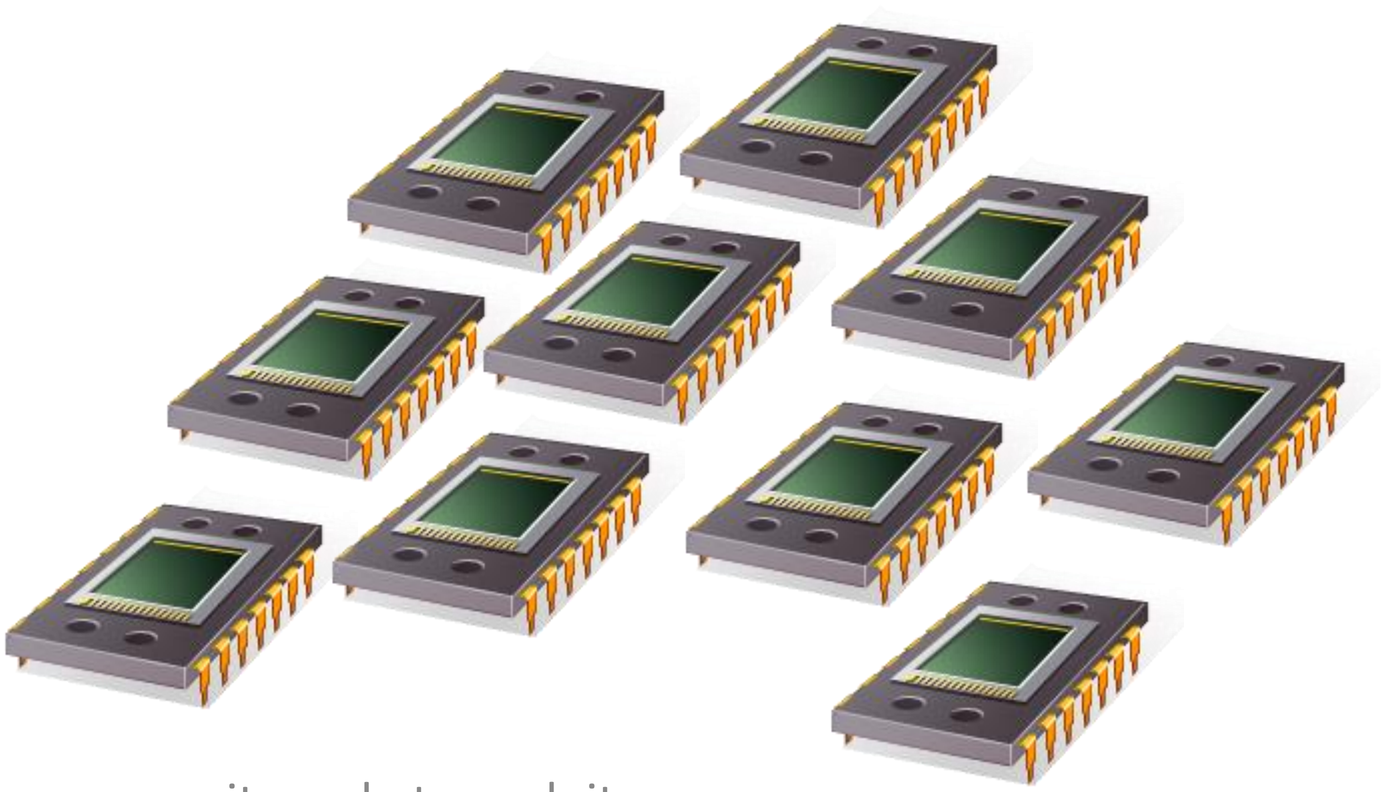
Parallel Programming

Parallel Hardware

Slides adapted from the lecture notes by Peter Pacheco



A programmer can write code to exploit.

PARALLEL HARDWARE

Flynn's Taxonomy

classic von Neumann

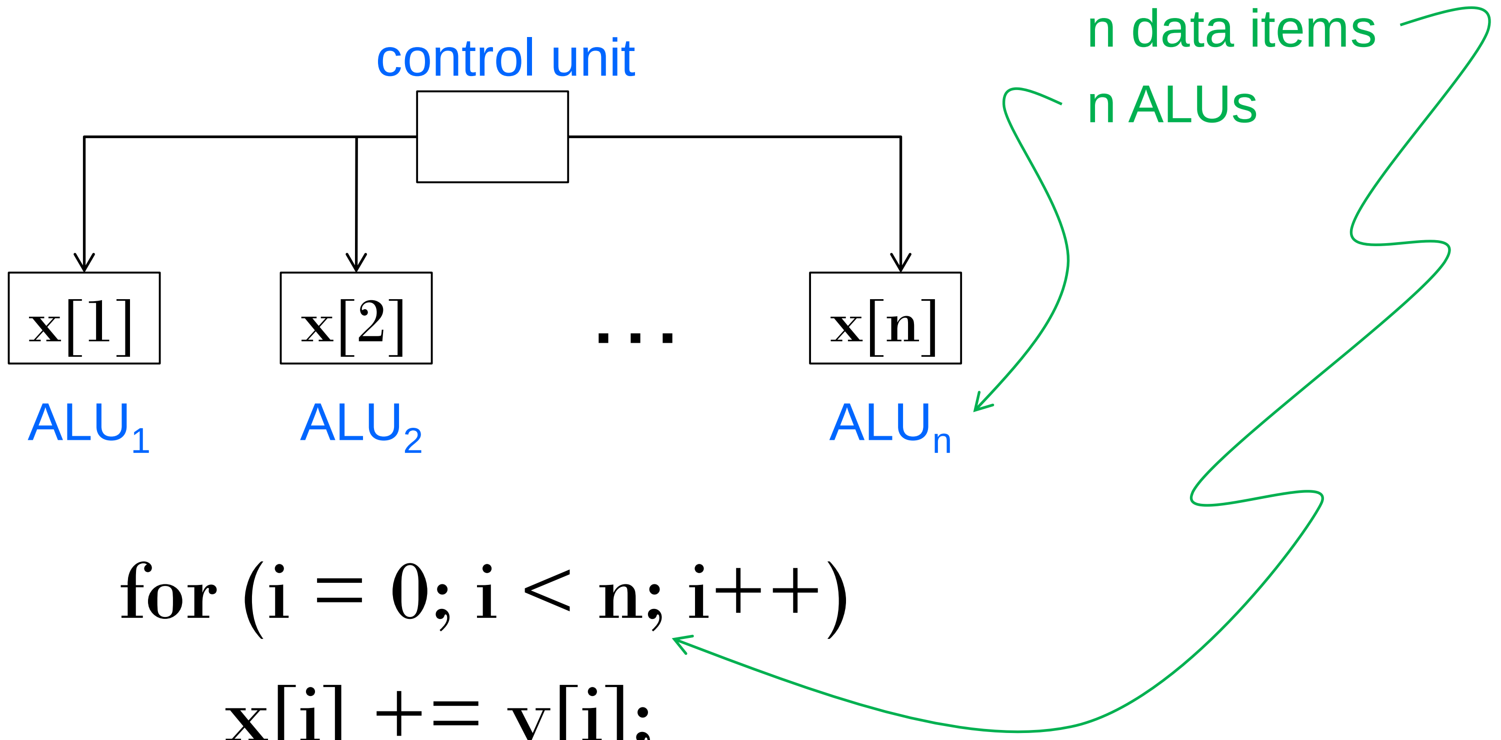
SISD Single instruction stream Single data stream	(SIMD) Single instruction stream Multiple data stream
MISD Multiple instruction stream Single data stream	(MIMD) Multiple instruction stream Multiple data stream

not covered

SIMD

- Parallelism achieved by dividing data among the processors.
- Applies the same instruction to multiple data items.
- Called data parallelism.

SIMD example



```
for (i = 0; i < n; i++)
    x[i] += y[i];
```

SIMD

- What if we don’t have as many ALUs as data items?
- Divide the work and process iteratively.
- Ex. m = 4 ALUs   and   n = 15 data items.

Round3	ALU_1	ALU_2	ALU_3	ALU_4
1	X[0]	X[1]	X[2]	X[3]
2	X[4]	X[5]	X[6]	X[7]
3	X[8]	X[9]	X[10]	X[11]
4	X[12]	X[13]	X[14]	

SIMD drawbacks

- All ALUs are required to execute the same instruction, or remain idle.
- In traditional design, they must also operate synchronously.
- The ALUs have no instruction storage.
- Efficient for large data parallel problems, but not other types of more complex parallel problems.

Vector processors (1)

- Operate on arrays or vectors of data while conventional CPU’s operate on individual data elements or scalars.

- Vector registers
  - Capable of storing a vector of operands and operating simultaneously on their contents.

Vector processors (2)

- Vectorized and pipelined functional units
  - The same operation is applied to each element in the vector (or pairs of elements).
- Vector instructions
  - Operate on vectors rather than scalars.

Vector processors (3)

- Interleaved memory
  - Multiple “banks” of memory, which can be accessed more or less independently.
  - Distribute elements of a vector across multiple banks, so reduce or eliminate delay in loading/storing successive elements.
- Strided memory access and hardware scatter/gather
  - The program accesses elements of a vector located at fixed intervals.

Vector processors - Pros

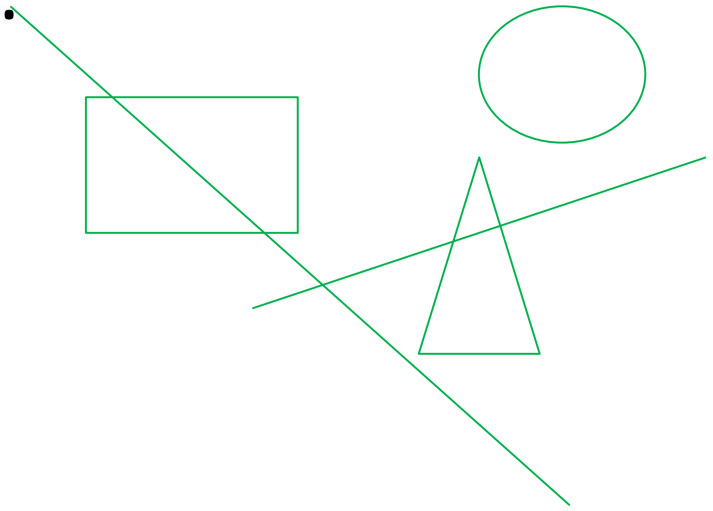
- Fast.
- Easy to use.
- Vectorizing compilers are good at identifying code to exploit.
- Compilers also can provide information about code that cannot be vectorized.
  - Helps the programmer re-evaluate code.
- High memory bandwidth.
- Uses every item in a cache line.

Vector processors - Cons

- They don’t handle irregular data structures as well as other parallel architectures.
- Limited to their ability to handle ever larger problems. (scalability)

Graphics Processing Units (GPU)

- Real time graphics application programming interfaces or API's use points, lines, and triangles to internally represent the surface of an object.



GPUs

- A graphics processing pipeline converts the internal representation into an array of pixels that can be sent to a computer screen.

- Several stages of this pipeline (called shader functions) are programmable.
  - Typically just a few lines of C code.

GPUs

- Shader functions are also implicitly parallel, since they can be applied to multiple elements in the graphics stream.

- GPU’s can often optimize performance by using SIMD parallelism.
- The current generation of GPU’s use SIMD parallelism.
  - Although they are not pure SIMD systems.

MIMD

- Supports multiple simultaneous instruction streams operating on multiple data streams.

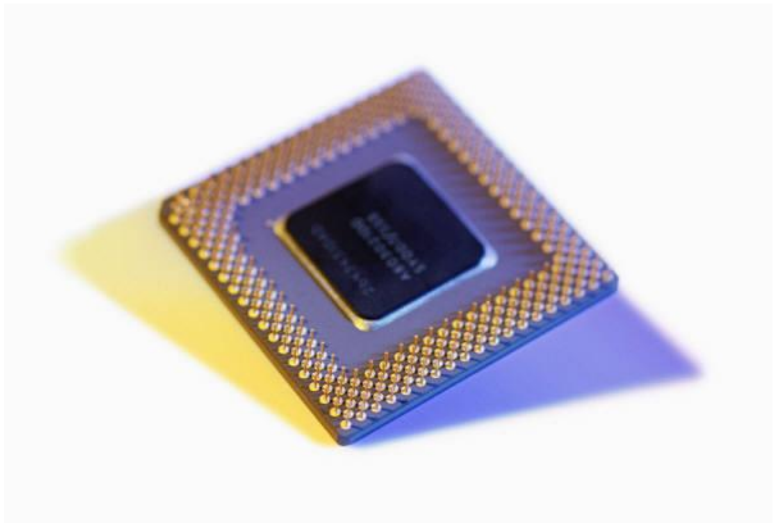
- Typically consist of a collection of fully independent processing units or cores, each of which has its own control unit and its own ALU.

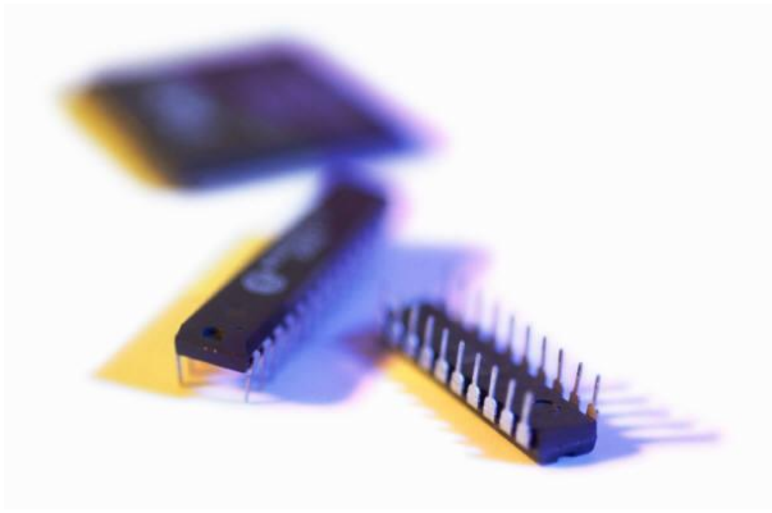
Shared Memory System (1)

- A collection of autonomous processors is connected to a memory system via an interconnection network.
- Each processor can access each memory location.
- The processors usually communicate implicitly by accessing shared data structures.

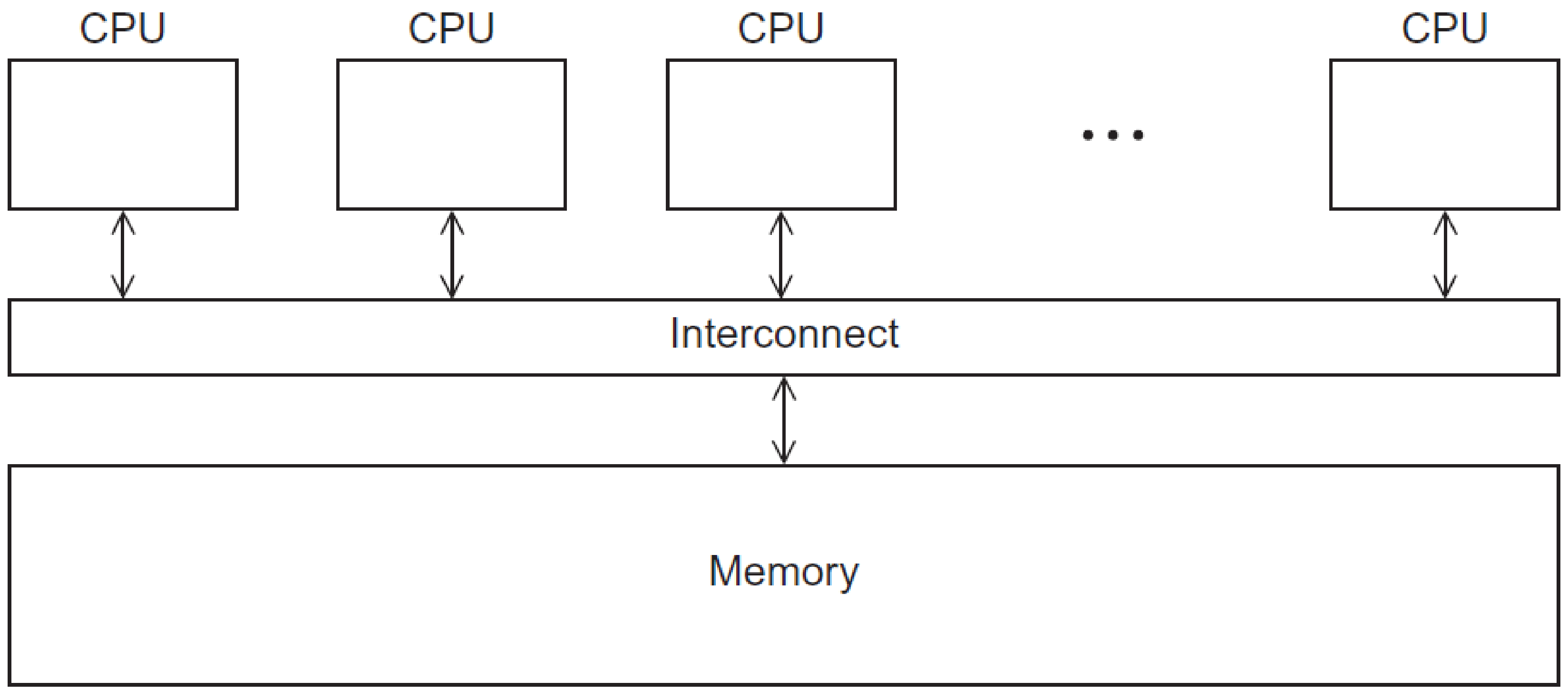
Shared Memory System (2)

- Most widely available shared memory systems use one or more multicore processors.
  - (multiple CPU's or cores on a single chip)

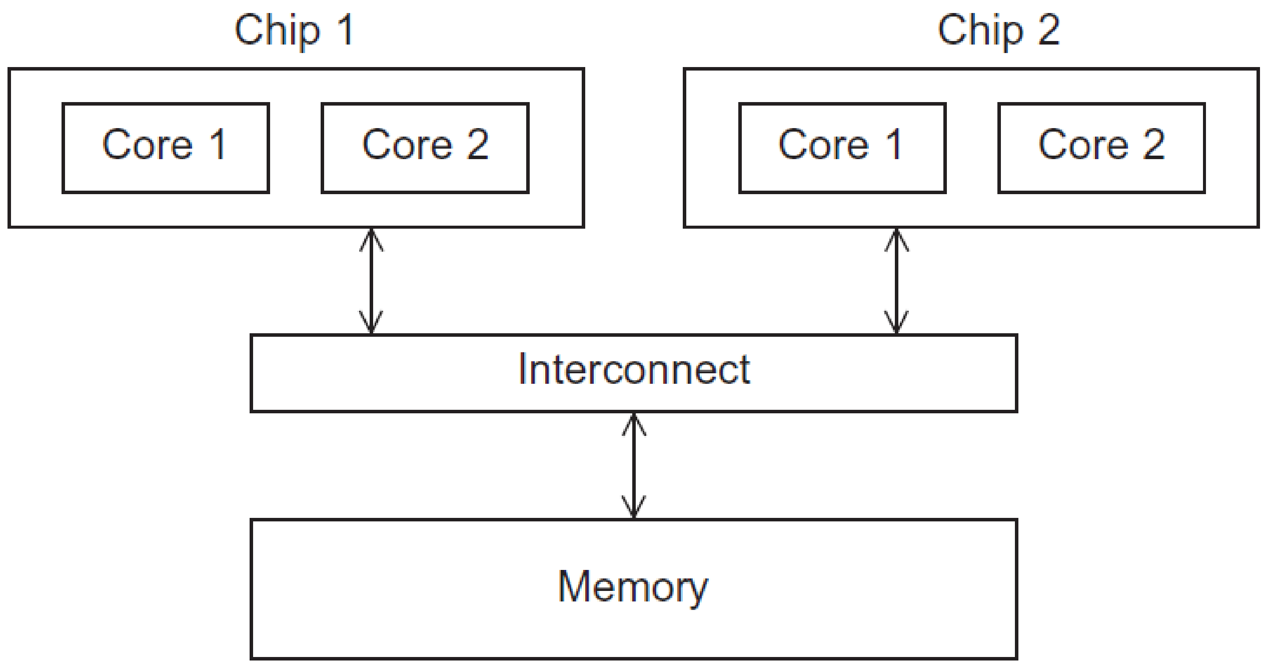




Shared Memory System

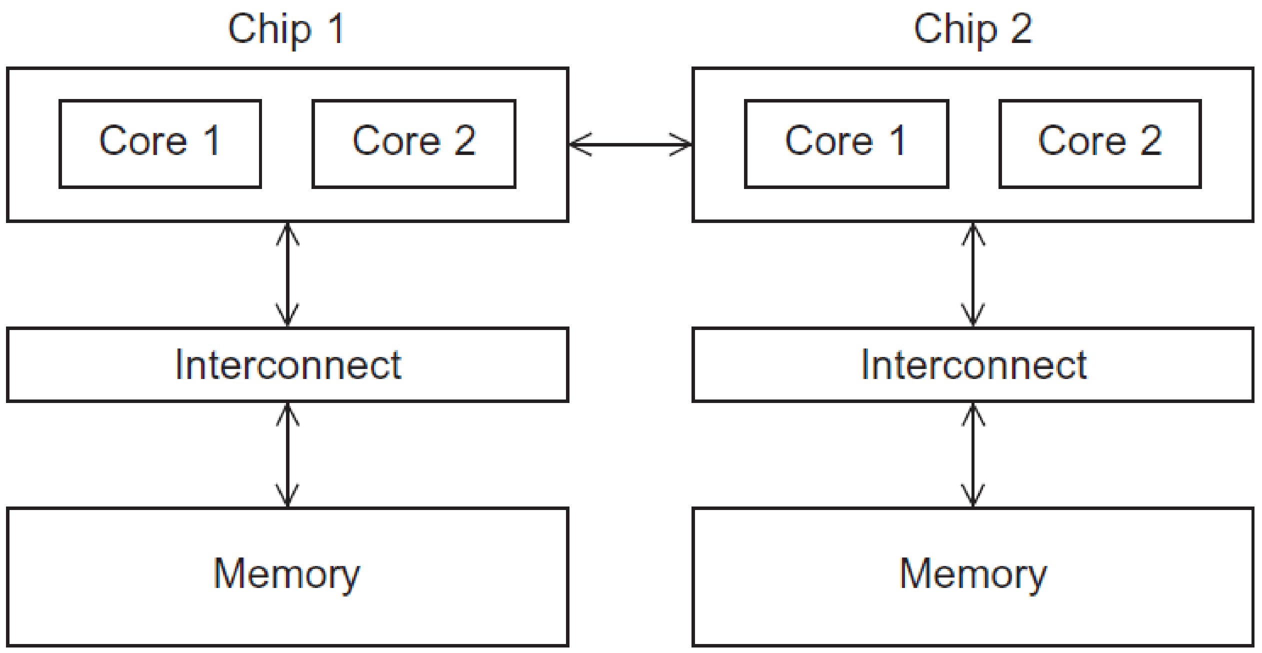


UMA multicore system



Time to access all the memory locations is the same for all the cores.

NUMA multicore system



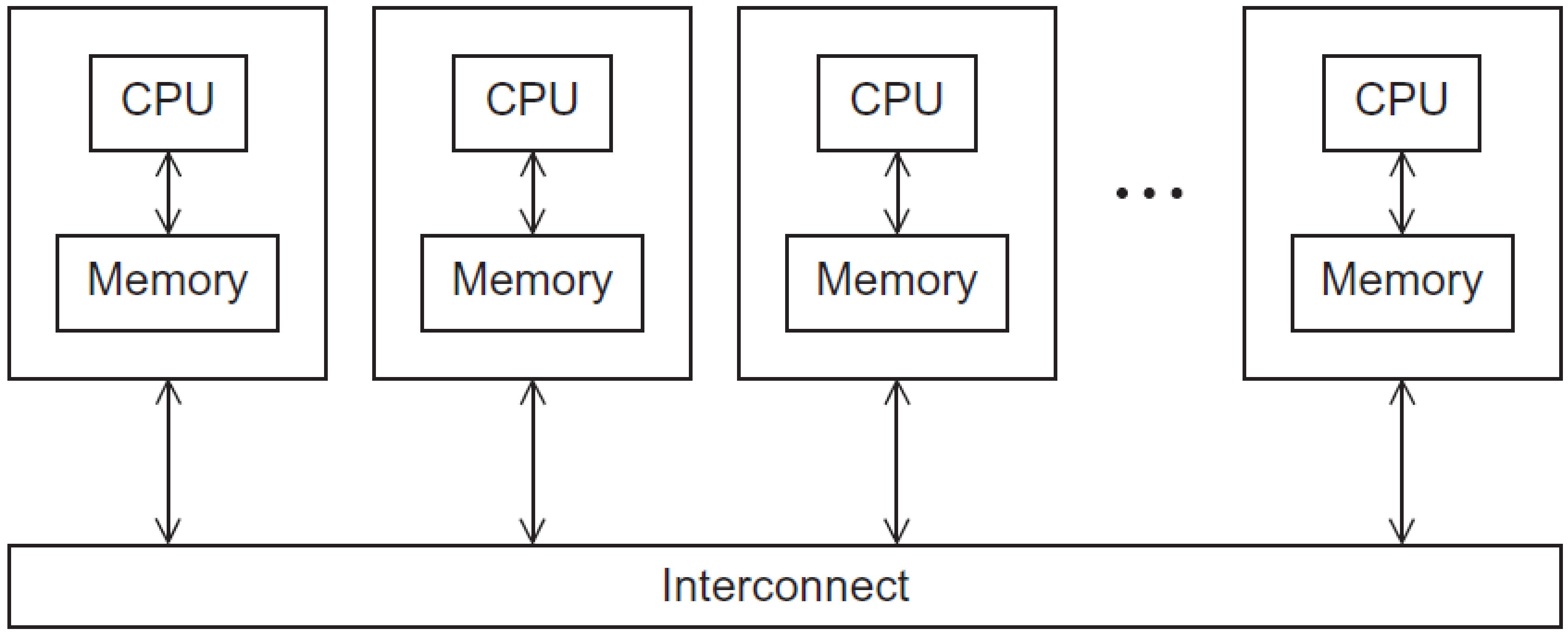
A memory location a core is directly connected to can be accessed faster than a memory location that must be accessed through another chip.

Distributed Memory System

- Clusters (most popular form in practice)
  - A collection of commodity systems.
  - Connected by a commodity interconnection network.
- Nodes of a cluster are individual computation units joined by a communication network.

a.k.a. hybrid systems

Distributed Memory System



Interconnection networks

- Affects performance of both distributed and shared memory systems.

- Two categories:
  - Shared memory interconnects
  - Distributed memory interconnects

Shared memory interconnects

- Bus interconnect
  - A collection of parallel communication wires together with some hardware that controls access to the bus.
  - Communication wires are shared by the devices that are connected to it.
  - As the number of devices connected to the bus increases, contention for use of the bus increases, and performance decreases.

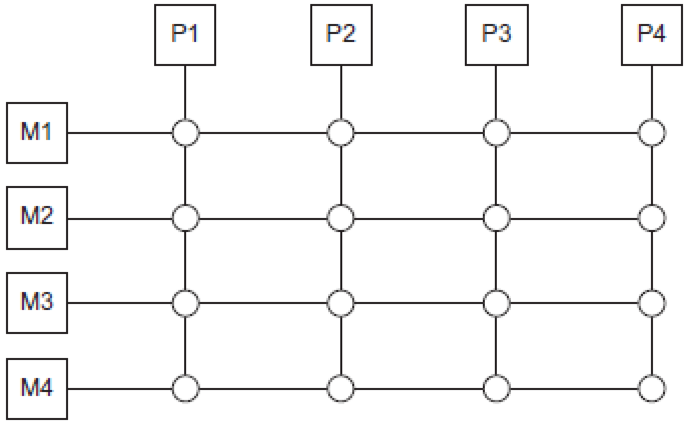
Shared memory interconnects

- Switched interconnect
  - Uses switches to control the routing of data among the connected devices.

  - Crossbar –
    - Allows simultaneous communication among different devices.
    - Faster than buses.
    - But the cost of the switches and links is relatively high.

(a)

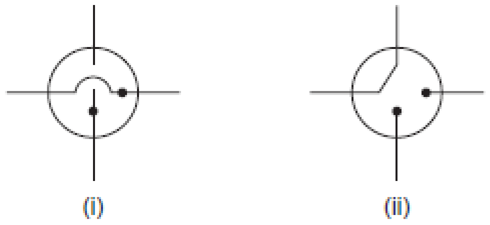
A crossbar switch connecting 4 processors (P_i) and 4 memory modules (M_j)



(a)

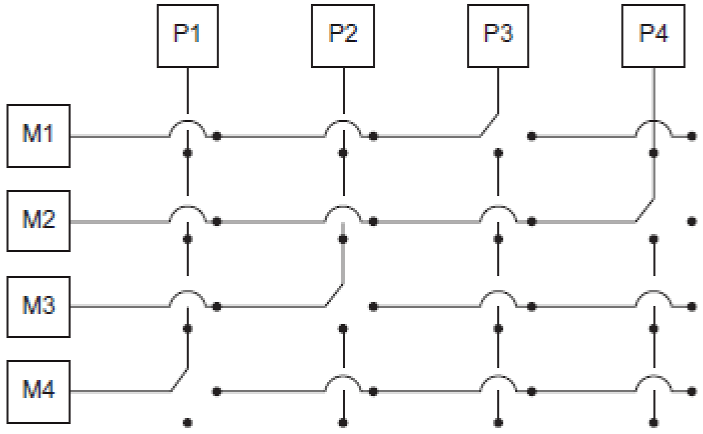
(b)

Configuration of internal switches in a crossbar



(b)

(c) Simultaneous memory accesses by the processors

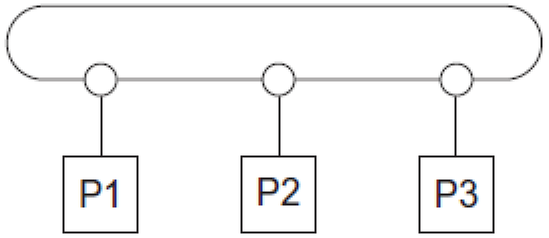


(c)

Distributed memory interconnects

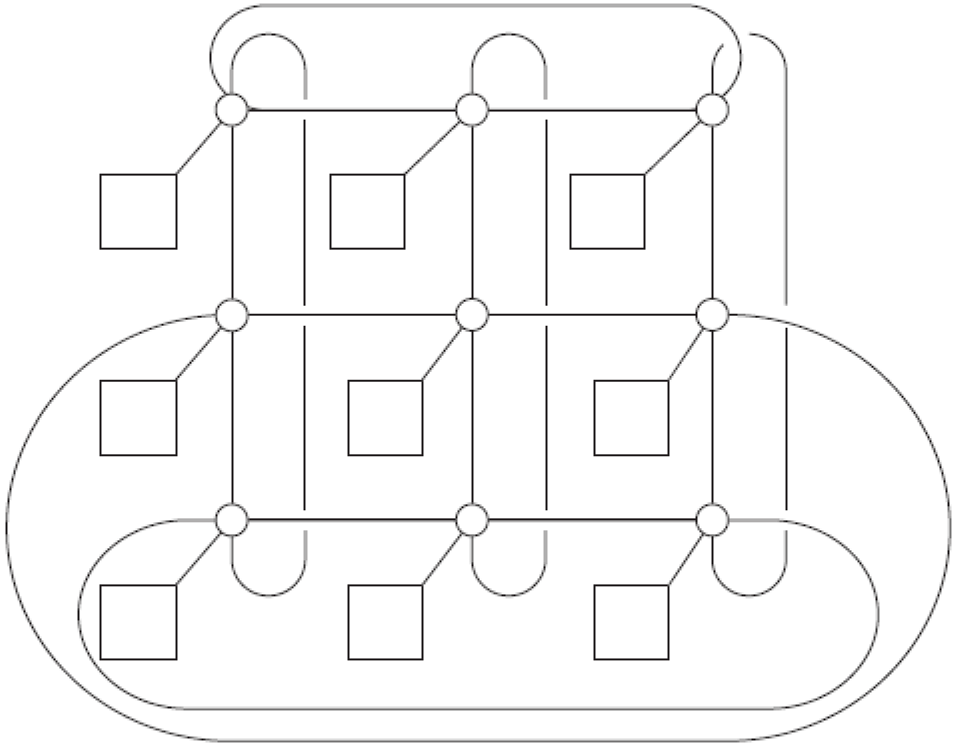
- Two groups
  - Direct interconnect
    - Each switch is directly connected to a processor memory pair, and the switches are connected to each other.
  - Indirect interconnect
    - Switches may not be directly connected to a processor.

Direct interconnect



(a)

ring



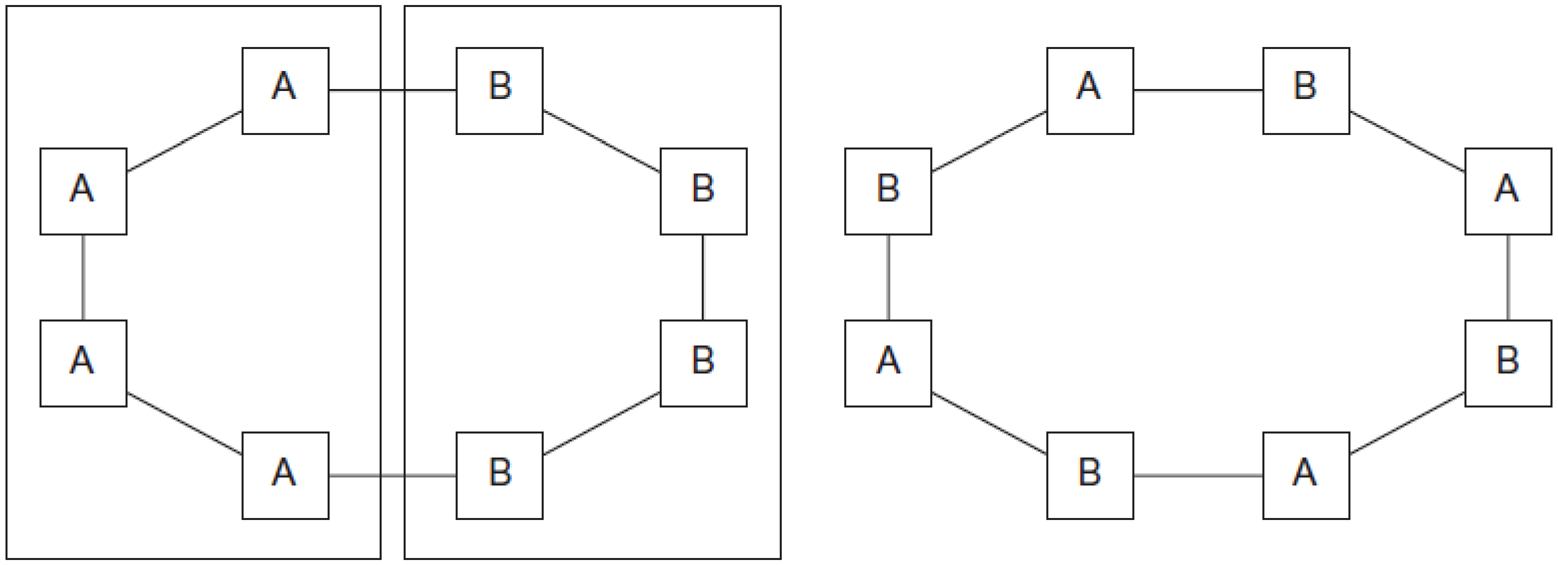
(b)

toroidal mesh

Bisection width

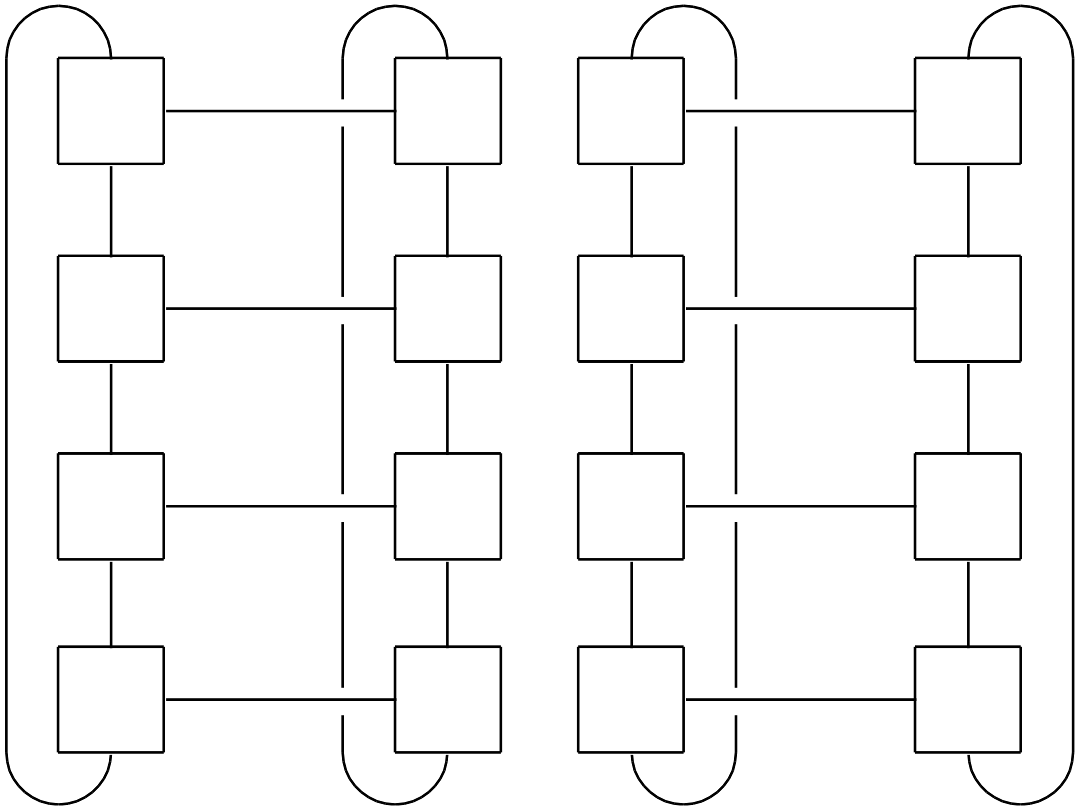
- A measure of "number of simultaneous communications" or "connectivity".
- How many simultaneous communications can take place "across the divide" between the halves?

Two bisections of a ring



(a) (b)

A bisection of a toroidal mesh



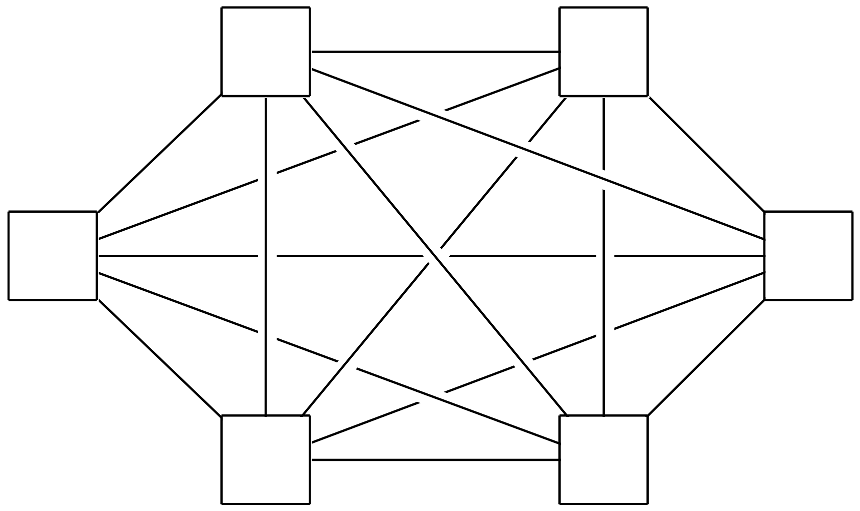
Definitions

- Bandwidth
  - The rate at which a link can transmit data.
  - Usually given in megabits or megabytes per second.
- Bisection bandwidth
  - A measure of network quality.
  - Instead of counting the number of links joining the halves, it sums the bandwidth of the links.

Fully connected network

- Each switch is directly connected to every other switch.

impractical

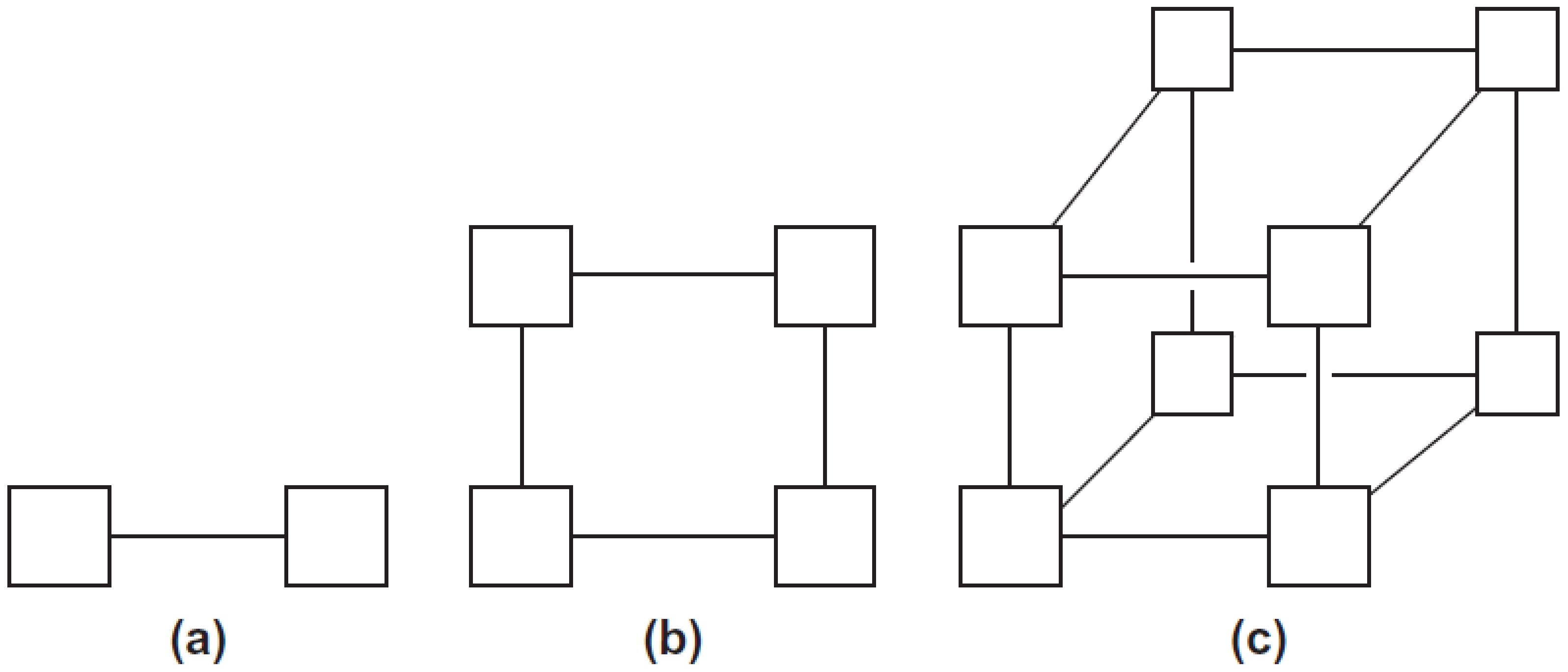


bisection width = $p^2/4$

Hypercube

- Highly connected direct interconnect.
- Built inductively:
  - A one-dimensional hypercube is a fully-connected system with two processors.
  - A two-dimensional hypercube is built from two one-dimensional hypercubes by joining "corresponding" switches.
  - Similarly a three-dimensional hypercube is built from two two-dimensional hypercubes.

Hypercubes



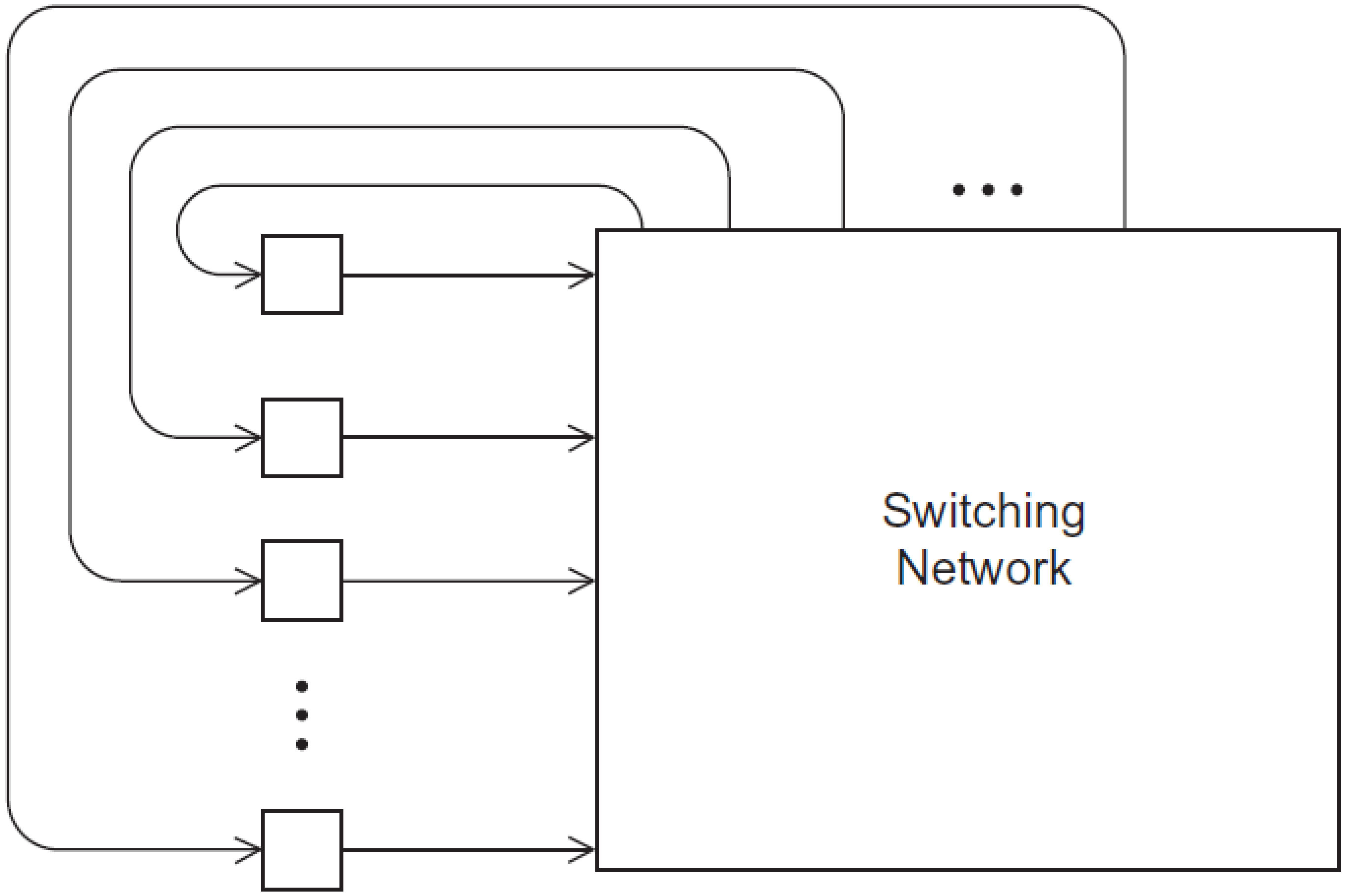
one- two- three-dimensional

Indirect interconnects

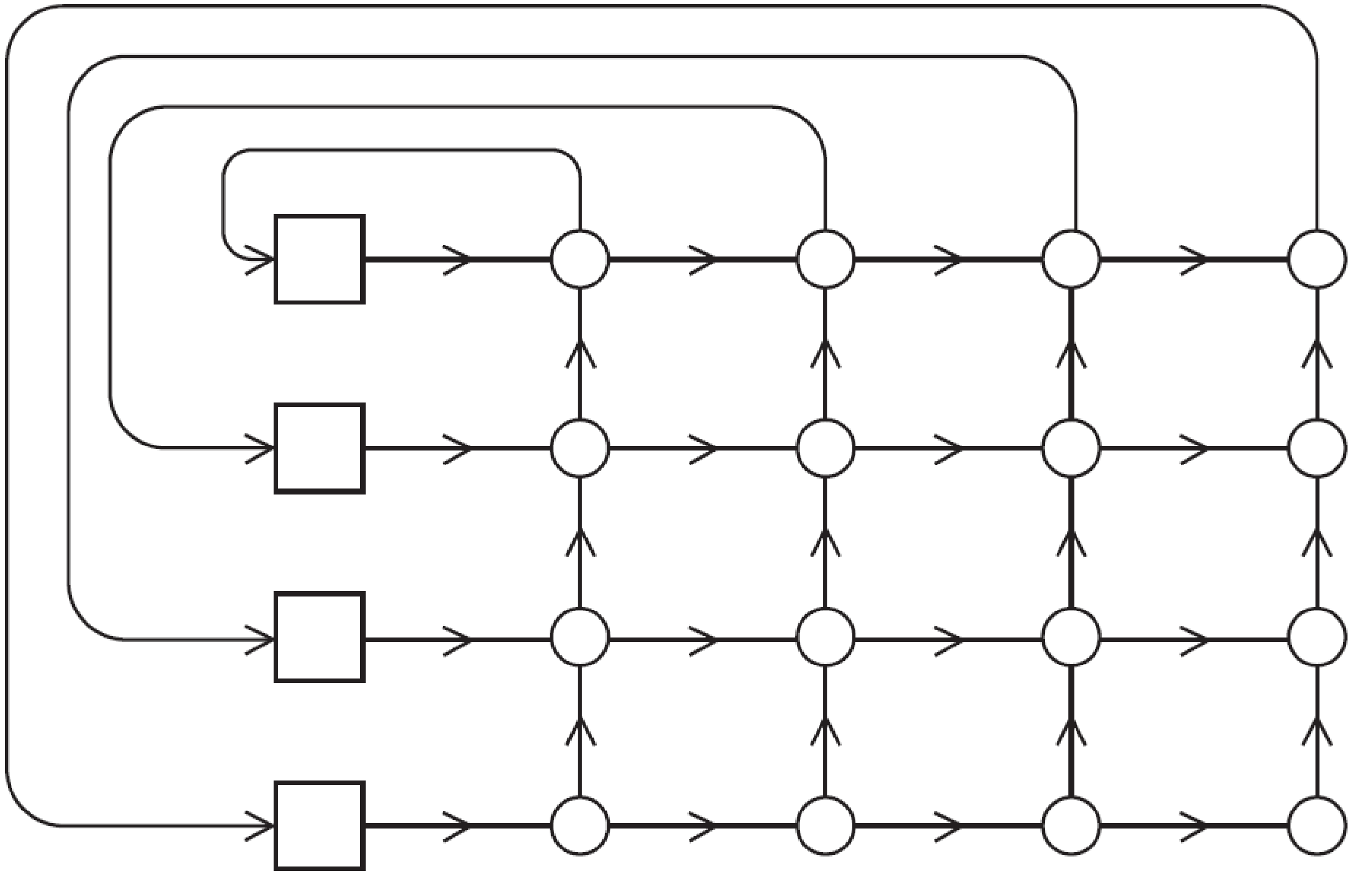
- Simple examples of indirect networks:
  - Crossbar
  - Omega network

- Often shown with unidirectional links and a collection of processors, each of which has an outgoing and an incoming link, and a switching network.

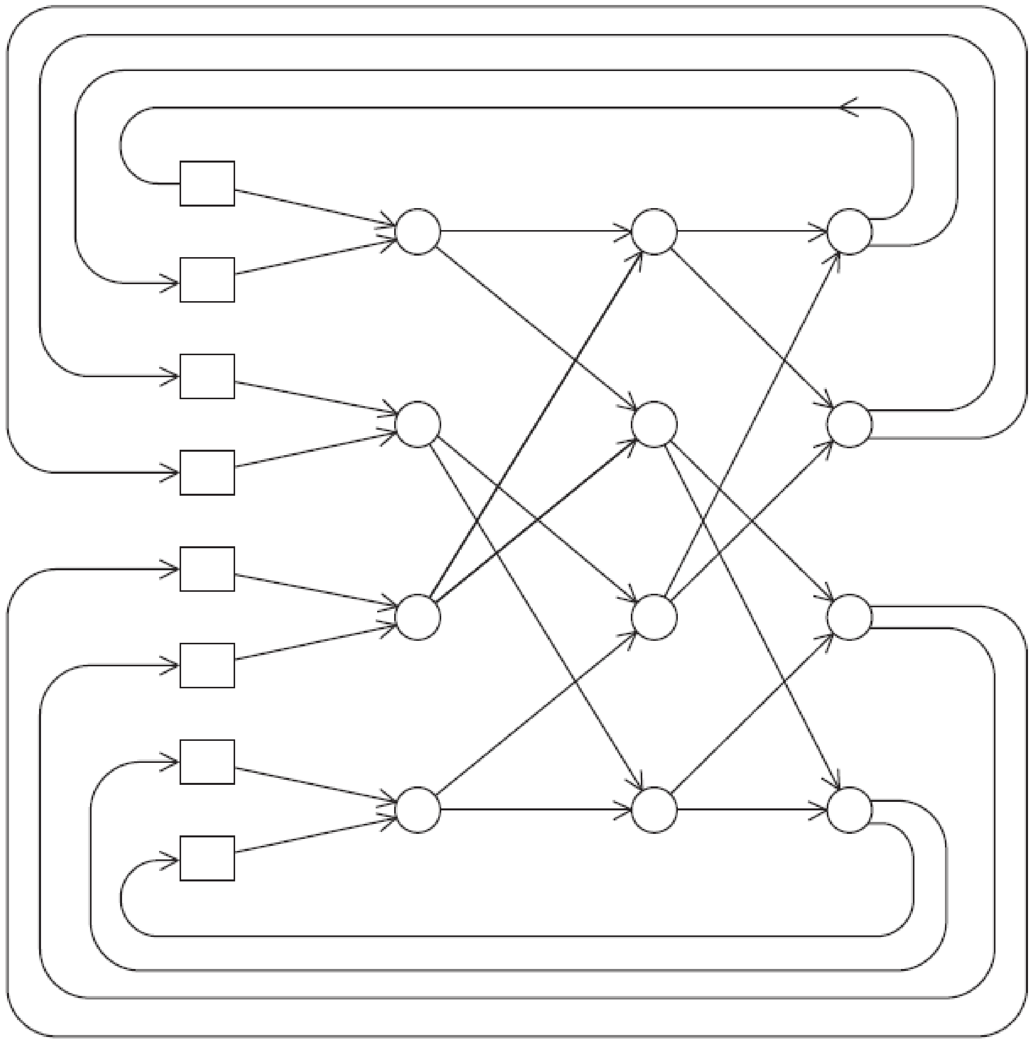
A generic indirect network



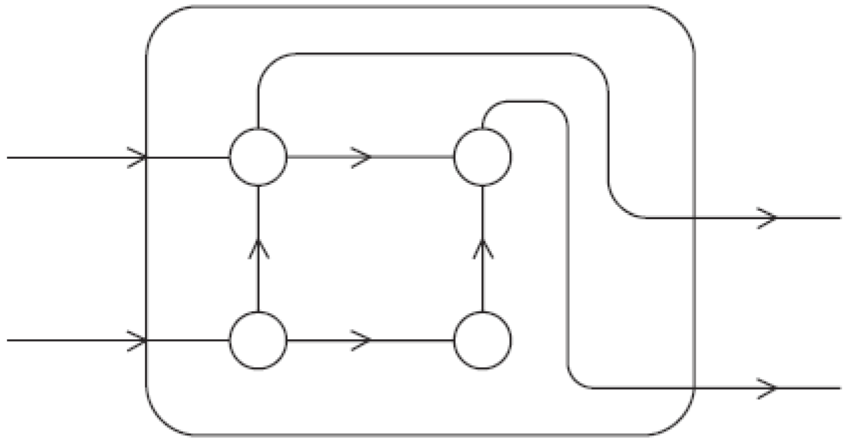
Crossbar interconnect for distributed memory



An omega network



A switch in an omega network



More definitions

- About transmitting data from a source to a destination:
- Latency
  - The time that elapses between the source's beginning to transmit the data and the destination's starting to receive the first byte.
- Bandwidth
  - The rate at which the destination receives data after it has started to receive the first byte.

Message transmission time = L + N / B

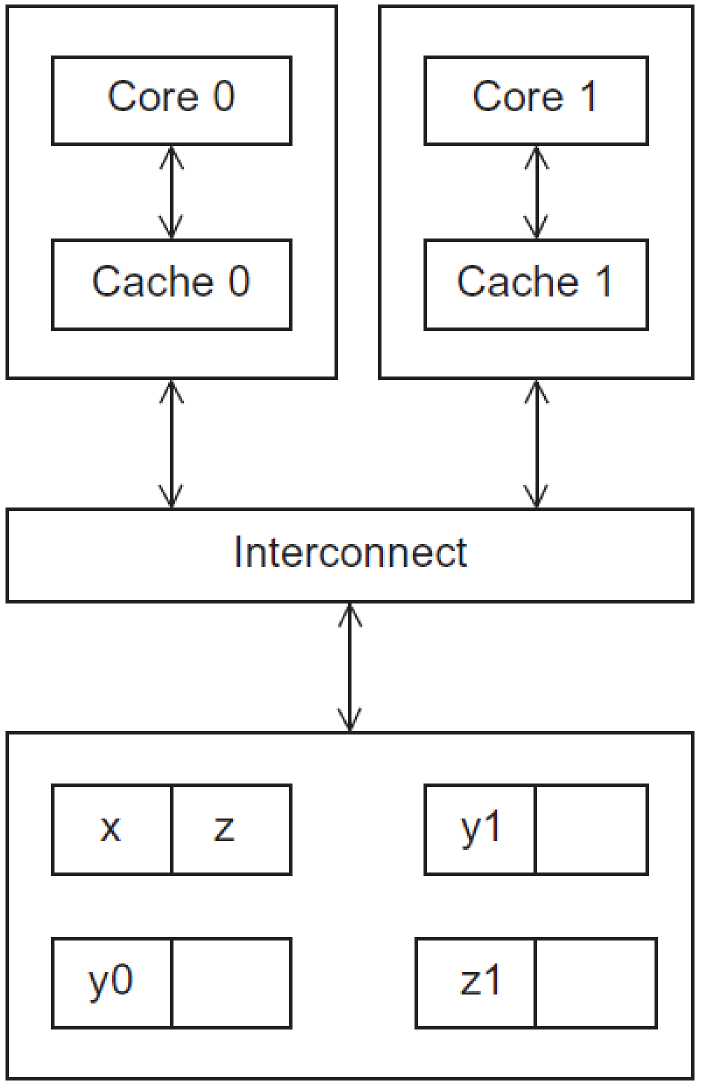
latency (seconds)

length of message (bytes)

bandwidth (bytes per second)

Cache coherence

- Programmers have no control over caches and when they get updated.



A shared memory system with two cores and two caches

Cache coherence

y0 privately owned by Core 0

y1 and z1 privately owned by Core 1

x = 2; /* shared variable */

Time	Core 0	Core 1
0	`y0 = x;`	`y1 = 3*x;`
1	`x = 7;`	Statement(s) not involving x
2	Statement(s) not involving x	`z1 = 4*x;`

y0 eventually ends up = 2

y1 eventually ends up = 6

z1 = ???

Snooping Cache Coherence

- The cores share a bus .
- Any signal transmitted on the bus can be “seen” by all cores connected to the bus.
- When core 0 updates the copy of x stored in its cache it also broadcasts this information across the bus.
- If core 1 is “snooping” the bus, it will see that x has been updated and it can mark its copy of x as invalid.

Directory Based Cache Coherence

- Uses a data structure called a directory that stores the status of each cache line.

- When a variable is updated, the directory is consulted, and the cache controllers of the cores that have that variable's cache line in their caches are invalidated.