(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2020-0123799
(43) 공개일자 2020년10월30일

(51) 국제특허분류(Int. Cl.)
G06F 11/36 (2006.01) *G06F 11/10* (2006.01)
G06F 13/42 (2006.01)
(52) CPC특허분류
G06F 11/3656 (2013.01)
G06F 11/10 (2013.01)
(21) 출원번호 10-2020-7026137
(22) 출원일자(국제) 2019년01월17일
심사청구일자 없음
(85) 번역문제출일자 2020년09월10일
(86) 국제출원번호 PCT/GB2019/050129
(87) 국제공개번호 WO 2019/166762
국제공개일자 2019년09월06일
(30) 우선권주장
1803179.9 2018년02월27일 영국(GB)

(71) 출원인
에이알엠 리미티드
영국 캠브리지 씨비1 9엔제이 체리힌톤 풀번로드 110
(72) 발명자
윌리암스 마이클 존
영국 캠브리지 캠브리지셔 씨비1 9엔제이 체리힌톤 풀번로드 110 에이알엠 리미티드
홀리 존 마이클
영국 캠브리지 캠브리지셔 씨비 1 9엔제이, 체리힌톤 풀번로드 110, 에이알엠 리미티드
(74) 대리인
이화익

전체 청구항 수 : 총 22 항

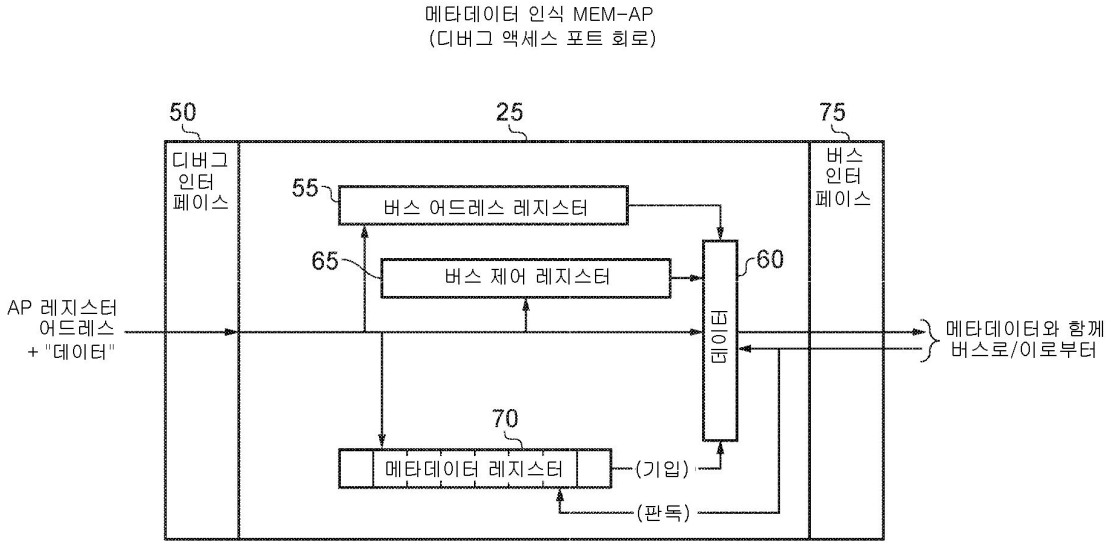
(54) 발명의 명칭 **디바이스를 디버깅할 때 메타데이터에 액세스하기 위한 장치 및 방법**

(57) 요 약

디바이스를 디버깅할 때 메타데이터에 액세스하기 위한 장치 및 방법이 제공된다. 특히, 디버그 액세스 포트 회로는 디버거로부터 커맨드를 수신하는 디버그 인터페이스, 및 디버거가 디바이스의 메모리 시스템에 액세스할 수 있게 버스에 결합하는 버스 인터페이스를 갖는다. 디바이스는 관련된 메타데이터 아이템을 갖는 데이터 그래뉼

(뒷면에 계속)

대 표 도



형태의 데이터 상에서 동작하도록 배열되고, 버스 인터페이스는 메모리 시스템과 버스 인터페이스 사이의 버스를 통한 데이터 그래뉼 및 메타데이터 아이템 모두의 통신을 할 수 있게 하도록 배열된다. 디버그 액세스 포트 회로는 버스 인터페이스를 통한 메모리 시스템 내에서 수행된 액세스가 커맨드에 의해서 액세스된 스토리지 엘리먼트에 의존해서 제어되도록 디버거로부터 발행된 커맨드를 통해서 액세스 가능한 복수의 스토리지 엘리먼트를 갖는다. 적어도 하나의 스토리지 엘리먼트는, 복수의 메타데이터 아이템을 저장하는 메타데이터 스토리지 엘리먼트와, 메타데이터 스토리지 엘리먼트와 메모리 시스템 사이에서 적어도 하나의 복수의 메타데이터 아이템을 전송하기 위해서, 디버거로부터의 적어도 하나의 커맨드에 응답해서, 메모리 시스템에 대한 액세스를 수행한다. 이는, 디버거가, 디버그되는 디바이스의 메모리 시스템 내에서 메타데이터 아이템에 직접 액세스할 수 있게 하기 위한 효율적인 메커니즘을 제공한다.

(52) CPC특허분류

G06F 13/4234 (2013.01)

명 세 서

청구범위

청구항 1

디버그 액세스 포트 회로로서:

디버거로부터 커맨드를 수신하는 디버그 인터페이스와;

디버거가 디바이스의 메모리 시스템에 액세스할 수 있게 하기 위해서 버스에 결합하는 버스 인터페이스를 포함하고, 디바이스는 관련된 메타데이터 아이템을 갖는 데이터 그래뉼 형태의 데이터 상에서 동작하고, 버스 인터페이스는 메모리 시스템과 버스 인터페이스 사이의 버스를 통한 데이터 그래뉼 및 메타데이터 아이템 모두의 통신을 할 수 있게 하며;

버스 인터페이스를 통한 메모리 시스템 내에서 수행된 액세스가 커맨드에 의해서 액세스된 스토리지 엘리먼트에 의존해서 제어되도록 디버거로부터 발행된 커맨드를 통해서 액세스 가능한 복수의 스토리지 엘리먼트와;

복수의 메타데이터 아이템을 저장하는 메타데이터 스토리지 엘리먼트를 포함하는 적어도 하나의 스토리지 엘리먼트를 더욱 포함하고;

디버그 액세스 포트 회로는, 메타데이터 스토리지 엘리먼트와 메모리 시스템 사이에서 적어도 하나의 복수의 메타데이터 아이템을 전송하기 위해서, 디버거로부터의 적어도 하나의 커맨드에 응답해서, 메모리 시스템에 대한 액세스를 수행하는, 디버그 액세스 포트 회로.

청구항 2

제1항에 있어서,

적어도 하나의 스토리지 엘리먼트는 액세스되는 메모리 어드레스의 인디케이션을 저장하는 메모리 어드레스 스토리지 엘리먼트를 포함하고;

디버그 액세스 포트 회로는, 적어도 하나의 복수의 메타데이터 아이템에 액세스하는 메모리 시스템 내의 위치를 결정하기 위해서 메모리 어드레스 스토리지 엘리먼트를 참조하도록 배열되는, 디버그 액세스 포트 회로.

청구항 3

제2항에 있어서,

디버그 액세스 포트 회로는, 적어도 하나의 복수의 메타데이터 아이템에 대해서 사용된 메타데이터 스토리지 엘리먼트 내의 위치를 결정하기 위해서 메모리 어드레스 스토리지 엘리먼트를 참조하도록 더 배열되는, 디버그 액세스 포트 회로.

청구항 4

제3항에 있어서,

메모리 어드레스의 특정 수의 하위 어드레스 비트가 메타데이터 스토리지 엘리먼트 내의 위치를 결정하기 위해서 사용되는, 디버그 액세스 포트 회로.

청구항 5

제3항 또는 제4항에 있어서,

메모리 시스템으로부터의 판독 액세스 동안, 메모리 시스템으로부터 검색된 메타데이터 아이템은 메타데이터 스토리지 엘리먼트 내의 결정된 위치에 기입되는 한편, 메타데이터 스토리지 엘리먼트 내의 다른 위치의 콘텐츠는 변경되지 않는, 디버그 액세스 포트 회로.

청구항 6

제1항 또는 제2항에 있어서,

메타데이터 스토리지 엘리먼트는 시프트 레지스터 구조로서 배열되어, 시프트 동작이 메모리 시스템으로부터의 판독 액세스 동안 메타데이터 스토리지 엘리먼트 내에 메타데이터 아이템을 부가, 및 메모리 시스템에 대한 기입 액세스 동안 메타데이터 스토리지 엘리먼트로부터 메타데이터 아이템을 출력하기 위해서 사용되는, 디버그 액세스 포트 회로.

청구항 7

선행하는 청구항 중 어느 한 항에 있어서,

복수의 스토리지 엘리먼트는, 디버거로부터 커맨드를 통해서 액세스될 때, 메모리 시스템에 대한 액세스가 디버그 액세스 포트 회로에 의해서 개시되게 하는, 하나 이상의 액세스 트리거 엘리먼트를 더 포함하는, 디버그 액세스 포트 회로.

청구항 8

제7항에 있어서,

복수의 스토리지 엘리먼트는, 상기 하나 이상의 액세스 트리거 엘리먼트가 액세스될 때 수행되는 액세스 동작의 타입을 식별하기 위해서 디버거로부터의 커맨드를 통해서 액세스 가능한 모드 제어 스토리지 엘리먼트를 더 포함하는, 디버그 액세스 포트 회로.

청구항 9

제7항에 있어서,

상기 하나 이상의 액세스 트리거 엘리먼트는 복수의 액세스 트리거 엘리먼트를 포함하고, 각각의 액세스 트리거 엘리먼트는 다른 타입의 액세스 동작에 관련되며, 디버거는, 메모리 시스템에 대한 액세스 동작을 개시하는 커맨드를 발행할 때, 개시되는 액세스 동작의 타입에 대한 적합한 액세스 트리거 엘리먼트를 커맨드 내에서 식별하기 위해서 배열되는, 디버그 액세스 포트 회로.

청구항 10

선행하는 청구항 중 어느 한 항에 있어서,

디버그 액세스 포트 회로는, 디버거로부터 수신된 커맨드에 의존해서, 지원된 액세스 동작의 세트로부터 액세스 동작을, 버스 인터페이스를 통해서, 개시하기 위해서 배열되는, 디버그 액세스 포트 회로.

청구항 11

제10항에 있어서,

지원된 액세스 동작의 세트는 하나 이상의 다음 기입 동작을 포함하는데, 다음은:

메타데이터 스토리지 엘리먼트로부터 획득된 값을 사용해서 메타데이터 아이템의 값을 갱신하는, 및 디버거로부터 커맨드에 의해서 제공된 데이터 값을 사용해서 관련된 데이터 그래뉼의 적어도 부분의 현재 값을 갱신하는 기입 동작;

메타데이터 스토리지 엘리먼트로부터 획득된 값을 사용해서 메타데이터 아이템의 값을 갱신하는 한편, 관련된 데이터 그래뉼의 현재 값을 보존하는 기입 동작;

메타데이터 스토리지 엘리먼트로부터 획득된 값을 사용해서 메타데이터 아이템의 값을 갱신하는, 및 관련된 데이터 그래뉼의 적어도 부분의 값을 공지된 값으로 갱신하는 기입 동작;

데이터 그래뉼의 적어도 부분을 메모리 시스템에 기입하여, 관련된 메타데이터 아이템의 현재 값을 보존하는, 기입 동작;

데이터 그래뉼의 적어도 부분을 메모리 시스템에 기입하여, 관련된 메타데이터 아이템의 값을 디폴트 값으로 갱신하는, 기입 동작인, 디버거 액세스 포트 회로.

청구항 12

제10항에 있어서,

지원된 액세스 동작의 세트는 하나 이상의 다음 판독 동작을 포함하는데, 다음은:

메모리 시스템으로부터 메타데이터 아이템을 판독하여 메타데이터 스토리지 엘리먼트 내에 그 판독 메타데이터 아이템의 값을 저장하는, 및 메모리 시스템으로부터 관련된 데이터 그래뉼의 적어도 부분을 추가적으로 판독하는 판독 동작;

관련된 데이터 그래뉼을 추가적으로 판독하지 않고, 메모리 시스템으로부터 메타데이터 아이템을 판독하여 메타데이터 스토리지 엘리먼트 내에 그 판독 메타데이터 아이템의 값을 저장하는 판독 동작;

관련된 메타데이터 아이템을 판독하지 않고, 메모리 시스템으로부터 데이터 그래뉼의 적어도 부분을 판독하는 판독 동작인, 디버거 액세스 포트 회로.

청구항 13

선행하는 청구항 중 어느 한 항에 있어서,

각각의 메타데이터 아이템은 하나 이상의 비트를 포함하는, 디버그 액세스 포트 회로.

청구항 14

선행하는 청구항 중 어느 한 항에 있어서,

각각의 메타데이터 아이템은, 관련된 데이터 그래뉼이 능력을 특정하는지를 식별하는 능력 태그인, 디버그 액세스 포트 회로.

청구항 15

선행하는 청구항 중 어느 한 항에 있어서,

각각의 메타데이터 아이템은 관련된 데이터 그래뉼의 할당 정책을 식별하는 할당 태그인, 디버그 액세스 포트 회로.

청구항 16

선행하는 청구항 중 어느 한 항에 있어서,

디버그 인터페이스는 JTAG, 직렬 와이어, 또는 PCI 인터페이스 중 하나인, 디버그 액세스 포트 회로.

청구항 17

선행하는 청구항 중 어느 한 항에 있어서,

복수의 스토리지 엘리먼트는 어드레스 스페이스 내에 존재하고, 디버그 인터페이스에서 수신된 커맨드는 어드레스 스페이스 내의 어드레스를 특정함으로써 그 커맨드가 관련되는 스토리지 엘리먼트를 식별하는, 디버그 액세스 포트 회로.

청구항 18

선행하는 청구항 중 어느 한 항에 있어서,

복수의 스토리지 엘리먼트는 복수의 레지스터를 포함하는, 디버그 액세스 포트 회로.

청구항 19

제2항에 의존할 때 선행하는 청구항 중 어느 한 항에 있어서,

디버그 액세스 포트 회로는, 액세스 동작의 다수의 반복(iteration)을, 버스 인터페이스를 통해서, 개시하도록 디버거로부터의 커맨드에 응답하고, 각각의 반복에 대해서, 메모리 어드레스 스토리지 엘리먼트 내에 저장된 메모리 어드레스 인디케이션은 증분되는, 디버그 액세스 포트 회로.

청구항 20

제19항에 있어서,

액세스 동작이 메타데이터 아이템 및 액세스되는 데이터 모두를 요구할 때, 메모리 어드레스 인디케이션은, 다음 반복에 대한 어드레스 인디케이션을 제공하기 위해서, 현재 반복에서 액세스된 데이터의 양에 기반해서 증분되고;

액세스 동작이 액세스되는 메타데이터 아이템만을 요구할 때, 메모리 어드레스 인디케이션은 데이터 그래뉼 사이즈를 n배함으로써 증분되며, n은, 다음 반복에 대한 어드레스 인디케이션을 제공하기 위해서, 현재 반복에서 액세스된 메타데이터 아이템의 수와 동등한, 디버그 액세스 포트 회로.

청구항 21

디버그 액세스 포트 회로를 동작하는 방법으로서:

디버거로부터 커맨드를 수신하는 단계와;

디버거가 디바이스의 메모리 시스템에 액세스할 수 있게 하기 위해서 버스에 결합하는 버스 인터페이스를 제공하는 단계로서, 디바이스는 관련된 메타데이터 아이템을 갖는 데이터 그래뉼 형태의 데이터 상에서 동작하고, 버스 인터페이스는 메모리 시스템과 버스 인터페이스 사이의 버스를 통한 데이터 그래뉼 및 메타데이터 아이템 모두의 통신을 할 수 있게 하는, 제공하는 단계와;

버스 인터페이스를 통한 메모리 시스템 내에서 수행된 액세스가 커맨드에 의해서 액세스된 스토리지 엘리먼트에 의존해서 제어되도록 디버거로부터 발행된 커맨드를 통해서 액세스 가능한 복수의 스토리지 엘리먼트를 제공하는 단계와;

복수의 메타데이터 아이템을 저장하는 메타데이터 스토리지 엘리먼트로서 적어도 하나의 스토리지 엘리먼트를 제공하는 단계와;

메타데이터 스토리지 엘리먼트와 메모리 시스템 사이에서 적어도 하나의 복수의 메타데이터 아이템을 전송하기 위해서, 디버거로부터의 적어도 하나의 커맨드에 응답해서, 메모리 시스템에 대한 액세스를 수행하는 단계를 포함하는, 방법.

청구항 22

디버그 액세스 포트 회로로서:

디버거로부터 커맨드를 수신하기 위한 디버그 인터페이스 수단과;

디버거가 디바이스의 메모리 시스템에 액세스할 수 있게 하기 위해서 버스에 결합하기 위한 버스 인터페이스 수단을 포함하고, 디바이스는 관련된 메타데이터 아이템을 갖는 데이터 그래뉼 형태의 데이터 상에서 동작하고, 버스 인터페이스 수단은 메모리 시스템과 버스 인터페이스 수단 사이의 버스를 통한 데이터 그래뉼 및 메타데이터 아이템 모두의 통신을 할 수 있게 하며;

버스 인터페이스 수단을 통한 메모리 시스템 내에서 수행된 액세스가 커맨드에 의해서 액세스된 스토리지 엘리먼트 수단에 의존해서 제어되도록 디버거로부터 발행된 커맨드를 통해서 액세스 가능한 복수의 스토리지 엘리먼트 수단과;

복수의 메타데이터 아이템을 저장하기 위한 메타데이터 스토리지 엘리먼트 수단을 포함하는 적어도 하나의 스토리지 엘리먼트 수단을 더욱 포함하고;

디버그 액세스 포트 회로는, 메타데이터 스토리지 엘리먼트 수단과 메모리 시스템 사이에서 적어도 하나의 복수의 메타데이터 아이템을 전송하기 위해서, 디버거로부터의 적어도 하나의 커맨드에 응답해서, 메모리 시스템에 대한 액세스를 수행하는, 디버그 액세스 포트 회로.

발명의 설명

기 술 분 야

[0001] 본 개시는 디바이스를 디버깅할 때 메타데이터에 액세스하기 위한 기술과 관련된다.

배 경 기 술

[0002] 디바이스는 다양한 형태를 취할 수 있고, 예를 들어, 디바이스의 메모리 시스템 내에 유지된 데이터에 대한 데이터 처리 동작을 위한 중앙 처리 유닛(CPU)과 같은 하나 이상의 처리 회로를 포함할 수 있다.

[0003] 디바이스는, 관련된 메타데이터 아이템을 갖는 데이터 그래뉼(data granule)로 형성된 데이터 상에서 동작하도록 배열될 수 있다. 데이터 그래뉼의 사이즈는 구현에 따라서 변할 수 있고, 실제로 각각의 데이터 그래뉼과 관련된 메타데이터 아이템의 사이즈는 구현에 따라서 변할 수 있다.

[0004] 디바이스에 대해서, 예를 들어, 처리 회로 중 하나에 대해서 실행되는 프로그램을 디버그하기 위해서 디버깅 동작을 수행한 때, 시간 내의 소정의 포인트에서 디버거는 대응하는 데이터 그래뉼에 관련되는 하나 이상의 메타데이터 아이템에 액세스하는 것을 원할 수 있다. 예를 들어, 각각의 이들 메타데이터 아이템의 값을 판독하는 것을 희망할 수 있다. 대안적으로, 각각의 이들 메타데이터 아이템의 현재 값을 갱신하기 위해서 기입 동작을 수행하는 것을 희망할 수 있다.

[0005] 디바이스의 처리 회로는 하나 이상의 메타데이터 아이템에 액세스하는 메타데이터 액세스 동작의 실행을 지원할 수 있고, 따라서 디버거는, 디버거가 메타데이터에 액세스하는 것을 원할 때, 이러한 메타데이터 액세스 동작을 수행하게 하도록 처리 회로에 커맨드를 발행할 수 있다.

발명의 내용

해결하려는 과제

[0006] 그런데, 이는, 관심의 메타데이터 아이템에 액세스하는데 있어서 디버거를 지원하기 위해서 처리 회로에 의해서 수행되는 현재 동작을 중단하는 것을 요구할 수 있다. 디바이스의 디버깅 동안 메타데이터 아이템에 액세스하기 위한 개선된 메커니즘을 제공하는 것이 바람직할 것이다.

과제의 해결 수단

[0007] 제1예의 배열에 있어서, 디버그 액세스 포트 회로가 제공되는데, 이는: 디버거로부터 커맨드를 수신하는 디버그 인터페이스와; 디버거가 디바이스의 메모리 시스템에 액세스할 수 있게 하기 위해서 버스에 결합하는 버스 인터페이스를 포함하고, 디바이스는 관련된 메타데이터 아이템을 갖는 데이터 그래뉼 형태의 데이터 상에서 동작하고, 버스 인터페이스는 메모리 시스템과 버스 인터페이스 사이의 버스를 통한 데이터 그래뉼 및 메타데이터 아이템 모두의 통신을 할 수 있게 하며; 버스 인터페이스를 통한 메모리 시스템 내에서 수행된 액세스가 커맨드에 의해서 액세스된 스토리지 엘리먼트에 의존해서 제어되도록 디버거로부터 발행된 커맨드를 통해서 액세스 가능한 복수의 스토리지 엘리먼트와; 복수의 메타데이터 아이템을 저장하는 메타데이터 스토리지 엘리먼트를 포함하는 적어도 하나의 스토리지 엘리먼트를 더욱 포함하고; 여기서 디버그 액세스 포트 회로는, 메타데이터 스토리지 엘리먼트와 메모리 시스템 사이에서 적어도 하나의 복수의 메타데이터 아이템을 전송하기 위해서, 디버거로부터의 적어도 하나의 커맨드에 응답해서, 메모리 시스템에 대한 액세스를 수행한다.

[0008] 또 다른 예의 배열에 있어서, 디버그 액세스 포트 회로를 동작하는 방법이 제공되는데, 이는: 디버거로부터 커맨드를 수신하는 것과; 디버거가 디바이스의 메모리 시스템에 액세스할 수 있게 하기 위해서 버스에 결합하는 버스 인터페이스를 제공하는 것을 포함하고, 디바이스는 관련된 메타데이터 아이템을 갖는 데이터 그래뉼 형태의 데이터 상에서 동작하고, 버스 인터페이스는 메모리 시스템과 버스 인터페이스 사이의 버스를 통한 데이터 그래뉼 및 메타데이터 아이템 모두의 통신을 할 수 있게 하며; 버스 인터페이스를 통한 메모리 시스템 내에서 수행된 액세스가 커맨드에 의해서 액세스된 스토리지 엘리먼트에 의존해서 제어되도록 디버거로부터 발행된 커맨드를 통해서 액세스 가능한 복수의 스토리지 엘리먼트를 제공하는 것과; 복수의 메타데이터 아이템을 저장하는 메타데이터 스토리지 엘리먼트로서 적어도 하나의 스토리지 엘리먼트를 제공하는 것과; 메타데이터 스토리지 엘리먼트와 메모리 시스템 사이에서 적어도 하나의 복수의 메타데이터 아이템을 전송하기 위해서, 디버거로부터의 적어도 하나의 커맨드에 응답해서, 메모리 시스템에 대한 액세스를 수행하는 것을 포함한다.

[0009] 또 다른 예의 구성에 있어서, 디버그 액세스 포트 회로가 제공되는데, 이는: 디버거로부터 커맨드를 수신하기 위한 디버그 인터페이스 수단과; 디버거가 디바이스의 메모리 시스템에 액세스할 수 있게 하기 위해서 버스에 결합하기 위한 버스 인터페이스 수단을 포함하고, 디바이스는 관련된 메타데이터 아이템을 갖는 데이터 그래뉼

형태의 데이터 상에서 동작하고, 버스 인터페이스 수단은 메모리 시스템과 버스 인터페이스 수단 사이의 버스를 통한 데이터 그래뉼 및 메타데이터 아이템 모두의 통신을 할 수 있게 하며; 버스 인터페이스 수단을 통한 메모리 시스템 내에서 수행된 액세스가 커맨드에 의해서 액세스된 스토리지 엘리먼트 수단에 의존해서 제어되도록 디버거로부터 발행된 커맨드를 통해서 액세스 가능한 복수의 스토리지 엘리먼트 수단과; 복수의 메타데이터 아이템을 저장하기 위한 메타데이터 스토리지 엘리먼트를 포함하는 적어도 하나의 스토리지 엘리먼트 수단을 포함하고; 여기서 디버그 액세스 포트 회로는, 메타데이터 스토리지 엘리먼트 수단과 메모리 시스템 사이에서 적어도 하나의 복수의 메타데이터 아이템을 전송하기 위해서, 디버거로부터의 적어도 하나의 커맨드에 응답해서, 메모리 시스템에 대한 액세스를 수행한다.

발명의 효과

[0010] 본 발명에 의하면, 개시는 디바이스를 디버깅할 때 메타데이터에 액세스하기 위한 개선된 기술이 제공된다.

도면의 간단한 설명

[0011] 본 발명 기술이, 첨부 도면에서 도시된 바와 같은 그 예들을 참조로, 예시적인 방식으로만, 더 기술될 것이다:

도 1은 일례에 따른 시스템의 블록도이다;

도 2는 일례에 따른 도 1의 메타데이터 인식 메모리 액세스 포트 내에 제공된 컴포넌트를 도시하는 블록도이다;

도 3a 및 3b는, 일례의 배열에 따른, 디버거가 메타데이터 아이템의 수를 판독하기 위한 메커니즘을 도시하는 흐름도를 제공한다;

도 4a 및 4b는, 일례의 배열에 따른, 디버거가 일련의 메타데이터 아이템을 판독하기 위한 메커니즘을 도시하는 흐름도를 제공한다;

도 5는, 대안적인 배열에 따른, 도 1의 메타데이터 인식 메모리 액세스 포트 내에 제공된 컴포넌트를 도시하는 블록도이다;

도 6a 및 6b는, 어떻게 디버거가, 도 5의 메타데이터 인식 메모리 액세스 포트를 사용할 때 일련의 메타데이터 아이템을 판독할 수 있는지를 도시하는 흐름도를 제공한다;

도 7a 및 7b는, 어떻게 디버거가, 도 5의 메타데이터 인식 메모리 액세스 포트를 사용할 때 일련의 메타데이터 아이템을 기입할 수 있는지를 도시하는 흐름도이다;

도 8a 및 8b는, 어떻게 메타데이터 아이템이 각각의 데이터 그래뉼과 관련해서 제공될 수 있는지를 도시한다;

도 9는 메타데이터 아이템이 능력(capability) 태그인 일례의 배열을 도시한다;

도 10은 메타데이터 아이템이 할당 태그인 또 다른 예의 배열을 도시한다;

도 11은, 일례의 배열에 따른, 메타데이터 인식 메모리 액세스 포트에 커맨드를 발행할 수 있는 디버거에 사용 가능한 제한된 어드레스 스페이스를 개략적으로 도시하는 도면이다.

발명을 실시하기 위한 구체적인 내용

[0012] 일례의 배열에 따라서, 디버그 액세스 포트 회로는, 디바이스와 관련해서 디버깅 활동을 수행한 때 디버거에 의해서 사용하기 위해서 제공된다. 특히, 디버그 액세스 포트 회로는 디버거로부터 커맨드를 수신하는 디버그 인터페이스, 및 디버거가 디바이스의 메모리 시스템에 액세스할 수 있게 하기 위해서 버스에 결합하는 버스 인터페이스를 갖는다. 디바이스는 관련된 메타데이터 아이템을 갖는 데이터 그래뉼 형태의 데이터 상에서 동작하도록 배열되고, 버스 인터페이스는 메모리 시스템과 버스 인터페이스 사이의 버스를 통한 데이터 그래뉼 및 메타데이터 아이템 모두의 통신을 할 수 있게 하도록 배열된다.

[0013] 디버그 액세스 포트 회로는 버스 인터페이스를 통한 메모리 시스템 내에서 수행된 액세스가 커맨드에 의해서 액세스된 스토리지 엘리먼트에 의존해서 제어되도록 디버거로부터 발행된 커맨드를 통해서 액세스 가능한 복수의 스토리지 엘리먼트를 갖는다. 본 개시에 기술된 예에 따라서, 적어도 하나의 스토리지 엘리먼트는 복수의 메타데이터 아이템을 저장하는 메타데이터 스토리지 엘리먼트를 포함한다. 디버그 액세스 포트 회로는, 그 다음, 메타데이터 스토리지 엘리먼트와 메모리 시스템 사이에서 적어도 하나의 복수의 메타데이터 아이템을 전송하기 위해서, 디버거로부터의 적어도 하나의 커맨드에 응답해서, 메모리 시스템에 대한 액세스를 수행한다.

[0014] 이 방식으로 디버그 액세스 포트 회로를 배열함으로써, 이는, 그 대신 액세스 동작을 수행하기 위해서 디바이스 내에 처리 회로를 채용할 필요 없이, 메타데이터 아이템 상에서 액세스 동작을 수행하기 위해서 디버거가 디바이스의 메모리 시스템에 효율적으로 액세스하기 위한 메커니즘을 제공한다. 복수의 메타데이터 아이템을 저장할 수 있는 메타데이터 스토리지 엘리먼트를 제공함으로써, 메모리 내의 일련의 메타데이터 아이템에 대한 값은 디버거에 의해서 발행된 적합한 커맨드를 사용해서 메타데이터 스토리지 엘리먼트 내에 판독될 수 있거나, 또는 대안적으로 메타데이터 아이템에 대한 일련의 갱신된 값은 메타데이터 스토리지 엘리먼트 내에 저장될 수 있고, 그 다음, 기입 동작은, 메타데이터 스토리지 엘리먼트 내에 유지된 값을 사용해서, 메모리 시스템 내에 저장된 일련의 메타데이터 아이템의 값을 갱신하기 위해서 디버거에 의해서 트리거될 수 있다.

[0015] 액세스되는 메타데이터 아이템이 디버거에 의해서 식별될 수 있는 다수의 방식이 있다. 일례의 배열에 있어서, 적어도 하나의 스토리지 엘리먼트는 액세스되는 메모리 어드레스의 인디케이션을 저장하는 메모리 어드레스 스토리지 엘리먼트를 포함하는 디버그 액세스 포트 회로가 배열된다. 메모리 어드레스 스토리지 엘리먼트에 발생된 커맨드를 통해서, 메모리 어드레스는, 그러므로, 메모리 어드레스 스토리지 엘리먼트 내에 저장될 수 있다. 디버그 액세스 포트 회로는, 그 다음, 적어도 하나의 복수의 메타데이터 아이템에 액세스하는 메모리 시스템 내의 위치를 결정하기 위해서 메모리 어드레스 스토리지 엘리먼트를 참조하도록 배열될 수 있다.

[0016] 어드레스는 다양한 형태를 취할 수 있다. 일례의 배열에 있어서, 어드레스는 특별한 데이터 그래뉼을 식별하기 위해서 사용되고, 그 어드레스에 기반해서, 메모리 내의 관련된 메타데이터 아이템의 위치가 결정될 수 있고, 따라서, 디버거가 메모리 어드레스 스토리지 엘리먼트 내에서 그 어드레스를 사용해서 메타데이터 액세스 동작을 트리거하는 커맨드를 발행할 때 액세스될 수 있다.

[0017] 디버그 액세스 포트 회로를 통한 디버거에 의해서 트리거된 각각의 메모리 액세스 동작에 응답해서 액세스된 메타데이터 아이템의 수는, 구현에 따라서 변할 수 있다. 예를 들어, 메모리 액세스 동작을 트리거하는 디버거로부터 싱글 커맨드에 응답해서 다수의 메타데이터 아이템에 액세스하는 것이 가능하게 될 수 있다. 그런데, 대안적인 접근에 있어서, 디버거가 메모리 액세스 동작을 트리거하는 커맨드를 발행할 때마다, 싱글 메타데이터 아이템이 액세스될 수 있다.

[0018] 메타데이터 스토리지 엘리먼트 내에서 사용되는 위치가 액세스되고 있는 각각의 메타데이터 아이템에 대해서 식별될 수 있는 다수의 방식이 있다. 일례의 배열에 있어서, 디버그 액세스 포트 회로는, 액세스되고 있는 메타데이터 아이템에 대해서 적용 가능한 메모리 어드레스 스토리지 엘리먼트 내에서 위치를 결정하기 위해서 메모리 어드레스 스토리지 엘리먼트를 참조하도록 더 배열된다. 예를 들어, 메모리 어드레스의 특정 수의 하위 어드레스 비트가 메타데이터 스토리지 엘리먼트 내의 위치를 결정하기 위해서 사용될 수 있다. 일련의 액세스를 통해서, 각각은 메모리 어드레스 스토리지 엘리먼트에 의해서 식별된 바와 같이 다른 어드레스와 관련되므로, 각각의 이러한 액세스에 대한 메타데이터 스토리지 엘리먼트 내에서 적용 가능한 위치를 식별하는 것이 가능하다.

[0019] 메타데이터 아이템의 수의 값을 검색하기 위해서 메모리 시스템으로부터 판독 액세스를 수행할 때, 그러면, 일례의 배열에 있어서, 메모리로부터 검색됨에 따라서 메타데이터 아이템의 획득된 값은 메타데이터 스토리지 엘리먼트 내의 결정된 위치 내에 쓰일 수 있는 한편, 메타데이터 스토리지 엘리먼트 내의 다른 위치의 콘텐츠는 변경되지 않는다. 메타데이터 액세스 동작의 반복된 실행을 통해서, 다른 어드레스와 각각 관련된, 이러한 접근에 의해서, 효율적인 방식으로 다수의 메타데이터 아이템에 대한 판독 값을 메타데이터 스토리지 엘리먼트 내에 구축하는 것이 가능하다. 디버거는, 그 다음, 이들 다수의 검색된 메타데이터 아이템 값을 획득하기 위해서, 메타데이터 스토리지 엘리먼트의 콘텐츠를 후속해서 판독할 수 있다.

[0020] 각각의 메타데이터 액세스 동작에 대해서 액세스하기 위해서 메타데이터 스토리지 엘리먼트 내에서 적합한 위치를 식별하는 메모리 어드레스 인디케이션을 사용하는 대안으로서, 메타데이터 스토리지 엘리먼트는 시프트 레지스터 구조로서 대신 배열될 수 있다. 이러한 배열에 따라서, 시프트 동작은 메모리 시스템으로부터 판독 액세스 동안 메타데이터 스토리지 엘리먼트 내에 메타데이터 아이템을 부가, 및 메모리 시스템에 대한 기입 액세스 동안 메타데이터 스토리지 엘리먼트로부터 메타데이터 아이템을 출력하기 위해서 사용될 수 있다.

[0021] 디버거가 메타데이터 아이템에 액세스하기 위해서 메모리 시스템에 대한 액세스를 트리거할 수 있는 다수의 방식이 있다. 그런데, 일례의 배열에 있어서, 복수의 스토리지 엘리먼트는, 디버거로부터 커맨드를 통해서 액세스될 때, 메모리 시스템에 대한 액세스가 디버그 액세스 포트 회로에 의해서 개시되게 하는, 하나 이상의 액세스 트리거 엘리먼트를 더 포함한다. 이들이 소정의 데이터를 저장할 수 있도록 액세스 트리거 엘리먼트가 배열될 수 있는 한편, 이들을 이러한 방식으로 편성하기 위한 요건은 없고, 대신, 이들은, 디버거로부터의 커맨드를 통해서 어드레스될 때, 메모리 시스템에 대한 요구된 액세스를 트리거하도록 단지 배열될 수 있는 것에 유의해야

한다.

[0022] 액세스가 메타데이터 아이템만 아니라 관련된 데이터 그래뉼의 적어도 부분을 수반할 때, 데이터 그래뉼(또는 그 관련 부분)의 값은, 희망하면, 액세스 트리거 엘리먼트 내에 일시적으로 저장될 수 있지만, 대안적인 배열에 있어서, 데이터 그래뉼(또는 그 관련 부분)의 값은 액세스 트리거 엘리먼트를 직접 통과할 수 있다(그러므로, 기입 동작에 대해서, 데이터 그래뉼 또는 그 부분의 값은, 버스로 직접 통과할 수 있는 한편, 판독 동작에 대해서 판독 데이터는 버스로부터 디버거로 직접 통과할 수도 있다).

[0023] 일례의 배열에 있어서, 다수의 다른 타입의 액세스 동작이 디버그 액세스 포트 회로에 의해서 지원될 수 있다. 일례에 있어서, 복수의 스토리지 엘리먼트는, 상기 하나 이상의 액세스 트리거 엘리먼트가 액세스될 때 수행되는 액세스 동작의 타입을 식별하기 위해서 디버거로부터의 커맨드를 통해서 액세스 가능한 모드 제어 스토리지 엘리먼트를 더 포함할 수 있다. 그러므로, 이러한 접근에 따라서, 액세스 트리거 엘리먼트는, 디버거로부터 커맨드를 통해서 어드레스될 때, 메모리 시스템에 대한 액세스가 발생하게 하는, 일반적인 엘리먼트가 될 수 있는데, 수행되는 액세스의 타입은, 그 다음, 모드 제어 스토리지 엘리먼트의 현재 콘텐츠에 의해서 구술된다.

[0024] 그런데, 대안적인 배열에 있어서, 이러한 모드 제어 스토리지 엘리먼트가 사용되지 않을 수 있고, 대신 다수의 다른 액세스 트리거 엘리먼트가 제공될 수 있다. 특히, 일례의 배열에 있어서, 상기 하나 이상의 액세스 트리거 엘리먼트는 복수의 액세스 트리거 엘리먼트를 포함할 수 있고, 각각의 액세스 트리거 엘리먼트는 다른 타입의 액세스 동작에 관련되며, 디버거는, 메모리 시스템에 대한 액세스 동작을 개시하는 커맨드를 발행할 때, 개시되는 액세스 동작의 타입에 대한 적합한 액세스 트리거 엘리먼트를 커맨드 내에서 식별하도록 배열된다. 그러므로, 이 실시예에 따라서, 다른 액세스 트리거 엘리먼트는 다른 타입의 액세스 동작과 관련되고, 디버거는 디버거가 수행하기 원하는 액세스 동작의 타입에 관한 것을 갖는 적합한 액세스 트리거 엘리먼트에 커맨드를 발행한다.

[0025] 모드 제어 스토리지 엘리먼트가 사용되는지, 또는 다수의 다른 액세스 트리거 엘리먼트가 사용되는지에 관계없이, 디버그 액세스 포트 회로는, 디버거로부터 수신된 커맨드에 의존해서, 지원된 액세스 동작의 세트로부터 액세스 동작을, 버스 인터페이스를 통해서, 개시하기 위해서 배열될 수 있다.

[0026] 예에 의해서, 지원된 액세스 동작의 세트는 하나 이상의 다음의 기입 동작을 포함할 수 있는데, 다음은: 메타데이터 스토리지 엘리먼트로부터 획득된 값을 사용해서 메타데이터 아이템의 값을 갱신하는, 및 디버거로부터 커맨드에 의해서 제공된 데이터 값을 사용해서 관련된 데이터 그래뉼의 적어도 부분의 현재 값을 갱신하는 기입 동작; 메타데이터 스토리지 엘리먼트로부터 획득된 값을 사용해서 메타데이터 아이템의 값을 갱신하는 한편, 관련된 데이터 그래뉼의 현재 값을 보존하는 기입 동작; 메타데이터 스토리지 엘리먼트로부터 획득된 값을 사용해서 메타데이터 아이템의 값을 갱신하는, 및 관련된 데이터 그래뉼의 적어도 부분의 값을 공지된 값으로 갱신하는 기입 동작; 데이터 그래뉼의 적어도 부분을 메모리 시스템에 기입하여, 관련된 메타데이터 아이템의 현재 값을 보존하는, 기입 동작; 데이터 그래뉼의 적어도 부분을 메모리 시스템에 기입하여, 관련된 메타데이터 아이템의 값을 디폴트 값으로 갱신하는, 기입 동작이다.

[0027] 그러므로, 넓은 다양한 다른 타입의 기입 동작은 디버그 액세스 포트 회로를 통해서 디버거에 의해서 트리거될 수 있다.

[0028] 또 다른 예로서, 지원된 액세스 동작의 세트는 하나 이상의 다음의 판독 동작을 포함할 수 있는데, 다음은: 메모리 시스템으로부터 메타데이터 아이템을 판독하는 및 메타데이터 스토리지 엘리먼트 내에 그 판독 메타데이터 아이템의 값을 저장하는, 및 메모리 시스템으로부터 관련된 데이터 그래뉼의 적어도 부분을 추가적으로 판독하는 판독 동작; 관련된 데이터 그래뉼을 추가적으로 판독하지 않고, 메모리 시스템으로부터 메타데이터 아이템을 판독하는 및 메타데이터 스토리지 엘리먼트 내에 그 판독 메타데이터 아이템의 값을 저장하는 판독 동작; 관련된 메타데이터 아이템을 판독하지 않고, 메모리 시스템으로부터 데이터 그래뉼의 적어도 부분을 판독하는 판독 동작이다. 그러므로, 디버그 액세스 포트 회로의 사용은, 또한, 수행될 수 있는 판독 동작의 타입에 대해서와 같이 디버거에 큰 유연성을 제공하는 것을 알 수 있다.

[0029] 각각의 메타데이터 아이템은 메타데이터의 형태에 의존하는 하나 이상의 비트를 포함할 수 있다. 특정 예로서, 각각의 메타데이터 아이템은 관련된 데이터 그래뉼이 능력을 특정하는지를 식별하는 능력 태그가 될 수 있다. 소정의 능력이 주어진 프로세스에 대해서 규정된 능력-기반 아키텍처에 관심이 증가하고 있고, 및 에러는, 규정된 능력 밖에서 동작을 수행하는 시도가 있으면, 에러가 트리거될 수 있다. 바운드된 포인터는 일례의 능력이다. 포인터 자체가, 예를 들어, 액세스되는 데이터 값 또는 실행되는 명령의 어드레스를 포인팅할 수 있

거나, 또는 이를 결정하기 위해서 사용될 수 있다. 그런데, 포인터는, 또한, 포인터를 사용할 때 어드레스의 허용 가능한 범위를 표시하는 범위 정보와 관련할 수 있다. 이는, 예를 들어, 포인터로부터 결정된 어드레스가, 소정의 바운드 내에서, 행동의 보안 또는 기능적인 정확성을 유지하도록 보장하기 위해서, 유용하게 될 수 있다. 이러한 능력-기반 시스템 내에서, 소정의 특별한 데이터 그래뉼이 능력에 또는 일반적인 목적의 데이터에 관련되는지를 아는 것은 중요하고, 능력 태그가 이 목적을 위해서 사용될 수 있다.

[0030] 메타데이터 아이템의 다른 예로서, 메타데이터 아이템은 관련된 데이터 그래뉼의 할당 정책을 식별하는 할당 태그가 될 수 있다. 이러한 할당 태그는, 또한, 가드 태그로서 언급될 수 있다. 소정의 메모리 사용 에러에 대항해서 보호하는 하나의 접근은, 하나 이상의 메모리 위치(이들 하나 이상의 메모리 위치 내의 데이터가 데이터 그래뉼이다)의 블록과 관련해서 메모리 시스템 내에 저장되는 가드 태그를 제공할 수 있다. 태그-가드된 메모리 액세스 동작이 메모리 시스템 내의 특별한 어드레스된 위치를 식별하는 타깃 어드레스에 기반해서 요청될 때, 메모리 액세스 회로는 타깃 어드레스에 의해서 식별된 어드레스된 위치를 포함하는 하나 이상의 메모리 위치의 블록과 관련해서 메모리 시스템 내에 저장되는 가드 태그와 관련된 어드레스 태그를 비교할 수 있다. 메모리 액세스 회로는, 매치가 가드 태그와 어드레스 태그 사이에서 검출되는지의 인디케이션을 생성할 수 있다. 이 인디케이션은, 메모리 액세스가 성공하도록 허용되는지 또는 후속 동작이 성공할 수 있는지를 제어하기 위해서 사용될 수 있거나, 또는 단지 보고될 수 있는 한편 메모리 액세스는 정상적으로 계속되도록 허용한다.

[0031] 디버거가 메모리 시스템에 액세스하기 위한 커맨드를 발행하도록 디버그 액세스 포트 회로에 의해서 제공된 디버그 인터페이스는 다양한 형태를 취할 수 있다. 그런데, 메모리 시스템 내에서 수행되는 액세스들이 디버그 액세스 포트 회로에 의해서 제공된 상대적으로 작은 세트의 스토리지 엘리먼트에 커맨드를 발행함으로써 착수되는 사실에 기인해서, 디버그 인터페이스는 상대적으로 콤팩트한 형태를 취할 수 있다. 순전히 예로서, 디버그 인터페이스는 JTAG, 직렬 와이어, 또는 PCI 인터페이스 중 하나가 될 수 있다.

[0032] 일례의 배열에 있어서, 복수의 스토리지 엘리먼트는 어드레스 스페이스 내에 존재하고, 디버그 인터페이스에서 수신된 커맨드는 어드레스 스페이스 내의 어드레스를 특정함으로써 그 커맨드가 관련되는 스토리지 엘리먼트를 식별한다. 이는, 다양한 스토리지 엘리먼트를 식별하기 위한 효율적인 메커니즘을 제공한다.

[0033] 복수의 스토리지 엘리먼트는 다양한 형태를 취할 수 있지만, 일례의 배열에 있어서, 복수의 레지스터를 포함한다.

[0034] 일례의 구현에 있어서, 디버그 액세스 포트 회로는, 액세스 동작의 다수의 반복(iteration)을, 버스 인터페이스를 통해서, 개시하도록 디버거로부터의 커맨드에 응답하고, 각각의 반복에 대해서, 메모리 어드레스 스토리지 엘리먼트 내에 저장된 메모리 어드레스 인디케이션은 증분된다.

[0035] 메모리 어드레스 인디케이션이 증분되는 양은, 일례에 있어서, 수행되는 액세스 동작의 타입에 따라서 변할 수 있다. 예들 들어, 일례의 구현에 있어서, 액세스 동작이 메타데이터 아이템 및 액세스되는 데이터 모두를 요구할 때, 메모리 어드레스 인디케이션은, 다음 반복에 대한 어드레스 인디케이션을 제공하기 위해서, 현재 반복에서 액세스된 데이터의 양에 기반해서 증분된다. 그런데, 액세스 동작이 액세스되는 메타데이터 아이템만을 요구할 때, 메모리 어드레스 인디케이션은 데이터 그래뉼 사이즈를 n배함으로써 증분될 수 있는데, 여기서 n은, 다음 반복에 대한 어드레스 인디케이션을 제공하기 위해서, 현재 반복에서 액세스된 메타데이터 아이템의 수와 동등하다.

[0036] 특별한 예들이 이제 도면을 참조해서 기술될 것이다.

[0037] 도 1은 일례의 구성에 따른 시스템의 블록도이다. 디버거(10)는 디바이스에 대해서 디버그 동작을 수행하기 위해서 사용된다. 디바이스는 다양한 형태를 취할 수 있지만, 도시하기 위해서 도 1 내의 디바이스는 인터커넥트(30)를 통해서 메모리(35) 및 일부 주변 디바이스(40)와 접속하는 하나 이상의 중앙 처리 유닛(CPU)(15)으로 이루어진다. 디버거는, 본 개시에서 디버그 액세스 포트 회로로서도 언급되는 메모리 액세스 포트와 같은 관련된 메모리 액세스 포트(20, 25)를 통해서 디바이스에 대한 다수의 버스 접속을 수립할 수 있다. 도 1에 나타낸 바와 같이, 제1메모리 액세스 포트(20)가 제공되는데, 이를 통해서, 디버거는 하나 이상의 CPU(15)가 소정의 동작을 수행하게 하게 하는 커맨드를 발행할 수 있다. 메모리 액세스 포트(20)는 버스(22)를 통해서 CPU에 결합될 수 있는데, 이를 통해서, 메모리 액세스 포트는 디버거로부터 수신된 커맨드에 의해서 표시된 바와 같은 소정의 동작을 수행하도록 CPU에 명령할 수 있고, 및 이를 통해서, 데이터는 메모리 액세스 포트와 CPU 사이에서 양쪽 방향으로 흐를 수 있다.

[0038] 본 개시에 기술된 기술에 따라서, 추가적인 메모리 액세스 포트(25)가 제공되는데, 이는, 메타데이터 인식이고,

특히, 이를 통해서, 데이터 그래뉼 및 관련된 메타데이터 아이템 모두가 전송될 수 있는 버스(27)에 결합될 수 있다. 메타데이터 인식 메모리 액세스 포트(25)는 메커니즘을 제공하는데, 이를 통해서, 디버거(10)는 메모리(35) 내에 저장된 메타데이터 아이템에 직접 액세스될 수 있고, 따라서, CPU(15) 중 하나 내에서 메타데이터 액세스 동작을 개시할 필요 없이, 메타데이터 아이템의 판독, 및 메타데이터 아이템에 대한 기입을 지원한다.

[0039] 도 2는, 일례의 배열에 따른, 메타데이터 인식 메모리 액세스 포트(25) 내에 제공된 컴포넌트를 더 상세히 도시하는 블록도이다. 메모리 액세스 포트(25)는 디버거(10)에 결합하기 위한 디버그 인터페이스(50) 및 버스(27)에 결합하기 위한 버스 인터페이스(75)를 갖는다. 메모리 액세스 포트 내에, 디버거(10)로부터 발행된 커맨드를 통해서 액세스 가능한 복수의 스토리지 엘리먼트가 제공된다. 특히, 도 2에 나타낸 예에 있어서, 다양한 스토리지 엘리먼트는 다른 레지스터의 형태를 취한다. 작은 어드레스 스페이스는 메모리 액세스 포트와 관련되고, 디버거에 의해서 발행된 커맨드는 그 레지스터와 관련된 어드레스를 식별함으로써 레지스터 중 특별한 하나를 타깃으로 할 수 있다. 각각의 커맨드는, 그러므로, 액세스 포트(AP) 레지스터 어드레스, 및 일부 첨부 "데이터"를 특정할 수 있다. 데이터는 커맨드에 의해서 액세스되고 있는 레지스터에 의존해서 다양한 형태를 취할 수 있다.

[0040] 그러므로, 디버거가, 예를 들어, 관련된 데이터 그래뉼 및/또는 데이터 그래뉼과 관련된 메타데이터 아이템에 대해서 액세스를 수행하게 메모리 내의 어드레스에 액세스하는 것을 원할 때, 이는, 버스 어드레스 레지스터(55)를 식별하는 커맨드를 발행할 것인데, 그 커맨드가 제공된 데이터는 액세스되는 메모리 내의 어드레스의 인디케이션이다. 이러한 커맨드가 버스 어드레스 레지스터(55)에 발행될 때, 관심의 메모리 어드레스를 식별하기 위해서, 버스 어드레스 레지스터의 콘텐츠가 커맨드에 의해서 제공된 데이터로 갱신된다.

[0041] 데이터 레지스터(60)가, 또한, 제공될 수 있고, 커맨드가 디버거로부터 데이터 레지스터에 발행될 때, 이는, 버스 어드레스 레지스터(55) 내에 제공된 메모리 어드레스를 사용해서, 메모리에 대한 액세스 동작을 개시하기 위해서 메모리 액세스 포트를 트리거한다. 기입 동작을 개시하기 위해서 사용된 커맨드에 대해서, 관련된 기입 데이터는, 예를 들어, 액세스되는 데이터 그래뉼의 적어도 부분에 대한 갱신된 값을 식별하기 위해서 특정될 수 있고, 그 데이터는, 희망하면, 데이터 레지스터(60) 내에서 일시적으로 버퍼될 수 있다. 대안적으로, 데이터는 데이터 레지스터 자체 내에 저장될 필요가 없을 수 있고, 대신, 데이터는 버스로 직접 통과될 수 있으며, 데이터 레지스터(60)에 대한 액세스는 단지 기입 동작의 실행을 트리거하는 메커니즘으로서 사용된다.

[0042] 유사하게, 판독 동작에 대해서, 타깃된 데이터 그래뉼의 적어도 부분에 속하는 판독 데이터가 메모리 액세스 포트(25)로 복귀할 때, 이는, 데이터 레지스터(60) 내에서 일시적으로 버퍼될 수 있거나, 또는 대안적으로 디버거(10)로의 복귀를 위한 액세스 포트로 직접 통과될 수 있는데, 이 경우, 다시, 데이터 레지스터(60) 자체는 데이터를 저장하기 위해서 사용되지 않지만, 이것에 대한 액세스는 단지 판독 동작을 트리거하기 위해서 사용된다.

[0043] 초기에 논의된 바와 같이, 메모리 내의 데이터 그래뉼에 대한 판독 및 기입 동작에 추가해서, 디버거는 시간 내의 소정의 포인트에서 이들 데이터 그래뉼과 관련된 메타데이터 아이템에 액세스하는 것을 원할 수 있다. 이러한 액세스는, 관련된 데이터 그래뉼이 액세스되는 동일한 시간에 수행될 수 있거나, 또는 대안적으로, 이는, 관련된 데이터 그래뉼을 액세스하지 않고, 메타데이터 아이템에 액세스하는 것을 희망할 수 있다. 이러한 메타데이터 액세스 동작을 용이하게 하기 위해서, 복수의 메타데이터 아이템을 저장할 수 있는 메타데이터 레지스터(70)가 제공된다. 메타데이터 레지스터 내에 저장될 수 있는 메타데이터 아이템의 수는 메타데이터 레지스터의 사이즈, 및 각각의 메타데이터 아이템을 형성하는 비트의 수에 의존할 것이다. 많은 경우에 있어서, 메타데이터 아이템은 관련된 데이터 그래뉼보다 상당히 작게 될 수 있고, 실제로 각각의 메타데이터 아이템은 정보의 하나 또는 약간의 비트만을 포함할 수 있다. 이러한 인스턴스에 있어서, 이는, 메타데이터 레지스터 내에 저장되는 상당한 수의 메타데이터 아이템에 대해서 가능하고, 그러므로, 메타데이터 인식 메모리 액세스 포트(25)를 통한 디버거(10)로부터의 적합한 커맨드의 발행을 통해서 발생하도록 전체 시리즈의 메타데이터 아이템에 대한 벌크 액세스를 허용한다.

[0044] 그러므로, 일련의 메타데이터 아이템에 대한 기입 동작을 수행하는 것이 희망 될 때, 이들 메타데이터 아이템을 위한 새로운 값이 메타데이터 레지스터(70)에 커맨드를 발행하는 디버거를 통해서 메타데이터 레지스터(70) 내에 저장될 수 있는데, 여기서 그 커맨드에 대한 관련된 "데이터"는 일련의 메타데이터 아이템에 대해서 갱신된 값을 제공한다. 메타데이터 레지스터(70)가 액세스될 때, 메타데이터 아이템에 대한 이들 새로운 값은, 그러면, 메타데이터 레지스터 내의 대응하는 위치 내에 저장될 것이다.

[0045] 일련의 기입 동작은, 그 다음, 데이터 레지스터(60)에 일련의 커맨드를 발행함으로써 트리거될 수 있고, 각각의

기입 동작을 위한 어드레스를 식별하기 위해서 적합한 것으로 갱신된 버스 어드레스 레지스터(55)의 콘텐츠와 함께 트리거될 수 있다. 일부 인스턴스에 있어서, 디버거는 각각의 기입 커맨드가 데이터 레지스터(60)에 발행되기 전에 버스 어드레스 레지스터(55)를 갱신하는 분리의 커맨드를 발행할 수 있는 한편, 대안적인 접근에 있어서, 메모리 액세스 포트는 어드레스 증분 회로를 포함할 수 있어서, 버스 어드레스 레지스터(55) 내의 어드레스가 각각의 기입 커맨드를 뒤따르는 소정 량에 의해서 증분될 수 있도록 되므로, 이에 의해서, 디버거가 각각의 기입 동작의 실행 사이에서 버스 어드레스 레지스터 콘텐츠(55)를 갱신하는 커맨드를 분리해서 발행할 필요는 회피된다.

[0046] 메타데이터 인식 메모리 액세스 포트(25)는, 또한, 디버거로부터 적합한 커맨드를 통해서 액세스될 수 있는, 및 커맨드가 데이터 레지스터(60)에 후속해서 발행될 때 그 콘텐츠가 수행되는 액세스 동작의 타입을 식별하기 위해서 사용되는, 버스 제어 레지스터(65)를 포함한다. 그러므로, 버스 제어 레지스터의 콘텐츠는 기입 액세스 동작이 수행되는지를, 또는 판독 액세스 동작이 수행되는지를 결정할 수 있고, 또한, 수행되는 기입 또는 판독 동작의 타입을, 예를 들어, 데이터 그래뉼이 액세스되는지를, 관련된 메타데이터 아이템이 액세스되는지를, 또는 데이터 그래뉼 및 관련된 메타데이터 아이템 모두가 액세스되는지를 식별할 수 있다.

[0047] 버스 어드레스 레지스터(55) 내의 어드레스는, 하나의 실시예에 있어서, 특별한 데이터 그래뉼을 식별하지만, 그 어드레스는, 또한, 액세스되는 관련된 메타데이터 아이템을 식별하기 위해서 사용될 수 있다. 그러므로, 데이터 레지스터(60)가 메타데이터 기입 동작을 개시하기 위해서 디버거로부터 커맨드에 의해서 접촉될 때, 이는, 버스 어드레스 레지스터(55)의 콘텐츠로부터 메타데이터 아이템의 위치를 결정할 수 있고, 및 메타데이터 레지스터(70)로부터 메타데이터 아이템에 대한 갱신된 값을 획득할 수 있다. 메타데이터 기입 동작은, 그 다음, 요구된 메타데이터 아이템이 갱신되게 하도록 메모리 액세스 포트(25)로부터 개시될 수 있다.

[0048] 유사하게, 메타데이터 판독 동작에 대해서, 버스 어드레스 레지스터(55) 내의 어드레스 정보는, 관련 메타데이터 아이템을 식별하기 위해서 다시 사용될 수 있고, 그 메타데이터 아이템의 값이 복귀될 때, 메타데이터 레지스터 내의 적합한 위치 내에 저장하기 위해서 메타데이터 레지스터(70)로 라우트될 수 있다.

[0049] 메타데이터 레지스터(70) 내에서 사용되는 적합한 위치가 수행된 각각의 메타데이터 아이템에 대해서 식별될 수 있는 다수의 방법이 있다. 예를 들어, 버스 어드레스 레지스터(55) 내에 저장된 어드레스의 소정 수의 하위 비트는, 메타데이터 기입 동작을 수행할 때 검색되어야 하는 어떤 값으로부터, 또는 메타데이터 판독 동작을 수행할 때 저장되어야 하는 어떤 검색된 값으로 메타데이터 레지스터 내의 대응하는 위치를 식별하기 위해서 사용될 수 있다. 대안적으로, 메타데이터 레지스터는 시프트 레지스터로서 배열될 수 있어서, 메타데이터 기입 동작을 수행할 때 메타데이터 아이템의 갱신된 값을 표현하는 소정 수의 비트가 시프트 아웃될 수 있거나, 또는 반대로 메타데이터 판독 동작을 수행할 때 판독 메타데이터 아이템을 표현하는 비트가 메타데이터 레지스터(70)로 시프트될 수 있도록 한다.

[0050] 도 3a 및 3b는, 일례의 배열에 따른, 메타데이터 아이템의 벌크 판독을 수행하기 위해서, 어떻게 메타데이터 인식 메모리 액세스 포트(25)가 사용될 수 있는지를 식별하는 흐름도를 제공한다. 단계 100에서, 디버거는 버스 어드레스에 대한 개시 어드레스를 결정하고, 그 다음, 단계 105에서, 그 버스 어드레스를 데이터로서 특정하는 커맨드를 버스 어드레스 레지스터(55)에 발행한다. 이는, 버스 어드레스 레지스터(55)가 개시 어드레스를 식별하기 위해서 자체의 콘텐츠를 갱신하게 한다.

[0051] 단계 110에서, 디버거는, 그 다음, 수행되는 판독 동작의 타입을 데이터로서 특정하는 커맨드를 버스 제어 레지스터(65)에 발행한다. 판독 동작은 다양한 형태를 취할 수 있고, 예를 들어, 메타데이터 아이템만이 판독되는 것을 특정할 수 있거나, 또는 대안적으로 메타데이터 아이템 및 관련된 데이터 그래뉼(또는 그 데이터 그래뉼의 적어도 부분) 모두가 판독되는 것을 식별할 수 있다.

[0052] 디버거는, 그 다음, 버스 제어 레지스터(65) 내에 규정된 타입의 판독 동작의 실행을 트리거하기 위해서 데이터 레지스터(60)에 커맨드를 발행한다. 더욱이, 버스 어드레스 레지스터(55) 내의 버스 어드레스는 액세스되고 있는 메모리 어드레스를 결정하기 위해서 사용된다. 초기에 논의된 바와 같이, 버스 어드레스 레지스터 내의 어드레스는 데이터 그래뉼 위치를 식별할 수 있지만, 또한, 관련된 메타데이터 아이템의 메모리 시스템 내의 위치를 결정하기 위해서 사용될 수 있다. 특히, 8a 및 8b를 참조해서 나중에 더 상세히 논의된 바와 같이, 일부 예의 배열에 있어서, 메타데이터 아이템 자체는 세미-히든(semi-hidden)될 수 있는데, 이들은, CPU에 의해서 발행될 수 있는 어드레스에 의해서 직접 어드레스 가능하게 되지 않을 수 있다. 그런데, 적합한 데이터 그래뉼을 특정함으로써, 대응하는 메타데이터 아이템이 식별 및 액세스될 수 있다.

[0053] 단계 120에서, 판독 메타데이터 아이템이 수신될 때, 그 메타데이터 아이템은, 그 다음, 메타데이터 레지스터(70)의 위치(엔트리로서도 본 개시에서 언급되는 그 위치) 내에 저장된다. 메타데이터 레지스터(70)가 시프트 레지스터로서 배열될 때, 이는, 메타데이터 레지스터 내로 판독 메타데이터 값을 시프팅함으로써 발생할 수 있다. 대안적으로, 메타데이터 레지스터가 시프트 레지스터로서 배열되지 않으면, 버스 어드레스 레지스터(55) 내의 버스 어드레스의 소정 수의 하위 비트는, 판독 메타데이터 아이템이 저장되어야 하는 메타데이터 레지스터 내의 적합한 위치를 식별하기 위해서 사용될 수 있다. 이 방식으로 메타데이터 레지스터를 갱신할 때, 갱신 프로세스가 전형적으로 배열될 것이므로, 다른 위치 중 아무 데도 영향받지 않고, 그들의 현재 콘텐츠를 유지한다.

[0054] 단계 120에서 나타낸 바와 같이, 판독 동작이, 또한, 데이터 그래뉼의 모든 또는 부분을 복귀시키면, 그 판독 데이터는, 희망하면, 디버그 인터페이스(50)를 통해서 디버거로 복귀하기 전에, 데이터 레지스터(60) 내에서 일시적으로 버퍼될 수 있다. 대안적으로, 이는, 데이터 레지스터(60) 내에서 버퍼되지 않고, 디버그 인터페이스(50)를 통해서 디버거로 직접 복귀될 수 있다.

[0055] 도 3a에서 분리의 커맨드가 단계 105, 110 및 115에서 발행되는 것으로 상정되는 한편, 대안적인 구현에 있어서, 단계 105 내지 120의 일부 또는 모두는 디버거에 의해서 발행된 싱글 커맨드에 응답해서 수행될 수 있는데, 예를 들어, 여기서 어드레스, 제어 및 액세스 커맨드는 싱글 패킷을 형성한다.

[0056] 단계 120에 뒤따라서, 프로세스는, 그 다음, 단계 125로 진행하는데, 여기서, 판독되는 더 많은 메타데이터 아이템이 있는지가 결정된다. 디버거는 판독된 메타데이터 아이템의 수를 제어할 수 있지만, 일례의 배열에 있어서, 이는, 메타데이터 레지스터(70)에 풀로(fully) 거주하는 메타데이터 아이템의 충분한 수를 판독하는 것을 원할 수 있는데, 풀로(fully) 거주된 메타데이터 레지스터(70)는, 그 다음, 모든 판독 메타데이터 아이템의 값을 검색하기 위해서 인터러게이트(interrogated)된다.

[0057] 단계 125에서, 판독할 더 많은 메타데이터 아이템이 있는 것으로 결정되면, 일례의 배열에 있어서, 디버거는, 단계 130에서, 갱신된 어드레스를 결정할 수 있다. 그런데, 대안적인 배열에 있어서, 메모리 액세스 포트(25)는, 판독 동작의 실행이 뒤따르는 버스 어드레스 레지스터(55)의 콘텐츠를 자동으로 갱신할 수 있는 일부 어드레스 증분 로직과 통합할 수 있으므로, 단계 130을 중복적이다. 단계 130이 수행되면, 그 다음, 프로세스는 단계 105로 복귀할 것인데, 여기서 디버거에 의해서 결정된 갱신된 어드레스가 버스 어드레스 레지스터(55) 내에 기입된다. 이 다음 반복 단계 110은 필요하지 않을 수 있다. 그런데, 버스 어드레스 레지스터 콘텐츠가 메모리 액세스 포트에 의해서 자동으로 갱신되면, 단계 125로부터 예스(yes) 경로는 도 3a의 단계 115로 직접 복귀할 수 있고, 동일한 타입의 판독 커맨드가 다음 반복 상에서 수행되고, 그러므로 단계 110은 재수행될 필요가 없다.

[0058] 버스 어드레스 레지스터(55) 내의 어떤 어드레스에 의한 양은, 현재 판독 동작에 의해서 액세스된 메타데이터 아이템의 수에 의존할 판독할 더 많은 메타데이터 아이템이 있는 것이, 단계 125에서, 결정될 때, 증분될 필요가 있다. 싱글 메타데이터 아이템이 액세스되면, 어드레스는, 예를 들어, 이것이 데이터 그래뉼 사이즈에 의해서 증분되도록 갱신될 수 있다. 그런데, 다수의 메타데이터 아이템이 액세스되면, 그 다음, 어드레스는 데이터 그래뉼 사이즈의 배수에 의해서 증분될 수 있다.

[0059] 더욱이, 데이터가 메타데이터 아이템에 추가해서 액세스되면, 싱글 데이터 액세스는 그래뉼 사이즈 미만 액세스할 수 있다. 결과적으로, 어드레스는, 그 다음, 그래뉼 사이즈 미만 증분될 수 있고, 특히, 액세스된 데이터의 사이즈만 증분될 수 있다. 어드레스는, 그러므로, 다음 반복 동안 동일한 그래뉼 내에 머물 수 있고, 후속 액세스는, 그 다음, 동일한 메타데이터 아이템에 액세스할 것이다.

[0060] 단계 125에서, 판독할 더 많은 메타데이터 아이템이 없는 것으로 결정되면, 프로세스는 단계 135로 진행할 수 있고, 여기서 디버거는 그것 내에 저장된 복수의 메타데이터 아이템을 판독하기 위해서 메타데이터 레지스터(70)에 커맨드를 발행할 수 있다.

[0061] 도 4a 및 4b는, 디버거가 도 2의 메타데이터 인식 메모리 액세스 포트를 사용하는 메타데이터 아이템의 벌크 기입을 수행할 수 있도록 수행될 수 있는 단계를 도시하는 흐름도를 제공한다. 단계 200에서, 디버거는, 복수의 메타데이터 아이템에 대한 데이터 값으로서 특정하는 커맨드를, 메타데이터 레지스터(70)에 발행해서, 이들 메타데이터 아이템 값이 메타데이터 레지스터 내에 저장되게 한다. 그 다음, 단계 205, 210, 215 및 220은 도 3a의 단계 100, 105, 110 및 115에 대응한다. 그런데, 단계 215에서, 버스 제어 레지스터에 발행된 커맨드로 제공된 데이터는 수행되는 기입 동작의 타입, 예를 들어, 기입 동작이 메타데이터 아이템의 기입을 수행하는 한편,

관련된 데이터 그래뉼의 값을 보존하는지, 또는, 대신, 메타데이터 아이템의 기입 및, 또한, 데이터 그래뉼의 적어도 부분을 갱신하는 기입을 수행하는지를 식별할 것이다.

[0062] 도 4a에서 분리의 커맨드가 단계 210, 215 및 220에서 발행되는 것으로 상정되는 한편, 대안적인 구현에 있어서, 단계 210 내지 220의 일부 또는 모두는 디버거에 의해서 발행된 싱글 커맨드에 응답해서 수행될 수 있는데, 예를 들어, 여기서 어드레스, 제어 및 액세스 커맨드는 싱글 패킷을 형성한다.

[0063] 커맨드가 기입 동작의 실행을 트리거하기 위해서 단계 220에서 데이터 레지스터(60)에 발행되었으면, 단계 225에서 메타데이터 레지스터의 엔트리로부터 추출된 메타데이터 아이템을 사용해서 기입 동작이 수행된다. 초기에 판독 동작과 마찬가지로, 버스 어드레스 레지스터(55) 내에 저장된 어드레스의 소정 수의 하위 비트는, 일례의 배열에 있어서, 메타데이터 레지스터 내에서 적합한 엔트리를 식별하기 위해서 사용될 수 있다. 그런데, 메타데이터 레지스터가 시프트 레지스터로서 배열된 또 다른 예의 배열에 있어서, 관련 메타데이터 아이템 값은 시프트 레지스터로부터 소정 수의 비트를 시프팅 아웃함으로써 획득될 수 있고, 시프트 레지스터로부터 시프트 아웃된 비트의 수는 각각의 메타데이터 아이템을 형성하는 비트의 수에 의존한다.

[0064] 더욱이, 데이터 그래뉼의 적어도 부분이, 또한, 기입 동작의 부분으로서 갱신되면, 그 데이터의 값은 기입 동작을 트리거하기 위해서 데이터 레지스터(60)에 송신된 커맨드 내에 특정될 수 있고, 그 기입 데이터는, 희망하면, 데이터 레지스터(60) 내에 일시적으로 버퍼된다. 대안적으로, 그 기입 데이터는 데이터 레지스터(60) 내에서 버퍼되지 않고 버스로 직접 통과될 수 있다.

[0065] 단계 230에서, 기입할 더 많은 메타데이터 아이템이 있는지가 결정된다. 디버거는, 전형적으로, 이것이 기입 동작을 수행하는 것을 원하지만, 일례의 배열에 있어서, 메타데이터 레지스터(70) 내에 저장될 수 있는 분리의 메타데이터 아이템의 수와 동등한 메타데이터 아이템의 수를 갱신하는 것을 원할 수 있는 메타데이터 아이템의 수를 결정할 것이고, 따라서, 메타데이터 레지스터(70) 내에서 아직 사용되지 않은 메타데이터 아이템 값으로서 적어도 하나가 있으면, 기입할 더 많은 메타데이터 아이템이 있는 것으로 결정될 것이다.

[0066] 기입할 더 많은 메타데이터 아이템이 있는 것으로 결정되면, 단계 235에서, 디버거는, 프로세스가 단계 210으로 복귀하기 전에, 갱신된 어드레스를 결정할 수 있다. 이 다음 반복 단계 215는 필요하지 않을 수 있다. 그런데, 대안적인 배열에 있어서, 메모리 액세스 포트(25)는, 기입 동작의 실행에 뒤따라서 버스 어드레스 레지스터(55) 내의 어드레스를 자동으로 증분하는 증분 회로를 포함할 수 있으므로, 프로세스가 도 4a의 단계 220으로 직접 복귀할 수 있는 단계 235에 대한 필요가 회피되며, 동일한 타입의 기입 커맨드가 다음 반복에서 수행되고 그러므로 단계 215는 재수행될 필요가 없는 것으로 상정한다.

[0067] 도 5는 메타데이터 인식 메모리 액세스 포트에 대한 대안적인 배열을 도시하는데, 여기서 버스 제어 레지스터(65)에 대한 필요가 회피될 수 있다. 도 5에 나타낸 바와 같이, 메타데이터 인식 메모리 액세스 포트(25')는 도 2에 따른 버스 어드레스 레지스터(55) 및 메타데이터 레지스터(70)를 포함하고, 또한, 디버그 인터페이스(50) 및 버스 인터페이스(75)를 포함한다. 그런데, 복수의 데이터 레지스터(250)가 제공되고, 하나의 데이터 레지스터는 지원되는 각각의 타입의 액세스 동작과 관련된다. 이러한 접근에 따라서, 버스 제어 레지스터(65)는 필요하지 않게 되고, 대신, 디버거는 요구되는 메모리 액세스와 관련된 특별한 데이터 레지스터에 커맨드를 발행한다. 그러므로, 다른 타입의 기입 커맨드와 관련된 데이터 레지스터의 수 및 다른 타입의 판독 커맨드와 관련된 데이터 레지스터의 수가 있을 수 있다. 이들 기입 및 판독 커맨드의 적어도 일부는, 메타데이터 아이템에 대해서 기입 동작 또는 판독 동작을 수행하는 것을 포함할 것이고, 메타데이터 아이템은 도 2를 참조해서 초기에 논의된 바와 같은 동일한 방식으로 메타데이터 레지스터(70) 내에 액세스될 수 있다.

[0068] 도 6a 및 6b는 도 3a 및 3b의 것과 유사한 흐름도를 제공하지만, 도 5에 나타낸 바와 같은 메타데이터 인식 메모리 액세스 포트(25')의 형태를 사용할 때만이다. 단계 300 및 305는 도 3a의 단계 100 및 105에 대응한다. 그 다음, 단계 310에서, 디버거는, 버스 어드레스 레지스터(55) 내에 저장된 버스 어드레스를 사용해서 그 요구된 판독 동작의 실행을 트리거하기 위해서, 요구된 판독 동작의 타입에 대한 적합한 데이터 레지스터에 커맨드를 발행한다. 그러므로, 싱글 커맨드는, 하나의 커맨드가 버스 제어 레지스터에 발행된 다음, 또 다른 커맨드가 데이터 레지스터에 발행되는 것보다, 이 단계에서 발행될 필요가 있는 것을 알 수 있다.

[0069] 단계 315, 320, 325 및 330은 도 3a 및 3b의 단계 120, 125, 130 및 135에 대응한다. 각각의 반복 사이에서의 어드레스의 갱신이 메모리 액세스 포트(25') 내에서 자동으로 발생하는 경우, 단계 320으로부터의 예스 경로는 단계 305보다 단계 310으로 복귀할 수 있다.

[0070] 도 7a 및 7b는 도 4a 및 4b의 흐름도에 대응하는 흐름도를 제공하지만, 도 5의 메타데이터 인식 메모리 액세스

포트(25')를 활용할 때, 어떻게 메타데이터 아이템의 벌크 기입이 수행될 수 있는지를 기술한다. 단계 400, 405 및 410은 도 4a의 단계 200, 205 및 210에 대응한다. 그 다음, 단계 415에서, 디버거는, 버스 어드레스 레지스터 내의 버스 어드레스를 사용해서 그 요구된 판독 동작의 실행을 트리거하기 위해서, 요구된 판독 동작의 타입에 대한 적합한 데이터 레지스터에 커맨드를 발행한다. 그러므로, 버스 제어 레지스터에 분리의 커맨드를 발행한 다음 후속해서 데이터 레지스터에 발행할 필요는 회피한다.

[0071] 그 다음, 단계 420, 425, 430 및 435는 도 4b의 단계 225, 230, 235 및 240에 대응한다. 어드레스가 각각의 반복 사이에서 메모리 액세스 포트(25') 내에서 자동으로 조정되면, 단계 430에 대한 필요는 회피되고, 그러면 단계 425로부터의 예스 경로는 단계 415로 직접 복귀할 수 있다.

[0072] 도 8a는 메모리 어드레스 스페이스 및 그 어드레스 스페이스 내의 다양한 데이터 그래뉼(450) 및 관련된 메타데이터 아이템(455)의 존재를 도시한다. 그런데, 일례의 배열에 있어서, 메타데이터 아이템 자체는 직접 어드레스 가능하지 않고, 따라서 세미-히든으로 볼 수 있다. 대신, 도 8a 내의 하부 도시에 도시된 바와 같이, 어드레스 가능한 어드레스 스페이스는 데이터 그래뉼의 면에서 특정된다. 이 예에 있어서, 0의 바이트 어드레스는 제1데이터 그래뉼이 액세스되게 할 것이고, 16의 바이트 어드레스는 제2데이터 그래뉼이 액세스되게 하는 등으로 될 것이다. 그런데, 메타데이터 아이템(455)은 각각의 데이터 그래뉼과 관련된다. 그 다음, 메타데이터 액세스 동작이 사용될 때, 어드레스는 여전히 데이터 그래뉼을 식별하는 한편, 그 어드레스는 관심의 대응하는 메타데이터 아이템(455)을 식별하기 위해서 사용될 수 있고, 그 메타데이터 아이템이 액세스되게 한다.

[0073] 도 8b는 메타데이터 아이템의 2개의 특정 예의 형태를 도시한다. 양쪽 예에 있어서, 데이터 그래뉼은 사이즈가 16 바이트이지만, 다른 구현에서 데이터 그래뉼 사이즈는 다르게 될 수 있는 것으로 상정된다. 능력 예에 따라서, 16 바이트의 각각의 데이터 그래뉼은, 본 개시에서 능력 태그로서도 언급되는, 메타데이터의 관련된 하나의 비트를 갖는데, 이는, 관련된 데이터 그래뉼이 능력을 식별하는지, 또는 단지 일반적인 목적의 데이터를 식별하는지를 식별하기 위해서 사용된다.

[0074] 도 8b의 제2예는 할당 태그를 도시하는데, 여기서 특별한 예에 있어서, 나타낸 4 비트의 메타데이터는 각각의 16 바이트 그래뉼과 관련된다. 이들 4 비트의 메타데이터는, 가드 태그로서도 언급되는 할당 태그로 언급될 수 있다. 메모리 내의 특별한 위치에 대한 액세스 요청이 있을 때, 어드레스 태그는 타깃 어드레스와 관련해서 특정될 수 있고, 어드레스 태그는 메모리 내의 관련 데이터 그래뉼과 관련된 할당 태그와 비교될 수 있다. 매치가 가드 태그와 어드레스 태그 사이에서 검출되는지에 의존해서 다른 액션이 취해질 수 있다.

[0075] 메타데이터 아이템은 다양한 형태를 취할 수 있지만, 2개의 예가 도 9 및 10을 참조해서, 이하 논의된다.

[0076] 도 9는, 이들 데이터 블록이 능력을 표현하는지(바운드된 포인터 및 관련된 제한 정보의 경우) 또는 정상 데이터를 표현하는지를 식별하기 위해서 개별 데이터 블록과의 관련해서 어떻게 사용되는지를 개략적으로 도시한다. 특히, 능력-인식 시스템 내의 메모리 어드레스 스페이스(510)는 일련의 데이터 블록(515)을 저장할 수 있는데, 이는, 전형적으로, 특정된 사이즈를 가질 것이다. 순전히 도시의 목적을 위해서, 이 예에 있어서, 각각의 데이터 블록은 128 비트를 포함하는 것으로 상정된다. 각각의 데이터 블록(515)과 관련해서, 태그 필드(520)가 제공되는데, 이는, 일례에 있어서, 태그 비트로서 언급된 싱글 비트 필드이고, 이는, 관련된 데이터 블록이 능력을 표현하는 것을 식별하도록 설정되며, 관련된 데이터 블록이 정상 데이터를 표현하는 것을 표시하기 위해서 클리어되므로, 능력으로서 처리될 수 없다. 세트 또는 클리어 상태와 관련된 실재 값은 실시예에 따라서 변할 수 있지만, 순전히 예시적인 방식으로, 하나의 실시예에 있어서, 태그 비트가 1의 값을 가지면, 이는, 관련된 데이터 블록이 능력을 갖고, 이것이 0의 값을 가지면, 관련된 데이터 블록이 정상 데이터를 포함하는 것을 표시하는 것으로 이해될 것이다.

[0077] 능력이 도 9에 나타낸 능력 레지스터(500)와 같은 바운드된 포인터 레지스터(또한, 본 개시에서 능력 레지스터로서 언급) 내에 로드될 때, 태그 비트는 능력 정보와 함께 이동한다. 따라서, 능력이 능력 레지스터(500) 내에 로드될 때, 포인터(502), 범위 정보(504) 및 제한 정보(506)(또한, 허가 정보로도 언급)는 능력 레지스터 내에 로드될 것이다. 추가적으로, 그 능력 레지스터와 관련해서, 또는 그것 내의 특정 비트 필드로서, 태그 비트(508)는 능력을 표현하는 그 콘텐츠를 식별하기 위해서 설정될 것이다. 유사하게, 능력이 메모리에 다시 저장될 때, 관련 태그 비트(520)는 능력이 저장되는 데이터 블록과 관련해서 설정될 것이다. 이러한 접근에 의해서, 능력과 정상 데이터 사이를 구별하는 것이 가능하게 되므로, 정상 데이터가 능력으로서 사용될 수 없는 것을 보장한다.

[0078] 본 발명 기술이 적용될 수 있는 메타데이터 아이템의 다른 예로서, 도 10은 태그-가드된 메모리 액세스(할당 태

그로서도 본 개시에서 언급되는 가드 태그)의 개념을 개략적으로 도시한다. 메모리 시스템 내의 메모리 위치를 언급하기 위해서 사용된 물리적인 어드레스 스페이스는 각각이 소정 수의 어드레스 가능한 위치를 포함하는 다수의 블록(630)으로 논리적으로 파티션될 수 있다. 간결함을 위해서, 도 10의 예에 있어서, 각각의 블록(630)은 4개의 메모리 위치를 포함하지만, 다른 블록 사이즈가 역시 사용될 수 있다. 각각의 블록(630)은 대응하는 가드 태그(632)와 관련된다. 소정 수의 블록(630)과 관련된 가드 태그는 함께 모을 수 있고 물리적인 어드레스 스페이스 내에서 다른 아키텍처하게 액세스 가능한 물리적인 위치(634) 내에, 또는 아키텍처하게 액세스 가능하지 않은(동일한 물리적인 어드레스 스페이스에 맵핑되지 않은) 메인 메모리 내에 제공된 추가적인 스토리지 위치 내에 저장될 수 있다. 분리의 비-아키텍처하게 액세스 가능한 스토리지의 사용은, 일부 경우에 있어서, 가드 태그 값을 캐싱하기 위한 데이터 캐시 내의 스페이스를 사용하는 것을 회피하는 것이 바람직할 수 있는데, 이는, 정규 코드의 실행에 영향을 줄 수 있고 코히어런시(Coherency) 관리를 더 복잡하게 만들 수 있다. 추가적인 태그 캐시가, 태그가 메인 메모리로부터 액세스되어야 하는 것보다 빠른 액세스를 위해서, 비-아키텍처하게 액세스 가능한 스토리지로부터 태그 값을 캐싱하기 위한 처리 회로(15)가 제공될 수 있다.

[0079] 태그 스토리지 위치(634)가 각각의 블록(630)에 대응하는 특별한 맵핑은 처리 회로(15)의 로드/저장 유닛에 의해서 제어될 수 있고, 하드와이어드될 수 있거나 또는 프로그램 가능하게 될 수 있다. 대안적으로, 메모리 시스템(35)은 태그가 저장되는 곳을 결정할 수 있고, 처리 회로는 공지 또는 케어(care)될 필요는 없다. 도 10에서 각각의 태그(632)는 물리적인 어드레스의 블록과 관련되는 한편, 이는, 또한, 가상의 메모리 어드레스 스페이스 내의 가상의 메모리 위치와 관련된 가드 태그(623)를 제공하는 것이 가능하지만, 이는, 각각의 메모리 액세스에 대한 일부 추가적인 어드레스 번역을 요구할 수 있다. 그러므로, 가드 태그(632)와 물리적인 메모리 위치를 관련시킴으로써, 이것은 실행을 개선할 수 있다. 일반적으로, 정확히 어떻게 가드 태그(632)가 물리적인 어드레스 스페이스의 대응하는 블록(630)과 관련되는지는 특별한 구현에 대한 초이스이다. 일반적으로, 요구된 모든 것은, 메모리의 주어진 블록과 관련된 가드 태그(632)가 액세스 및 비교될 수 있는 것이다.

[0080] 그러므로, 태그-가드된 메모리 액세스가 요구될 때, 어드레스 태그(640)(이는, 액세스되는 어드레스된 위치(644)를 식별하는 타깃 어드레스(642)와 관련되는)는, 어드레스된 위치(644)를 포함하는 메모리 위치(630)의 블록과 관련되는 가드 태그(632)에 대해서 비교된다. 예를 들어, 도 10에 있어서, 타깃 어드레스(642)는 도 10의 어드레스 스페이스 내에서 마크된(644) 메모리 내의 소정의 위치 B1을 포인트한다. 그러므로, 위치 B1을 포함하는 위치 B의 블록과 관련되는 가드 태그 B는 타깃 어드레스(642)와 관련된 어드레스 태그(640)에 대항해서 비교한다. 도 10의 상단에 나타낸 바와 같이, 어드레스 태그(640)는 타깃 어드레스 자체의 선택된 비트의 기능으로서 결정될 수 있다. 특히, 어드레스 태그는 어드레스된 위치(644)로서 선택되는 특정 메모리 위치를 표시하기 위해서 사용되지 않은 타깃 어드레스의 부분 내에서 비트로부터 결정될 수 있다. 예를 들어, 일부 구현에 있어서, 타깃 어드레스의 비트의 상단 부분은 사인 확장(모든 0들 또는 모든 1들)과 같은 소정의 고정된 값을 항상 갖고, 그러므로, 어드레스는, 임의의 태그 값으로 이들 사용되지 않은 비트를 오버라이팅함으로써 어드레스 태그(640)로 태그될 수 있다. 특별한 어드레스 태그 값은, 예를 들어, 프로그래머 또는 컴파일러에 의해서 선택될 수 있다. 어드레스 태그 및 가드 태그(632)는 상대적으로 작은 비트의 수, 예를 들어, 4 비트가 될 수 있고, 그러므로, 메모리 내에서 및 타깃 어드레스 내에서 많은 스페이스를 점유하지 않을 필요가 있다. 4 비트의 태그 스페이스를 제공하는 것은, 즉, 태그의 16개의 가능한 값을 제공하는 것은, 흔히, 많은 공통 타입의 메모리 액세스 에러를 검출하는데 충분할 수 있다.

[0081] 그러므로, 태그-가드된 메모리 액세스가 소정의 CPU(15)에 의해서 수행될 때, 그 CPU(15)의 로드/저장 유닛은 어드레스 태그(640) 및 어드레스된 위치(644)를 포함하는 블록(630)과 관련된 가드 태그(632)를 포함하고, 이들이 매치하는지를 결정한다(대안적으로, 태그 비교가 메모리 시스템(35) 내에서 수행될 수 있다). 로드/저장 유닛은, 어드레스 태그(640) 및 가드 태그(632)가 매치되는지를 가리키는 매치 인디케이션을 생성한다. 예를 들어, 이 매치 인디케이션은, 어드레스 태그(640)와 가드 태그(632) 사이에 미스매치가 있으면 생성되는 폴트(fault) 신호가 될 수 있거나, 또는 매치가 있었는지를 표시하는 상태 레지스터 내에 위치된 인디케이션, 또는 에러가 검출되었던 어드레스를 표시하는 에러 보고 및/또는 에러를 트리거한 명령의 명령 어드레스에 부가된 엔트리가 될 수 있다.

[0082] 도 2 또는 5를 참조해서 논의된 형태의 메모리 액세스 포트를 사용함으로써, 상대적으로 작은 어드레스 스페이스가 도 11에 나타낸 어드레스 스페이스(700)와 같은 디버거에 사용 가능하게 만들어질 수 있다. 그 작은 어드레스 스페이스를 통해서, 디버거는, 액세스가, 디버그되는 디바이스의 메모리(35) 내의 매우 더 큰 어드레스 스페이스에 수행되게 할 수 있는 커맨드를 발생할 수 있고, 수행되는 넓은 다양한 다른 타입의 액세스 동작을 특정할 수 있다. 도 11에 개략적으로 도시된 바와 같이, 어드레스 스페이스(700) 내의 다른 어드레스는 메모리 액

세스 포트 내의 다른 레지스터를 식별하기 위해서 사용될 수 있다. 예를 들어, 커맨드 내에 특정된 어드레스(705)는 버스 어드레스 레지스터(55)를 식별할 수 있고, 또 다른 어드레스(710)는 데이터 레지스터(60)를 식별할 수 있으며, 또 다른 어드레스(715)는 메타데이터 레지스터(70)를 식별할 수 있는 한편, 또 다른 어드레스(720)는 버스 제어 레지스터(65)를 식별할 수 있다.

[0083] 버스 제어 레지스터가 사용되지 않은 도 5의 대안적인 접근을 사용할 때, 다양한 다른 어드레스(710, 725, 730)가 커맨드가 발행될 수 있는 다른 데이터 레지스터를 식별하기 위해서 사용될 수 있다.

[0084] 요구된 작은 어드레스 스페이스(700)에 기인해서, 상대적으로 단순한 인터페이스가 디버거(10)와 메모리 액세스 포트(25) 사이에 제공될 수 있다. 예를 들어, 디버그 인터페이스는 JTAG, 직렬 와이어 또는 PCI 인터페이스, 또는 소정의 다른 적합한 인터페이스로 형성될 수 있다.

[0085] 도 11에서 어드레스 스페이스가 4 K바이트 사이즈로서 특정된 예로서 나타내는 반면, 다른 구현에서 어드레스 스페이스는 다른 사이즈로 될 수 있다. 실제로, 어드레스 사이즈에 대한 일부 구현에 있어서, 예를 들어, 256 바이트가 충분할 수 있는데, 이것보다 상당히 작게 되는 것이 가능하다.

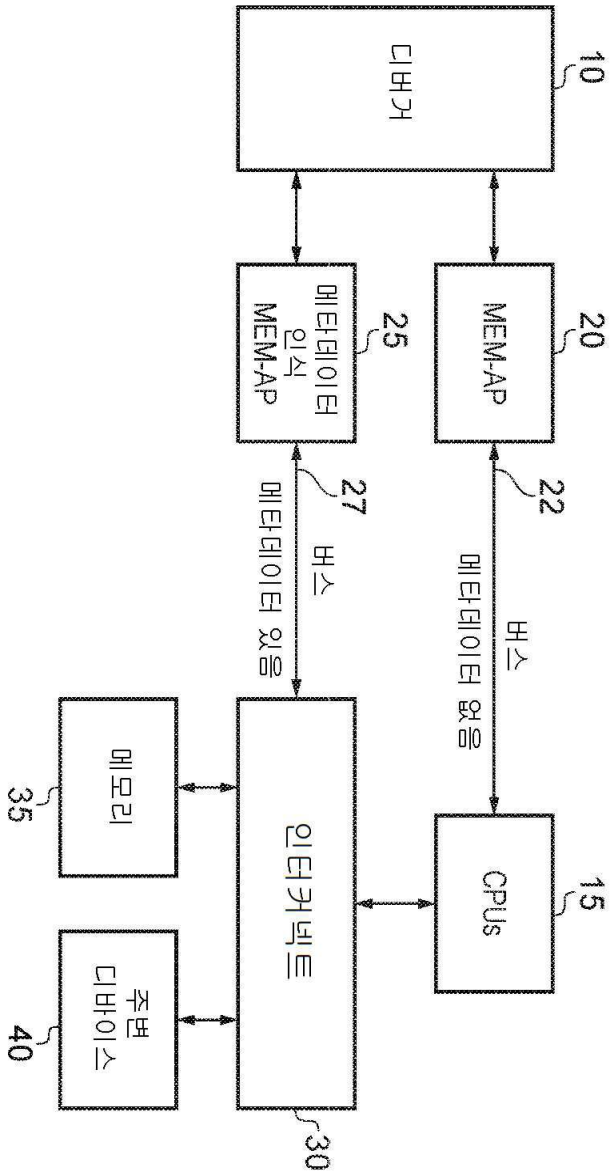
[0086] 상기 다양한 예의 구성으로부터, 본 개시에 기술된 기술이, 관련된 데이터 그래뉼에 부가해서, 또는 그 대신, 메타데이터 아이템이 메모리 내에 액세스되게 허용하는 방식으로, 디버거가 메모리 액세스를 직접 개시하는 효율적인 메커니즘을 제공하는 것으로 이해될 것이다. 요구된 액세스 동작은, 그 다음, 디바이스의 처리 회로 상에서 처리를 중단할 필요 없이, 필요한 곳에서 효율적으로 및 원자적으로 수행될 수 있다. 특히, 본 개시에 기술된 메타데이터 인식 메모리 액세스 포트의 사용을 통해서, 디버거는, 필요한 메타데이터 액세스 동작을 수행하기 위해서, 디바이스의 처리 회로를 사용할 필요 없이, 메모리 내의 메타데이터 아이템에 액세스할 수 있다.

[0087] 본 출원에서, 단어 "...하도록 구성된"은 디바이스의 엘리먼트가 규정된 동작을 수행할 수 있게 하는 구성을 갖는 것을 의미하도록 사용된다. 이 콘텍스트에 있어서, "구성"은 하드웨어 또는 소프트웨어의 상호 접속의 배열 또는 방식을 의미한다. 예를 들어, 디바이스는 규정된 동작을 제공하는 전용의 하드웨어를 가질 수 있거나, 또는 프로세서 또는 다른 처리 디바이스가 기능을 수행하도록 프로그래밍 될 수 있다. "하도록 구성된"은 규정된 동작을 제공하기 위해 소정의 방법으로 디바이스 엘리먼트가 변경될 필요가 있는 것을 의미하지는 않는다.

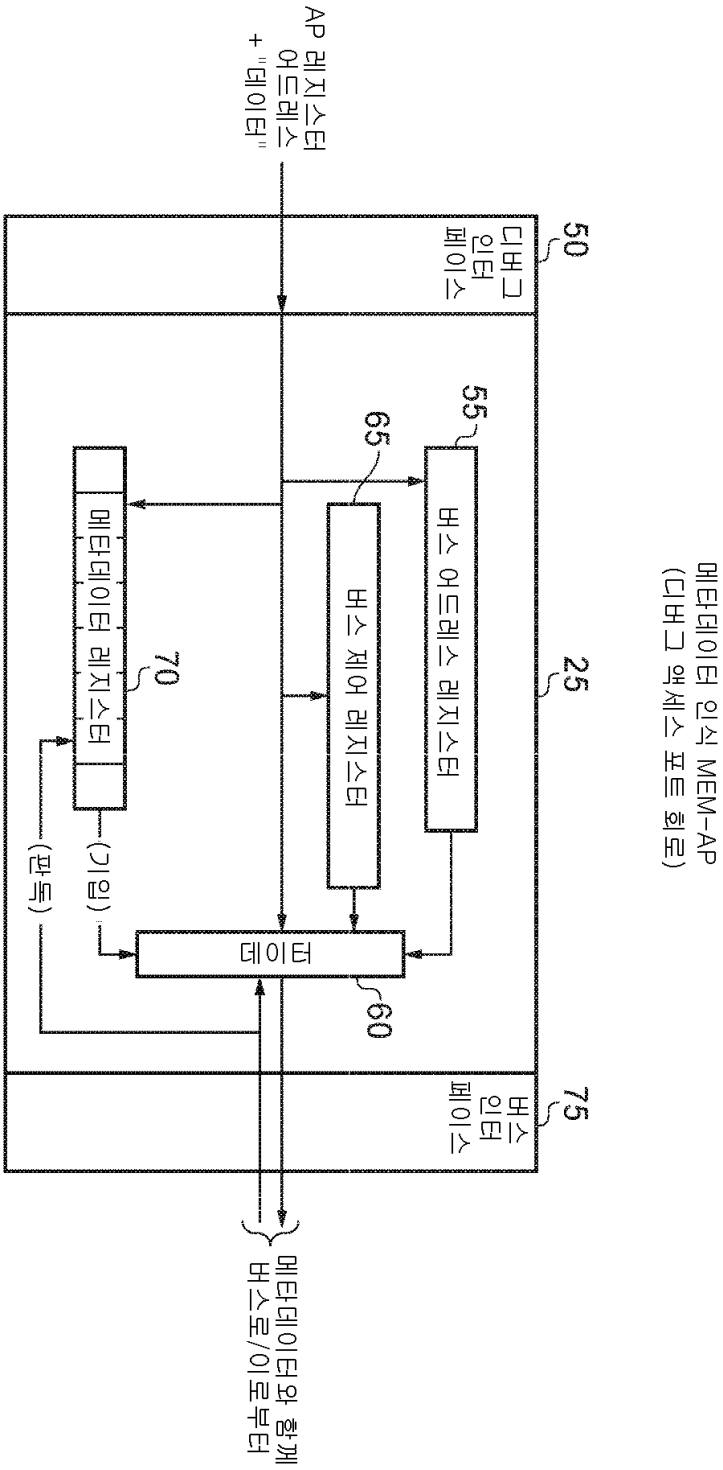
[0088] 예시적인 실시예가 첨부 도면을 참조해서 본 명세서에서 상세하게 설명되었지만, 본 발명은, 첨부된 청구 범위에 의해 규정된 바와 같은 본 발명의 범위 및 사상을 벗어남이 없이, 이러한 정확한 실시예로 제한되지 않으며, 본 기술 분야의 통상의 기술자에게는 다양한 변경, 추가 및 수정이 이루어질 수 있음을 이해해야 한다. 예를 들어, 종속항의 특징의 다양한 조합은 본 발명의 범위를 벗어나지 않고 독립항의 특징과 함께 만들어질 수 있다.

도면

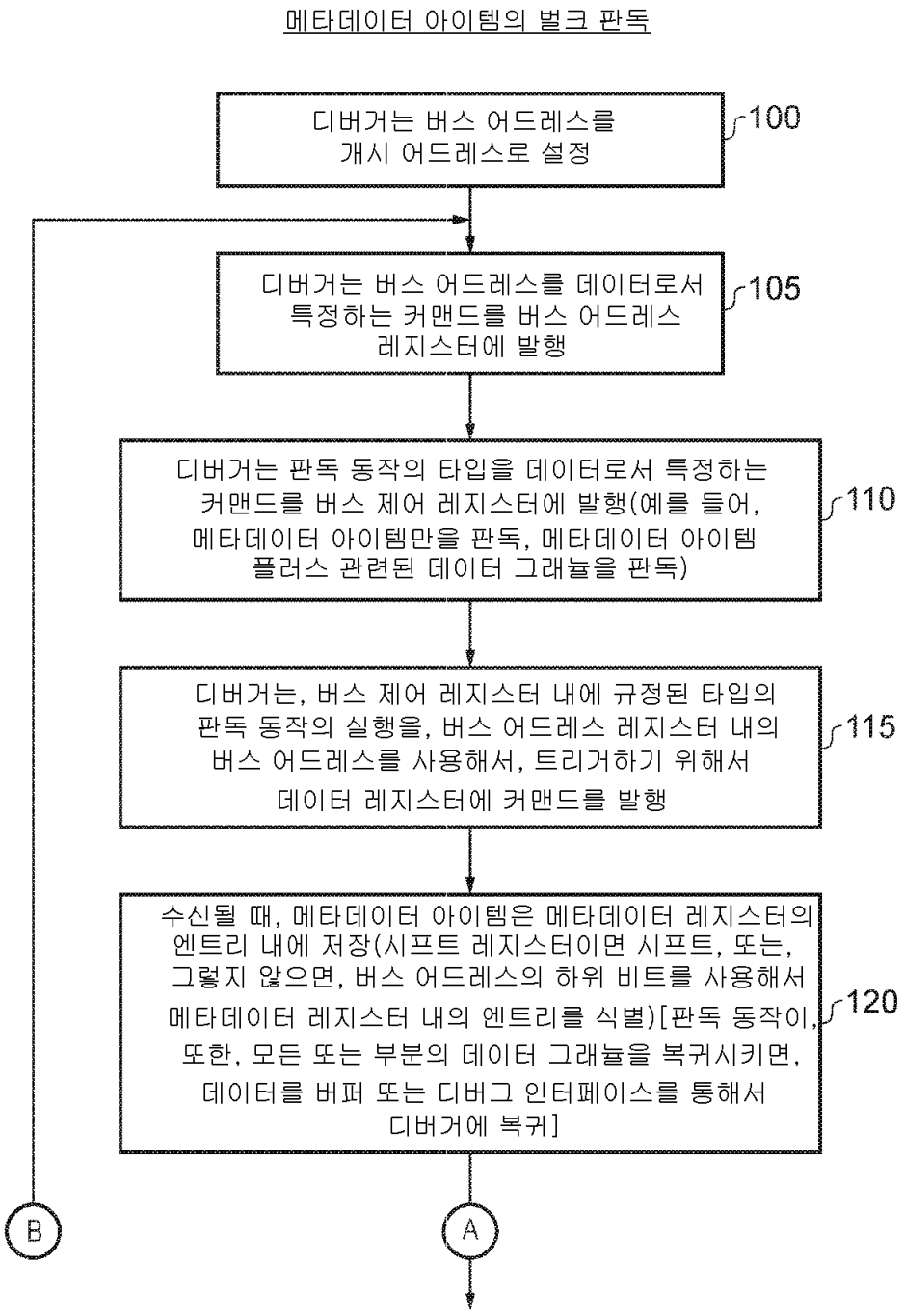
도면1


디버거
10
메타데이터 인식 MEM-AP
25
MEM-AP
20
27
메타데이터 있음
버스
22
메타데이터 없음
버스
메모리
35
인터커넥트
30
CPUs
15
주변 디바이스
40

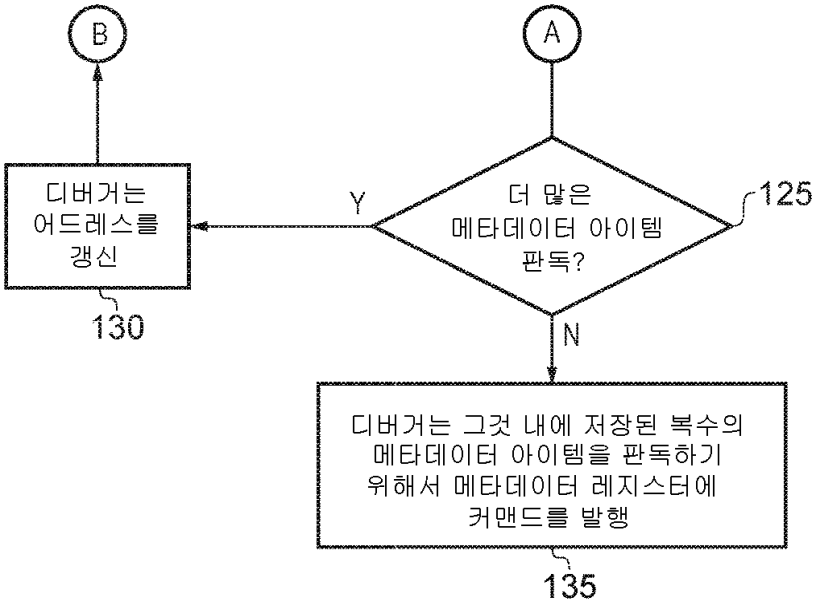
도면2


AP 레지스터 어드레스 + "데이터"
디버그 인터페이스
50
55
65
버스 어드레스 레지스터
버스 제어 레지스터
메타데이터 레지스터
70
25
(기입)
(판독)
데이터
60
버스 인터페이스
75
메타데이터와 함께 버스로/으로부터
메타데이터 인식 MEM-AP
(디버그 액세스 포트 회로)

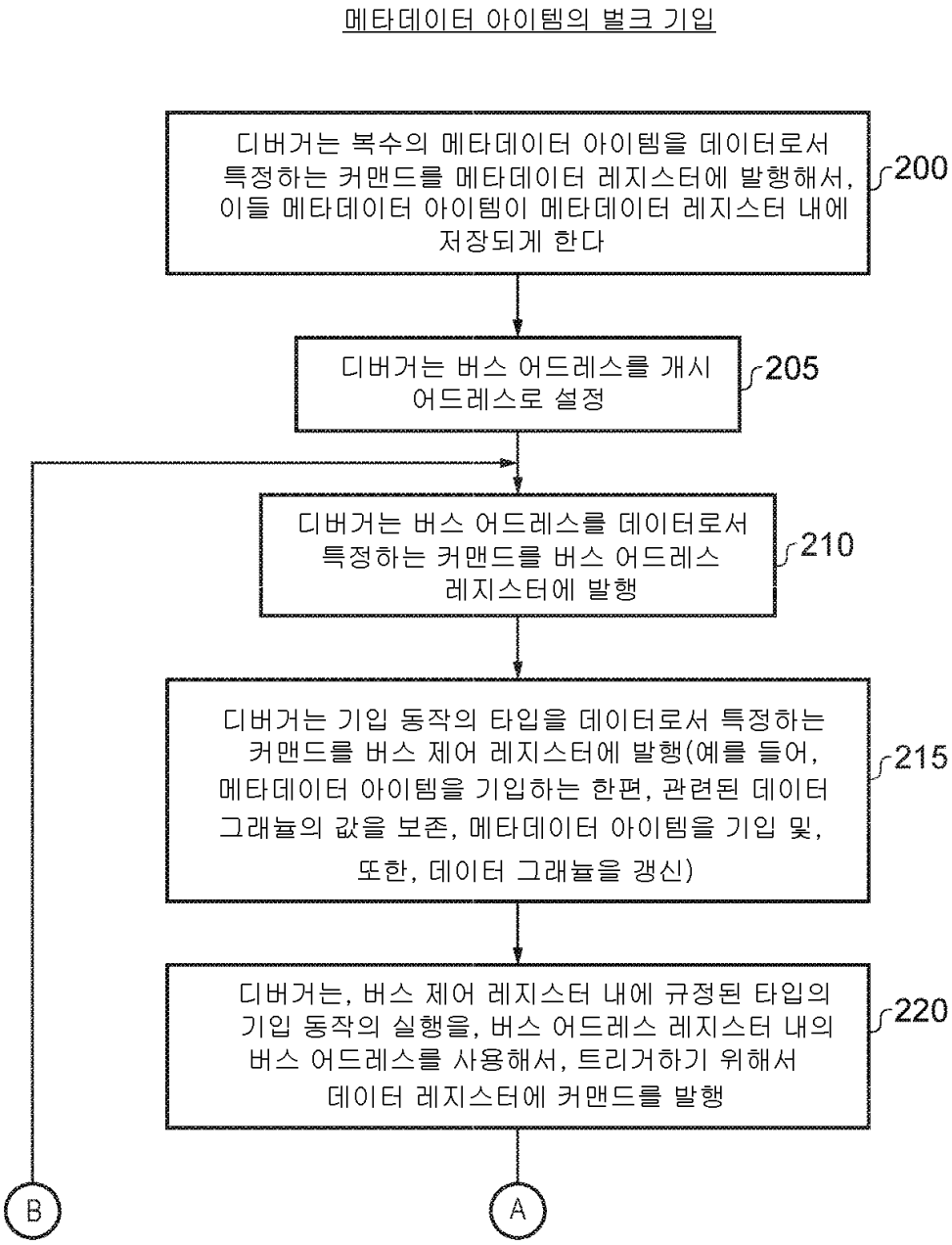
도면3a


메타데이터 아이템의 벌크 판독
디버거는 버스 어드레스를 개시 어드레스로 설정
100
디버거는 버스 어드레스를 데이터로서 특정하는 커맨드를 버스 어드레스 레지스터에 발행
105
디버거는 판독 동작의 타입을 데이터로서 특정하는 커맨드를 버스 제어 레지스터에 발행(예를 들어, 메타데이터 아이템만을 판독, 메타데이터 아이템 플러스 관련된 데이터 그래뉼을 판독)
110
디버거는, 버스 제어 레지스터 내에 규정된 타입의 판독 동작의 실행을, 버스 어드레스 레지스터 내의 버스 어드레스를 사용해서, 트리거하기 위해서 데이터 레지스터에 커맨드를 발행
115
수신될 때, 메타데이터 아이템은 메타데이터 레지스터의 엔트리 내에 저장(시프트 레지스터이면 시프트, 또는, 그렇지 않으면, 버스 어드레스의 하위 비트를 사용해서 메타데이터 레지스터 내의 엔트리를 식별)[판독 동작이, 또한, 모든 또는 부분의 데이터 그래뉼을 복귀시키면, 데이터를 버퍼 또는 디버그 인터페이스를 통해서 디버거에 복귀]
120
B
A

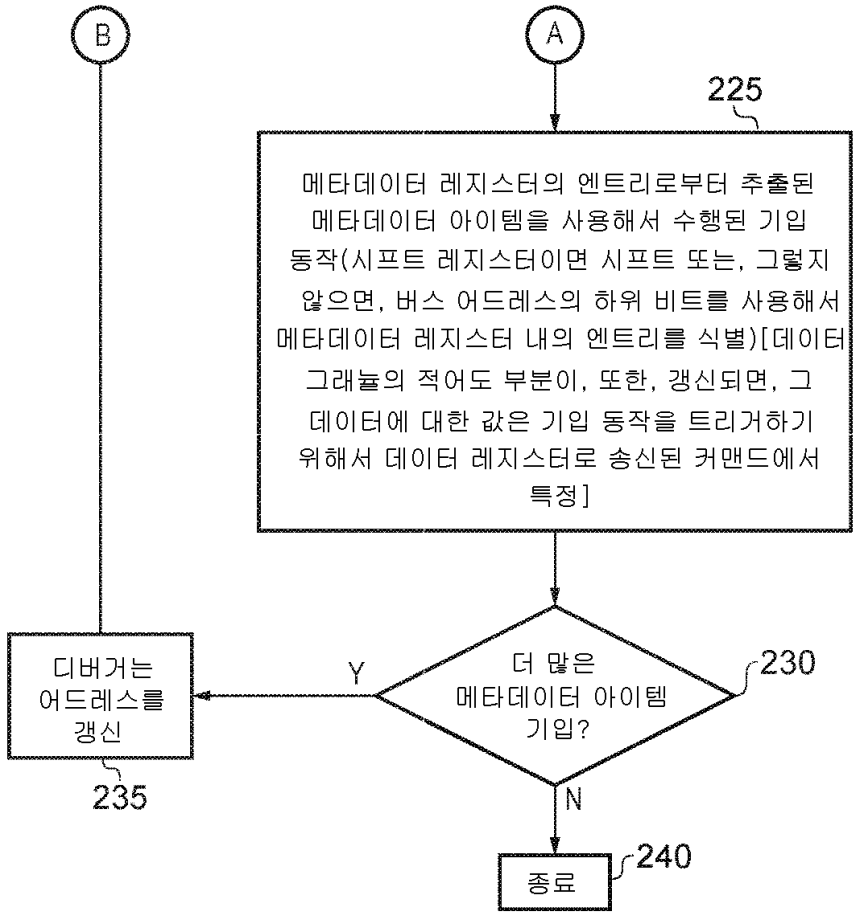
도면3b


B
A
디버거는
어드레스를
갱신
130
Y
더 많은
메타데이터 아이템
판독?
125
N
디버거는 그것 내에 저장된 복수의
메타데이터 아이템을 판독하기
위해서 메타데이터 레지스터에
커맨드를 발행
135

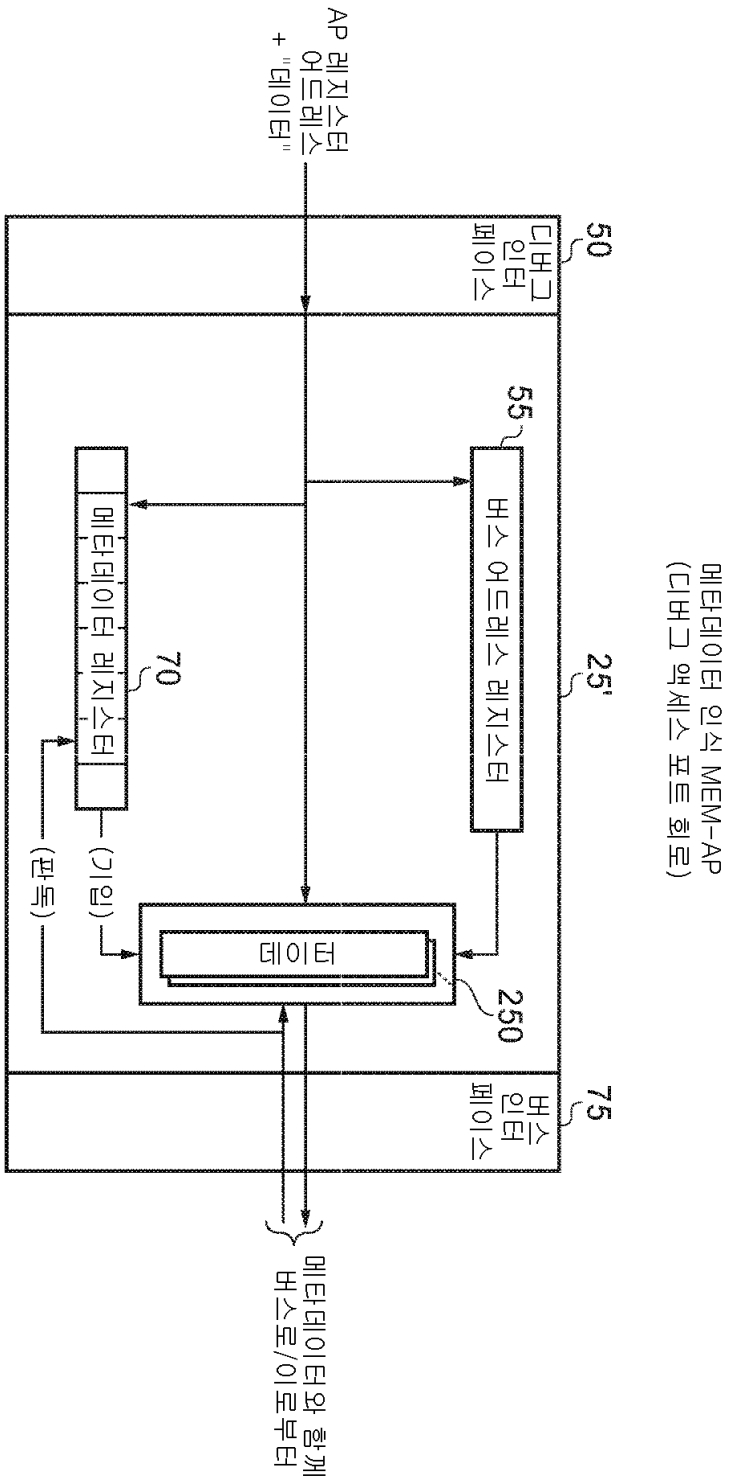
도면4a


메타데이터 아이템의 벌크 기입
디버거는 복수의 메타데이터 아이템을 데이터로서 특정하는 커맨드를 메타데이터 레지스터에 발행해서, 이들 메타데이터 아이템이 메타데이터 레지스터 내에 저장되게 한다
200
디버거는 버스 어드레스를 개시 어드레스로 설정
205
디버거는 버스 어드레스를 데이터로서 특정하는 커맨드를 버스 어드레스 레지스터에 발행
210
디버거는 기입 동작의 타입을 데이터로서 특정하는 커맨드를 버스 제어 레지스터에 발행(예를 들어, 메타데이터 아이템을 기입하는 한편, 관련된 데이터 그래뉼의 값을 보존, 메타데이터 아이템을 기입 및, 또한, 데이터 그래뉼을 갱신)
215
디버거는, 버스 제어 레지스터 내에 규정된 타입의 기입 동작의 실행을, 버스 어드레스 레지스터 내의 버스 어드레스를 사용해서, 트리거하기 위해서 데이터 레지스터에 커맨드를 발행
220
B
A

도면4b

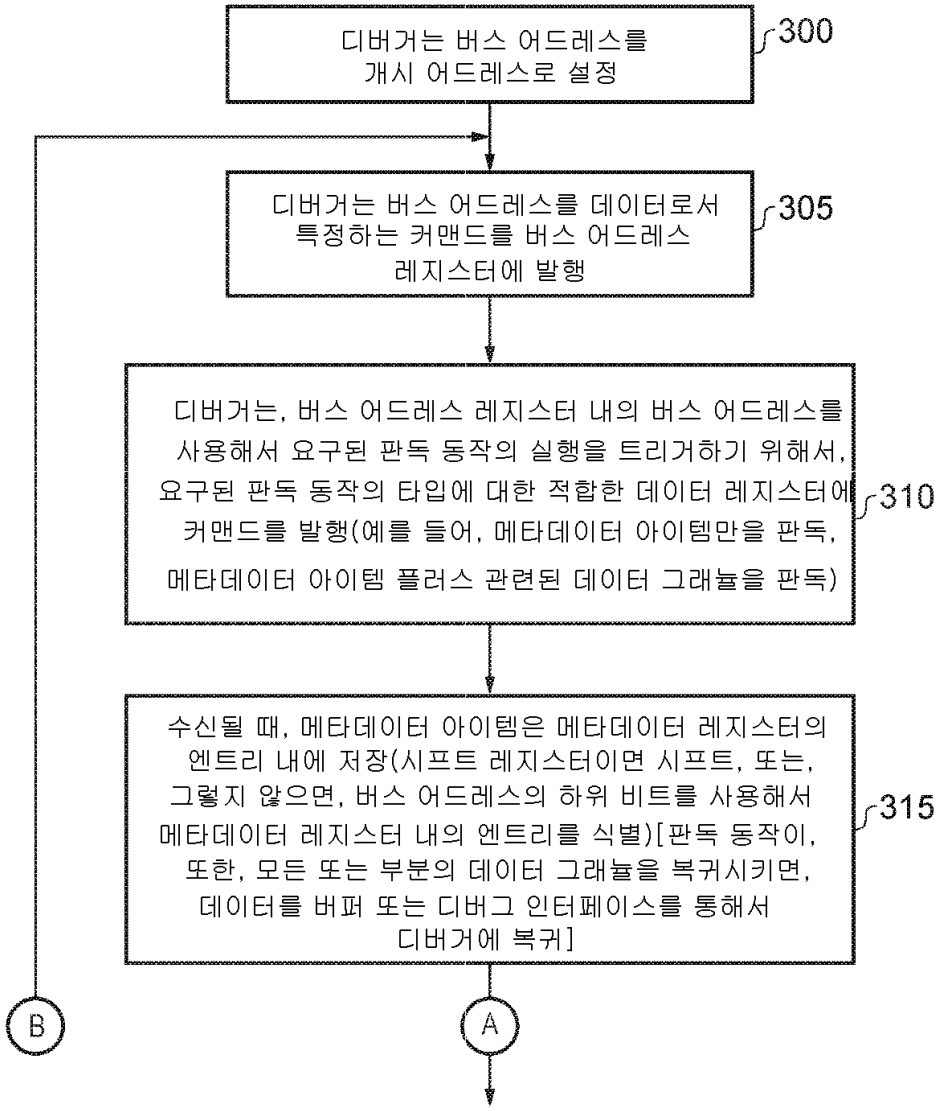

B
A
225
메타데이터 레지스터의 엔트리로부터 추출된 메타데이터 아이템을 사용해서 수행된 기입 동작(시프트 레지스터이면 시프트 또는, 그렇지 않으면, 버스 어드레스의 하위 비트를 사용해서 메타데이터 레지스터 내의 엔트리를 식별)[데이터 그래뉼의 적어도 부분이, 또한, 갱신되면, 그 데이터에 대한 값은 기입 동작을 트리거하기 위해서 데이터 레지스터로 송신된 커맨드에서 특정]
더 많은 메타데이터 아이템 기입?
230
Y
N
디버거는 어드레스를 갱신
235
종료
240

도면5

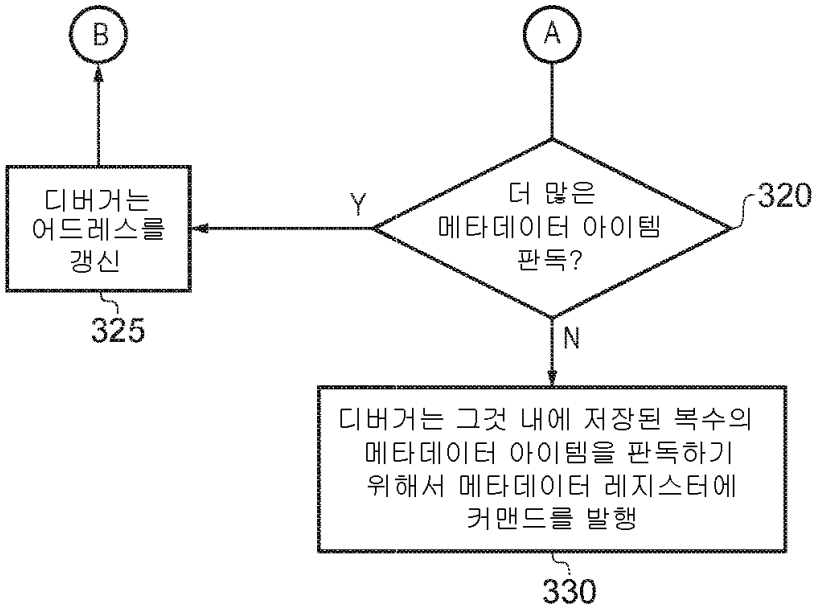

AP 레지스터 어드레스 + "데이터"
50
디버그 인터페이스
55
버스 어드레스 레지스터
25'
메타데이터 레지스터
70
(기입)
(판독)
데이터
250
75
버스 인터페이스
메타데이터와 함께 버스로/으로부터
메타데이터 인식 MEM-AP
(디버그 액세스 포트 회로)

도면6a

메타데이터 아이템의 벌크 판독(도 5 접근 사용)

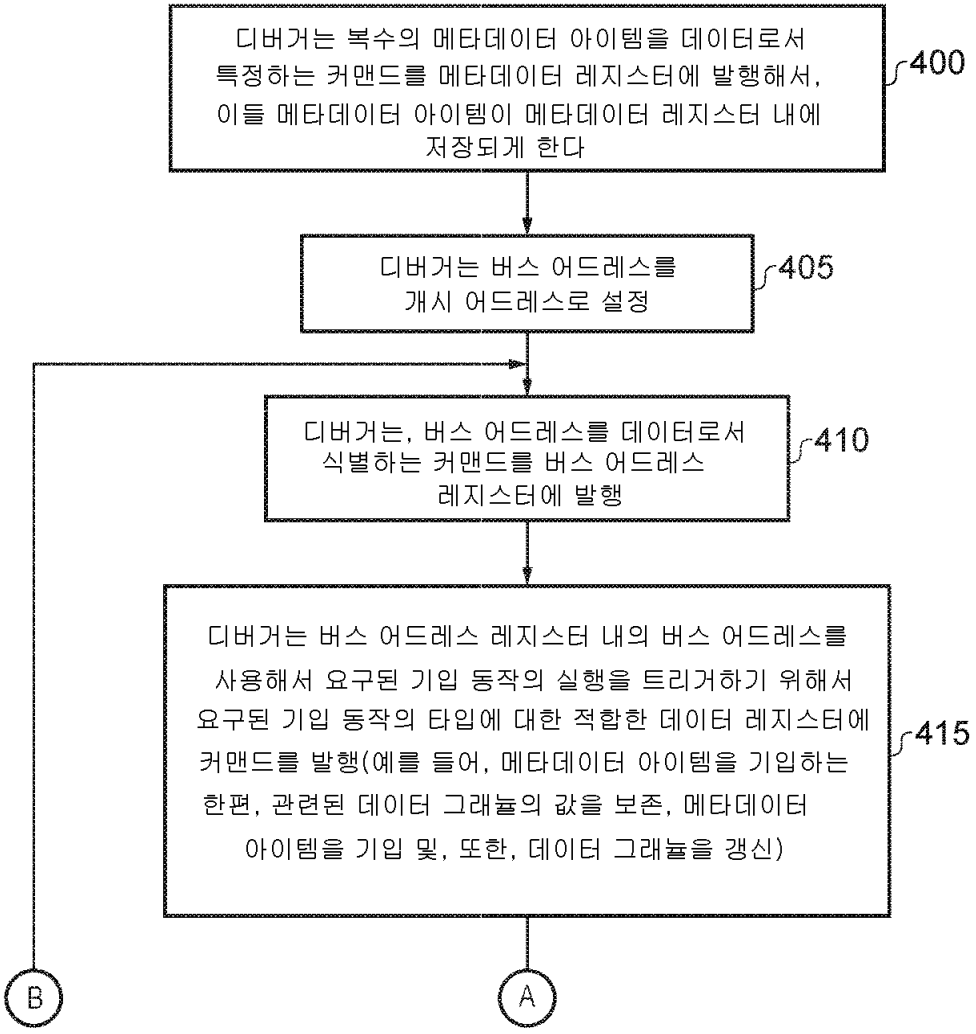


도면6b

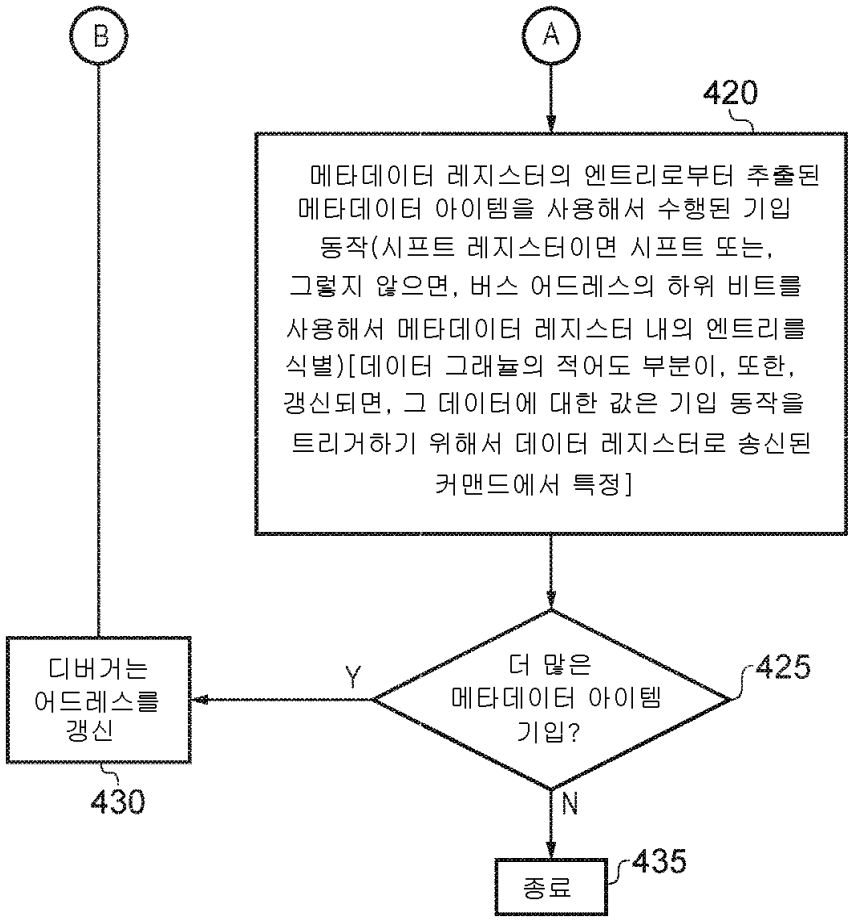

B
A
디버거는
어드레스를
갱신
325
Y
더 많은
메타데이터 아이템
판독?
320
N
디버거는 그것 내에 저장된 복수의
메타데이터 아이템을 판독하기
위해서 메타데이터 레지스터에
커맨드를 발행
330

도면7a

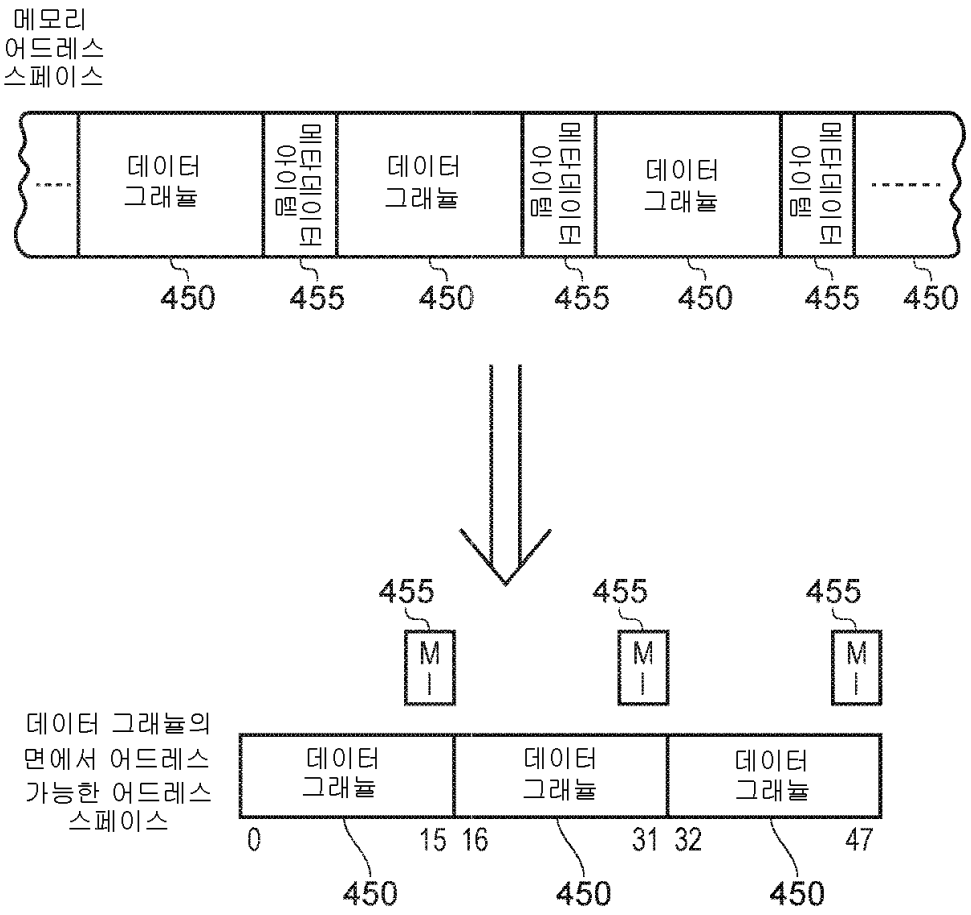
메타데이터 아이템의 벌크 기입(도 5의 접근 사용)



도면7b


B
A
420
메타데이터 레지스터의 엔트리로부터 추출된 메타데이터 아이템을 사용해서 수행된 기입 동작(시프트 레지스터이면 시프트 또는, 그렇지 않으면, 버스 어드레스의 하위 비트를 사용해서 메타데이터 레지스터 내의 엔트리를 식별)[데이터 그래픞의 적어도 부분이, 또한, 갱신되면, 그 데이터에 대한 값은 기입 동작을 트리거하기 위해서 데이터 레지스터로 송신된 커맨드에서 특정]
더 많은 메타데이터 아이템 기입?
425
Y
N
디버거는 어드레스를 갱신
430
종료
435

도면8a



도면8b

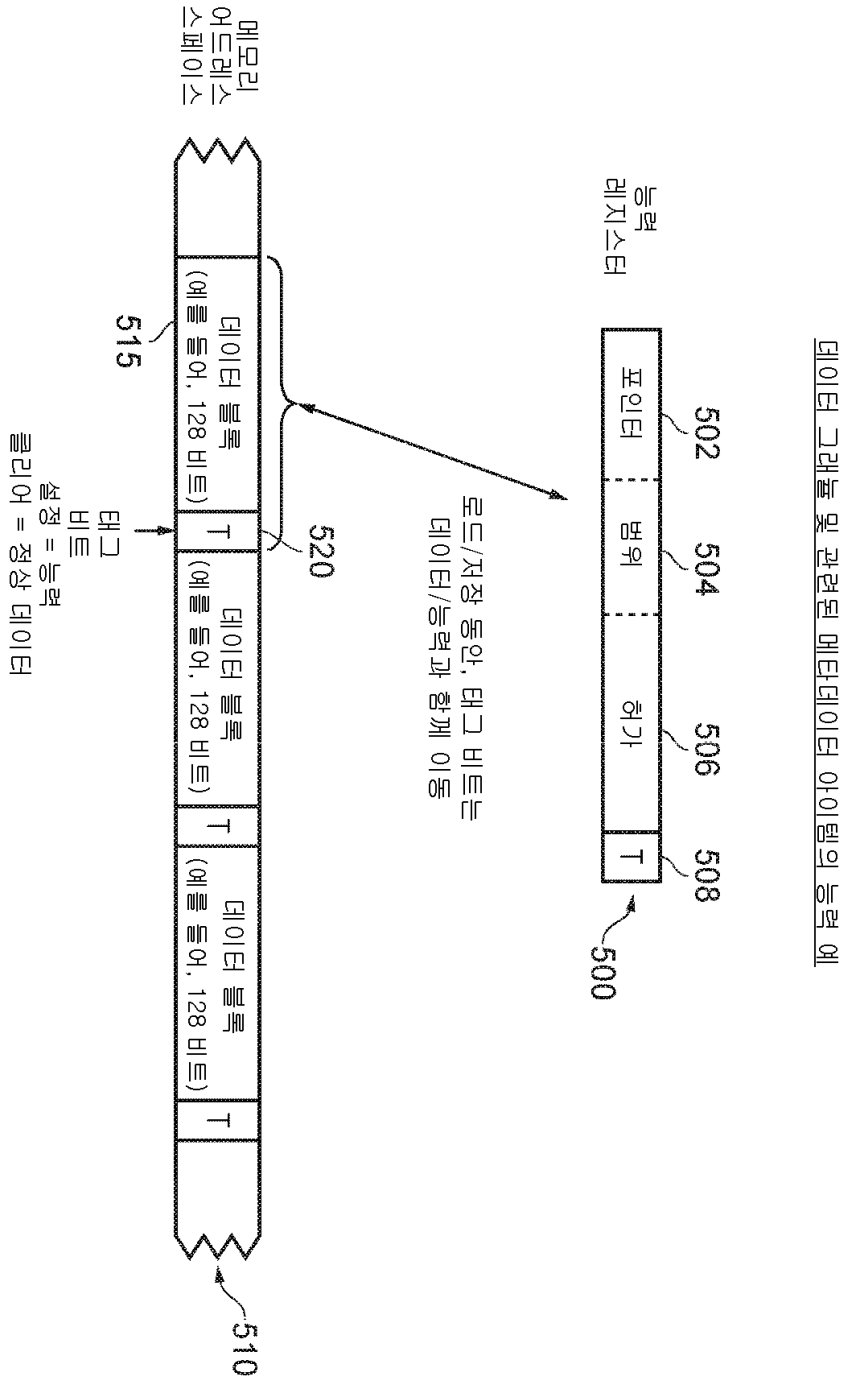
능력(각각의 16 바이트 데이터에 대한 1 비트 메타데이터)

어드레스	데이터
0x1000:	16 바이트 데이터, 1 비트 메타데이터
0x1010:	16 바이트 데이터, 1 비트 메타데이터
0x1020:	16 바이트 데이터, 1 비트 메타데이터

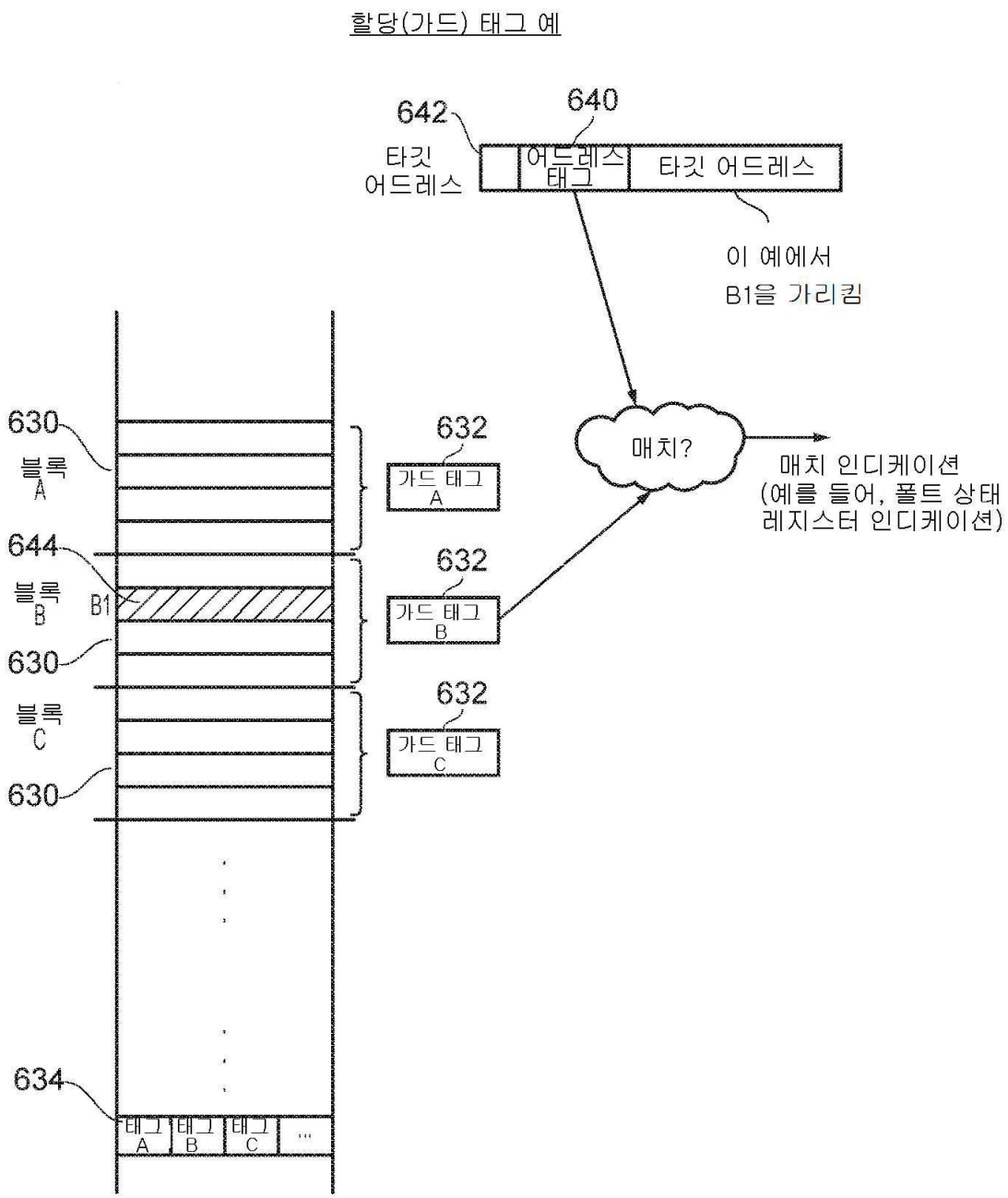
할당 태깅(각각의 16 바이트 데이터에 대한 4 비트 메타데이터)

어드레스	데이터
0x1000:	16 바이트 데이터, 4 비트 메타데이터
0x1010:	16 바이트 데이터, 4 비트 메타데이터
0x1020:	16 바이트 데이터, 4 비트 메타데이터

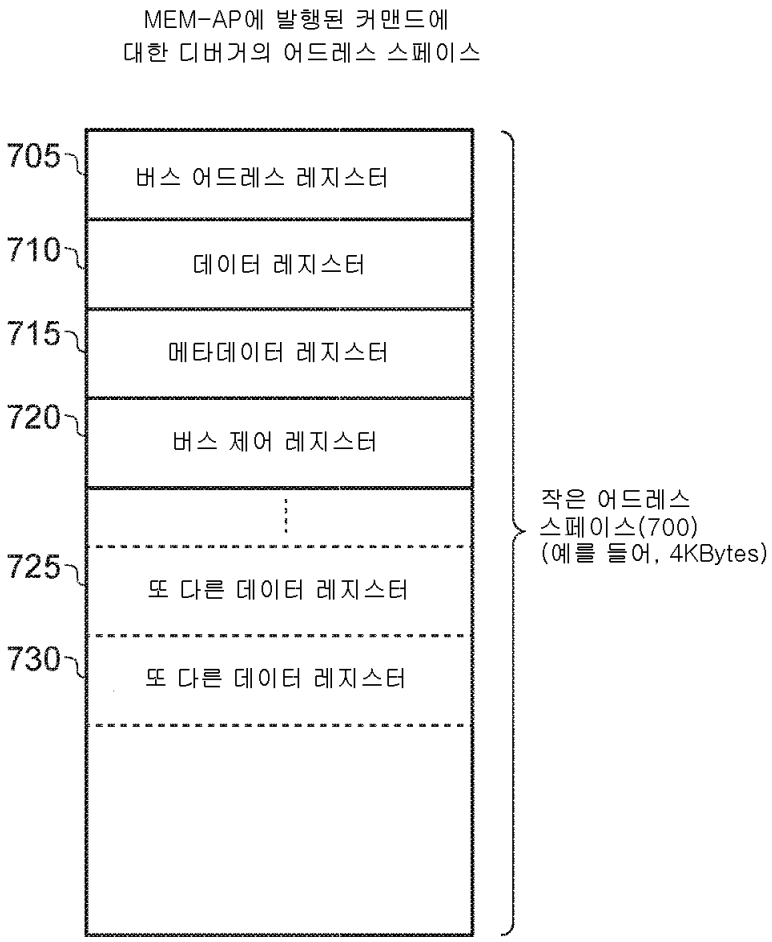
도면9


데이터 그래프 및 관련된 메타데이터 아이템의 등록 예
등록 레지스터
포인터
범위
허가
T
502
504
506
508
500
로드/저장 동안, 태그 비트는 데이터/능력과 함께 이동
메모리 어드레스 스페이스
데이터 블록 (예를 들어, 128 비트)
T
515
520
510
태그 비트 설정 = 능력 클리어 = 정상 데이터

도면10


할당(가드) 태그 예
642
640
타깃
어드레스
어드레스
태그
타깃 어드레스
이 예에서
B1을 가리킴
630
블록
A
632
가드 태그
A
매치?
매치 인디케이션
(예를 들어, 폴트 상태
레지스터 인디케이션)
644
블록
B
B1
632
가드 태그
B
630
블록
C
632
가드 태그
C
630
634
태그
A
태그
B
태그
C
...

도면11


MEM-AP에 발행된 커맨드에
대한 디버거의 어드레스 스페이스
705
버스 어드레스 레지스터
710
데이터 레지스터
715
메타데이터 레지스터
720
버스 제어 레지스터
725
또 다른 데이터 레지스터
730
또 다른 데이터 레지스터
작은 어드레스
스페이스(700)
(예를 들어, 4KBytes)