Models of Parallel Computers

Outlines

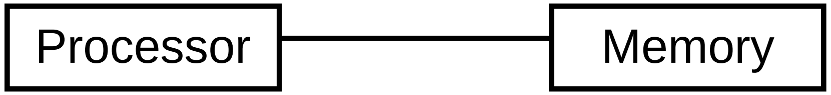
- Sequential Computers
- A Taxonomy of Parallel Architectures
- Dynamic Interconnection Networks
- Static Interconnection Networks
- Evaluating Static Interconnection Networks

Sequential Computers

- Sequential Computers
- Sequential Computer with Memory Interleaving
- Sequential Computer with Memory Interleaving and cache
- Pipelined Processor with d-stages

Sequential Computers

- Sequential computers are based on John von Neumann model
  - A simple sequential computer:



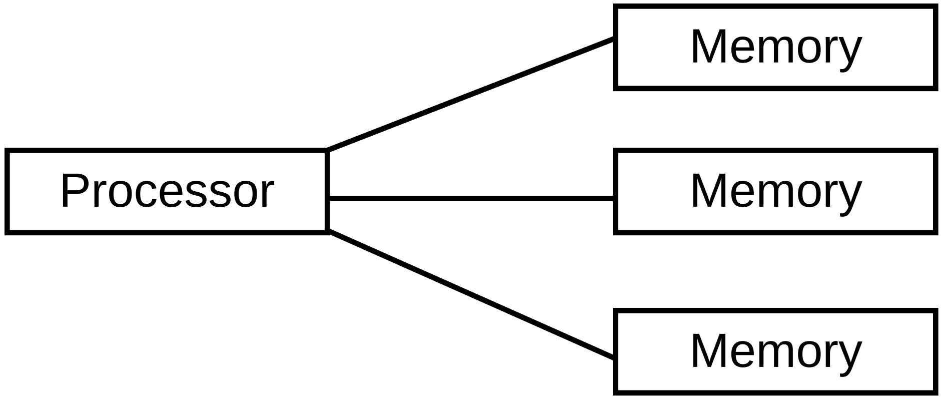
  - This computation model takes a single sequence of instructions and operates on a single sequence of data
  - Referred to as Single Instruction Stream Single Data Stream (SISD) computers

Sequential Computers (Continues)

- The speed of an SISD computer is limited by two factors:

1) The execution rate of instruction
2) The speed at which information is exchanged between memory and CPU

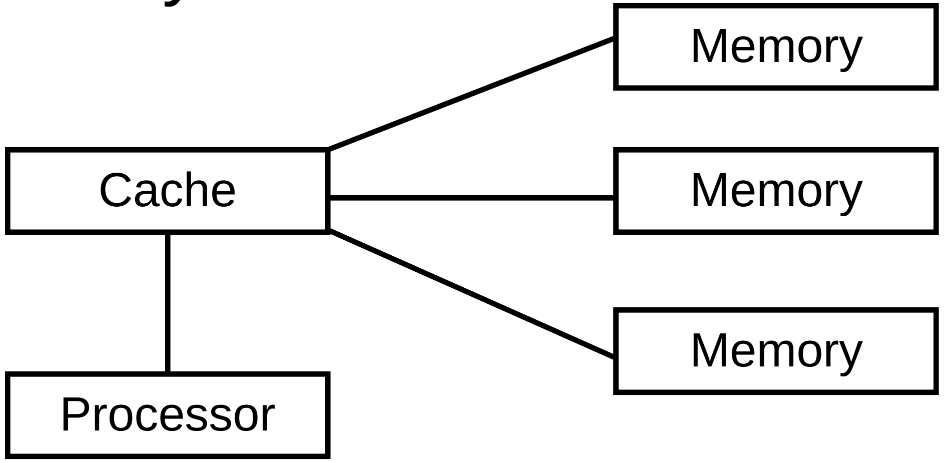
Sequential Computer with Memory Interleaving



- Increasing the speed of exchanging information between memory and CPU by increasing number of channels.
- This is done by dividing memory into a number of banks, each of which is accessed independently

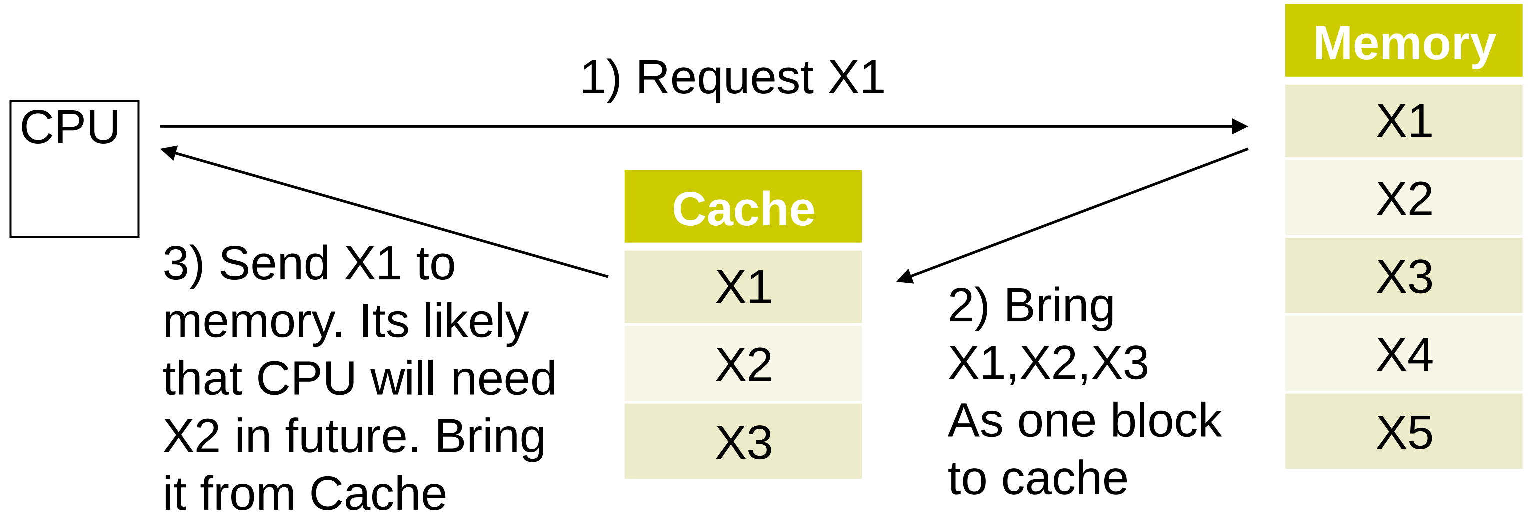
Sequential Computer with Memory Interleaving and cache

- Another way to increase the rate of information exchange between CPU and memory is to use a relatively small and very fast memory (**cache memory**) to act as a buffer to larger and slower main memory.

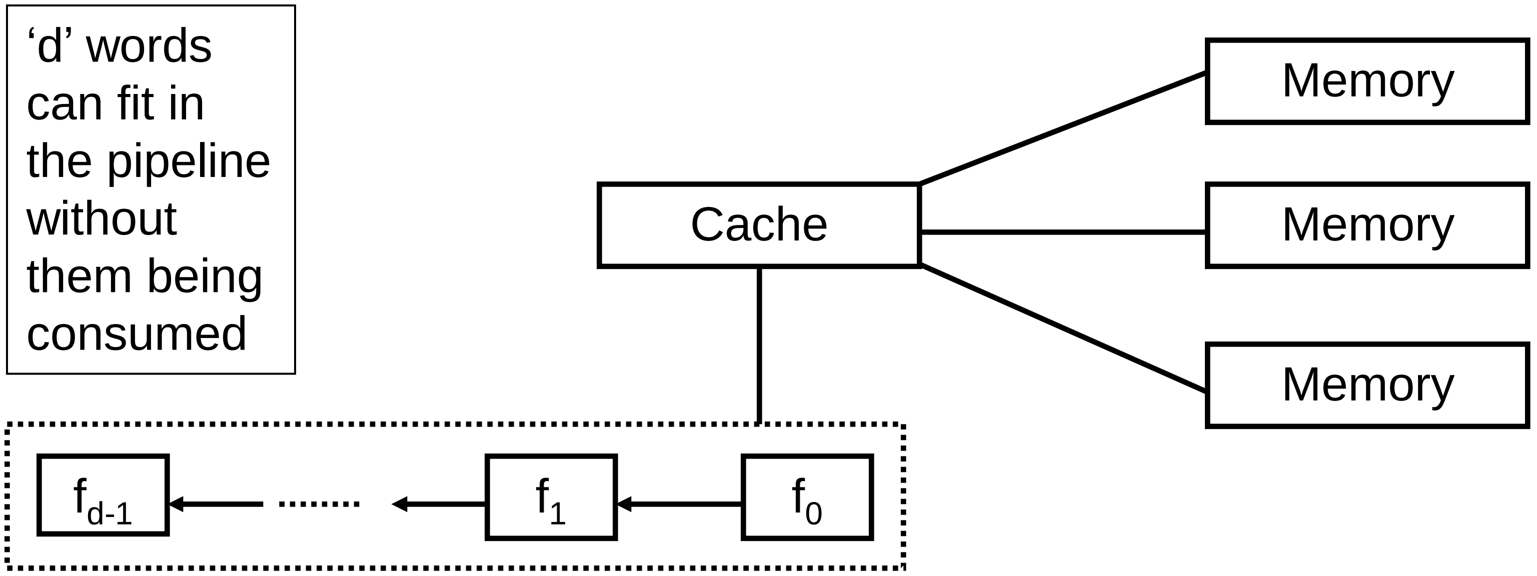


Sequential Computer with Memory Interleaving and cache (Continues)

- **Cache memory** uses the principle that if a word is accessed from a part of memory, it is likely that subsequent memory accesses will be to words in the neighbourhood of this memory location.

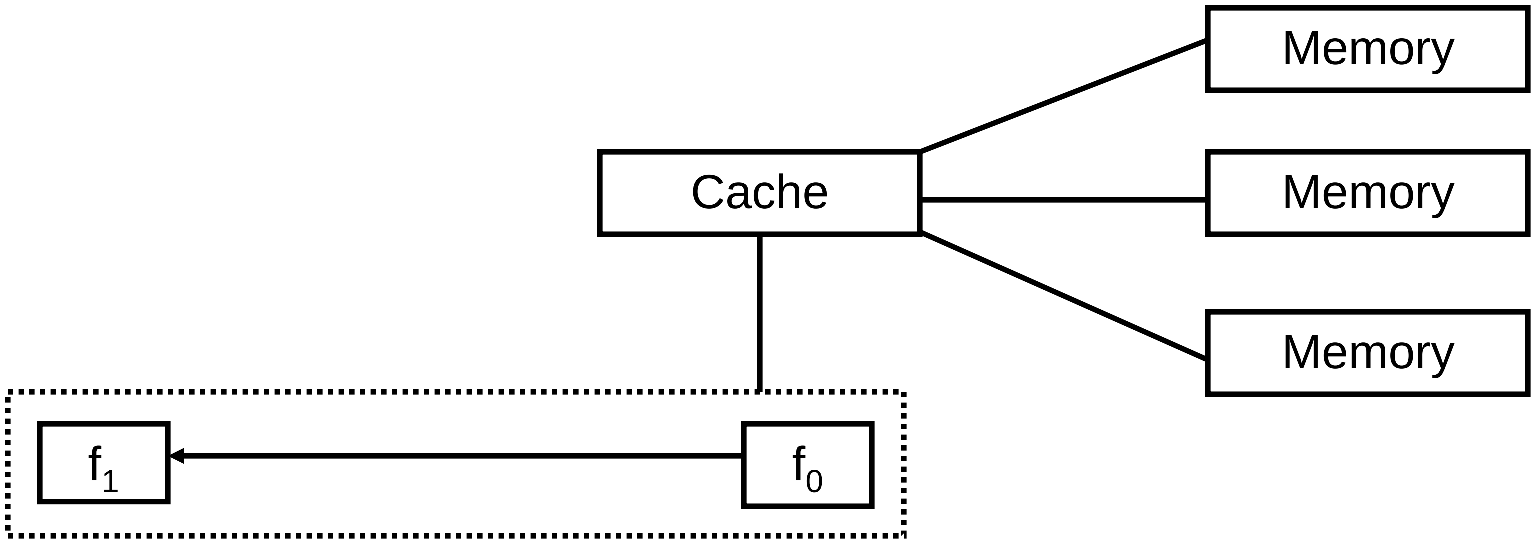

CPU
1) Request X1
Memory
X1
X2
X3
X4
X5
Cache
X1
X2
X3
3) Send X1 to memory. Its likely that CPU will need X2 in future. Bring it from Cache
2) Bring X1,X2,X3 As one block to cache

Pipelined Processor with d-stages


'd' words can fit in the pipeline without them being consumed
Cache
Memory
Memory
Memory
f_{d-1}
....
f_1
f_0

- Rate of execution of instruction can be increased by overlapping execution of an instruction with operation of fetching next instruction to be executed

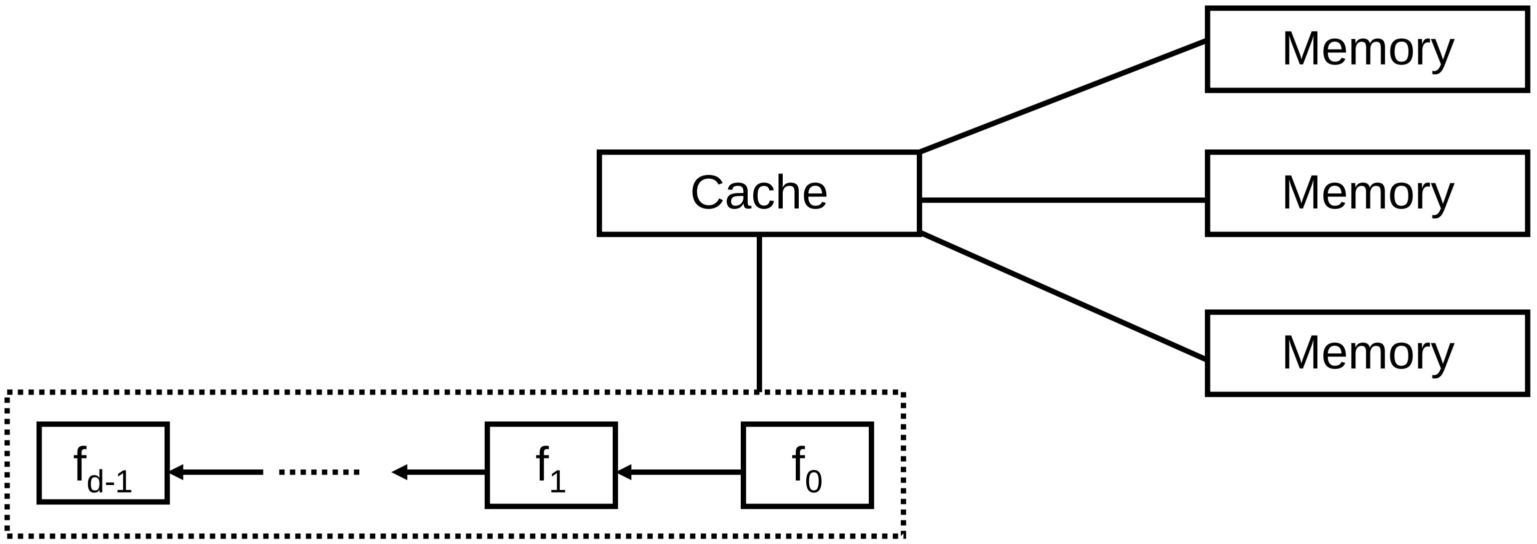
Without Pipelining



Without Pipelining:

- Instruction f1 cannot be brought from cache until instruction f0 finishes execution

Without Pipelining



With Pipelining:

- instructions are brought from cache to pipeline
- By the time f0 is executed by processor, f1 will be ready to be executed
- By the time f1 is executed by processor, f2 will be ready to be executed
- And so on.

Pipelined Processor with d-stages (Continues)

Adders and Multipliers are used by CPU for arithmetic computation

- **Instruction Pipelining:** while CPU is busy executing current instruction, next instruction is brought from memory into instruction queue
- **Execution Pipelining:** multiple instructions are allowed to be in various stages of execution in functional units such as multipliers and adders

Execution Pipelining Example:

- Suppose instructions f0,f1 need to use adder, then multiplier in order
- No need for f1 to wait until f0 finishes using the adder and the multiplier
- As soon as f0 finishes using the adder and starts using the multiplier, f1 can immediately start using the adder

Memory Interleaving, Cache Memory, and Pipelining

- Memory Interleaving, Cache Memory, and Pipelining are commonly used in high-performance SISD computers
- Memory Interleaving and Pipelining are useful only if a small set of operations is performed on large arrays of data
- Cache memories do increase processor-memory bandwidth, but their speed is still limited by hardware technology
- Another way to speed up the rate of instruction execution is to use multiple CPUs and memory units interconnected in some fashion

A Taxonomy of Parallel Architectures

- Parallel Computers differ along various dimensions such as:
  - **Control Mechanism:** *single global control unit or multiple independent control units*
  - **Address-Space Organization:** *distributed or shared memory*
  - **Interconnected Network:** *static or dynamic*
  - **Granularity of Processors:** *course or fine grain*

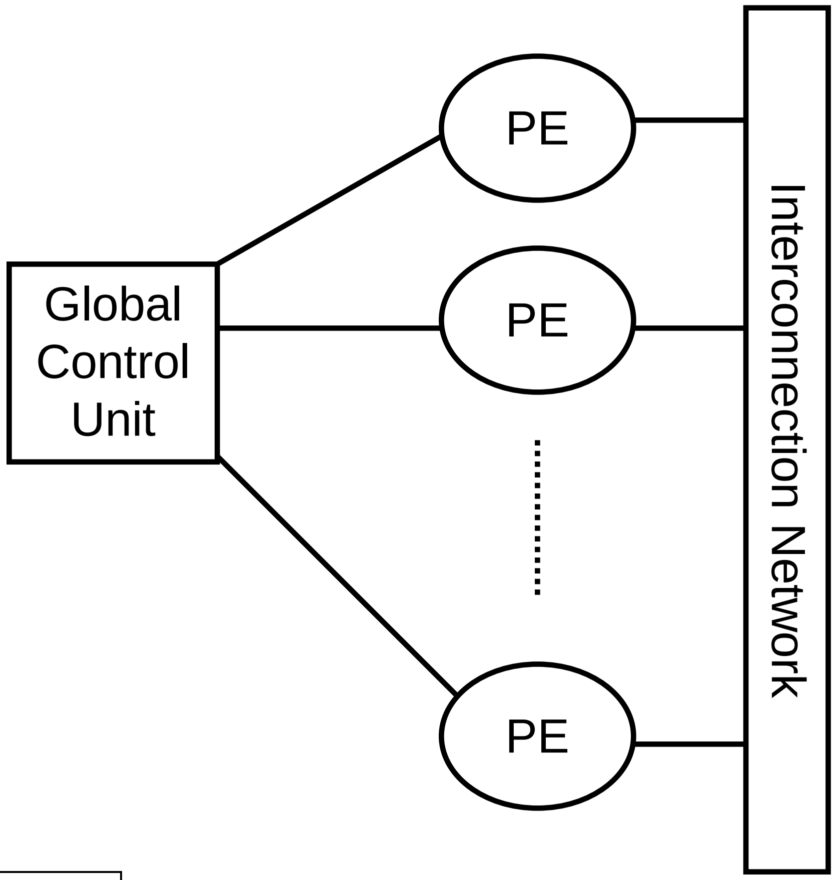
Control Mechanism

- Processing units in parallel computers either operate under centralized control of a single control unit or work independently.
- **Two Models:**
  1) Single Instruction Stream, Multiple Data Stream (SIMD).

  2) Multiple Instruction Stream, Multiple Data Stream (MIMD).

Single Instruction Stream, Multiple Data Stream (SIMD)

- A single control unit dispatches instructions to each processing unit.
- The same instruction is executed synchronously by all processing units.
- Examples of SIMD parallel computers: MPP, CM-2, MasPar MP-1 and MP-2.



PE (Processing Element - Processor)

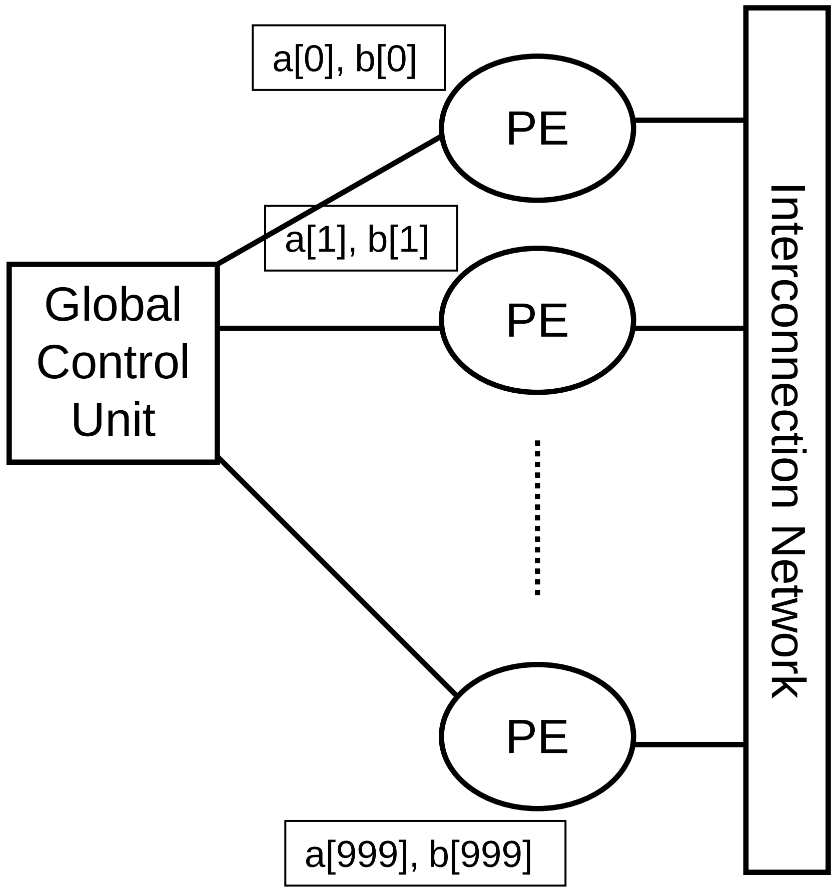
Single Instruction Stream, Multiple Data Stream (SIMD)

```
For(int i=0;i<1000;i++){
    c[i] = a[i] + b[i];
}
```

c[0] = a[0] + b[0];
c[1] = a[1] + b[1];
……..
All of them are independent of each other.

So, same instruction
c[i] = a[i] + b[i];
Is sent to all processors and multiple data is sent (different data for each processor) (eg., a[0] & b[0] are sent to first PE, … etc.)

All processors work in parallel to finish the job, and global control unit orchestrates work.



Single Instruction Stream, Multiple Data Stream (SIMD)

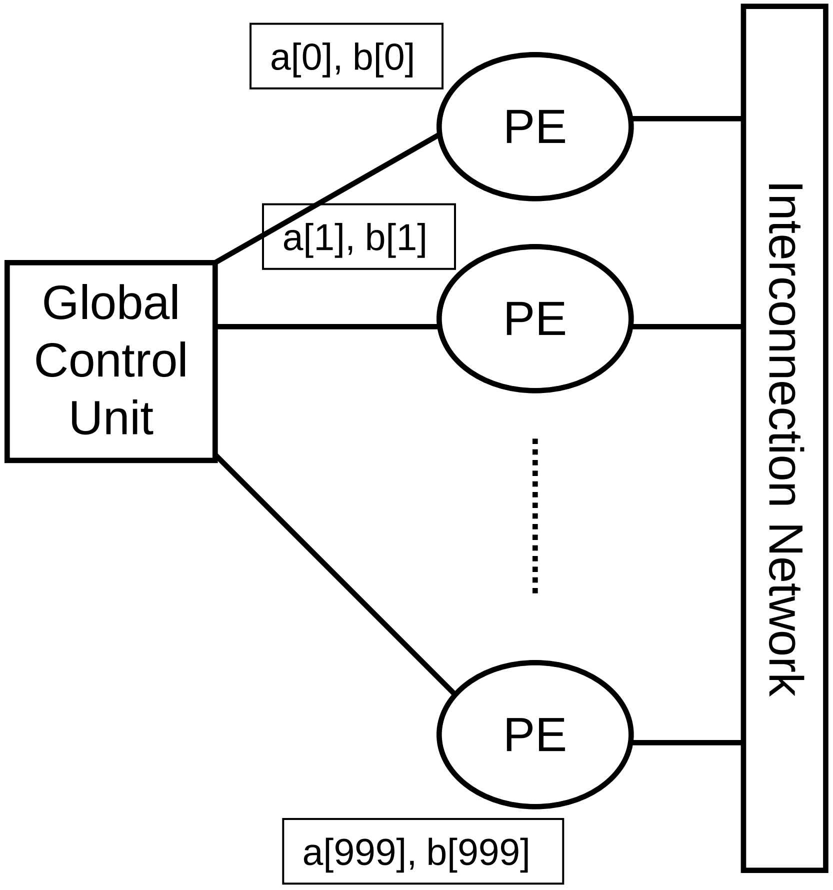
In some cases, it would be bad to use this scheme

```
For(int i=0;i<1000;i++){
   If (b[i]==0){
       c[i] = a[i];
   }
   Else{
     c[i] = a[i] / b[i];
   }
}
```

First Round:

- Single instruction: All processors are given one instruction c[i]=a[i]
- However, only when b[i] = 0, the PE will execute the statement
- Otherwise, the PE will be idle

Second round: Same as before but with c = a / b given to all PEs

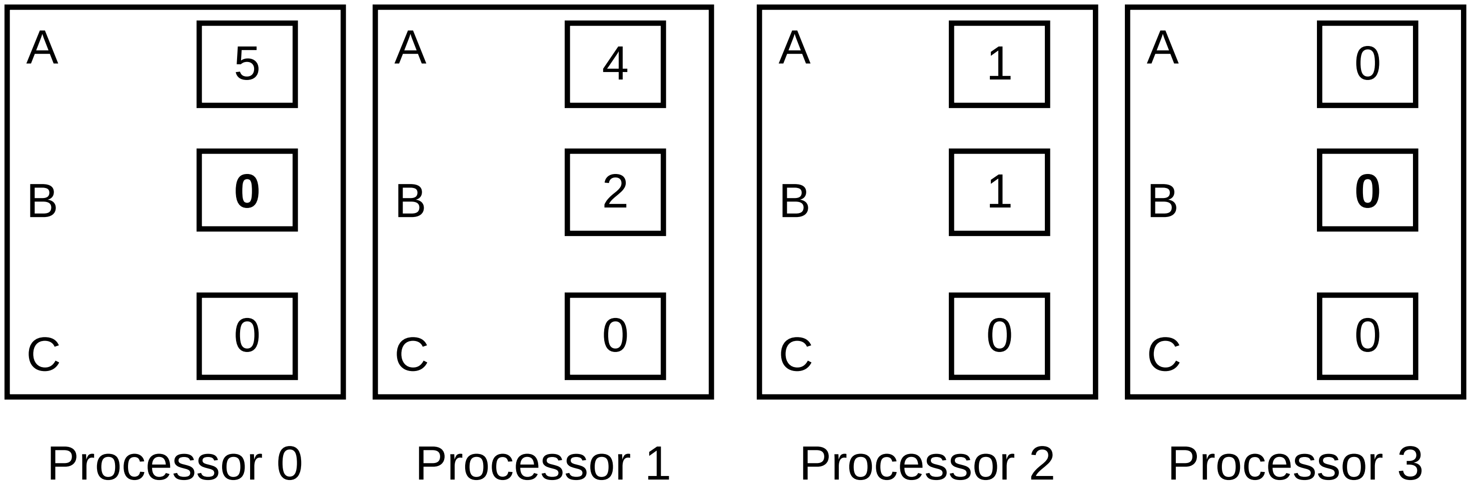


Example: Executing a conditional statement on an SIMD computer with 4 processors

The conditional statement:

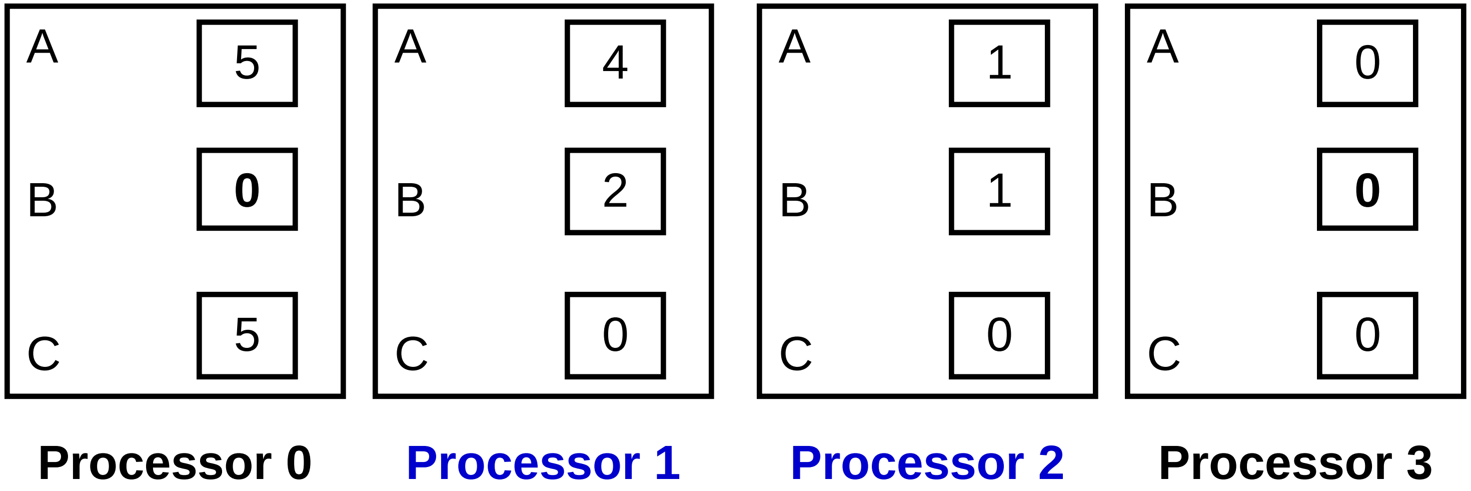
```
if (B == 0)    then    C = A;
               else    C = A / B;
```

Initial Values:



Example: Executing … (Continues)

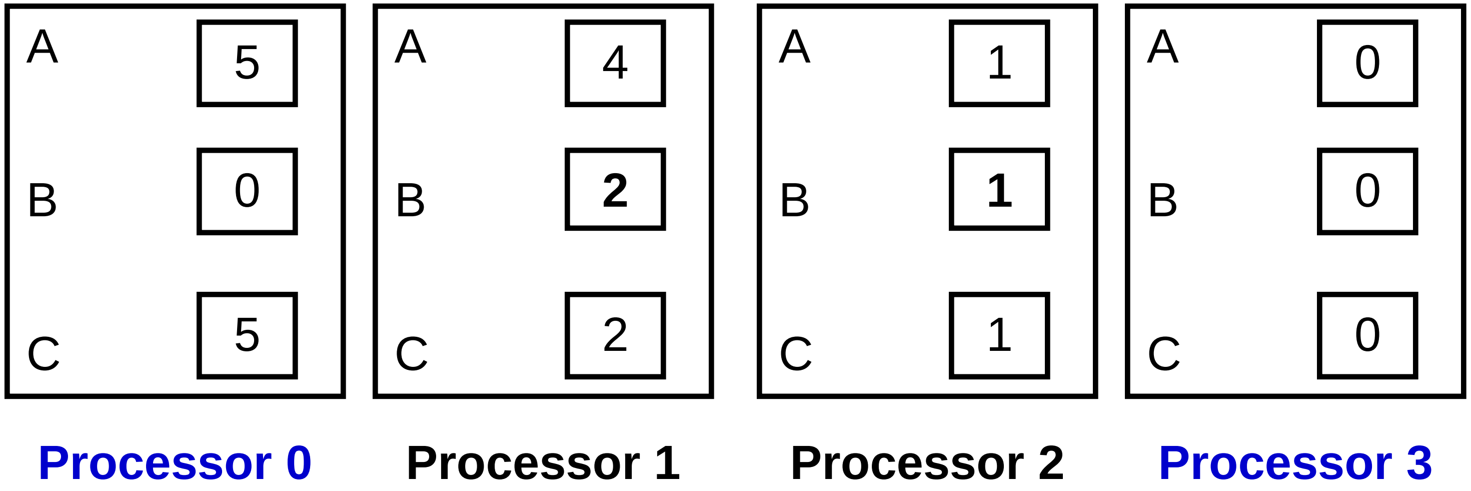
- Execution of the statement in two steps:
- Step 1:



- In Step 1, all processors that have B == 0 execute instruction C = A. All other processors are idle (that is Processor 1 and Processor 2 are idle).

Example: Executing … (Continues)

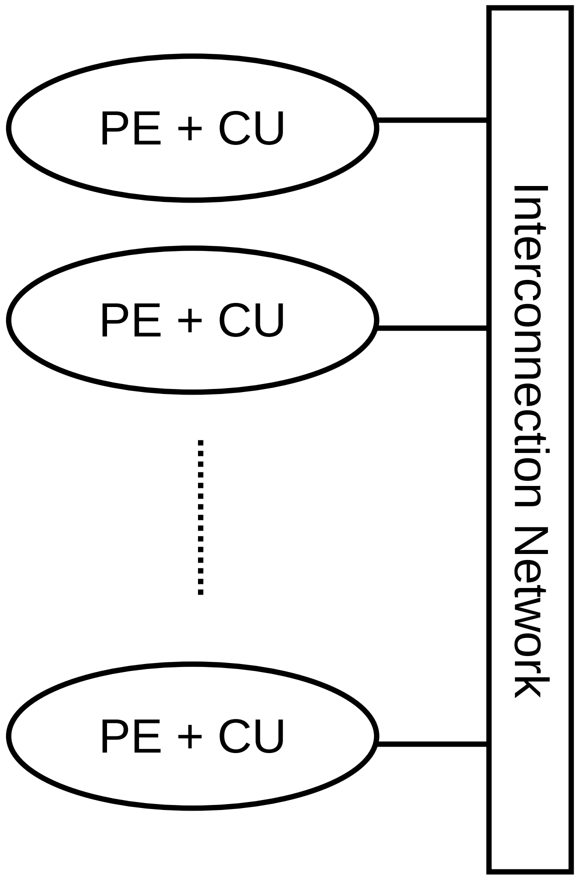
- Step 2:



- In Step 2, the 'else' part of the instruction (C = A / B) is executed. The processors that were active in Step 1 now become idle (that is Processor 0 and Processor 3 are idle).

Multiple Instruction Stream, Multiple Data Stream (MIMD)

- Each processor is capable of executing a different program independent of other processor.
- Examples of MIMD computers: Cosmic Cube, nCUBE2, CM-5.



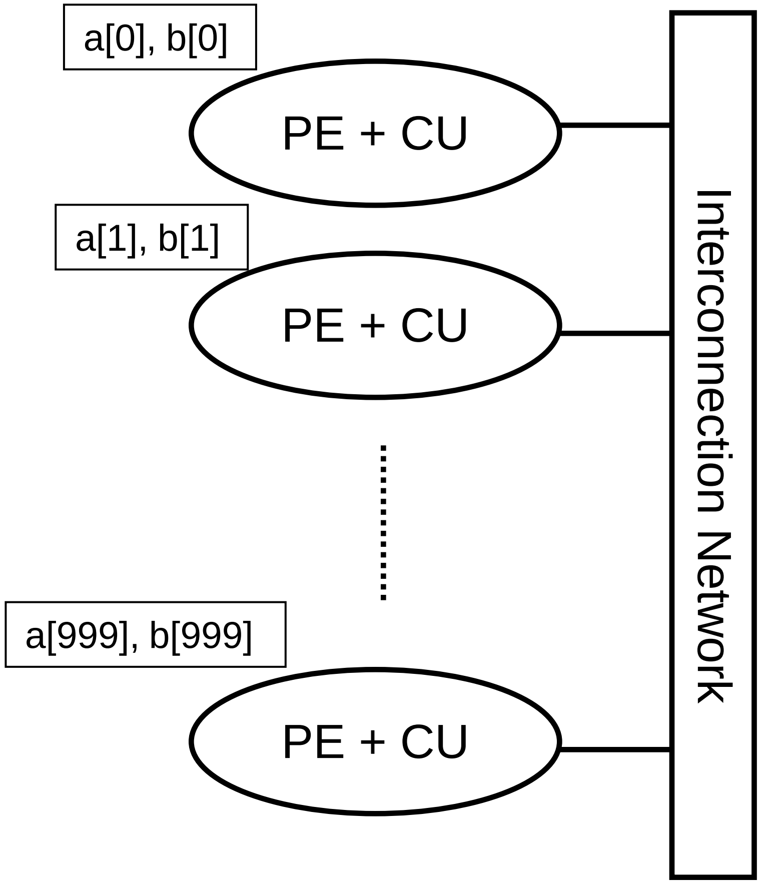
Multiple Instruction Stream, Multiple Data Stream (MIMD)

```
For(int i=0;i<1000;i++){
   If (b[i]==0){
       c[i] = a[i];
   }
   Else{
     c[i] = a[i] / b[i];
   }
}
```

Multiple Instruction:
- c[i] = a[i] is given to some PEs
- c[i] = a[i] / b[i]; is given to other PEs

Multiple Data: Values of b and a are coming from arrays.

All processors work in parallel to finish the job because each processor has multiple statements, and it can decide (Control Unit) which one to execute based on the value of b.



Comparing SIMD with MIMD Computers

Although MIMD style can exist within a single computer, it is easier for the sake of comparison between SIMD and MIMD to think of them as follows:

SIMD is equivalent to:

- One computer with multiprocessors and they interact via shared memory in the same computer.

MIMD is equivalent to:

- Multi-computers network and they interact via message passing through the network (e.g., Internet).

Comparing SIMD with MIMD Computers

1) SIMD requires less hardware than MIMD because they have only one global control unit.
    - Because the single computer in SIMD requires less hardware than several interconnected computers.
2) SIMD requires less memory.
    - Because only one copy of program needs to be stored.
3) MIMD stores program and operating system at each processor.
    - In multi-computer network each computer will have its own memory and operating system.

Comparing SIMD with MIMD Computers

4) SIMD computers are naturally suited for data-parallel programs; that is programs in which the same set of instructions are executed on a large data set.

   - Examples: matrix addition and matrix multiplication.

5) SIMD computers require less start up time for communicating with neighbouring processors.

   - Because in this case multiprocessors exist in the same machine, the network connecting them is local physical network consisting of wires (such as bus), and therefore communication takes less time.

Comparing SIMD with MIMD Computers (Continues)

6) A drawback of SIMD computers is that different processors cannot execute different instructions in same clock cycle.

   - This was explained when we tried to implement the for loop example that contained “if statement” on the SIMD model.

7) Data-parallel programs in which significant parts of the computation are contained in conditional statements are better suited to MIMD computers than to SIMD computers.

   - This was explained when we tried to implement the for loop example that contained “if statement” on the MIMD model.

Comparing SIMD with MIMD Computers (Continues)

8) CPU used in SIMD computers has to be specially designed.

- Because CPU in a single computer need to interact with other CPUs in the same computer. So, special design has to be taken care of so that proper communication can be done among CPUs. However, in MIMD model, communication between computers occurs via the network (e.g., Internet).

9) Processors in MIMD computers may be both cheaper and more powerful than processors in SIMD computers.

- In its basic form, MIMD consists of multi-computers, each with one CPU. Since the power of this model comes from having many interconnected computers, then we don’t need to spend more money on very powerful CPUs. Although, having powerful CPUs would improve the system.

Comparing SIMD with MIMD Computers (Continues)

10) Individual processors in an MIMD computer are more complex, because each processor has its own control unit.

- In its basic form, MIMD model have multi-computers with one CPU at each computer, so this CPU is responsible of running and interacting with the operating system at the given computer, so, MIMD is more complex. Meanwhile, processors in SIMD are controlled by only one central control unit and they share one large memory.

Comparing SIMD with MIMD Computers (Continues)

11) SIMD computers are better suited to parallel programs that require frequent synchronization.

   - If MIMD model is used in this case, then more synchronization implies more communication between computers which causes communication delays. Therefore, SIMD is more suitable here because communication is local among CPUs in the same machine.

12) Many MIMD computers have extra hardware to provide fast synchronization, which enables them to operate in SIMD mode as well.

   - In its advanced form, each computer in MIMD model may also have several CPUs (multiprocessors). In this case extra hardware and design considerations have to be paid so that proper CPUs functionality can be guaranteed.
   - Examples: CM-5 and DADO.

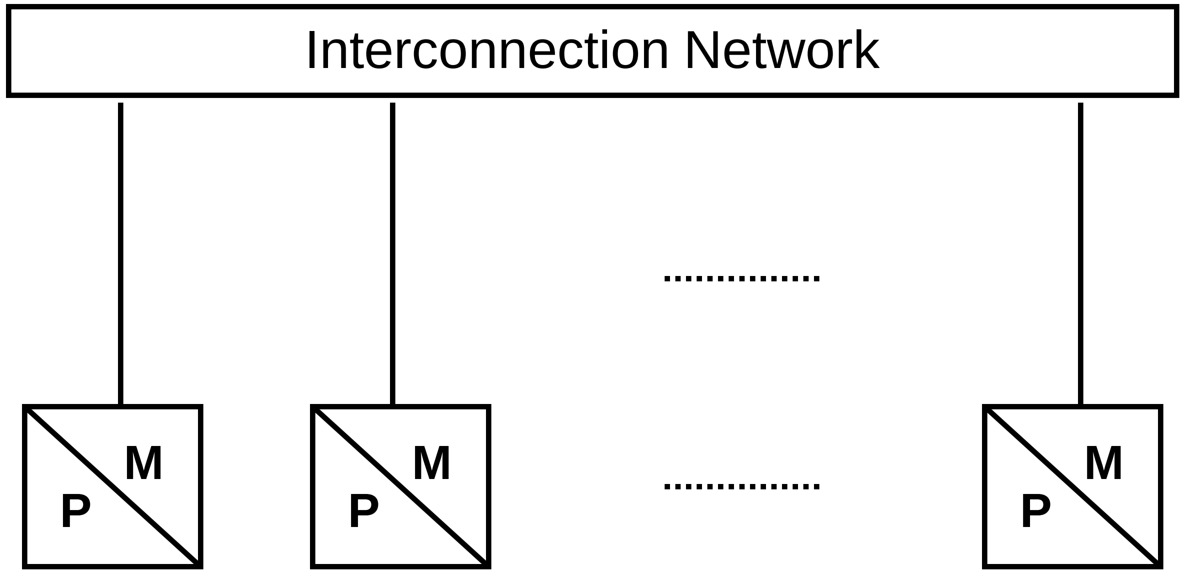
Address-Space Organization

- Two Models:
  1) Message-Passing Architecture.
  2) Shared-Address Space Architecture.

Message-Passing Architecture

- Processors are connected using a message-passing interconnection network.
- Each processor has its own memory called local or private memory, which is accessible only to that processor.
- This architecture is referred to as a distributed-memory or private-memory architecture.

A Typical Message-Passing Architecture



P: Processor **M: Memory**

- MIMD message-passing computers are referred to as multi-computers.
- Examples: Cosmic Cube, CM-5, and nCube2.

Shared-Address Space Computers

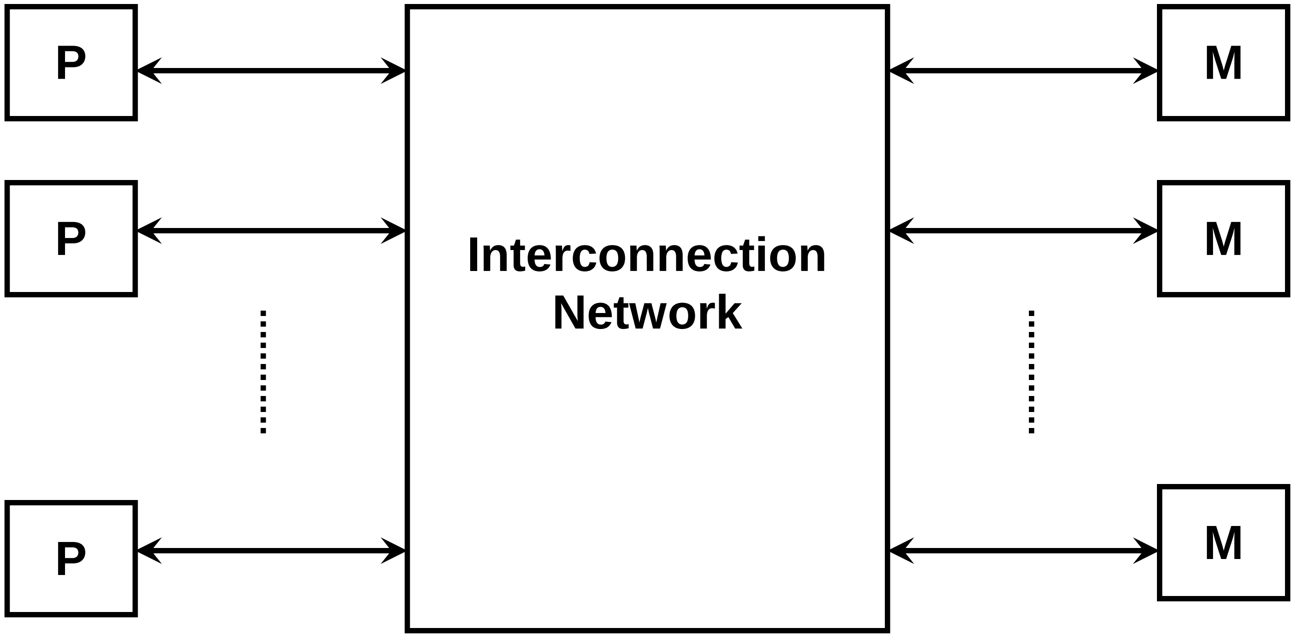
- Shared-address space computers are classified into two categories based on amount of time a processor takes to access local and global memories:

1) UMA computer: If the time taken by a processor to access any memory word in the system is identical.
2) NUMA computer: If the time to access a remote memory bank is longer than the time to access a local one.

Shared-Address Space Architecture

- MIMD shared-address space computers are referred to as multiprocessors (One Computer).
- Three Models for both Multicomputer and Multiprocessor-Computer:
  1) Uniform-Memory-Access (UMA).
  2) Non-Uniform-Memory-Access with local and global memories (NUMA).
  3) Non-Uniform-Memory-Access with local memory only (NUMA).

Uniform-Memory-Access Shared-Address Space Computer (UMA)



- These architectures are called shared-memory parallel computers.
- Examples: C.mmp and NYU Ultracomputer.

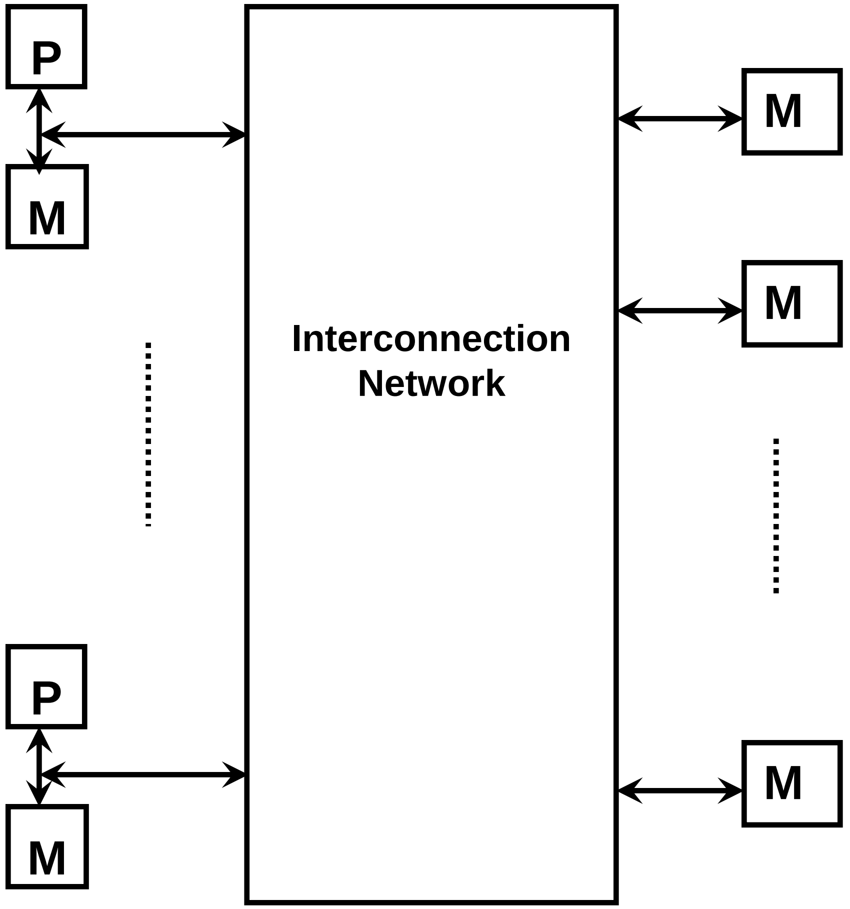
Uniform-Memory-Access Shared-Address Space Computer (UMA) Cont…

- **Drawbacks:**
  - The bandwidth of interconnection network must be substantial to ensure good performance.
  - Memory access through interconnection network can be slow, since a read or write request may have to pass through multiple stages in network.

Non-Uniform-Memory-Access Shared-Address Space Computer with local and global memories (NUMA)

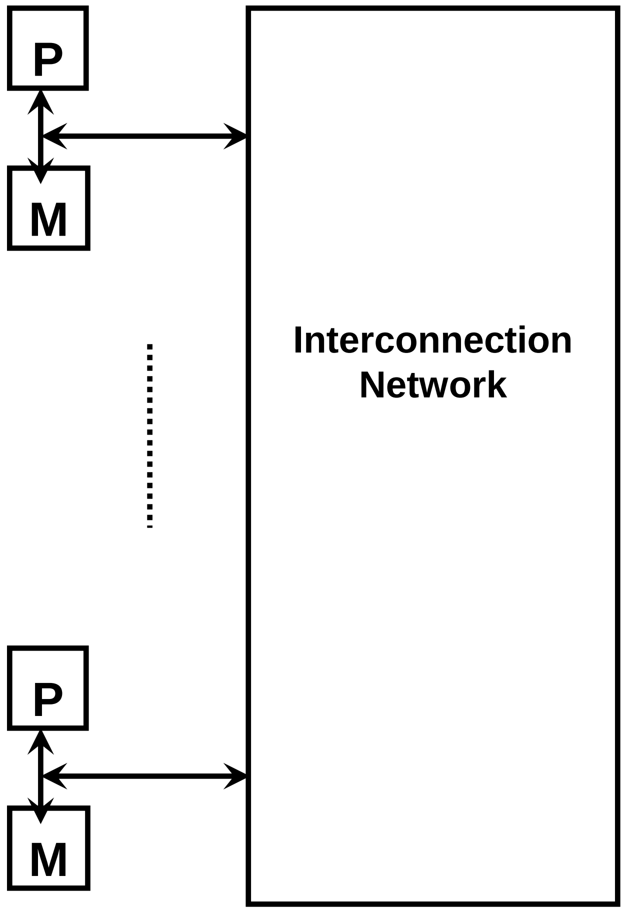
- Provide each processor with a local memory, which stores program being executed on processor and any non-shared data structures.

- Global data structures are stored in shared memory, which eliminates repeated memory references across interconnection network and improves performance.



Non-Uniform-Memory-Access Shared-Address Space Computer with local memory only (NUMA)

- Local memory access time is much smaller than remote memory access time.



Shared-Address Space Computers (Cont…)

- Most shared-address space computers have a local cache at each processor to increase their effective processor-memory bandwidth.

- Cache coherence occurs when a processor modifies a shared variable in its cache; after this modification, different processors have different values of the variable, unless copies of the variable in the other caches are simultaneously updated.

NUMA versus Message-Passing

- NUMA architecture is <u>similar</u> to a message-passing architecture; that is memory is physically distributed in both.
- The <u>major difference</u> between NUMA and Message-Passing is that a NUMA provides hardware support for read and write access to other processors' memories, whereas in a Message-Passing, remote access must be emulated by explicit message passing.
  - In other words:
    - NUMA is physical
    - Message Passing is logical

Shared-Address Space versus Message-Passing

- Shared-Address Space computers provide greater flexibility in programming than Message-Passing.

- A Shared-Address Space tends to be more expensive than Message-Passing.
  - Because it is physical (not virtual), which requires extra hardware expenses.

Interconnection Networks

- Shared-Address Space computers and Message-Passing computers can be constructed by connecting processors and memory units using a variety of interconnection networks.

- <u>Interconnection networks</u> can be classified as <u>static</u> or <u>dynamic.</u>

Static & Dynamic Networks

- Static networks consist of point-to-point communication links among processors and are also referred to as direct network.
- Static networks are used to construct message-passing computers.
- Dynamic networks are built using switches and communication links.
- Dynamic networks are referred to as indirect networks and are used to construct shared-address space computers.

Processor Granularity

- Coarse-Grain computers composed of a small number of very powerful processors.
  - Example: Cray Y-MP 8-16 processors each capable of Gflops (10^9 flop per second).
- Fine-Grain computers composed of a large number of less powerful processors.
  - Example: CM-2 up to 65,536 one-bit processors.

Processor Granularity Cont…

- The granularity of a parallel computer can be defined as the ratio of the time required for a basic computation operation over the time required for a basic communication

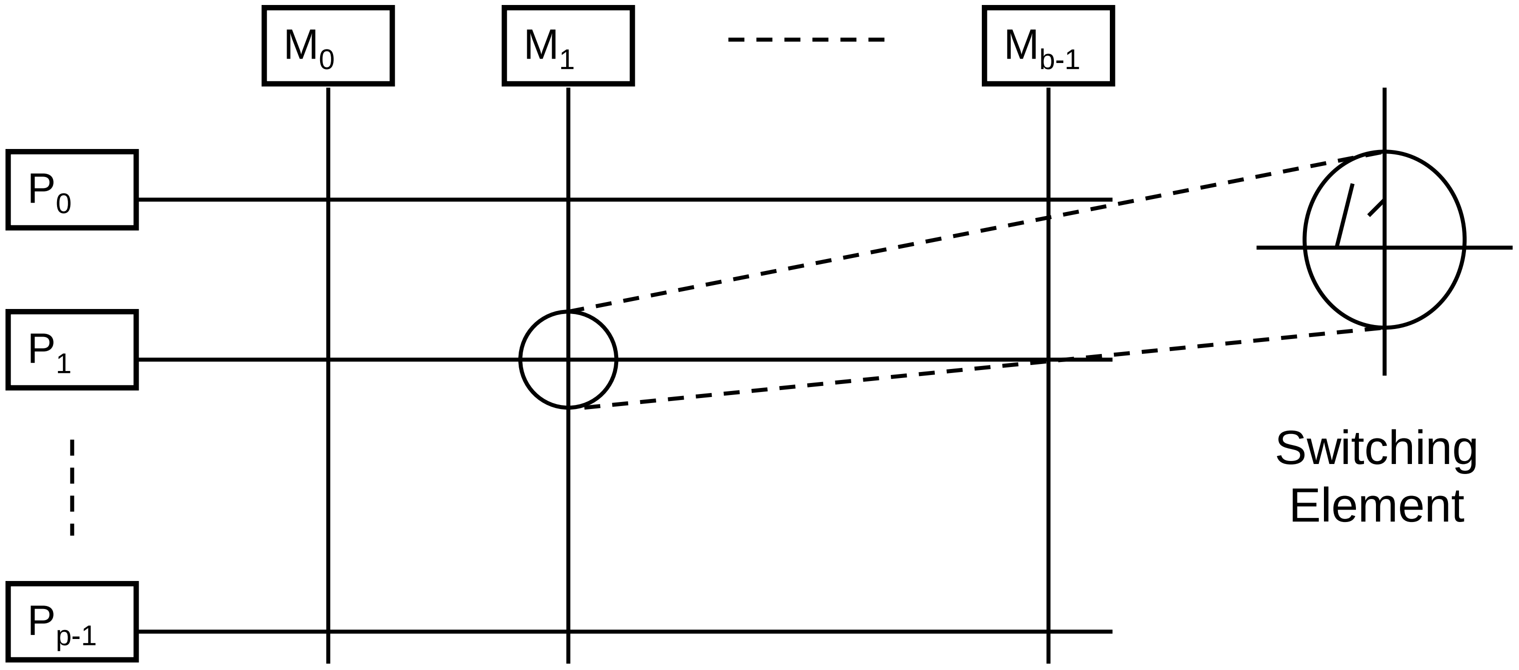
  (That is computation / communication).
- Parallel computers for which this ratio is small are suitable for algorithms requiring frequent communication (more communication than computation); that is, algorithms in which the grain size of the computation is small (fine-grain).
- Parallel computers for which this ratio is large are suited to algorithms that do not require frequent communication (more computation than communication) (coarse-grain).

Dynamic Interconnection Networks

1) Crossbar Switching Networks
2) Bus-based networks
3) Multistage Interconnection Networks

Crossbar Switching Networks

- A completely non-blocking crossbar switch connecting ***p*** processors to ***b*** memory banks.

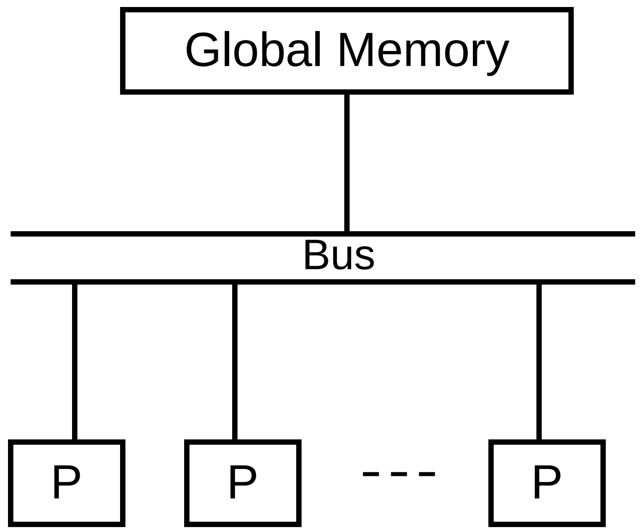


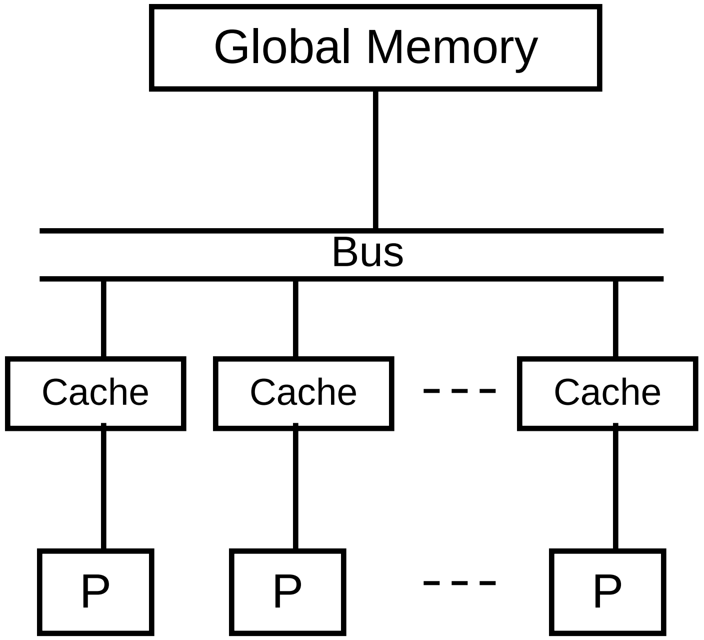
Crossbar Switching Networks (Continues)

- Crossbar switching network is a non-blocking network in the sense that the connection of a processor to a memory bank does not block the connection of any other processor to any other memory bank.
- Normally, ***b*** memory banks is greater than or equal to ***p*** processors, so that each processor has at least one memory bank to access.
- Total number of switching elements required to implement such a network is Θ(p x b).

Bus-Based Networks

- A typical bus-based architecture with no cache; and with cache memory at each processor.





Bus-Based Networks (Continues)

- Whenever a processor accesses global memory, that processor generates a request over the bus. The data is then fetched from memory over the same bus.

- Bus contention: Each processor spends an increasing amount of time waiting for memory access while the bus is in use by other processor due to large number of processors sharing same bus.
  - So, If a processor is using the shared memory through the bus, the other processors are blocked from accessing the bus.

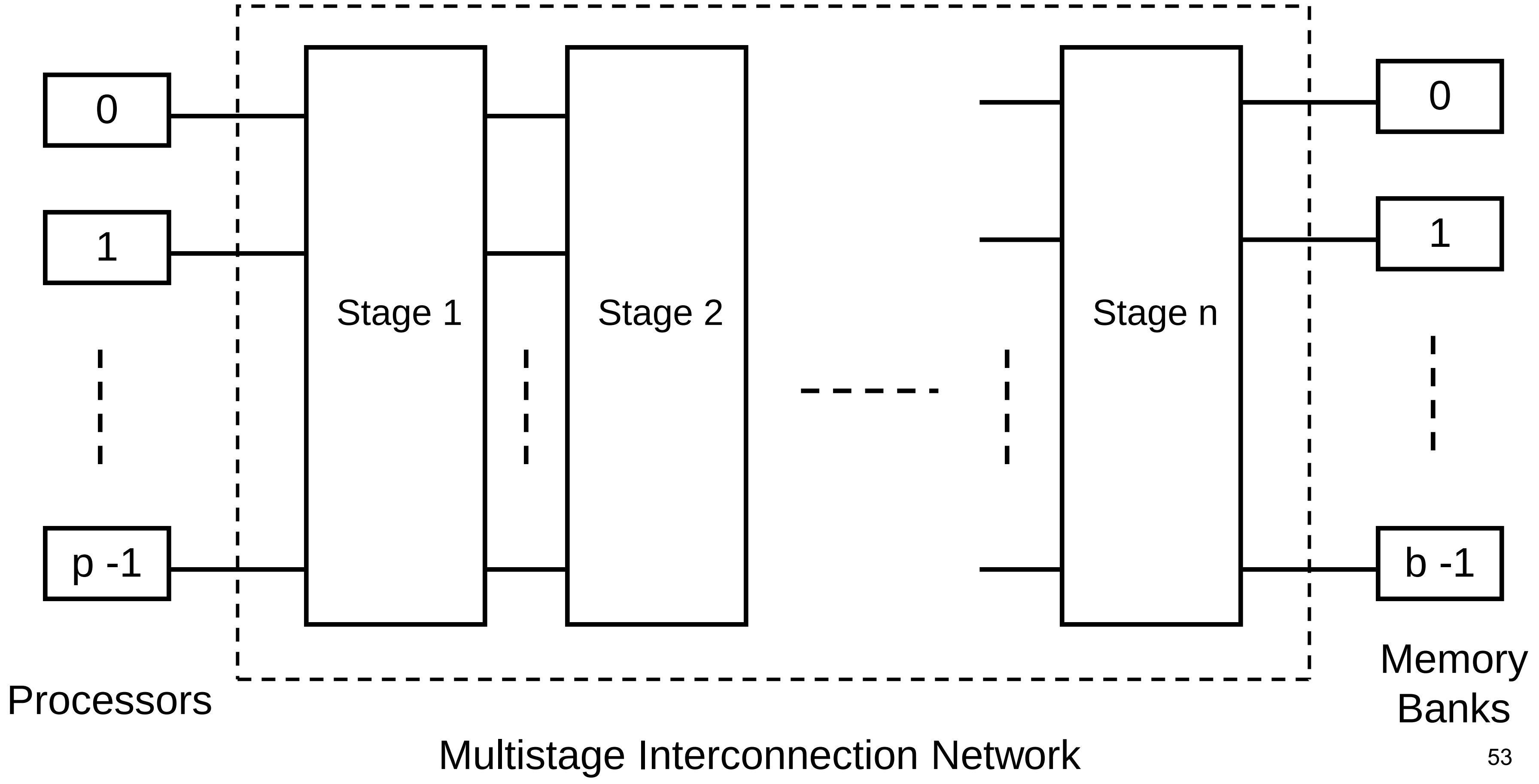
Bus-Based Networks (Continues)

- To reduce bus contention, each processor has a local cache memory:
  - Local cache memory reduces the total number of accesses to global memory, because when a reference is made to a memory location, subsequent references are likely to be made to memory locations in the neighborhood of this location, were fetched into a processor cache memory too (locality in space).

- Contention: Processors blocking each other.

Multistage Interconnection Networks

- Schematic of a multistage network consisting of ***p*** processors and ***b*** memory banks.



Example: Omega Network

- Omega network consists of log p stages; where p is the number of processors and also the number of memory banks.
  - Each stage of Omega network consists of an interconnection pattern that connects ***p*** inputs and ***p*** outputs.
  - A link exist between input ***i*** and output ***j*** if the following is true:

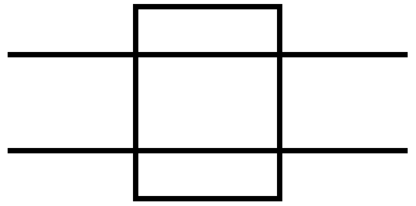
<u>Left Rotation</u>

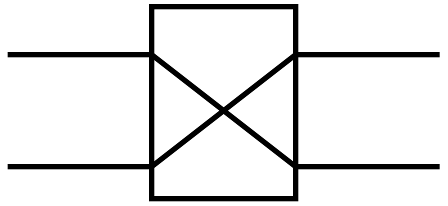
$$j = \begin{cases} 2 \times i; & 0 \le i \le (p/2) - 1 \quad \text{------(1)} \\ 2 \times i + 1 - p; & (p/2) \le i \le p - 1 \quad \text{------(2)} \end{cases}$$

  - This interconnection pattern is called <u>perfect shuffle.</u>

Omega Network

- In each stage of an Omega Network, a perfect shuffle interconnection pattern feeds into a set of (p / 2) switching elements.
- Two switching configurations of the 2 x 2 switch:

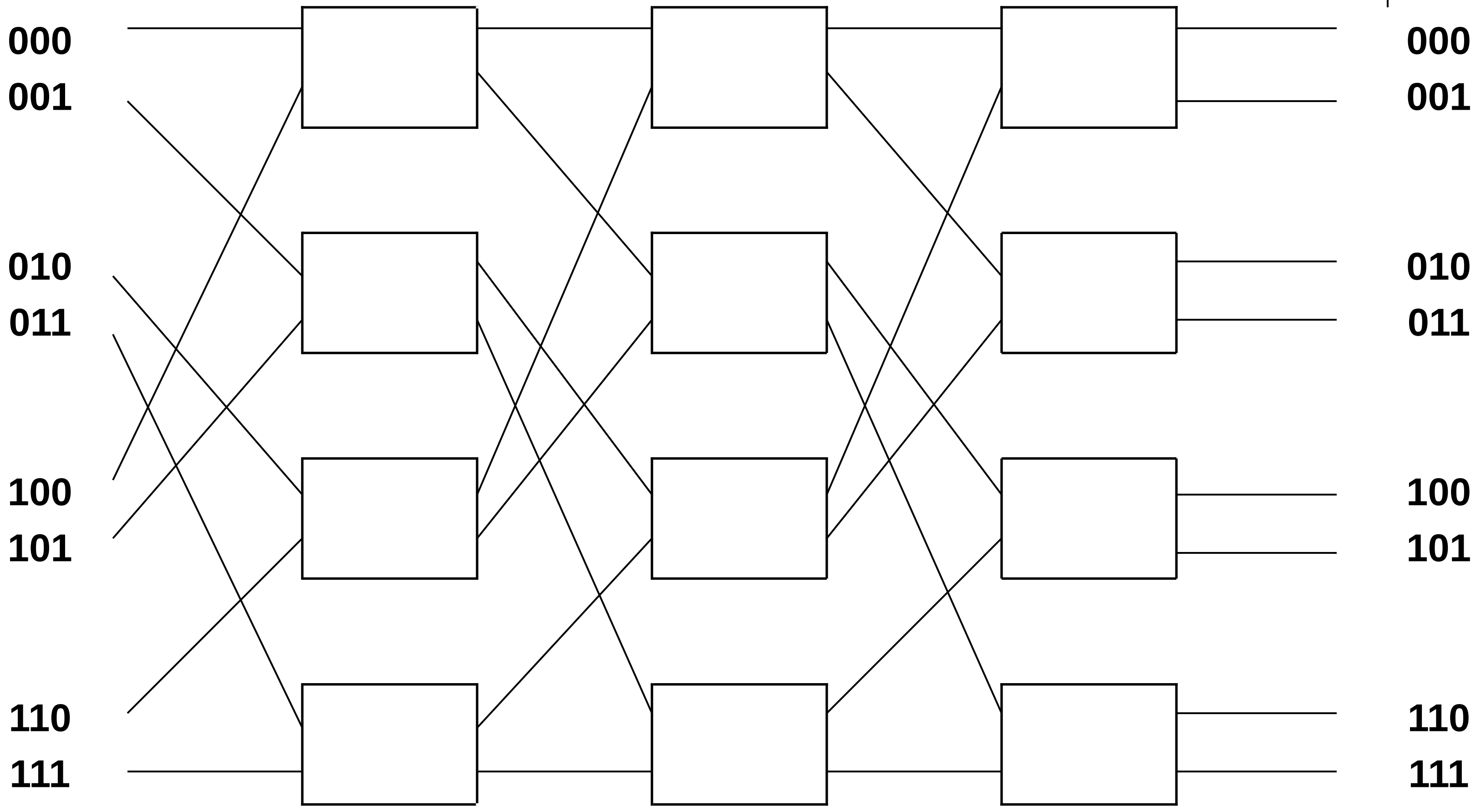

Pass Through


Cross Over

Example: Omega Network

- Assume the number of processors p = 8.
- Then the number of memory banks is also 8.
- So, the number of stages is 3; that is $\log_2(8) = 3$.
- The number of switching elements in each stage is 4; that is p / 2 = 8 / 2 = 4.

A Complete Omega Network connecting 8 inputs (processors) and 8 outputs (memory banks)



Processors
(Input nodes)

Interconnection Network
(Switching elements)

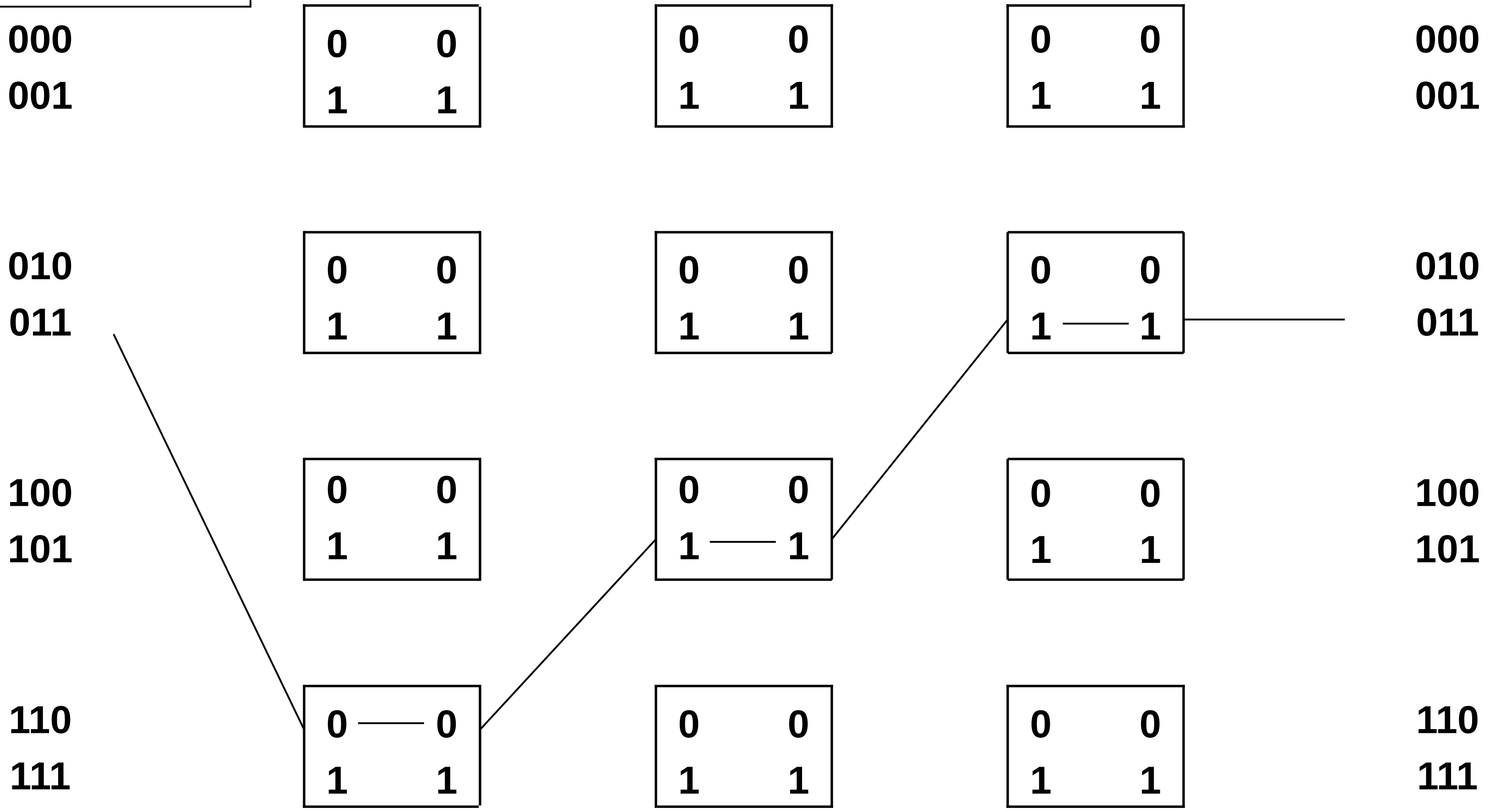
Memory Banks
(Output nodes)

How the previous Omega network is constructed?

Source is always as destination
Source: 011
Destination: 011

Use Circular Left Rotation 3 times for 3 stages
011 → 110 → 101 → 011

Each bit decides whether to go for the 0 route or the 1 route at each switch

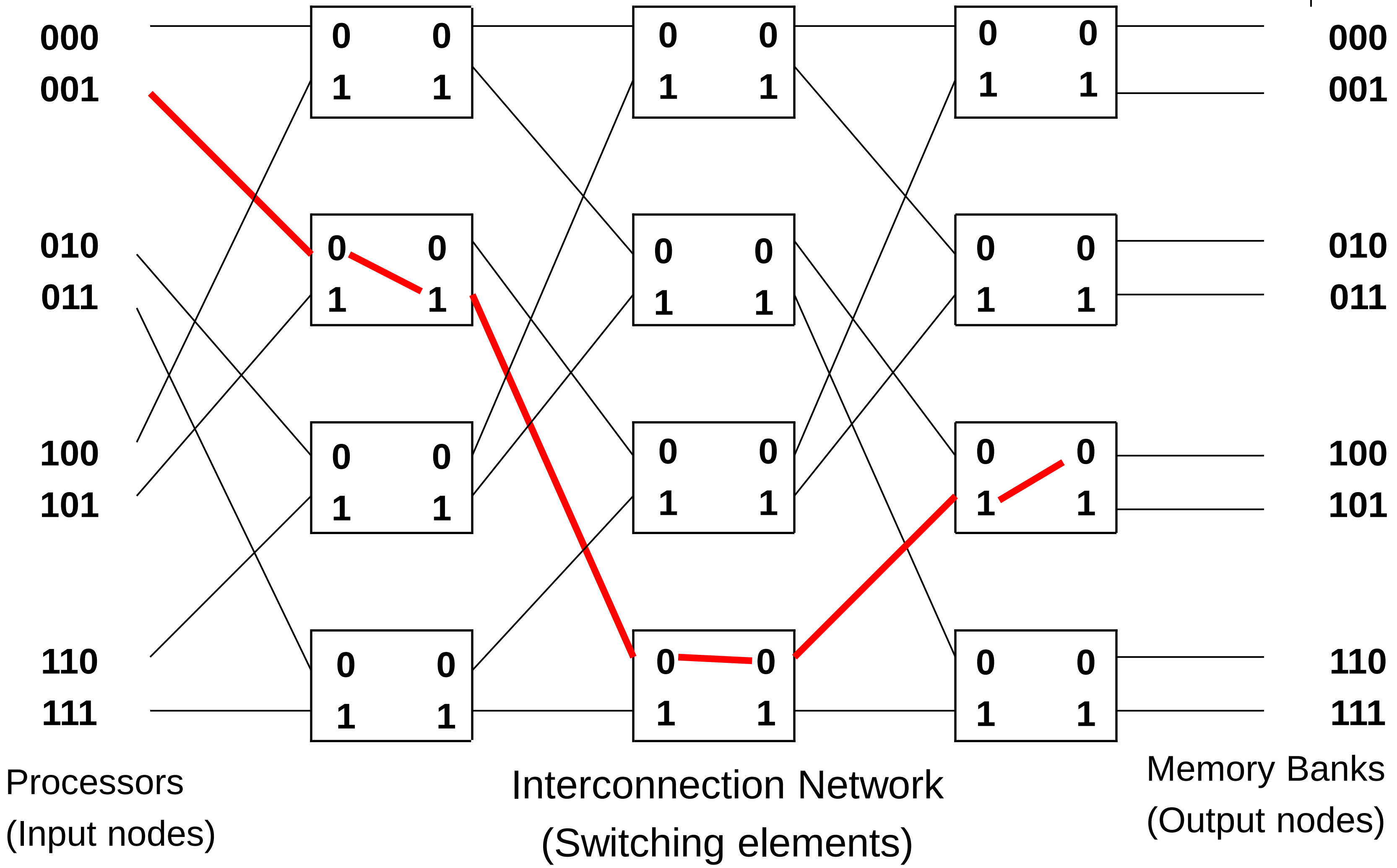


Omega Network (Continue)

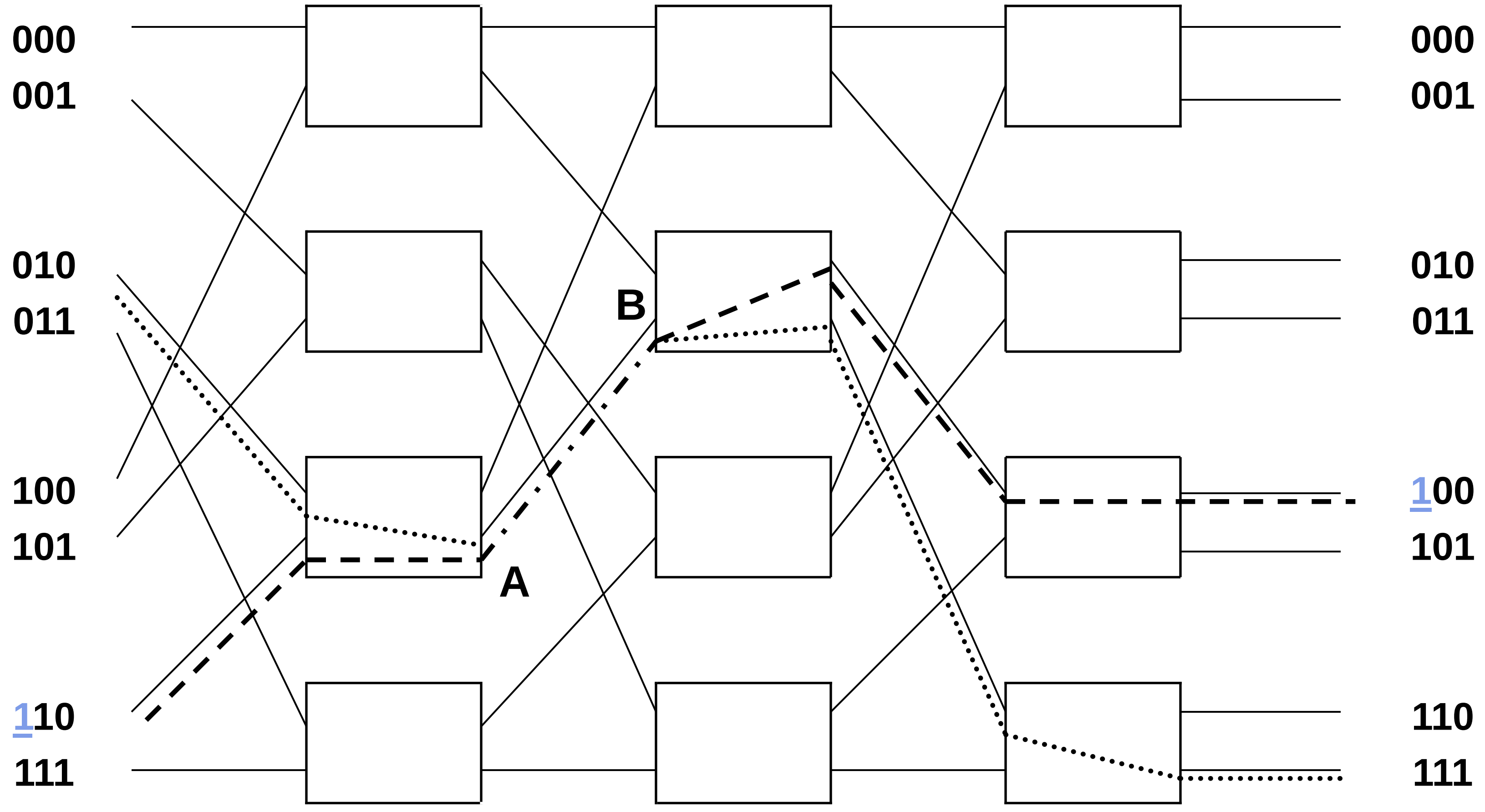
- Let ***s*** and ***t*** be the binary representation of the source and destination of the message.
- If the most significant bits of ***s*** and ***t*** are the same, then the message is routed in pass through mode by the switch.
- If these bits are different, then the message is routed through in cross over mode.
- This scheme is repeated at the next switching stage using the next most significant bit.

How routing is done in Omega network?

Simply use the destination bits and follow routes
Example: Source 001 and Destination is 100.



Example of Blocking in Omega Network: One of messages 010 to 111 or 110 to 100 is blocked at link AB.



Processors (Input nodes)

Interconnection Network (Switching elements)

Memory Banks (Output nodes)

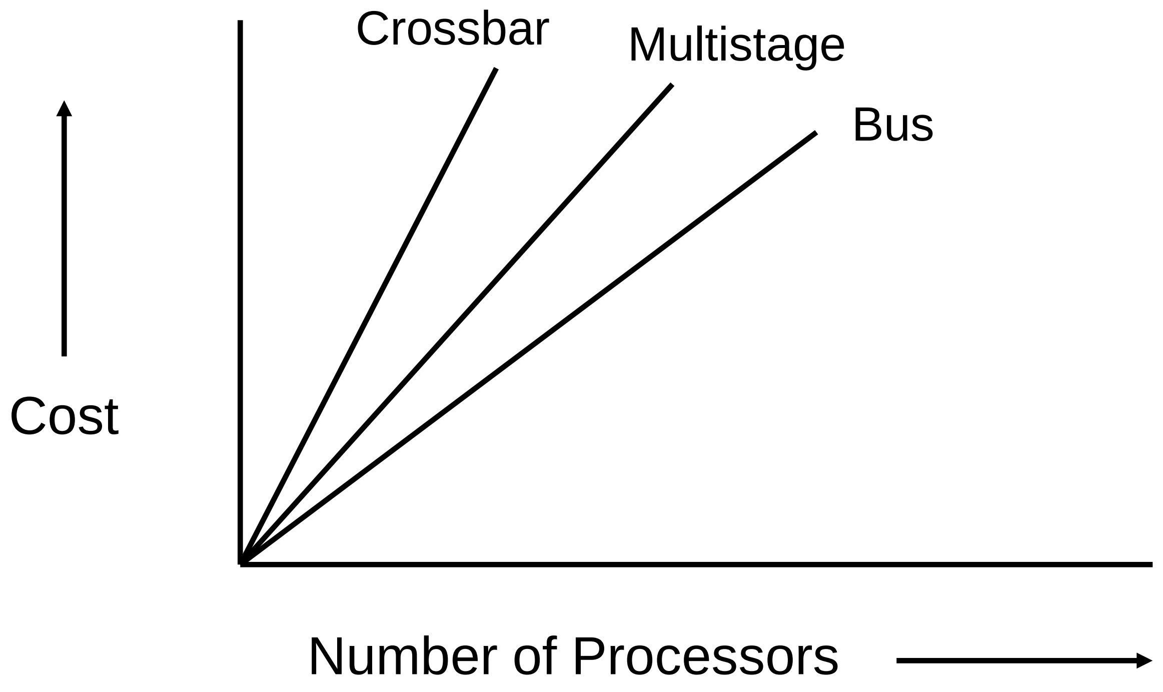
Omega Network (Continue)

- An Omega Network has $(p / 2) \times (\log_2 p)$ switching elements and the cost of such a network grows as $\Theta(p \log_2 p)$.
- Example: For a Complete Omega Network connecting 8 inputs (8 processors) and 8 outputs (8 memory banks) we have $(8 / 2) \times (\log_2 2^3) = 12$ switching elements and a cost of $\Theta(8 \times \log_2 2^3) = 8 \times 3 = 24$.
- Note: This cost $\Theta(p \log_2 p)$ is less than $\Theta(p^2)$ cost of a complete crossbar switch.

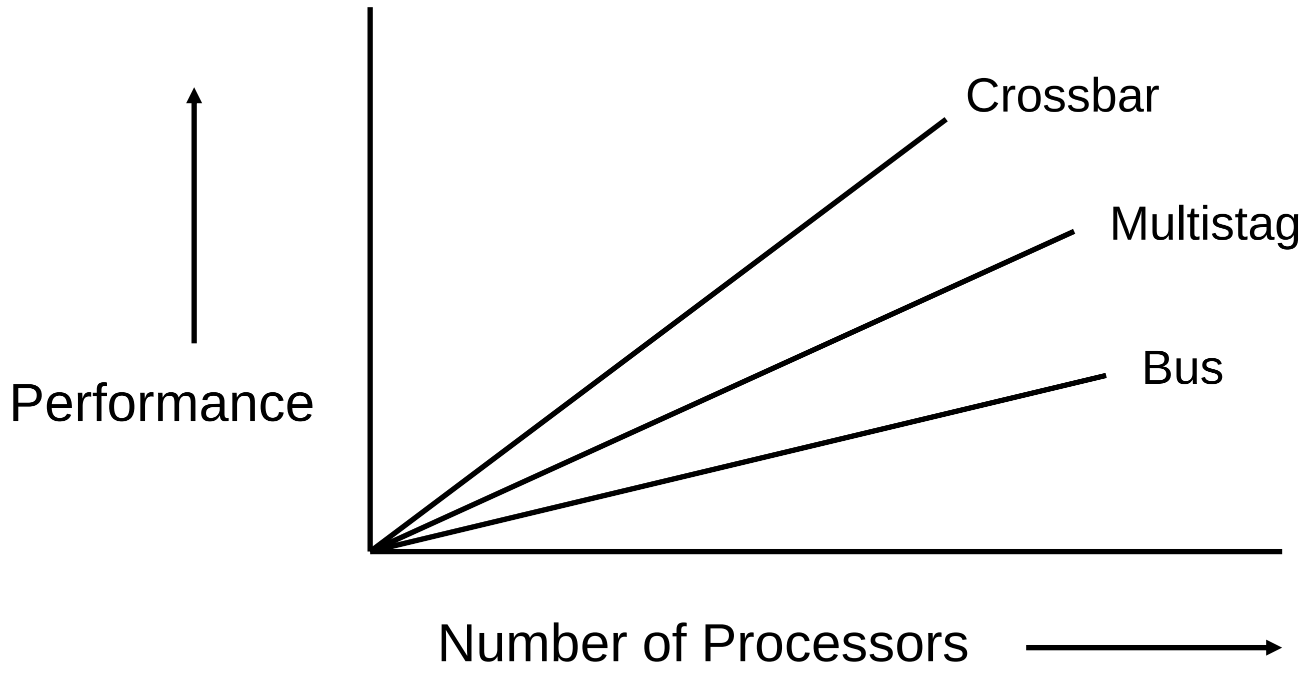
Crossbar, Shared Bus, & Multistage Interconnection Networks

- The crossbar interconnection network is scalable in terms of performance but un-scalable in terms of cost.
- The shared bus network is scalable in terms of cost but un-scalable in terms of performance.
- The multistage interconnection networks is more scalable than the bus in terms of performance and more scalable than crossbar in terms of cost.

Cost versus Number of Processors



Performance versus Number of Processors

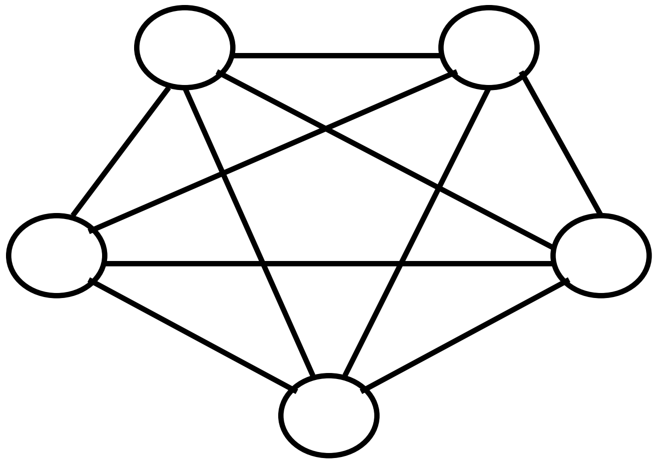


Static Interconnection Networks

- Message-Passing architecture typically use static interconnection networks to connect processors.
- Types of Static Interconnection Networks:
  - Completely-Connected Network
  - Star-Connected Network
  - Linear Array and Ring
  - Mesh Network
  - Tree Network
  - Hypercube Network
- Evaluating Static Interconnection Networks:
  - Diameter; Connectivity; Bisection Width; Bisection Bandwidth; Cost; and Size.

Completely-Connected Network

- Each processor has a direct communication link to every other processor in the network.
- A Completely-Connected Network of 5 Processors:

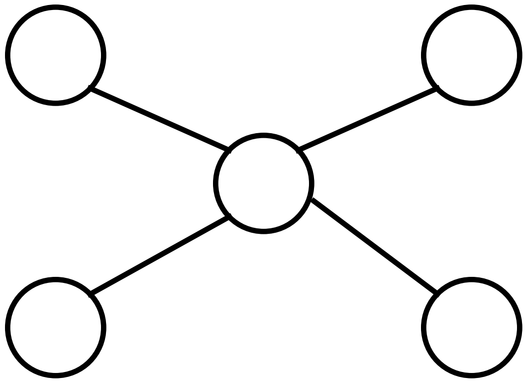


Advantages of Completely-Connected Network

1) In this network a processor can send a message to another processor in a single step, since a communication link exists between them.
2) The communication between any input / output pair does not block communication between any other pair.
3) The network supports communications over multiple channels originating at the same processor.

Star-Connected Network

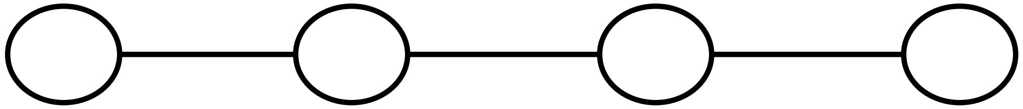
- One processor acts as the central processor and every other processor has a communication link connecting it to this processor.
- A Star-Connected Network of 5 Processors:



- Communication between any pair of processors is routed through the central processor.
- The central processor is the **bottleneck** in the star topology.

Linear Array and Ring

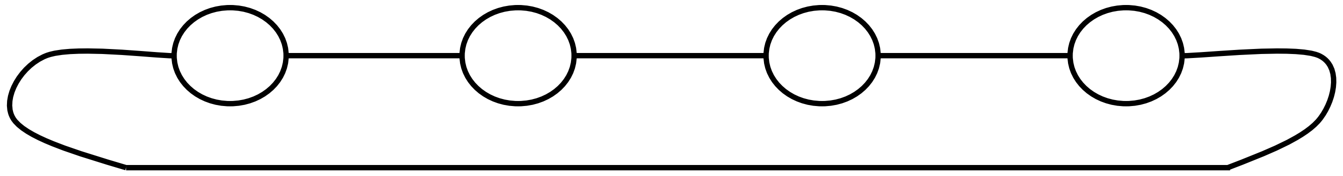
- A 4-Processor **Linear Array**:



- Each processor in this network (except processors at the ends) has a direct communication link to 2 other processors.
- If a wraparound connection is provided between the processors at the ends, then it becomes a **ring**.
- A **linear array** with a wraparound connection is called a **ring**.

Ring (Continues)

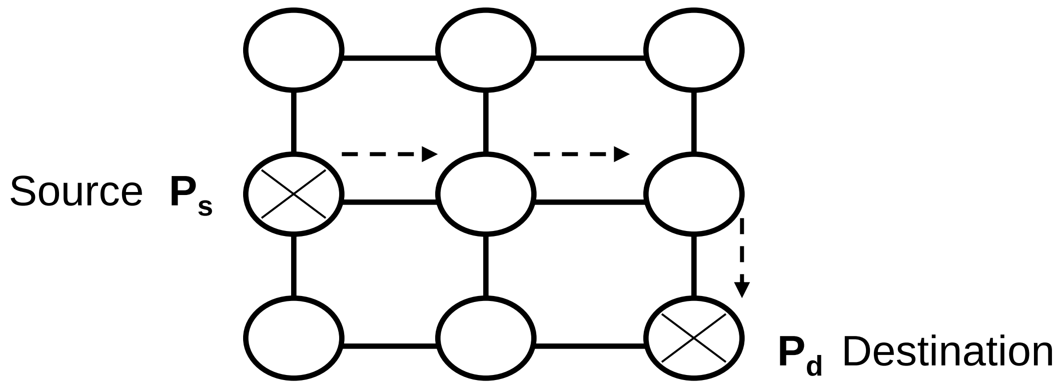
- A 4-Processor **Ring**:



- One way of communicating a message between processors is by repeatedly passing it to the processor immediately to the right (or left, depending on which direction yields a shorter path) until it reaches its destination.

Mesh Network

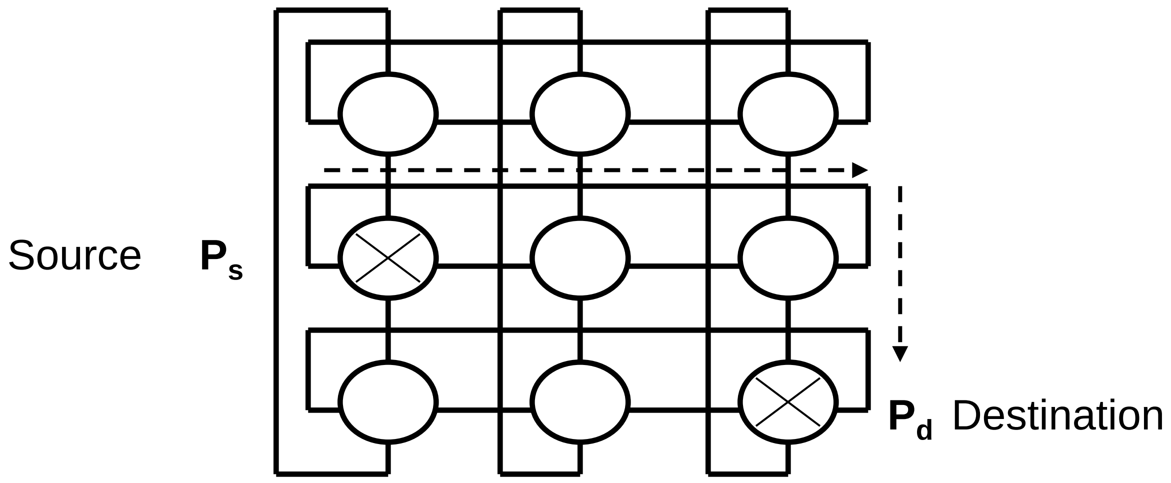
- A **two-dimensional mesh** of 9 processors:



- Routing a message from processor P_s to processor P_d.
- In a 2-D mesh, each processor has a direct communication link connecting it to 4 other processors, except the ones at the first and last rows and columns.

Mesh Network (Continues)

- A **wraparound mesh** or torus (2-D):
  - Routing a message from processor P_s to processor P_d.
- In wraparound 2-D mesh, each processor has a direct communication link connecting it to 4 other processors.



Mesh Network (Continues)

- The Cray T3D is an example of 3-D Mesh:

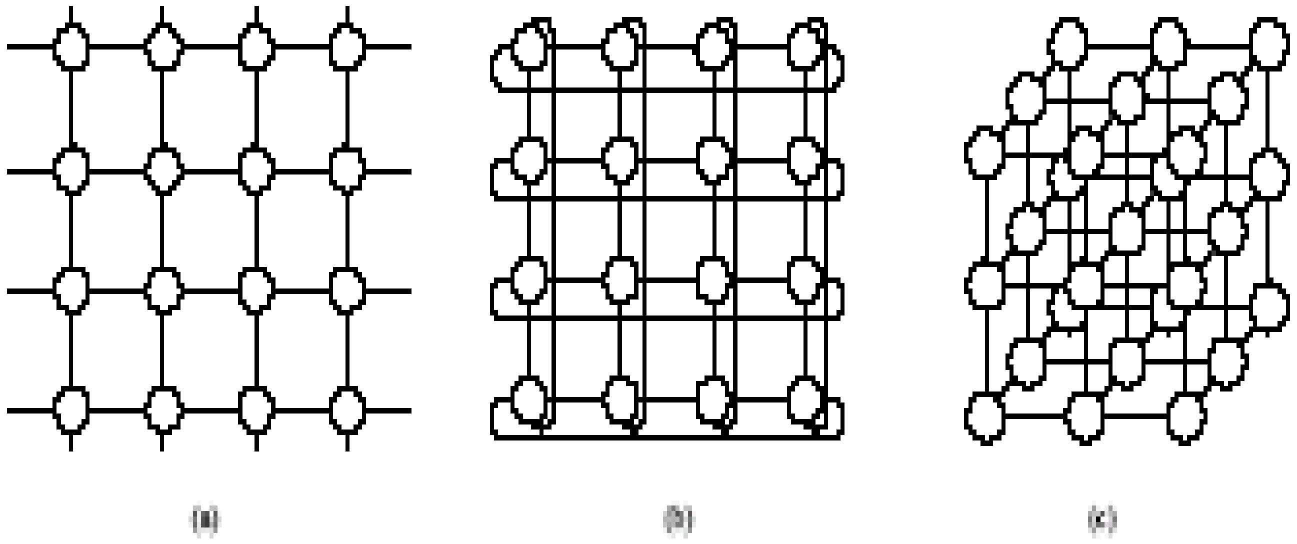


Figure 2.16 Two and three dimensional meshes: (a) 2-D mesh with no wraparound; (b) 2-D mesh with wraparound link (2-D torus); and (c) a 3-D mesh with no wraparound.

Tree Network

- A **tree** is a connected network without cycles.
- A **tree** network is one in which there is only one path between any pair of processors.
- Both **linear arrays** and **star-connected** networks are special cases of tree network.
- **Complete binary tree** networks and message routing in them:
  - ➢ <u>Static tree networks:</u> A processor at each node of the tree.
  - ➢ <u>Dynamic tree networks:</u> Nodes at intermediate levels are switching elements and the leaf nodes are processors.

Static vs. Dynamic Tree Networks

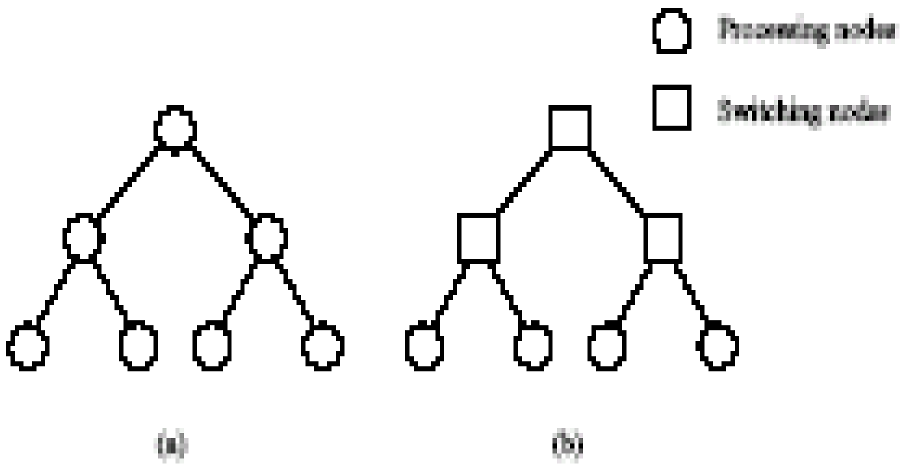


Figure 2.18 Complete binary tree networks: (a) a static tree network; and (b) a dynamic tree network.

Tree Networks (Continues)

- Tree networks suffer from a communication **bottleneck** at higher levels of the tree.
- For example, when many processors in the left sub-tree of a node communicate with processors in the right sub-tree, the root node has to handle all the messages.
- This problem can be solved by increasing the number of communication links between processors that are closer to the root.

Example: A Fat Tree Network

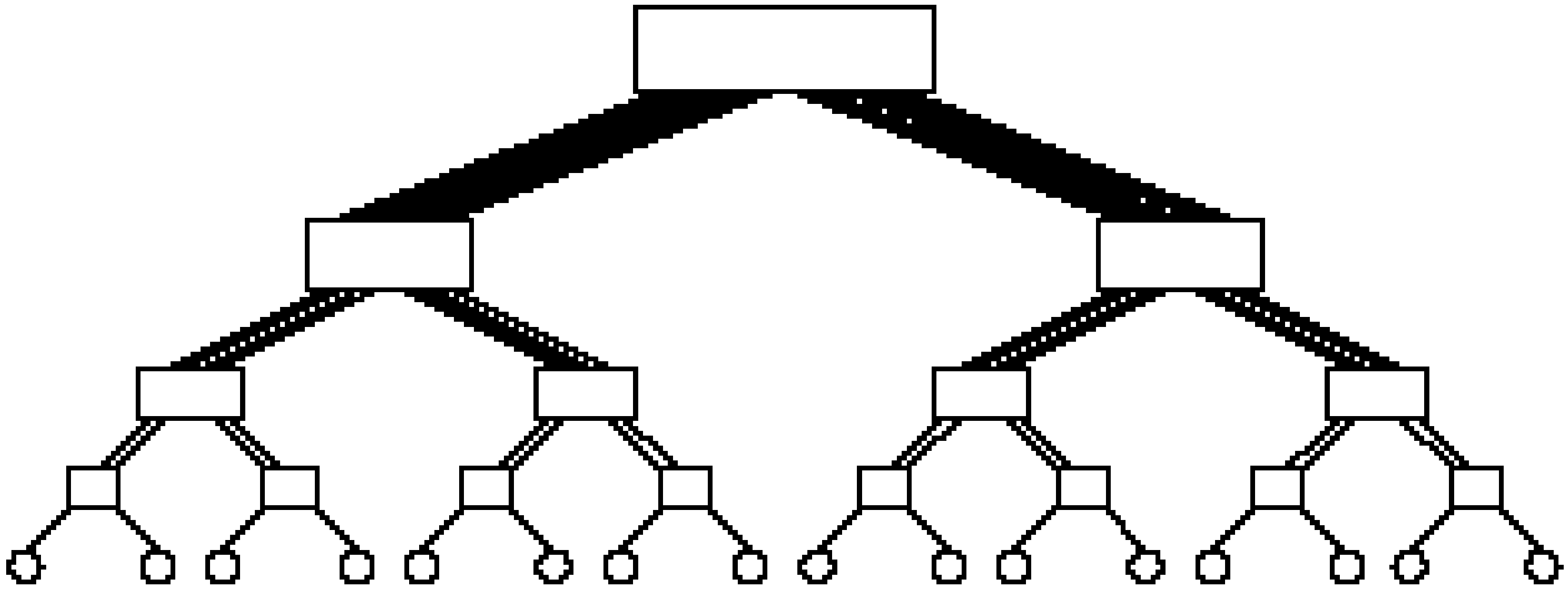
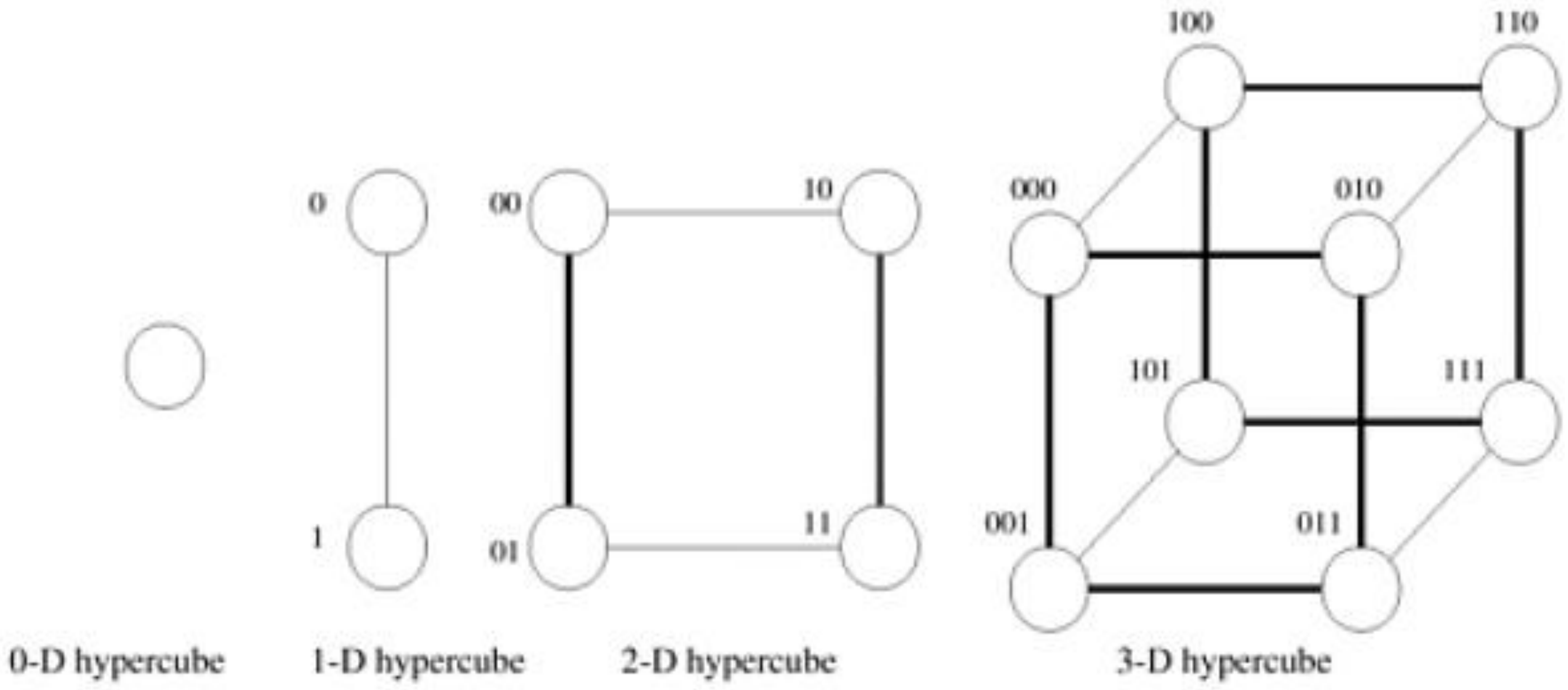


Figure 2.19 A fat tree network of 16 processing nodes.

Hypercube Network

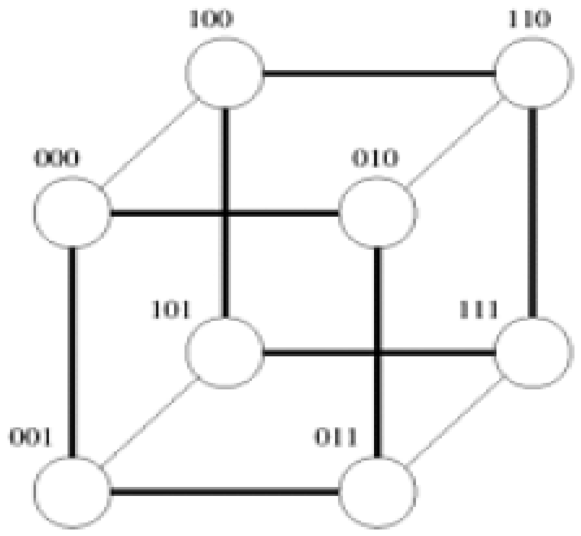
- A d-dimensional hypercube consists of $p = 2^d$ processors.
- A zero-dimensional hypercube is a single processor.
- A one-dimensional hypercube is constructed by connecting 2 zero-dimensional hypercubes.
- In general, a (d+1)-dimensional hypercube is constructed by connecting the corresponding processors of 2 d-dimensional hypercubes.

Hypercubes of Dimensions Zero to Three



Hypercube Labels

- When a (d+1)-dimensional hypercube is constructed by connecting 2 d-dimensional hypercubes, the labels of the processors of one hypercube are prefixed with 0 and those of the second hypercube are prefixed with a 1.



3-D hypercube

Two 2-D hypercubes connected to each other.

Labels of the first one starts with 0.

Labels of the second one starts with 1.

Properties of Hypercube Network

1) Two processors are connected by a direct link if and only if the binary representation of their labels differ at exactly one bit position.
2) In a d-dimensional hypercube, each processor is directly connected to d other processors.
3) A d-dimensional hypercube can be partitioned into 2 (d – 1)-dimensional sub-cubes.

Properties (Continues)

4) Hamming distance between 2 labels (***s*** & ***t***) of 2 processors in hypercube is the total number of bit positions at which these 2 labels differ.

Example: **Hamming Distance**:

$\underline{01}1$ and $\underline{10}1 = 2$

$\underline{101}$ and $\underline{010} = 3$

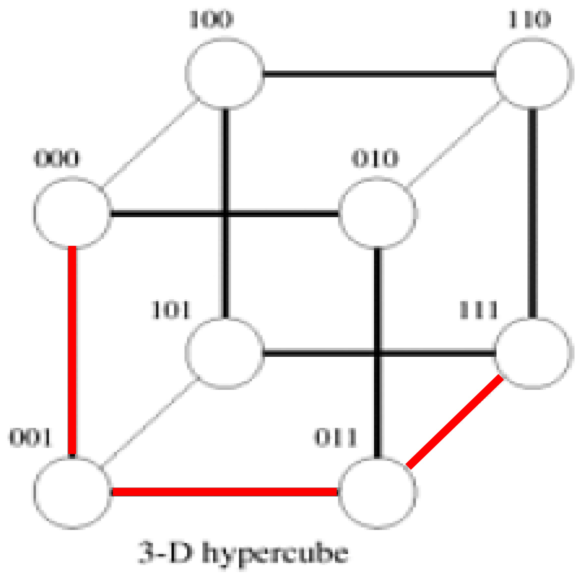
Properties (Continues)

- Hamming distance between ***s*** and ***t*** labels is the number of bits that are 1 in the binary representation of ***s*** **X-OR** ***t***.
- Example:
  - s = 011, t = 101, s **x-or** t =

```
011
101
-----
110 (So, hamming distance = 2)
```

Properties (Continues)

- The number of communication links in the shortest distance between 2 processors is the Hamming Distance between their labels.



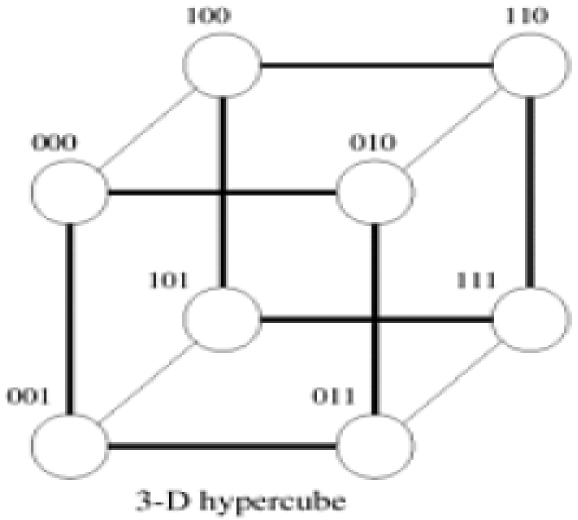
3-D hypercube

Number of links between processors 000 and 111 is 3.

Evaluating Static Interconnection Networks

1) **Diameter:** is the **maximum distance** between any 2 processors in the network.

- The **distance** between 2 processors is defined as the shortest path (in terms of number of links) between them.
- Networks with smaller diameters are better, since distance largely determines communication time.

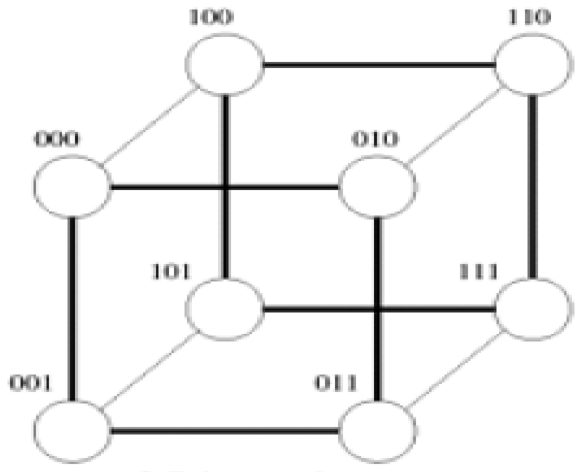


In this case, diameter is 3.

Evaluating Static Interconnection Networks

2) **Connectivity:** is a measure of the multiplicity of paths between any 2 processors.

- A network with high connectivity is desirable, because it lowers contention for communication resources.
- For example, processor 000 and 111 are connected using 6 different paths. If one path is busy with other communications, we can follow another path.

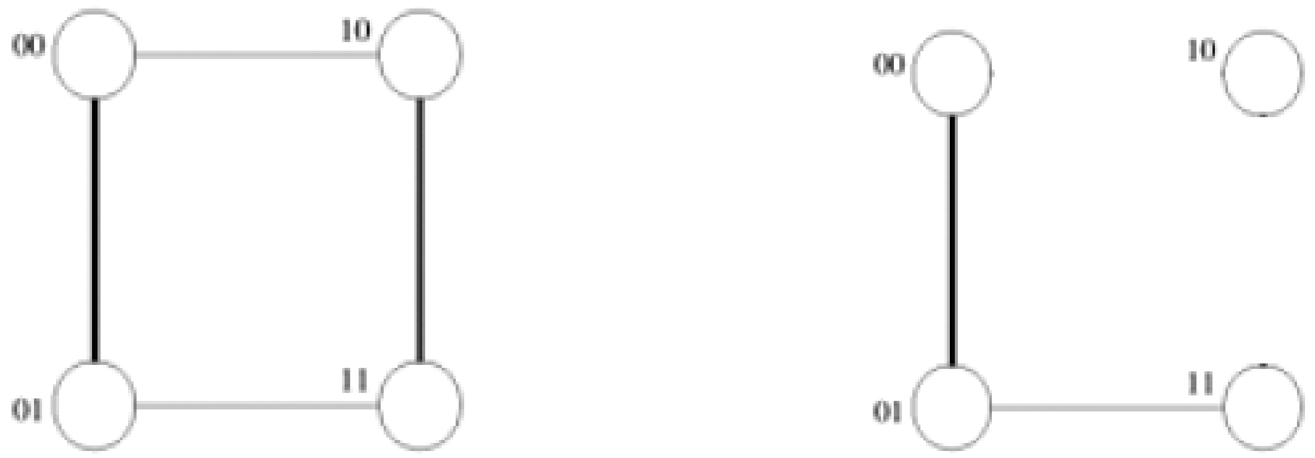


3-D hypercube

Evaluating Static Interconnection Networks

- **Arc connectivity:** is the minimum number of arcs that must be removed from the network to break it into 2 disconnected networks.

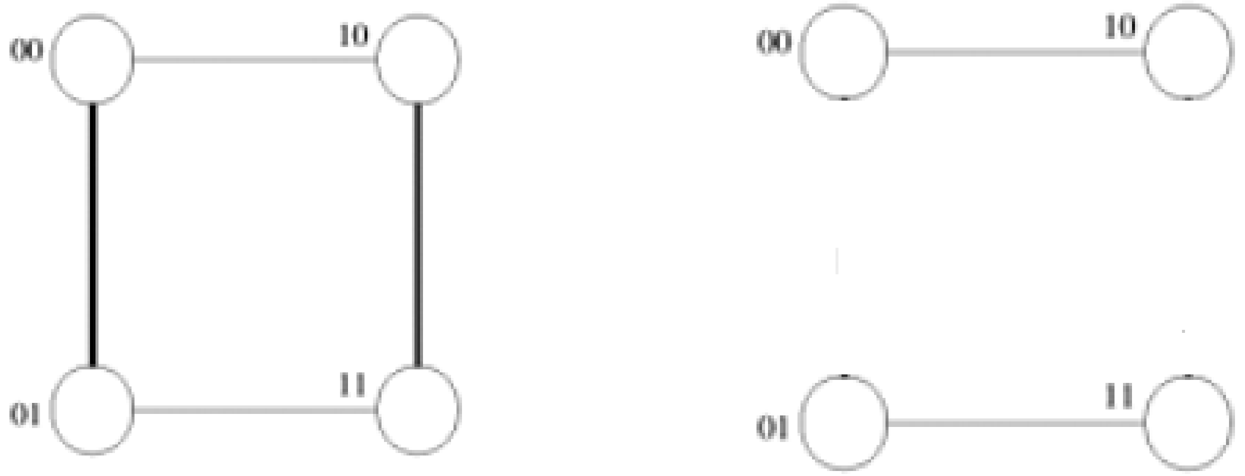
In this case, arc connectivity = 2.



Evaluating Static Interconnection Networks

3) **Bisection Width:** is the minimum number of communication links that have to be removed to partition the network into 2 equal halves.

In this case, bisection width = 2.

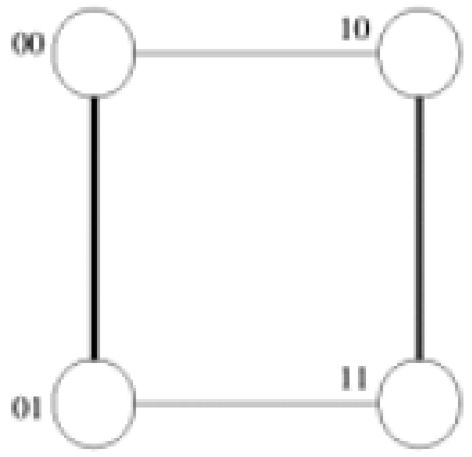


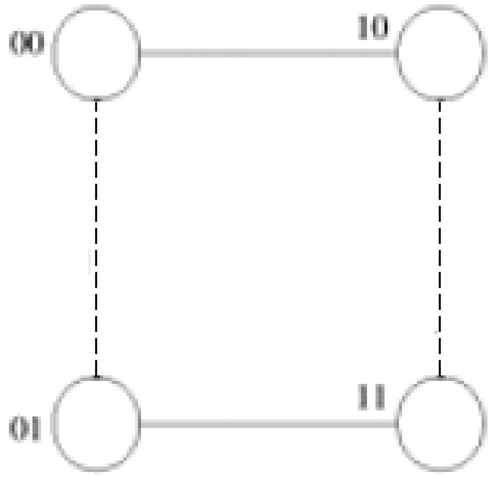
Evaluating Static Interconnection Networks

4) **Bisection Bandwidth** of a network is the minimum volume of communication allowed between any 2 halves of the network with an equal number of processors.

OR

Bisection Bandwidth = Bisection Width x Channel Bandwidth





In this case, assume channel bandwidth is 10kb/s, then bisection bandwidth = 2 x 10 = 20.

Evaluating Static Interconnection Networks

- **Channel bandwidth:** is the peak rate at which data can be communicated between the ends of a communication link.

  OR

  Channel Bandwidth = Channel Rate x Channel Width
- **Channel width** is equal to the number of physical wires in each communication link.

  OR it is the number of bits that can be communicated simultaneously over a link connecting 2 processors.
- **Channel rate:** is the peak rate at which a single physical wire can deliver bits.

Evaluating Static Interconnection Networks

5) **Cost:** is the number of communication links or the number of wires required by the network.

6) **Size:** is the number of nodes in the network.

Characteristics of various static network topologies connecting p processors

Network	Diameter	Bisection Width	Arc Connectivity	Cost
Completely Connected	1	p/2 x p/2 = p^2/4	p – 1	p x (p-1) / 2
Star	2	-	1	p – 1
Linear Array	p – 1	1	1	p – 1
Ring	FL(p / 2)	2	2	p
Hypercube	log p	p / 2	log p	(p log p) / 2

Characteristics of various static network topologies connecting p processors

- **Exercise:** Find the diameter, bisection width, arc connectivity, and cost of the following networks:
  - Complete binary tree – when number of nodes is even.
  - Two-dimensional mesh non-wraparound.
  - Two-dimensional mesh wraparound.