(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.
G06F 9/46 (2006.01)

(45) 공고일자 2006년09월26일
(11) 등록번호 10-0628492
(24) 등록일자 2006년09월19일

(21) 출원번호 10-2004-0077323
(22) 출원일자 2004년09월24일

(65) 공개번호 10-2005-0030871
(43) 공개일자 2005년03월31일

(30) 우선권주장 JP-P-2003-00335498 2003년09월26일 일본(JP)

(73) 특허권자 가부시끼가이샤 도시바
일본국 도쿄도 미나또꾸 시바우라 1쪼메 1방 1고

(72) 발명자 가나이다츠노리
일본국 도쿄도 미나또꾸 시바우라 1쪼메 1방 1고 가부시끼가이샤 도시바 지적재산부내

마에다세이지
일본국 도쿄도 미나또꾸 시바우라 1쪼메 1방 1고 가부시끼가이샤 도시바 지적재산부내

야노히로쿠니
일본국 도쿄도 미나또꾸 시바우라 1쪼메 1방 1고 가부시끼가이샤 도시바 지적재산부내

요시이겐이치로
일본국 도쿄도 미나또꾸 시바우라 1쪼메 1방 1고 가부시끼가이샤 도시바 지적재산부내

(74) 대리인 김윤배
이범일

(56) 선행기술조사문헌
KR1019990050550 A
KR1020000056310 A
KR1020020035580 A
KR1020030018048 A
US5956321 A
* 심사관에 의하여 인용된 문헌

심사관 : 이재훈

(54) 실시간 동작 수행방법 및 시스템

요약

본 발명의 정보처리 시스템은, 특정의 시간간격 내에 복수의 작업을 수행한다. 이 시스템은, 데이터 전송속도가 변경될 수 있는 버스와, 이 버스를 매개로 데이터를 전송하는 복수의 프로세서, 상기 작업 중 적어도 2개의 작업의 실행기간을 중복하지 않게 하고서 특정의 시간간격 내에 그들 작업을 수행하기 위해, 각 작업을 수행하는데 필요한 시간과 관련된 비용정보와 각 작업에 의해 요구되는 데이터 전송대역폭과 관련된 대역폭 정보에 기초해서, 각 작업의 실행시작 타이밍과 상기 작업을 실행하는 프로세서 중 적어도 하나를 결정하는 스케줄링 동작을 수행하는 수단, 상기 스케줄링 동작의 결과와 상기 대역폭 정보에 기초해서, 특정의 시간간격 내에 상기 프로세서 중 적어도 하나에 의해 수행되어야 할 데이터 전송의 데이터 전송대역폭의 피크값을 계산하는 수단 및, 상기 버스의 데이터 전송속도를 상기 피크값보다 높은 값으로 설정하는 수단을 구비하여 구성되되, 상기 2개의 작업이 나머지 작업의 데이터 전송대역폭보다 작은 데이터 전송대역폭을 필요로 하는 것을 특징으로 한다.

대표도

도 1

명세서

도면의 간단한 설명

도 1은 본 발명의 실시예에 따른 실시간 처리시스템을 구성하는 컴퓨터 시스템의 예를 나타낸 블록도이고,

도 2는 본 발명의 실시예에 따른 실시간 처리시스템에 설치되는 MPU(master processing unit) 및 VPU(versatile processing unit)의 블록도,

도 3은 본 발명의 실시예에 따른 실시간 처리시스템에 사용되는 가상주소번역 메커니즘의 예를 나타낸 도면,

도 4는 본 발명의 실시예에 따른 실시간 처리시스템에서 실제 주소공간에 사상된 데이터의 예를 나타낸 도면,

도 5는 본 발명의 실시예에 따른 실시간 처리시스템에서 유효주소공간, 가상주소공간 및 실제 주소공간을 나타낸 도면,

도 6은 디지털 TV 방송을 위한 수신기의 블록도,

도 7은 본 발명의 실시예에 따른 실시간 처리시스템에 의해 실행되는 프로그램 모듈의 예를 나타낸 도면,

도 8은 도 7에 도시된 프로그램 모듈에 포함된 구조적 기술의 예를 나타낸 테이블,

도 9는 도 7에 도시된 프로그램 모듈에 대응하는 프로그램 사이에서의 데이터의 흐름을 나타낸 차트,

도 10은 2개의 VPU에 의해 수행되는 도 7에 도시된 프로그램 모듈의 병렬 동작을 나타낸 차트,

도 11은 2개의 VPU에 의해 수행되는 도 7에 도시된 프로그램 모듈의 파이프라인 동작을 나타낸 차트,

도 12는 실시간 동작의 각 작업의 실행기간과 요구되는 데이터 전송대역폭의 관계를 나타낸 차트,

도 13은 각 작업에 의해 요구되는 데이터 전송대역폭을 고려하여 가능할 만큼 많은 주기 내에서 요구되는 데이터 전송대역폭을 균일화하는 스케줄링의 예를 나타낸 차트,

도 14는 본 발명의 실시예에 따른 실시간 처리시스템에 의해 수행되는 전력절약 제어동작을 위한 단계의 예를 나타낸 플로우차트,

도 15는 하나의 VPU에 의해 실시간 동작의 스레드를 주기적으로 실행하는 스케줄링의 차트,

도 16은 동시에 2개의 VPU에 의해 2개의 실시간 동작을 수행하는 스케줄링의 예를 나타낸 차트,

도 17은 본 발명의 실시예에 따른 스케줄링 방법을 통해 동시에 2개의 VPU에 의해 2개의 실시간 동작을 수행하는 스케줄링의 예를 나타낸 차트,

도 18은 본 발명의 실시예에 따른 실시간 처리시스템에서의 동작 시스템의 예를 나타낸 도면,

도 19는 본 발명의 실시예에 따른 실시간 처리시스템에서의 동작 시스템의 다른 예를 나타낸 도면,

도 20은 본 발명의 실시예에 따른 실시간 처리시스템에서 가상기계 OS와 게스트 OS 사이의 관계를 나타낸 도면,

도 21은 본 발명의 실시예에 따른 실시간 처리시스템에서 복수의 게스트 OS에 시분할로 할당되는 리소스를 나타낸 차트,

도 22는 본 발명의 실시예에 따른 실시간 처리시스템에서 특정 게스트 OS에 의해 획득되는 특정 리소스를 나타낸 차트,

도 23은 본 발명의 실시예에 따른 실시간 처리시스템에서 스케줄러로 사용된 VPU 실행시간 환경을 나타낸 도면,

도 24는 본 발명의 실시예에 따른 실시간 처리시스템에서 사용된 가상기계 OS로 구현되는 VPU 실행시간 환경의 예를 나타낸 도면,

도 25는 본 발명의 실시예에 따른 실시간 처리시스템에서 사용된 게스트 OS로서 구현되는 VPU 실행시간 환경의 예를 나타낸 도면,

도 26은 본 발명의 실시예에 따른 실시간 처리시스템에서 사용된 각 게스트 OS로 구현되는 VPU 실행시간 환경의 예를 나타낸 도면,

도 27은 본 발명의 실시예에 따른 실시간 처리시스템에서 사용된 1개의 게스트 OS로 구현되는 VPU 실행시간 환경의 예를 나타낸 도면,

도 28은 본 발명의 실시예에 따른 실시간 처리시스템에서 사용된 MPU 측 VPU 실행시간 환경 및 VPU 측 VPU 실행시간 환경을 나타낸 도면,

도 29는 본 발명의 실시예에 따른 실시간 처리시스템에서 사용된 VPU 측 VPU 실행시간 환경에 의해 수행되는 수순을 나타낸 플로우차트,

도 30은 본 발명의 실시예에 따른 실시간 처리시스템에서 사용된 MPU 측 VPU 실행시간 환경에 의해 수행되는 수순을 나타낸 플로우차트,

도 31은 본 발명의 실시예에 따른 실시간 처리시스템에서 다른 프로세서에 의해 실행되고 단단히 결합된 스레드 그룹에 속하는 스레드를 나타낸 도면,

도 32는 본 발명의 실시예에 따른 실시간 처리시스템에서 단단히 결합된 스레드 사이의 상호작용을 나타낸 도면,

도 33은 본 발명의 실시예에 따른 실시간 처리시스템에서 단단히 결합된 스레드의 유효주소공간에서 파트너 스레드를 실행하는 VPU의 로컬기억장치의 사상을 나타낸 도면,

도 34는 본 발명의 실시예에 따른 실시간 처리시스템에서 프로세서를 느슨히 결합된 스레드 그룹에 속하는 스레드에 할당하는 것을 나타낸 도면,

도 35는 본 발명의 실시예에 따른 실시간 처리시스템에서 느슨히 결합된 스레드 사이의 상호작용을 나타낸 도면,

도 36은 본 발명의 실시예에 따른 실시간 처리시스템에서 프로세스와 스레드 사이의 관계를 나타낸 도면,

도 37은 본 발명의 실시예에 따른 실시간 처리시스템에서 스케줄링 동작을 수행하기 위한 수순을 나타낸 플로우차트,

도 38은 본 발명의 실시예에 따른 실시간 처리시스템에서 스레드의 상태천이를 나타낸 도면,

도 39는 본 발명의 실시예에 따른 실시간 처리시스템에서 스레드와 그 실행기간 사이의 관계를 나타낸 차트,

도 40은 본 발명의 실시예에 따른 실시간 처리시스템에서 실행기간에서 동시에 실행되는 단단히 결합된 스레드를 나타낸 차트,

도 41은 본 발명의 실시예에 따른 실시간 처리시스템에서 주기적인 실행모델을 나타낸 차트,

도 42는 본 발명의 실시예에 따른 실시간 처리시스템에서 비주기적인 실행모델을 나타낸 차트,

도 43은 작업 그래프를 나타낸 도면,

도 44는 본 발명의 실시예에 따른 실시간 처리시스템에서 사용되는 예약그래프의 원리를 나타낸 도면,

도 45는 본 발명의 실시예에 따른 실시간 처리시스템에서 사용되는 예약그래프의 예를 나타낸 도면,

도 46은 본 발명의 실시예에 따른 실시간 처리시스템에서 사용되는 계층적인 스케줄러를 나타낸 도면,

도 47은 본 발명의 실시예에 따른 실시간 처리시스템에 의한 하드 실시간 클래스에서의 스케줄링에 사용되는 파라미터의 예를 나타낸 차트,

도 48은 본 발명의 실시예에 따른 실시간 처리시스템에서 사용된 절대적인 타이밍 제약조건을 나타낸 도면,

도 49는 본 발명의 실시예에 따른 실시간 처리시스템에서 사용된 상대적인 타이밍 제약조건을 나타낸 도면,

도 50은 본 발명의 실시예에 따른 실시간 처리시스템에서 사용되는 상호배타적인 제약조건을 나타낸 도면,

도 51은 본 발명의 실시예에 따른 실시간 처리시스템에서 동기화 메커니즘을 나타낸 테이블,

도 52는 본 발명의 실시예에 따른 실시간 처리시스템에서 동기화 메커니즘을 선택적으로 사용하기 위한 수순을 나타낸 플로우차트,

도 53은 본 발명의 실시예에 따른 실시간 처리시스템에서 사용되는 예약그래프의 예를 나타낸 도면,

도 54는 본 발명의 실시예에 따른 실시간 처리시스템에서 생성된 예약요청의 예를 나타낸 도면,

도 55는 도 54에 도시된 예약요청에 기초해서 본 발명의 실시예에 따른 실시간 처리시스템에 의해 수행되는 스케줄링의 예를 나타낸 차트,

도 56은 본 발명의 실시예에 따른 실시간 처리시스템에 의해 수행되는 소프트웨어 파이프라인 타입의 스케줄링의 제1예를 나타낸 차트,

도 57은 본 발명의 실시예에 따른 실시간 처리시스템에 의해 수행되는 소프트웨어 파이프라인 타입의 스케줄링의 제2예를 나타낸 차트,

도 58은 본 발명의 실시예에 따른 실시간 처리시스템에 의해 수행되는 소프트웨어 파이프라인 타입의 스케줄링을 위한 수순을 나타낸 플로우차트,

도 59는 본 발명의 실시예에 따른 실시간 처리시스템에 의해 수행되는 소프트웨어 파이프라인 타입의 스케줄링의 제3예를 나타낸 차트,

도 60은 동시에 2개의 VPU에 의해 2개의 실시간 동작을 수행하는 스케줄링의 예를 나타낸 차트,

도 61은 본 발명의 실시예에 따른 실시간 처리시스템에서 2개의 VPU에 의해 2개의 실시간 동작을 파리프라인 모드로 동시에 수행하는 스케줄링을 나타낸 차트,

도 62는 도 61에 도시된 스케줄링에 의해 요구되는 버스 대역폭에서의 축소를 나타낸 차트,

도 63은 본 발명의 실시예에 따른 실시간 처리시스템에서 사용되는 계층적인 구조를 갖는 예약그래프의 예를 나타낸 도면,

도 64는 본 발명의 실시예에 따른 실시간 처리시스템에서 사용되는 예약 리스트의 예를 나타낸 도면,

도 65는 본 발명의 실시예에 따른 실시간 처리시스템에서 실행기간을 예약하기 위한 수순을 나타낸 플로우차트이다.

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 특정의 시간간격으로 주기적으로 실시간 동작을 수행하기 위한 스케줄링방법 및 정보처리 시스템에 관한 것이다.

종래, 서버 컴퓨터 등과 같은 컴퓨터 시스템은 스루풋(throughput: 처리량)을 향상시키기 위해 멀티프로세서(multiprocessor) 및 병렬(parallel) 프로세서 등과 같은 시스템 구조를 이용하고 있었다. 상기 양 프로세서는 복수의 처리유니트를 이용하여 병렬계산동작을 달성한다.

일본 특허공개공보 제10-143380호에는 복수의 처리유니트를 갖는 시스템이 개시되어 있다. 이러한 시스템은 단일의 고속 CPU, 복수의 저속 CPU 및 공유 메모리를 포함하고 있다. 프로세스는 각 프로세스의 상관관계와 실행시간을 고려하여 고속 및 저속 CPU에 할당되도록 되어 있다.

일본 특허공개공보 제8-180025호에는 동일한 프로세서가 동일한 프로세스에 속하는 스레드(thread)를 실행하도록 스레드를 스케줄링하는 스케줄링 기법이 개시되어 있다.

실시간으로 AV(audio video) 데이터 등과 같은 대량의 데이터를 처리할 필요가 있는 컴퓨터 시스템뿐만 아니라 내장형의 장치(embedded device)도 최근 스루풋을 향상시키기 위해 도입되는 멀티프로세서 및 병렬 프로세서와 같은 시스템 구조를 필요로 하고 있다.

그러나, 현재의 환경 하에서는, 상기의 시스템 구조 상에서 예상되는 실시간 처리시스템은 거의 보고되고 있지 않다.

실시간 처리시스템에서, 각 동작은 허가된 시간의 한계내에서 완료될 필요가 있다. 주기적으로 특정의 시간간격으로 복수의 연쇄작업(chained task)의 조합을 포함하는 실시간 동작을 수행하기 위해, 모든 연쇄작업은 각 주기의 시간간격 내에 완료될 필요가 있다.

실시간 처리시스템은 종종 내장형 시스템으로 사용되기 때문에, 그 중대한 문제는 전력소비를 줄이는데 있다. 이 시스템 내에 포함되는 처리유니트의 수가 많으면 많을수록, 요구되는 데이터 전송속도(데이터 전송대역폭)는 높아진다. 그리고 데이터 전송대역폭이 크면 클수록, 전력소비는 커진다. 멀티프로세서와 병렬 프로세서 등과 같은 시스템 아키텍처를 실시간 처리시스템에 적용하는 경우에는, 주어진 기간 내에 실시간 동작을 완료하면서 요구되는 데이터 전송대역폭을 줄이기 위해 새로운 메커니즘이 필요하게 된다.

발명이 이루고자 하는 기술적 과제

본 발명의 목적은, 어떤 실시간 동작도 손상시키지 않고 요구되는 데이터 전송대역폭을 줄일 수 있는 방법 및 정보처리 시스템을 제공하는데 있다.

발명의 구성 및 작용

본 발명의 실시예에 따르면, 데이터 전송속도가 변경될 수 있는 버스를 매개로 데이터를 전송하는 복수의 프로세서를 이용하여 특정의 시간간격 내에 복수의 작업을 수행하는 방법이 제공된다. 이 방법은, 각 작업을 수행하는데 필요한 시간과 관련된 비용정보와 각 작업에 의해 요구되는 데이터 전송대역폭과 관련된 대역폭 정보를 입력하는 단계와, 상기 작업 중 적어도 2개의 작업의 실행기간을 중복하지 않게 하고서 특정의 시간간격 내에 그들 작업을 수행하기 위해, 상기 입력된 비용정보와 대역폭 정보에 기초해서, 각 작업의 실행시작 타이밍과 상기 작업을 실행하는 프로세서 중 적어도 하나를 결정하는 스케줄링 동작을 수행하는 단계, 상기 스케줄링 동작의 결과와 상기 대역폭 정보에 기초해서, 특정의 시간간격 내에 상기 프로세서 중 적어도 하나에 의해 수행되어야 할 데이터 전송의 데이터 전송대역폭의 피크값을 계산하는 단계 및, 상기 버스의 데이터 전송속도를 상기 피크값보다 높은 값으로 설정하는 단계를 구비하여 이루어지되, 상기 2개의 작업이 나머지 작업의 데이터 전송대역폭보다 작은 데이터 전송대역폭을 필요로 하는 것을 특징으로 한다.

(실시예)

이하, 예시도면을 참조하여 본 발명의 실시예를 상세히 설명한다.

도 1은 본 발명의 실시예에 따른 실시간 처리시스템을 실현하기 위한 컴퓨터 시스템의 구성례를 나타낸다. 이 컴퓨터 시스템은 시간 제약조건 하에서 실시간으로 수행될 필요가 있는 여러 가지의 동작을 수행하는 정보처리 시스템이다. 이 컴퓨터 시스템은 범용 컴퓨터뿐만 아니라, 실시간으로 수행될 필요가 있는 동작을 수행하기 위한 각종의 전자장치용의 내장형 시스템(embeded system)으로도 사용될 수 있다. 도 1을 참조하면, 컴퓨터 시스템은 MPU(마스터 처리유니트; 11), 복수의 VPU(다목적 처리유니트; 12), 연결장치(connecting device; 13), 주메모리(14) 및 I/O(입/출력) 제어기(15)를 포함하고 있다. MPU(11), VPU(12), 주메모리(14) 및 IO 제어기(15)는 연결장치(13)에 의해 서로 연결되어 있다. 연결장치(13)는 버스(bus)를 포함하는 데이터 전송경로이다. 예컨대, 크로스바(crossbar) 스위치와 같은 링(ring)모양의 버스구조 또는 상호접속 네트워크가 버스로 사용될 수 있다. 버스가 연결장치(13)로 사용되는 경우에는, 링과 같은 모양으로 형성될 수 있다. MPU(11)는 컴퓨터 시스템의 동작을 제어하는 주프로세서이다. MPU(11)는 주로 OS(operating system)를 실행한다. VPU(12)와 IO 제어기(15)는 OS의 몇 가지 기능을 실행할 수 있다. VPU(12)의 각각은 MPU(11)의 제어 하에 갖가지 동작을 수행하기 위한 프로세서이다. MPU(11)는 이들 동작(작업)을 병렬로 수행하기 위해 이들 동작(작업)을 VPU(12)로 분배한다. 따라서, 이들 동작은 고속에서 고효율로 수행될 수 있다. 주메모리(14)는 MPU(11), VPU(12) 및 I/O 제어기(15)에 의해 공유되고 있는 기억장치(공유 메모리)이다. 주메모리(14)는 OS와 응용 프로그램을 저장한다. I/O 제어기(15)는 하나 이상의 I/O 장치(16)에 연결되어 있다. 또한, 제어기(15)는 브리지 장치(bridge device)라고도 불린다.

연결장치(13)는 데이터 전송속도를 보장하는 QoS(서비스 품질)기능을 갖는다. QoS기능은 연결장치(13)를 통해 예약된 대역폭(전송속도)으로 데이터를 전송함으로써 이행된다. QoS기능은 기록 데이터(write data)가 하나의 VPU(12)로부터 5Mbps로 메모리(14)로 송신될 때, 또는 하나의 VPU(12)와 다른 VPU(12) 사이에서 100Mbps로 송신될 때 사용된다. VPU(12)의 각각은 연결장치(13)에 대해 대역폭(전송속도)을 지정한다(예약한다). 연결장치(13)는 지정된 대역폭을 우선순위에 따라 VPU(12)에 할당한다. VPU(12)의 데이터 전송을 위해 대역폭이 예약되어 있는 경우에는, 이전의 VPU(12)의 데이터 전송 중에 다른 VPU(12), MPU(11) 또는 IO 제어기(15)가 대량의 데이터를 전송한다고 하더라도 이것이 보호된다. QoS기능은 실시간으로 동작을 수행하는 컴퓨터에 있어서 특히 중요하다.

도 1에 나타낸 컴퓨터 시스템은 하나의 MPU(11), 4개의 VPU(12), 하나의 메모리(14) 및 하나의 IO 제어기(15)를 구비하고 있다. VPU(12)의 수는 제한되지 않는다. 시스템은 MPU를 갖출 필요가 없고, 이 경우 하나의 VPU(12)가 MPU(11)의 동작을 수행한다. 즉, 하나의 VPU(12)가 가상의 MPU(11)로서 기능한다.

또, 컴퓨터 시스템은 전력절약 제어기(17)를 구비하고 있다. 이 제어기(17)는 시스템의 전체 또는 일부의 전력소비를 줄이기 위해 다음과 같은 기능을 이행한다.

1. 전체 컴퓨터 시스템의 클럭주파수를 낮춘다.

2. 전체 컴퓨터 시스템의 전원전압을 낮춘다.

3. 전체 컴퓨터 시스템의 전원을 턴오프(turn off)한다.

4. 하나 이상의 모듈(MPU, VPU, 메모리, I/O 제어기 등)의 클럭주파수를 낮춘다.

5. 하나 이상의 모듈(MPU, VPU, 메모리, I/O 제어기 등)의 전원전압을 낮춘다.

6. 하나 이상의 모듈(MPU, VPU, 메모리, I/O 제어기 등)의 전원을 턴오프한다.

7. 연결장치의 클럭주파수(동작주파수)를 낮춘다.

8. 연결장치의 전송속도를 줄인다.

9. 연결장치의 대역폭을 좁게 한다.

10. 연결장치의 전원을 턴오프한다.

11. 메모리 뱅크의 유니트에서 전원을 턴오프한다.

12. 메모리 뱅크의 유니트에서 재생(refresh)을 중단한다.

13. MPU와 VPU에서 동시에 동작되는 기능모듈을 줄인다(프로세서가 복수의 동작 유니트를 포함하고 있는 경우에는, 동시에 사용되는 동작 유니트의 수를 제한한다).

상기의 전력절약 기능은 소프트웨어의 제어 하에서 이행될 수 있다. 상기의 전력절약기능 1～13은 단독으로 실행되거나 서로 협력하여 실행될 수 있다.

도 2는 MPU(11)와 VPU(12)를 나타낸다. MPU(11)는 처리유니트(processing unit; 21)와 메모리 관리 유니트(memory management unit; 22)를 갖추고 있다. 처리유니트(21)는 메모리 관리 유니트(22)를 통해 메모리(14)에 억세스한다. 메모리 관리 유니트(22)는 가상 메모리 관리기능을 수행하고, 당해 메모리 관리 유니트(22)의 캐시 메모리를 관리한다. VPU(12)의 각각은 처리유니트(21), 로컬기억장치(로컬 메모리; 32) 및 메모리 제어기(33)를 포함하고 있다. 처리유니트(31)는 동일한 VPU(12)의 로컬기억장치(32)에 직접 억세스할 수 있다. 메모리 제어기(33)는 로컬기억장치(32)와 메모리(14) 사이에서 데이터를 전송하는 DMA(direct memory access) 제어기로서 작용한다. 메모리 제어기(33)는 연결장치(13)의 QoS기능을 이용해서 대역폭을 지정하는 기능과 지정된 대역폭에서 데이터를 입/출력하는 기능을 갖는다. 또한, 메모리 제어기(33)는 MPU(11)의 메모리 관리 유니트(22)와 같은 동일한 가상 메모리 관리기능을 갖는다. 처리유니트(31)는 주메모리로서 로컬기억장치(32)를 사용한다. 처리유니트(31)는 메모리(14)에 직접 억세스하지는 않지만, 메모리 제어기(33)가 메모리(14)의 내용을 로컬기억장치(32)로 전송하도록 지시한다. 처리유니트(31)는 데이터를 독출/기록하기 위해 로컬기억장치(32)에 억세스한다. 더욱이, 처리유니트(31)는 메모리 제어기(33)가 로컬기억장치(32)의 내용을 메모리(14)에 기록하도록 지시한다.

MPU(11)의 메모리 관리 유니트(22)와 VPU(12)의 메모리 제어기(33)는 도 3에 나타낸 바와 같은 가상 메모리 관리를 수행한다. MPU(11)의 처리유니트(21)나 VPU(12)의 메모리 제어기(33)로부터 보여지는 주소는, 도 3의 윗부분에 나타낸 것처럼 64비트 주소이다. 64비트 주소에 있어서, 상위 36비트 부분은 세그먼트 수(segment number)를 나타내고, 중간의 16비트 부분은 페이지 수를 나타내며, 하위 12비트 부분은 페이지 오프셋(page offset)을 나타낸다. 메모리 관리 유니트(22)와 메모리 제어기(33)는 각각 세그먼트 테이블(50)과 페이지 테이블(60)을 갖추고 있다. 세그먼트 테이블(50)과 페이지 테이블(60)은 64비트 주소를 연결장치(13)를 매개로 실제로 억세스되는 실제 주소(real address; RA)공간으로 변환한다.

예컨대, 다음의 데이터 항목(item)들이 도 4에 나타낸 바와 같이 MPU(11)와 각 VPU(12)에서 보여지는 실제주소공간에 사상(map)된다.

1. 메모리(14)(주기억장치)

2. MPU(11)의 제어 레지스터

3. VPU(12)의 제어 레지스터

4. VPU(12)의 로컬기억장치

5. I/O 장치의 제어 레지스터(I/O 제어기(15)의 제어 레지스터를 포함함)

MPU(11)와 VPU(12)는 데이터 항목 1～5를 독출/기록하기 위해 실제주소공간 내의 임의의 주소에 억세스할 수 있다. 실제주소공간에 억세스할 수 있고, 이로써 MPU(11)와 VPU(12)로부터, 그리고 심지어 I/O 제어기(15)로부터 어떤 VPU(12)의 로컬기억장치(32)에도 억세스할 수 있다는 것은 특히 중요한 것이다. 더욱이, 세그먼트 테이블(50)이나 페이지 테이블(60)은 각 VPU(12)의 로컬기억장치(32)의 내용이 자유롭게 독출되거나 기록되는 것을 방지할 수 있다.

도 5는 도 3에 나타낸 가상 메모리 관리기능에 의해 관리되는 메모리 주소공간을 나타낸다. MPU(11)나 VPU(12)로 실행되는 프로그램으로부터 직접 보여지는 것은 EA(effective address; 유효주소)공간이다. 유효주소는 세그먼트 테이블(50)에 의해 VA(virtual address; 가상주소)공간에 사상된다. 가상주소는 페이지 테이블(60)에 의해 RA공간에 사상된다. RA공간은 도 4에 나타낸 바와 같은 구조를 갖는다.

MPU(11)는 제어 레지스터와 같은 하드웨어 메커니즘을 이용하여 VPU(12)를 관리할 수 있다. 예컨대, MPU(11)는 각 VPU의 레지스터로부터 데이터를 독출하거나 각 VPU의 레지스터로 데이터를 기록할 수 있고, 프로그램을 실행하기 위해 각 VPU(12)를 시동시키거나 정지시킬 수 있다. VPU(12) 사이에서 통신 및 동기화(synchronization)가 이루어질 수 있는 것처럼, MPU(11)와 각 VPU(12)간의 통신 및 동기화가 메일박스(mailbox) 및 이벤트 플래그(event flag)와 같은 하드웨어 메커니즘에 의해 수행될 수 있다.

본 실시예에 따른 컴퓨터 시스템은, 소프트웨어가 일반적으로 구현되는 것과 같은 실시간 동작에 대해 엄격한 요구를 하는 전기장치의 그러한 동작을 수행하도록 한다. 예컨대, 하나의 VPU(12)는 전기장치를 구성하는 몇개의 하드웨어 구성요소에 대응하는 계산을 수행하고, 그와 동시에 다른 VPU(12)는 전기장치를 구성하는 다른 하드웨어 구성요소에 대응하는 계산을 수행한다.

도 6은 디지털 TV 방송용 수신기의 하드웨어 구조를 간단히 나타낸다. 이 수신기에 있어서, DEMUX(디멀티플렉서) 회로(101)는 수신된 방송신호를 오디오 데이터와 비디오 데이터 및 자막데이터에 대응하는 압축 부호화된 데이터 스트림(compressing-encoded data stream)으로 분할한다. A-DEC(audio decoder: 오디오 디코더) 회로(102)는 압축 부호화된 오디오 데이터 스트림을 복호화한다. V-DEC(video decoder: 비디오 디코더) 회로(103)는 압축 부호화된 비디오 데이터 스트림을 복호화한다. 복호화된 비디오 데이터 스트림은 PROG(progressive conversion: 순차 변환) 회로(105)로 전송되어 순차 비디오신호로 변환된다. 순차 비디오신호는 BLEND(image blending: 이미지 혼합) 회로(106)로 전송된다. TEXT(자막데이터 처리) 회로(104)는 압축 부호화된 자막데이터 스트림을 자막 비디오신호로 변환하고, 그것을 BLEND 회로(106)로 전송한다. BLEND 회로(106)는 PROG 회로(105)로부터 전송된 비디오신호와, TEXT 회로(104)로부터 보내진 자막 비디오신호를 혼합하여 이 혼합된 신호를 비디오 스트림으로서 출력한다. 상술한 일련의 동작은 비디오 프레임 속도(예컨대, 초당 30프레임, 32프레임 또는 60프레임)로 반복된다.

도 6에 나타낸 하드웨어의 동작을 소프트웨어로 수행하기 위해, 본 실시예는 도 7에 나타낸 바와 같은 프로그램 모듈(100)을 제공한다. 프로그램 모듈(100)은 컴퓨터 시스템이 도 6에 나타낸 DEMUX 회로(101), A-DEC 회로(102), V-DEC 회로(103), TEXT 회로(104), PROG 회로(105) 및 BLEND 회로(106)의 동작을 수행하도록 하는 응용 프로그램이다. 응용 프로그램은 멀티 스레드 프로그래밍(multi-thread programming)에 의해 기술되고, 실시간 동작을 실행하기 위한 스레드 그룹으로서 구성된다. 실시간 동작은 복수 작업의 조합을 포함한다. 프로그램 모듈(100)은 각각 스레드로서 실행되는 복수의 프로그램(복수의 루틴)을 포함한다. 구체적으로, 프로그램 모듈(100)은 DEMUX 프로그램(111), A-DEC 프로그램(112), V-DEC 프로그램(113), TEXT 프로그램(114), PROG 프로그램(115) 및 BLEND 프로그램(116)을 포함하고 있다. 이들 프로그램(111～116)은 회로(101～106)의 동작(DEMUX 동작, A-DEC 동작, V-DEC 동작, TEXT 동작, PROG 동작 및 BLEND 동작)에 대응하는 작업의 수순을 기술하는 프로그램이다. 보다 구체적으로는, 프로그램 모듈(100)이 작동할 때, 각 프로그램(111～116)에 대응하는 스레드가 생성되어 하나 이상의 VPU(12)로 발송되어 실행된다. VPU(12)로 발송된 스레드에 대응하는 프로그램은 VPU(12)의 로컬기억장치(32)에 로드(load)되고, 스레드는 로컬기억장치(32)의 프로그램을 실행한다. 프로그램 모듈(100)은 디지털 TV 방송용 수신기를 구성하는 하드웨어 모듈에 대응하는 프로그램(111～116)을 구조적 기술(structural description; 117)이라 불리는 데이터와 함께 패키징(packaging)함으로써 얻어진다.

구조적 기술(117)은 프로그램 모듈(100) 내의 프로그램(스레드)이 어떻게 결합되고 실행되는지를 나타내는 정보이다. 구조적 기술(117)은, 연쇄 프로그램(111～116)간의 (연쇄적인) 입/출력관계와 프로그램(111～116)의 각각을 실행하는데 필요한 비용(시간)을 나타내는 정보를 포함하고 있다. 도 8은 구조적 기술(117)의 일례를 나타낸다.

구조적 기술(117)은 스레드로서 각각 실행되는 모듈(프로그램 모듈(100) 내의 프로그램)과, 이들 모듈의 대응하는 입력, 출력, 실행비용 및 출력에 필요한 버퍼 사이즈를 나타낸다. 예컨대, (3)번의 V-DEC 프로그램은 (1)번의 DEMUX 프로그

램의 출력을 입력으로서 받고, 그 출력을 (5)번의 PROG 프로그램으로 전송한다. V-DEC 프로그램의 출력에 필요한 버퍼는 1MB이고, V-DEC 프로그램을 그 자체로서 실행하기 위한 비용은 50이다. 비용은 프로그램을 실행하는데 필요한 시간단위(시간주기)나 프로그램의 단계 수로 기술될 수 있다. 또한, 몇개의 가상의 세부사항(specification: 명세)을 갖는 가상 프로세서에 의해 프로그램을 실행하는데 요구되는 시간단위로 기술될 수도 있다. VPU 세부사항과 성능은 컴퓨터에 따라 달라질 수 있기 때문에, 그러한 가상 단위로 비용을 기술하는 것이 바람직하다. 구조적 기술(117)에서의 버스 대역폭은 프로그램(111～116)의 각각에 의해 연결장치(13)를 매개로 데이터를 전송하기 위해 필요하게 되는 데이터 전송대역폭(데이터 전송속도)을 나타내는 정보이다. 데이터 전송은 VPU 사이에서, VPU와 메모리(14) 사이에서, 또는 VPU와 I/O장치(16) 사이에서 수행된다. 상기의 QoS 기능은 프로그램(111～116)의 각각에 대응하는 동작을 수행하기 위해 필요하게 되는 대역폭이 확보되도록 해준다. 프로그램이 도 8에 나타낸 구조적 기술(117)에 따라 실행되는 경우에는, 데이터는 도 9에 나타낸 바와 같이 프로그램 사이에서 흐른다.

또한, 구조적 기술(117)은 스레드 파라미터로서 프로그램(111～116)에 대응하는 스레드간의 결합속성(coupling attribute)을 나타내는 결합속성정보를 나타낸다. 결합속성은 단단히 결합된 속성과 느슨히 결합된 속성의 2가지의 다른 속성을 포함한다. 단단히 결합된 속성을 갖는 복수의 스레드는 서로 협력하여 실행되고, 단단히 결합된 스레드 그룹(tightly coupled thread group)이라 일컬어진다. 본 실시예의 컴퓨터 시스템은 동일한 단단히 결합된 스레드 그룹에 속하는 스레드가 다른 VPU에 의해 동시에 실행될 수 있도록 각각의 단단히 결합된 스레드 그룹에 속한 스레드를 스케줄링한다. 느슨히 결합된 속성을 갖는 복수의 스레드는 느슨히 결합된 스레드 그룹(loosely coupled thread group)이라 일컬어진다. 프로그래머는 스레드 파라미터를 이용하여 프로그램(11～16)에 대응하는 스레드간의 결합속성을 지정할 수 있다. 도 25 등을 참조하여 단단히 결합된 스레드 그룹과 느슨히 결합된 스레드 그룹에 대해 상세히 설명한다. 결합속성정보를 포함하는 스레드 파라미터는 구조적 기술(117)로서가 아니라, 프로그램(111～116) 내의 코드로서 직접 기술될 수 있다.

도 10과 도 11을 참조하면서, 본 실시예의 컴퓨터 시스템이 어떻게 프로그램(111～116)을 실행하는가에 대해 설명한다. 여기에서, 컴퓨터 시스템은 VPU0와 VPU1의 2개의 VPU를 포함하고 있는 것으로 가정한다. 도 10은 30프레임의 비디오 데이터가 매초마다 디스플레이될 때, VPU의 각각에 프로그램을 할당하는 시간을 나타낸다. 1프레임에 대한 오디오 및 비디오 데이터는 1주기에 대응하는 시간간격 내에서 출력된다. 먼저, VPU0는 DEMUX 동작을 수행하기 위해 DEMUX 프로그램을 실행하고, 그 결과로서 생기는 오디오, 비디오 및 자막데이터를 버퍼에 기록한다. 그 후, VPU1은 A-DEC 동작과 TEXT 동작을 차례로 수행하기 위해 A-DEC 프로그램과 TEXT 프로그램을 실행하고, 그들 결과를 버퍼에 기록한다. 이어서, VPU0은 V-DEC 동작을 수행하기 위해 V-DEC 프로그램을 실행하고, 그 결과를 버퍼에 기록한다. VPU0은 PROG 동작을 수행하기 위해 PROG 프로그램을 실행하고, 그 결과를 버퍼에 기록한다. 이 때, VPU1은 TEXT 프로그램을 이미 완료했기 때문에, 최종 비디오 데이터를 만들기 위해 VPU0이 최후의 BLEND 프로그램을 실행해서 BLEND 동작을 수행하도록 한다. 상기의 처리는 매 주기마다 반복된다.

원하는 동작을 지연없이 수행하기 위해 어떤 프로그램이 각 VPU에 의해 실행되고, 언제 행해지는지를 결정하는 동작을 스케줄링(scheduling)이라고 한다. 스케줄링을 실행하는 모듈을 스케줄러(scheduler)라고 한다. 본 실시예에서는, 스케줄링은 프로그램 모듈(100) 내에 포함된 상기 구조적 기술(117)에 기초해서 실행된다. 스케줄링 동작에서, 프로그램(111～116)을 실행하는 각 스레드의 실행시작 타이밍(execution start timing)과 실행기간은 모두 구조적 기술(117)에 기초해서 결정되고, 이에 따라 스레드의 각각을 하나 이상의 VPU(12)에 할당하게 된다. 프로그램 모듈(100)이 실행될 때 다음의 동작이 수행된다.

1. 동작 시스템은 외부 기억장치나 메모리(14)로부터 프로그램 모듈(100)을 수신하고, 프로그램 모듈(100)로부터 복수의 프로그램(111～116)과 구조적 기술(117)을 독출한다.

2. 구조적 기술(117)에 기초해서, 동작 시스템의 스케줄러는 스레드(DEMUX, V-DEC, A-DEC, TEXT, PROG, BLEND)를 하나 이상의 VPU에 할당하도록 프로그램 모듈(100) 내의 프로그램(111～116)을 실행하기 위해 스레드(DEMUX, V-DEC, A-DEC, TEXT, PROG, BLEND)의 각각의 실행시작 타이밍과 실행기간을 결정한다.

상술한 바와 같이, 실시간 처리시스템에서는, 프로그램 모듈(100) 내의 연쇄 프로그램(111～116)을 실행하는 스레드(DEMUX, V-DEC, A-DEC, TEXT, PROG, BLEND)의 각각의 실행시작 타이밍과 실행기간은 구조적 기술(117)에 기초해서 결정된다. 따라서, 실시간 동작을 수행하기 위한 스레드는, 프로그램의 코드에서의 각 동작의 타이밍 제약조건을 기술하지 않고 효과적으로 스케줄링될 수 있다.

도 11은 60프레임의 비디오 데이터가 매초마다 디스플레이될 때 실행되는 프로그램을 나타낸다. 도 11은 다음과 같이 도 10과는 다르다. 도 11에서는 60프레임의 데이터가 매초마다 처리될 필요가 있지만, 도 10에서는 30프레임의 데이터가 매초마다 처리되어 1프레임에 대한 데이터 처리가 1주기(1/30초)에 완료될 수 있다. 즉, 1프레임 데이터 처리가 1주기(1/60

초)에 완료될 수 없기 때문에, 복수의 주기(2주기)에 걸친 소프트웨어 파이프라인 동작이 도 11에서 수행된다. 예컨대, 주기 1에서는, VPU0이 입력신호를 위한 DEMUX 프로그램과 V-DEC 프로그램을 실행한다. 그 후, 주기 2에서는 VPU1이 A-DEC, TEXT, PROG 및 BLEND 프로그램을 실행하고, 최종 비디오 데이터를 출력한다. 주기 2에서는 VPU0이 다음 프레임에서 DEMUX와 V-DEC 프로그램을 실행한다. VPU0의 DEMUX 및 V-DEC 프로그램과, VPU1의 A-DEC, TEXT, PROG, BLEND 프로그램은 파이프라인 모드로 2주기에 걸쳐 실행된다.

상기의 파이프라인 동작을 실행하기 위해서는, 프로그램 모듈(100)이 실행될 때에 다음의 동작을 수행한다:

1. 동작 시스템은 외부 기억장치나 메모리(14)로부터 프로그램 모듈(100)을 수신하고, 프로그램 모듈(100)로부터 구조적 기술(117)을 독출한다.

2. 동작 시스템의 스케줄러는 복수의 작업(DEMUX, V-DEC, A-DEC, TEXT, PROG, BLEND)이 구조적 기술(117)에 기초해서 프로그램 모듈(100) 내의 프로그램(111～116)에 의해 실행되는 순서를 결정한다. 그 후, 스케줄러는 작업을 제1작업그룹과 제2작업그룹으로 분할한다. 제2작업그룹은 제1작업그룹을 뒤따른다. 예컨대, 작업(DEMUX, V-DEC)은 제1작업그룹에 속하고, 작업(A-DEC, TEXT, PROG, BLEND)은 제2작업그룹에 속한다.

3. 스케줄러는 적어도 2개의 프로세서(VPU0, VPU1)를 사용하고, 파이프라인 모드로 제1작업그룹(DEMUX, V-DEC)과 제2작업그룹(A-DEC, TEXT, PROG, BLEND)을 주기적으로 실행하기 위해 적어도 하나의 프로세서를 제1 및 제2작업그룹의 각각에 주기적으로 할당한다. 스케줄러가 2개의 프로세서(VPU0, VPU1)를 이용하여 파이프라인 동작을 수행하는 경우에는, 1/60초의 시간간격으로 VPU0 상의 제1작업그룹을 주기적으로 실행하기 위해 제1작업그룹(DEMUX, V-DEC)을 VPU0에 주기적으로 할당한다. 스케줄러는, VPU1 상의 제2작업그룹을 제1작업그룹에 비해 1주기 지연시켜 1/60초의 시간간격으로 주기적으로 실행하기 위해 제2작업그룹(A-DEC, TEXT, PROG, BLEND)을 VPU1에 주기적으로 할당한다.

2개의 프로세서(VPU0, VPU2)는 제2작업그룹을 동시에 실행할 수 있다. 예컨대, VPU1이 작업(A-DEC, TEXT)을 실행하는 동안, VPU2는 작업(PROG, BLEND)을 실행한다.

도 7에 나타낸 프로그램 모듈(100)에 있어서는, 복수의 작업(DEMUX, V-DEC, A-DEC, TEXT, PROG, BLEND)이 서로 다른 스레드에 의해 실행된다. 따라서, 상기 작업그룹은 스레드 그룹이라 부를 수 있다.

도 7에 나타낸 프로그램 모듈(100)은 본 실시예의 컴퓨터 시스템을 갖춘 장치 내의 플래시 ROM과 하드 디스크에 미리 기록(prerecord)될 수 있거나 네트워크를 통해 전달될 수 있다. 이 경우, 컴퓨터 시스템에 의해 수행되어야 할 동작의 내용은 네트워크를 통해 다운로드되는 프로그램 모듈의 타입에 따라 바뀔 수 있다. 따라서, 컴퓨터 시스템을 갖춘 장치는 전용 하드웨어의 여러 부분의 각각에 대응하는 실시간 동작을 수행할 수 있다. 새로운 내용을 재생하는데 필요한 새로운 플레이어 소프트웨어, 디코더 소프트웨어 및 암호화(encryption) 소프트웨어가 컴퓨터 시스템에 의해 실행가능한 프로그램 모듈로서 내용과 함께 분배되면, 컴퓨터 시스템을 갖춘 어떠한 장치도 가능한 능력 내에서 내용을 재생할 수 있다.

<u>전력절약제어</u>

본 발명의 실시예에 따른 컴퓨터 시스템은, 상기 프로그램 모듈(100) 등의 실시간 동작이 제한된 시간주기 이내에 완료되도록 하면서, 전력소비를 줄이기 위해 전력절약제어를 수행한다. 실시간 동작이 특정의 시간간격으로 복수의 VPU에 의해 주기적으로 수행되는 경우에는, 특정의 시간간격 내에서 실시간 동작의 복수의 작업을 완료하고, 가능할 만큼 많은 주기 내에서 필요한 데이터 전송대역폭을 균일화하기 위해 스케줄링 동작이 수행된다. 각 작업에 의해 요구되는 데이터 전송대역폭에 기초해서, 큰 데이터 전송대역폭을 갖는 적어도 2개의 더 높은 순위의 작업의 실행기간이 서로 겹치는 것을 방지하도록 작업을 실행하는 하나 이상의 VPU와 각각의 작업의 실행시작 타이밍이 결정된다.

도 12는 VPU0와 VPU1에 의해 3개의 작업(A, B, C)을 포함하는 실시간 동작을 주기적으로 수행하도록 스케줄링하는 예를 나타낸다. 작업(A, B, C)의 각각을 실행하기 위해 필요하게 되는 총 비용(시간)이 1주기에 대응하는 시간간격보다 길다고 가정한다. 1개의 VPU는 1주기에 대응하는 시간간격 내에 3개의 작업(A, B, C)을 실행할 수 없기 때문에, 작업(A, B, C)이 VPU0 및 VPU1에 분배된다. VPU0에 의해 실행해야 할 작업의 실행기간 및 VPU1에 의해 실행해야 할 작업의 실행기간이 서로 겹치면, 도 12에 나타낸 바와 같이 겹치는 기간 내에 데이터 전송의 양(연결장치를 위해 필요하게 되는 대역폭)이 증가된다. 도 12에 있어서, 작업(A, B, C)에 의해 요구되는 버스 대역폭은 각각 100Gbps, 90Gbps 및 20Gbps이다. 작업(A, B)의 실행기간이 겹치는 기간에는, 190Gbps의 버스 대역폭이 필요하게 된다. 연결장치(버스; 13)의 데이터 전송속도는 각 주기에서 요구되는 버스 대역폭의 피크값을 만족하도록 설정할 필요가 있다. 피크값이 길어질수록, 일어나야 하는 연결장치(버스; 13)의 데이터 전송속도가 높아진다. 따라서, 연결장치(버스; 13)는 전력소비를 증가시킨다.

본 실시예에 따른 스케줄링 동작은 구조적 기술(117)에 의해 제공되는 작업(A, B, C)의 각각의 버스 대역폭을 고려하여 피크값을 최소화하도록 수행된다. 동작 시스템은, 작업(A, B, C)이 1주기에 대응하는 시간간격 내에서 실행되고, 큰 버스 대역폭을 갖는 적어도 2개의 더 높은 순위의 작업(이 경우에는 A 및 B)의 실행기간이 서로 중복하지 않도록 작업(A, B, C)의 스케줄링을 수행한다. 도 13은 작업(A, B)의 실행기간이 서로 겹치는 것을 방지하기 위한 스케줄링의 예를 나타낸다. 요구되는 데이터 전송대역폭은 주기 내에서 거의 균일하게 될 수 있고, 그 피크는 작은 값으로 낮추어질 수 있다. 그 결과, 연결장치(버스; 13)의 데이터 전송속도는 낮게 설정될 수 있고, 따라서 실시간 동작이 특정의 시간간격으로 주기적으로 작업(A, B, C)을 실행하도록 보장하면서 전력소비를 줄일 수 있게 된다.

도 14에 나타낸 플로우차트를 참조하여 전력절약 제어동작에 대한 수순을 설명한다.

단계 S1: 동작 시스템은 실시간 동작의 작업의 실행순서, 각 작업을 실행하기 위해 필요하게 되는 비용 및 각 작업에 의해 요구되는 데이터 전송대역폭을 체크하기 위해 외부기억장치 또는 메모리(14)로부터 구조적 기술(117)을 수신한다.

단계 S2: 상기의 실행순서, 비용 및 데이터 전송대역폭에 기초해서, 동작 시스템은 작업을 실행하는 하나 이상의 VPU와, (1) 상기 작업의 실행순서의 제약조건을 만족시키기 위한 조건, (2) 1주기에 대응하는 시간간격 내에 모든 작업을 실행하기 위한 조건 및 (3) 데이터 전송대역폭이 그 이외의 작업의 데이터 전송대역폭과 같거나 그보다 큰 적어도 2개의 더 높은 순위의 작업의 실행기간이 겹치는 것을 방지하기 위한 조건의 3가지 조건을 만족시키도록 각 작업의 실행시작 타이밍을 결정하는 스케줄링 동작을 수행한다.

단계 S3: 동작 시스템은 단계 S2에서의 스케줄링의 결과 및 각 작업에 의해 요구되는 데이터 전송대역폭에 기초해서 시간간격 내에서 결정된 하나 이상의 VPU에 의해 수행되어야 할 데이터 전송의 데이터 전송대역폭의 피크값을 계산한다(결정한다).

단계 S4: 동작 시스템은 연결장치(버스; 13)의 데이터 전송능력 및 계산된 피크값에 기초해서 그 계산된 피크값대 연결장치(버스; 13)의 데이터 전송대역폭(최대 버스 대역폭)의 비율을 계산한다.

단계 S5: 동작 시스템은 단계 S4에서 계산된 비율에 기초해서 연결장치(버스; 13)의 데이터 전송속도를 최대 데이터 전송대역폭보다 낮은 값으로 설정한다. 연결장치(버스; 13)의 데이터 전송속도는 장치(13)의 최대 데이터 전송대역폭을 계산된 비율에 곱함으로써 얻어질 수 있다. 동작 시스템은 장치(13)의 동작 주파수 또는 장치(13)의 버스 대역폭을 지정하기 위한 명령(command)을 전력절약 제어기(17)로 송신한다. 전력절약 제어기(17)는 장치(13)의 동작 주파수 또는 그 버스 대역폭을 제어하기 위한 회로를 포함하고 있다. 전력절약 제어기(17)는 동작 주파수 또는 버스 대역폭을 동작 시스템으로부터의 명령에 의해 지정되는 것으로 설정한다.

다음에는 도 7에 나타낸 프로그램 모듈(100)에 의해 수행되는 디지탈 TV방송 수신동작에 대한 전력절약제어에 대해 설명한다.

도 15는 하나의 VPU가 디지탈 TV방송 수신동작을 수행할 때 요구되는 버스 대역폭을 나타낸다. 디지탈 TV방송 수신동작은 다수의 작업(D: DEMUX, V: V-DEC, A: A-DEC, T: TEXT, P: PROG, B: BLEND)을 내포하고 있다. 이들 작업 중 버스 대역폭이 큰 작업(BLEND)이 단독으로 실행된다. 따라서, 각 주기에 있어서 요구되는 버스 대역폭의 피크값은 그 작업(BLEND)에 의해 요구되는 버스 대역폭과 일치한다.

도 16은 2개의 채널에 대한 디지탈 TV방송 수신동작이 동시에 수행될 때 요구되는 버스 대역폭을 나타낸다. VPU0는 한 채널에 대한 디지탈 TV방송 수신동작을 수행하고, VPU1은 다른 채널에 대한 디지탈 TV방송 수신동작을 수행한다. 버스 대역폭이 가장 큰 작업(BLEND)이 서로 겹치기 때문에, 각 주기에 있어서 요구되는 버스 대역폭의 피크값은 크게 증가한다.

도 17은 본 발명의 실시예에 따른 스케줄링 방법을 통해 2개의 채널에 대한 디지탈 TV방송 수신동작을 수행하기 위한 스케줄링의 예를 나타낸다. 이 예에서는, VPU1에 의해 실행되어야 할 작업(BLEND)의 실행기간은 버스 대역폭이 가장 긴 작업(BLEND)이 서로 겹치는 것을 방지하기 위해 실행되는 작업이 없는 주기의 예비비시간을 이용하여 시프트된다. 따라서, 각 주기에서 요구되는 버스 대역폭의 피크값은 도 16에서의 값의 반으로 줄일 수 있다.

상술한 바와 같이, 각 작업에 의해 요구되는 버스 대역폭을 고려하여 수행되는 본 실시예의 스케줄링 동작은 하나의 실시간 동작에 포함된 작업들에 대한 스케줄링뿐만 아니라 특정의 시간간격 내에 수행될 필요가 있는 2개 이상의 실시간 동작

의 각각에 대한 스케줄링에도 적용될 수 있다. 각 실시간 동작은 하나 이상의 작업을 포함하고 있고, 실시간 동작 사이에 실행동작의 제약조건은 없다. 2개의 실시간 동작의 각각이 하나의 작업만을 포함하고 있는 경우는, 작업을 실행하기 위해 필요하게 되는 비용과 작업의 버스 대역폭의 양쪽에만 기초해서 그들 작업의 실행기간이 서로 겹치는 것을 방지하도록 스케줄링을 수행할 수 있다.

동작 시스템

본 실시예의 컴퓨터 시스템으로 하나의 OS(operating system; 201)만이 로드되는 경우에는, 도 18에 나타낸 바와 같이 그 OS가 실제의 리소스(MPU(11), VPU(12), 메모리(14), I/O 제어기(15), I/O 장치(16) 등)를 전부 관리하게 된다.

한편, 가상 기계 시스템은 복수의 OS를 동시에 수행할 수 있다. 이 경우, 도 19에 나타낸 바와 같이, 가상 기계 OS(301)는 실제의 리소스(MPU(11), VPU(12), 메모리(14), I/O 제어기(15), I/O 장치(16) 등)를 전부 관리하기 위해 컴퓨터 시스템으로 로드된다. 가상 기계 OS(301)는 호스트(host) OS라고도 불리운다. 게스트(guest) OS라고도 불리우는 하나 이상의 OS(302, 303)가 가상 기계 OS(301)로 로드된다. 도 20을 참조하면, 게스트 OS(302, 303)는 각각 가상 기계 OS(301)에 의해 주어진 가상 기계 리소스를 포함하는 컴퓨터 상에서 실행되고, 게스트 OS(302, 303)에 의해 관리되는 응용 프로그램에 갖가지 서비스를 제공한다. 도 20의 예에서는, 게스트 OS(302)는 1개의 MPU(11), 2개의 VPU(12) 및 1개의 메모리(14)를 갖추고 있는 컴퓨터상에서 동작되는 것처럼 보이고, 게스트 OS(303)는 1개의 MPU(11), 4개의 VPU(12), 1개의 메모리(14)를 갖추고 있는 컴퓨터상에서 동작되는 것처럼 보인다. 가상 기계 OS(301)는 실제의 리소스의 어느 VPU(12)가 게스트 OS(302)로부터 보여진 VPU(12)와 게스트 OS(303)로부터 보여진 VPU(12)에 실질적으로 대응하는지를 관리한다. 게스트 OS(302, 303)는 그 대응관계를 알 필요는 없다.

가상 기계 OS(301)는 시분할에 기초해서 컴퓨터 시스템의 모든 리소스를 게스트 OS(302, 303)에 할당하기 위해 게스트 OS(302, 303)를 스케줄링한다. 게스트 OS(302)는 실시간 동작을 실행한다고 가정한다. 정확한 페이스(pace)로 초당 30번의 동작을 수행하기 위해, 게스트 OS(302)는 그 파라미터를 가상 기계 OS(301)에 세트한다. 가상 기계 OS(301)는, 필요한 동작시간을 1/30초마다 한번씩 게스트 OS(302)에 신뢰성 좋게 할당하도록 게스트 OS(302)를 스케줄링한다. 동작시간은 실시간 동작을 요구하는 게스트 OS보다 우선순위가 낮은 실시간 동작을 요구하지 않는 게스트 OS에 할당된다. 도 21은 수평축이 시간을 나타내고, 게스트 OS(302, 303)가 교대로 실행되는 것을 나타내고 있다. 게스트 OS(OS1; 302)가 실행되는 동안, MPU(11)와 모든 VPU(12)가 게스트 OS(OS1; 302)의 리소스로서 사용된다. 게스트 OS(OS2; 303)가 실행되는 동안, MPU(11)와 모든 VPU(12)는 게스트 OS(OS2; 303)의 리소스로서 사용된다.

도 22는 다른 동작모드를 나타낸다. VPU(12)가 목표하는 어플리케이션에 따라 계속해서 사용되는 것을 바라는 경우가 있다. 예컨대, 데이터와 이벤트의 감시를 계속해서 필요로 하는 응용의 경우이다. 가상 기계 OS(301)의 스케줄러는, 게스트 OS가 특정한 VPU(12)를 점유하도록 특정한 게스트 OS의 스케줄링을 관리한다. 도 22에서, VPU(3)는 게스트 OS(OS1; 302)를 위해 리소스로서 독점적으로 지정된다. 가상 기계 OS(301)가 게스트 OS(OS1; 302)와 게스트 OS(OS2; 303)를 서로 전환한다고 하더라도, VPU(3)는 게스트 OS(OS1; 302)의 제어 하에 항상 동작을 계속한다.

본 실시예에서는, 복수의 VPU(12)를 이용하여 프로그램을 실행하기 위해, VPU 실행시간(runtime) 환경이라 불리는 소프트웨어 모듈이 이용된다. 소프트웨어 모듈은, VPU(12)에 할당될 스레드를 스케줄링하기 위한 스케줄러를 갖추고 있다. 하나의 OS(201)만이 본 실시예의 컴퓨터 시스템에서 구현될 때, VPU 실행시간 환경(401)은 도 23에 나타낸 바와 같이 OS(201)로 구현된다. VPU 실행시간 환경(401)은 OS(201)의 커널(kernel)이나 사용자 프로그램으로 구현될 수 있고, 또 커널이나 사용자 프로그램이 서로 협력하여 실행되도록 2개로 분할될 수 있다. 하나 이상의 게스트 OS가 가상 기계 OS(301)에서 실행될 때, VPU 실행시간 환경(401)을 구현하기 위해 다음의 모드가 제공된다.

1. 가상 기계 OS(301; 도 24)로 VPU 실행시간 환경(401)을 구현하는 모드.

2. 가상 기계 OS(301; 도 25)에 의해 관리되는 하나의 OS로서 VPU 실행시간 환경(401)을 구현하는 모드. 도 25에 있어서, 가상 기계 OS(301)에서 실행되는 게스트 OS(304)는 VPU 실행시간 환경(401)이다.

3. 가상 기계 OS(301; 도 26)에 의해 관리되는 각 게스트 OS에서 전용 VPU 실행시간 환경을 구현하는 모드. 도 26에 있어서, VPU 실행시간 환경(401, 402)은 그들 각각의 게스트 OS(302, 303)로 구현된다. VPU 실행시간 환경(401, 402)은 서로 관련되어 실행되고, 필요한 경우 가상 기계 OS(301)에 의해 제공되는 게스트 OS간의 통신기능을 사용한다.

4. 가상 기계 OS(301; 도 27)에 의해 관리되는 게스트 OS 중 하나로 VPU 실행시간 환경(401)을 구현하는 모드. VPU 실행시간 환경이 없는 게스트 OS(303)는 가상 기계 OS(301)에 의해 제공되는 게스트 0S간의 통신기능을 사용함으로써 게스트 OS(302)의 VPU 실행시간 환경(401)을 사용한다.

상기의 모드는 다음과 같은 이점을 갖는다.

모드 1의 이점

가상 기계 OS(301)에 의해 관리되는 게스트 OS의 스케줄링과 VPU의 스케줄링은 하나로 결합될 수 있다. 따라서, 스케줄링은 효과적이고 정교하게 행해질 수 있고, 리소스도 효과적으로 사용될 수 있다.

그리고, VPU 실행시간 환경을 복수의 게스트 OS에서 공유할 수 있기 때문에, 새로운 게스트 0S가 도입되더라도 새로운 VPU 실행시간 환경을 생성할 필요는 없다.

모드 2의 이점

VPU를 위한 스케줄러를 가상 기계 OS의 게스트 OS 사이에서 서로 공유할 수 있기 때문에, 스케줄링을 효과적이고 정교하게 수행할 수 있고, 리소스도 효과적으로 사용할 수 있다.

VPU 실행시간 환경을 복수의 게스트 OS 사이에서 서로 공유할 수 있기 때문에, 새로운 게스트 OS가 도입되는 경우에도, 새로운 VPU 실행시간 환경을 생성할 필요는 없다.

가상 기계 OS나 특정 게스트 OS에 의존하지 않고 VPU 실행시간 환경을 생성할 수 있기 때문에, 쉽게 표준화할 수 있고 다른 것으로 교체할 수 있다. 장치의 특성을 이용하여 스케줄링을 수행하도록 내장된 특정 장치에 적합한 VPU 실행시간 환경이 생성되는 경우에는, 스케줄링을 효율좋게 행할 수 있다.

모드 3의 이점

VPU 실행시간 환경을 각 게스트 OS에 있어서 최적으로 구현할 수 있기 때문에, 스케줄링을 효과적이고 정교하게 수행할 수 있고, 리소스도 효과적으로 사용할 수 있다.

모드 4의 이점

VPU 실행시간 환경을 모든 게스트 OS에서 수행할 필요가 없기 때문에, 새로운 게스트 OS를 추가하는 것이 용이하다.

상기로부터 명백하게 된 바와 같이, 모드 1～4 전부를 VPU 실행시간 환경을 구현하기 위해 사용할 수 있다. 필요하다면, 다른 모드도 사용할 수 있다.

서비스 제공자

본 실시예에 따른 컴퓨터 시스템에서는, VPU 실행시간 환경(401)은 VPU(12)와 관련된 다양한 리소스를 관리하고 스케줄링하는 기능(각 VPU의 동작시간, 메모리, 연결장치의 대역폭 등)뿐만 아니라, 다양한 서비스(네트워크를 이용한 통신기능, 파일 입/출력 기능, 코덱과 같은 라이브러리(library) 기능을 호출하는 것, 사용자와 인터페이스하는 것, I/O 장치를 이용한 입/출력 동작, 데이터와 시간을 독출하는 것 등)를 제공한다. 이러한 서비스는 VPU(12)에서 실행되는 응용 프로그램으로부터 호출된다. 간단한 서비스가 호출되면, 이 서비스는 VPU(12)의 서비스 프로그램에 의해 처리된다. VPU(12)에 의해서만 처리될 수 없는 서비스, 즉 통신처리 및 파일처리는 MPU(11)의 서비스 프로그램에 의해 처리된다. 이러한 서비스를 제공하는 프로그램을 서비스 제공자(SP)라고 부른다.

도 28은 VPU 실행시간 환경의 예를 나타낸다. VPU 실행시간 환경의 주요부는 MPU(11)에 제공되고, MPU측 VPU 실행시간 환경(501)에 대응한다. VPU측 VPU 실행시간 환경(502)은 VPU(12)의 각각에 제공되고, VPU(12)에서 처리될 수 있는 서비스를 수행하는 최소한의 기능만을 가진다. MPU측 VPU 실행시간 환경(501)의 기능은 VPU 제어기(511)와 서비스 중개자(service broker; 512)로 개략적으로 분할된다. VPU 제어기(511)는 VPU(12)에 관련된 갖가지 리소스(각 VPU의 동작시간, 메모리, 가상공간, 연결장치의 대역폭 등)를 위해 주로 관리 메커니즘, 동기화 메커니즘, 안전관리 메커니즘

및 스케줄링 메커니즘을 제공한다. VPU 제어기(511)는 스케줄링의 결과에 기초해서 VPU(12)로 프로그램을 발송한다. 각 VPU(12)의 응용 프로그램에 의해 호출된 서비스 요청을 받자마자, 서비스 중개자(512)는 알맞은 서비스 프로그램을 호출하여 서비스를 제공한다.

각 VPU(12)의 응용 프로그램에 의해 호출된 서비스 요청을 받자마자, VPU측 VPU 실행시간 환경(502)은 VPU(12)에서 처리될 수 있는 서비스만을 처리하고, 서비스 중개자(512)에게 처리할 수 없는 서비스를 처리하라고 요청한다.

도 29는 VPU측 VPU 실행시간 환경(502)에 의한 서비스 요청을 처리하기 위한 수순을 나타낸다. 응용 프로그램으로부터 서비스 호출(service call)을 받자마자(단계 S101), VPU측 VPU 실행시간 환경(502)은 그 서비스를 처리할 수 있는지의 여부를 결정(판단)한다(단계 S102). 그 서비스를 처리할 수 있는 경우에는, VPU 실행시간 환경(502)은 서비스를 실행하고, 그 결과를 호출된 부분으로 반환한다(단계 S103와 단계 S107). 그렇지 않은 경우에는, VPU 실행시간 환경(502)은 서비스를 실행할 수 있는 서비스 프로그램이 각 VPU(12)에 실행가능하게 등록되어 있는지의 여부를 결정한다(단계 S104). 서비스 프로그램이 등록되어 있는 경우에는, VPU 실행시간 환경(502)은 그 서비스 프로그램을 실행하고, 그 결과를 호출된 부분으로 반환한다(단계 S105와 단계 S107). 그렇지 않은 경우에는, VPU 실행시간 환경(502)은 서비스 중개자(512)에게 서비스 프로그램을 실행할 것을 요청하고, 서비스 중개자(512)로부터의 서비스의 결과를 호출된 부분으로 반환한다(단계 S106과 단계 S107).

도 30은 MPU측 VPU 실행시간 환경(501)의 서비스 중개자(512)에 의해 VPU측 VPU 실행시간 환경(502)으로부터 요청된 서비스를 처리하기 위한 수순을 나타낸다. VPU측 VPU 실행시간 환경(502)으로부터 서비스 호출을 받자마자, 서비스 중개자(512)는 VPU 실행시간 환경(501)이 그 서비스를 처리할 수 있는지의 여부를 결정한다(단계 S112). 그 서비스를 처리할 수 있는 경우에는, 서비스 중개자(512)는 서비스를 실행하고, 그 결과를 호출된 부분의 VPU측 VPU 실행시간 환경(502)으로 반환한다(단계 S113과 단계 S114). 그렇지 않은 경우에는, 서비스 중개자(512)는 서비스를 실행하는 서비스 프로그램이 MPU(11)에서 실행가능하게 등록되어 있는지의 여부를 결정한다(단계 S114). 서비스 프로그램이 등록되어 있는 경우에는, 서비스 중개자(512)는 서비스 프로그램을 실행하고, 그 결과를 호출된 부분의 VPU측 VPU 실행시간 환경(502)으로 반환한다(단계 S116과 단계 S114). 그렇지 않은 경우에는, 서비스 중개자(512)는 호출된 부분의 VPU측 VPU 실행시간 환경(502)으로 에러(error)를 반환한다(단계 S117).

그들 결과는, 각 VPU(12)에 의해 실행되는 프로그램으로부터 발행된 몇몇 서비스 요청에 응답하지만, 다른 서비스 요청에는 응답하지 않는다. 응답의 수신지는 통상 서비스 요청을 발행하는 스레드이지만, 또 다른 스레드, 스레드 그룹 또는 프로세스가 응답의 수신지로서 지정될 수 있다. 따라서, 서비스를 요청하는 메시지에 수신지를 포함시키는 것이 바람직하다. 서비스 중개자(512)는 널리 사용되는 객체 요청 중개자(object request broker)를 이용하여 실현될 수 있다.

실시간 동작

본 실시예에 따른 컴퓨터 시스템은 실시간 처리시스템으로서 작용한다. 실시간 처리시스템에 의해 수행되는 동작은 대략적으로 다음의 3가지 타입으로 나눌 수 있다.

1. 하드 실시간 동작(hard real-time operation)

2. 소프트 실시간 동작(soft real-time operation)

3. 베스트 에포트(best effort: 최상 노력) 동작(비실시간 동작)

하드 및 소프트 실시간 동작은 실시간 동작이라 불린다. 본 실시예의 실시간 처리시스템은 현존하는 복수의 OS처럼 스레드와 프로세스 양자의 개념을 갖는다. 우선, 실시간 처리시스템의 스레드와 프로세스를 설명한다.

스레드는 다음의 3가지 클래스(class)를 갖는다.

1. 하드 실시간 클래스

타이밍 요구조건이 매우 중요하다. 이 스레드 클래스는 요구조건이 만족되지 않는 경우에 중대한 상황을 초래할 수 있는 매우 중요한 응용에 사용된다.

2. 소프트 실시간 클래스

이 스레드 클래스는 타이밍 요구조건이 만족되지 않는다고 하더라도 품질이 단순히 더 낮은 응용에 사용된다.

3. 베스트 에포트 클래스

이 스레드 클래스는 타이밍 요구조건이 없는 응용에 사용된다.

본 실시예에서, 스레드는 실시간 동작을 위한 실행 유니트이다. 이들 스레드는 그들 스레드에 의해 실행되는 관련된 프로그램을 갖는다. 각 스레드는 스레드 문맥(context)이라고 불리우는 고유의 정보를 갖는다. 예컨대, 스레드 문맥은 스택의 정보 및 프로세서의 레지스터에 저장된 값(value)을 포함한다.

실시간 처리시스템에 있어서는, MPU와 VPU 스레드의 2개의 스레드가 있다. 이러한 2개의 스레드는 프로세서(MPU(11), VPU(12))에 의해 분류되고, 그 모델은 서로 동일하다. VPU 스레드의 스레드 문맥은 VPU(12)의 로컬기억장치(32)의 내용 및 메모리 제어기(33)의 DMA 제어기의 조건을 포함한다.

스레드의 그룹은 스레드 그룹이라 불린다. 스레드 그룹은, 예컨대 그 그룹의 스레드에 동일한 속성을 주는 동작을 효과적이고 쉽게 수행한다는 이점을 갖는다. 하드 또는 소프트 실시간 클래스의 스레드 그룹은 대략 단단히 결합된 스레드 그룹과 느슨히 결합된 스레드 그룹으로 나누어진다. 단단히 결합된 스레드 그룹과 느슨히 결합된 스레드 그룹은 스레드 그룹에 부가된 속성정보(결합속성정보)에 의해 서로 구별된다. 스레드 그룹의 결합속성은 상술한 구조적 기술 또는 응용 프로그램의 코드에 의해 명백히 지정될 수 있다.

단단히 결합된 스레드 그룹은 서로 협력하여 실행하는 스레드로 이루어진 스레드 그룹이다. 즉, 단단히 결합된 스레드 그룹에 속하는 스레드는 서로 단단히 협력한다. 단단한 협력은 빈번한 스레드간 통신 및 동기화와 같은 상호작용, 또는 지연(latency)을 감소시키는 상호작용을 의미한다. 단단히 결합된 동일 스레드 그룹에 속하는 스레드는 항상 동시에 실행된다. 반면에, 느슨히 결합된 스레드 그룹은 그 그룹에 속하는 스레드 사이에서 단단한 협력을 방지하는 스레드 그룹이다. 느슨히 결합된 스레드 그룹에 속하는 스레드는 메모리(14)의 버퍼를 통해 데이터를 전송하기 위한 통신을 수행한다.

단단히 결합된 스레드 그룹

도 31에 나타낸 바와 같이, 서로 다른 VPU가 단단히 결합된 스레드 그룹의 스레드에 할당되고, 그들 스레드가 동시에 실행된다. 이들 스레드는 단단히 결합된 스레드라 불린다. 단단히 결합된 스레드의 실행기간은 그들 각각의 VPU에서 예약되고, 단단히 결합된 스레드는 동시에 실행된다. 도 31에 있어서, 단단히 결합된 스레드 그룹은 2개의 단단히 결합된 스레드 A와 B를 포함하고 있고, 스레드 A와 B는 VPU0과 VPU1에 의해 각각 동시에 실행된다. 본 실시예의 실시간 처리시스템은 스레드 A와 B가 다른 VPU에 의해 동시에 실행되도록 한다. 스레드 중 하나는 다른 스레드를 실행하는 VPU의 제어 레지스터나 로컬기억장치를 통해 다른 스레드와 직접 통신할 수 있다.

도 32는 스레드 A와 B 사이의 통신을 나타내는데, 이것은 각각 스레드 A와 B를 실행하는 VPU0과 VPU1의 로컬기억장치를 통해 수행된다. 스레드 A를 실행하는 VPU0에 있어서는, 스레드 B를 실행하는 VPU1의 로컬기억장치(32)에 대응하는 RA공간이 스레드 A의 EA공간의 일부에 사상된다. 이러한 사상을 위해, VPU0의 메모리 제어기(33)에 제공된 주소번역 유니트(331)가 세그먼트 테이블과 페이지 테이블을 이용하여 주소번역을 수행한다. 주소번역 유니트(331)는 스레드 A의 EA공간의 일부를 VPU1의 로컬기억장치(32)에 대응하는 RA공간으로 변환(번역)하고, 그에 따라 VPU1의 로컬기억장치(32)에 대응하는 RA공간을 스레드 A의 EA공간의 일부에 사상한다. 스레드 B를 실행하는 VPU1에 있어서는, 스레드 A를 실행하는 VPU0의 로컬기억장치(32)에 대응하는 RA공간은 스레드 B의 EA공간의 일부에 사상된다. 이러한 사상을 위해, VPU1의 메모리 제어기(33)에 제공되는 주소번역 유니트(331)가 세그먼트 테이블과 페이지 테이블을 이용하여 주소번역을 수행한다. 주소번역 유니트(331)는 스레드 B의 EA공간의 일부를 VPU0의 로컬기억장치(32)에 대응하는 RA공간으로 변환하고, 그에 따라 VPU0의 로컬기억장치(32)에 대응하는 RA공간을 스레드 B의 EA공간의 일부에 사상한다.

도 33은 스레드 B를 실행하는 VPU1의 로컬기억장치(LS1; 32)가 VPU0에 의해 실행되는 스레드 A의 EA공간에 사상되는 것과, 스레드 A를 실행하는 VPU0의 로컬기억장치(LS0; 32)가 VPU1에 의해 실행되는 스레드 B의 EA공간에 사상되는 것을 나타낸다. 예컨대, 스레드 B로 전송될 데이터가 로컬기억장치(LS0)에 준비될 때, 스레드 A는 이러한 준비를 나타내는 플래그(flag)를 스레드 B를 실행하는 VPU1의 로컬기억장치(LS1), 또는 VPU0의 로컬기억장치(LS0)에 설정한다. 플래그의 설정에 응답하여, 스레드 B는 로컬기억장치(LS0)로부터 데이터를 독출한다.

상술한 본 실시예에 따르면, 단단히 결합된 스레드는 결합속성정보에 의해 구체화될 수 있고, 단단히 결합된 스레드 A와 B는 각기 다른 VPU에 의해 동시에 실행될 수 있다. 따라서, 스레드 A와 B간의 통신 및 동기의 상호작용은 지연없이 보다 쉽게 수행될 수 있다.

느슨히 결합된 스레드 그룹

느슨히 결합된 스레드 그룹에 속하는 각 스레드의 실행기간은 스레드간의 입/출력의 관계에 의존하고 있다. 스레드는 실행순서에 아무런 제약을 받지 않는다고 하더라도, 그것들이 동시에 실행된다는 보장은 없다. 느슨히 결합된 스레드 그룹에 속하는 스레드는 느슨히 결합된 스레드라 불린다. 도 34는 느슨히 결합된 스레드로서 스레드 C와 D를 포함하는 느슨히 결합된 스레드 그룹을 보여주는데, 각각 VPU0과 VPU1에 의해 실행된다. 도 34로부터 명백하게 된 바와 같이, 스레드 C와 D는 실행기간이 서로 다르다. 스레드 C와 D간의 통신은 도 35에 나타낸 바와 같은 주메모리(14)에 준비된 버퍼에 의해 수행된다. VPU0에 의해 실행되는 스레드 C는, 로컬기억장치(LS0)에 준비된 데이터를 DMA 전송에 의해 주메모리(14)에 준비된 버퍼에 기록한다. VPU1에 의해 실행되는 스레드 D는 주메모리(14)에 있는 버퍼로부터 데이터를 독출하고, 스레드 D가 실행되기 시작할 때, DMA전송에 의해 로컬기억장치(LS1)에 기록한다.

프로세스와 스레드

도 36에 나타낸 바와 같이, 프로세스는 하나의 주소공간과 하나 이상의 스레드를 포함한다. 스레드는 숫자나 타입에는 상관없이 프로세서에 포함될 수 있다. 예컨대, VPU 스레드만이 프로세스에 포함될 수 있고, 따라서 VPU와 MPU 스레드의 혼합으로 될 수 있다. 스레드가 그 고유의 정보로서 스레드 문맥을 유지하는 것처럼, 프로세스는 그 고유의 정보로서 프로세스 문맥을 유지한다. 프로세스 문맥은 프로세스 고유의 주소공간과 프로세스에 포함되는 모든 스레드의 스레드 문맥의 양자를 포함한다. 주소공간은 프로세스의 모든 스레드 사이에서 공유될 수 있다. 하나의 프로세스는 복수의 스레드 그룹을 포함할 수 있으나, 하나의 스레드 그룹은 복수의 프로세스에 속할 수 없다. 따라서, 프로세스에 속하는 스레드 그룹은 프로세스에 있어서 고유한 것이다.

본 실시예의 실시간 처리시스템에 있어서, 새로운 스레드를 생성하는 방법으로서는, 스레드 우선 모델(thread first model)과 주소공간 우선 모델(address space first model)의 2가지 모델이 있다. 주소공간 우선 모델은 현재 OS에 채택된 것과 동일하기 때문에, MPU와 VPU 스레드의 양자에 적용될 수 있다. 반면에, 스레드 우선 모델은 VPU 스레드에만 적용될 수 있고, 본 실시예의 실시간 처리시스템에 특별한 것이다. 스레드 우선 모델에 있어서, 현재의 스레드(새로운 스레드를 만들기 위한 스레드, 즉 새로운 스레드의 부모 스레드)는 새로운 스레드에 의해 수행될 프로그램을 지정하고, 새로운 스레드가 프로그램을 실행하는 것을 시작하도록 한다. 이어서 프로그램은 VPU의 로컬기억장치에 저장되고, 소정 주소로부터 실행되기 시작한다. 이 때 아무런 주소공간도 새로운 스레드와 관련되어 있지 않기 때문에, 새로운 스레드는 VPU의 로컬기억장치에만 접근할 수 있고, 메모리(14)로의 접근은 불가능하다. 그 후, 필요가 생기면, 새로운 스레드는 VPU 실행시간 환경의 서비스를 요청하고, 주소공간을 생성한다. 주소공간은 새로운 스레드와 관련되어 있고, 새로운 스레드는 메모리(14)에 접근할 수 있게 된다. 주소공간 우선 모델에서는, 현재의 스레드가 새로운 주소공간을 생성하거나, 현재의 주소공간을 지정하고, 새로운 스레드에 의해 실행할 프로그램을 주소공간에 할당한다. 그 후, 새로운 스레드가 프로그램을 실행하기 시작한다. 스레드 우선 모델의 장점은, 스레드를 생성하고, 발송하며, 내보내는데 요구되는 총비용을 줄이기 위해 스레드가 로컬기억장치에 의해서만 실행될 수 있다는 점이다.

스레드의 스케줄링

도 37에 나타낸 플로우차트를 참조하여 VPU 실행시간 환경(401)에 의해 수행되는 스케줄링 동작에 대해 설명한다. VPU 실행시간 환경(401)의 스케줄러는, 스케줄링되기 위해 스레드의 각 그룹에 부가된 결합속성정보에 기초해서 스레드간의 결합속성을 체크한다(단계 S121). 스케줄러는 각 스레드 그룹이 단단히 결합된 스레드 그룹인지, 느슨히 결합된 스레드 그룹인지를 결정한다(단계 S122). 결합속성은 프로그램 코드의 스레드 기술이나, 상기 구조적 기술(117)의 스레드 파라미터를 참조하여 체크된다. 단단히 결합된 스레드 그룹과 느슨히 결합된 스레드 그룹이 각각 구체화되면, 스케줄링되는 스레드가 단단히 결합된 스레드 그룹과 느슨히 결합된 스레드 그룹으로 분리된다.

단단히 결합된 스레드 그룹에 속하는 스레드의 스케줄링은 다음과 같이 수행된다. 스케줄링될 스레드로부터 선택된 단단히 결합된 스레드 그룹의 스레드를, 각 VPU에 의해 동시에 실행하기 위해, VPU 실행시간 환경(401)의 스케줄러가 스레드의 수와 동일한 VPU의 각 실행기간을 예약하고, VPU에 스레드를 동시에 발송한다(단계 S123). 스케줄러는 스레드를 실행하는 VPU의 주소번역 유니트(331)를 이용하여 스레드의 EA공간의 일부에 RA공간을 사상하는데(단계 S124), RA공간

은 이전 스레드와 상호작용하는 파트너 스레드를 실행하는 VPU의 로컬기억장치에 대응한다. 스케줄링될 스레드로부터 선택되는 느슨히 결합된 스레드 그룹에 속하는 스레드에 관해서는, 스케줄러는 스레드간의 입/출력 관계에 기초해서 하나 이상의 VPU로 연속해서 스레드를 발송한다(단계 S125).

서로 협력해서 실행하는 스레드의 세트인 단단히 결합된 스레드 그룹이 결합속성정보에 기초해서 선택되는 경우에는, 단단히 결합된 스레드 그룹에 속하는 스레드가 다른 프로세서에 의해 동시에 실행된다는 것을 보증할 수 있다. 따라서, 스레드간 통신은, 서로 파트너 스레드를 실행하는 프로세서의 레지스터에 대해 직접 접근을 얻는 경량(lightweight)의 메커니즘에 의해 달성될 수 있다. 따라서, 통신은 쉽고 빠르게 수행될 수 있다.

스레드의 상태천이

스레드는 일반적으로 생성될 때로부터 삭제될 때까지 상태천이(state transition)를 한다. 도 38에 나타낸 바와 같이, 스레드는 다음과 같이 7개의 상태천이를 한다.

1. 존재하지 않는 상태(Not-existent state): 이 상태는 논리적인 것으로, 유효 스레드에서는 존재하지 않는다.

2. DORMANT 상태: 스레드가 생성되나, 아직 실행을 시작하지는 않는다.

3. READY 상태: 스레드가 실행을 시작할 준비가 되어 있다.

4. WAITING 상태: 스레드가 실행을 시작할 조건이 만족되기를 기다린다.

5. RUNNING 상태: 스레드가 VPU나 MPU에서 실제적으로 실행된다

6. SUSPENDED 상태: 스레드가 VPU 실행시간 환경이나 다른 스레드에 의해 강제로 중지된다.

7. WAITING-SUSPENDED 상태: 대기와 중지상태가 서로 겹쳐진다.

상기 7개의 상태 사이의 천이조건과, 천이에 관련된 스레드 문맥은 다음과 같다.

[NOT EXISTENT 상태로부터 DORMANT상태로의 천이]

이 천이는 스레드를 생성함으로써 행해진다.

스레드 문맥이 생성되나, 그 내용은 초기상태에 있다.

[DORMANT 상태로부터 NOT EXISTENT 상태로의 천이]

이 천이는 스레드를 삭제함으로써 행해진다.

스레드가 그 스레드 문맥을 저장하도록 설정되면, 저장된 스레드 문맥은 천이에 의해 폐기된다.

[DORMANT 상태로부터 WAITING 상태로의 천이]

이 천이는 스레드가 실행시간 환경에 스레드를 스케줄링할 것을 요청할 때 행해진다.

[WAITING 상태로부터 READY 상태로의 천이]

이 천이는 스레드가 기다리는 이벤트(예컨대, 동기화, 통신, 타이머 인터럽션)가 생성될 때 행해진다.

[READY 상태로부터 RUNNING 상태로의 천이]

이 천이는 스레드가 실행시간 환경에 의해 MPU나 VPU로 발송될 때 행해진다.

스레드 문맥이 로드된다. 스레드 문맥이 저장될 때, 그것이 복구된다.

[RUNNING 상태로부터 READY 상태로의 천이]

이 천이는 스레드의 실행이 선취(先取)되었을 때에 행해진다.

[RUNNING 상태로부터 WAITING 상태로의 천이]

이 천이는 스레드가 동기화 메커니즘, 통신 메커니즘 등을 이용하여 이벤트를 기다리기 위해 실행을 중지할 때에 행해진다.

모든 클래스의 스레드는 그 스레드 문맥을 저장하도록 설정될 수 있다. 스레드가 그 스레드 문맥을 저장하도록 설정되어 있는 경우에는, 스레드가 RUNNING 상태로부터 WAITING 상태로 천이할 때 실행시간 환경에 의해 스레드 문맥이 저장된다. 스레드가 DORMANT 상태로 천이하지 않는 한, 저장된 스레드 문맥은 유지되고, 스레드가 RUNNING 상태로 천이할 때 복구된다.

[RUNNING 상태로부터 SUSPENDED 상태로의 천이]

이 천이는 스레드의 실행이 실행시간 환경이나 다른 스레드의 지시에 응답하여 강제로 중지될 때 행해진다.

모든 클래스의 스레드는 그 스레드 문맥을 저장하도록 설정될 수 있다. 스레드가 그 스레드 문맥을 저장하도록 설정되어 있는 경우에는, 스레드가 RUNNING 상태로부터 SUSPENDED 상태로 천이할 때 실행시간 환경에 의해 스레드 문맥이 저장된다. 스레드가 DORMANT 상태로 천이하지 않는 한, 저장된 스레드 문맥은 유지되고, 스레드가 RUNNING 상태로 천이할 때 복구된다.

[RUNNING 상태로부터 DORMANT상태로의 천이]

이 천이는 스레드가 그 자신의 실행을 본질적으로 벗어날 때 행해진다.

스레드가 그 스레드 문맥을 저장하도록 설정되어 있을 때, 그 스레드 문맥의 내용은 천이에 의해 폐기된다.

[WAITING 상태로부터 WAITING-SUSPENDED 상태로의 천이]

이 천이는, WAITING 상태에서 이벤트를 생성하는 것을 기다리고 있는 동안 외부로부터의 지시에 의해 스레드가 강제로 정지될 때에 행해진다.

[WAITING-SUSPENDED 상태로부터 WAITING 상태로의 천이]

이 천이는, 스레드가 WAITING-SUSPENDED 상태에 있는 동안, 외부로부터의 지시에 의해 실행을 재개할 때에 행해진다.

[WAITING-SUSPENDED 상태로부터 SUSPENDED 상태로의 천이]

이 천이는 스레드가 WAITING 상태에서 기다리고 있는 이벤트가 생성될 때에 행해진다.

[SUSPENDED 상태로부터 READY 상태로의 천이]

이 천이는 스레드가 외부로부터의 지시에 의해 실행을 재개할 때에 행해진다.

[READY 상태로부터 SUSPENDED 상태로의 천이]

이 천이는 외부 환경에 의해 스레드가 실행을 정지할 때에 행해진다.

스레드의 실행기간

VPU가 할당되는 스레드의 실행상태의 기간을 실행기간이라고 한다. 일반적으로, 스레드의 생성에서부터 삭제까지의 기간은 스레드의 복수의 실행기간을 포함한다. 도 39는 생성에서부터 삭제까지 변화하는 스레드 상태의 일례를 나타낸다. 이러한 예는 스레드가 존재하는 동안의 2개의 실행기간을 포함한다. 스레드 문맥은 저장될 수 있고, 갖가지 방법을 이용하여 복구될 수 있다. 대부분의 일반적인 스레드는 실행기간의 끝부분에서 문맥을 저장하고, 다음 실행기간의 시작부분에서 문맥을 복구하도록 실행한다. 어떤 주기적인 동작에서는, 스레드는 실행기간의 시작부분에서 새로운 문맥을 생성하고, 실행기간동안 문맥을 이용하며, 모든 주기에서의 실행의 끝부분에서 문맥을 폐기하도록 실행한다.

단단히 결합된 스레드 그룹에 속하는 스레드의 실행기간

도 40은 마찬가지로 단단히 결합된 스레드 그룹에 속하는 스레드의 실행기간을 나타낸다. 어떤 단단히 결합된 스레드 그룹에 속하는 모든 스레드는, 한 실행기간동안 동시에 실행될 수 있도록 VPU 실행시간 환경(401)에 의해 스케줄링된다. 이러한 단단히 결합된 스레드 그룹은 주로 하드 실시간 스레드를 위해 이용된다. 따라서, 이 동작을 실행하기 위해, 실행기간이 하드 실시간 클래스에 관해 예약되어 있을 때, VPU 실행시간 환경(401)은 동시에 이용되는 프로세서와 그 수를 지정한다. 더욱이, VPU 실행시간 환경(401)은 프로세서에 대응하여 동시에 실행하는 스레드 문맥을 만든다.

어떤 실행기간에 단단히 결합된 스레드 그룹에 속하는 스레드는, 다른 실행기간에서 그들의 단단히 결합된 관계를 취소함으로써 서로 분리해서 실행할 수 있다. 각 스레드는 단단히 결합된 스레드로서 실행할 것인지 또는 다른 스레드로부터 분리해서 실행할 것인지를 감지해야 하고, 그 파트너 스레드와 함께 통신 및 동기의 동작을 수행해야 한다. 스레드의 각각에는 선취(preemptive) 또는 무선취(non-preemptive)를 나타내는 속성이 갖추어져 있다. 선취속성은 스레드가 그것의 실행기간동안 선취될 수 있도록 허용하는 것으로, 즉 스레드가 실행을 정지하는 것을 허용한다. 무선취속성은 스레드가 그 실행기간동안 선취될 수 없다는 것을 보증한다. 무선취속성은 스레드 클래스에 따라 다양한 의미를 갖는다. 하드 실시간 클래스에서는, 스레드가 실행을 시작할 때, 그 실행기간이 끝날 때까지 스레드만이 본질적으로 실행을 정지할 수 있다. 소프트 실시간 클래스에서는, 선취가 필수적이고, 따라서 무선취속성은 지원되지 않는다. 베스트 에포트 클래스에서는, 스레드는 또 다른 베스트 에포트 클래스로부터 선취되는 것에서 보호될 수 있으나, 하드 실시간 클래스나 소프트 실시간 클래스와 같은 상급의 클래스로부터는 선취될 수 있다.

스레드의 실행모델

스레드의 실행모델은 대략 2가지의 모델, 즉 도 41에 나타낸 바와 같은 주기적인 실행모델과 도 42에 나타낸 바와 같은 비주기적인 실행모델로 분류될 수 있다. 주기적인 실행모델에서는, 스레드는 주기적으로 실행된다. 비주기적인 실행모델에서는, 스레드는 이벤트에 기초해서 실행된다. 주기적인 실행모델은 소프트웨어 인터럽트(interrupt)나 동기 프리미티브(primitive)와 같은 이벤트 객체를 이용하여 구현될 수 있다. 하드 실시간 클래스에 있어서, 주기적인 실행모델은 소프트웨어 인터럽트를 이용하여 구현된다. 즉, VPU 실행시간 환경(401)은 주기적인 동작을 시작하는 타이밍에 따라 소정의 방법에 의해 결정되는 스레드의 엔트리 포인트로 점프하거나, 소정의 수순에 의해 미리 등록된 콜백 기능(callback function)을 호출한다. 소프트 실시간 클래스에 있어서, 주기적인 실행모델은 이벤트 객체를 이용하여 구현된다. 즉, VPU 실행시간 환경(401)이 각 주기에서 미리 등록된 이벤트 객체의 생성을 통보하기 때문에, 소프트 실시간 스레드가 각 주기에서 이벤트 객체를 기다리고, 이벤트가 생성되면 소정의 동작을 수행함으로써, 주기적인 실행모델을 실현한다. 베스트 에포트 클래스에서는, 주기적인 실행모델은 소프트웨어 인터럽트나 이벤트 객체 중 하나를 이용하여 구현될 수 있다. 실제적인 실행은 항상 각 주기의 시작부분에서 시작되는 것이 아니라, 제약조건 내에서 지연되기도 한다.

이벤트 모델을 사용함으로써, 비주기적인 실행모델은 주기적인 실행모델로서 실현될 수 있다. 소프트 실시간 클래스와 베스트 에포트 클래스에서는, 비주기적인 실행모델은 이벤트가 통보되는 타이밍에서만 주기적인 실행모델과 다르고, 이들 모델은 구현방법에서는 동일하다. 하드 실시간 클래스에서, 시간 요구조건을 보증하는데 필요한 최소한의 상호도착(inter-arrival) 시간과 기한(deadline)은 시스템의 동작을 심각하게 제한한다. 따라서, 비주기적 실행이 제한된다.

문맥 전환

본 실시예에 따른 실시간 처리시스템에서는, VPU 스레드의 실행기간의 끝부분에서 문맥을 전환(switching: 절환)하기 위한 방법의 하나가 선택될 수 있다. 문맥을 전환하는 비용이 매우 고가이기 때문에, 한 방법을 선택하는 것은 전환의 효율을

향상시킨다. 선택된 방법은 예약된 스레드의 실행기간의 끝부분에서 사용된다. 문맥이 실행기간동안 또는 선취의 시간에 전환되는 경우, 현재의 스레드의 모든 문맥은 어떤 경우에도 저장되어야 하고, 스레드가 다음 실행을 재개할 때 복구될 필요가 있다. 예컨대, 다음과 같은 VPU문맥의 전환방법이 있다.

1. 문맥의 폐기

어떠한 문맥도 저장되지 않는다.

2. 문맥의 완전한 저장(complete saving)

레지스터 및 VPU의 로컬기억장치의 상태, 그리고 메모리 제어기에서 DMA제어기의 상태를 포함하여, VPU의 모든 문맥이 저장된다.

3. 문맥의 우아한 저장(graceful saving)

문맥 전환은 VPU의 메모리 제어기 내의 DMA제어기의 모든 동작이 끝날 때까지 지연된다. 그 후, 레지스터와 VPU의 로컬기억장치의 내용이 저장된다. 이 방법에서는, 완전한 저장뿐만 아니라 VPU의 모든 내용이 저장된다.

MPU와 VPU 스레드의 양자를 스케줄링하도록 하나의 스케줄러가 구현될 수 있고, 그들 각각의 MPU와 VPU 스레드를 스케줄링하도록 다른 스케줄러가 구현될 수 있다. MPU와 VPU 스레드는 문맥을 전환하는 비용이 다르기 때문에, 다른 스케줄러의 구현이 더욱 유효하게 된다.

하드 실시간 클래스의 스케줄링

하드 실시간 클래스의 스레드 스케줄링은 확장된 작업 그래프의 예약그래프를 이용하여 수행된다. 도 43은 작업 그래프의 예를 나타낸다. 작업 그래프는 작업간의 관계를 나타낸다. 도 43에서, 작업간의 화살표는 작업간의 의존도(작업간의 입/출력 관계)를 표시한다. 도 43의 예에 따르면, 작업 1과 2는 자유롭게 실행을 시작할 수 있고, 작업 3은 작업 1과 2의 실행이 정지된 후 실행을 시작할 수 있으며, 작업 4와 5는 작업 3의 실행이 정지된 후 실행을 시작할 수 있다. 작업 그래프는 문맥의 개념을 갖지 않는다. 예컨대, 작업 1과 4가 동일한 문맥을 사용하여 처리되어야 하는 경우에는, 그것은 작업 그래프에 기술될 수 없다. 따라서, 확장된 작업 그래프의 다음의 예약그래프가 본 실시예의 실시간 처리시스템에 있어서 사용된다.

우선, 작업 그래프가 작업간의 관계가 아니라, 실행기간 사이의 관계로 된다고 생각한다. 문맥을 실행기간의 각각에 관련시킴으로써, 문맥에 대응하는 스레드가 실행기간에서 실행한다. 동일한 문맥이 복수의 실행기간에 관련되는 경우에는, 그 대응하는 스레드가 실행기간의 각각에서 실행된다. 도 44에 나타낸 예에서는, 스레드 1의 문맥은 실행기간 1과 2에 관련되고, 스레드 1은 각 실행기간 1과 2에서 실행된다. 실행시간 환경에 의해 보증되는 하드 실시간의 제약조건을 나타내는 속성이 실행기간 사이의 각 화살표에 첨가된다. 이렇게 만들어진 예약그래프를 사용함으로써, 실시간 응용의 모델에 어떠한 변형도 행하지 않고도 동작모델 및 실시간 응용의 시간 요구조건과 같은 제약조건을 설명할 수 있다. 도 45는 도 44에 나타낸 그래프에 기초해서 만들어진 예약그래프의 예를 나타낸다. 도 45의 문맥 1, 2, 3은 도 44의 스레드 1, 2, 3에 각각 대응한다.

소프트 실시간 클래스에서의 스케줄링

소프트 실시간 클래스에서 스레드의 스케줄링은 스레드의 실행패턴이 예측되도록 하기 위해 고정된 우선순위 스케줄링방법(fixed priority scheduling method)을 이용하여 수행된다. 이 스케줄링방법을 위해 2개의 다른 스케줄링 알고리즘, 즉 고정된 우선순위 FIFO 스케줄링과 고정된 우선순위 순환순서(round robin) 스케줄링이 준비된다. 상위 우선순위의 스레드를 우선순위에 의해 실행하기 위해, 하위 우선순위의 스레드가 실행되는 동안에라도, 하위 우선순위의 스레드는 선취되고, 즉시 상위 우선순위의 스레드가 실행되기 시작한다. 임계부분(critical section)에서 발생하는 우선순위의 반전문제를 회피하기 위해, 우선순위 계승 프로토콜(priority inheritance protocol)이나 우선순위 상한 프로토콜(priority ceiling protocol)과 같은 동기화 메커니즘을 수행하는 것이 바람직하다.

베스트 에포트 클래스에서의 스케줄링

베스트 에포트 클래스에서의 스레드의 스케줄링은 동적인 우선순위 스케줄링(dynamic priority scheduling)과 같은 것을 이용하여 수행된다.

계층형 스케줄러

VPU 실행시간 환경(401)에서의 스케줄링 기능은 도 46에 나타낸 바와 같이 계층형(hierarchical) 스케줄러로서 실행될 수 있다. 즉, 스레드 레벨 스케줄링은 스레드 인터클래스(thread inter-class: 클래스간 스레드) 스케줄링과 스레드 인트라클래스(thread intra-class: 클래스 내 스레드) 스케줄링의 2개의 계층을 갖는다. 따라서, VPU 실행시간 환경(401)의 스케줄러는 스레드 인트라클래스 스케줄링부(601)와 스레드 인터클래스 스케줄링부(602)를 갖는다. 스레드 인터클래스 스케줄링부(602)는 스레드 클래스에 걸쳐 있는 스레드를 스케줄링한다. 스레드 인트라클래스 스케줄링부(601)는 각 스레드 클래스에 속하는 스레드를 스케줄링한다. 스레드 인트라클래스 스케줄링부(601)는 하드 실시간(하드 RT) 클래스 스케줄링부(611), 소프트 실시간(소프트 RT) 클래스 스케줄링부(612) 및 베스트 에포트 클래스 스케줄링부(613)를 포함하고 있다.

스레드 인터클래스 스케줄링과 스레드 인트라클래스 스케줄링은 계층형 구조를 갖는다. 우선, 스레드 인터클래스 스케줄링은 어느 스레드 클래스가 실행되고, 그리고 나서 스레드 클래스의 어느 스레드가 실행될지를 결정하도록 동작한다. 스레드 인터클래스 스케줄링은 선취의 고정된 우선순위 스케줄링을 이용한다. 하드 실시간 클래스가 최상위 우선순위을 갖고, 소프트 실시간 클래스와 베스트 에포트 클래스가 순서대로 뒤따른다. 보다 상위의 우선순위 클래스의 스레드가 실행할 준비가 되어 있을 때는, 최하위 우선순위의 스레드가 선취된다. 스레드 클래스간의 동기는 VPU 실행시간 환경(401)에 의해 제공되는 동기식 프리미티브(synchronous primitive)에 의해 달성된다. 특히, 블록이 하드 실시간 스레드에서 발생하는 것을 방지하기 위해 프리미티브만이 하드 실시간 스레드에서 사용될 수 있다. 베스트 에포트 스레드가 소프트 실시간 스레드를 차단하는 경우에는, 우선순위가 스레드 클래스간에서 반전되는 것을 방지하기 위해 소프트 실시간 스레드로서 처리된다. 더욱이, 예컨대 우선순위 계승 프로토콜을 사용함으로써, 또 다른 소프트 실시간 스레드가 베스트 에포트 스레드를 차단하는 것을 방지할 수 있다.

스레드 파라미터

본 실시예에 따른 실시간 처리시스템에서는, 스레드는 갖가지 파라미터를 이용하여 스케줄링된다. 각 클래스에서 스레드에 대하여 공통인 파라미터는 다음과 같다.

스레드 클래스(하드 실시간, 소프트 실시간, 베스트 에포트);

이용하기 위한 리소스(MPU나 VPU의 수, 대역폭, 물리적인 메모리 크기 및 I/O 장치);

우선순위; 및

선취 또는 비선취.

다음은 하드 실시간 클래스에서의 스레드에 대한 파라미터이다. 즉,

실행기간;

기한(dead line);

주기나 최소 상호도착 시간; 및

VPU 문맥 전환방법.

도 47은 하드 실시간 클래스에 대한 기본적인 파라미터의 예를 나타낸다. 도 47의 최상부에 나타낸 실행기간을 지정하기 위한 예 1에서는, 1개의 MPU와 2개의 VPU가 지정된 실행기간에 동시에 예약되고, 각 VPU의 문맥은 완전히 저장된다. 이 경우, 스레드는 3개의 프로세서에서 동시에 실행되고, 실행기간 후에 VPU 스레드의 문맥과 MPU의 스레드 문맥이 완전히 저장된다. 도 47의 상부 오른쪽에서, 예 2는 VPU의 수와 그 실행기간에 의해 표현되는 동작이 기한 전에 수행되는 것을 보증하기 위해 기한을 지정하는 방법을 나타내고 있다. 기한은 예약요청이 행해졌을 때, 그 요청 시간에서 시작하는 상대시간(relative time)에 의해 지정된다. 도 47의 최하부에서, 예 3은 주기적인 실행을 지정하는 방법을 나타내고 있다. 이

예에서는, 2개의 VPU(12)를 지정하는 실행기간이 주기적으로 반복되고, 각 주기에 대한 실행기간 후에 VPU 스레드의 문맥이 폐기되고, 그 결과로서 모든 동작이 새로운 문맥에 의해 수행된다. 더욱이, 기한은 그 주기의 시작부분에서 시작하는 상대시간에 의하여 지정된다.

예컨대, 하드 실시간 클래스에서 사용되는 다른 파라미터로서 다음과 같은 제약조건이 있다. 즉,

타이밍 제약조건(절대적인 타이밍 제약조건과 상대적인 타이밍 제약조건);

선행 제약조건(precedence constraint); 및

상호배타적인 제약조건.

타이밍 제약조건은 실행시간을 지연하는 유니트를 제공한다. 절대적인 타이밍 제약조건은, 도 48에 나타낸 바와 같이, 주기의 시작시간과 같은 정적인 타이밍(static timing)을 참조하여 지연시간을 지정하기 위한 조건이다. 상대적인 타이밍 제약조건은, 도 49에 나타낸 바와 같이, 일정(一定)한 시작시간 및 종료시간과 같은 동적인 타이밍이나 이벤트를 참조하여 허용가능한 지연시간(permissible delay time)을 지정하기 위한 조건이다. 선행 제약조건은 상대적인 타이밍 제약조건을 이용하여 일정한 실행기간의 종료시간을 참조해 지연시간을 0이나 그 이상으로 지정함으로써 달성될 수 있기 때문에, 상대적인 타이밍 제약조건을 위한 특별한 것으로 생각될 수 있다.

상호배타적인 제약조건은 도 50에 나타낸 바와 같이 실행기간이 서로 중복하지 않게 하는 것을 보증하기 위한 조건이다. 상호배타적인 제약조건은 잠금수단(lock)에 의해 야기되는 실행기간의 예측 불가능성을 줄이는 것을 가능하게 한다. 즉, 어떠한 리소스에 공통되는 모든 스레드는 리소스와 관련한 잠금수단을 제거하기 위해 동시에 실행되는 것이 방지된다.

스레드를 위한 동기화 메커니즘

본 실시예에 따른 실시간 처리시스템에서는, 스레드를 위한 동기화 메커니즘으로서 다음과 같은 동기식 프리미티브가 사용된다. 즉,

세마포어(semahpore);

메시지 대기열(queue);

메시지 버퍼;

이벤트 플래그;

장벽(barrier); 및

뮤텍스(Mutex).

다른 동기식 프리미티브가 사용될 수도 있다. 본 실시예의 실시간 처리시스템은 상기 동기식 메커니즘을 달성하기 위해 다음과 같은 3가지의 방법을 제공한다.

동기화 메커니즘은 TEST & SET와 같은 명령을 이용해서 VPU의 메모리(주기억장치; 14)나 로컬기억장치(32) 상에서 구현된다.

동기화 메커니즘은 메일 박스와 신호 레지스터와 같은 하드웨어 메커니즘에 의해 구현된다.

그리고 동기화 메커니즘은 VPU 실행시간 환경에 의해 서비스로서 제공되는 메커니즘을 이용하여 구현된다.

동기화 메커니즘은 장점과 단점을 갖기 때문에, 도 51에 나타낸 바와 같이 스레드의 속성에 따라서 선별적으로 이용하는 것이 바람직하다. 즉, MPU와 VPU에 의해 공유되고 접근되는 메모리(주기억장치; MS(14))를 이용하여 실현되는 동기화 메커니즘은, 모든 클래스의 스레드에 사용될 수 있다. 반면에, VPU(12)의 로컬기억장치(LS)에서 실현되는 동기화 메커니즘은 단단히 결합된 스레드 그룹에 속하는 스레드에만 사용될 수 있다. 이것은, 단단히 결합된 스레드 그룹에 속하는 스레

드만이 동기화를 위해 그들의 파트너 스레드가 동시에 실행되는 것을 보증하기 때문이다. 예컨대, 단단히 결합된 스레드 그룹에 속하는 스레드가 파트너 스레드를 실행하는 VPU의 로컬기억장치 상에서 구현되는 동기화 메커니즘에 이용되는 경우에는, 동기화 메커니즘이 이용될 때 파트너 스레드의 실행이 보증된다. 따라서, 파트너 스레드를 실행하는 VPU의 로컬기억장치는 동기화 메커니즘을 위한 정보를 항상 저장한다.

메모리(주기억장치: MS)와 로컬기억장치(LS) 이외의 유니트를 이용하는 동기화 메커니즘은 하드웨어 메커니즘이나 VPU 실행시간 환경(401)의 서비스에 의해 구현될 수 있다. 단단히 결합된 스레드에 속하는 스레드나 하드 실시간 클래스의 스레드는 고속의 동기화 메커니즘을 필요로 하기 때문에, 하드웨어 메커니즘에 의해 구현되는 동기화 메커니즘은 스레드에서 사용하는 것이 바람직하다. 반면에, 실행시간 환경에 의해 제공되는 동기화 메커니즘은 느슨히 결합된 스레드 그룹이나 소프트 실시간 클래스 및 베스트 에포트 클래스에 속하는 스레드에서 사용하는 것이 바람직하다.

동기화 메커니즘의 자동선택

본 실시예에 따른 실시간 처리시스템에서는, 상기의 동기화 메커니즘은 스레드의 속성이나 상태에 따라서 자동적으로 선택되거나 전환될 수 있다. 이러한 동작은 도 52에 나타낸 바와 같은 수순에 의해 수행된다. 동기화를 위한 스레드가 단단히 결합된 스레드 그룹에 속하는 동안(단계 S201에서 예(YES)), 메모리(14), 각 VPU(12)의 로컬기억장치(32) 또는 하드웨어 메커니즘에 의해 구현되는 고속의 동기화 메커니즘이 사용된다(단계 S202, S203, S204, S205). 스레드가 단단히 결합된 관계를 취소하기 위해 상태를 변환할 때(단계 S201에서 아니오(NO)), 고속의 동기화 메커니즘은 메모리(14)나 VPU 실행시간 환경(401)의 서비스에서 동기화 메커니즘으로서 구현되는 동기화 메커니즘으로 전환된다(단계 S206, S207, S208).

상기의 전환은 라이브러리 형태로, 또는 각 VPU(12)의 VPU 실행시간 환경(502)의 서비스로서 VPU(12)에서 실행하는 프로그램에 제공될 수 있다. 복수의 동기화 메커니즘이 다음과 같이 전환될 수 있다. 동기화 메커니즘은 미리 보호되어 선택적으로 이용될 수 있거나, 또는 전환이 수행될 때 새로운 동기화 메커니즘이 보호될 수 있다.

VPU(12)의 로컬기억장치를 이용하는 동기화 메커니즘을 위해서는, 스레드를 단단히 결합된 스레드 그룹에 속하는 스레드와 마찬가지로 VPU에 의해 동시에 실행할 필요가 있다. 이 제약조건은 다음과 같이 완화될 수 있다. 스레드가 실행되고 있지 않은 동안에는, 스레드가 최후의 것을 실행할 때 로컬기억장치의 내용이 메모리(14)에 저장되고, 저장된 내용이 로컬기억장치를 지시하는 세그먼트 테이블이나, 페이지 테이블의 엔트리에 의해 지시되도록 사상이 제어되게 된다. 이러한 방법에 따르면, 파트너 스레드가 실행되고 있지 않은 동안에는, 스레드는 파트너 스레드와 관련된 로컬기억장치가 있는 것처럼 실행을 계속할 수 있다. VPU(12)를 할당함으로써 스레드가 실행을 시작할 때는, 대응하는 페이지 테이블이나 세그먼트 테이블의 사상을 변화시키기 위해 메모리(14)에 저장된 내용이 VPU(12)의 로컬기억장치에 대해 복구된다. VPU(12)의 로컬기억장치의 백업 복사본(copy)을 이용함으로써, VPU(12)의 로컬기억장치를 이용하는 동기화 메커니즘은 단단히 결합된 그룹에 속하지 않는 스레드에 대해서도 이용될 수 있다.

예약그래프

도 53은 도 9에 나타낸 데이터 흐름에 대응하는 예약그래프를 나타내고 있다. 도 53에서, 6개의 박스는 실행기간을 나타낸다. 각 박스의 상부 왼쪽의 수는 예약되는 실행기간의 ID를 나타낸다. 각 박스의 심볼은 실행기간과 관련된 스레드 문맥의 식별자(identifier)를 나타낸다. 각 박스의 오른쪽 하부의 수는 실행기간의 길이(비용)를 나타낸다. 박스를 연결하는 화살표는 모두 선행 제약조건을 나타낸다. 즉, 하나의 박스에서 다른 박스로 이어지는 화살표는, 후자의 박스의 실행기간의 동작이 전자의 박스의 동작이 완료된 후에 시작됨을 나타낸다. 따라서, 일련의 실행주기가 제시될 수 있다. 각 화살표의 수는 화살표에 의해 연결된 실행시간 사이의 데이터 전송을 위해 사용되는 버퍼의 ID를 나타내고, 각 수의 값은 버퍼의 크기를 나타낸다. 다음에는 도 53에 나타낸 예약그래프에 따른 동작을 수행하기 위한 수순(1～7)을 설명한다.

1. DEMUX 프로그램(111)을 실행하고 그 식별자 DEMUX를 호출하는 스레드 문맥을 생성한다.

2. A-DEC 프로그램(112)을 실행하고 그 식별자 A-DEC를 호출하는 스레드 문맥을 생성한다.

3. V-DEC 프로그램(113)을 실행하고 그 식별자 V-DEC를 호출하는 스레드 문맥을 생성한다.

4. TEXT 프로그램(114)을 실행하고 그 식별자 TEXT를 호출하는 스레드 문맥을 생성한다.

5. PROG 프로그램(115)을 실행하고 그 식별자 PROG를 호출하는 스레드 문맥을 생성한다.

6. BLEND 프로그램(116)을 실행하고 그 식별자 BLEND를 호출하는 스레드 문맥을 생성한다.

7. 도 54에 나타낸 바와 같은 데이터 구조를 갖는 예약요청을 생성하고, 예약을 행하기 위해 이것을 VPU 실행시간환경(401)으로 보낸다.

상기의 수순(1～6)의 각각에 따르면, 프로그램이 스레드로서 실행되도록 지정되는 경우에는, VPU 실행시간 환경(401)은 스레드 문맥을 생성하는데 필요한 리소스를 프로그램에 할당한다. 스레드 문맥의 핸들(handle)은 되돌려지는 바, 따라서 식별자라고 부른다.

도 54는 BUFFER로서 쓰여진 버퍼 데이터 및 TASK로서 쓰여진 실행기간 데이터를 포함하는 예약요청을 나타낸다. 버퍼 데이터는 실행기간 사이의 데이터 전송을 위해 메모리(14)에 버퍼를 선언하도록 사용된다. 버퍼 데이터에서, "Id"는 버퍼의 수를, "Size"는 버퍼의 크기를 나타내고, "SrcTask"는 데이터를 기록하는 실행기간 수를, "DstTask"는 데이터를 독출하는 실행기간 수를 나타낸다. 실행기간 데이터에서, "Id"는 실행기간 수를 나타내고, "Class"는 스레드 클래스를 나타내며(VPU는 VPU 스레드를, HRT는 하드 실시간 클래스를 나타낸다. 이들에 추가해서, MPU 스레드를 나타내는 MPU, 소프트 실시간 클래스를 나타내는 SRT, 베스트 에포트 클래스를 나타내는 BST 등이 있다), "ThreadContext"는 실행기간에 대응하는 스레드 문맥을 나타내고, "Cost"는 실행기간의 길이 또는 비용을 나타내며, "Constraint"는 실행기간에 기초한 갖가지 제약조건을 나타내고, "InputBuffer"는 실행기간에 독출된 버퍼의 식별자 리스트를 나타내며, "OutputBuffer"는 실행기간에 기록된 버퍼의 식별자 리스트를 나타낸다. 또한, "Constraint"는 선행 제약조건을 보여주는 "Precedence", 절대적인 타이밍 제약조건을 나타내는 "Absolute Timing", 상대적인 타이밍 제약조건을 나타내는 "Relative Timing" 및 상호배타적인 제약조건을 보여주는 "Exclusive"를 포함하고 있다. "Constraint"는 제약조건을 위한 파트너 스레드의 실행기간 수의 리스트를 갖고 있다.

도 54에 나타낸 예약요청에 의해 예약된 버퍼영역은 주메모리(14)로 할당되고, VPU 실행시간 환경(401)에 의해 그 다음부터 해제된다. 버퍼영역의 할당은 버퍼영역에 데이터를 기록하는 스레드가 실행을 시작할 때 수행된다. 버퍼영역의 해제는 버퍼영역으로부터 데이터를 독출하는 스레드가 나갈 때 수행된다. 스레드는, 스레드가 실행을 시작할 때 미리 결정되어 있는 주소, 변수 또는 레지스터를 이용하여 할당된 버퍼의 주소를 통보받을 수 있다. 본 실시예의 실시간 처리시스템에서는, 도 7에 나타낸 프로그램 모듈(100)이 제공되면, 도 8에 나타낸 구조적 기술(117)은 그 구조적 기술(117)에 기초해서 프로그램 모듈(100)을 독출하고, 상기 수순에 의해 스레드 문맥이 생성되며, 도 54에 나타낸 예약요청이 생성되어 발행되고, 이에 따라서 프로그램 모듈(100)을 실행하는 기능을 제공한다. 이 기능은 도 7에 나타낸 프로그램 모듈(100)에 의해 설명되는 전용 하드웨어의 동작이 복수의 프로세서에 의해 소프트웨어를 처리함으로써 수행되도록 한다. 도 7에 나타낸 구조를 갖는 프로그램 모듈은 구현되어야 할 각 하드웨어에 대해 생성되고, 본 실시예의 실시간 처리시스템에 맞는 기능을 갖는 장치에 의해 실행되며, 결과적으로 장치는 원하는 하드웨어로 동작될 수 있게 된다. 다른 예로서, 도 54에 나타낸 예약요청을 생성하는 동작을 응용 프로그램으로 기술하는 것이 있는데, 이 경우 응용 프로그램은 그것만으로 예약요청을 생성할 수 있으며, 그것을 VPU 실행시간 환경(401)으로 전송할 수 있다.

도 54에 나타낸 예약요청을 제공함으로써, VPU 실행시간 환경(401)은 어느 VPU(12)가 주기의 어느 타이밍에 각 작업을 수행할지를 결정한다. 이것이 스케줄링이다. 실제적으로, 복수의 예약요청은 동시에 제공될 수 있다. 따라서, 동작 타이밍은 그것들이 서로 모순되는 것을 방지하도록 결정된다(소정 제약이 만족되지 않는 것을 방지한다). 도 55에 나타낸 바와 같이 2개의 VPU(12)가 있을 때는 도 54에 나타낸 예약요청만이 행해진다고 가정하면, VPU0가 동시에 수행될 수 없는 DEMUX, V-DEC, PROG, BLEND 동작을 연속하여 수행하도록 스케줄링이 수행되고, DEMUX 동작 후에 VPU1은 동시에 수행될 수 있는 A-DEC과 TEST 동작을 수행한다.

<u>소프트웨어 파이프라인</u>

DEMUX, V-DEC, PROG, BLEND동작을 1주기 내에서 연속해서 수행하기에 충분한 시간이 없는 경우에는, 복수의 주기에 걸쳐 소프트웨어 파이프라인 처리가 수행된다. 예컨대, 도 56에 나타낸 바와 같이, VPU0은 첫번째 주기에 DEMUX와 V-DEC동작을 수행하고, VPU1은 두번째 주기에 A-DEC, TEXT, PROG, BLEND동작을 수행한다. 두번째 주기에서, VPU0은 다음 프레임에서의 DEMUX와 V-DEC 동작을 A-DEC, TEXT, PROG, BLEND동작과 동시에 수행한다. 즉, 도 57에 나타낸 바와 같이, VPU0이 DEMUX와 V-DEC동작을 수행하고 있는 동안에는, 이전 주기의 DEMUX와 V-DEC동작으로부터의 출력을 받자마자 VPU1이 A-DEC, TEXT, PROG, BLEND동작을 수행하는 것과 같이 파이프라인 처리가 수행된다. 파이프라인 동작을 채용함으로써, 실시간 동작을 보다 짧은 시간동안 각 주기에서 완료할 수 있게 된다.

도 58은 소프트웨어 파이프라인 동작을 달성하도록 스케줄링하기 위한 수순의 플로우차트이다.

VPU 실행시간 환경(401)은 연속해서 실행될 필요가 있는 모든 스레드(DEMUX, V-DEC, PROG, BLEND)가 1주기 내에서 실행될 수 있는지를 결정한다(단계 S401). 1주기의 길이는 VPU 실행시간 환경(401)에 대해 프로그램 모듈(100)의 실행 조건으로서 미리 설정된다. 그 길이는 구조적 기술(117)에 명백히 설명될 수 있다. 단계 S401에서, 스레드(DEMUX, V-DEC, PROG, BLEND)의 총 실행기간은 이들 스레드의 비용에 기초해서 예측된다. 예측된 총 실행기간은 1주기의 길이와 비교된다.

VPU 실행시간 환경(401)이 스레드(DEMUX, V-DEC, PROG, BLEND)가 1주기 내에서 실행될 수 없다고 결정한 경우(단계 S401에서 아니오(NO))에는, 프로그램 모듈(100)을 실행하기 위해 모든 스레드(DEMUX, V-DEC, A-DEC, TEXT, PROG, BLEND)를 스레드(DEMUX, V-DEC, A-DEC, TEXT, PROG, BLEND)의 실행순서에 기초해서 연속해서 실행될 수 있는 2개의 그룹(이하, 제1 및 제2스레드 그룹)으로 분할한다(단계 S402). 제1스레드 그룹은 제2스레드 그룹 전에 실행되는 하나 이상의 스레드 세트이고, 제2스레드 그룹은 제1스레드 그룹 다음에 실행되는 하나 이상의 스레드 세트이다. 본 실시예에서는, 스레드 사이의 선행 제약조건을 만족시키고, 각 그룹의 총 실행기간을 1주기에 대응하는 시간간격보다 길어지지 않도록 하기 위해, 스레드(DEMUX, V-DEC)는 제1스레드 그룹에 속하게 하고, 스레드(A-DEC, TEXT, PROG, BLEND)는 제2스레드 그룹에 속하게 한다.

VPU 실행시간 환경(401)은 VPU0에 의해 제1스레드 그룹을 1/60초의 시간간격으로 주기적으로 실행하도록 제1스레드 그룹(DEMUX, V-DEC)에 속하는 각 스레드의 실행기간을 주기적으로 VPU0에 할당하기 위해 스케줄링 동작을 수행한다(단계 S403). 단계 S403에서, 스레드(DEMUX, V-DEC)의 각각의 주기적인 실행은 VPU0에 예약된다. 다음으로, VPU 실행시간 환경(401)은 제2스레드 그룹(A-DEC, TEXT, PROG, BLEND)에 속하는 각 스레드를 주기적으로 VPU1에 할당하도록 스케줄링 동작을 수행하고, VPU1은 제2스레드 그룹을 제1스레드 그룹에 비해 1주기 지연시켜 1/60초의 시간간격으로 주기적으로 실행한다(단계 S404). 단계 S404에서, 스레드(A-DEC, TEXT, PROG, BLEND)의 각각의 주기적인 실행은 VPU1에 예약된다.

2개의 프로세서(VPU0, VPU1)는 제1스레드 그룹(DEMUX, V-DEC)과 제2스레드 그룹(A-DEC, TEXT, PROG, BLEND)을 파이프라인 모드로 실행한다. 그 결과, 제2스레드 그룹이 제1스레드 그룹에 비해 1주기 지연되는 동안, 제1스레드 그룹과 제2스레드 그룹이 동시에 실행되고, 따라서 1/60초의 각 주기에 대해 프레임 데이터 처리결과를 출력하게 된다.

상기의 예에서는, VPU0이 항상 제1스레드 그룹(DEMUX, V-DEC)을 실행하고, VPU1이 항상 제2스레드 그룹(A-DEC, TEXT, PROG, BLEND)을 실행하도록 되어 있다. 그러나, 도 59에 나타낸 바와 같이, 스케줄링은 제1스레드 그룹이 할당되는 프로세서와 제2스레드 그룹이 할당되는 프로세서를 주기적으로 대체하도록 수행될 수 있다. 스케줄링 동작에 있어서, 제1 및 제2스레드 그룹 각각의 실행 타이밍과 제1 및 제2스레드 그룹을 실행하기 위한 다른 프로세서는 각 주기동안 결정되어 제2스레드 그룹이 제1스레드 그룹에 비해 1주기만큼 지연되는 동안 프로세서로 동시에 제1 및 제2스레드 그룹을 실행하도록 한다.

파이프라인 동작을 이용한 전력절약제어

상술한 파이프라인 동작은 각 작업의 실행 타이밍의 제약조건이 작업의 실행순서의 제약조건을 만족하는 범위 내에서 완화되도록 해준다. 비록 각 주기가 예비시간을 갖지 않더라도, 파이프라인 동작을 이용하는 것에 의해 버스 대역폭이 큰 작업의 실행기간이 서로 겹치는 것을 방지하도록 스케줄링이 수행될 수 있다.

도 60은 2개의 채널에 대한 디지탈 TV방송 수신동작이 동시에 수행될 때 필요하게 되는 버스 대역폭을 나타낸다. 각 주기가 예비시간을 갖고 있지 않은 경우에는, VPU0에 의해 실행되어야 할 BLEND의 실행기간과 VPU1에 의해 실행되어야 할 BLEND의 실행기간은 서로로부터 시프트될 수는 없다.

도 61은 상기의 BLEND의 실행기간이 파이프라인 동작에 의해 시프트되는 예를 나타낸다. VPU1에 의해 수행되어야 할 실시간 동작(D2: DEMUX, V2: V-DEC, A2: A-DEC, T2: TEXT, P2: PROG, B2: BLEND)은 제1 스레드그룹(V2, A2, T2, D2)과 제2 스레드그룹(P2, B2)으로 분류된다. 도 61에 나타낸 바와 같이, 제2 스레드그룹(P2, B2)은 제1 스레드그룹(V2, A2, T2, D2)에 비해 1주기 지연되어 실행되는 바, 제2 스레드그룹(P2, B2)은 주기 2에서 제1 스레드그룹(V2, A2, T2, D2) 전에 실행된다. 실시간 동작(D2: DEMUX, V2: V-DEC, A2: A-DEC, T2: TEXT, P2: PROG, B2: BLEND)이 VPU1에 의해 2주기에 걸쳐 수행되기 때문에, VPU0 및 VPU1에 의해 실행되어야 할 BLEND의 실행기간은 서로 겹치는 것이 방지될 수 있다. 따라서, 도 62에 나타낸 바와 같이, 각 주기에서 요구되는 버스 대역폭의 피크값은 도 60에서의 값의 절반으로 저감될 수 있다.

계층적인 구조를 갖는 예약그래프

도 53에 나타낸 예약그래프가 계층적인 구조를 갖지 않더라도, 계층적인 구조를 갖는 예약그래프가 도 63에 나타낸 바와 같이 이용될 수 있다. 도 63에서, 실행기간 A는 실행기간 B에 선행하고, 실행기간 B는 실행기간 C에 선행한다. 실행기간 B에서, 실행기간 D는 실행기간 E와 F에 선행한다. 계층을 분해하면, 실행기간 A는 실행기간 D에 선행하고, 실행기간 E와 F는 실행기간 C에 선행한다.

구조적 기술에 기초한 스케줄링 알고리즘

다음에는, 프로그램 모듈에 합체된 구조적 기술에 기초해서 각 스레드의 실행기간을 예약하기 위한 수순에 대해 설명한다.

도 8은 도 7에 도시된 프로그램 모듈(100)에 합체된 구조적 기술(117)의 예를 나타낸다. 구조적 기술(117)에 의해, VPU 실행시간 환경(401)은 다음의 단계를 수행한다.

1. 구조적 기술(117)의 모듈 필드에 기록되어 있는 프로그램이 그 프로그램을 실행하는 스레드를 생성하기 위해 로드된다. 본 실시예에서는, 구조적 기술(117)의 엔트리의 각각에 대해 하나의 스레드가 생성된다. 구조적 기술(117)이 같은 모듈이름을 갖는 엔트리를 포함하고 있는 경우에는, 같은 모듈을 실행하는 복수의 스레드가 그들 각각의 엔트리에 대응하도록 생성된다. 도 8의 예에서는, 한 프로세스에 속하도록 모든 스레드가 생성되지만, 그 스레드가 다른 프로세스에 속할 수 있거나, 또는 스레드 그룹이 다른 프로세스에 속할 수 있다.

2. 구조적 기술(117)의 정보에 기초해서, 도 54에 나타낸 데이터 구조를 갖는 예약요청이 생성된다.

3. 스레드를 스케줄링하고 스레드의 실행을 시작하기 위해, 예약요청이 VPU 실행시간 환경으로 보내진다.

예약요청을 생성하는 상기 단계 2는 다음과 같이 수행된다.

먼저, BUFFER 레코드가 구조적 기술(117)의 출력 필드에 대응하도록 일대일(one-to-one)방식으로 생성되고, 예약요청에 부가된다. 예컨대, 도 8의 예에서는, DEMUX 모듈의 두번째 출력 데이터가 1MB 버퍼를 통해 V-DEC로 공급되고, 따라서 도 54에 나타낸 Id가 2인 BUFFER 레코드가 생성된다. 이러한 BUFFER 레코드에서는, 버퍼 크기는 Size 필드에서 1MB로서 기술되고, Id가 1이고 버퍼에 데이터를 기록하는 DEMUX 모듈에 대응하는 TASK 레코드에 대한 참조(reference)는 SrcTask 필드에 기술되며, Id가 3이고 버퍼로부터 데이터를 독출하는 V-DEC 모듈에 대응하는 TASK 레코드에 대한 참조는 DstTask 필드에 기술된다.

다음으로, TASK 레코드가 구조적 기술(117)의 모듈 필드에 대응하도록 일대일방식으로 생성되어 예약요청에 부가된다. 예컨대, 도 8의 예에서는, 도 54에 나타낸 바와 같이 Id가 3인 TASK 레코드가 V-DEC 모듈에 대응하는 것으로서 생성된다. 이러한 TASK 레코드는 다음의 정보를 갖는다.

Class 필드: TASK 레코드에서 지정된 스레드를 실행하는데 어느 속성이 사용되는지를 지시하기 위한 플래그.

이 필드에서, "VPU"는 VPU에서 실행되는 스레드를 나타내고, "HRT"는 하드 실시간 클래스에서의 스레드를 나타낸다. 이들 정보 항목은, 도 8에 나타낸 구조적 기술(117)의 스레드 파라미터에서 기술된 정보에 기초해서 설정된다.

ThreadContext 필드: 실행이 TASK 레코드에 예약되어야 할 스레드의 스레드 문맥을 지정하기 위한 플래그. 보다 상세하게는, 구조적 기술(117)의 모듈 필드에서 지정된 프로그램 모듈이 로드되고, 프로그램 모듈을 실행하는 스레드가 VPU 실행시간 환경(401)에 의해 생성되며, 스레드의 스레드 문맥의 식별자(포인터 등)가 "ThreadContext" 필드에 기록된다.

Constraint 필드: TASK 레코드의 제약조건을 기록하기 위한 플래그. 제약조건이 선행 제약조건인 경우, TASK 레코드에 의해 선행되는 다른 TASK 레코드의 필요한 Id 수는 "Precede" 필드 후에 지정된다. 예컨대, Id가 3인 TASK 레코드는 Id가 5인 PROG 모듈에 대응하는 TASK 레코드에 선행한다.

InputBuffer 필드: TASK 레코드에 의해 지정된 스레드에 의해 데이터가 독출되는 버퍼의 Buffer 레코드의 필요한 Id 수를 지정하기 위한 플래그.

OutputBuffer 필드: TASK 레코드에 의해 지정된 스레드에 의해 데이터가 기록되는 버퍼의 Buffer 레코드의 필요한 Id 수를 지정하기 위한 플래그.

밴드 필드: TASK 레코드에 의해 지정된 스레드에 의해 요구되는 버스 대역폭을 지정하기 위한 플래그.

구조적 기술이 상기에 설명한 바와 같이 제공되면, 그 대응하는 예약요청이 생성된다.

예약요청이 VPU 실행시간 환경(401)의 스케줄러로 보내질 때, 스케줄러는 예약요청을 수행하는데 필요한 스케줄을 생성한다. 이 스케줄은, 도 55에 나타낸 바와 같이, 어느 VPU가 어느 시간에 어느 스레드에 할당되는지, 그리고 1주기에서 얼마나 오랫동안 VPU가 할당되는지를 나타낸다. 실질적으로, 스케줄은 도 64에 나타낸 예약리스트에 의해 나타낼 수 있다.

도 64에 나타낸 예약리스트는 각 VPU에 관련된 예약엔트리를 포함한다. 각 예약엔트리는, 각 주기에서 언제 스레드가 VPU에 의해 실행되는지를 나타내는 시작시간 필드(스레드의 실행시작 타이밍), 얼마나 오래 VPU가 스레드에 할당되는지를 나타내는 실행기간 필드(스레드의 실행기간) 및 스레드의 식별자를 나타내는 실행 스레드 필드를 포함한다. 예약엔트리는 VPU에 따라 시작시간의 순서로 정렬되고, 예약리스트로 링크(link)된다.

도 54에 나타낸 예약요청으로부터 도 64에 나타낸 예약리스트를 생성하기 위한 수순은 도 65에 나타낸 플로우차트에 의해 수행될 수 있다.

기본적으로, 예약요청의 TASK 레코드는 BUFFER를 이용한 입/출력 관계를 고려하여 연속되어야 하고, VPU의 실행시간은 데이터 흐름의 순서로 각 TASK 레코드에 할당되어야 한다. 다음으로, VPU를 단단히 결합된 스레드 그룹에 속하는 Task에 동시에 할당할 필요가 있다. 2개 이상의 VPU가 사용될 때에는, TASK 레코드의 각각의 버스 대역폭을 고려하여 TASK는 적어도 2개의 더 높은 순위의 TASK의 실행기간 및 큰 버스 대역폭이 서로 겹치는 것을 방지하기 위해 차례로 나열된다.

그 수순이 도 65에 나타내어져 있다. 예약요청을 받자 마자, VPU 실행시간 환경(401)은 예약요청의 TASK 레코드에 의해 지정된 모든 작업을 다음의 단계에 의해 스케줄링한다(즉, VPU 실행시간 환경(401)은 각 작업이 할당되는 VPU를 예약하기 위해 예약리스트, 실행시작 타이밍, 작업의 실행기간을 생성한다).

단계 S301: VPU 실행시간 환경(401)은 스케줄링되어 있지 않은 작업 중에서, 모든 선행 작업(입력작업)이 이미 스케줄링되어 있고, 단단히 결합된 속성을 갖지 않는 작업을 선택한다. 작업이 입력작업에 의해 선행되는 작업이 없으면, 이미 스케줄링된 입력작업으로서 결정된다.

입력작업이 이미 스케줄링되어 있고, 단단히 결합된 속성을 가지고 있지 않은 작업이 존재하는 경우에는, VPU 실행시간 환경(401)은 그것을 선택하고, 단계 S302로 이동한다. 그렇지 않은 경우에는, 단계 S304로 이동한다.

단계 S302: 만족스러운 제약조건 하에서 선택된 작업의 실행시작 타이밍과 실행기간을 할당할 수 있는 VPU가 존재하면, VPU 실행시간 환경(401)은 단계 S303으로 이동한다. 그렇지 않은 경우에는, VPU 실행시간 환경(401)은 스케줄링에 실패하고, 그 실패를 통보한다.

단계 S303: VPU 실행시간 환경(401)은 선택된 작업의 예약엔트리를 생성하고, 그것들을 예약리스트로 링크한다. 작업의 실행 타이밍은 상술한 바와 같이 그 버스 대역폭을 고려하여 결정된다.

단계 S304: VPU 실행시간 환경(401)은 스케줄링되어 있지 않은 작업중에서, 모든 입력작업이 이미 스케줄링되어 있고, 단단히 결합된 그룹에 속하는 작업을 선택한다. 입력작업에 의해 선행되는 작업이 없으면, 이미 스케줄링된 입력작업으로서 결정된다.

입력작업이 이미 스케줄링되어 있고, 단단히 결합된 그룹에 속하는 작업이 존재하는 경우에는, VPU 실행시간 환경(401)은 그것을 선택하고, 단계 S305로 이동한다. 그렇지 않은 경우에는, 스케줄링을 종료한다.

단계 S305: 선택된 작업에 포함된 모든 작업을 (동일한 실행시작 타이밍과 동일한 실행기간을 갖도록) 동시에 예약할 수 있는 VPU가 존재하면, VPU 실행시간 환경(401)은 단계 S306으로 이동한다. 그렇지 않은 경우에는, VPU 실행시간 환경(401)은 스케줄링에 실패하고, 실패를 통보한다.

단계 S306: 선택된 작업 세트의 모든 작업의 예약엔트리가 생성되고, 예약리스트로 링크된다.

하나의 예약요청에 대한 스케줄링 단계를 설명했다. 실제로는, 하나의 시스템에서 복수의 예약요청이 동시에 나타나는 것이 일반적이다. 이 경우, 예약요청은 상기의 단계를 통해 스케줄링될 수 있고, 더욱 바람직하게는 상기의 단계를 통해 동시에 스케줄링될 수 있다.

본 실시예는 디지털 TV방송 수신기의 동작을 설명하는 프로그램 모듈을 예로 들어 설명했다. 그러나, 다양한 형태의 하드웨어의 동작을 설명하는 프로그램 모듈이 준비되면, 하드웨어의 동작은 소프트웨어에 의해 수행될 수 있다.

도 1에 나타낸 컴퓨터 시스템에 제공되는 MPU(11)와 VPU(12)는 칩에 혼합된 병렬 프로세서로서 구현될 수 있다. 이 경우, MPU(11)에 의해 실행되는 VPU 실행시간 환경 또는 특정 VPU에 의해 실행되는 VPU 실행시간 환경 등은 VPU(12)를 위한 스케줄링 및 버스(13)의 데이터 전송속도를 제어할 수 있다.

VPU 실행환경으로서 실행하는 프로그램 또는 VPU 실행환경을 포함하는 동작 시스템의 프로그램이 컴퓨터 독출가능한 저장매체에 저장되고, 그리고 로컬 메모리를 각각 갖추고 있는 복수의 프로세서를 포함하는 컴퓨터로 도입되어 실행되는 경우, 본 발명의 상기 실시예와 동일한 장점이 얻어질 수 있다.

부가적인 장점과 변형은 당업자에게 자명하다. 따라서, 이러한 넓은 관점의 발명은 여기에 보여지고 설명된 특정 세부사항과 대표적인 실시예에 한정되지 않는다. 따라서, 첨부된 청구항에 의해 정의되는 발명의 개념이나 범위를 벗어나지 않는 한 다양한 변형이 이루어져도 좋다.

발명의 효과

이상 설명한 바와 같이 본 발명에 의하면, 어떤 실시간 동작도 손상시키지 않고 요구되는 데이터 전송대역폭을 줄일 수 있는 방법 및 정보처리 시스템을 제공할 수 있다.

(57) 청구의 범위

청구항 1.

데이터 전송속도가 변경될 수 있는 버스를 매개로 데이터를 전송하는 복수의 프로세서를 이용하여 특정의 시간간격 내에 복수의 작업을 수행하는 방법으로,

각 작업을 수행하는데 필요한 시간과 관련된 비용정보와 각 작업에 의해 요구되는 데이터 전송대역폭과 관련된 대역폭 정보를 입력하는 단계와,

상기 작업 중 적어도 2개의 작업의 실행기간을 중복하지 않게 하고서 특정의 시간간격 내에 그들 작업을 수행하기 위해, 상기 입력된 비용정보와 대역폭 정보에 기초해서, 각 작업의 실행시작 타이밍과 상기 작업을 실행하는 프로세서 중 적어도 하나를 결정하는 스케줄링 동작을 수행하는 단계,

상기 스케줄링 동작의 결과와 상기 대역폭 정보에 기초해서, 특정의 시간간격 내에 상기 프로세서 중 적어도 하나에 의해 수행되어야 할 데이터 전송의 데이터 전송대역폭의 피크값을 계산하는 단계 및,

상기 버스의 데이터 전송속도를 상기 피크값보다 높은 값으로 설정하는 단계를 구비하여 이루어지되,

상기 2개의 작업이 나머지 작업의 데이터 전송대역폭보다 작은 데이터 전송대역폭을 필요로 하는 것을 특징으로 하는 작업수행방법.

청구항 2.

제1항에 있어서, 상기 피크값대 상기 최대 데이터 전송대역폭의 비율에 기초해서, 상기 버스의 데이터 전송속도를 상기 버스의 최대 데이터 전송대역폭보다 작은 값으로 설정하는 단계를 더 구비한 것을 특징으로 하는 작업수행방법.

청구항 3.

제2항에 있어서, 상기 설정단계가 상기 버스의 동작 주파수를 제어함으로써 상기 버스의 데이터 전송속도를 제어하는 것을 포함하도록 된 것을 특징으로 하는 작업수행방법.

청구항 4.

특정의 시간간격 내에 복수의 작업을 수행하는 정보처리 시스템으로,

데이터 전송속도가 변경될 수 있는 버스와,

상기 버스를 매개로 데이터를 전송하는 복수의 프로세서,

상기 작업 중 적어도 2개의 작업의 실행기간을 중복하지 않게 하고서 특정의 시간간격 내에 그들 작업을 수행하기 위해, 각 작업을 수행하는데 필요한 시간과 관련된 비용정보와 각 작업에 의해 요구되는 데이터 전송대역폭과 관련된 대역폭 정보에 기초해서, 각 작업의 실행시작 타이밍과 상기 작업을 실행하는 프로세서 중 적어도 하나를 결정하는 스케줄링 동작을 수행하는 수단,

상기 스케줄링 동작의 결과와 상기 대역폭 정보에 기초해서, 특정의 시간간격 내에 상기 프로세서 중 적어도 하나에 의해 수행되어야 할 데이터 전송의 데이터 전송대역폭의 피크값을 계산하는 수단 및,

상기 버스의 데이터 전송속도를 상기 피크값보다 높은 값으로 설정하는 수단을 구비하여 구성되되,

상기 2개의 작업이 나머지 작업의 데이터 전송대역폭보다 작은 데이터 전송대역폭을 필요로 하는 것을 특징으로 하는 정보처리 시스템.

청구항 5.

제4항에 있어서, 상기 피크값대 최대 데이터 전송대역폭의 비율에 기초해서, 상기 버스의 데이터 전송속도를 상기 버스의 최대 데이터 전송대역폭보다 작은 값으로 설정하는 수단을 더 구비한 것을 특징으로 하는 정보처리 시스템.

청구항 6.

제5항에 있어서, 상기 데이터 전송속도를 설정하는 수단이 상기 버스의 동작 주파수를 제어함으로써 상기 버스의 데이터 전송속도를 제어하는 수단을 포함하도록 되어 있는 것을 특징으로 하는 정보처리 시스템.

청구항 7.

제4항에 있어서, 상기 버스가 상호접속 네트워크를 포함하도록 되어 있는 것을 특징으로 하는 정보처리 시스템.

청구항 8.

제4항에 있어서, 상기 버스에 접속되는 메모리를 더 구비하고,

상기 프로세서가 상기 버스를 매개로 상기 프로세서와 상기 메모리 사이에서 데이터를 전송하도록 되어 있는 것을 특징으로 하는 정보처리 시스템.

청구항 9.

특정의 시간간격 내에 복수의 작업을 수행하는 정보처리 시스템으로,

데이터 전송속도가 변경될 수 있는 버스와,

상기 버스를 매개로 데이터를 전송하는 복수의 제1프로세서,

상기 작업 중 적어도 2개의 작업의 실행기간을 중복하지 않게 하고서 특정의 시간간격 내에 그들 작업을 수행하기 위해, 각 작업을 수행하는데 필요한 시간과 관련된 비용정보와 각 작업에 의해 요구되는 데이터 전송대역폭과 관련된 대역폭 정보에 기초해서, 각 작업의 실행시작 타이밍과 상기 작업을 실행하는 제1프로세서 중 적어도 하나를 결정하는 스케줄링 동작을 수행하는 제2프로세서,

상기 스케줄링 동작의 결과와 상기 대역폭 정보에 기초해서, 특정의 시간간격 내에 상기 제1프로세서 중 적어도 하나에 의해 수행되어야 할 데이터 전송의 데이터 전송대역폭의 피크값을 계산하는 수단 및,

상기 버스의 데이터 전송속도를 상기 피크값보다 높은 값으로 설정하는 수단을 구비하여 구성되되,

상기 2개의 작업이 나머지 작업의 데이터 전송대역폭보다 작은 데이터 전송대역폭을 필요로 하는 것을 특징으로 하는 정보처리 시스템.

청구항 10.

제9항에 있어서, 상기 피크값대 최대 데이터 전송대역폭의 비율에 기초해서, 상기 버스의 데이터 전송속도를 상기 버스의 최대 데이터 전송대역폭보다 작은 값으로 설정하는 데이터 전송속도 제어유니트를 더 구비한 것을 특징으로 하는 정보처리 시스템.

청구항 11.

제10항에 있어서, 상기 데이터 전송속도 제어유니트가 상기 버스의 동작 주파수를 제어함으로써 상기 버스의 데이터 전송속도를 제어하도록 되어 있는 것을 특징으로 하는 정보처리 시스템.

청구항 12.

제9항에 있어서, 상기 버스가 상호접속 네트워크를 포함하도록 되어 있는 것을 특징으로 하는 정보처리 시스템.

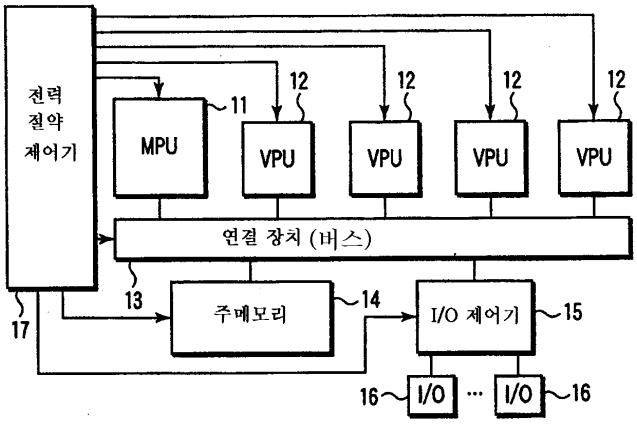
청구항 13.

제9항에 있어서, 상기 버스에 접속되는 메모리를 더 구비하고,

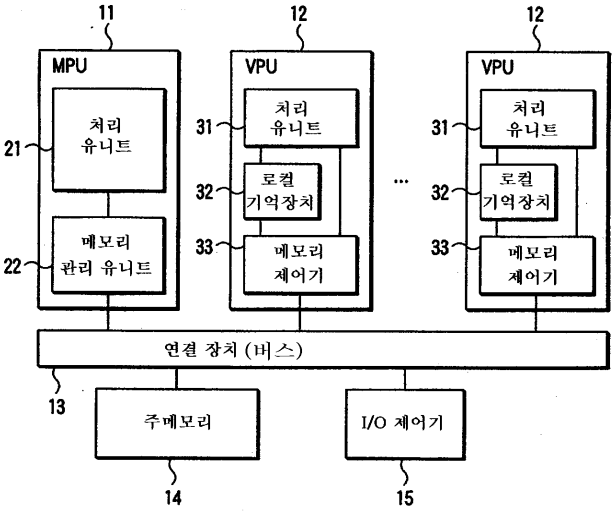
상기 제1프로세서가 상기 버스를 매개로 상기 제1프로세서와 상기 메모리 사이에서 데이터를 전송하도록 되어 있는 것을 특징으로 하는 정보처리 시스템.

도면

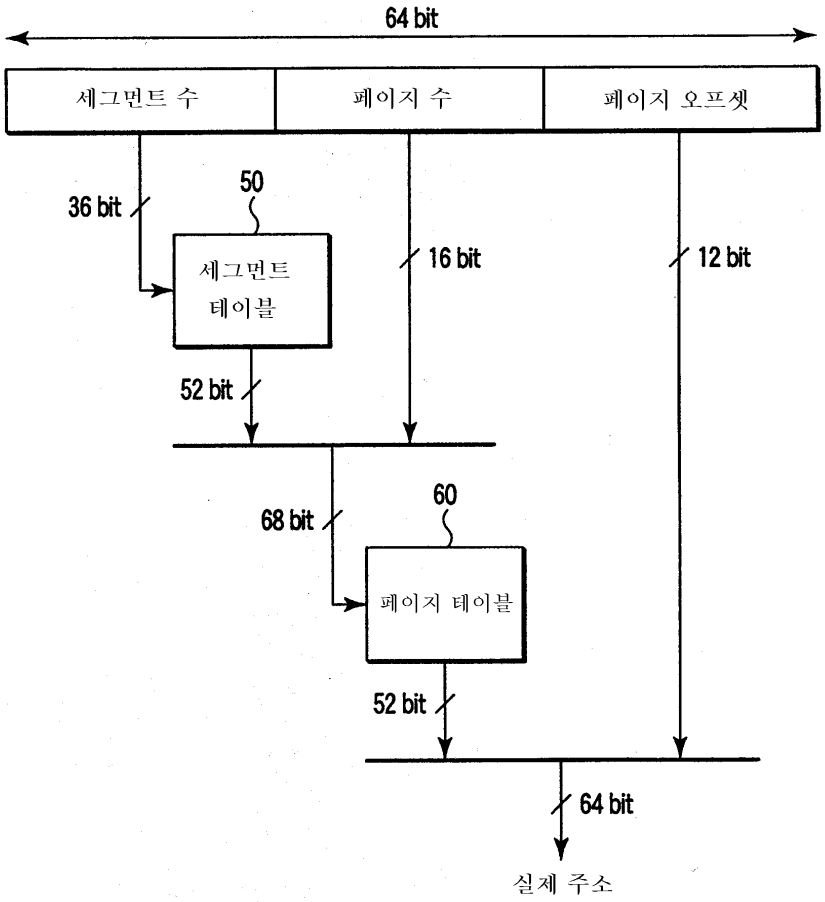
도면1


전력
절약
제어기
MPU
11
VPU
12
VPU
12
VPU
12
VPU
12
연결 장치 (버스)
13
주메모리
14
I/O 제어기
15
17
16
I/O
I/O
16

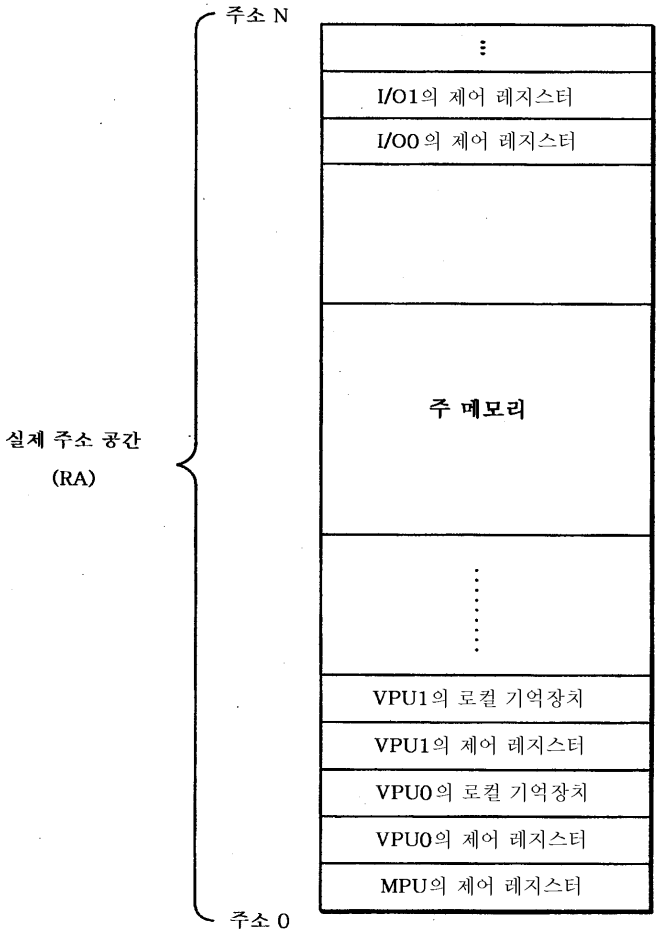
도면2


11
MPU
처리
유니트
21
메모리
관리 유니트
22
12
VPU
처리
유니트
31
로컬
기억장치
32
메모리
제어기
33
12
VPU
처리
유니트
31
로컬
기억장치
32
메모리
제어기
33
연결 장치 (버스)
13
주메모리
14
I/O 제어기
15

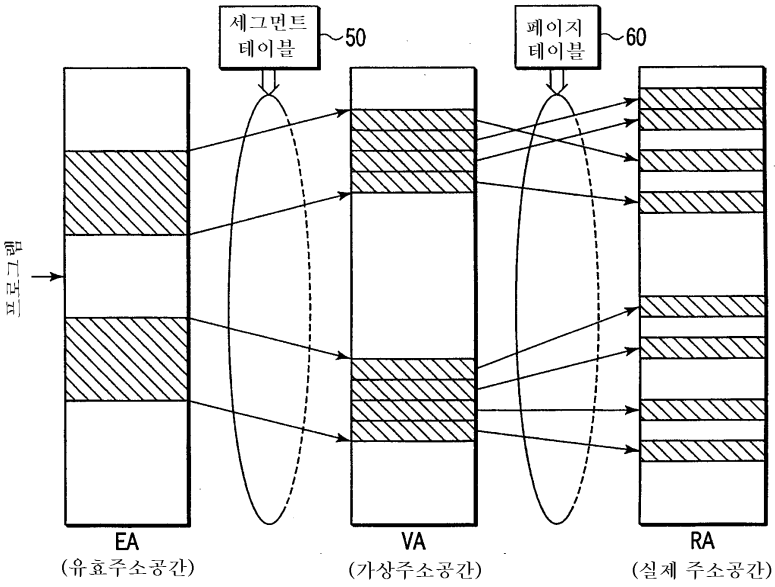
도면3


64 bit
세그먼트 수
페이지 수
페이지 오프셋
36 bit
50
세그먼트
테이블
16 bit
12 bit
52 bit
68 bit
60
페이지 테이블
52 bit
64 bit
실제 주소

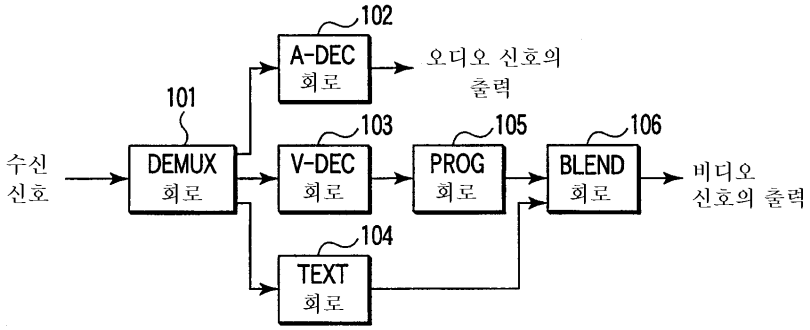
도면4


주소 N
I/O1의 제어 레지스터
I/O0의 제어 레지스터
주 메모리
실제 주소 공간
(RA)
VPU1의 로컬 기억장치
VPU1의 제어 레지스터
VPU0의 로컬 기억장치
VPU0의 제어 레지스터
MPU의 제어 레지스터
주소 0

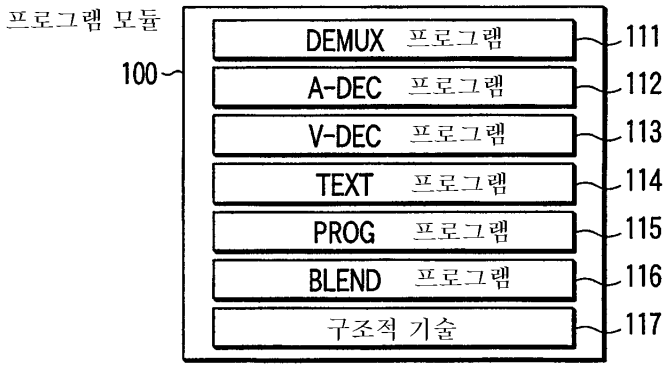
도면5


세그먼트 테이블
50
페이지 테이블
60
프로그램
EA
(유효주소공간)
VA
(가상주소공간)
RA
(실제 주소공간)

도면6


102
A-DEC
회로
오디오 신호의
출력
101
수신
신호
DEMUX
회로
103
V-DEC
회로
105
PROG
회로
106
BLEND
회로
비디오
신호의 출력
104
TEXT
회로

도면7

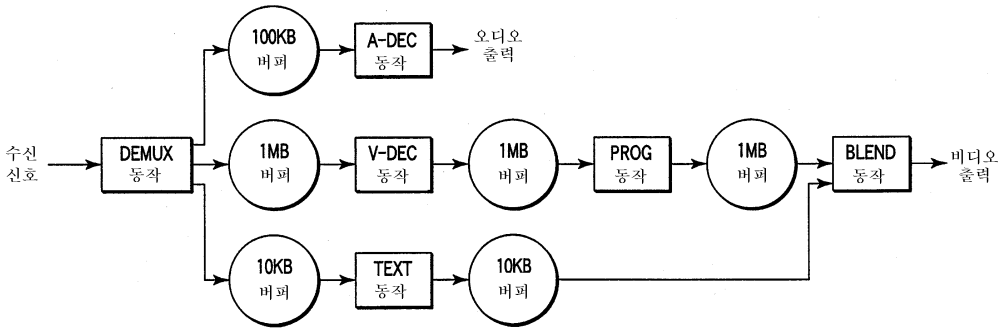

프로그램 모듈
100
DEMUX 프로그램
111
A-DEC 프로그램
112
V-DEC 프로그램
113
TEXT 프로그램
114
PROG 프로그램
115
BLEND 프로그램
116
구조적 기술
117

도면8

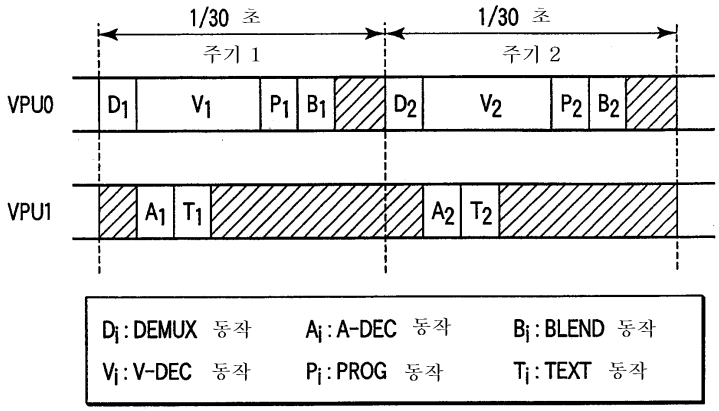
구조적 기술 117

번호	프로그램	입력	출력	비용	버퍼
(1)	DEMUX	수신신호	(2) (3) (4)	5	100KB 1MB 10KB
(2)	A-DEC	(1)	오디오 출력	10	———
(3)	V-DEC	(1)	(5)	50	1MB
(4)	TEXT	(1)	(6)	5	10KB
(5)	PROG	(3)	(6)	20	1MB
(6)	BLEND	(4) (5)	비디오 출력	10	———
스레드 파라미터 { • 단단히 결합된 스레드 그룹 : • 느슨히 결합된 스레드 그룹 : }					
그 외					

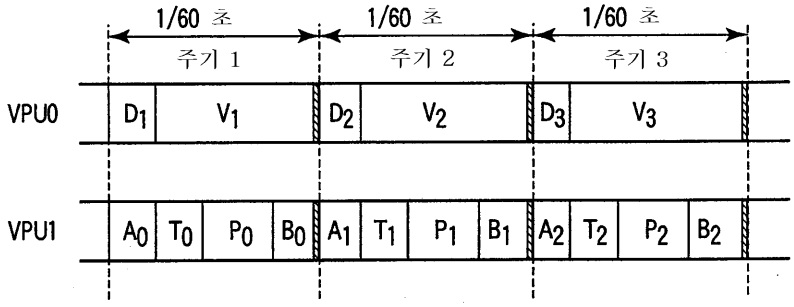
도면9



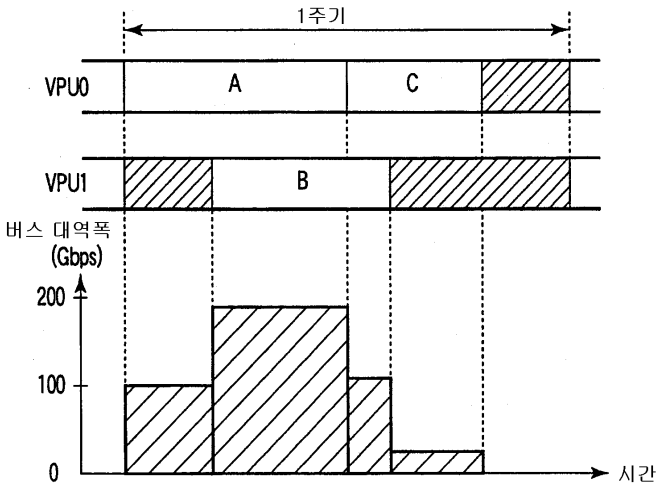
도면10



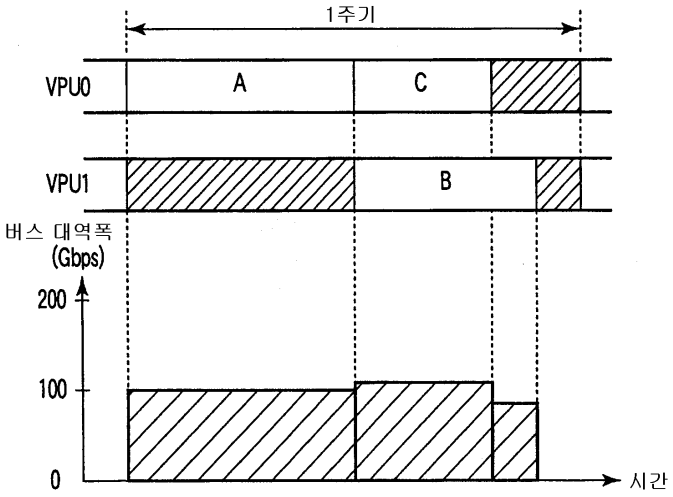
도면11



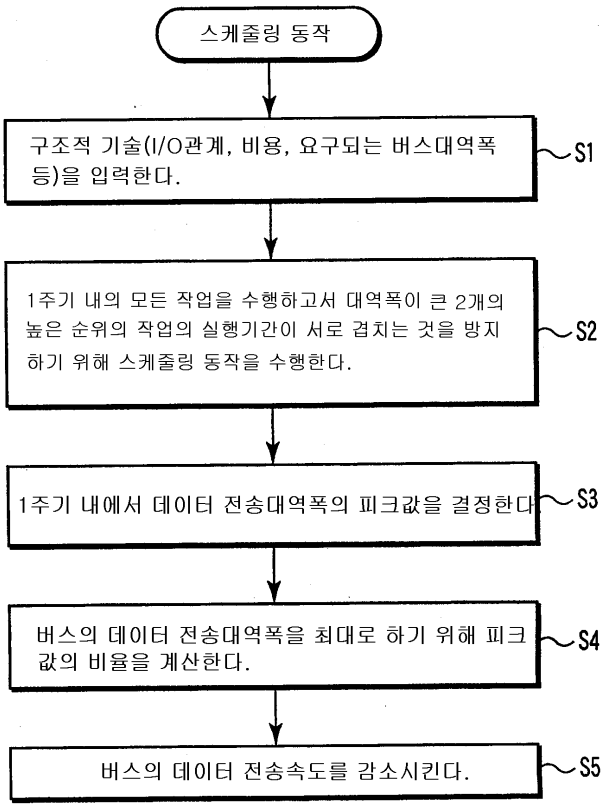
도면12



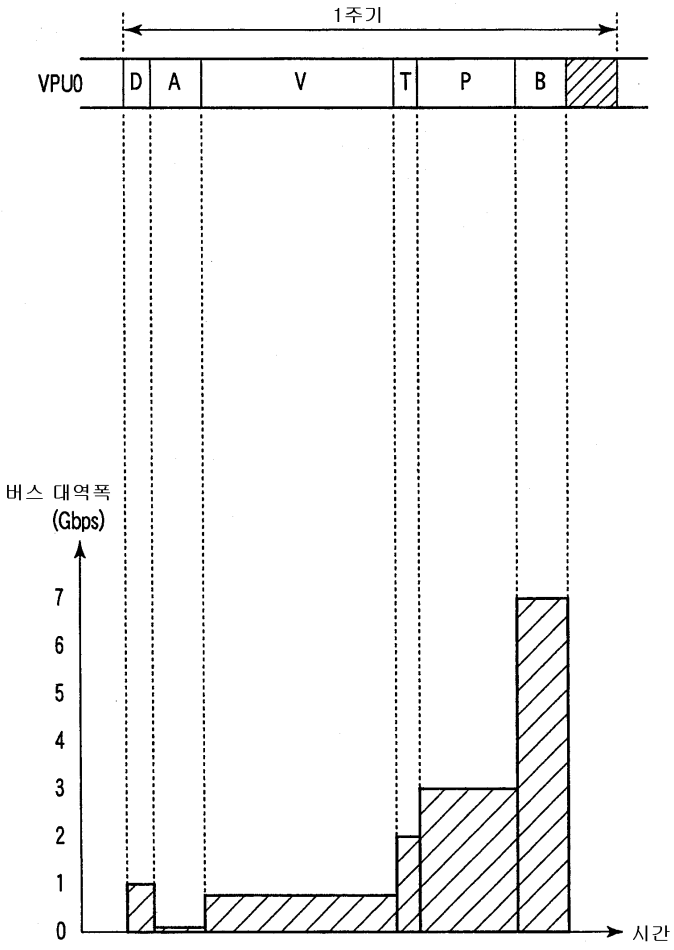
도면13


1주기
VPU0
A
C
VPU1
B
버스 대역폭
(Gbps)
200
100
0
시간

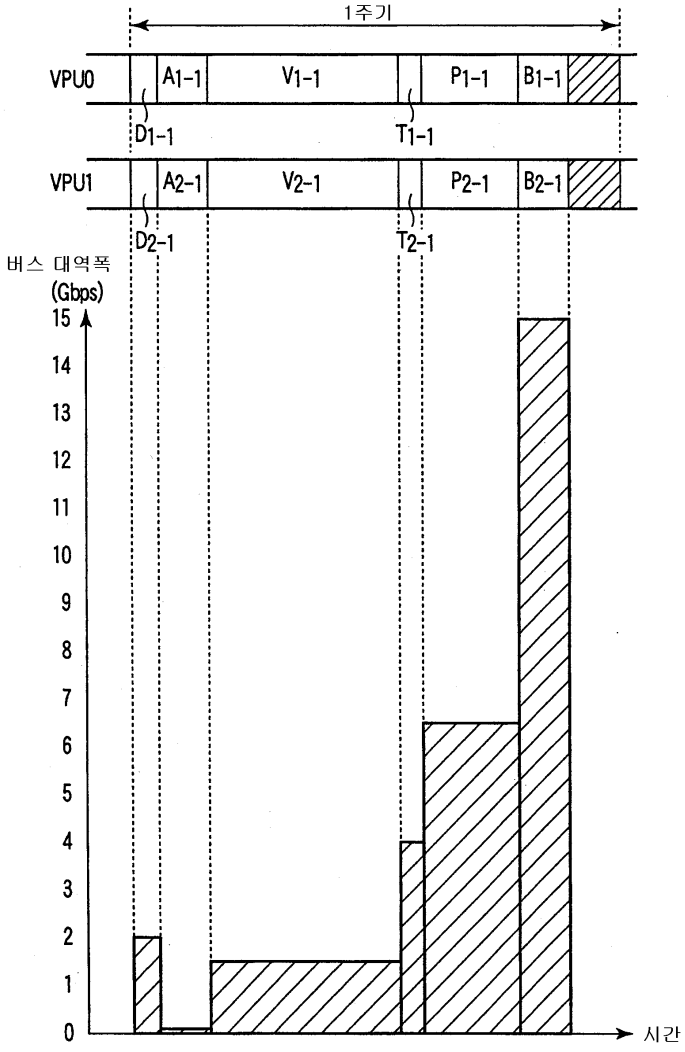
도면14


스케줄링 동작
구조적 기술(I/O관계, 비용, 요구되는 버스대역폭 등)을 입력한다.
S1
1주기 내의 모든 작업을 수행하고서 대역폭이 큰 2개의 높은 순위의 작업의 실행기간이 서로 겹치는 것을 방지하기 위해 스케줄링 동작을 수행한다.
S2
1주기 내에서 데이터 전송대역폭의 피크값을 결정한다
S3
버스의 데이터 전송대역폭을 최대로 하기 위해 피크값의 비율을 계산한다.
S4
버스의 데이터 전송속도를 감소시킨다.
S5

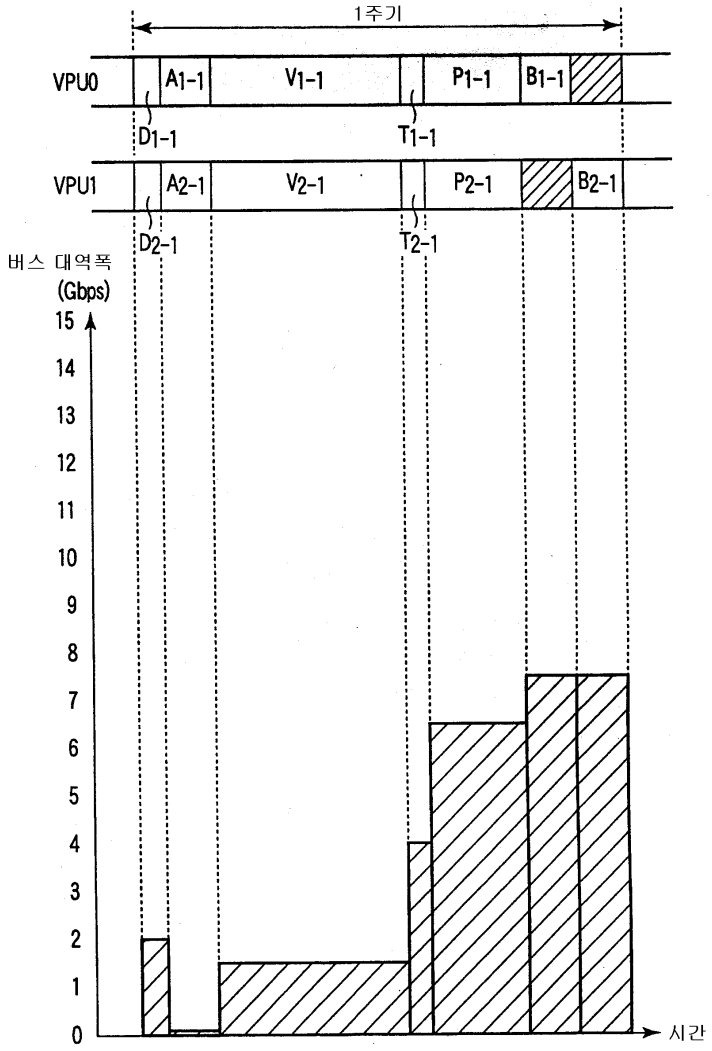
도면15



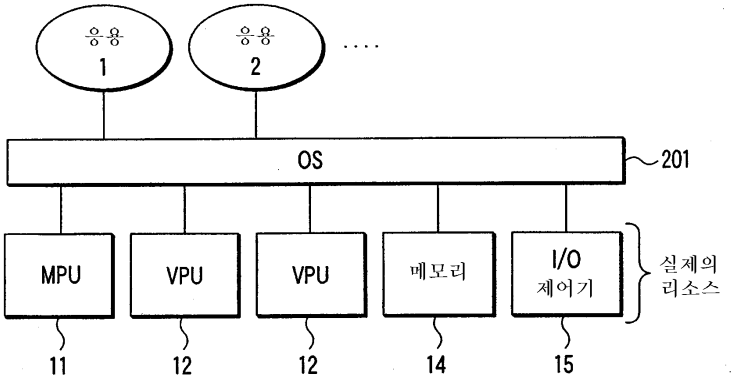
도면16


1주기
VPU0
A1-1
V1-1
P1-1
B1-1
D1-1
T1-1
VPU1
A2-1
V2-1
P2-1
B2-1
D2-1
T2-1
버스 대역폭
(Gbps)
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
시간

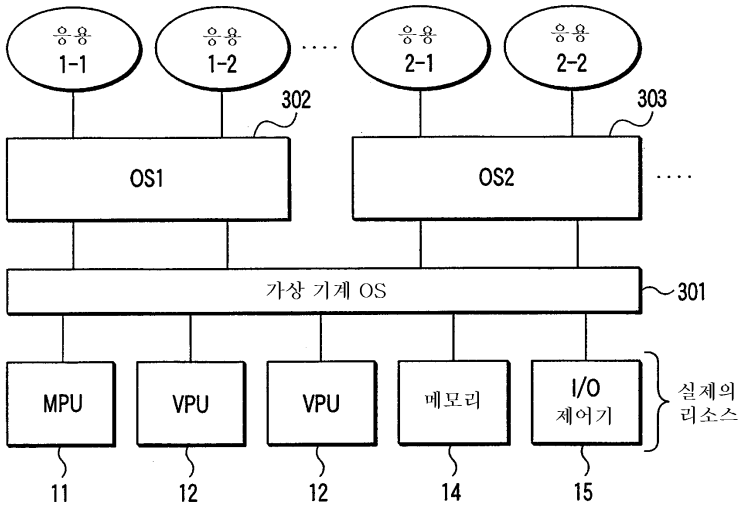
도면17


1주기
VPU0
A1-1
V1-1
P1-1
B1-1
D1-1
T1-1
VPU1
A2-1
V2-1
P2-1
B2-1
D2-1
T2-1
버스 대역폭
(Gbps)
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
시간

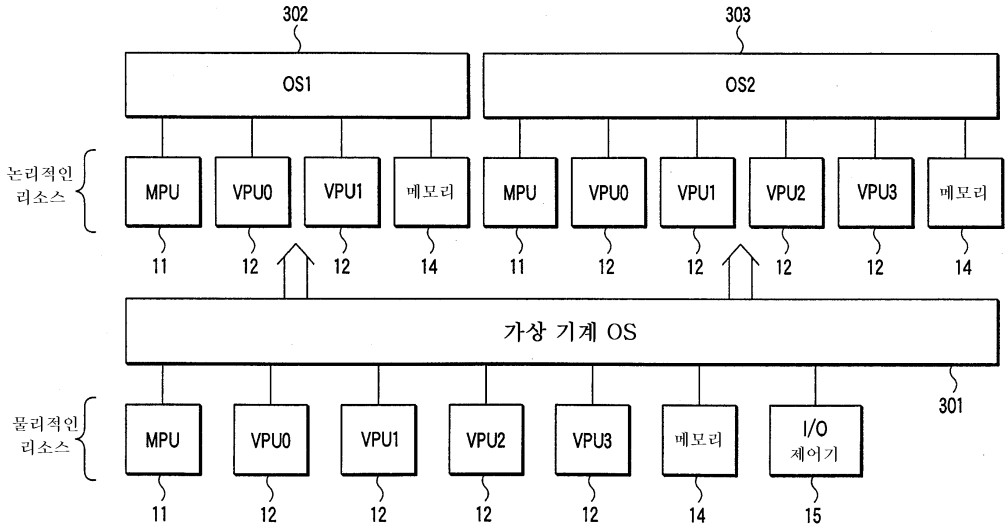
도면18


응용
1
응용
2
....
OS
201
MPU
VPU
VPU
메모리
I/O
제어기
실제의
리소스
11
12
12
14
15

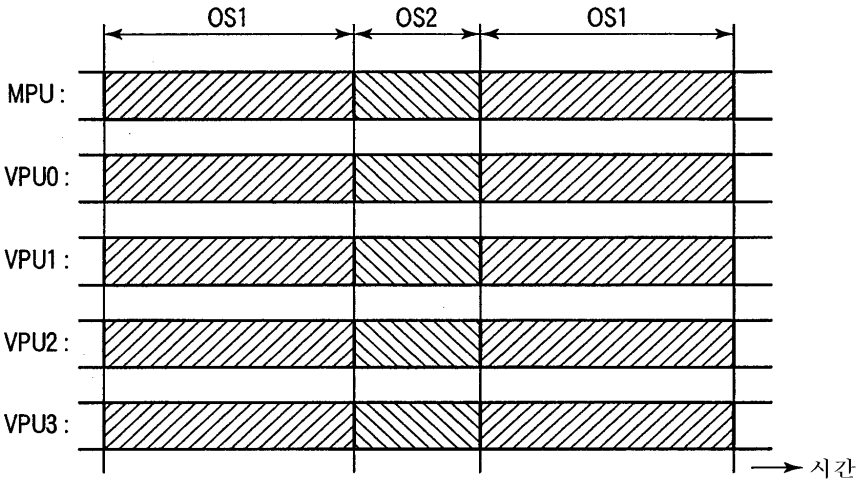
도면19



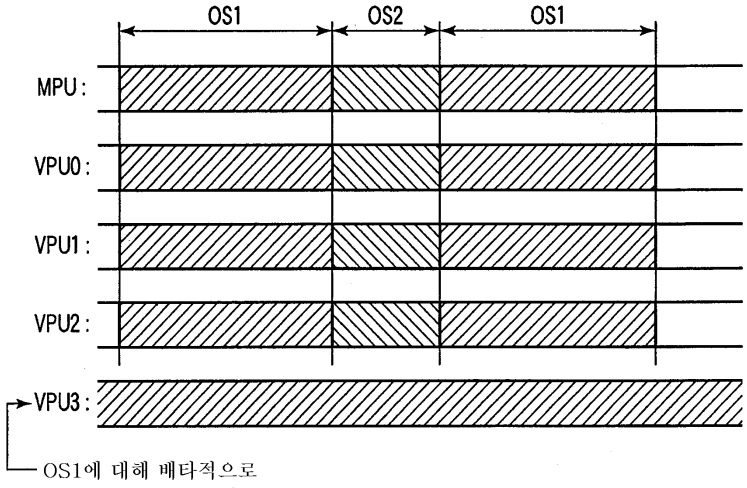
도면20



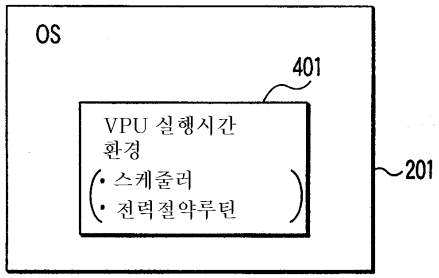
도면21



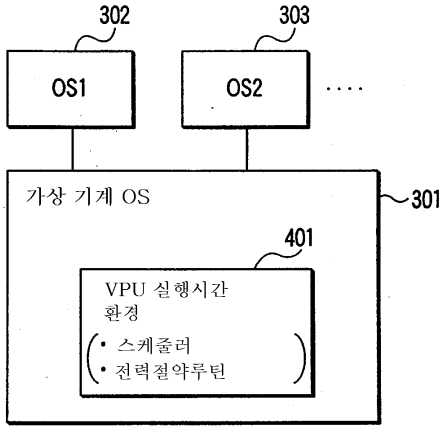
도면22



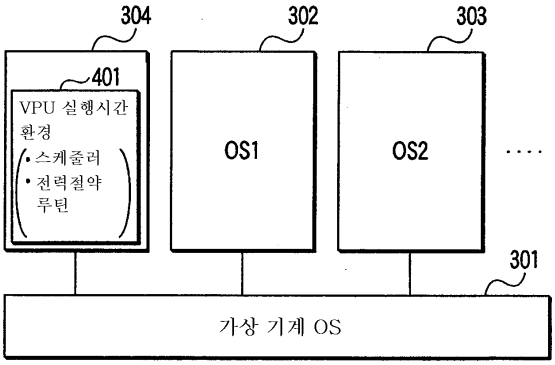
도면23



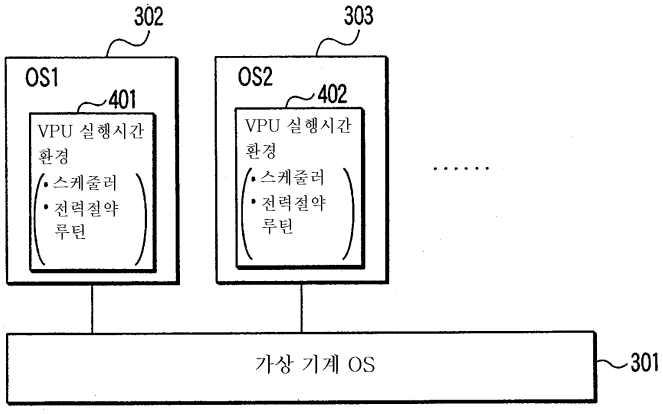
도면24



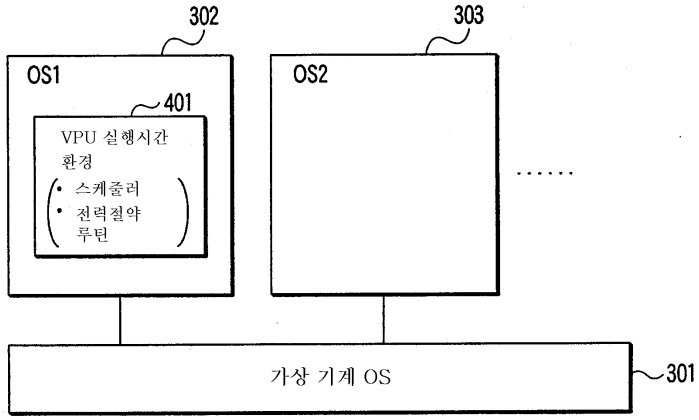
도면25



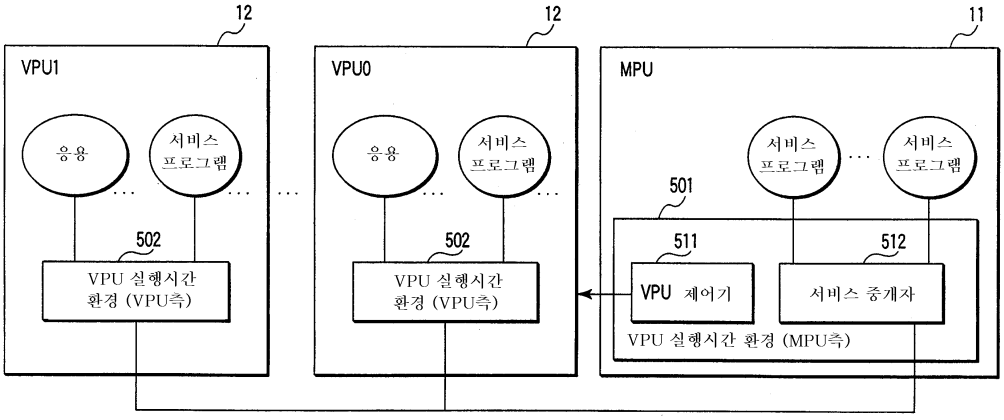
도면26



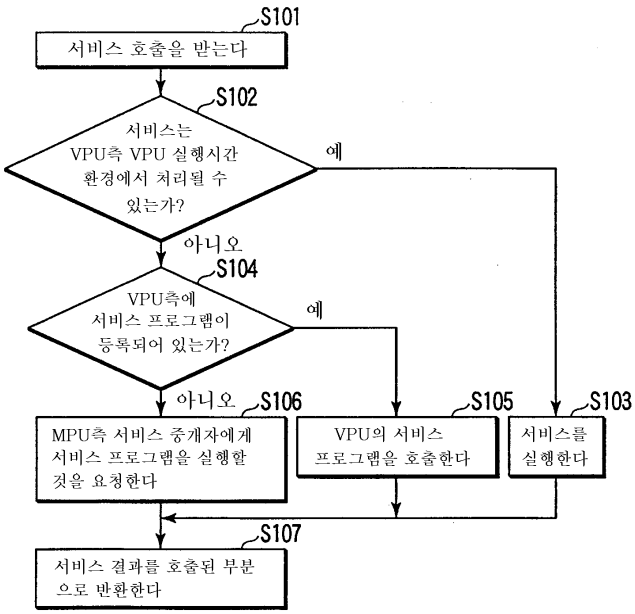
도면27



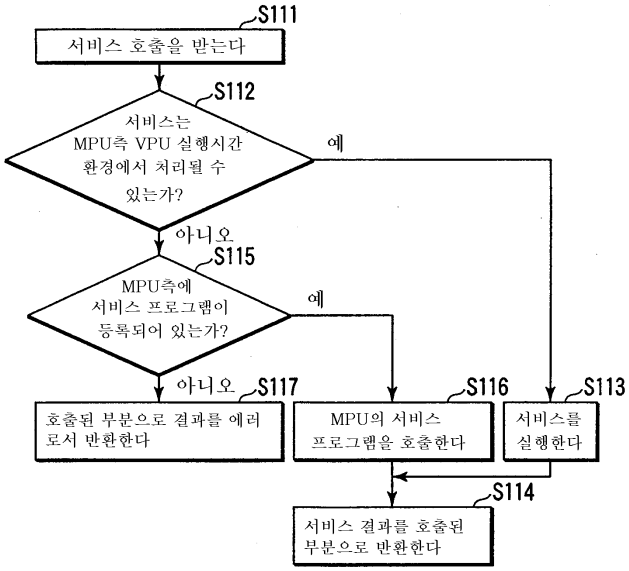
도면28


12
12
11
VPU1
VPU0
MPU
응용
서비스 프로그램
서비스 프로그램
서비스 프로그램
응용
서비스 프로그램
서비스 프로그램
서비스 프로그램
501
502
502
511
512
VPU 실행시간 환경 (VPU측)
VPU 실행시간 환경 (VPU측)
VPU 제어기
서비스 중개자
VPU 실행시간 환경 (MPU측)

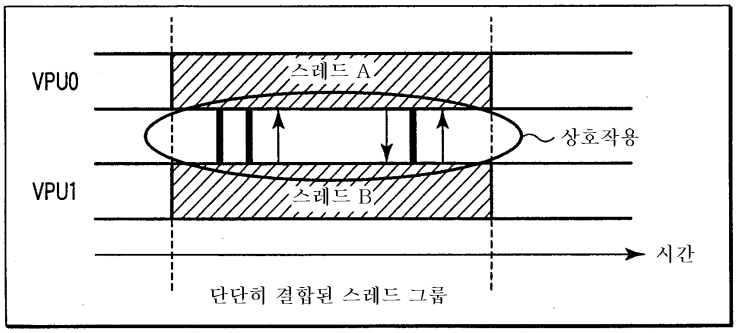
도면29


S101
서비스 호출을 받는다
S102
서비스는 VPU측 VPU 실행시간 환경에서 처리될 수 있는가?
예
아니오
S104
VPU측에 서비스 프로그램이 등록되어 있는가?
예
아니오
S106
S105
S103
MPU측 서비스 중개자에게 서비스 프로그램을 실행할 것을 요청한다
VPU의 서비스 프로그램을 호출한다
서비스를 실행한다
S107
서비스 결과를 호출된 부분으로 반환한다

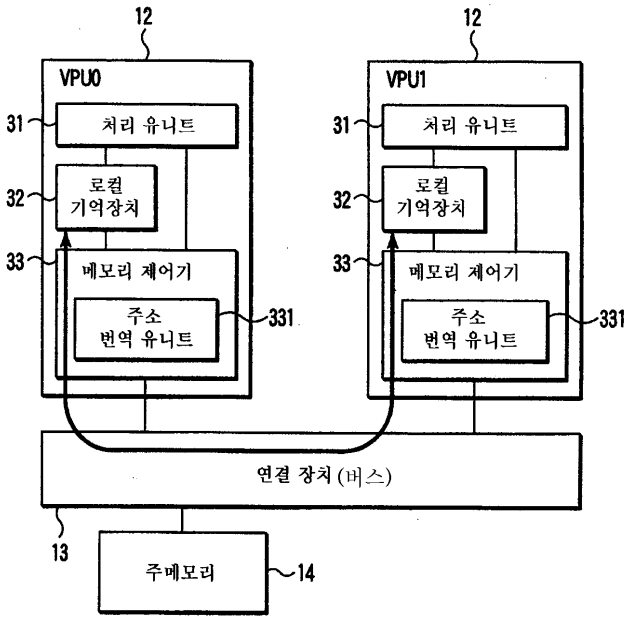
도면30


S111
서비스 호출을 받는다
S112
서비스는 MPU측 VPU 실행시간 환경에서 처리될 수 있는가?
예
아니오
S115
MPU측에 서비스 프로그램이 등록되어 있는가?
예
아니오
S117
호출된 부분으로 결과를 에러로서 반환한다
S116
MPU의 서비스 프로그램을 호출한다
S113
서비스를 실행한다
S114
서비스 결과를 호출된 부분으로 반환한다

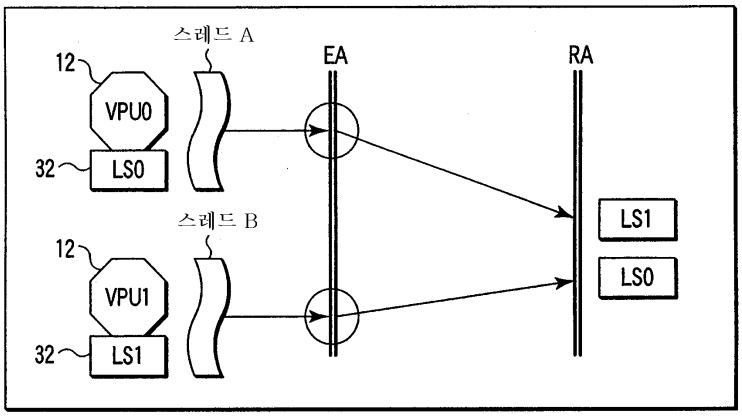
도면31


VPU0
VPU1
스레드 A
스레드 B
상호작용
시간
단단히 결합된 스레드 그룹

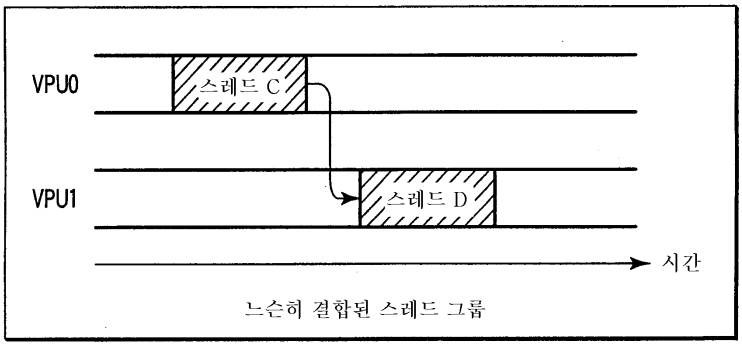
도면32



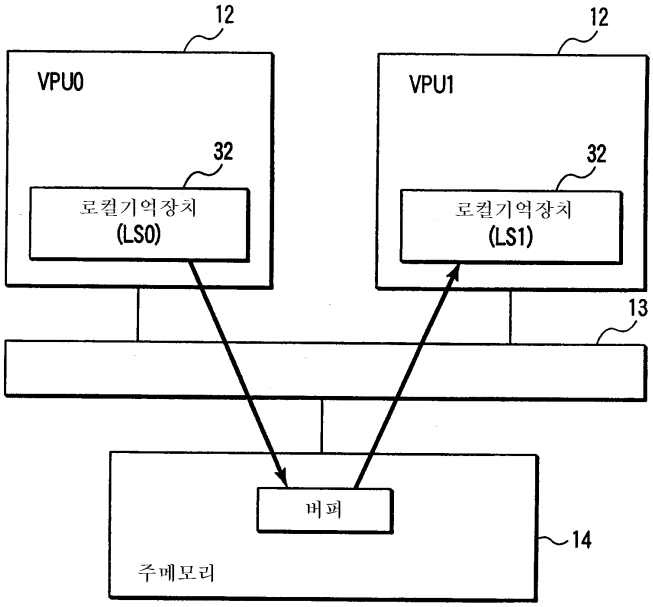
도면33



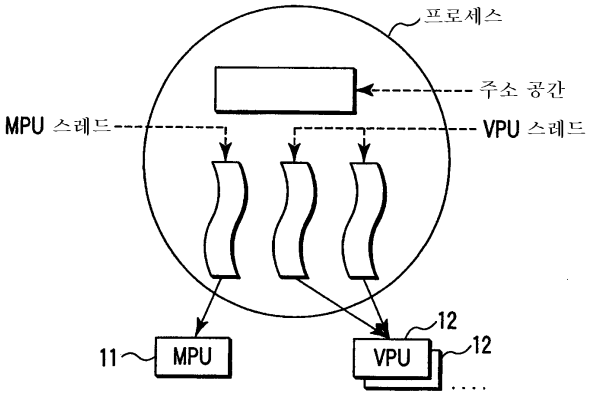
도면34



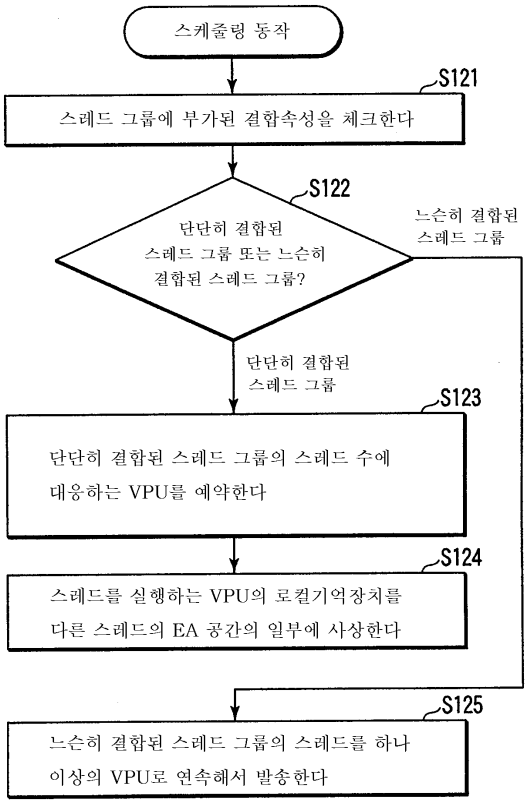
도면35


12
12
VPU0
VPU1
32
32
로컬기억장치
(LS0)
로컬기억장치
(LS1)
13
버퍼
14
주메모리

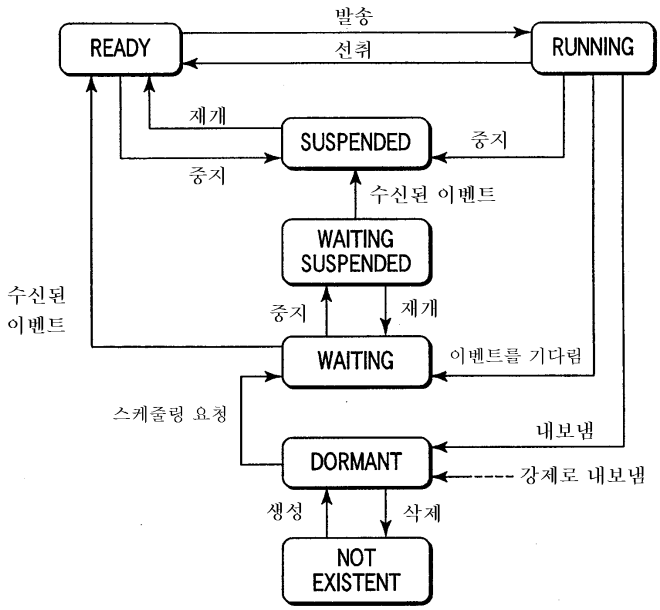
도면36


프로세스
주소 공간
MPU 스레드
VPU 스레드
12
11
MPU
VPU
12

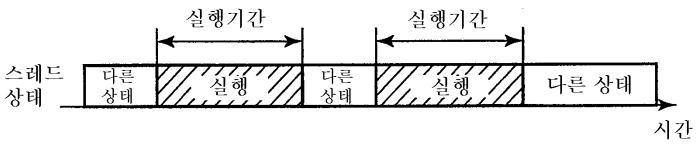
도면37


스케줄링 동작
S121
스레드 그룹에 부가된 결합속성을 체크한다
S122
단단히 결합된
스레드 그룹 또는 느슨히
결합된 스레드 그룹?
느슨히 결합된
스레드 그룹
단단히 결합된
스레드 그룹
S123
단단히 결합된 스레드 그룹의 스레드 수에
대응하는 VPU를 예약한다
S124
스레드를 실행하는 VPU의 로컬기억장치를
다른 스레드의 EA 공간의 일부에 사상한다
S125
느슨히 결합된 스레드 그룹의 스레드를 하나
이상의 VPU로 연속해서 발송한다

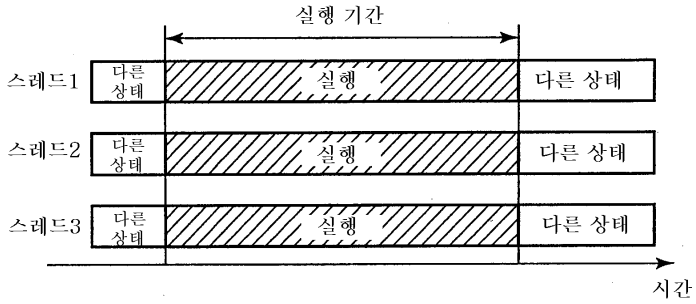
도면38


발송
READY
선취
RUNNING
재개
SUSPENDED
중지
중지
수신된 이벤트
WAITING
SUSPENDED
수신된
이벤트
중지
재개
WAITING
이벤트를 기다림
스케줄링 요청
내보냄
DORMANT
강제로 내보냄
생성
삭제
NOT
EXISTENT

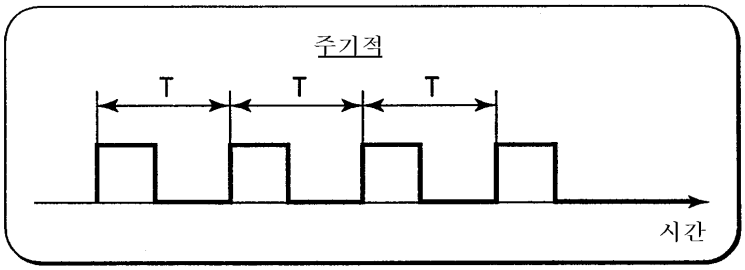
도면39



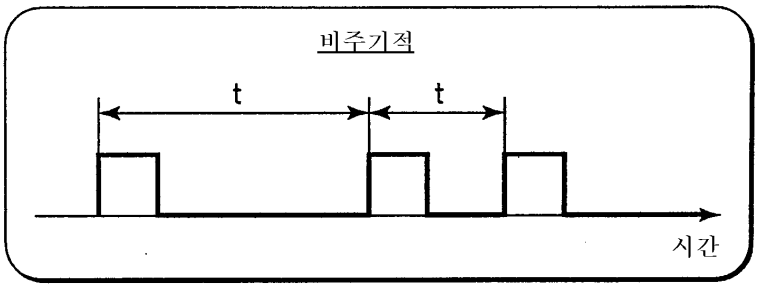
도면40



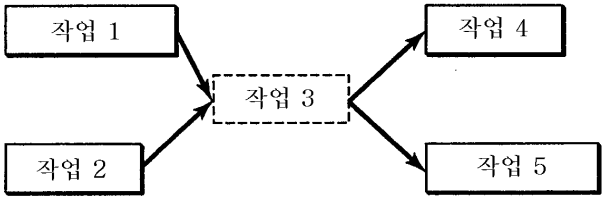
도면41



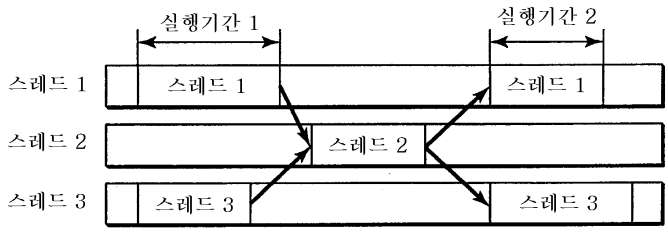
도면42



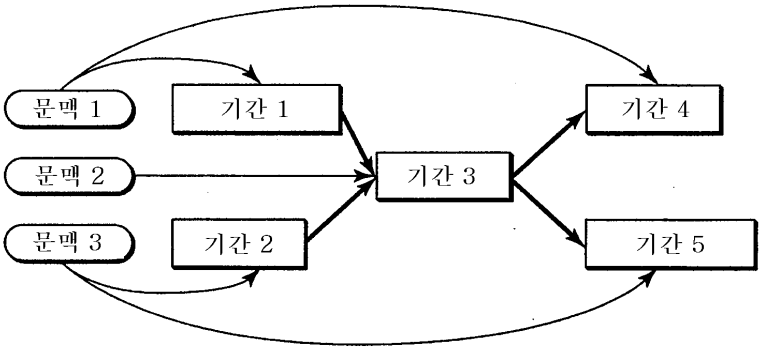
도면43



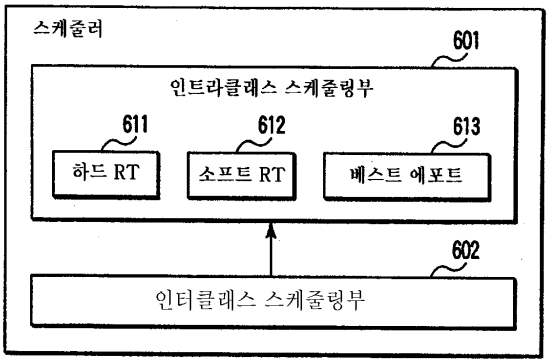
도면44



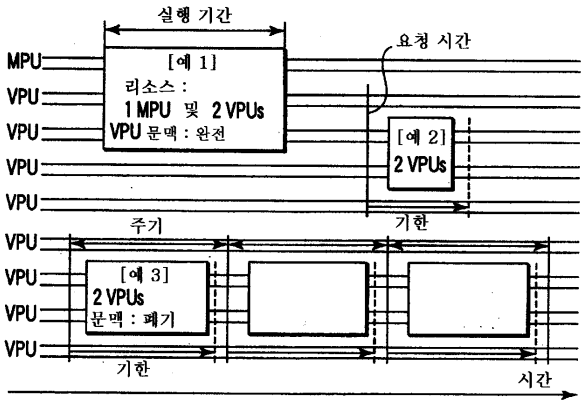
도면45



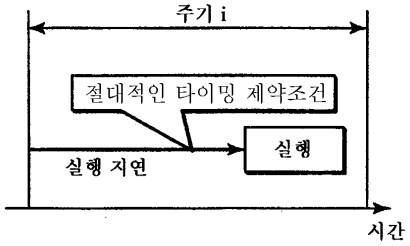
도면46



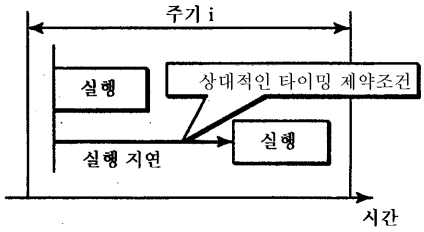
도면47



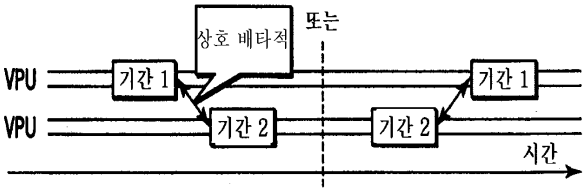
도면48



도면49



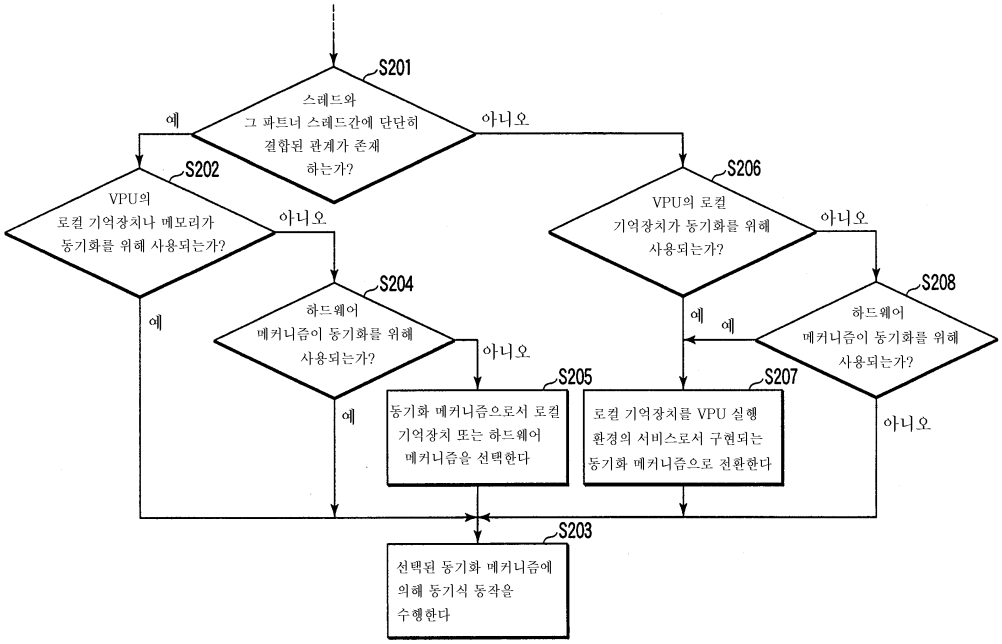
도면50



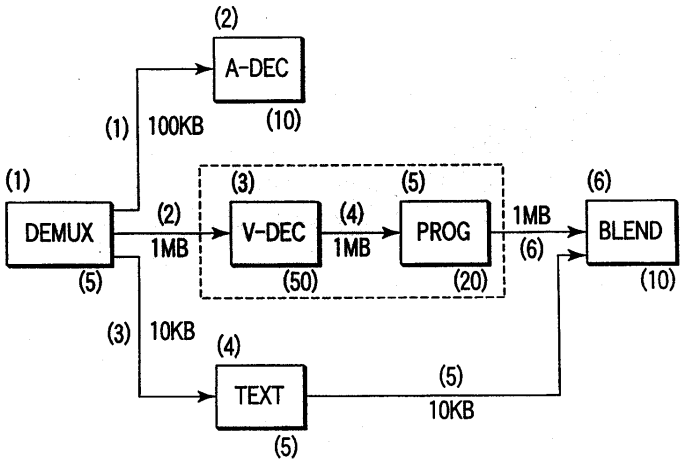
도면51

		단단히 결합된 스레드 그룹	느슨히 결합된 스레드 그룹
메모리	LS	사용 가능	사용 불가능
	MS	사용 가능	
다른것		하드웨어 프리미티브를 사용해야 함	VPU 실행시간 환경에 의해 제공되는 메커니즘을 사용해야 함

도면52


S201
스레드와 그 파트너 스레드간에 단단히 결합된 관계가 존재하는가?
예
아니오
S202
VPU의 로컬 기억장치나 메모리가 동기화를 위해 사용되는가?
아니오
예
S204
하드웨어 메커니즘이 동기화를 위해 사용되는가?
아니오
예
S205
동기화 메커니즘으로서 로컬 기억장치 또는 하드웨어 메커니즘을 선택한다
S206
VPU의 로컬 기억장치가 동기화를 위해 사용되는가?
아니오
예
S207
로컬 기억장치를 VPU 실행 환경의 서비스로서 구현되는 동기화 메커니즘으로 전환한다
S208
하드웨어 메커니즘이 동기화를 위해 사용되는가?
예
아니오
S203
선택된 동기화 메커니즘에 의해 동기식 동작을 수행한다

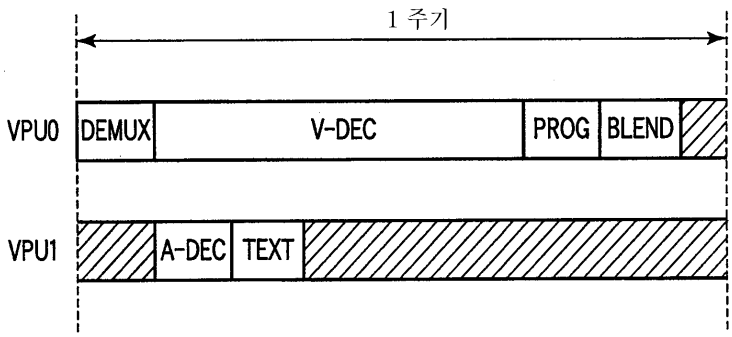
도면53


(2)
A-DEC
(10)
(1)
100KB
(1)
DEMUX
(5)
(2)
1MB
(3)
V-DEC
(50)
(4)
1MB
(5)
PROG
(20)
1MB
(6)
(6)
BLEND
(10)
(3)
10KB
(4)
TEXT
(5)
(5)
10KB

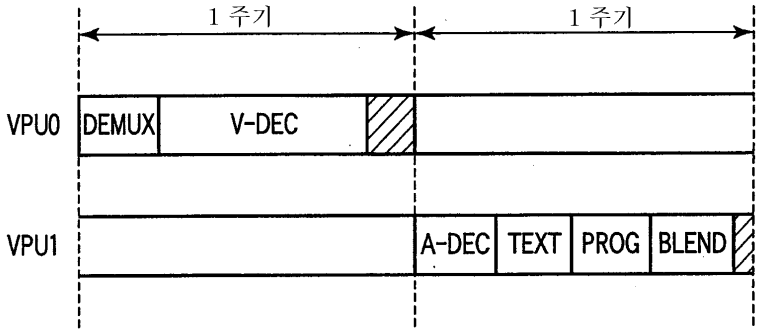
도면54

BUFFER Id : 1 Size : 100KB SrcTask : 1 DstTask : 2
BUFFER Id : 2 Size : 1MB SrcTask : 1 DstTask : 3
BUFFER Id : 3 Size : 10KB SrcTask : 1 DstTask : 4
BUFFER Id : 4 Size : 1MB SrcTask : 3 DstTask : 5
BUFFER Id : 5 Size : 10KB SrcTask : 4 DstTask : 6
BUFFER Id : 6 Size : 1MB SrcTask : 5 DstTask : 6
TASK Id : 1 Class : VPU,HRT ThreadContext : DEMUX Cost : 5 Constraint : Precede : 2,3,4 InputBuffer : OutputBuffer : 1,2,3
TASK Id : 2 Class : VPU,HRT ThreadContext : A-DEC Cost : 10 Constraint : Precede : InputBuffer : 1 OutputBuffer :
TASK Id : 3 Class : VPU,HRT ThreadContext : V-DEC Cost : 50 Constraint : Precede : 5 InputBuffer : 2 OutputBuffer : 4
TASK Id : 4 Class : VPU,HRT ThreadContext : TEXT Cost : 5 Constraint : Precede : 6 InputBuffer : 3 OutputBuffer : 5
TASK Id : 5 Class : VPU,HRT ThreadContext : PROG Cost : 20 Constraint : Precede : 6 InputBuffer : 4 OutputBuffer : 6
TASK Id : 6 Class : VPU,HRT ThreadContext : BLEND Cost : 10 Constraint : Precede : 5 InputBuffer : 5,6 OutputBuffer :

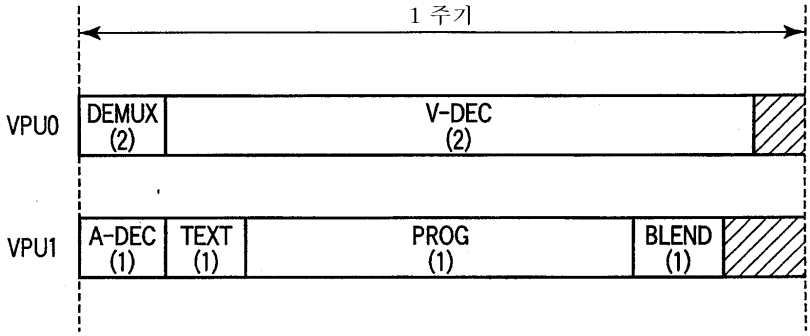
도면55



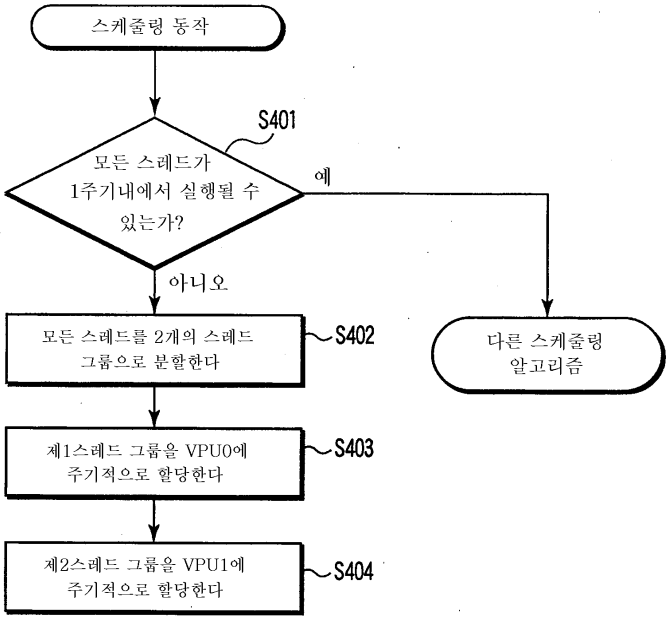
도면56



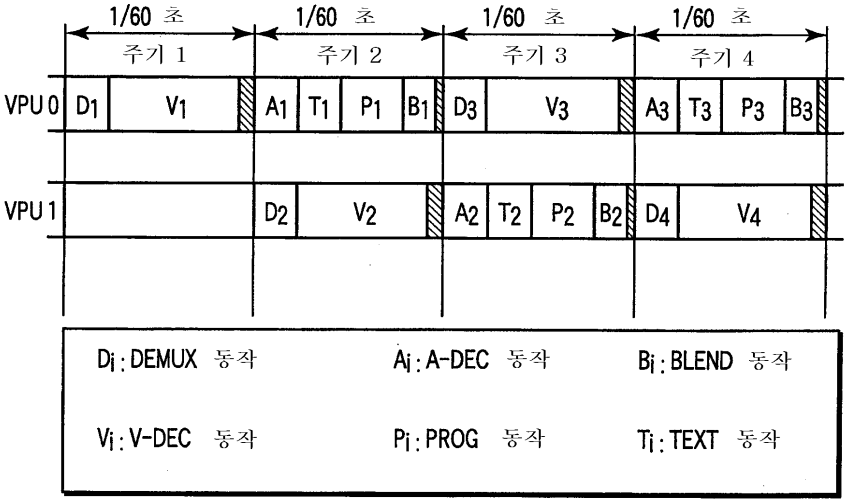
도면57



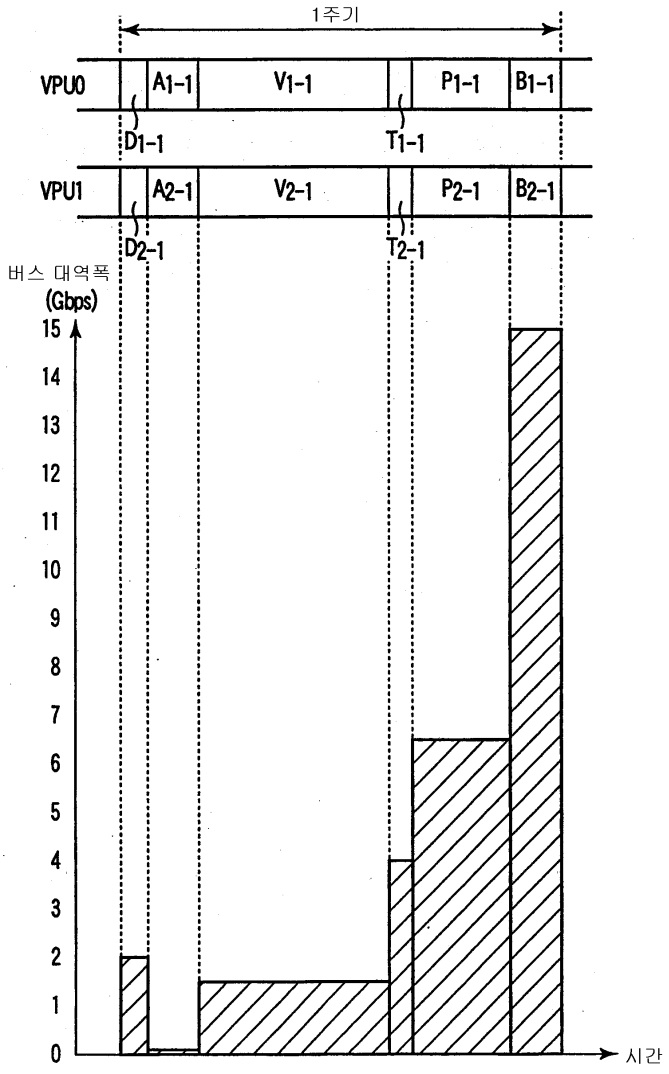
도면58



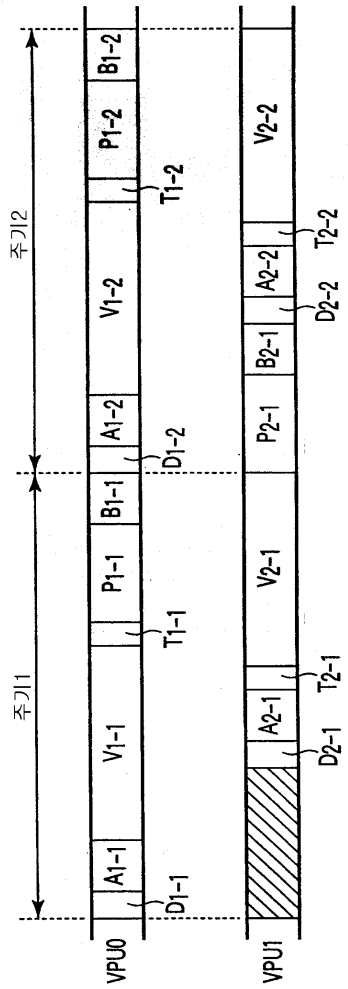
도면59


1/60 초
주기 1
1/60 초
주기 2
1/60 초
주기 3
1/60 초
주기 4
VPU 0
D1
V1
A1
T1
P1
B1
D3
V3
A3
T3
P3
B3
VPU 1
D2
V2
A2
T2
P2
B2
D4
V4
Di : DEMUX 동작
Ai : A-DEC 동작
Bi : BLEND 동작
Vi : V-DEC 동작
Pi : PROG 동작
Ti : TEXT 동작

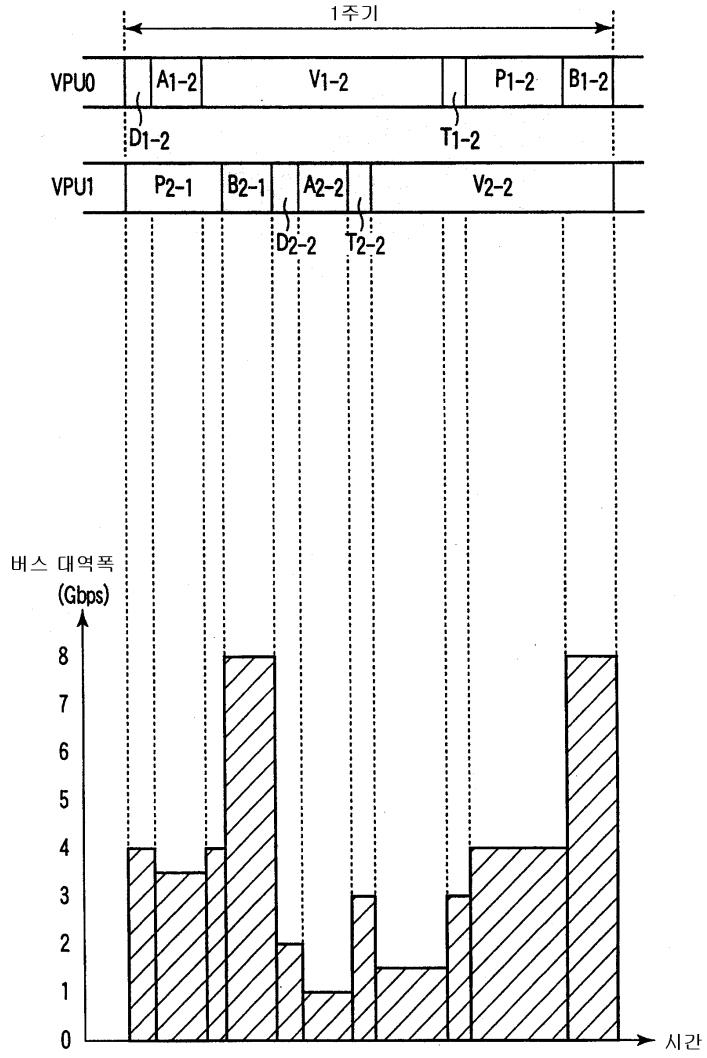
도면60


1주기
VPU0
A1-1
V1-1
P1-1
B1-1
D1-1
T1-1
VPU1
A2-1
V2-1
P2-1
B2-1
D2-1
T2-1
버스 대역폭
(Gbps)
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
시간

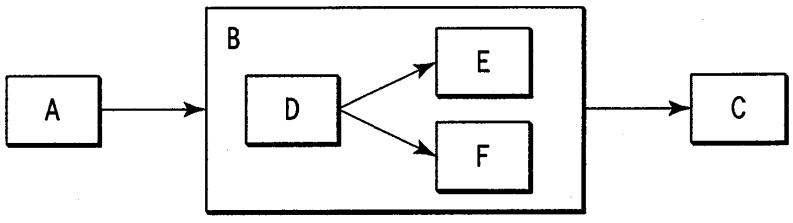
도면61


주기2
주기1
VPU0
VPU1
A1-1
V1-1
D1-1
P1-1
B1-1
T1-1
A1-2
V1-2
D1-2
P1-2
B1-2
T1-2
A2-1
V2-1
D2-1
P2-1
B2-1
T2-1
A2-2
V2-2
D2-2
T2-2

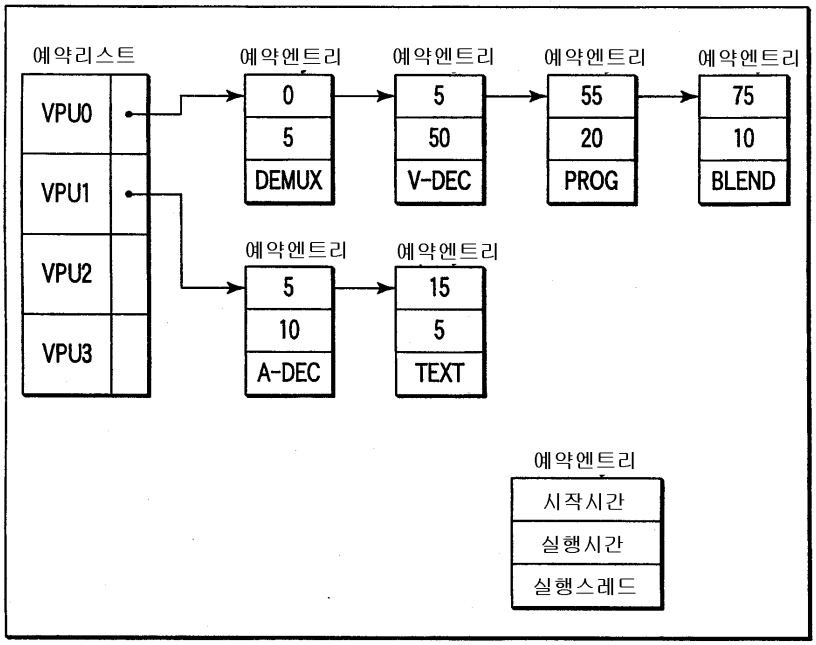
도면62


1주기
VPU0
A1-2
V1-2
P1-2
B1-2
D1-2
T1-2
VPU1
P2-1
B2-1
A2-2
V2-2
D2-2
T2-2
버스 대역폭
(Gbps)
8
7
6
5
4
3
2
1
0
시간

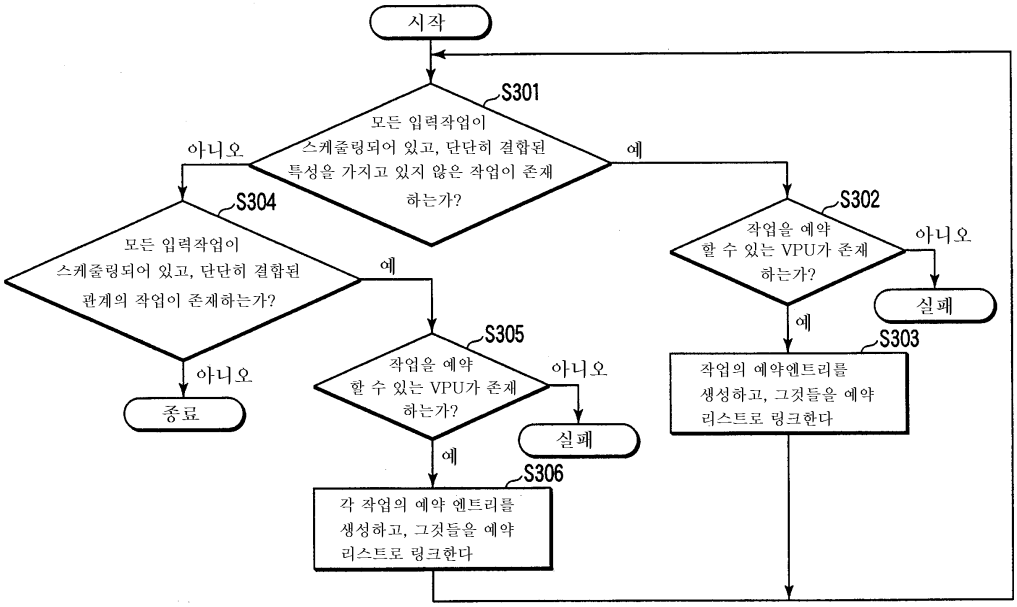
도면63


A
B
D
E
F
C

도면64


예약리스트
VPU0
VPU1
VPU2
VPU3
예약엔트리
0
5
DEMUX
예약엔트리
5
50
V-DEC
예약엔트리
55
20
PROG
예약엔트리
75
10
BLEND
예약엔트리
5
10
A-DEC
예약엔트리
15
5
TEXT
예약엔트리
시작시간
실행시간
실행스레드

도면65


시작
S301
모든 입력작업이 스케줄링되어 있고, 단단히 결합된 특성을 가지고 있지 않은 작업이 존재하는가?
아니오
예
S304
모든 입력작업이 스케줄링되어 있고, 단단히 결합된 관계의 작업이 존재하는가?
예
아니오
종료
S305
작업을 예약할 수 있는 VPU가 존재하는가?
아니오
실패
예
S306
각 작업의 예약 엔트리를 생성하고, 그것들을 예약 리스트로 링크한다
S302
작업을 예약할 수 있는 VPU가 존재하는가?
아니오
실패
예
S303
작업의 예약엔트리를 생성하고, 그것들을 예약 리스트로 링크한다