Multiprocessor

Dr e v prasad

Dt. 12.10.17

Multiprocessor

Contents

Introduction

- A multiprocessor system is an interconnection of two or more CPUs with memory and I/O equipment.
- IOPs are generally not included in the definitions of multiprocessor system unless they have computational facilities comparable to CPUs.
- Multiprocessor are MIMD system.
- Multicomputer system includes number of computers connected together by means of communication lines.

- It improves reliability.
- If one system fails, the whole system continue to function with perhaps low efficiency.
- The computation can proceed in parallel in two ways:
  - Multiple independent jobs operate in parallel
  - A single job can be partitioned into multiple parallel tasks
  - Multiprocessor systems are classified by the way their memory is organized:

    1. tightly coupled systems (Shared memory)
    2. loosely coupled systems (Distributed memory)

Interconnection Structures

- The components of multiprocessor system include:
- CPUs
- IOPs
- I/O devices
- Memory

Interconnection networks for multiprocessor systems are:

- Time Shared Common Bus
- Multiport Memory
- Crossbar Switch
- Multistage switching network
- Hypercube System (Binary n- cube)

Assessing Network Alternatives

- Buses
  - excellent cost scalability
  - poor performance scalability
- Crossbars
  - excellent performance scalability
  - poor cost scalability
- Multistage interconnects
  - compromise between these extremes

Time Shared Common Bus

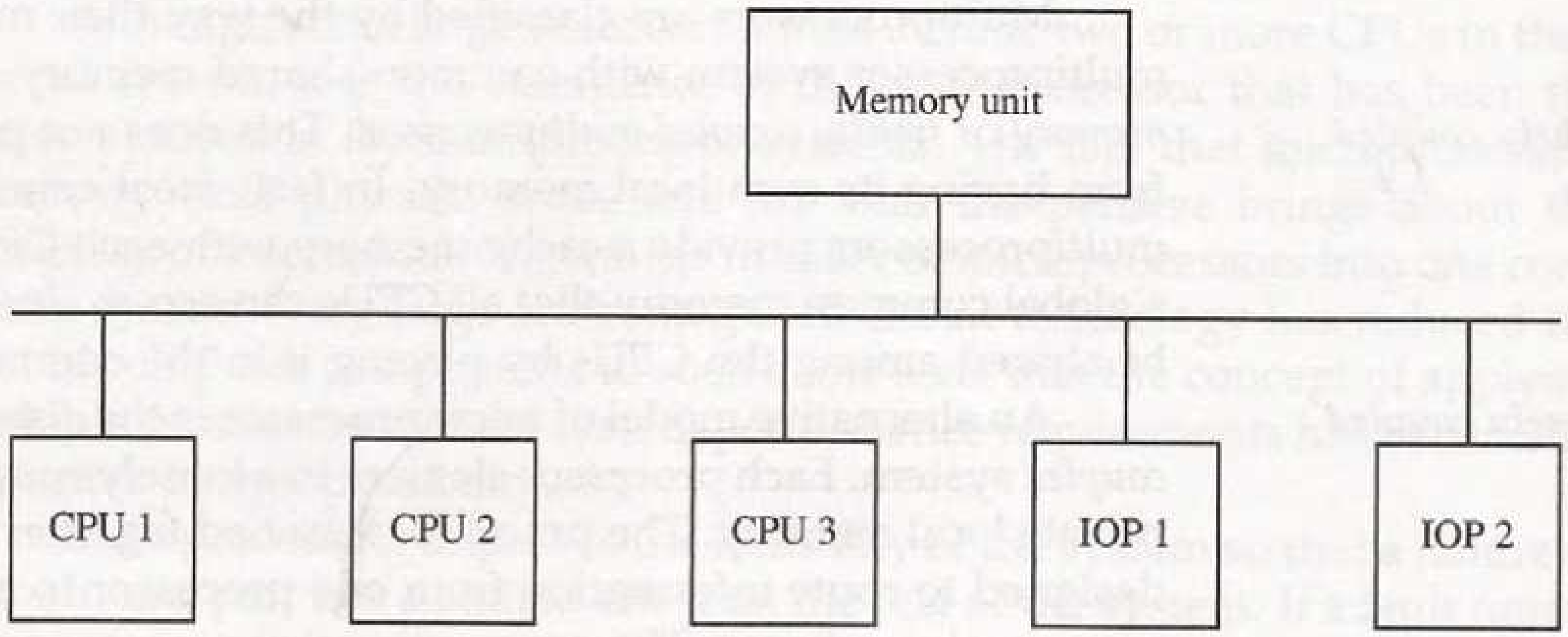


Figure Time-shared common bus organization.

Multiprocessor systems –system bus

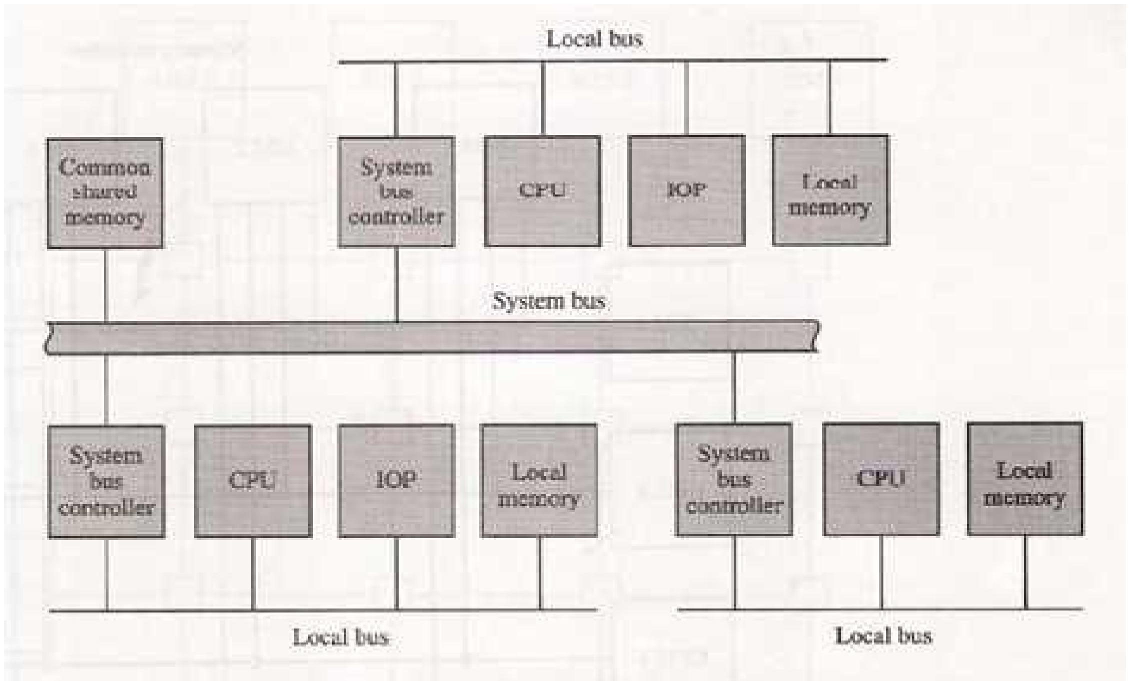
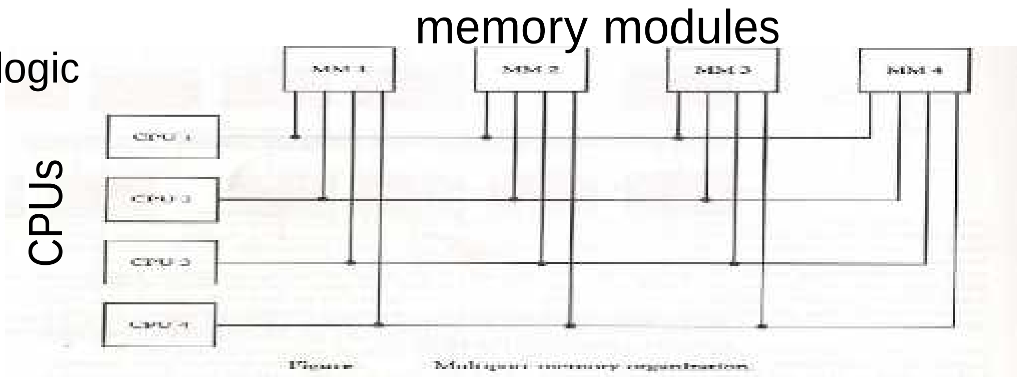


Figure System bus structure for multiprocessors.

Multiport memory system

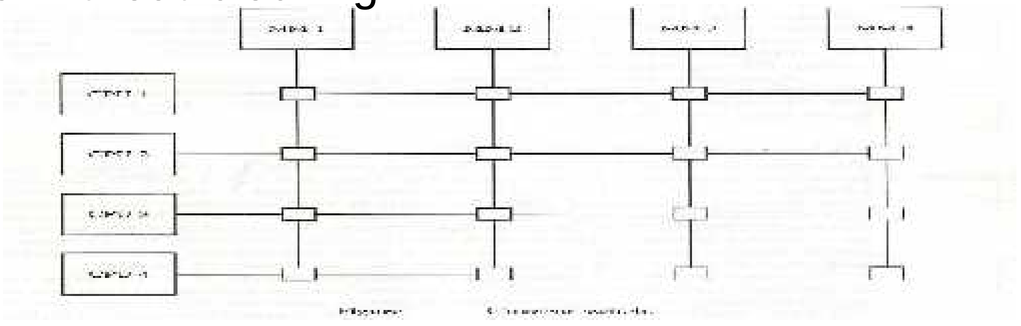
Employs separate buses between each memory module and each CPU.
This is shown in Fig.for four CPUs and four memory modules (MMs).
Each processor bus is connected to each memory module (MM)
A processor bus consists of the address, data, and control lines
required to communicate with memory.
The MM is said to have 4 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.

- ❖ high transfer rate
- ❖ expensive memory control logic



Crossbar Switch

- Crossbar is the simplest and most flexible switch architecture.
- It uses an n×m grid of switches to connect n inputs to m outputs
- It can be used to establish simultaneously n connections between n inputs and m outputs.
- When two or more inputs request the same output, it is called CONFLICT. Only one of them is connected and others are either dropped or buffered
- A crossbar switch contains a matrix of simple switch elements that can switch on and off to create or break a connection.
- Each crosspoint can be switched on or off under program control
- Turning on a switch element in the matrix, a connection between a processor and a memory can be made.
- Crossbar switches are non-blocking, that is all communication permutations can be performed without blocking.
- Crossbar switch is not scalable



The crossbar switch organization consists of a number of crosspoints that are placed at intersections between processor buses and memory module paths.

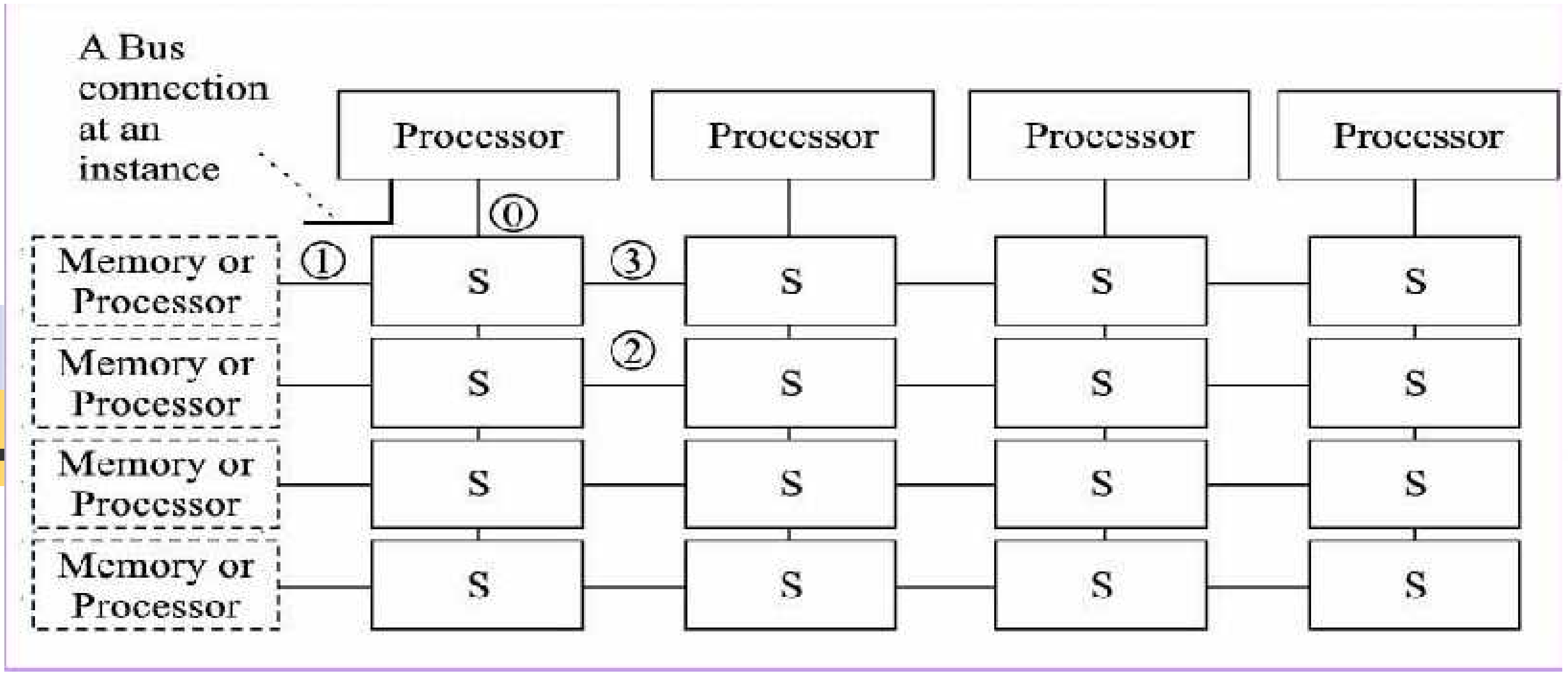
• Figure shows a crossbar switch interconnection between four CPUs and four memory modules.

• The small square in each crosspoint 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. • It also resolves multiple requests for access to the same memory module on a predetermined priority basis.

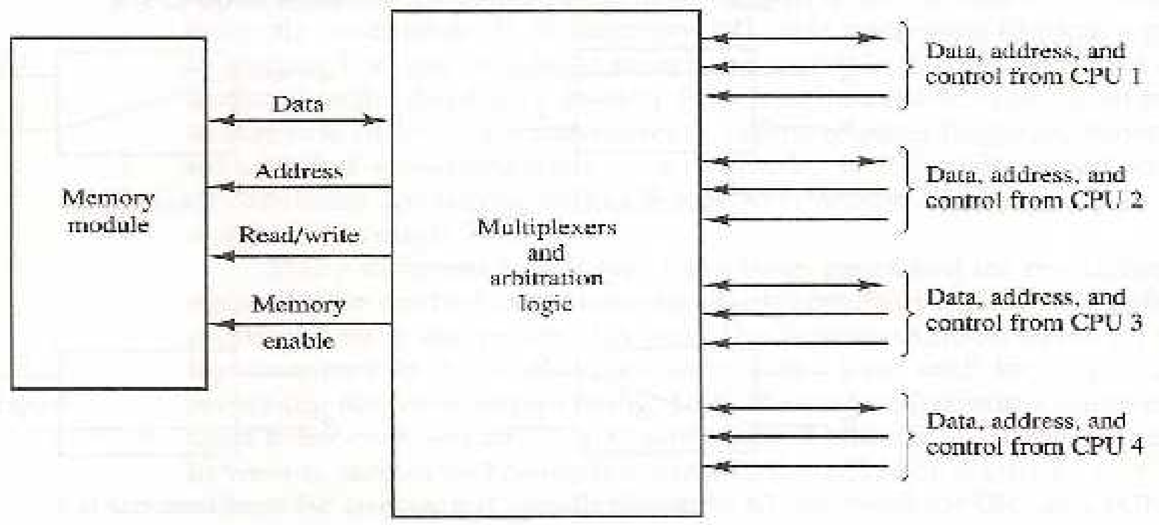
•



Each processor has switch to memory bus horizontally and processor-to-switch links vertically
A switch S having four I/O paths (0, 1, 2, 3)
: 0-1, 0-2, 0-3 : 1-0, 1-2,1-3 : 2-0, 2-1,2-3 : 3-0, 3-1,3-2

Figure 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.
A crossbar switch organization supports simultaneous transfers from all 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.

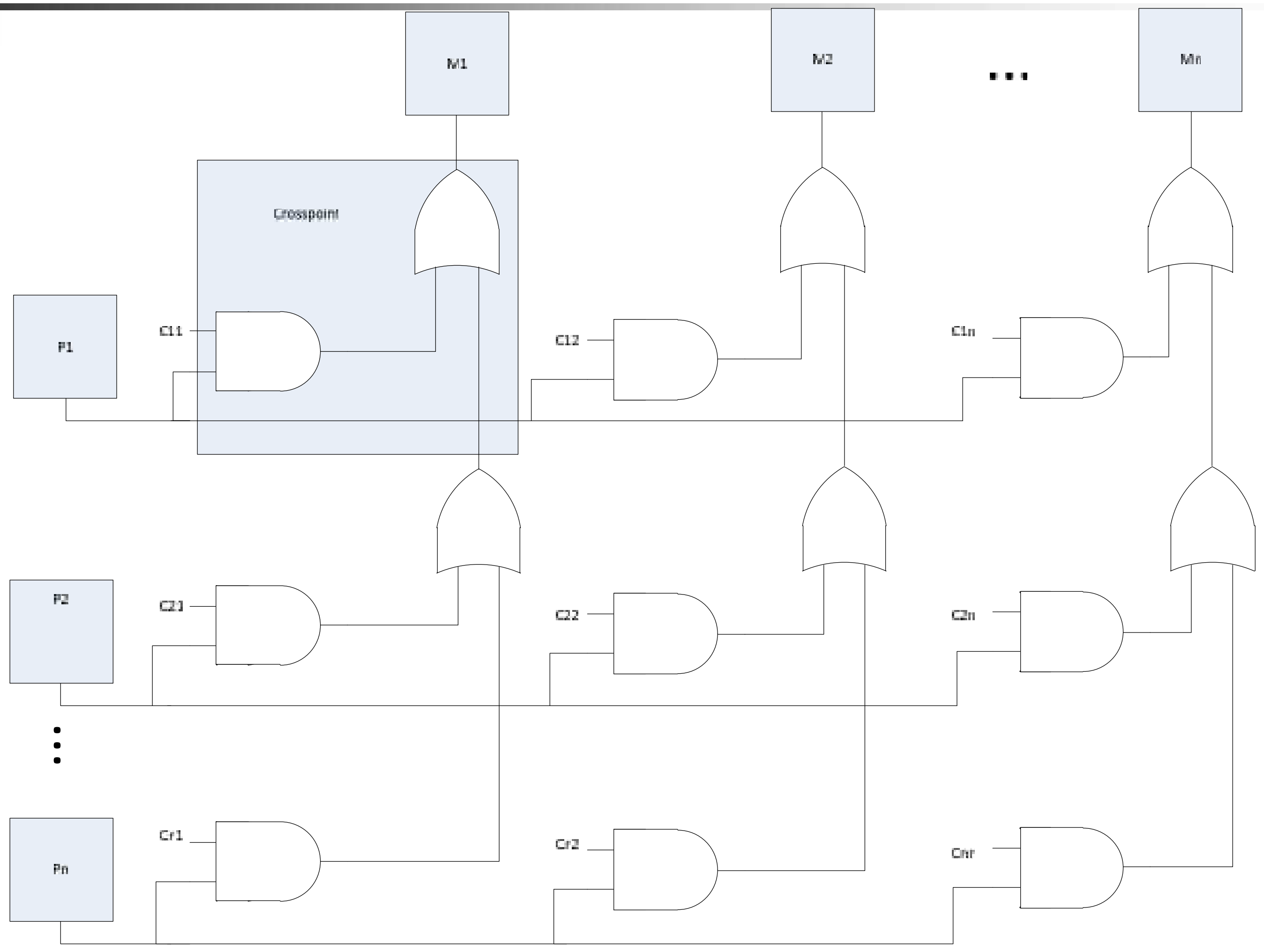
Figure Block diagram of crossbar switch.


Memory module
Data
Address
Read/write
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

1(a) Use two-input AND and OR gates to construct NxN crossbar switch network between N processors and N memory modules. Use cij signal as the enable signal for the switch in ith row and jth column. Let the width of each crosspoint be w bits.

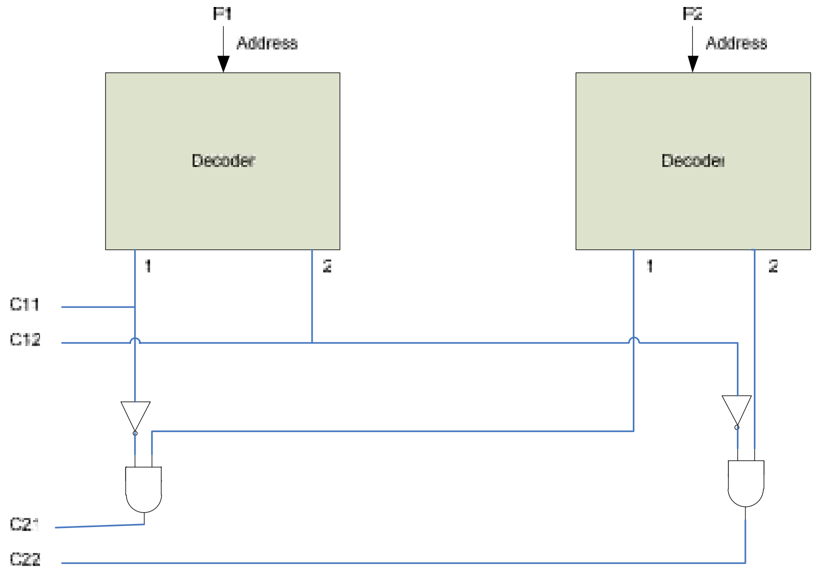
(b) Estimate the total number of AND and OR gates needed as a function of N and w.

Problem (cont.)



Problem (cont.)

A simplify version of the arbiter. 2 x 2 crossbar



The crossbar uses priority to determine who gets to go first when Two PE try to communicate with a single memory.

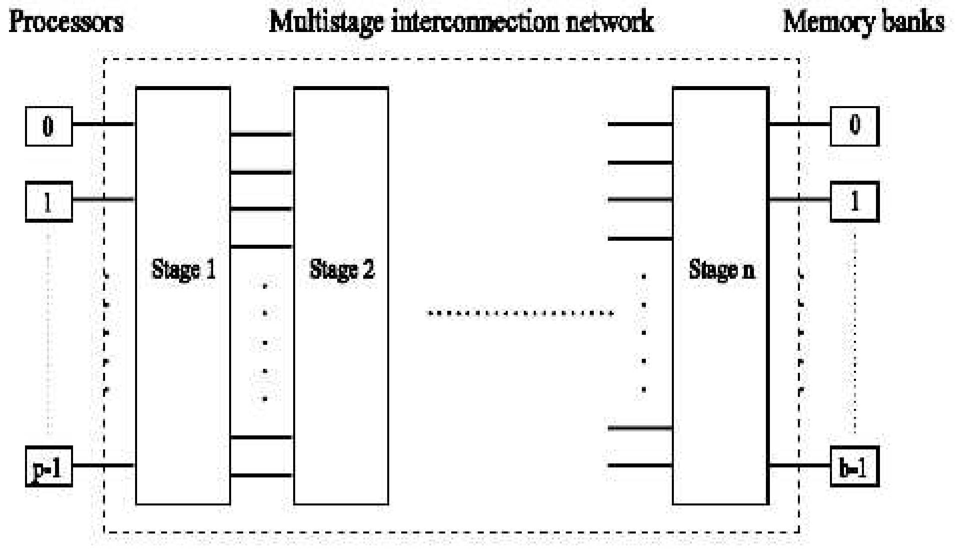
P_1 has priority over P_2 , P_2 over P_3, P_{N-1} over P_N.

Cij is the control signals to determine which cross point gets “activated”.

The decoder gets an address (to determine which memory the PE wants to communicate with). So for example, if P1 wants to communicate with M1, it would send 1 to C11 and C21 would get 0 (since there is a NOT gate). What that means if P2 wanted to c

Multistage Network

- MINs are a class of high-speed computer networks
- An MIN consists of a sequence of switching stages, each of which consists of several switches.
- The switching stages are connected with inter stage links between successive stages, usually composed of processing elements (PEs) on one end of the network and memory elements (MEs) on the other end, connected by switching elements (SEs).
- The switching elements themselves are usually connected to each other in stages.



Multistage Switching Interconnection Networks (MINs)

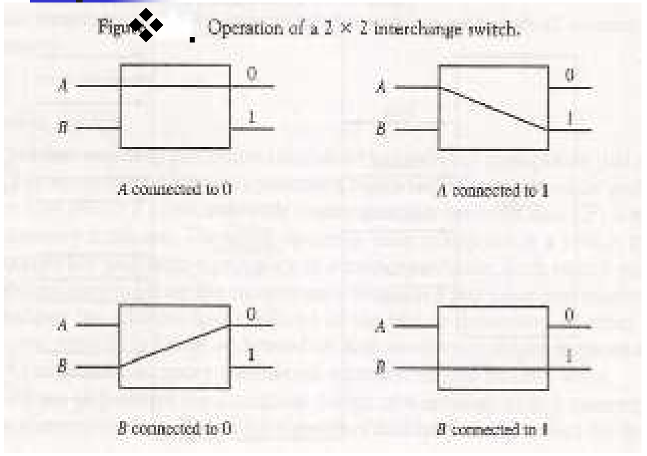
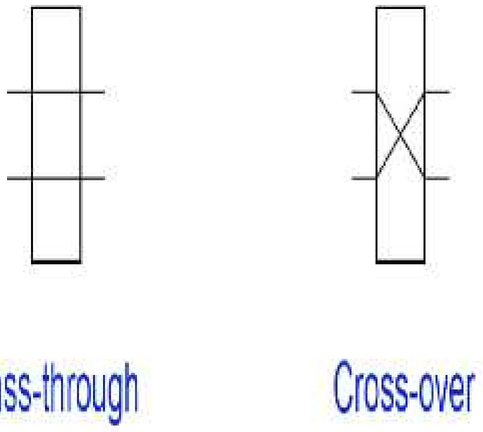


Figure . Operation of a 2 × 2 interchange switch.

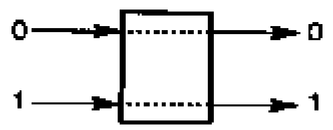
- The basic component of MIN is a 2 X 2 interchange switch.
- 2 x 2 switch has two inputs, labeled A and B, and two outputs, labeled 0 and 1.
- There are control signals associated with the switch that establish the interconnection between the input and output terminals.

switching elements

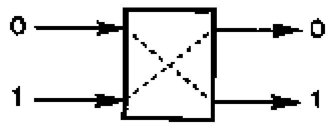


2 × 2 Switches

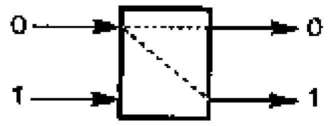
control signals



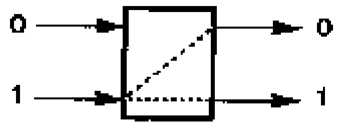
(a) Straight



(b) Crossover



(c) Upper broadcast



(d) Lower broadcast

❖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

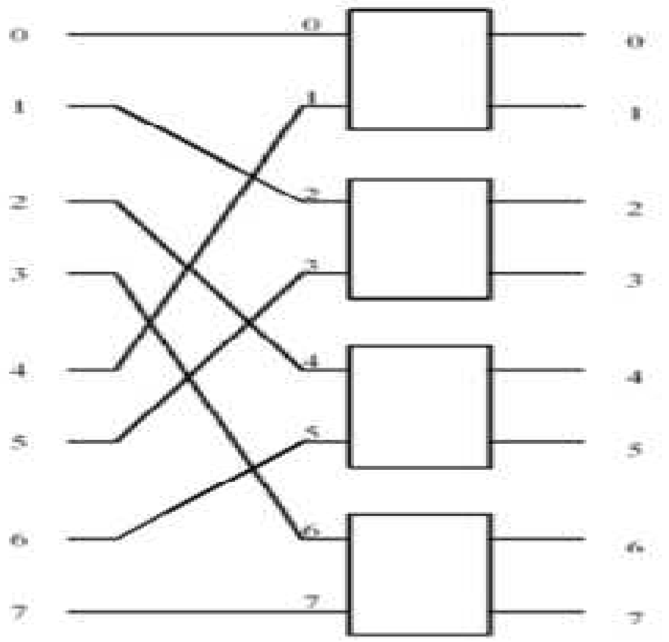
Network Topology

MINs networks can be categorized on the basis of their topology.
Topology is the pattern in which one node is connected to other nodes.
There are two main types of topology: static and dynamic
Static interconnect networks are hard-wired and cannot change their configurations. point-to-point communication links
– that don't change dynamically (e.g., trees, rings, meshes)
Dynamic networks: that change interconnectivity dynamically
Implemented with switched communication links
(e.g.,system buses, crossbar switches, multistage networks)
The regular structure signifies that the nodes are arranged in specific shape and the shape is maintained throughout the networks.
The way input units are connected with the output units, determine the functional characteristics of the network, i.e., the allowable interconnections
In a single stage network, data may have to be passed through the switches several times before reaching the final destination.
In multistage network, one pass of multistage stages of switches is usually sufficient.

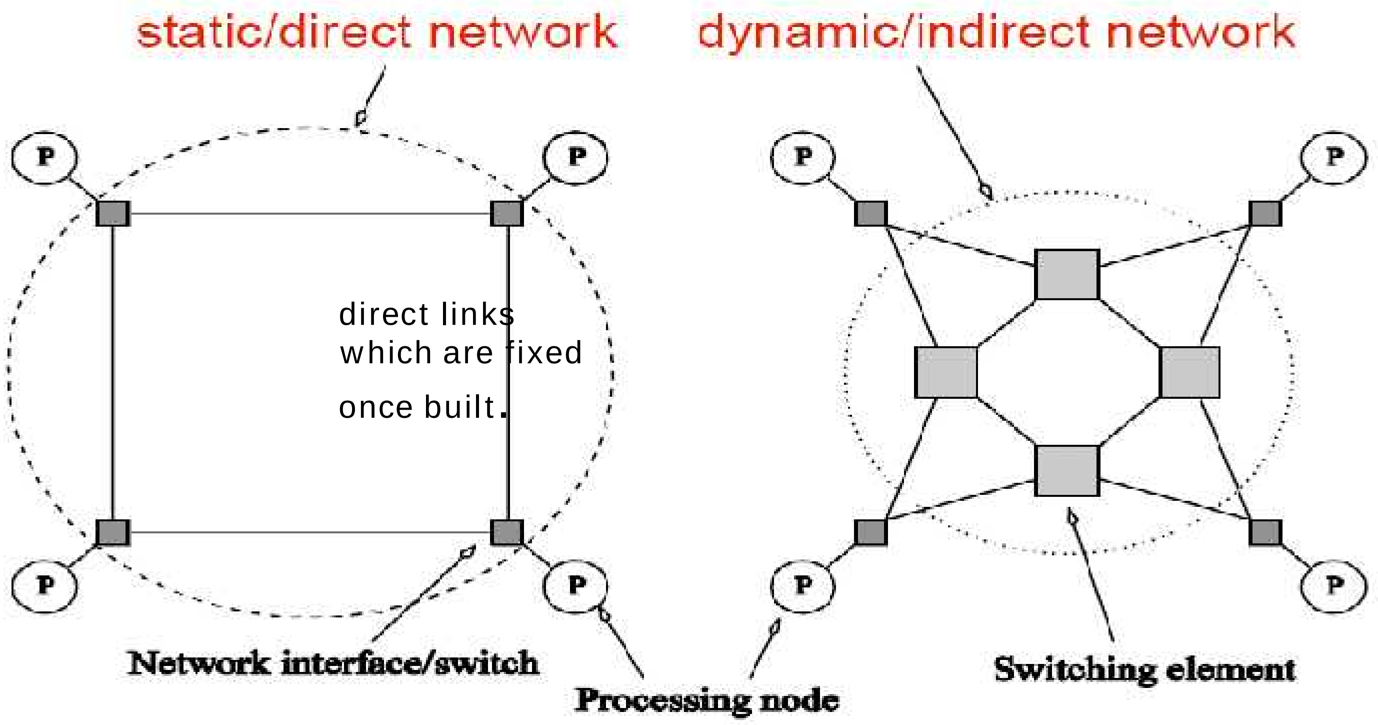
Single Stage Interconnect Network

The input nodes are connected to output via a single stage of switches. The figure shows 8*8 single stage switch using shuffle exchange.

The way input units are connected with the output units, determine the functional characteristics of the network, i.e., the allowable interconnections.



Static vs. Dynamic



Switching element : maps a fixed number of inputs to outputs

Classification of Multistage Interconnect Network

- Multistage switches provide better blocking performance than single stage switches, as they provide alternative paths for a particular source destination pair. nodes at ends, switches in middle
- Increasing the number of stages will lead to advantage in the number of Cross Points. But the complexity also increases, so overall advantage is not much.

Classification

Non-blocking: A non-blocking network can connect any idle input to any idle output, regardless of the connections already established across the network. Crossbar is an example of this type of network.

Rearrangeable non-blocking: This type of network can establish all possible connections between inputs and outputs by rearranging its existing connections.

Blocking: It can perform many, but cannot realize all possible connections between terminals. Example: the Omega network

This is because a connection between one free input to another free output is blocked by an existing connection in network.

Multi Stage Interconnect Network

- ❖ A multistage network consists of multiple stages of switches. 2x2 switch elements are a common choice for many multistage networks.
- ❖ The number of stages determine the delay of the network.
- ❖ By choosing different interstage connection patterns, various types of multistage network can be created.

❖ The two processors P1 and P2 are connected through switches to eight memory modules marked in binary from 0 0 0 through 111.
❖ The path from a source to a destination is determined from the binary bits of the destination number

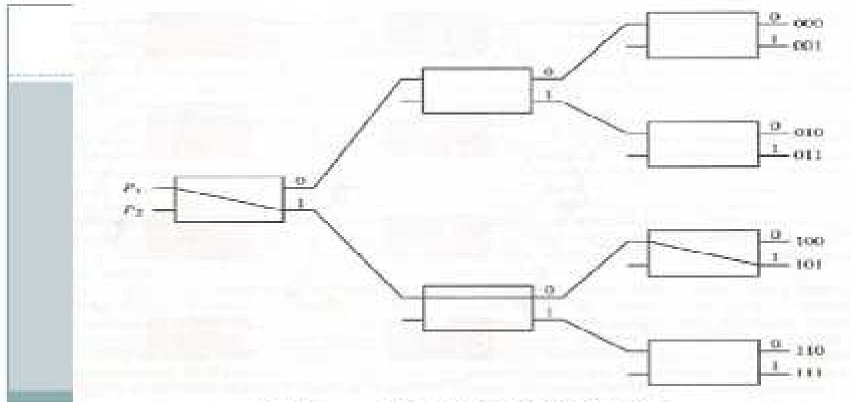
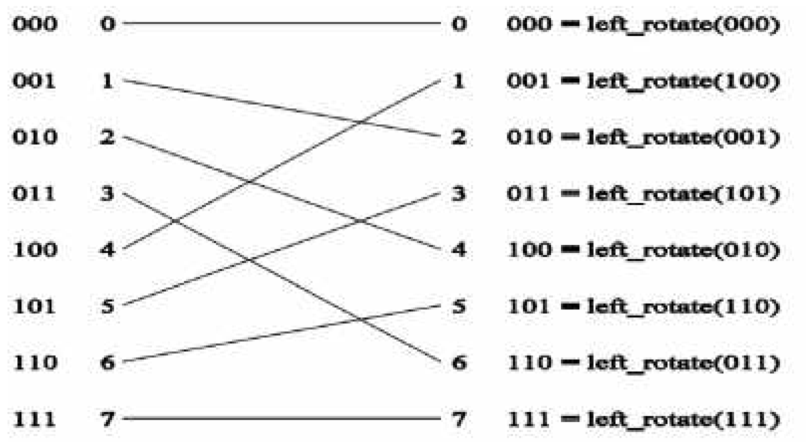


Figure Binary tree with 2 × 2 switches.

e.g, 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.
It is clear that either P1 or P2 can be connected to any one of the eight memories, Certain request patterns, however, cannot be satisfied simultaneously.
For example, if P1 is connected to one of the destinations 0 0 0 through 0 11 ,P2 can be connected to only one of the destinations 1 0 0 through 111.

Omega Network Stage

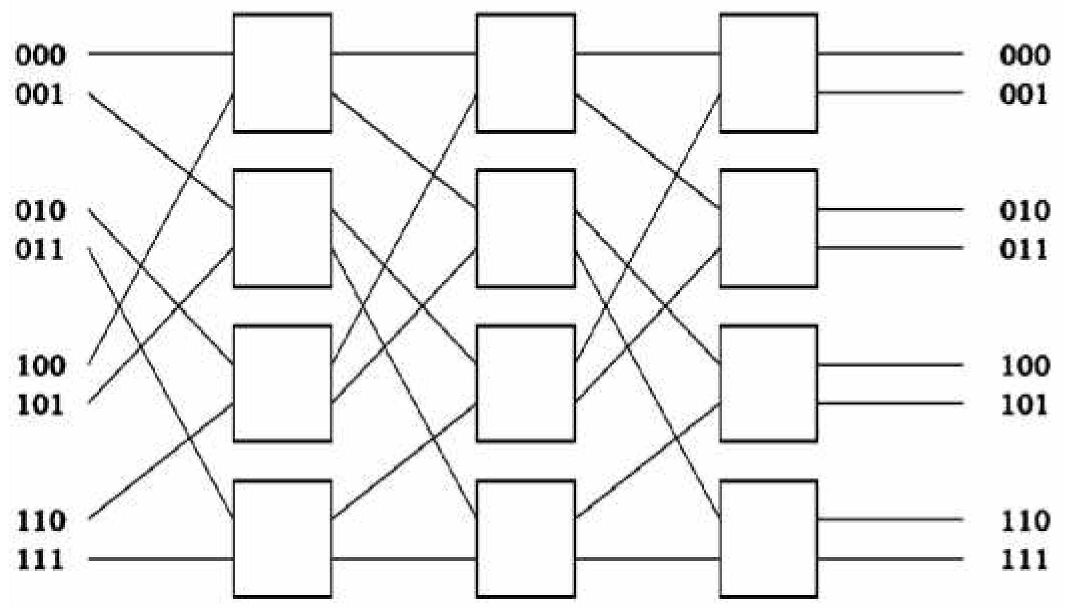
One topology proposed for MINs to control processor—memory communication in a tightly coupled multiprocessor system or to control the communication between the processing elements in a loosely coupled.
One such topology is the omega switching network shown in Fig.
In this configuration, there is exactly one path from each source to any particular destination. Some request patterns, however, cannot be connected simultaneously. For example, any two sources cannot he connected simultaneously to destinations 0 0 0 through 111.
Some request patterns, however, cannot be connected simultaneously.
For example, any two sources cannot he connected simultaneously to destinations 0 0 0 through 111



Omega Network

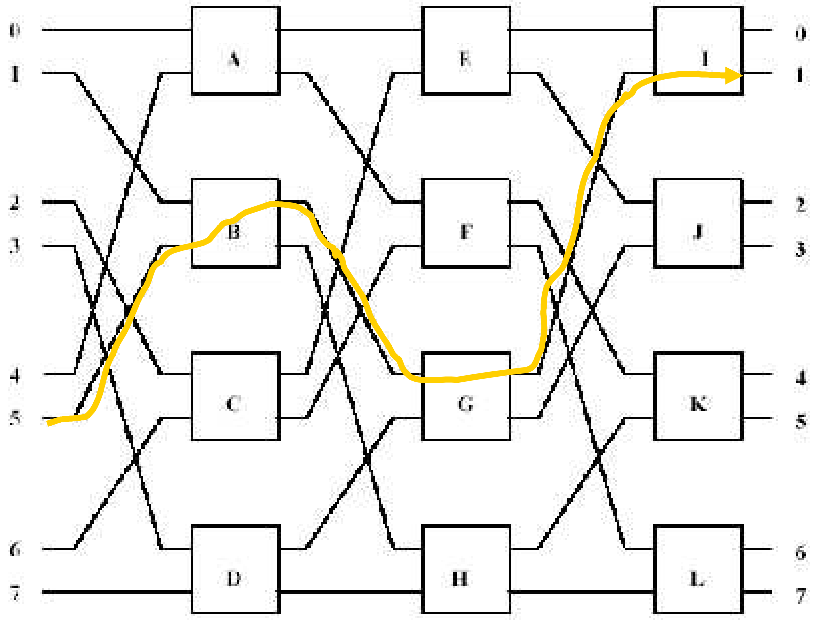
- ❖ An omega network consists of multiple stages of 2X2 switching elements.
- ❖ Each input has a dedicated connection to an output.
- ❖ An N*N omega network, (N processing element) has $\log_2$ (N) number of stages and N/2 number of switching elements in each stage
- ❖ uses perfect shuffle between stages.

perfect shuffle of inputs of n PEs to n/2 switches



Omega networks

- A multi-stage IN using 2 × 2 switch boxes and a perfect shuffle interconnect pattern between the stages
- In the Omega MIN there is one unique path from each input to each output.
- No redundant paths → no fault tolerance and the possibility of blocking.



Example:

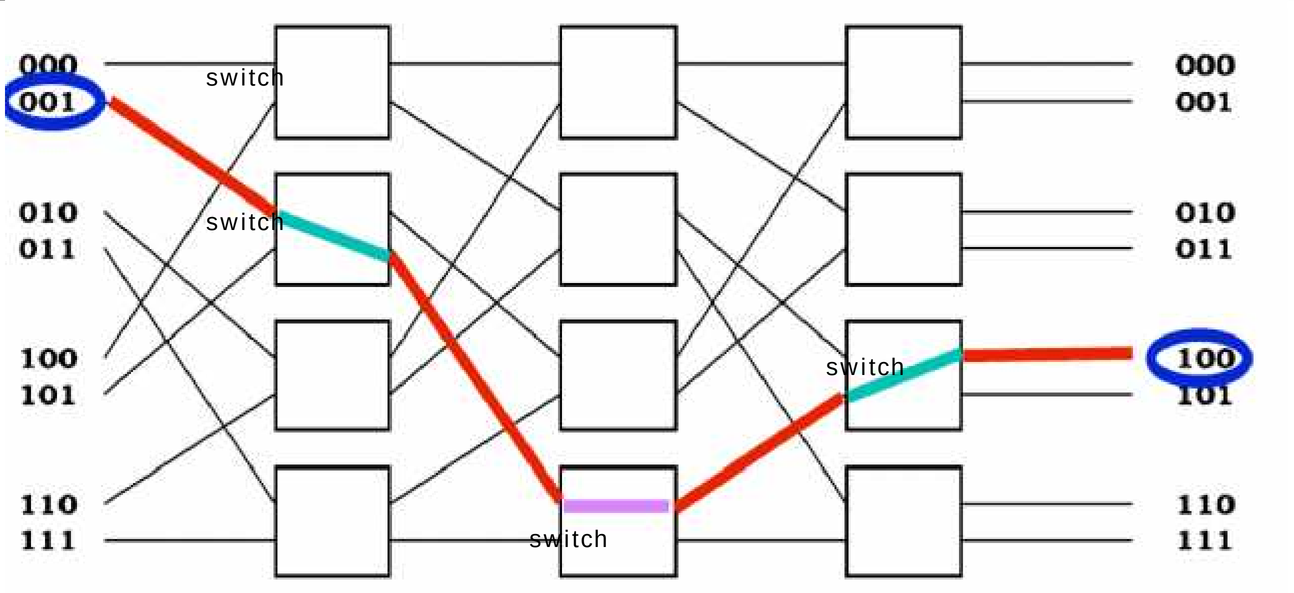
- Connect input 101 to output 001
- Use the bits of the destination address, 001, for dynamically selecting a path
- Routing:
  - 0 means use upper output
  - 1 means use lower output

Omega Network Routing

- Let
  - s = binary representation of the source processor
  - d = binary representation of the destination processor or memory
- The data traverses the link to the first switching node
  - if the most significant bit of s and d are the same
    - route data in pass-through mode by the switch
  - else
    - use crossover path
- Strip off leftmost bit of s and d
- Repeat for each of the $\log_2 N$ switching stages

Omega Network Routing

How to connect PE 001 to Memory module 100 ?

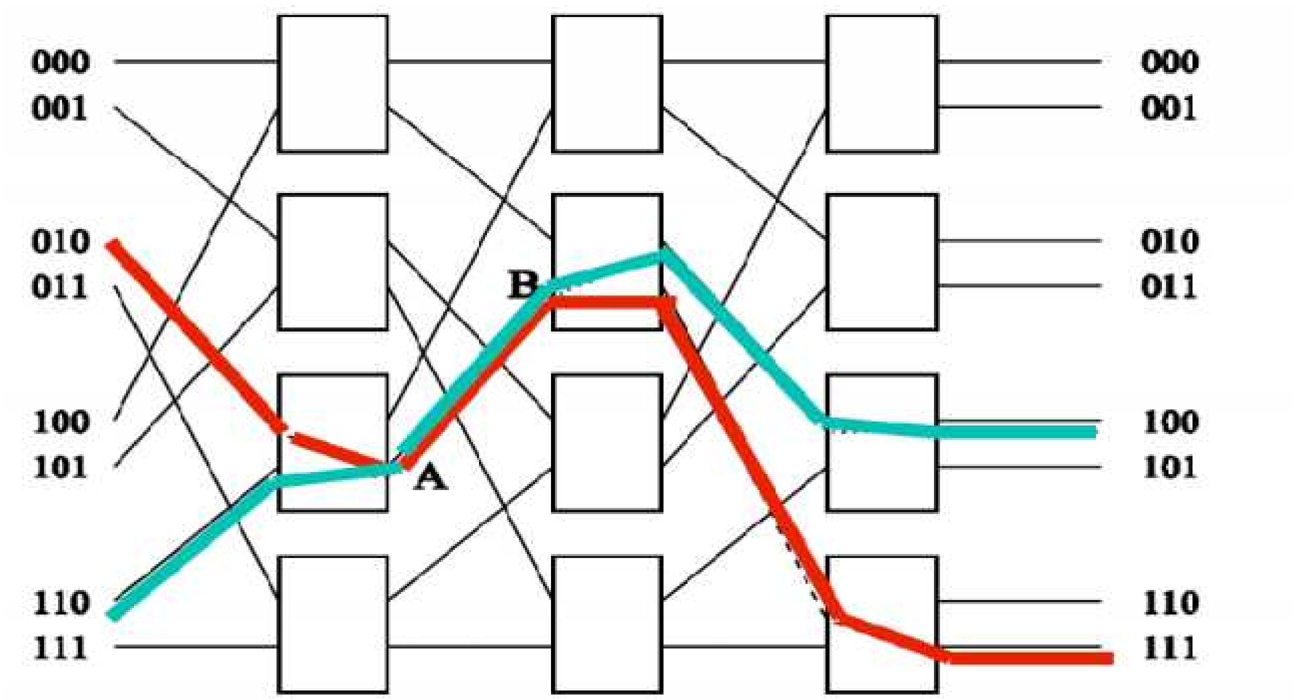


The source node generates a tag, which is binary equivalent Of the destination. At each switch, the corresponding tag bit is checked.

- If the bit is 0, the input is connected to the upper output. If it is 1, the Input is connected to the lower output.

Blocking in an Omega Network

Are u able to connect simultaneously :
(i) PE 010 to Memory module 111 and (ii) PE 110 to Memory module 100 ?



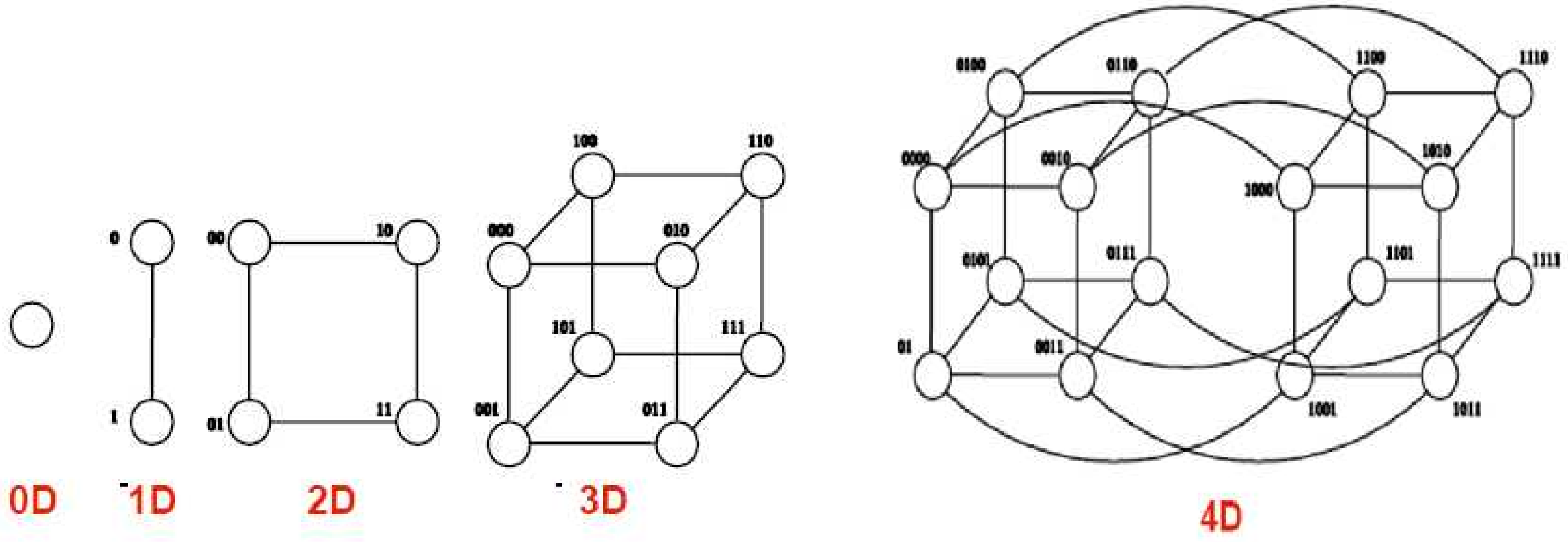
switch conflict : If both inputs of the switch have either 0 or 1. One of them is connected. The other one is rejected or buffered at the switch

Hypercube Interconnection

- Hypercube or binary n-cube multiprocessor structure is a loosely coupled system.
- It composed of $N=2^n$ processors interconnected in n-dimensional binary cube.
- Each processor form the node of the cube.
- Each processor has direct communication path with (n) other neighbor processor.
- There are 2^n distinct n-bit binary address that can be assigned to each processor.

Hypercube Connection

Special d-dimensional mesh: p nodes, d = log p



n-cube can be formed by interconnecting corresponding nodes of two (n-1)-cubes
2 nodes are adjacent if they differ in exactly 1 bit.

Static Interconnects

Hypercube (cont.)

Point-to-point Routing

compare IDs of S & D, if S > D

look at left most bit

Ex. S 101 and D 000

Broadcasting (suppose from 0)

Step 1
0 – 1

Step 2
0 - 2
1 - 3

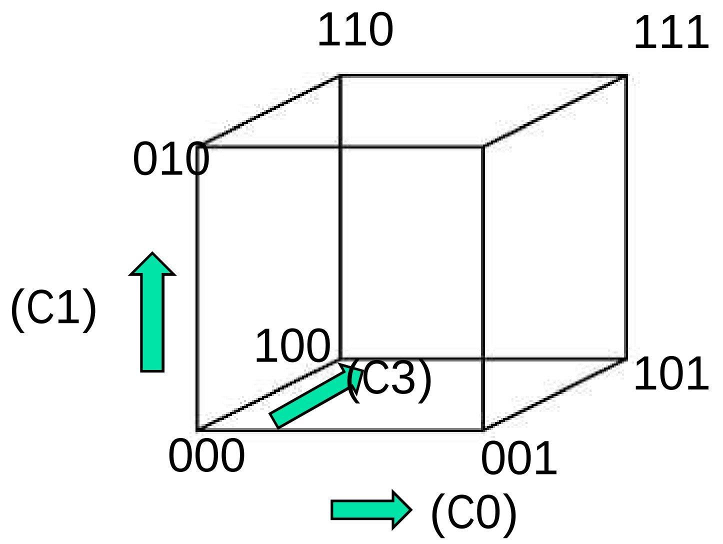
Step 3
0 - 4
1 - 5
2 - 6
3 – 7

Multi stage 3-cube

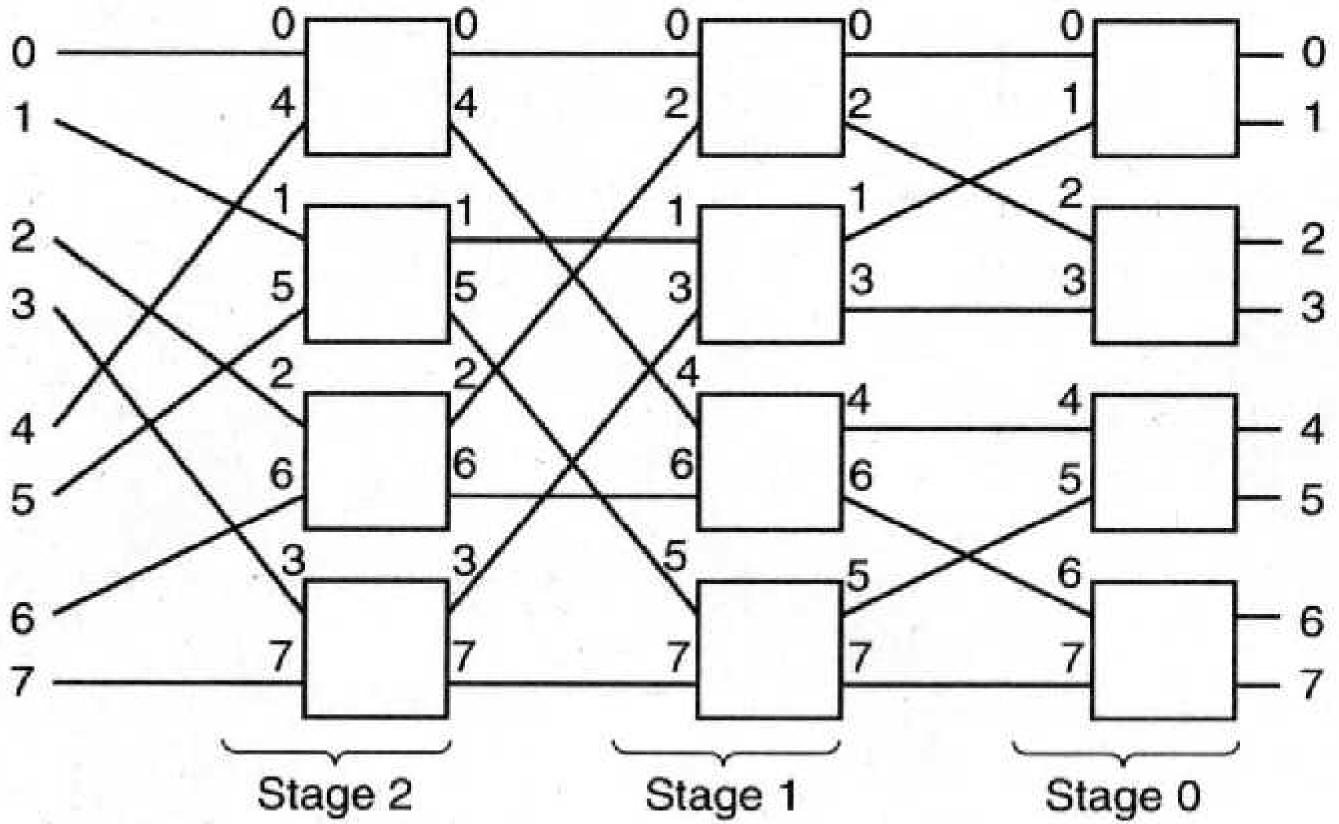
i) Routing by least significant bit (C0)
0 – 1 , 2–3 , 4 – 5 , 6–7

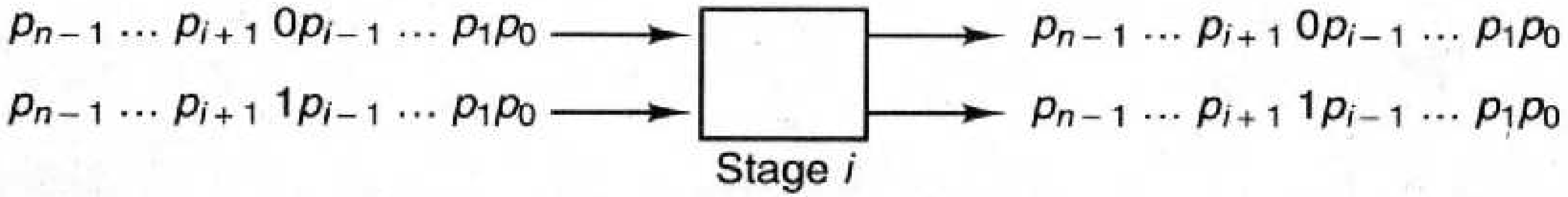
ii) Routing by least significant bit (C1)
0 – 2 , 1–3 , 4 – 6 , 5–7

iii) Routing by least significant bit (C2)
0 – 4 , 1–5, 2 – 6 , 3–7



Multi stage 3-cube




$p_{n-1} \cdots p_{i+1} 0 p_{i-1} \cdots p_1 p_0$
$p_{n-1} \cdots p_{i+1} 1 p_{i-1} \cdots p_1 p_0$
$p_{n-1} \cdots p_{i+1} 0 p_{i-1} \cdots p_1 p_0$
$p_{n-1} \cdots p_{i+1} 1 p_{i-1} \cdots p_1 p_0$
Stage *i*

Link labels of each switch

System Bus

- A typical system bus consists of approximately 100 signal lines.
- System bus is divided into 3 functional groups of lines :

  data bus , address bus and control bus.
- In addition there are power distribution lines that supply power to the components.
- Ex. IEEE standard 796 multi bus system has 16 data lines, 24 address lines, 26 control lines and 20 power lines for total of 86 lines.
- Data lines provide a path for the transfer of data between processor and common memory.
- The number of data lines are usually multiple of 8, with 16 and 32 being the most common.

- Address lines are used to indentify memory location or any other source and destination units.
- The number of address lines determine the maximum possible memory capacity in the system.
- The control lines provides signal for controlling information transfer.
- Timing signals indicate validity of data and address.
- Command signal specify the operation to be performed.
- Data transfer on the system can be either Synchronous

  Or Asynchronous

Arbitration Procedure

- Arbitration procedure services all processor requests on the basis of established priorities.
- It can be implemented either HW (static) or SW (dynamic) Arbitration techniques:

  i) Static Techniques

  - Serial Arbitration (Serial Connection of Units)
  - Parallel Arbitration (Parallel Connection of Units)

  In static techniques the priorities assigned are fixed.

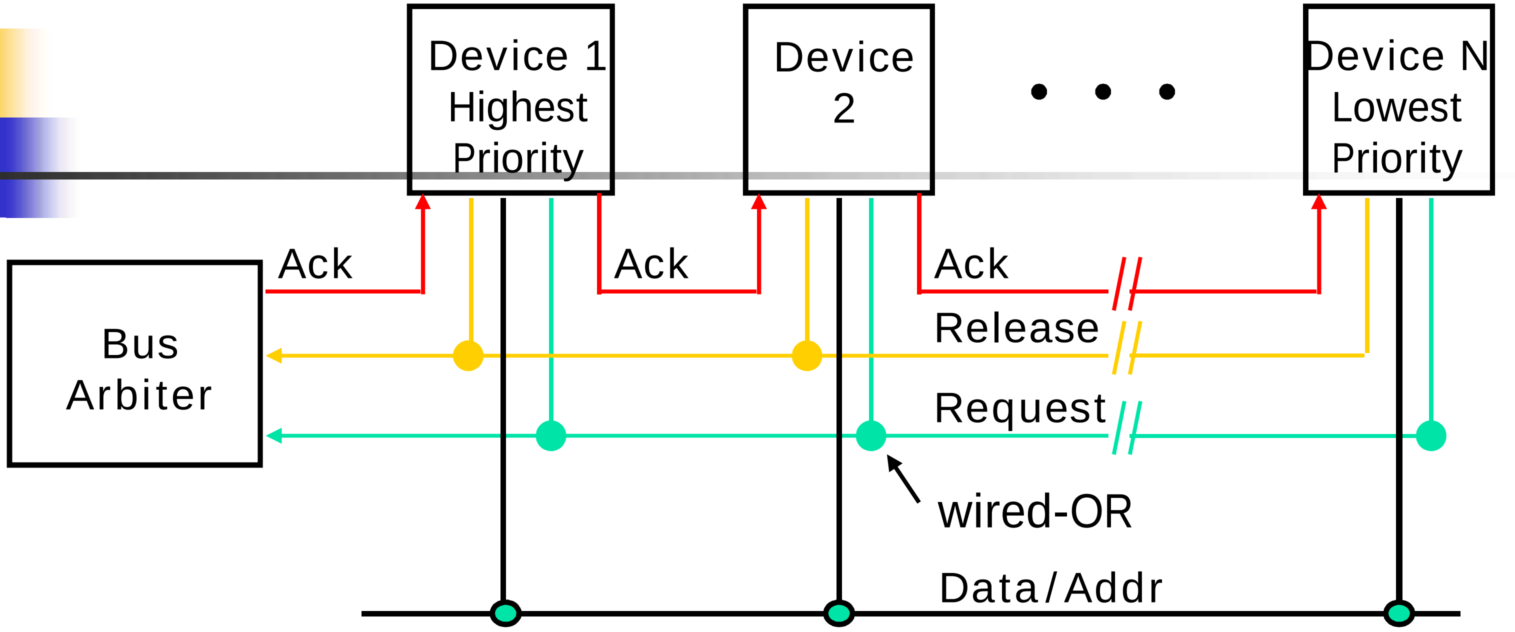
  ii) Dynamic Techniques

  Dynamic arbitration priorities of the system change while the system is in operation.

Serial Arbitration Procedure

- It is obtained from daisy chain connection of bus arbitration circuit similar to the case of priority interrupt.
- Each bus has its own arbiter logic which are arranged according to the priorities from highest to the lowest.
- The daisy-chaining method involves connecting all the devices that can request an interrupt in a serial manner.
  This configuration is governed by the priority of the devices.
  The device with the highest priority is placed first followed by the second highest priority device and so on.
  The given figure depicts this arrangement.

Daisy Chain Bus Arbitration



- Advantage: simple
- Disadvantages:
  - Cannot assure fairness – a low-priority device may be locked out indefinitely
  - Slower – the daisy chain grant signal limits the bus speed

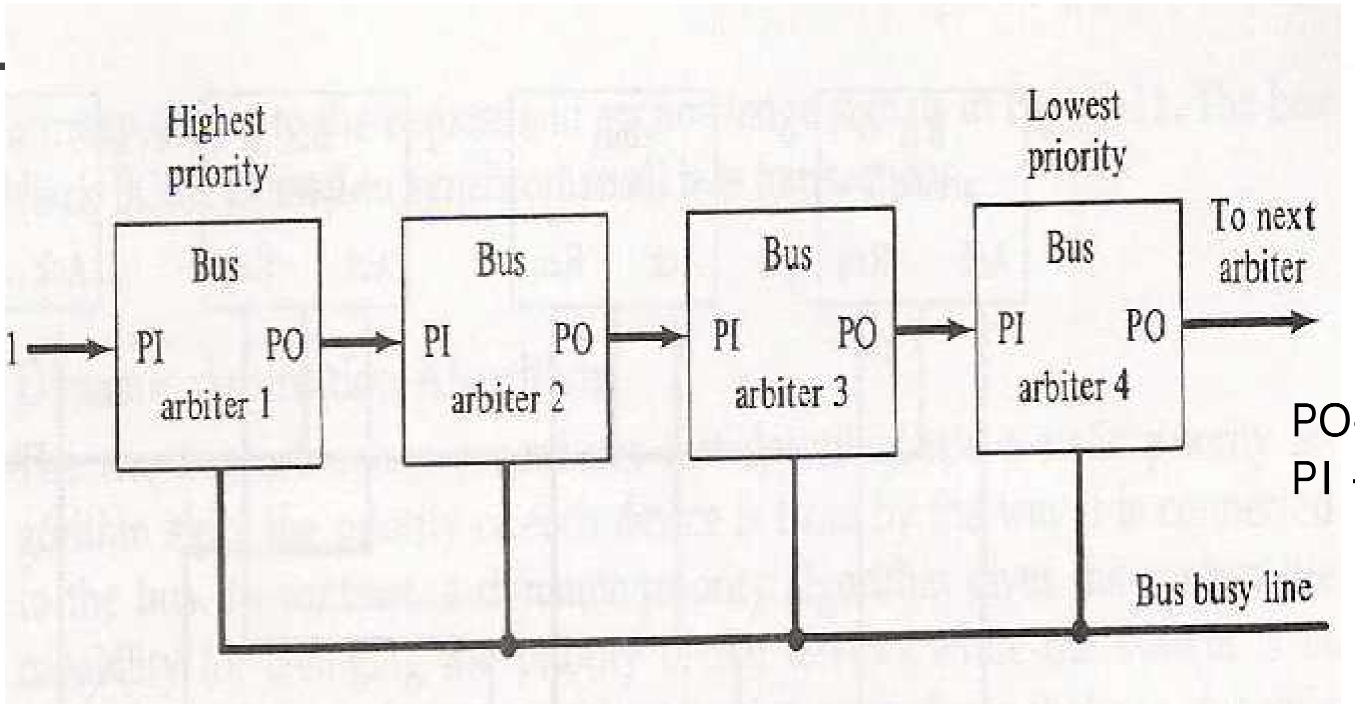


Figure Serial (daisy-chain) arbitration.

PO-priority out
PI -priority in

- ❖ Daisy chain is a wiring scheme in which multiple devices are wired together in sequence.
- ❖ The higher priority device will pass the grant line to the lower priority device only if it does not want to use the bus.
- ❖ Then priority is forwarded to the next in the sequence.

WORKING

- The interrupt request line which is common to all the devices and CPU.
- When there is no interrupt, the interrupt request line (IRL)is in HIGH state.
- A device that raises an interrupt places the IRL in the LOW state.
- The CPU acknowledges this interrupt request from the line and then enables the interrupt acknowledge line in response to the request.
- This signal is received at the PI (Priority in) input of device 1.
- If the device has not requested the interrupt, it passes this signal to the next device through its PO (priority out) output. (PI = 1 & PO = 1)
- However, if the device had requested the interrupt, (PI =1 & PO = 0)
  - The device consumes the acknowledge signal and block its further use by placing 0 at its PO (priority out) output.
  - The device then proceeds to place its interrupt vector address (VAD) into the data bus of CPU.
  - The device puts its interrupt request signal in HIGH state to indicate its interrupt has been taken care of.

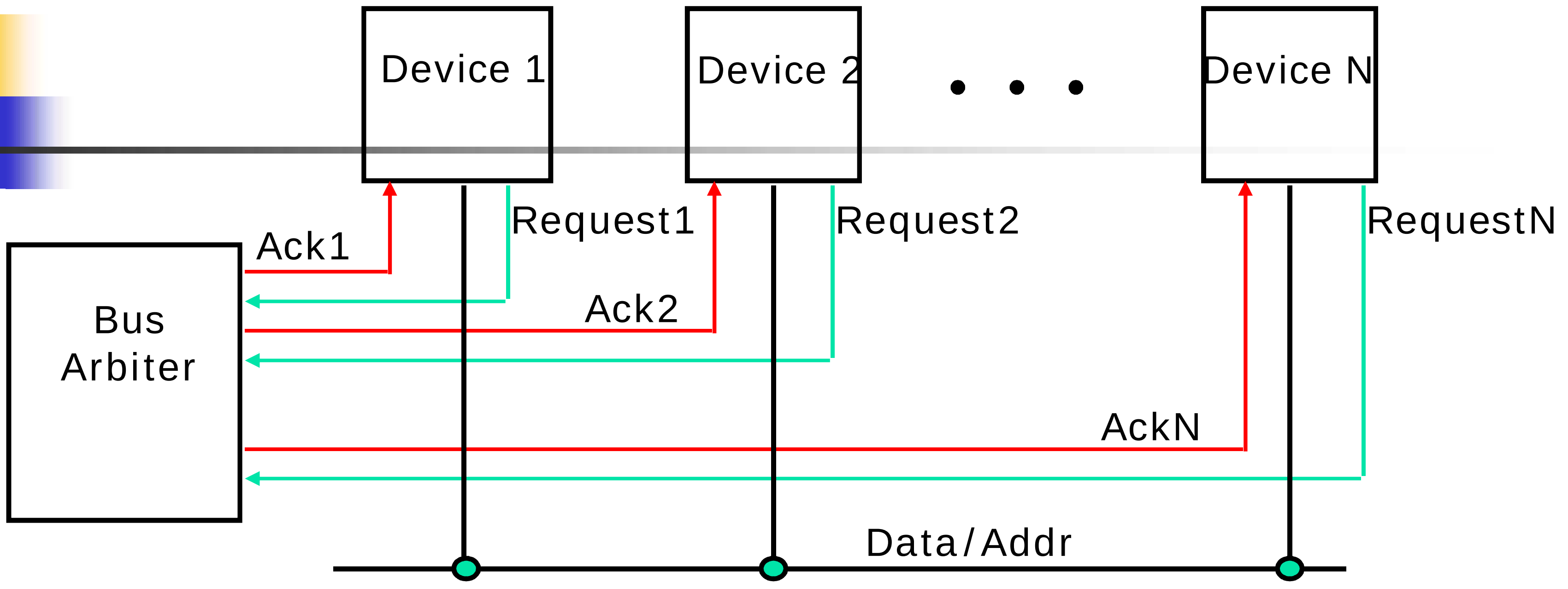
NOTE:

- Interrupt vector address (VAD) is the address of the service routine which services that device.
- If a device gets 0 at its PI input, it generates 0 at the PO output to tell other devices that acknowledge signal has been blocked. (PI = 0 & PO = 0)
- Hence, the device having PI = 1 and PO = 0 is the highest priority device that is requesting an interrupt.
- Therefore, by daisy chain arrangement ensures that the highest priority interrupt gets serviced first and have established a hierarchy.
- The farther a device is from the first device, the lower its priority.

Parallel Arbitration Logic

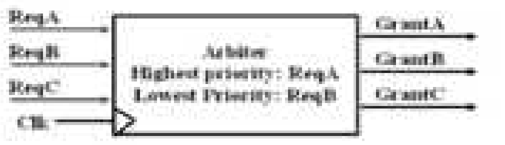
- It uses an external priority encoder and decoder.
- Each bus arbiter has a bus request output lines and a bus acknowledge input lines.
- Each arbiter enables request lines when its processor is requesting the system bus.
- The one with highest priority determine by the output of the decoder get access to the bus.

Centralized Parallel Arbitration



- Advantages: flexible, can assure fairness
- Disadvantages: more complicated arbiter hardware
- Used in essentially all processor-memory buses and in high-speed I/O buses

Centralized Arbitration (Parallel Arbitration)



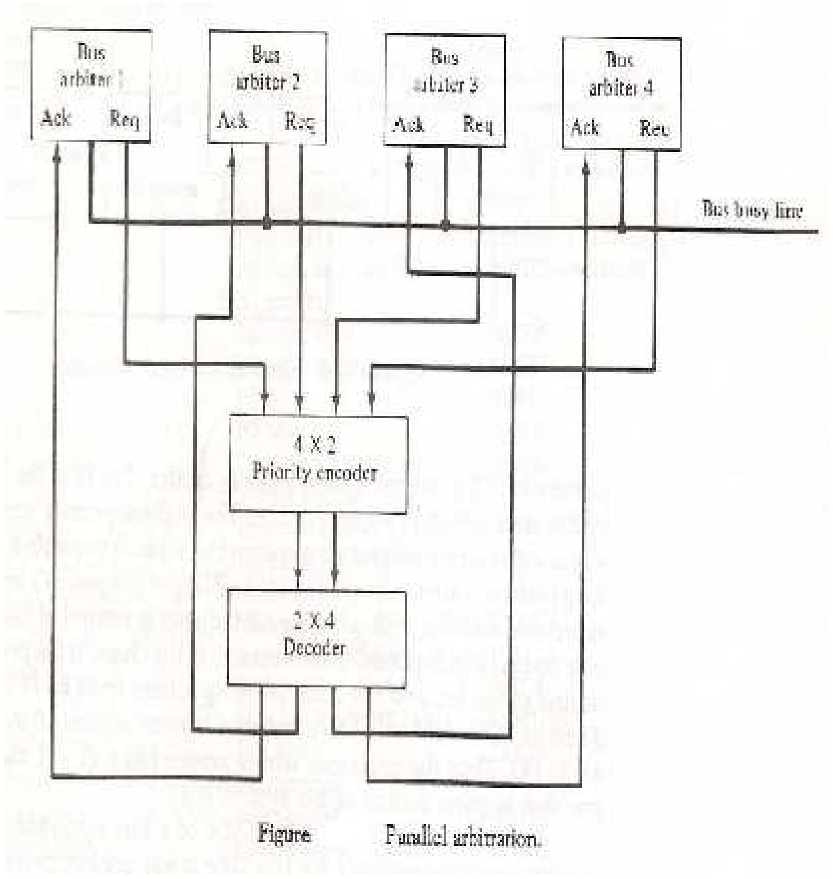


Figure Parallel arbitration.

- The devices independently request the bus by using multiple request lines.
- A centralized arbiter chooses from among the devices requesting bus access and notifies the selected device that it is now the bus master via one of the grant line.
- Consider: A has the highest priority and C the lowest. H

How bus is granted to different bus masters?

- Since A has the highest priority, Grant A will be asserted even though both requests A and B are asserted.
- Device A will keep Request A asserted until it no longer needs the bus so when Request A goes low, the arbiter will disable Grant A.
- Since Request B remains asserted (Device B has not gotten the bus yet) at this time, the arbiter will then assert Grant B to grant Device B access to the bus.
- Similarly, Device B will not disable Request B until it is done with the bus.

Priority Encoder (or priority decoder)

- Priority encoders are typically used when a set of components (e.g., processor, memory, I/O devices, etc.) are to share a common resource (e.g., a bus).
- Each component is assigned a certain priority according to its nature (importance), so that when more than one components request the resource, the one with the highest priority will be granted the usage.
- The most significant bit of the input corresponds to the highest priority while the least significant bit the lowest priority.
- All inputs with lower priorities are ignored.
- The output represents the component granted with the usage of the resource.
- Output index number is either binary encoded y1y0 or one-hot encoded by ignoring all lower priorities input .(treated as don't cares).

x_3	x_2	x_1	x_0	y_1	y_0	z_3	z_2	z_1	z_0
1	X	X	X	1	1	1	0	0	0
0	1	X	X	1	0	0	1	0	0
0	0	1	X	0	1	0	0	1	0
0	0	0	X	0	0	0	0	0	1

Dynamic Arbitration Algorithm

- Serial and parallel bus arbitration are static since the priorities assigned is fixed.
- In dynamic arbitration priorities of the system change while the system is in operation.
- The various algorithms used are:
  - Time Slice
  - Polling
  - Least Recently Used (LRU)
  - First In First Out (FIFO)
  - Rotating Daisy Chain

Inter processor Communication

- All the processors in the multi processor system need to communicate with each other.

Problems inherent in Inter processor Communication

- ➢Starvation
- ➢Deadlock
- ➢Data Inconsistency
- ➢Shared Buffer Problems

i) The most common way is to set aside portion of memory that is accessible to all processor (common memory)

- ✓ Sending processor puts the data and the address of the receiving processor in the memory.
- ✓ All the processor periodically check the memory for any information.
- ✓ If they find their address they read the data.
- ✓ This procedure is time consuming

- ii)Use of interrupt facility

To send the interrupt signal to the receiving processor whenever the sending processor leaves the message.

- In addition to shared memory, multiprocessor system may have other shared resources.
- Two primary forms of data exchange between parallel tasks - accessing a shared data space and exchanging messages.
- Platforms that provide a shared data space are called shared-address-space machines or tightly coupled systems or multiprocessors.
- Platforms that support messaging are also called message passing platforms or multi-computers or loosely coupled systems (Distributed memory)
- To prevent the conflicting use of shared resources by several processor there must be provision for assigning resources to processor.
- This task is handled by the Operating System.

OS for Multiprocessors

- There are three organization that have been used in design of OS of multiprocessor:
  - Master-Slave Configuration
  - Separate OS
  - Distributed OS

i) Master slave configuration

- One processor designated as master execute OS function.
- If slave processor needs OS service, it must place request to the master processor.

ii) Separate OS

- Every processor have its own copy of the entire OS.
- Each processor can execute the OS routine as and when it needed.
- This organization is suited for loosely coupled system.

iii) Distributed OS

- The OS routines are distributed among the processors.
- Each particular OS function is assigned to only one processor at a time.
- Since the routines float from one processor to another it is also called floating OS.

Inter processor Synchronization

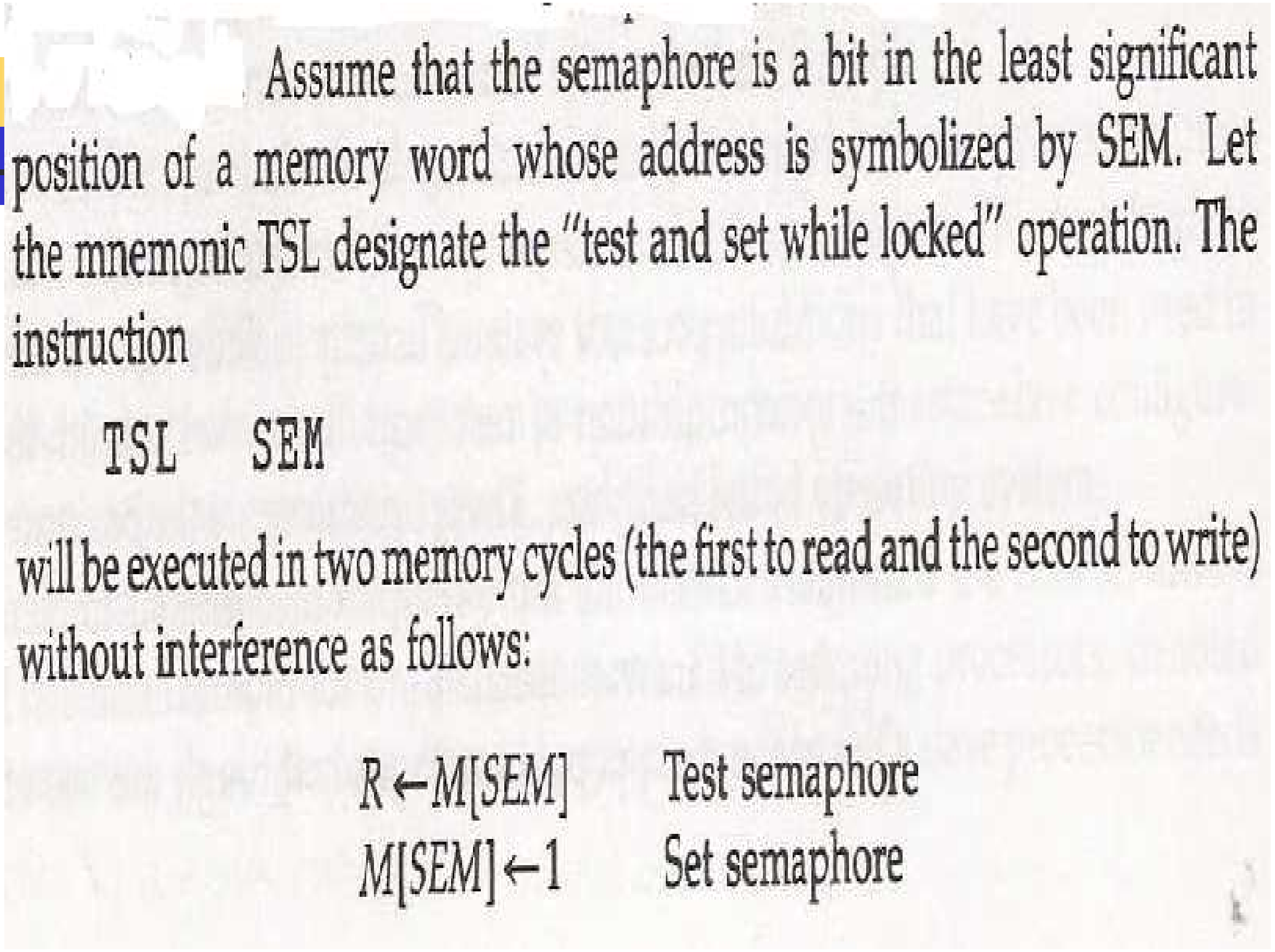
- Synchronization is a special case of communication where data used to communicate between processors is a control information.
- It is needed to enforce correct sequence of process and to ensure mutually exclusive access to shared writable data.
- A number of hardware mechanisms for mutual exclusion have been developed.
- One of the most popular is through the use of binary semaphore.

Mutual Exclusion with Semaphore

- Proper functioning multi-processor should guarantee orderly access to shared resources. So that data can not be changed simultaneously by two processor.
- This is called mutual exclusion.
- Mutual exclusion must be provided to enable one processor to lock out access to shared resources once it entered in critical section.
- Critical section is a program sequence, that must be completed once begun.i.e. set of instructions that must be controlled so as to allow exclusive access to one process.
- Execution of the critical section by processes is mutually exclusive in time.

- Binary variable semaphore is used to indicate whether the processor is in critical section.
- Semaphore is software controlled flag stored in common memory.
- If it is 1, processor is executing critical section. If it is 0, it indicate common memory is available for the other processor.
- Testing and setting semaphore is itself a critical task and must be performed in single indivisible task.

- A semaphore can be initialized by means of test and set instruction in conjunction with a hardware lock mechanism.
- A hardware lock is processor generated signal that prevent other processor from using system bus as long as signal is active.
- Test and set instruction tests and sets semaphore and activate lock mechanism during the time that it is testing and setting it.



Assume that the semaphore is a bit in the least significant position of a memory word whose address is symbolized by SEM. Let the mnemonic TSL designate the "test and set while locked" operation. The instruction

```
TSL   SEM
```

will be executed in two memory cycles (the first to read and the second to write) without interference as follows:

$R \leftarrow M[SEM]$	Test semaphore
$M[SEM] \leftarrow 1$	Set semaphore

- The last instruction in the program must be clear location SEM to 0 to released shared resource to other processor.
- Lock signal must be active during execution of test and set instruction.
- It does not have to be active once semaphore is set.

Using binary Semaphore p1 or queue semaphore p2

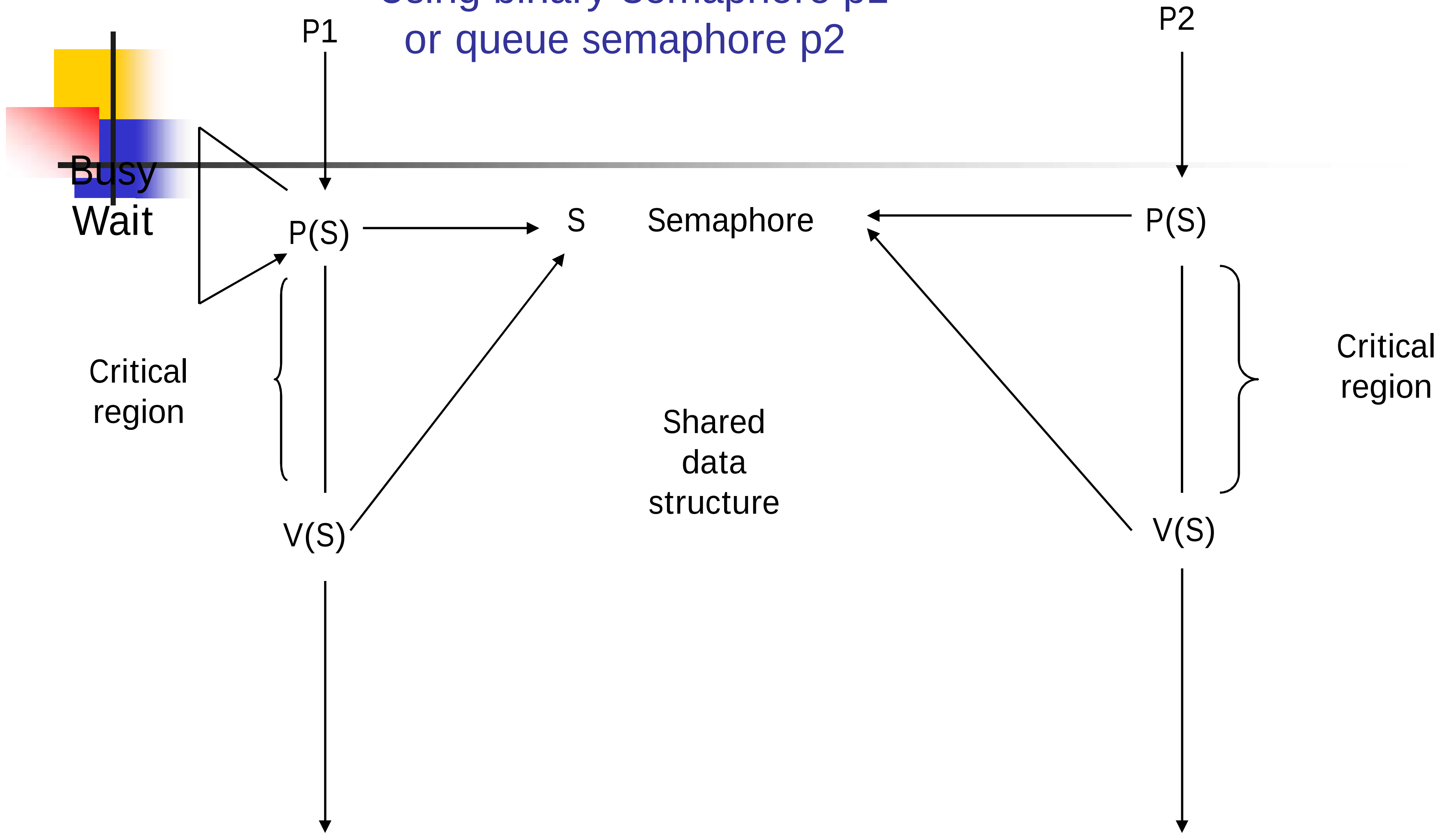


Figure . Using a semaphore to solve the mutual execution problem

Cache Coherence

- In shared memory multi-processor system it is customary to have one or two levels of cache associated with each processor.
- Caches in such machines require coordinated access to multiple copies as read-write data to shared data must be coordinated .
- This organization is essential to achieve high performance.
- Cache creates a problem which is known as the cache coherence problem.
- The cache coherence problem is: Multiple copies of the same data can exist in different caches simultaneously, and if processors are allowed to update their own copies freely, an inconsistent view of memory can result.
- To ensure ability of the system to execute memory instruction independently, multiple copies of the data must be identical which is called cache coherence.

Condition to cache Incoherence

- This condition arise when the processor need to share the writable data.
- In both policy write back and write through incoherence condition is created.
- Write back: Write operations are usually made only to the cache. Main memory is only updated when the corresponding cache line is flushed from the cache.
- Write through: All write operations are made to main memory as well as to the cache, ensuring that main memory is always valid

Cache Coherence

- In shared memory multi-processor system, processor share memory and they have local memory(part or all of which is cache).
- To ensure ability of the system to execute memory instruction independently, multiple copies of the data must be identical which is called cache coherence.

local memories

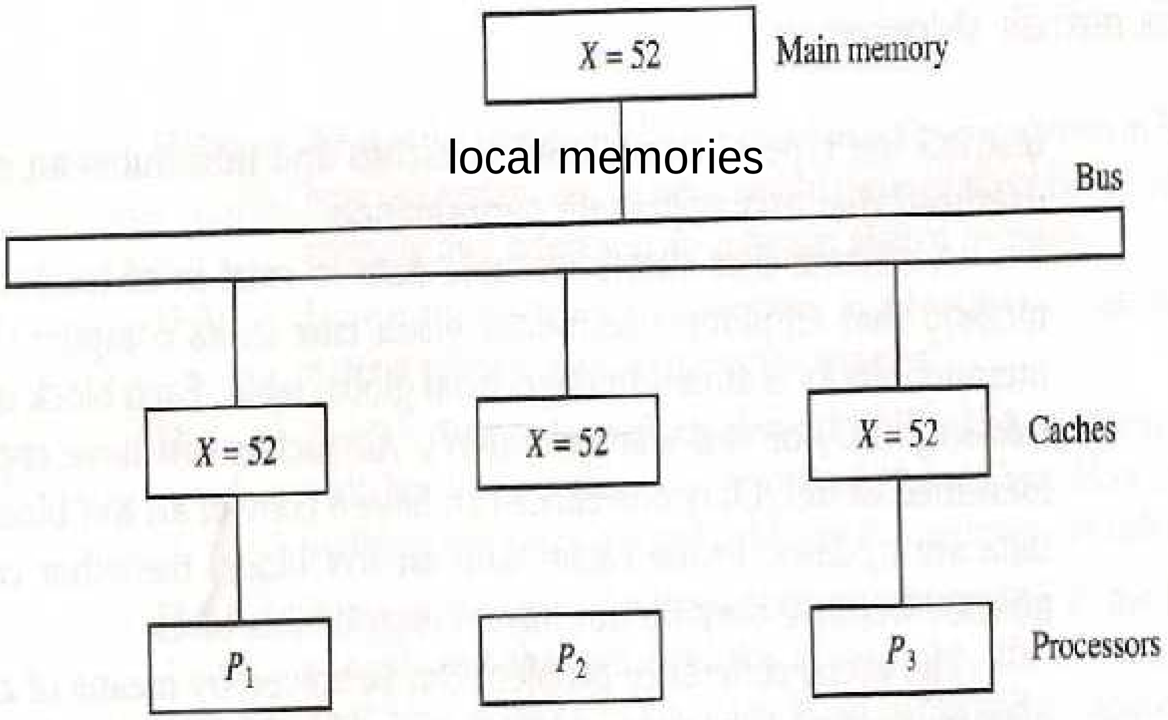
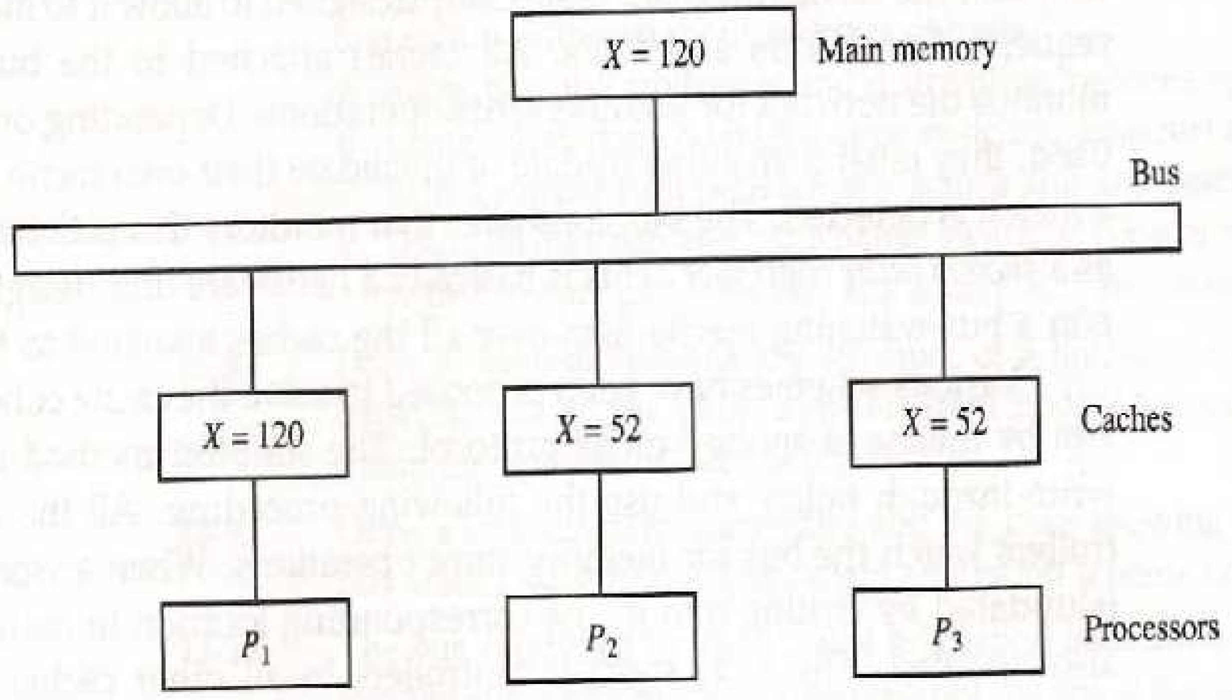


Figure Cache configuration after a load on X.

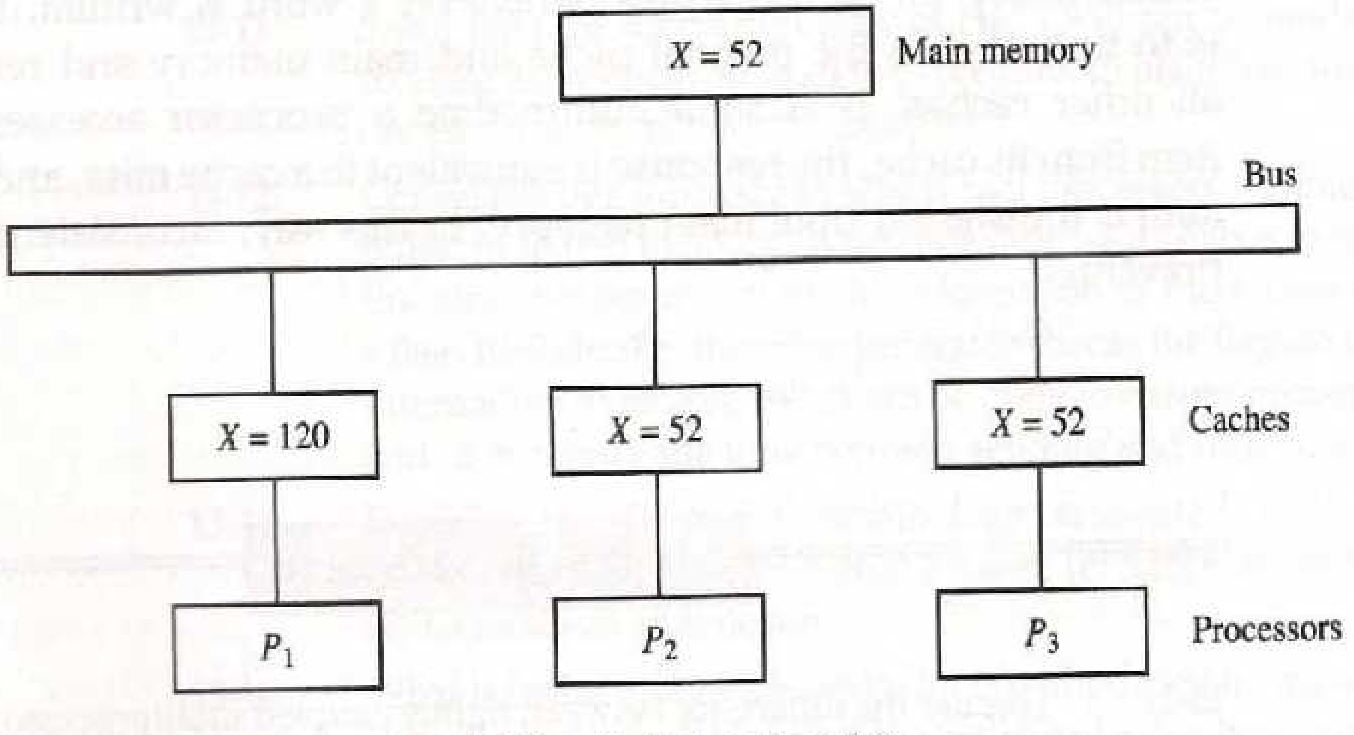
Cache Coherence due to Write through policy

Figure Cache configuration after a store to X by processor P_1.



With write-through cache policy

Cache Coherence due to Write back policy



With write-back cache policy

Solution using Software

- A simple way is to disallow to have private caches and use the common memory for shared resources.
- Another way is to cache only the read only data in the local cache and use the common memory for writable data.
- A scheme that allows writable data to reside in only one cache .
- A centralized global table is used to maintain details:
  - Status of each memory block as read only (RO) or read write (RW) is stored.
  - All local cache can store RO memory blocks.
  - RW memory blocks are stored only in one cache that is shared by all.

Hardware Solution

Snoopy cache controller

- A hardware specially designed to watch the system bus for write operation.
- When a word is updated in cache, main memory is also updated.
- Local snoopy controller watch the cache if they have the address updated.
  - If the copy exist, it is marked invalid.
  - If CPU request the same address, it is fetch from the main memory and updated in cache.