MULTIPROCESSORS

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

TERMINOLOGY

Parallel Computing

Simultaneous use of multiple processors, all components of a single architecture, to solve a task. Typically processors identical, single user (even if machine multiuser)

Distributed Computing

Use of a network of processors, each capable of being viewed as a computer in its own right, to solve a problem. Processors may be heterogeneous, multiuser, usually individual task is assigned to a single processors

Concurrent Computing

All of the above?

TERMINOLOGY

Supercomputing

Use of fastest, biggest machines to solve big, computationally intensive problems. Historically machines were vector computers, but parallel/vector or parallel becoming the norm

Pipelining

Breaking a task into steps performed by different units, and multiple inputs stream through the units, with next input starting in a unit when previous input done with the unit but not necessarily done with the task

Vector Computing

Use of vector processors, where operation such as multiply broken into several steps, and is applied to a stream of operands ("vectors"). Most common special case of pipelining

Systolic

Similar to pipelining, but units are not necessarily arranged linearly, steps are typically small and more numerous, performed in lockstep fashion. Often used in special-purpose hardware such as image or signal processors

SPEEDUP AND EFFICIENCY

A: Given problem

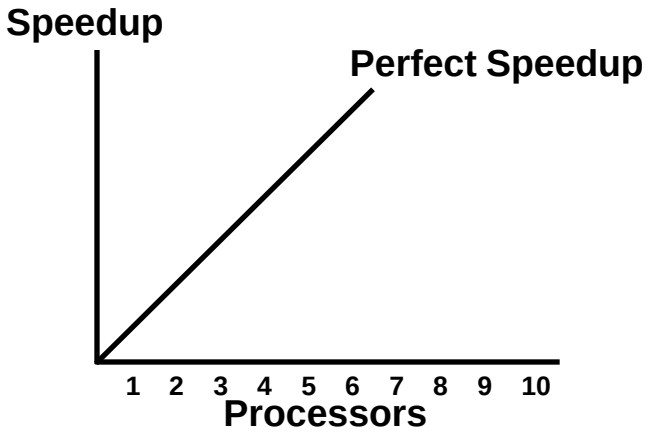
$T^*(n)$: Time of best sequential algorithm to solve an instance of A of size n on 1 processor

$T_p(n)$: Time needed by a given parallel algorithm and given parallel architecture to solve an instance of A of size n, using p processors

Note: $T^*(n) \leq T_1(n)$

Speedup: $T^*(n) / T_p(n)$

Efficiency: $T^*(n) / [pT_p(n)]$



Speedup should be between 0 and p, and Efficiency should be between 0 and 1

Speedup is *linear* if there is a constant c > 0 so that speedup is always at least cp.

AMDAHL'S LAW

Given a program

f **: Fraction of time that represents operations that must be performed serially**

Maximum Possible Speedup: *S*

$$S \leq \frac{1}{f + (1 - f) / p}$$ **, with p processors**

$$S < 1 / f$$ **, with unlimited number of processors**

- Ignores possibility of new algorithm, with much smaller *f*

- Ignores possibility that more of program is run from higher speed memory such as Registers, Cache, Main Memory

- Often problem is scaled with number of processors, and *f* is a function of size which may be decreasing (Serial code may take constant amount of time, independent of size)

FLYNN's HARDWARE TAXONOMY

I: Instruction Stream
D: Data Stream

$$\left[\begin{matrix}M\\S\end{matrix}\right] I \left[\begin{matrix}M\\S\end{matrix}\right] D$$

SI: Single Instruction Stream
- All processors are executing the same instruction in the same cycle
- Instruction may be conditional
- For Multiple processors, the control processor issues an instruction

MI: Multiple Instruction Stream
- Different processors may be simultaneously executing different instructions

SD: Single Data Stream
- All of the processors are operating on the same data items at any given time

MD: Multiple Data Stream
- Different processors may be simultaneously operating on different data items

SISD : standard serial computer
MISD : very rare
MIMD and *SIMD* : Parallel processing computers

COUPLING OF PROCESSORS

Tightly Coupled System

- Tasks and/or processors communicate in a highly synchronized fashion
- Communicates through a common shared memory
- Shared memory system

Loosely Coupled System

- Tasks or processors do not communicate in a synchronized fashion
- Communicates by message passing packets
- Overhead for data exchange is high
- Distributed memory system

GRANULARITY OF PARALLELISM

Granularity of Parallelism

Coarse-grain

- **A task is broken into a handful of pieces, each of which is executed by a powerful processor**
- **Processors may be heterogeneous**
- **Computation/communication ratio is very high**

Medium-grain

- **Tens to few thousands of pieces**
- **Processors typically run the same code**
- **Computation/communication ratio is often hundreds or more**

Fine-grain

- **Thousands to perhaps millions of small pieces, executed by very small, simple processors or through pipelines**
- **Processors typically have instructions broadcasted to them**
- **Compute/communicate ratio often near unity**

MEMORY

Shared (Global) Memory

- A Global Memory Space accessible by all processors
- Processors may also have some local memory

Distributed (Local, Message-Passing) Memory

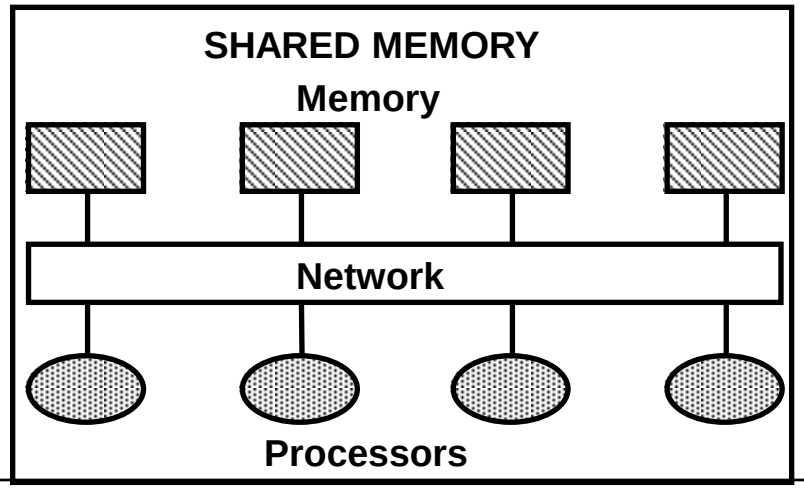
- All memory units are associated with processors
- To retrieve information from another processor's memory a message must be sent there

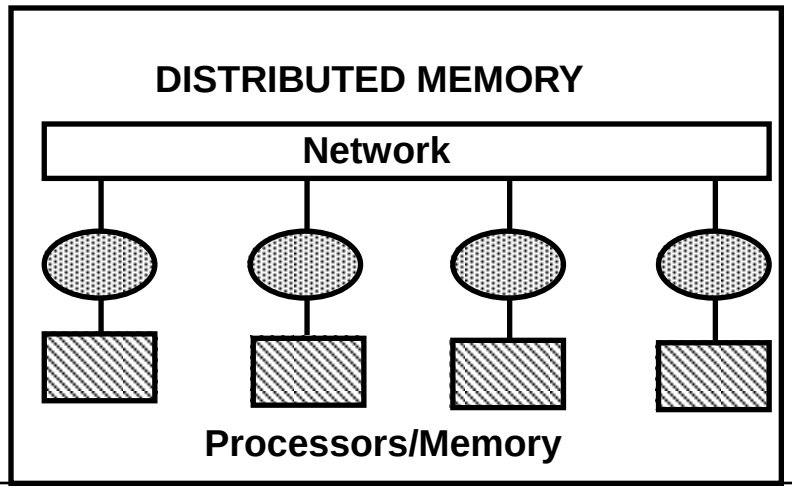
Uniform Memory

- All processors take the same time to reach all memory locations

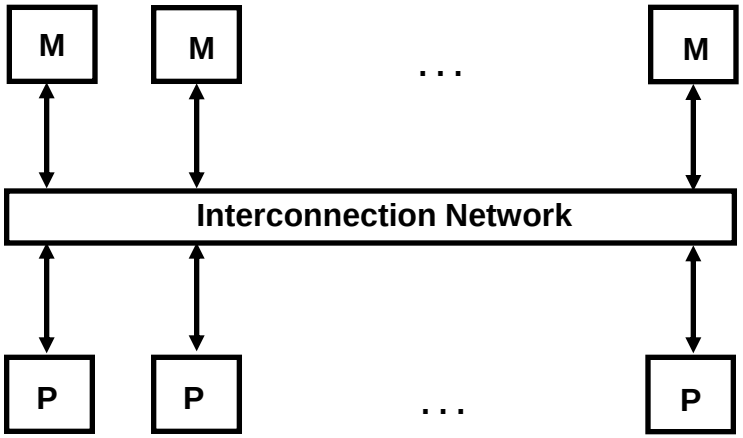
Nonuniform (NUMA) Memory

- Memory access is not uniform





SHARED MEMORY MULTIPROCESSORS



Buses,
Multistage IN,
Crossbar Switch

Characteristics

All processors have equally direct access to one large memory address space

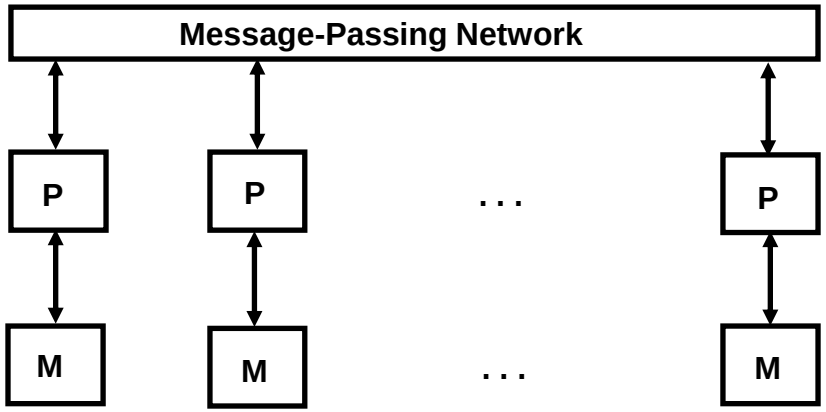
Example systems

- Bus and cache-based systems: Sequent Balance, Encore Multimax
- Multistage IN-based systems: Ultracomputer, Butterfly, RP3, HEP
- Crossbar switch-based systems: C.mmp, Alliant FX/8

Limitations

Memory access latency; Hot spot problem

MESSAGE-PASSING MULTIPROCESSORS



Characteristics

- Interconnected computers
- Each processor has its own memory, and communicate via message-passing

Example systems

- Tree structure: Teradata, DADO
- Mesh-connected: Rediflow, Series 2010, J-Machine
- Hypercube: Cosmic Cube, iPSC, NCUBE, FPS T Series, Mark III

Limitations

- Communication overhead; Hard to programming

INTERCONNECTION STRUCTURES

* Time-Shared Common Bus
* Multiport Memory
* Crossbar Switch
* Multistage Switching Network
* Hypercube System

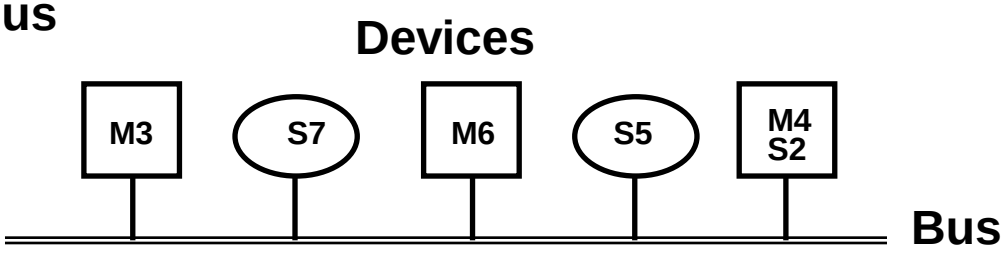
Bus

All processors (and memory) are connected to a common bus or busses
- Memory access is fairly uniform, but not very scalable

BUS

- A collection of signal lines that carry module-to-module communication
- Data highways connecting several digital system elements

Operations of Bus



M3 wishes to communicate with S5

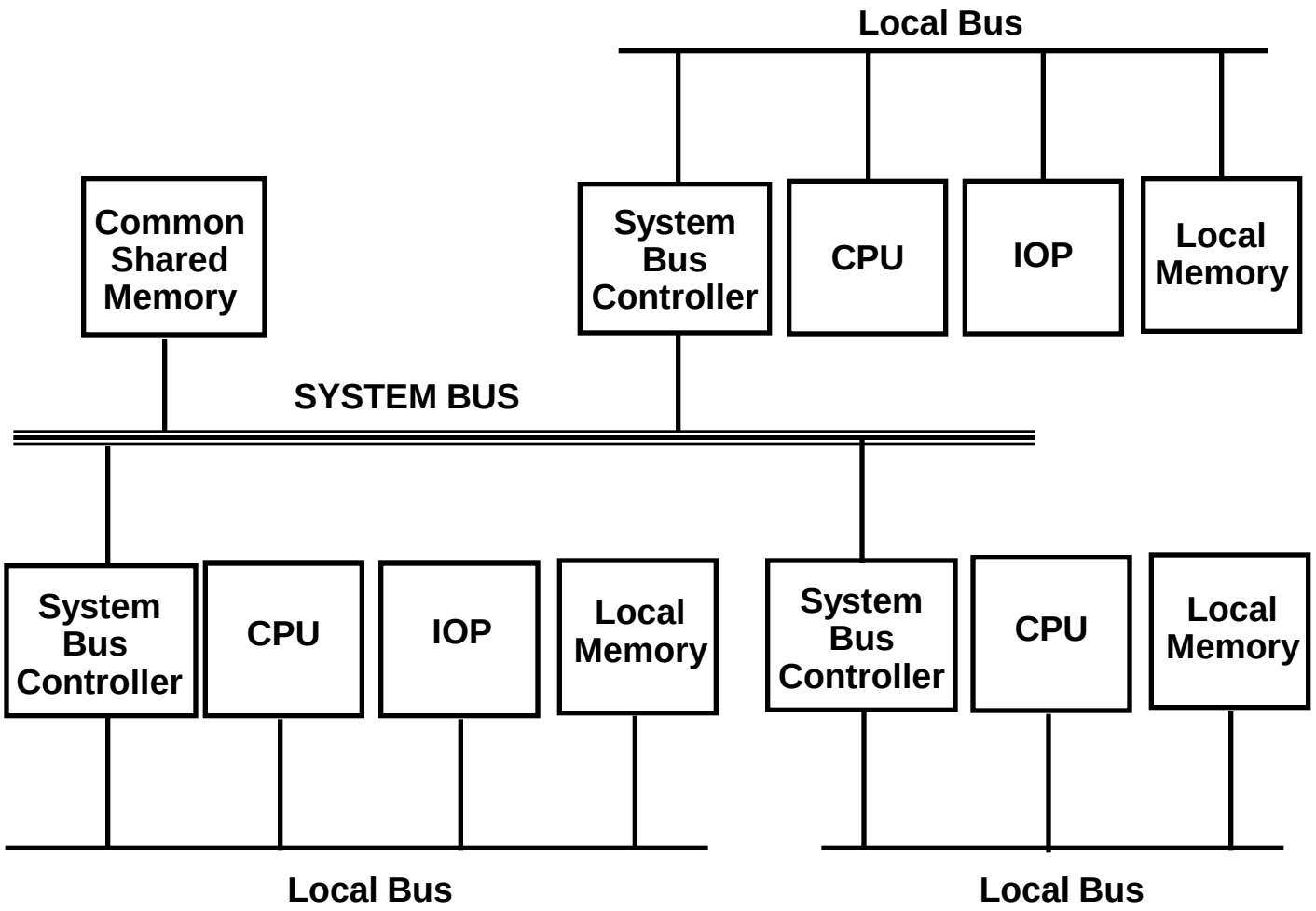
[1] M3 sends signals (address) on the bus that causes S5 to respond
[2] M3 sends data to S5 or S5 sends data to M3(determined by the command line)

Master Device: Device that initiates and controls the communication

Slave Device: Responding device

Multiple-master buses
-> Bus conflict
-> need bus arbitration

SYSTEM BUS STRUCTURE FOR MULTIPROCESSORS



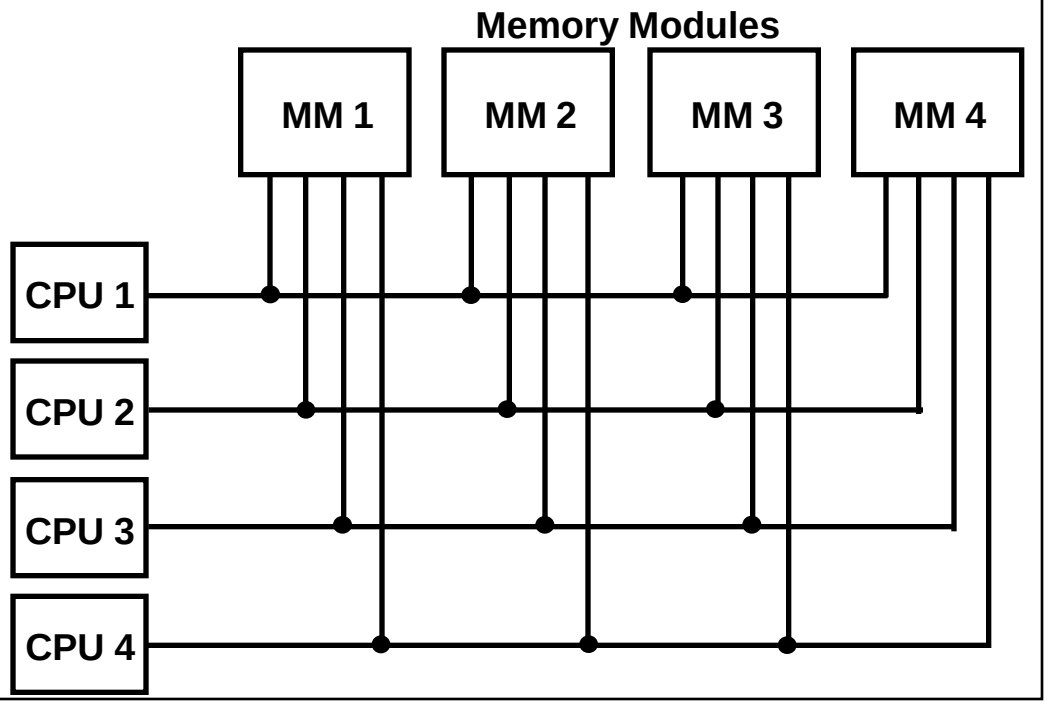
MULTIPORT MEMORY

Multiport Memory Module
- Each port serves a CPU

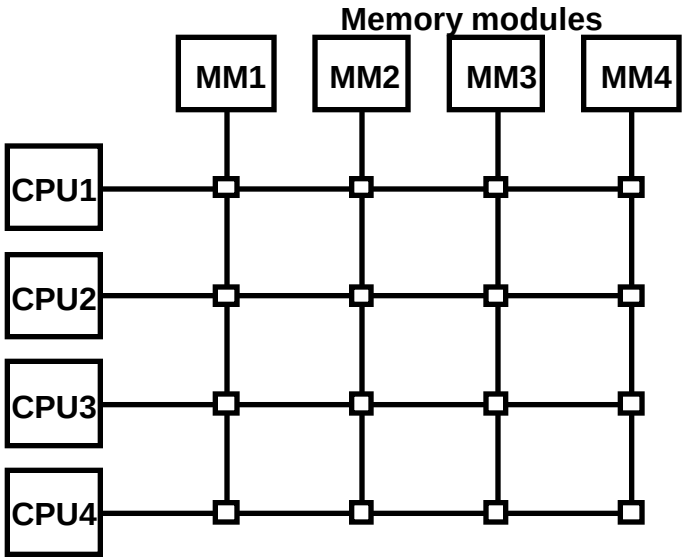
Memory Module Control Logic
- Each memory module has control logic
- Resolve memory module conflicts Fixed priority among CPUs

Advantages
- Multiple paths -> high transfer rate

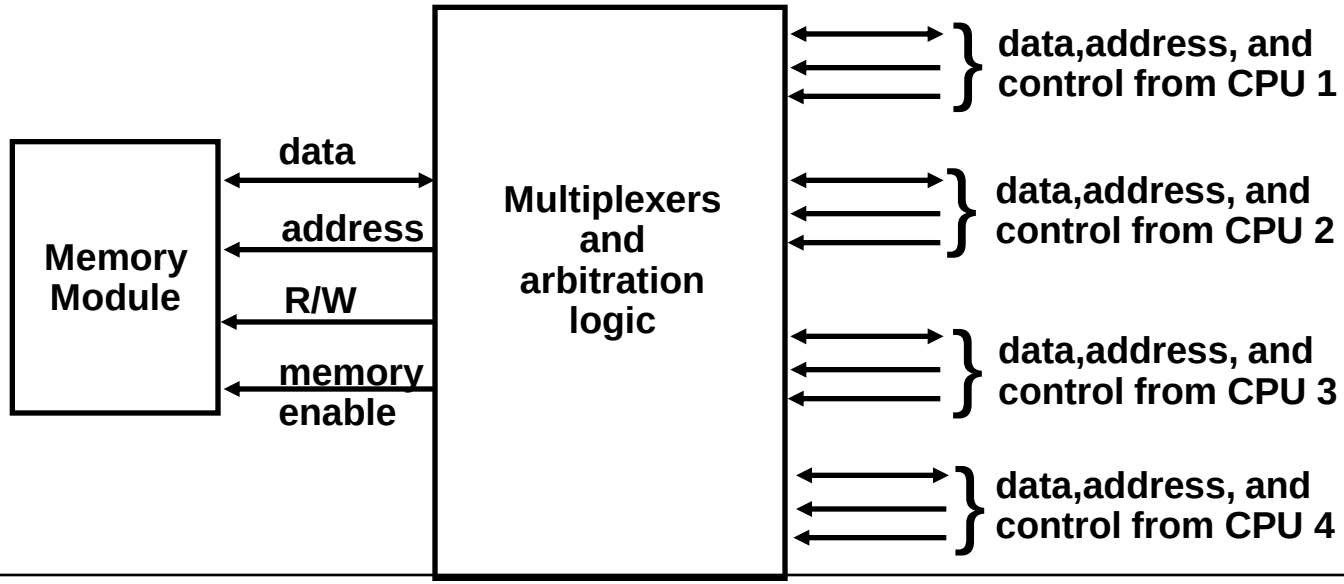
Disadvantages
- Memory control logic
- Large number of cables and connections



CROSSBAR SWITCH

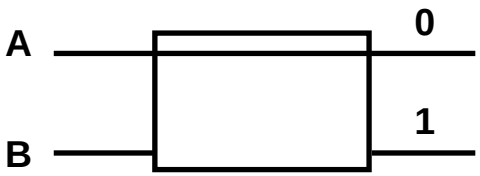


Block Diagram of Crossbar Switch

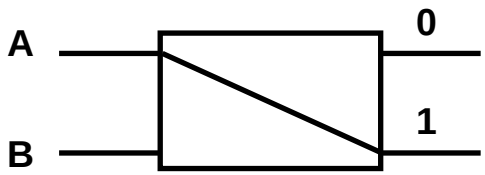


MULTISTAGE SWITCHING NETWORK

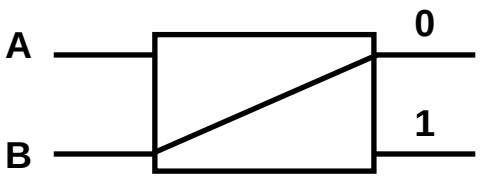
Interstage Switch



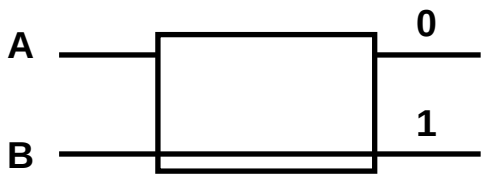
A connected to 0



A connected to 1



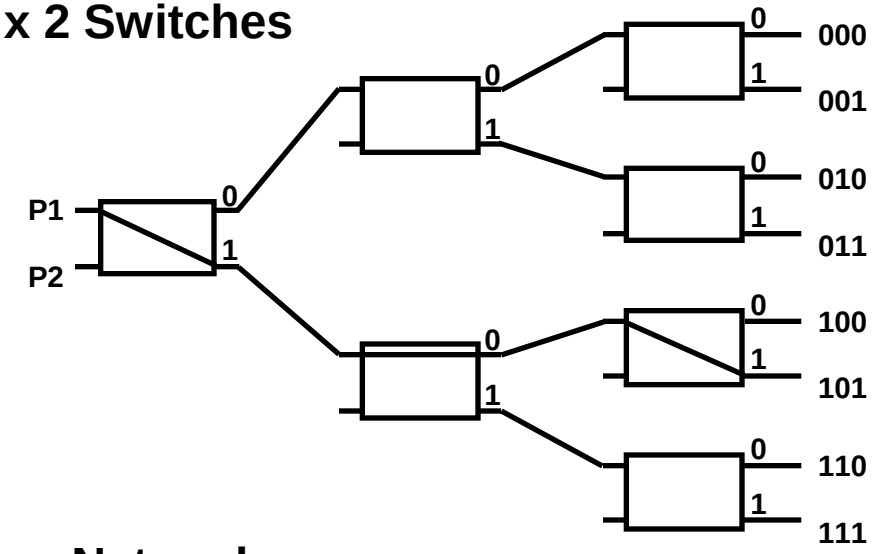
B connected to 0



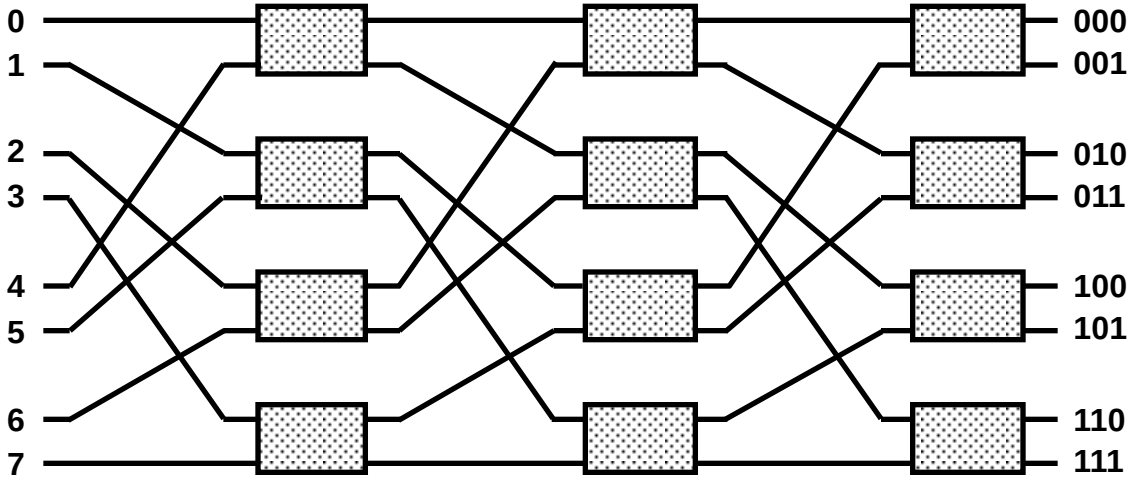
B connected to 1

MULTISTAGE INTERCONNECTION NETWORK

Binary Tree with 2 x 2 Switches



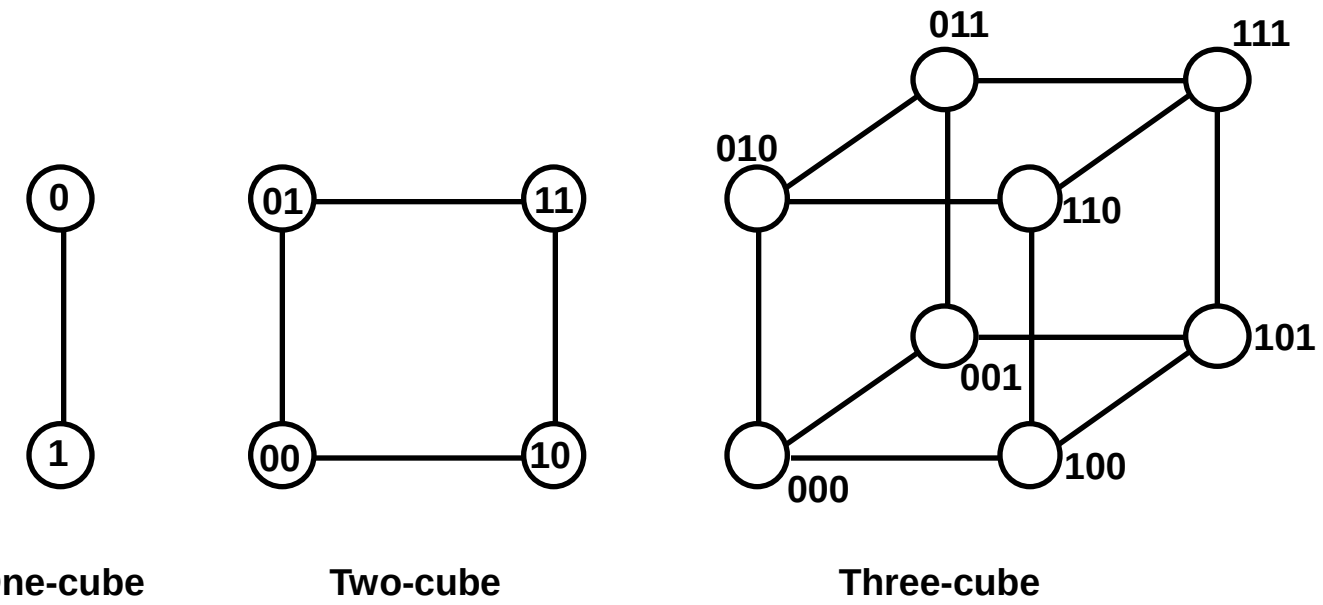
8x8 Omega Switching Network



HYPERCUBE INTERCONNECTION

n-dimensional hypercube (binary n-cube)

- $p = 2^n$
- processors are conceptually on the corners of a n-dimensional hypercube, and each is directly connected to the n neighboring nodes
- Degree = n



INTERPROCESSOR ARBITRATION

Bus

- **Board level bus**
- **Backplane level bus**
- **Interface level bus**

System Bus - A Backplane level bus

- **Printed Circuit Board**
- **Connects CPU, IOP, and Memory**
- **Each of CPU, IOP, and Memory board can be plugged into a slot in the backplane(system bus)**
- **Bus signals are grouped into 3 groups**

  Data, Address, and Control(plus power)

e.g. IEEE standard 796 bus
- 86 lines
Data: 16(multiple of 8)
Address: 24
Control: 26
Power: 20

- **Only one of CPU, IOP, and Memory can be granted to use the bus at a time**
- **Arbitration mechanism is needed to handle multiple requests**

SYNCHRONOUS & ASYNCHRONOUS DATA TRANSFER

Synchronous Bus

Each data item is transferred over a time slice known to both source and destination unit

- **Common clock source**
- **Or separate clock and synchronization signal is transmitted periodically to synchronize the clocks in the system**

Asynchronous Bus

*** Each data item is transferred by *Handshake* mechanism**

- **Unit that transmits the data transmits a control signal that indicates the presence of data**
- **Unit that receiving the data responds with another control signal to acknowledge the receipt of the data**

*** Strobe pulse - supplied by one of the units to indicate to the other unit when the data transfer has to occur**

BUS SIGNALS

Bus signal allocation

- address
- data
- control
- arbitration
- interrupt
- timing
- power, ground

IEEE Standard 796 Multibus Signals

Data and address	
Data lines (16 lines)	DATA0 - DATA15
Address lines (24 lines)	ADRS0 - ADRS23
Data transfer	
Memory read	MRDC
Memory write	MWTC
IO read	IORC
IO write	IOWC
Transfer acknowledge	TACK (XACK)
Interrupt control	
Interrupt request	INT0 - INT7
interrupt acknowledge	INTA

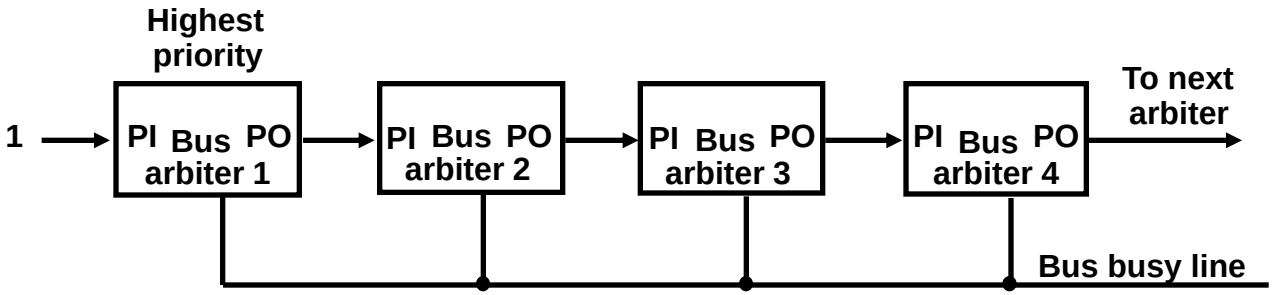
BUS SIGNALS

IEEE Standard 796 Multibus Signals (Cont'd)

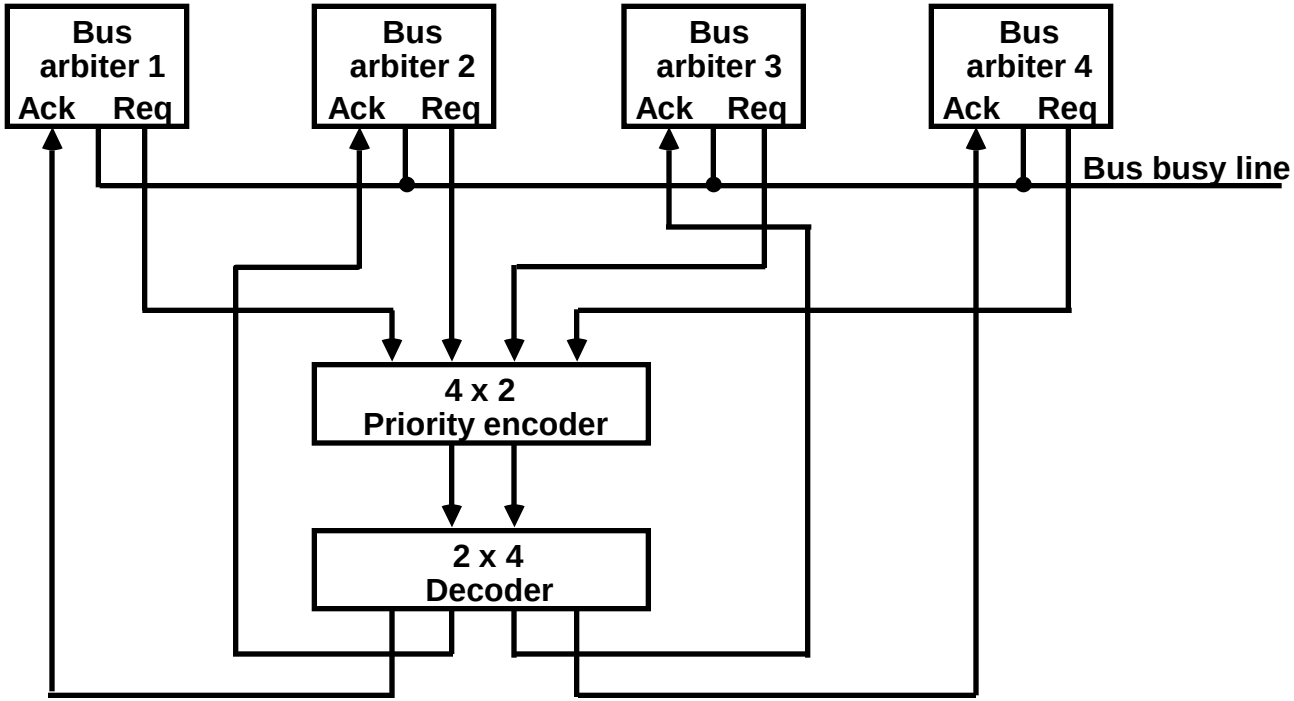
Miscellaneous control	
Master clock	CCLK
System initialization	INIT
Byte high enable	BHEN
Memory inhibit (2 lines)	INH1 - INH2
Bus lock	LOCK
Bus arbitration	
Bus request	BREQ
Common bus request	CBRQ
Bus busy	BUSY
Bus clock	BCLK
Bus priority in	BPRN
Bus priority out	BPRO
Power and ground (20 lines)	

INTERPROCESSOR ARBITRATION STATIC ARBITRATION

Serial Arbitration Procedure



Parallel Arbitration Procedure



INTERPROCESSOR ARBITRATION DYNAMIC ARBITRATION

Priorities of the units can be dynamically changeable while the system is in operation

Time Slice

- **Fixed length time slice is given sequentially to each processor, round-robin fashion**

Polling

- **Unit address polling - Bus controller advances the address to identify the requesting unit**

LRU

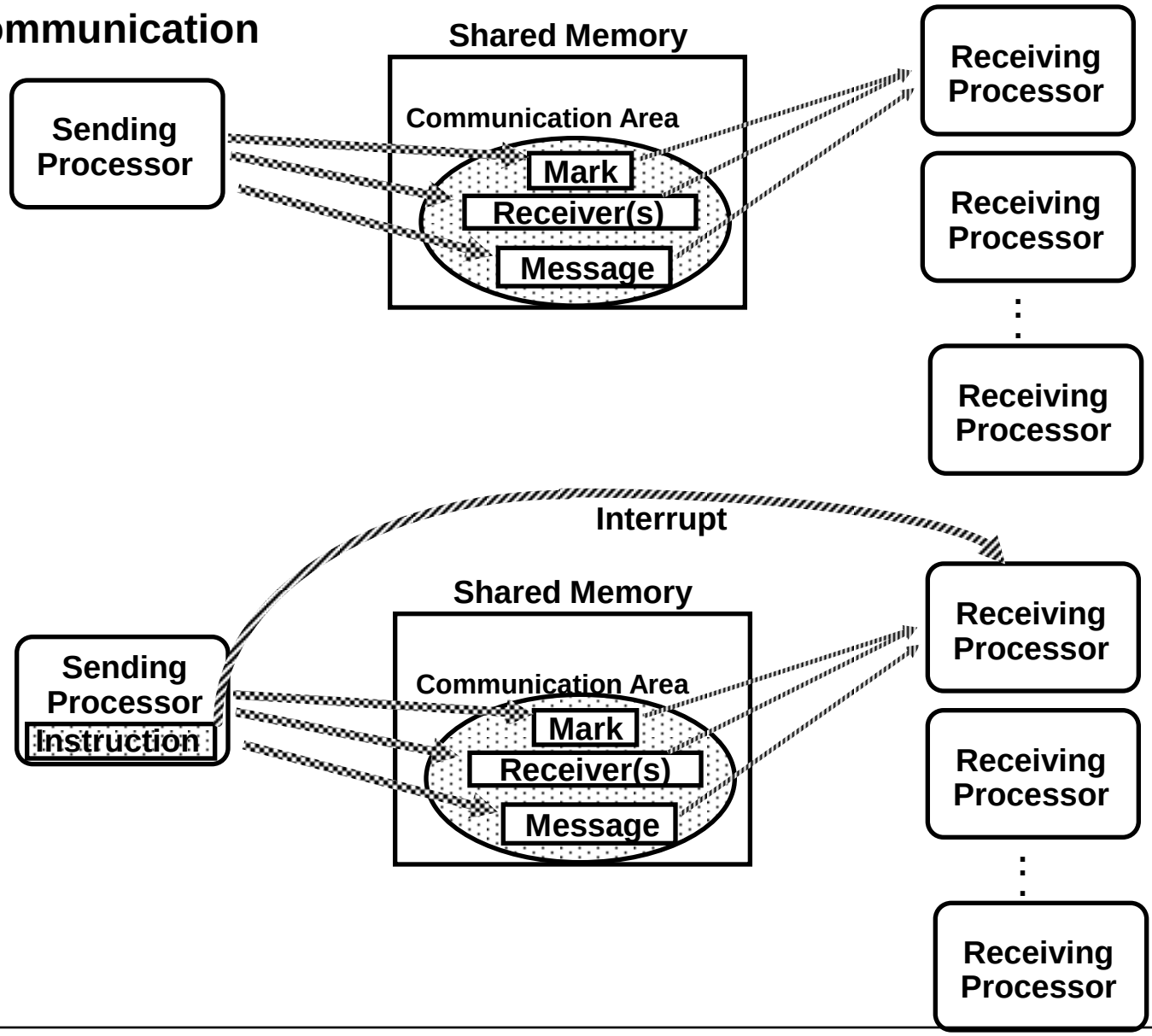
FIFO

Rotating Daisy Chain

- **Conventional Daisy Chain - Highest priority to the nearest unit to the bus controller**
- **Rotating Daisy Chain - Highest priority to the unit that is nearest to the unit that has most recently accessed the bus(it becomes the bus controller)**

INTERPROCESSOR COMMUNICATION

Interprocessor Communication



INTERPROCESSOR SYNCHRONIZATION

Synchronization

Communication of control information between processors

- To enforce the correct sequence of processes

- To ensure mutually exclusive access to shared writable data

Hardware Implementation

Mutual Exclusion with a *Semaphore*

Mutual Exclusion

- One processor to exclude or lock out access to shared resource by other processors when it is in a *Critical Section*

- Critical Section is a program sequence that, once begun, must complete execution before another processor accesses the same shared resource

Semaphore

- A binary variable

- 1: A processor is executing a critical section, that not available to other processors

0: Available to any requesting processor

- Software controlled Flag that is stored in memory that all processors can be access

SEMAPHORE

Testing and Setting the Semaphore

- **Avoid two or more processors test or set the same semaphore**
- **May cause two or more processors enter the same critical section at the same time**
- **Must be implemented with an indivisible operation**

```
R <- M[SEM]        / Test semaphore /
M[SEM] <- 1        / Set semaphore /
```

These are being done while *lock*ed, so that other processors cannot test and set while current processor is being executing these instructions

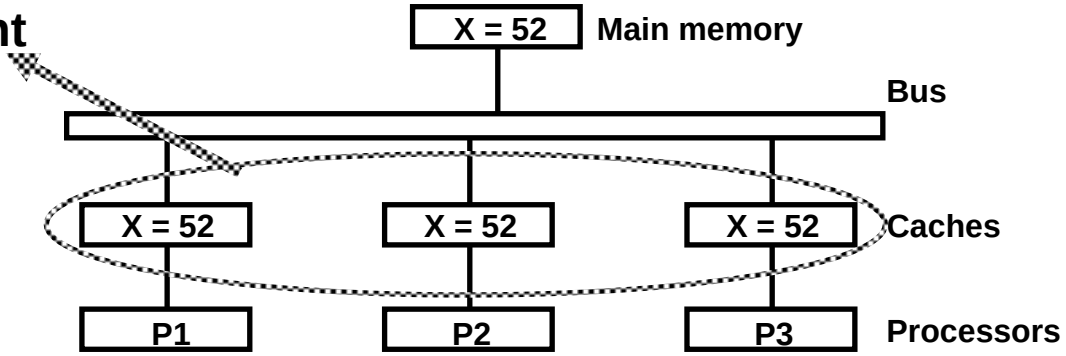
If R=1, another processor is executing the critical section, the processor executed this instruction does not access the shared memory

If R=0, available for access, set the semaphore to 1 and access

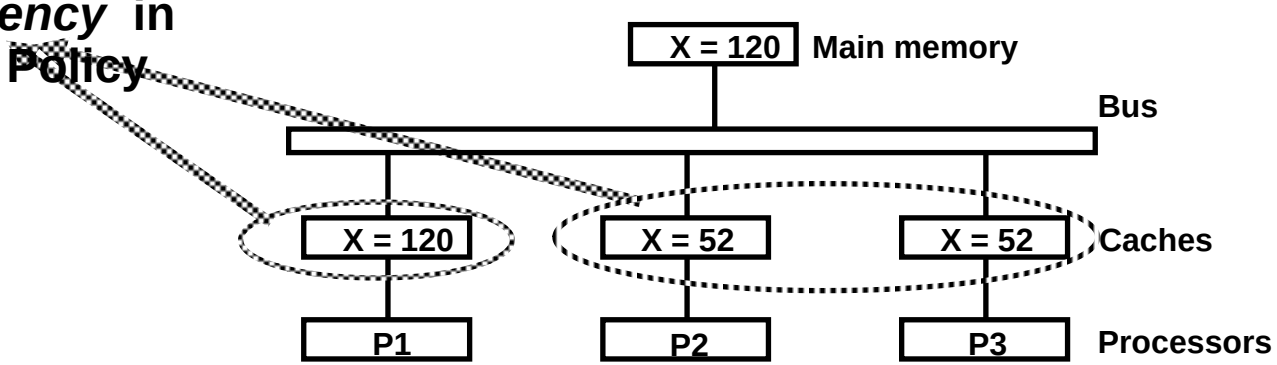
The last instruction in the program must clear the semaphore

CACHE COHERENCE

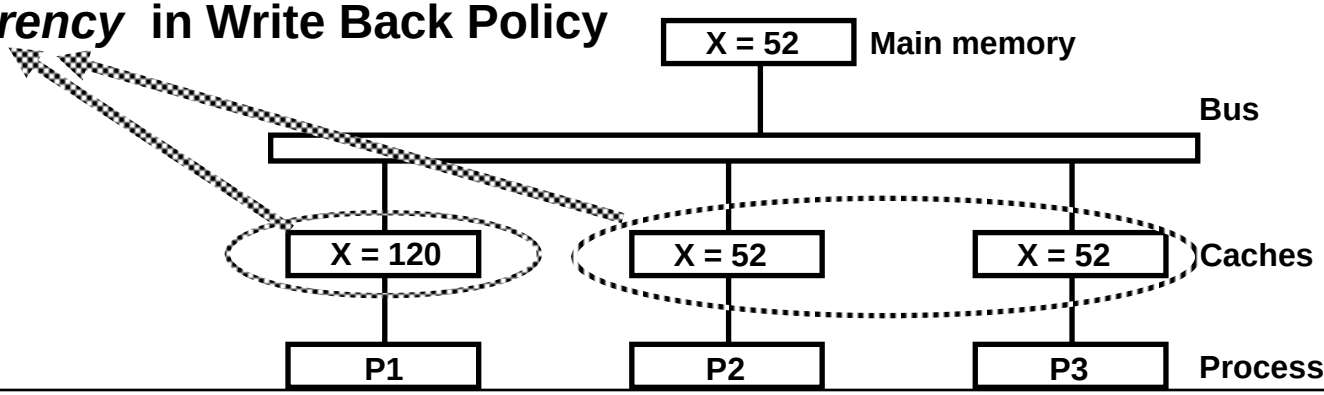
Caches are Coherent



Cache Incoherency in Write Through Policy



Cache Incoherency in Write Back Policy



MAINTAINING CACHE COHERENCY

Shared Cache

- Disallow private cache
- Access time delay

Software Approaches

* Read-Only Data are Cacheable

- Private Cache is for Read-Only data
- Shared Writable Data are not cacheable
- Compiler tags data as cacheable and noncacheable
- Degrade performance due to software overhead

* Centralized Global Table

- Status of each memory block is maintained in CGT: RO(Read-Only); RW(Read and Write)
- All caches can have copies of RO blocks
- Only one cache can have a copy of RW block

Hardware Approaches

* Snoopy Cache Controller

- Cache Controllers monitor all the bus requests from CPUs and IOPs
- All caches attached to the bus monitor the write operations
- When a word in a cache is written, memory is also updated (write through)
- Local snoopy controllers in all other caches check their memory to determine if they have a copy of that word; If they have, that location is marked invalid(future reference to this location causes cache miss)

PARALLEL COMPUTING

Grosche's Law

Grosch's Law states that the speed of computers is proportional to the square of their cost. Thus if you are looking for a fast computer, you are better off spending your money buying one large computer than two small computers and connecting them.

Grosch's Law is true within classes of computers, but not true between classes. Computers may be priced according to Groach's Law, but the Law cannot be true asymptotically.

Minsky's Conjecture

Minsky's conjecture states that the speedup achievable by a parallel computer increases as the logarithm of the number of processing elements,thus making large-scale parallelism unproductive.

Many experimental results have shown linear speedup for over 100 processors.

PARALLEL COMPUTING

History

History tells us that the speed of traditional single CPU Computers has increased 10 folds every 5 years.
Why should great effort be expended to devise a parallel computer that will perform tasks 10 times faster when, by the time the new architecture is developed and implemented, single CPU computers will be just as fast.

Utilizing parallelism is better than waiting.

Amdahl's Law

A small number of sequential operations can effectively limit the speedup of a parallel algorithm.

Let f be the fraction of operations in a computation that must be performed sequentially, where 0 < f < 1. Then the maximum speedup S achievable by a parallel computer with p processors performing the computation is $S < 1 / [f + (1 - f) / p]$. For example, if 10% of the computation must be performed sequentially, then the maximum speedup achievable is 10, no matter how many processors a parallel computer has.

There exist some parallel algorithms with almost no sequential operations. As the problem size(n) increases, f becomes smaller (f -> 0 as n->∞). In this case, $\lim_{n \to \infty} S = p$.

PARALLEL COMPUTING

Pipelined Computers are Sufficient

Most supercomputers are vector computers, and most of the successes attributed to supercomputers have accomplished on pipelined vector processors, especially Cray=1 and Cyber-205.

If only vector operations can be executed at high speed, supercomputers will not be able to tackle a large number of important problems. The latest supercomputers incorporate both pipelining and high level parallelism (e.g., Cray-2)

Software Inertia

Billions of dollars worth of FORTRAN software exists. Who will rewrite them? Virtually no programmers have any experience with a machine other than a single CPU computer. Who will retrain them ?

INTERCONNECTION NETWORKS

Switching Network (Dynamic Network)

Processors (and Memory) are connected to routing switches like in telephone system

- Switches might have queues(combining logic), which improve functionality but increase latency
- Switch settings may be determined by message headers or preset by controller
- Connections can be packet-switched or circuit-switched(remain connected as long as it is needed)
- Usually NUMA, blocking, often scalable and upgradable

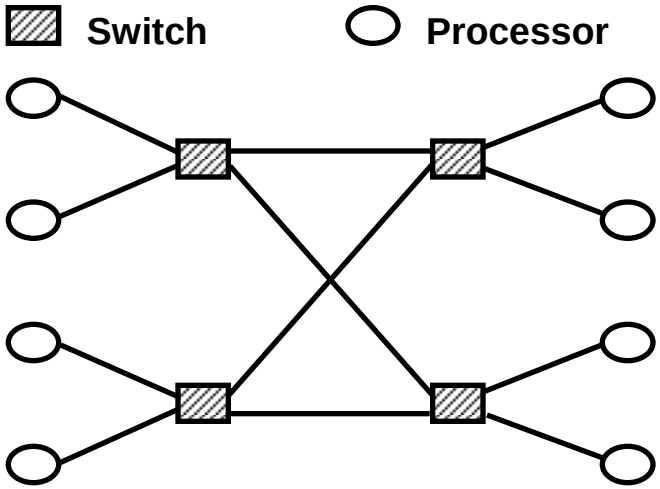
Point-Point (Static Network)

Processors are directly connected to only certain other processors and must go multiple hops to get to additional processors

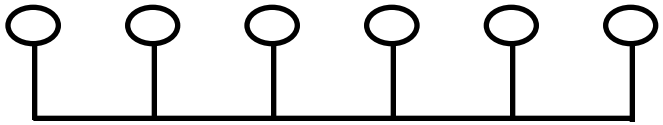
- Usually distributed memory
- Hardware may handle only single hops, or multiple hops
- Software may mask hardware limitations
- Latency is related to graph diameter, among many other factors
- Usually NUMA, nonblocking, scalable, upgradable
- Ring, Mesh, Torus, Hypercube, Binary Tree

INTERCONNECTION NETWORKS

Multistage Interconnect



Bus



INTERCONNECTION NETWORKS

Static Topology - Direct Connection

- Provide a direct inter-processor communication path
- Usually for distributed-memory multiprocessor

Dynamic Topology - Indirect Connection

- Provide a physically separate switching network for inter-processor communication
- Usually for shared-memory multiprocessor

Direct Connection

Interconnection Network

A graph G(V,E)

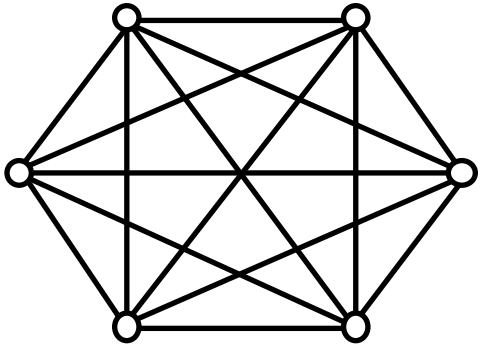
V: a set of processors (nodes)

E: a set of wires (edges)

Performance Measures: - degree, diameter, etc

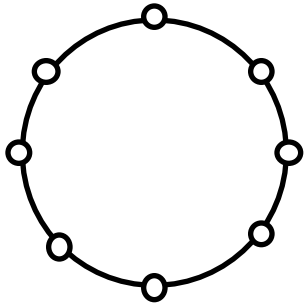
INTERCONNECTION NETWORKS

Complete connection



- Every processor is directly connected to every other processors
- Diameter = 1, Degree = p - 1
- # of wires = p (p - 1) / 2; dominant cost
- Fan-in/fanout limitation makes it impractical for large p
- Interesting as a theoretical model because algorithm bounds for this model are automatically lower bounds for all direct connection machines

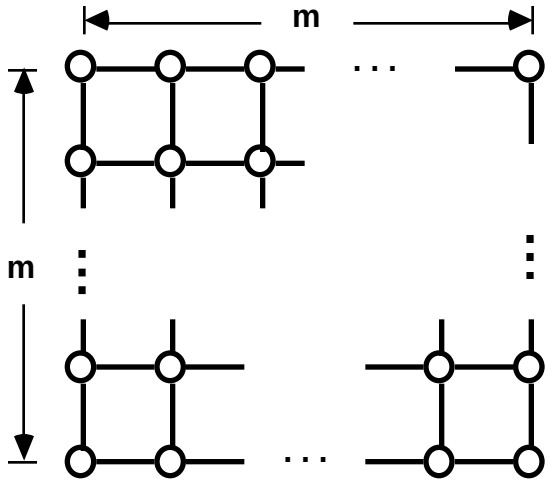
Ring



- Degree = 2, (not a function of p)
- Diameter = $\lfloor p/2 \rfloor$

INTERCONNECTION NETWORKS

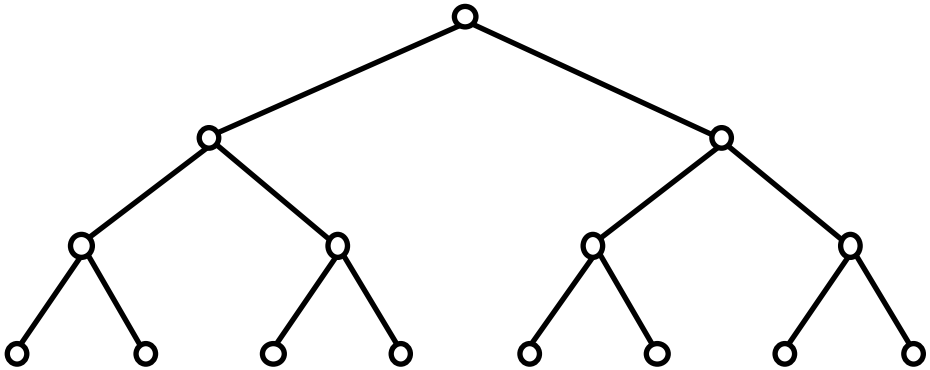
- **2-Mesh**



- Degree = 4
- Diameter = 2(m - 1)
- In general, an n-dimensional mesh has
  diameter = d ($p^{1/n}$ - 1)
- Diameter can be halved by having wrap-around connections (-> Torus)
- Ring is a 1-dimensional mesh with wrap-around connection

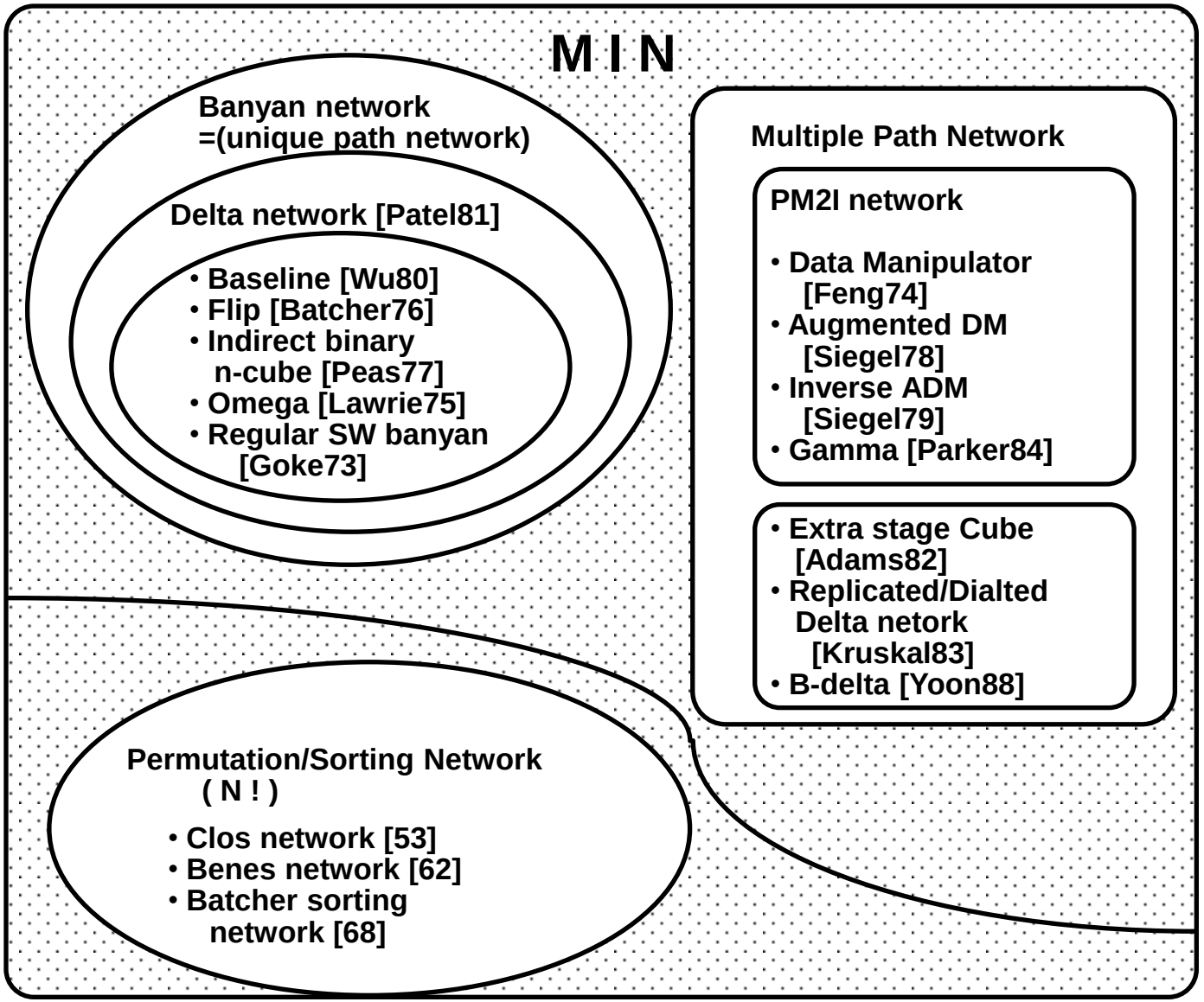
INTERCONNECTION NETWORK

Binary Tree



- Degree = 3
- Diameter = $2 \log \frac{p + 1}{2}$

MIN SPACE


MIN
Banyan network
=(unique path network)
Delta network [Patel81]
• Baseline [Wu80]
• Flip [Batcher76]
• Indirect binary
n-cube [Peas77]
• Omega [Lawrie75]
• Regular SW banyan
[Goke73]
Multiple Path Network
PM2I network
• Data Manipulator
[Feng74]
• Augmented DM
[Siegel78]
• Inverse ADM
[Siegel79]
• Gamma [Parker84]
• Extra stage Cube
[Adams82]
• Replicated/Dialted
Delta netork
[Kruskal83]
• B-delta [Yoon88]
Permutation/Sorting Network
(N !)
• Clos network [53]
• Benes network [62]
• Batcher sorting
network [68]

SOME CURRENT PARALLEL COMPUTERS

DM-SIMD

- AMT DAP
- Goodyear MPP
- Thinking Machines CM series
- MasPar MP1
- IBM GF11

SM-MIMD

- Alliant FX
- BBN Butterfly
- Encore Multimax
- Sequent Balance/Symmetry
- CRAY 2, X-MP, Y-MP
- IBM RP3
- U. Illinois CEDAR

DM-MIMD

- Intel iPSC series, Delta machine
- NCUBE series
- Meiko Computing Surface
- Carnegie-Mellon/ Intel iWarp