CSE 431
Computer Architecture
Fall 2022
Introduction to Parallel Computers
Part II

(based partially on the material in Introduction to Parallel Computing by A. Grama, A. Gupta, G. Karypis, V. Kumar Second Edition, Addison Wesley

Kiwan Maeng

Slides adapted from Mahmut Taylan Kandemir (www.cse.psu.edu/~kandemir)

Project 2: Quick Run-Through

- setSizesOffsetAndMaskFields(acache, size, assoc, blocksize)
  - You must set the following variables
  - acache->numways
  - acache->blocksize
  - acache->numsets
  - acache->numBitsForBlockOffset
  - acache->numBitsForIndex
  - acache->VAImask
  - acache->VATmask
  - Try printing these out to see if you are getting reasonable results before doing anything!!!

Project 2: Quick Run-Through

- getindex(acache, address)
  - Get the index bits from the address
- gettag(acache, address)
  - Get the tag bits from the address

Project 2: Quick Run-Through

- writeback(acache, index, oldestway)
  - Write back the oldestway of the index to the next level (you only need to care about the data block)
  - 0. look at csim.h for the structure of acache
  - 1. select the block to write back
  - 2. write the block out to the next level
    - Calculate the address from the cache parameters
    - Get the value from the word in a block (you can only do it word by word)
    - Call Store(acache->nextcache, address, value)
    - Note that for multiword block cache, you have to do this multiple times!!!!

Project 2: Quick Run-Through

- fill(acache, index, oldestway, address)
  - Read the block from the next level and put it in the current block (you only need to care about the data block)
  - 0. look at csim.h for the structure of acache
  - 1. select the block to write into
  - 2. read the block from the next level and put it in the block
    - Calculate the address you want to read from **this can be a bit trickly**
    - Load(acache->nextcache, address) to read a word from the address
    - Put it in the block
    - Note that for multiword block cache, you have to do this multiple times!!!!
  - If you use a similar logic with the writeback() to calculate the address, it also kinda works… (because the tag is already set)

Project 2: Quick Run-Through

- This is a 64-bit machine!! So things are different that what we learned.
  - Byte offset is 3
  - Word size is 8-byte
- THIS IS MUCH HARDER!!!!!! START NOW!!!!!!
- Use Canvas discussion as much as possible (although it is bad..)

Dichotomy of Parallel Computing Platforms

- An explicitly parallel program must specify concurrency and interaction between concurrent subtasks.
- The former is sometimes also referred to as the control structure and the latter as the communication model.

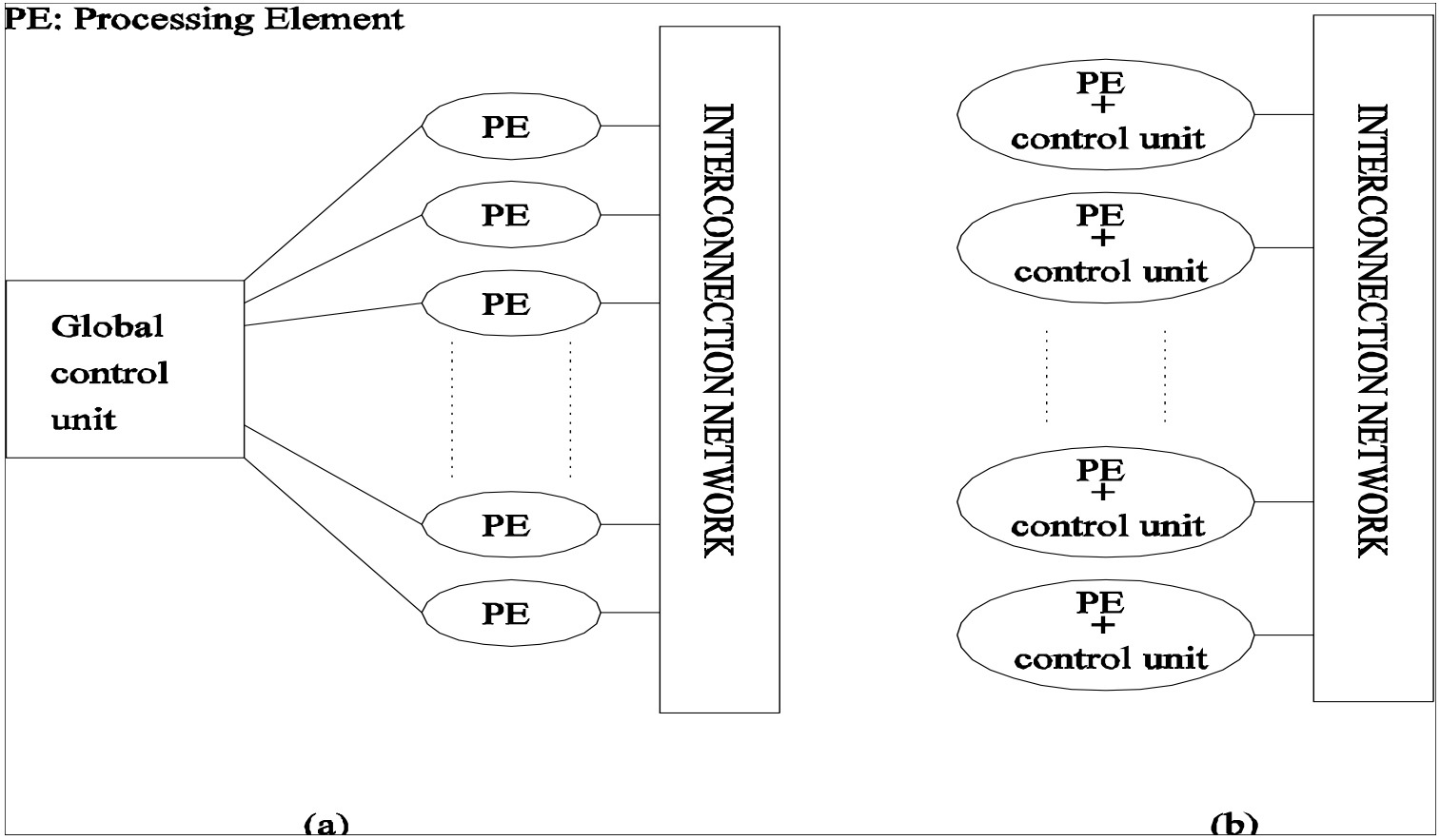
Control Structure of Parallel Programs

- Parallelism can be expressed at various levels of granularity - from instruction level to processes.
- Between these extremes exist a range of models, along with corresponding architectural support.

Control Structure of Parallel Programs

- Processing units in parallel computers either operate under the centralized control of a single control unit or work independently.
- If there is a single control unit that dispatches the same instruction to various processors (that work on different data), the model is referred to as Single Instruction, Multiple Data (SIMD).
- If each processor has its own control control unit, each processor can execute different instructions on different data items. This model is called Multiple Instruction, Multiple Data (MIMD).

SIMD and MIMD Processors

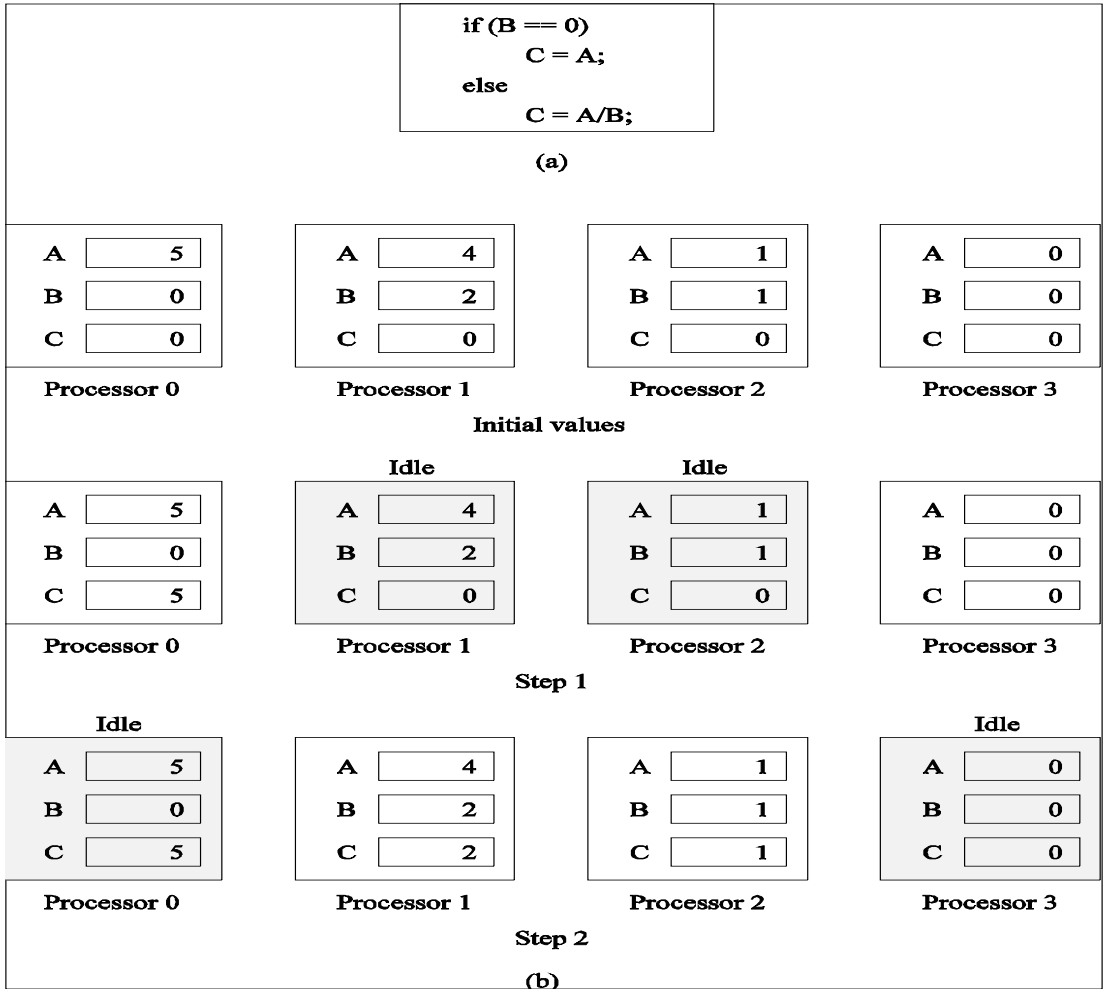


A typical SIMD architecture (a) and a typical MIMD architecture (b).

SIMD Processors

- Some of the earliest parallel computers such as the Illiac IV, MPP, DAP, CM-2, and MasPar MP-1 belonged to this class of machines.
- Variants of this concept have found use in co-processing units such as the MMX units in Intel processors and DSP chips such as the Sharc.
- SIMD relies on the regular structure of computations (such as those in image processing).
- It is often necessary to selectively turn off operations on certain data items. For this reason, most SIMD programming paradigms allow for an ``activity mask'', which determines if a processor should participate in a computation or not.

Conditional Execution in SIMD Processors



Executing a conditional statement on an SIMD computer with four processors: (a) the conditional statement; (b) the execution of the statement in two steps.

MIMD Processors

- In contrast to SIMD processors, MIMD processors can execute different programs on different processors.
- A variant of this, called single program multiple data streams (SPMD) executes the same program on different processors.
- It is easy to see that SPMD and MIMD are closely related in terms of programming flexibility and underlying architectural support.
- Examples of such platforms include current generation multiprocessor PCs and workstation clusters.

SIMD-MIMD Comparison

- SIMD computers require less hardware than MIMD computers (due to single control unit).
- However, since SIMD processors are specially designed, they tend to be expensive and have long design cycles.
- Not all applications are naturally suited to SIMD processors (image processing applications are).
- In contrast, platforms supporting the SPMD paradigm can be built from inexpensive off-the-shelf components with relatively little effort in a short amount of time.

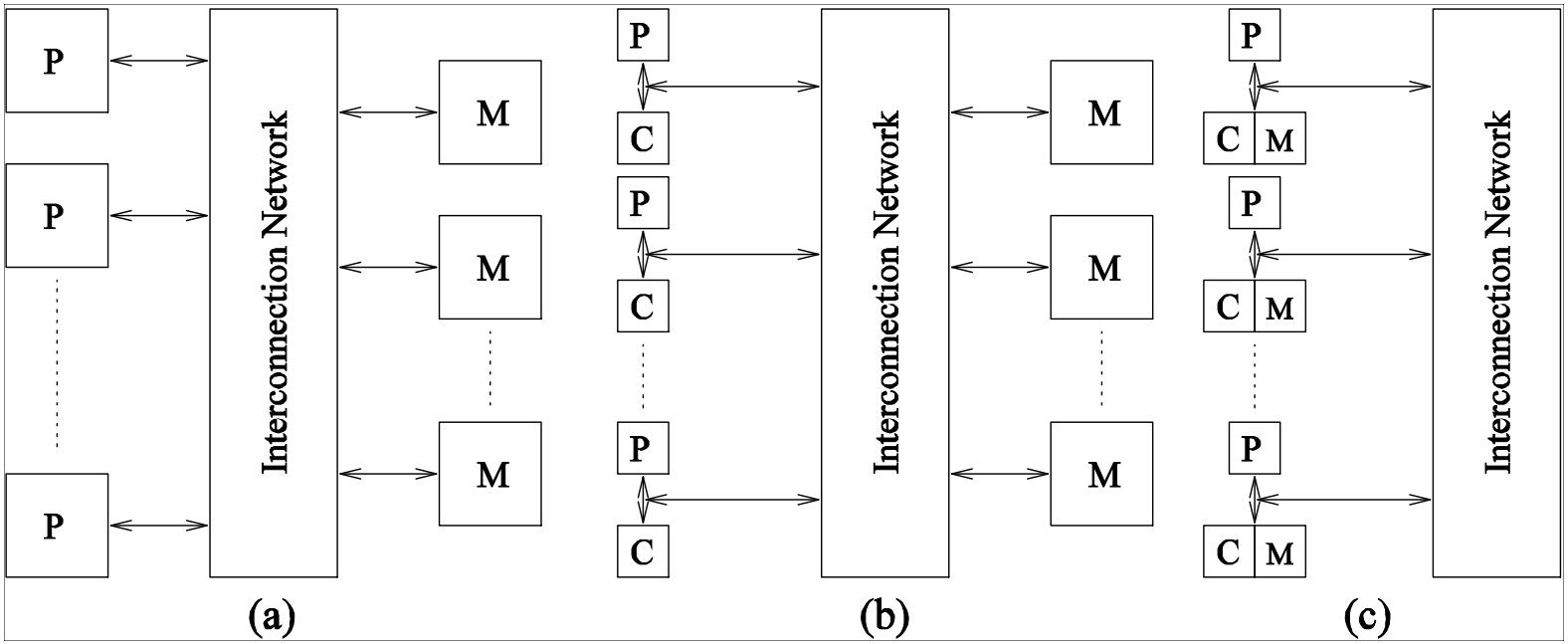
Communication Model of Parallel Platforms

- There are two primary forms of data exchange between parallel tasks - accessing a shared data space and exchanging messages.
- Platforms that provide a shared data space are called shared-address-space machines or multiprocessors.
- Platforms that support messaging are also called message-passing platforms or multi-computers.

Shared-Address-Space Platforms

- Part (or all) of the memory is accessible to all processors.
- Processors interact by modifying data objects stored in this shared-address-space.
- If the time taken by a processor to access any memory word in the system global or local is identical, the platform is classified as a uniform memory access (UMA), else, a non-uniform memory access (NUMA) machine.

NUMA and UMA Shared-Address-Space Platforms



Typical shared-address-space architectures: (a) Uniform-memory access shared-address-space computer; (b) Uniform-memory-access shared-address-space computer with caches and memories; (c) Non-uniform-memory-access shared-address-space computer with local memory only.

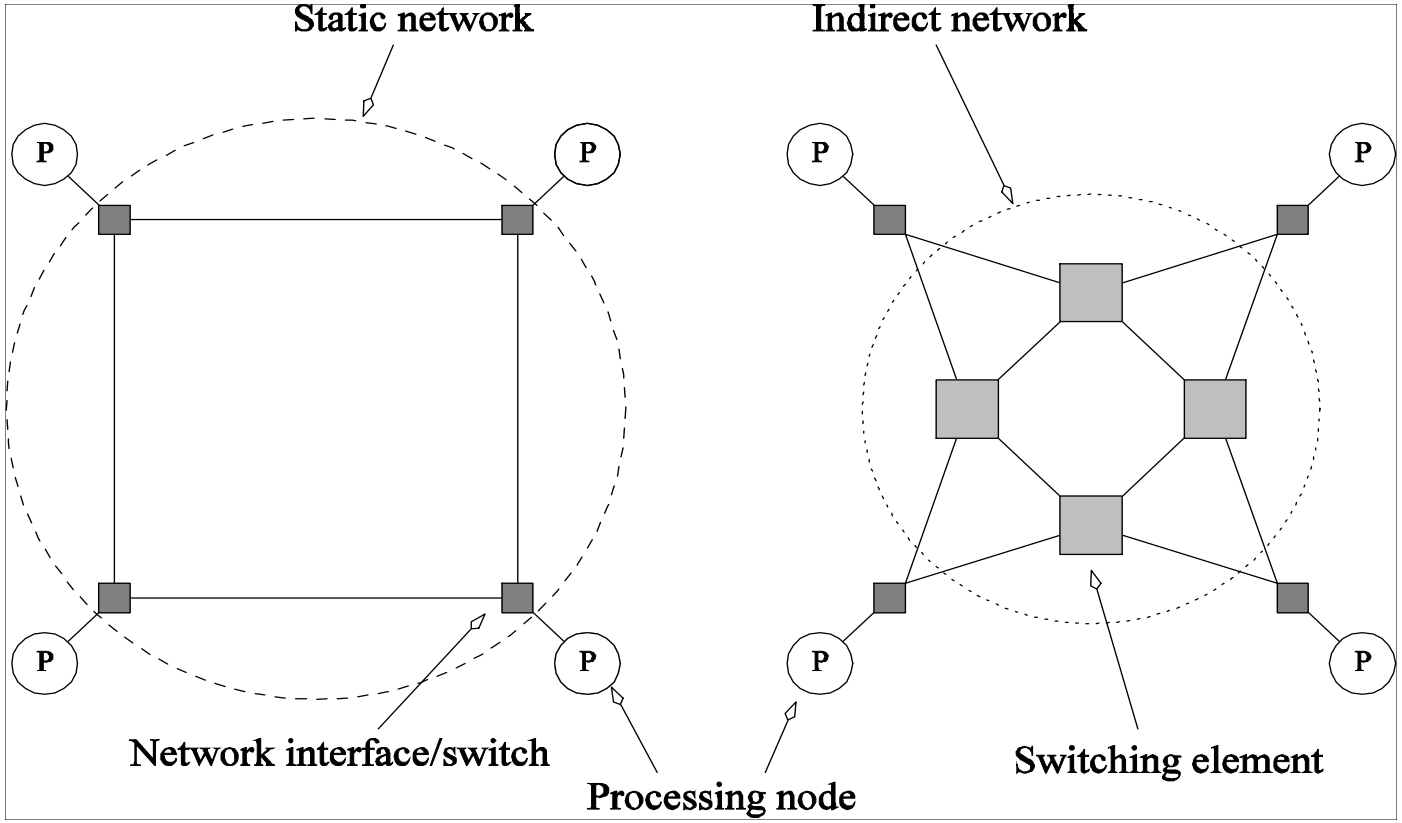
Message-Passing Platforms

- These platforms comprise of a set of processors and their own (exclusive) memory.
- Instances of such a view come naturally from clustered workstations and non-shared-address-space multicomputers.
- These platforms are programmed using (variants of) send and receive primitives.
- Libraries such as MPI provide such primitives.

Interconnection Networks for Parallel Computers

- Interconnection networks carry data between processors and to memory.
- Interconnects are made of switches and links (wires, fiber).
- Interconnects are classified as Static or Dynamic.
- Static networks consist of point-to-point communication links among processing nodes and are also referred to as Direct networks.
- Dynamic networks are built using switches and communication links. Dynamic networks are also referred to as Indirect networks.

Static and Dynamic Interconnection Networks



Classification of interconnection networks: (a) a static network; and (b) a dynamic network.

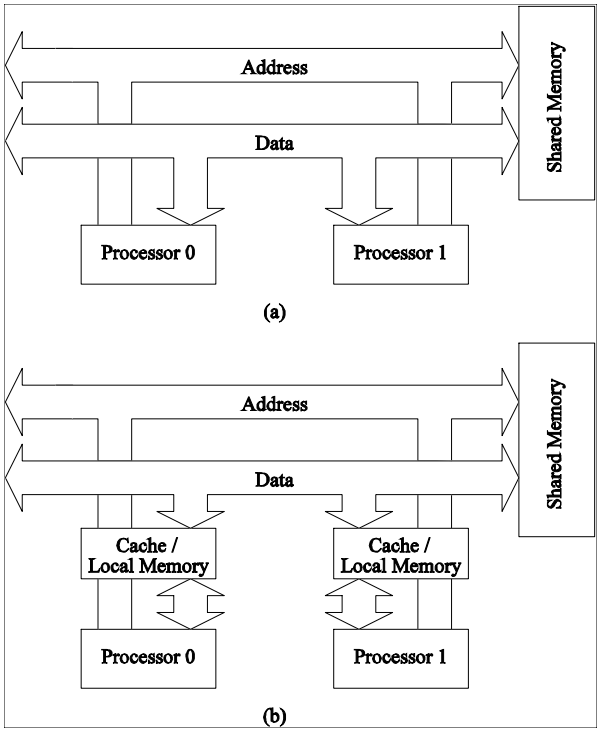
Network Topologies

- A variety of network topologies have been proposed and implemented.
- These topologies tradeoff performance for cost.
- Commercial machines often implement hybrids of multiple topologies for reasons of packaging, cost, and available components.

Network Topologies: Buses

- Some of the simplest and earliest parallel machines used buses.
- All processors access a common bus for exchanging data.
- The distance between any two nodes is *O(1)* in a bus. The bus also provides a convenient broadcast media.
- However, the bandwidth of the shared bus is a major bottleneck.
- Typical bus based machines are limited to dozens of nodes. Sun Enterprise servers and Intel Pentium based shared-bus multiprocessors are examples of such architectures.

Network Topologies: Buses

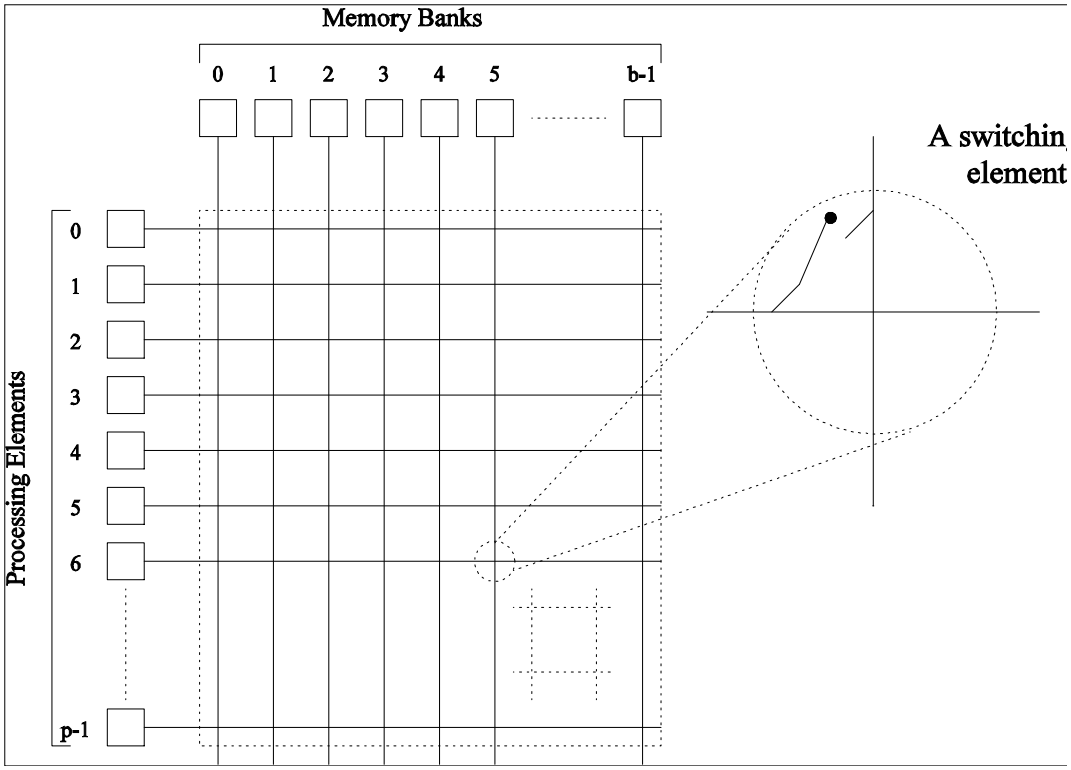


Bus-based interconnects (a) with no local caches; (b) with local memory/caches.

Since much of the data accessed by processors is local to the processor, a local memory can improve the performance of bus-based machines.

Network Topologies: Crossbars

A crossbar network uses an $p \times m$ grid of switches to connect p inputs to m outputs in a non-blocking manner.

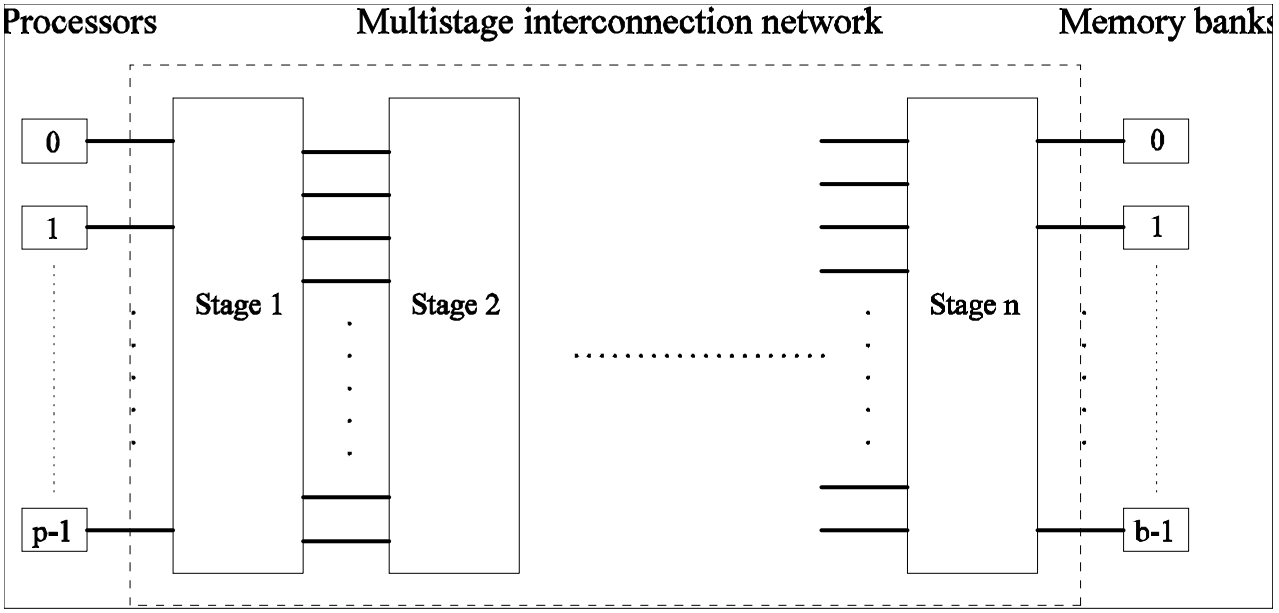


A completely non-blocking crossbar network connecting p processors to b memory banks.

Network Topologies: Multistage Networks

- Crossbars have excellent performance scalability but poor cost scalability.
- Buses have excellent cost scalability, but poor performance scalability.
- Multistage interconnects strike a *compromise* between these extremes.

Network Topologies: Multistage Networks



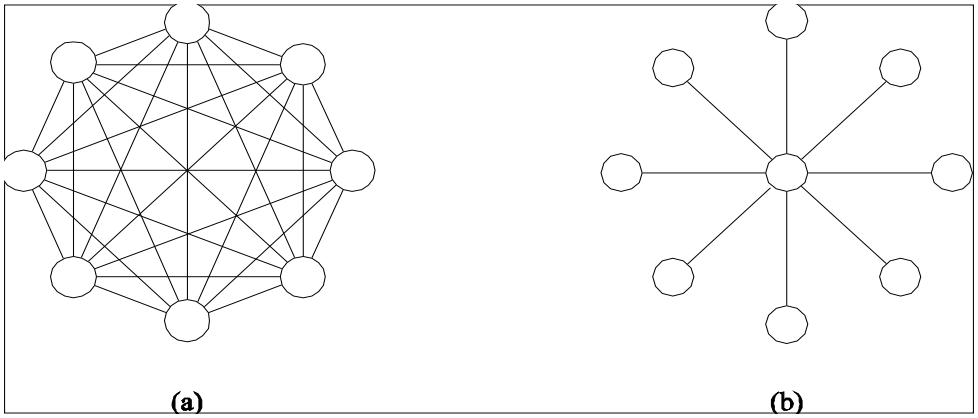
The schematic of a typical multistage interconnection network.

Network Topologies: Multistage Omega Network

- One of the most commonly used multistage interconnects is the Omega network.
- This network consists of *log p* stages, where *p* is the number of inputs/outputs.
- At each stage, input *i* is connected to output *j* if:

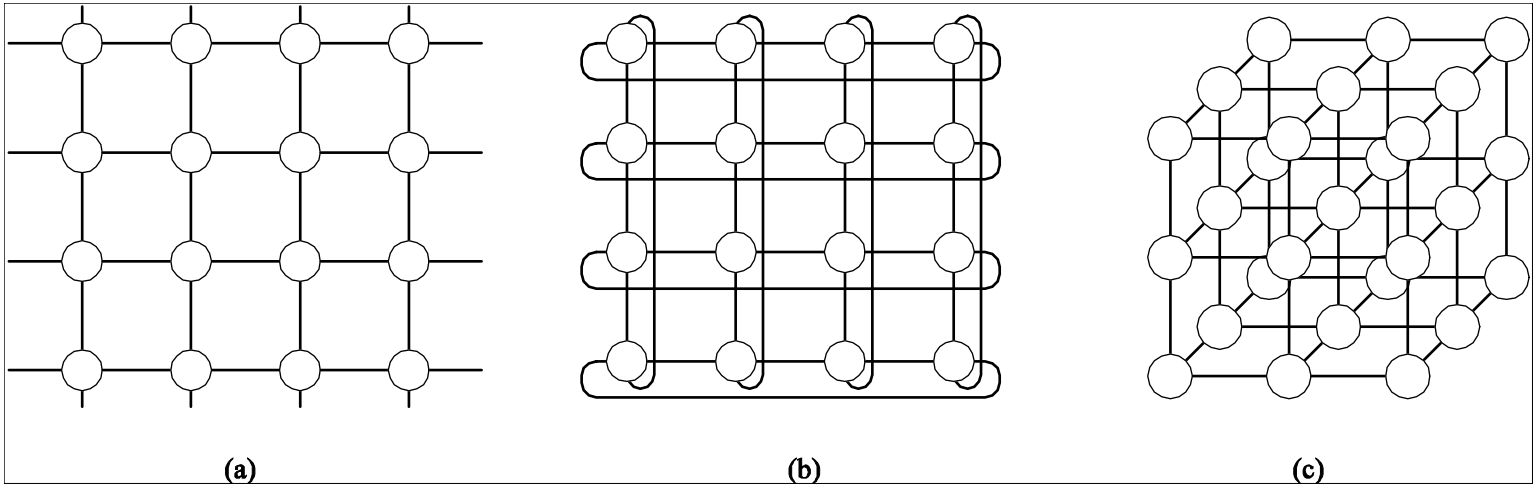
$$j = \begin{cases} 2i, & 0 \leq i \leq p/2 - 1 \\ 2i + 1 - p, & p/2 \leq i \leq p - 1 \end{cases}$$

Network Topologies: Completely Connected and Star Connected Networks



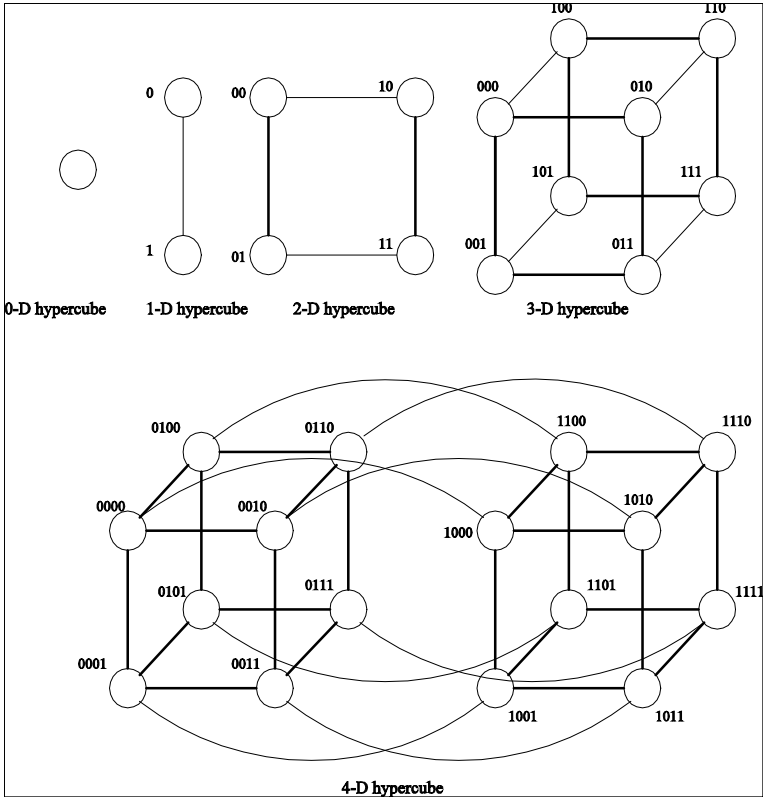
(a) A completely-connected network of eight nodes;
(b) a star connected network of nine nodes.

Network Topologies: Meshes



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.

Network Topologies: Hypercubes

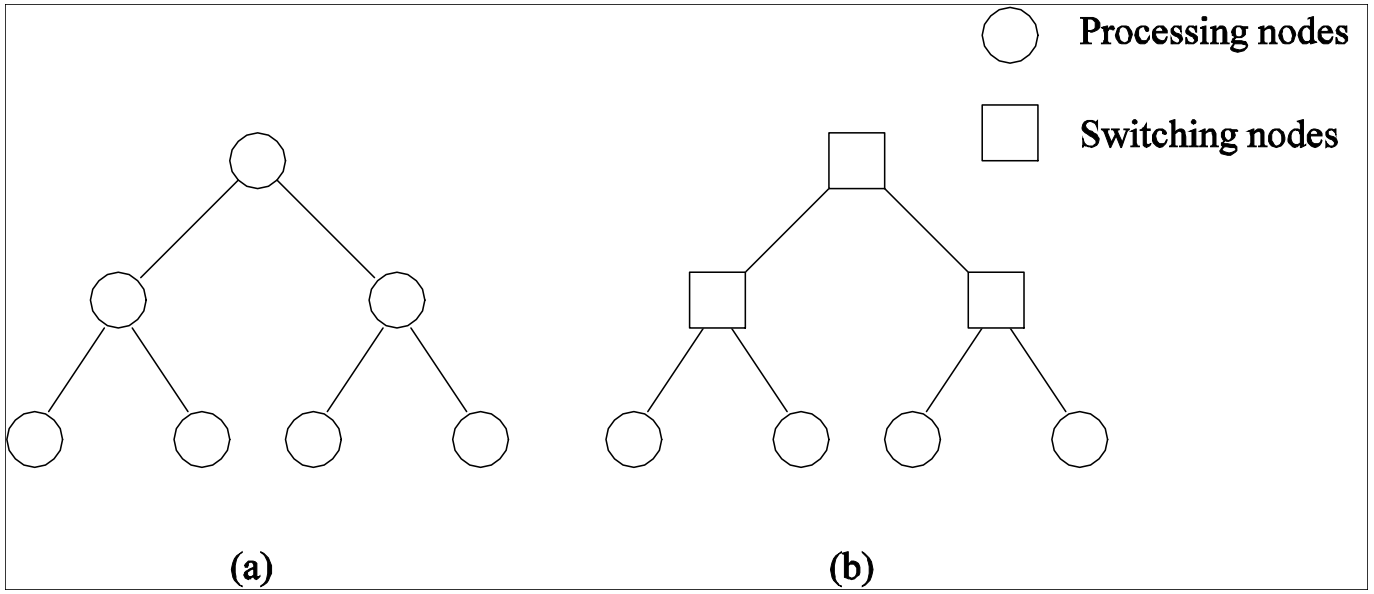


Construction of hypercubes from hypercubes of lower dimension.

Network Topologies: Properties of Hypercubes

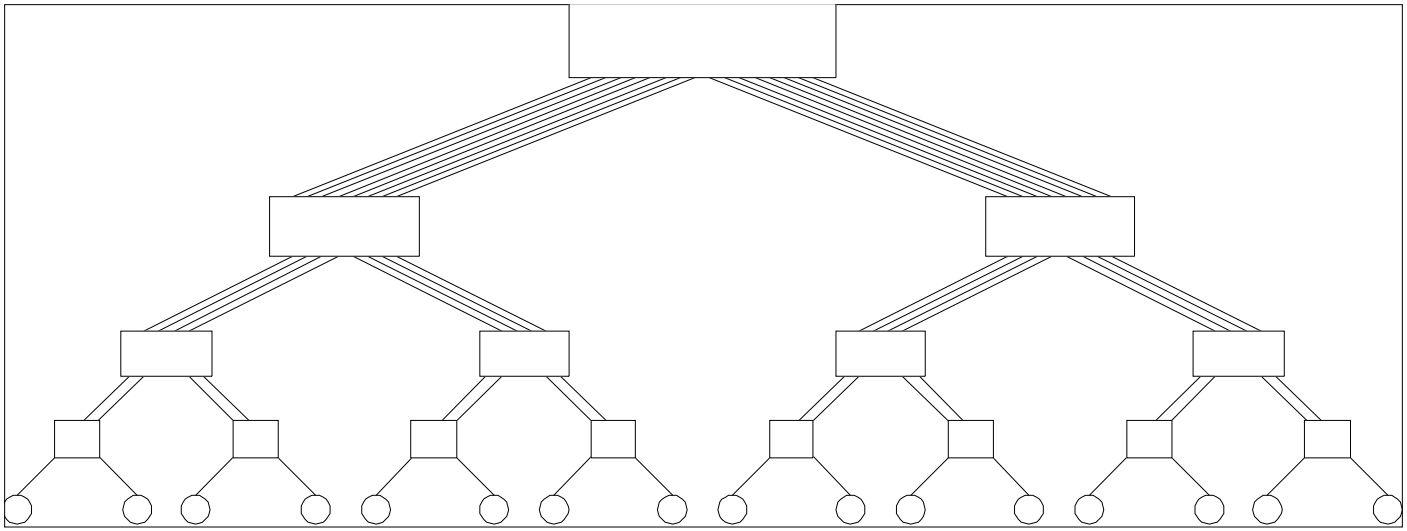
- The distance between any two nodes is at most *log p*.
- Each node has *log p* neighbors.
- The distance between two nodes is given by the number of bit positions at which the two nodes differ.

Network Topologies: Tree-Based Networks



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

Network Topologies: Fat Trees



A fat tree network of 16 processing nodes.

Manycores with On-Chip Networks

- An example 5x5 manycore system connected through 2D mesh on-chip network – also known as Network-on-Chip (NoC)
- A node contains a core, a private L1, private LLC/shared LLC bank, and network routing circuitry
- Data movements in NoC are caused by data accesses
- Directory accesses are not shown here