Unit-6
Multiprocessors

By- Yadvir Kaur

Multiprocessors

- Characteristics of Multiprocessors
- Interconnection Structures
- Interprocessor Arbitration
- Interprocessor Communication & Synchronization
- Cache Coherence

Characteristics of Multiprocessors

- **A multiprocessor system is an interconnection of two or more CPUs with memory and input-output equipment.**
- Multiprocessors are classified as multiple instruction stream, multiple data stream (MIMD) systems.
- Very-large-scale integrated (VLSI) circuit technology has reduced the cost of computer components to such a low level that the concept of applying multiple processors to meet system performance requirements has become an attractive design possibility.
- **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 benefit derived from a multiprocessor organization is an **improved system performance. The system derives its high performance from the fact that computations can proceed in parallel in one of two ways.**

    1. Multiple independent jobs can be made to operate in parallel.

    2. A single job can be partitioned into multiple parallel tasks.

Characteristics of Multiprocessors

Multiprocessors are classified by the way their memory is organized:

1) multiprocessor system with common shared memory is classified as a **shared memory or tightly coupled multiprocessor.**

» Local memory + Shared memory (Information can therefore be shared among the CPUs by placing it in the common global memory.)

» higher degree of interaction between tasks

2) Distribute memory or Loosely-coupled system

» Local memory + message passing scheme (The processors are tied together by a switching scheme designed to route information from one processor to another through a mes sage-passing scheme.)

» most efficient when the interaction between tasks is minimal

Interconnection Structures

The components that form a multiprocessor system are CPUs, IOPs connected to input-output devices, and a memory unit that may be partitioned into a number of separate modules. **There are several physical forms available for establishing an interconnection network. Some of these schemes are presented in this section:**

1. Time-shared common bus

2. Multiport memory

3. Crossbar switch

4. Multistage switching network

5. 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.

- **Only one processor can communicate with the memory or another processor at any given time.**

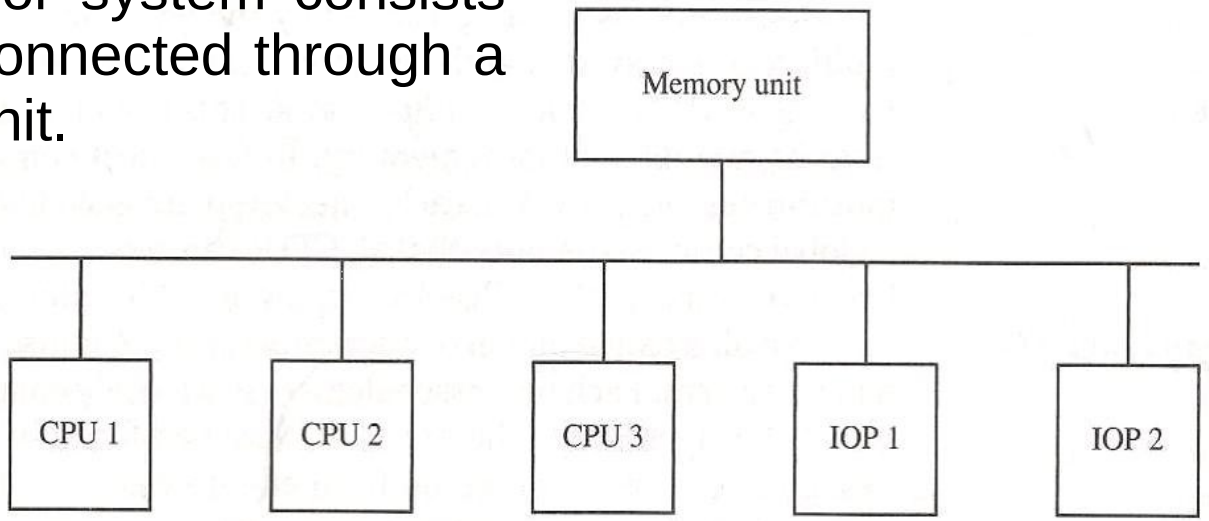
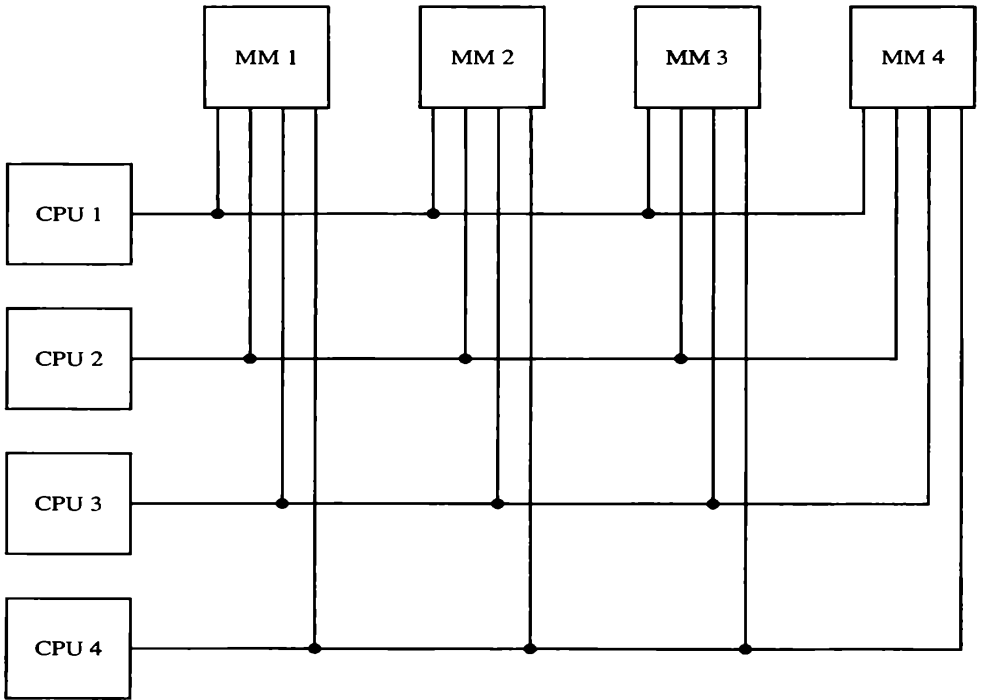


Figure 13-1 Time-shared common bus organization.

Multi-port memory

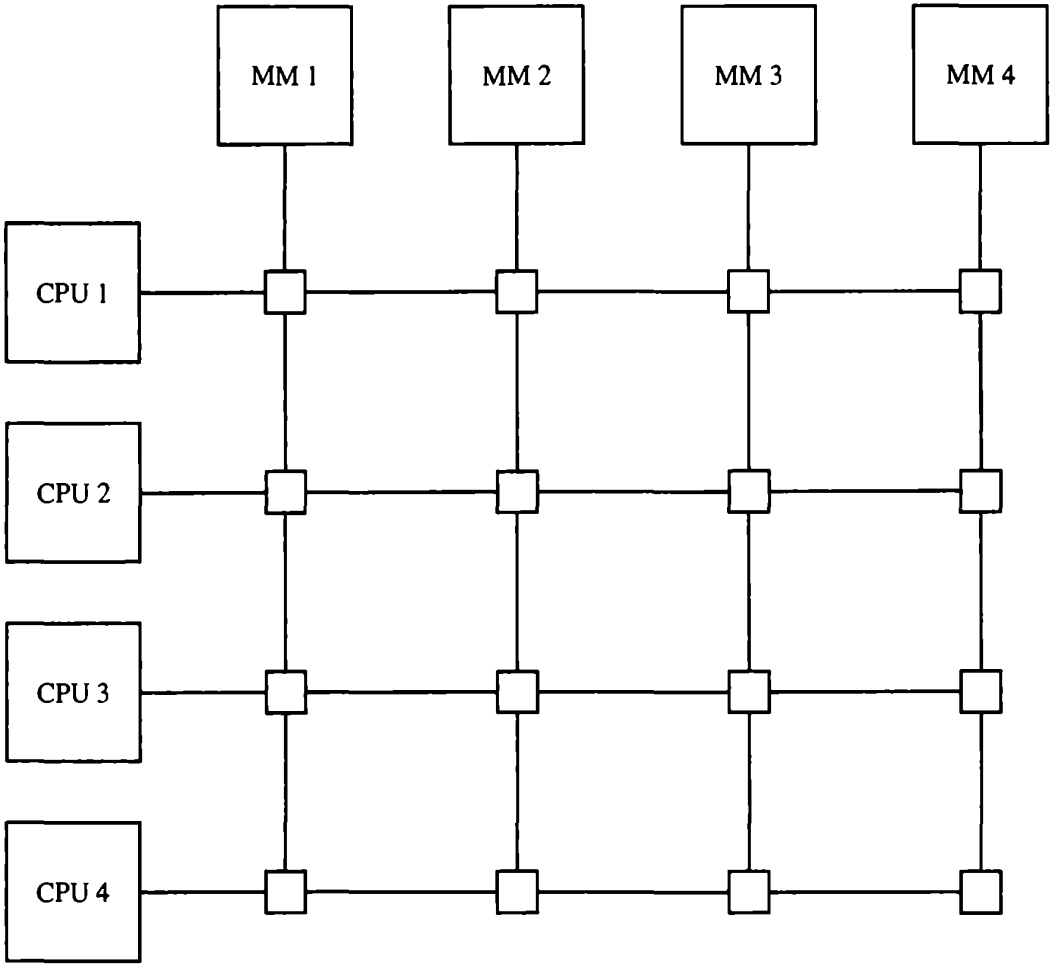
- multiple paths between processors and memory
- Advantage : **high transfer rate can be achieved**
- Disadvantage : **expensive memory control logic / large number of cables & connectors**



Crossbar Switch

- The crossbar switch organization **consists of a number of crosspoints(switches)** that are placed at intersections between processor buses and memory module paths, **that determines the path from a processor to a memory module.**

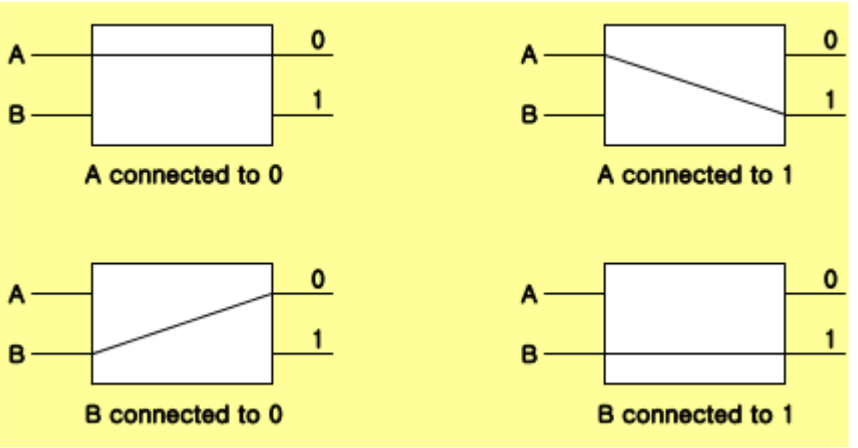
- It also resolves multiple requests for access to the same memory module on a predetermined priority basis.
- A crossbar switch organization supports simultaneous transfers from memory modules because there is a separate path associated with each module.
- However, **the hardware required to implement the switch can become quite large and complex.**



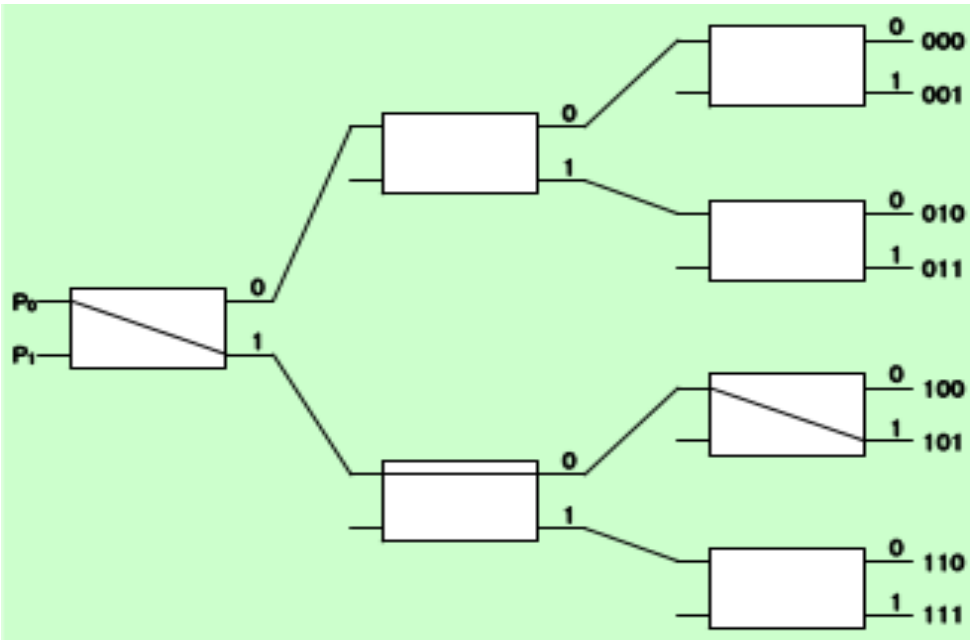
Multistage Switching Network

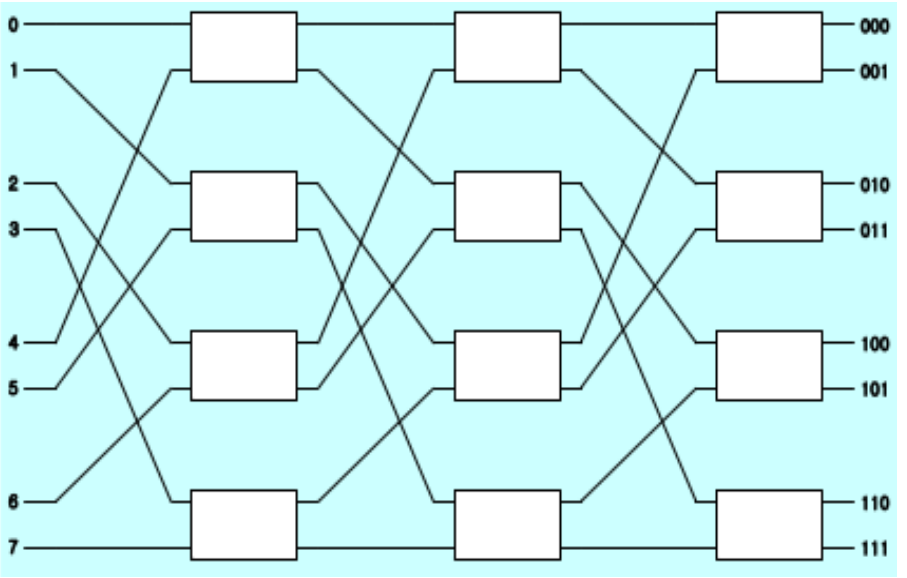
Control the communication between a number of sources and destinations.

The basic component of a multistage network is a two-input, two-output interchange switch.



In the 2nd Figure, 2 Processor (P1 and P2) are connected through switches to 8 memory modules (000 - 111). 3rd Figure is showing omega network.





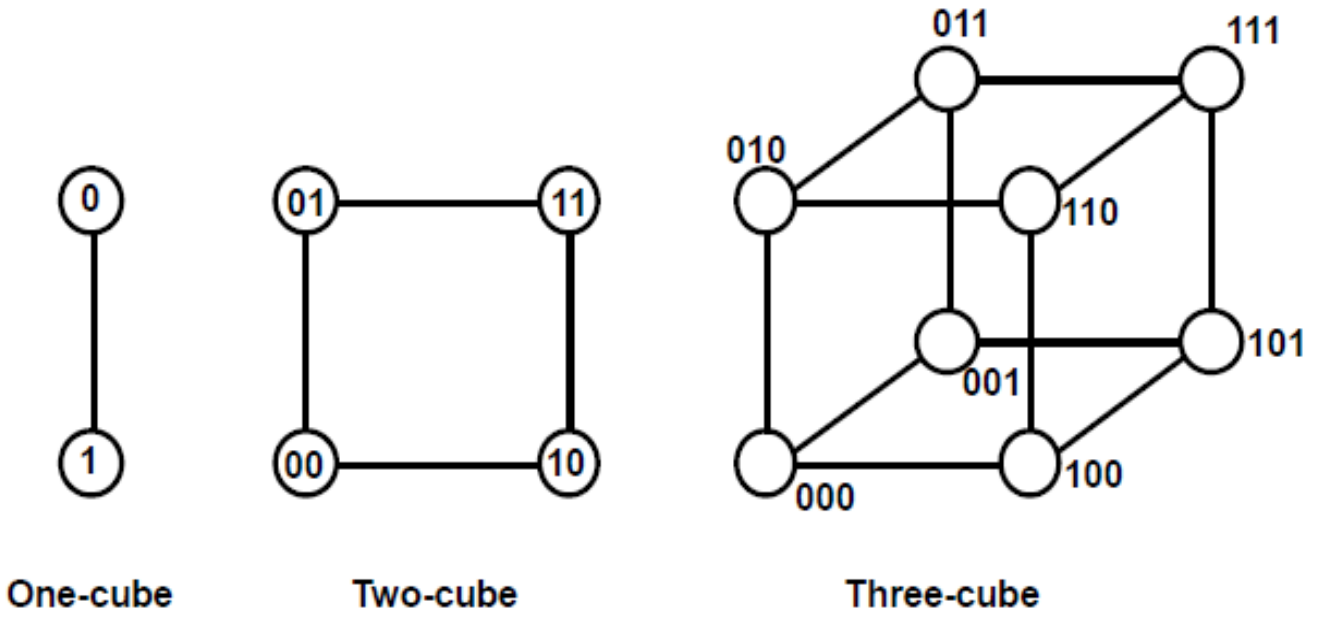
Hypercube Interconnection

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 has direct communication paths to n other neighbor processors.

Fig. below shows the hypercube structure for n=1, 2, and 3.



Interprocessor Arbitration

The processors in a shared memory multiprocessor system request access to common memory or other common resources through the system bus. If no other processor is currently utilizing the bus, the requesting processor may be granted access immediately. However, the requesting processor must wait if another processor is currently utilizing the system bus. Furthermore, other processors may request the system bus at the same time. **Arbitration must then be performed to resolve this multiple contention for the shared resources. The arbitration logic would be part of the system bus controller placed between the local bus and the system bus.**

Data transfers over the system bus may be synchronous or asynchronous.

- **In a synchronous bus**, each data item is transferred during a time slice known in advance to both source and destination units. Synchronization is achieved by driving both units from a **common clock source**.
- **In an asynchronous bus**, each data item being transferred is accompanied by **handshaking control signals** to indicate when the data are transferred from the source and received by the destination.

Interprocessor Arbitration

Bus Arbitration Algorithm : Static / Dynamic

- **Static : priority fixed**
  - »Serial (daisy-chain) arbitration
  - »Parallel arbitration
- **Dynamic : priority flexible**
  - »Time slice (fixed length time)
  - »Polling
  - »LRU
  - »FIFO
  - »Rotating daisy-chain

Interprocessor Communication & Synchronization

The various processors in a multiprocessor system must be provided with a facility for communicating with each other. 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. For example, parameters passed to a procedure in a different processor constitute interprocessor communication.

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.

Interprocessor Communication & Synchronization

In a shared memory multiprocessor system (tightly coupled):

- The most common procedure is to set aside a **portion of memory that is accessible to all processors**. The primary use of the common memory is to act as a message center similar to a mailbox, where each processor can leave messages for other processors and pick up messages intended for it.
- The sending processor structures a request, a message, or a procedure, and places it in the memory mailbox. **Status bits residing in common memory are generally used to indicate the condition of the mailbox,** whether it has meaningful information, and for which processor it is intended. The receiving processor can check the mailbox periodically to determine if there are valid messages for it. 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.

Interprocessor Communication & Synchronization

In a loosely coupled multiprocessor system(no shared memory):

- The memory is distributed among the processors and 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. When the sending processor and receiving processor name each other as a source and destination, a channel of communication is established. **A message is then sent with a header and various data objects used to communicate between nodes.** There may be a number of possible paths available to send the message between any two nodes. The operating system in each node contains routing information indicating the alternative paths that can be used to send a message to other nodes. The communication efficiency of the interprocessor network depends on the communication routing protocol, processor speed, data link speed, and the topology of the network.

Interprocessor Communication & Synchronization

I**nterprocessor Synchronization:** Multiprocessor systems usually include various mechanisms to deal with the synchronization of resources. A number of hardware mechanisms for mutual exclusion have been developed. One of the most popular methods is through the use of a **binary semaphore**.

Mutual Exclusion: A properly functioning multiprocessor system must provide a mechanism that will guarantee orderly access to shared memory and other shared resources. This is necessary **to protect data from being changed simultaneously by two or more processors.** Mutual exclusion must be provided in a multiprocessor system to enable one processor to lock out access to a shared resource by other processors when it is in a critical section.

Interprocessor Communication & Synchronization

Mutual Exclusion with Semaphore

» **Critical Session**: A critical section is a **program sequence that, once begun, must complete execution** before another processor accesses the same shared resource.

» **Semaphore**: **Indicate whether or not a processor is executing a critical section.** When the semaphore is equal to 1, it means that a processor is executing a critical program, so that the shared memory is not available to other processors. When the semaphore is equal to 0, the shared memory is available to any requesting processor.

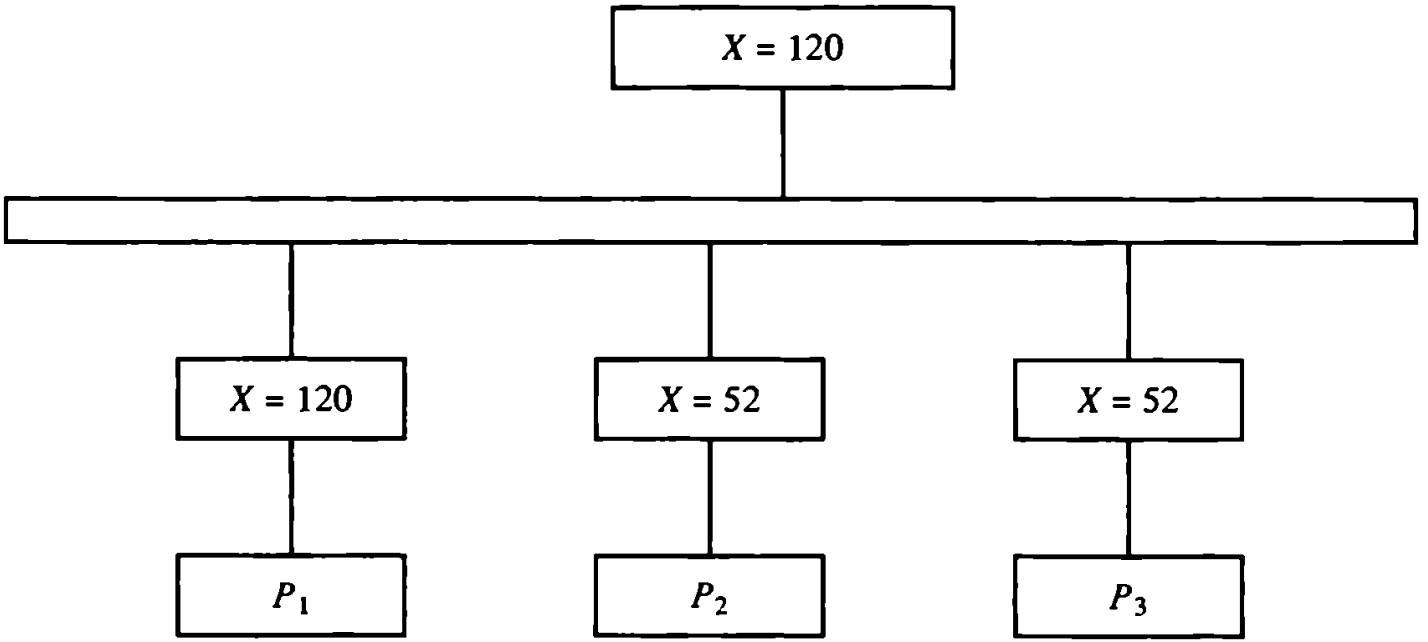
» **Hardware Lock**: A semaphore can be initialized by means of a **test and set instruction** in conjunction with a hardware lock mechanism. **A hardware lock is a processor generated signal that serves to prevent other processors from using the system bus as long as the signal is active**. The test-and-set instruction tests and sets a semaphore and activates the lock mechanism during the time that the instruction is being executed. This prevents other processors from changing the semaphore between the time that the processor is testing it and the time that it is setting it.

Cache Coherence

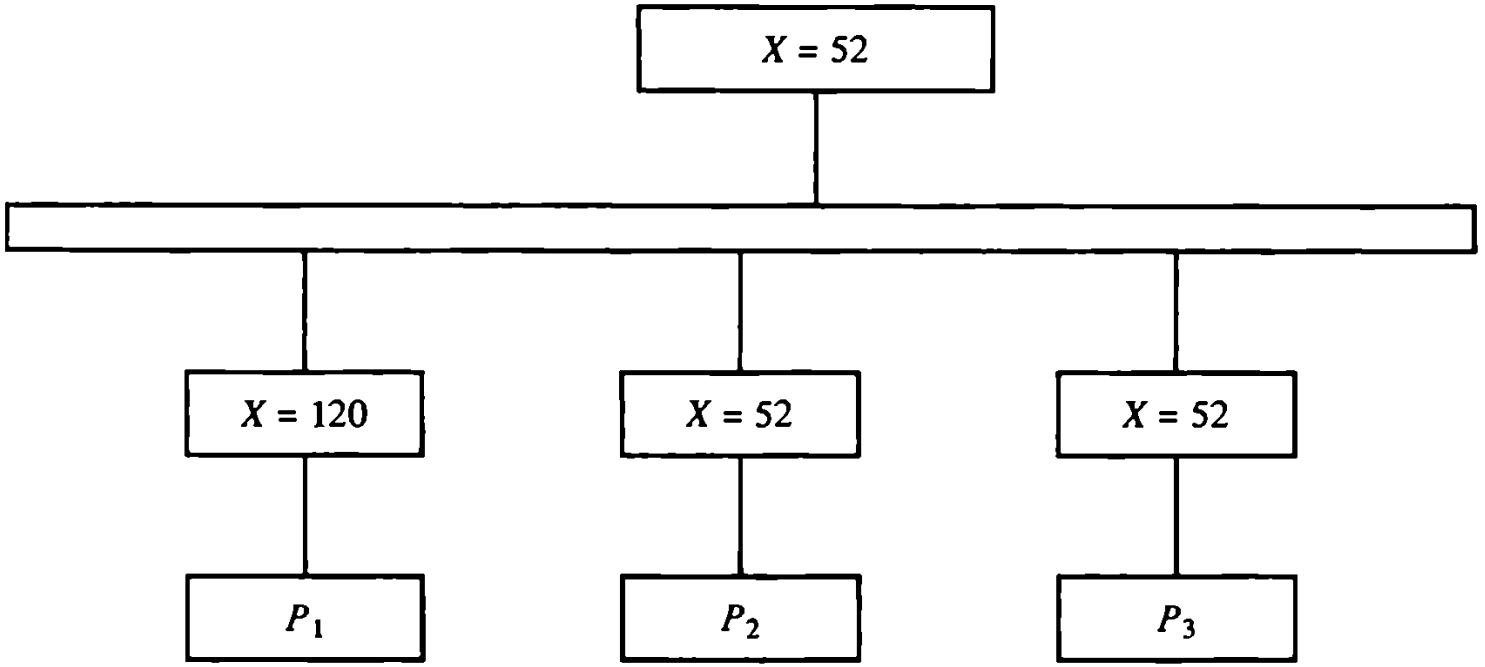
- In a shared memory multiprocessor system, all the processors share a common memory. In addition, each processor may have a local memory, part or all of which may be a cache. The compelling reason for having separate caches for each processor is to reduce the average access time in each processor.
- **The same information may reside in a number of copies in some caches and main memory. To ensure the ability of the system to execute memory operations correctly, the multiple copies must be kept identical. This requirement imposes a cache coherence problem.**
- Without a proper solution to the cache coherence problem, caching cannot be used in bus-oriented multiprocessors with two or more processors.

Conditions for Incoherence:

- **Cache coherence problems exist in multiprocessors with private caches because of the need to share writable data. Read-only data can safely be replicated without cache coherence enforcement mechanisms.**
- To illustrate the problem, consider the three-processor configuration with private caches. Sometime during the operation an element X from main memory is loaded into the three processors, P1, P2, and P3. As a consequence, it is also copied into the private caches of the three processors. For simplicity, we assume that X contains the value of 52. The load on X to the three processors results in consistent copies in the caches and main memory.
- If one of the processors performs a store to X, the copies of X in the caches become inconsistent. A load by the other processors will not return the latest value. Depending on the memory update policy used in the cache, the main memory may also be inconsistent with respect to the cache. This is shown in Fig. 13-13.
- A store to X (of the value of 120) into the cache of processor P1 updates memory to the new value in a **write-through policy**. **A write-through policy maintains consistency between memory and the originating cache, but the other two caches are inconsistent since they still hold the old value.**
- **In a write-back policy, main memory is not updated at the time of the store. The copies in the other two caches and main memory are inconsistent.** Memory is updated eventually when the modified data in the cache are copied back into memory.



A write-through policy maintains consistency between memory and the originating cache, but the other two caches are inconsistent since they still hold the old value.



In a write-back policy, main memory is not updated at the time of the store. The copies in the other two caches and main memory are inconsistent.

Solutions to the Cache Coherence Problem

Software:

» 1) Shared writable data are non-cacheable

» 2) Writable data exists in one cache : Centralized global table

Hardware:

» 1) Monitor possible write operation : Snoopy cache controller

Practice Questions

- What are the advantages of a multiprocessor system?
- Discuss the difference between tightly coupled and loosely coupled multiprocessors.
- Describe the following terminology associated with multiprocessors.

  (a) mutual exclusion; (b) critical section; (c) hardware lock; (d) semaphore;

  (e) test-and-set instruction.
- What is cache coherence, and why is it important in shared-memory multiprocessor systems? How can the problem be resolved with a snoopy cache controller?
- Explain Interconnection structures?
- Explain the importance of multiprocessor system.
- Explain shared memory multiprocessors?
- What is Inter processor arbitration?
- Explain inter processor communication.
- What is synchronization?
- What do you understand by Cache coherence Problem? Give an example.