Slide 0

13장. 멀티 프로세서

멀티프로세서의 특성

- 멀티프로세서 시스템이란?
  - 메모리와 입출력 장치를 공유하는 두개 이상의 CPU를 가진 시스템
  - MIMD 로 분류
  - 병렬처리로 성능향상
  - fault tolerant 컴퓨터로 동작할수 있음

Slide 1

- 멀티 프로세서의 병렬처리
  - 다수의 독립적인 작업이 병렬로 수행
  - 하나의 작업이 여러부분으로 나뉘어 병렬적으로 수행
    - 사용자가 나누는 방식
    - 컴파일러가 나눈는 방식 (병렬 컴파일러 사용)
- 멀티 프로세서의 분류
  - 공유메모리 (shared-memory) 혹은 tightly coupled
    - 모든 프로세서가 공통으로 사용하는 공유메모리가 있음

멀티프로세서의 특성 (계속)

Slide 2

  * 각각의 프로세서는 캐쉬메모리와 같은 local memory 를 가질수 있음
  * 상호작용이 많은 프로그램에 적합

– 분산메모리 (distributed memory) 혹은 loosely coupled

  * 각 프로세서는 local memory 를 가짐
  * 프로세서간 정보전달은 메세지 패싱 (message-passing) 방식으로 이루어짐
  * 상호작용이 적은 프로그램에 적합

상호 연결 구조

Slide 3

- 상호연결 구조의 형태
  1. 시분할 공통 버스
  2. 다중 포트 메모리
  3. 크로스바 스위치
  4. 다단 교환망
  5. 하이퍼큐브 시스템
- 시분할 공통 버스
  – 다섯개의 프로세서를 위한 시분할 공통 버스 구조 (그림 13-1)

상호 연결 구조 (계속)

Slide 4

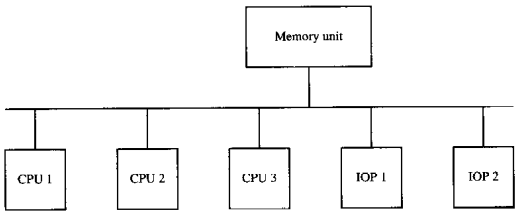


그림 13-1 시분할 공통 버스 구조

- 공통의 버스 사용 →버스 효율성이 줄어듬
- 이중 버스구조 (그림 13-2)

상호 연결 구조 (계속)

Slide 5

[illegible] 프로세서에 의하여 쓸 수 있다. 단 하나의 프로세서만이 주어진 시간에 시스템 버스를 사용하여 공통의 메모리나 다른 공용 시설과 통신할 수 있으며 다른 프로세서는 자신의 로컬 메모리와 입출력 장치와 계속 통신할 수 있다. 이러한 방식은 멀티컴퓨터 시스템으로 분류되기도 하는데 이는 CPU, IOP, 메모

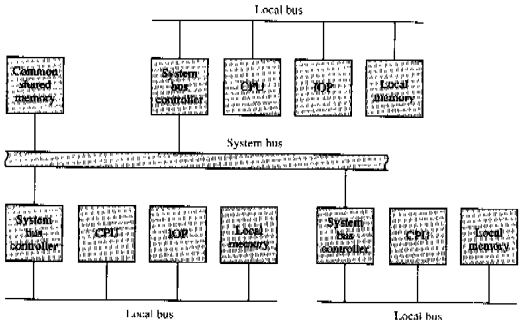


그림 13-2 [illegible]

- 다중 포트 메모리
  - 구조 (그림 13-3)

상호 연결 구조 (계속)

Slide 6

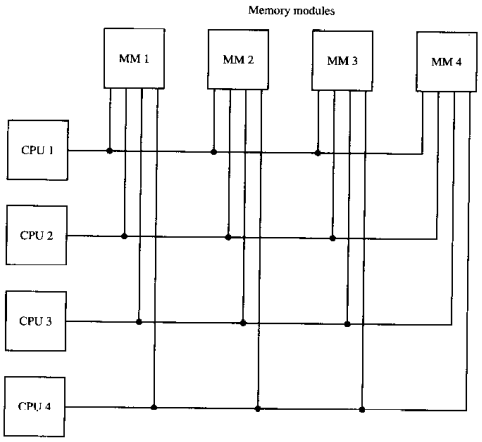


그림 13-3 다중프로 메모리 구조

- 다중 통로에 의하여 높은 전송률을 지님

상호 연결 구조 (계속)

Slide 7

- 비싼 메모리 제어 논리와 많은 수의 전선및 커넥터 필요 →비용 증가

- 크로스바 스위치
  - 크로스바 스위치의 구성 (그림 13-4) 및 블럭도 (그림 13-5)

상호 연결 구조 (계속)

Slide 8

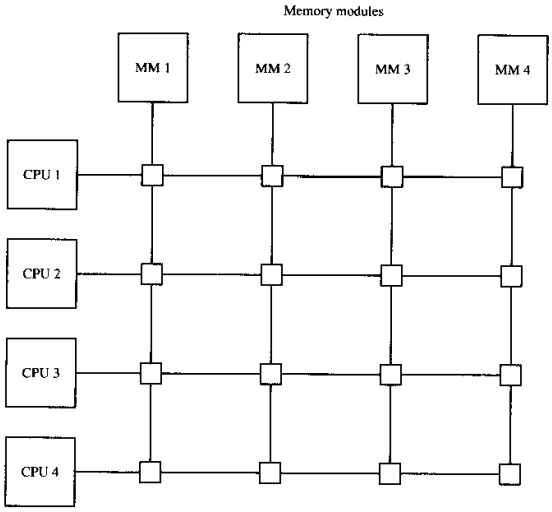


그림 13-4 크로스바 스위치

상호 연결 구조 (계속)

Slide 9

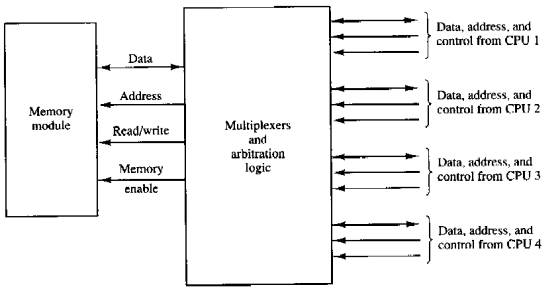
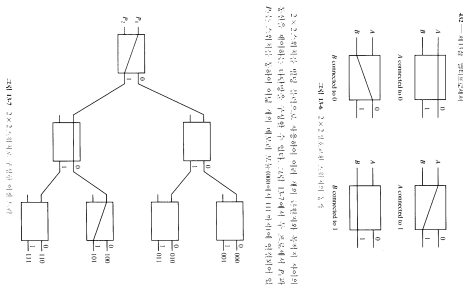


그림 13-5 크로스바 스위치의 블럭도

- 다단 교환망
  - 기본요소 →2입력, 2출력 상호교환 스위치 (그림 13-6,7)

상호 연결 구조 (계속)

Slide 10



- 8 × 8 오메가 교환망 (그림 13-8)

상호 연결 구조 (계속)

Slide 11

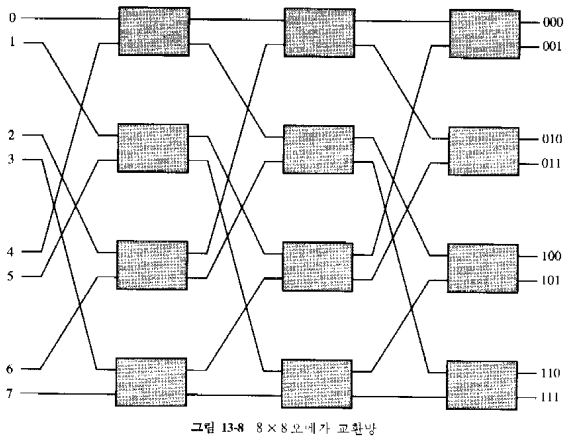


그림 13-8 8×8 오메가 교환망

- 하이퍼큐브 연결
  - 하이퍼큐브 연결 방식 (그림 13-9)

상호 연결 구조 (계속)

Slide 12

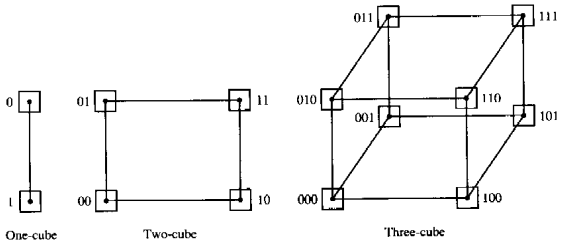


그림 13-9 $n = 1, 2, 3$에 대한 하이퍼큐브

- n큐브 구조에서 메세지 전송은 하나에서 n 개까지의 링크를 거침

프로세서간의 중재

Slide 13

- 여러프로세가선의 버스 요구를 중재할 필요가 있음
- 시스템 버스
  - 여러개의 프로세서가 프로세서간 연결을위해 필요한 버스
- 직렬 중재 절차
  - 직렬 중재 (그림 13-10)

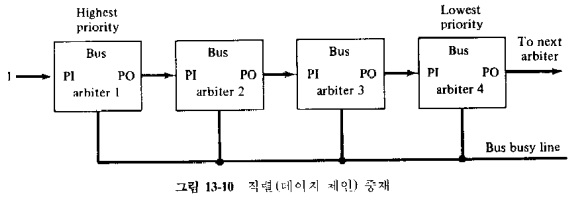


그림 13-10 직렬(데이지 체인) 중재

프로세서간의 중재 (계속)

Slide 14

  - 우선순위 제어라인에 가까운 프로세서의 우선순위가 높음

- 병렬 중재 논리
  - 병렬 중재 (그림 13-11)

프로세서간의 중재 (계속)

Slide 15

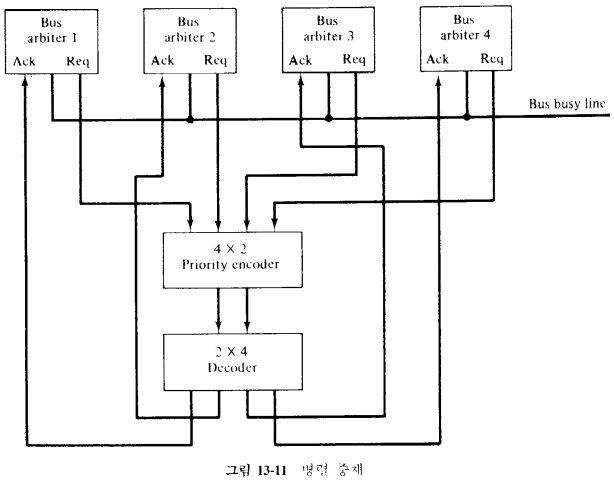


그림 13-11 병렬 중재

- 동적 중재 알고리즘

프로세서간의 중재 (계속)

Slide 16

- 시간 분할 (time slice)
  - 시간을 균등 분할하여 각 프로세서에게 할당
- 폴링 (polling)
  - 프로그램으로 각각의 프로세서에게 버스 할당
- LRU (least recently used) 알고리즘
  - 가장 오랫동안 버스를 사용하지 않는 프로세서에게 가장 높은 우선순위를 부여
- first-come, first-serve
  - 요구한 순서대로 버스를 사용하게 함
- 회전 데이지 체인 (rotating daisy chain)
  - 중앙 버스 제어기가 없음
  - 맨끝의 PO 를 맨처음의 PI 에 연결
  - 현재 버스를 사용하는것이 버스 제어 역할 수행
  - 현재 버스를 사용하는것 바로 뒤에것이 우선순위 가장 높음

프로세서간 통신과 동기화

Slide 17

- 멀티프로세서 시스템에서 프로세서간 자원이용
  - 공유메모리 사용시
    - 해당 공유메모리에 전달할 데이타를 넣음
    - 수신 프로세서는 폴링이나 인터럽트 (송신 프로세서가 활성화) 를 받고 처리
  - 멀티 프로세서에서의 운영체제 방식
    - 주종 (master-slave) 형식
      - 주 프로세서만 운용체제 수행 종 프로세서는 인터럽트로 서비스 요구
    - 분리형 (separate) 형식
      - 각자의 프로세서가 필요한 운영체제 루틴을 수행
    - 분산 (distrubuted) 형식
      - 운영체제가 여러 프로세서에 분산되어 있음
- 프로세서간 동기화
  - 공유의 기록 장치에 상호배재 (mutual exclusive) 보장
  - 방법

프로세서간 통신과 동기화 (계속)

Slide 18

* 세마포를 이용한 상호배제
  - 하나의 프로세서가 임계 프로그램 영역 (critical program section)에 있을때
  - 다른 프로세서의 공유영역 사용 방지
  - 세마포 라는 이진변수를 이용
  - 세마포가 1이면 다른 프로세서는 공용의 메모리 사용 못함
  - 세마포 변경시 lock 을 걸어 다른 프로세서가 세마포 변경을 금지

캐쉬의 일관성

- 멀티프로세서에서 각각의 캐쉬에 있는 데이타는 일관성이 있어야함
- 읽기 전용 공용메모리는 일관성이 유지됨
- 캐쉬 일관성의 문제
  - 캐쉬의 구성 (그림 13-12)

Slide 19

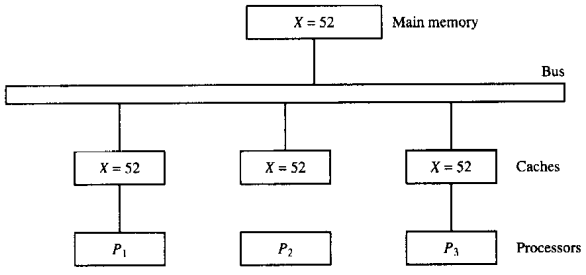
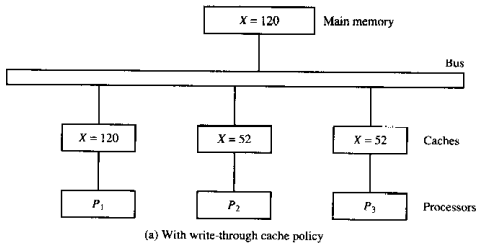


그림 13-12 X를 적재한 후의 캐시 구성

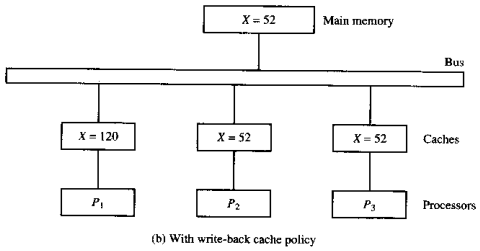
캐쉬의 일관성 (계속)

Slide 20

– 캐쉬 비일관성 (그림 13-13)



(a) With write-through cache policy



(b) With write-back cache policy

그림 13-13 P_1에 의한 X로의 store 후의 캐시 구성

캐쉬의 일관성 (계속)

Slide 21

- 캐쉬 일관성 문제에 대한 해결책
  - 공통의 캐쉬 사용
    - 평균 접근 시간 증가
  - 캐쉬 제한
    - 공유 기록 가능 데이타는 캐쉬하지 않음
    - 소프트웨어의 오버헤드 발생
  - 표를 이용
    - 중앙집중식 전역표를 이용
    - 각각의 블럭을 읽기 전용과 읽기및 쓰기가능으로 분류
    - 읽기 전용만 캐쉬, 단 하나의 캐쉬만 읽기 및 쓰기 데이타 캐쉬
  - 버스 감시
    - 캐쉬가 버스 쓰기 요구를 감시하여 자신의 캐쉬의 valid 비트를 제어