MODULE 3

MULTI PROCESSORS & MULTI COMPUTERS

Multi processor System Interconnects

- Parallel processing needs the use of efficient system interconnects for fast communication between multiple processors and shared memory, I/0, and peripheral devices.
- Commonly used interconnects are Hierarchical buses, crossbar switches and multistage networks
- Dynamic networks are used in multiprocessors in which the interconnections are under program control.
- The three major operational characteristics of an interconnection network are

Timing , **Switching** and **Control**

- Timing control can be synchronous or asynchronous
- Synchronous networks are controlled by a global clock that synchronizes all network activities.
- Asynchronous networks use handshaking or interlocking mechanisms to coordinate fast and slow devices
- A network can transfer data using either circuit-switching or packet –switching.
- In circuit switching, once a device is granted a path in the network, it occupies the path for the entire duration of the data transfer

- In packet switching, the information is broken into small packets individually competing for a path in the network.
- Network control strategy is classified as centralized or distributed.
- With centralized control, a global controller receives requests from all devices attached to the network and grants the network access to one or more requesters.
- In a distributed system, requests are handled by local devices independently

Hierarchical Bus Systems

- Buses connects various system and subsystem components in a computer.
- Each bus is formed with a number of signal, control, and power lines.
- Different buses are used to perform different interconnection functions.
- Different levels of bus systems are local buses on boards, back plane buses, and I/O buses

Local Bus

- Buses implemented Within processor chips or on printed circuit board
- It provides a common communication path among major components (chips) mounted on the board.
- A memory board uses a memory bus. An I/O or network interface chip or board uses a data bus.

Backplane Bus

- It is a printed circuit on which many connectors are used to plug in functional boards
- Eg: VME bus, Multibus II, Futurebus+

I/O Bus

- Input/output devices are connected to a computer system through an I/O bus such as the SCSI (Small Computer Systems Interface) bus.
- This bus is made of coaxial cables with taps connecting disks, printer and other devices to a processor through an I/O controller

- IF (Interface logic)
- LM (Local Memory)
- IOC (I/O Controller)
- MC (Memory Controller)
- IOP (I/O Processor)
- CC (Communication Controller)

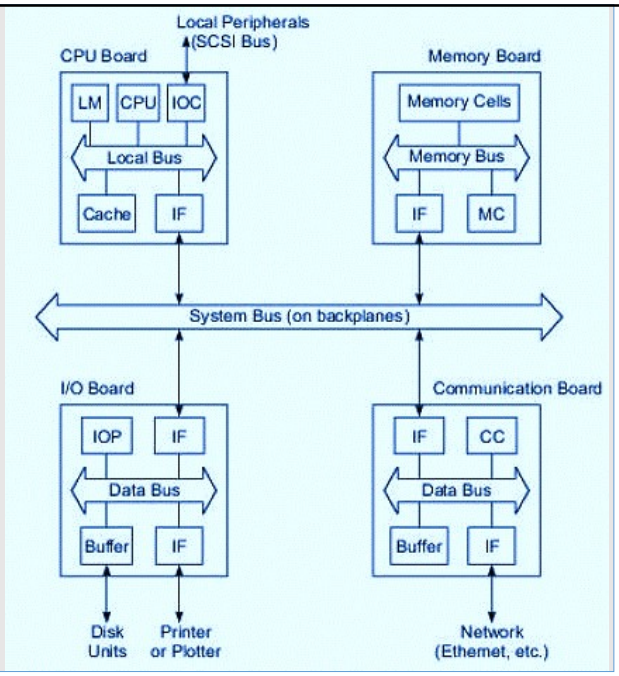
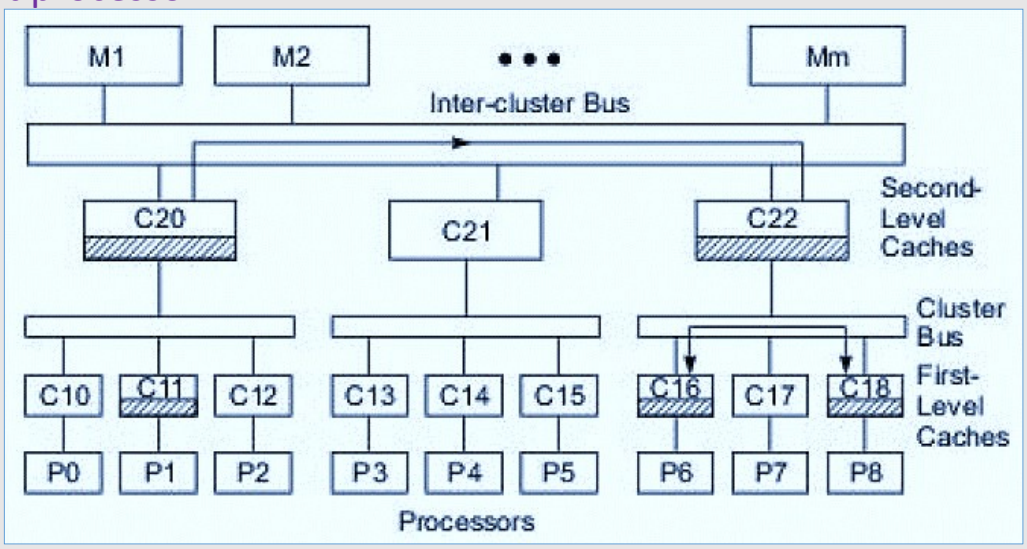


Fig: Bus systems at different levels

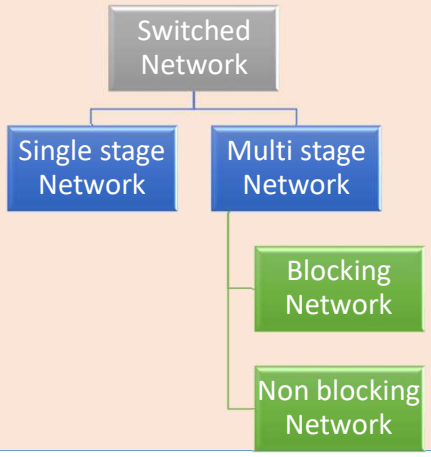
❖A hierarchical cache/bus architecture for designing a scalable multiprocessor



Prepared By Mr.EBIN PM, AP, IESCE EDULINE 9

Crossbar Switch & Multiport Memory

- Switched networks provide dynamic interconnections between the inputs and outputs



Prepared By Mr.EBIN PM, AP, IESCE EDULINE 10

➢Singlestage Network

- Also called recirculating network , because data items recirculate many times before reaching the destination
- Eg: Cross bar switch , Multiport memory

➢Multistage Network

- It has more than one stage of switch boxes
- It should be able to connect from any input to any output
- Eg: Omega Network, Flip Network, Baseline Network
- It can be divided in to 2. Blocking Network and Non blocking Network

Blocking Network

- Here, simultaneous connections of some multiple input-output pairs may result in conflicts in the use of switches or communication links.
- Eg: Omega, Baseline, Delta network
- Most multistage networks are blocking in nature

Nonblocking Network

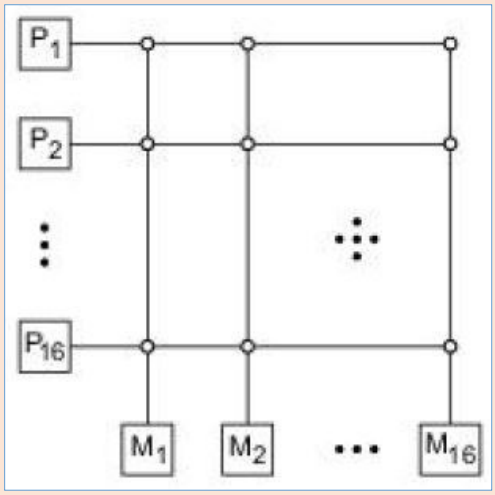
- It can perform all possible connections between inputs and outputs by rearranging its connections.
- Eg: Benes network, Clos Network

Switch Modules

- An a×b switch module has a inputs and b outputs
- A binary switch corresponds to a 2×2 switch module in which a=b=2
- Commonly used switch modules are 2×2, 4×4 and 8×8
- Each input can be connected to one or more of the outputs. That is, one-to-one and one-to-many mappings are allowed.
- Many-to-one mapping not allowed due to the conflicts at the output terminal
- A 2×2 crossbar switch can connect 2 possible patterns. I.e., straight and crossover.

- Every input port is connected to a free out put port through a cross point switch without blocking
- It is a single stage network
- Small circles in each cross point is a switch
- Support simultaneous transfer from all memory modules
- Each cross point switch requires the use of a large number of connecting lines accommodating address, data path and control signals

Crossbar networks



Limitations of Crossbar Network

- Crossbar network is cost effective only for small multiprocessors
- A single stage crossbar network is not expandable once it is built
- Redundancy or parity check lines can be built in to each cross point switch to enhance the fault tolerance and reliability of the cross bar network.

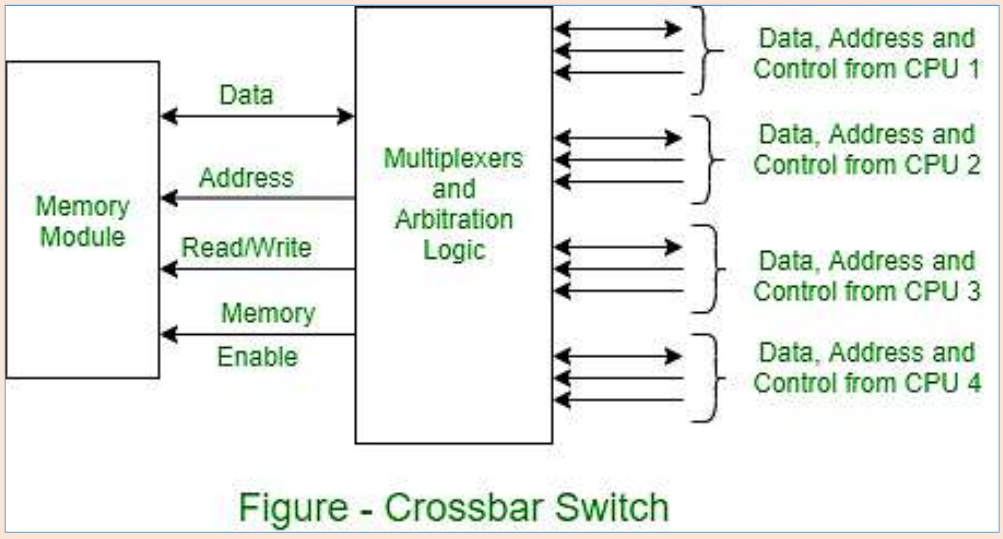


Figure - Crossbar Switch

- Requires the most expensive memory units since most of the control and switching circuitry is included in the memory unit
- A large number of cable and connections are required
- Memory access conflicts are resolved by assigning fixed priorities to each memory port
- There is a potential for a very high total transfer rate in the overall system.

Multiport Memory

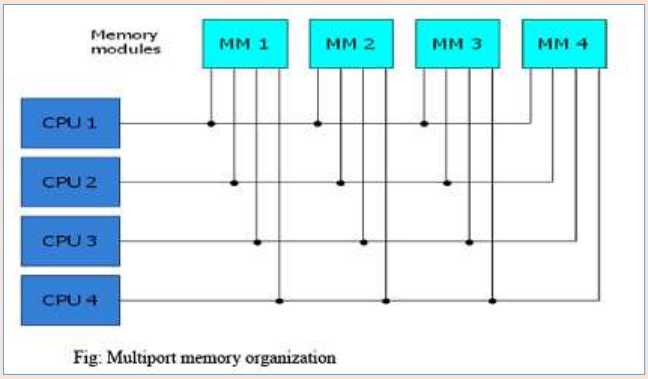


Fig: Multiport memory organization

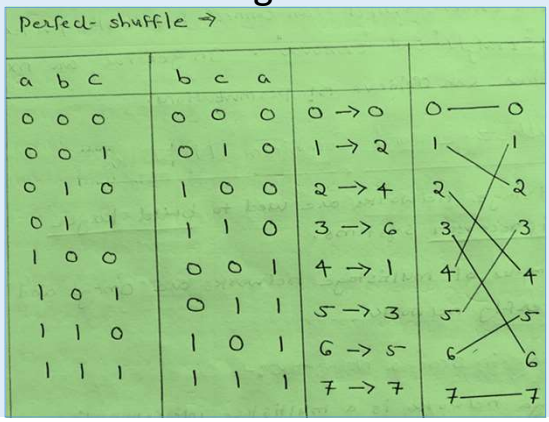
Multistage and Combining Networks

- Multistage networks are used to build larger multiprocessor systems.
- Examples of multistage networks are Omega and Butterfly network.

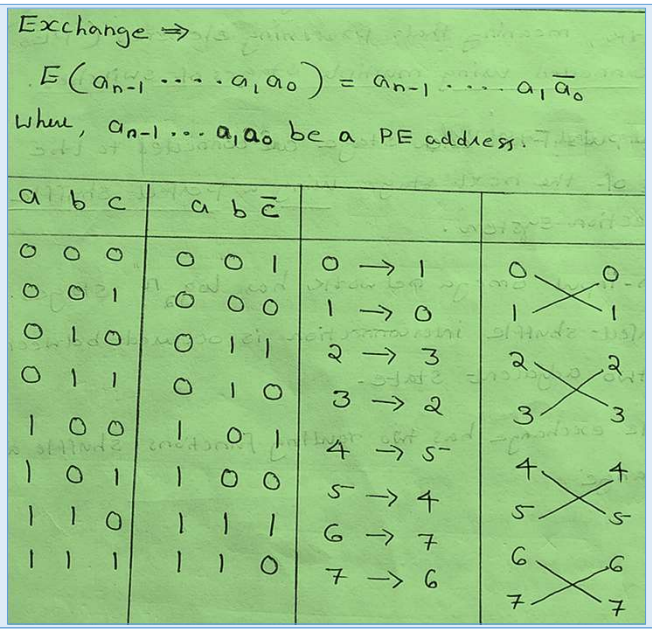
Routing in Omega Network

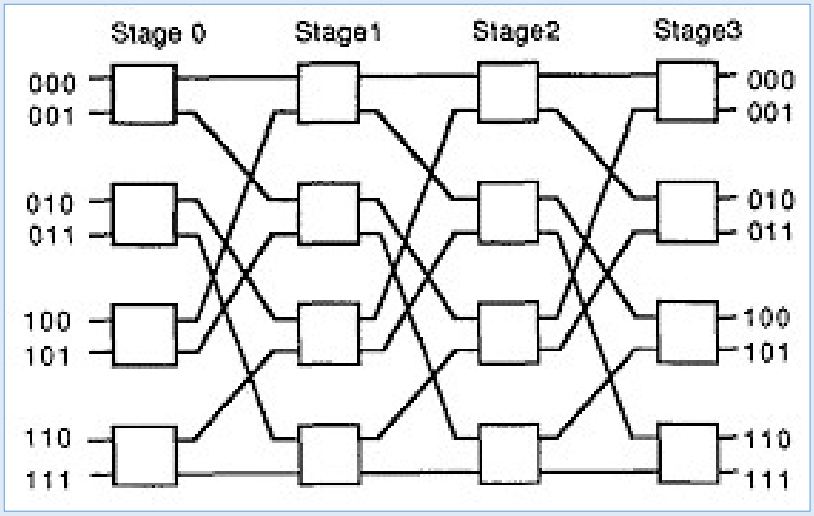
- Omega network is a multistage interconnection network, meaning that processing elements(PEs) are connected using multiple stages of switches
- The output from each stage are connected to the inputs of the next stage using a perfect shuffle connection system.

- An n-input omega network has $\log_2 n$ stages
- A perfect shuffle interconnection is occurred between the two adjacent state.
- Shuffle exchange has two routing functions shuffle and exchange



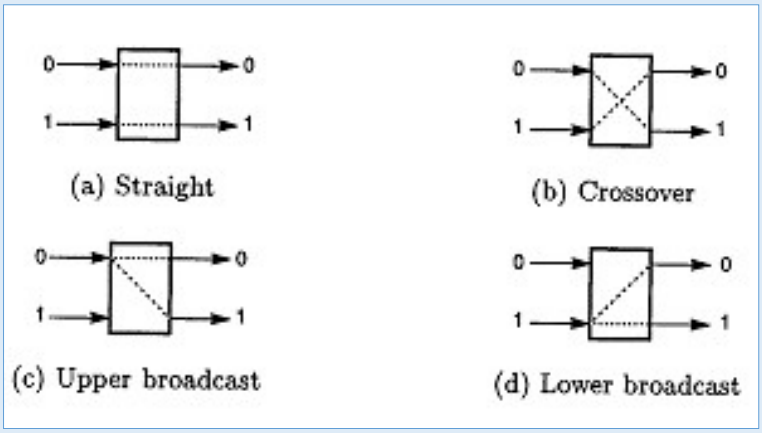
- **Exchange**





Prepared By Mr.EBIN PM, AP, IESCE EDULINE 21

The following figure shows four possible connections of 2×2 switches used in constructing the omega network



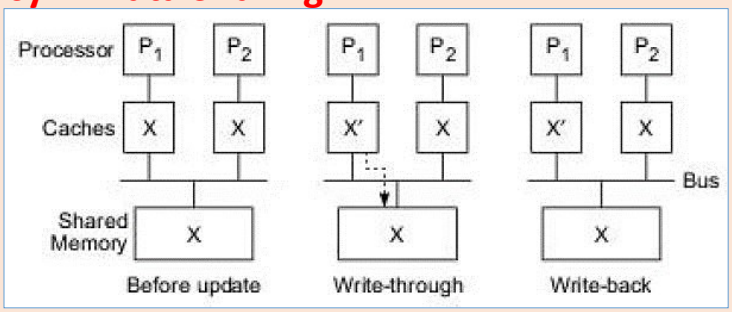
Prepared By Mr.EBIN PM, AP, IESCE EDULINE 22

Cache Coherence Problem

- Caches in a multiprocessing environment introduce the cache coherence problem.
- It is a cache inconsistency problem
- When multiple processors maintain locally cached copies of a unique shared-memory location, any local modification of the location can result in a globally inconsistent view of memory.
- Multiple caches may possess different copies of the same memory block because multiple processors operate asynchronously and independently.

- Cache inconsistencies caused by data sharing, process migration, or I/O

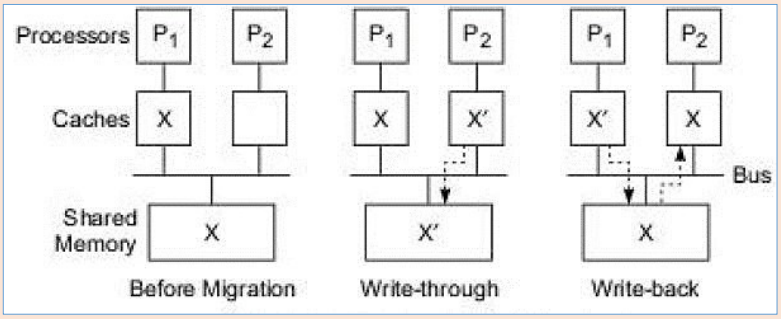
➢ **Inconsistency in Data Sharing**



- Consider a multiprocessor with two processors, each using a private cache and both sharing the main memory.

- Let X be a shared data element which has been referenced by both processors. Before update, the three copies of X are consistent.
- If processor P_1 writes new data X' into the cache, the same copy will be written immediately into the shared memory under a write-through policy.
- In this case inconsistency occurs between the two copies (X'and X) in the two caches
- inconsistency may also occur when a write-back policy is used
- The main memory will be eventually updated when the modified data in the cache are replaced or invalidated

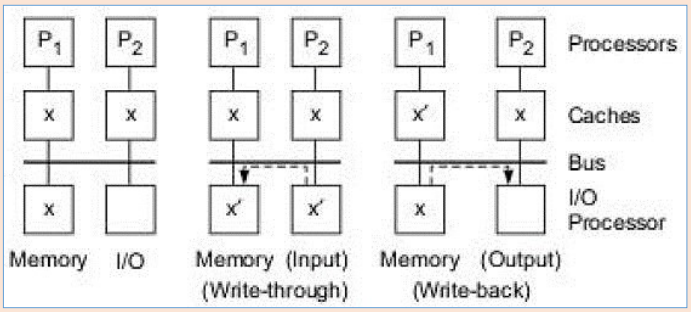
➢Inconsistency in Process Migration



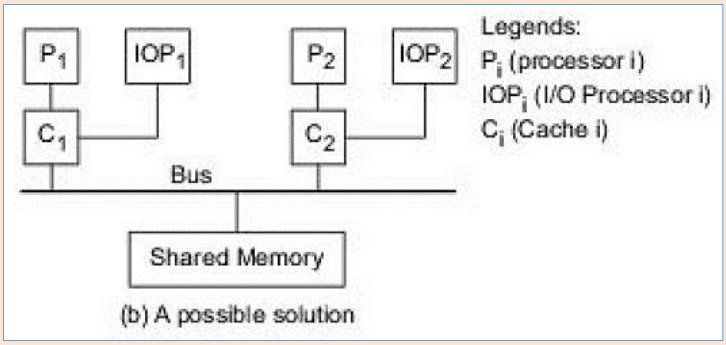
- A coherence protocol must be established before processes can safely migrate from one processor to another.

➢Inconsistency in I/O Activity

- Inconsistency problems may occur during I/O operations that bypass the caches.
- when the I/O processor loads a new data X' into the main memory, bypassing the write through caches, inconsistency occurs between cache 1 and the shared memory



- When outputting a data directly from the shared memory (bypassing the caches), the write-back caches also create inconsistency.
- One possible solution to the I/O inconsistency problem is to attach the I/O processors (IOP_1 and IOP_2) to the private caches (C_1 and C_2), respectively



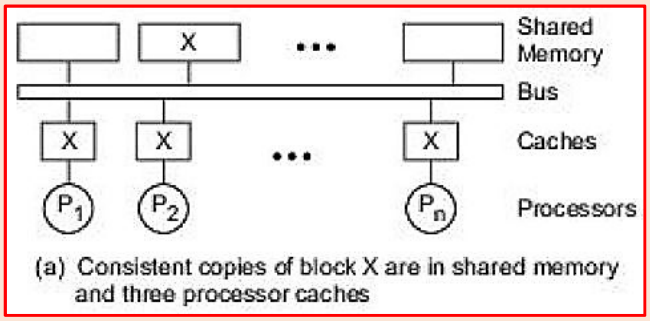
SNOOPY BUS PROTOCOLS

- commercially available multiprocessors used bus-based memory systems.
- Bus ensure cache coherence because it allows all processors to observe ongoing memory transactions.
- If a bus transaction threatens the consistent state of a locally cached object, the cache controller can take appropriate actions to invalidate the local copy.
- Protocols using this mechanism to ensure coherence are called snoopy protocols because each cache snoops on the transactions of other caches.

- Snoopy protocols achieve data consistency among the caches and shared memory through a bus watching mechanism.

➢Write invalidate and Write update protocols are used for maintaining cache consistency.

➢Consider three processors (P_1, P_2, and P_3) maintaining consistent copies of block X in their local caches and in the shared-memory module.

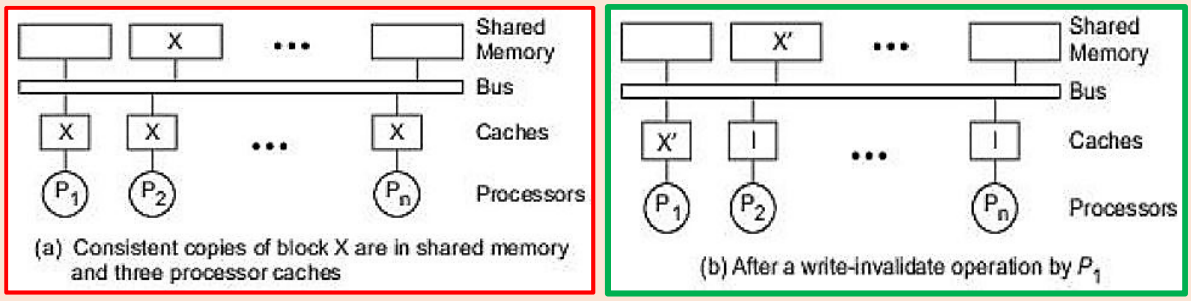


(a) Consistent copies of block X are in shared memory and three processor caches

❖ Write Invalidate Protocol

- the processor P_1 modifies (writes) its cache from X to X', and all other copies are invalidated (I) via the bus.
- invalidated blocks are sometimes called dirty, meaning they should not be used.

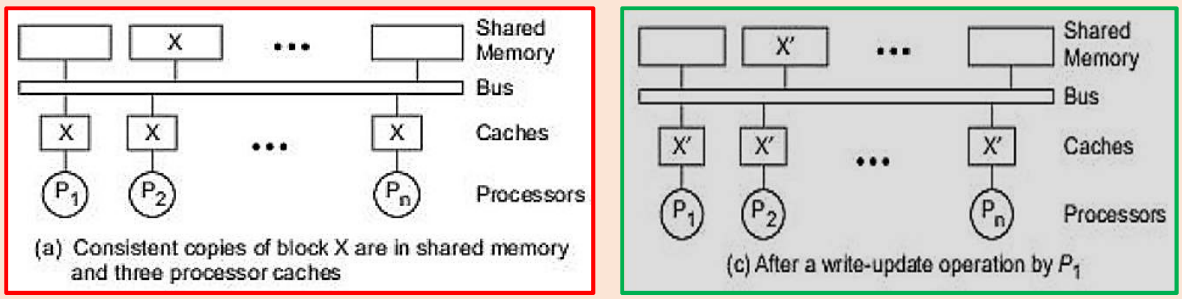
Consistent state | **After write-invalidate**



Prepared By Mr.EBIN PM, AP, IESCE EDULINE 31

❖ Write Update Protocol

- The new block content X' is broadcast to all cache copies via the bus. The memory copy is also updated if write-through caches are used.
- In using write-back caches, the memory copy is updated later at block replacement time.



Prepared By Mr.EBIN PM, AP, IESCE EDULINE 32

- In snoopy protocol , transactions on bus are visible to all processors
- Processors or bus-watching mechanisms can snoop(monitor) the bus and take action on relevant events (eg. Change state) to ensure data consistency among private caches and shared memory.

❖ Write Once Protocol

- James Goodman proposed this protocol and it is the first snoopy cache protocol
- Multiple memory readers allowed simultaneously , not multiple writers
- The 4 cache states are

1. Valid – cache block is consistent with memory copy. Has been read from shared memory and has not been modified.
2. Invalid - The block is not found in the cache or is inconsistent with the memory copy.

3. Reserved – The block is the only copy of the memory, but it is still coherent. No write back is needed if the block is replaced

4. Dirty – The cache block has been modified more than once, and it is coherent. This copy was written one or more times.

❖ cache events and actions

➢Read miss

- When a processor wants to read a block that is not in the cache, a read-miss occurs
- A bus read operation will be initiated.
- If no dirty copy exist, then main memory has a consistent copy and supplies a copy to the requesting cache

- If a dirty copy exist in a remote cache, that cache will inhibit the main memory and send a copy to the requesting cache

➢Write miss

- When a processor fails to write in a local cache, the copy must come either from the main memory or from a remote cache with a dirty block
- This is accomplished by sending a read-invalidate command which will invalidate all cache copies.
- The local copy is thus updated and ends up in a dirty state

➢Read hit

- Read-hits can always be performed in a local cache without causing a state transition or using the snoopy bus for invalidation.

➢Write hit

- If the copy is in the dirty or reserved state, the write can be carried out locally and the new state is dirty
- If the new state is valid, a write invalidate command is broadcast to all caches, invalidating their copies.
- The shared memory is written through, and the resulting state is reserved after this first write.

➢Block replacement

- If a copy is dirty, it has to be written back to main memory by block replacement
- If the copy is clean, no replacement will take place