 제 1 레벨 트리거는 어드레스 $0000FACE에 $FEED가 아닌 다른 값의 사용자 모드 피연산자 워드 기록이 된다. 제 2의 레벨 트리거는 프로그램 카운터가 $DEADBEEF의 값과 동일하게 될 때 발생한다.

이 동작을 실행하기 위하여, 외부 개발 시스템(7)은 $DEADBEEF의 값을 프로그램 카운터 중단점 레지스터(PBR) (115)에, 또한 $000000000의 값을 프로그램 카운터 중단점 마스크 레지스터(PBMR)(190)에 기록할 수 있다. 모든 '0' 이 PBMR(190)에 기록될 때, PBR(115)에 저장된 모든 비트는 임시 PC레지스터(125)에 의해 제공된 프로그램 카운터 값과 비교된다. 비교 동작은 다수의 제어 레지스터(40)의 TDR에 저장된 값에 따라서 실행됨을 주목해야 한다.

외부 개발 시스템(7)은 $0000FACE 값을 어드레스 중단점 하위 레지스터(ABLR)에 또한 $0041 값을 도 6의 어드레스 속성 중단점 레지스터(AABR)에 기록해야만 한다. ABLR(120)내의 $0000FACE 값은 TDR의 구성이 비교의 발생을 허용하면, 비교기(140)가 중단점 신호를 단정으로 하는 값을 나타낸다. AABR내의 $0041 값은 TM 값을 %001로, TT 값을 %00으로, SZ 값을 %10으로, 또한 R(판독) 비트를 0 으로 설정한다. TM 과 TT값은 트리거가 사용자 모드 피연산자 워드 기록 동작인 것을 보장하도록 동작한다.

외부 개발 시스템(7)은 $0000FEED의 값을 중단점 회로(100)의 데이터 중단점 레지스터(DBR)(160)에 기록해야만 한다. 외부 개발 시스템(7)은 또한 $000000000의 값을 중단점 회로(100)의 데이터 중단점 마스크 레지스터(DBMR)(150)에 기록해야만 한다. 모든 '0'이 DBMR(150)에 기록될 때, DBR(160)에 저장된 모든 비트는 비교기(180)를 사용하여 KDATA 신호에 의해 제공된 데이터 값과 비교된다. 비교 동작은 다수의 제어 레지스터(40)의 TDR에 저장된 값에 따라서 실행됨을 주목해야 한다.

최종적으로 외부 개발 시스템(7)은 $20022824의 값을 다수의 제어 레지스터(40)의 TDR에 기록해야만 한다. TDR이 이 값을 가질 때, 제 1 레벨 트리거는 다음의 특성을 갖는다. 이 예에서, EBL 비트는 단정, EDLW 비트는 부정, EDWL

비트는 단정, 및 EDWU, EDLL, EDLM, EDUM, 및 EDUU 비트는 모두 부정이다. DI 비트는 단정, AI 및 EAR 비트는 부정, EAL 비트는 단정, 또한 EPC 및 PCI 비트는 부정이다. 이 구성에 대해서, TDR은 DBR(160)내에 기록된 값과 동일하게 저장되지 않는 워드 길이 데이터 값에 대해 중단점 신호를 단정으로 하기 위하여 가능이 될 제 1 레벨 트리거를 구성한다. 더욱이, 상기 조건들이 각각 부합되고 ABLR(120)에 저장된 어드레스 값이 KADDR 신호에 의해 제공된 값에 상응할 때, TDR은 제 1 레벨 트리거를 단정으로 하도록 구성된다. TDR에 의해 결정된 상기 조건들 각각이 만족될 때, 제 1레벨 트리거는 단정이 되고, 제 2 레벨 트리거는 중단점 신호가 단정이 되기 전에 만족해야만 한다.

더욱이, TDR이 이 값을 가질 때, 제 2 레벨 트리거는 다음의 특성을 갖는다. 이 예에서, TRC 값은 %00이고, EBL 비트는 단정, EDLW, EDWL, EDWU, EDLL, EDLM, EDUM, 및 EDUU 비트는 모두 부정이다. DI, AI, EAR, EAL 비트는 부정이고, EPC 비트는 단정 및 PCI 비트는 부정이다. 이 구성에 대해서, TDR은 PBR(115)내에 기록된 값과 동일한 프로그램 카운터 값에 대한 중단점 신호를 단정으로 하도록 가능이 될 제 2의 레벨 트리거를 구성한다. TDR에 의해 결정된 상기 조건들 각각이 만족될 때, 제 2레벨 트리거는 단정이 되고, 중단점 신호는 단정이 될 수 있다.

이전에 언급한 바와 같이, 중단점 신호의 단정은 프로그램 가능한 트리거 반응을 유도할 수 있다. 트리거 반응은 다수의 제어 레지스터(40)의 TDR의 비트(31 및 30)에 프로그램된다. 정확히 프로그램되었을 때, 데이터 프로세서(3)의 CPU(2)의 트리거 반응은 DDATA 신호를 통해 외부 개발 시스템(7)에 가시화 시키고 내부 반응을 야기하지 않을 수 있다. 제 2 트리거 반응은 CPU(2)가 실행을 중지시키고 CPU(2)가 정지 상태임을 PST 신호를 통해 나타내도록 강요한다. 제 3 트리거 반응은 CPU(2)에 대한 디버그 인터럽트를 생성한다. 이전에 설명한 바와 같이, 디버그 인터럽트는 다른 모든 인터럽트보다 우선권에서 높고, CPU(2)를 에뮬레이터 동작 모드로 진입하여 단일의 예외 처리 루틴을 억세스하도록 한다. 에뮬레이터 동작 모드에 또는 이로부터의 진입 과 빠져나옴은 PST 출력 상에 가시화 된다.

디버그 모듈 과 코어의 동시 동작

상기 실시간 디버그 기능을 지원하기 위하여, 본 발명은 코어(9)와 디버그 모듈(10)의 동시의 동작을 지원한다. 디버그 모듈(10)을 위한 K-버스(25)상의 버스 주기를 효과적으로 시간 관리하는 조정 구조는 본 발명에 있어서, 실시간 디버그 동작의 효율적 실행을 허용하도록 실현된다. 조정 구조는 CPU(2)의 내부 지령 페치 파이프라인을 스톨 함으로써 디버그 모듈(10)을 위한 버스 주기를 효과적으로 시간 관리한다. 조정 구조는 그랜팅 디버그 모듈(10)이 K-버스(25)를 억세스하기 전에 모든 피연산자 요구가 CPU(2)에 의해 서비스될 때까지 기다린다. 디버그 모듈(10)은 버스 그랜트 신호를 수신하고, ABHR(110)내에 저장된 요구되는 어드레스와 도 6의 AABR내에 저장된 속성으로 K-버스(25)를 구동시킨다. 판독 요구에 대해서, 반응은 KDATA 신호 상에서 유도되고 디버그 모듈(10)에 의해 임시 데이터 레지스터(170)에 포착된다. 기록 레퍼런스를 위하여, 기록 피연산자는 DBR(160)으로부터 KDATA 신호 상에 유도된다. 디버그 모듈(10)은 K-버스(25)를 CPU(2)에 되돌려 포기하기 전에 한 버스 트랜잭션에 이 동작을 수행한다. 따라서 디버그 모듈(10)은 CPU(2)의 실시간 동작 특성에 강제성을 최소화한다. K-버스(25)에 대한 조정 구조는 CPU(2)에 저장된 논리에 의해 실행된다.

CPU(2)와 디버그 모듈(10)의 동시 동작을 도시하는 예에 있어서, 외부 개발 시스템(7)은 디버그 기능 중 메모리 판독 동작을 실행한다고 가정한다. 메모리 판독 동작을 실행하기 위하여, 외부 개발 시스템(7)은 메모리 위치를 억세서 위한 명령 및 어드레스 내에서 DSDI 및 DSCLK 신호를 사용하여 직렬 인터페이스(30)로 이동한다. 본 발명의 한 실시예에 있어서, 명령의 제 1 부분은 DSDI 신호에 의해 제공되고, DSCLK 신호에 의해 동기가 맞추어진다. 명령의 부가적인 부분은 정보의 16비트 패킷이 종료될 때까지 레지스터(32)에 저장된다. 본 발명의 선택적인 실시예는 본 발명의 현재의 실시예에 있어서 요구되는 16비트보다 많거나 적은 비트를 포함할 수 있다.

여기에서 설명된 본 발명의 실시예에 있어서, 사전에 선택된 메모리 위치에 값을 기록하는 지령은 16비트 옵코드와 32비트 어드레스를 갖는 48비트 길이이다. 레지스터(32)는 16 비트만의 길이이다. 따라서, 정보의 제 1의 16비트 패킷이 종료될 때, 그 정보는 제어 버스(15)를 통해 제어 회로(20)에 전송된다. 제어 회로(20)는 수행되어야 할 동작의 형태를 결정하기 위하여 정보의 제 1의 16비트 패킷을 디코드 한다. 제어 회로(20)가 한번 데이터의 제 1의 16비트 패킷을 디코드하고 프로그램된 메모리 위치로부터 판독 어드레스가 수행될 것임을 결정하면, 다수의 하드웨어 중단점 레지스터(50)의 ABHR(110)에 로드될 어드레스 정보의 후속의 32비트를 가능으로 하기 위하여, 제어회로(20)는 제어 버스(15)를 통해 제어 신호를 제공한다.

후속의 데이터는 외부 개발 시스템(7)에 의해 DSDI신호를 통해 제공된다. 이전에 설명한 바와 같이, DSDI신호에 의해 제공된 후속의 데이터 DSCLK 신호를 통해 동기가 맞추어진다. 각각의 데이터 비트는 정보의 16 비트 패킷이 종료될 때까지 레지스터(32)에 저장된다. 정보의 제 2의 16 비트 패킷이 종료될 때, 정보는 직렬 정보 신호를 통해 다수의 하드웨어 중단점 레지스터(50)의 ABHR(110)에 전송된다. 후속하여, 부가적인 정보는 외부 개발 시스템(7)에 의해 DSDI 및 DSCLK 신호를 통하여 레지스터(32)에 직렬로 제공된다. 정보의 제 3의 16 비트 패킷이 종료될 때, 정보는 직렬 정보 신호를 통해 다수의 하드웨어 중단점 레지스터(50)의 ABHR(110)에 전송된다.

피연산자 및 어드레스 정보가 수신되었을 때, 제어 회로(20)는 버스 요구 신호를 단정으로 한다. 버스 요구 신호는 K-버스(25)를 통해 CPU(2)에 전송된다. 또, 버스 요구 및 버스 그랜트 신호는 디버그 모듈(10)과 CPU(2) 사이의 직접 접속이다. 버스 조정의 상세한 사항은 이전에 설명되었다.

버스 그랜트 신호가 단정일 때, ABHR(110)은 자체에 저장된 어드레스 정보를 KADDR 신호를 통해 K-버스(25)에 제공하기 위하여 가능이 된다. 어드레스 정보는 자체의 메모리 위치를 판독하기 위하여 메모리(6)에 의해 디코드 된다. 판독 종작이 실행되기 때문에, 메모리(6)는 K-버스(25)의 KDATA 신호 상의 메모리 위치에 상응하는 피연산자를 출력한다. 억세스되는 메모리는 메모리(6)에 제한되지 않음을 주목해야 한다. 억세스되는 메모리는 메모리(6)가 될 수 있거나 또는 데이터 처리 시스템(5)내의 다른 메모리 위치가 될 수 있다. KDATA 신호를 통해 전송되는 피연산자는 후속하여 중단점 회로(100)의 임시 데이터 레지스터(170)에 포착된다. 피연산자는 후속적으로 내부 데이터 신호를 통해 레지스터(32)에 전송된다. 그 후 피연산자는 직렬 인터페이스(30)에 의해 DSDO신호를 통해 32 타이밍 주기 내에서 직렬로 이동해 나온다. DSDO 신호는 외부 개발 시스템(7)에 제공된다.

부가적으로 피연산자가 KDATA 신호를 통해 전송된 후, CPU(2)는 처리 속도에 최소의 강요를 갖는 데이터 처리 지령을 다시 시작한다. 따라서, CPU(2)는 직렬 인터페이스(30)가 정보를 출력하고 있는 동안 데이터 처리 기능을 수행할 수 있다.

본 발명은 데이터 프로세서(3)가 특정 디버그 모드에서 동작하는 것을 요구함이 없이 CPU(2)와 디버그 모듈(10)의 동시의 동작을 제공한다. 코어(9)와 디버그 모듈(10) 사이의 데이터, 어드레스 및 제어 정보를 통신하기 위한 K-버스(25)의 사용은 디버그 모듈(10)이 CPU(2)와 같이 동일한 메모리 위치와 내부 레지스터를 억세스 하는 것을 허용한다. 따라서 디버그 모듈은 CPU(2)가 억세스 가능한 모든 메모리 위치와 내부 레지스터를 억세스할 수 있다.

CPU(2)와 디버그 모듈(10)을 동시에 동작시키는 능력은 외부 개발 시스템(7)이 프로세서의 동작에 중요한 충돌을 일으키지 않고 내부 레지스터와 데이터 프로세서(3)의 메모리 내에서 증가된 가시화를 갖도록 허용한다.

부가적으로, 디버그 모듈(10)과 CPU(2)가 동일한 내부 레지스터와 메모리 위치를 억세스할 능력을 갖는 동안, CPU(2)와 코어(9)의 나머지 부분은, 다수의 제어 레지스터(40)의 CSR내에 디버그 레지스터에 금지 프로세서 기록(IPW) 비트가 설정되었을 때, 다수의 중단점 레지스터(50)내에 저장된 값을 수정하지 않을 수도 있다. CSR내에 IPW 비트가

설정되었을 때, CPU(2)에 의해 초기화된 임의의 기록 동작은 금지된다. 더욱이, IPW 비트는 외부 개발 시스템(7)에 의해 제공되는 명령에 의해서만 수정될 수 있다.

IPW 비트를 설정하기 위해, 외부 개발 시스템(7)은 데이터를 CSR에 로드하는 명령 내에서 DSDI 및 DSCLK 신호를 사용하여 직렬 인터페이스(30)로 이동한다. 이전에 언급된 바와 같이, 본 발명의 한 실시예에 있어서, 명령의 제 1부분은 DSDI 신호에 의해 제공되고 DSCLK 신호에 의해 동기가 맞춰진다. 명령의 부가적인 부분은 후속하여 제공된다. 명령의 모든 부분은 정보의 16 비트 패킷이 종료될 때까지 레지스터(32)에 저장된다. 본 발명의 선택적인 실시예는 본 발명의 현재의 실시예에서 요구되는 16 비트 보다 많거나 또는 적은 비트를 포함할 수 있다.

여기에서 설명된 본 발명의 실시예에 있어서, 값을 CSR에 기록하는 지령은 16비트의 옵코드와 32비트의 데이터를 갖는 48비트 길이이다. 레지스터(32)는 16비트만의 길이이다. 이 기록 지령을 수행하기 위하여, 외부 개발 시스템(7)은 데이터를 CSR에 로드 하는 명령 내에서 DSDI및 DSCLK 신호를 사용하여 직렬 인터페이스(30)로 이동한다. 명령의 제 1부분은 DSDI 신호에 의해 제공되고 DSCLK 신호에 의해 동기가 맞춰진다. 명령의 부가적인 부분은 후속하여 제공된다. 명령의 모든 부분은 정보의 16 비트 패킷이 종료될 때까지 레지스터(32)에 저장된다.

여기에서 설명된 본 발명의 실시예에 있어서, 값을 CSR에 기록하는 지령은 16비트의 옵코드와 32비트의 데이터를 갖는 48비트 길이이다. 레지스터(32)는 16비트만의 길이이다. 따라서 정보의 제 1의 16비트 패킷이 종료될 때, 그 정보가 디코드 되고 후속 하는 동작이 구성되는 제어회로(20)에 정보가 전송된다. 정보의 제 2의 16 비트 패킷이 레지스터(32)에 로드 되었을 때, 정보는 직렬 정보 신호를 통해 다수의 제어 레지스터(40)에 전송된다. 제어 버스(15)에 의해 제어 회로(20)로부터 제공되는 제어 정보를 사용하여, 직렬 정보 신호의 내용은 지정 레지스터, CSR의 상부 부분에 로드 된다. 유사한 방법으로, 정보의 제 3의 16 비트 패킷이 레지스터(32)에 로드 되고, 이후 다수의 제어 레지스터(40)에 전송된다. 또, 제어 회로(20)에 의해 소스로 된 신호를 사용하여, 직렬 정보 신호의 내용은 CSR의 낮은 부분에 로드 된다. 이 방법으로, 전체 48 비트 지령이 직렬로 디버그 모듈(10) 내로 이동하고, 여기에서 이 정보는 디코드 되고 피연산자 값은 CSR에 로드 된다.

후속 하는 데이터는 DSDI 신호를 통해 외부 개발 시스템(7)에 의해 제공된다. 이전에 설명한 바와 같이, DSDI신호에 의해 제공되는 후속 하는 데이터는 DSCLK 신호를 통해 동기가 맞추어진다. 각 데이터 비트는 정보의 16비트 패킷이 종료될 때까지 레지스터 (32)내에 저장된다. 정보의 제 2의 16비트 패킷이 종료될 때, 정보는 직렬 정보 신호를 통해 다수의 제어 레지스터(40)에 전송된다. 정보의 제 2의 16비트 패킷은 비트(31 내지 16)로써 CSR에 저장된다. 후속하여, 부가적인 정보는 외부 개발 시스템(7)에 의해 DSDI 및 DSCLK 신호를 통해 레지스터(32)에 직렬로 제공된다. 정보의 제 3의 16비트 패킷이 종료될 때, 그 정보는 직렬 정보 신호를 통해 다수의 제어 레지스터(40)에 전송된다. 정보의 제 3의 16비트 패킷은 비트(15 내지 0)로써 CSR에 저장된다.

본 발명의 다른 실시예에 있어서, IPW비트가 아닌 모든 값은 데이터 프로세서(4)의 CPU(2)에 의해 CSR에 기록된 값으로써 구성될 수 있다. CPU(2)가 상기 값을 CSR에 기록할 때, 상기 값은 도 6에 도시된 중단점 회로(100)의 임시 레지스터(170)에 제공된다. 상기 값은 K-버스(25)의 KDATA 신호를 통해 임시 레지스터(170)에 제공된다. 임시 레지스터(170)로 부터, 상기 값은 16 비트 패킷 내에서 레지스터(32)에 전송되고 거기에 저장된다. 따라서 다수의 제어 레지스터(40)에서, 상기 값은 외부 개발 시스템(7)으로 부터 레지스터(32)에 제공된 것과 동일하게 나타난다.

IPW 비트 및 CSR에 저장된 다른 값은 제어 버스(15)를 통해 하드웨어 회로(100)에 제공된다. IPW 비트가 단정이고 CPU(2)가 ABHR(110), ABLR(120), DBMR(150), DBR(160), PBMR(190), 및 PBR(125) 중 하나에 저장된 값을 수정하려 할 때, IPW 비트는 코어(9)에 의해 초기화된 임의의 기록 동작을 불가능으로 하기 위하여 이전에 언급한 각 레지스터에 제공된다.

다수의 하드웨어 중단점 레지스터(50)에 대한 기록 동작을 효과적으로 불가능으로 하기 위한 IPW비트의 사용은, 데이터 프로세서(3)내의 실패 회로와 에러 프로그램을 분리하고 확인하기 위하여 외부 개발 시스템(7)에 의해 프로그램된 조건이 CPU(2)에 의해 재 기록되지 않게 보장한다.

리세트 예외 처리 중 디버그 동작

본 발명은, 리세트 동작을 위한 예외 처리 루틴이 초기화되기 전에 외부 개발 시스템(7)이 데이터 프로세서(3)내의 중단점 동작을 실행하는 것을 허용하는 메카니즘을 제공한다. 외부 리세트 신호가 단정일 때, 다수의 제어 레지스터(40) 각각의 내용은 초기화된다. 외부 리세트 신호가 후속적으로 부정일 때, 실제 예외 처리 루틴의 시작 전에 데이터 프로세서(3)가 활동하지 않는 시간의 윈도우가 존재한다. 활동하지 않는 시간 중, 외부 중단점 신호 $\overline{BKPT}$ 가 외부 개발 시스템(7)에 의해 단정일 때, 본 발명은 타겟 메모리 값을 데이터 처리 시스템(5)에 다운 로드하고 요구되는 임의의 하드웨어 레지스터 구성을 수행하기 위한 메카니즘 및 방법을 제공한다. 타겟 메모리 값은 단일의 값 또는 다수의 값이 될 수 있다.

타겟 메모리 값이 외부 개발 시스템(7)에 의해 한번 다운 로드 되면, 다수의 제어 레지스터(40) 및 다수의 하드웨어 중단점 레지스터(50) 중 하나 또는 둘 모두 구성될 수 있다. 부가적으로, CPU(2), MMU/제어기(4), 및 시스템 버스 제어기(8)내에 포함된 하드웨어 레지스터 역시 프로그램될 수 있다. 최종적으로, 데이터 프로세서(3)는 " Go" 명령을 사용하여 재 시작되어야만 한다. 그러나, 본 발명에 있어서, 일련의 하드웨어 동작은 CPY(2)가 중단점 동작을 실행하는 동안 수정되는 하드웨어 레지스터의 기능으로써 " Go" 명령의 수신 상에서 실행된다. 외부 개발 시스템(7)이 CPU(2)의 프로그램 카운터 레지스터를 수정하지 않으면, " Go" 명령은 데이터 프로세서가 정상의 리세트 예외 처리 루틴을 실행할 수 있게 한다. 그러나, 외부 개발 시스템(7)이 CPU(2)의 프로그램 카운터 레지스터를 수정하면, " Go" 명령은 데이터 프로세서가 정상의 리세트 예외 처리 루틴을 통과시키고 프로그램 카운터 레지스터에 로드될 어드레스에 의해 지정된 지령의 실행을 시작 할 수 있게 한다.

이와 같은 리세트 및 중단점 동작 중의 데이터 프로세서(3)의 동작은 후속하여 보다 상세하게 설명된다. 리세트 신호는 단정이고 데이터 프로세서(3)의 CPU(2)에 제공되는 것으로 가정한다. 리세트 신호가 부정일 때, 외부 개발 시스템(7)은 $\overline{BKPT}$ 신호를 단정으로 한다. 디버그 모듈(10)의 제어 회로(10)는 다수의 중단점 제어 신호의 BKPT신호를 K-버스(25)를 통해 CPU(2)에 단정으로 한다. 또, 제어 인터페이스는 디버그 모듈(10)과 CPU(2) 사이의 직접 접속임을 주목해야 한다.

CPU(2)가 단정이 된 BKPT신호를 수신할 때, CPU(2)는 활동하지 않는 시간 윈도우와 데이터 프로세서(3)가 정지했다는 신호를 지연시킨다. 상기 시간 중, 데이터 프로세서(3)가 정지된 동작을 갖는 것을 외부 개발 시스템(7)에 표시하기 위하여 PST 신호는 $F 의 값을 갖는다. 이와 같이, CPU(2)와 데이터 프로세서(3)는 리세트 예외 처리 루틴을 시작하기 전에 정지된다.

데이터 프로세서(3)가 정지된 동작을 갖는 것을 PST신호가 한번 표시하면, 외부 개발 시스템(7)은 다수의 데이터, 어드레스 및 제어 값을 DSDI신호와 DSCLK 신호를 통해 데이터 프로세서(3)에 다운 로드 시킨다. 다수의 데이터, 어드레스, 및 제어 값은 프로그램의 메모리 영상 또는 메모리(6)에 저장될 데이터 값이 될 수 있다. 부가적으로 다수의 데

이터, 어드레스, 및 제어 값은 다수의 제어 레지스터(40) 및 다수의 하드웨어 중단점 레지스터(50)와 같은 다수의 구성 레지스터 각각을 위하여 다수의 초기화 값을 제공할 수 있다. 다수의 데이터, 어드레스, 및 제어 값이 DSDI 신호를 통해 데이터 프로세서(3)에 제공될 때, 상기 값의 각 비트는 직렬 인터페이스(30)를 통해 레지스터(32)에 제공됨을 주목해야 한다. 레지스터(32)가 풀(full)일 때, 정보 한 패킷은 직렬 정보 신호를 통해 다수의 하드웨어 중단점 레지스터(50)에 제공된다. 다수의 하드웨어 중단점 레지스터(50)에 있어서, 정보의 상기 패킷은 임시 데이터 레지스터(170)와 같은 레지스터에 저장된다. 임시 데이터 레지스터(170)는 후속하여 저장된 정보를 KDATA 신호를 통해 K-버스(25)에 제공한다. K-버스(25)는 상기 정보를 코어(9)의 나머지 부분에 전송한다. 이 동작은 이전에 상세하게 설명되었으므로, 내부 저장 절차의 부가적인 설명은 생략한다. 다수의 데이터, 어드레스, 및 제어 값이 데이터 프로세서(3)의 적절한 메모리 또는 레지스터 위치에 저장된 후, 외부 개발 시스템(7)은 DSDI 및 DSCLK 신호를 통해 " Go" 명령을 데이터 프로세서(3)에 제공한다. 다수의 데이터, 어드레스, 및 제어 값과 같이, " Go" 명령은 DSDI 신호를 통해 데이터 프로세서(3)에 제공된다. " Go" 명령의 각 비트는 직렬 인터페이스(30)를 통해 레지스터(32)에 제공된다. 레지스티(32)가 풀(full)일 때, 정보의 한 패킷은 제어 버스를 통해 제어 회로(20)에 제공된다. 제어 회로(20)는 " Go" 명령을 디코드하고 K-버스(25)를 통해 CPU(2)에 Go 신호를 제공한다.

CPU(2)가 Go신호에 응답할 때, 실행하는 지령은 외부 개발 시스템(7)에 의해 제공된 다수의 어드레스, 데이터 및 제어 값에 의해 수정된 레지스터에 의해 결정된다. 예컨대 CPU(2)내의 프로그램 카운터(상세하게 도시 안됨)가 수정되면, CPU(2)는 수정된 프로그램 카운터 내에 저장된 새로운 지령 어드레스로부터 프로세서 실행을 시작한다. 그러나, CPU(2)내의 프로그램 카운터가 수정이 되지 않으면, CPU(2)는 정상 리세트 예외 처리 루틴을 계속한다.

리세트 예외 처리 루틴의 실행 이전에 중단점 데이터 처리기(3)에 대한 본 발명의 능력은, 외부 개발 시스템(7)이, 요구되는 방법으로, 리세트 예외 처리 루틴 중 그 값의 재 기록되는 기회가 없이, 데이터 프로세서(3)를 구성하도록 허용하기 위한 효율적이고 유용한 메카니즘을 증명한다.

리세트 예외 처리 루틴의 실행되기 전에 중단점 동작을 실행하는 능력은 또한, 외부 개발 시스템(7)이 임의의 비휘발성 메모리 장치의 프로그래밍을 통과하도록 허용한다. 새로운 소프트웨어 영상을 기록 가능 메모리 장치에 로딩 함으로써, 비휘발성 메모리 장치에 프로그램된 코드는 쉽게 통과될 수 있다. 외부 개발 시스템(7)은 80비트 지령에서 이동함으로써 임의의 하드웨어 레지스터를 억세스할 수 있다. 80비트 지령은 16비트의 옵코드, CPU 공간에 맵핑된 32 비트의 어드레스, 및 32 비트의 피연산자 데이터를 포함한다. 완성된 명령이 한번 수신되면, 디버그 모듈(10)은 메모리 레퍼런스 동작과 유사한 방법으로 동작한다. K-버스(25)는 요구되고, CPU(2)에 의해 그랜트 된다. CPU 공간 K-버스 주기는 디버그 모듈(10)에 의해 초기화되고, 적절한 하드웨어 레지스터가 기록된다.

백그라운드 디버그 모드

데이터 프로세서(3)가 특정 백그운드 디버그 모드에서 동작할 때, 디버그 모듈(10)이 모든 메모리와 레지스터 값을 완전히 억세스 하여 실패 회로와 에러 프로그램이 분리되고 확인될 수 있도록 하기 위하여, CPU(2)는 정지한다. 레지스터와 메모리 위치는, 백그라운드 디버그 모드에 있을 때, 보여질 수 있고 선택적으로 변환될 수 있다. 더욱이, 시험 특성이 가능이 될 수 있다. 외부 개발 시스템(7)은 특정 백그라운드 디버그 모드에서 동작 중 일 때, 디버그 모듈(10)을 통해 내부 정보를 쉽게 억세스할 수 있고, CPU(2)가 이 동작 모드에서는 정지해야만 하므로, 정상 지령 실행은 지연된다. CPU(2)가 정지하였을 때, PST신호는 $F값을 갖음을 주목해야 한다.

동작의 특정 백그라운드 디버그 동작 모드는 많은 소스에 의해 생성될 수 있다. 예컨대, 백그라운드 디버그 동작 모드로 진입하는 데이터 프로세서(3)를 선택적으로 기능으로 하기 위하여, $\overline{BKPT}$ 신호는 외부 개발 시스템(7)에 의해 외부적으로 생성될 수 있다. 부가적으로, 내부적으로 생성된 중단점 동작과 돌발 예외 조건은 백그라운드 디버그 동작 모드로 진입하는 데이터 프로세서(3)를 가능으로 할 수 있다.

데이터 프로세서(3)가 백그라운드 디버그 동작 모드로 한번 진입하면, 데이터 프로세서(3)의 정상 동작은 정지하고, 개발 시스템(7)의 제어 아래, 특정 기능이 디버그 모듈(10)에 의해 수행된다. 디버그 모듈(10)과 외부 개발 시스템(7) 사이의 모든 통신은 직렬 인터페이스(30)와 DSDI, DSDO, 및 DSCLK 신호의 각각을 통한다.

이전에 언급한 바와 같이, 본 발명의 현재의 실시예에 있어서, 백그라운드 디버그 모드는 PST 신호를 통해 외부 사용자에게 알려진다. PST 신호 상의 $F 값은 프로세서가 정지했고 백그라운드 디버그 동작 모드에서 동작함을 표시한다. 백그라운드 디버그 모드로 진입의 표시는 매우 중요하다, 그 이유는 이것이 외부 사용자가 직렬 통신 채널을 통해 지원되는 디버그 명령을 초기화하도록 허용하는 신호를 제공하기 때문이다. 따라서 직렬 인터페이스(30)를 통해 제공되는 직렬 포트는 메모리 위치와 내부 레지스터로부터 디버그 정보를 검색하기 위하여 사용될 수 있다.

MAP 비트가 부정이면, 모든 에뮬레이터 모드 메모리 레퍼런스는 정상의 수퍼바이저 억세스로 맵핑되어, 온-칩 메모리, 외부 메모리, 및 입력/출력 장치와 같은 시스템 자원이 레퍼런스될 수 있도록 한다.

최종적으로, " Go" 명령이 외부 개발 시스템(7)으로부터 디버그 모듈(10)에 직렬로 이동된 후, Go 신호는 단정이 된다. 일부 이벤트가 데이터 프로세서(3)를 정지시킨 후, Go신호는, 백그라운드 디버그 동작 모드로의 진입을 강요하면서, 데이터 프로세서(3)의 실행을 재 시작한다. TDR이 이 형태의 반응을 강요하도록 프로그램되었다면, 데이터 프로세서(3)는 제어 회로(20)에 의한 중단점 신호의 단정에 의해 정지될 수 있다. 부가적으로 데이터 프로세서(3)는 데이터 프로세서(3)의 정지 지령의 실행 또는 폴트-온-폴트 (fault-on-fault)조건 의 발생에 의해 정지될 수 있다.

발명의 효과

요약에서, 본 발명은, 데이터 프로세서가 백그라운드 디버그 모드와 같은 특정 동작 모드로 진입할 필요가 없는 실시간 트래이스 및 디버그 기능을 실행되도록 하는, 데이터 프로세서 및 동작 방법을 제공한다. 부가적으로 본 발명은 실시간 디버그 동작 중, 멀티-레벨 트리거를 실행하기 위한 방법을 제공한다. 이들 유일의 특성 각각은 데이터 처리 시스템(5)에서의 실패를 확인하고 분리시킴에 있어서 외부 사용자에게 큰 기능성과 유연성을 제공한다.

이전에 언급한 바와 같이, 본 발명은 중요하고 지금까지 없었던 실시간 트래이스 기능을 제공한다. 실시간 트래끼스 기능은 외부 개발 시스템(7)과 같은 외부 사용자가, 외부 버스의 형태 또는 유용성을 상정함이 없이, 데이터 프로세서(3)의 내부 동작을 동적으로 관찰할 수 있도록 한다. 더욱이, 실시간 트래이스 기능은, 데이터 처리 시스템(5)의 효율과 속도와 충돌을 일으킴이 없이, 시스템의 내부 동작에 이와 같은 관찰 능력을 제공한다. 데이터 처리 시스템(5)의 디버그 모듈(10)은 내부 동작 정보를 제공하기 위한 병렬 출력 포트를 DDATA 및 PST신호를 통해 외부 사용자에게 제공한다. DDATA신호는 피연산자 값을 반영하는 데이터를 제공하고, PST신호는 CPU(2)의 실행 상태를 반영하는 엔코드된 상태 정보를 제공한다. 더욱이, 외부의 가시적인 어드레스 버스 또는 외부의 가시적인 데이터 버스를 요구함이 없이 외부 개발 시스템이 정확한 프로그램 흐름을 트래이스 하는 것을 허용하기 위하여, DDATA신호는 또한 포착된 지령 어드레스 프로그램 흐름 변화를 제공한다. DDATA 신호 상에 디스플레이되는 정보는 PST 신호 상의 상태 정보와 동기가 맞추어짐을 주목해야 한다. 따라서 데이터 프로세서(3)의 정상 동작을 지연시킴이 없이 데이터 프로세서(3)내의 키 변수의 상태를 결정하기 위하여, 외부 사용자는 DDATA 및 PST 신호를 디코드할 수 있다. 달리 설명하면, 내부 정보를 외부 개발 시스템에 제공하기 위하여 데이터 프로세서(3)는 특정 디버그 또는 에뮬레이터 모드에서 동작하도록 요구되지는 않는다.

부가적으로, 본 발명은 실시간 디버그 기능을 실행하기 위한 회로와 방법을 제공한다. 많은 데이터 프로세서, 특히 내장된 시스템에 있어서, 시스템 제한, 또는 데이터 프로세서가 더 이상 정상 모드에서 동작하지 않을 때 실패 회로 또는 프로그램을 확인할 수 없는 능력에 기인하여, 데이터 프로세서는 디버그 동작 중 정지 안할 수 도 있다. 본 발명은 데이터

프로세서의 실시간 동작상에서 강제를 최소화함에 의해 또한 데이터 프로세서를 정지시킴이 없는 디버그 동작을 실행한다.

중단점 동작은 전형적으로 프로그래밍 이벤트를 확인하기 위하여 디버깅 동작 중 사용된다. 프로그래밍 이벤트가 발생하였을 때, 소프트웨어 프로그램의 상태를 결정하기 위하여 중단점 신호는 단정이 되고 데이터는 검색된다. 트래이스 기능과 같이, 중단점 동작은 외부 사용자가 다수의 선택된 레지스터 와 메모리 위치 각각의 상태를 확인할 수 있게 하여 데이터 처리 에러가 확인될 수 있게 한다. 이와 같은 중단점 동작을 수행하기 위하여, 본 발명은 다수의 하드웨어 중단점 레지스터(50)와 중단점 회로(100)를 제공한다. 본 발명에 있어서, 프로그램 카운터 또는 프로그램 카운터 마스크, 피연산자 어드레스 범위, 및 데이터 값 또는 데이터 마스크 값에 기초한 중단점 동작은 모두 실현된다. 부가적으로 본 발명에서 실현되는 중단점 동작은, 정확한 트리거 응답이 다수의 제어 레지스터(40)의 트리거 한정 레지스터(TDR)를 통해 프로그램될 수 있는, 1 레벨 또는 2레벨 트리거로 구성될 수 있다.

또한, 본 발명은, 데이터 프로세서(3)가 특정 디버그 모드에서 동작하는 것을 요구함이 없이, CPU(2)와 디버그 모듈(10)의 동시의 동작을 제공한다. 코어(9)와 디버그 모듈(10) 사이의 데이터, 어드레스 및 제어 정보를 통신하기 위한 K-버스(25)의 사용은 디버그 모듈(10)이 CPU(2)의 동일한 내부 레지스터와 메모리 위치를 억세스 하도록 허용한다. 따라서, 디버그 모듈은 CPU(2)가 억세스할 수 있는 모든 메모리 위치와 내부 레지스터를 억세스할 수 있다. CPU(2)와 디버그 모듈(10)을 동시에 동작시킬 수 있는 능력은 외부 개발 시스템(7)이 프로세서의 동작과 심각한 충돌 없이 데이트 프로세서(3)의 내부 레지스터 및 메모리 내에서 증가된 가시성을 갖도록 한다.

더욱이, 디버그 모들(10)과 CPU(2)가 둘다 동일한 내부 레지스터와 메모리 위치를 억세스 하는 능력을 갖는 동안, 다수의 제어 레지스터(40)의 CSR내의 디버그 레지스터에 대한 금지 프로세서 기록(IPW)비트가 설정되었을 때, CPU(2)와 코어(9)의 나머지 부분은 다수의 중단점 레지스터(50)에 저장된 값을 수정 안할 수도 있다. CSR 내의 IPW 비트가 설정되었을 때, CPU(2)에 의해 초기화되는 임의의 기록 동작은 금지된다. 더욱이, IPW비트는 외부 개발 시스템(7)에 의해 제공되는 명령에 의해서만 수정될 수 있다. 다수의 하드웨어 중단점 레지스터(50)에 대한 기록 동작을 불가능으로 하기 위한 IPW비트의 사용은 외부 개발 시스템(7)에 의해 프로그램되고 데이터 프로세서(3)내에 실패 회로와 에러 프로그램을 분리하고 확인하기 위해 요구되는 이벤트는 CPU(2)에 의해 재 기록되지 않는다.

더욱이, 본 발명은, 리세트 동작을 위한 예외 처리 루틴이 초기화되기 전에 외부 개발 시스템(7)이 데이터 프로세서(3)내의 중단점 동작을 실행하도록 하는 메카니즘을 제공한다. 외부 리세트 신호가 단정일 때, 다수의 제어 레지스터(40)의 각 내용은 초기화된다. 외부 리세트 신호가 후속하여 부정일 때, 실제 예외 처리 루틴의 시작 전에 데이터 프로세서(3)가 활동하지 않는 시간의 윈도우가 존재한다. 활동하지 않는 시간 중, 외부 중단점 신호 $\overline{BKPT}$ 가 외부 개발 시스템(7)에 의해 단정일 때, 본 발명은 타겟 메모리 값을 데이터 처리 시스템(5)에 다운 로드 하기 위한 메카니즘 및 방법을 제공하여, 임의의 하드웨어 레지스터 구성이 수행될 수 있다. 타겟 메모리 값은 단일의 값 또는 다수의 값이 될 수 있다.

리세트 예외 처리 루틴이 실행되기 전에 중단점 동작을 실행하는 능력은 또한 외부 개발 시스템(7)이 임의의 비휘발성 메모리 기억장치의 프로그래밍을 통과시킬 수 있게 한다. 새로운 소프트웨어 영상을 기록 가능 메모리 장치에 로딩함으로써, 비휘발성 메모리 장치에 프로그램된 코드는 쉽게 통과될 수 있다. 외부 개발 시스템(7)은 80비트 지령에서 이동함으로써 임의의 하드웨어 레지스터를 억세스할 수 있다. 80비트 지령은 16비트의 옵코드, CPU 공간에 맵핑된 32 비트의 어드레스, 및 32 비트의 피연산자 데이터를 포함한다. 완성된 명령이 한번 수신되면, 디버그 모듈(10)은 메모리 레퍼런스 동작과 유사한 방법으로 동작한다. K-버스(25)는 요구되고, CPU(2)에 의해 그랜트 된다. CPU 공간 K-버

스 주기는 디버그 모듈(10)에 의해 초기화되고, 적절한 하드웨어 레지스터가 기록된다.

리세트 예외 처리 루틴의 실행 이전에 중단점 데이터 처리기(3)에 대한 본 발명의 능력은, 외부 개발 시스템(7)이, 요구되는 방법으로, 리세트 예외 처리 루틴 중 그 값의 재 기록되는 기회가 없이, 데이터 프로세서(3)를 구성하도록 허용하기 위한 효율적이고 유용한 메카니즘을 증명한다.

여기에서 설명된 본 발명의 실시예는 한 예로써 제공되었다. 그러나 다른 많은 실시예가 여기에서 설명된 기능을 실행하기 위해 존재할 수 있다. 본 발명이 특정 실시예를 참조하여 도시되고 설명되었지만, 당업자에 있어서, 추가의 수정과 개선을 일어날 수 있다. 따라서, 본 발명은 도시된 특정 형태에 제한되지 않고, 첨부된 청구 범위가 본 발명의 범주에서 벗어나지 않는 모든 수정을 보호하는 것을 이해해야 한다.

(57) 청구의 범위

청구항 1.

데이터 프로세서에 있어서,

중단점 예외 루틴을 실행하기 위한 제 1 디버그 스테이트와 외부 사건을 펜딩(pending)하는 지령의 순차적인 실행을 정지하기 위한 제 2 디버그 스테이트를 갖는 지령을 순차적으로 실행하기 위한 중앙 처리 유니트,

중단점 값과 복수의 중단점 응답 값 중 하나를 수신하기 위한 입력 수단,

상기 중단점 응답 값을 저장하기 위한 상기 입력 수단에 결합된 제어 레지스터,

상기 중단점 값을 저장하기 위한 상기 입력 수단에 결합된 중단점 레지스터,

상기 중단점 레지스터에 결합된 중단점 회로와, 상기 중단점 값이 데이터 버스 상의 값과 메칭될 때 중단점 신호를 발생하기 위한 데이터 버스, 및

상기 제어 레지스터와 중앙 처리 유니트에 결합되고, 상기 중단점 신호를 수신하기 위해 결합된 제어 회로로서, 상기 제어 회로는 상기 중단점 응답 값이 제 1 값일 때 상기 중앙 처리 유니트가 제 1 스테이트에서 동작하게 하고, 상기 제어 회로는 상기 중단점 응답 값이 제 2 값일 때 상기 중앙 처리 유니트가 제 2 스테이트에서 동작하도록 하는 제어 회로를 포함하는 데이터 프로세서.

청구항 2.

데이터 프로세서 상의 디버그 기능을 수행하는 방법에 있어서,

제 1 메모리 위치에 중단점 값을 저장하는 단계,

제 2 메모리 위치에 중단점 응답 값을 저장하는 단계,

현재의 데이터 값을 갖는 중앙 처리 유니트 상의 지령을 순차적으로 실행하는 단계,

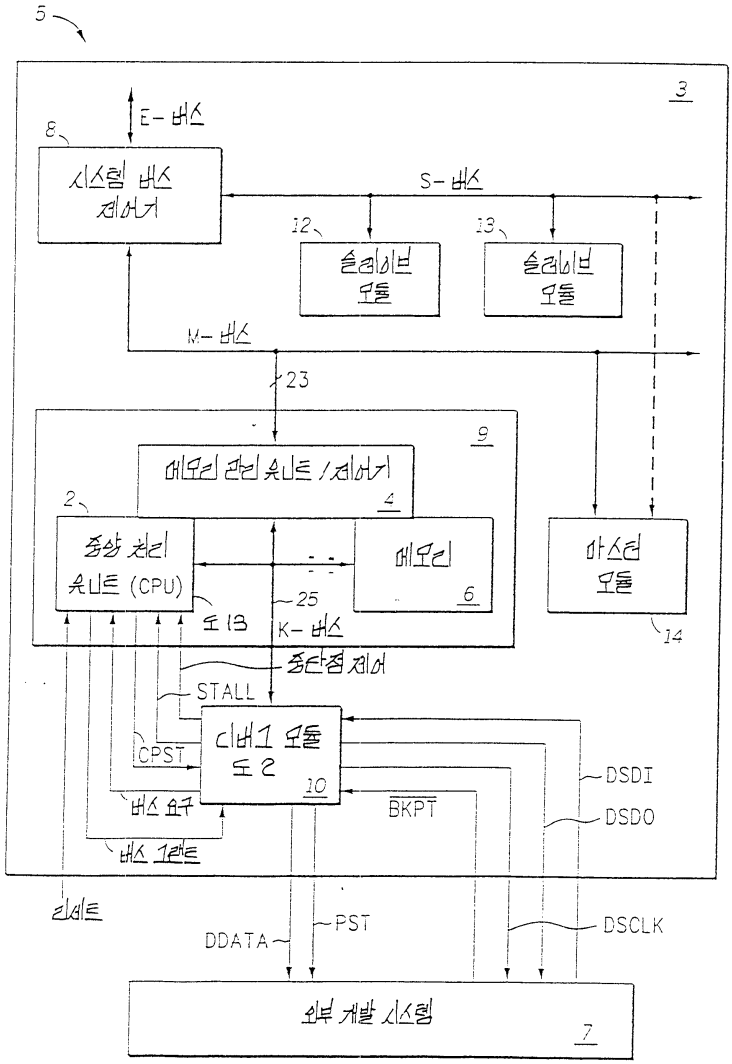
중단점 메칭이 발생했는지를 결정하기 위해 상기 현재 데이터 값을 상기 중단점 값과 비교하는 단계,

상기 중단점 메칭이 발생하고 상기 중단점 응답 값이 제 1 값일 때 상기 중앙 처리 유니트 상의 지령의 실행을 정지하는 단계, 및

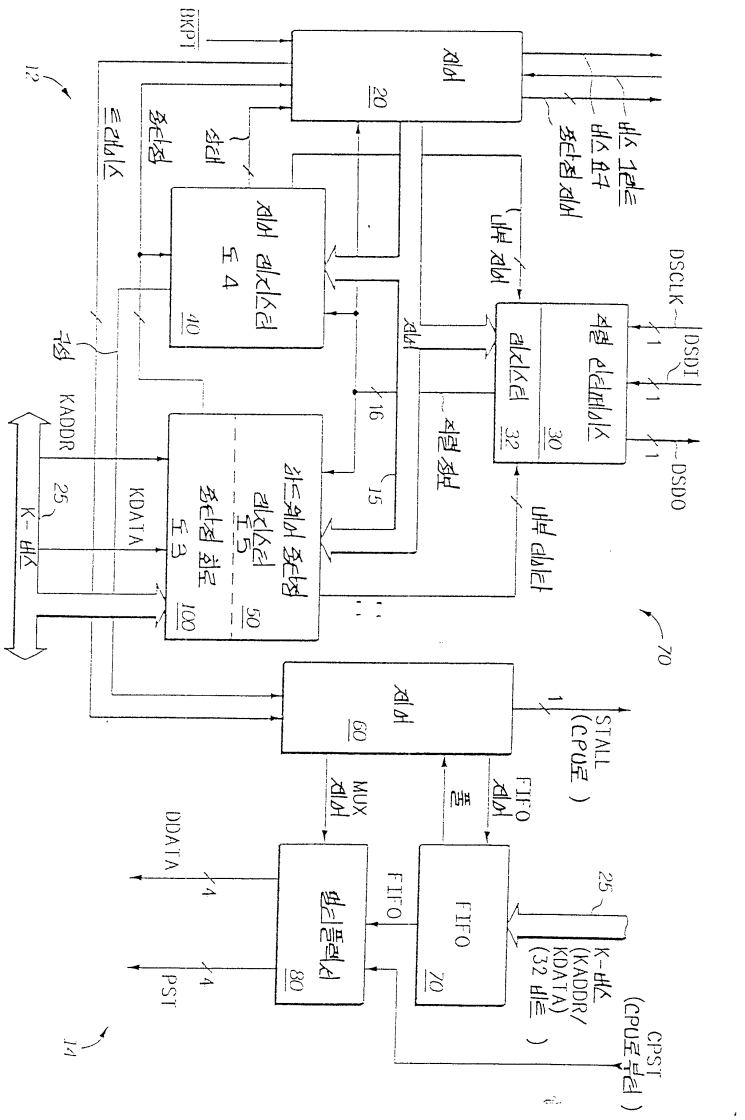
상기 중단점 메칭이 발생하고 상기 중단점 응답 값이 제 2 값일 때 상기 중앙 처리 유니트 상의 중단점 루틴을 실행하는 단계를 포함하는 디버그 기능 수행 방법.

도면

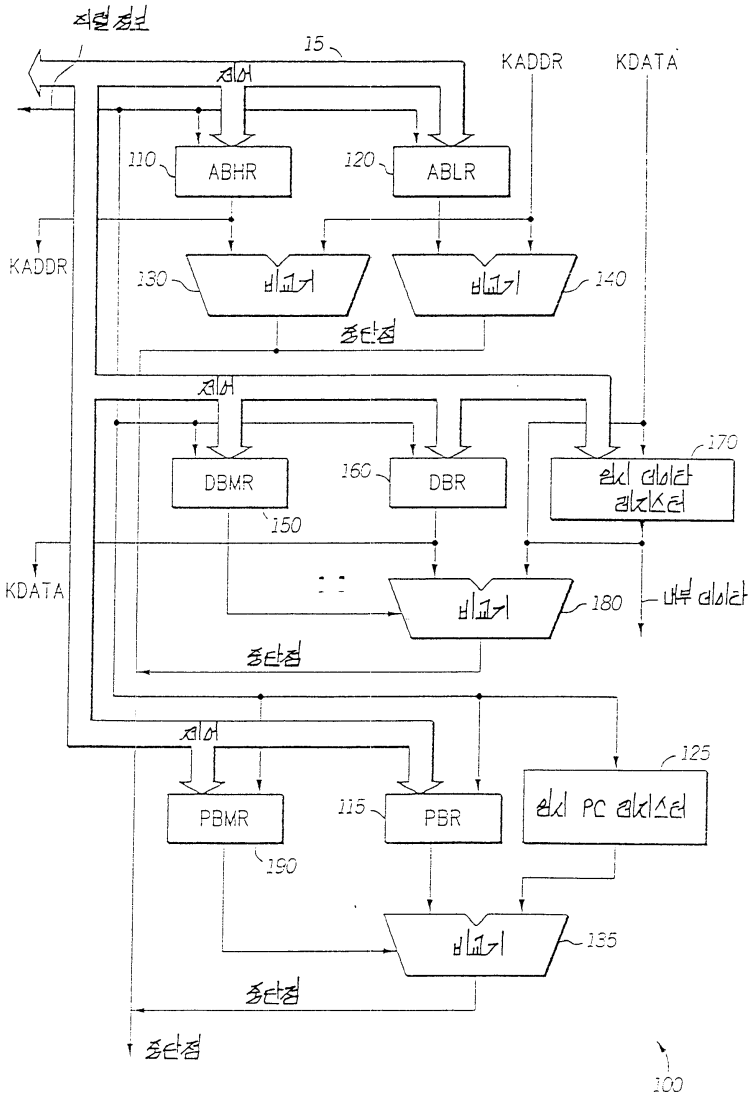
도면 1



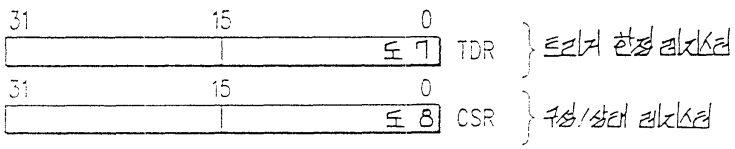
도면 2


BKPT
제어
20
12
트레이스
중단점
상태
제어 레지스터
도 4
40
구성
KADDR
25
K-버스
KDATA
하드웨어 중단점
레지스터
도 5
중단점 회로
도 3
50
100
16
15
제어
직렬 정보
버퍼 제어
직렬 인터페이스
레지스터
30
32
DSCLK
DSDI
DSDO
버퍼 데이타
70
제어
60
STALL
(CPU로)
MUX
제어
FIFO
제어
풀
DDATA
PST
멀티플렉서
80
FIFO
70
K-버스
(KADDR/
KDATA)
(32 비트)
CPST
(CPU로부터)
14

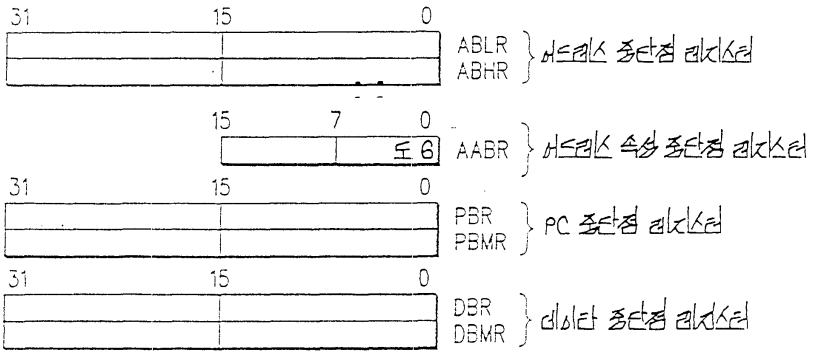
도면 3


처리정보
15
제어
KADDR
KDATA
110
ABHR
120
ABLR
KADDR
130
비교기
비교기
140
중단점
제어
DBMR
160
DBR
170
현재 데이타 레지스터
150
KDATA
비교기
180
내부 데이타
중단점
제어
PBMR
115
PBR
125
현재 PC 레지스터
190
비교기
135
중단점
중단점
100

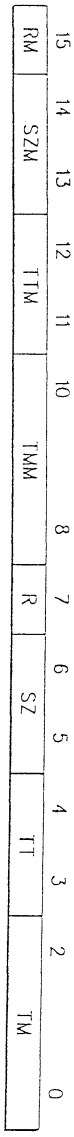
도면 4


31
15
0
도 7
TDR
트리거 한정 레지스터
31
15
0
도 8
CSR
구성/상태 레지스터

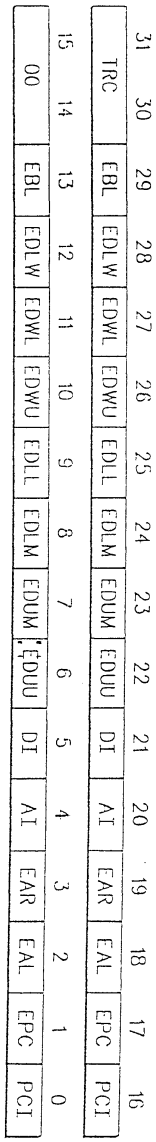
도면 5


31
15
0
ABLR
ABHR
어드레스 중단점 레지스터
15
7
0
도 6
AABR
어드레스 속성 중단점 레지스터
31
15
0
PBR
PBMR
PC 중단점 레지스터
31
15
0
DBR
DBMR
데이타 중단점 레지스터

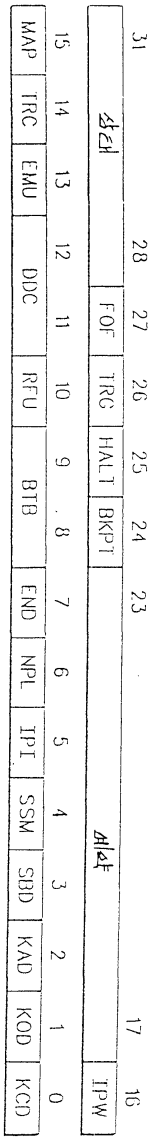
도면 6



도면 7


31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
TRC EBL EDLW EDWL EDWU EDLL EDLM EDUM EDUU DI AI EAR EAL EPC PCI
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
00 EBL EDLW EDWL EDWU EDLL EDLM EDUM EDUU DI AI EAR EAL EPC PCI

도면 8


31
28
27
26
25
24
23
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
MAP
TRC
EMU
DDC
RFU
BTB
END
NPL
IPI
SSM
SBD
KAD
KOD
KCD
IPW
BKPT
HALT
TRC
FOF

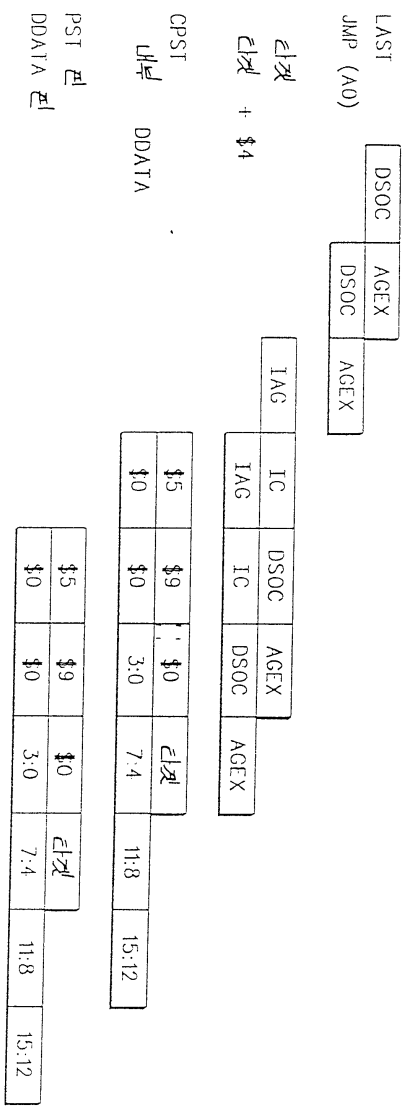
도면 9

DDATA[3:0], CSR[31:28]	중단점 상태
$0	버퍼한 중단점도 가능하지 않음
$1	레벨 1 중단점을 기다림
$2	레벨 1 중단점이 트리거됨
$3-4	예약
$5	레벨 2 중단점을 기다림
$6	레벨 2 중단점이 트리거됨
$7-$F	예약

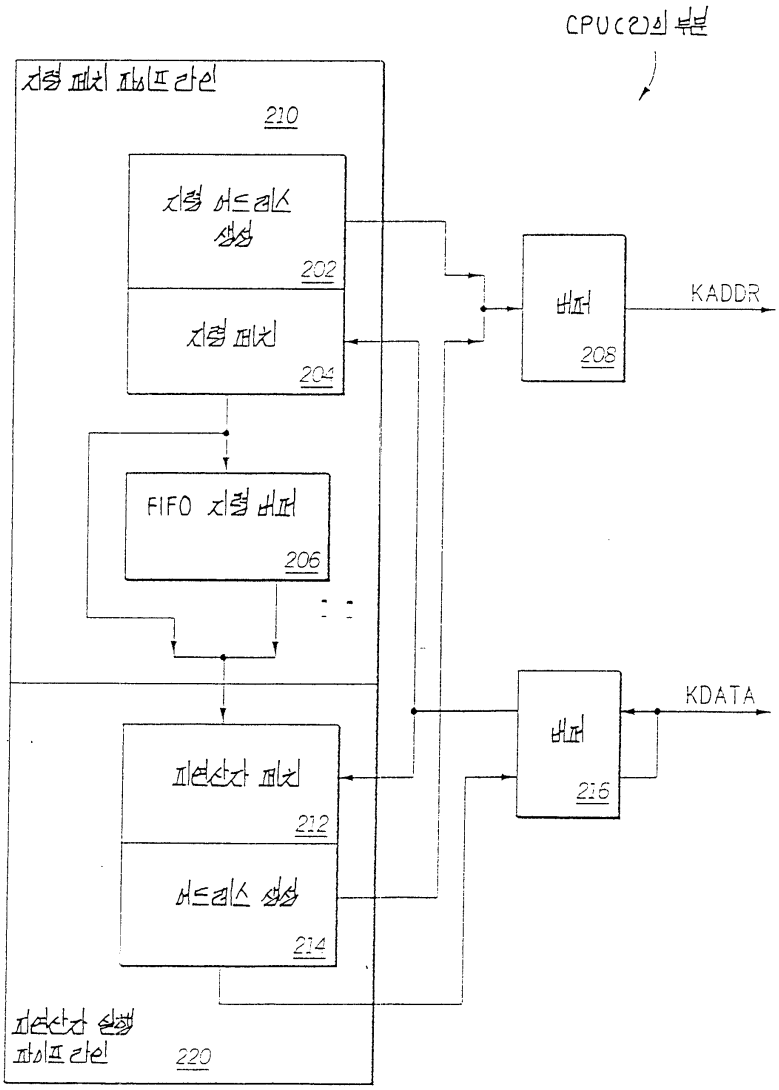
도면 10

PST[3:0]	정의
0000	확장 지속
0001	지령 실행 개시
0010	예약
0011	사용자 모드로 전입
0100	펄스 지령 실행 개시
0101	체제된 브랜치 실행 개시
0110	예약
0111	RTE 지령 실행 개시
1000	데이타의 1 바이트 전송 개시
1001	데이타의 2 바이트 전송 개시
1010	데이타의 3 바이트 전송 개시
1011	데이타의 4 바이트 전송 개시
1100	예외처리
1101	예물레이팅 모드 전입 예외처리
1110	프로세서는 정지하고, 리셋팁트를 기다린다
1111	프로세서 정지

도면 11


LAST
JMP (A0)
DSOC
AGEX
DSOC
AGEX
타겟
타겟 + $4
IAG
IC
DSOC
AGEX
IAG
IC
DSOC
AGEX
CPST
내부 DDATA
$5
$0
$9
$0
$0
3:0
타겟
7:4
11:8
15:12
PST 핀
DDATA 핀
$5
$0
$9
$0
$0
3:0
타겟
7:4
11:8
15:12

도면 12


CPU(C2)의 부분
지령 패치 파이프라인
210
지령 어드레스 생성
202
지령 패치
204
FIFO 지령 버퍼
206
버퍼
208
KADDR
피연산자 패치
212
어드레스 생성
214
버퍼
216
KDATA
피연산자 실행 파이프라인
220