(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.[6]
G06F 17/00

(45) 공고일자 2003년08월25일
(11) 등록번호 10-0387193
(24) 등록일자 2003년05월29일

(21) 출원번호 10-1996-0036639
(22) 출원일자 1996년08월30일

(65) 공개번호 특1997-0012203
(43) 공개일자 1997년03월29일

(30) 우선권주장 520,945 1995년08월30일 미국(US)

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

(72) 발명자 죠셉 씨. 써셀로
미국, 아리조나 85023-5230, 피닉스 웨스트 콘라드 드라이브 44

클라우스 알. 리델
미국, 텍사스 78759, 오스틴 하니써클 트레일 8836

(74) 대리인 이병호

심사관 : 송대종

(54) 트레이스함수와그에따른방법을실행하기위한데이타처리시스템

요약

데이타 처리장치(3)는 외부개발시스템(7)이 내부 버스의 유형 또는 사용 가능성을 가정하지 않고서 그리고 데이타 처리장치(3)의 효율과 속도에 심히 영향을 끼치지 않고서 데이타 처리장치(3)의 내부 연산을 동적으로 관찰하도록 허용하는 실시간 트레이스 함수를 실행한다. 데이타 처리장치(3)의 디버그 모듈(10)은, DDATA 신호와 PST 신호를 매개로하여 내부 연산 정보를 제공하기 위한 병렬 출력 포트(port)를 제공한다. 상기 DDATA 신호는 오퍼랜드 값을 반영하는 데이타를 제공하고 상기 PST 신호는 중앙처리장치(2)의 실행상태를 반영하는 코드화된 상태 정보를 제공한다. 더욱이, 상기 DDATA 신호는 또한 포착된 명령 어드레스 프로그램 흐름 변화를 제공하여 외부개발 시스템(7)이 외부에서 가시가능한 어드레스 버스 또는 외부에서 가시가능한 데이타 버스를 필요치 않고서도 정확한 프로그램 흐름을 추적하게 한다.

대표도

도 1

명세서

도면의 간단한 설명

도 1은 본 발명의 한 실시예에 따른 데이타 처리 시스템을 나타내는 블럭도.
도 2는 도 1의 데이타 처리시스템의 디버그 모듈의 일부를 나타내는 블럭도.
도 3은 도 2의 디버그 모듈의 브레이크포인트 회로를 나타내는 블럭도.
도 4는 도 2의 디버그 모듈의 다수의 제어 레지스터를 나타내는 블럭도.

도 5는 도 2의 다수의 하드웨어 브레이크포인트 레지스터를 나타내는 블럭도.
도 6은 도 5의 다수의 하드웨어 브레이크포인트 레지스터의 어드레싱 속성 브레이크포인트 레지스터를 나타내는 블럭도.
도 7은 도 4의 다수의 제어 레지스터의 트리거 정의 레지스터를 나타내는 블럭도.
도 8은 도 4의 다수의 제어 레지스터의 구성 상태 레지스터를 나타내는 블럭도.
도 9는 도 2의 디버그 모듈의 다수의 브레이크포인트 신호의 출력을 나타내는 표.
도 10은 도 2의 디버그 모듈의 다수의 처리장치 상태신호의 출력을 나타내는 표.
도 11은 본 발명의 한 실시에에 따른 파이프라인 타이밍 예를 나타내는 도표.
도 12는 도 1의 데이타 처리 시스템의 CPU의 일부를 나타내는 블럭도.

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 일반적으로 데이타 처리 시스템에 관한 것이고, 더 상세히 설명하면 데이타 처리 시스템에서 디버그(debug) 함수를 실행하기 위한 방법과 장치에 관한 것이다.
데이타 처리 시스템이 의도한 대로 작동하지 않는 경우, 장애의 원인을 식별하기 위해 다양한 분석 기술이 동시 사용될 수 있다. 통상, 트레이스(trac역e) 함수와 브레이크포인트(breakpoint) 함수는 데이타 처리 시스템내에서 실현되어, 장애를 일으킨 회로를 분리시키고 장애를 일으킨 소프트웨어 프로그램을 수정하도록 도와준다.
트레이스 함수는 외부의 사용자가 데이타 처리 동작의 중간 실행결과를 관찰하도록 해주는 수단을 제공해준다. 트레이스 함수는 통상 각각의 명령 또는 소프트 프로그램 명령중의 일단의 예정된 명령이 데이타 처리 시스템에 의해 실행된후, 데이타 처리 시스템내에 포함된 선택된 레지스터 및 기억장치의 상태(status)를 제공한다. 상기 선택된 레지스터 및 기억장치의 상태를 나타냄으로서 상기 트레이스 함수는 외부의 사용자에게 데이타 처리 장치 또는 데이타 처리 시스템내의 프로그래 밍 모델에 관한 매우 상세한 정보를 제공한다. 상기 정보로 인해 많은 유형의 에러가 식별되고 계속 수정될 수가 있다. 또한 브레이크포인트 함수는 데이타 처리 시스템내의 잘못된 소프트웨어 코드 또는 결함있는 회로를 식별하고 분리하기 위한 방법을 제공한다. 브레이크포인트 함수부는 사실상, 소프트웨어 프로그램의 정지 원인이 되는 사전에 프로그래밍된 사상(event)이 발생하는 곳이다. 그때 데이타는 검색되어 소프트웨어 프로그램의 상태를 결정한다. 상기 트레이스 함수와 마찬가지로 상기 브레이크포인트 함수는 데이타 처리상의 에러가 식별될 수 있도록 외부의 사용자가 선택된 레지스터 및 기억장치 각각의 상태를 확인하게 해준다.
상기 트레이스 함수 및 브레이크포인트 함수 둘 모두는 이전에 기술된 분리 및 식별기능을 제공하기 위해 현재 수정가능한 데이타 처리 시스템에 통합되어 있다. 예를들어 캘리포니아 산타 클라라에 소재한 인텔사로 부터 입수가능한 i960Jx 마이크로프로세서는, 트레이스 사상과 트레이스 장애를 생성하는 내부 브레이크포인트 레지스터를 포함하는 있는, 집적 회로 마이크로프로세서 이다. 상기 i960Jx 마이크로프로세서에서, 내부 브레이크포인트 레지스터는, 명령 실행 어드레스 또는 여러 유형의 데이타 액세스의 어드레스에 대한 트래핑(trapping)에 전용된다. 상기 i960Jx 마이크로프로세서의 트레싱 설비를 사용하기 위해, 상기 마이크로프로세서를 이용하는 소프트웨어는 고장처리 과정 또는 디버깅(debugging) 모니터 프로그램과의 인터페이스를 제공할 필요가 있다. 또한 소프트웨어는, 다양한 트레이싱 모드를 사용가능하게 하고 트레이싱 함수를 선택적으로 사용가능하게 하거나 사용금지 하게 하기 위해, 여러 레지스터 및 제어 비트를 조작하는 것이 필요해 진다.
유사하게, 캘리포니아의 산타 클라라에 소재한 인텔사로부터 입수가능한 80486 마이크로프로세서는 내부 브레이크포인트 레지스터를 포함하며 트레이스 함수를 실행할 수 있는 집적회로 마이크로프로세서이다. 상기 80486 마이크로프로세서에서 브레이크포인트 명령은 내부 소프트웨어 디버거 장치에 의해서 사용하기 위해 실현되고, 여기서 내부 소프트웨어는 데이타 처리 시스템상의 소프트웨어 실행을 참조한다. 통상의 동작중에 소프트웨어 디버그 프로그램은, 실행된 소프트웨어 코드의 유형에 의존하는 모든 소요되는 브레이크포인트에서, 브레이크포인트 명령을 실현할 수 있다. 상기 80486 마이크로 프로세서는 또한 트랩 루틴이 실행된 후 인터럽트(interrupt)를 실행하는 단하나의 스텝 트랩 기능을 실현하기 위한 메카니즘을 제공한다.
상기 i960Jx 마이크로프로세서와 상기 80486 마이크로프로세서는 브레이크포인트 및 트레이스 함수를 통합했는데, 상기 마이크로프로세서 디버그 기능은 정규의 작동을 보류하고 특정한 디버그 예외(exception) 루틴속으로 들어가는 마이크로프로세서를 필요로한다. 상기 디버그 예외루틴 중에 상기 마이크로프로세서는 정규의 작동중에서와 같은 작동을 하지 않아, 회로 또는 소프트웨어의 고장이 유사한 방법으로 발생하지 않는다. 따라서 외부 사용자는 실시간 고장조건이 실제로 발생하지 않는 경우 마이크로프로세서 둘중 어느 하나에서 장애의 근원을 검증 및 분리할 수 없을 것이다.
더욱이 상기 i960Jx 및 80486 마이크로프로세서는, 브레이크포인트 및 트레이스 함수가 이루어지는 동안 프로그램 제어를 하기 위해 마이크로프로세서에 의해 실행되는 내부 소프트웨어 디버그 프로그램이 필요하다. 상기 디버그 제어를 사용하면 상당한 량의 오버헤드를 필요로 하는 몹시 강제적인 디버깅 방식이 제공된다. 뿐만아니라, 디버그 소프트웨어를 사용하면 고장난 마이크로프로세서에서 결함있는 회로 또는 시퀀스를 진정으로 반영할 수 없다.

발명이 이루고자 하는 기술적 과제

따라서, 실시간 트레이스 및 실시간 디버그 함수 둘다를 제공해주는 데이타 처리장치의 필요성이 존재한다. 만일, 트레이스 및 디버그 함수가 특정한 디버그 작동방식이 아닌 정규의 작동중에 데이타 처리 장치 작동을 반영하는 방법으로 실행되는 경우에는, 외부 사용자는 데이타 프로세서의 회로와 소프트웨어내의 장애를 더욱 실제적으로 식별하고 수정할 수가 있다.

발명의 구성 및 작용

다음의 토론에서, 본 발명을 실현하는 데이타 처리 시스템의 한 실시예에 관한 상호 접속과 동작은 매우 상세히 제공될 것이다.
본 발명의 상호접속에 대한 아래의 기술에서, '버스(bus)' 라는 용어는, 데이타, 어드레스, 제어, 또는 상태(status)와 같은 하나 또는 그 이상의 다양한 형태의 정보를 전송하고자 사용될 수 있는 다수의 신호 또는 전도체를 언급하기 위해 사용할 것이다. '표명하다(assert)' 및 '부정하다(negate)'라는 용어는, 신호, 상태(status) 비트, 또는 유사한 장치를 논리적인 참(true) 또는 논리적인 거짓(false)의 상태로 각각 만드는것을 언급하는 경우에 사용될 것이다. 만일 논리적 참의 상태가 논리 레벨 1 이라면 논리적 거짓의 상태는 논리 레벨 0 이 될 것이 다.
더욱이, 숫자앞의 부호 '$'는, 상기 숫자가 16진법 또는 16진법 형식으로 표현된다는 것을 타나내준다. 숫자앞의 부호 '%'는 상기 숫자가 2진법 또는 2진법 형식으로 표현된다는 것을 타나내준다.
이제 도 1에 관해 언급해 보면, 도 1은 본 발명의 하나의 실시예에 따른 데이타 처리 시스템(5)을 나타낸다. 데이타 처리 시스템(5)은 데이타 처리장치(3)와 외부개발시스템(7)을 포함한다. 데이타 처리장치(3)는 시스템 버스제어장치(8), 코어(9), 종속 모듈(12), 종속 모듈(13), 주 모듈(14), 디버그 모듈(10)을 포함한다. 시스템 버스 제어장치(8)는 E-버스를 경유하여 여기에서는 나타나지 않은 외부 장치에 결합된다. 시스템 버스 제어장치(8)는 S-버스를 경유하여 종속 뮬듈(12)과 종속 모듈(13)에 결합된다. 상기 S-버스는 종속 모듈 버스이다. 시스템 버스 제어장치(8)는 M-버스(23)를 경유하여 코어(9)와 주 모듈(14)에 결합된다. 상기 M-버스(23)는 주 모듈 버스이다.
코어(9)는 CPU(2), 기억관리장치(MMU)/제어장치(4), 메모리(6)를 포함한다. CPU(2), MMU/제어장치(4), 메모리(6), 디버그 모듈(10)은 각각 K-버스(25)를 경유하여 서로에 결합된다. 또한 CPU(2)와 메모리(6) 둘 모두는 MMU/제어장치(4)에 직접 접속된다. MMU/제4장치(4)는 M-버스(23)을 경유하여 데이타 처리 시스템(5)의 잔여부분에 정보를 제공한다. CPU(2)는 디버그 모듈(10)에 결합되어 중앙처리장치상태(CPST) 신호를 제공하며, 기능정지(Stall) 신호를 수신한다. CPU(2)는 또한 외부장치로부터 리셋(Reset) 신호를 수신한다. 디버그 모듈(10)은 외부의 사용자에게 처리장치상태(PST) 신호, 디버그 데이타(DDATA) 신호, 개발 직렬 데이타 출력(DSDO: development serial data out) 신호를 제공한다. 디버그 모듈(10)은 개발 직렬 데이타 입력(DSDI: development serial data in) 신호, 개발 직렬 클록(DSCLK: development serial clock), 브레이크포인트(BKPT) 신호를 데이타 처리 시스템(5)의 외부에 있는 여기에서 도시되지 않은 장치로 부터 수신한다.
도 2는 디버그 모듈(10)을 매우 상세히 나타낸다. 디버그 모듈(10)은 제어회로(20), 직렬 인터페이스(30), 레지스터(32), 다수의 제어레지스터(40), 다수의 하드웨어 브레이크포린트 레지스터(50), 제어회로(60), 선입선출(FIFO: First-in First-out) 장치(70), 다수의 다중화 장치(80), 브레이크포인트 회로(100)를 포함한다.
브레이크포인트(BKPT) 신호는 도 2에서 상세히 나타나 있지 않은 외부의 집적회로 핀을 경유하여 제어회로(20)에 제공된다. 제어회로(20)는 다수의 제어 레지스터(40)에 결합되어 다수의 비트를 포함하는 상태신호(Status signal)를 수신한다. 제어회로(20)는 브레이크포인트 회로(100)에 결합되어 다수의 비트를 포함하는 브레이크포인트 신호를 수신한다. 제어회로(20)는 제어회로(60)에 결합되어 다수의 비트를 포함하는 트레이스 신호를 제공한다. 제어회로(20)는 제어버스(15)를 경유하여 다수의 제어 레지스터(40), 다수의 하드웨어 브레이크포인트 레지스터(50), 레지스터(32)에 결합된다. 제어회로(20)는 다수이 브레이크포인트 제어신호와 버스 요구(Bus Request) 신호를 CPU(2)에 제공한다. 제어회로(20)는 또한 CPU(2)에 의해 제공된 버스 허가(Bus Grant)신호를 수신한다.
다수의 제어 레지스터(40)는 다중 비트(multi-bit) 직렬 정보를 레지스터(32)로 부터 수신한다. 다수의 제어 레지스터(40)는 다수의 비트를 포함하는 구성(configuration) 신호를 제어회로(60)에 제공한다. 브레이크포인트 회로(100)는 KADDR(K 어드레스) 신호와 KDATA(K 데이타) 신호를 K-버스(25)로 부터 수신한다. 또한 다수의 하드웨어 브레이크포인트 레지스터(50)는 다수의 비트를 포함하는 내부 데이타 신호를 레지스터(32)에 제공한다. 또한 K-버스(25)는 다수의 하드웨어 브레이크 포인트 레지스터(50)에 제공된다. K-버스(25)는 K 어드레스 신호, K 데이타 신호, K 제어(K CONTROL) 신호의 수집이라는 것을 주목하라. 상기 K 어드레스, K 데이타 및 K 제어신호 각각은 브레이크포인트 회로(100)에 입력된다.
직렬 인터페이스(30)는 상기 DSDI 및 DSCLK 신호를 제 1 및 제 2 직접회로 핀으로부터 각각 수신한다. 직렬 인터페이스(30)는 상기 DSDO 신호를 제 3 집적회로 핀에 제공한다. 직렬 인터페이스(30)는 레지스터(32)에 결합된다. 레지스터(32)는 다수의 제어 레지스터(40)와 다수의 하드웨어 브레이크포인트 레지스터(50)에 결합되어 다중 비트 직렬 정보신호를 제공한다.
제어회로(60)는 기능정지(Stalle) 신호를 CPU(2)에 제공한다. 제어회로(60)는 또한 선입선출(FIFO: first-in first-out) 제어신호를 선입선출 장치(70)에, 그리고 시스템 다중화(Mux: system multiplex) 제어신호를 다수의 다중화 장치(80)에 제공한다. 선입선출 장치(70)는 K-버스(25)에 결합되어 K 어드레스 및 K 데이타 신호를 수신한다. 또한 선입선출 장치(70)는 다수의 다중화 장치(80)에 결합되어 선입선출 신호를 제공한다. 선입선출 장치(70)는 또한 전(Full) 신호를 제어회로(60) 에 제공한다. 다수의 다중화 장치(80)는 중앙처리 장치 상태(CPST) 신호를 CPU(2)로 부터

수신한다. 다수의 다중화 장치(80)는 DDATA 및 PST신호를 제공한다.
제어회로(20)는 직렬 인터페이스(30)를 작동시키는데 필요하며 레지스터(32)에서 정보의 기억을 제어하는데 필요한 신호를 제공한다. 일단 레지스터(32)에 데이타가 로드되면, 제어회로(20)는 제어회로(40) 또는 다수의 하드웨어 브레이크포인트 레지스터(50)내의 적절한 행선(destination) 속으로 데이타의 전송을 인도한다. 제어회로(20)에 의해 실행된 제어함수는 제어 버스(15)에 의해 전달된다. 로드 작동의 실행을 제어하는 것 외에 또, 제어회로(20)는 제어회로(40)로부터 상태(Status) 정보를 수신하고, 브레이크포인트 회로(100)으로부터 브레이크포인트 정보를 수신한다. 제어회로(20)는 상태정보 및 브레이크포인트 정보를 함께 사용하여 버스 요구신호, 브레이크포인트 제어신호, 트레이스 신호를 발생시킨다.
도 3은 브레이크포인트 회로(100)을 매우 상세히 나타낸다. 브레이크포인트 회로(100)는 어드레스 브레이크포인트 상위 레지스터(ABHR, 110), 어드레스 브레이크포인트 하위 레지스터(ABLR, 110), 비교기(130), 비교기(140), 데이타 브레이크포인트 마스크 레지스터(DBMR, 150), 데이타 브레이크포인트 레지스터(DBR, 160), 임시 데이타 레지스터(170), 비교기(180), 프로그램 카운터 브레이크포인트 마스크 레지스터(PBMR, 190), 프로그램 카운터 브레이크포인트 레지스터(PBR, 115), 임시 프로그램 카운터(PC) 레지스터(125), 비교기(135)를 포함한다.
제어 버스(15)는 상기 ABHR(110), ABLR(120), DBMR(150), DBR(160), PBMR(190), PBR(115)의 각각의 입력부에 결합된다. 직렬 정보신호는 상기 ABHR(110), ABLR(120), DBMR(150), DBR(160)의 각각의 입력부에 결합된다. K 제어신호는 상기 PBMR(190), PBR(115), 임시 PC 레지스터(125)의 각각의 입력부에 결합된다. ABHR(110)의 제 1 출력부는 K 어드레스신호에 결합된다. 제어 버스(15)는 또한 임시 레지스터(170, 125)에 결합된다. 직렬 정보신호는 비교기(130)의 제 1출력부에 결합된다. K 어드레스 신호는 제 2 입력부를 비교기(130)에 제공한다. K 어드레스 신호는 제 2 입력부를 비교기(130)에 제공한다. ABLR(120)은 제 1 입력부를 비교기(140)에 제공하고, K 어드레스 신호는 제 2 입력부를 비교기(140)에 제공한다. 비교기(130)의 출력부와 비교기(140)의 출력부는 둘다 브레이크포인트 신호 전도체에 결합된다.
DBMR(150)의 출력부는 비교기(180)의 가능 입력부에 결합된다. DBR(160)의 출력부는 비교기(180)의 제 1 입력부에 결합된다. K 데이타 신호는 임시 데이타 레지스터(170)의 입력부에 결합된다. 임시 데이타 레지스터(170)의 출력부는 비교기(180)의 제 2 입력부와 내부 데이타 신호에 결합된다. 비교기(180)의 출력부는 브레이크포인트 신호 전도체에 결합된다.
PBMR(190)의 출력부는 비교기(135)의 가능 입력부에 결합된다. PBR(115)의 출력부는 비교기(135)의 제 1 입력부에 결합된다. K 제어신호는 임시 프로그램 카운터 레지스터(125)의 입력부에 결합된다. 임시 PC 레지스터(135)의 출력부는 비교기(135)의 제 2 입력부에 결합된다. 비교기(135)의 출력부는 브레이크포인트 신호 전도체에 결합된다.
도 12는 CPU(2)의 일부분을 매우 상세히 나타낸다. CPU(2)의 상기 부분은 명 령 인출 파이프라인(IFP, 210)과 오퍼랜드(operand) 실행 파이프라인(OEP, 220)을 포함한다. IFP는 명령 어드레스 생성 회로(202), 명령 인출 회로(204), 선입선출(FIFO) 명령 버퍼(206)을 포함한다. OEP(220)는 오퍼랜드 인출 회로(212)와 어드레스 생성 회로(214)를 포함한다.
명령 어드레스 생성 회로(202)의 출력부는 버퍼(208)의 제 1 입력부에 결합된다. 명령 인출 회로(204)의 출력부는 선입선출 명령 버퍼(206)와 오퍼랜드 인출 회로(212) 둘다에 결합된다. 선입선출 명령 버퍼(206)의 출력부는 오퍼랜드 인출 회로(212)에 결합된다. 어드레스 생성 회로(214)의 제 1 출력부는 버퍼(208)의 제 2 입력부에 결합된다. 어드레스 생성회로(214)의 제 2 출력부는 버퍼(216)에 결합된다. 버퍼(216)의 제 1 출력부는 오퍼랜드 인출 회로(212)와 명령 인출 회로(204)에 결합된다. 버퍼(216)의 제 2 출력부는 K 데이타 신호를 제공한다. K 데이타 신호는 또한 버퍼(216)에 제공된다. 버퍼(208)의 출력부는 K 어드레스 신호를 제공한다.

레지스터에 대한 설명

도 4는 다수의 제어 레지스터(40)의 일부를 매우 상세히 나타낸다. 본 발명의 하나의 실시예에서, 다수의 제어 레지스터(40)의 일부는 트리거 정의 레지스터(TDR)와 구성/상태 레지스터(CSR)를 포함한다.
상기 TDR은 디버그 모들(10)의 구성을 기억하고, 정의된 조건하에 실행된 선택된 함수에 대해 제어를 한다. 본 발명의 실시예에서, 트리거는 1 또는 2 레벨 트리거로 구성될 수 있고 여기서 비트(31 부터 16)는 제 2 레벨 트리거를 정의하고, 비트(15 부터 0)는 제 1레벨 트리거를 정의한다. 트리거가 2레벨 트리거로 구성된 경우, 트리거 함수는 조건 구성체와 유사한 방법으로 작동한다. 예를들어 트리거의 제 1조건 또는 제 1레벨이 만족되면, 트리거가 브레이크포인트 함수를 데이타 처리시스템(5)에 의해 실행되게 하기전에 제 2 조건 또는 제 2레벨은 만족되어야 한다. 다른 방법으로 표현하면;
만일 제 1조건이 만족되고 그후 제 2 조건이 만조되면 트리거 된다.
TDR에 포함된 비트 각각의 기능은 차후에 매우 상세히 논의될 것이다.
TDR은 도 7에서 매우 상세히 나타나 있다. 상기 TDR의 PCI 비트는 프로그램 카운터 브레이크포인트 역 비트 이다. 만일 상기 PCI 비트가 표명되면, 프로그램 카운터 브레이크포인트는 상기 프로그램 카운터 브레이크포인트 레지스터(PBR) 및 프로그램 카운터 브레이크포인트 마스크 레지스터(PBMR)에 의해 정의된 범위의 외부에서 개시된다. 만일 상기 PCI 비트가 부정되면, 프로그램 카운터 브레이크포인트 상기 PBR과 PBMR에 의해 정의된 영역내에서 정의된다.
가능 프로그램 카운터 브레이크포인트(EPC: Enable Program Counter Breakpoint) 비트는, 상기가 표명되는 경우에, 선택적으로 상기 프로그램 카운터 브레이크포인트, PBR 그리고 PBMR에 대한 트리거의 의존을 가능케 한다. 상기 EPC 비트가 부정된 경우, 상기 프로그램 카운터 브레이크포인트는 불능하게 된다.
가능 어드레스 브레이크포인트 하위(EAL: Enable Address Breakpoint Low) 비트는, 표명된 경우에, 다수의 하드웨어 브레이크포인트 레지스터(50)의 어드레스 브레이크포인트 레지스터의 ABLR에서 기억된 어드레스가 브레이크포

인트 트리거로 기능하는 것을 가능케 한다. 따라서, 만일 상기 EAL 비트가 표명된다면, KADDR 신호를 경유하여 전송된 어드레스 값이 어드레스 브레이크포인트 레지스터의 ABLR에서 기억된 어드레스 값과 같은 경우에 브레이크포인트는 트리거한다.
가능 어드레스 브레이크포인트(EAR: Enable Address Breakpoint Range) 비트는, 비트가 표명된 경우, 다수의 하드웨어 브레이크포인트 레지스터(50)의 어드레스 브레이크포인트 레지스터의 ABLR과 ABHR에 의해 정의된 포함적 범위상에 브레이크 포인트가 트리거 하는 것을 가능하게 한다. 따라서, 만일 상기 EAR 비트가 표명 된다면 K 어드레스 신호를 경유하여 전송된 어드레스 값이 어드렝스 브레이크포인트 레지스터의 ABLR과 ABHR에 기억된 어드레스 값에 의해 정의된 범위내인 경우 브레이크포인트는 트리거한다.
EAI(Enable Address Breakpoint Inverted) 비트는, 상기 EAI가 표명된 경우, 다수의 하드웨어 브레이크포인트 레지스터(50)의 어드레스 브레이크포인트 레지스터의 ABLR과 ABHR에 의해 정의된 범위를 제외하거나 또는 외부에 있는 어드레스 범위상에 브레이크포인트가 트리거 하는 것을 가능하게 한다. 따라서 상기 EAI 비트가 표명된 경우, 브레이크포인트는, KADDR 신호를 경우하여 전송된 어드레스 값이 어드레스 브레이크포인트 레지스터의 ABLR에 기억된 어드레스 값 보다 적거나 또한 어드레스 브레이크포인트 레지스터의 ABHR에 기억된 어드레스 값 보다 큰 경우에, 트리거 한다.
DI(Data Breakpoint Invert) 비트는, 상기 DI 비트가 표명된 경우, 다수의 데이타 브레이크포인트 비교기의 논리적 센스를 반전시킨다. 상기 DI 비트는 다수 의 하드웨어 브레이크포인트 레지스터(50)의 DBR 속으로 프로그램된 데이타 값과 동등하지 않은 데이타 값의 출력에 근거한 트리거를 발생시키고자 사용될 수 있다.
EDUU(Enable Data Breakpoint for the Upper Upper Data Byte) 비트는 상기 EDUU 비트가 표명된 경우, KDATA 신호 및 K-버스(25)를 경유하여 전송된 데이타 값의 비트(31 부터 24)에 근거한 데이타 브레이크포인트 트리거를 가능하게 한다. 유사하게, EDMU(Enable Data Breakpoint for the Upper Middle Data Byte) 비트는, 상기 EDLM(Enable Data Breakpoint for the Lower Middle Data Byte) 비트는, 상기 EDLM 비트가 표명된 경우, KDATA 신호 및 K-버스(25)를 경유하여 전송된 데이타 값의 비트(15 부터 8)에 근거한 데이타 브레이크포인트 트리거를 가능하게 한다. 또한, EDLL(Enable Data Breakpoint for the Lower Lower Data Byte) 비트는, 상기 EDLL 비트가 표명된 경우, KDATA 신호 및 K-버스(25)를 경유하여 전송된 데이타 값의 비트(31 부터 0)에 근거한 데이타 브레이크포인트 트리거를 가능하게 한다. 상기 EDUU, EDUM, EDLM, EDLL, EDWU, EDWL, EDLW 비트는 어느것이라도 데이타 브레이크포인트 트리거를 가능하게 한다는 것을 주목해야 한다. 통상 요구되는 바와 같이 소정의 조건은 데이타 브레이크포인트가 실제 트리거되기 전에 발생해야 한다. 만일 EDUU, EDUM, EDLM, EDLL, EDWU, EDWL, EDLW 비트의 각각이 부정되면, 상기 데이타 브레이크포인트 트리거는 본 발명의 한 실시예에서 불능하게 된다.
게다가, 본 발명의 본 실시예에서 데이타 브레이크포인트 트리거는 잘못 정렬되거나 정렬되지 않은 오퍼랜드 참조상에서 발생될 수 있다는 점을 주목해야 한다. 잘못 정렬되거나 정렬되지 않은 오퍼랜드 참조에서의 상기 브레이크포인트의 발생은 다음과 같은 방법을 사용한다. 바이트-크기 참조에 대해, 적절한 한 세트의 데이타 비트는 오퍼랜드 어드레스의 두개의 하위 배열 비트의 함수로 프로그램 될 것이다. 다음의 예를 보아라.
KADDR[1:0]=00 이면, KDATA[31:24]는 프로그램 된다.
KADDR[1:0]=01 이면, KDATA[23:16]는 프로그램 된다.
KADDR[1:0]=10 이면, KDATA[15:8]는 프로그램 된다.
KADDR[1:0]=11 이면, KDATA[7:0]는 프로그램 된다.
워드-사이즈 참조에 대해, 상기 KDATA 신호를 경유하여 전송된 데이타 값의 비트(31 부터 16) 또는 비트(16 부터 0)는 KADDR 신호의 비트하나의 함수로서 프로그램될 것이다. 만일 상기 KADDR 신호의 비트하나가 논리값 0이면, 비트(31 부터 16)는 프로그램 된다. 만일 상기 KADDR 신호의 비트 하나가 논리값 1이면, 비트(31 부터 0)가 프로그램 된다. 배길이 단어 참조에 대해, 상기 KADDR 신호의 비트(31 부터 0)가 프로그램 된다.
EBL(Enable breakpoint Level) 비트는, 표명된 경우에, 브레이크포인트 트리거에 대한 광역 인에이블(a global enable)로서 소용된다. 만일 상기 EBL 비트가 표명되지 않는다면, 모드 브레이크포인트 트리거는 불능하게 된다.
TRC(Trigger Response Control) 값은, 데이타 처리 시스템(5)이 완료된 트리거 조건에 응답하는 방법을 결정한다. 상기 트리기 응답은, 다수의 집적회로 핀에 의해 제공된 상기 DDATA 신호를 경유하여 외부 사용자에게 표시된다. 상기 TRC 값이 논리값 %01을 갖는 경우, 브레이크포인트 회로(100)에 의해 검출된 하드웨어 브 레이크포인트 조건의 발생은 제어회로(20)가 CPU(2)에 브레이크포인트 신호를 표명하게 한다. 상기 브레이크포인트 신호의 표명은 CPU(2)가 인터럽트 가능 점에서 데이타 처리 명령의 정상적인 실행을 정지시키는 것을 야기시킨다. 상기 TRC 값이 논리값 %10을 갖는 경우, 하드웨어 브레이크포인트 조건의 발생은 제어회로(20)가 디버그 인터럽트 신호를 CPU(2)에 표명하는 것을 가능하게 한다. 상기 디버그 인터럽트 신호가 표명되면, CPU(2)는 인터럽트 가능점에서 정상적인 실행을 정지시키고 선결된 예외(exception) 처리 루틴을 실행한다.
상기 CSR(Configuration Status Register: 구성상태 레지스터)은 도 8에 매우 상세히 나타나 있다. 상기 CSR은 CPU(2)의 연산구성, MMU/제어장치(4), 메모리(6), 그리고 데이타 처리 시스템(5)의 구성을 정의하는 것 외에, 상기 CSR은 브레이크포인트 회로(100)의 상태를 지시하는 상태 정보를 포함한다.
상기 CSR은 데이타 처리 시스템이 리셀되면 클리어 된다는 점을 주목하라. 더욱이 상기 CSR은 데이타 처리 시스템이 백그라운드 디버그 연산 모드 상태인 경우 여기서 도시되지 않은 외부개발시스템에 의해 판독될 수 있고 기입될 수 있다. 백그라운드 디버그 연산모드는 더 상세히 계속하여 기술될 것이다. 상기 CSR에 포함된 비트 각각의 함수는 더 상세히 계속하여 논의될 것이다.
도 7에서 나타나 바와 같이, 상기 CSR은 데이타 처리 시스템(5)내의 하드웨어 브레이크포인트 함수에 대해 판독전용 상태를 정의하고 제공하는 브레이크포인트 상태 필드(field)를 포함한다. 상기 브레이크포인트 상태 필드에서 기억된

정보는 브레이크포인트 회로(100)에 의해 제공된 브레이크포인트 트리거 정보에 대응하 여 다수의 제어 레지스터(40)에 의해 생성된다. 상기 상태 필드에서 기억된 정보는 다수의 제어 레지스터(40)에서 TDR의 함수이다. 상기 상태 필드가 논리값 $0을 기억하는 경우 하드웨어 브레이크포인트는 가능하지 않는다. 상기 상태필드가 논리값 $1을 기억하는 경우, 디버그 모듈(10)은 트리거되는 레벨 1 브레이크포인트를 기다리고 있다. 이전에 논의되었듯이, 레벨 1 브레이크포인트의 속성은 TDR(트리거 정의 레지스터)의 비트(16 부터 0)에 의해 정의된다. 만일 논리값 $2가 상기 CSR의 상태필드에서 기억되면, 레벨 1 브레이크포인트는 트리거 된다. 상기 상태 필드가 논리값 $5를 갖는 경우, 디버그 모듈(10)은 트리거 되는 레벨 2 브레이크포인트를 기다리고 있다. 이전에 논의한 바와 같이, 레벨 2 브레이크포인트의 속성은 TDR(트리거 정의 레지스터)의 비트(31 부터 16)에 의해 정의된다. 만일 논리값 $6이 상기 CSR의 상태필드 내에 기억되면, 레벨 2 브레이크포인트는 트리거 된다.

상기 CSR의 상태필드에서 나타난 브레이크포인트 상태는 DDATA 신호를 제공하는 다수의 외부 집적 회로 핀을 경유하여 도 2에 나타나지 않은 외부 사용자에게 제공된다. 도 9를 보자. DDATA 신호는, 상기 신호가 K-버스(25)에 포착된 데이타를 표시하지 않고 있는 경우, 브레이크포인트 상태를 반영한다. 게다가, CSR의 내용은 어느때라도 직렬 인터페이스(30)를 사용하여 판독될 수 있다는 점을 주목하라. 따라서 브레이크포인트 상태는 DDATA 신호가 포착된 데이타 또는 직렬 인터페이스(30)를 사용하여 판독 CSR 명령을 통해 포착된 데이터를 표시하지 않는 경우, 두개의 방식 DDATA 신호를 경유하여 사용가능하다.

상기 CSR은 또한 외부시스템(7)애 백그라운드 디버그 모드로의 엔트리 상태 를 지시하는 4비트 값을 제공한다. 4비트 값의 제 1 비트는 FOF 비트이다. 상기 FOF(Fault-on-Fault) 비트는, 상기 FOF 비트가 표명된 경우, 데이타 처리 시스템(5)의 CPU(2)의 파국적 중지가 발생하고 데이타 처리 시스템(5)은 백그라운드 디버그 연산 모드를 입력해야 한다는 것을 표시한다. 상기 백그라운드 디버그 연산 모드는 계속해서 더 상세히 논의 될 것이다. 상기 FOF 비트는 CSR의 내용의 판독 부상에서 클리어된다.

TRG(Hardware Breakpoint Trigger) 비트는, 상기 TRG 비트가 표명된 경우, 하드웨어 브레이크포인트 데이타 처리 시스템(5)의 CPU(2)의 연산을 중지시킨다는 것을 표시한다. 상기 TRG 비트는, 상기 TRG 비트가 표명된 경우, 데이타 처리 시스템(5)은 백그라운드 디버그 연산모드를 입력해야 된다는 것을 표시한다. 상기 TRG 비트는 CSR의 내용의 판독부상에서, 또는 직렬 인터페이스(30)을 통해 직렬로 제공된 'Go' 명령의 수신부 상에서 클리어된다.

처리장치 정지 비트는, CPU(2) 가 표명된 경우, CPU(2)는 정지해야 한다는 명령을 CPU(2)가 실행했다는 것을 표시한다. 상기 처리장치 정지 비트가 표명된 경우 상기 정지 비트는 데이타 처리 시스템(5)은 백그라운드 디버그 연산모드를 입력해야 한다는 것을 표시한다. 상기 정지 비트는 CSR의 내용의 판독부상에서 또는 직렬 인터페이스(30)을 통과하여 직렬로 제공된 'Go' 명령의 수신부 상에서 클리어 된다.

BKPT 표명 (BKPT) 비트는 BKPT 신호가 CPU(2)에 표명되고 제공된다는 것을 표시한다. 상기 BKPT 비트가 표명된 경우, 상기 BKPT 비트는 데이타 처리 시스템(5)은 백그라운드 디버그 연산모드를 입력시켜야 한다는 것을 표시한다. 상기 BKPT는 CSR의 내용의 판독부상에서 또는 직렬 인터페이스(30)를 통해 직렬로 제공된 'Go' 명령의 수신부 상에서 클리어 된다.

IPW(Inhibit Processor Writes to Debug Registers) 비트는 상기 IPW가 표명되면 다수의 하드웨어 브레이크포인트 레지스터(50)와 다수의 제어 레지스터(40)에 대해 쓰기 조작을 금지시킨다. 상기 IPW 비트는 외부개발시스템(7)과 같은 외부개발시스템으로부터 제공된 명령에 의해 단지 변경될 수 있을 것이다. 상기 IPW 비트는 CPU(2)를 폐쇠하여 외부개발시스템 (7)에 의해 다수의 제어 레지스터(40) 및 다수의 브레이크포인트 레지스터(50)에 기록된 디버그 정보를 겹쳐쓰지 못하게 한다.

에뮬레이터 모드내의 강제 처리장치 참조(MAP) 비트는, 표명된 경우, CPU(2)가 특정 어드레스 공간에 대해 데이타가 처리 시스템이 에뮬레이터 모드에서 작동하고 있는 경우 모든 참조를 사상하도록 강제한다. 만일 상기 MAP 비트가 부정되면, 모든 에뮬레이타 방식 참조는 메모리(6)의 감시프로그램 본문과 데이타 공간으로 사상된다.

본 발명은 감시 프로그램 또는 특전 모드의 논리적 확장부인 특정 연산모드로서 에뮬레이터 모드를 포함한다. 상기 특정 연산모드는 본 발명의 본 실시예의 3개의 가능 표시장치의 하나의 표명상에 입력된다. 일단 에뮬레이터 모드에서, 모든 정상적인 입/출력 인터럽트는 무시될 수 있고 데이타 처리장치(3)는 선택적으로 모드 메모리 참조를 대체 공간으로 밀어 넣을 수 있다. 상기 MAP 비트는 데이타 처리장치(3)의 상기 연산을 제어한다. 에뮬레이터 작동 모드는 전형적으로 외부개발시 스템(7)이 상기 대체 공간으로 모든 메모리 참조를 인터셉트하게 하고, 몇몇 디버그 함수형을 지원하는데 필요한 명령과 데이타 값을 제공하도록 한다. 상기 교체 공간으로 모든 메모리 참조를 인터셉트 하게 하고, 몇몇 디버그 함수형을 지원하는데 필요한 명령과 데이타 값을 제공하도록 한다. 상기 교체 공간 재사상은 외부개발시스템(7)이 상태 또는 정상적인 시스템 메모리를 변경시키지 않고 데이타 처리 장치(3)의 연산을 제어하는 것을 달성하게 하기 위해 제공된다. 본 발명의 본 실시예에서, 데이타 처리장치(3)는 RTE(Return from Exception)과 같은 특정 명령이 실행될 때까지 에뮬레이터 모드에 남아 있다. 그때, 데이타 처리장치(3)는 에뮬레이터 모드를 나와 CPU(2)는 정상의 작동 모드로 귀환한다.

본 발명의 본 실시예에서, 에뮬레이터 모드 입구와 축구는 PST 신호를 경유하여 외부의 사용자에게 신호가 보내진다는 것을 주목해야 한다. PST 신호상의 값 $D는 에뮬레이터 모드로의 입구를 지시하고 PST 신호 상의 값 $7는 에뮬레이터 모드로 부터의 출구를 지시한다. PST 신호에 의해 제공된 에뮬레이터 작동 모드로 들어가는 입구 그리고 에뮬레이터 작동 모드로부터 나오는 출구의 지시는 상기가 외부의 개발시스템(7)에 지시를 하는 것과 같이 특히 중요하여 하드웨어 브레이크포인트는 트리거 된다. 따라서 직렬 인터페이스(30)를 경유하여 제공된 직렬 포트(Port)는 외부 메모리 위치로부터 디버그 정보를 검색하기 위해 사용될 수 있다.

만일 상기 MAP 비트가 부정되면, 모든 에뮬레이터 모드 메모리 참조는 온-칩(on-chip) 메모리, 외부 메모리, 그리고 입력/출력 장치 같은 시스템 자원이 참조될 수 있을 만큼 정상적으로 감시 프로그램 액세스에 사상된다.

게다가 본 발명의 본 실시예에서, 에뮬레이터 작동 모드 속으로 들어가기 위한 3가지 방법이 실현되어 있다. 제 1 방법에서, 만일 TRC(Force Enulation Mode on Treace Exception) 비트가 표명되면, 소프트웨어 트레이스 예외가 발생하여 데이타 처리장치(3)가 에뮬레이터 작동 모드속으로 들어가는 것을 강제한다. 상기는 데이타 처리장치(3)로 가동하는 소프트웨어 디버거에 의해서가 아니라 외부개발시스템(7)에 의해 제어되는 단하나의 명령 디버거 루틴을 생성하는 메카니즘 이다.
제 2 방법에서, 만일 EMU(Force Enulation Mode) 비트가 표명되면, 데이타 처리장치(3)는 예외 처리장치를 리셋 하기를 시작한다. 다시, 상기 방법은 상기가 명령 처리를 시작하기 전에 외부개발시스템(7)이 데이타 처리 시스템(10)을 제어하게 한다.
제 3 방법은, 디버그 인터럽트가 트리거된 하드웨어 브레이크포인트에 응답하여 표명된 경우, 데이타 처리장치(3)가 에뮬레이터 모드로 들어가는 것을 가능하게 한다. TDR의 TRC 필드는 하드웨어 브레이크포인트 트리거의 사용을 프로그램 하기 위해 사용될 수 있어 에뮬레이터 모드로의 입구와의 디버그 인터럽트를 생성한다.
DDC(Debug Data Control) 값은 오퍼렌드 데이타를 포착키 위한 구성 제어 정보를 제공하여 DDATA 신호를 경유하여 외부개발시스템에 제공된다. 상기 DDC 값이 %00인 경우, 오퍼랜드 데이타는 DDATA 신호상에 표시되지 않는다. 상기 DDC 값이 %01인 경우, M-버스(23)를 경유하여 전달된 모든 기록 데이타는 포착되어 DDATA 신 호상에 표시된다. 상기 DDC 값이 %10인 경우, M-버스(23)을 경유하여 전달된 모든 기록 데이타는 포착되고 DDATA 신호상에 표시된다. 상기 DDC 값이 %11인 경우, M-버스(23)를 경유하여 전달된 모든 판독 및 기록 데이타는 포착되고 DDATA 신호상에 표시된다. 상기 DDC 값에 의해 특정된 정보의 유형에 관계없이, DDATA 신호는 오퍼랜드 참조 크기에 의해 정의된 많은 바이크를 표시한다. 오퍼랜드가 CSR의 DDC 필드의 제어하에 포착되는 경우, DDATA 신호상에 표시된 많은 바이트는 실행된 한 유형의 참조에 의해 결정된다. 만일 오퍼랜드가 바이트로 참조되면 8비트의 데이타가 표시된다. 만일 오퍼랜드가 워드로서 참조되면, 16 비트의 데이타가 표시된다. 유사하게, 만일 오퍼랜드가 배길이 단어로서 참조되면, 32 비트의 데이타가 표시된다. DDC는 M-버스(23) 판독 및 기록 연산을 포착하는 가능성을 제공한다. 따라서, 오퍼랜드 참조 크리는 DDATA 신호상에 실제로 표시된 많은 바이트를 결정한다.
BTB(Branch Tarset Bytes) 값은 분기 목표 어드레스의 많은 바이트는 DDATA 신호상에 표시된다는 것을 정의한다. 상기 BTB 값이 %00인 경우, 브렌치 타겟 어드레스의 바이트는 DDATA 신호상에 표시되지 않는다. 상기 BTB 값이 %01인 경우, 상기 분기 목표 어드레스의 보다 하위의 두개의 바이트는 DDATA 신호상에 표시된다. 상기 BTB 값이 %10인 경우, 상기 분기 목표 어드레스의 보다 하위의 3개의 바이트는 DDATA 신호상에 표시된다. 상기 BTB 값이 %11인 경우, 상기 분기 목표 어드레스의 모든 4개의 바이트는 DDATA 신호상에 표시된다. 상기 DDATA 신호는 통상 최하위 순서에서 최상위 순서를 따라 데이타를 출력한다. 상기는 분기 목표 어드레스 뿐만아니라 포착된 오퍼랜드에 대해서도 사실이다. 게다가, 연산중에, FIFO 기억 버퍼(70)는 제어회로(60)의 제어하에 다른 어드레싱 모드를 사용하는 포획된 분기 연산과 연관된 상기 목표 어드레스만을 단지 포착한다. 상기 다른 어드레싱 모드는, 프로그램-카운터 상대 또는 절대 어드레스에 의해 제공되지 않고, 몇몇 다른 수단에 의해 계산되는 어드레싱 모드이다. 상기 목표 어드레스는 전형적으로 RTE(Return from Exception: 예외로부터 복귀) 명령, RTS(서브루틴으로부터 복귀) 명령, 그리고 몇몇 JMP(forms of Jump) 및 JSR(Jump-to-Subroutire) 뿐만아니라 예외 벡터와 관련된다.
END(Enable Debug Mode) 비트는, 표명되면, PST 및 DDATA 신호의 정상적인 연산을 불능하게 한다. 더욱이 상기 비트가 사용 가능하게 되면, 데이타 처리장치(3)의 CPU(2)는 8비트 벡터의 내부상태 정보를 출력한다.
NPL(Non-Pipelined Mode) 비트는, 표명되면, CPU(2)가 비 파이프라인 연산의 모드에서 작동하게 강제한다. 데이타 처리 시스템(3)이 비 파이프라인 연산모드에 있는 경우, 데이타 처리 시스템(3)은 효과적으로 둘 또는 그 이상의 연속적인 명령 사이의 오버랩이 없이 한번에 단하나의 명령을 실행한다.
IPI(Ignore Pending Interrupts) 비트는, 표명되는 경우, CPU(2)가 데이타 처리 시스템(5)이 단 하나의-명령-스텝 모드에서 작동하고 있는 경우 미정의 인터럽트 요구 어느것 이라도 무시하게 강제한다.
SSM(Sirgle-Step Mode) 비트는, 표명된 경우 단하나의-명령-스텝-모드에서 CPU(2)가 작동하게 강제한다. CPU(2)가 단하나의-명령-스텝 모드 내에 있는 동안, CPU(2)는 단하나의 명령을 실행하고 그후 정지한다. CPU(2)가 정지되면 백그라운드 디버그 모드와 연관된 어떠한 명령이라도 실행될 수 있다. Go 명령을 수신하자마자, CPU(2)는 다음의 명령을 실행하고 그후 다시 정지한다. 상기 과정은 단하나의-명령-스텝 모드가 불능하게 될 때까지 계속된다.
SBD(Store Buffer Disable) 비트, KAD(K-Bus RAM Disable) 비트, KOD(K-Bus ROM Disable) 비트는 외부개발시스템(7)이 디버그 목적을 위해 데이타 처리장치(3)의 정상적인 구성을 무사하게 하는데 함께 사용된다. 특히, 상기 SBD 비트는 MMU/제어장치(4)에 위치하고 FIFO 기억장치 버퍼(70)와 혼란되어서는 안되는 임의의 FIFO 기억장치 버퍼의 사용을 불능하게 한다. 상기 KAD 비트는 메모리 블록(6)의 사용이 바이패스되고 M-버스(23) 엑세스를 강제하게 한다. 메모리 블럭(6)은 램 모듈, 롬 모듈, 또한 캐시 모듈을 포함할 수 있다. 상기 KAD, KOD, 그리고 KCD 비트는 상기 모듈의 각각에 의해 제공된 함수를 불능하게 하기 위해 각각 사용된다.
다수의 하드웨어 브레이크포인트 레지스터(50)는 도 5에 매우 상세히 나타나 있다. 상기 다수의 하드웨어 브레이크포인트 레지스터(50) 각각의 함수에 대한 일반적인 설명은 여기서 제공될 것이다. 상기 다수의 하드웨어 브레이크포인트 레지스터(50) 각각의 함수에 대한 보다 상세한 설명은 계속해서 더욱 상세히 기술될 것이다.
도 5에서, 어드레스 브레이크포인트 레지스터(ABLR 과 ABHR)는 브레이크포인트 함수를 트리거 하기 위해 사용될 수 있는 데이타 처리 시스템(5)의 오퍼랜드 논리 어드레스 공간내의 영역을 정의한다. ABLR과 ABHR 각각에 기억된 어드레스 값과 비교된다. 게다가, 다음의 논의에서 브레이크포인트 함수를 실행하기 위해 충족되 어야 하는 한 세트의 조건이 트리거로서 언급되어 있다는 점을 주목하라.
AABR(Address Attribute Breakpoint Register)은 도 6 에 매우 상세히 나타나 있다. 상기 AABR은 다수의 어드레스 속성과 마스크가 브레이크포인트 한수에 대한 트리거에서 부합되는 것을 정의한다. 상기 ABLR과 ABLR 어드레스

값과 같이, ABBR에 기억된 속성값은 K-버스(25)를 경유하여 전송된 어드레스 속성값과 비교된다.
AABR의 보다 낮은 다섯개의 비트, TT(Transfer Type) 값과 TM(Transfer Modifier) 값은 백그라운드 디버그 모드 상태인 동안, 데이타 처리 시스템(5)이 메모리 기준을 실행하는 경우 사용되는 어드레스 공간을 함께 정의한다. 상기 TT 값은 상기 전송이 감시프로그램 오퍼랜드 인지, 감시프로그램 명령인지, 사용자 오퍼랜드 또는 사용자 명령인지를 표시한다.
다수의 하드웨어 브레이크포인트 레지스터(50)는 두가지 함수로 소용된다는 것을 주목해라. 데이타 처리장치(3)가 백그라운드 디버그 연산모드에서 작동하고 있는 경우, 다수의 하드웨어 브레이크포인트 레지스터(50)는 메모리 및 처리장치 참조 연산의 실행중에 사용된다. 데이타 처리장치(3)가 백그라운드 디버그 연산 모드에서 작동하지 않고 있는 경우, 다수의 하드웨어 브레이크포인트 레지스터(50)는 하드웨어 브레이크포인트를 검색하는데 사용된다.
다수의 하드웨어 브레이크포인트 레지스터(50)가 하드웨어 브레이크포인트를 검색하는데 사용되는 경우, 상기 AABR 레지스터는 4개의 동등 마스크 필드외에 4개의 필드를 포함한다. 상기 TT 및 TM 필드는 액세스 되는 어드레스 공간을 정의한 다. 표 1은 필드의 각각에 대한 부호화를 나타낸다.
[표 1]

TT	TM	어드레스 공간
00	001	사용자 오퍼랜드
00	010	사용자 명령
00	101	감시프로그램 오퍼랜드
00	110	감시프로그램 명령
01	XXX	사용되지 않음
10	101	에뮬레이터 모드 오퍼랜드(만일MAP=1이면)
10	110	에뮬레이터 모드 명령(만일 MAP=1이면)
11	000	CPU-공간
11	001	인터럽트 응답
11	01X	인터럽트 응답
11	1XX	인터럽트 응답

'어드레스 공간'은 특권의 작동레벨(사용자 모드 또는 감시프로그램모드)과 수행된 참조(오퍼랜드, 명령인출)의 형태에 의해 가끔 정의된다는 것을 주목해라. 동시에, 상기 TT와 TM값은 브레이크포인트 위치를 더 정확히 정의하고자 사용될 수 있다.
상기 AABR에서, 다음 값, 즉 SZ(Size) 값은 어드레스 범위외에 K-버스(25)를 경유하여 전송된 크기 특성은 AABR의 상기 SZ 비트에서 저장된 값에 대응해야 한다는 것을 표시한다. 유사하게, 판독/기록(R)값은 상기 AABR에서 기억된다. 상기 R 값은, 판독 또는 기록 연산에 근거를 둔 트리거 조건은 어드레스에 근거한 비교에 가해진다는 것을 표시한다.
앞서 언급한 비트는 다수의 브레이크포인트 레지스터(50)가 브레이크 퐁닛 모니터로서 작동하는 경우에 사용된다. 상기 비트는 K-버스(25) 속성 정보와 비교 된다. 예를들어, 다수의 브레이트포인트 레지스터(50)를 프로그램하여 사용자 모드 판독된 워드-크기 오퍼랜드만이 트리거된다. 따라서, AABR의 상기 비트는 어드레스 브레이크포인트 정의를 위한 부가적인 수식(qualification)을 제공하는데 사용된다.
게다가, AABR은 브레이크포인트 정의부로 부터 각개의 비트를 마스크 아웃하는 능력을 제공하는 속성 마스크 비트를 포함한다.
AABR의 최상위 비트, RM(Read/Write Mask) 비트는, 판독 도는 기록연산에 근거한 트리거 조건은 어드레스에 근거한 비교에서 무시된다는 것을 표시한다. 따라서, 상기 RM 비트가 표명되면, R 비트는 비교 연산중에 무시된다. 다음의 비트의 세트, SM(Size mask) 값은 어드레스의 크기에 근거한 트리거 조건은 어드레스에 근거한 비교에서 무시된다는 것을 표시한다. 예를들어 SM 값내의 비트가 표명되면 SZ값 내의 대응 비트는 어드레스 비교 함수내에서 무시된다.
TTM(Transfer Type Mask) 값은 상기 TT값에 대응한다. 상기 TTM값 내의 비트가 표명되면, 상기 TT값 내의 대응 비트는 계속적인 어드레스 비교에서 무시된다. 유사하게, TMM(Transfer Modifier Mask)값은 상기 TM값에 대응한다. 상기 TMM 필드내의 비트가 표명되면, 상기 TM값 내의 비트는 계속적인 어드레스 비교에서 무시된다.
다수의 하드웨어 브레이크포인트 레지스터(50)는 또한 다수의 프로그램 카운터 브레이크포인트 레지스터를 포함한다. 상기 다수의 프로그램 카운터 브레이크포인트 레지스터는 브레이크포인트 함수를 트리거 하는데 사용될 수 있는 데이타 처 리시스템(5)내의 명령 논리 어드레스 공간에서의 영역을 정의한다. 프로그램 카운터 브레이크포인트(PBR) 값은 PMBR(Program Countor Breakpoint Mask) 레지스터에 기억된 값에 의해 마스크 될 수 있다. 본 발명의 한 실시예에서 PBMR에서 기억된 대응 제로 값을 갖는 PBR에서의 상기 비트만이 데이타 처리 시스템(5)의 프로그램 카운터 값과 비교 연산하는데 사용된다. 프로그램 카운터 값은 데이타 처리 시스템(5)의 CPU(2)의 여기서 나타나지 않은 프로그램 카운터 레지스터에서 기억된다는 것을 주목해라.
프로그램 카운터 브레이크포인트 레지스터 외에, 다수의 하드웨어 브레이크포인트 레지스터(50)는 다수의 데이타 브레이크포인트 레지스터를 포함한다. 상기 다수의 데이타 브레이크포인트 레지스터는 DBR(Data Breakpoint Register)와 DBMR(Data Breakpoint Mask Register)를 포함한다. 상기 DBR은 데이타 처리 시스템(5)이 디버그 작용을 수

행하는 경우 트리거로 부터 사용될 수 있는 특정 데이타 패턴을 정의하는 데이타 값을 기억한다. 상기 DBMR에 기억된 데이타 마스크 값은 상기 DBR에 기억된 데이타 값을 마스크한다. 따라서 본 발명의 한 실시예에서 DBMR에 기억된 대응하는 제로 값을 가진 DBR에 기억된 상기 버트만이 K-버스(25)를 경유하여 전송된 데이타신호와 비교될 수 있어 트리거 조건이 충족되었는가를 결정한다.

연산에 대한 기술

계속적인 토론을 위해, 디버그 지원에 관한 일반적인 논제는 독특한 필요조건과 독특한 기능을 가진 3개의 분리된 영역으로 분리될 수 있다. 본 발명은 실시 간 트레이스 지원, 백그라운드 디버그 연산모드, 그리고 실시간 디버그 지원을 제공한다. 도 1의 데이타 처리 시스템(5)은 본 발명의 한 실시예를 나타낸다.

데이타 처리 시스템(5)에서, 데이타 처리장치(3)는 외부개발시스템(7)에 접속되어 정보를 전달한다. 그후, 외부개발시스템(7)은 전달된 정보를 사용하여 데이타 처리장치(3)에서 고정난 회로와 결함있는 있는 소프트웨어 코드를 식별하고 분리하고자 설계된 디버그 연산을 수행한다. 외부개발시스템(7)은 상기 DSDI, DSCLK, DSDO, DDATA, PST, 그리고 다른 입/출력 신호를 경유하여 데이타 처리장치(3)와 통신한다. 상기 DSDI, DSCLK, 그리고 DSDO 신호는 외부개발시스템(7)과 데이타 처리장치(3) 사이에 직렬 전달 수단을 제공한다. 상기 DDATA 신호는 데이타 처리장치(3)로부터 외부개발시스템(7)에 디버그 정보를 제공한다. 상기 CSR의 구성에 의존하며, 디버그 모듈(10)은 DDATA 신호상에 표시하기 위한 확실한 오퍼랜드와 분기 목표 어드레스를 포착할 수 있다. 게다가, CPU(2)는 상기 CSR에 기억된 구성값에 관계없이 DDATA 신호상에 포착되고 표시된 명령을 실행할 수 있다. 상기 PST 신호는 외부개발시스템(7)에 내부 처리장치 상태를 제공한다. 따라서 프로그램 연산의 실행동안에, 상기 DDATA 및 PST 신호는 함께 엑서스되어 데이타 처리장치(3)에 의해 실행된 현재 오퍼랜드를 반영한다.

연산중에, 코어(9)는 K-버스(25)를 사용하여 CPU(2), MMU/제어장치(4), 그리고 메모리(6)에 접속한다. 본 발명의 본 실시예에서, K-버스(25)는 빠른 속도, 단일 사이클 엑세스 버스이다. 메모리(6)는 RAM, 판독전용 ROM, 캐쉬 블럭, 그리고 상기의 조합을 포함한다. 모든 다른 시스템 모듈과 주변장치는 M-버스(23)는 다중 버스 마스터의 하나에 의해 개시된 데이타 전송을 실행하는데 사용되는 내부 다중-마스타 버스이다. 시스템 버스 제어장치(8)는 다중 함수를 제공한다. 시스템 버스 제어장치(8)는, 만일 외부 E-버스가 존재한다면, 내부 M-버스(23)와 외부 E-버스 사이에 인터페이스를 제공한다. 게다가 시스템 버스 제어장치(8)는 S-버스상의 모든 데이타 전송을 제어하기 위한 촛점으로 소용된다. S-버스는 타이머와 직렬 전달 채널같은 단순한 종속 주변 모듈(12, 13)을 접속시키는데 사용된다.

본 발명에서, 데이타 처리장치(3)는 여러층의 버스 밴드폭을 포함하여 데이타 전송을 위한 비용면에서 효과적인 메카니즘을 제공한다. 코어(9)는 높은 속도, 단일의 사이클 K-버스(25)와 상호접속 되있어 성능을 최대화 시킨다. 상기 높은 속도 버스에 직접결합 되지 않은 전송부에 대해, M-버스(23)는 코어(9)와 주 모듈(14)과 같은 내부 버스 마스타의 어느것으로 부터 밴드폭을 제공한다. 시스템 버스 제어장치(8)는 종속 모듈(12, 13)과 같은 종속 주변 모듈에 대한 낮은 비용, 보다 하위의 밴드폭 S-버스 상의 모든 데이타 전송을 위한 제어함수를 또한 제공하는 반면에, 내부 M-버스(23)와 외부 E-버스(만일 E-버스가 존재한다면) 사이의 접속을 제공한다. 디버그 모듈(10)은 K-버스(25)에 접속되어 모든 처리장치-개시 메모리 엑서스의 비 강제적인 진찰을 허용한다. 디버그 모듈(10)은 또한 외부개발시스템(7)에 접속을 제공한다.

도 1의 디버그 모듈(10)은 상기에 기술된 3개의 디버그 연산 각각을 지원하기 위한 논리 회로를 제공한다. 디버그 모듈(10)의 연산은 계속해서 매우 상세히 논의 될 것이다.

실시간 트레이스 함수

본 발명은 중요하지만 여기에 비존재하는 실시간 트레이스 함수를 제공한다. 상기 실시간 트레이스 함수는 외부개발시스템(7)과 같은 외부 사용자가 데이타 처리장치(3)의 내부 연산을 동적으로 관찰하게 허용한다. 게다가 상기 실시간 트레이스 함수는 데이타 처리 시스템(5)의 효율과 속도에 나쁜 영향을 주지 않고서 시스템의 내부 연산에 대한 관찰을 가능하게 해준다. 데이타 처리 시스템(5)의 디버그 모듈(10)은 DDATA 및 DST 신호를 매개로하여 배두 연산 정보를 외부 사용자에게 제공하기 위한 병렬 출력 포트를 제공한다. 상기 DDATA 신호는 오퍼랜드 데이타를 반영하는 데이타를 제공하고 상기 PST 신호는 CPU(2)의 실행상태를 반영하는 코드화된 상태 정보를 제공한다. 게다가, DDATA 신호는 또한 포착된 명령 어드레스 프로그램 흐름 변화를 제공하여 외부개발시스템이 외부에서 가시 가능한 어드레스 버스 또는 외부에서 가시가능한 데이타 버스를 필요치 않고서도 정확한 프로그램 흐름을 트레이스 하게 해준다. DDATA 신호상에 표시된 정보는 PST 신호상의 상태정보와 동기가 된다는 것을 주목해라. 따라서, 외부사용자는 DDATA 및 PST 신호를 해독할 수 있어 데이타 처리장치(3)의 연산을 정지시키지 않고 또한 연산의 특정 디버그 모드가 상기 실행 패스를 가시화 하는 것을 필요로 하지 않고 프로그램의 동적인 실행 패스를 결정한다.

실시간 트레이스의 실행동안에, 데이타 처리시스템(5)의 연산은 계속해서 매우 상세히 논의 될 것이다. 상기 논의의 이전에, 상기 PST 신호의 코드화는 간략히 기술될 것이다. 상기 PST 신호의 코드화를 나타내는 표는 도 10에 제공되어 있다.

도 10은 상기 PST 신호의 비트(3부터 0)를 경유하여 제공된 값과 CPU(2)의 상태사이의 관계를 나타낸다. 상기 PST 신호가 값 % 0000을 제공하면, 이전의 연산은 아직 완결되지 않고 실행이 단순히 계속된다. 만일 값 % 0001이 제공되면, 상기 PST 신호는 명령의 실행이 시작되고 있다는 것을 표시한다. 상기 PST 신호가 값 % 0011을 갖는 경우, 데이타 처리 시스템은 사용자 연산모드로 들어간다. 만일, 값 %0100이 PST 신호에 의해 제공되면, 데이타 처리 시스템(5)은 펄스 또는 WDDATA 명령의 실행을 시작하고 있다. 상기 펄스 연산코드는, 상기 WDDATA 연산 코드는 오퍼랜드가 CSR에 의해 정의된 구성에 관계없이 DDATA상에 포착되고 표시되는 것을 장제하는 반면에, 단순히 상기 특정 PST 부호화를 생성시킨다. 상기는 메모리 오퍼랜드, 예외 스택 프레임, 또는 소프트웨어 마커를 직접 DDATA 신호상에 표시하는 단순한 메카니즘을 제공한다.

만일 값 %0101 이 PST 신호에 의해 제공되면, 데이타 처리 시스템(5)은 선택된 분기 연산의 실행을 시작하고 있다. 만일 값 %0111이 PST 신호에 의해 제공되면, 데이타 처리시스템(5)은 RTE 명령의 실행을 시작하고 있다. PST 신호가 값 %1000을 제공하는 경우, 데이타 처리 시스템(5)은 DDDATA 신호상의 하나와 바이트 전송을 시작하고 있다. PST 신호가 값 %1001을 제공하는 경우, 데이타 처리 시스템(5)은 DDATA 신호상의 두개의 바이트 전송을 시작하고 있다. PST 신호가 값 %1010을 제공하면, 데이타 처리 시스템(5)은 DDATA 신호상의 3개의 바이트 전송을 시작하고 있다. PST 신호가 값 %1011을 제공하는 경우 데이타 처리 시스템(5)은 DDATA 신호상의 4번째의 바이트 전송을 시작하고 있다. 만일 PST 신호가 논리값 %1100을 갖게 되면, 데이타 처리 시스템(5)의 CPU(2)는 예외 처리 루틴을 실행하고 있다. 유사하게, 만일 PST 신호가 논리값 %1101을 갖게되면, 데이타 처리시스템(5)의 CPU(2)는 예외를 실행하고 있다. 상기 값은 어떤 시간으로 부터의 PST 신호상에 표명되고 예외는 예의 처리 루틴의 제 1 명령이 인출될 때까지 검출된다. 유사하게, 만일 PST 신호가 는리값 %1101을 갖게 되면, 데이타 처리 시스템(5)의 CPU(2)는 연산의 에뮬레이터 모드속으로 진입을 야기하는 예외루틴을 처리하고 있다. PST 신호가 %1110인 경우 데이타 처리시스템(5)의 CPU(2)는 정지되고 인터럽트를 기다린다. PST 신호가 %1111인 경우, 데이타 처리 시스템(5)의 CPU(2)는 정지된다.

상술한 바와 같이, 코드화된 PST 및 DDATA 신호에 의해 함께 제공된 처리장치 상태 정보는, 외부 프로그램 모델과 같이 사용될 수 있어 완전히 데이타 처리 시스템(5)의 동적 실행 패스를 추적한다.

실시간 트레이스 함수를 사용 가능하게 하기 위해 상기 DDATA 신호는 분기 목표 명령 어드레스를 표시하게 구성되어야 한다. 전형적으로, 상기 구성은 데이타 처리 시스템(5)이 정상적인 실행을 시작하기 전에 수행된다. DDATA 신호를 구성하기 위해 외부개발시스템(7)은 다수의 제어 레지스터(40)내와 CSR 레지스터에 값을 기록할 수 있다. 상기 CSR 레지스터는 도 8에 나타나 있다. 상기 기록 함수를 실행키 위해 외부개발시스템(7)은 커맨드에서 자리 이동하여 데이타를 DSDI 및 DSCLK 신호 모두를 사용하여 직렬 인터페이스(30)에 대한 CSR에 로드시킨다. 본 발명의 한 실시예에서, 커맨드의 제 1 부분은 DSDI 신호에 의해 제공되고 DSCLK 신호에 의해 클록인 된다. 커맨드의 추가부분은 계속해서 제공된다. 커맨드의 모든 부분은 16 비트 패킷의 정보가 완결될때까지 레지스터(32)에 기억된다. 또다른 본 발명의 실시예는 본 발명의 본 실시예에서 소요되는 16 비트 보다 많거나 적은 비트를 포함할 수 있다.

여기서 기술된 본 발명의 본 실시예에서, CSR에 값을 기록하는 명령의 길이는 16 비트 연산코드와 32 비트 데이타로 이루어진 48 비트이다. 레지스터(32)는 단지 16 비트 길이이다. 따라서 제 1 16 비트 패킷(Packet)의 정보가 완비되면, 상기 정보는 제어회로(20)에 전송되어 여기서 상기 정보는 해독되어 계속적인 연산이 형성된다. 제 2 16비트 패킷의 정보가 레지스터(32)속으로 로드되어 있는 경우 상기 정보는 직렬 정보 신호를 매개로 하여 다수의 제어 레지스터(32)에 전송된다. 제어회로(20)로 부터의 제어버스(15)에 의해 제공된 제어 정보를 사용하여, 직렬 정보신호의 내용은 목표 레지스터 즉 CSR의 상부속으로 로드된다. 유사한 방법으로 제 3 16비트 패킷의 정보는 레지스터(32) 속으로 로드되고 그후 다수의 제어 레지스터(40)에 전송된다. 다시, 제어회로(20)에 의해 개시된 신호를 사용하여, 직렬 정보신호의 내용은 CSR의 하위 부분속으로 로드된다. 이런 식으로 모든 48비트 명령은 계속적으로 디버그 모듈(10) 속으로 자리이동하고, 여기서 상기 명령은 해독되어 오퍼랜드 값은 CSR 속으로 로드된다.

계속적인 데이타는 직렬로 DSDI 신호를 매개로 하여 외부발전시스템(7)에 의해 제공된다. 상술한 바와 같이, DSDI 신호에 의해 제공된 계속적인 데이타는 DSCLK 신호를 매개로하여 클록인 된다. 데이타 비트 각각은 제 2 16비트 패킷의 정보가 완비될 때까지 레지스터(32)에 기억된다. 제 2 16비트 패킷의 정보가 완비된 경우, 정보는 직렬 정보신호를 매개로하여 다수의 제어레지스터(40)에 전송된다. 제 2 16 비트 괘킷의 정보는 비트(31에서 16)와 같이 CSR에 기억된다. 또한, 추가적인 정보는 계속해서 DSDI 및 DSCLK 신호를 매개로하여 레지스터(32)에 제공된다. 제 3 16비트 패킷의 정보가 완비된 경우 정보는 직렬 정보신호를 경유하여 다수의 제어 레지스터(40)에 전송된다. 제 3 16비트 패킷이 정보는 비트(15 에서 0)와 같이 CSR에 기억된다.

상기 문장에서, DDATA 신호의 구성은 외부발전시스템(7)에 의해 CSR속으로 기록된 값에 의해 결정된다. 본 발명의 다른 실시예에서, 상기 DDATA 신호는 데이타 처리장치(4)의 CPU(2)에 의해 CSR 속으로 기록된 값에 의해 형성될 수 있다. CPU(2)가 상기 값을 CSR 속으로 기록하는 경우, 상기 값은 도 6에 나타난 브레이크 포인트 회로(100)의 임시 레지스터(170)에 제공된다. 상기 값은 K-버스(25)의 KDATA 신호를 매개로 하여 임시 레지스터(170)에 제공된다. 임시 레지스터(170)로부터 상기 값은 16비트 패킷내의 레지스터(32)에 전송되고 기억된다. CSR로의 CPU 기록에 대해, KDATA 신호를 매개로하여 전송된 오퍼랜드 값은 CPU(2)로부터 K-버스(25)를 경유하여 제어회로에 제공된 제어신호를 사용하여 임시 데이타 레지스터(170)에 포착된다. 따라서, 다수의 제어 레지스터(40)로 향하는 상기 값은 외부개발시스템(T)으로부터 레지스터(32)에 제공된 것과 같이 동일하게 나타난다.

오퍼랜드가 외부개발시스템(7)에 의해서 또는 CPU(2)에 의해서 제공되던지 간에, 오퍼랜드와 관련 오퍼랜드는 DDATA 신호를 구성하는 CSR에 기억되어 실시간 트레이스 함수를 실행한다. 상기 CSR에서, DDC(Debug Data Control) 및 BTB(Branch Target Bytes) 값은 외부 발전 시스템(7)에 트레이스 정보를 제공하기 위해 DDATA 신호를 함께 구성하도록 기록되어야 한다. 상기에서 언급한 바와같이, DDC(Debug Data Control) 값은 DDATA 신호를 매개로하여, 외부개발시스템에 제공된 오퍼랜드 데이타를 포착하기 위한 구성제어 정보를 제공한다. 상기 DDC 값이 %00인 경우, 오퍼랜드 데이타는 DDATA 신호상에 표시되지 않는다. 상기 DDC 값이 %01인 경우, M-버스(23)를 경유하여 전달된 모든 기록 데이타는 DDATA 신호상에 포착되어 표시된다. 상기 DDC 값이 %10인 경우, M-버스(23)를 경유하여 전달된 모든 판독 데이타는 DDATA 신호상에 포착되고 표시된다. 상기 DDC 값이 %11인 경우, M-버스(23)를 경유하여 전달된 모든 판독 및 기록 데이타는 DDATA 신호상에 포착되고 표시된다. 상기 DDC 값에 의해 지정된 정보의 유형에 관계없이, DDATA 신호는 오퍼랜드 참조 크기에 의해 정의된 많은 바이트를 표시한다. 예를들어, 바이트 디스플레이는 길이가 8비트 이고, 워드 디스플레이는 길이가 8비트이고, 워드 디스플레이는 깊이가 16비트이고, 장(long) 디스플레이는 길이가 32비트이다. 오퍼랜드가 CSR의 DDC 필드의 제어하에 포착되는 경우, DDATA 신호상에

표시된 많은 바이트는 실행된 기준의 유형에 의해 결정된다. 만일 오퍼랜드가 바이트로서 참조되면, 8비트의 데이타가 표시된다. 만일 오퍼랜드가 워드로서 참조되면, 16비트의 데이타가 표시된다. 유사하게 만일 오퍼랜드가 배길이 단어(long-word)로서 참조되었다면, 32 비트의 정보가 표시된다. 상기 DDC는 M-버스(23) 판독 및 기록 연산을 포착하는 능력을 제공한다. 따라서 오퍼랜드 참조크기는 DDATA 신호상에 실제로 표시된 많은 바이트로 결정한다.
BTB(Branch Target Bytes) 값은 분기 목표 어드레스의 많은 바이트를 정의하고 상기 어드레스는 DDATA상에 표시된다. 상기 BTB 값이 %00인 경우, 분기 목표 어드레스의 바이트는 DDATA 신호상에 표시된다. 상기 BTB 값이 %01 인 경우, 분기 목표 어드레스의 보다 하위의 두개의 바이트는 DDATA 신호상에 표시된다. 상기 BTB 값이 %10 인 경우, 분기 목표 어드레스의 보다 하위의 3개의 바이트는 DDATA 신호상에 표시된다. 상기 BTB 값이 %11인 경우, 분기 목표 어드레스의 모든 4개의 바이트는 DDATA 신호상에 표시된다. DDATA 신호는 통상 최하위 순서 내지 최상위 순서에서 데이타를 출력한다는 것을 주목해야 한다. 상기는 분기 목표 어드레스 뿐만아니라 포착된 오퍼랜드에 대해서도 사실이다. 더욱이, 연산도중에 제어회로(60)의 제어하에 FIFO 기억 버퍼(70)는 다른 어드레싱 모드를 사용하는 선택된 분기연산과 관련된 상기 목표 번지만을 포착만한다. 상기 다른 어드레싱모드는 어드레스가 프로그램-카운터 상대 또는 절대 어드레스에 의해 제공되지 않고, 몇몇 다른 수단에 의해 계산된 어드레싱 모드이다. 상기 목표 어드레스는, RTE(Return-from-exception), RTS(Return-from-Subroutire) 명령과 몇몇 JMP(jump) 형식과 JSR(jump-to-subroutire) 뿐만아니라 예외 벡터와 전형적으로 연관된다.
상기 CSR이 실시간 트레이스 함수를 실행하는데 필요한 정보를 제공하기 위해 DDATA 신호를 구성하도륵 기록이 되어진 후, CPU(2)는 다수의 목표 어드레스를 K-버스(25)에 의해 전달된 KADDR 신호를 매개로 FIFO 기억 버퍼(70)에 제공한다. 게다가 CPU(2)는 다수의 어퍼랜드 데이타 값을 K-버스(25)에 의해 전달된 KDATA 신호를 매개로하여 FIFO 기억버퍼(70)에 제공한다. 상기 다수의 목표어드레스 및 상 기 다수의 오퍼렌드 데이타 값은 제어회로(60)애 의해 제공된 FIFO 제어신호에 대응하여 FIFO 기억 버퍼(70)에 기억된다.
제어회로(60)는 FIFO 제어신호를 제어회로(20)에 의해 제공된 트레이스 신호 및 다수의 제어 레지스터(40)의 CSR에 의개 제공된 구성신호에 대응하여 출력한다. FIFO 기억버퍼(70)는 선입선출 순서에 따라 저장된 KADDR과 KDATA 정보를 출력한다. FIFO 기억버퍼(70)는 어드레스 및 오퍼랜드 정보를 기억하기 위한 다수의 기억위치를 포함하고 있으므로, FIFO 기억버퍼는, 각각이 데이타 처리장치(3)의 정상적인 연산중에 억세스된 오퍼랜드의 값과 명령 어드레스를 반영하는 정보를 전달하는 것과 같이, KADDR과 KDATA 신호를 수신할 수 있다. 다른식으로 말하면, FIFO 기억버퍼(70)에 제공되고 기억된 KADDR 및 KDATA 신호를 매개로 하여 전송된 값은 실시간에서 데이타 처리 시스템(5)의 현재 연산을 나타낸다.
FIFO 기억버퍼(70)는, FIFO 기억버퍼(70)내의 모든 기억 위치가 DDATA 신호상에 출력되어야 하는 유효한 정보를 포함하는 경우, CPU(2)의 연산 속도에 영향을 끼칠뿐이다. 모든 기억 위치가 채워진 경우, FIFO 기억버퍼(70)는 전(Full) 신호를 표명한다. 상기 전 신호가 표명되면, 제어회로(60)는 기능정지(Stall) 신호를 표명한다. 상기 기능정지 신호는 FIFO 기억버퍼(70)내의 최소한 하나의 기억위치가 다음의 값을 기억하는데 사용가능하게 될때까지 코어(9)의 연산을 기능정지 하기 위해 CPU(2)에 제공된다. 모든 다른 연산에서, 어드레스 및 오퍼랜드 정보를 DDATA 상에 제공하는 것은 데이타 처리장치(3)의 실행속도에 크게 영향을 미치지 않는다.
FIFO 기억버퍼(70)는 어드레스 및 오퍼랜드 정보를 상기 FIFO 신호를 매개로 하여 다수의 다중화장치(80)에 제공한다. CPU(2)는 또한 CPST(CPU Processor Status) 신호를 다수의 멀티플렉서(80)에 제공한다. 상기 CPST 신호는 처리장치 상태 정보를 제공하여 현재 데이타에 의해 실행되고 있는 오퍼랜드의 유형을 나타낸다. 예를들어, 상기 CPST 신호는 명령의 실행이 시작된 경우를 명령의 실행이 계속되어야 하는 경우를 데이타 처리장치(3)가 연산의 선택된 모드속으로 들어가는 경우를 사전에 선택된 분기 명령이 실행되는 경우를, 그리고 데이타 처리장치(3)의 연산이 정지되는 경우를 나타낸다. 상기 CPST 신호는 처리 장치의 연산의 어떤 유형이라도 나타낼 수 있고 여기서 실린예에 한정되지 않는다는 것을 주목하라.
FIFO 및 CPST 신호가 다수의 다중화장치(80)에 제공된 경우, 다수의 다중화 장치의 출력은 시스템 다중화(Mux) 제어신호에 의해 제어된다. Mux 제어신호는 다수의 다중화 장치(80)가 동기적으로 DDATA 및 PST 신호를 제공하는 것을 가능케 한다. DDATA 및 PST 신호는 여기서 도시되지 안은 데이타 처리장치(3)의 클록의 상승 에지(edge)와 동기하여, 따라서 PST 신호를 매개로 하여 제공된 상태는 어느 소정의 버스 사이클의 발생(transaction)과 무관할 수 있다.
더욱이, PST 신호가 Mux 제어신호에 의해 가능하게 된 경우, PST 신호는 CPU(2)의 상태를 나타내는 마커(marker)를 제공한다. 예를들어, 상태값 $C, $E, 또는 $F가 PST 신호에 의해 제공된 경우, PST 신호는 멀티-사이클 모드가 또는 특정 연산이 데이타 처리장치(3)에 의해 실행되고 있다는 것을 나타낸다. 코드화된 PST 신호에 관한 더 많은 정보를 위해 도 10을 언급한다. PST 신호가 특정 연산의 멀티 사이클 모드는 데이타 처리장치(3)에 의해 실행되고 있다는 것을 반영하는 경우, PST 신호는 상기 모드가 나오게 되거나 연산이 완비될 때까지 상기 상태를 유지한다.
PST 신호가 상태값 $0 내지 $7을 출력하는 경우, PST 신호는 각각의 타이밍 사이클 상에서 갱신된 정보를 제공한다.
PST 신호가 상태값 $8, $9, $A, 또는 $B를 제공하면, PST 신호는 DDATA 신호를 매개로하여 외부에 제공된 값을 수식하기 위해 사용되고 있는 중이다. 상기 연산에서, PST 신호는 DDATA 신호상의 어떤값이 필요한 어드레스 및 데이타 정부를 제공하고 있는 지를 나타내는 '마커(marker)'로서 사용된다. PST 신호는 실제 데이타가 DDATA 신호상에 표시되기 원 타이밍 사이클전에 상기 상태값으로 코드화 된다. 따라서, 외부개발시스템은 올바르게 데이타 처리장치(3)의 연산을 관측할 것이다.
다음의 예는 실시간 트레이스 연산에서 DDATA 및 PST 신호의 사용을 보다 상세히 나타낼 것이다. 상기 예는 도 11에 나타나 있다. 도 11은 간접적인 JMP(jump) 명령의 실행을 나타내고 여기서 점프 명령의 목표 어드레스의 보다 하

위의 16비트는 DDATA 신호상에 표시된다. 점프 명령이 실행되는 경우, 프로그램의 실행은 상기 명령의 오퍼랜드에 의해 특정된 유효 어드레스에서 계속된다. 본 예에서, 유효 어드레스 값은 레지스터(40)에 의해 특정된 메모리 위치에 위치한다.
다음의 예에서, CSR은 DDATA 신호를 정확히 구성하기 위해 기록되어서 적절한 포맷으로 데이타를 제공한다는 것을 가정하라. 도 11에서 나타난 예에서 JMP(AO) 명령을 도 11에서 '라스트(Last)'라고 불리는 이전의 명령을 뒤따른다. JMP 명령은 목표 명령 어드레스를 간접적으로 어드레스 한다는 것을 주목해라. 예를들어, CPU(2)에 기억된 레지스터(AO)의 내용은 소요의 목표 명령 어드레스를 정의한다. 더욱이, 본 특허출원에서 상세히 도시되지 않았지만, AO 레지스터는 CPU(2)에 위치할 수도 있다. 상기 간접 어드레싱의 사용은, 목표 명령 어드레스는 AO 레지스터의 내용에 의존하고 PC-상대 또는 절대 어드레스가 아니므로, 다른 어드레싱으로 참조된다.
다음 예의 실시동안에, DSOC와 AGEX는 데이타 처리장치(3)의 내부 파이프라인 내의 위치를 나타낸다. 용어 DSOC, AGEX, IAG, 그리고 IC를 설명하기 위해 CPU(2)의 내부구조에 관한 간단하 설명이 제공되어야 한다.
데이타 처리장치(3)의 CPU(2)에서, 두개의 독립적이며 결합되지 않은 파이프라인이 실현되 있다. 제 1 파이프 라인은 여기서 도시안된 명령을 사전에 인출하기 위한 IFP(명령인출 파이프라인: instraction fetch pipeline) 이고 제 2 파이프라인은 실제적인 명령의 실행을 수행하고 해독하기 위한 OEP(오퍼랜드 실행 파이프라인 : operand execution pipeline) 이다. 여기서 도시되지 않은 FIFO 명령 버퍼는 두개의 파이프라린 사이의 해독 메카니즘으로 소용된다.
상기 명령 인출 파이프라인은 두 단계를 포함한다. 제 1 단계는 다음의 사전 인출 어드레스가 계산되는 IAG(명령 어드레스 생성: Instruction Address Generation)단계이다. 제 2 단계는 소요의 명령이 메모리로부터 인출되는 명령 인출 사이클이다. 오퍼랜드 실행 파이프라인은 또한 두개의 단계를 포함하는데, 각 단계는 단 하나의 명령을 실행하는 동안 두개의 함수를 위해 사용될 수 있다. 제 1 단계는 OC(오퍼랜드 인출 사이클)를 구비한 DS(Decode Select: 해독 및 선택) 상태이다. 제 2 단계는 EX(실행) 단계와 더불어 있는 AG(어드레스 생성) 단계이다.
두개의 레지스터 사이에서 정보가 전달되는 명령을 위해 두개의 OEP 단계는 각각 한번 엑세스 된다. 레지스터 오퍼랜드는 OC 단계동안에 엑세스되고 그후 명령 실행을 EX 단계에서 실제로 수행된다. 따라서, OEP 연산은 OC로부터 EX까지 처리된다.
메모리 위치와 레지스터 사이에서 정보가 전달되는 명령을 위해 각 파이프라인 단계는 두번 엑세스 된다. 첫번째는, 오퍼랜드 어드레스를 위한 구성요소는 DS 단계에서 선택된다. 두번째로, 상기 구성요소 각각은 합계되어 AG 단계에서 오퍼렌드 어드레스를 형성한다. 세번째로, 메모리 오퍼랜드는 OC 단계에서 어느 다른 레지스터 오퍼랜드와 함께 억세스된다. 네번째로, 실제의 명령 실행은 EX 단계에서 수행된다. 상기 유형의 명령을 위해, OEP 연산은 DS로부터 AG까지, OC까지, EX까지 이다.
다른 유형의 명령을 위해, 각 파이프라인 상태내의 두개의 연산은 동시에 수행된다. 상기 유형의 명령을 위해 OEP 연산은 DSOC부터 AG EX 까지이다.
상기 예에서, 레지스터 AO의 내용은 '목표'의 명령 어드레스를 정의한다고 가정된다. JMP 명령이 오퍼랜드 실행 파이프라인의 AGEX를 차지함에 따라, 명령 목표 어드레스 '목표'로서 계산된다. 상기 흐름 변경 연산은 명령 인출 파이프라인이 인출 어드레스의 상기 현재 흐름을 버리게하고 '타켓' 어드레스에서 새로운 흐름 시작을 정의하게 한다. 본 발명의 상기 실시예에서, 상기 흐름 변경 JMP 명령은 두 개의 파이프라인이 새로운 명령흐름이 설정되게 야기하는 AGEX ALC IAG 단계에서 결합되도록 야기시킨다. 일단 흐름변경이 발생하면, 두개의 파이프라인은 새로운 어드레스에서 연속적으로 사전인출을 계속하는 정상적이고, 결합되지 않은 명령인출 파이프라인과 더불어 연산모드로 복귀한다. 따라서 어드레스 목표, 목표+4, 그리고 목표+8은 계속되어 인출될 것이다.
JMP 명령이 상기 오퍼랜드 실행 파이프라인 단계를 완결함에 따라, CPU(2)는 내부 CPST 신호상에 '선택(taken) 분기' 값을 신호에 준다. 따라서, 내부 CPST 신호는 값 $5를 갖는다. 본 발명의 상기 실시예에서 실제의 PST 및 DDATA 신호는 원(one) 타이밍 사이클에 의해 내부 짝(counterpart)으로부터 지연된다. CPU(2)가 새로운 어드레스에서 명령을 사전인출함에 따라 FIFO 기억 버퍼(70)는 명령 인출 파이프라인 IC 사이클의 마지막에 임하여 명령 어드레스 목표를 포착한다. 다음의 타이밍 사이클때에 다수의 다중화장치(80)는 마커를 PST 신호속으로 삽입하여 두개의 바이트 정보가 DDATA 신호상에 표시된다는 것을 나타낸다. 따라서, PST 신호는 값 $9를 갖는다. 다수의 다중화 장치(80)의 연산은 제어회로(60)에 의해 제어된다. PST 신호상의 마커의 등장 이후에 다수의 다중화장치(80)는 최하위 니블(nibble)로 즉 비트(3 부터 0)에서 비트(15부터 12)까지로 시작하는 포착된 명령 어드레스 목표의 하위 순서 16 비트를 계속해서 표시한다.
내부 PST 값(도 2의 CPST 신호)은 제 1 타이밍 사이클에서 $5 갑승 전송한다. 도 10에서 나타나 바와같이, PST값이 $5 값을 갖는 경우, 데이타처리장치는 선택된 분기 명령을 실행하기 시작하고 있다. 상기는 레지스터 AO에 의해 간접적으로 특정된 어드레스에 프로그램 흐름을 전송하고 있는 JMP 명령에 대응한다. 도 11에 나타난 제 1 타이밍 사이클 동안에 내부 DDATA 신호(도 2의 FIFO 신호)SMS $0 값을 전송한다. 도 9에서 나타난 바와 같이, DDATA 값이 $0 값을 기억하는 경우 브레이크포인트 데이타 처리장치(3)에서 사용 가능해 진다.
제 2 타이밍 사이클 동안에 내부 PST 값은 $9 값을 전송하여 JMP 명령을 위한 목표 어드레스가 내부 DDATA 값을 매개로하여 제공된 다음의 2바이트 정보에서 특정될 수 있다는 것을 나타낸다. 상기에서, 내부 DDATA 신호는 $0 값의 전송을 계속한다. 이전에 언급한 바와같이, 내부 PST 값이 $8, $9, $A 또는 $B의 상태값을 전송하는 경우, 내부 PST 신호는 DDATA 신호를 매개로 외부로 제공된 값을 한정하는데 사용되고 있다. 상기 연산에서 내부 PST 신호는 내부 DDATA 신호상의 어떤값이 소요의 목표 어드레스 정보를 제공하고 있는지를 나타내는 '마커'로서 사용된다. 내부 PST 신호는 실제 데이타가 내부 DDATA 신호상에 표시되기 원 타이밍 사이클전에 상기 상태값으로 코드화된다.
제 3 타이밍 사이클 동안, 내부 PST 값은 $0 값을 전송하여 명령 실행이 게속된 다는 것을 나타낸다. 게다가, 제 3 타이밍 사이클 동안 내부 DDATA 신호는 목표 어드레스 값의 비트(3부터 0)를 제공한다.

제 4 타이밍 사이클 동안, CPST 값은 목표 어드레스에 위치한 제 1 명령에 의해 정의된 값을 나타낸다. 상기 시간에 내부 DDATA 신호는 목표 어드레스 값의 비트(7부터 4)를 전송한다. 계속하여 제 5 및 6 타이밍 사이클에서, 내부 DDATA 신호는 목표 어드레스 값의 비트(11 부터 8)와 비트(15 부터 12)를 각각 제공한다. 상기 방법에서, 외부개발시스템(7)은 다른 어드레싱을 갖는 분기 연산의 현재기록과 실시간에서 간접적으로 결정된 번지 위치를 둘다 관찰 할 수 있다.

다수의 다중화장치(80)는 PST 및 DDATA 신호와 같이 내부 PST 및 내부 DDATA 신호를 각각 외부개발시스템(7)에 제공한다. PST 및 DDATA 신호는 상세히 도시되지 않은 집적회로 핀을 경유하여 제공된다. 게다가 집적회로 핀을 경유하여 제공된 PST 및 DDATA 신호는 내부 PST 및 내부 DDATA 신호 각각을 매개로하여 정보가 전송된 원 타이밍 사이클 후에 정보를 전달한다.

상기 실시간 관찰을 허용함으로서, 외부개발시스템(7)은 고장을 더 신속히, 더 정확히 식별하고 분리 할 수 있다. 더욱이, 실시간 추적관찰이 외부 개발시스템에 제공되는 반면에, 데이타 처리장치(3)의 CPU(2)는 명령의 실행을 계속하고 통상 DDATA 및 PST 신호의 제공에 의해 영향을 받지 않는다.

더욱이, 상기에서 제공된 예는 JMP 명령을 실행키 위한 트레이스 함수의 사용을 기술했는데, 본 발명은 데이타 처리장치(3)의 동적인 실행 경로(path)를 완전히 추적하는데 사용될 수 있다. JMP 명령과 더불어 나타난 바와 같이, 동적인 실행 경로의 추적은 흐름 연산의 변경에 의해 복잡하게 된다. 예를들어 몇몇 분기 명령은 목표 명령 어드레스의 계산이 상대 또는 절대 PC(프로그램 카운터)가 아닌 프로그램 가시 가능 레지스터의 사용을 포함하는 다른 번지지정(addressing)을 사용하여 실현될 수 있다.

다른 번지지정을 이용하는 명령을 위해, 본 발명은 목표 어드레스가 정확히 결정되고 외부개발시스템(7)에 제공될 수 있게 특정 순서대로 DDATA 및 PST 신호를 제공한다. 첫째로, 선택된 분기 명령은 다수의 집적회로핀을 경유하여 전송된 PST 신호에 의해 식별된다. 도 10의 부호화에서 나타난 바와 같이, PST 신호는 선택된 분기 명령이, PST 신호가 $5의 값을 갖는 경우, 실행되고 있다는 것을 나타낸다. 다음에, PST 신호는 목표 어드레스는 DDATA 신호상에 표시된다는 것을 나타내기 위해 사용될 수 있다. PST 신호의 코드화는 DDATA 신호를 매개로하여 표시되는 많은 바이트를 식별한다. 목표 어드레스는 그후 DDATA 신호를 사용하여 계속적인 타이밍 사이클 동안에 계속 유용 가능하다. DDATA 신호상에 표시된 목표 어드레스의 많은 비트는 외부사용자에게 제어되는 구성가능한 변수이다.

본 발명은 데이타 처리장치의 CPU(2)의 실행에 중요한 영향을 미치지않고 동적으로 실행되는 독특한 트레이스 함수를 제공한다.

본 발명에 의해 실현된 실시간 트레이스 함수에서 PST 신호에 의해 정의된 바와 같이 실행된 명령 흐름과 DDATA 신호에 의해 제공된 포착된 데이타의 가용성 사이에는 엄숙한 동기가 존재한다는 것을 주목하라. 여기에서 기술된 본 발명의 실싱케에서, DDATA 신호가 적절한 명령과 용이하게 연관될 수 있도록 엄격한 순서가 존재한다. 상기 순서는 다음의 예에서 매우 상세히 기술된다. 이후의 코드는 다음 의 예에서 실행된다는 점을 가정해라

명령어드레스	명령	
00001316	movq	1, d0
00001318	mov. 1	d0,(-4,a6)
0000131c	pea	(-68,a6)
00001320	pea	(-36, a6)
00001324	bsr.w	Func 2
0000115c	mov. 1	d7, -(a7)

상기 예에서, 만일 CSR이 두개의 바이트 분기 목표 어드레스 정보를 표시하도록 구성된다면, 만일 모든 오퍼랜드가 M-버스(23) 엑세스를 생성시킨다면 그리고 판독 및 기록 어퍼랜드 모굳 포착된다면 PST 신호 및 DDATA 신호 상의 결과축렬은 다음과 같이 나타난다.

PST	DDATA	기술
$1	----	1316에서 명령을 시작
$1	----	1318에서 명령을 시작
$b	----	명령 1318에 대해 오퍼랜드를 기입
at		
0	[3:0]	1318에서 명령에 대한 기록 어퍼랜드로부터의 비트
0	[7:4]	1318에서 명령에 대한 기록 어퍼랜드로부터의 비트
0	[11:8]	1318에서 명령에 대한 기록 어퍼랜드로부터의 비트
0	[15:12]	1318에서 명령에 대한 기록 어퍼랜드로부터의 비트
0	[19:16]	1318에서 명령에 대한 기록 어퍼랜드로부터의 비트
0	[23:20]	1318에서 명령에 대한 기록 어퍼랜드로부터의 비트
0	[27:24]	1318에서 명령에 대한 기록 어퍼랜드로부터의 비트
0	[31:28]	1318에서 명령에 대한 기록 어퍼랜드로부터의 비트
$1	----	131c에서 명령을 시작
$b	----	명령 131c에 대해 오퍼랜드를 기입
at		
0	[3:0]	131c에서 명령에 대한 기록 오퍼랜드로부터의 비트
0	[7:4]	131c에서 명령에 대한 기록 오퍼랜드로부터의 비트
0	[11:8]	131c에서 명령에 대한 기록 오퍼랜드로부터의 비트
0	[15:12]	131c에서 명령에 대한 기록 오퍼랜드로부터의 비트
0	[19:16]	131c에서 명령에 대한 기록 오퍼랜드로부터의 비트

0	[23:20]	131c에서 명령에 대한 기록 오퍼랜드로부터의 비트
0	[27:24]	131c에서 명령에 대한 기록 오퍼랜드로부터의 비트
0	[31:28]	131c에서 명령에 대한 기록 오퍼랜드로부터의 비트
$1	----	1320에서 명령을 시작
$b	----	명령 1320에 대해 오퍼래느를 기록
at		
0	[3:0]	1320에서 명령에 대한 기록 오퍼랜드로부터의 비트
0	[7:4]	1320에서 명령에 대한 기록 오퍼랜드로부터의 비트
0	[11:8]	1320에서 명령에 대한 기록 오퍼랜드로부터의 비트
0	[15:12]	1320에서 명령에 대한 기록 오퍼랜드로부터의 비트
0	[19:16]	1320에서 명령에 대한 기록 오퍼랜드로부터의 비트
0	[23:20]	1320에서 명령에 대한 기록 오퍼랜드로부터의 비트
0	[27:24]	1320에서 명령에 대한 기록 오퍼랜드로부터의 비트
0	[31:28]	1320에서 명령에 대한 기록 오퍼랜드로부터의 비트
$5	----	선택된 분기 명령 1324를 시작
at		

$9 ---- 1324에 대한 목표 어드레스

instruction at

0 $c 목표 어드레스(115c)로 부터의 비트

0 $5 목표 어드레스(115c)로 부터의 비트

0 $1 목표 어드레스(115c)로 부터의 비트

0 $1 목표 어드레스(115c)로 부터의 비트

$b ---- 1324에서 명령에 대해 오퍼랜드를 기옥

0 $8 1324에서 명령에 대한 복귀 어드레스로부터의 비트

0 $2 1324에서 명령에 대한 복귀 어드레스로부터의 비트

0 $3 1324에서 명령에 대한 복귀 어드레스로부터의 비트

0 $1 1324에서 명령에 대한 복귀 어드레스로부터의 비트

0 $0 1324에서 명령에 대한 복귀 어드레스로부터의 비트

0 $0 1324에서 명령에 대한 복귀 어드레스로부터의 비트

0 $0 1324에서 명령에 대한 복귀 어드레스로부터의 비트

0 $0 1324에서 명령에 대한 복귀 어드레스로부터의 비트

$1 ---- 115c에서 명령을 시작

상기에서 제공된 예에서, PST 및 DDATA 신호는, 동기상태를 유지하여 DDATA 신호상에 표시된 소정의 오퍼랜드는 항상 소정의 명령과 연관된다는 것이 보장될 수 있도록 동기되어 유지된다. 설명의 명확화를 위해, 상기 예는 DDATA 신호와 비-제로 PST값 사이의 오버랩을 나타내지 않았다. 따라서, 상기 오버랩은 발생할 수 있으나 PST 및 DDATA 신호 사이의 관계는 침해받지 않는다.

실시간 디버그 지원

실시간 트레이스 함수외에, 본 발명은 또한 실시간 디버그 함수를 실행키 위한 회로와 방법을 제공한다. 많은 데이타 처리장치, 특히 내장 시스템에서 데이타 처리장치는 데이타 처리장치가 더이상 정상적인 모드에서 작동하지 않는 경우 디버그 연산중에 시스템 제약 조건 또는 고장난 회로 또는 프로그램을 수정할 수 없음에 기인하여 디버그 연산중에 중지되지 않을 수 있다. 본 발명은 데이타 처리 장치를 정지시키지 않고 데이타 처리장치의 실시간 연산을 최소한도로만 방해 함으로써 디버그 연산을 실행한다.

브레이크포인트 연산은 전형적으로 다버깅 연산중에 사용되어 프로그래밍 사상을 식별한다. 프로그래밍 사상이 출회(encounter)된 경우, 브레이크포인트 신호는 표명되고 데이타는 검색되어 소프트웨어 프로그램의 상태를 결정한다. 트레이스 함수와 마찬가지로 데이타 처리장치 에러가 식별될 수 있도록 브레이크포인트 연산은 외부의 사용자에게 다수의 선택된 레지스터 및 메모리 위치 각각의 상태를 확인하게 해준다.

본 발명은 다수이 하드웨어 브레이크포인트 레지스터(50)와 브레이크포인트 회로(100)를 제공하여 상기 브레이크포인트 연산을 수행한다. 본 발명에서, 프로그 램 카운터 또는 프로그램 카운터 마스크의 값, 오퍼랜드 어드레스 범위, 그리고 데이타 값 또는 데이타 마스크 값은 모두 실현된다. 게다가, 본 발명에서 실현된 브레이크포인트 연산은 1 또

는 2 레벨 크리거로 또한 구성될 수 있고, 여기서 정확한 트리거 반응은 다수의 제어 레지스터(40)의 TDR (트리거 정의 레지스터)를 매개로하여 프로그램 가능하다. 본 발명을 사용하는 브레이크포인트 연산의 실행은 계속하여 매우 상세히 기술될 것이다.

다수의 하드웨어 브레이크포인트 레지스터(50) 각각은 외부개발시스템(7) 또는 데이타 처리장치(3)의 CPU(2)에 의해 기록될 수 있다. 도 3에서 나타난 바와같이 어드레스 브레이크포인트 레지스터(ABLTR과 ABHR)는 브레이크포인트 함수를 트리거 하기 위헤 사용될 수 있는 데이타 처리 시스템(5)의 오퍼렌드 논리 어드레스 공간내의 영역을 정의한다. 도 6에서 매우 상세히 나타난 AABR(Address Attribute Breakpoint Registor)은 브레이크포인트 함수를 위한 트리거 내에서 일치되도록 다수의 어드레스 속성과 마스크를 정의한다. ABLR과 ABLR 어드레스 값처럼 AABR에 기억된 속성값은 K-버스(25)를 경유하여 전송된 어드레스 속성 신호와 비교된다.

상기 다수의 하드웨어 브레이크포인트 레지스터(50)는 또한 다수의 프로그램 카운터 브레이크포인트 레지스터를 포함한다. 다수의 프로그램 카운터 브레이크포인트 레지스터는 브레이크 함수를 트리거 하기 위해 사용될 수 있는 데이타 처리 시스템(5)의 명령 논리 어드레스 스페이스 내의 영역을 정의한다. PBR(Program Countor Breakpoint) 값은 PBMR(Program Countor Breakpoint Mask) 레지스터 내에 저장된 값에 의해 마스크 될 수 있다. 본 발명의 한 실시예에서, PBMR에서 저장된 대응하는 제로값을 갖는 PBR내의 비트만이 데이타 처리 시스템(5)의 프로그램 카운터 값과의 비교 연산에 사용된다. 프로그램 카운터 값은 데이타 처리 시스템(5)의 CPU(2)의 여기서 나타나지 않은 프로그램 카운터 레지스터에 저장된다는 것을 주목해라.

프로그램 카운터 브레이크포인트 레지스터 외에, 다수의 하드웨어 브레이크포인트 레지스터(50)는 다수의 데이타 브레이크포인트 레지스터를 포함한다. 다수의 데이타 브레이크포인트 레지스터는 DBR(Data Breakpoint Registor)과 DBMR(Data Breakpoint Mask Registor)을 포함한다. 상기 DBR은 데이타 처리 시스템(5)이 디버그 연산을 실행하고 있는 경우 트리거를 형성키 위해 사용될 수 있는 특정 데이타 패턴을 정의하는 데이타 값을 저장한다. DBMR에 저장된 데이타 마스크 값은 DBR에 저장된 데이타 값을 마스크한다. 따라서 본 발명의 한 실시예에서 DBMR에 저장된 대응하는 제로 값을 갖는 DBR에 저장된 비트만이 K-버스(25)를 경유하여 전송된 데이타 신호와 비교될 수 있어 트리거 조건이 충족되었는지를 결정한다.

다수의 하드웨어 브레이크포인트 레지스터(50) 각각은 다수의 제어 레지스터(40)의 TDR을 사용하는 트리거에 응답하게 다르게 구성될 수 있다. TDR에 의해 표시된 구성에 관계없이 트리거된 브레이크포인트의 표시는 DDATA 신호가 포착된 오퍼랜드와 어드레스를 표시하지 않을때는 언제든지 DDATA 신호상에 제공된다. DDATA 신호의 출력은 도 9에 나타나 있다.

도 9는 DDATA 신호의 비트(3부터 0)를 매개로 하여 제공된 값과 브레이크포인트 상태 사이의 관계를 나타난다. CSR의 비트(31 부터 28)는 DDATA 신호의 비트(3부터 0)에 대응한다.

DDATA 신호가 $0의 값을 갖는 경우, 브레이크포인트는 사용 가능해진다. DDATA 신호가 $1의 논리 값을 제공하면 디버그 모듈(10)은 레벨 1 브레이크포인트를 기다려 트리거 된다. 이전에 논의한 바와 같이, 레벨 1 브레이크포인트의 속성은 TDR(Trigger Definition Register)의 비트(16부터 0)에 의해 정의된다. 만일 $2의 논리값이 DDATA 신호에 의해 제공되면, 레벨 1 브레이크포인트는 트리거 되었다. DDATA 신호가 $5의 논리값을 제공하는 경우, 디버그 모듈(10)은 레벨 2 브레이크포인트를 기다려 크리거 된다. 이전에 논의 되었듯이, 레벨 2 브레이크포인트의 속성은 TDR(Trigger Definition Register)의 비트(31 부터 16)에 의해 정의된다. 만일 $6의 논리값이 DDATA 신호에 의해 제공된다면, 레벨 2 브레이크포인트는 트리거 된다.

브레이크포인트 연산의 상태는 또한 CSR의 비트(31 부터 28)내에 기억된다.

디버그 연산의 실행 이전에, 브레이크포인트 값은 다수의 하드웨어 브레이크포인트 레지스터(56)의 적절한 하나에 기억된다 브레이크포인트 값은 직렬 인터패이스(30)를 통하는 외부개발시스템(7) 또는 K-버스(25)를 통하는 CPU(2)에 의해 제공될 수 있다.

외부 장치로부터의 상기 기록함수를 실행키 위해, 외부개발시스템(7)은 명령내에 자리 이동하여 데이타를 DSDI 및 DSCLK 신호를 사용하는 직렬 인터페이스(30)에 대한 다수의 하드웨어 브레이크포인트 레지스터(50)의 선택된 하나 속으로 로드시킨다. 본 발명의 한 실시예에서, 커맨드의 제 1 부분은 DSDI 신호에 의해 제공되 고 DSCLK 신호에 의해 클록인(clock in) 된다. 커맨드의 추가부분은 계속하여 제공된다. 커맨드의 모든 부분은 16 비트 패킷 정보가 완비될 때까지 레지스터(32)에 기억된다. 본 발명의 또다른 실시예는 본 발명의 본 실시예에서 소요되는 16 비트 보다 많거나 적은 비트를 포함할 수 있다는 것을 주목하라.

본 발명의 또다른 실시예에서, 브레이크포인트 값은 데이타 처리장치(3)의 CPU(2)에 의해 사전에 선택된 다수의 브레이크포인트 레지스터(50)중의 하나 속으로 기록된 값에 의해 결정될 수 있다. CPU(2)가 사전에 선택된 다수의 브레이크포인트 레지스터(50) 중의 하나에 값을 기록하는 경우, 상기 값은 도 3에 나타난 브레이크포인트 회로(100)의 임시 레지스터(170)에 제공된다. 상기 값은 K-버스(25)를 매개로 임시 레지스터(170)에 제공된다. 임시 레지스터(170)로부터, 상기 값은 16 비트 패킷내의 레지스터(32)에 전송 및 기억된다. 사전에 언급된 바와같이, CSR로의 CPU 기록에 대해, KDDATA 신호를 매개체로하여 전송된 오퍼랜드 값은 CPU(2)에 의해 제어버스(15)를 경유하여 제어회로(20)에 제공된 제어신호를 사용하여 임시 데이타 레지스터(170)에 도착된다. 따라서 다수의 제어 레지스터(40)에, 브레이크포인트 값은 외부개발시스템(7)으로 부터의 레지스터에 제공된 것과 동일하게 보인다.

일단 브레이크포인트 값이 사전에 선택된 다수의 브레이크포인트 레지스터(50)의 하나에 기록되면, CPU(2)는 정상적인 연산모드로 프로그래밍 코드를 실행한다. 상기 프로그래밍 코드의 실행중에, 어드레스 정보는 K- 버스(25)의 KDDR 신호를 매개로하여 전달되고 데이타 정보는 K-버스(25)의 KDATA 신호를 매개 로 하여 전달된다. 게다가 CPU(2)의 프로그램 카운터정보는 프로그램 카운터 브레이크포인트 비교연산을 위해 임시 PC 레지스터(125)에 기억된 값을 연속하여 갱신한다.

어드레스와 같이, 데이타 및 프로그램 카운터 정보는 하드웨어 회로(100)에 전송되고, 비교기(130, 140, 180, 그리고 135)는 정보를 수신하고 데이타 처리장치(3)의 실행에 역 효과를 미치지 않고 비교 연산을 선택적으로 실행한다. 더욱이, 다수의 제어 값은 제어버스(15)를 경유하여 다수의 제어 레지스터(50)의 TDR로부터 하드웨어 회로(100)에 제공된다. 다수의 제어값은 하드웨어 회로(100)를 사용하는 트리거 연산을 선택적으로 구성하는 TDR의 사전 기술된 값을 포함한다.

예를들어, 외부개발시스템(7)이 브레이크포인트 어드레스가 어드레스의 어떤 범위내에 내려가야 하는 경우에 표명되어야 한다는 것을 결정한 경우, 외부개발시스템(7)은 직렬 인터페이스(30)를 사용하여 상위 어드레스 값을 도 3의 ABHR(110)로 기록하여야 하고, 하위 어드레스 값을 도 3의 ABLR(120)에 기록하여야 한다. 그러나 비교기(130)와 비교기(140)는 K-버스(25)의 KADDR 신호에 의해 제공된 어드레스가 ABLR(120)과 ABHR(110)에 기억된 어드레스에 의해 정의된 범위 내인지를 결정할 수 있기 전에, 다수의 제어 레지스터(40)의 TDR은 제어버스(15)를 경유하여 적절한 제어를 제공해야 한다.

상기예에서, 적절한 예는 표명된 EAR(Enable Address Range) 비트 또는 표명된 EAI(Enable Address Breakpoint Inverted) 비트를 포함한다. 이전에 논의된 바와 같이, EAR 비트는 상기 비트가 표명된 경우 ABLR(120)과 ABHR(110)에 의해 정의 된 포하적 범위상에 브레이크포인트가 트리거 되게 한다. 마찬가지로, 표명된 경우, EAI 비트는 브레이크포인트가 ABLR(120)과 ABHR(110)에 의해 정의된 범위를 제외하거나 외부에 있는 어드레스 범위 상에 트리거 되게 한다.

EAR 및 EAI 비트의 논리값에 따라, 비교기(130, 140)는 KADDR 신호를 매개로 하여 전송된 어드레스 값이 ABHR(110)과 ABLR(120)에서 연관된 값보다 크거나 작은지를 결정할 것이다. 만일 EAR 비트가 표명되고 KADDR 신호를 매개로하여 전송된 어드레스 값이 ABHR(110)과 ABLR(120)에서의 어드레스 값에 의해 정의된 범위내이면, 비교기(130, 140)는 브레이크포인트 신호를 표명하여 사전에 선택된 트리거포인트는 엑세스 된다는 것을 표시한다. 브레이크포인트 신호가 표명된 경우, 제어회로(20)는 코어(9)의 CPU(2)에 BKPT(Breakpoint) 신호, 디버그 이터럽트 신호, 그리고 다중-전도체 브레이크포인트 제어 신호를 매개로 하여 전송된 Go 신호의 하나를 표명한다.

브레이크포인트 신호가 표명되면, 브레이크포인트 트리거의 발생은 TDR에서 정의된 바와같은 응답을 발생시킨다. 만일 데이타 처리장치(3)가 정지될 수 있다면, BKPT 신호의 표명은 CPU(2)가 백그라운드 디버그 모드내의 실행을 정지하게 강제할 수 있다. 백그라운드 디버그 모드는 계속하여 약간 더 상세히 논의 될 것 이라는 점을 주목하라. 백그라운드 디버그 방식으로 들어가기 위해서 TDR의 TRC 비트는 %01과 동등해야 한다. TRC 비트는 TDR의 비트(31, 30)에 대응한다. 백그라운드 디버그 모드가 들어가고, 프로그램 카운터 값에 근거한 브레이크포인트가 표명되면 CPU(2)는 목표화된 명령이 실행되기 전에 데이타 처리 장치(3)의 실행을 정지시킨 다. 어드레스에 근거한 브레이크포인트와 데이타 비교를 위해, CPU(2)는 정지전에 몇몇 추가명령을 실행할 수 있다. 상기 유형의 브레이크포인트 연산은 데이타 처리장치(3)가 트리거에 대응하여 정지되므로 비 실시간 디버그 연산을 위해 단지 사용되어야 한다는 점을 주목하라.

그라나, 만일 데이타 처리장치(3)가 정지될 수 없으며, 제어회로(20)는 디버그 인터럽트 신호를 표명하여 특정 디버그 인터럽트는 CPU(2)에 제공되어야 한다는 것을 나타낸다. 특정 디버그 인터럽트가 CPU(2)에 제공되어야 하는 경우, TDR의 TRC 비트는 %10의 논리 값에 설정된다. TRC 비트가 %10에 설정되면, 브레이크포인트 트리거는 제어회로(20)에 의해 CPU(2)으로의 특정 디버그 인터럽트로 변환된다. 특정 디버그 인터럽트는 CPU(2)가 인터럽트에 대해 샘플링 할때까지 미정 상태가 된다. CPU(2)는 각 명령당 한번 샘플링 한다. 특정 디버그 인터럽트가 표명된 경우 프로그램 카누터 값에 근거한 브레이크포인트는 목표한 명령이 실행되기 전에 발생한다. 상기는 프로그램 카운터 비교 연산 CPU(2)가 인터럽트가 미정인지를 결정키 위해 샘플링하는 것과 동시에 사용 가능해지므로 가능하다. 어드레스에 근거한 브레이크포인트와 데이타 비교를 위해 CPU(2)는 인터럽트 연산을 시행하기 전에 몇몇 추가 명령을 실행할 수 있다.

특정 디버그 인터럽트 CPU(2)에 의해 인식되는 경우, CPU(2)는 소프트웨어 프로그램이 실행을 중지하고 예외 처리 루틴을 개시한다. 예외 처리 루틴의 개시할때에, CPU(2)는 에뮬레이터 연산모드로 들어간다. 모든 인터럽트는 CPU(2)가 에뮬레이터 연산모드에서 작동하고 있는 경우 무시된다는 점을 주목하라. 다수의 제어 레지스터(40)의 CSR 내의 MAP 비트의 상태에 따라서, 에뮬레이션 연산모드는 모든 메모리 엑세스를 특별히 사상되고, 다수의 하드웨어 브레이크포인트 레지스터(50)의 AABR 레지스터의 TT 및 TM 값에 의해 표시되어 배열된 어드레스 속으로 강제한다. 상기 예에서, TT 값은 $2에 동등하고 TM 값은 $5(오퍼랜드) 또는 $6(명령)에 동등하다.

표준 8바이트 에외 스택(Stack)이 생성된 후, CPU(2)는 독특한 예의 벡터를 인출한다. CPU(2)는 상기 독특한 예외 벡터내에 포함된 명령 어드레스에서 실행을 계속한다. CPU(2)내의 디버그 인터럽트 조정기는 감시 프로그램 명령 세트를 사용하여 다수의 필요한 저장(Save) 연산을 수행하기 위해 프로그램 될 수 있다. 예로서 디버그 인터럽트 조정기는 모든 프로그램 가시 레지스터 및 특정 메모리 위치 어느것의 상태를 확보된 메모리 공간에 저장할 수 있다. 상기 확보된 메모리 공간은 메모리(6)에 위치될 수 있고 상기는 데이타 처리 시스템(5)의 어느 다른 메모리에 위치될 수 있다. 일단 소요된 연산이 완결되면, RTE(Return From Exception) 명령은 실행되고 CPU(2)는 에뮬레이터 연산 모드에서 나온다. CPU(2)가 디버그 인터럽트 신호를 수신하고 에뮬레이터 연산모드를 입력한 경우, PST 신호는 상기 조건에 신호를 보내기 위한 $D의 값을 갖는다. PST 신호는 디버그 인터럽트 예외 처리 루틴이 인출될 때까지 디버그 인터럽트가 처리되는 시간으로 부터 상기 값을 갖는다. 게다가, PST 신호는 $7의 값을 갖으며 CPU(2)는 에뮬레이터 연산모드로 부터 나온다는 것을 표시한다. 더욱이, 일단 디버그 인터럽트 조정기가 디버그 예외 처리 루틴이 상기 실행을 완결하면, 외부개발시스템(7)은 그후 확보된 메모리 위치를 포함하는 데이타 처리 시스템(5)내의 다른 메모리 또는 메모리(6)내의 확보된 메모리 위치로 엑세스 할 수 있다.

디버그 인터럽트는 정규의 데이타 처리장치(3)의 연산내의 사소한 관입이 용인될 수 있는 실시간 디버그 연산에 사용될 수 있으나, 데이타 처리장치(3)를 정지시키는 것을 선택할 수 있는 사항이 아니다. 데이타 처리 장치의 에뮬레이터 모드의 입구와 출구는 PST 신호상의 독특한 값에 의해 신호가 가해진다.

게다가, 범위 보다는 단하나의 어드레스, 데이타 값에 근거한 또는 데이타 값의 마스크, 그리고 프로그램 카운터 값 또는 프로그램 카운터 값의 마스크에 근거한 다른 비교는 본 발명에서 실현될 수 있다. 상기 예에서 기술된 어드레스 범위 비교에서와 같이, 데이타 값, 데이타 값의 마스크 프로그램 카운터 값, 그리고 프로그램 카운터 값의 마스크에 근거한 비교는 TDR을 필요로하여 제어 버스(15)를 매개로 하여 적절한 제어를 제공한다.

다중-레벨 트리거 구조

본 발명예에 의해 실행된 실시간 디버그 연산은 확장될 수 있어 2 레벨이 트리거 값을 제공한다. 본 발명의 본 실시예에서, 비트(31 부터 16)는 제 2 레벨 트리거를 정의하고 비트(15 부터 0)는 제 1 레벨 트리거를 정의한다. 트리거가 2 레벨 트리거로서 구성된 경우 트리거는 조건부 'if-then-if' 구조와 유사한 방법으로 작용한다. 예를들어, 트리거의 제 1 조건 또는 레벨이 충족되면, 제 2 조건 또는 레벨은, 트리거가 브레이크포인트 함수를 데이타처리 시스템(5)에 의해 실행되는 것이 가능하게 하기 전에, 만족되어야 한다. 상기 조건부 트리거 구조는 외부개발 시스템(7)이 증가된 유연성과 정밀도를 가진채 작동하게 허용한다. 본 발명의 본 실시예에서 트리거 구조의 단지 2레벨이 실현된다. 그러나 트리거 구조의 추가 레벨이 실현될 수도 있다.

2레벨 트리거 연산의 예와같이, 외부개발시스템(7)은 프로그램 카운터가 $DEADBEEF의 값과 동등한 이후에 사용자 모드 오퍼랜드 워드 기록의 값 어느것으로서 트리거를 어드레스 $0000FACE에 정의하고자 한다. 제 1 레벨 트리거는 $FEED 이외의 사용자 어드레스 $0000FACE에 모드 오퍼랜드 워드 기록의 어느 값이 될 것이다. 제 2 레벨 트리거는 프로그램 카운터가 $DEADBEEF의 값에 동등한 경우에 발생한다.

상기 연산을 실행키 위해, 외부개발시스템(7)은 프로그램 카운터 브레이크포인트 레지스터(PBR, 115)에 $DEADBEEF 값을, PBMR(Program Counter Breakpoint Mask Register, 190)에 $000000000 값을 기록한 것이다. 모든 제로가 PBMR(190)에 기록되면, PBR(115)에 기억된 모든 비트는 임시 PC 레지스터(125)에 의해 제공된 프로그램 카운터 값에 비교된다. 비교 연산은 다수의 제어 레지스터(40)의 TDR에 기억된 값에 대응하여 실행된다는 점을 주목하라.

외부개발시스템(7)은 $0000FACE 값을 어드레스 브레이크포인트 하위 레지스터(ABLR)에 $0041 값을 도 6의 어드레스 속성 브레이크포인트 레지스터(AABR)에 또한 기록해야 한다. ABLR(120)내의 $0000FACE의 값은 TDR의 구성이 비교가 발생하도록 허용한다면, 비교기(140)는 브레이크포인트 신호를 표명할 수 있는 값을 나타낸다. AABR내의 $0041 값은 TM 값을 %001에, TT 값을 %00에, SZ 값을 %10에, 그리고 R(Read) 비트를 0에 세트한다. TM과 TT 값은 트리거는 사용자 모드 오퍼랜드 워드 기록 연산이 될 것이라는 점을 확실히 하는데 소용된다.

외부개발시스템(7)은 브레이크포인트 회로(100)의 DBR(data breakpoint register, 160)에 $0000FEED 값을 기록하여야 한다. 외부개발시스템(7)은 DBMR(data breakpoint mask register, 150)에 $000000000의 값을 또한 기록하여야 한다. 모든 제로가 DBMR(150)에 기록된 경우, DBR(160)에 기억된 모든 비는는 비교기(180)를 사용하여 KDATA 신호에 의해 제공된 데이타 값과 비교된다. 비교 연산은 다수의 제어 레지스터(40)의 TDR에 기억된 값에 대응하여 실행된다는 점을 주목해라.

마지막으로 외부개발시스템(7)은 $20022824의 값으로 다수의 제어 레지스터의 TDR을 기록하여야 한다. TDR이 상기 값을 갖는 경우 제 1 레벨 트리거는 다음과 같은 특성을 갖는다. 상기 예에서, EBL 비트는 표명되고, EDLW 비트는 부정되고, EDWL 비트는 표명되고, 그리고 EDWU, EDLL, EDLM, EDUM, 그리고 EDUU 비트는 모두 부정된다. DI 비트는 표명되고, AI 및 EAR 비트는 부정되고, EAL 비트는 표명되고, 그리고 EPL 및 PCI 비트는 부정된다. 상기 구성에서, TDR은 제 1 레벨 트리거가 사용되게 구성하여 DBR(160)에 기억된 값과 동일하지 않게 기억된 워드 길이 데이타 값을 위한 브레이크포인트 신호를 ABLR에 기억된 어드레스 값이 KADDR 신호에 의해 제공된 값에 대응하는 경우, 제 1 레벨 트리거를 표명하게 구성된다. TDR에 의해 결정된 상기 조건 각각이 만족되면, 제 1 레벨 트리거는 표명되고 그후 제 2 레벨 트리거는 브레이크포인트 신호가 표명될 수 있기 전에 만족되어야 한다.

더욱이, TDR이 상기 값을 갖는 경우, 제 2 레벨 트리거는 다음과 같은 특성을 갖는다. 상기 예에서, TRC 값은 %00이고, EBL 비트는 표명되고 EDLW, EDWL, EDWU, EDLL, EDLM, EDUM 그리고 EDUU 비튼 모두 부정된다. DI, AI, EAR 그리고 EAL 비트는 부정되고, EPC 비트는 표명되고, PIC 비트는 부정된다. 상기 구성과 더불어, TDR은 제 2 베렐 트리거가 PBR(115)에 기억된 것과 동일한 프로그램 카운터 값에 대해 브레이크포인트 신호를 표명하게 사용되어 지도록 구성한다. TDR에 의해 결정된 상기 조건 각각이 만족되어지는 경우, 제 2 레벨 트리거는 표명되고 브레이크포인트 신호는 표명될 수 있다.

이전에 언급한 바와같이, 브레이크포인트 신호의 표명은 결과적으로 프로그램가능한 트리거 반응이 될 수 있다. 트리거 반응은 다수의 제어 레지스터(40)의 TDR의 비트(31, 30)속으로 프로그램 된다 올바르게 프로그램되면, 데이타 처리장치(3)의 CPU(2)의 트리거 반응은 DDATA 신호를 매개로하여 외부개발시스템에 트리거를 가시적이게 만들 수 있고 내부 반응을 야기하지 않는다. 제 2 트리거 반응은 CPU(2)가 실행을 정지하게 강제하고, 외부에 CPU(2)가 PST 신호를 매개로 하여 정지된 상태가 된다는 점을 표시하는 것을 강제한다. 제 3 트리거 반응은 CPU(2)에 디버그 인터럽트를 생성케 할 수 있다. 이전에 언급한 바와같이, 디버그 인터럽트는 모든 다른 인터럽트 보다 우선 순위에서 보다 상위이고 CPU(2)가 에뮬레이터 연산 모드로 들어가게 하고 독특한 예외 처리 루틴에 엑세스하게 한다. 에뮬레이터 연산 모드의 입구와 축구는 PST 출력부 상에 외부에 가시적이게 만들어 진다.

디버그 모듈과 코어의 동시 연산

상기에 기숙된 실시간 디버그 함수를 지원하기 위해, 본 발명은 코어(9)와 디버그 모듈(10)의 동시 연산을 지원한다. 디버그 모듈(10)에 대한 K-버스(25) 상의 버스 사이클을 효과적으로 스케줄하는 중재 도표는 본 발명에서 실현되어

실시간 디버그 연산의 효율적인 실행을 고려한다. 중재 도표는 CPU(2)의 내부 명령 인출 파이프라인을 도입 설치함으로서 디버그 모듈(10)에 대한 버스 사이클을 스케줄 한다. 중재 도표는 모든 오퍼랜드 요구가 K-버스(25)에 디버그 모듈(10) 엑세스를 승인하기 전에 CPU(2)에 의해 처리된다. 디버그 모듈(10)은 버스 승인 신호를 수신하고 ABHR(110)에 기억된 소요의 어드레스와 도 6의 AABR에 기억된 속성으로 K-버스(25)를 구동한다. 판독 요구에 대해, 반응은 KDATA 신호상으로 구동되어 지고 임시 데이타 레지스터(170)에서 디버그 모듈(10)에 의해 포착된다. 기록 참조에 대해, 기록 오퍼랜드는 DBR(160)로 부터 KDDATA 신호상으로 구동되어 진다. 디버그 모듈(10)은 K-버스(25)를 CPU(2)에 되돌려 포기 하기 전에 하나의 버스 트랜젝션(transaction)에서 상기 연산을 실행한다. 따라서, 디버그 모듈(10)은 CPU(2)의 실시간 연산 행동으로의 관입을 축소시킨다. K-버스(25)에 대한 중재 도표는 CPU(2)에 기억된 논리에 의해 실행된다.

CPU(2)와 디버그 모듈(10)의 동시 연산을 나타내는 예에서, 외부개발시스템(7)은 디버그 함수중에 메모리 판독 연산을 실행하고 있다는 점을 가정하라. 메모리 판독 명령을 실행키 위해, 외부개발시스템(7)을 커맨드와 어드레스에서 자리 이동하여 DSDI 및 DSCLK 신호를 사용하여 직렬 인터페이트(30)에 메모 리 위치를 엑세스 한다. 본 발명의 한 실시예에서 커맨드의 제 1부분은 DSDI 신호에 의해 제공되고, DSCLK 신호에 의해 크록인 된다. 추가 커맨드 부분은 계속해서 제공된다. 모든 커맨드 부분은 16비트 패킷 정보가 완결될 때까지 레지스터(32)에 기억된다. 본 발명의 또 다른 실시예는 본 발명의 본 실시예에서 소요되는 16비트 보다 많거나 적은 비트를 포함한다.

여기에 기술된 본 발명의 실시예에서, 사전에 선택된 메모리 위치에 값을 기록하는 명령은 길이가 48비트이고 연산코드는 16비트이고 어드레스는 32비트이다. 레지스터(32)는 폭이 단지 16비트이다. 따라서 16비트 패킷의 정보가 완결되면 정보는 제어버스(15)를 경유하여 제어회로(20)에 전송된다. 제어회로(20)는 제 1 16비트 패킷의 정보를 해독하여 수행되어야 할 연산 유형을 결정한다. 일단 제어회로(20)가 제 1 16비트 패킷의 데이타를 해독하고 프로그램된 메모리 위치로부터 판독 엑세스는 실행되어야 한다는 것을 결정하면, 제어회로(20)는 제어버스(15)를 경유하여 제어신호를 제공하여 계속적인 32 비트의 어드레스 정보가 다수의 하드웨어 브레이크 포인트 레지스터(50)의 ABHR(110)에 로드되게 한다.

계속적인 데이타는 DSDI 신호를 매개로 하여 외부개발시스템(7)에 의해 직렬로 제공된다. 이전에 기술된 바와 같이, DSDI 신호에 의해 제공된 연속적인 데이타는 DSCLK 신호를 매개로 하여 클록인 된다. 데이타 비트의 각각은 레지스터(32)에 16비트 패킷의 정보가 완결될 때까지 기억된다.

제 2 16비트 패킷의 정보가 완결된 경우, 정보는 직렬 정보 신호를 매기로하여 다수의 하드웨어 브레이크포인트 레지스터(50)의 ABHR(110)에 전송된다. 계속하 여, 추가정보는 DSDI 및 DSCLK 신호를 매개로 하여 외부개발시스템(7)에 의해 레지스터(32)에 직렬로 제공된다. 제 3 16비트 패킷의 정보가 완결된 경우, 상기 정보는 직렬 정보 신호를 매개로하여 다수의 하드웨어 브레이크포이트 레지스터(50)의 ABHR에 전송된다.

오퍼랜드 및 어드레스 정보가 수신되는 경우 제어회로(20)은 버스 요구신호를 표명한다. 버스 요구 신호는 K-버스(25)를 매개로 하여 CPU(2)에 전송된다. 다시 버스 요구 및 버스 승인 신호는 디버그 모듈(10)과 CPU(2)사이의 직접 접속부이다. 버스 중재에 관한 상세는 이전에 설명되었었다.

버스 승인 신호가 표명되면 ABHR(110)은 사용되어 KADDR 신호를 매기로하여 기억된 어드레스 정보를 K-버스(25)에 제공한다. 어드레스 정보는 메로리(6)에 의해 해독되어 메모리 위치를 한독한다. 판독연산이 실행 되었으므로, 메모리(6)는 K-버스(25)의 KDATA 신호상의 메모리 위치에 대응하는 오퍼랜드를 출력한다. 접근된 메모리는 메모리(6)에 한정되지 않는다는 점을 주목하라. 오히려, 접근된 메모리는 데이타 처리 시스템(5)내의 메모리(6) 또는 다른 메모리 위치 일 수 있다. KDATA 신호를 매개로 전송된 오퍼랜드는 계속하여 브레이크포인트 회로(100)의 임시 데이타 레지스터(170)에서 포착된다. 오퍼랜드는 계속하여 내부 데이타 신호를 매개로하여 레지스터(32)에 전송된다. 오퍼랜드는 그후 직렬 인터페이트(30)에 의해 제공된 DSDO 신호를 매개로하여 32 타이밍 사이클에서 직렬로 시프트 아웃(shift-out)한다. DSDO 신호는 외부개발시스템(7)에 제공된다.

게다가 오퍼랜드가 KDATA 신호를 매개로하여 전송된 후, CPU(2)는 처리속도 에 최소로 관입해서 데이타 처리 명령의 실행을 다시 시작한다는 것을 주목하라. 따라서 CPU(2)는 직렬 인터페이스(30)가 정보를 출력하고 있는 동안 데이타 처리 함수를 실행할 수 있다.

본 발명은 특정 디버그 모드에서 데이타 처리장치(B)가 연산하는 것을 요구하지 않고도 CPU(2)와 디버그 모듈(10)의 동시 연산을 제공한다는 것을 주목하여야 한다. 데이타, 어드레스, 그리고 코어(9)와 디버그 모듈(10) 사이의 제어 정보를 전달하기 위해 K-버스(25)를 이용함으로서 디버그 모듈(10)은 CPU(2)와 같이 동일 메모리 위치와 내부 레지스터에 접근이 허용된다. 따라서, 디버그 모듈은 CPU(2)가 액세스 할 수 있는 모든 메모리 위치와 내부 레지스터에 액세스 한다. 동시에 CPU(2)와 디버그 모듈(10)을 작동시키는 가능성으로 인해 외부개발시스템이 처리장치의 연산에 중대히 영향을 끼치지 않고 데이타 처리장치(3)의 내부 레지스터 및 메모리에 가시성을 증가시키는 것이 허용된다.

게다가, 디버그 모듈(10)과 CPU(2)가 모두 동일 내부 레지스터 및 메모리 위치에 액세스 하는 가능성을 갖지만, CPU(2)와 코어(9)의 잔여 부분은 다수의 제어 레지스터(40)의 CSR내의 IPW(Inhibit Processor Writes to Debug Register) 비트가 세트된 경우, 다수이 브레이크포인트 레지스터(50)에 기억된 값을 변경하지 못할 수 있다. CSR내의 IPW 비트가 세트된 경우, CPU(2)에 의해 개시된 어느 기록 연산은 금지된다. 더욱이, IPW 비트는 외부개발시스템(7)에 의해 제공된 커맨드에 의해 단지 변경될 수 있다.

IPW 비트를 세트하기 위해, 외부개발시스템(7)은 커맨드를 시프트 인(shift in) 해서, DSDI 및 DSCLK 신호를 사용하여 직렬 인터페이스(30)로 향한 CSR에 데이타를 로드한다. 이전에 언급된 바와 같이, 본 발명의 한 실시예에서, 커맨드의 제 1 부분은 DSDI 신호에 의해 제공되고, DSCLK 신호에 의해서 클록 인 된다. 커맨트의 추가부분은 연속해서 제공된다. 커맨드의 모든 부분은 16비트 패킷의 정보가 완료될 때까지 레지스터(32)에 기억된다. 본 발명의 또다른 실시예는 본 발명의 본 실시예에서 소요되는 16비트 보다 많거나 적은 비트를 포함 할 수 있다.

여기에 기술된 본 발명의 실시예에서, CSR에 값을 기록하는 명령은 길이는 48비트이고, 연산코드는 16비트이고, 데이타는 32 비트이다. 레지스터(32)는 폭이 단지 16비트 이다. 상기 기록 명령을 실행키 위해 외부개발시스템(7)은 데이타를 CSR에 로드시키는 명령을 DSDI 및 DSCLK 신호 모두를 사용하여 직렬 인터페이스(30)에 시프트 인 한다. 커맨드의 제 1 부분은 DSDI 신호에 의해 제공되고, DSCLK 신호에 의해 클록 인 된다. 커맨드의 추가부분은 계속하여 제공된다. 커맨드의 모든 부분은 16비트 패킷의 정보가 완료될 때까지 레지스터(32)에 기억된다.
여기서 기술된 본 발명의 실시예에서, 값을 CSR에 기록하는 명령은 길이가 48비트이고, 연산 코드는 16비트 이고 데이타는 32 비트이다. 레지스터(32)는 폭이 단지 16비트이다. 따라서, 제 1 16비트 패킷의 정보가 완료된 경우, 정보는 제어회로(20)에 전송되고 상기 회로에서 해독되어 계속적인 연산이 형성된다. 제 2 16비트 패킷의 정보가 레지스터(32)에 로드된 경우 정보는 직렬 정보 신호를 매개체로하여 다수의 제어 레지스터(40)애 전송된다. 제어회로(20)로부터 제어버스(15)에 의해 제공된 제어정보를 사용하여, 직렬 정보 신호의 내용은 목표 레지스터 즉 CSR의 상부 부분에 로드된다. 유사한 방법으로 제 3 16비트 패킷의 정보는 레지스터(32)에 로드되고 그후 다수의 제어 레지스터(40)에 전송된다. 다시, 제어회로(20)에서 나온 신호를 사용하여, 직렬 정보 신호의 내용은 CSR의 하부부분에 로드된다. 상기 방법에서, 모든 48비트 명령은 디버그 모듈(10)에 직렬로 시프트 인 되고, 상기 디버그 모듈에서 상기 명령은 해독되어 오퍼랜드 값은 CSR에 로드된다.
연속적인 데이타는 DSDI 신호를 매개로하여 외부개발시스템에 의해 직렬로 제공된다. 이전에 기술된 바와 같이, DSDI 신호에 의해 제공된 연속적인 데이타 비트 각각은 16비트 패킷의 정보가 완료될 때까지 레지스터(32)에 기억된다. 제 2 16비트 패킷의 정보가 완료된 경우, 정보는 직렬 정보 신호를 매개로하여 다수의 제어 레지스터(40)에 전송된다. 제 2 16비트 패킷의 정보는 비트(31 내지 16)와 같이 CSR에 기억된다. 계속하여, 추가정보는 DSDI 및 DSCLK 신호를 매개로하여 레지스터(32)에 외부개발시스템(7)에 의해 직렬로 제공된다. 제 3 16비트 패킷의 정보가 완료된 경우, 정보는 직렬 정보 신호를 매개로 하여 다수의 제어 레지스터(40)에 전송된다. 제 3 16비트 패킷의 정보는 비트(15 내지 0)와 같이 CSR에 기억된다.
본 발명의 또다른 실시예에서, IPW 비트 이외의 모든 값은 데이타 처리 장치(4)의 CPU(2)에 의해 CSR에 기록된 값에 의해 형성될 수 있다. CPU(2)가 값을 CSR에 기록한 경우, 값은 도 6에 나타난 브레이크포인트 회로(100)의 임시 레지스 터(170)에 제공된다. 값은 K-버스(25)의 KDATA 신호를 매개로하여 임시 레지스터(170)에 제공된다. 임시 레지스터(170)로부터 값은 16비트 패킷 내의 레지스터(32)에 기억된다. 따라서, 다수의 제어 레지스터(40)에 대해, 값은 외부개발시스템(7)으로부터 레지스터(32)에 제공된 값과 동일하게 나타난다.
CSR에 기억된 IPW 비트와 다른 값은 제어버스(15)를 매개로하여 하드웨어 회로(100)에 제공된다. IPW 비트가 표명되고 CPU(2)가 ABHR(110), ABLR(120), DBMR(150), DBR(160), PBMR(190), 그리고 PBR(125)의 하나에 기억된 값을 변경하려고 시도하는 경우, IPW 비트는 상기 언급한 각각에 제공되어 코어(9)에 의해 개시된 기록 연산을 불능하게 한다.
다수의 하드웨어 브레이크포인트 레지스터(50)에 기록연산을 효과적으로 불능하게 하는 IPW 비트의 사용으로 인해, 데이타 처리장치(3)내의 고장난 회로와 장애를 일으킨 프로그램을 분리하고 식별하기 위해 외부개발시스템에 의해 프로그램된 조건은 CPU(2)에 의해 겹쳐 써지지 않을 것이라는 점이 확실해 진다.

리셋 예외 처리중의 디버그 연산

본 발명은 리셋 연산을 위한 예외 처리 루틴이 개시되기 전에 외부개발시스템(7)이 데이타 처리장치(3)에서 브레이크포인트 연산을 실행하게 하는 메카니즘을 제공한다. 외부 리셋 신호가 표명된 경우 다수의 제어 레지스터(40) 각각의 내용은 초기화 된다. 외부 리셋 신호가 계속하여 부정된 경우, 실제 리셋 예의 처리 루틴을 개시하기 전에 데이타 처리장치(3)가 정지하고 있는 시간의 윈도우가 존재한다. 만일 외부 브레이크포인트 신호 BKPT가 외부개발시스템(7)에 의한 정지 시간 동안 표명된다면, 본 발명은 목표 메모리 값을 데이타 처리 시스템(5)에 다운로드 시키고 어느 소요된 하드웨어 레지스터 구성을 실행하기 위한 메카니즘과 방법을 제공한다. 목표 메모리 값을 유일한 값인 수도 다수의 값일 수도 있다.
일단 목표 메모리 값이 외부개발시스템(7)에 의해 다룬로드 되면, 다수의 제어 레지스터(40) 및 다수의 하드웨어 브레이크포인트 레지스터(50) 둘다 또는 하나는 구성될 수 있다. 게다가, CPU(2), MMU/제어장치(4), 그리고 시스템 버스 제어장치(8)에 포함된 하드웨어 레지스터는 또한 프로그램 될 수 있다. 마지막으로, 데이타 처리장치(3)는 'Go' 커맨드를 사용하여 재시작 되어야 한다. 따라서, 본 발명에서, 'Go' 커맨드를 수신하자 마자 실행된 일련의 하드웨어 연산은 CPU(2)가 브레이크포인트 연산을 실행하고 있었던 동안에 변경되었던 하드웨어 레지스터의 함수이다. 만일, 외부개발시스템(7)이 CPU(2)의 CPU(2)의 프로그램 카운터 레지스터를 변경하지 않았다면, 'Go' 커맨드는 데이타 처리장치(3)가 정상적인 리셋 예외 처리 루틴을 실행하게 한다. 따라서, 만일 외부 개발 시스템이 CPU(2)의 프로그램 카운터 레지스터를 변경하지 않았다면, 'Go' 명령은 데이타 처리장치(3)가 정상적인 리셋 예외 처리루틴을 바이패스 하고 프로그램 카운터 레지스터에 로드된 어드레스에 의해 식별된 명령의 실행을 시작하게 한다.
상기 리셋 및 브레이크포인트 연산중에 데이타 처리장치(3)의 연산은 매우 상세히 계속하여 논의될 것이다. 리셋 신호는 표명되고 데이타 처리장치(3)의 CPU(2)에 제공된다는 점을 가정하라. 리셋 신호가 부정된 경우, 외부개발시스템(7)은 BKPT 신호를 표명한다. 디버그 모듈(10)의 제어회로(20)는 K-버스(25)를 경유하 여 다수의 브레이크포인트 제어 신호의 BKPT 신호를 표명한다. 다시, 제어 인터페이스는 더버그 모듈(10)과 CPU(2)의 직접 접속부라는 점을 주목하라.
CPU(2)가 표명된 BKPT 신호를 수신하는 경우, CPU(2)는 정지 시간 윈도우를 중지시키고 데이타 처리장치(3)는 정지된다는 것을 신호한다. 상기 동안에, PST 신호는 $F의 논리값을 가져 외부개발시스템에 데이타 처리장치(3)가 연산을 중지했다는 것을 표시해 준다. 따라서, CPU(2)와 데이타 처리장치(3)는 리셋 예외 처리 루틴을 시작하기 전에 정지된다.

일단 PST 신호가, 데이타 처리장치(3)가 연산을 중지했다는 것을 표시하면, 외부개발시스템(7)은 다수의 데이타, 어드레스, 그리고 제어 값을 DSDI 신호와 DSCLK 신호를 매개로하여 데이타 처리장치(3)에 다운로드한다. 다수의 데이타, 어드레스, 그리고 제어값은 프로그램의 메모리 화상 또는 메모리(6)에 기억된 데이타 값이 될 수 있다. 게다가, 다수의 데이타, 어드레스, 그리고 제어 값은 다수의 제어 레지스터(40)와 다수의 하드웨어 브레이크포인트 레지스터(50)와 같은 다수의 구성 레지스터 각각에 대해 다수의 초기값을 제공할 수 있다. 다수의 데이타, 어드레스, 그리고 제어값이 DSDI 신호를 매개로하여 데이타 처리장치(3)에 제공된 경우, 값의 각각의 비트는 직렬 인터페이스(30)를 매개로하여 레지스터(32)에 제공된다는 점을 주목하라. 레지스터(32)가 가득찬 경우, 패킷의 정보는 직렬 정보 신호를 매개로 하여 다수의 하드웨어 브레이크포인트 레지스터(50)에서, 패킷의 정보는 임시 데이타 레지스터(170)와 같은 레지스터에 기억된다. 임시 데이타 레지스터(170)는 그곳에 저장된 정보를 KDATA 신호를 매개로하여 K-버스(25)에 계속적으로 제공한다. K-버스(25)는 정보를 코어(9)의 잔여부분에 전송한다. 상기 연산이 매우 상세히 이전에 기술되었으므로 내부 기억 과정에 대한 추가적인 설명은 여기에 제공되지 않을 것이다.

다수의 데이타, 어드레스, 그리고 제어값이 적절한 메모리 또는 데이타 처리장치(3)의 레지스터에 기억된 후, 외부개발시스템(7)은 'Go' 커맨드를 DSDI 및 DSCLK 신호를 매개로하여 데이타 처리장치(3)에 제공한다. 다수의 데이타, 어드레스, 그리고 제어값 처럼 'Go' 커맨드는 DSDI 신호를 매개로 하여 데이타 처리장치(3)에 제공된다. 'Go' 커맨드의 각 비트는 직렬 인터페이스(30)를 매개로 하여 레지스터(32)에 제공된다. 레지스터(32)가 가득한 경우, 패킷의 정보는 제어버스(15)를 매개로하여 제어회로(20)에 제공된다. 제어회로(20)는 'Go' 커맨드를 해독하여 K-버스(25)를 경유하여 CPU(2)에 Go 신호를 제공한다.

CPU(2)가 Go 신호에 대응하는 경우, 상기가 실행하는 명령은 다수의 어드레스, 데이타 그리고 외부개발시스템(7)에 의해 제공된 제어값에 의해 변경된 레지스터에 의해 결정된다. 예를들어 CPU(2)내의(상세히 도시되지 않은) 프로그램 카운터가 변경되었다면, CPU(2)는 변경된 프로그램 카운터에 기억된 새로운 명령 어드레스로 부터 처리 장치 실행을 시작한다. 따라서, 만일 CPU(2)내의 프로그램 카운터가 변경되지 않았다면 CPU(2)는 정상적인 리셋 예의 처리 루틴과 더불어 지속된다.

리셋 예외 처리 루틴의 실행 이전에 데이타 처리장치(3)를 브레이크포인트 하는 능력은, 값이 리셋 예외 처리 루틴 중에 겹쳐쓰일 가능성이 없이 외부개발시 스템(7)이 소요된 방법으로 데이타 처리장치(3)를 구성하게 하는 효과적이며 유용한 메카니즘을 입증한다. 연산을 실행할 수 있음으로 인해 리셋 예외 처리 루틴이 실행되기 이전에 브레이크포인트 비 축적형 메모리 기억장치를 프로그래밍 함으로서 외부개발시스템이 바이패스 하게 하는 것이 또한 허용된다. 새로운 소프트웨어 화상을 기록가능한 메모리 기억장치에 로드함으로서 비축적형 기억장치에서 프로그램된 코드는 용이하게 바이패스 될 수 있다. 외부개발시스템(7)은 80비트 명령에 시프트 인 함으로서 어느 하드웨어 레지스터에 접근할 수 있다. 상기 80비트 명령은 16비트 연산코드, CPU 공간내에 맵핑된 32 비트 어드레스, 32 비트의 오퍼랜드 데이타를 포함한다. 일단 완전한 커맨드가 수신되면, 디버그 모듈(10)은 메모리 참조 연산과 유사한 방법으로 작동한다. K-버스(25)는 요구되고, 그후 CPU(2)에 의해 승인된다. CPU-공간 K-버스 사이클은 디버그 모듈(10)에 의해 개시되고 적절한 하드웨어 레지스터가 기록된다.

<u>백그라운드 디버그 모드</u>

데이타 처리장치(3)가 특정 백그라운드 디버그 모드에서 작동하는 경우 CPU(2)는 정지되어 고장난 회로와 장애를 일으킨 프로그램이 식별되고 분리되기 위하여 디버그 모듈(10)이 완전히 모든 메모리 및 레지스터 값에 액세스하게 한다. 레지스터 및 메모리 위치는 백그라운드 디버그 모드에 있는 경우, 판단되고 선택적으로 변경된다. 더욱이 테스트 기능이 가능해질 수 있다. 외부개발시스템(7)이 상기가 특정 백그라운드 디버그 모드에서 작동하고 있는 경우 디버그 모듈(10)을 매개로 하여 내부 정보에 용이하게 액세스 할 수 있는 동안, CPU(2)는 상기 연산 모 드에서 정지되어야 하므로 정상적인 명령 실행이 정지된다. CPU(2)가 정지된 경우 PST 신호는 $F의 값을 갖는다는 점을 주목하라.

연산의 특정 백그라운드 디버그 모드는 많은 소스(source)로부터 생성될 수 있다. 예를들어, BKPT 신호는, 데이타 처리장치(3)가 연산의 백그라운드 디버그 모드에 들어가게 하기 위해서 외부개발시스템(7)에 의해 외부에서 생성될 수 있다. 게다가, 내부에서 생성된 브레이크포인트 연산과 파국적 예외 조건은 또한 데이타 처리장치(3)가 연산의 백그라운드 디버그 모드로 들어가게 할 수 있다.

일단 데이타 처리장치(3)가 연산의 백그라운드 디버그 모드를 입력하면 데이타 처리장치(3)의 정상적인 연산은 정지되고, 특정 함수가 외부개발시스템(7)의 제어하에 디버그 모듈(10)에 의해 실행된다. 디버그 모듈(10)과 외부개발시스템(7) 사이의 모든 전달은 직렬 인터페이스(30)와 DSDI, DSDO 그리고 DSCLK 신호 각각을 통해 이루어 진다.

이전에 언급된 바와 같이 본 발명의 본 실시예에서, 디버그 모드는 PST 신호를 매개로하여 외부사용자에게 신호가 가해진다. PST 신호상의 SF의 값은 처리장치가 정지되고 연산의 백그라운드 디버그 모드에서 작동하고 있다는 것을 표시한다. 백그라운드 디버그 모드로의 입구의 표시는 상기가 외부사용자에게 직렬 전달 채널을 매개로하여 지원된 디버그 커맨드를 개시하게 하는 신호를 제공하는 바와 같이 증요하다. 따라서 직렬 인터페이스(30)를 매개로하여 제공된 직렬 포트는 메모리 위치와 내부 레지스터로 부터 디버그 정보를 검색하도록 사용될 수 있다.

만일 MAP 비트가 부정된다면, 모든 에뮬레이터 모드 메모리 참조는, 온 칩(on-chip) 메모리, 외부 메모리, 그리고 입력/출력 장치와 같은 시스템 자원이 참조될 수 있도록 감시 프로그램이 정상적으로 맵핑된다.

마지막으로 Go 신호는, 'Go' 커맨드가 외부개발시스템(7)으로부터 디버그 모듈(10)로 직렬로 시프트 인 된 후 표명된다. Go 신호는 몇몇 사상이 데이타 처리 장치(3)를 중지한 후 데이타 처리 장치(3)의 실행을 재개시하여 연산의 백그라운드 디버그 모드로 강제한다. 데이타 처리장치(3)는 TDR이 상기 유형의 반응을 강제하도록 프로그램 되었다면 제어회로(20)에 의한 브레이크포인트 신호의 표명에 의해 증지될 수 있다. 게다가, 데이타 처리장치(3)는 데이타 처리장치(3)에 의한 중지 명령의 실행에 의해 또는 폴트-온 폴트(fault-on-fault) 조건의 발생에 의해 정지될 수 있다.

요약하면, 본 발명은 실시간 트레이스 및 디버그 함수가, 백그라운드 디버그 모드와 같은 특정 연산 모드로 들어가는 데이타 처리장치를 필요치 않고도, 실행되게 하는 데이타 처리장치 와 연산모드를 제공한다. 게다가 본 발명은 실시간 디버그 함수의 실행중에 다중레벨 트리거를 실행 하기위한 방법을 제공한다. 상기 독특한 특징 각각은 외부의 사용자에게 데이타 처리 시스템(5)내의 고장을 식별하고 분리하는 경우 보다 큰 기능성과 융통성을 제공한다.
이전에 언급한 바와같이, 본 발명은 중대하고 여기에 비존재하는 실시간 트레이스 함수를 제공한다. 실시간 트레이스 함수는, 외부개발시스템(7)과 같은 외부 사용자가 외부버스의 형식 또는 사용가능성을 가정하지 않고서도 데이타 처리장치(3)의 내부연산을 동적으로 관찰하게 한다. 더욱이, 실시간 트레이스 함수 는 데이타 처리시스템(5)의 효율성과 속도에 심히 나쁜영향을 끼치지 않고 시스템의 내부 연산에 대한 상기의 관찰 가능성을 제공한다. 데이타 처리 시스템(5)의 디버그 모듈(10)은 외부의 사용자에게 DDATA 및 PST 신호를 매개로하여 내부 연산 정보를 제공하기 위한 병렬 출력 포트를 제공한다. DDATA 신호는 오퍼랜드 값을 반영하는 데이타를 제공하고, PST 신호는 CPU(2)의 실행 상태를 반영하는 코드화된 상태 정보를 제공한다. 더욱이, DDATA 신호는 포착된 명령 어드레스 프로그램 흐름 변화를 또한 제공하여 외부개발시스템이 외부에서 가시 가능한 어드레스 버스 또는 외부에서 가시가능한 데이타 버스를 필요치 않고도 정확한 프로그램 흐름을 추적하게 한다. DDATA 신호상에 표시된 정보는 PST 신호상의 상태 정보와 동기된다는 점을 주목하라. 따라서, 외부 사용자는 DDATA 및 PST 신호를 해독할 수 있어 데이타 처리장치(3)의 정상적인 연산을 중단시키지 않고 데이타 처리장치(3)내의 중요한 변수의 상태를 결정한다. 다른 방법으로 말하면, 데이타처리장치(3)는 특정 디버그 또는 에뮬레이터 모드에서 작동하는 것이 필요치 않고서도 내부정보를 외부개발시스템에 제공한다.
게다가, 본 발명은 실시간 디버그 함수를 실행하기 위한 회로와 방법을 제공한다. 많은 처리장치, 특히 삽입된 시스템에서, 데이타 처리장치는 데이타 처리장치가 더이상 정상 모드에서 작동하지 않은 경우, 시스템 제약조건 또는 장애를 일으킨 회로 또는 프로그램을 식별할 수 없음에 기인하여, 디버그 연산중에 정지되지 않을 수 있다. 본 발명은 데이타 처리장치를 정지시키지 않고 데이타 처리장치의 실시간 연산에 대해 단지 최소한도로 방해함으로서 디버그 연산을 실행한다.
브레이크포인트 연산은 전형적으로 디버깅 연산중에 사용되어 프로그래밍 사상을 식별한다. 프로그래밍 사상이 발생한 경우, 브레이크포인트 신호는 표명되고 데이타는 소프트웨어 프로그램의 상태를 결정하도록 검색된다. 트레이스 함수 처럼 브레이크포인트 연산은, 데이타 처리 에러가 식별될 수 있도록, 외부사용자가 다수의 선택된 레지스터 각각의 상태를 확인하게 한다. 본 발명은 다수의 하드웨어 브레이크포인트 레지스터(50)와 브레이크포인트 회로(100)를 제공하여 상기 브레이크포인트 연산을 실행한다. 본 발명에서, 프로그램 카운터 값 또는 프로그램 카운터 마스크, 오퍼랜드 어드레스 범위, 그리고 데이타 값 또는 데이타 마스크 값에 근거한 브레이크포인트 연산은 모두 실현된다. 게다가, 본 발명에서 실현된 브레이크포인트 연산은 1 또는 2레벨 트리거로 구성될 수 있고 상기 트리거에서 정확한 트리거 반응은 다수의 제어 레지스터(40)의 TDR(트리거 정의 레지스터)을 매개로하여 프로그램 가능하다.
또한, 본 발명은 데이타 처리장치(3)가 특정 디버그 모드에서 작동할 필요가 없이 CPU(2)와 디버그 모듈(10)의 동시 연산을 제공한다. 코어(9)와 디버그 모듈(10) 사이의 데이타, 어드레스, 그리고 제어정보를 전달하기 위한 K-버스(25)의 사용으로 인해 디버그 모듈(10)이 CPU(2)와 동일한 내부 레지스터와 메모리 위치에 액세스하는 것이 허용된다. 따라서, 디버그 모듈은 CPU(2)가 액세스 할 수 있는 모든 메모리 위치와 내부 레지스터에 액세스 할 수 있다. CPU(2)와 디버그 모듈(10)을 동시에 작동할 수 있음으로 인해 외부개발시스템이, 처리장치의 연산에 심히 영향을 끼치지 않고도 데이타 처리장치(3)의 내부레지스터 및 메모리에서 및 메모리에서 가시성이 증가되게 된다.
더욱이, 디버그 모듈(10)과 CPU(2) 모두가 동일한 내부 레지스터와 메모리 위치에 액세스 할 능력을 가진 반면에, CPU(2)와 코어(9)의 잔여부분은, 다수의 제어 레지스터(40)의 CSR 내의 IPW(Inhibit Processor Writes to Debug Resistor: 디버그 레지스터로의 기록을 금지하는 처리장치) 비트가 세트된 경우, 다수의 브레이크포인트 레지스터(50)에 기억된 값을 변경하지 못할 수 있다. CSR 내의 IPW 비트가 세트된 경우, CPU(2)에 의해 개시된 기록 연산 어느것도 금지된다. 더욱이, IPW 비트는 외부 개발시스템(7)에 의해 제공된 커맨드에 의해 단지 변경될 수 있다. 기록 연산을 다수의 하드웨어 브레이크포인트 레지스터(50)에 효과적으로 불능하게 하는 IPW의 사용으로 인해 외부개발시스템(7)에 의해 프로그램 되고 고장난 회로와 장애를 일으킨 프로그램을 식별하고 분리하는데 필요한 사상은 CPU(2)에 의해 겹쳐쓰이지 않을 것이라는 점이 확실해 진다.
더욱이, 본 발명은, 외부개발시스템(7)이 리셋 연산을 위한 예외 처리 루틴이 개시되기 전에 데이타 처리장치(3)에서 브레이크포인트 연산을 실행하게 하는, 메카니즘을 제공한다. 외부 리셋 신호가 표명된 경우, 다수의 제어회로(40) 각각의 내용은 초기화 된다. 외부 리셋 신호가 계속하여 부정된 경우, 실제 리셋 예의 처리 루틴을 실행하기전에 데이타 처리장치(3)가 정지하고 있는 시간의 윈도우(window of time)가 존재한다. 만일 외부 브레이크포인트신호, 즉 BKPT가 외부개발시스템(7)에 의해 정지시간 동안 표명되면, 본 발명은 하드웨어 레지스터 구성 어느것 이라도 실행될 수 있도록 목표 메모리 값을 데이타 처리 시스템(5)에 다 운로드 시키기 위한 메카니즘과 방법을 제공한다. 목표 메모리 값은 유일한 값일 수도 다수의 값 일수도 있다.
리셋 예의 처리 루틴이 실행되기 전에 브레이크포인트 연산을 실행하는 능력으로 인해 외부개발시스템(7)이(어느 비축적형 메모리 기억장치를 프로그래밍 함으로서) 마이패스 하는 것이 허용된다. 새로운 소프트웨어 화상을 기록가능한 메모리 기억장치에 로드함으로서 비축적형 메모리 저장장치에 프로그램된 코드는 용이하게 바이패스 될 수 있다.
외부개발시스템(7)은 80 비트 명령에 시프트 인 함으로서 하드웨어 레지스터 어디에도 액세스 할 수 있다. 80 비트 명령은 16 비트 연산코드, CPU 공간에 맵핑된 32 비트 어드레스, 그리고 32 비트의 오퍼랜드 데이타를 포함하고 있다. 일단 완전한 커맨드가 수신되면, 디버그 모듈(10)은 메모리 참조 연산과 유사한 방법으로 작동한다. K-버스(25)는 요구되고 그후 CPU(2)에 의해 승인된다. CPU 공간의 K-버스 사이클은 디버그 모듈(10)에 의해 개시되고 적절한 하드웨어 레지스터가 기록된다.

리셋 예의 처리 루틴의 실행 이전에 데이타 처리장치(3)를 브레이크포인트 하는 본 발명의 능력은 외부개발시스템(7)이(값이 리셋 예의 처리 루틴의 실행중에 겹쳐쓰일 가능성이 없이도 소요의 방법으로) 데이타 처리장치(3)를 구성하게 허용하는 효율 적이며 유용한 메카니즘을 입증한다.
여기에 기술된 본 발명의 수단은 단지 예로서 제공되어 있다. 따라서, 많은 다른 수단은 여기에 기술된 함수를 실행하기 위해 존재할 수 있다. 본 발명은 특정 실시예에 관련하여 나타나고 기술되었는데, 또다른 변형과 개선이 본 분야의 기술 자에게는 떠오를 것이다. 따라서 상기 발명은 나타난 특정 형태에 제한되지 않고 첨부된 청구항은 본 발명의 범위에서 벗어나지 않는 모든 변형을 포함하고 있다는 점을 이해해야 한다.

발명의 효과

본 발명은 특정 작동 방식이 필요치 않고 실시간으로 브레이크포인트 및 트레이스 함수 둘다를 제공할 수 있는 데이타 처리 시스템을 제공한다. 게다가, 브레이크포인트 및 트레이스 함수는, 외부 어드레스 버스 또는 외부 데이타 버스를 경유하여 어드레스 및 데이타 정보를 제공하기 위해 검사되는 데이타 처리장치가 필요없이도 데이타 처리 시스템에 의해 실행될 수 있다. 외부 버스로부터 상기와 같은 독립으로 인해 본 발명이 크기, 형태, 규약(protocol), 심지어 외부 버스의 존재에 관계없이 다양한 데이타 처리장치 시스템에 가해질 수 있는 효율적인 디버깅 메카니즘을 제공하는 것은 가능해 진다.
게다가, 본 발명은 실시간으로 트레이스 및 브레이크포인트 함수를 실현키 위한 구조와 기법을 제공한다. 검사되는 데이타 처리장치는 전형적으로, 트레이스 및 브레이크포인트 함수가 실행되기 이전에, 작동을 정지시키거나 수정하는 것이 필요치 않는다. 따라서 본 발명은, 데이타 처리장치가 디버깅 동작을 위한 특정한 방식이 아닌 정규방식대로 작동하는 동안에 외부의 사용자가 장애를 분리시키는 것을 가능케 해준다. 또한, 본 발명은, 실시간 트레이스 또는 실시간 디버그 동작을 실행하자마자 외부사용자에게 직접 데이타를 제공한다. 외부의 소프트웨어 모니터 프로그램 또는 트레이스 함수 또는 디버그 함수를 실행하는 에뮬레이터를 필요로하 는 선행기술 장치와는 다르게, 본 발명은, 외부의 사용자에게, 데이타 처리장치의 현재 동작을 지시하는 데이타를 제공한다. 따라서 본 발명은 데이타 처리장치를 과도히 밀어 넣거나 역으로 데이타 처리장치에 영향을 끼치지 않고도 트레이스 및 디버그 기능을 제공해 준다.

(57) 청구의 범위

청구항 1.
데이타 처리 장치에 있어서,
프로그램 카운터 관계 브랜치 명령과 변형 브랜치 명령을 포함하는 명령을 실행하는 중앙 처리 장치(CPU)와,
상기 CPU에 접속된 입력과, CPU 상태 출력부와 데이타 출력부를 제공하기 위한 디버그 상태 출력을 갖는 디버그 장치를 포함하고,
상기 디버그 장치는 상기 프로그램 관계 브랜치 명령을 실행하는 상기 CPU에 응답하여, 상기 CPU 상태 출력부에 활성 신호를 제공하고 상기 데이타 출력부에 비활성 신호를 제공하며,
상기 디버그 장치는 변형 브랜치 명령을 실행하는 상기 CPU에 응답하여, 상기 CPU 상태 출력부에 활성 신호를 제공하고 상기 데이타 출력부에 비활성 신호를 제공하는 데이타 처리 장치.

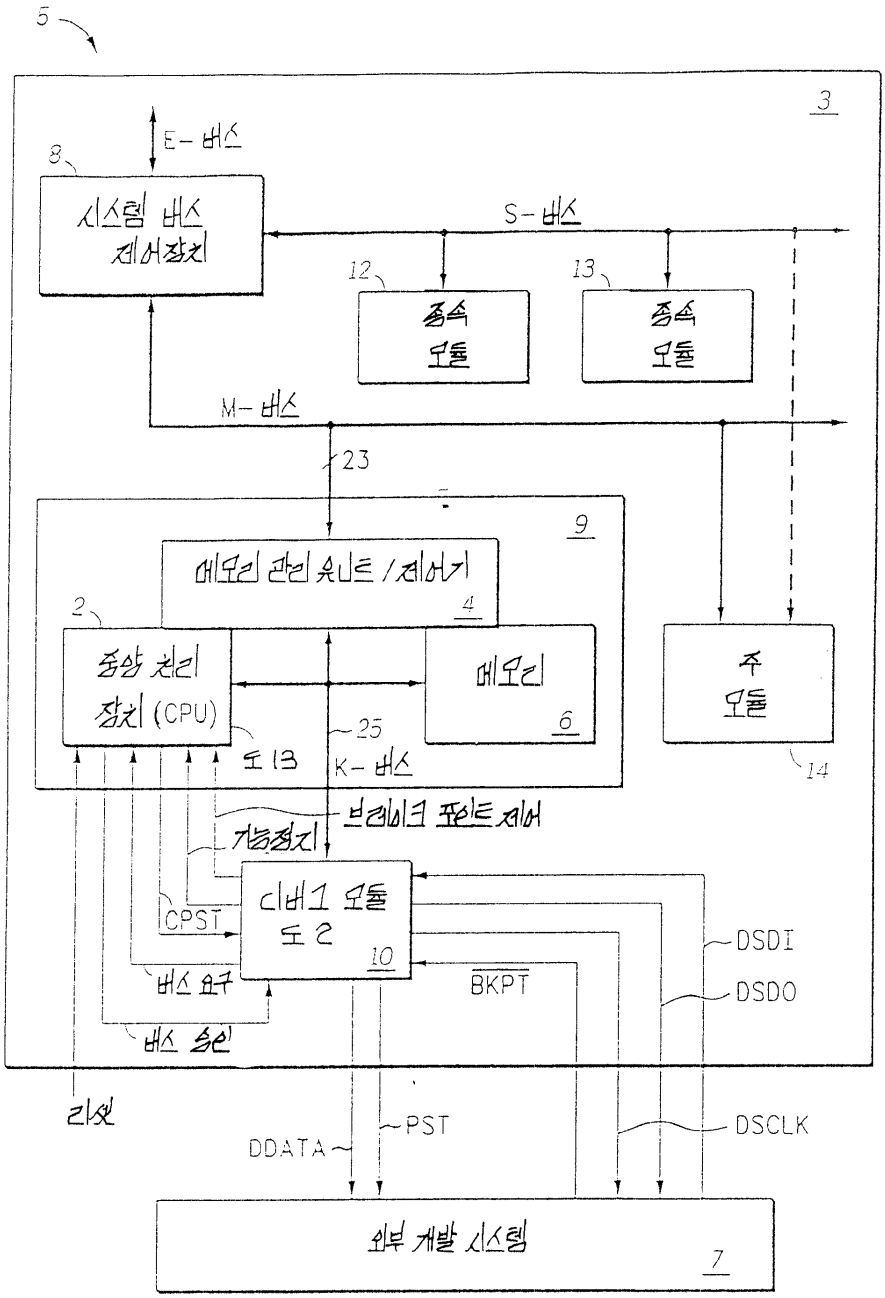
청구항 2.
데이타 처리 장치를 모니터하는 방법에 있어서,
1) 명령을 갖는 브랜치의 실행의 개시를 나타내는 상태 코드와,
2) 예외로부터 복귀의 개시를 나타내는 상태 코드와,
3) 펄스 명령의 개시 실행을 나타내는 상태 코드와,
4) 다른 명령들에 대한 개시 실행을 나타내는 상태 코드를 포함하는 상태 코드들의 세트를 규정하는 단계와,
제 1 버스로부터 제 1 명령을 수신하는 단계와,
중앙 처리 장치(CPU)를 이용하여 상기 제 1 명령 실행의 정상 모드를 개시하는 단계와,
상기 제 1 상태 코드로 상기 데이타 처리장치 외부에 디스플레이하는 단계를 포함하고,
상기 제 1 상태 코드는 상기 제 1 명령에 따라 상기 상태 코드들의 세트로부터 선택되는 데이타 처리 장치 모니터 방법.

청구항 3.
데이타 처리 장치를 모니터하는 방법에 있어서,
제 1 버스로부터 제 1 명령을 수신하는 단계와,
중앙 처리 장치(CPU)를 이용하여, 상기 제 1 명령의 실행을 개시하는 단계로, 상기 제 1 명령과 관련된 데이타가 존재하는 상기 제 1 명령의 실행 개시단계와,
상기 CPU가 상기 제 1 명령을 개시할 때, 상기 CPU로부터 상기 데이타 처리 장치와 디버그 부분까지 제 2 버스를 통해 내부 상태 코드를 전송하는 단계와,
상기 디버그 부분에 의해 사용하기 위한 데이타를 포착하는 단계와,
상기 데이타 처리 장치의 상기 디버그 부분을 거쳐 외부 상태 포트의 제 1 부분에, 상기 내부 상태 코드의 표시를 디스플레이하는 단계와,
데이타가 디버그 데이타 포트를 통해 전송되는 것을 나타내는 데이타 전송 상태를, 외부 상태 포트의 제 1 부분에 디

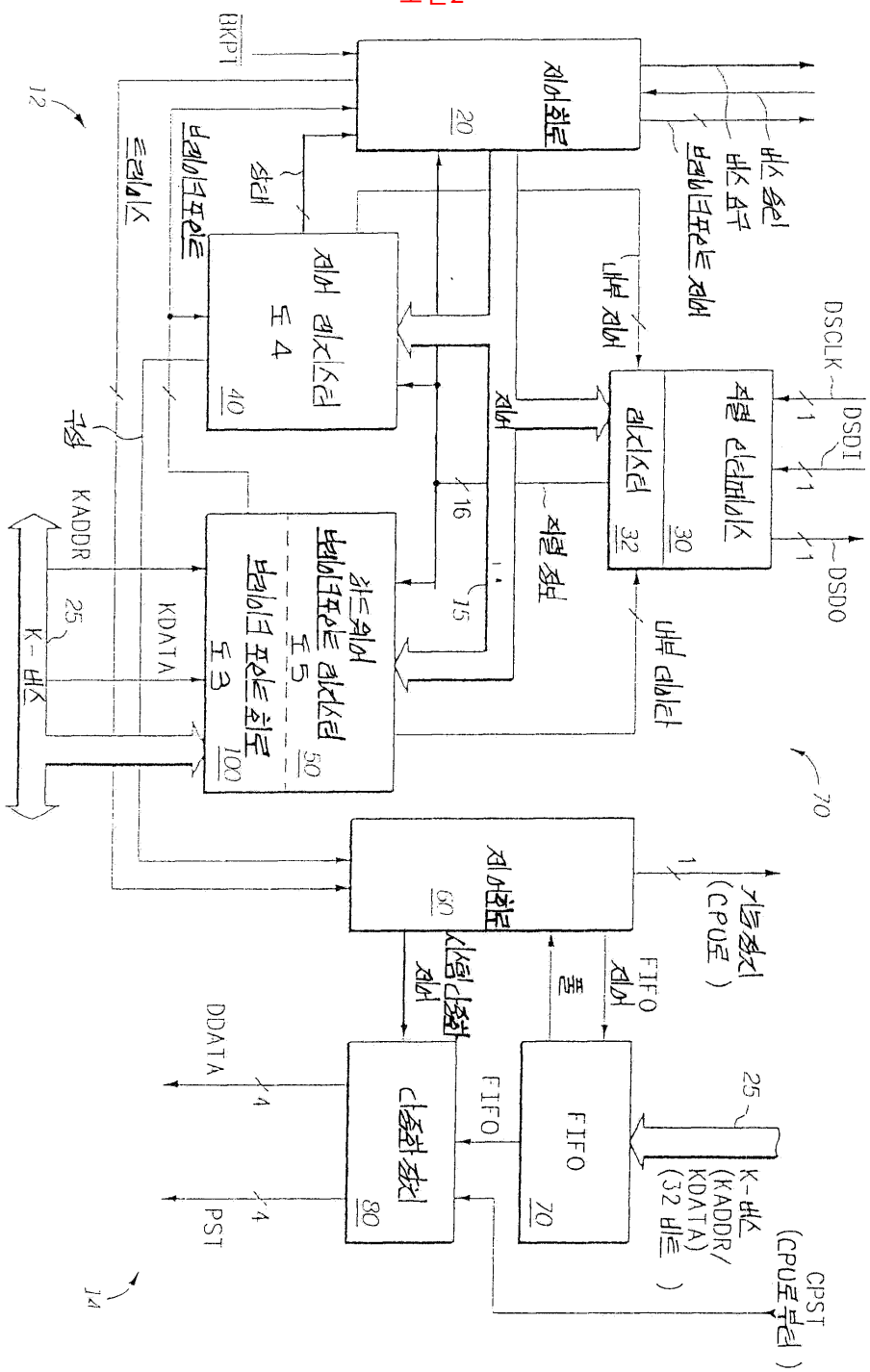
스플레이하는 단계와,
상기 데이타 전송 상태를 디스플레이한 후에, 소정 시간에 상기 디버그 데이타 포트에 적어도 상기 데이타의 일부를 디스플레이하는 단계를 포함하는 데이타 처리 장치 모니터 방법.

도면

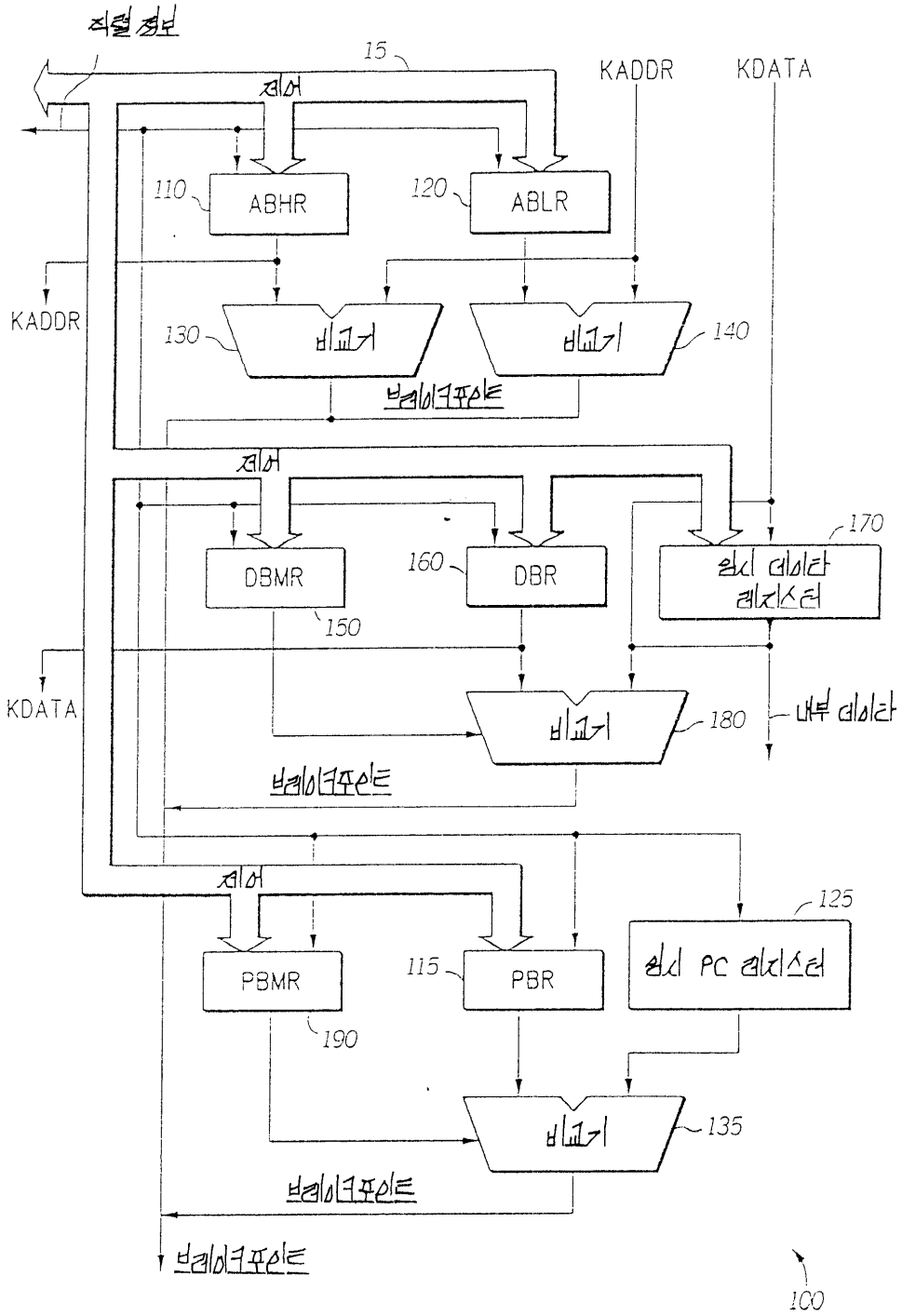
도면1



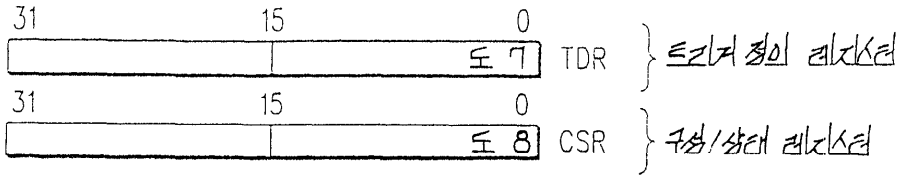
도면2


BKPT
제어회로
20
12
트레이스
브레이크포인트
상태
제어 레지스터
도 4
40
구성
KADDR
25
K-버스
KDATA
하드웨어
브레이크포인트 레지스터
도 5
50
브레이크 포인트 회로
도 3
100
16
15
제어
직렬 정보
버스 용인
버스 요구
브레이크포인트 제어
내부 제어
직렬 인터페이스
레지스터
30
32
DSCLK
DSDI
DSDO
내부 데이터
70
제어회로
60
기능정지
(CPU로)
시퀀 다중화 제어
FIFO 제어
풀
DDATA
PST
다중화 정치
80
FIFO
FIFO
70
25
K-버스
(KADDR/
KDATA)
(32 비트)
CPST
(CPU로부터)
14

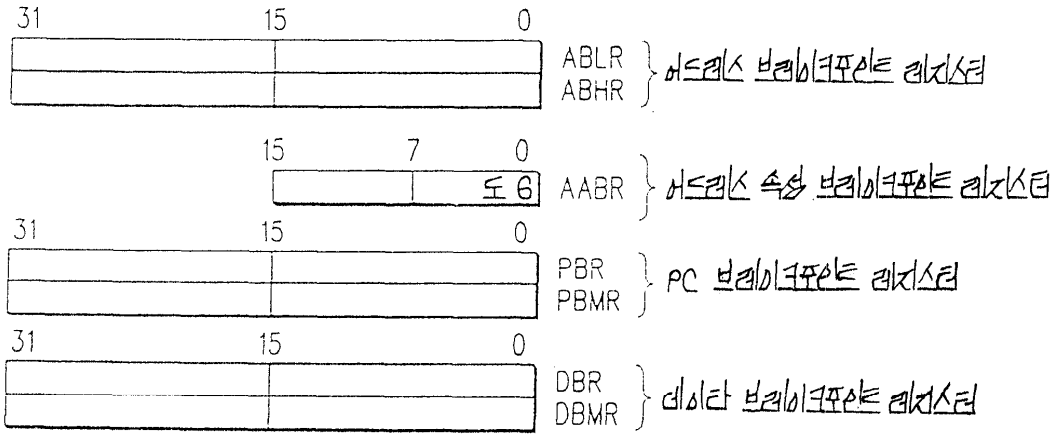
도면3


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

도면4


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

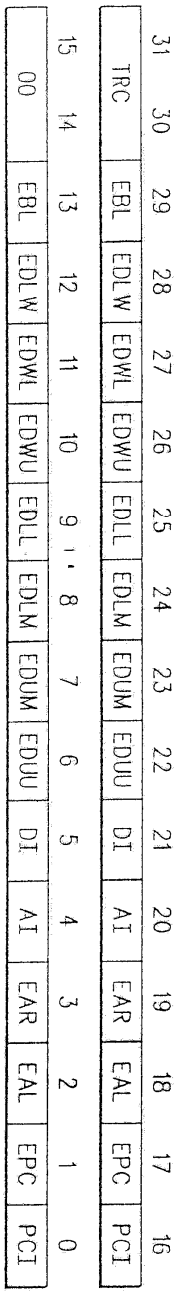
도면5


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

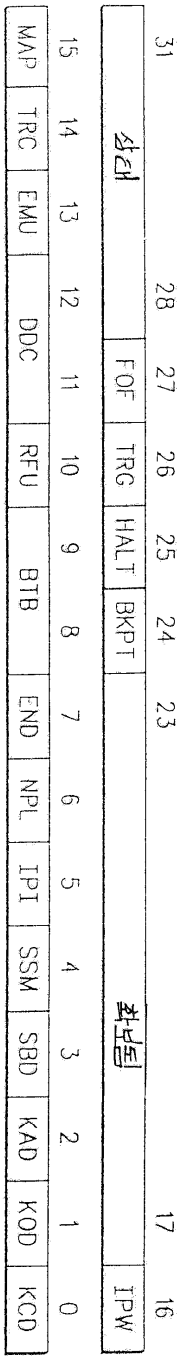
도면6


RM
SZM
TTM
TMM
R
SZ
TT
TM
15
14
13
12
11
10
8
7
6
5
4
3
2
0

도면7


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

도면8



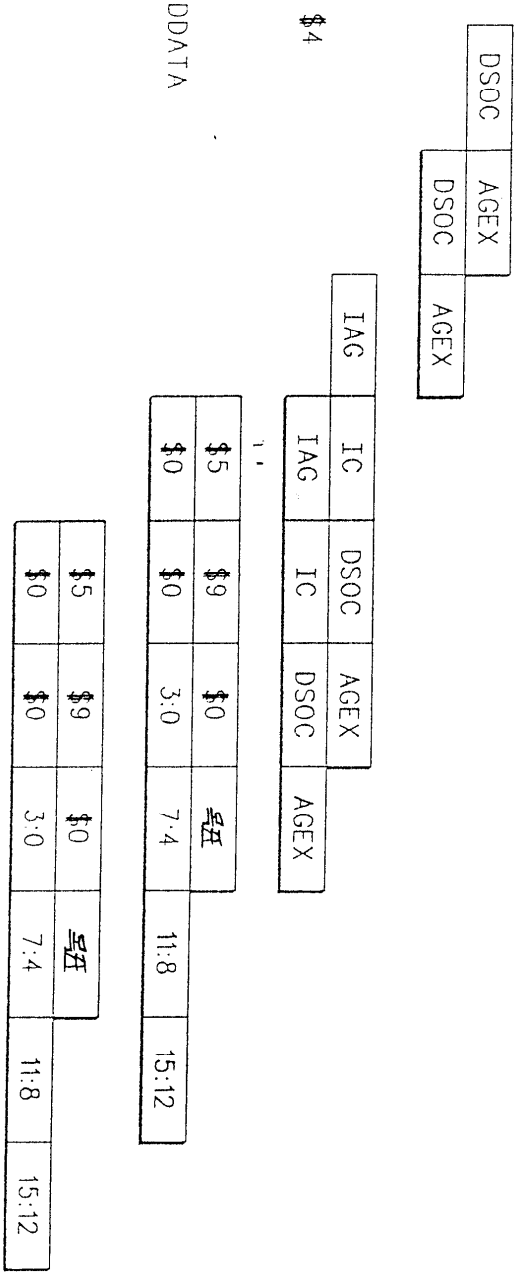
도면9

DDATA[3:0], CSR[31:28]	브레이크포인트 상태
$0	브레이크포인트 사용가능하지 않음
$1	레벨 1 브레이크포인트에 대한 기록
$2	트리거된 레벨 1 브레이크포인트
$3-4	확보됨
$5	레벨 2 브레이크포인트에 대한 기록
$6	트리거된 레벨 2 브레이크포인트
$7-$F	확보됨

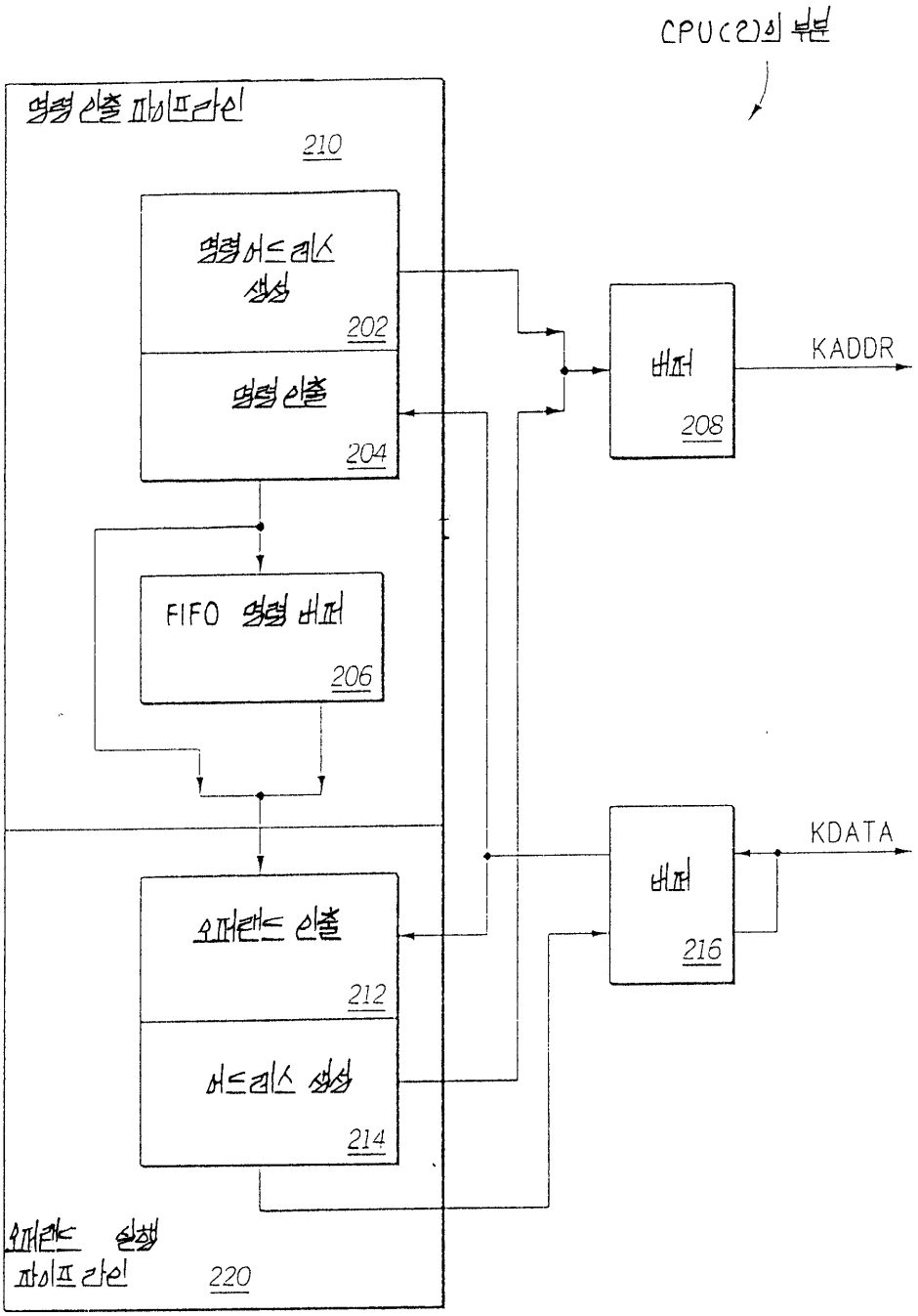
도면10

PST[3:0]	정의
0000	확장 지속
0001	지령 실행 게시
0010	확보됨
0011	사용자 모드로 진입
0100	펄스 명령의 실행을 시작
0101	선택된 분기의 실행을 시작
0110	확보됨
0111	RTE 명령의 실행을 시작
1000	DDATA에 대해 1 바이트 전송을 시작
1001	DDATA에 대해 2 바이트 전송을 시작
1010	DDATA에 대해 3 바이트 전송을 시작
1011	DDATA에 대해 4 바이트 전송을 시작
1100	예외처리
1101	에뮬레이터 모드 진입 예외처리
1110	프로세서는 정지하고, 인터럽트를 기다린다
1111	프로세서 정지

도면11


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

도면12


CPU(C)의 부분
명령 인출 파이프라인
210
명령 어드레스 생성
202
명령 인출
204
FIFO 명령 버퍼
206
버퍼
208
KADDR
오퍼랜드 인출
212
어드레스 생성
214
버퍼
216
KDATA
오퍼랜드 실행 파이프라인
220