(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2008년04월07일
(11) 등록번호 10-0819720
(24) 등록일자 2008년03월31일

(51) Int. Cl.
G06F 11/36 (2006.01)
(21) 출원번호 10-2003-7010937
(22) 출원일자 2003년08월20일
심사청구일자 2006년12월18일
번역문제출일자 2003년08월20일
(65) 공개번호 10-2003-0075202
(43) 공개일자 2003년09월22일
(86) 국제출원번호 PCT/US2001/049157
국제출원일자 2001년12월18일
(87) 국제공개번호 WO 2002/69146
국제공개일자 2002년09월06일
(30) 우선권주장
09/788,816 2001년02월21일 미국(US)
(56) 선행기술조사문헌
US05361392 A1
전체 청구항 수 : 총 7 항

(73) 특허권자
프리스케일 세미컨덕터, 인크.
미합중국 텍사스 (우편번호 78735) 오스틴 윌리암 캐논 드라이브 웨스트 6501
(72) 발명자
우드마이클씨
미국,텍사스78660,프플루거빌,팍크레스트코트1000
베커조지이
미국,텍사스78736,오스틴,콜브람레인7806
시빅트러스제임스엠
미국,텍사스78681,라운드락,왈시드라이브2412
(74) 대리인
이병호, 장훈

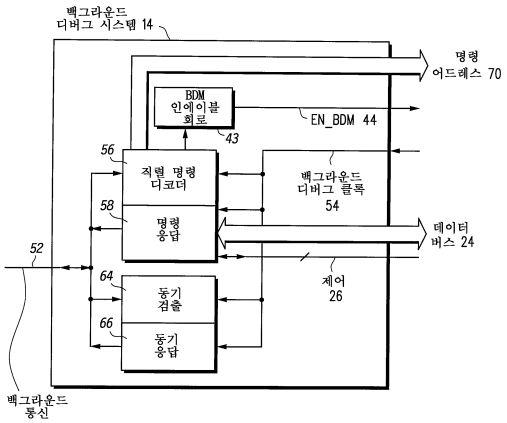
심사관 : 김견수

(54) 온 칩 백그라운드 디버그 시스템 및 그 방법을 갖는데이터 처리 시스템

(57) 요 약

본 발명의 실시예들은, 호스트 개발 시스템이 백그라운드 디버그 통신 인터페이스(52)에 결합되고 백그라운드 디버그 모드가 인에이블되었을 때, 발진기가 정지되는 것을 금지하는 기구에 관한 것이다. 이것은 타겟 데이터 처리 시스템이 저 전력 모드내에 있을 때, 백그라운드 디버깅 동작들이 계속하는 것을 허용한다. 다른 실시예들은 호스트 개발 시스템이 타겟 데이터 처리 시스템으로부터 동기화 타이밍 펄스를 요청하는 것을 허용하고, 그리하여 정확한 클록 속도가 백그라운드 통신들을 위해 결정될 수 있는 기구에 관한 것이다. 대안의 실시예들은 백그라운드 디버그 시스템이 시스템 클록 유닛으로부터 분리한 백그라운드 디버그 클록 유닛, 및 인에이블 제어(44)를 포함하는 백그라운드 디버그 시스템(14), 및 시스템 클록 유닛을 가진 데이터 처리 시스템에 관한 것이다. 인에이블 제어가 어서팅될 때, 시스템 클록 유닛과 관계없이, 백그라운드 디버그 클록 유닛이 인에이블된다.

대표도 - 도3



특허청구의 범위

청구항 1

데이터 처리 시스템에 있어서,

인에이블 제어(enable control;44)를 갖는 백그라운드 디버그 시스템(background debug system;14); 및

상기 백그라운드 디버그 시스템에 결합되어 상기 인에이블 제어를 수신하도록 적응된 클록 유닛(19)으로서, 상기 클록 유닛은 발진을 정지시킬 수 있는, 상기 클록 유닛(19)을 포함하며,

상기 인에이블 제어가 어서팅(asserting)될 때, 상기 클록 유닛이 발진을 정지하는 것이 금지되는, 데이터 처리 시스템.

청구항 2

클록 유닛(19)에 결합된 백그라운드 디버그 시스템(14)을 갖는 데이터 처리 시스템을 동작시키는 방법에 있어서,

저 전력 모드에 들어하는 단계로서, 상기 저 전력 모드 동안, 상기 클록 유닛은 발진을 정지시킬 수 있는, 상기 저 전력 모드에 들어가는 단계;

백그라운드 디버그 인에이블 제어(44)를 어서팅하는 단계; 및

상기 백그라운드 디버그 인에이블 제어를 어서팅하는 것에 응답하여, 상기 백그라운드 디버그 시스템이 상기 클록 유닛으로 하여금 발진을 정지시키는 것을 금지하는 단계를 포함하는, 데이터 처리 시스템을 동작시키는 방법.

청구항 3

데이터 처리 시스템에 있어서,

미리 결정된 통신 프로토콜에 따라 미리 결정된 심볼 지속기간을 갖는 심볼들을 전송할 수 있는 통신 인터페이스;

상기 통신 인터페이스에 결합되어 동기화 요청을 수신하도록 적응된 동기화 검출 유닛으로서, 상기 동기화 요청은 상기 심볼 지속기간보다 더 긴 지속기간을 갖는, 상기 동기화 검출 유닛; 및

상기 통신 인터페이스에 결합되어 상기 동기화 요청에 동기화 응답을 제공하도록 적응된 동기화 응답 유닛으로서, 상기 동기화 응답은 상기 데이터 처리 시스템에 의해 호스트 유닛에 제공되고, 상기 호스트 유닛은 상기 동기화 응답으로부터 상기 미리 결정된 심볼 지속기간을 결정할 수 있는, 상기 동기화 응답 유닛을 포함하는, 데이터 처리 시스템.

청구항 4

삭제

청구항 5

삭제

청구항 6

데이터 처리 시스템에 있어서,

백그라운드 디버그 속도로 상기 데이터 처리 시스템 외부의 호스트 유닛과 통신할 수 있는 비동기 통신 인터페이스; 및

상기 비동기 통신 인터페이스에 결합된 백그라운드 디버그 시스템을 포함하며,

상기 백그라운드 디버그 시스템은,

상기 호스트 유닛으로부터 동기화 요청을 수신하도록 적응된 동기화 검출 유닛; 및

상기 동기화 요청에 응답하여 상기 호스트 유닛에 동기화 응답을 제공하도록 적응된 동기화 응답 유닛으로서, 상기 동기화 응답은 상기 백그라운드 디버그 속도를 결정하도록 사용되는, 상기 동기화 응답 유닛을 포함하는, 데이터 처리 시스템.

청구항 7

삭제

청구항 8

제 1 데이터 처리 시스템을 제 2 데이터 처리 시스템에 동기화하는 방법으로서, 상기 제 1 및 제 2 데이터 처리 시스템들은 상기 제 2 데이터 처리 시스템의 클록 속도에 비례하여 미리 결정된 심볼 지속기간을 갖는 심볼들을 전송하는 통신 프로토콜에 따라 통신하는, 상기 동기화 방법에 있어서,

상기 제 1 데이터 처리 시스템으로부터 상기 미리 결정된 심볼 지속기간보다 더 긴 지속기간을 갖는 동기화 요청을 제공하는 단계; 및

상기 동기화 요청에 응답하여, 상기 제 2 데이터 처리 시스템은 고정되고 미리 결정된 지속기간을 갖는 동기화 응답을 제공하고, 상기 제 1 데이터 처리 시스템은 상기 동기화 응답으로부터 상기 미리 결정된 심볼 지속기간을 결정하는 단계를 포함하는, 동기화 방법.

청구항 9

데이터 처리 시스템에 있어서,

백그라운드 디버그 클록 유닛(19)과 상기 백그라운드 디버그 클록 유닛에 결합된 인에이블 제어(44)를 갖는 백그라운드 디버그 시스템(14); 및

상기 백그라운드 디버그 시스템에 결합되어, 시스템 발진기에 결합되도록 적응되고 상기 시스템 발진기의 발진을 정지시킬 수 있는 시스템 클록 유닛을 포함하며,

상기 인에이블 제어가 어서팅될 때, 상기 백그라운드 디버그 클록 유닛이 인에이블되는, 데이터 처리 시스템.

청구항 10

시스템 클록 유닛에 결합된 백그라운드 디버그 시스템을 갖는 데이터 처리 시스템을 동작시키는 방법으로서, 상기 백그라운드 디버그 시스템은 백그라운드 디버그 클록 유닛을 갖는, 상기 데이터 처리 시스템을 동작시키는 방법에 있어서,

저 전력 모드에 들어가는 단계로서, 상기 저 전력 모드 동안, 상기 시스템 클록 유닛은 상기 데이터 처리 시스템에 대한 발진을 금지하는, 상기 저 전력 모드에 들어가는 단계;

백그라운드 디버그 인에이블 제어를 어서팅하는 단계; 및

상기 백그라운드 디버그 인에이블 제어를 어서팅하는 것에 응답하여, 상기 시스템 클록 유닛과 관계없이, 상기 백그라운드 디버그 클록 유닛의 발진을 인에이블하는 단계를 포함하는, 데이터 처리 시스템을 동작시키는 방법.

명 세 서

기 술 분 야

<1> 관련 출원

<2> 본원은 동일 출원된 발명의 명칭 "디버그 정보를 저장하는 온 칩 FIFO를 갖는 데이터 처리 시스템 및 그 방법(DATA PROCESSING SYSTEM WITH ON-CHIP FIFO FOR STORING DEBUG INFORMATION AND METHOD THEREFOR)"의 사건 번호 SC11064TH인 미국특허공보에 관련되어 있고, 본원에 참조로서 포함되어 있으며, 이것의 현재 양수인에게 양도되어 있다.

<3> 발명의 분야

<4> 본 발명은 일반적으로 데이터 처리 시스템들에 관한 것으로, 특히 온 칩 백그라운드 디버그 시스템(on-chip

background debug system)들을 갖는 데이터 처리 시스템에 관한 것이다.

배 경 기 술

<5> 관련 기술

<6> 전력 소모를 감소시키기 위하여, 최신의 데이터 처리 시스템들은 종종 응용 프로그램이 시스템 클록 속도(system clock speed)를 조정하거나 심지어 발진기(oscillator)를 정지시키는 것을 허용한다. 어떤 경우에, 이러한 행동들은 호스트 개발 시스템에게 그것의 통신 속도를 조정하여 타겟 데이터 처리 시스템 내의 변화들에 적응하라고 요청한다. 타겟 시스템 발진기가 정지되는 경우에, 백그라운드 통신(background communication)이 또한 정지되고, 그리하여 관독 또는 기록 타겟 시스템 메모리 위치들과 같은 정규 디버깅 동작들을 금지한다. 따라서, 응용 프로그램이 시스템 클록 속도를 정지하거나 조정하는 동안 정규 디버깅 동작들을 허용할 필요성이 존재한다. 또한, 타겟 데이터 처리 시스템을 갖는 백그라운드 통신을 위해 올바른 클록 속도를 결정하는 호스트 개발 시스템에 대한 필요성이 존재한다.

실 시 예

<11> 본 발명은 첨부 도면들에 예로서 도시되지만 이에 제한되지 않고, 상기 도면들에서 동일한 참조 부호들은 유사한 구성 요소들을 나타낸다.

<12> 도면들에서의 구성 요소들이 단순성과 명확성을 위해 도시되며 비례대로 그려질 필요가 없다는 것을 숙련된 자들은 알고 있다. 예를 들면, 일부 구성 요소들의 크기들은 본 발명의 실시예들의 이해를 높이는데 도움이 되도록 다른 실시예들에 비해 확장될 수도 있다.

<13> 본 명세서에서 사용되는 것으로서, "버스(bus)"라는 용어는 데이터, 어드레스, 제어 또는 상태와 같은 하나 또는 그 이상의 다양한 형태의 정보를 전달하기 위해 이용될 수 있는 복수의 신호들 또는 도선들(conductors)을 지칭하는데 사용된다. "어서팅(asserting)"이라는 용어는 신호, 상태 비트, 또는 유사 장치를 그의 논리적으로 트루 상태(true state)로 하는 것을 지칭할 때 사용된다. "부정(negate)" 또는 "디어서팅(deasserting)"는 신호, 상태 비트 또는 유사 장치를 그의 논리적으로 폴스 상태(false state)로 하는 것을 지칭할 때 사용된다. 신호(또는 상태 비트 등)가 액티브 하이일 때, 논리적으로 트루 상태는 논리 레벨 1이고, 논리적으로 폴스 상태는 논리 레벨 0이다. 신호(또는 상태 비트 등)가 액티브 로우일 때, 논리적으로 트루 상태는 논리 레벨 0이고, 논리적으로 폴스 상태는 논리 레벨 1이다. 또한, "하이(high)"는 논리 레벨 1을 지칭하는데 사용될 수도 있고, "로우(low)"는 논리 레벨 0을 지칭하는데 사용될 수도 있다.

<14> 도 1은 본 발명의 일 실시예에 따라 데이터 처리 시스템(10)을 블록 다이어그램 형태로 도시한다. 데이터 처리 시스템(10)은 마이크로컨트롤러, 마이크로프로세서, 디지털 신호 처리기(DSP) 등의 소정의 처리 시스템일 수 있다. 데이터 처리 시스템(10)은 CPU(12), 클록 유닛(19), 메모리 모듈(16), 다른 모듈(18), 디버그 모듈(20), 내부 어드레스 버스(22), 내부 데이터 버스(24) 및 제어 신호들(26)을 포함한다. CPU(12)는 백그라운드 디버그 시스템(BDS)(14)을 포함한다. BDS(14)는 백그라운드 통신 인터페이스(52)를 포함한다. 내부 데이터 버스(24), 내부 어드레스 버스(22) 및 제어 신호들(26)은 CPU(12)와 데이터 처리 시스템(10) 상의 각각의 주변 모듈들 사이에 결합된다. 클록 유닛(19)은 제어 신호들(26)을 통하여 CPU(12)에 결합되고, 신호들(44 및 54)을 통하여 BDS(14)에 결합된다. 또한, 클록 유닛(19)은 발진기 회로에 결합하는 인터페이스 신호들(48 및 49)을 포함한다.

<15> 동작 중, CPU(12)는 데이터 버스(24)를 통하여 메모리 모듈(16)에 저장된 소프트웨어 프로그램으로부터 명령들을 수신하여 실행한다. 다음에, CPU(12)는 어떤 업무들을 수행하기 위하여 데이터 처리 시스템의 다른 리소스들을 감시하고 사용한다. 메모리 모듈(16)은 정적 랜덤 액세스 메모리, 동적 랜덤 액세스 메모리, 또는 예컨대 플래시와 같은 소정 형태의 비휘발성 메모리를 포함하는 소정 형태의 메모리일 수 있지만, 그에 제한되지는 않는다. 다른 모듈(18)은 또다른 메모리 모듈, 아날로그-디지털 변환기, 타이머 모듈, 예컨대 CAN 모듈인 직렬 통신 모듈, 범용 입력/출력 모듈 등일 수 있다. 디버그 모듈(20)은 프로그램 디버깅을 고려한 소정의 적당한 디버그 모듈일 수 있다.

<16> 클록 유닛(19)은 EN_BDM(44)과 백그라운드 디버그 클록(54)을 통하여 백그라운드 디버그 시스템(14)에 결합된다. 신호들(48 및 49)은 외부 발진기 구성 요소들에 인터페이스 신호들을 제공한다. 또한, 클록 유닛(19)은 제어 신호들(26)을 통하여 제어 신호들을 수신하고 제공한다. 예를 들어, 클록 유닛(19)은 CPU(12)에 클록 신호들을 제공하고 제어 신호들(26)을 통하여 STOP 신호를 수신한다. 또한, 클록 유닛(19)은 CPU(12), 메모리 모듈(16), 디버그 모듈(20) 및 다른 모듈(18)에 시스템 클록들을 제공한다.(클록 유닛(19)은 하기에 도 2를 참

조하여 더 논의될 것이다.)

<17> 또한, BDS(14)는, 호스트 개발 시스템이 (타겟 시스템으로서 또한 지칭될 수 있는) 데이터 처리 시스템(10)에 결합되는 것을 허용하는 백그라운드 통신 인터페이스(52)를 포함한다. 그러므로, 호스트 개발 시스템은 백그라운드 통신 인터페이스(52)를 통하여 디버그 동작들을 수행할 수 있다. 일 실시예에서, 백그라운드 통신 인터페이스(52)는 비동기 양방향 신호선 인터페이스(asynchronous b-directional single-wire interface)일 수 있다. 본 실시예에서, BDS(14)는 호스트 시스템으로 디버그 동작들을 수행하는데 하나의 외부 핀만을 필요로 한다. 대안의 실시예에서, 다른 적당한 통신 인터페이스들은 JTAG 인터페이스와 같이 사용될 수 있다.

<18> 도 2는 도 1의 클록 유닛(19) 및 CPU(12)의 일부와, 외부 발진기 구성 요소들(30)을 도시한다. 일 실시예에서, 외부 발진기 구성 요소들(30)은 크리스털(crystal) 또는 공진기(32), 피드백 레지스터(34), 및 2개의 부하 캐패시터들(36 및 38)을 포함하며, 모두 전형적인 피어스 발진기(Pierce oscillator) 형상에 결합된다. 그러나, 대안의 실시예들은 다른 적당한 발진기 구성 요소들 및 형상들을 이용할 수 있다. 클록 유닛(19)은 인버터(42), NAND 게이트들(66 및 62), 및 클록 제어(46)를 포함한다. NAND 게이트(66)는 STOP 신호(68), 인버터(42), 및 NAND 게이트(62)에 결합된다. NAND 게이트(62)(또한 발진기 증폭기(62)로 지칭)는 발진기 구성 요소들(30) 및 클록 제어(46)에 결합된다. 인버터(42)는 BDS(14)로부터 EN_BDM(44)을 수신하고, 클록 제어(46)는 BDS(14)에 백그라운드 디버그 클록(54)을 제공한다. CPU(12)는 명령 어드레스(70)를 통하여 어드레스 발생 유닛(74)에 결합된 BDS(14)를 포함한다. 또한, 어드레스 발생 유닛(74)은 CPU 어드레스(72)를 수신하고, 어드레스 버스(22)를 통하여 시스템 어드레스들을 제공한다. 또한, BDS(14)는 백그라운드 통신 인터페이스(52), 데이터 버스(24) 및 제어 신호들(26)에 양방향으로 결합된다.

<19> 동작 중, 클록 유닛(19)에서의 발진기 증폭기(62)는 액티브 하이 인에이블 신호(64; active high enable signal)를 디어서팅함으로써 디스에이블링될 수 있다. 종래 기술의 시스템들에서, 발진기 증폭기는 전력 소모를 감소하기 위하여 데이터 처리 시스템이 정지 모드에 들어갈 때 디스에이블링될 것이다(그러므로 발진기를 디스에이블링할 것이다). 그러나, 도 2의 실시예에서, 인에이블 신호(64)는 NAND 게이트(66)의 출력에 의해 구동된다. NAND 게이트(66)에의 2개의 입력들은 STOP 신호(68) 및 정지 인에이블 제어 신호(40)이다. 본 실시예에서, 양자의 신호들(68 및 40)이 액티브 하이 신호들이라는 것에 주목한다. 또한, STOP 신호(68)가 CPU 명령에 의해 전력을 감소하기 위하여 데이터 처리 시스템(10)을 정지 모드(또는 저 전력 모드)로 두는 것이 발생될 수 있다는 것에 주목한다. 정규 동작 동안, 통상적으로 호스트 시스템은 백그라운드 통신 인터페이스(52)에 결합되지 않고, EN_BDM((44)은 로우이다(즉, BDS(14)은 인에이블링되지 않는다). 따라서, 정규 동작 동안, STOP 신호(68)는, STOP 신호(68)가 각각 로우 또는 하이인지에 따라, 발진기가 가동할 것인지 또는 정지할 것인지를 결정한다. 또한, 정규 동작 동안, 어드레스 발생 유닛(74)은 CPU 어드레스(72)로부터의 정보를 어드레스 버스(22)에 건넨다. 예를 들어, CPU 어드레스(72)는 CPU(12)내의 실행 유닛(도시 안됨)으로부터 어드레스들을 수신할 수 있다.

<20> 그러나, 개발 및 디버그 동작 동안, 호스트 시스템은 백그라운드 통신 인터페이스(52)에 일반적으로 결합되고, EN_BDM(44)는 하이이다. 이는 BDS(14)를 인에이블링시키고, 정지 인에이블 제어 신호(40)가 로우인 인버터(42)에 입력을 보낸다. 이는 STOP 신호(68)의 상태와 관계없이 NAND 게이트(66)의 출력이 하이가 되게 한다. 따라서, 이는 BDS(14)가 STOP 신호(68)와 관계없이 발진기 증폭기(62)를 인에블링하는 것을 허용한다. 또한, STOP 신호(68)가, 디버그 동작 동안 BDS(14)가 발진기를 가동시킬 때, 다른 시스템 클록들이 여전히 정지될 수 있도록, 데이터 처리 시스템(10)내의 다른 회로들을 구동한다는 것에 주목한다. 또한, BDS(14)는 데이터 버스(24), 제어 신호들(26) 및 어드레스 버스(22)를 사용하여, 예를 들어 메모리로부터 판독하고 기록하는 등의 백그라운드 디버그 동작들을 수행한다. 이러한 디버그 동작들 동안, 어드레스 발생 유닛(74)은 명령 어드레스(70)상의 BDS(14)로부터의 어드레스들을, BDS(14)가 데이터 처리 시스템 메모리들에 접근하는 것을 허용하도록 어드레스 버스(22)에 건넨다.

<21> 따라서, 본 발명의 여러 실시예들에 따라, 호스트 개발 시스템이 백그라운드 통신 인터페이스에 결합되고 백그라운드 디버그 모드가 인에이블링될 때, 발진기 정지 모드의 정규 활동이 오버라이드(override)되어 발진기가 계속해서 가동한다는 것을 이해할 수 있다. 이는 백그라운드 디버그 통신 인터페이스가 계속해서 동작하는 것을 허용하는데, 그 이유는 다른 데이터 처리 시스템 모듈들이 전력을 절약하기 위하여 폐쇄되는 동안에도 정규 디버그 동작들이 여전히 수행될 수 있기 때문이다. 예를 들어, 백그라운드 디버그 모드가 인에이블링될 때, 호스트 개발 시스템은 데이터 처리 시스템 모듈들이 폐쇄되는 동안 타겟 시스템의 상태를 결정하도록 READ_STATUS 명령을 보낼 수 있다.

<22> 도 2를 참조하면, 클록 제어(46)는 발진기 구성 요소들(30)로부터의 기준 발진기 신호(48)를 수신하고, 기준 신호(48)에 근거하여 백그라운드 디버그 클록(54)을 BDS(14)에 제공한다. 클록 제어(46)는 기준 발진기 신호를 조정하도록, 분할기(divider)들과 같은 회로를 포함할 수 있다. 따라서, 발진기 기준 신호의 주파수만을 알고 있는 호스트 개발 시스템과 클록 제어(46)의 특정되지 않은 것은 클록 제어(46)에 의해 발생된 시스템 클록들의 실제 주파수를 결정할 수 없다. 따라서, 도 3 및 도 4를 참조하여 논의될 바와 같이, BDS(14)는 호스트 시스템으로 그것의 디버그 동작들을 적당히 수행하기 위하여 이러한 문제점을 처리하여야 한다.

<23> 도 3은 BDS(14)의 일 실시예를 도시한다. 외부 호스트 개발 시스템(즉, 외부 디버그 호스트 시스템)으로부터 신호들을 수신하는 백그라운드 통신 인터페이스(52)는 직렬 명령 디코더 블록(56), 명령 응답 블록(58), 동기화(동기) 검출 블록(64), 및 동기화(동기) 응답 블록(66)에 결합된다. 백그라운드 디버그 클록 신호(54)는 직렬 명령 디코더 블록(56), 명령 응답 블록(58), 동기 검출 블록(64) 및 동기 응답 블록(66)에서 동작들의 타이밍을 제어한다. 또한, 명령 응답 블록(58)은 데이터 버스(24) 및 제어 신호들(26)에 결합되어, 직렬 백그라운드 명령들이 메모리와 레지스터 값들을 판독하거나 기록하고, 또는 GO, TRACE, 또는 enter-active-GACKGRPUND과 같은 디버그 명령들을 착수하는 것을 허용한다. 또한, 직렬 명령 검출기(56)는 EN_DBM(44)을 제공하는 BDM 인에이블 회로(43)에 결합된다.

<24> BDM 인에이블 회로(43)는 그의 제어 비트들의 하나로서 EN_BDM((44)를 저장하기 위한 제어 레지스터를 포함할 수 있거나, 또는 EN_BDM(44)를 어서팅하기 위하여 설계된 다른 회로를 포함할 수 있다. 일 실시예에서, EN_BDM(44)는 백그라운드 통신 인터페이스(52)를 통하여 호스트 개발 시스템에 의해 발행된 BDS 명령에 의해서만 접근할 수 있는 제어 레지스터에 저장된 비트일 수 있다. 이것은 사용자 코드가 EN_BDM(44)를 고의로 또는 의도적이지 않게 어서팅하고 STOP 신호(68) 오버라이드를 채용하는 것을 금지한다. 대안의 실시예에서, EN_BDM(44)는 제어 비트로서 저장될 수 없고, 유효한 디버그 통신들이 백그라운드 통신 인터페이스(52)를 통하여 일어날 때를 검출하는 논리 회로에 의해 그 대신 어서팅될 수 있다. 대안의 실시예들은 BDS(14)를 인에이블링하기 위하여 EN_BDM(44)를 어서팅하는, BDM 인에이블 회로(43)에 관련하여 기술된 것과는 다른 서로 다른 기구들 및 회로들을 사용할 수 있다.

<25> 백그라운드 디버그 동작들 동안, 직렬 명령들 및 데이터는 백그라운드 통신 인터페이스(52)를 통하여 수신되고, 직렬 명령 디코더(56)에 의해 디코딩된다. 다음에, 명령 응답 블록(58)은 데이터 버스(24) 및 제어 신호들(26)에서의 신호들을 사용하는 요청 명령을 수행한다. 일부 명령들에 있어서, 데이터는 데이터 버스(24) 및 제어 신호들(26)을 통하여 데이터 처리 시스템(10)에 기록된다. 다른 명령들에 있어서, 데이터는 데이터 버스(24) 및 제어 신호들(26)을 통하여 데이터 처리 시스템(10)으로부터 판독되고, 직렬 데이터 스트림으로서 백그라운드 통신 인터페이스(52)를 통하여 호스트 개발 시스템에 역으로 전달된다. 일 실시예에서, 미리 결정된 통신 프로토콜에 따라, 모든 명령들과 직렬 명령 디코더(56)에 의해 처리된 데이터 및 명령 응답 블록(58)은 도 4에 도시된 심볼 타이밍(symbol timing)에 따른다. 이러한 예제 프로토콜에서, 호스트 개발 시스템을 백그라운드 통신 인터페이스(52)에 연결하는 선은, 본 실시예에서는 백그라운드 디버그 클록(54)의 16 사이클들인 심볼 지속기간의 대략 $3\frac{1}{4}$ 보다 그 이상 동안 로우로 어서팅되지 않는다.(하기 기재에서, 백그라운드 통신 인터페이스(52)가 또한 호스트 개발 시스템으로부터 백그라운드 통신 인터페이스(52)에 연결된 통신선이라고 지칭될 수 있다는 것에 주목한다. 즉, 호스트 개발 시스템에 연결될 때, 백그라운드 통신 인터페이스(52)는 또한 백그라운드 통신선(52)으로 지칭될 수 있다.)

<26> 도 4는 백그라운드 통신선(52) 상의 정규 직렬 명령들 및 데이터를 위해 논리 1 심볼 지속기간(즉, 비트 타임) 및 논리 0 심볼 지속기간(즉, 비트 타임)를 위한 타이밍을 도시한다. 본 실시예에서, 각각의 심볼은 하강 에지(falling edge)로 시작하며, 백그라운드 디버그 클록(54)의 16 사이클들이다. 각각의 비트 타임을 위한 논리값은 "SAMPLING POINTS" 라벨에 의해 도 4에 도시된 바와 같이, 비트 타임의 중간에 가깝게 샘플링된다. 논리 1 심볼의 경우에, 신호는 비트 타임의 중간에서 샘플링될 때 논리 1이도록 비트 타임의 대략 $1\frac{1}{4}$ 동안 로우로 어서팅된다. 또한, 신호는 샘플링될 때 신호에 정정값(논리 1)을 주기 위하여 샘플링 포인트 이전에 디어서팅되는 동안은 소정양의 시간 동안 로우로 어서팅될 수 있다. 유사하게, 논리 0 심볼의 경우에, 신호는 비트 타임의 중간에서 샘플링될 때 논리 0에 있도록 비트 타임의 대략 $3\frac{1}{4}$ 동안 로우로 어서팅된다. 또한, 신호는 새로운 심볼의 시작 전에 디어서팅되고 샘플링될 때 신호에 정정값(논리 0)을 주기 위하여 샘플링 포인트에서 로우로 어서팅되는 동안은 소정량의 시간동안 로우로 어서팅될 수 있다. 즉, 논리 0 심볼 동안, 신호는 새로운 심볼의 시작 이전의 어떤 시간에서 디어서팅되기 전에 현재 심볼의 지속기간 이상 로우로 어서팅된 채로 있다. 본 실시예에

서, 결코 정규 디버그 통신 중에는 논리 0 경우 동안 보다 더 긴 동안 로우로 어서팅된 백그라운드 통신선(52)이 아니다.

<27> 또한, 도 4는 상기 기재된 예제 프로토콜을 사용하는, 동기화(동기) 요청 및 동기화(동기) 응답 동안의 타이밍을 도시한다. 동기 요청은 외부 호스트 개발 시스템이 백그라운드 디버그 클록(54)의 적어도 128 사이클들 동안 로우로 백그라운드 통신선(52)을 어서팅할 때 착수된다. 데이터 처리 시스템(10)(즉, 타겟 시스템)에서 동기 검출 블록(64)이 이러한 동기 요청을 검출할 때, 백그라운드 통신선(52)이 디어서팅된 하이 레벨로 되돌아 갈 때까지 대기한다. 다음에, 동기 응답 블록(66)은 백그라운드 디버그 클록(54)의 소수의 사이클들 동안 (백그라운드 통신선(52)을 위해 요청과 응답 사이의 클리어 블레이크(clear break)를 제공하는 하이 상태로 적어도 되돌아가기에 충분한) 지연하고, 다음에 백그라운드 디버그 클록(54)의 128 사이클들 동안 백그라운드 통신선(52)을 어서팅한다. 다음에, 외부 호스트 개발 시스템은 후속하는 백그라운드 통신들을 위해 정확한 속도를 결정하도록 이런 로우인 동기 응답 신호의 지속기간을 측정할 수 있다.

<28> 상기 기재된 바와 같이, 외부 호스트 개발 시스템은 디버그 동작들을 수행하는 정확한 프로세서 속도를 검출할 수 없다. 따라서, 동기 요청 및 응답 기구는 호스트 개발 시스템이 백그라운드 통신들을 위해 타겟 시스템으로 정확한 클록 속도를 결정하는 것을 허용한다. 요약하면, 호스트 개발 시스템은 정규 통신들 동안 어서팅될 수 있는 것보다 더 많은 동안 백그라운드 통신 신호를 어서팅함으로써 타겟 시스템으로부터 동기화 타이밍 펄스를 요청한다. 이러한 요청을 인지하면, 타겟 시스템은 통신 클록 신호의 특정 수의 사이클들 동안 백그라운드 통신 신호를 어서팅함으로써 응답한다. 호스트는 정확한 통신 속도를 결정하기 위하여 이러한 응답 펄스의 길이를 추정한다. 따라서, 동기 요청 및 응답은 제 1 데이터 처리 시스템을 제 2 데이터 처리 시스템으로 동기화하기 위해 사용될 수 있고, 단지 호스트 및 타겟 시스템들에만 제한되지 않는다.

<29> 대안의 실시예들은 서로 다른 지속기간 및 포맷을 갖기 때문에 심볼들을 규정할 수 있는 다른 통신 프로토콜들에 따라 통신할 수 있다. 예를 들어, 심볼은 도 4에 도시된 16 사이클들보다 많거나 적은 지속기간을 가질 수 있으며, 심볼 지속기간 동안 서로 다른 포인트에서 샘플링될 수 있다. 따라서, 동기 요청은 상기 예시된 128 사이클들보다 더 많거나 적을 수 있다. 일반적으로, 동기 요청은 실질적으로 최대 심볼 지속기간보다 더 길다. 예를 들어, 도 4의 실시예에서, 128 사이클은 적어도 (정규 심볼 지속기간에 대응하는) 16 사이클들보다 더 길고, 클록 신호들에서 충분한 허용 공차와 소정의 측정 에러들을 고려한다. 어떤 실시예들에서, 동기 요청의 지속기간은 적어도 정규 심볼의 지속기간의 2배가 되게 규정될 수 있다. 한편, 타겟 시스템에 의해 전송되는 동기 응답은, 호스트 시스템이 전송된 고정수의 사이클들을 이용하여 적당한 타임 측정을 하게 하도록 고정된 지속기간이다. 그러나, 호스트가 타겟 시스템에 비동기적일 수 있기 때문에, 동기 응답의 측정에서 어떤 불확실성이 있을 수 있다. 따라서, (정규 심볼의 지속기간에 비하여) 충분히 긴 지속기간은 측정 불확실성의 영향을 최소화하도록 동기 응답에 대해 선택되어, 호스트 시스템이 보다 정확한 타이밍 정보를 얻는 것을 허용하여야 한다. 그래도, 대안의 실시예들은 동기 요청 및 동기 응답의 지속기간들을 위해 서로 다른 많은 조합들을 사용할 수 있다.

<30> 데이터 처리 시스템(10)의 대안의 실시예에서, BDS(14)는 클록 유닛(19)과 관계없는, 레지스터-캐패시터(RC) 발진기 등의 독립된 발진기(self-contained oscillator)를 포함할 수 있다. 이런 시스템에서, STOP 신호(68)를 오버라이드하는데 사용되는 것과 유사한 논리가 독립된 발진기를 적당히 인에이블하는데 사용될 수 있다. 예를 들어, 상기 독립된 발진기는, 유효한 통신이 백그라운드 통신선(52)에서 검출될 때, 시스템 발진기의 동작과 관계없이 인에이블될 수 있다. 따라서, 데이터 처리 시스템을 위한 시스템 발진기는 정지 모드로 놓일 수 있고, BDS(14)는 여전히 계속 무관한 디버그 동작들을 수행할 수 있다. RC 발진기 등의 독립된 발진기가 BDM(14)에 의해 사용되면, 호스트 시스템은 일반적으로 그 타이밍 정보의 이전 지식을 갖고 있지 않다. 이는 일부분 RC 발진기들이 일반적으로 넓은 주파수 허용 공차들을 갖기 때문에, 상기 주파수를 알기 어렵다. 그러나, 상술된 바와 같이, 호스트 시스템은 동기 응답을 통하여 DBM(14)로부터 적당한 타이 및 정보를 얻기 위하여 동기 요청을 사용할 수 있다.

<31> 특정한 도전성 형태들 또는 전위들의 극성에 대해 본 발명을 기재하였지만, 숙련된 자들은 도전성 타입들 및 전위들의 극성들이 반대될 수 있다는 것을 이해할 수 있다. 예를 들어, 액티브 하이로 설계된 신호들은 액티브 로우로 설계될 수 있고, 액티브 로우로 설계된 신호들은 액티브 하이로 설계될 수 있다. 본 분야의 통상의 기술자는 이러한 변화들을 수용하기 위하여 회로를 어떻게 조정하는지를 이해할 것이다.

<32> 전술한 명세서에서는 특정 실시예들을 참조하여 본 발명을 기재하였다. 그러나, 다양한 변형들 및 변경들이 하기 청구범위에 설명되는 본 발명의 범위로부터 벗어나지 않고 이루어질 수 있다는 것을 본 분야의 통상의 기술

자는 이해할 것이다. 따라서, 명세서 및 도면은 제한하기보다는 예시하는 의미로 간주되며, 이러한 모든 변형들은 본 발명의 범위내에 포함되게 한다.

<33> 특정 실시예들에 관하여 이점들, 다른 장점들 및 문제들에 대한 해결책들을 상술하였다. 그러나, 어떤 이점, 장점 또는 해결책이 발생하거나 보다 명확해지게 할 수 있는 이점들, 장점들, 문제점들에 대한 해결책들, 및 어떤 요소(들)은 어떤 또는 모든 청구범위의 중대한, 요구되는, 또는 필수적인 특징이나 요소로서 해석되지 않는다. 본원에 사용되는 바와 같이, "포함하다", "포함하는"의 용어 또는 그의 어떤 다른 변화는, 요소들의 리스트를 포함하는 공정, 방법, 물품, 또는 장치가 그의 요소들만을 포함하지 않고 명백하게 이러한 공정, 방법, 물품, 또는 장치에 리스트되지 않거나 고유하지 않는 다른 요소들을 포함할 수 배타적이지 않는 포함을 커버하도록 의도된다.

도면의 간단한 설명

<7> 도 1은 본 발명의 일 실시예를 도시하는 데이터 처리 시스템을 블록 다이어그램 형태로 도시하는 도면.

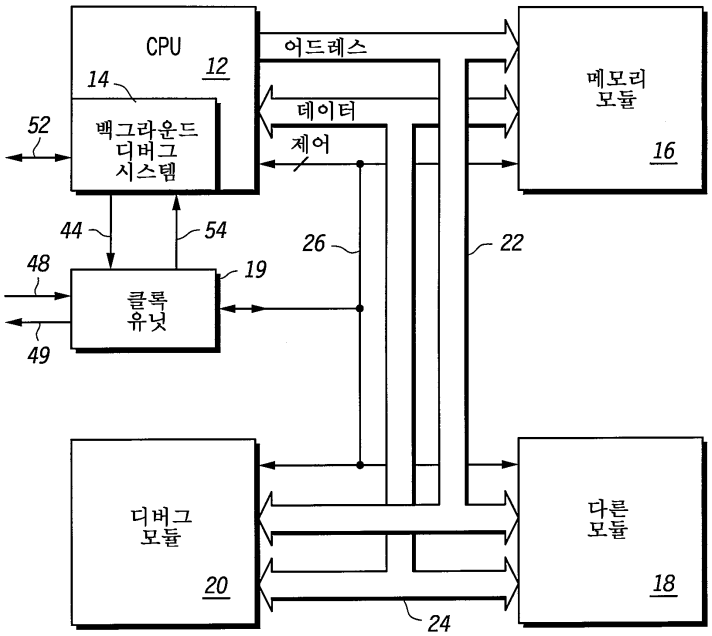
<8> 도 2는 본 발명의 일 실시예에 따라 도 1의 중앙 처리 장치 및 클록 유닛의 일부를 부분적인 블록 다이어그램 및 부분적인 개략도 형태로 도시하는 도면.

<9> 도 3은 본 발명의 일 실시예에 따라 도 1의 백그라운드 디버그 시스템을 블록 다이어그램 형태로 도시하는 도면.

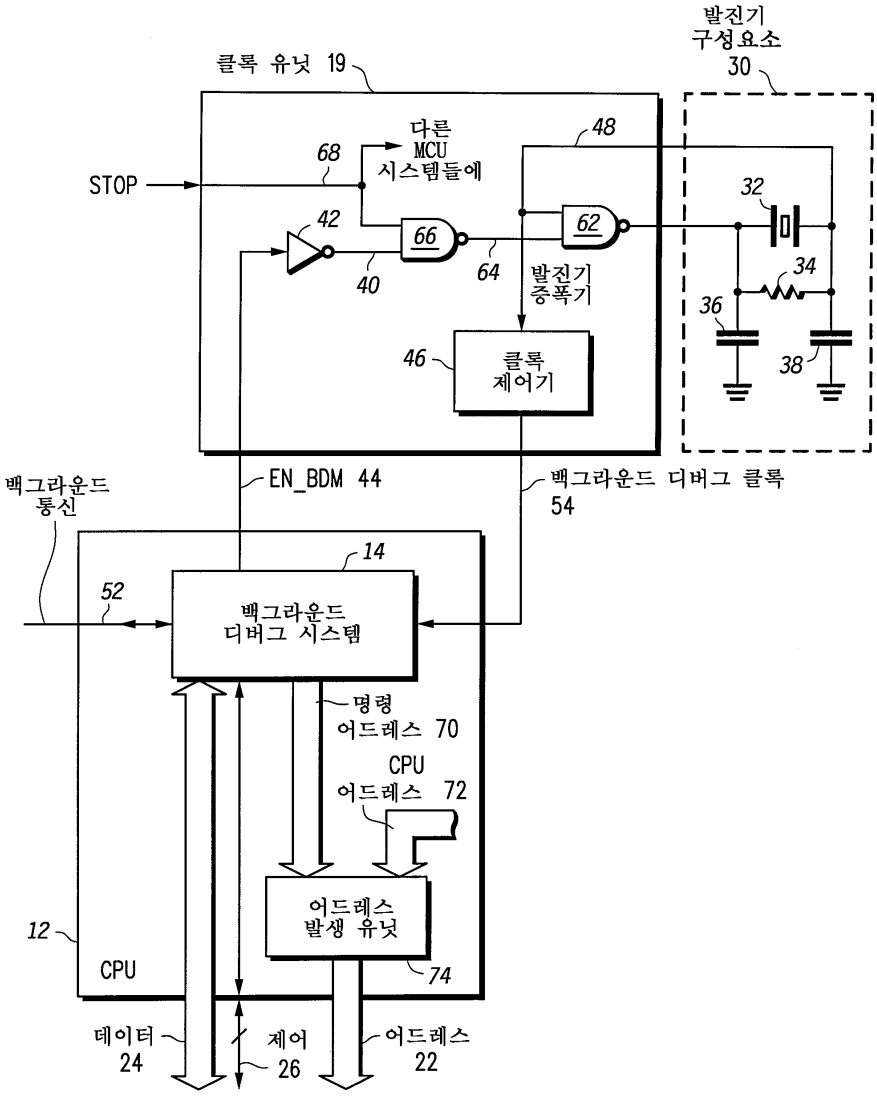
<10> 도 4는 본 발명의 일 실시예에 따라 논리 1 심볼, 논리 0 심볼, 및 동조 요청과 응답을 타이밍 다이어그램 형태로 도시하는 도면.

도면

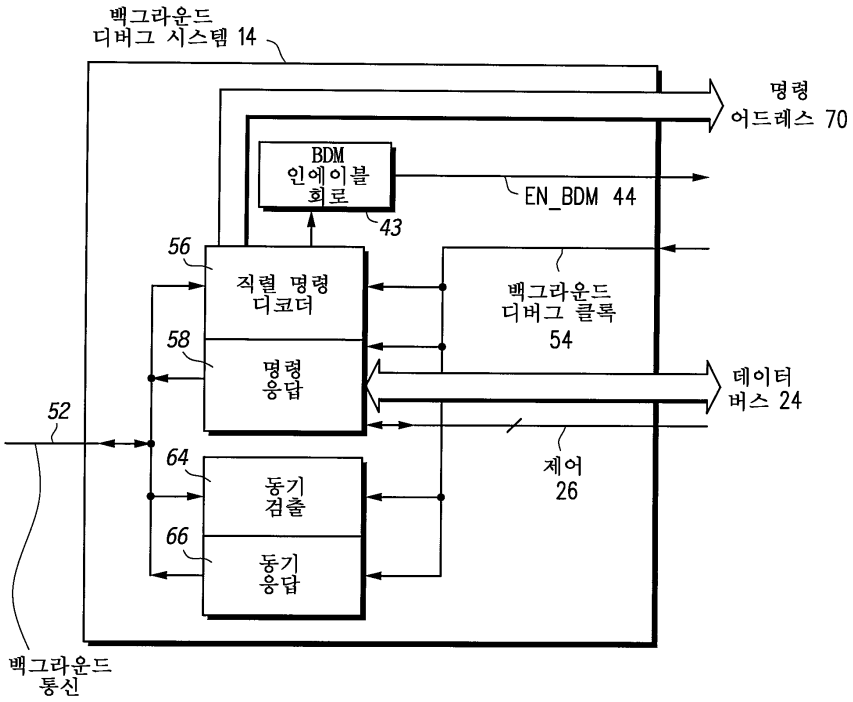
도면1



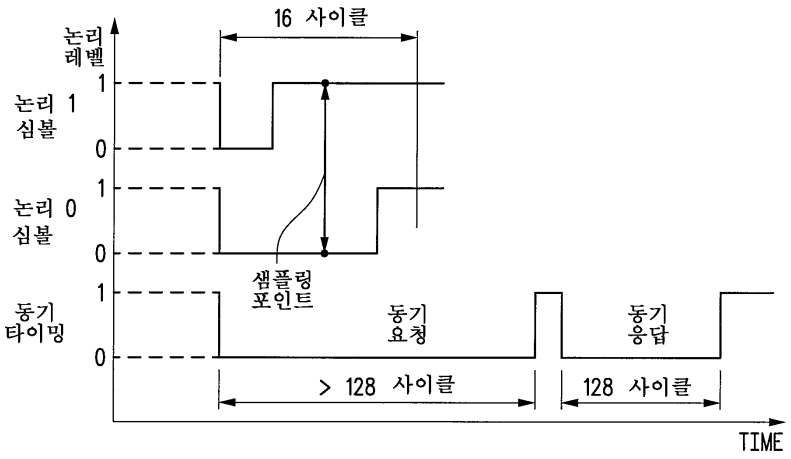
도면2


발진기
구성요소
30
클록 유닛 19
다른
MCU
시스템들에
68
48
STOP
32
42
66
62
34
64
발진기
증폭기
40
36
46
클록
제어기
38
백그라운드
통신
EN_BDM 44
백그라운드 디버그 클록
54
14
52
백그라운드
디버그 시스템
명령
어드레스 70
CPU
어드레스 72
어드레스
발생 유닛
74
12
CPU
데이터
24
제어
26
어드레스
22

도면3


백그라운드
디버그 시스템 14
명령
어드레스 70
BDM
인에이블
회로
43
EN_BDM 44
56
직렬 명령
디코더
58
명령
응답
백그라운드
디버그 클록
54
데이터
버스 24
52
64
동기
검출
66
동기
응답
제어
26
백그라운드
통신

도면4


16 사이클
논리
레벨
논리 1
심볼
논리 0
심볼
동기
타이밍
1
0
1
0
1
0
샘플링
포인트
동기
요청
동기
응답
> 128 사이클
128 사이클
TIME