TEMPUS: Activity 2

Parallel Algorithms,
Chapter 1 Fundamentals,
Section 1, Parallel Models

István Erényi, KFKI MSZKI

20th July 1997

Overview

1 Sequential machine model: RAM
2 Requirements toward parallel models
3 Parallel Random Access Machine, PRAM
4 Models for Parallel Algorithm Behavior
- Directed Acyclic Graph model
- Shared Memory model
- Network model

5 Comparison of Parallel Models
6 Questions closer to the implementation
7 Taxonomy of parallel architectures
8 Shared address space architectures
9 Interconnection networks
10 Processor granularity
11 Summary

Fundamentals: Parallel Models
1 Sequential machine model: RAM

Running time estimation of a sequential algorithm is based upon the number of basic operations as a function of the input size (data amount)

What is a basic operation?
Memory reads/writes and
basic arithmetic/logic operations
(*uniform cost criterion:* word size independence)

Random Access Machine model (RAM):
a sequential machine model
containing: *Central Processing Unit,*
Random Access Memory, and
some input/output handling operations

Fundamentals: Parallel Models
2 Requirements

Parallel models: algorithmic models that may be used as general frameworks for describing and analyzing parallel algorithms

Requirements:

Simplicity in order to describe its operation easily and to analyze the basic performance measures (speed, communication, memory utilization)

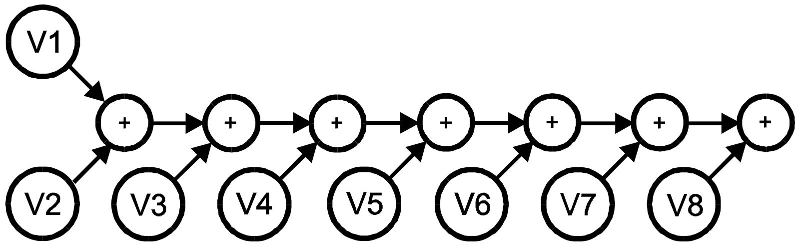
Hardware independence not to be tied to any particular architecture

Implementability the algorithmic model should be easily implementable on parallel computers

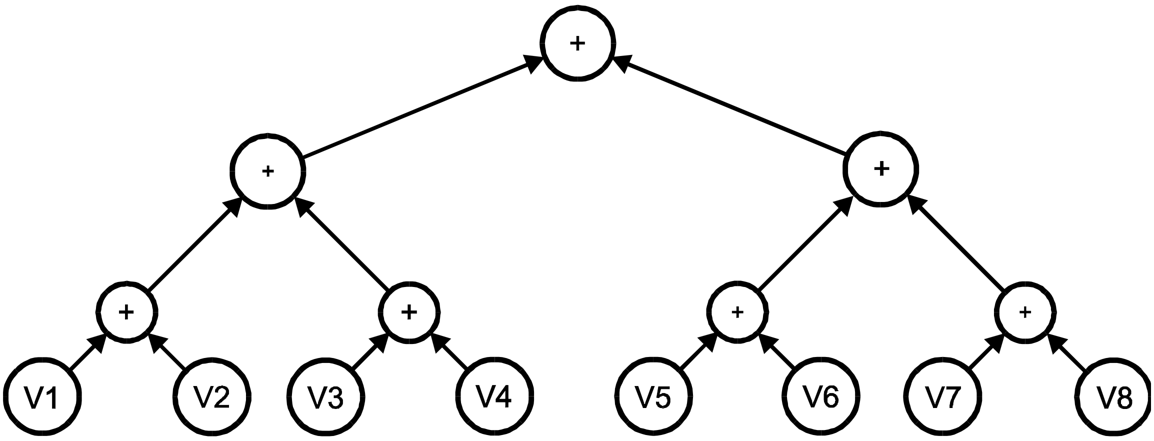
So far there is no model that has been proven to be acceptable by most researchers.

Examples for algorithm data-flow

Two possible algorithms for adding eight values:

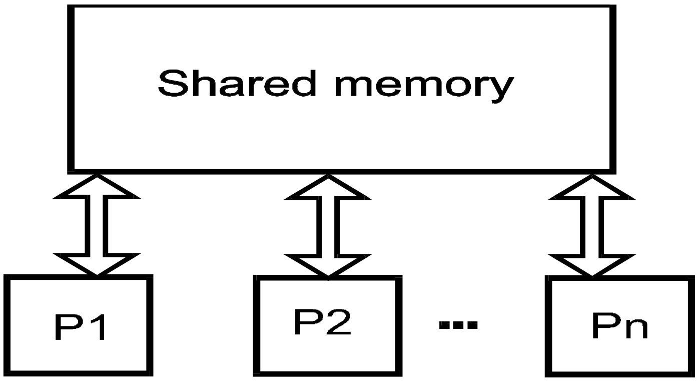


a.) Serial

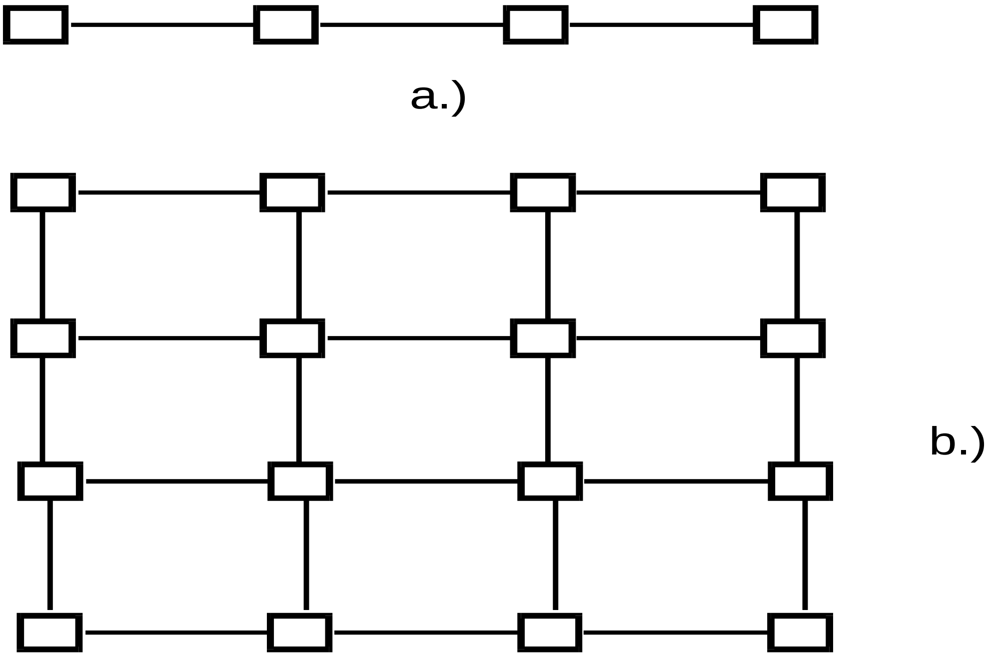


b.) Parallel

Various parallel computer architectures



The Shared-memory model



Networks: a) linear array, b) 4 x 4 mesh

Parallel Models

3 Parallel Random Access Machine PRAM

PRAM is an idealized parallel computer**:**

p processors,

global memory of unbounded size,

uniformly accessible to all processors

(processors share same address space)

common clock control

processors execute different instructions

in each cycle

EREW: Exclusive-read, exclusive-write PRAM

CREW: Concurrent-read, exclusive-write PRAM

ERCW: Exclusive-read, concurrent-write PRAM

CRCW: Concurrent-read, concurrent-write PRAM

for CRCW PRAM concurrent write needs resolving protocols:

common: identical values may be concurrently written

arbitrary: one arbitrary processor is allowed to write

priority: the highest priority processor may write

sum: sum (or other functional result) of all quantities is written

Parallel Models
4 Models for Algorithm Behavior

Directed Acyclic Graphs (DAGs) for computation representation**:**

inputs are nodes with no incoming arcs

operations are nodes with incoming arcs

operands are represented by incoming arcs for operation nodes

indegree of the internal nodes is at most 2

outputs are nodes with zero outdegree

(unit cost criterion is assumed, i.e. each node operation takes unit time)

DAG with n input nodes represents a computation with no branches, and has an input size of n. An algorithm is represented with a family of DAGs $\{G_n\}$, where G_n refers to an algorithm with input size n.

DAG implies **precedence constraints** on the order of operations. It is architecture-independent.

Parallel Models
4/a Directed Acyclic Graphs (DAGs)
(continued)

Simplification:

any processor can access the data computed by any other processor, without incurring additional cost

Scheduling appears in a particular implementation with *p* processors in a the way of assigning one or more node functions to any of the processors

Performance analysis via DAGs:

assuming uniform time complexity for each of the node functions, (t_i), the *parallel complexity* of the DAG is defined by the minimum execution time $min\ (T_p)$ of all possible schedules that use *p* processors. Clearly the depth of the DAG (the longest path from between an input and output node) is a lower bound on T_p for any number *p* of processors

Parallel Models
4/b The Shared Memory Model

This Model consists of *p* processors, each of them may have its own local memory and may execute its own local program, and all of them communicate by exchanging data through a shared (common, global) memory unit.

Modes of operation:

Synchronous: all the processors operate synchronously under common clock control. This is the **Parallel Random Access machine (PRAM)**

Asynchronous: each processor has its own separate clock. The program's responsibility: setting synchronization points at data or control exchanges: **Multiple Instruction Multiple Data (MIMD)** type.

Parallel Models
4/c The Network Model

The Network can be modeled by a graph $G=(N,E)$, where each node $i \in N$ represents a processor, and each edge $(i,j) \in E$ a communication link between processors *i* and *j*. Each processor has its own local memory, and no shared memory exists.

Asynchronous network operation: the processors' activity is coordinated by exchanging messages: *Message-passing model.* Communicating processors do not have to be adjacent. In this case the process that delivers messages is called *routing*.

Synchronous networks: they are controlled by one global clock, *systolic paradigm.*

Parallel Models
4/c The Network Model
(continued)

Network topology: the interconnection topology between the processors defines the structure of the network.

Topology evaluation parameters:

Diameter of the network: the maximum distance between any pair of nodes

Maximum degree of any node in *G*

Connectivity (node connectivity) of *G* is the measure of multiplicity of paths between any of two processors

Arc or edge connectivity of *G* is the minimum number of arcs that must be removed to break the network into two disconnected networks

Bisection bandwidth is the minimum number of communication links that have to be removed to cut the network into two halves

Parallel Models
5 Comparison of the Parallel Models

DAG: simple, but it could be very difficult to analyze the algorithm

scheduling and processor allocation problem is not represented by DAGs

there is no mechanism to handle interprocessor and memory communication

Network model:

it is very difficult to analyze algorithms for this model

the model depends heavily on the particular topology, different topologies require totally different algorithms

Parallel Models
5 Comparison of the Models
(continued)

Shared memory model:
PRAM:
well-developed techniques exist

simplifies the algorithm analysis by removing communication and synchronization problems

offers explicit understanding of the operations in each time unit, and processor-job allocation

PRAM paradigm is robust, many network algorithms may be derived from it

synchronization and communication issues may be easily incorporated

more general problems can be analyzed by this model

Parallel Models
6 Closer to implementation issues

Evolution of Computers:

John von Neumann model: CPU and memory

Single-Instruction Single Data, (SISD)

Speed defined by instruction execution and memory-CPU exchange rate

Speed increase:

for execution with **pipelining**

for memory exchange with **memory interleaving** and **cache memories**

Parallel computers with multiple processing units

They differ along various dimensions:

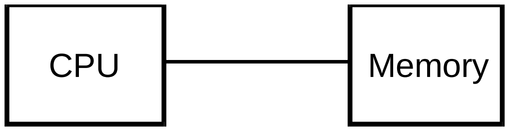
control mechanism

address-space organization

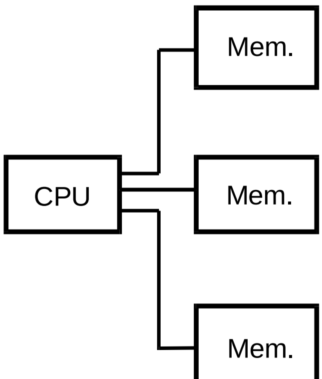
interconnection network

processor granularity

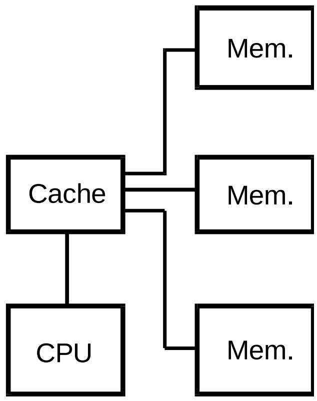
Evolution of single processor computers



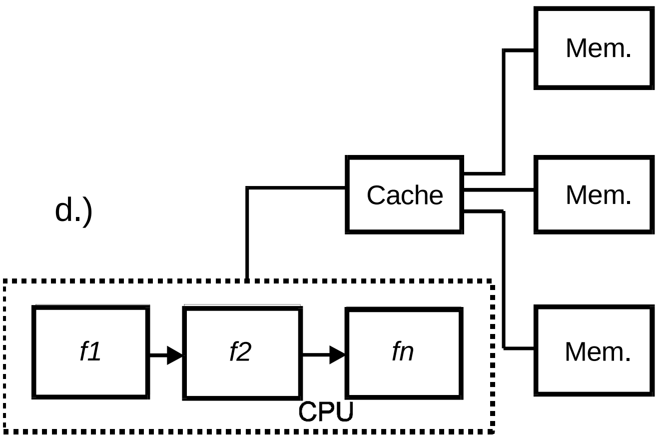
a.) SISD model



b.) memory interleaving



c.) with cache



d.) pipeline CPU with n stages

Parallel Models

6 Taxonomy of Parallel Architectures:

Control Mechanism

Centralized control, single control unit

Single-Instruction Multiple-Data, (SIMD)

The control unit dispatches instructions to each processing unit. Processing units can be selectively switched out.

Special purpose control and processing units are used

Primarily for data parallel programs

Independent program control

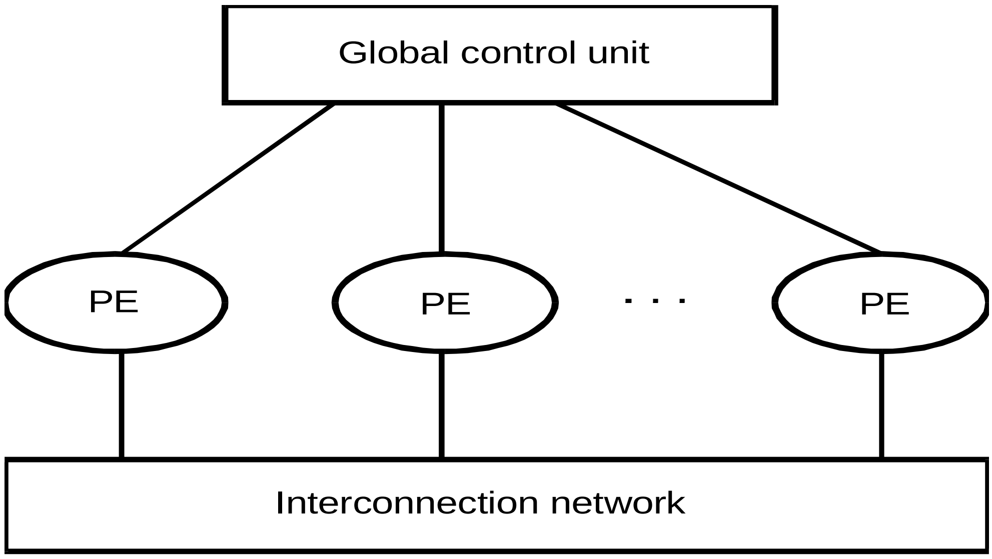
Multiple-Instruction Multiple-Data, (MIMD)

Each processor executes its own program

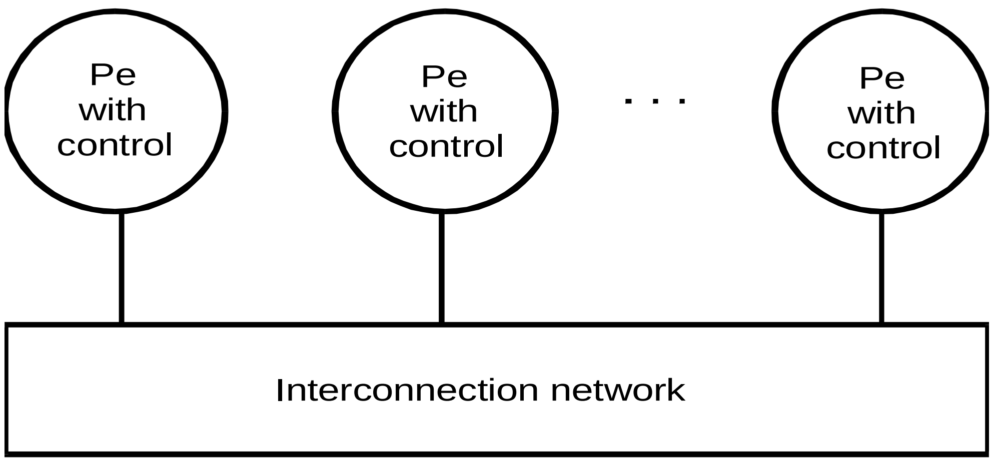
Extra measures (software or hardware) are needed for synchronization at data and control exchange actions

General purpose processors may be used

Synchronous and Asynchronous control



SIMD architecture



MIMD architecture

Parallel Models
6 Taxonomy of Parallel Architectures:
(continued)

Address-Space Organization

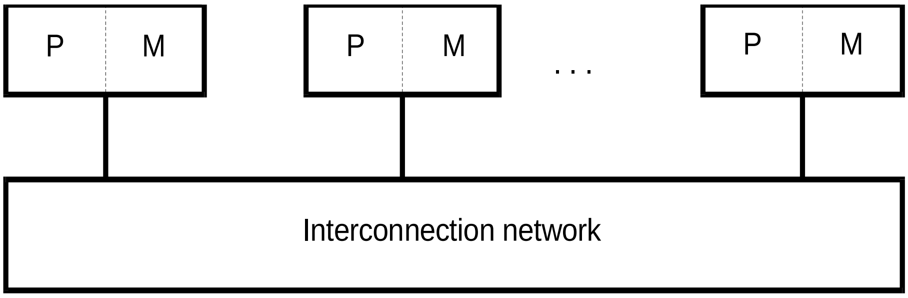
Shared-Address-Space Architecture

Hardware support exists for read and write access by all processors to a shared address space (of global memory). Processors interact by modifying data in the shared address space. In order to avoid memory access bottleneck, processors may have their own private or local memory units, as well.

Message-Passing Architecture

Processors are connected using message passing interconnection network. Each processor has its own *local or private memory*. An alternative name is *distributed memory architecture.*

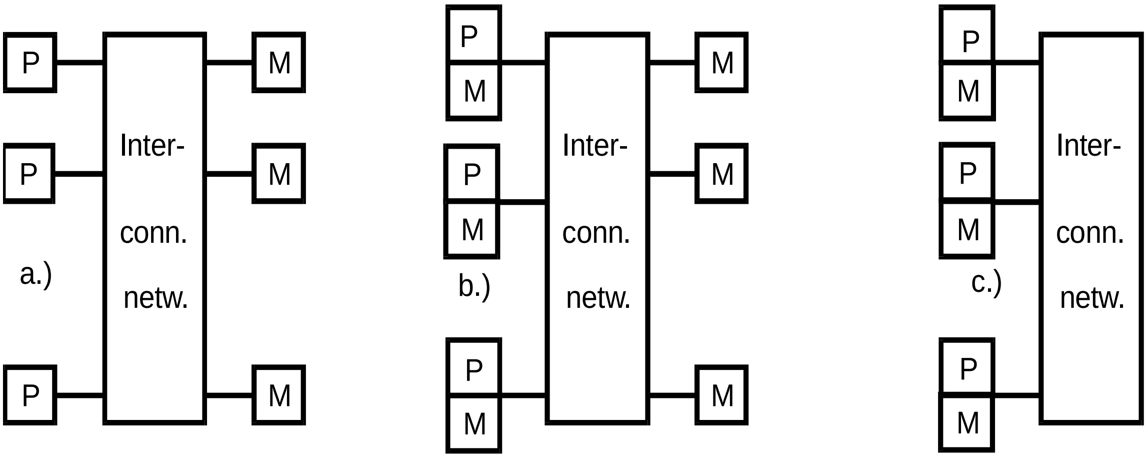
Shared memory and Message passing architectures



P: processor

M; memory

Message passing architecture



Shared-Memory Access architectures

a.) UMA; b.) NUMA with global and local memories; c.) NUMA with local memories only

Parallel Models
8 Shared-Address-Space Architecture

Architectures with local and global memories

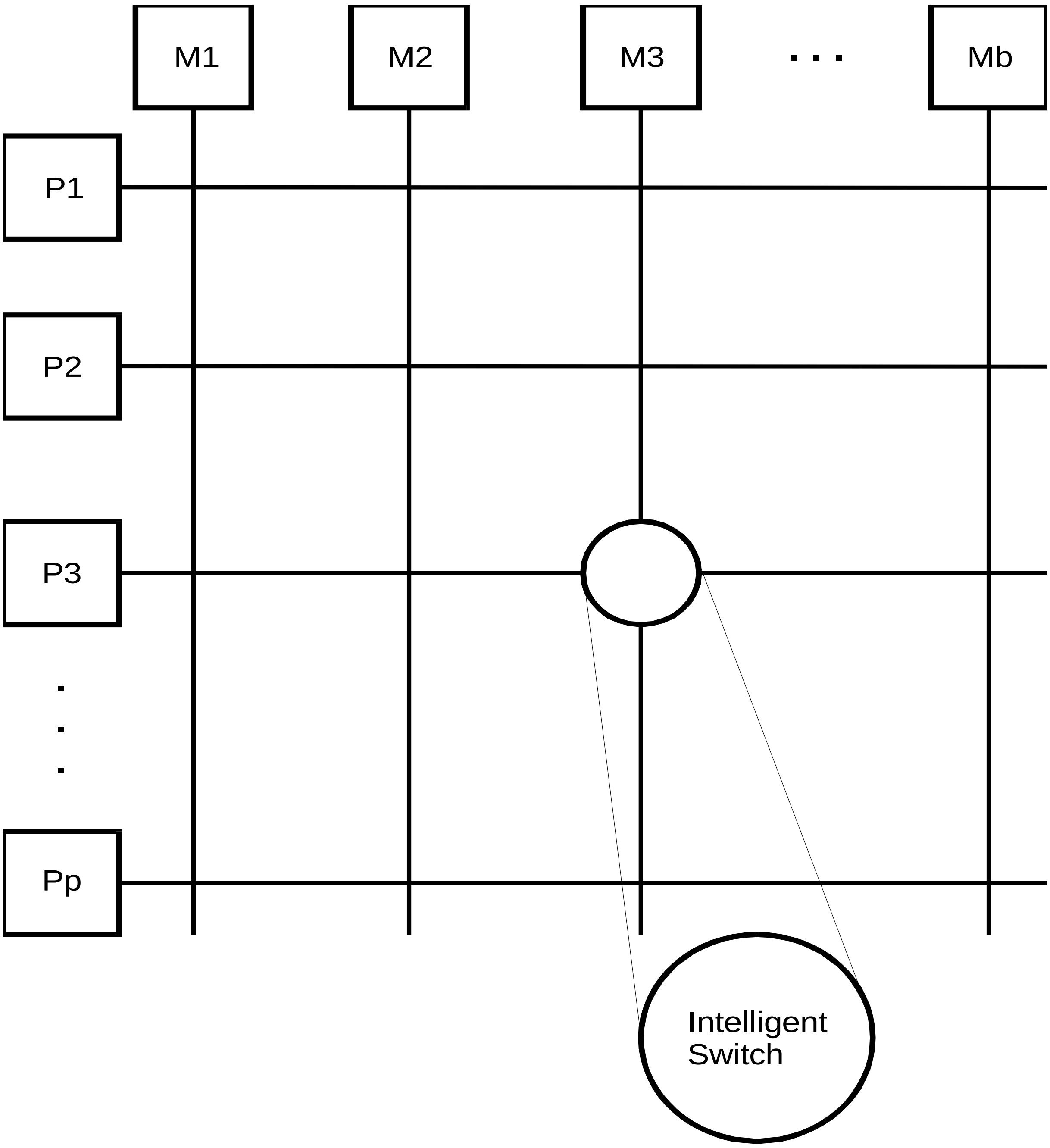
Uniform Memory Access computers (UMA)
Any memory word can be accessed by any processor within identical time

Non-Uniform Memory Access computers (NUMA), Usually a remote memory location access takes longer time than a local one.

Cache memories: usually local caches are used at each processor. Additional cache coherence problem exists in these systems.

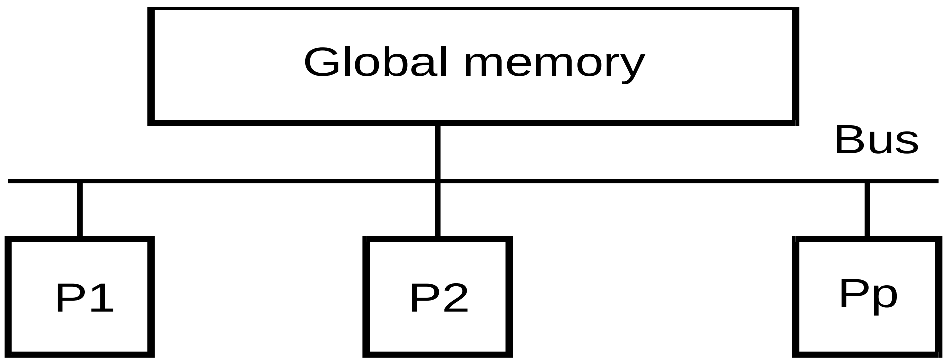
Cache-Only Memory Access computers (COMA) with global only memory, but local caches at the processors.

Cross-bar architecture

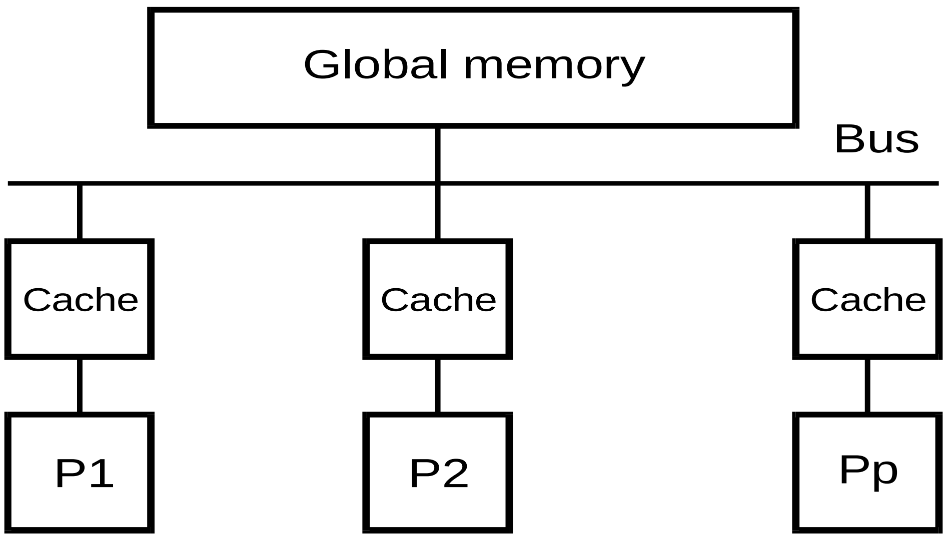


A crossbar parallel computer with
p processors and *b memory blocks*

Bus-based architectures



Bus-based architecture



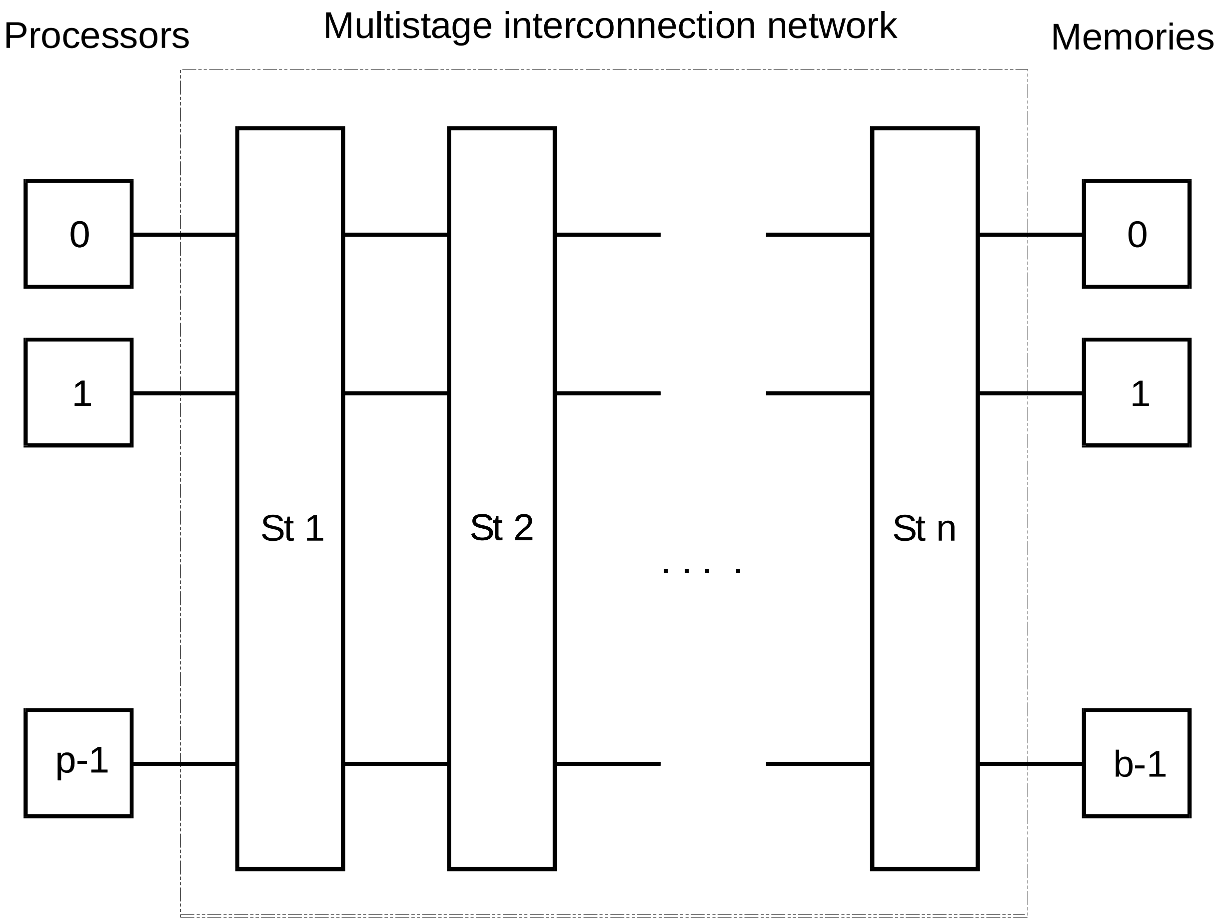
Bus-based architecture with cache memories

Parallel Models
9 Interconnection networks

Static or direct interconnections: they consist of point-to-point communications links among processors
Some typical types of them:
Star, *Ring*, *Mesh (2-D, n-D)*, *Torus*, *Linear array*, *Tree*, *Hypercube*, *k-ary d-cube*

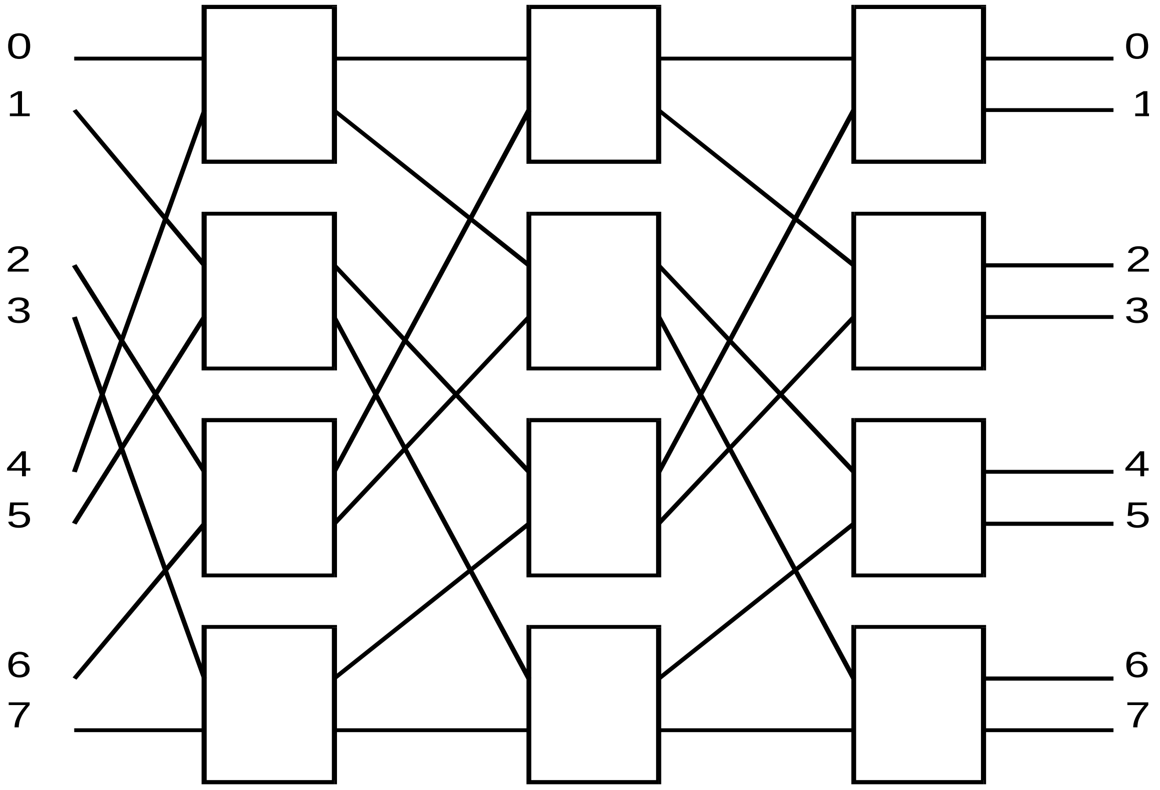
Dynamic or indirect networks: they use switches and communication links. Switches are programmed to set up the required paths among the processors.
Some typical types of the networks:
Crossbar switching, *Bus-based*, *Multistage interconnection networks*.

Multistage switching network



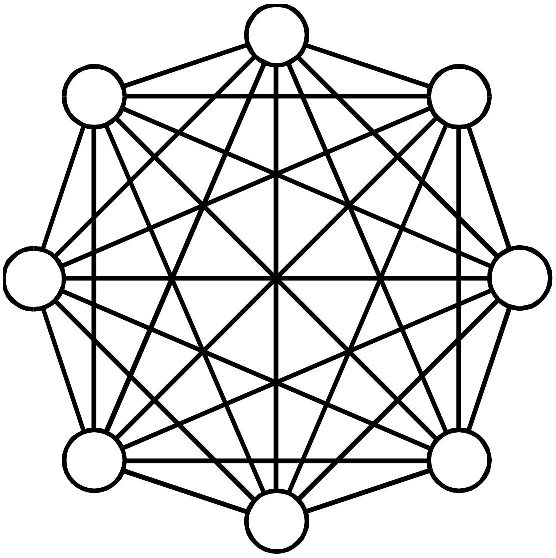
Omega switchning network

Two switch configurations

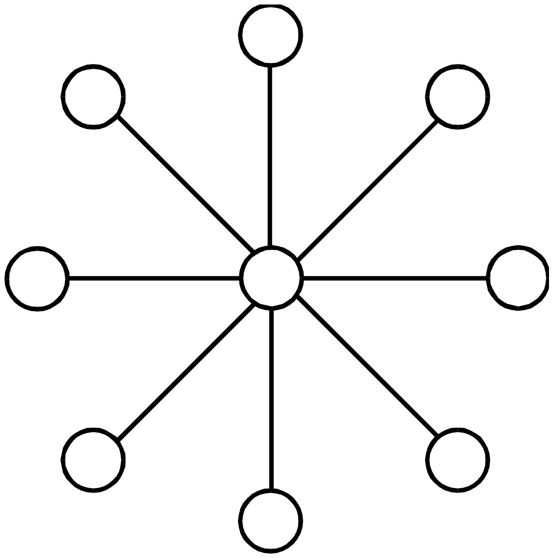


A complete omega network connecting
eight inputs and eight outputs

Completely connected and Star networks

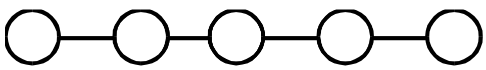


Completely connected network of eight processors

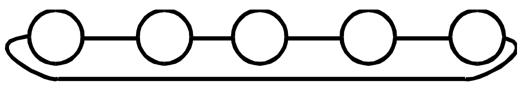


Star connected network of nine processors

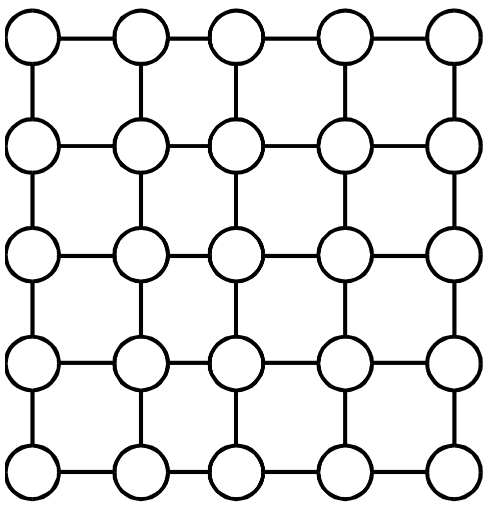
Linear array, Ring 2-D mesh, and 2-D torus topologies



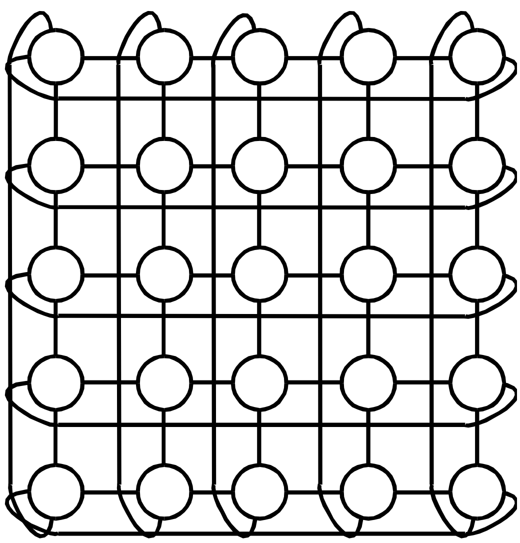
A five processor liear array



A five-processor ring

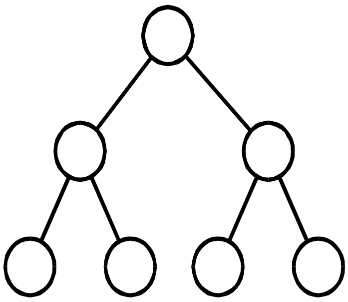


A two dimensional (5 by 5) mesh

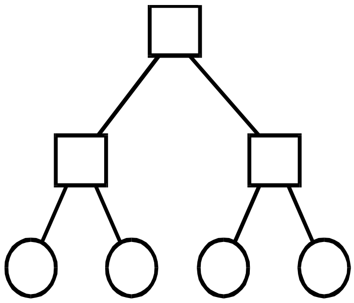


A two dimensional (5 by 5) torus

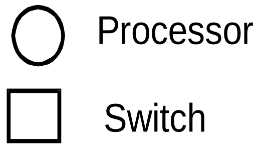
Tree topologies

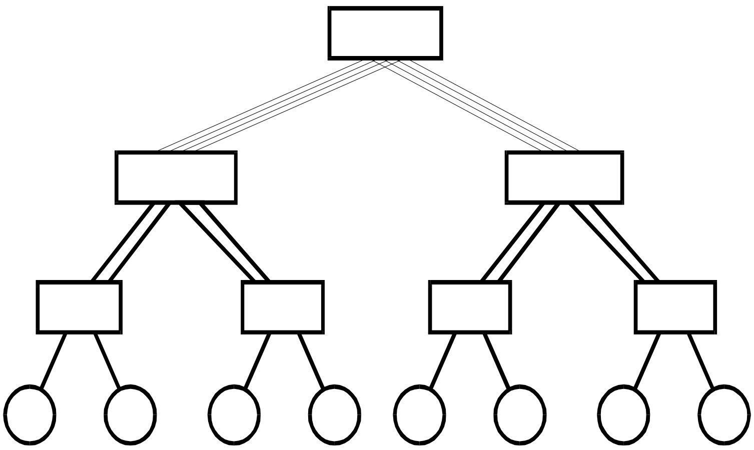


Binary tree with 7 processors



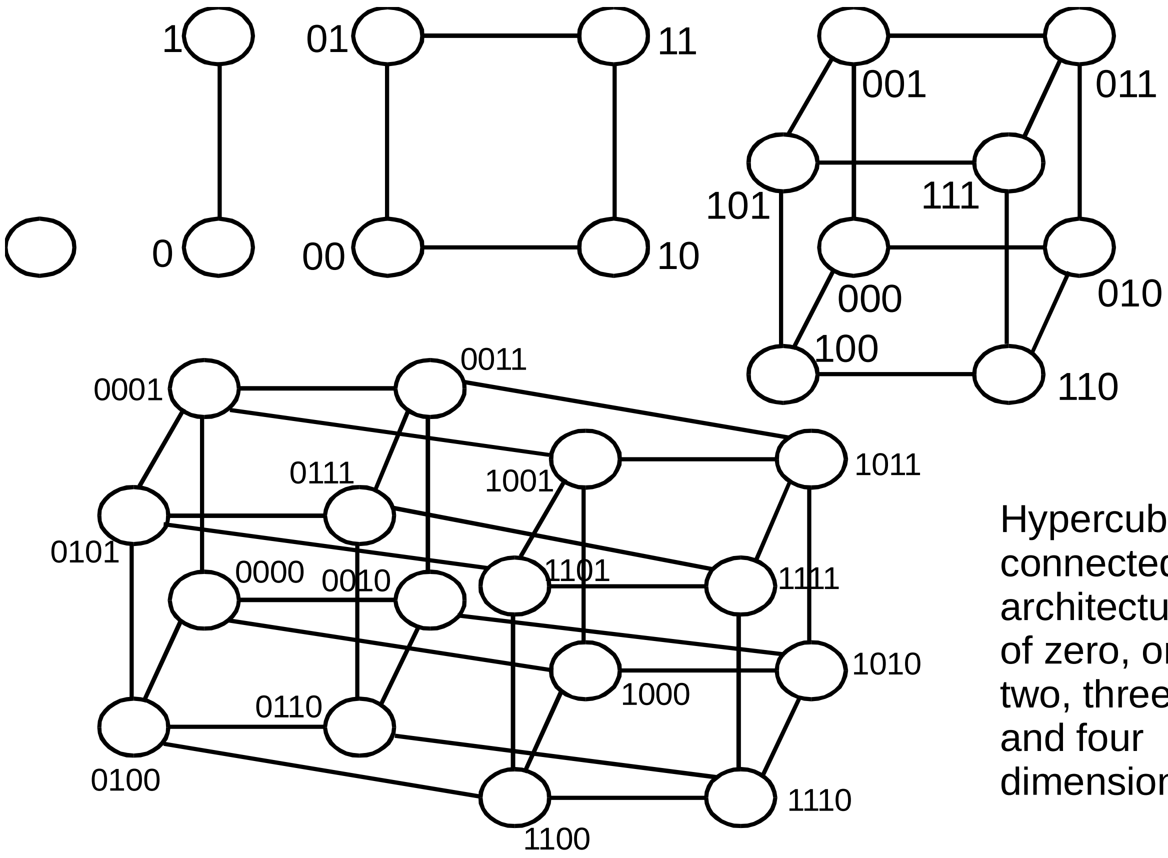
Binary tree with 4 processors
and 3 switching elements





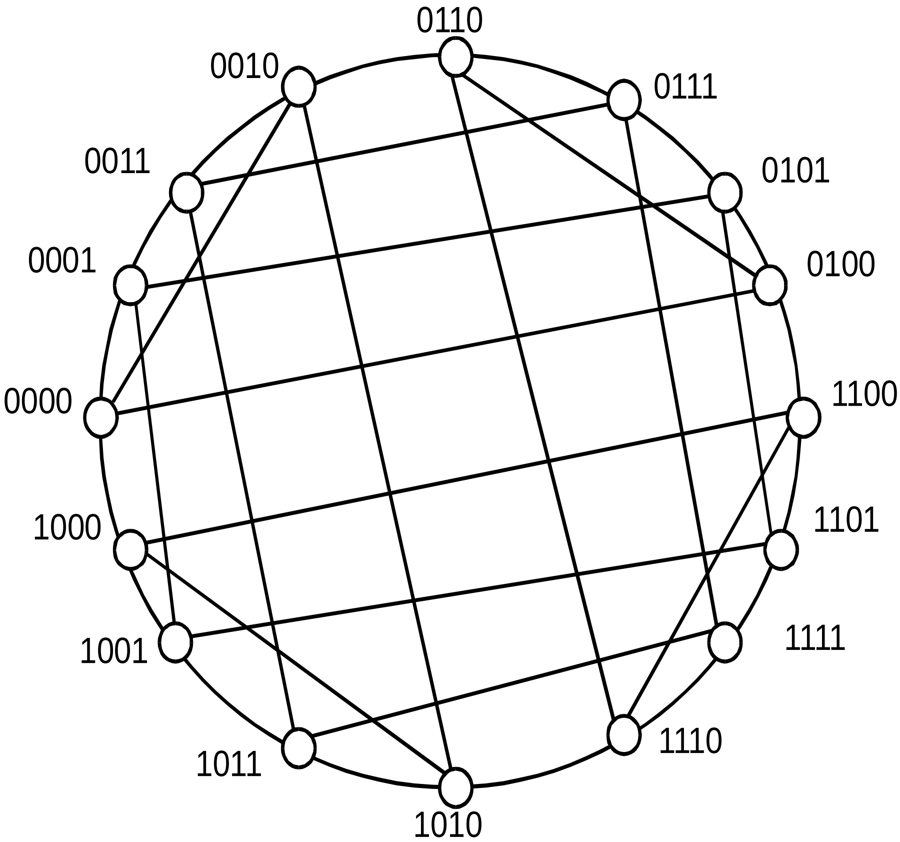
A fat tree network with 8 processors

0-D, 1-D, 2-D, 3-D, and 4-D Hypercube topologies



Hypercube connected architecture of zero, one, two, three and four dimensions

A 4-D Hypercube structure



Another way to create a hypercube: arranging processors along a circle, then numbering them by Gray-codes and connecting processors with numbers of unit Hamming distance

Parallel Models
10 Processor granularity

Granularity of a parallel computer may be defined as the ratio of time required for a basic communication operation to the time required for a basic computation.

Computers with a small ratio are suitable for algorithms requiring frequent communication; i.e. in these algorithms the size of computations between communications is small. These are algorithms with fine grain parallelism.

Classification based on the grain size:

Fine granularity
Medium granularity
Coarse granularity

Network	**Diameter**	**Bisection width**	**Arc connectivity**	**No. of links (cost)**
Completely-connected	1	$p^2/4$	p-1	p(p-1)/2
Star	2	1	1	p-1
Complete binary tree	2log((p+1)/2)	1	1	p-1
Linear array	p-1	1	1	p-1
Ring	p/2	2	2	p
2-D mesh	$2(\sqrt{p-1})$	$\sqrt{p}$	2	$2(p-\sqrt{p})$
2-D wraparound mesh	$2\sqrt{p}/2$	$2\sqrt{p}$	4	2p
Hypercube	log p	p/2	log p	(p log p)/2
Wraparound k-ary d cube	d k/2	$2k^{d-1}$	2d	dp

Summary of the characteristics of various static network topologies with *p* processors

Parallel Models
11 Summary of Models:

Computing Tool kit:

Algorithm Theory
Computer Languages
Software Implementation
Architectures
Hardware Implementation

Models exist for all levels:

Data-flow Model
Control-Flow Model
Functional Model
Operational Model
Topological Model
Connectivity Model
Routing Model
Data Exchange Protocol Model
Signal-Flow Model

Need for mapping (embedding) different models