Chapter – 8
Multiprocessors

8.1 Characteristics of multiprocessors

- A multiprocessor system is an interconnection of two or more CPUs with memory and input-output equipment.
- The term "processor" in multiprocessor can mean either a central processing unit (CPU) or an input-output processor (IOP).
- Multiprocessors are classified as *multiple instruction stream, multiple data stream* (MIMD) systems
- The similarity and distinction between multiprocessor and multicomputer are
    - Similarity
        - Both support concurrent operations
    - Distinction
        - The network consists of several autonomous computers that may or may not communicate with each other.
        - A multiprocessor system is controlled by one operating system that provides interaction between processors and all the components of the system cooperate in the solution of a problem.
- Multiprocessing improves the reliability of the system.
- The benefit derived from a multiprocessor organization is an improved system performance.
    - Multiple independent jobs can be made to operate in parallel.
    - A single job can be partitioned into multiple parallel tasks.
- Multiprocessing can improve performance by decomposing a program into parallel executable tasks.
    - The user can explicitly declare that certain tasks of the program be executed in parallel.
        - This must be done prior to loading the program by specifying the parallel executable segments.
    - The other is to provide a compiler with multiprocessor software that can automatically detect parallelism in a user's program.
- Multiprocessor are classified by the way their memory is organized.
    - A multiprocessor system with *common shared memory* is classified as a *shared-memory* or *tightly coupled multiprocessor*.
        - Tolerate a *higher degree* of interaction between tasks.
    - Each processor element with its own *private local memory* is classified as a *distributed-memory* or *loosely coupled system.*
        - Are most efficient when the interaction between tasks is *minimal*

8.2 Interconnection Structures

- The components that form a multiprocessor system are CPUs, IOPs connected to input-output devices, and a memory unit.
- The interconnection between the components can have different physical configurations, depending on the number of transfer paths that are available
  - Between the processors and memory in a shared memory system
  - Among the processing elements in a loosely coupled system
- There are several physical forms available for establishing an interconnection network.
  - Time-shared common bus
  - Multiport memory
  - Crossbar switch
  - Multistage switching network
  - Hypercube system

Time Shared Common Bus

- A common-bus multiprocessor system consists of a number of processors connected through a common path to a memory unit.
- *Disadv.*:
  - 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
- A more economical implementation of a dual bus structure is depicted in Fig. below.
- Part of the local memory may be designed as a *cache memory* attached to the CPU.

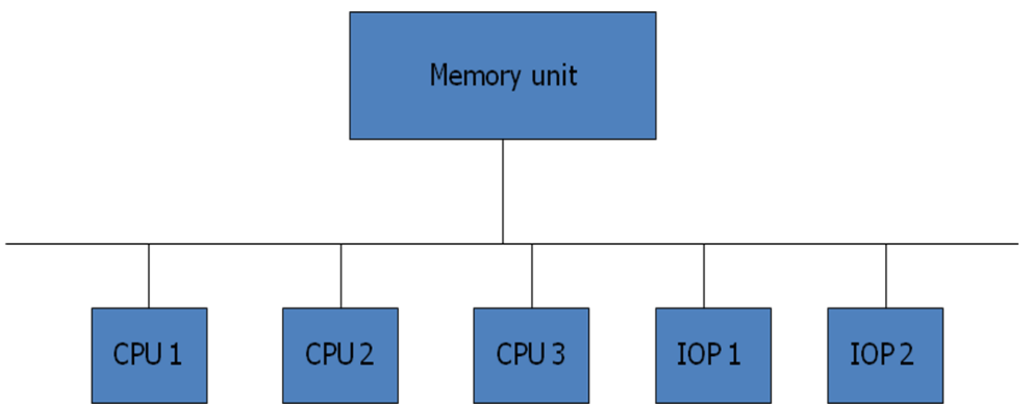


Fig: Time shared common bus organization

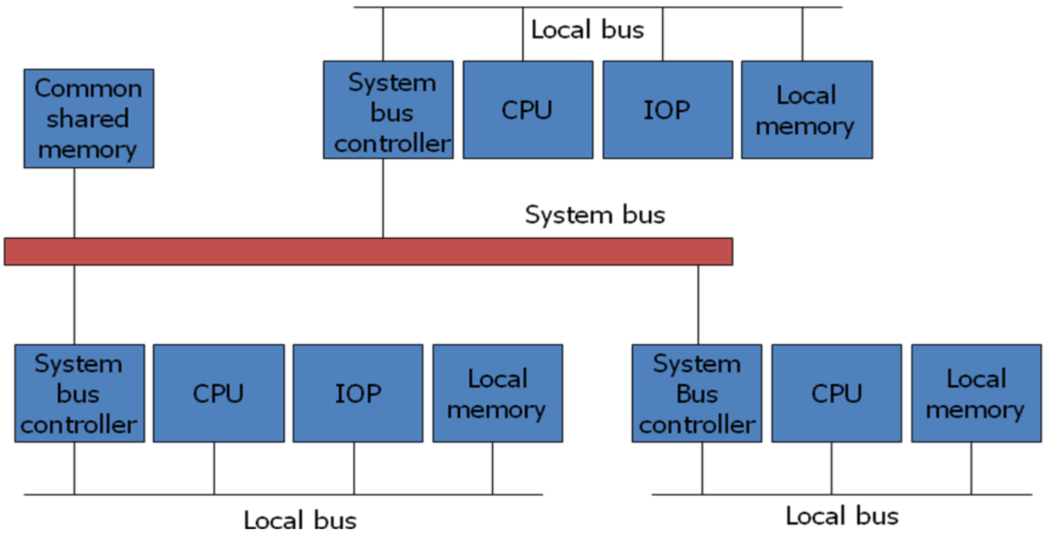


Fig: System bus structure for multiprocessorsa

Multiport Memory

- A multiport memory system employs separate buses between each memory module and each CPU.
- The module must have internal control logic to determine which port will have access to memory at any given time.
- Memory access conflicts are resolved by assigning fixed priorities to each memory port.
- *Adv.*:
    - The high transfer rate can be achieved because of the multiple paths.
- *Disadv.*:
    - It requires expensive memory control logic and a large number of cables and connections

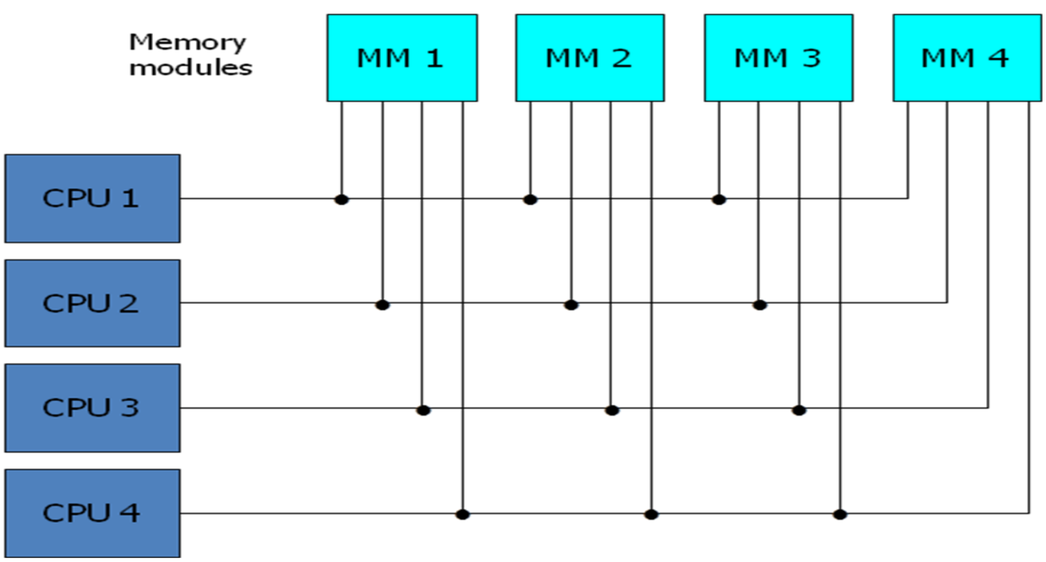


Fig: Multiport memory organization

Crossbar Switch

- Consists of a number of *crosspoints* that are placed at intersections between processor buses and memory module paths.
- The small square in each crosspoint is a *switch* that determines the path from a processor to a memory module.
- Adv.:
    - Supports simultaneous transfers from all memory modules
- Disadv.:
    - The hardware required to implement the switch can become quite large and complex.
- Below fig. shows the functional design of a crossbar switch connected to one memory module.

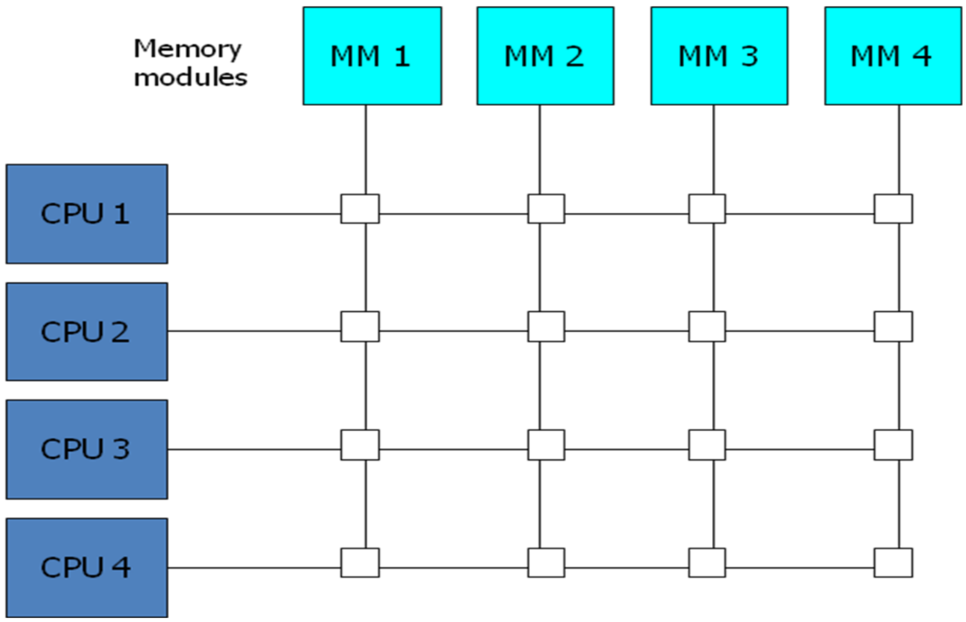


Fig: Crossbar switch

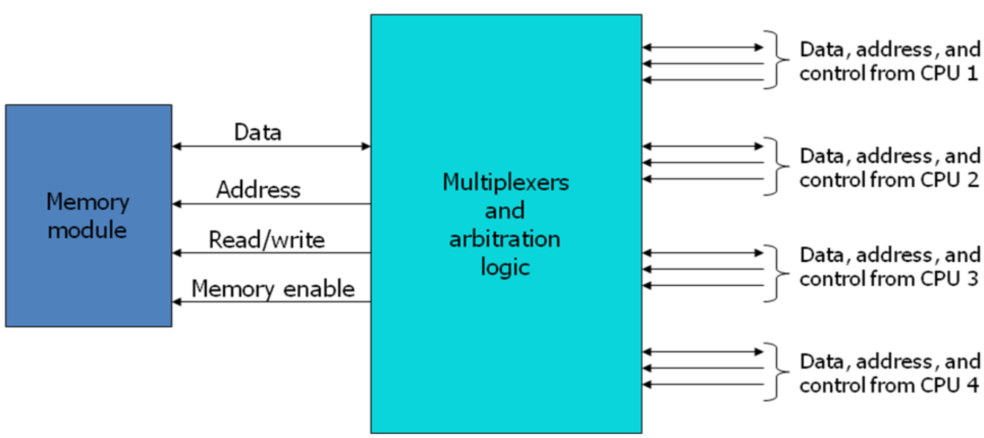
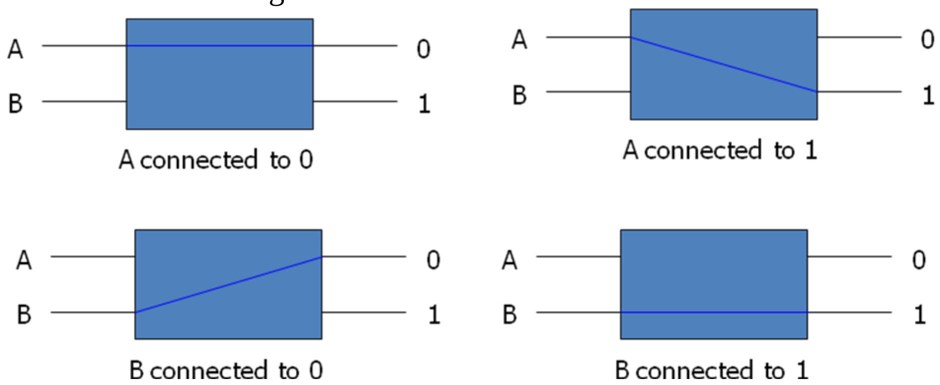


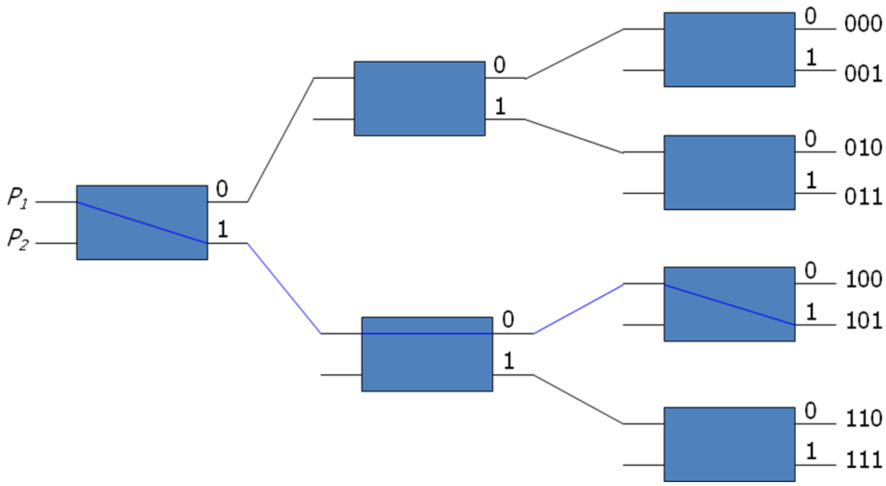
Fig: Block diagram of crossbar switch

Multistage Switching Network

- The basic component of a multistage network is a two-input, two-output interchange switch as shown in Fig. below.



- 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 P_1 → 000~011, then P_2 → 100~111



- One such topology is the omega switching network shown in Fig. below

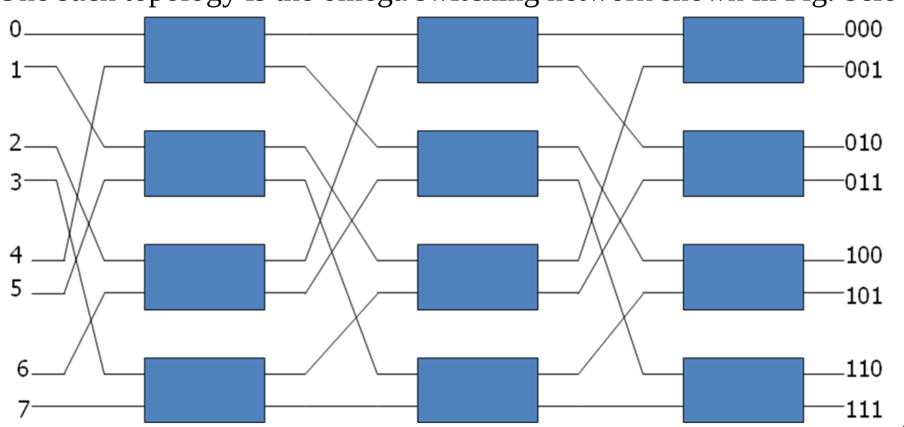


Fig: 8 x 8 Omega Switching Network

- 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 processing elements.

Hypercube System

- The hypercube or binary n-cube multiprocessor structure is a loosely coupled system composed of $N=2^n$ 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.

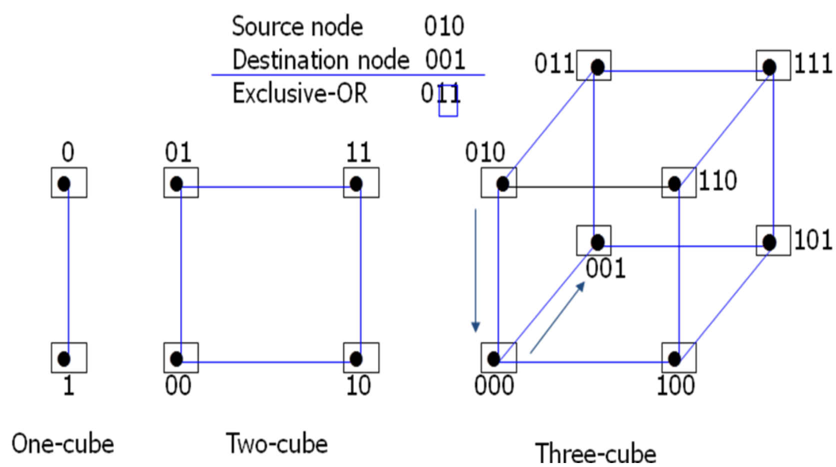


Fig: Hypercube structures for n=1,2,3

8.3 Inter processor Communication and Synchronization

- 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 ← M[*SEM*], M[*SEM*] ← 1
- Note that the lock signal must be active during the execution of the test-and-set instruction.