Module-4

7 a. Explain routing in omega network. (08 Marks)

b. What are different vector – access memory schemes? Explain any two of them. (08 Marks)

OR

8 a. What are the implementation models of SIMD? Explain them. (08 Marks)

b. Explain four context-switching policies. (08 Marks)

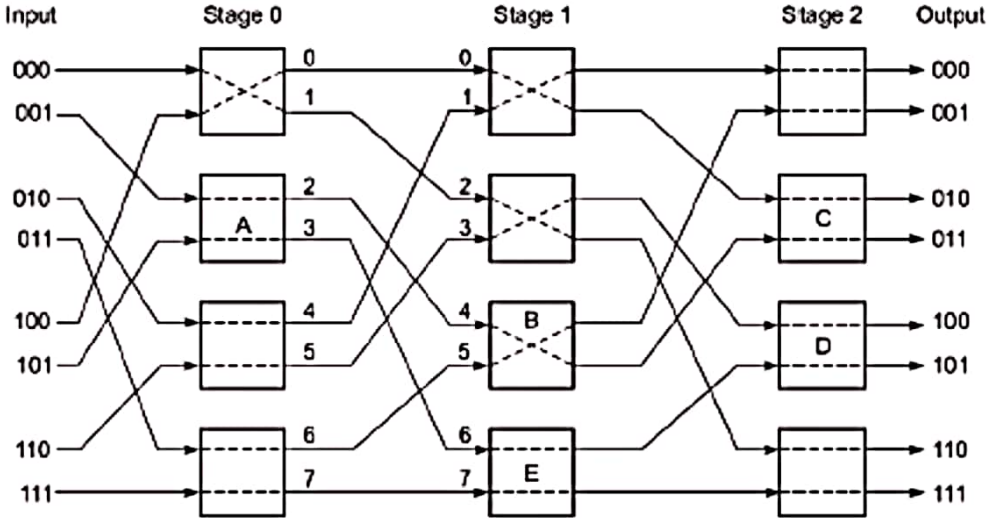
7A)

Routing in Omega Networks

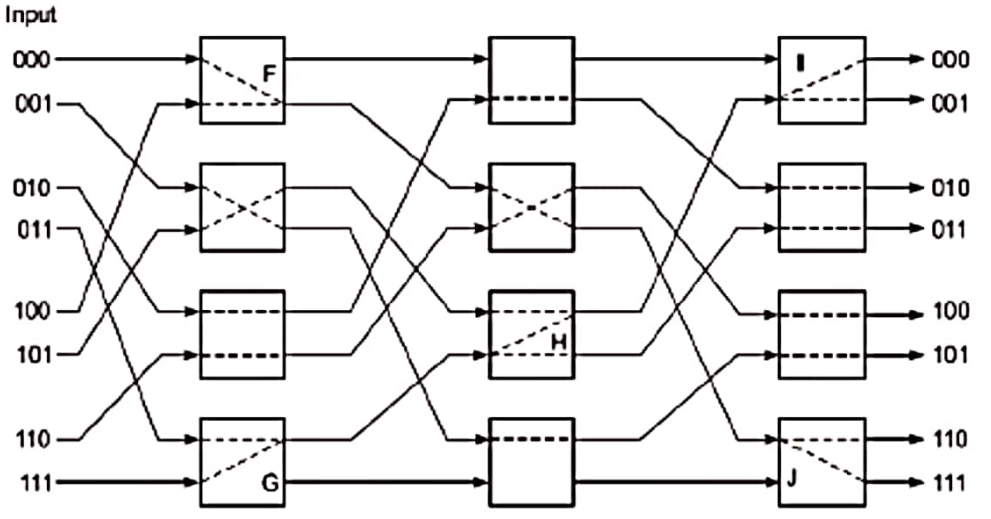
An 8-input Omega network is shown in Fig. 7.8.

In general, an n-input Omega network has **$\log_2 n$** stages. The stages are labeled from 0 to $\log_2 n$ — 1 from the input end to the output end.

Data routing is controlled by inspecting the destination code in binary. When the ith high-order bit of the destination code is a 0, a 2 x 2 switch at stage *i* connects the input to the upper output. Otherwise, the input is directed to the lower output.



(a) Permutation π_1 = (0, 7, 6, 4, 2) (1, 3) (5) implemented on an Omega network without blocking



(b) Permutation π_2 = (0, 6, 4, 7, 3) (1, 5) (2) blocked at switches marked F, G, and H

Fig. 7.8 Two switch settings of an 8 × 8 Omega network built with 2 × 2 switches

- Two switch settings are shown in Figs. 7.8a and b with respect to permutations Π_1 = (0,7,6,4,2) (1,3)(5) and Π_2= (0,6,4,7,3) (1,5)(2), respectively.
- The switch settings in Fig. 7.8a are for the implementation of Π_1, which maps 0 →7, 7 →6, 6→4, 4→2, 2 →0, 1 →3, 3 →1, 5 →5.
- Consider the routing of a message from input 001 to output 011. This involves the use of switches A, B, and C. Since the most significant bit of the destination 011 is a "zero," switch A must be set straight so that the input 001 is connected to the upper output (labeled 2).

- The middle bit in 011 is a "one," thus input 4 to switch B is connected to the lower output with a "crossover" connection.
- The least significant bit in 011 is a "one," implying a flat connection in switch C.
- Similarly, the switches A, E, and D are set for routing a message from input 101 to output 101. There exists no conflict in all the switch settings needed to implement the permutation Π_1 in Fig. 7.8a.

- Now consider implementing the permutation Π_2 in the 8-input Omega network (Fig. 7.8b0. Conflicts in switch settings do exist in three switches identified as F, G, and H. The conflicts occurring at F are caused by the desired routings 000→ 110 and 100→ 111.

7B)

8.1.2 Vector-Access Memory Schemes

The flow of vector operands between the main memory and vector registers is usually pipelined with multiple access paths.

Vector Operand Specifications

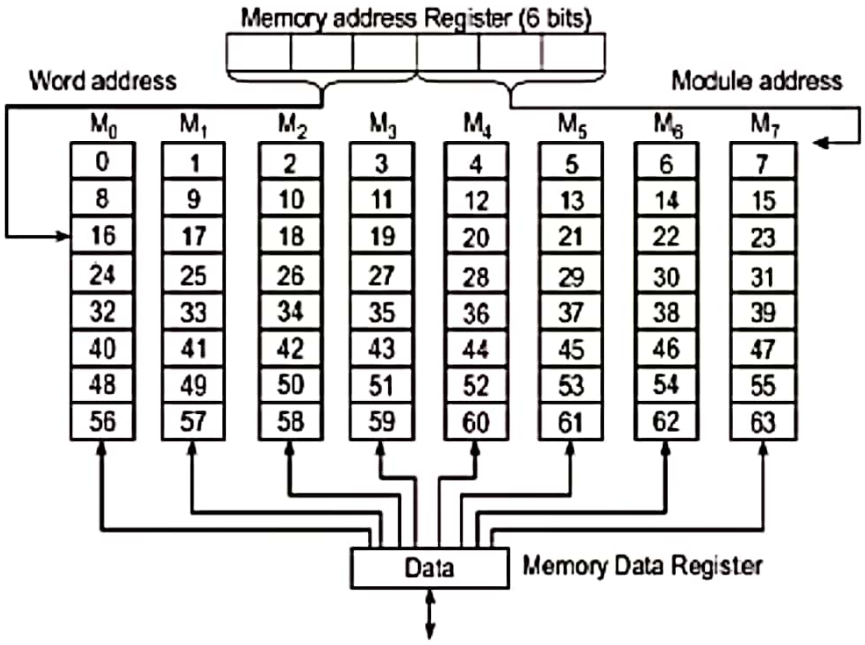
- Vector operands may have arbitrary length.
- Vector elements are not necessarily stored in contiguous memory locations.
- To access a vector a memory, one must specify its base, stride, and length.
- Since each vector register has fixed length, only a segment of the vector can be loaded into a vector register.
- Vector operands should be stored in memory to allow pipelined and parallel access. Access itself should be pipelined.

Three types of Vector-access memory organization schemes

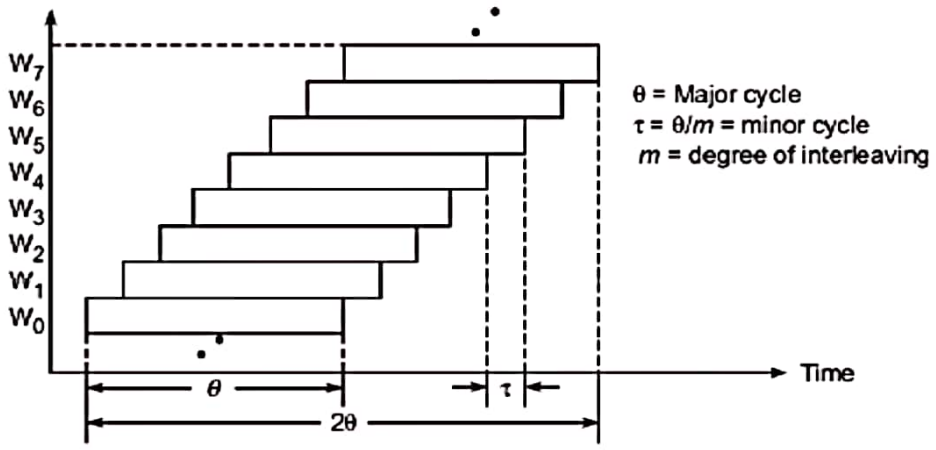
1. C-Access memory organization

The m-way low-order memory structure, allows m words to be accessed concurrently and overlapped.

The access cycles in different memory modules are staggered. The low-order a bits select the modules, and the high-order b bits select the word within each module, where $m=2^a$ and $a+b = n$ is the address length.



(a) Eight-way low-order interleaving (absolute address shown in each memory word)



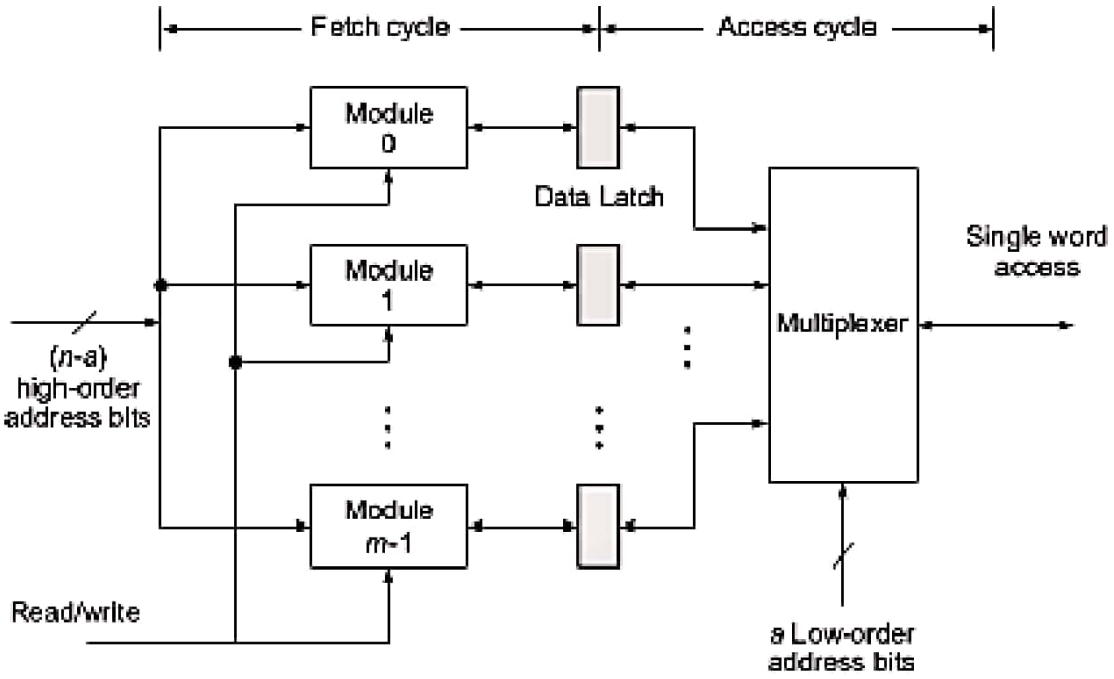
(b) Pipelined access of eight consecutive words in a C-access memory

Fig. 5.16 Multiway interleaved memory organization and the C-access timing chart

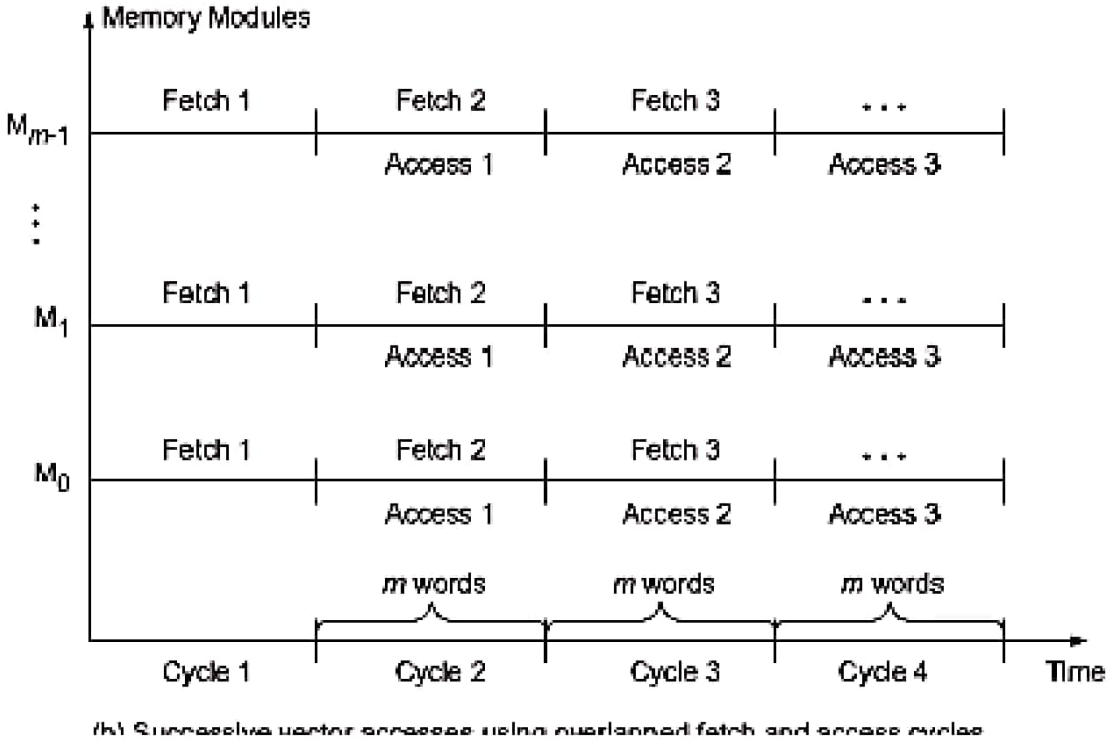
- To access a vector with a stride of 1, successive addresses are latched in the address buffer at the rate of one per cycle.
- Effectively it takes m minor cycles to fetch m words, which equals one (major) memory cycle as stated in Fig. 5.16b.
- If the stride is 2, the successive accesses must be separated by two minor cycles in order to avoid access conflicts. This reduces the memory throughput by one-half.
- If the stride is 3, there is no module conflict and the maximum throughput (m words) results.
- In general, C-access will yield the maximum throughput of m words per memory cycle if the stride is relatively prime to m, the number of interleaved memory modules.

2. S-Access memory organization

All memory modules are accessed simultaneously in a synchronized manner. The high order (n-a) bits select the same offset word from each module.



(a) S-access organization for an *m*-way interleaved memory



(b) Successive vector accesses using overlapped fetch and access cycles

3. C/S-Access memory organization

- Here C-access and S-access are combined.
- n access buses are used with m interleaved memory modules attached to each bus.

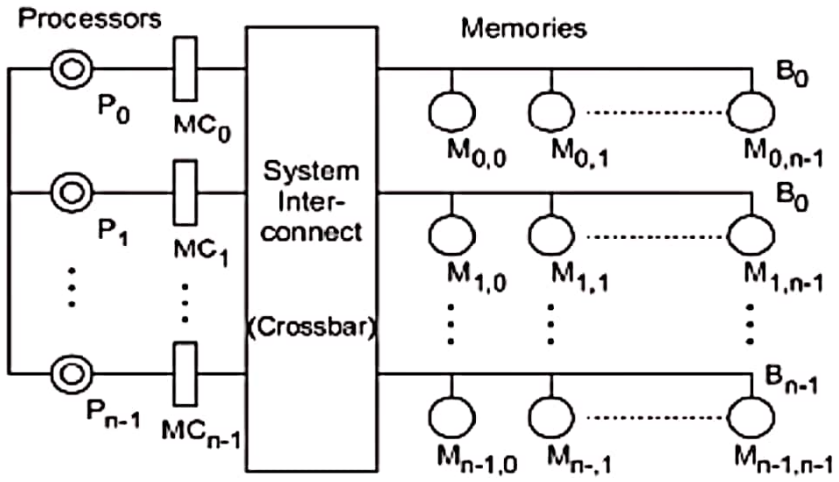


Fig. 8.4 The C/S memory organization with $m = n$. (Courtesy of D.K. Panda, 1990)

- The C/S-access memory is suitable for use in vector multiprocessor configurations.
- It provides parallel pipelined access of a vector data set with high bandwidth.
- A special vector cache design is needed within each processor in order to guarantee smooth data movement between the memory and multiple vector processors.

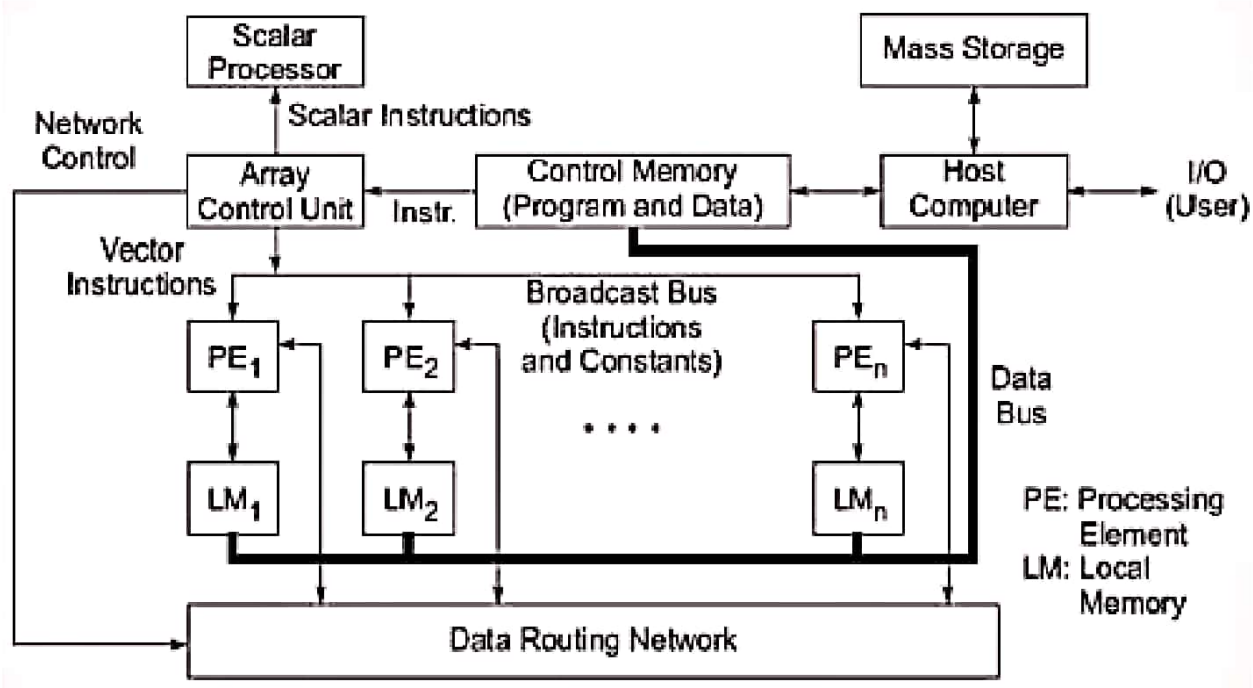
8A)

SIMD Implementation Models OR (Two models for constructing SIMD Super Computers)

SIMD models differentiates on base of memory distribution and addressing scheme used.

Most SIMD computers use a single control unit and distributed memories, except for a few that use associative memories.

1. Distributed memory model

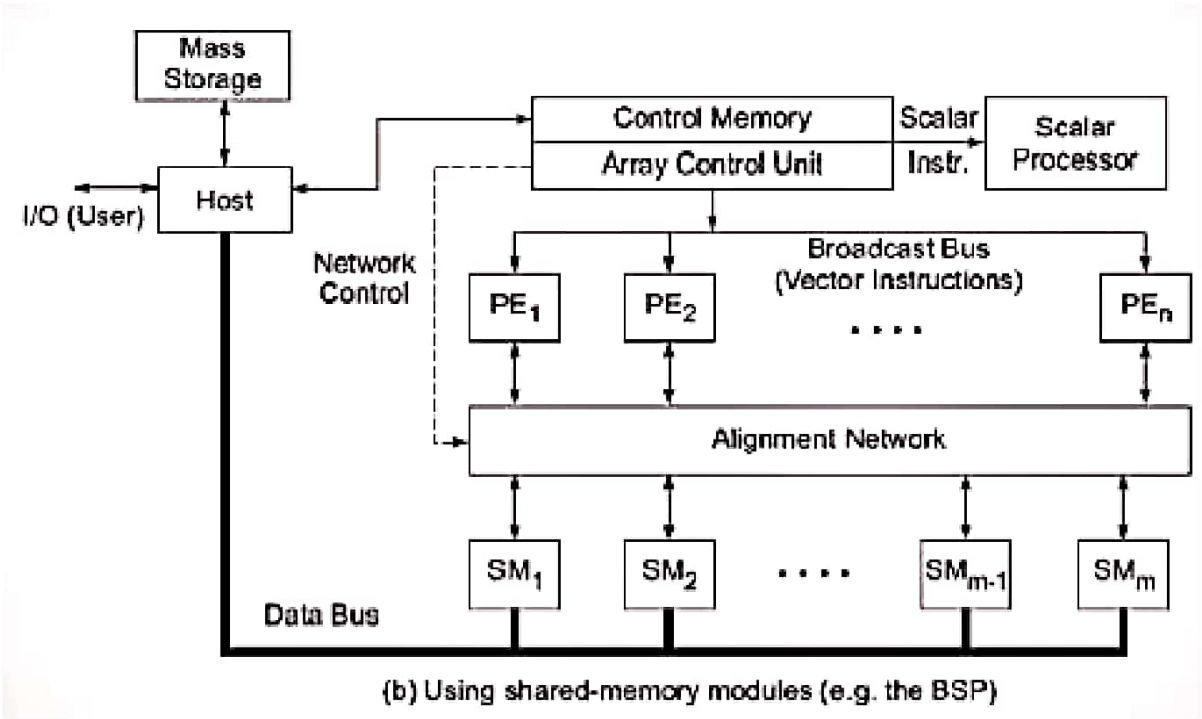


(a) Using distributed local memories (e.g. the Illiac IV)

- Spatial parallelism is exploited among the PEs.
- A distributed memory SIMD consists of an array of PEs (supplied with local memory) which are controlled by the array control unit.
- Program and data are loaded into the control memory through the host computer and distributed from there to PEs local memories.
- An instruction is sent to the control unit for decoding. If it is a scalar or program control operation, it will be directly executed by a scalar processor attached to the control unit.
- If the decoded instruction is a vector operation, it will be broadcast to all the PEs for parallel execution.
- Partitioned data sets are distributed to all the local memories attached to the PEs trough a vector data bus.

2. Shared Memory Model

- An alignment network is used as the inter-PE memory communication network. This network is controlled by control unit.
- The alignment network must be properly set to avoid access conflicts.
- Figure below shows a variation of the SIMD computer using shared memory among the PEs.
- Most SIMD computers were built with distributed memories.



(b) Using shared-memory modules (e.g. the BSP)

8B)

Context-Switching Policies

Different multithreaded architectures are distinguished by the context-switching policies adopted.

Four switching policies are:

1. **Switch on Cache miss** – This policy corresponds to the case where a context is preempted when it causes a cache miss.

   In this case, R is taken to be the average interval between misses (in Cycles) and L the time required to satisfy the miss.

   Here, the processor switches contexts only when it is certain that the current one will be delayed for a significant number of cycles.

2. **Switch on every load** - This policy allows switching on every load, independent of whether it will cause a miss or not.

   In this case, R represents the average interval between loads. A general multithreading model assumes that a context is blocked for L cycles after every switch; but in the case of a switch-on-load processor, this happens only if the load causes a cache miss.

3. **Switch on every instruction** – This policy allows switching on every instruction, independent of whether it is a load or not. Successive instructions become independent , which will benefit pipelined execution.

4. **Switch on block of instruction** – Blocks of instructions from different threads are interleaved. This will improve the cache-hit ratio due to locality. It will also benefit single-context performance.

Module-4

7 a. Explain crossbar networks and cross-point switch design in multiprocessor system. (08 Mar

b. With necessary sketches, explain the cache-coherence problems in data sharing and process migration. (08 Mar

OR

8 a. With a diagram, explain the architecture of the connection machine CM-2. (08 Mar

b. Explain the context–switching policies. (08 Mar

7a)

Crossbar Networks

Crossbar networks connect every input to every output through a crosspoint switch. A crossbar network is a single stage, non-blocking permutation network.

In an n-processor, m-memory system, n * □m crosspoint switches will be required. Each crosspoint is a unary switch which can be open or closed, providing a point-to-point connection path between the processor and a memory module.

Crosspoint Switch Design

Out of n crosspoint switches in each column of an *n* *□*m* crossbar mesh, only one can be connected at a time.

Crosspoint switches must be designed to handle the potential contention for each memory module. A crossbar switch avoids competition for bandwidth by using $O(N^2)$ switches to connect N inputs to N outputs.

Although highly non-scalable, crossbar switches are a popular mechanism for connecting a small number of workstations, typically 20 or fewer.

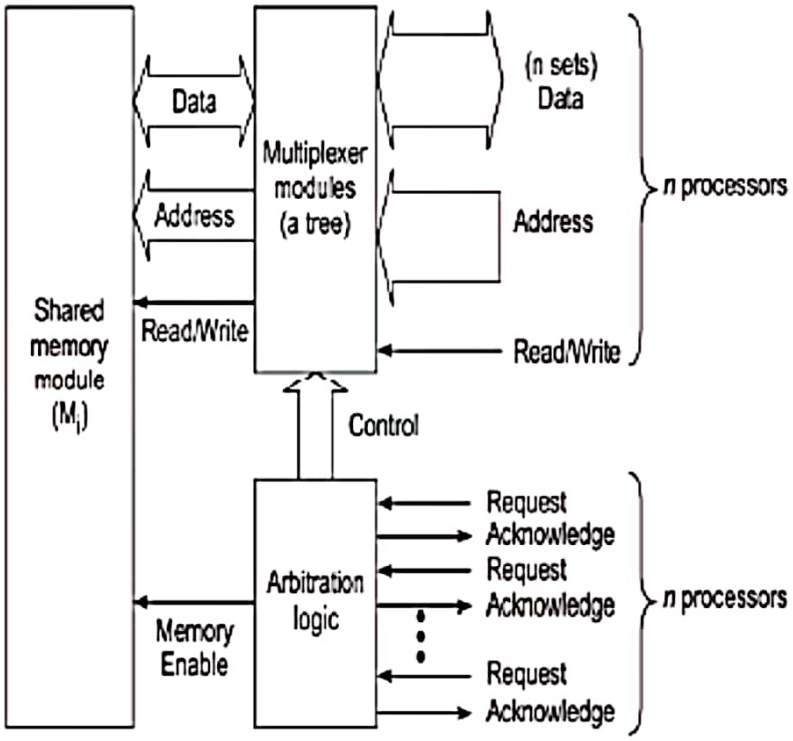


Fig. 7.6 Schematic design of a row of crosspoint switches in a crossbar network

Each processor provides a request line, a read/write line, a set of address lines, and a set of data lines to a crosspoint switch for a single column. The crosspoint switch eventually responds with an acknowledgement when the access has been completed.

7b)

Inconsistency in Data sharing:

The cache inconsistency problem occurs only when multiple private caches are used.

In general, three sources of the problem are identified:

- ✓ sharing of writable data,
- ✓ process migration
- ✓ I/O activity.

- 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 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.
- On the other hand, inconsistency may also occur when a write back policy is used, as shown on the right.
- The main memory will be eventually updated when the modified data in the cache are replaced or invalidated.

Process Migration and I/O

The figure shows the occurrence of inconsistency after a process containing a shared variable X migrates from processor 1 to processor 2 using the write-back cache on the right. In the middle, a process migrates from processor 2 to processor1 when using write-through caches.

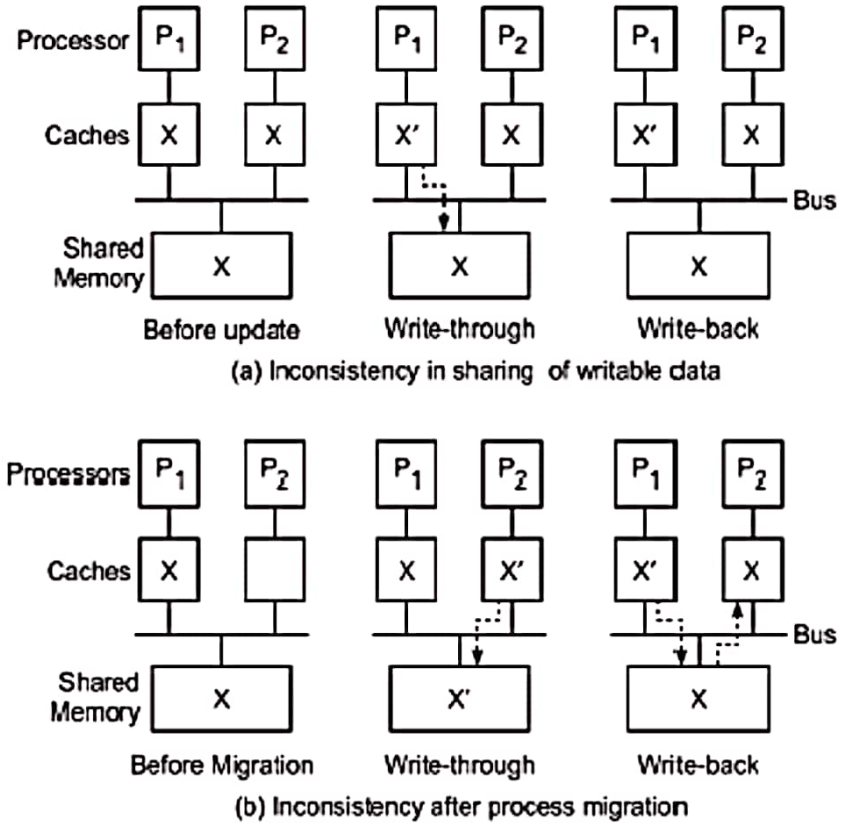


Fig. 7.12 Cache coherence problems in data sharing and in process migration (Adapted from Dubois, Scheurich, and Briggs 1988)

In both cases, inconsistency appears between the two cache copies, labeled X and X'. Special precautions must be exercised to avoid such inconsistencies. A coherence protocol must be established before processes can safely migrate from one processor to another.

8a)CM-2

8.4.2 CM-2 Architecture

The Connection Machine CM-2 produced by Thinking Machines Corporation was a fine-grain MPP computer using thousands of bit-slice PEs in parallel to achieve a peak processing speed of above 10 Gflops.

Program Execution Paradigm

All programs started execution on a front-end, which issued microinstructions to the back-end processing array when data-parallel operations were desired. The sequencer broke down these microinstructions and broadcast them to all data processors in the array.

Data sets and results could be exchanged between the front-end and the processing array in one of three ways as shown in the figure:

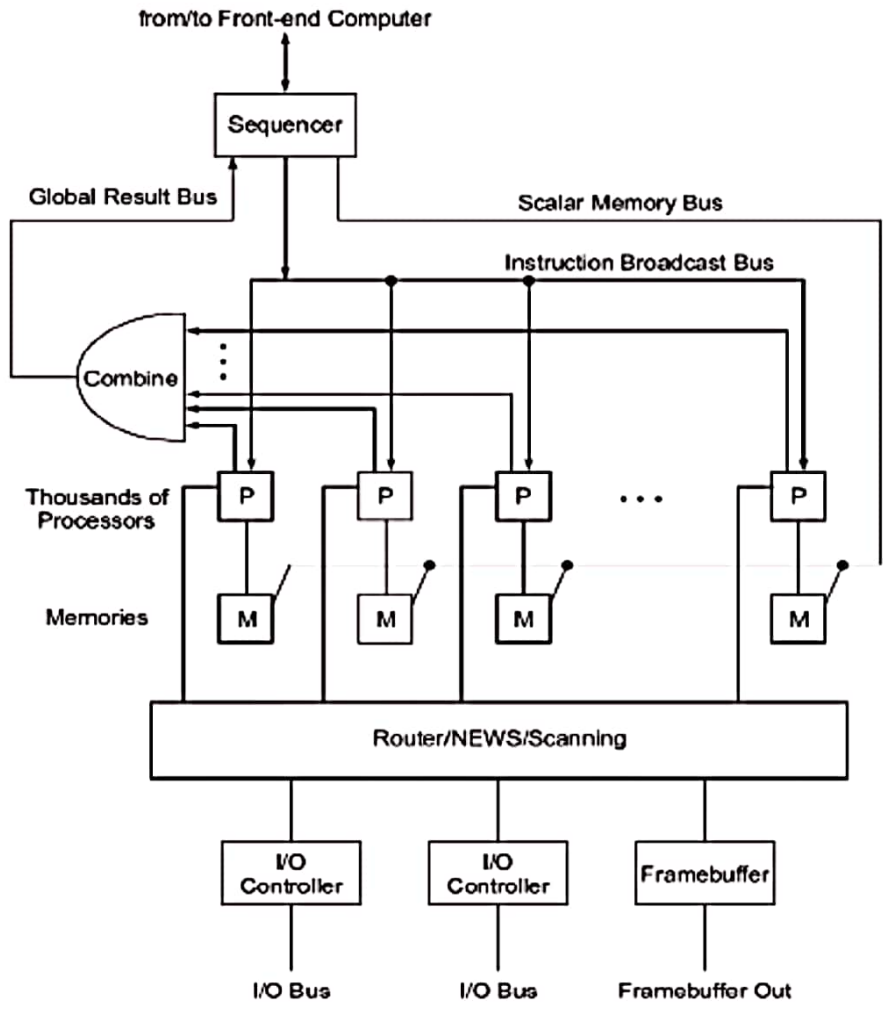


Fig. 8.23 The architecture of the Connection Machine CM-2 (Courtesy of Thinking Machines Corporation, 1990)

- **Broadcasting:** Broadcasting was carried out through the broadcast bus to all data processors at once.
- **Global combining:** Global combining allowed the front-end to obtain the sum, largest value, logical OR etc, of values one from each processor.
- **Scalar memory bus:** Scalar bus allowed the front-end to read or to write one 32-bit value at a time from or to the memories attached to the data processors.

Processing Array

The processing array contained from 4K to 64K bit-slice data processors(PEs), all of which were controlled by a sequencer.

Processing Nodes

Each data processing node contained 32 bit-slice data processors, an optional floating point accelerator and interfaces for inter processor communication.

Hypercube Routers

The router nodes on all processor chips were wired together to frm a Boolean n-cube. A full configuration of CM-2 had 4096 router nodes on processor chips interconnected as a 12-dimensional hypercube.

Module-4

7 a. Explain hierarchical bus system with neat diagram. (08 Marks
b. Explain crossbar networks along with its advantages and limitations. (08 Marks

OR

8 a. Explain snoopy protocols with its approaches. (08 Marks
b. Briefly explain message routing schemes. (08 Marks

7.1.1 Hierarchical Bus Systems

- A *bus system* consists of a hierarchy of buses connecting 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.
- In general, the hierarchy of bus systems are packaged at different levels as depicted in Fig. 7.2, including local buses on boards, backplane buses, and I/O buses.

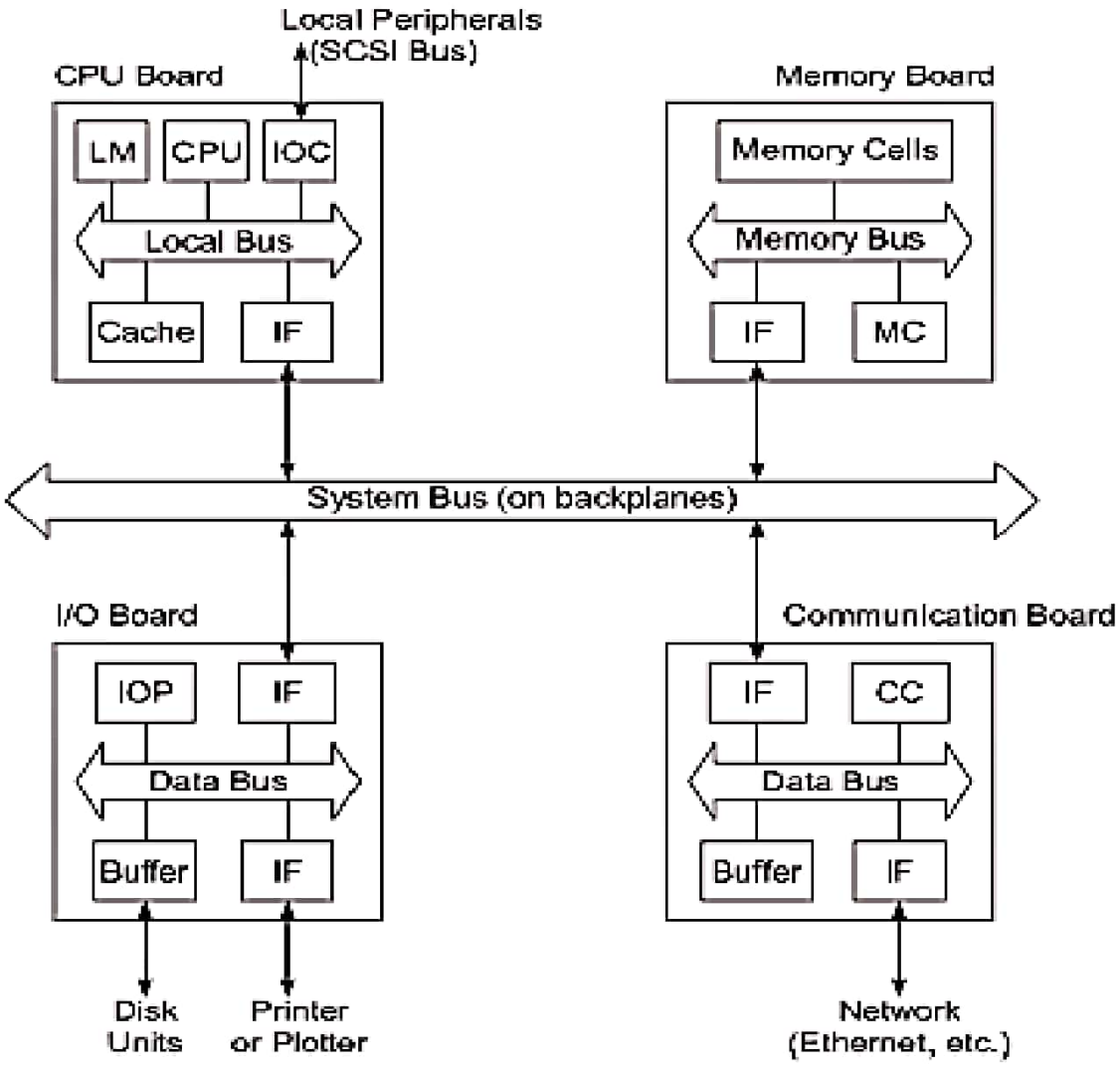


Fig. 7.2 Bus systems at board level, backplane level, and I/O level

- **Local Bus** Buses implemented on *printed-circuit* boards are called *local buses*.
- On a processor board one often finds a local bus which provides a common communication path among major components (chips) mounted on the board.
- A **memory board** uses a *memory bus* to connect the memory with the interface logic.
- An **I/O board** or network interface board uses a *data bus*. Each of these board buses consists of signal and utility lines.

Backplane Bus

A backplane is a printed circuit on which many connectors are used to plug in functional boards. A system bus, consisting of shared signal pathsand utility lines, is built on the backplane.This system bus provides a common communication path among all plug-in boards.

I/O Bus

Input/Output devices are connected to a comuter 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.

Special interface logic is used to connect various board types to the backplane bus.

7b)

Advantages of Crossbar Networks:

1. It is a Non Blocking Network that allows multiple i/o connections to be achieved simultaneously.
2. It provides full connectivity, i.e., any permutation can be implemented using a crossbar.
3. It is highly useful in a multiprocessor system, as all processors can send memory requests independently and asynchronously.
4. It gives maximum utilization of bandwidth as compared to other networks like bus system, multistage networks, etc.

Limitations:

1. The main disadvantage of this system is that, the failure of a single switch will make some subscribers inaccessible.
2. The Crosspoint switch is the abstract of any switch such as the time or space switch. If N connections can be made simultaneously in an NXN switch matrix, it is called the **Non**-blocking Switch

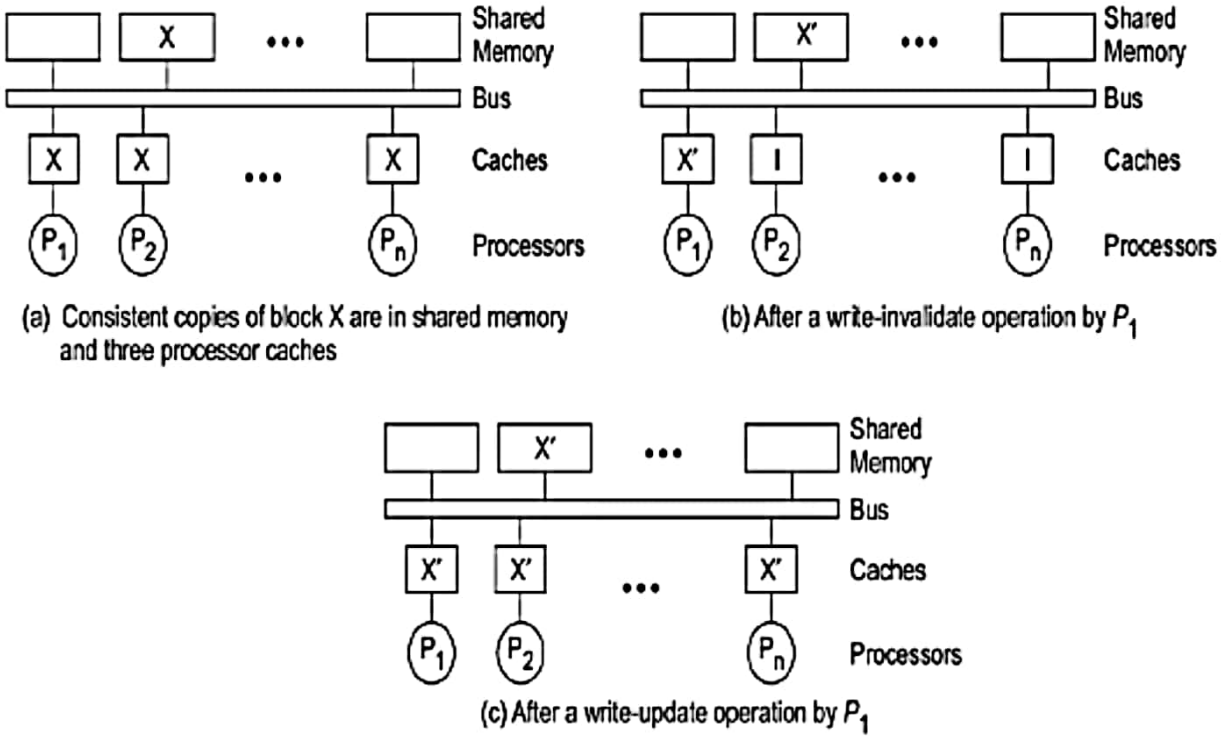
8a)

Protocol Approaches for Cache Coherence:

1. Snoopy Bus Protocol
2. Directory Based Protocol

1. Snoopy Bus Protocol

- Snoopy protocols achieve data consistency among the caches and shared memory through a bus watching mechanism.
- In the following diagram, two snoopy bus protocols create different results. Consider 3 processors (P_1, P_2, P_n) maintaining consistent copies of block X in their local caches (Fig. 7.14a) and in the shared memory module marked X.



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

(b) After a write-invalidate operation by P_1

(c) After a write-update operation by P_1

Fig. 7.14 Write-invalidate and write-update coherence protocols for write through caches (1: invalidate)

- Using a write-invalidate protocol, the processor P_1 modifies (writes) its cache from X to X', and all other copies are invalidated via the bus (denoted I in Fig. 7.14b). Invalidated blocks are called dirty, meaning they should not be used.
- The write-update protocol (Fig. 7.14c) demands the new block content X' be broadcast to all cache copies via the bus.
- The memory copy also updated if write through caches are used. In using write-back caches, the memory copy is updated later at block replacement time.

8b)

7.4.1 Message Routing Schemes

- Message formats are introduced below. Refined formats led to the improvement from store-and-forward to wormhole routing in two generations of multicomputers.
- A handshaking protocol is described for asynchronous pipelining of successive routers along a communication puth. Finally, latency analysis is conducted to show the time difference between the two routing schemes presented

- Message Formats Information units used in message routing are specified in Fig. 7.26.

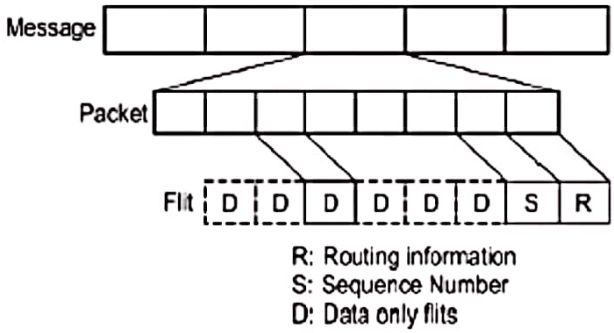


Fig.7.26 The format of message, packets, and flits (control flow digits) used as information units of communication in a message-passing network

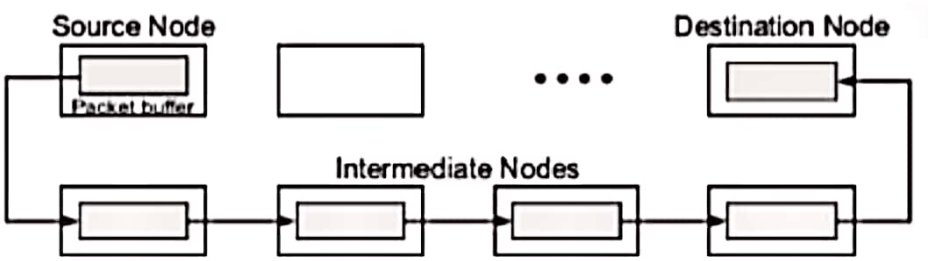
- Among is the logical unit for internode communication. It is often assembled from an arbitrary number of fixed-length packets, thus it may have a variable length
- A parker is the basic unit containing the destination address for routing purpose. Because different packets may arrive at the destination asynchronously, a sequence number is needed in each packet to allow reassembly of the message transmitted
- A packet can be further divided into a number of fixed-length flits (flow control digits).
- Routing information(destination) and sequence number occupy the header fits.
- The remaining fits are the data elements of a pucket In multicomputers with store-and-forward routing, packets are the smallest unit of informationtransmission.

Two Message Passing mechanisms are:

1 Store and Forward Routing

2 Wormhole Routing

1. Store and Forward Routing

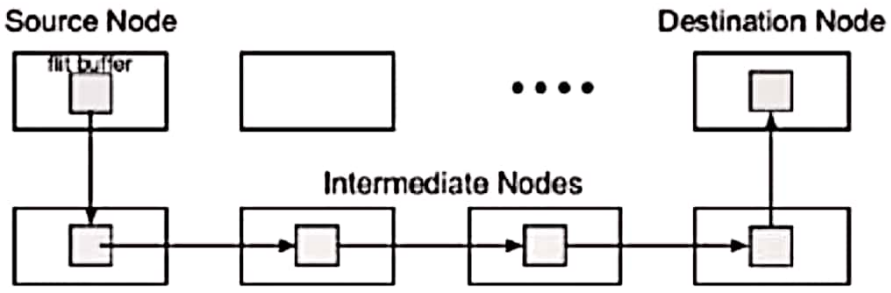


(a) Store-and-forward routing using packet buffers in successive nodes

- Packets are the basic unit of information flow in a store-and-forward network.
- Each node is required to use a packet buffer.

- A packet is transmitted from a source node to a destination node through a sequence of intermediate nodes.
- When a packet reaches an intermediate node, it is first stored in the buffer.
- Then it is forwarded to the next node if the desired output channel and a packet buffer in the receiving node are both available.

2. Wormhole Routing



(b) Wormhole routing using flit buffers in successive routers

- Packets are subdivided into smaller flits. Flit buffers are used in the hardware routers attached to nodes.
- The transmission from the source node to the destination node is done through a sequence of routers.
- All the flits in the same packet are transmitted in order as inseparable companions in a pipelined fashion.
- Only the header flit knows where the packet is going.
- All the data flits must follow the header flit.
- Flits from different packets cannot be mixed up. Otherwise they may be towed to the wrong destination.