(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0036972
(43) 공개일자 2022년03월23일

(51) 국제특허분류(Int. Cl.)
G06F 12/109 (2016.01) *G06F 12/02* (2018.01)
G06F 12/1009 (2016.01) *G06F 9/54* (2018.01)
(52) CPC특허분류
G06F 12/109 (2013.01)
G06F 12/0246 (2013.01)
(21) 출원번호 10-2022-7005983
(22) 출원일자(국제) 2020년07월22일
심사청구일자 없음
(85) 번역문제출일자 2022년02월22일
(86) 국제출원번호 PCT/IB2020/056906
(87) 국제공개번호 WO 2021/014382
국제공개일자 2021년01월28일
(30) 우선권주장
16/519,311 2019년07월23일 미국(US)

(71) 출원인
에이티아이 테크놀로지스 유엘씨
캐나다 온타리오 엘3티 7엑스6 마크햄 커머스 밸리 드라이브 이스트 1
(72) 발명자
췽 퉁 추엔
캐나다 온타리오 엘3티 7엑스6 마크햄 커머스 밸리 드라이브 이스트 1
찬 벤자민 쿤 판
캐나다 온타리오 엘3티 7엑스6 마크햄 커머스 밸리 드라이브 이스트 1
앳킨슨 윌리엄 로이드
캐나다 온타리오 엘3티 7엑스6 마크햄 커머스 밸리 드라이브 이스트 1
(74) 대리인
박장원

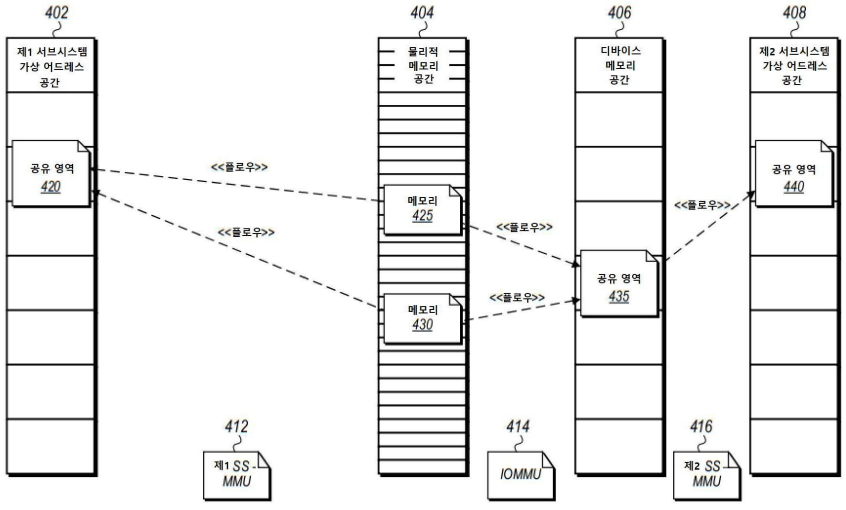
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 **이종 컴퓨팅을 위한 통합 커널 가상 어드레스 공간**

(57) 요 약

이종 컴퓨팅을 위한 통합 커널 가상 어드레스 공간을 구현하기 위한 시스템들, 장치들 및 방법들이 개시된다. 시스템은 적어도 제1 커널을 실행하는(running) 제1 서브시스템, 입력/출력 메모리 관리 유닛(IOMMU), 및 제2 커널을 실행하는 제2 서브시스템을 포함한다. 2개의 서브시스템들 사이에서 메모리 버퍼를 공유하기 위해, 제1 서브시스템은 제1 서브시스템에 의해 제어되는 시스템 메모리의 일부에 메모리의 블록을 할당한다. 제1 서브시스템의 커널 어드레스 공간의 제1 논리 어드레스로부터 메모리의 블록으로의 제1 매핑이 생성된다. 그 후, IOMMU는 제2 서브시스템의 커널 어드레스 공간의 제2 논리 어드레스로부터 메모리의 그 블록의 물리적 어드레스를 매핑하기 위한 제2 매핑을 생성한다. 이러한 매핑은 제1 및 제2 서브시스템이 메모리의 블록을 참조하는 버퍼 포인터를 공유할 수 있게 한다.

대 표 도



(52) CPC특허분류
G06F 12/1009 (2013.01)
G06F 9/544 (2013.01)
G06F 9/545 (2013.01)
G06F 2212/657 (2013.01)

명 세 서

청구범위

청구항 1

시스템에 있어서,

메모리 서브시스템;

제1 커널 어드레스 공간을 갖는 제1 서브시스템;

제2 커널 어드레스 공간을 갖는 제2 서브시스템; 및

입력/출력 메모리 관리 유닛(IOMMU)을 포함하고;

상기 시스템은

상기 제1 서브시스템에 의해, 상기 메모리 서브시스템에 대응하는 물리적 어드레스 공간 내의 제1 물리적 어드레스에서 메모리의 블록을 할당하고;

상기 제1 서브시스템에 의해, 상기 제1 커널 어드레스 공간 내의 제1 논리 어드레스의 상기 제1 물리적 어드레스로의 매핑을 생성하고 - 상기 제1 커널 어드레스 공간 내의 상기 제1 논리 어드레스는 상기 제1 물리적 어드레스로부터 떨어진 제1 오프셋임 - ;

상기 IOMMU에 의해, 상기 제2 커널 어드레스 공간에서 상기 제1 논리 어드레스로부터 떨어진 제2 오프셋인 디바이스 어드레스를 선택하고;

상기 디바이스 어드레스 공간에서 상기 제1 물리적 어드레스에 대한 상기 선택된 디바이스 어드레스의 IOMMU 매핑을 생성하고; 및

상기 IOMMU 매핑을 사용하여, 상기 제1 서브시스템 및 상기 제2 서브시스템 모두에 의해 사용하기 위해 상기 제1 커널 어드레스 공간 및 상기 제2 커널 어드레스 공간 모두에서 공통 공유 영역을 인에이블하도록 구성되는, 시스템.

청구항 2

제1항에 있어서, 상기 제1 서브시스템은 상기 메모리의 블록을 가리키는 버퍼 포인터를 상기 제2 서브시스템에 전달하도록 구성되고, 상기 버퍼 포인터는 상기 제1 커널 어드레스 공간에서 특정되는, 시스템.

청구항 3

제2항에 있어서, 상기 시스템은,

상기 제2 서브시스템에 의해, 상기 버퍼 포인터의 변환을 수행하지 않고 상기 제1 커널 어드레스 공간에서 버퍼 포인터를 참조하는 메모리 액세스 명령을 실행하고; 및

상기 IOMMU 매핑을 사용하여, 상기 제2 서브시스템에 의해 상기 메모리의 블록에 액세스할 때 상기 버퍼 포인터를 상기 물리적 어드레스 공간으로 변환하도록 더 구성되는, 시스템.

청구항 4

제1항에 있어서,

상기 제1 커널 어드레스 공간 내의 논리 어드레스들은 상기 물리적 어드레스 공간 내의 대응하는 어드레스들로부터의 제1 오프셋이고;

상기 제2 커널 어드레스 공간 내의 논리 어드레스들은 디바이스 어드레스 공간 내의 대응하는 어드레스들로부터의 제2 오프셋이고; 및

상기 IOMMU 매핑은 상기 메모리의 블록을 가리키는 상기 제2 커널 어드레스 공간에서의 논리 어드레스들이 상기

제1 커널 어드레스 공간에서의 논리 어드레스들과 동일하도록 오버레이를 생성하는, 시스템.

청구항 5

제1항에 있어서,

상기 제1 서브시스템은 제1 운영 체제를 실행하고; 및

상기 제2 서브시스템은 상기 제1 운영 체제와 상이한 제2 운영 체제를 실행하는, 시스템.

청구항 6

제5항에 있어서,

상기 제1 서브시스템은 제1 메모리 관리 유닛(MMU)을 포함하고; 및

상기 제2 서브시스템은 상기 제1 MMU와 상이한 제2 MMU를 포함하는, 시스템.

청구항 7

제1항에 있어서, 상기 공통 공유 영역은 상기 물리적 어드레스 공간 내의 상기 메모리의 블록에 매핑되는, 시스템.

청구항 8

방법에 있어서,

제1 커널 어드레스 공간을 갖는 제1 서브시스템에 의해, 물리적 어드레스 공간 내의 제1 물리적 어드레스에 메모리의 블록을 할당하는 단계;

상기 제1 서브시스템에 의해, 상기 제1 커널 어드레스 공간 내의 제1 논리 어드레스의 상기 제1 물리적 어드레스로의 매핑을 생성하는 단계 - 상기 제1 커널 어드레스 공간 내의 상기 제1 논리 어드레스는 상기 제1 물리적 어드레스로부터 떨어진 제1 오프셋임 - ;

입력/출력 메모리 관리 유닛(IOMMU)에 의해, 제2 서브시스템의 제2 커널 어드레스 공간에서 상기 제1 논리 어드레스로부터 떨어진 제2 오프셋인 디바이스 어드레스를 선택하는 단계;

상기 IOMMU에 의해, 상기 디바이스 어드레스 공간에서 상기 선택된 디바이스 어드레스의 상기 제1 물리적 어드레스로의 IOMMU 매핑을 생성하는 단계; 및

상기 IOMMU 매핑을 사용하여, 상기 제1 서브시스템 및 상기 제2 서브시스템 모두에 의해 사용하기 위해 상기 제1 커널 어드레스 공간 및 상기 제2 커널 어드레스 공간 모두에서 공통 공유 영역을 인에이블하는 단계를 포함하는, 방법.

청구항 9

제8항에 있어서, 상기 제1 서브시스템으로부터 상기 제2 서브시스템으로, 상기 메모리의 블록을 가리키는 버퍼 포인터를 전달하는 단계를 더 포함하고, 상기 버퍼 포인터는 상기 제1 커널 어드레스 공간에서 특정되는, 방법.

청구항 10

제9항에 있어서,

상기 제2 서브시스템에 의해, 상기 버퍼 포인터의 변환을 수행하지 않고 상기 제1 커널 어드레스 공간에서 버퍼 포인터를 참조하는 메모리 액세스 명령을 실행하는 단계; 및

상기 IOMMU 매핑을 사용하여, 상기 제2 서브시스템에 의해 상기 메모리의 블록에 액세스할 때 상기 버퍼 포인터를 상기 물리적 어드레스 공간으로 변환하는 단계를 더 포함하는, 방법.

청구항 11

제8항에 있어서,

상기 제1 커널 어드레스 공간 내의 논리 어드레스들은 상기 물리적 어드레스 공간 내의 대응하는 어드레스들로

부터의 제1 오프셋이고;

상기 제2 커널 어드레스 공간 내의 논리 어드레스들은 디바이스 어드레스 공간 내의 대응하는 어드레스들로부터의 제2 오프셋이고;

상기 방법은, 상기 IOMMU 매핑에 의해, 상기 메모리의 블록을 가리키는 상기 제2 커널 어드레스 공간에서의 논리 어드레스들이 상기 제1 커널 어드레스 공간에서의 논리 어드레스들과 동일하도록 오버레이를 생성하는 단계를 더 포함하는, 방법.

청구항 12

제8항에 있어서,

상기 제1 서브시스템은 제1 운영 체제를 실행하고; 및

상기 제2 서브시스템은 상기 제1 운영 체제와 상이한 제2 운영 체제를 실행하는, 방법.

청구항 13

제12항에 있어서,

상기 제1 서브시스템은 제1 메모리 관리 유닛(MMU)을 포함하고; 및

상기 제2 서브시스템은 상기 제1 MMU와 상이한 제2 MMU를 포함하는, 방법.

청구항 14

제8항에 있어서, 상기 공통 공유 영역은 상기 물리적 어드레스 공간 내의 상기 메모리의 블록에 매핑되는, 방법.

청구항 15

장치에 있어서,

프로그램 명령어들을 저장하는 메모리; 및

상기 메모리에 결합된 적어도 하나의 프로세서로서, 상기 프로그램 명령어들은 상기 적어도 하나의 프로세서에 의해:

제1 커널 어드레스 공간을 갖는 제1 서브시스템에 의해, 물리적 어드레스 공간 내의 제1 물리적 어드레스에 메모리의 블록을 할당하고;

상기 제1 서브시스템에 의해, 상기 제1 커널 어드레스 공간 내의 제1 논리 어드레스의 상기 제1 물리적 어드레스로의 매핑을 생성하고 - 상기 제1 커널 어드레스 공간 내의 상기 제1 논리 어드레스는 상기 제1 물리적 어드레스로부터 떨어진 제1 오프셋임 - ;

입력/출력 메모리 관리 유닛(IOMMU)에 의해, 제2 서브시스템의 제2 커널 어드레스 공간에서 상기 제1 논리 어드레스로부터 떨어진 제2 오프셋인 디바이스 어드레스를 선택하고;

상기 IOMMU에 의해, 상기 디바이스 어드레스 공간에서 상기 선택된 디바이스 어드레스의 상기 제1 물리적 어드레스로의 IOMMU 매핑을 생성하고; 및

상기 IOMMU 매핑을 사용하여, 상기 제1 서브시스템 및 상기 제2 서브시스템 모두에 의해 사용하기 위해 상기 제1 커널 어드레스 공간 및 상기 제2 커널 어드레스 공간 모두에서 공통 공유 영역을 인에이블하도록 실행가능한, 장치.

청구항 16

제15항에 있어서, 상기 프로그램 명령어들은 상기 제1 서브시스템으로부터 상기 제2 서브시스템으로, 상기 메모리의 블록을 가리키는 버퍼 포인터를 전달하도록 상기 적어도 하나의 프로세서에 의해 실행가능하고, 상기 버퍼 포인터는 상기 제1 커널 어드레스 공간에서 특정되는, 장치.

청구항 17

제16항에 있어서, 상기 프로그램 명령어들은, 상기 적어도 하나의 프로세서에 의해,

상기 제2 서브시스템에 의해, 상기 버퍼 포인터의 변환을 수행하지 않고 상기 제1 커널 어드레스 공간에서 버퍼 포인터를 참조하는 메모리 액세스 명령을 실행하고; 및

상기 IOMMU 매핑을 사용하여, 상기 제2 서브시스템에 의해 상기 메모리의 블록에 액세스할 때 상기 버퍼 포인터를 상기 물리적 어드레스 공간으로 변환하도록 실행가능한, 장치.

청구항 18

제15항에 있어서,

상기 제1 커널 어드레스 공간 내의 논리 어드레스들은 상기 물리적 어드레스 공간 내의 대응하는 어드레스들로부터의 제1 오프셋이고;

상기 제2 커널 어드레스 공간 내의 논리 어드레스들은 디바이스 어드레스 공간 내의 대응하는 어드레스들로부터의 제2 오프셋이고; 및

상기 IOMMU 매핑은 상기 메모리의 블록을 가리키는 상기 제2 커널 어드레스 공간에서의 논리 어드레스들이 상기 제1 커널 어드레스 공간에서의 논리 어드레스들과 동일하도록 오버레이를 생성하는, 장치.

청구항 19

제15항에 있어서,

상기 제1 서브시스템은 제1 운영 체제를 실행하고; 및

상기 제2 서브시스템은 상기 제1 운영 체제와 상이한 제2 운영 체제를 실행하는, 장치.

청구항 20

제19항에 있어서,

상기 제1 서브시스템은 제1 메모리 관리 유닛(MMU)을 포함하고; 및

상기 제2 서브시스템은 상기 제1 MMU와 상이한 제2 MMU를 포함하는, 장치.

발명의 설명

기 술 분 야

배 경 기 술

[0001] 전형적으로, 컴퓨팅 디바이스들 사이의 메모리 버퍼들의 공유는 디바이스들이 디바이스 어드레스 공간 또는 물리적 어드레스 공간에서 포인터 또는 핸들(handle)을 교환할 것을 요구한다.

[0002] 대부분의 경우, 이 포인터를 사용하기 전에, 컴퓨팅 디바이스는 메모리 관리 유닛(MMU) 페이지 테이블 매핑(mapping)을 사용하여 포인터를 로컬 가상 어드레스 공간으로 변환한다. 이것은 포인터 자체 또는 포인터를 포함하는 구조의 공유를 비효율적이고 어렵게 만든다.

발명의 내용

도면의 간단한 설명

[0003] 본 명세서에 설명된 방법 및 메커니즘의 장점은 첨부 도면과 함께 다음 설명을 참조하여 더 잘 이해될 수 있다.

도 1은 컴퓨팅 시스템의 일 구현예의 블록도이다.

도 2는 이종(heterogeneous) 컴퓨팅을 위한 통합 커널 가상 어드레스 공간을 생성하는 일 구현예의 도면이다.

도 3은 2개의 개별 서브시스템들 사이의 공유 버퍼들의 일 구현예의 도면이다.

도 4는 메모리의 영역들을 다수의 커널 어드레스 공간들에 매핑하는 일 구현예의 도면이다.

도 5는 통합 커널 가상 어드레스 공간을 생성하기 위한 방법의 일 구현예를 예시하는 일반화된 흐름도이다.

도 6은 다수의 커널 어드레스 공간들에서 공통 공유 영역을 인에이블하기 위한 방법의 일 구현예를 예시하는 일반화된 흐름도이다.

도 7은 2개의 상이한 서브시스템들 사이에서 버퍼 포인터 공유를 인에이블하기 위한 방법의 일 구현예를 예시하는 일반화된 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0004] 다음 설명에서는 본 명세서에 제시된 방법 및 메커니즘에 대한 철저한 이해를 제공하기 위해 수많은 특정 세부사항이 설명된다. 그러나, 당업자는 이러한 특정 세부사항 없이 다양한 구현예가 실시될 수 있음을 인식해야 한다. 일부 예에서, 공지의 구조, 컴포넌트, 신호, 컴퓨터 프로그램 명령어 및 기술은 본 명세서에 설명된 접근 방식을 모호하게 하는 것을 피하기 위해 자세히 표시되지 않았다. 예시의 단순성과 명료성을 위해, 도면에 도시된 엘리먼트는 반드시 축척대로 그려지지는 않았다는 것이 이해될 것이다. 예를 들어, 일부 엘리먼트의 치수는 다른 엘리먼트에 비해 과장될 수 있다.

[0005] 이종 컴퓨팅을 위한 통합 커널 가상 어드레스 공간을 구현하기 위한 시스템들, 장치들 및 방법들이 본 명세서에 개시된다. 일 구현예에서, 시스템은 적어도 제1 커널을 실행하는 제1 서브시스템, 입력/출력 메모리 관리 유닛(IOMMU), 및 제2 커널을 실행하는 제2 서브시스템을 포함한다. 일 구현예에서, IOMMU는 제1및 제2서브시스템이 커널 레벨에서 메모리 버퍼를 공유할 수 있도록 하는 통합 커널 어드레스 공간을 생성한다. 2개의 서브시스템들 사이에서 메모리 버퍼를 공유하기 위해, 제1 서브시스템은 제1 서브시스템에 의해 제어되는 시스템 메모리의 일부에 메모리의 블록을 할당한다. 제1 서브시스템의 제1 커널 어드레스 공간의 제1 논리 어드레스로부터 메모리의 블록으로의 제1 매핑이 생성된다. 그 후, IOMMU는 제2 서브시스템의 제2 커널 어드레스 공간의 제2 논리 어드레스로부터 메모리의 그 블록의 물리적 어드레스를 매핑하기 위한 제2 매핑을 생성한다. 이러한 매핑은 제1 및 제2 서브시스템이 메모리의 블록을 참조하는 커널 어드레스 공간의 버퍼 포인터를 공유할 수 있게 한다.

[0006] 도 1을 참조하면, 컴퓨팅 시스템(100)의 일 구현예의 블록도가 도시되어 있다. 일 구현예에서, 컴퓨팅 시스템(100)은 적어도 제1 서브시스템(110), 제2 서브시스템(115), 입력/출력(I/O) 인터페이스(들)(120), 입력/출력 메모리 관리 유닛(IOMMU)(125), 메모리 서브시스템(130), 및 주변 디바이스(들)(135)를 포함한다. 다른 구현예들에서, 컴퓨팅 시스템(100)은 다른 컴포넌트들을 포함할 수 있고 및/또는 컴퓨팅 시스템(100)은 상이하게 배열될 수 있다.

[0007] 일 구현예에서, 제1 및 제2 서브시스템들(110 및 115)은 상이한 커널 어드레스 공간들을 갖지만, 통합 커널 어드레스 공간은 제1 및 제2 서브시스템들(110 및 115)에 대한 IOMMU(125)에 의해 생성된다. 통합 커널 어드레스 공간은 제1 및 제2 서브시스템(110, 115)이 서로 간에 포인터를 전달하고 버퍼를 공유할 수 있게 한다. 제1 서브시스템(110) 및 제2 서브시스템(115)에 대해, 그들 각각의 커널 어드레스 공간은 커널 논리 어드레스 및 커널 가상 어드레스를 포함한다. 일부 아키텍처에서, 커널 논리 어드레스 및 그 연관된 물리적 어드레스는 일정한 오프셋만큼 상이하다. 커널 가상 어드레스들은 커널 논리 어드레스들을 특징짓는 물리적 어드레스들에 대한 선형, 일대일 매핑을 반드시 가질 필요는 없다. 모든 커널 논리 어드레스는 커널 가상 어드레스이지만 커널 가상 어드레스가 반드시 커널 논리 어드레스일 필요는 없다.

[0008] 일 구현예에서, 제1 서브시스템(110) 및 제2 서브시스템(115) 각각은 운영 체제를 실행하는 하나 이상의 프로세서를 포함한다. 프로세서(들)는 또한 다양한 구현예들에서 하나 이상의 소프트웨어 프로그램들을 실행한다. 제1 서브시스템(110) 및 제2 서브시스템(115)의 프로세서(들)는 임의의 수 및 유형의 프로세싱 유닛(예를 들어, 중앙 프로세싱 유닛(CPU), 그래픽 프로세싱 유닛(GPU), 디지털 신호 프로세서(DSP), 필드 프로그래머블 게이트 어레이(FPGA), 주문형 집적 회로(ASIC))을 포함한다. 또한, 제1 서브시스템(110)은 메모리 관리 유닛(MMU)(112)을 포함하고, 제2 서브시스템(115)은 MMU(117)를 포함하며, 각각의 MMU는 그의 대응하는 서브시스템에 대한 가상에서 물리적 어드레스로 변환(translation)을 처리한다. 제1 및 제2 서브시스템들(110 및 115)이 상이한 커널 어드레스 공간들을 갖지만, 통합 커널 어드레스 공간이 제1 및 제2 서브시스템들(110 및 115)에 대해 IOMMU(125)에 의해 생성되어, 제1 및 제2 서브시스템들(110 및 115)이 포인터들을 앞뒤로 전달하고 메모리 서브시스템(130)에서 버퍼들을 공유할 수 있게 한다.

[0009] 메모리 서브시스템(130)은 임의의 수 및 유형의 메모리 디바이스를 포함한다. 예를 들어, 메모리 서브시스템

(130) 내의 메모리의 유형은 고-대역폭 메모리(HBM), 비휘발성 메모리(NVM), 동적 랜덤 액세스 메모리(DRAM), 정적 랜덤 액세스 메모리(SRAM), NAND 플래시 메모리, NOR 플래시 메모리, 강유전성 랜덤 액세스 메모리(FeRAM), 또는 다른 것들을 포함할 수 있다. I/O 인터페이스들(120)은 임의의 수 및 유형의 I/O 인터페이스들(예를 들어, 주변 컴포넌트 상호접속(PCI) 버스, PCI-확장(PCI-X), PCIE((PCI Express) 버스, 기가비트 이더넷(GBE) 버스, 범용 직렬 버스(USB))을 나타낸다. 다양한 유형의 주변 디바이스(135)가 I/O 인터페이스(120)에 결합될 수 있다. 이러한 주변 디바이스들(135)은 디스플레이들, 키보드들, 마우스들, 프린터들, 스캐너들, 조이스틱들 또는 다른 유형들의 게임 제어기들, 미디어 기록 디바이스들, 외부 저장 디바이스들, 네트워크 인터페이스 카드들 등을 포함한다(그러나 이들로 제한되지 않음).

[0010] 일 구현예에서, 제1 서브시스템(110)과 제2 서브시스템(115) 사이에 공유되는 메모리 공간을 생성하기 위해, 메모리의 블록이 제1 서브시스템(110)에 의해 관리되는 시스템 메모리의 일부에 할당된다. 메모리의 초기 블록이 할당된 후에, 적절한 I/O 가상 어드레스(VA)가 제2 서브시스템(115)에 할당된다. 일 구현예에서, IOMMU 매핑은 제2 서브시스템(115)의 커널 어드레스 공간으로부터 메모리의 블록의 물리적 어드레스로 생성된다. 이 구현예에서, IOMMU(125)는 메모리의 블록에 대한 제2 서브시스템(115)에 대한 가상 어드레스 매핑을 수행한다. 그런 다음, 추가 메모리가 할당되면, 힙(heap) 할당 함수가 호출되고, 이전에 생성된 동일한 I/O VA 어드레스를 기반으로 어드레스가 매핑된다. 그 후, 통합 어드레스를 제2 서브시스템(115)에 통지하는 메시지가 제2 서브시스템(115)으로 전송된다.

[0011] 다양한 구현예들에서, 컴퓨팅 시스템(100)은 컴퓨터, 랩탑, 모바일 디바이스, 게임 콘솔, 서버, 스트리밍 디바이스, 웨어러블 디바이스, 또는 다양한 다른 유형들의 컴퓨팅 시스템들 또는 디바이스들 중 임의의 것이다. 컴퓨팅 시스템(100)의 컴포넌트들의 수는 구현예마다 달라짐에 유의한다. 다른 구현예들에서, 도 1에 도시된 수보다 더 많거나 더 적은 각각의 컴포넌트가 있다. 또한, 다른 구현예들에서, 컴퓨팅 시스템(100)은 도 1에 도시되지 않은 다른 컴포넌트들을 포함한다. 추가적으로, 다른 구현예들에서, 컴퓨팅 시스템(100)은 도 1에 도시된 것 이외의 다른 방식들로 구성된다.

[0012] 이제 도 2를 참조하면, 이종 컴퓨팅을 위한 통합 커널 가상 어드레스 공간을 생성하는 일 구현예의 다이어그램이 도시된다. 도 2의 상부로부터 좌측에서 우측으로, 점선의 수직선은 제1 서브시스템(202), 제1 서브시스템(202)의 MMU(204), 제1 서브시스템(202) 상에서 실행되는 디바이스 드라이버(206), 메모리의 공유 영역(208), IOMMU(211), 제2 서브시스템(214)의 MMU(212), 및 제2 서브시스템(214)을 나타낸다. 본 논의의 목적을 위해, 제1 서브시스템(202)은 제1 운영 체제를 갖고, 제2 서브시스템(214)은 제2 운영 체제를 가지며, 제2 운영 체제는 제1 운영 체제와 상이한 것으로 가정된다. 또한, 본 논의의 목적들을 위해, 제1 서브시스템(202) 및 제2 서브시스템(214)은 이종 컴퓨팅 시스템의 일부라고 가정된다. 일 구현예에서, 이러한 이종 컴퓨팅 시스템에 대해, 제1 서브시스템(202) 및 제2 서브시스템(214)은 각각 워크로드(workload)의 일부를 수행하고 있고, 워크로드를 수행하기 위해, 제1 서브시스템(202) 및 제2 서브시스템(214)은 서로 간에 버퍼 포인터들 및 버퍼들을 공유한다. 도 2의 다이어그램은 제1 서브시스템(202)과 제2 서브시스템(214) 사이에 공유될 메모리를 할당하는 일 예를 도시한다.

[0013] 일 구현예에서, 제1 단계(205)는 힙(heap)(즉, 공유 메모리 영역)을 생성하기 위해 디바이스 드라이버(206)에 의해 수행된다. 본 명세서에서 사용되는 바와 같이, "힙(heap)"이라는 용어는 물리적 메모리 풀(pool)에 매핑되는 가상 메모리 풀로서 정의된다. 다음으로, 단계(210)에서, 원하는 크기의 힙이 물리적 메모리 서브시스템에 할당된다. 단계(215)에서, 커널로부터 힙으로의 매핑이 생성된다. 일 구현예에서, 캐버아웃(carveout) 힙이 처음 생성될 때, 새로운 플래그는 힙이 커널 논리 어드레스 공간으로부터 와야 하는지를 나타낸다. 예를 들어, kmalloc 함수는 커널 논리 어드레스 공간에서 메모리를 리턴한다. 일 구현예에서, Linux® 운영 체제에 대해, 메모리 관리자는 Linux 젠풀 라이브러리를 사용하여 힙에 대해 할당된 버퍼를 관리한다. 이 구현예에서, 버퍼가 할당될 때, 버퍼는 내부 풀에 마킹되고 물리적 어드레스가 리턴된다. 그런 다음 캐버아웃 힙은 이 물리적 어드레스를 버퍼 및 sg_table 설명자로 래핑(wrap)한다. 일 구현예에서, 버퍼가 커널 어드레스 공간에 매핑될 때, heap_map_kernel 함수는 단계(220)에서 vmap 대신에 kmap을 사용하여 버퍼를 매핑한다. 함수 kmap은 논리적 매핑에 기초하여 버퍼를 주어진 가상 어드레스에 매핑한다. kmap 또는 vmap의 선택은 캐버아웃 힙 생성 동안 주어진 새로운 플래그에 의해 제어된다. 또는, 젠풀(Genpool) 라이브러리, 캐버아웃 API(application programming interface), 새로운 플래그를 이용하여 커널 매핑과 함께 힙이 생성된다. 이 경우, 커널 맵은 미리 매핑된 어드레스를 리턴한다. 사용자 모드 매핑은 계속 즉시 적용된다. 제2 서브시스템(214)에 대해, 제1 서브시스템(202)에 의해 할당된 임의의 공유 메모리 버퍼는 캐버아웃 힙 래핑된 젠풀과 유사하게 관리된다. 새로운 API를 사용하면 외부 버퍼를 캐버아웃 힙에 추가할 수 있다. 제2 서브시스템(214) 상의 로컬 태스크들은 동일한 API를 사

용하여 이 캐버아웃 힙으로부터 할당한다. 단계(220) 이후에, 공유 메모리 영역의 할당이 완료된다.

[0014] 다음으로, 단계(225)에서, 입력/출력(I/O) 가상 어드레스가 공유 메모리 영역에 대해 IOMMU(211)에 의해 할당된다. 그 후, 단계(230)에서, DMA 어드레스 공간의 연속 블록이 공유 메모리 영역을 위해 예약된다. 다음으로, 단계(235)에서, 공유 메모리 영역은 IOMMU(211)에 의해 제2 서브시스템(214)의 커널 어드레스 공간에 매핑된다. IOMMU 매핑은 디바이스 드라이버(206)가 셧다운될 때까지 해제되어서는 안 된다는 점에 유의한다. 제1 서브시스템(202)의 커널 어드레스 공간으로부터 공유 메모리 영역으로의 매핑은 단계(240)에서 유효하지 않게 된다. 그 후, 단계(245)에서, 커널 어드레스 공간 매핑은 메모리 릴리스 함수를 실행함으로써 해제(freed)된다.

[0015] 이제 도 3을 참조하면, 2개의 개별 서브시스템들 사이의 공유 버퍼들의 일 구현예의 다이어그램이 도시된다. 도 3의 상부로부터 아래로, 좌측에서 우측으로 연장하는 점선의 수직선들은 제 1 서브시스템(302), 제 1 서브시스템(302)의 MMU(304), 제 1 서브시스템(302) 상에서 실행하는 디바이스 드라이버(306), 메모리의 공유 영역(308), IOMMU(310), 제 2 서브시스템(314)의 MMU(312), 및 제 2 서브시스템(314)을 나타낸다. 본 논의의 목적을 위해, 제1 서브시스템(302)은 제1 운영 체제를 갖고, 제2 서브시스템(314)은 제2 운영 체제를 가지며, 제2 운영 체제는 제1 운영 체제와 상이한 것으로 가정된다.

[0016] 루프(Loop) 교환(305)의 경우, 메모리의 블록이 할당되고, 제1 서브시스템(302)의 커널 어드레스 공간으로부터 메모리 블록의 물리적 어드레스로의 매핑이 디바이스 드라이버(306)에 의해 생성된다. 그 후, 매핑은 MMU(304)에 의해 유지된다. 메모리 블록은 제2 서브시스템(314)에 배타적으로 할당되거나, 제1 서브시스템(302)과 제2 서브시스템(314) 사이에 공유되거나, 제1 서브시스템(302)에 배타적으로 할당될 수 있다. 그 후, 메모리의 블록의 어드레스 및 크기와 함께 메시지가 제1 서브시스템(302)으로부터 제2 서브시스템(314)으로 전송된다. 일 구현예에서, 메시지는 대역외(out-of-band)로 전송될 수 있다. 메시지가 제 2 서브시스템(314)에 의해 수신된 후, 제 2 서브시스템(314)의 커널 어드레스 공간으로부터 메모리 블록의 물리적 어드레스로의 매핑이 생성되고 MMU(312)에 의해 유지된다. 루프 교환(315)의 경우, 데이터는 공유 영역(308)을 사용하여 제1 서브시스템(302)과 제2 서브시스템(314) 사이에서 교환된다. 통합 커널 가상 어드레스로, 버퍼 포인터들(1^{st}_SS_buf 및 2^{nd}_SS_buf)은 동일하고, 제1 서브시스템(302)과 제2 서브시스템(314) 사이에서 자유롭게 교환될 수 있고, 또한 젠풀 라이브러리를 사용하여 분할(partitioned)될 수 있다.

[0017] 이제 도 4를 참조하면, 다수의 커널 어드레스 공간들에 대한 메모리의 매핑 영역들의 일 구현예의 다이어그램이 도시된다. 일 구현예에서, 이종 컴퓨팅 시스템(예를 들어, 도 1의 시스템(100))은 그 자신의 독립적인 운영 체제들을 갖는 다수의 상이한 서브시스템들을 포함한다. 도 4에 도시된 수직으로 배향된 직사각형 블록들은 이종 컴퓨팅 시스템의 상이한 컴포넌트들에 대한 어드레스 공간들을 나타낸다. 좌측에서 우측으로, 도 4에 도시된 어드레스 공간은 제1 서브시스템 가상 어드레스 공간(402), 물리적 메모리 공간(404), 디바이스 메모리 공간(406), 및 제2 서브시스템 가상 어드레스 공간(408)이다.

[0018] 일 구현예에서, 제1 서브시스템 가상 어드레스 공간(402)은 물리적 메모리 공간(404)의 메모리 블록(425) 및 메모리 블록(430)에 매핑되는 공유 영역(420)을 포함한다. 일 구현예에서, 메모리 블록(425) 및 메모리 블록(430)에 대한 공유 영역(420)의 매핑은 제 1 서브시스템 MMU(412)에 의해 생성되고 유지된다. 공유 영역(420)이 제 2 서브시스템과 공유되도록 허용하기 위해, 메모리 블록(425) 및 메모리 블록(430)은 IOMMU(414)에 의해 디바이스 메모리 공간(406)의 공유 영역(435)에 매핑된다. 그 후, 공유 영역(435)은 제 2 서브시스템 MMU(416)에 의해 제 2 서브시스템 가상 어드레스 공간(408)의 공유 영역(440)에 매핑된다. 이러한 매핑 방식을 통해, 제1 서브시스템과 제2 서브시스템은 커널 어드레스 공간에서 서로 버퍼 포인터를 공유할 수 있다.

[0019] 이제 도 5를 참조하면, 통합 커널 가상 어드레스 공간을 생성하기 위한 방법(500)의 일 구현예가 도시된다. 논의를 위해, 이 구현예에서의 단계들 및 도 6-7의 단계들이 순차적인 순서로 도시된다. 그러나, 설명된 방법들의 다양한 구현예들에서, 설명된 엘리먼트들 중 하나 이상은 도시된 것과 상이한 순서로 동시에 수행되거나, 또는 완전히 생략된다는 것에 유의한다. 다른 추가적인 엘리먼트들이 또한 원하는 대로 수행된다. 본 명세서에 설명된 다양한 시스템들 또는 장치들 중 임의의 것은 방법(500)을 구현하도록 구성된다.

[0020] 제1 서브시스템은 메모리 서브시스템에 대응하는 물리적 어드레스 공간 내의 제1 물리적 어드레스에서 메모리의 블록을 할당한다(블록 505). 일 구현예에서, 제1 서브시스템은 제1 커널 어드레스 공간을 갖는 제1 운영 체제를 실행한다. 다음으로, 제1 서브시스템은 제1 커널 어드레스 공간 내의 제1 논리 어드레스의 제1 물리적 어드레스로의 매핑을 생성한다(블록 510). 제1 커널 어드레스 공간의 제1 논리 어드레스 공간은 제1 물리적 어드레스로부터의 제1 선형 오프셋이라는 것에 유의한다. 그 후, IOMMU는 제2 커널 어드레스 공간의 제2 논리 어드레스를

제1 물리적 어드레스에 대한 IOMMU 매핑을 생성한다(블록 515). 일 구현예에서, 제2 커널 어드레스 공간은 제2 서브시스템과 연관된다. 제2 커널 어드레스 공간의 제2 논리 어드레스 공간은 제1 물리적 어드레스로부터의 제2 선형 오프셋이라는 것에 유의한다.

[0021] 다음으로, 제1 서브시스템은 버퍼 포인터를 제2 서브시스템으로 전달하고, 여기서 버퍼 포인터는 제1 논리 어드레스를 가리킨다(블록 520). 그 후, 제2 서브시스템은 버퍼 포인터로 액세스 요청을 생성하고 액세스 요청을 IOMMU에 전달한다(블록 525). 다음으로, IOMMU는 이전에 생성된 IOMMU 매핑을 사용하여 버퍼 포인터의 가상 어드레스를 제1 물리적 어드레스로 변환(translates)한다(블록 530). 그 후, 제2 서브시스템은 제1 물리적 어드레스에서 메모리의 블록에 액세스한다(블록 535). 블록(535) 후에, 방법(500)은 종료된다.

[0022] 이제 도 6을 참조하면, 다수의 커널 어드레스 공간들에서 공통 공유 영역을 인에이블하기 위한 방법(600)의 일 구현예가 도시된다. 제1 서브시스템은 메모리 서브시스템에 대응하는 물리적 어드레스 공간 내의 제1 물리적 어드레스에서 메모리의 블록을 할당한다(블록 605). 본 논의의 목적을 위해, 방법(600)에 의해 설명된 시스템은 제1 서브시스템 및 제2 서브시스템을 포함한다고 가정된다. 또한, 제1 서브시스템은 제1 커널 어드레스 공간을 갖는 제1 운영 체제를 실행하고, 제2 서브시스템은 제2 커널 어드레스 공간을 갖는 제2 운영 체제를 실행하는 것으로 가정된다. 제1 서브시스템은 제1 커널 어드레스 공간 내의 제1 논리 어드레스의 제1 물리적 어드레스로의 매핑을 생성하며, 여기서 제1 커널 어드레스 공간 내의 제1 논리 어드레스는 제1 물리적 어드레스로부터 떨어진 제1 오프셋이다(블록 610).

[0023] 다음으로, IOMMU는 제2 커널 어드레스 공간에서 제1 논리 어드레스로부터 떨어진 제2 오프셋인 디바이스 어드레스를 선택한다(블록 615). 그 후, IOMMU는 디바이스 어드레스 공간에서 선택된 디바이스 어드레스의 제1 물리적 어드레스로의 IOMMU 매핑을 생성한다(블록 620). IOMMU 매핑은 제1 서브시스템과 제2 서브시스템 모두에 의해 사용하기 위해 제1 커널 어드레스 공간과 제2 커널 어드레스 공간 모두에서 공통 공유 영역을 인에이블하게 한다(블록 625). 블록(625) 후에, 방법(600)은 종료된다.

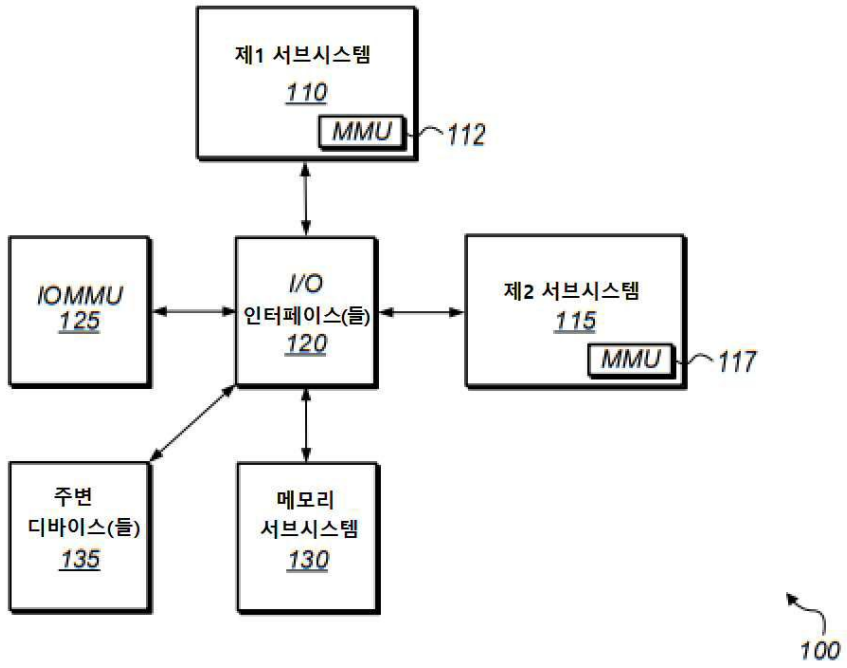
[0024] 이제 도 7을 참조하면, 2개의 상이한 서브시스템들 사이에서 버퍼 포인터 공유를 인에이블하기 위한 방법(700)의 일 구현예가 도시된다. 제1 서브시스템은 제1 커널 어드레스 공간의 제1 영역을 물리적 어드레스 공간의 제2 영역에 매핑한다(블록 705). 일 구현예에서, 물리적 어드레스 공간은 제1 서브시스템에 의해 제어되는 시스템 메모리에 대응한다. 다음으로, IOMMU는 제2 커널 어드레스 공간의 제3 영역을 디바이스 어드레스 공간의 제4 영역에 매핑한다(블록 710). 본 논의의 목적들을 위해, 제2 커널 어드레스 공간은 제1 서브시스템과 상이한 제2 서브시스템에 대응하는 것으로 가정된다. 그 후, IOMMU는 디바이스 어드레스 공간의 제4 영역을 물리적 어드레스 공간의 제2 영역에 매핑하여, 제1 및 제3 영역들에서의 어드레스들이 제2 영역에서의 매칭(즉, 동일한) 어드레스들을 가리키게 한다(블록 715). 제1 서브시스템 및 제2 서브시스템은 물리적 어드레스 공간 내의 제2 영역을 참조하는 제1 또는 제2 커널 어드레스 공간 내의 버퍼 포인터들을 공유할 수 있다(블록 720). 블록(720) 이후에, 방법(700)은 종료된다.

[0025] 다양한 구현예들에서, 소프트웨어 애플리케이션의 프로그램 명령어들은 본 명세서에 설명된 방법들 및/또는 메커니즘들을 구현하기 위해 사용된다. 예를 들어, 범용 또는 특수 목적 프로세서에 의해 실행 가능한 프로그램 명령어들이 고려된다. 다양한 구현예들에서, 이러한 프로그램 명령어들은 고급(high level) 프로그래밍 언어로 표현될 수 있다. 다른 구현예들에서, 프로그램 명령어들은 고급 프로그래밍 언어로부터 이진, 중간, 또는 다른 형태로 컴파일될 수 있다. 대안적으로, 하드웨어의 동작 또는 설계를 설명하는 프로그램 명령어들이 기록될 수 있다. 이러한 프로그램 명령어들은 C와 같은 고급 프로그래밍 언어로 표현될 수 있다. 대안적으로, Verilog와 같은 하드웨어 설계 언어(HDL)가 사용될 수 있다. 다양한 구현예들에서, 프로그램 명령어들은 다양한 비일시적 컴퓨터 판독가능 저장 매체들 중 임의의 것 상에 저장된다. 저장 매체는 프로그램 실행을 위해 프로그램 명령어들을 컴퓨팅 시스템에 제공하기 위해 사용 동안 컴퓨팅 시스템에 의해 액세스가능하다. 일반적으로 말하면, 이러한 컴퓨팅 시스템은 적어도 하나 이상의 메모리 및 프로그램 명령어를 실행하도록 구성된 하나 이상의 프로세서를 포함한다.

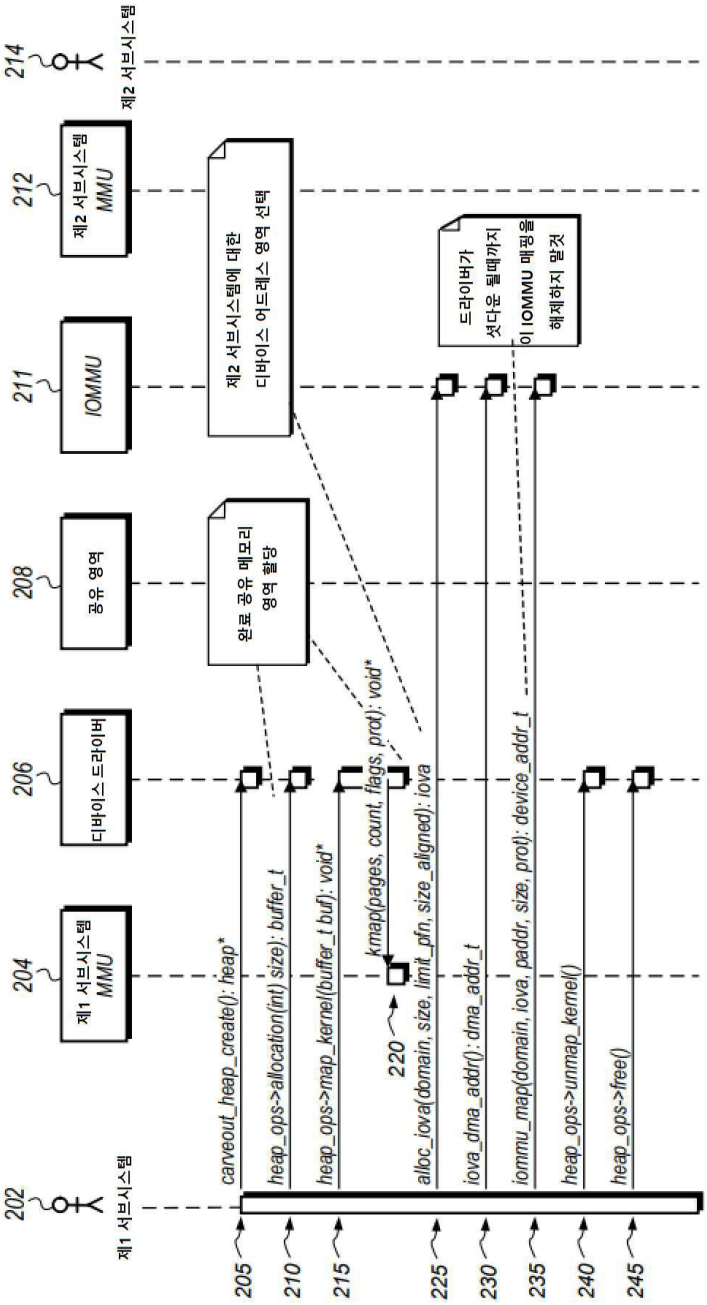
[0026] 상술한 구현예들은 구현예들의 비제한적인 예일뿐이라는 것이 강조되어야 한다. 상기 개시 내용이 충분히 이해되면 수많은 변형 및 수정들이 당업자에게 명백해질 것이다. 다음 청구범위는 그러한 모든 변형 및 수정을 포함하는 것으로 해석되어야 한다.

도면

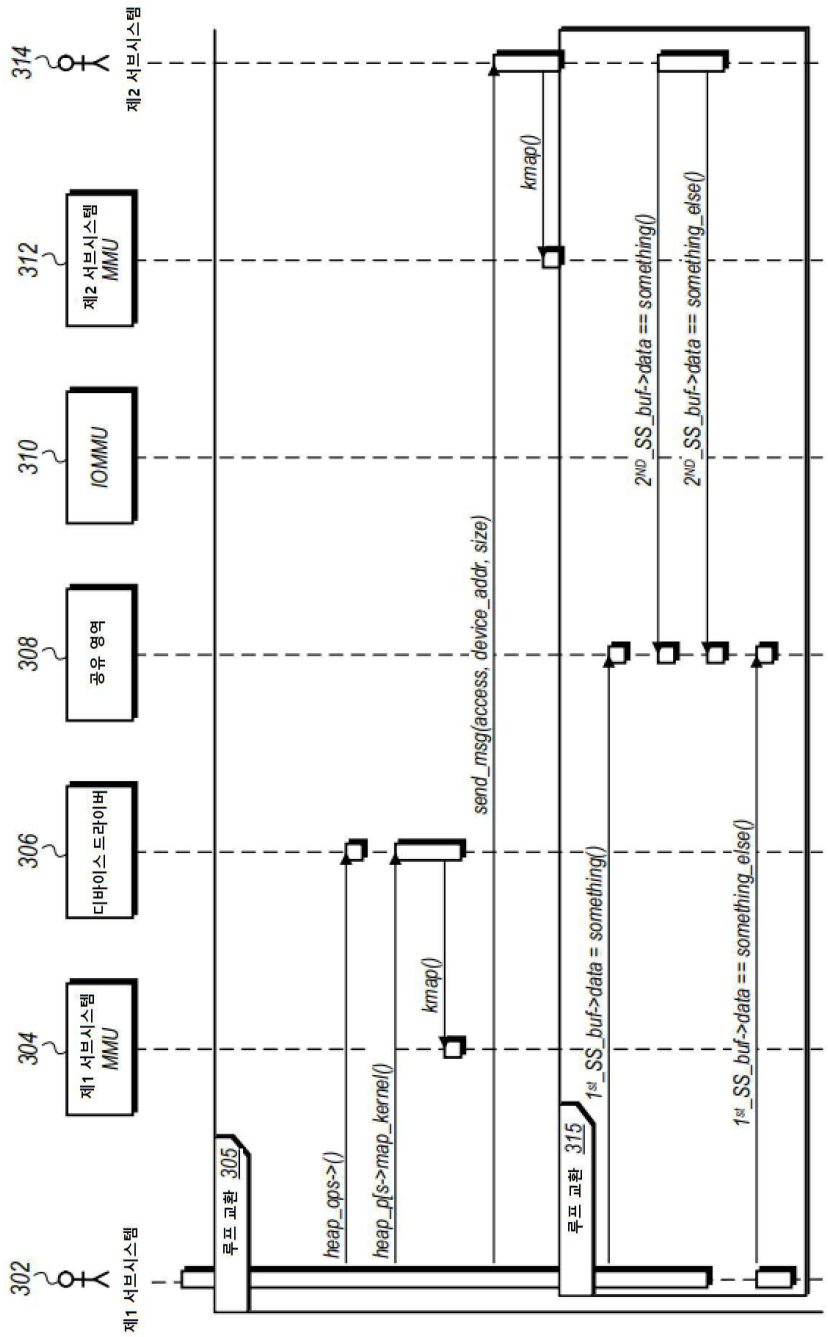
도면1


제1 서브시스템
110
MMU
112
IOMMU
125
I/O
인터페이스(들)
120
제2 서브시스템
115
MMU
117
주변
디바이스(들)
135
메모리
서브시스템
130
100

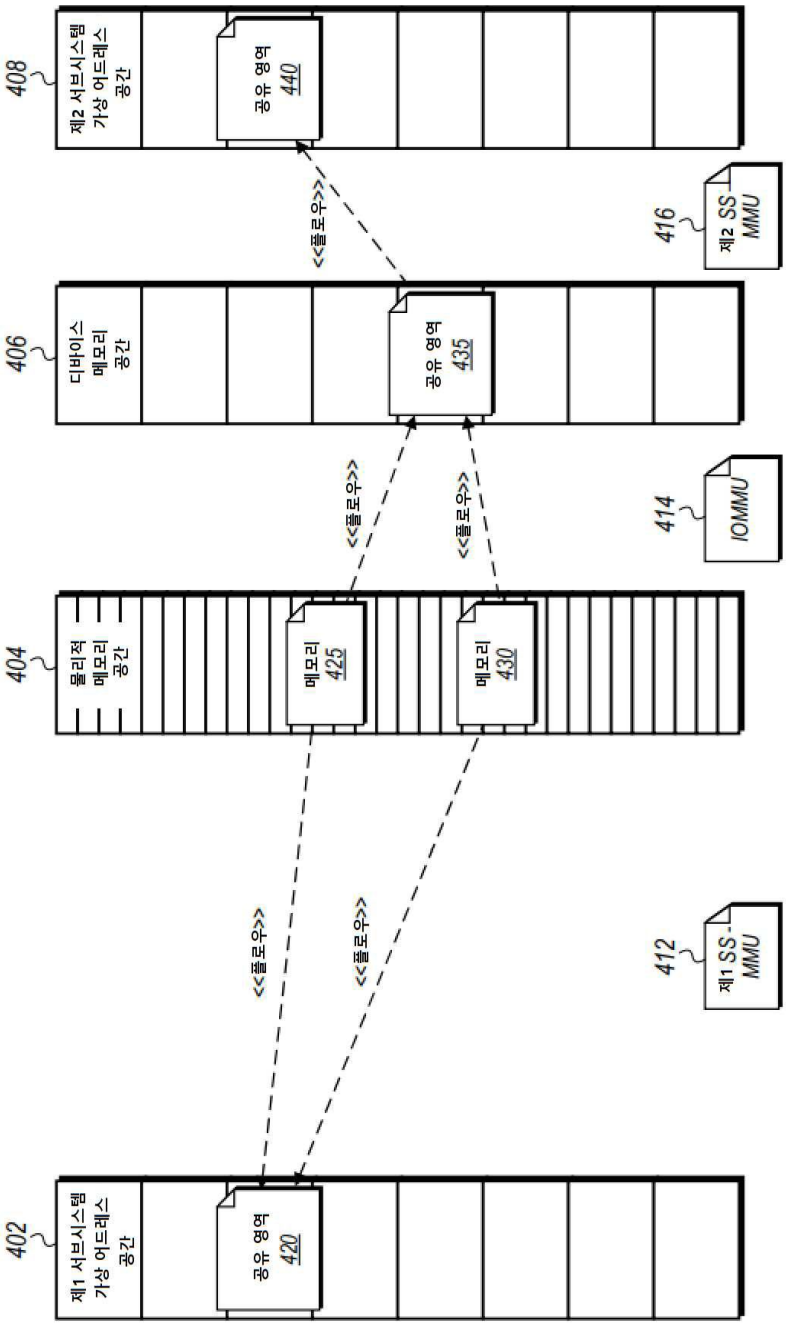
도면2


202
제1 서브시스템
204
제1 서브시스템 MMU
206
디바이스 드라이버
208
공유 영역
211
IOMMU
212
제2 서브시스템 MMU
214
제2 서브시스템
완료 공유 메모리 영역 할당
제2 서브시스템에 대한 디바이스 어드레스 영역 선택
드라이버가 셧다운 될때까지 이 IOMMU 매핑을 해제하지 말것
carveout_heap_create(): heap*
heap_ops->allocation(int) size): buffer_t
heap_ops->map_kernel(buffer_t buf): void*
kmap(pages, count, flags, prot): void*
220
alloc_iova(domain, size, limit_pfn, size_aligned): iova
iova_dma_addr(): dma_addr_t
iommu_map(domain, iova, paddr, size, prot): device_addr_t
heap_ops->unmap_kernel()
heap_ops->free()
205
210
215
225
230
235
240
245

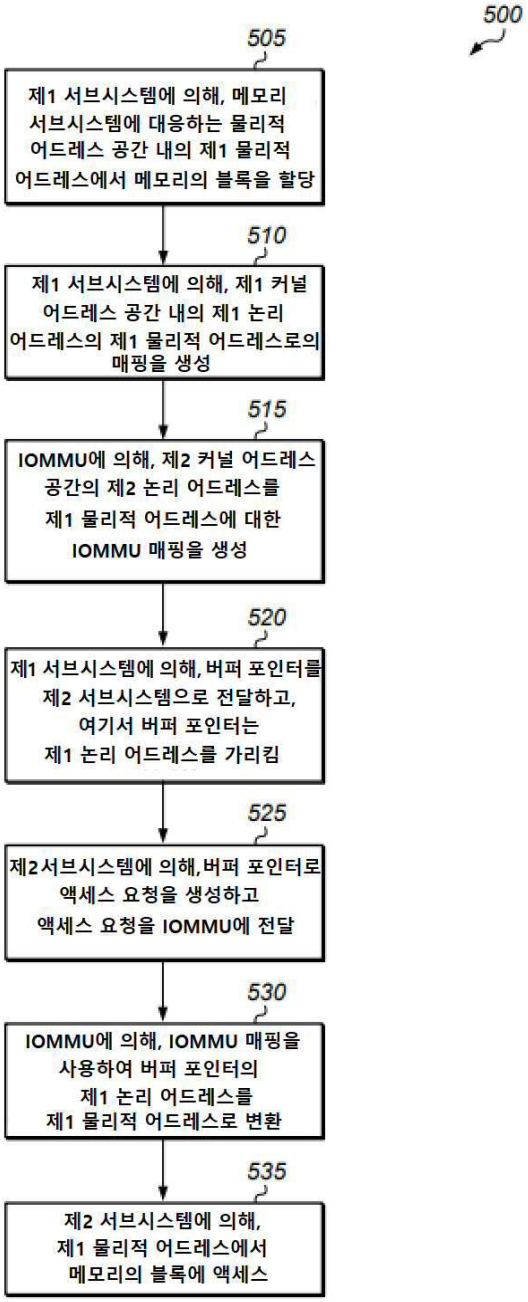
도면3


302
제1 서브시스템
304
제1 서브시스템 MMU
306
디바이스 드라이버
308
공유 영역
310
IOMMU
312
제2 서브시스템 MMU
314
제2 서브시스템
루프 교환 305
heap_ops->()
heap_pfs->map_kernel()
kmap()
send_msg(access, device_addr, size)
kmap()
2ND_SS_buf->data == something()
2ND_SS_buf->data == something_else()
루프 교환 315
1st_SS_buf->data = something()
1st_SS_buf->data == something_else()

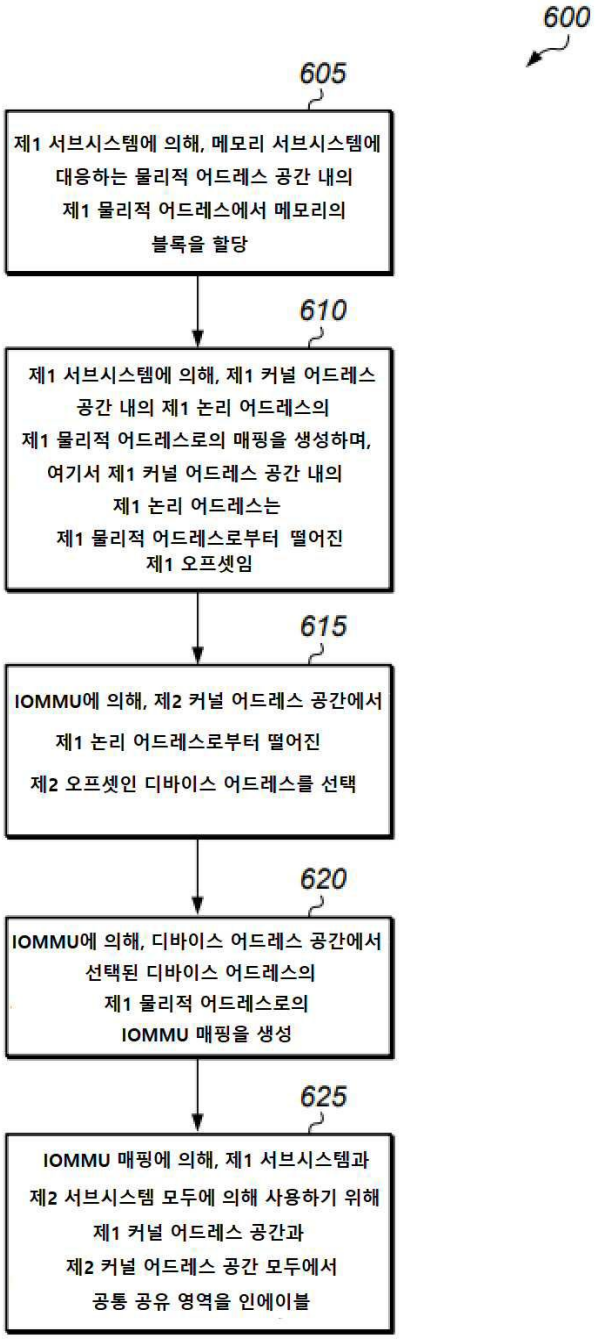
도면4


408
제2 서브시스템 가상 어드레스 공간
공유 영역 440
416
제2 SS MMU
406
디바이스 메모리 공간
공유 영역 435
414
IOMMU
404
물리적 메모리 공간
메모리 425
메모리 430
412
제1 SS-MMU
402
제1 서브시스템 가상 어드레스 공간
공유 영역 420
<<플로우>>
<<플로우>>
<<플로우>>
<<플로우>>

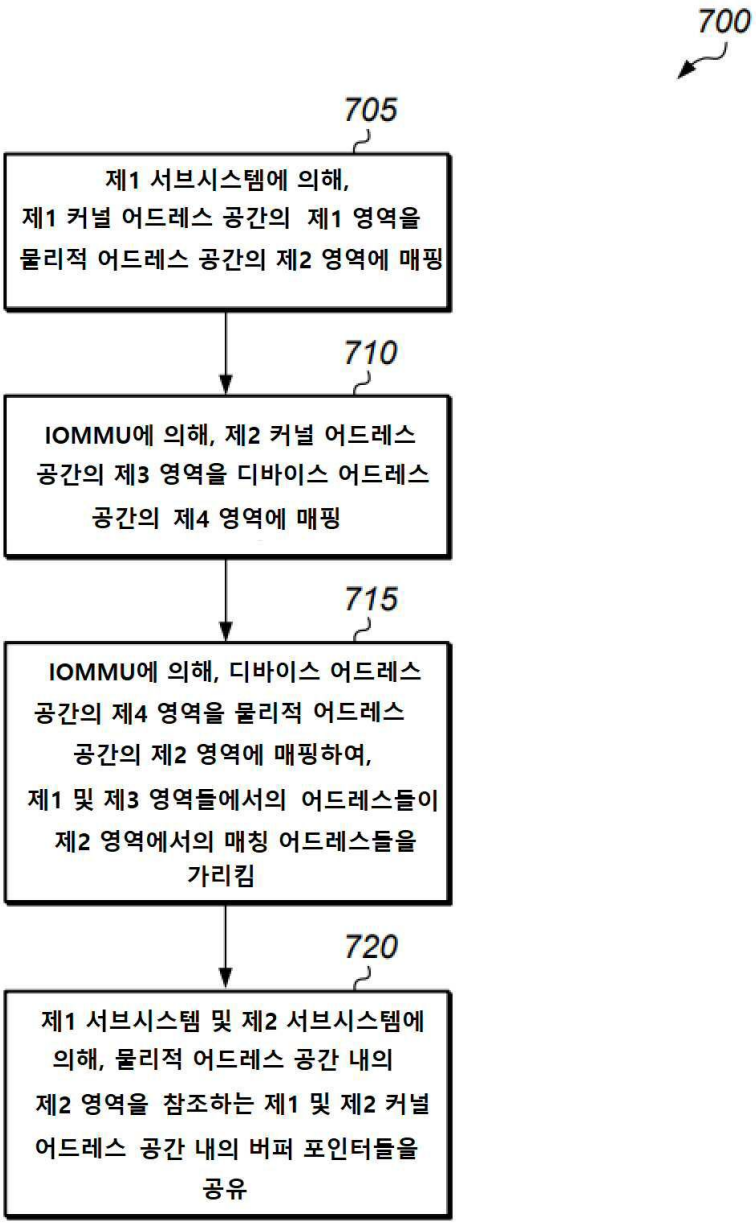
도면5


500
505
제1 서브시스템에 의해, 메모리 서브시스템에 대응하는 물리적 어드레스 공간 내의 제1 물리적 어드레스에서 메모리의 블록을 할당
510
제1 서브시스템에 의해, 제1 커널 어드레스 공간 내의 제1 논리 어드레스의 제1 물리적 어드레스로의 매핑을 생성
515
IOMMU에 의해, 제2 커널 어드레스 공간의 제2 논리 어드레스를 제1 물리적 어드레스에 대한 IOMMU 매핑을 생성
520
제1 서브시스템에 의해, 버퍼 포인터를 제2 서브시스템으로 전달하고, 여기서 버퍼 포인터는 제1 논리 어드레스를 가리킴
525
제2서브시스템에 의해, 버퍼 포인터로 액세스 요청을 생성하고 액세스 요청을 IOMMU에 전달
530
IOMMU에 의해, IOMMU 매핑을 사용하여 버퍼 포인터의 제1 논리 어드레스를 제1 물리적 어드레스로 변환
535
제2 서브시스템에 의해, 제1 물리적 어드레스에서 메모리의 블록에 액세스

도면6


600
605
제1 서브시스템에 의해, 메모리 서브시스템에 대응하는 물리적 어드레스 공간 내의 제1 물리적 어드레스에서 메모리의 블록을 할당
610
제1 서브시스템에 의해, 제1 커널 어드레스 공간 내의 제1 논리 어드레스의 제1 물리적 어드레스로의 매핑을 생성하며, 여기서 제1 커널 어드레스 공간 내의 제1 논리 어드레스는 제1 물리적 어드레스로부터 떨어진 제1 오프셋임
615
IOMMU에 의해, 제2 커널 어드레스 공간에서 제1 논리 어드레스로부터 떨어진 제2 오프셋인 디바이스 어드레스를 선택
620
IOMMU에 의해, 디바이스 어드레스 공간에서 선택된 디바이스 어드레스의 제1 물리적 어드레스로의 IOMMU 매핑을 생성
625
IOMMU 매핑에 의해, 제1 서브시스템과 제2 서브시스템 모두에 의해 사용하기 위해 제1 커널 어드레스 공간과 제2 커널 어드레스 공간 모두에서 공통 공유 영역을 인에이블

도면7


700
705
제1 서브시스템에 의해,
제1 커널 어드레스 공간의 제1 영역을
물리적 어드레스 공간의 제2 영역에 매핑
710
IOMMU에 의해, 제2 커널 어드레스
공간의 제3 영역을 디바이스 어드레스
공간의 제4 영역에 매핑
715
IOMMU에 의해, 디바이스 어드레스
공간의 제4 영역을 물리적 어드레스
공간의 제2 영역에 매핑하여,
제1 및 제3 영역들에서의 어드레스들이
제2 영역에서의 매칭 어드레스들을
가리킴
720
제1 서브시스템 및 제2 서브시스템에
의해, 물리적 어드레스 공간 내의
제2 영역을 참조하는 제1 및 제2 커널
어드레스 공간 내의 버퍼 포인터들을
공유