(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2024년09월20일
(11) 등록번호 10-2707683
(24) 등록일자 2024년09월11일

(51) 국제특허분류(Int. Cl.)
G06F 13/16 (2006.01) *G06F 13/42* (2006.01)
(52) CPC특허분류
G06F 13/1689 (2013.01)
G06F 13/4234 (2013.01)
(21) 출원번호 10-2016-0088038
(22) 출원일자 2016년07월12일
심사청구일자 2021년07월09일
(65) 공개번호 10-2018-0007374
(43) 공개일자 2018년01월23일
(56) 선행기술조사문헌
US20140369597 A1*
US20080320254 A1*
US20140250281 A1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
삼성전자주식회사
경기도 수원시 영통구 삼성로 129 (매탄동)
(72) 발명자
차규환
경기도 수원시 영통구 영통로 232, 813동 1001호 (영통동, 두산.우성.한신아파트)
(74) 대리인
특허법인 고려

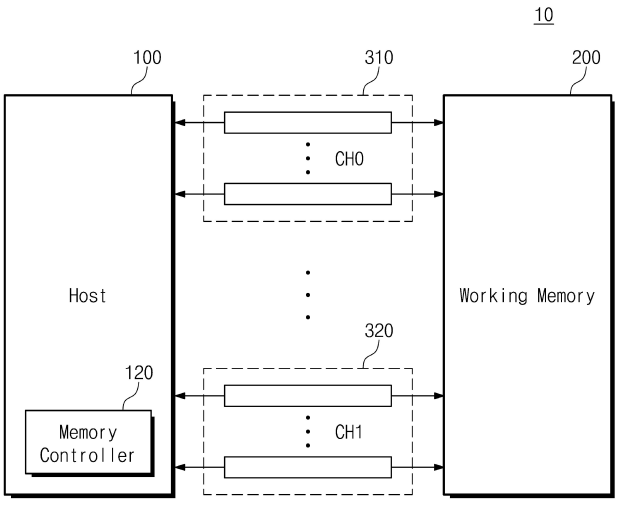
전체 청구항 수 : 총 20 항 심사관 : 이진

(54) 발명의 명칭 **메모리 채널의 소프트웨어 트레이닝을 수행하는 전자 장치 및 그것의 메모리 채널 트레이닝 방법**

(57) 요 약

본 발명의 실시 예에 따른 전자 장치는, 데이터를 저장하는 메모리 장치, 그리고 상기 메모리 장치를 워킹 메모리로 사용하는 시스템 온 칩을 포함하되, 상기 시스템 온 칩은 상기 메모리 장치의 제 1 영역에 트레이닝 코드를 로드하고, 로드된 상기 트레이닝 코드를 실행하여 상기 메모리 장치의 제 2 영역에 대한 소프트웨어 트레이닝을 수행한다.

대 표 도 - 도1



명 세 서

청구범위

청구항 1

데이터를 저장하도록 구성되고 제1 영역 및 제2 영역을 포함하는 메모리 장치; 그리고

상기 메모리 장치를 워킹 메모리로 사용하는 시스템 온 칩을 포함하되,

상기 시스템 온 칩은 상기 제 1 영역에 상기 제2 영역의 동작 특성을 최적화하기 위한 트레이닝 코드를 로드하고, 로드된 상기 트레이닝 코드를 실행하여 상기 제 2 영역에 대한 소프트웨어 트레이닝을 수행하여 상기 제2 영역의 상기 동작 특성을 최적화하는 전자 장치.

청구항 2

제 1 항에 있어서,

상기 시스템 온 칩은 상기 트레이닝 코드의 실행시 상기 제 2 영역의 접근 파라미터를 테스트하고 조정하기 위한 트레이닝 데이터를 로드하는 전자 장치.

청구항 3

제 1 항에 있어서,

상기 시스템 온 칩은 상기 메모리 장치에 대한 채널 인터리빙 모드를 해제한 후에 상기 제 1 영역에 상기 트레이닝 코드를 로드하는 전자 장치.

청구항 4

제 3 항에 있어서,

상기 제 1 영역은 상기 메모리 장치의 제 1 채널에 대응하는 메모리 단위에 포함되고, 상기 제 2 영역은 제 2 채널에 대응하는 메모리 단위에 포함되는 전자 장치.

청구항 5

제 1 항에 있어서,

상기 시스템 온 칩은 상기 메모리 장치에 대한 채널 인터리빙을 위한 인터리빙 단위의 크기를 조정한 후에 상기 제 1 영역에 상기 트레이닝 코드를 로드하는 전자 장치.

청구항 6

제 5 항에 있어서,

상기 채널 인터리빙 단위의 크기는 상기 트레이닝 코드의 용량을 참조하여 조정되는 전자 장치.

청구항 7

제 6 항에 있어서,

상기 채널 인터리빙 단위의 크기는 상기 트레이닝 코드의 용량보다 큰 사이즈로 조정되는 전자 장치.

청구항 8

제 1 항에 있어서,

상기 시스템 온 칩은, 상기 소프트웨어 트레이닝이 수행되는 동작 모드에서 생성되는 채널 인터리빙 어드레스를 상기 트레이닝 코드가 상기 메모리 장치의 제 1 영역에 연속적으로 저장되도록 맵핑된 변환 어드레스를 생성하는 아비터를 포함하는 전자 장치.

청구항 9

제 8 항에 있어서,

상기 아비터는 상기 채널 인터리빙 어드레스의 특정 비트를 마스킹하여 상기 변환 어드레스를 생성하는 전자 장치.

청구항 10

제 1 항에 있어서,

상기 시스템 온 칩은 상기 제 2 영역에 상기 트레이닝 코드를 로드하고, 로드된 상기 트레이닝 코드를 실행하여 상기 제 1 영역에 대한 소프트웨어 트레이닝을 추가로 수행하는 전자 장치.

청구항 11

채널 단위로 관리되는 메모리 장치를 워킹 메모리로 구동하는 전자 장치의 소프트웨어 트레이닝 방법에 있어서:

상기 메모리 장치의 제 1 채널에 대응하는 제 1 영역에 제 2 영역의 동작 특성을 최적화하기 위한 트레이닝 코드를, 상기 메모리 장치의 제 2 채널에 대응하는 상기 제 2 영역에 트레이닝 데이터를 로드하는 단계;

상기 제 1 영역에 로드된 트레이닝 코드를 실행에 따라 상기 제 2 영역에 저장된 트레이닝 데이터에 접근하는 단계;

상기 제 2 영역으로부터 독출된 상기 트레이닝 데이터를 평가하여 상기 제 2 채널의 접근 파라미터를 결정하는 단계; 그리고

상기 접근 파라미터를 사용하여 상기 제 2 채널의 동작 조건을 설정하는 단계를 포함하는 메모리 장치의 트레이닝 방법.

청구항 12

제 11 항에 있어서,

상기 메모리 장치에 대한 채널 인터리빙 모드를 해제하는 단계를 더 포함하는 트레이닝 방법.

청구항 13

제 11 항에 있어서,

상기 메모리 장치에 대한 채널 인터리빙 단위의 크기를 조정하는 단계를 더 포함하는 트레이닝 방법.

청구항 14

제 13 항에 있어서,

상기 채널 인터리빙의 단위의 크기는 상기 트레이닝 코드의 크기보다 큰 값으로 설정되는 트레이닝 방법.

청구항 15

제 11 항에 있어서,

상기 트레이닝 코드가 상기 메모리 장치의 제 1 영역에 연속적으로 저장되도록 상기 메모리 장치에 대한 채널 인터리빙 어드레스를 리맵핑하는 단계를 더 포함하는 트레이닝 방법.

청구항 16

제 11 항에 있어서,

상기 제 1 영역에 상기 트레이닝 데이터를, 상기 제 2 영역에 상기 트레이닝 코드를 로드하는 단계;

상기 제 2 영역에 로드된 트레이닝 코드를 실행하는 단계; 그리고

상기 트레이닝 코드의 실행에 따라 상기 트레이닝 데이터에 접근하여 상기 제 1 채널의 접근 파라미터를 결정하

는 단계를 더 포함하는 트레이닝 방법.

청구항 17

제 11 항에 있어서,

상기 접근 파라미터는 데이터 신호(DQ)와 스트로브 신호(DQS) 간의 타이밍 또는 신호 레벨의 기준값을 포함하는 트레이닝 방법.

청구항 18

워킹 메모리로 구동되는 메모리 장치; 그리고

상기 메모리 장치에 데이터를 로드하고 패치하며, 복수의 채널들로 연결되는 호스트를 포함하되,

상기 호스트는 상기 복수의 채널들 중 제 1 채널에 대응하는 메모리 영역에 제 2 채널에 대응하는 최적의 접근 파라미터를 결정하기 위한 트레이닝 코드를 로드하고, 로드된 상기 트레이닝 코드를 실행하여 상기 제 2 채널에 대한 소프트웨어 트레이닝을 수행하여 상기 제 2 채널에 대응하는 상기 최적의 접근 파라미터를 결정하는 컴퓨팅 시스템.

청구항 19

제 18 항에 있어서,

상기 호스트는 상기 메모리 장치에 대한 채널 인터리빙 모드를 해제한 후에 상기 제 1 채널에 대응하는 메모리 영역에 상기 트레이닝 코드를 로드하여 상기 제 2 채널에 대한 소프트웨어 트레이닝을 수행하는 컴퓨팅 시스템.

청구항 20

제 18 항에 있어서,

상기 호스트는 상기 메모리 장치에 대한 채널 인터리빙 단위의 크기를 조정한 후에 상기 제 1 채널에 대응하는 메모리 영역에 상기 트레이닝 코드를 로드하는 컴퓨팅 시스템.

발명의 설명

기 술 분 야

[0001] 본 발명은 반도체 장치에 관한 것으로, 좀 더 구체적으로는 메모리 채널의 소프트웨어 트레이닝을 수행하는 전자 장치 및 그것의 메모리 채널 트레이닝 방법에 관한 것이다.

배 경 기 술

[0002] 스마트폰, 테이블릿 PC, 디지털 카메라, MP3 플레이어, PDA 그리고 웨어러블 기기 등과 같은 모바일 장치의 이용이 폭발적으로 증가하고 있다. 이들 모바일 장치의 핵심 구동 프로세서로서 소위 응용 프로세서(Application Processor: 이하, AP)가 사용되고 있다. 더불어, 다양한 기능 블록들(Intellectual Property: 이하, IP)을 포함하는 응용 프로세서(AP)의 메인 메모리 또는 워킹 메모리(Working memory)로 디램과 같은 휘발성 메모리 장치가 사용된다. 고성능 및 고용량의 메모리 요구에 따라, 워킹 메모리의 동작 주파수와 저장 용량은 급격히 증가하는 추세이다.

[0003] 동작 속도의 증가에 따라 응용 프로세서와 워킹 메모리 예컨대, 디램(DRAM)이 교환하는 데이터의 신뢰성을 보장하는 것이 점점 어려워지고 있다. 데이터 신뢰성의 보장을 위해서는 인터페이스를 통해 전달되는 데이터 또는 신호의 클록 타이밍이나 레퍼런스 전압의 최적화가 필수적이다. 클록 타이밍의 최적화는 클록 신호(또는 스트로브 신호)와 데이터 간의 최적 동기화를 의미한다. 레퍼런스 전압은 데이터의 논리값을 결정하기 위한 기준을 제공한다.

[0004] 일반적으로 이러한 데이터의 신뢰성을 위해 디램에 대한 트레이닝(Training)이 수행된다. 이러한 트레이닝을 위한 코드의 실행을 통해서 디램 채널의 최적화가 수행될 수 있다. 보다 많은 기능과 높은 신뢰도를 제공하기 위한 트레이닝 코드의 사이즈는 증가하고 있는 추세이다. 하지만, 증가하는 트레이닝 코드를 로드(Load)하기 위한 에스램(SRAM)의 고용량화는 과도한 비용 증가를 초래한다. 또한, 충분한 리소스를 갖는 디램에 트레이닝 코드를

로드(Load)하는 시도는 채널 인터리빙(Channel interleaving)의 장벽에 의해서 차단되고 있는 실정이다.

발명의 내용

해결하려는 과제

[0005] 본 발명의 목적은 메모리의 채널 트레이닝을 효율적으로 수행하는 전자 장치 및 그것의 메모리 트레이닝 방법을 제공하는 데 있다.

과제의 해결 수단

[0006] 상기 목적을 달성하기 위한 본 발명의 실시 예에 따른 전자 장치는, 데이터를 저장하는 메모리 장치, 그리고 상기 메모리 장치를 워킹 메모리로 사용하는 시스템 온 칩을 포함하되, 상기 시스템 온 칩은 상기 메모리 장치의 제 1 영역에 트레이닝 코드를 로드하고, 로드된 상기 트레이닝 코드를 실행하여 상기 메모리 장치의 제 2 영역에 대한 소프트웨어 트레이닝을 수행한다.

[0007] 상기 목적을 달성하기 위한 본 발명의 실시 예에 따른 채널 단위로 관리되는 메모리 장치를 워킹 메모리로 구동하는 전자 장치의 소프트웨어 트레이닝 방법은, 상기 메모리 장치의 제 1 채널에 대응하는 제 1 영역에 트레이닝 코드를, 상기 메모리 장치의 제 2 채널에 대응하는 제 2 영역에 트레이닝 데이터를 로드하는 단계, 상기 제 1 영역에 로드된 트레이닝 코드를 실행에 따라 상기 제 2 영역에 저장된 트레이닝 데이터에 접근하는 단계, 상기 제 2 영역으로부터 독출된 상기 트레이닝 데이터를 평가하여 상기 제 2 채널의 접근 파라미터를 결정하는 단계, 그리고 상기 접근 파라미터를 사용하여 상기 제 2 채널의 동작 조건을 설정하는 단계를 포함한다.

[0008] 상기 목적을 달성하기 위한 본 발명의 실시 예에 따른 컴퓨팅 시스템은, 워킹 메모리로 구동되는 메모리 장치, 그리고 상기 메모리 장치에 데이터를 로드하고 패치하며, 복수의 채널들로 연결되는 호스트를 포함하되, 상기 호스트는 상기 복수의 채널들 중 제 1 채널에 대응하는 메모리 영역에 트레이닝 코드를 로드하고, 로드된 상기 트레이닝 코드를 실행하여 제 2 채널에 대한 소프트웨어 트레이닝을 수행한다.

발명의 효과

[0009] 이상과 같은 본 발명의 실시 예에 따르면, 메모리의 채널 트레이닝을 위한 트레이닝 코드를 디램에 로드하여 실행할 수 있다. 따라서, 트레이닝 코드의 사이즈 제약없이 보다 높은 신뢰도의 메모리 채널 트레이닝이 가능하다. 더불어, 고속으로 데이터 전송이 가능한 디램으로부터 패치되는 트레이닝 코드에 의해서 메모리 채널의 트레이닝에 소요되는 시간을 획기적으로 줄일 수 있다.

도면의 간단한 설명

[0010] 도 1은 본 발명의 실시 예에 따른 전자 장치(10)를 개략적으로 보여주는 블록도이다.

도 2는 본 발명의 호스트의 구조를 예시적으로 보여주는 블록도이다.

도 3은 본 발명의 실시 예에 따른 워킹 메모리(200)의 구조를 예시적으로 보여주는 블록도이다.

도 4a 및 도 4b는 메모리의 소프트웨어 트레이닝의 예를 보여주는 파형도이다.

도 5는 본 발명의 트레이닝 코드(TC)가 로드되는 워킹 메모리의 트레이닝 방법을 간략히 보여주는 순서도이다.

도 6은 도 5의 소프트웨어 트레이닝 절차(S140 단계)를 세부적으로 보여주는 순서도이다.

도 7은 도 6에서 설명된 메모리 컨트롤러(120)의 채널 인터리빙을 해제하는 방법을 보여주는 블록도이다.

도 8은 채널 인터리빙이 해제된 워킹 메모리(200)에 트레이닝 코드(TC)와 트레이닝 데이터를 로드하는 방법을 간략히 보여주는 도면이다.

도 9는 본 발명의 다른 실시 예에 따른 소프트웨어 트레이닝 방법을 보여주는 순서도이다.

도 10은 도 9에서 설명된 실시 예에 따른 트레이닝 코드의 로드 방법을 보여주는 블록도이다.

도 11은 본 발명의 또 다른 실시 예에 따른 소프트웨어 트레이닝을 수행하는 전자 장치(10')를 보여주는 블록도이다.

도 12는 도 11의 호스트(100')의 예시적인 구성을 보여주는 블록도이다.

도 13은 아비터(190)의 동작을 구체적으로 보여주는 블록도이다.

도 14는 아비터(190)를 포함하는 호스트(100)의 소프트웨어 트레이닝 절차를 보여주는 순서도이다.

도 15는 본 발명의 실시 예에 따라 소프트웨어 트레이닝을 통해 생성된 접근 파라미터를 이용하는 방법을 예시적으로 보여주는 순서도이다.

도 16은 본 발명의 실시 예에 따라 소프트웨어 트레이닝을 통해 생성된 접근 파라미터를 이용하는 다른 방법을 보여주는 순서도이다.

발명을 실시하기 위한 구체적인 내용

[0011] 앞의 일반적인 설명 및 다음의 상세한 설명 모두 예시적이라는 것이 이해되어야 하며, 청구된 발명의 부가적인 설명이 제공되는 것으로 여겨져야 한다. 참조 부호들이 본 발명의 바람직한 실시 예들에 상세히 표시되어 있으며, 그것의 예들이 참조 도면들에 표시되어 있다. 가능한 어떤 경우에도, 동일한 참조 번호들이 동일한 또는 유사한 부분을 참조하기 위해서 설명 및 도면들에 사용된다.

[0012] 이하에서는, 본 발명의 특징 및 기능을 설명하기 위한 워킹 메모리로서 디램(DRAM)을 한 예로서 기술될 것이다. 하지만, 이 기술 분야에 정통한 사람은 여기에 기재된 내용에 따라 본 발명의 다른 이점들 및 성능을 쉽게 이해할 수 있을 것이다. 예를 들면, 워킹 메모리 장치로서 PRAM, MRAM, ReRAM, FRAM, NOR 플래시 메모리 등에도 본 발명의 기술적 특징이 적용될 수 있을 것이다. 여기서, 워킹 메모리는 전자 장치에서 처리되는 명령어 또는 데이터를 저장하는 메모리일 수 있다. 더불어, 본 발명에서 사용되는 용어 '트레이닝(Training)'은 최적의 신뢰성을 제공하기 위한 메모리 채널의 레이턴시 또는 신호 레벨을 탐색하는 동작을 의미한다.

[0013] 본 발명은 다른 실시 예들을 통해 구현되거나 적용될 수 있을 것이다. 게다가, 상세한 설명은 본 발명의 범위, 기술적 사상 그리고 다른 목적으로부터 상당히 벗어나지 않고 관점 및 응용에 따라 수정되거나 변경될 수 있다. 이하, 본 발명에 따른 실시 예를 첨부된 도면을 참조하여 상세히 설명한다.

[0014] 도 1은 본 발명의 실시 예에 따른 전자 장치(10)를 개략적으로 보여주는 블록도이다. 도 1을 참조하면, 전자 장치(10)는 호스트(100)와 워킹 메모리(200)를 포함한다. 호스트(100)와 워킹 메모리(200) 사이에는 신호 및 데이터 교환을 위한 복수의 채널들(310, 320)이 제공된다.

[0015] 호스트(100)는 사용자의 요청에 따라 다양한 애플리케이션(Application)을 실행할 수 있다. 호스트(100)는 애플리케이션을 수행하기 위해, 워킹 메모리(200)에 애플리케이션을 로드(Load)하고 실행한다. 호스트(100)는 운영 체제(OS)를 구동할 수 있고, 운영 체제(OS) 상에서 다양한 애플리케이션들을 실행할 수 있다. 이러한 동작을 위해 호스트(100)는 워킹 메모리(200)에 데이터를 쓰거나 또는 워킹 메모리(200)에 저장된 데이터를 읽을 수 있다.

[0016] 호스트(100)는 워킹 메모리(200)를 제어하기 위한 메모리 컨트롤러(120)를 포함할 수 있다. 호스트(100)는 워킹 메모리(200)에 저장된 데이터를 읽기 위해 메모리 컨트롤러(120)에 읽기 요청을 전달할 수 있다. 메모리 컨트롤러(120)는 호스트(100)가 요청한 데이터를 읽기 위해서 워킹 메모리(200)에 접근할 것이다. 호스트(100)는 적어도 하나의 채널(310, 320)을 통하여 워킹 메모리(200)에 연결될 수 있다. 메모리 컨트롤러(120)는 접근 효율을 높이기 위하여 복수의 채널(310, 320)을 이용하여 채널 인터리빙(Channel Interleaving) 방식에 따라 워킹 메모리(200)를 제어할 수 있다.

[0017] 호스트(100)는 부팅이나 특정 상황에서 워킹 메모리(200)에 대한 트레이닝(Training)을 수행할 수 있다. 트레이닝을 통해서 호스트(100)는 워킹 메모리(200)와의 데이터나 신호 교환의 신뢰성을 높일 수 있다. 예를 들면, 호스트(100)는 트레이닝 데이터(Training Data: TD)를 다양한 조건에서 워킹 메모리(200)에 기입하거나 독출하여 최적의 클록 타이밍이나 레퍼런스 레벨을 결정할 수 있다.

[0018] 특히, 본 발명의 호스트(100)는 트레이닝을 수행하기 위한 트레이닝 코드(Training Code: 이하, TC)를 워킹 메모리(200)에 로드할 수 있다. 트레이닝 코드(TC)는 실질적으로 트레이닝을 수행하기 위한 실행 코드에 해당한다. 만일, 트레이닝 코드에 오류가 발생하면, 제반 트레이닝 동작은 실패(Failure)하게 될 것이다. 따라서, 데이터나 신호의 신뢰성을 확보하기 위한 트레이닝 코드(TC)의 안정성 확보가 필요하다. 본 발명의 호스트(100)는 워킹 메모리(100)의 제 1 영역에 트레이닝 코드(TC)를 로드하고, 워킹 메모리(200)의 제 2 영역에 트레이닝 데이터(TD)를 로드한다. 그리고 호스트(100)는 제 1 영역으로부터 트레이닝 코드(TC)를 패치하여

실행하고, 제 2 영역에 대한 트레이닝을 수행하여 최적 신뢰도를 갖는 파라미터를 결정할 수 있다. 물론, 제 2 영역에 트레이닝 코드(TC)를 로드하고, 제 1 영역에 대한 트레이닝을 수행할 수 있다. 제 1 영역과 제 2 영역의 메모리 영역의 분할이나 확보 방법은 후술하는 도면들을 통해서 자세히 설명하기로 한다.

[0019] 메모리 컨트롤러(120)는 호스트(100)의 다양한 마스터 기능 블록(Master IP)의 요청에 따라 워킹 메모리(200)에 접근한다. 메모리 컨트롤러(120)는 예를 들면, 워킹 메모리(200)로의 접근 요청에 따라 데이터(DQ)를 스트로브 신호(DQS)에 동기하여 전달할 수 있다. 또는, 스트로브 신호(DQS)에 동기되어 워킹 메모리(200)로부터 출력되는 데이터(DQ)를 수신할 수 있다. 메모리 컨트롤러(120)는 예를 들면 채널 인터리빙(Channel Interleaving) 방식으로 워킹 메모리(200)에 접근할 수 있다. 채널 인터리빙 방식에 따른 워킹 메모리(200)로의 접근에 따라, 데이터 교환은 채널 단위로 독립적으로 수행될 수 있다. 이러한 채널 인터리빙에 대한 설명은 후술하는 도 3에서 상세히 설명될 것이다.

[0020] 워킹 메모리(200)는 전자 장치(10)의 메인 메모리로서 구동된다. 워킹 메모리(200)에는 전자 장치(10)의 부팅시에 운영 체제(OS)나 기본 응용 프로그램들이 로드될 것이다. 예를 들면, 호스트(100)의 부팅시에 스토리지 장치(미도시)에 저장된 OS 이미지가 부팅 시퀀스에 의거하여 워킹 메모리(200)에 로드된다. 운영 체제에 의해서 호스트(100)의 제반 입출력 동작들이 지원될 수 있다. 마찬가지로, 사용자의 의하여 선택되거나 기본적인 서비스 제공을 위해서 응용 프로그램들이 워킹 메모리(200)에 로드될 수 있다. 워킹 메모리(200)는 이뿐 아니라, 카메라와 같은 이미지 센서로부터 제공되는 영상 데이터를 저장하는 버퍼 메모리로 사용될 수도 있을 것이다.

[0021] 워킹 메모리(200)는 바이트 단위 액세스(Byte Access)가 가능한 다이나믹 랜덤 액세스 메모리(DRAM)일 수 있다. 워킹 메모리(200)는 덮어쓰기가 가능한 불휘발성 메모리 장치로 제공될 수도 있다. 예를 들면, 워킹 메모리(200)는 PRAM, MRAM, ReRAM, FRAM, NOR 플래시와 같은 불휘발성 램으로 구성될 수 있다. 워킹 메모리(200)에는 전자 장치(10)의 구동시에 운영 체제(OS), 구동중인 응용 프로그램(Application Program), 업데이트되는 데이터 등이 저장된다. 워킹 메모리(200)는 멀티-칩이 적층되는 멀티칩 패키지 또는 모듈 형태로 제공될 수 있을 것이다. 하지만, 워킹 메모리(200)의 구성 방식은 여기의 게시에 국한되지 않는다.

[0022] 채널(310, 320)은 호스트(100)와 워킹 메모리(200) 상이에 데이터나 신호의 전송 경로를 제공한다. 채널(310, 320)은 각각 독립적으로 워킹 메모리(200)와 호스트(100)가 데이터를 전송할 수 있는 데이터 경로에 해당한다. 예를 들면, 제 1 채널(310)을 통해서 데이터(DQ) 및 스트로브 신호(DQS)가 전송되는 동안에, 제 2 채널(320)을 통해서 또 다른 데이터(DQ) 및 스트로브 신호(DQS)가 전송될 수 있다. 채널(310, 320)의 구동 및 운영은 메모리 컨트롤러(120)에 의해서 제어될 것이다.

[0023] 이상에서는 본 발명의 트레이닝 코드(TC)를 로드하는 워킹 메모리(200) 및 그것을 사용하여 워킹 메모리(200)를 트레이닝하는 호스트(100)의 구성에 대해서 간략히 설명되었다. 이러한 구조를 통해서 기존의 호스트(100) 내의 에스램(SRAM) 내에 트레이닝 코드(TC)를 로드하는 방식보다 로드 및 패치 속도가 향상될 수 있다. 더불어, 트레이닝 코드(TC)가 로드되는 채널 또는 메모리 영역은 트레이닝 대상에서 제외함으로써 트레이닝 코드(TC)에 대한 안정성을 확보할 수 있다.

[0024] 도 2는 본 발명의 호스트의 구조를 예시적으로 보여주는 블록도이다. 도 2를 참조하면, 호스트(100)는 시스템 온 칩(SoC)과 스토리지 장치(180)를 포함할 수 있다. 도시되지는 않았지만, 호스트(100)는 액정 표시 장치나 터치 패널과 같은 장치를 더 포함할 수 있을 것이다.

[0025] 시스템 온 칩(SoC)은 CPU(110), 메모리 컨트롤러(120), GPU(130), 사용자 인터페이스 컨트롤러(140), 스토리지 인터페이스(150), 가속기(160) 등을 포함할 수 있다. 시스템 온 칩(SoC)의 구성 요소들은 도시된 구성 요소들에 국한되지 않음은 잘 이해될 것이다. 예를 들면, 시스템 온 칩(SoC)은 영상 데이터를 처리하기 위한 하드웨어 코덱, 보안 블록 등을 더 포함할 수 있다.

[0026] CPU(110)는 호스트(100)에서 수행될 소프트웨어(응용 프로그램, 운영 체제, 장치 드라이버들)를 실행한다. CPU(110)는 워킹 메모리(200)에 로드되는 운영 체제(OS)를 실행할 것이다. CPU(110)는 운영 체제(OS) 기반에서 구동될 다양한 응용 프로그램들(Application Program)을 실행할 것이다. 특히, CPU(110)는 워킹 메모리(200)에 로드되는 트레이닝 코드(TC)를 패치하여 실행할 것이다. CPU(110)는 트레이닝 코드(TC)의 실행에 따라 요청되는 워킹 메모리(200)의 트레이닝 동작을 수행하도록 메모리 컨트롤러(120)를 제어할 것이다. CPU(110)는 동종 멀티-코어 프로세서(Homogeneous Multi-Core Processor) 또는 이종 멀티-코어 프로세서(Heterogeneous Multi-Core Processor)로 제공될 수 있다.

[0027] 메모리 컨트롤러(120)는 워킹 메모리(200)와 시스템 온 칩(SoC) 사이에서 인터페이싱을 제공한다. 메모리 컨트

롤러(120)는 CPU(110)나 다른 기능 블록(IP)의 요청에 따라 워킹 메모리(200)를 액세스할 것이다. 예를 들면, 메모리 컨트롤러(120)는 CPU(110)의 쓰기 요청에 따라 데이터를 워킹 메모리(200)에 기입할 수 있다. 또는, 메모리 컨트롤러(120)는 워킹 메모리(200)로부터 데이터를 읽어내어 CPU(100)나 스토리지 인터페이스(150)에 전달할 수 있다. 메모리 컨트롤러(120)는 트레이닝 동작시, CPU(110)의 요청에 따라 채널 인터리빙을 해제하거나 채널 인터리빙의 단위를 조정할 수 있다.

[0028] GPU(130)는 CPU(110)의 요청에 따라 다양한 그래픽 연산을 수행한다. 즉, GPU(130)는 처리 요청된 데이터를 디스플레이(미도시)에 적합한 데이터로 변환할 수 있다. GPU(130)는 유사한 연산을 반복해서 처리하는 병렬 처리에 유리한 연산 구조를 가진다. 따라서, 최근에는 GPU(130)는 그래픽 연산뿐만 아니라 고속의 병렬 처리를 요구하는 다양한 연산에 사용될 수 있는 구조로 개발되고 있다.

[0029] 사용자 인터페이스 컨트롤러(140)는 사용자 인터페이스 장치들(예를 들면, 키보드나 터치 패널, 또는 디스플레이)로부터의 사용자 입력 및 출력을 제어한다. 예를 들면, 사용자 인터페이스 컨트롤러(140)는 CPU(110)의 제어에 따라 디스플레이(미도시)에 데이터를 입력하기 위한 키보드 화면 등을 표시할 수 있다. 또는, 사용자 인터페이스 컨트롤러(140)는 사용자가 요청한 데이터를 표시하도록 디스플레이를 제어할 수 있다. 사용자 인터페이스 컨트롤러(140)는 키보드, 마우스, 터치 패널과 같은 사용자 입력 수단으로부터의 제공되는 데이터를 사용자 입력 데이터로 디코딩할 수 있다.

[0030] 스토리지 인터페이스(150)는 CPU(110)의 요청에 따라 스토리지 장치(180)를 액세스한다. 즉, 스토리지 인터페이스(150)는 시스템 온 칩(SoC)과 스토리지 장치(180) 사이의 인터페이스를 제공한다. 예를 들면, CPU(110)에 의해서 처리된 데이터가 스토리지 인터페이스(150)를 통해 스토리지 장치(180)에 저장된다. 더불어, 스토리지 장치(180)에 저장된 데이터는 스토리지 인터페이스(150)를 통해 CPU(110)에 제공될 수 있다.

[0031] 가속기(160)는 멀티미디어 또는 멀티미디어 데이터의 처리 속도를 향상하기 위한 별도의 기능 블록으로 제공된다. 예를 들면, 가속기(180)는 텍스트(Text), 오디오(Audio), 정지 영상들(Still images), 애니메이션(Animation), 비디오(Video), 2차원 데이터, 또는 3차원 데이터의 처리 성능을 향상시키기 위한 기능 블록(IP)으로 제공될 수 있다.

[0032] 시스템 인터커넥터(170)는 시스템 온 칩(SoC)의 내부에서 온칩 네트워크를 제공하기 위한 시스템 버스(System Bus)이다. 시스템 인터커넥터(170)는 예를 들면, 데이터 버스(Data bus), 어드레스 버스(Address bus) 및 컨트롤 버스(Control bus)를 포함할 것이다. 데이터 버스(Data bus)는 데이터가 이동하는 경로이다. 데이터 버스(Data bus)는 주로, 워킹 메모리(200)이나 스토리지 장치(180)로의 메모리 접근 경로를 제공할 것이다. 어드레스 버스(Address bus)는 기능 블록들(IPs) 간의 어드레스 교환 경로를 제공한다. 컨트롤 버스(Control bus)는 기능 블록들(IPs) 간의 제어 신호를 전달하는 경로를 제공한다. 하지만, 시스템 인터커넥터(170)의 구성은 상술한 설명에만 국한되지 않으며, 효율적인 관리를 위한 중재 수단들을 더 포함할 수 있다.

[0033] 스토리지 장치(180)는 호스트(100)의 저장 매체(Storage Medium)로서 제공된다. 스토리지 장치(180)는 응용 프로그램들, 운영 체제 이미지(OS Image) 및 각종 데이터를 저장할 수 있다. 특히, 스토리지 장치(180)의 특정 영역에는 워킹 메모리(200)를 트레이닝하기 위한 트레이닝 코드(TC)가 저장될 수 있다. 하지만, 트레이닝 코드(TC)는 스토리지 장치(180)가 아닌 다른 비휘발성 메모리에 저장될 수도 있음은 잘 이해될 것이다. 스토리지 장치(180)는 메모리 카드(MMC, eMMC, SD, MicroSD 등)로 제공될 수도 있다. 스토리지 장치(180)는 대용량의 저장 능력을 가지는 낸드 플래시 메모리(NAND-type Flash memory)를 포함할 수 있다. 또는, 스토리지 장치(180)는 PRAM, MRAM, ReRAM, FRAM 등의 차세대 불휘발성 메모리나 NOR 플래시 메모리를 포함할 수도 있다. 본 발명의 다른 실시 예에서는 스토리지 장치(180)는 시스템 온 칩(SoC)의 내부에 구비되는 내장 메모리일 수 있다.

[0034] 이상의 설명에 따르면, 워킹 메모리(200)의 트레이닝을 위해서 시스템 온 칩(SoC)은 트레이닝 코드(TC)를 워킹 메모리(200)에 로드할 수 있다. 그리고 트레이닝 코드(TC)의 워킹 메모리(200)로의 로드를 위해, 메모리 컨트롤러(120)는 워킹 메모리(200)의 채널 인터리빙 모드를 해제(또는 비활성화)할 수 있다. 또는, 트레이닝 코드(TC)의 워킹 메모리(200)로의 로드를 위해, 메모리 컨트롤러(120)는 워킹 메모리(200)의 채널 인터리빙의 단위 사이즈를 조정할 수도 있다.

[0035] 도 3은 본 발명의 실시 예에 따른 워킹 메모리(200)의 구조를 예시적으로 보여주는 블록도이다. 도 3을 참조하면, 워킹 메모리(200)는 복수의 채널들(CH0, CH1, …, CHn-1) 단위로 관리되는 복수의 디램 다이들(Die_0, Die_1, Die_2, …, Die_k-1)을 포함할 수 있다.

[0036] 워킹 메모리(200)는 메모리 컨트롤러(120)에 의해서 채널 인터리빙 방식으로 관리될 수 있다. 예를 들면, 메모

리 컨트롤러(120)가 데이터를 워킹 메모리(200)에 기입하는 경우, 쓰기 요청된 데이터는 인터리빙 단위(IU)로 파티셔닝된다. 인터리빙 단위(IU)가 128 바이트(Byte)라 가정하면, 쓰기 요청된 데이터는 128 바이트(Byte) 단위로 분할되고, 미리 정의된 순서에 따라 각각의 채널에 대응하는 메모리 다이(Die)에 기입된다. 예를 들면, 채널 인터리빙의 시퀀스가 'CH0→CH1→CH2→CH3→…→CHn-1'이라 가정하면, 각각의 채널에 128 바이트(Byte)의 분할된 인터리빙 단위(IU)의 데이터가 파이프라인 방식으로 기입될 수 있다. 읽기 동작에서도 동일한 방식으로 채널들이 선택되어 데이터가 출력될 수 있다.

[0037] 2개의 다이(Die_0, Die_1)가 포함되는 채널(CH0)이 선택되면, 메모리 컨트롤러(120)는 2개의 다이(Die_0, Die_1) 중 어느 하나를 선택하기 위한 칩 선택 신호(CS0, CS1)를 활성화한다. 그리고 스트로브 신호(DQS)와 데이터(DQ)를 전달할 것이다. 이때, 스트로브 신호(DQS)와 데이터(DQ)를 전달하는 신호 라인은 2개의 다이(Die_0, Die_1)에 의해서 공유되며, 칩 선택 신호(CS0, CS1)에 의해서 2개의 다이(Die_0, Die_1) 중 어느 하나가 선택된다.

[0038] 상술한 채널 인터이빙 구조로 관리되는 워킹 메모리(200)에 트레이닝 코드(TC)를 로드하는 것은 트레이닝 코드(TC)의 안정성을 저하시킬 수 있다. 하지만, 본 발명의 실시 예에 따라, 트레이닝 코드(TC)가 로드되는 시점에서 채널 인터리빙은 해제될 수 있다. 다른 실시 예에서, 채널 인터리빙 단위(IU)의 사이즈가 트레이닝 코드(TC)를 수용할 정도의 사이즈로 조정될 수 있다. 그리고 소프트웨어 트레이닝(S/W Training)은 트레이닝 코드가 저장되지 않은 영역에 대해서만 수행될 것이다. 이러한 소프트웨어 트레이닝 방법에 따라 트레이닝 코드의 안정성을 높이고, 메모리의 트레이닝 신뢰성도 보장될 수 있다.

[0039] 도 4a 및 도 4b는 메모리의 소프트웨어 트레이닝의 예를 보여주는 파형도이다. 도 4a는 워킹 메모리(200)의 읽기 트레이닝 방법을, 도 4b는 워킹 메모리(200)의 쓰기 트레이닝을 보여주는 파형도 들이다. 좀더 구체적으로, 도 4a는 읽기 동작시의 스트로브 신호(DQS)의 트레이닝을 보여주는 파형도이고, 도 4b는 쓰기 동작시 스트로브 신호(DQS)의 트레이닝을 보여주는 파형도이다.

[0040] 도 4a를 참조하면, 워킹 메모리(200)의 제 1 영역에 로드된 트레이닝 코드(TC)의 실행에 따라 제 2 영역에 대한 워킹 메모리(200)의 소프트웨어 트레이닝이 실시된다. 여기서, 제 1 영역은 예를 들면 채널(CH0)에 대응하는 메모리들(210)일 수 있다. 그리고 제 2 영역은 트레이닝 코드(TC)가 로드된 채널(CH0) 영역(또는 제 1 영역)을 제외한 메모리들(예를 들면, 220, 230)일 수 있다.

[0041] 트레이닝 코드(TC)의 실행에 따라, 제 2 영역의 메모리들에 트레이닝 데이터(TD)가 기입되고 독출 동작이 도시된 절차에 따라 진행된다. 트레이닝 코드(TC)의 실행에 의해, 메모리 컨트롤러(120)는 최적 타이밍을 갖는 스트로브 신호(tDQS)를 찾기 위한 트레이닝 동작을 수행할 것이다. 즉, 메모리 컨트롤러(120)는 다양한 지연값(t0, t1, t2, t3)을 갖는 스트로브 신호들(tDQS0, tDQS1, tDQS2, tDQS3)을 사용하여 트레이닝 데이터(D0, D1, D2, D3)를 출력하도록 제 2 영역에 대응하는 메모리 다이들을 제어할 것이다. 그리고 트레이닝 코드(TC)에 의해서 가장 높은 신뢰도를 제공하는 스트로브 신호(DQS)의 타이밍이 선택될 수 있다.

[0042] 이상에서는 읽기 동작시 스트로브 신호(DQS)의 지연을 다양하게 제공하는 방식으로 트레이닝이 수행되는 예가 설명되었으나 이것은 설명의 편의를 위한 예시에 불과함은 잘 이해될 것이다. 읽기 트레이닝을 위해 워킹 메모리(200)의 클록(CK) 신호나 데이터(DQ)의 조정도 가능하다.

[0043] 도 4b를 참조하면, 워킹 메모리(200)의 제 1 영역에 로드된 트레이닝 코드(TC)의 실행에 따라 제 2 영역에 대한 워킹 메모리(200)의 쓰기 트레이닝이 실시된다. 여기서, 제 1 영역은 예를 들면 채널(CH0)에 대응하는 메모리들(210)일 수 있다. 그리고 제 2 영역은 트레이닝 코드(TC)가 로드된 채널(CH0) 영역(또는 제 1 영역)을 제외한 메모리들(예를 들면, 220, 230)일 수 있다.

[0044] 트레이닝 코드(TC)의 실행에 따라, 제 2 영역의 메모리들에 다양한 지연의 스트로브 신호들(tDQS0, tDQS1, tDQS2, tDQS3)을 사용하여 트레이닝 데이터(D0, D1, D2, D3)가 기입되고 독출될 것이다. 독출된 데이터의 신뢰도나 에러율을 검출하여 최적의 스트로브 신호가 선택될 것이다.

[0045] 이상에서는 소프트웨어 트레이닝의 예로 데이터(DQ) 및 스트로브 신호(DQS)의 최적 타이밍을 검출하기 위한 동작이 예시적으로 설명되었다. 하지만, 워킹 메모리(200)에 대한 소프트웨어 트레이닝은 이뿐 아니라 다양한 신호나 설정에 대해서 수행될 수 있다.

[0046] 도 5는 본 발명의 트레이닝 코드(TC)가 로드되는 워킹 메모리의 트레이닝 방법을 간략히 보여주는 순서도이다. 도 5를 참조하면, 워킹 메모리(200)의 제 1 영역에 트레이닝 코드(TC)가 로드되고, 트레이닝 코드(TC)의 실행에

의해서 워킹 메모리의 제 2 영역에 대한 소프트웨어 트레이닝이 수행된다.

[0047] S110 단계에서, 전자 장치(10)에 대한 부팅이 수행된다. 예를 들면, 전자 장치(10)의 전원 스위치가 턴온되거나, 장치 리셋에 의한 초기화의 수행시에 부팅 동작이 수행될 수 있을 것이다.

[0048] S120 단계에서, 가장 기본적으로 전자 장치(10)의 제반 입출력 인터페이스(I/O Interface)에 대한 트레이닝이 수행될 수 있다. 워킹 메모리(200)에 대한 메모리 트레이닝의 이전에, 최소한의 신뢰성을 보장하기 위한 트레이닝 동작들이 이 단계에서 수행될 수 있을 것이다.

[0049] S130 단계에서, 워킹 메모리(200)에 대한 하드웨어 트레이닝(H/W training)이 수행된다. 하드웨어 트레이닝은 예를 들면, 워킹 메모리(200)에 미리 정의된 초기값 파라미터로 메모리 컨트롤러(120)를 설정하는 동작일 수 있다. 하드웨어 트레이닝을 통해서 워킹 메모리(200)로의 접근을 위한 최소한의 신뢰성은 획득될 수 있을 것이다. 따라서, 하드웨어 트레이닝(H/W training)은 다른 표현으로 코어스 트레이닝(Coarse training)이라 불리기도 한다.

[0050] S140 단계에서, 워킹 메모리(200)에 대한 소프트웨어 트레이닝(S/W training)이 수행된다. 소프트웨어 트레이닝(S/W training)은 하드웨어 트레이닝(H/W training)에 의해서 결정된 파라미터를 보다 미세하게 조정하기 위한 파인 트레이닝(Fine training)에 해당한다. 도 4a 및 도 4b에 도시된 바와 같이, 소프트웨어 트레이닝(S/W training)은 데이터의 타이밍을 최적의 신뢰성을 제공하기 위한 값으로 하드웨어 트레이닝(H/W training)에 의해서 설정된 파라미터를 재조정하기 위한 절차이다. 본 발명의 소프트웨어 트레이닝(S/W training)시 트레이닝 코드(TC)는 트레이닝의 대상인 워킹 메모리(200)에 로드될 수 있다. 즉, 소프트웨어 트레이닝(S/W training)을 수행하기 위한 트레이닝 코드(TC)가 워킹 메모리(200)의 제 1 영역에 로드된다. 트레이닝 코드(TC)의 실행시, 트레이닝 코드(TC)가 로드되지 않은 제 2 영역의 메모리들에 대한 소프트웨어 트레이닝이 수행될 것이다.

[0051] S150 단계에서, 소프트웨어 트레이닝(S/W training)의 결과로 확정된 최적화된 동작 파라미터가 스토리지 장치(180)에 저장될 것이다. 스토리지 장치(180)에 저장된 최적화된 동작 파라미터는 이후 부팅 동작시 소프트웨어 트레이닝(S/W training) 절차를 스킵하기 위하여 사용될 수 있을 것이다.

[0052] 이상의 본 발명의 실시 예에 따른 소프트웨어 트레이닝 방법에 따르면, 트레이닝 코드(TC)는 트레이닝의 대상이 되는 워킹 메모리(200)에 로드될 수 있다. 따라서, 트레이닝 코드(TC)의 사이즈 제한이 제거되어, 다양한 기능과 성능의 소프트웨어 트레이닝이 가능하다. 더불어, 시스템 인터커넥터(170) 또는 버스를 통해서 접근되는 에스램(SRAM)보다 고속의 워킹 메모리(200)에 트레이닝 코드(TC)가 로드됨에 따라, 로드와 패치에 소요되는 시간 단축이 가능하다. 따라서, 워킹 메모리(200)에 대한 소프트웨어 트레이닝의 속도를 높일 수 있다.

[0053] 도 6은 도 5의 소프트웨어 트레이닝 절차(S140 단계)를 세부적으로 보여주는 순서도이다. 도 6을 참조하면, 채널 인터리빙을 해제(또는 비활성화)하는 방법을 적용하여 트레이닝 코드(TC)를 워킹 메모리(200)에 로드하는 소프트웨어 트레이닝 방법이 설명될 것이다.

[0054] S141 단계에서, 소프트웨어 트레이닝 절차(S140 단계)가 시작되면 호스트(100)는 메모리 컨트롤러(120)의 채널 인터리빙 모드를 해제시킨다. 채널 인터리빙 모드의 해제에 따라, 호스트(100)는 특정 데이터를 워킹 메모리(200)의 어느 하나의 채널에 대응하는 영역에 집중적으로 기입할 수 있다. 즉, 호스트(100)는 채널 인터리빙의 해제에 따라 데이터를 파티셔닝(Partitioning)하여 워킹 메모리(200)에 저장하지 않아도 된다.

[0055] S143 단계에서, 호스트(100)는 채널 인터리빙의 해제 상태에서 트레이닝 코드(TC)와 트레이닝 데이터(TD)를 워킹 메모리(200)에 로드한다. 호스트(100)는 트레이닝 코드(TC)를 제 1 채널에 대응하는 제 1 영역에 로드한다. 그리고 호스트(100)는 트레이닝 데이터(TD)를 제 2 채널에 대응하는 제 2 영역에 기입할 것이다. 제 1 채널은 복수의 채널들 중 어느 하나의 채널일 수 있다. 그리고 제 1 영역은 제 1 채널을 통해서 접근 가능한 메모리 영역 또는 메모리 다이(Die)일 수 있다. 더불어, 제 2 채널은 복수의 채널들 중 제 1 채널을 제외한 하나 또는 그 이상의 채널일 수 있다. 따라서 제 2 영역은 워킹 메모리(200)의 모든 메모리 영역들 중 제 1 영역을 제외한 메모리 영역 또는 메모리 다이(Die)일 수 있다.

[0056] S145 단계에서, 호스트(100)는 제 1 영역에 로드된 트레이닝 코드(TC)를 패치하여 실행할 것이다. 예를 들면, 소프트웨어 트레이닝을 위해 할당된 CPU(110)에서 트레이닝 코드(TC)를 패치하여 실행할 수 있다. 트레이닝 코드(TC)의 실행에 따라 제 2 영역에 대응하는 메모리 영역이나 다이에 대한 트레이닝이 수행된다. 이때, 제 2 영역에 기입된 트레이닝 데이터(TD)가 독출되고, 호스트(100)는 독출된 트레이닝 데이터를 평가하여 최적의 파라미터를 결정할 수 있다.

[0057] S147 단계에서, 호스트(100)는 트레이닝 코드(TC)와 트레이닝 데이터(TD)를 워킹 메모리(200)에 다시 로드한다. 호스트(100)는 트레이닝 코드(TC)를 제 2 채널에 대응하는 제 2 영역에 로드한다. 그리고 호스트(100)는 트레이닝 데이터(TD)를 제 1 채널에 대응하는 제 1 영역에 기입할 것이다.

[0058] S149 단계에서, 호스트(100)는 제 2 영역에 로드된 트레이닝 코드(TC)를 패치하여 실행할 것이다. 예를 들면, 소프트웨어 트레이닝을 위해 할당된 CPU(110)에서 트레이닝 코드(TC)를 패치하여 실행할 수 있다. 트레이닝 코드(TC)의 실행에 따라 제 1 영역에 대응하는 메모리 영역이나 다이에 대한 트레이닝이 수행된다. 이때, 제 1 영역에 기입된 트레이닝 데이터(TD)가 독출되고, 호스트(100)는 독출된 트레이닝 데이터를 평가하여 제 1 영역에 대응하는 최적의 파라미터를 결정할 수 있다. 상술한 절차에 따라 워킹 메모리(200)에 대한 제반 영역이나 다이(또는, 칩)에 대한 소프트웨어 트레이닝이 완료되면, S140 단계는 종료된다.

[0059] 이상에서는 채널 인터리빙을 해제하는 방식으로 트레이닝 코드(TC)를 워킹 메모리에 로드하는 방법이 설명되었다. 트레이닝 코드(TC)가 로드되는 영역은 실질적으로 트레이닝에서 제외된다. 따라서, 트레이닝 코드(TC)가 로드된 메모리 영역이나 채널에 대한 트레이닝까지 수행하기 위해서는, 소프트웨어 트레이닝은 적어도 2회 실시되어야 한다.

[0060] 도 7은 도 6에서 설명된 메모리 컨트롤러(120)의 채널 인터리빙을 해제하는 방법을 보여주는 블록도이다. 도 7을 참조하면, 채널 인터리빙이 해제되기 전의 워킹 메모리(200)의 메모리 맵 (a)와 해제된 후의 메모리 맵 (b)가 도시되어 있다.

[0061] 채널 인터리빙이 해제되기 전의 메모리 맵 (a)를 참조하면, 인터리빙 단위(Interleaving Unit: 이하, IU)로 워킹 메모리(200)의 데이터가 기입되거나 독출된다. 예를 들면, 데이터를 기입하는 경우에 하나의 데이터는 인터리빙 단위(IU)로 분할되고, 분할된 인터리빙 단위의 데이터가 각각의 채널에 분산되어 기입된다. 예를 들면, 인터리빙 시퀀스가 (CH0→CH1→CH2→CH3→…→CHn-1)이라 가정하면, 각각의 채널에는 분할된 인터리빙 단위(IU)의 분할 데이터가 할당되고 기입될 수 있다. 읽기 동작에서도 동일한 방식으로 채널들이 선택되어 독출된 데이터가 호스트(100)에 전달될 수 있다.

[0062] 채널 인터리빙이 해제된 후의 메모리 맵 (b)를 참조하면, 인터리빙 단위(IU)는 더 이상 의미가 없어진다. 즉, 쓰기 데이터를 선택된 메모리 영역이나 칩(또는, 다이)에 연속적으로 기입할 수 있다. 여기서 채널(CH0)에 대응하는 영역은 채널(CH0)에 연결된 모든 연속된 메모리 영역(211)을 나타낸다. 따라서, 채널(CH0)에 대응하는 메모리 영역(211)에는 파티셔닝되지 않은 트레이닝 코드(TC)를 로드할 수 있다. 채널들(CH1, CH2, CH3) 각각에 대응하는 메모리 영역들(212, 213, 214) 각각에도 분할되지 않은 트레이닝 코드(TC)의 기입이 가능하다.

[0063] 도 8은 채널 인터리빙이 해제된 워킹 메모리(200)에 트레이닝 코드(TC)와 트레이닝 데이터를 로드하는 방법을 간략히 보여주는 도면이다. 도 8을 참조하면, 채널 인터리빙이 해제된 워킹 메모리(200)에 2회의 트레이닝 코드(TC)의 로드가 수행될 수 있다. 2회의 트레이닝 코드(TC)의 로드는 각각 워킹 메모리(200)의 서로 다른 영역에서 이루어진다.

[0064] 먼저, 트레이닝 코드(TC)의 로드 상태 (a)에는 도 6의 S143 단계에 대응하는 방식으로 트레이닝 코드(TC)와 트레이닝 데이터(TD)가 로드될 수 있다. 채널(CH0)에 대응하는 메모리 영역에는 트레이닝 코드(TC)가 로드된다. 그리고 채널들(CH1, CH2, CH3) 각각에 대응하는 메모리 영역들에는 트레이닝 데이터(TD)가 기입될 것이다. 채널(CH0)에 대응하는 메모리 영역의 용량은 트레이닝 코드(TC)가 충분히 수용 가능한 용량으로 제공될 것이다. 상술한 트레이닝 코드(TC)의 로드 상태에서, 호스트(100)는 트레이닝 코드(TC)를 패치 및 실행할 것이다. 그러면, 트레이닝 데이터(TD)가 저장된 채널들(CH1, CH2, CH3) 각각에 대응하는 메모리 영역(또는 칩이나 다이)들에 대한 소프트웨어 트레이닝이 수행될 것이다.

[0065] 이어서, 트레이닝 코드(TC)의 로드 상태 (b)에는 도 6의 S147 단계에 대응하는 방식으로 트레이닝 코드(TC)와 트레이닝 데이터(TD)가 로드될 수 있다. 예를 들면, 채널(CH3)에 대응하는 메모리 영역에는 트레이닝 코드(TC)가 로드된다. 그리고 채널들(CH0)에 대응하는 메모리 영역들에는 트레이닝 데이터(TD)가 기입될 것이다. 이러한 트레이닝 코드(TC)와 트레이닝 데이터(TD)의 로드는 S143 단계 및 S145 단계에서 트레이닝의 대상이 되지 못한 채널(CH0)의 메모리 영역을 트레이닝하기 위한 설정이다. 상술한 트레이닝 코드(TC)의 로드 상태에서, 호스트(100)는 트레이닝 코드(TC)를 패치 및 실행할 것이다. 그러면, 트레이닝 데이터(TD)가 저장된 채널(CH0)에 대응하는 메모리 영역(또는 칩이나 다이)들에 대한 소프트웨어 트레이닝이 수행될 것이다.

[0066] 도 9는 본 발명의 다른 실시 예에 따른 소프트웨어 트레이닝 방법을 보여주는 순서도이다. 도 9를 참조하면, 채널 인터리빙의 해제와 같은 절차 없이도 트레이닝 코드를 워킹 메모리(200)에 로드하여 소프트웨어 트레이닝을

수행할 수 있다. 이러한 다른 실시 예에 따른 트레이닝 코드(TC)의 로드 및 실행은 도 5의 S140 단계에서 수행될 것이다.

[0067] S142 단계에서, 소프트웨어 트레이닝 절차(S140 단계)가 시작되면 호스트(100)는 채널의 인터리빙 단위(IU)의 크기를 결정한다. 예를 들면, 호스트(100)는 트레이닝 코드(TC)의 사이즈를 참조하여 채널의 인터리빙 단위(IU)의 크기를 결정할 수 있다. 채널 인터리빙 단위(IU)의 디폴트값이 128 바이트(Byte)이고, 트레이닝 코드(TC)의 사이즈가 120메가 바이트(MByte)인 경우를 가정하기로 하자. 그러면, 호스트(100)는 채널의 인터리빙 단위(IU)의 크기를 10^6배로 증가시킬 수 있을 것이다.

[0068] S144 단계에서, 호스트(100)는 결정된 채널의 인터리빙 단위(IU)에 따라 채널 인터리빙의 설정을 변경할 것이다. 즉, 호스트(100)는 하나의 채널 인터리빙 단위(IU)의 사이즈가 트레이닝 코드(TC)를 수용 가능한 정도로 증가시킬 수 있을 것이다.

[0069] S146a 단계에서, 호스트(100)는 조정된 채널 인터리빙 단위(IU) 조건에서 트레이닝 코드(TC)와 트레이닝 데이터(TD)를 워킹 메모리(200)에 로드한다. 호스트(100)는 트레이닝 코드(TC)를 제 1 채널에 대응하는 제 1 영역에 로드한다. 그리고 호스트(100)는 트레이닝 데이터(TD)를 제 2 채널에 대응하는 제 2 영역에 로드할 것이다. 여기서, 제 1 채널은 채널 인터리빙을 위해 할당되는 복수의 메모리 영역 중 하나이다. 제 2 채널은 채널 인터리빙을 위해 할당되는 제 1 채널과 다른 채널이다.

[0070] S146b 단계에서, 호스트(100)는 제 1 채널의 메모리 영역에 로드된 트레이닝 코드(TC)를 패치하여 실행할 것이다. 트레이닝 코드(TC)의 실행에 따라 제 2 채널의 메모리 영역에 대응하는 메모리 영역이나 다이에 대한 트레이닝이 수행된다. 이때, 제 2 영역에 기입된 트레이닝 데이터(TD)가 호스트(100)로 전달될 것이다. 그러면, 호스트(100)는 독출된 트레이닝 데이터(TD)를 평가하여, 제 2 채널에 대응하는 최적의 접근 파라미터를 결정할 수 있다.

[0071] S148a 단계에서, 호스트(100)는 트레이닝 코드(TC)와 트레이닝 데이터(TD)를 워킹 메모리(200)에 다시 로드한다. 이때, 호스트(100)는 트레이닝 코드(TC)를 제 2 채널에 대응하는 제 2 영역에 로드한다. 그리고 호스트(100)는 트레이닝 데이터(TD)를 제 1 채널에 대응하는 제 1 영역에 기입할 것이다.

[0072] S148b 단계에서, 호스트(100)는 제 2 채널의 메모리 영역에 로드된 트레이닝 코드(TC)를 패치하여 실행할 것이다. 트레이닝 코드(TC)의 실행에 따라 제 1 채널의 메모리 영역에 대응하는 메모리 영역이나 다이에 대한 트레이닝이 수행된다. 이때, 제 1 영역에 기입된 트레이닝 데이터(TD)가 읽혀지고 호스트(100)로 전달될 것이다. 그러면, 호스트(100)는 독출된 트레이닝 데이터(TD)를 평가하여, 제 1 채널에 대응하는 최적의 접근 파라미터를 결정할 것이다.

[0073] 상술한 절차에 따라 채널 인터리빙의 해제없이 워킹 메모리(200)에 대한 제반 영역이나 다이(또는 칩)에 대한 소프트웨어 트레이닝이 수행될 수 있다.

[0074] 도 10은 도 9에서 설명된 실시 예에 따른 트레이닝 코드의 로드 방법을 보여주는 블록도이다. 도 10을 참조하면, 워킹 메모리(200)에 대한 채널 인터리빙의 해제 절차 없이도 트레이닝 코드(TC)는 파티셔닝되지 않고 워킹 메모리(200)에 로드될 수 있다.

[0075] 디폴트값에 대응하는 채널 인터리빙 단위(IU) 상태에서 메모리 맵 (a)를 참조하면, 인터리빙 단위(IU)는 128 바이트(Byte)라 가정하기로 하자. 그리고 파티셔닝되지 않은 트레이닝 코드(TC)의 크기는 120MByte라 가정하기로 하자. 이러한 상황에서 트레이닝 코드(TC)를 워킹 메모리(200)에 로드하면, 트레이닝 코드(TC)는 128바이트 단위로 분리된다. 그리고 분리된 트레이닝 코드(TC)는 워킹 메모리(200)에 존재하는 모든 메모리 칩(또는, 다이)에 분산될 것이다.

[0076] 하지만, 트레이닝 코드(TC)의 사이즈를 고려하여 채널 인터리빙 단위(IU)를 조정한 상태인 (b)를 참조하면, 하나의 채널 인터리빙 유닛(IU)에 트레이닝 코드(TC)가 파티셔닝되지 않고 로드될 수 있다. 즉, 조정된 128MByte의 채널 인터리빙 단위(IU)에 120MByte 사이즈의 트레이닝 코드(TC)가 수용될 수 있다. 따라서, 하나의 칩이나 다이 내에 트레이닝 코드(TC)가 모두 저장될 수 있음을 알 수 있다.

[0077] 이러한 상태에서, 채널 인터리빙의 해제없이 채널(CH0)에 로드된 트레이닝 코드(TC)를 실행하면 나머지 채널들(CH1, CH2, CH3)에 대한 소프트웨어 트레이닝이 가능하다.

[0078] 도 11은 본 발명의 또 다른 실시 예에 따른 소프트웨어 트레이닝을 수행하는 전자 장치(10')를 보여주는 블록도

이다. 도 11을 참조하면, 호스트(100')는 소프트웨어 트레이닝 모드시에 활성화되는 아비터(190)를 포함할 수 있다. 아비터(190)에 의해서 채널 인터리빙 시퀀스에 따라 생성되는 어드레스가 리맵핑된 어드레스(ADD_TR)로 워킹 메모리(200)에 전달될 수 있다.

[0079] 호스트(100')는 부팅이나 리셋 동작시에 워킹 메모리(200)에 대한 소프트웨어 트레이닝을 수행할 것이다. 이때, 채널 인터리빙 모드로 생성되는 워킹 메모리(200)에 대한 접근 어드레스는 아비터(190)를 통해서 중재될 수 있다. 예를 들면, 채널 인터리빙 모드로 생성되는 어드레스의 특정 비트값을 마스킹하는 방식으로 채널 인터리빙 효과가 제거될 수 있다. 아비터(190)는 어드레스 비트의 특정 비트를 마스킹하는 논리 회로로 간단히 구성될 수 있을 것이다. 하지만, 아비터(190)에 의한 어드레스의 변환 방법은 여기에만 국한되지 않음은 잘 이해될 것이다. 그리고 아비터(190)는 하나의 구성(Element)으로 도시되어 있으나, 아비터(190)는 소프트웨어 또는 알고리즘으로 구현될 수도 있음은 잘 이해될 것이다.

[0080] 도 12는 도 11의 호스트(100')의 예시적인 구성을 보여주는 블록도이다. 도 12를 참조하면, 호스트(100')는 워킹 메모리(200)로의 접근을 요청하는 복수의 마스터 구성들을 포함할 수 있다. 특히, 복수의 CPU(110) 중에서 어느 하나가 소프트웨어 트레이닝을 위한 워킹 메모리(200)로의 접근 요청을 발행하는 것으로 간주할 수 있다. 아비터(190)는 이러한 CPU(110)의 접근 어드레스를 변환하여 메모리 컨트롤러(120)에 전달할 수 있다. 여기서, 아비터(190)를 제외하면, 호스트(100')의 구성은 도 2의 호스트(100)와 실질적으로 동일하다. 따라서, CPU(110), 메모리 컨트롤러(120), GPU(130), 사용자 인터페이스 컨트롤러(140), 스토리지 인터페이스(150), 가속기(160), 시스템 인터커넥터(170) 등에 대한 설명은 생략하기로 한다.

[0081] 아비터(190)는 소프트웨어 트레이닝을 수행하는 CPU(110)와 시스템 인터커넥터(170) 사이에 위치할 수 있다. 그리고 소프트웨어 트레이닝 모드시, 아비터(190)의 어드레스 변환 동작이 활성화된다. 트레이닝 코드(TC)나 트레이닝 데이터(TD)를 로드하기 위하여 CPU(110)가 발행하는 어드레스는 아비터(190)에 의해서 변환 어드레스(ADD_TR)로 조정된다. 변환 어드레스(ADD_TR)는 시스템 인터커넥터(170), 메모리 컨트롤러(120)를 경유하여 워킹 메모리(200)에 제공될 것이다.

[0082] 도 13은 아비터(190)의 동작을 구체적으로 보여주는 블록도이다. 도 13을 참조하면, 아비터(190)는 소프트웨어 트레이닝 모드에서 CPU(110)에서 발행되는 채널 인터리빙을 위한 어드레스(ADD)를 변환 어드레스(ADD_TR)로 리맵핑 또는 재배열할 수 있다.

[0083] 소프트웨어 트레이닝 모드에서, CPU(110)는 통상의 트레이닝 절차에 따라 스토리지 장치(180)로부터 트레이닝 코드(TC)를 요청할 것이다. 그리고 CPU(110)는 트레이닝 코드(TC)를 워킹 메모리(200)에 로드하기 위한 명령어와 채널 인터리빙 모드의 어드레스(ADD)를 발행할 것이다. 이때 CPU(110)가 발행하는 채널 인터리빙 모드에 해당되는 어드레스(ADD)에 따라 CPU(110)에서 인식하는 워킹 메모리(200)는 메모리 맵(191)과 같다.

[0084] 아비터(190)는 CPU(110)가 발행하는 채널 인터리빙 모드에 해당되는 어드레스(ADD)를 소프트웨어 트레이닝을 위한 변환 어드레스(ADD_TR)로 재배열한다. 예를 들면, 아비터(190)는 채널 인터리빙 방식으로 생성되는 어드레스(ADD)의 특정 비트를 마스킹(Masking)하는 방식으로 변환 어드레스(ADD_TR)를 생성할 수 있을 것이다. 변환 어드레스(ADD_TR)에 의해서 아비터(190)에서 인식하는 워킹 메모리(200)는 메모리 맵(193)과 같다.

[0085] 상술한 아비터(190)의 활성화에 의해서 CPU(110)는 채널 인터리빙 모드를 유지한 상태에서 트레이닝 코드(TC)를 워킹 메모리(200)에 로드할 것이다. 하지만, 아비터(190)의 중재에 의해서 트레이닝 코드(TC)는 파티셔닝되지 않고 어느 하나의 채널에 대응하는 메모리 영역에 저장될 수 있다.

[0086] 도 14는 아비터(190)를 포함하는 호스트(100)의 소프트웨어 트레이닝 절차를 보여주는 순서도이다. 도 14를 참조하면, 소프트웨어 트레이닝 모드에서 어드레스 해싱을 수행하는 아비터(190)가 활성화된다. 그리고 트레이닝 코드(TC)는 파티셔닝되지 않고 어느 하나의 채널에 대응하는 메모리 영역에 로드될 수 있다.

[0087] S241 단계에서, CPU(110)는 현재 동작 모드가 소프트웨어 트레이닝(S/W Training) 모드인지를 판단한다. 소프트웨어 트레이닝(S/W Training) 모드로 판단되면, 절차는 아비터(190)를 활성화하기 위한 S242 단계로 이동할 것이다. 하지만, 현재 동작 모드가 소프트웨어 트레이닝(S/W Training) 모드가 아닌 경우에, 절차는 소프트웨어 트레이닝(S/W Training) 모드가 활성화되는 시점을 계속 검출할 것이다.

[0088] S242 단계에서, CPU(110)는 트레이닝 코드(TC) 또는 트레이닝 데이터(TD)를 로드하기 위한 워킹 메모리(200)의 어드레스를 조정하기 위한 아비터(190)를 활성화한다. 즉, 아비터(190)의 활성화에 따라 아비터(190)가 수행하는 어드레스 해싱(Address Hashing) 기능이 활성화될 것이다.

[0089] S243 단계에서, CPU(110)는 트레이닝 코드(TC)와 트레이닝 데이터(TD)를 워킹 메모리(200)에 로드하기 위한 명령어와 어드레스(ADD)를 발행할 것이다. 아비터(190)의 어드레스 해싱 기능에 의해서 어드레스(ADD)는 변환 어드레스(ADD_TR)로 변환된다. 따라서, 트레이닝 코드(TC)는 파티셔닝되지 않은 상태로 제 1 채널에 대응하는 제 1 영역에 로드될 수 있다. 그리고 트레이닝 데이터(TD)는 제 2 채널에 대응하는 제 2 영역에 로드될 것이다.

[0090] S244 단계에서, 호스트(100)는 제 1 채널의 메모리 영역에 로드된 트레이닝 코드(TC)를 패치하여 실행할 것이다. 트레이닝 코드(TC)의 실행에 따라 제 2 채널의 메모리 영역에 대응하는 메모리 영역이나 다이에 대한 트레이닝이 수행된다. 이때, 제 2 영역에 기입된 트레이닝 데이터(TD)가 호스트(100)로 전달될 것이다. 그러면, 호스트(100)는 독출된 트레이닝 데이터(TD)를 평가하여, 제 2 채널에 대응하는 최적의 접근 파라미터를 결정할 수 있다.

[0091] S245 단계에서, CPU(110)는 트레이닝 코드(TC)와 트레이닝 데이터(TD)를 S243 단계에서와는 다른 채널에 할당하기 위한 명령어와 어드레스(ADD)를 발행할 것이다. 아비터(190)의 어드레스 해싱 기능에 의해서 어드레스(ADD)는 변환 어드레스(ADD_TR)로 변환된다. 따라서, 트레이닝 코드(TC)는 파티셔닝되지 않은 상태로 제 2 채널에 대응하는 제 2 영역에 로드될 수 있다. 그리고 트레이닝 데이터(TD)는 제 1 채널에 대응하는 제 1 영역에 로드될 것이다.

[0092] S246 단계에서, 호스트(100)는 제 2 채널의 메모리 영역에 로드된 트레이닝 코드(TC)를 패치하여 실행할 것이다. 트레이닝 코드(TC)의 실행에 따라 제 1 채널의 메모리 영역에 대응하는 메모리 영역이나 다이에 대한 트레이닝이 수행된다. 이때, 제 1 영역에 기입된 트레이닝 데이터(TD)가 읽혀지고 호스트(100)로 전달될 것이다. 그러면, 호스트(100)는 독출된 트레이닝 데이터(TD)를 평가하여, 제 1 채널에 대응하는 최적의 접근 파라미터를 결정할 것이다.

[0093] S247 단계에서, 호스트(100)는 소프트웨어 트레이닝 모드가 완료됨에 따라 아비터(190)의 어드레스 해싱 기능을 해제시킨다. 상술한 절차에 따라 소프트웨어 트레이닝을 위한 트레이닝 코드(TC)를 워킹 메모리(200)에 로드하여 실행할 수 있다.

[0094] 도 15는 본 발명의 실시 예에 따라 소프트웨어 트레이닝을 통해 생성된 접근 파라미터를 이용하는 방법을 예시적으로 보여주는 순서도이다. 도 15를 참조하면, 소프트웨어 트레이닝의 결과는 추후에 수행되는 부팅 동작에 적용하여 전자 장치(10)의 부팅 속도를 높일 수 있다. 여기서, 소프트웨어 트레이닝을 통해 생성된 접근 파라미터는 스토리지 장치(170)나 호스트(100) 내의 비휘발성 메모리 소자에 저장되어 있는 것으로 가정한다.

[0095] S310 단계에서, 전자 장치(10)에 대한 부팅이 수행된다. 예를 들면, 전자 장치(10)의 전원 스위치가 턴온되거나, 리셋에 의해서 초기화의 수행시에 부팅 동작이 수행될 수 있을 것이다.

[0096] S320 단계에서, 가장 기본적으로 전자 장치(10)의 제반 입출력 인터페이스(I/O Interface)에 대한 트레이닝이 수행될 수 있다. 워킹 메모리(200)에 대한 트레이닝의 이전에 최소한의 신뢰성을 보장하기 위한 다양한 트레이닝 동작들이 이 단계에서 수행될 수 있을 것이다.

[0097] S330 단계에서, 워킹 메모리(200)에 대한 하드웨어 트레이닝(H/W training)이 수행된다. 하드웨어 트레이닝은 예를 들면, 워킹 메모리(200)에 미리 정의된 초기값 파라미터로 메모리 컨트롤러(120)를 설정하는 동작일 수 있다. 하드웨어 트레이닝을 통해서 워킹 메모리(200)로의 접근을 위한 최소한의 신뢰성은 획득될 수 있을 것이다. 따라서, 하드웨어 트레이닝(H/W training)은 다른 표현으로 코어스 트레이닝(Coarse training)이라 불리기도 한다.

[0098] S340 단계에서, 전자 장치(10)의 내부에 이전의 소프트웨어 트레이닝을 통해서 생성된 접근 파라미터가 존재하는지 검출된다. 만일, 이전의 소프트웨어 트레이닝을 통해서 생성된 접근 파라미터가 존재하는 경우(Yes 방향), 워킹 메모리(200)에 대한 트레이닝은 종료된다. 반면, 이전의 소프트웨어 트레이닝을 통해서 생성된 접근 파라미터가 존재하지 않는 경우(No 방향), 절차는 S350 단계로 이동한다.

[0099] S350 단계에서, 워킹 메모리(200)에 대한 소프트웨어 트레이닝(S/W training)이 수행된다. 소프트웨어 트레이닝(S/W training)은 하드웨어 트레이닝(H/W training)에 의해서 결정된 파라미터를 보다 미세하게 조정하기 위한 파인 트레이닝(Fine training)에 해당한다.

[0100] S360 단계에서, 소프트웨어 트레이닝(S/W training)의 결과로 확정된 최적화된 접근 파라미터가 스토리지 장치(180)에 저장될 것이다.

[0101] 이상의 본 발명의 실시 예에 따르면, 소프트웨어 트레이닝을 통해서 획득된 접근 파라미터는 매 부팅 동작시 소

프트웨어 트레이닝의 실행없이 적용될 수 있다. 따라서, 매 부팅 동작시 소프트웨어 트레이닝에 소요되는 시간이나 자원이 절약될 수 있다.

[0102] 도 16은 본 발명의 실시 예에 따라 소프트웨어 트레이닝을 통해 생성된 접근 파라미터를 이용하는 다른 방법을 보여주는 순서도이다. 도 16을 참조하면, 소프트웨어 트레이닝을 통해 생성된 접근 파라미터를 사용하여 하드웨어 트레이닝과 소프트웨어 트레이닝 절차를 모두 생략할 수 있다. S410 단계, S420 단계, S440 단계, S450 단계, S460 단계는 도 15의 단계들과 실질적으로 동일하므로 이 단계들의 상세한 설명은 생략하기로 한다.

[0103] S410 단계와 S420 단계에서, 전자 장치(10)에 대한 부팅 및 입출력 인터페이스(I/O Interface)에 대한 트레이닝이 수행된다.

[0104] S430 단계에서, 이전의 소프트웨어 트레이닝을 통해서 생성된 접근 파라미터가 존재하는지 검출된다. 만일, 이전의 소프트웨어 트레이닝을 통해서 생성된 접근 파라미터가 존재하는 경우(Yes 방향), 워킹 메모리(200)에 대한 트레이닝 절차는 종료된다. 반면, 이전의 소프트웨어 트레이닝을 통해서 생성된 접근 파라미터가 존재하지 않는 경우(No 방향), 절차는 S440 단계로 이동한다.

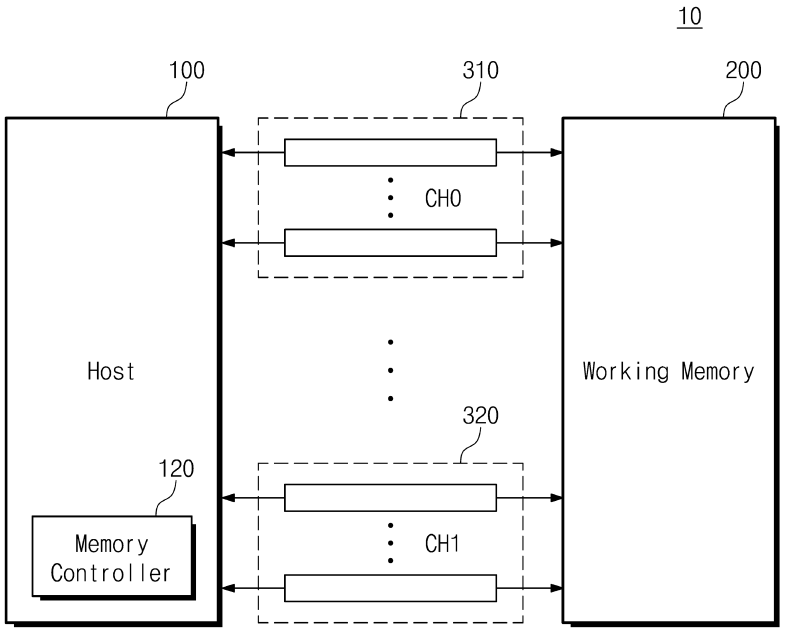
[0105] S440 단계에서, 워킹 메모리(200)에 대한 하드웨어 트레이닝(H/W training)이 수행된다. S450 단계에서, 소프트웨어 트레이닝(S/W training)이 수행된다. 그리고 S460 단계에서, 소프트웨어 트레이닝(S/W training)의 결과로 확정된 최적화된 접근 파라미터의 저장이 수행된다.

[0106] 이상에서는 소프트웨어 트레이닝을 통해서 획득된 접근 파라미터가 전자 장치(10)의 부팅시 하드웨어 트레이닝 및 소프트웨어 트레이닝을 생략할 수 있는 기회를 제공할 수 있다. 따라서, 매 부팅 동작시 트레이닝에 소요되는 시간이나 자원이 절약될 수 있다.

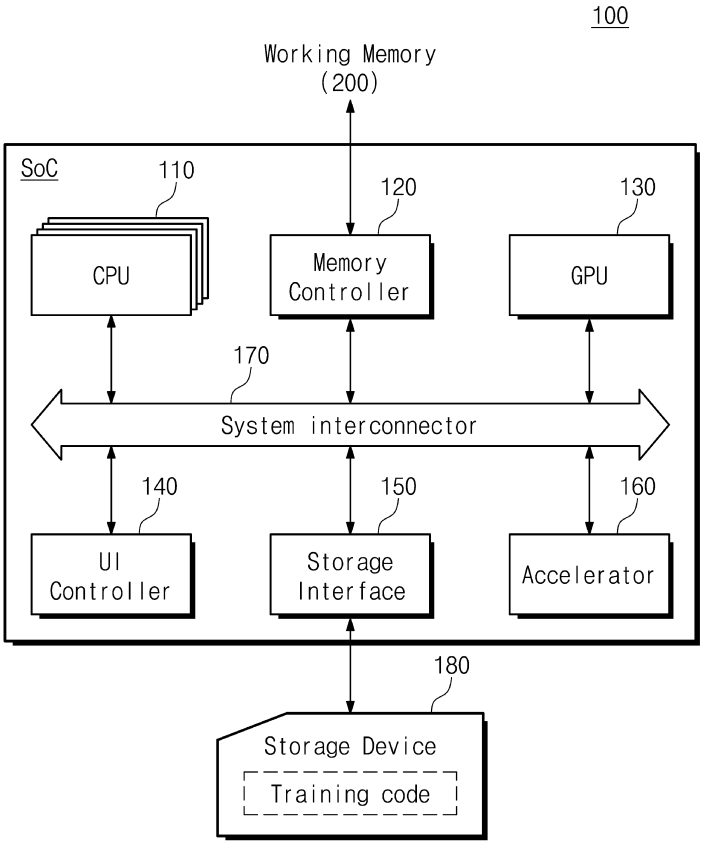
[0107] 이상에서와 같이 도면과 명세서에서 실시 예가 개시되었다. 여기서 특정한 용어들이 사용되었으나, 이는 단지 본 발명을 설명하기 위한 목적에서 사용된 것이지 의미 한정이나 특허 청구범위에 기재된 본 발명의 범위를 제한하기 위하여 사용된 것은 아니다. 그러므로 본 기술분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시 예가 가능하다는 점을 이해할 것이다. 따라서 본 발명의 진정한 기술적 보호 범위는 첨부된 특허 청구범위의 기술적 사상에 의해 정해져야 할 것이다.

도면

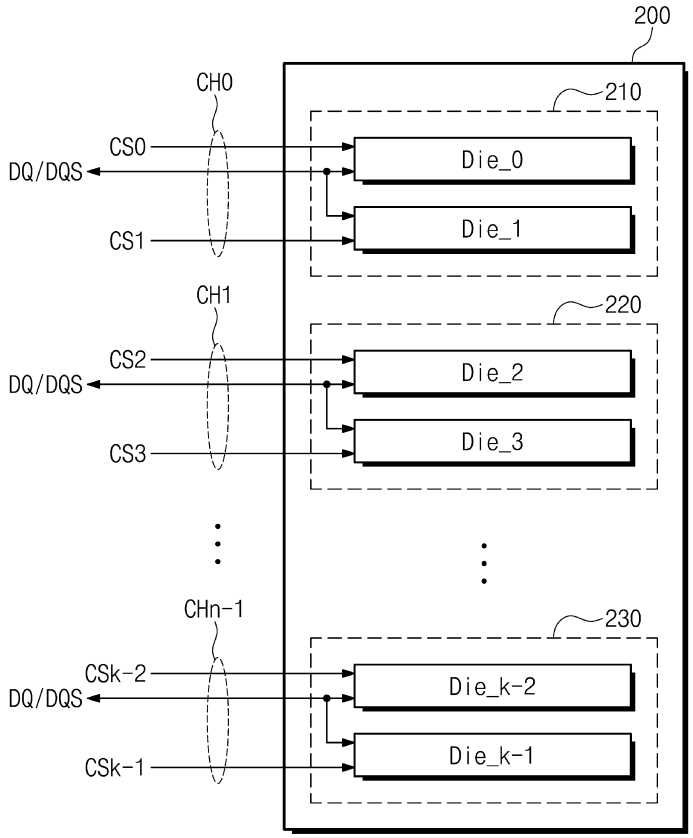
도면1



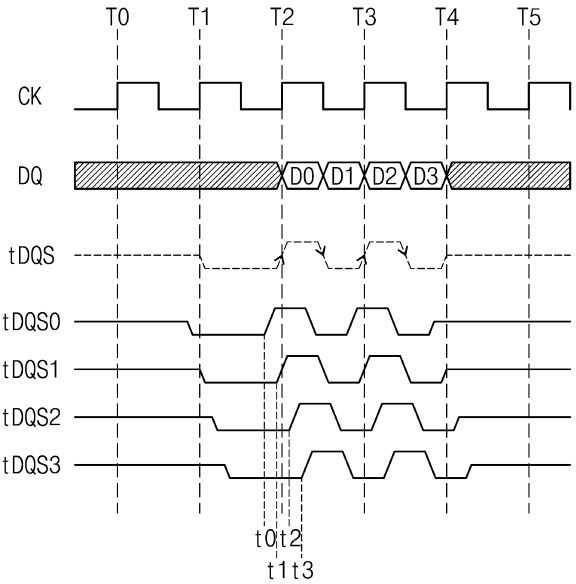
도면2


100
Working Memory
(200)
SoC
110
120
130
CPU
Memory
Controller
GPU
170
System interconnector
140
150
160
UI
Controller
Storage
Interface
Accelerator
180
Storage Device
Training code

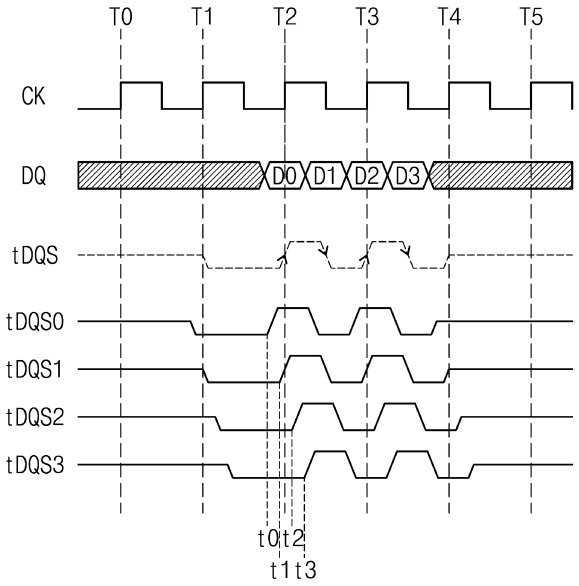
도면3


200
CH0
210
CS0
DQ/DQS
Die_0
Die_1
CS1
CH1
220
CS2
DQ/DQS
Die_2
Die_3
CS3
CHn-1
230
CSk-2
DQ/DQS
Die_k-2
Die_k-1
CSk-1

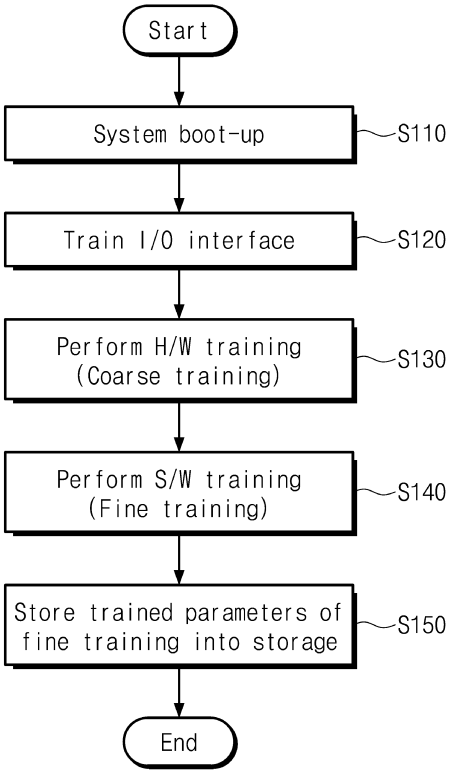
도면4a


T0
T1
T2
T3
T4
T5
CK
DQ
D0
D1
D2
D3
tDQS
tDQS0
tDQS1
tDQS2
tDQS3
t0
t1
t2
t3

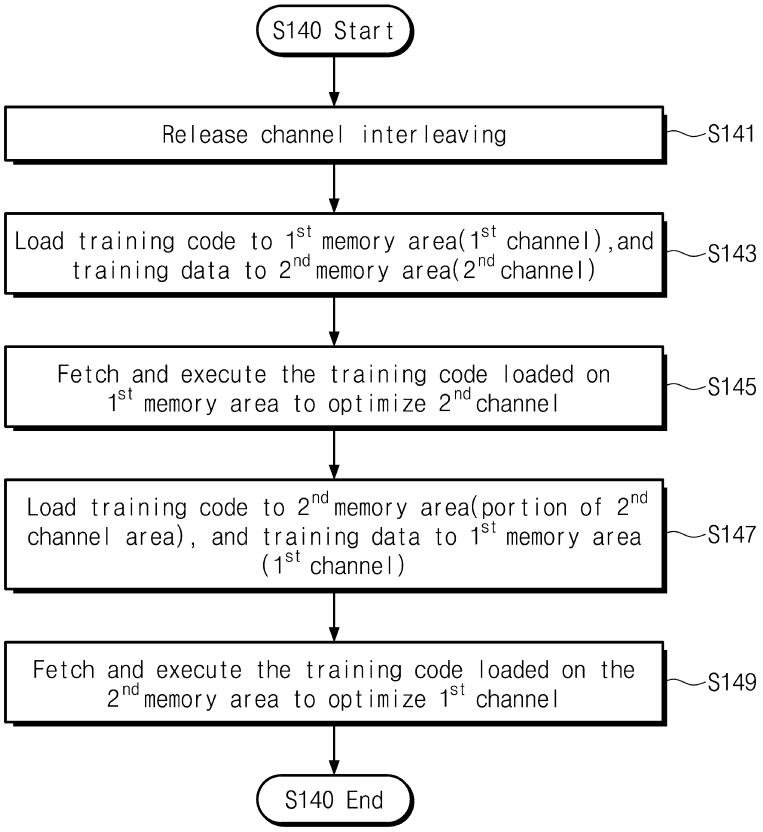
도면4b


T0
T1
T2
T3
T4
T5
CK
DQ
D0
D1
D2
D3
tDQS
tDQS0
tDQS1
tDQS2
tDQS3
t0
t2
t1
t3

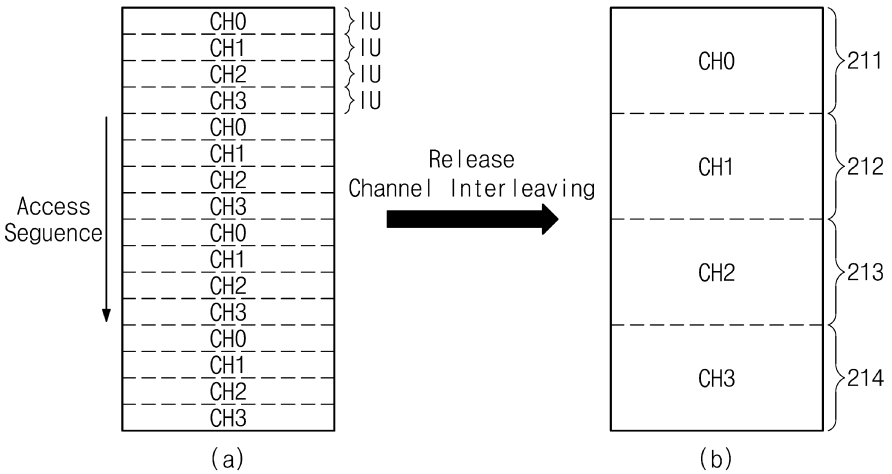
도면5


Start
System boot-up
S110
Train I/O interface
S120
Perform H/W training
(Coarse training)
S130
Perform S/W training
(Fine training)
S140
Store trained parameters of
fine training into storage
S150
End

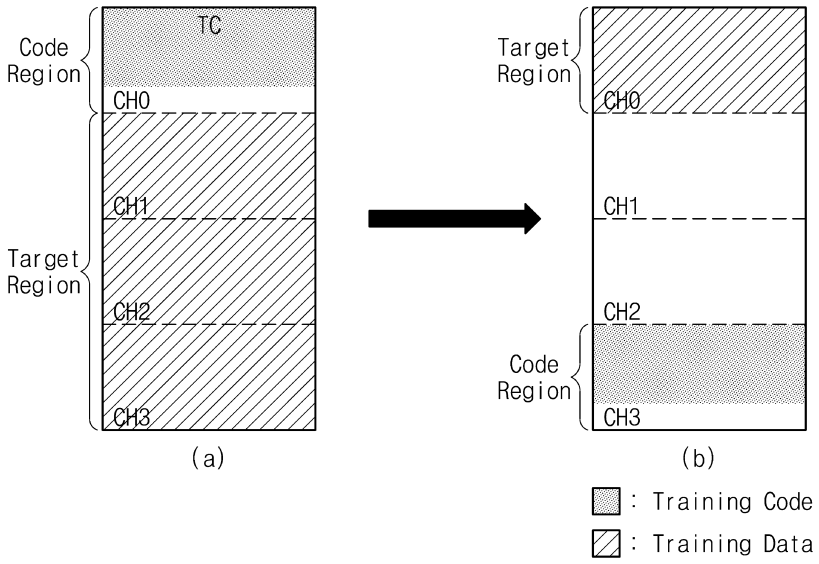
도면6


S140 Start
Release channel interleaving
S141
Load training code to 1st memory area(1st channel),and training data to 2nd memory area(2nd channel)
S143
Fetch and execute the training code loaded on 1st memory area to optimize 2nd channel
S145
Load training code to 2nd memory area(portion of 2nd channel area), and training data to 1st memory area (1st channel)
S147
Fetch and execute the training code loaded on the 2nd memory area to optimize 1st channel
S149
S140 End

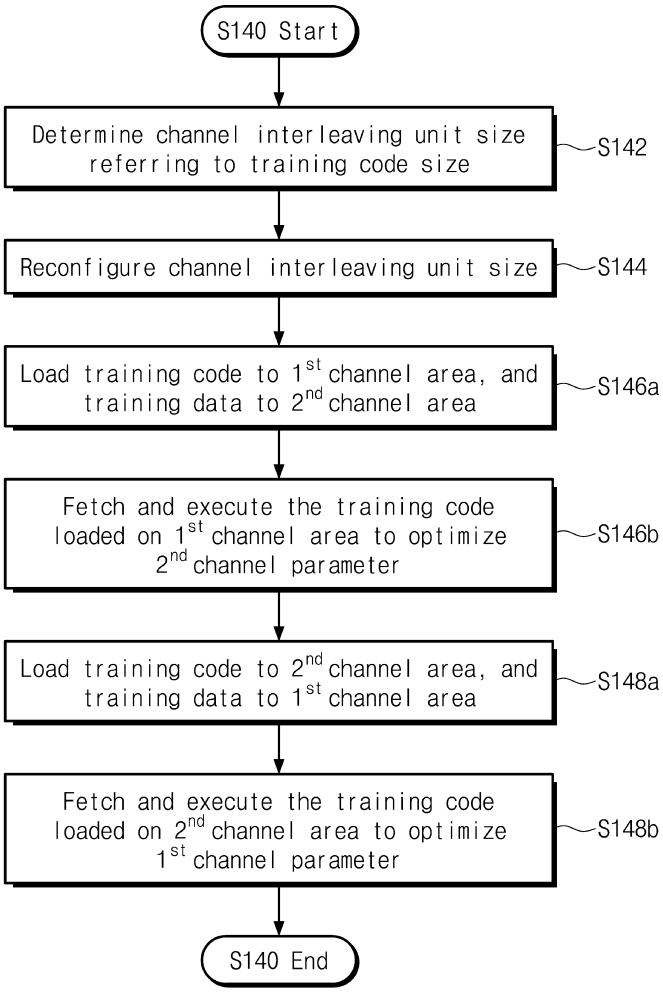
도면7


CH0
CH1
CH2
CH3
IU
Access Seguence
Release Channel Interleaving
211
212
213
214
(a)
(b)

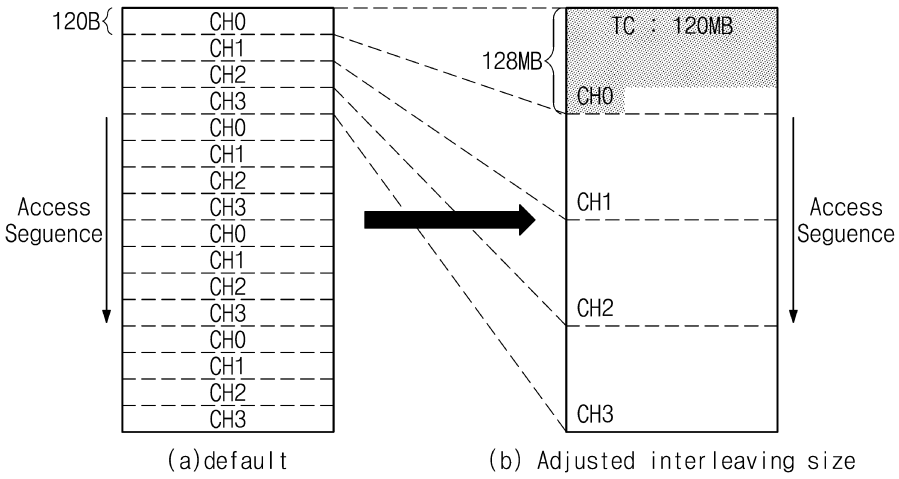
도면8


Code Region
TC
CH0
CH1
Target Region
CH2
CH3
(a)
Target Region
CH0
CH1
CH2
Code Region
CH3
(b)
: Training Code
: Training Data

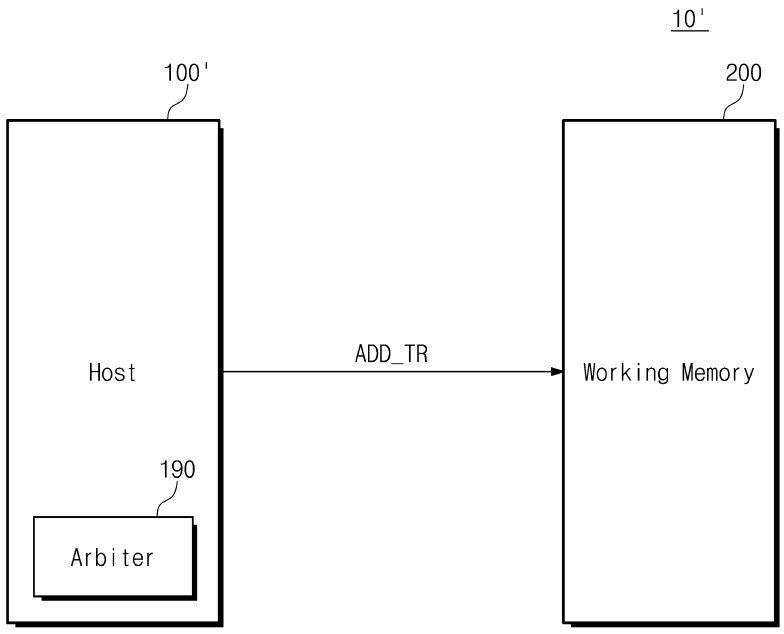
도면9


S140 Start
Determine channel interleaving unit size referring to training code size
S142
Reconfigure channel interleaving unit size
S144
Load training code to 1st channel area, and training data to 2nd channel area
S146a
Fetch and execute the training code loaded on 1st channel area to optimize 2nd channel parameter
S146b
Load training code to 2nd channel area, and training data to 1st channel area
S148a
Fetch and execute the training code loaded on 2nd channel area to optimize 1st channel parameter
S148b
S140 End

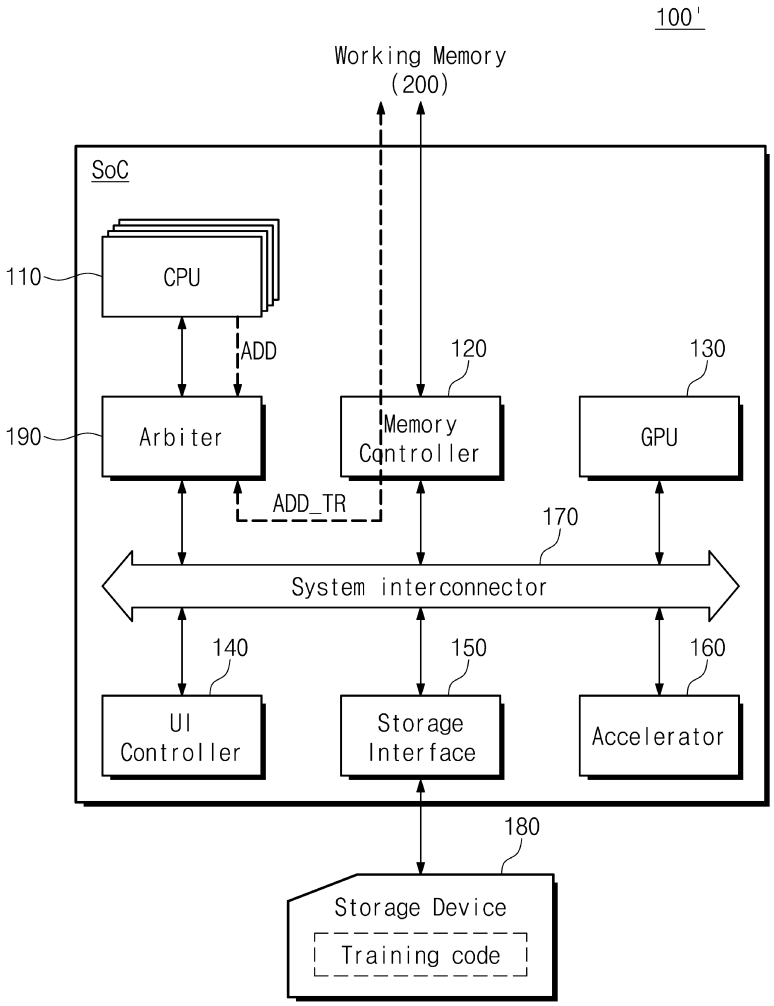
도면10


120B
CH0
CH1
CH2
CH3
CH0
CH1
CH2
CH3
CH0
CH1
CH2
CH3
CH0
CH1
CH2
CH3
Access Seguence
128MB
TC : 120MB
CH0
CH1
CH2
CH3
Access Seguence
(a)default
(b) Adjusted interleaving size

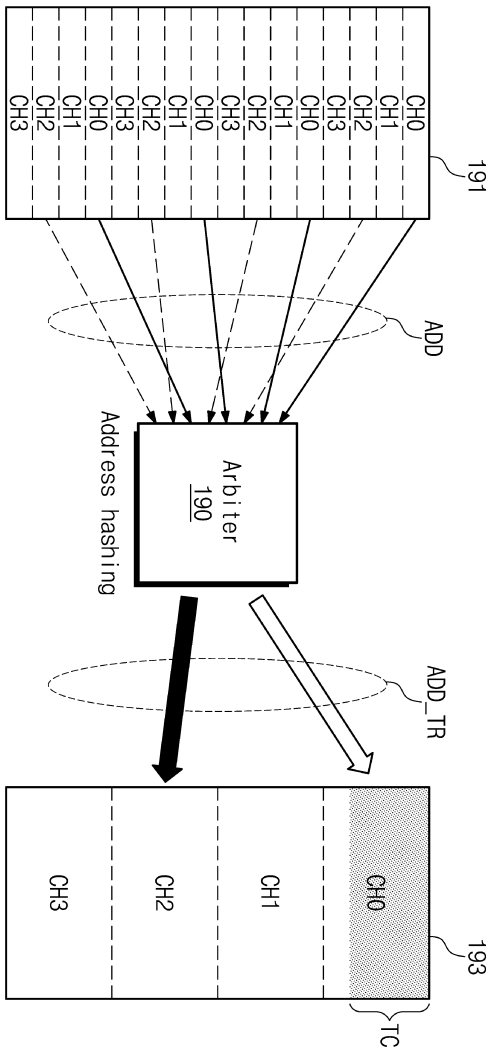
도면11


10'
100'
200
Host
ADD_TR
Working Memory
190
Arbiter

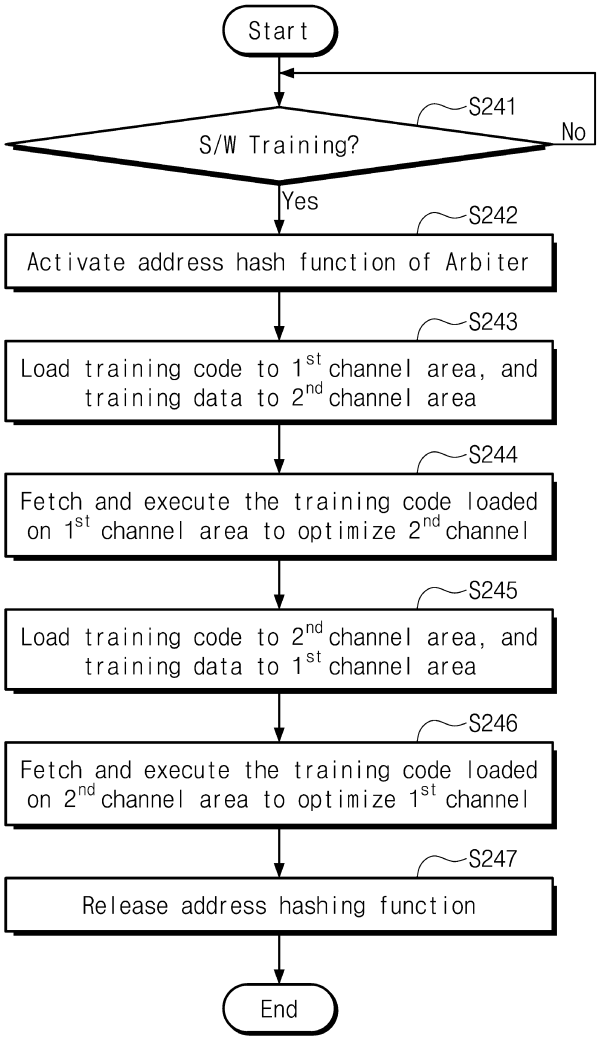
도면12


100'
Working Memory
(200)
SoC
110
CPU
ADD
120
130
190
Arbiter
Memory
Controller
GPU
ADD_TR
170
System interconnector
140
150
160
UI
Controller
Storage
Interface
Accelerator
180
Storage Device
Training code

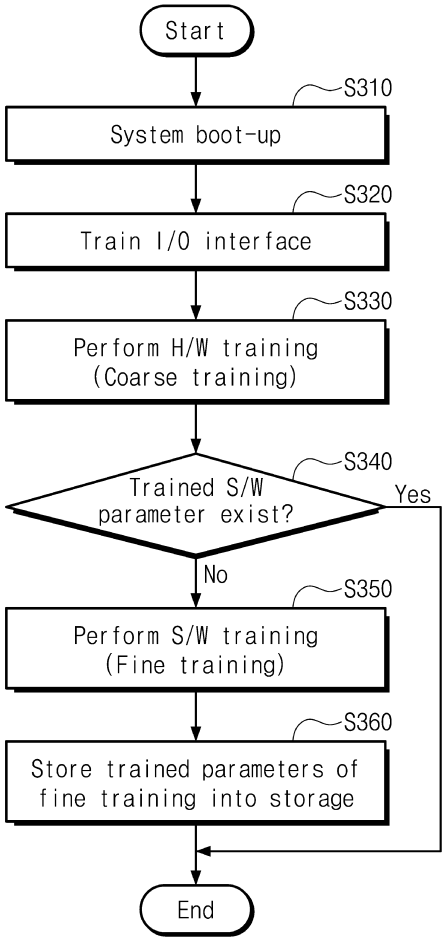
도면13


CH0
CH1
CH2
CH3
191
ADD
Address hashing
Arbiter
190
ADD_TR
CH3
CH2
CH1
CH0
193
TC

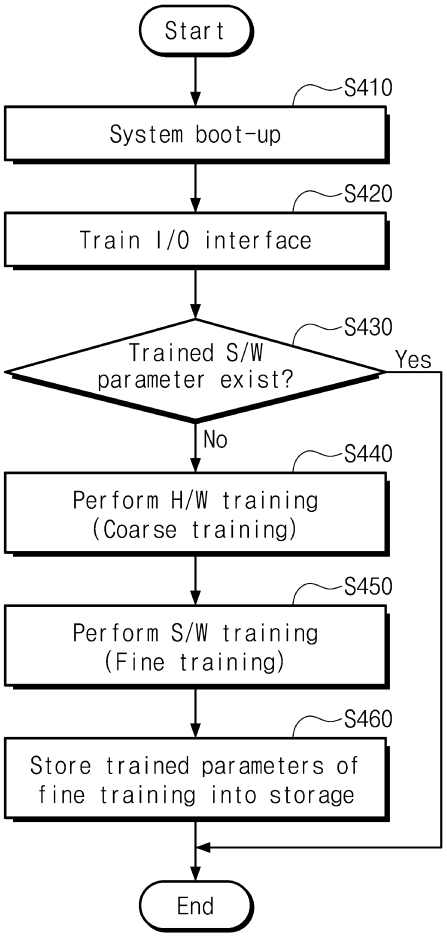
도면14


Start
S241
S/W Training?
No
Yes
S242
Activate address hash function of Arbiter
S243
Load training code to 1st channel area, and training data to 2nd channel area
S244
Fetch and execute the training code loaded on 1st channel area to optimize 2nd channel
S245
Load training code to 2nd channel area, and training data to 1st channel area
S246
Fetch and execute the training code loaded on 2nd channel area to optimize 1st channel
S247
Release address hashing function
End

도면15


Start
S310
System boot-up
S320
Train I/O interface
S330
Perform H/W training
(Coarse training)
S340
Trained S/W
parameter exist?
Yes
No
S350
Perform S/W training
(Fine training)
S360
Store trained parameters of
fine training into storage
End

도면16


Start
S410
System boot-up
S420
Train I/O interface
S430
Trained S/W parameter exist?
Yes
No
S440
Perform H/W training (Coarse training)
S450
Perform S/W training (Fine training)
S460
Store trained parameters of fine training into storage
End