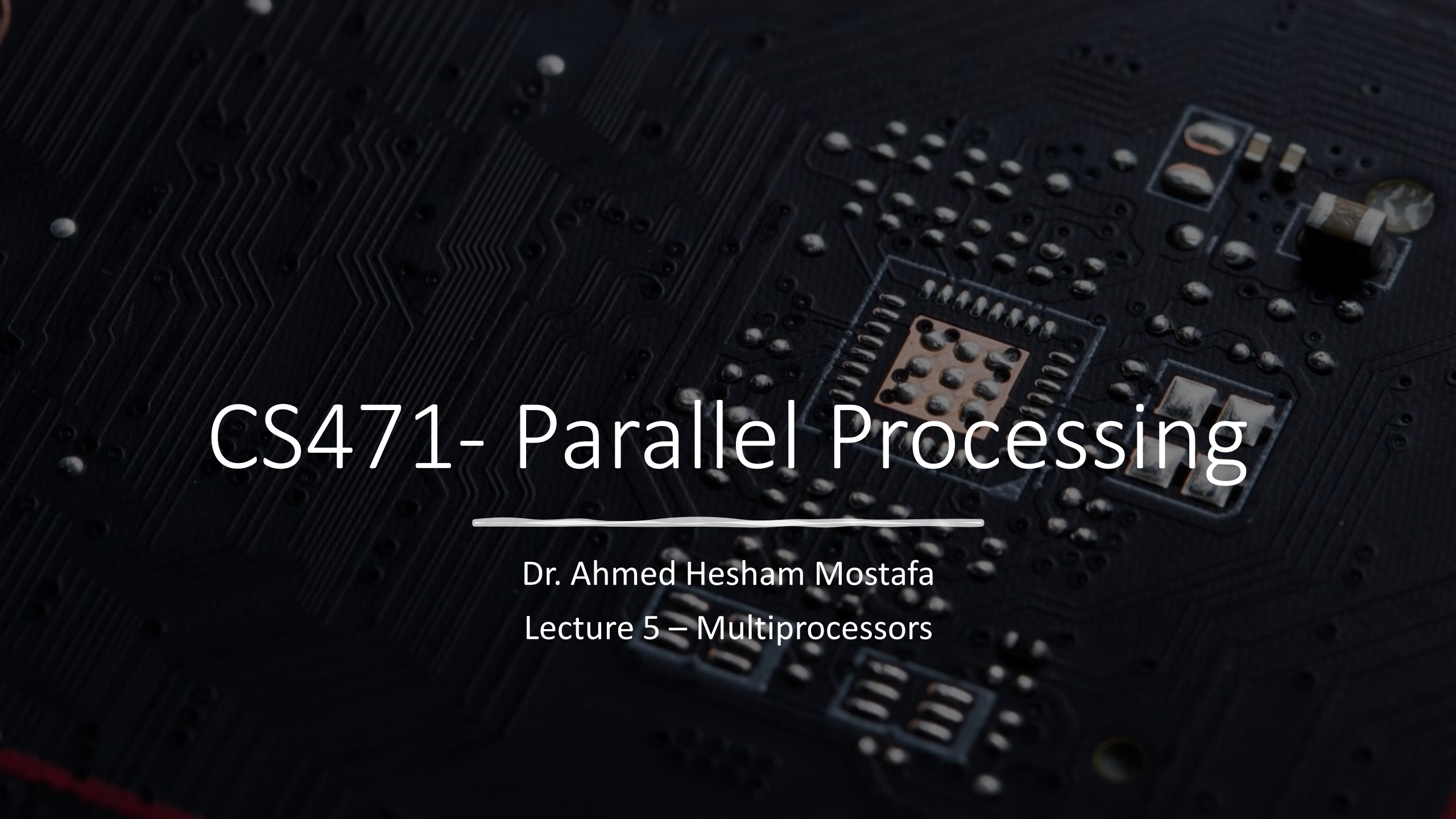


# CS471- Parallel Processing

Dr. Ahmed Hesham Mostafa

Lecture 5 – Multiprocessors

# Introduction

- Multiprocessors
  - A computer system in which two or more CPUs share full access to a common RAM
- Multicomputer
  - Tightly-coupled CPUs that do not share memory Also known as cluster computers or clusters of workstations (COWs)

# Introduction

- There are some similarities between Multiprocessor and Multicomputer systems since both support concurrent operations.
- However, there exists an important distinction between a system with multiple computers and a system with multiple processors.
- **Computers** are interconnected with each other by means of communication lines to form a computer network.
- The network consists of several 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.

# Introduction

- Multiprocessing improves the reliability of the system so that a failure or error in one part has a 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 processor.
- The system can continue to function correctly with perhaps some loss in efficiency.
- The benefit derived from a multiprocessor organization is improved system performance.
- The system derives its high performance from the fact that computations can proceed in parallel in one of two ways.
  - Multiple independent jobs can be made to operate in parallel.
  - A single job can be partitioned into multiple parallel tasks.

# Processor Coupling

- Multiprocessors are classified by the way their memory is organized.
  - Tightly coupled multiprocessor systems
  - Loosely-coupled multiprocessor systems

# Tightly coupled multiprocessor systems

- A multiprocessor system with common shared memory is classified as a shared memory or tightly coupled multiprocessor.
- In fact, most commercial tightly coupled multiprocessors provide a cache memory with each CPU.
- in addition, there is a global common memory that all CPUs can access.
- Information can therefore be shared among the CPUs by placing it in the common global memory
- These CPUs may have access to a central shared memory (Symmetric Multiprocessing, or SMP), or may participate in a memory hierarchy with both local and shared memory (Non-Uniform Memory Access, or NUMA).

# Loosely-coupled multiprocessor systems

- Each processor element in a loosely coupled system has its own private local memory.
- Often referred to as clusters
- Based on multiple standalone singles or dual processor computers interconnected via a high-speed communication system, such as Gigabit ethernet.
- The processors are tied together by a switching scheme designed to route information from one processor to another through a message passing scheme.

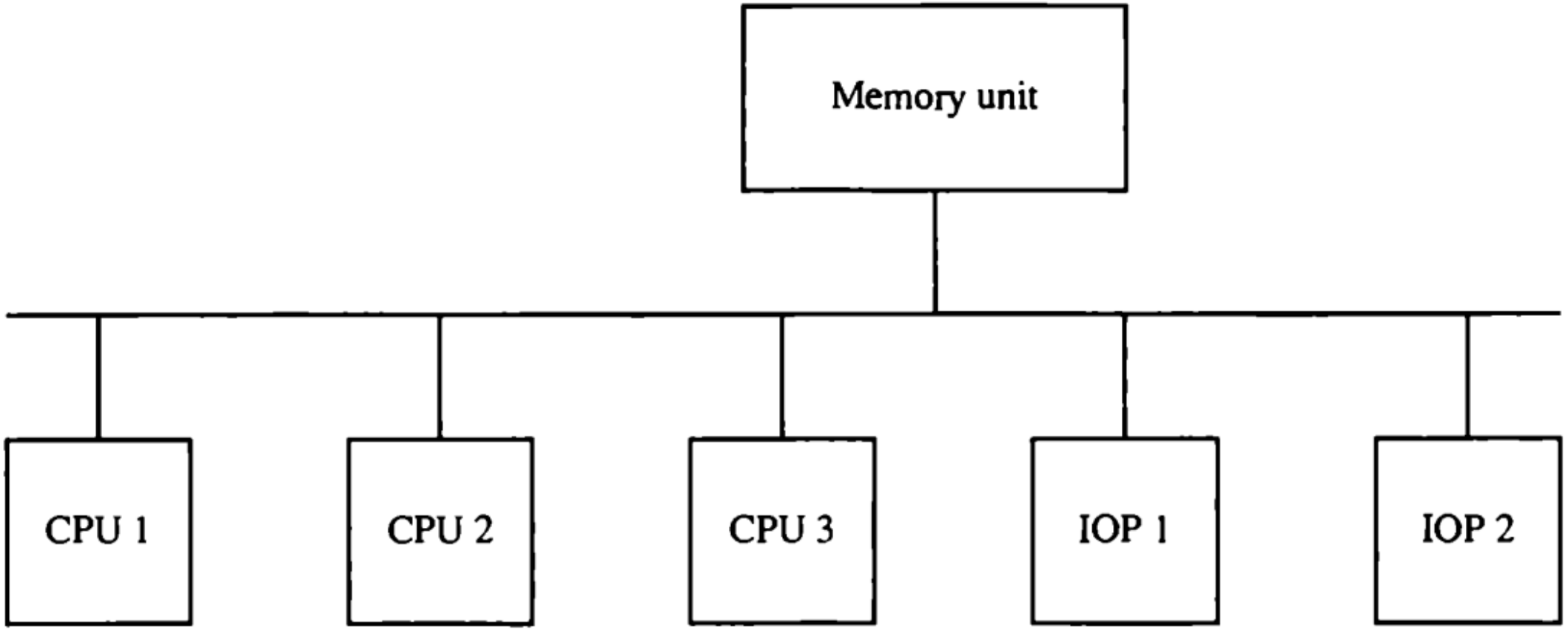
# Loosely-coupled multiprocessor systems

- The processors relay program and data to other processors in packets.
- A packet consists of an address, the data content, and some error detection code.
- The packets are addressed to a specific processor or taken by the first available processor, depending on the communication system used.
- Loosely coupled systems are most efficient when the interaction between tasks is minimal
- whereas tightly coupled systems can tolerate a higher degree of interaction between tasks.

# Interconnection Structures

- 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 or among the processing elements in a loosely coupled system.
- There are several physical forms available for establishing an interconnection network, Some of these schemes are presented in this section:
  - **Time-shared common bus**
  - **Multiport memory**
  - **Crossbar switch**
  - **Multistage switching network**
  - **Hypercube system**

# Time-Shared Common Bus



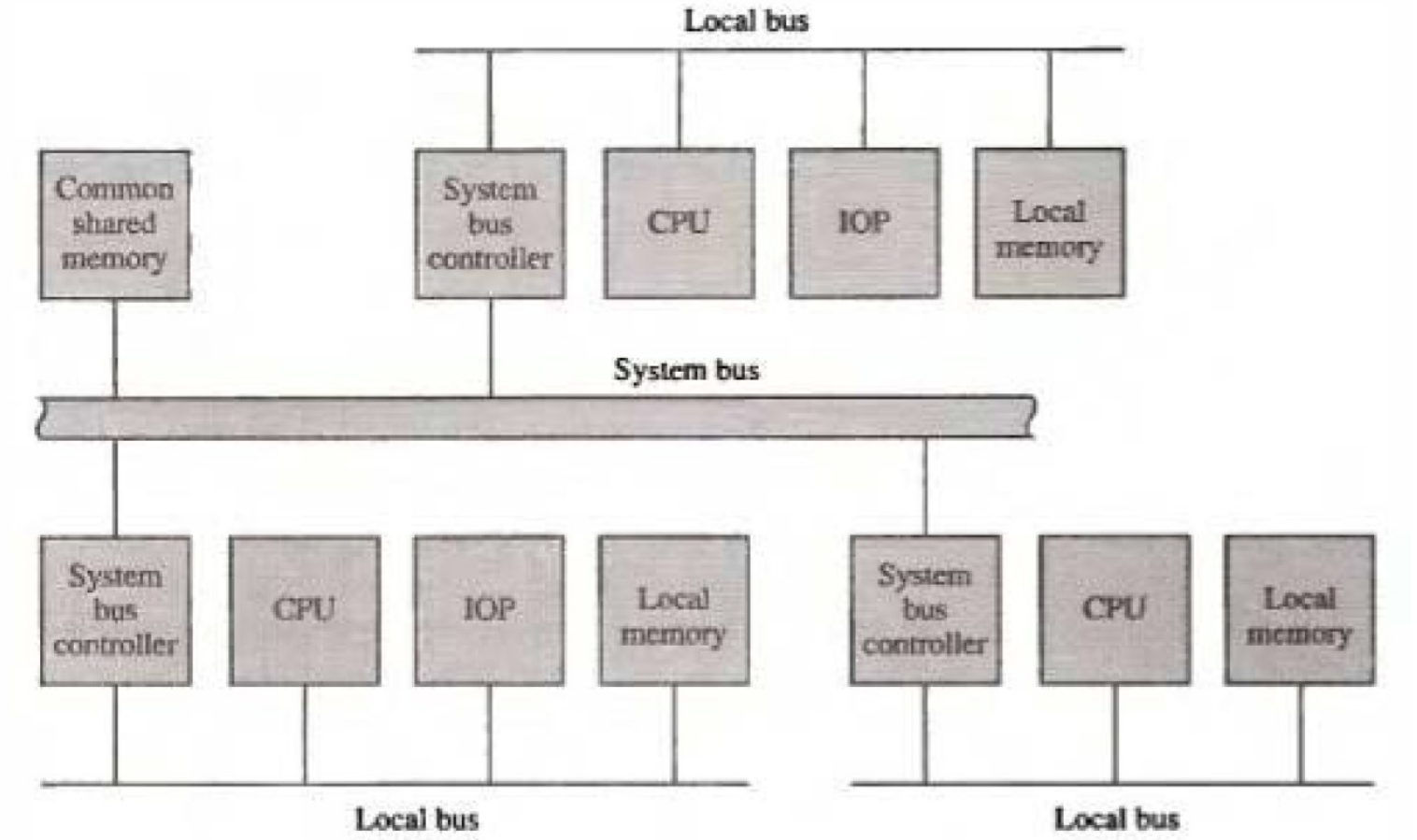
# Time-Shared Common Bus

- A common-bus multiprocessor system consists of a number of processors connected through a common path to a memory unit.
- Only one processor can communicate with the memory or another processor at any given time.
- Transfer operations are conducted by the processor that is in control of the bus at the time.
- Any other processor wishing to initiate a transfer must first determine the availability status of the bus, and only after the bus becomes available can the processor address the destination unit to initiate the transfer.
- A command is issued to inform the destination unit what operation is to be performed

# Time-Shared Common Bus

- The receiving unit recognizes its address in the bus and responds to the control signals from the sender, after which the transfer is initiated.
- The system may exhibit transfer conflicts since one common bus is shared by all processors.
- These conflicts must be resolved by incorporating a bus controller that establishes priorities among the requesting units.
- A more economical implementation of a dual bus structure is depicted in Fig. 13-2.

# Time-Shared Common Bus

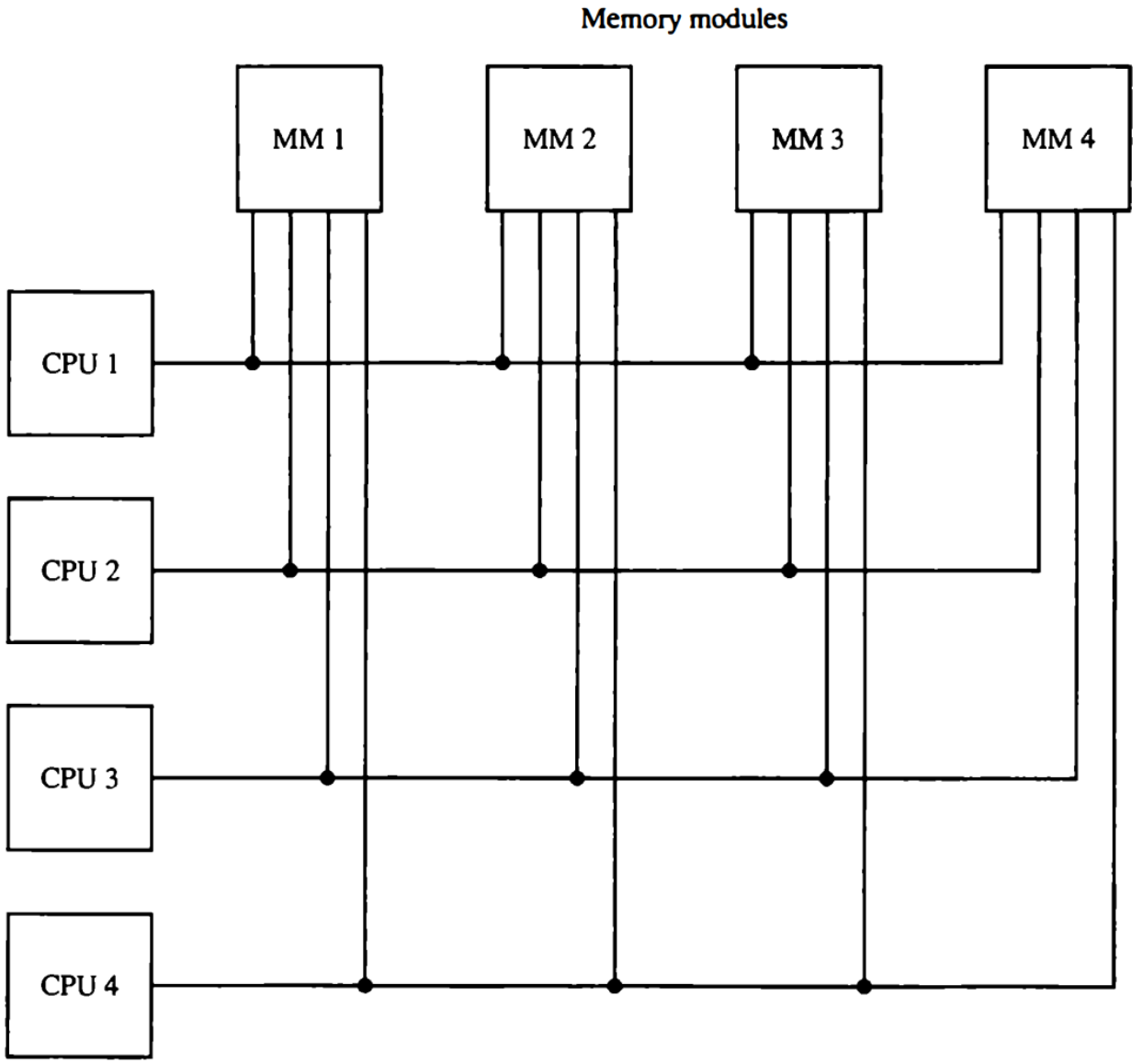



**The other processors are kept busy communicating with their local memory and I/O devices.**

# Time-Shared Common Bus

- Here we have a number of local buses each connected to its own local memory and to one or more processors.
- Each local bus may be connected to a CPU, an lOP, or any combination 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.
- Only one processor can communicate with the shared memory and other common resources through the system bus at any given time.

# Multiport Memory



# Multiport Memory

- A multiport memory system employs separate buses between each memory module and each CPU
- There are four CPUs and four memory modules (MMs).
- Each processor bus is connected to each memory module.
- A processor bus consists of the address, data, and control lines required to communicate with memory.

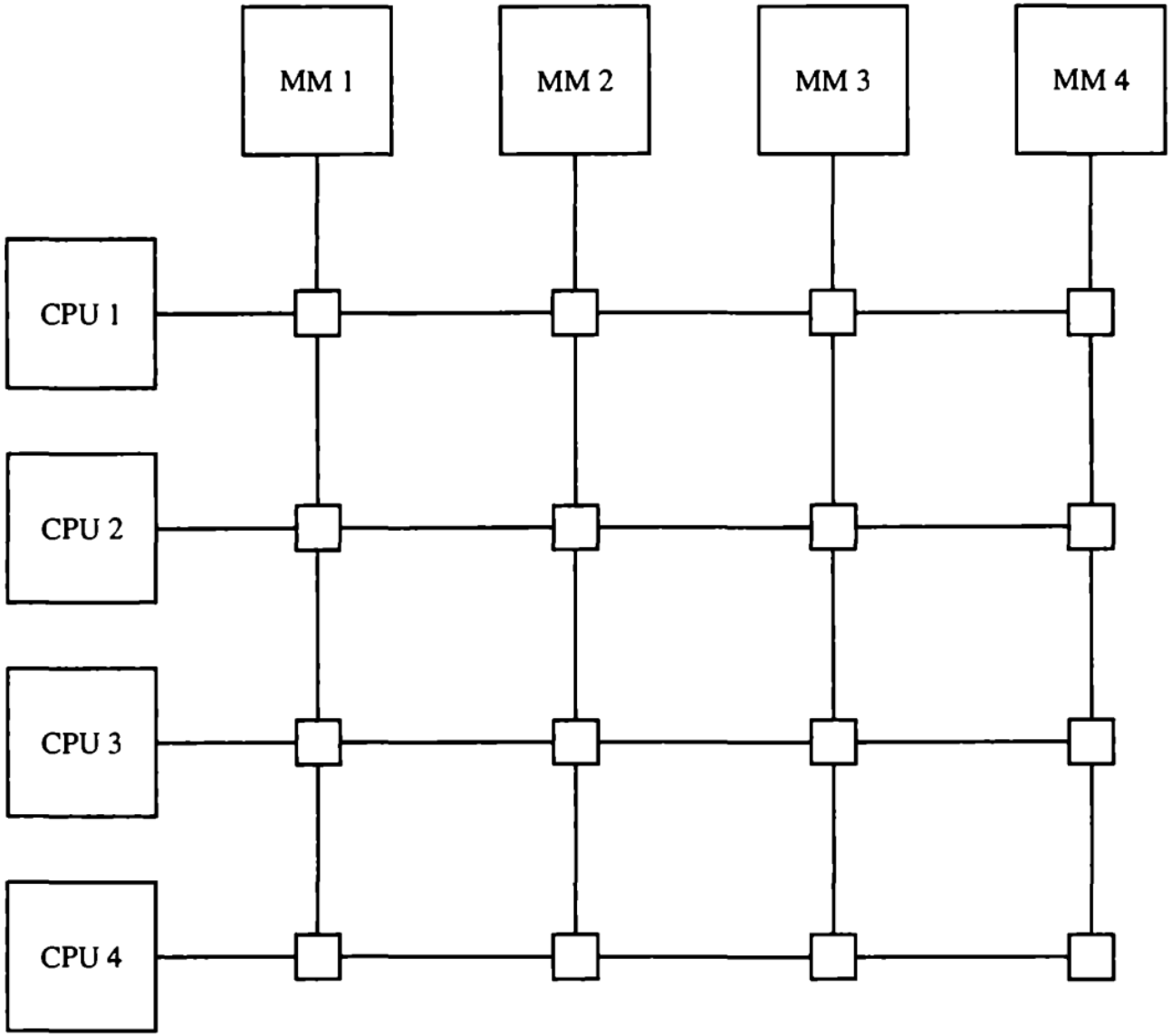
# Multiport Memory

- The memory module is said to have four ports and each port accommodates one of the buses.
- 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.
- The priority for memory access associated with each processor may be established by the physical port position that its bus occupies in each module.
- Thus CPU 1 will have priority over CPU 2, CPU 2 will have priority over CPU 3, and CPU 4 will have the lowest priority.

# Multiport Memory

- The advantage of the multiport memory organization is the high transfer rate that can be achieved because of the multiple paths between processors and memory.
- The disadvantage is that it requires expensive memory control logic and a large number of cables and connectors.
- As a consequence, this interconnection structure is usually appropriate for systems with a small number of processors.

# Crossbar Switch



# Crossbar Switch

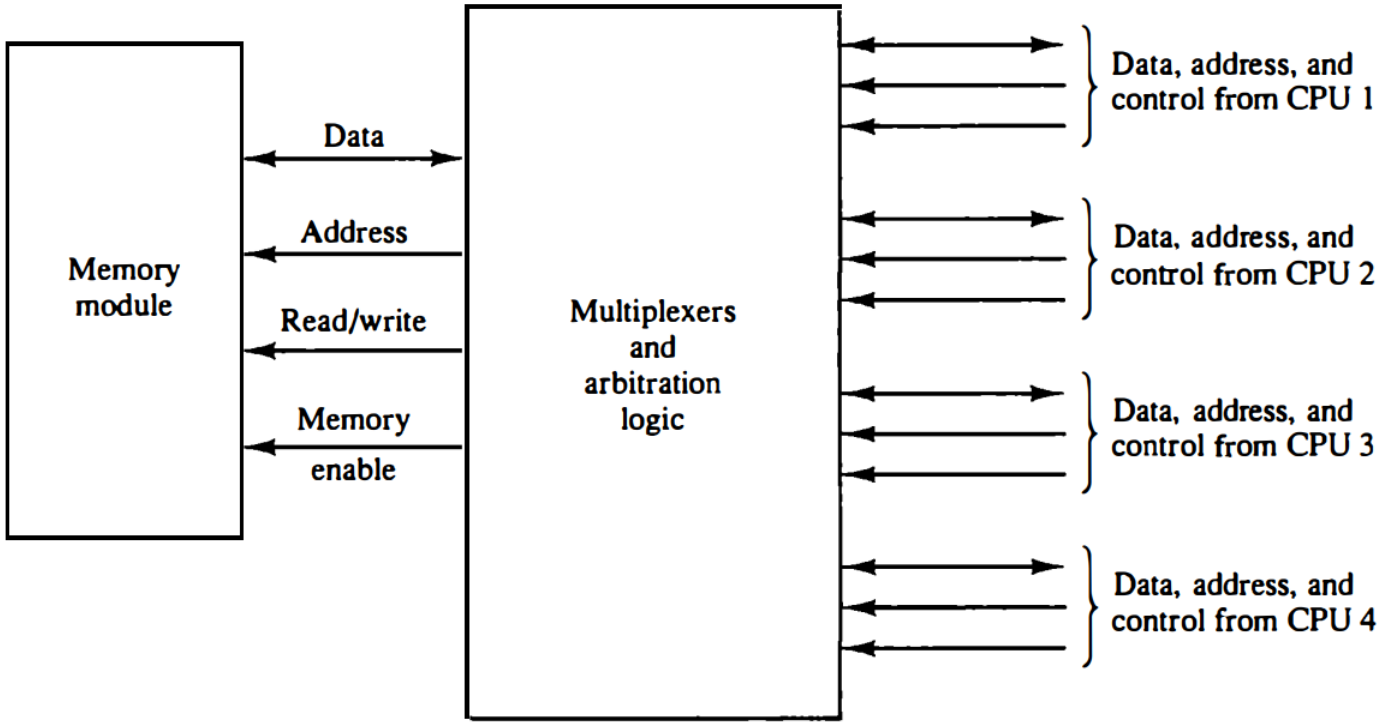
- The crossbar switch organization consists of a number of cross points that are placed at intersections between processor buses and memory module paths.
- Figure 13-4 shows a crossbar switch interconnection between four CPUs and four memory modules.
- The small square in each cross point is a switch that determines the path from a processor to a memory module.
- Each switch point has control logic to set up the transfer path between a processor and memory.
- It examines the address that is placed in the bus to determine whether its particular module is being addressed.

# Crossbar Switch

- It also resolves multiple requests for access to the same memory module on a predetermined priority basis.
- Figure 13-5 shows the functional design of a crossbar switch connected to one memory module.
- The circuit consists of multiplexers that select the data, address, and control from one CPU for communication with the memory module.
- Priority levels are established by the arbitration logic to select one CPU when two or more CPUs attempt to access the same memory.

# Crossbar Switch

**Figure 13-5** Block diagram of crossbar switch.

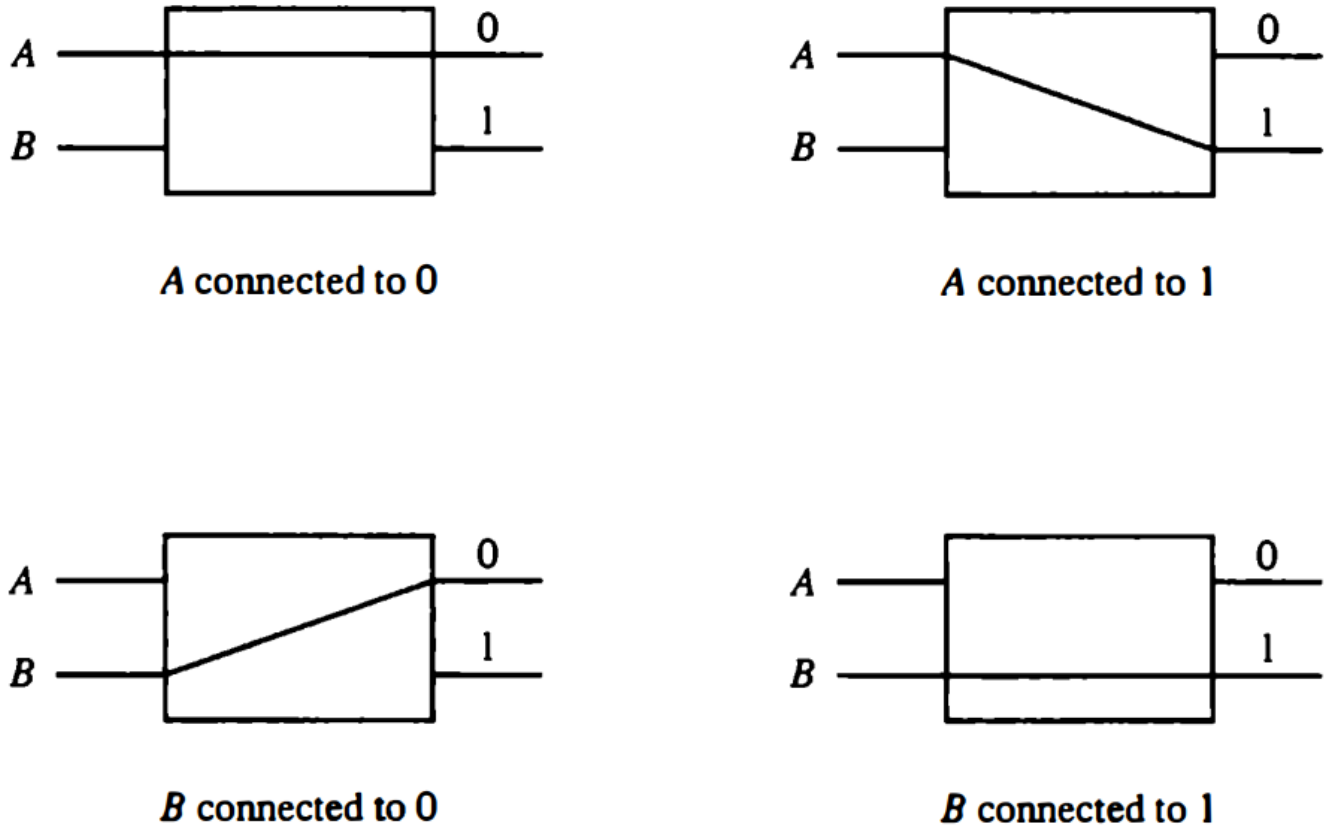


# Crossbar Switch

- There is a separate path associated with each module.
- However, the hardware required to implement the switch can become quite large and complex.

# Multistage switching Network

**Figure 13-6** Operation of a 2 × 2 interchange switch.



# Multistage switching Network

- The basic component of a multistage network is a two-input, two-output interchange switch.
- As shown in Fig. 13-6, the 2 x 2 switch has two inputs, labeled A and B, and two outputs labeled 0 and 1.
- There are control signals (not shown) associated with the switch that establish the interconnection between the input and output terminals.
- The switch has the capability of connecting input A to either of the outputs.
- Terminal B of the switch; behaves in a similar fashion. The switch also has the capability to arbitrate between conflicting requests.
- If inputs A and B both request the same output terminal, only one of them will be connected; the other will be blocked.
- Using the 2 x 2 switch as a building block, it is possible to build a multistage network to control the communication between a number of sources and destinations

# Multistage switching Network

- To see how this is done, consider the binary tree shown in Fig. 13-7.
- The two processors P1 and P2 are connected through switches to eight memory modules marked in binary from 000 through 111.
- The path from a source to a destination is determined from the binary bits of the destination number.
- The first bit of the destination number determines the switch output in the first level.
- The second bit specifies the output of the switch in the second level, and the third bit specifies the output of the switch in the third level.

# Multistage switching Network

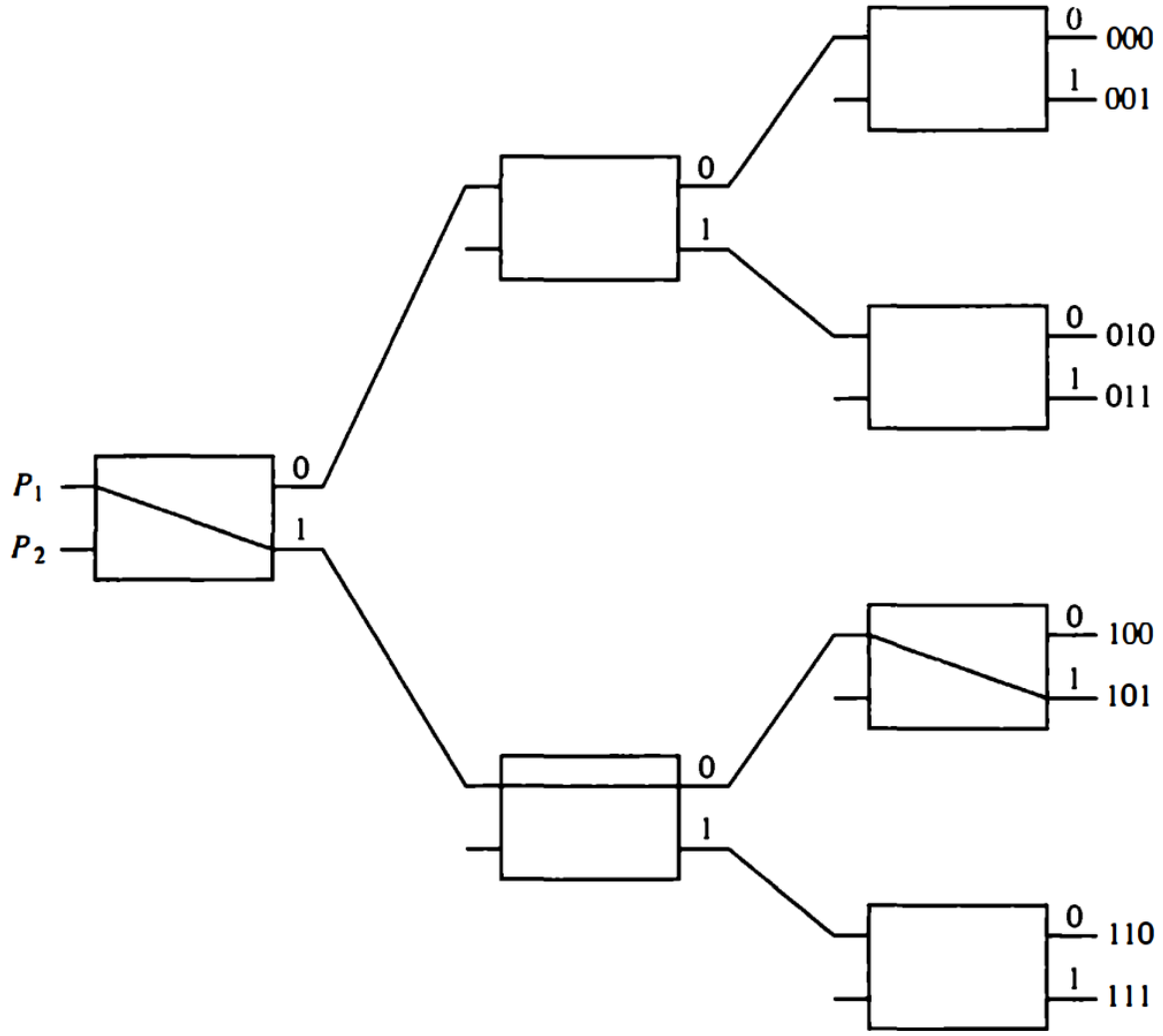



**Figure 13-7** Binary tree with 2 × 2 switches.

# Multistage switching Network

- For example, to connect P1 to memory 101, it is necessary to form a path from P to output 1 in the first-level switch, output 0 in the second-level switch, and output 1 in the third-level switch.
- Either P1 or P2 can be connected to any one of the eight memories, Certain request patterns, however, cannot be satisfied at the same time.
- For example, if P1 is connected to one of the destinations 000 to 011,P2 can be connected to only one of the destinations 100 to 111.

# Multistage switching Network

- Many different topologies have been proposed for multistage switching networks to control processor—memory communication in a tightly coupled multiprocessor system or to control the communication between the processing elements in a loosely coupled
- One such topology is the omega switching network shown in Fig. 13- 8.
- In this configuration, there is exactly one path from each source to any destination.
- Some request patterns, however, cannot be connected at the same time.
- For example, any two sources cannot he connected simultaneously to destinations 000 to 111.

# Multistage switching Network

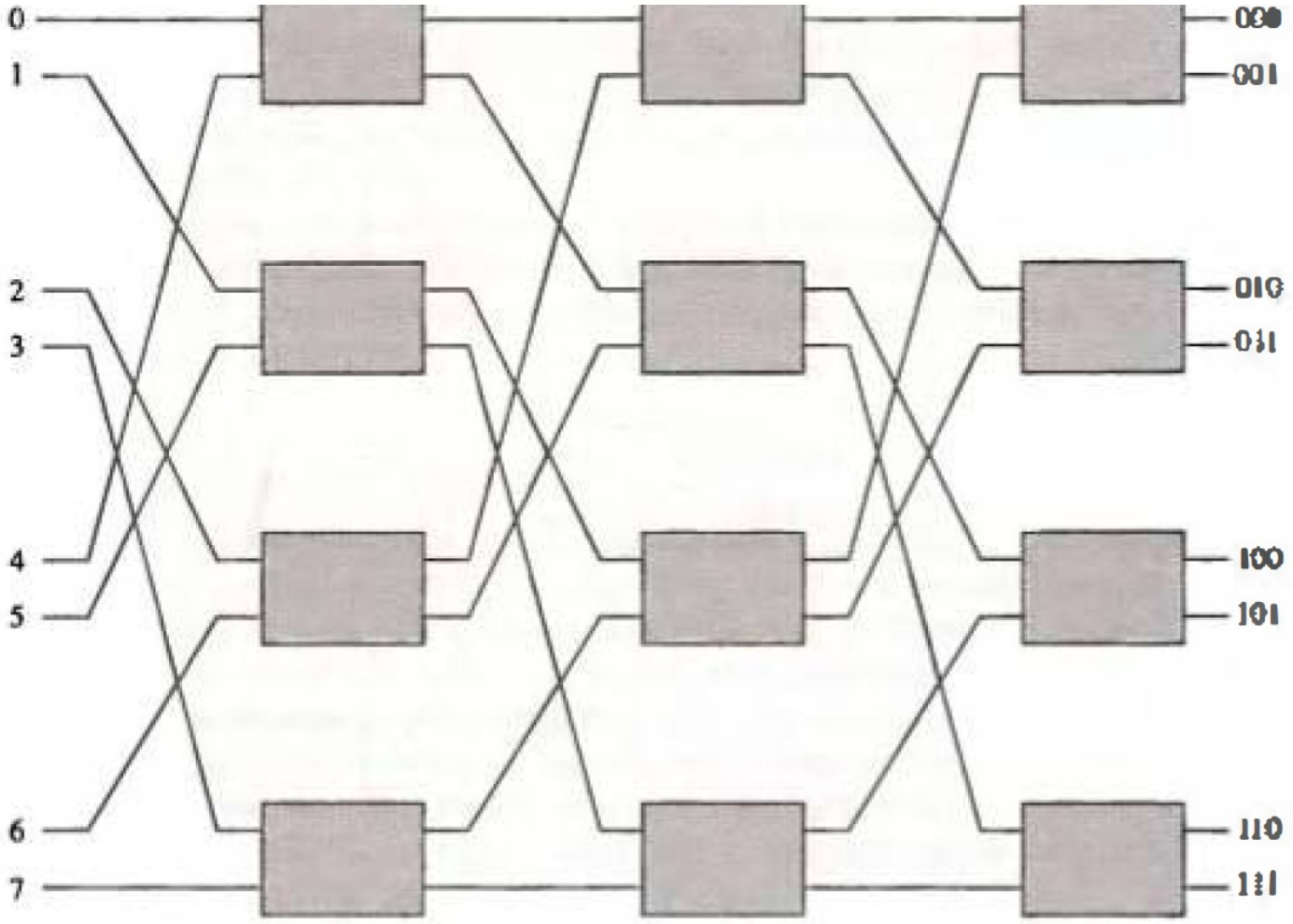



Figure 13-8 8 × 8 omega switching network.

# Multistage switching Network

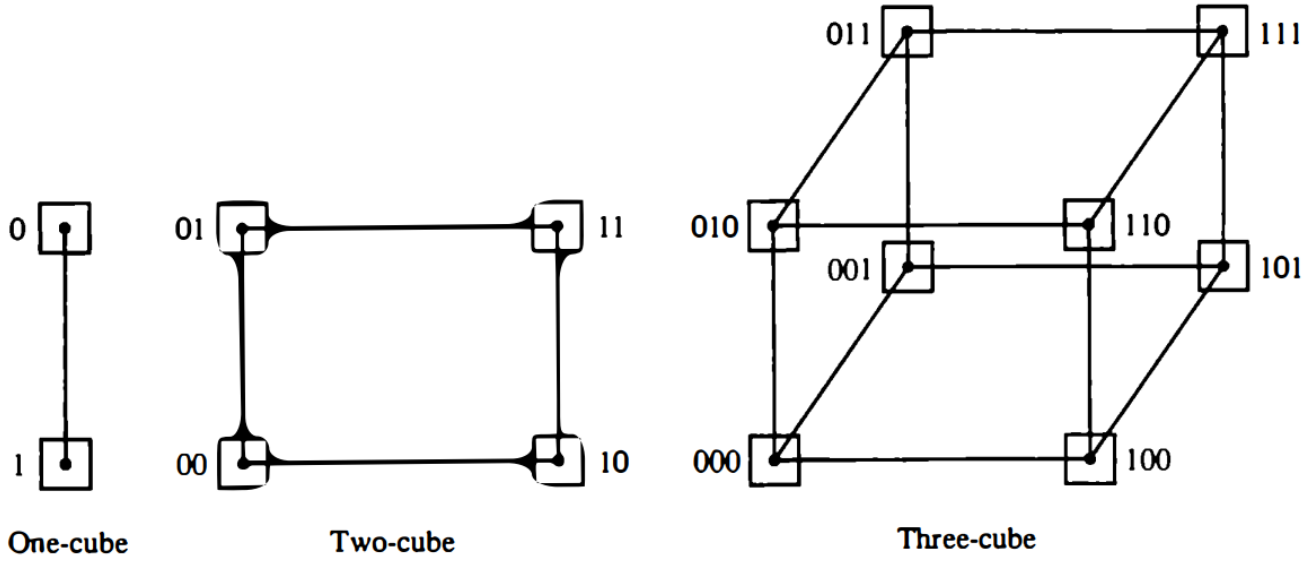
- A particular request is initiated in the switching network by the source, which sends a 3-bit pattern representing the destination number.
- As the binary pattern moves through the network, each level examines a different bit to determine the 2 x 2 switch setting.
- Level 1 inspects the most significant bit, level 2 inspects the middle bit, and level 3 inspects the least significant bit.
- When the request arrives on either input of the 2 x 2 switch, it is routed a the upper output if the specified bit is 0 or to the lower output if the bit is 1.

# Multistage switching Network

- In a tightly coupled multiprocessor system, the source is a processor, and the destination is a memory module.
- The first pass through the network sets up the path.
- Succeeding passes are used to transfer the address into memory and then transfer the data in either direction, depending on whether the request is a read or a write.
- In a loosely coupled multiprocessor system, both the source and destination are processing elements. After the path is established, the source processor transfers a message to the destination processor.

# Hypercube Interconnection

**Figure 13-9** Hypercube structures for $n = 1,2,3$.



# Hypercube Interconnection

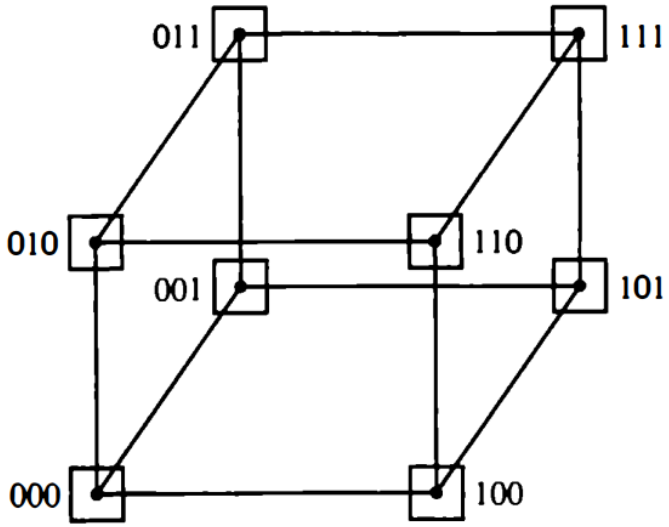
- The hypercube or binary n-cube multiprocessor structure is a loosely coupled system composed of N = 2^ n processors interconnected in an n-dimensional binary cube.
- Each processor forms a node of the cube.
- Although it is customary to refer to each node as having a processor, in effect it contains not only a CPU but also local memory and an I/O interface.
- Each processor has direct communication paths -to n other neighbor processors.
- These paths correspond to the edges of the cube.
- There are 2^ n distinct n-bit binary addresses that can be assigned to the processors.
- Each processor address differs from that of each of its n neighbors by exactly one-bit position.

# Hypercube Interconnection

- Figure 13- 9 shows the hypercube structure for n = 1, 2, and 3.
- A one-cube structure has n = 1 and 2^ n = 2.
- It contains two processors interconnected by a single path.
- A two-cube structure has n= 2 and 2 ^ 2 = 4.
- lt contains four nodes interconnected as a square.
- A three-cube structure has eight nodes interconnected as a cube.
- An n-cube structure has 2^ n nodes with a processor residing in each node.
- Each node is assigned a binary address in such a way that the addresses of two neighbors differ in exactly one-bit position.
- For example, the three neighbors of the node with address 100 in a three-cube structure are 000,110, and 101.
- Each of these binary numbers differs from address 100 by one-bit value.

# Hypercube Interconnection

- Routing messages through an n-cube structure may take from one to n links from a source node to a destination node.
- For example, in a three-cube structure, node 000 can communicate directly with node 001.
- It must cross at least two links to communicate with 011 (from 000 to 001 to 011 or from 000 to 010 to 011).
- It is necessary to go through at least three links to communicate from node 000 to node 111.
- A routing procedure can be developed by computing the exclusive OR of the source node address with the destination node address.



# Hypercube Interconnection

- The resulting binary value will have 1 bit corresponding to the axes on which the two nodes differ.
- EX. Assume S= 2(010) ,and D= 1(001)
- These paths have to → S2S1S0 XOR D2D1D0
- 010 XOR 001 = 011
- The change occurs at the position of the bits number 0 and number 1
- So, the path from 2 → 1 will follow the route by traversing the following dimensions (0 and 1)
- 2(010) → 0(000) → 1(001)

# Example :

**Assume that S =S5S4 . . . S1S0 to be the source node address, and that D = D5D4 . . . D1D0 is the destination node address in a six-dimensional hypercube message passing system. Let R = S XOR D be the exclusive OR function executed bitwise for each node in the path. Consider the case whereby S = 10(001010) and D = 39(100111)**

**= 5 4 3 2 1 0**

**These paths have to → (001010) XOR (100111) = 1 0 1 1 0 1**

**So, the path from 10 → 39 will following dimensions(0, 2, 3, and 5)**

**Let us assume that the message will follow the route by traversing the following dimensions 5, 3, 2, and 0. Then the route is totally determined as:**

**10 (001010) → 42(101010) → 34(100010) → 38(100110) → 39 (100111).**

**5 3 2 0**

# References

# Thanks