UNIT V MULTIPROCESSORS

Characteristics of multiprocessors – Interconnection structures – Inter processor arbitration – Inter processor communication and synchronization – Cache coherence

5.1 Multiprocessor:

- A set of processors connected by a communications network

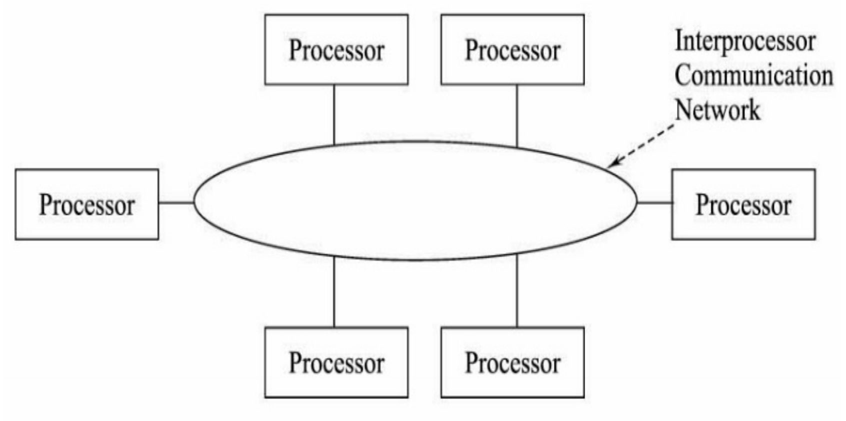


Fig. 5.1 Basic multiprocessor architecure

- A multiprocessor system is an interconnection of two or more CPU's with memory and input-output equipment.
- Multiprocessors system are classified as multiple instruction stream, multiple data stream systems(MIMD).
- There exists a distinction between multiprocessor and multicomputers that though both support concurrent operations.
- In multicomputers several autonomous computers are connected through a network and they may or may not communicate but in a multiprocessor system there is a single OS Control that provides interaction between processors and all the components of the system to cooperate in the solution of the problem.
- VLSI circuit technology has reduced the cost of the computers to such a low Level that the concept of applying multiple processors to meet system performance requirements has become an attractive design possibility.

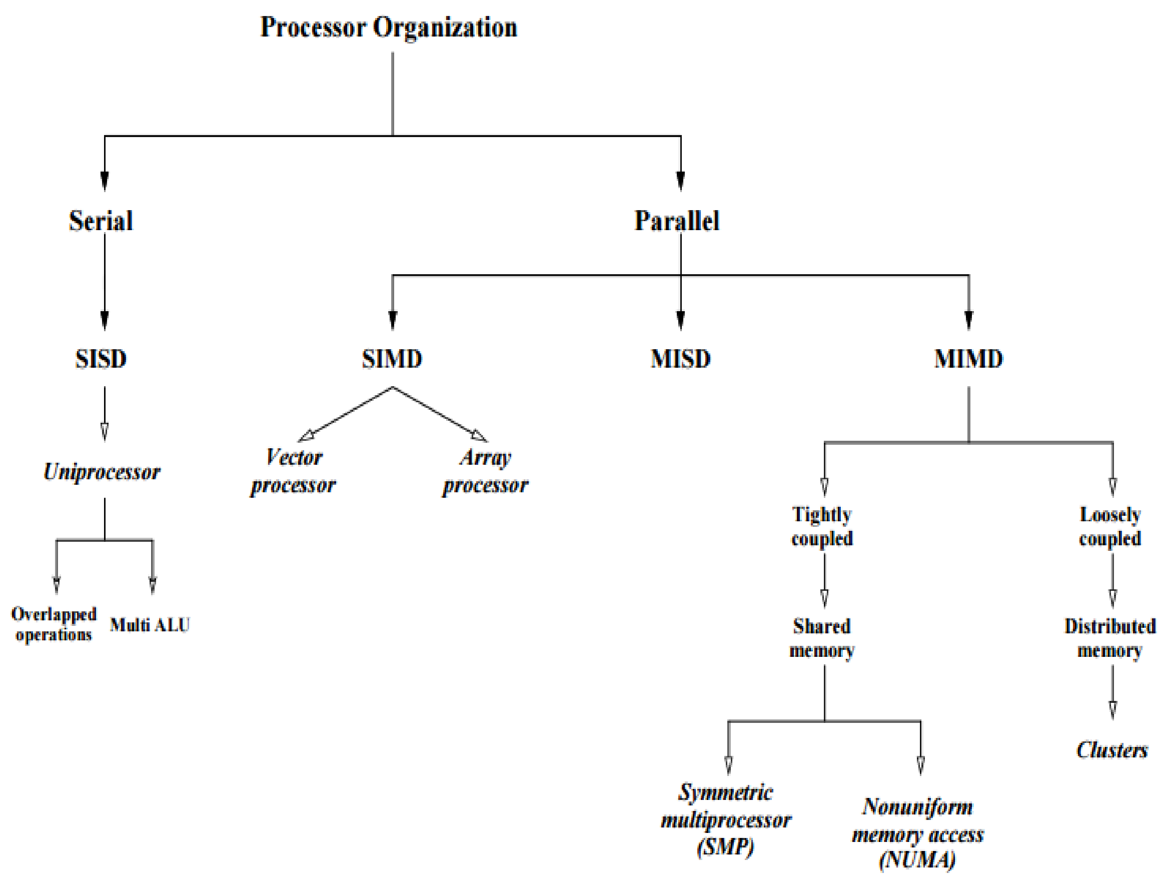


Fig. 5.2 Taxonomy of mono- mulitporcessor organizations

<u>Characteristics of Multiprocessors:</u>

Benefits of Multiprocessing:

1. Multiprocessing increases the reliability of the system so that a failure or error in one part has limited effect on the rest of the system. If a fault causes one processor to fail, a second processor can be assigned to perform the functions of the disabled one.

2. Improved System performance. System derives high performance from the fact that computations can proceed in parallel in one of the two ways:

a) Multiple independent jobs can be made to operate in parallel.

b) A single job can be partitioned into multiple parallel tasks.

This can be achieved in two ways:

- The user explicitly declares that the tasks of the program be executed in parallel

- The compiler provided with multiprocessor s/w that can automatically detect parallelism in program. Actually it checks for Data dependency

COUPLING OF PROCESSORS

Tightly Coupled System/Shared Memory:

- Tasks and/or processors communicate in a highly synchronized fashion
- Communicates through a common global shared memory
- Shared memory system. This doesn’t preclude each processor from having its own local memory(cache memory)

Loosely Coupled System/Distributed Memory

- Tasks or processors do not communicate in a synchronized fashion.
- Communicates by message passing packets consisting of an address, the data content, and some error detection code.
- Overhead for data exchange is high
- Distributed memory system

Loosely coupled systems are more efficient when the interaction between tasks is minimal, whereas tightly coupled system can tolerate a higher degree of interaction between tasks.

Shared (Global) Memory

- A Global Memory Space accessible by all processors
- Processors may also have some local memory

Distributed (Local, Message-Passing) Memory

- All memory units are associated with processors
- To retrieve information from another processor's memory a message must be sent there

Uniform Memory

- All processors take the same time to reach all memory locations

Non-uniform (NUMA) Memory

- Memory access is not uniform

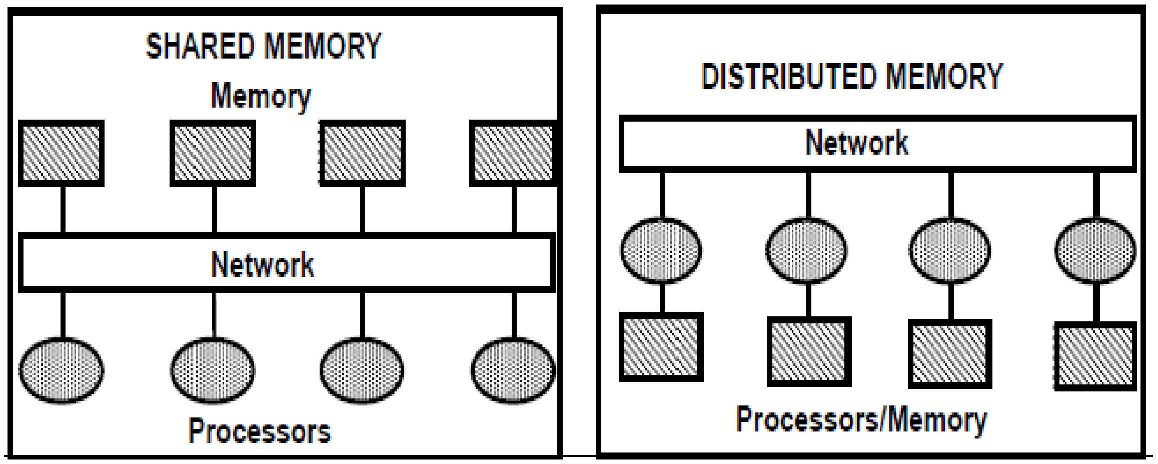


Fig. 5.3 Shared and distributed memory

Shared memory multiprocessor:

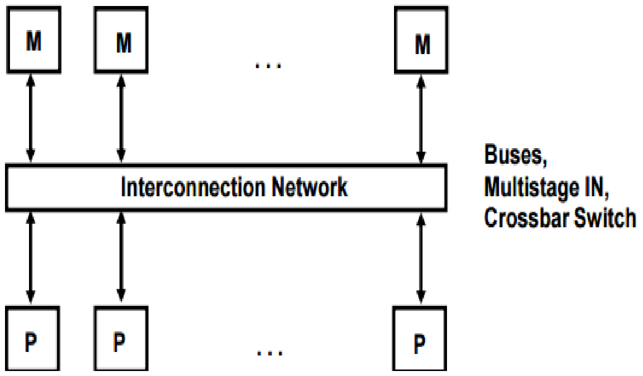


Fig 5.4 Shared memory multiprocessor

Characteristics

- All processors have equally direct access to one large memory address space

Limitations

- Memory access latency; Hot spot problem

<u>5.2 Interconnection Structures</u>:

The interconnection between the components of a multiprocessor System can have different physical configurations depending n the number of transfer paths that are available between the processors and memory in a shared memory system and among the processing elements in a loosely coupled system.

Some of the schemes are as:

- Time-Shared Common Bus
- Multiport Memory
- Crossbar Switch
- Multistage Switching Network
- Hypercube System

a. Time shared common Bus

- All processors (and memory) are connected to a common bus or busses
- Memory access is fairly uniform, but not very scalable
- A collection of signal lines that carry module-to-module communication
- Data highways connecting several digital system elements
- Operations of Bus

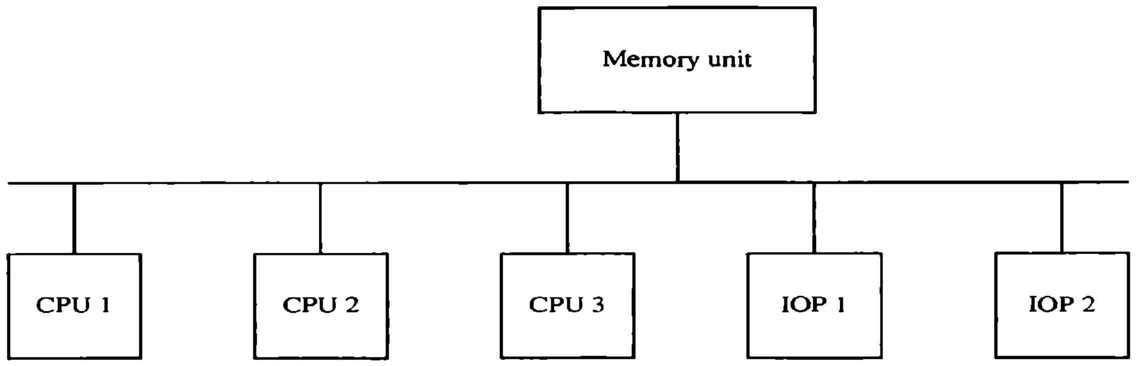


Fig. 5.5 Time shared common bus organization

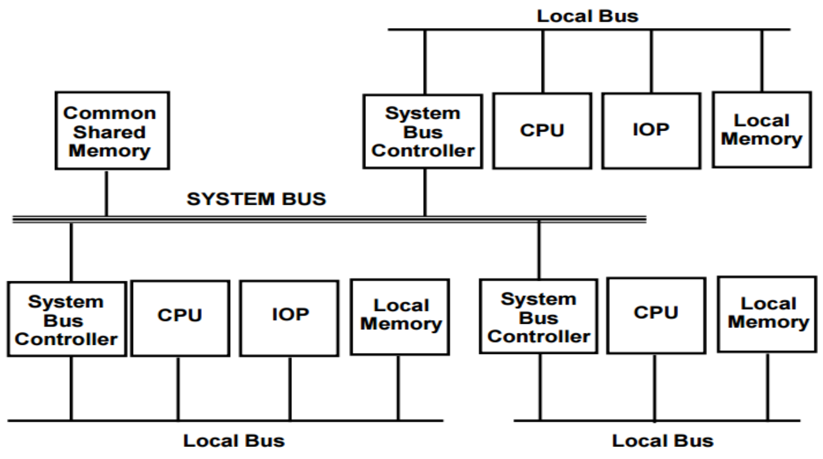


Fig. 5.6 system bus structure for multiprocessor

In the above figure we have number of local buses to its own local memory and to one or more processors. Each local bus may be connected to a CPU, an IOP, or any combinations of processors. A system bus controller links each local bus to a common system bus. The I/O devices connected to the local IOP, as well as the local memory, are available to the local processor. The memory connected to the common system bus is shared by all processors. If an IOP is connected directly to the system bus the I/O devices attached to it may be made available to all processors

Disadvantage.:

- Only one processor can communicate with the memory or another processor at any given time.
- As a consequence, the total overall transfer rate within the system is limited by the speed of the single path

b. Multiport Memory:

Multiport Memory Module

- Each port serves a CPU

Memory Module Control Logic

- Each memory module has control logic
- Resolve memory module conflicts Fixed priority among CPUs

Advantages

- The high transfer rate can be achieved because of the multiple paths.

Disadvantages:

- It requires expensive memory control logic and a large number of cables and connections

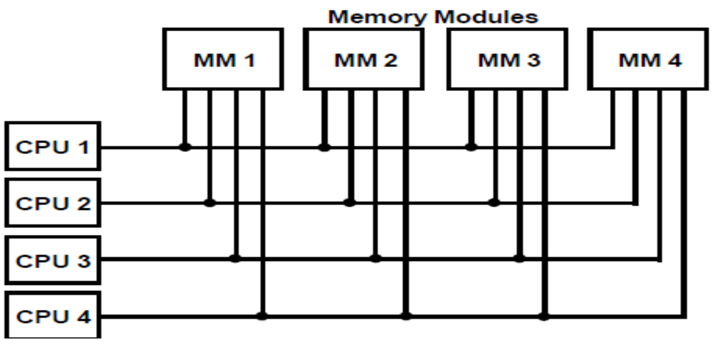
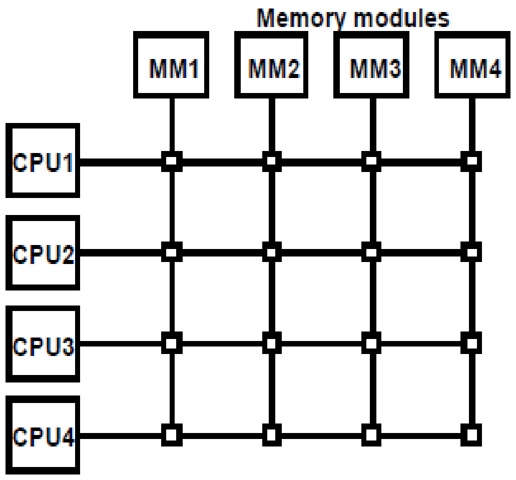
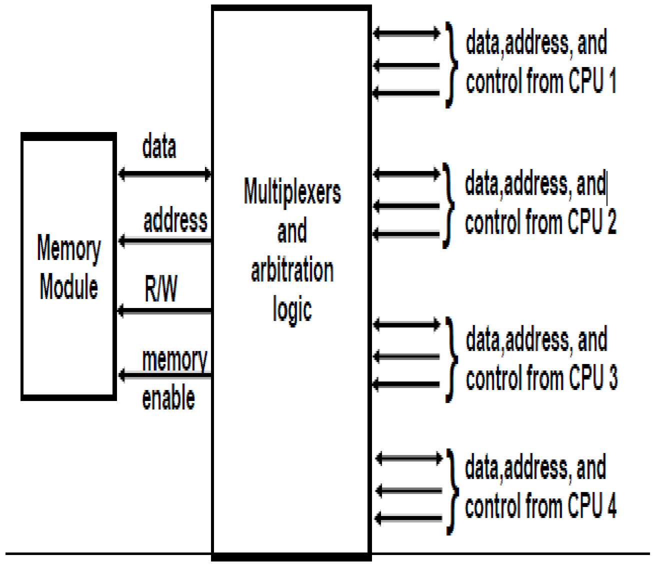


Fig. 5.7 Multiport memory

c. Crossbar switch:

- Each switch point has control logic to set up the transfer path between a processor and a memory.
- It also resolves the multiple requests for access to the same memory on the predetermined priority basis.
- Though this organization supports simultaneous transfers from all memory modules because there is a separate path associated with each Module.
- The H/w required to implement the switch can become quite large and complex




Memory Module
data
address
R/W
memory enable
Multiplexers and arbitration logic
data,address, and control from CPU 1
data,address, and control from CPU 2
data,address, and control from CPU 3
data,address, and control from CPU 4

a) b)

Fig. 5.8 a) cross bar switch b) Block diagram of cross bar switch

Advantage:

- Supports simultaneous transfers from all memory modules

Disadvantage:

- The hardware required to implement the switch can become quite large and complex.

d. Multistage Switching Network:

- The basic component of a multi stage switching network is a two-input, two-output interchange switch.

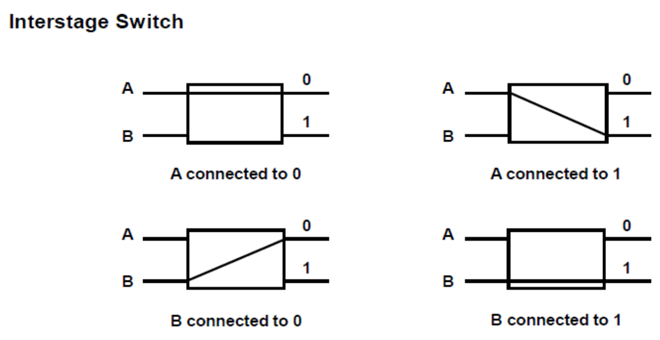


Fig. 5.9 operation of 2X2 interconnection switch

Using the 2x2 switch as a building block, it is possible to build a multistage network to control the communication between a number of sources and destinations.

- To see how this is done, consider the binary tree shown in Fig. below.
- Certain request patterns cannot be satisfied simultaneously.

  i.e., if P1 → 000~011, then P2 → 100~111

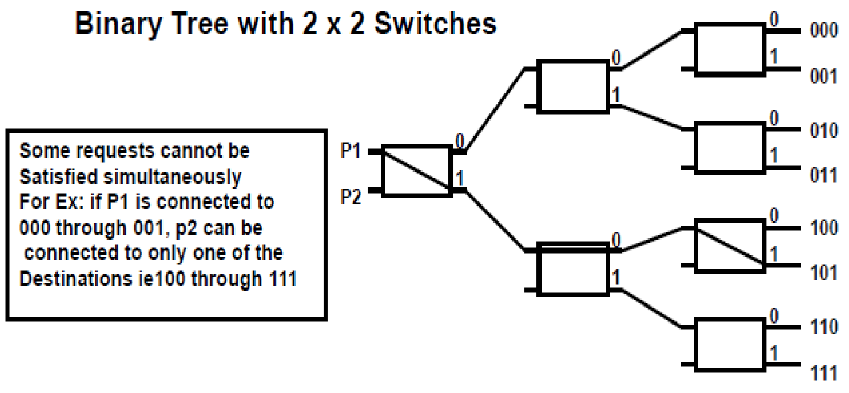

Binary Tree with 2 x 2 Switches
Some requests cannot be Satisfied simultaneously For Ex: if P1 is connected to 000 through 001, p2 can be connected to only one of the Destinations ie100 through 111
P1
P2
0
1
000
001
010
011
100
101
110
111

Fig 5.10 Binary tree with 2x2 switches

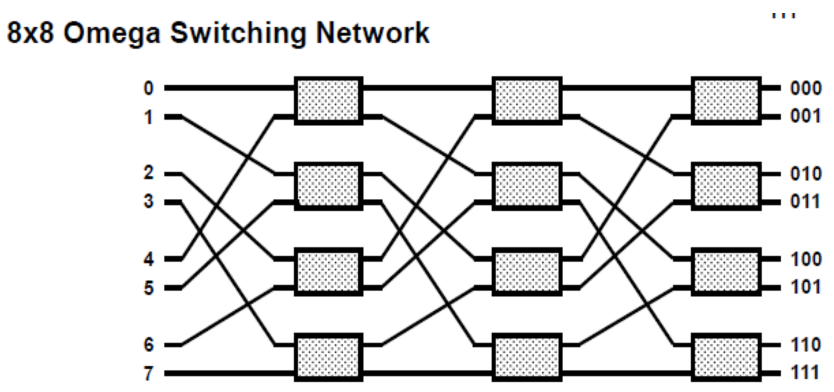


Fig. 5.11 8X8 Omega switching network

- Some request patterns cannot be connected simultaneously. i.e., any two sources cannot be connected simultaneously to destination 000 and 001
- In a tightly coupled multiprocessor system, the source is a processor and the destination is a memory module.
- Set up the path → transfer the address into memory → transfer the data
- In a loosely coupled multiprocessor system, both the source and destination are Processsing elements.

e. Hypercube System:

The hypercube or binary n-cube multiprocessor structure is a loosely coupled system composed of N=2n processors interconnected in an n-dimensional binary cube.

- Each processor forms a node of the cube, in effect it contains not only a CPU but also local memory and I/O interface.
- Each processor address differs from that of each of its n neighbors by exactly one bit position.
- Fig. below shows the hypercube structure for n=1, 2, and 3.
- Routing messages through an *n*-cube structure may take from one to *n* links from a source node to a destination node.
- A routing procedure can be developed by computing the exclusive-OR of the source node address with the destination node address.
- The message is then sent along any one of the axes that the resulting binary value will have 1 bits corresponding to the axes on which the two nodes differ.
- A representative of the hypercube architecture is the Intel iPSC computer complex.
- It consists of 128(*n*=7) microcomputers, each node consists of a CPU, a floating point processor, local memory, and serial communication interface units

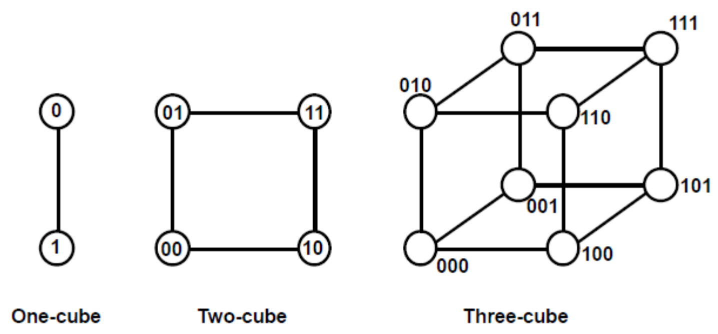


Fig. 5.12 Hypercube structures for n=1,2,3

5.3 Inter-processor Arbitration

- Only one of CPU, IOP, and Memory can be granted to use the bus at a time
- Arbitration mechanism is needed to handle multiple requests to the shared resources to resolve multiple contention
- SYSTEM BUS:
  - A bus that connects the major components such as CPU's, IOP's and memory
  - A typical System bus consists of 100 signal lines divided into three functional groups: data, address and control lines. In addition there are power distribution lines to the components.
- Synchronous Bus
  - Each data item is transferred over a time slice
  - known to both source and destination unit
  - Common clock source or separate clock and synchronization signal is transmitted periodically to synchronize the clocks in the system
- Asynchronous Bus
  - Each data item is transferred by Handshake mechanism
    - Unit that transmits the data transmits a control signal that indicates the presence of data
    - Unit that receiving the data responds with another control signal to acknowledge the receipt of the data

- Strobe pulse -supplied by one of the units to indicate to the other unit when the data transfer has to occur

Table 5.1 IEEE standard 796 multibus signals

	Signal name
Data and address	
Data lines (16 lines)	DATA0–DATA15
Address lines (24 lines)	ADRS0–ADRS23
Data transfer	
Memory read	MRDC
Memory write	MWTC
IO read	IORC
IO write	IOWC
Transfer acknowledge	TACK
Interrupt control	
Interrupt request (8 lines)	INT0–INT7
Interrupt acknowledge	INTA
Miscellaneous control	
Master clock	CCLK
System initialization	INIT
Byte high enable	BHEN
Memory inhibit (2 lines)	INH1–INH2
Bus lock	LOCK
Bus arbitration	
Bus request	BREQ
Common bus request	CBRQ
Bus busy	BUSY
Bus clock	BCLK
Bus priority in	BPRN
Bus priority out	BPRO
Power and ground (20 lines)	

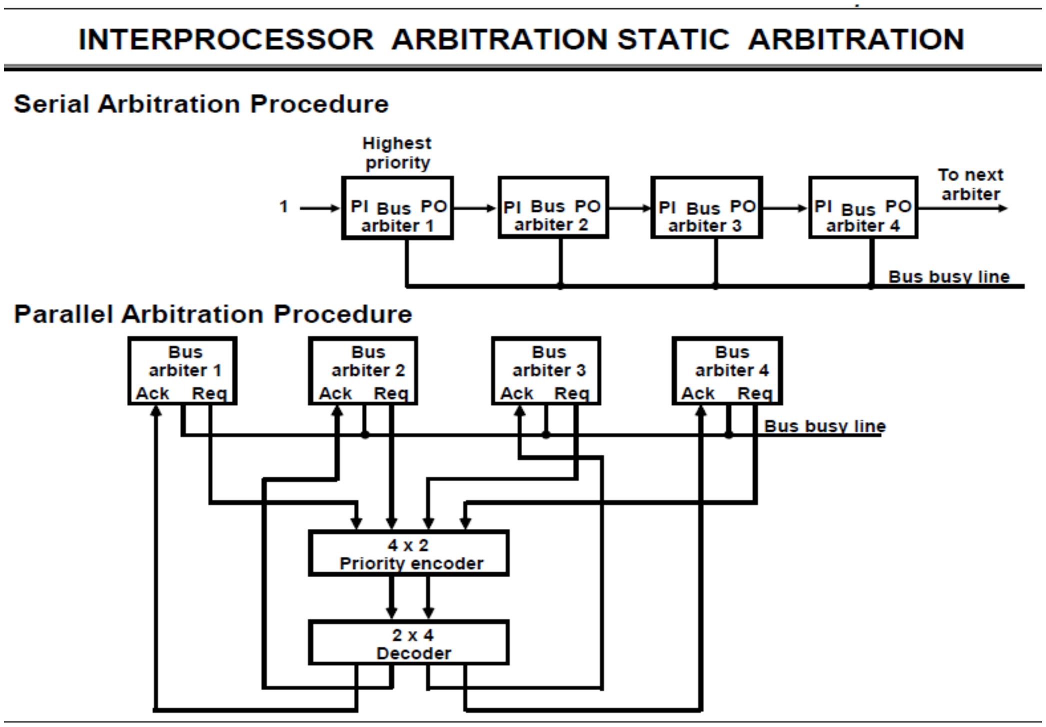


Fig. 5.13 Inter-processor arbitration static arbitration

Interprocessor Arbitration Dynamic Arbitration

- Priorities of the units can be dynamically changeable while the system is in operation
- Time Slice
  - Fixed length time slice is given sequentially to each processor, round-robin fashion
- Polling
  - Unit address polling -Bus controller advances the address to identify the requesting unit. When processor that requires the access recognizes its address, it activates the bus busy line and then accesses the bus. After a number of bus cycles, the polling continues by choosing a different processor.
- LRU
  - The least recently used algorithm gives the highest priority to the requesting device that has not used bus for the longest interval.
- FIFO
  - The first come first serve scheme requests are served in the order received. The bus controller here maintains a queue data structure.
- Rotating Daisy Chain
  - Conventional Daisy Chain -Highest priority to the nearest unit to the bus controller
  - Rotating Daisy Chain –The PO output of the last device is connected to the PI of the first one. Highest priority to the unit that is nearest to the unit that has most recently accessed the bus(it becomes the bus controller)

<u>5.4 Inter processor communication and synchronization:</u>

- The various processors in a multiprocessor system must be provided with a facility for *communicating* with each other.
  - A communication path can be established through *a portion of memory* or *a common input-output channels*.

- The sending processor structures a request, a message, or a procedure, and places it in the memory mailbox.
    - *Status bits* residing in common memory
    - The receiving processor can check the mailbox *periodically*.
    - The response time of this procedure can be time consuming.
- A more efficient procedure is for the sending processor to alert the receiving processor directly by means of an *interrupt signal*.
- In addition to shared memory, a multiprocessor system may have other shared resources.
    - e.g., a magnetic disk storage unit.
- To prevent conflicting use of shared resources by several processors there must be a provision for assigning resources to processors. i.e., operating system.
- There are three organizations that have been used in the design of operating system for multiprocessors: *master-slave configuration*, *separate operating system*, and *distributed operating system*.
- In a master-slave mode, one processor, master, always executes the operating system functions.
- In the separate operating system organization, each processor can execute the operating system routines it needs. This organization is more suitable for *loosely coupled systems*.
- In the distributed operating system organization, the operating system routines are distributed among the available processors. However, each particular operating system function is assigned to only one processor at a time. It is also referred to as a *floating operating system*.

Loosely Coupled System

- There is *no shared memory* for passing information.
- The communication between processors is by means of message passing through *I/O channels*.
- The communication is initiated by one processor calling a *procedure* that resides in the memory of the processor with which it wishes to communicate.

- The communication efficiency of the interprocessor network depends on the *communication routing protocol, processor speed, data link speed, and the topology of the network*.

Interprocess Synchronization

- The instruction set of a multiprocessor contains basic instructions that are used to implement communication and synchronization between cooperating processes.
    - Communication refers to the exchange of data between different processes.
    - Synchronization refers to the special case where the data used to communicate between processors is control information.
- Synchronization is needed to enforce the *correct sequence of processes* and to ensure *mutually exclusive access* to shared writable data.
- Multiprocessor systems usually include various mechanisms to deal with the synchronization of resources.
    - Low-level primitives are implemented directly by the hardware.
    - These primitives are the basic mechanisms that enforce mutual exclusion for more complex mechanisms implemented in software.
    - A number of hardware mechanisms for mutual exclusion have been developed.
        - A binary semaphore

Mutual Exclusion with Semaphore

- A properly functioning multiprocessor system must provide a mechanism that will guarantee orderly access to shared memory and other shared resources.
    - Mutual exclusion: This is necessary to protect data from being changed simultaneously by two or more processors.
    - Critical section: is a program sequence that must complete execution before another processor accesses the same shared resource.
- A *binary variable* called a *semaphore* is often used to indicate whether or not a processor is executing a critical section.

- Testing and setting the semaphore is itself a critical operation and must be performed as a single indivisible operation.
- A semaphore can be initialized by means of a *test and set instruction* in conjunction with a hardware *lock* mechanism.
- The instruction TSL SEM will be executed in two memory cycles (the first to read and the second to write) as follows:

$$R \leftarrow M[SEM],\ M[SEM] \leftarrow 1$$

5.5 Cache Coherence

cache coherence is the consistency of shared resource data that ends up stored in multiple local **caches**. When clients in a system maintain **caches** of a common memory resource, problems may arise with inconsistent data, which is particularly the case with CPUs in a multiprocessing system.

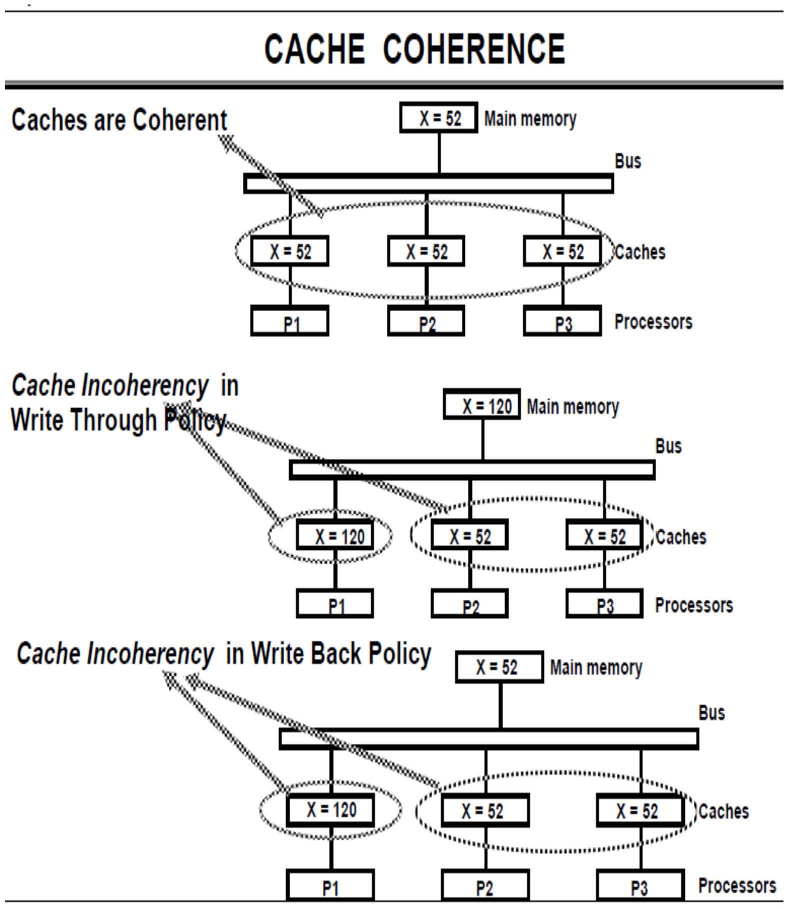


Fig. 5.14 cache coherence

Shared Cache

-Disallow private cache

-Access time delay

Software Approaches

* Read-Only Data are Cacheable
  - Private Cache is for Read-Only data
  - Shared Writable Data are not cacheable
  - Compiler tags data as cacheable and noncacheable
  - Degrade performance due to software overhead

* Centralized Global Table
  - Status of each memory block is maintained in CGT: RO(Read-Only); RW(Read and Write)
  - All caches can have copies of RO blocks
  - Only one cache can have a copy of RW block
  - Hardware Approaches

* Snoopy Cache Controller
  - Cache Controllers monitor all the bus requests from CPUs and IOPs
  - All caches attached to the bus monitor the write operations
  - When a word in a cache is written, memory is also updated (write through)
  - Local snoopy controllers in all other caches check their memory to determine if they have a copy of that word; If they have, that location is marked invalid(future reference to this location causes cache miss)