Parallel and Distributed Computing
CS3006 (BCS-6C/6D)
Lecture 04

Instructor: Dr. Syed Mohammad Irteza

Assistant Professor, Department of Computer Science, FAST

02 February, 2023

Previous Lecture

- Multi-processor vs. multi-computer
  - Centralized multi-processor (UMA)
  - Distributed multi-processor (NUMA)
  - Asymmetric and symmetric multi-computers
- Network of workstations vs. cluster
- Flynn’s Taxonomy:
  - SISD, SIMD
  - MISD, MIMD

Reading Assignment

- Cache Coherence and Snooping
- Branch prediction and issues while pipelining the problem

Assigned reading pointers:

- Cache Coherence:
  - When we are in a distributed environment, each CPU's cache needs to be consistent (continuously needs to be updated for current values), which is known as cache coherence.
- Snooping:
  - Snoopy protocols achieve data consistency between the cache memory and the shared memory through a bus-based memory system. Write-invalidate and write-update policies are used for maintaining cache consistency.
- Branch Prediction:
  - Branch prediction is a technique used in CPU design that attempts to guess the outcome of a conditional operation and prepare for the most likely result.

Parallel and Distributed Systems

- Parallel and distributed computing is going to be more and more important
  - Dual and quad core processors are very common
  - Up to six and eight cores for each CPU
  - Multithreading is growing
- Hardware structure or architecture is important for understanding how much speed up is possible beyond a single CPU
- Also, the capability of compilers to generate efficient code is very important
- It is always difficult to distinguish between HW and SW influences

Parallel Computing vs. Distributed Computing

Parallel Computing	Distributed Computing
Shared memory	Distributed memory
Multiprocessors	Clusters and networks of workstations
Processors share logical address spaces	Processors do not share address space
Processors share physical memory	No sharing of physical memory
Also referred to the study of parallel algorithms	Study of theoretical distributed algorithms or neural networks

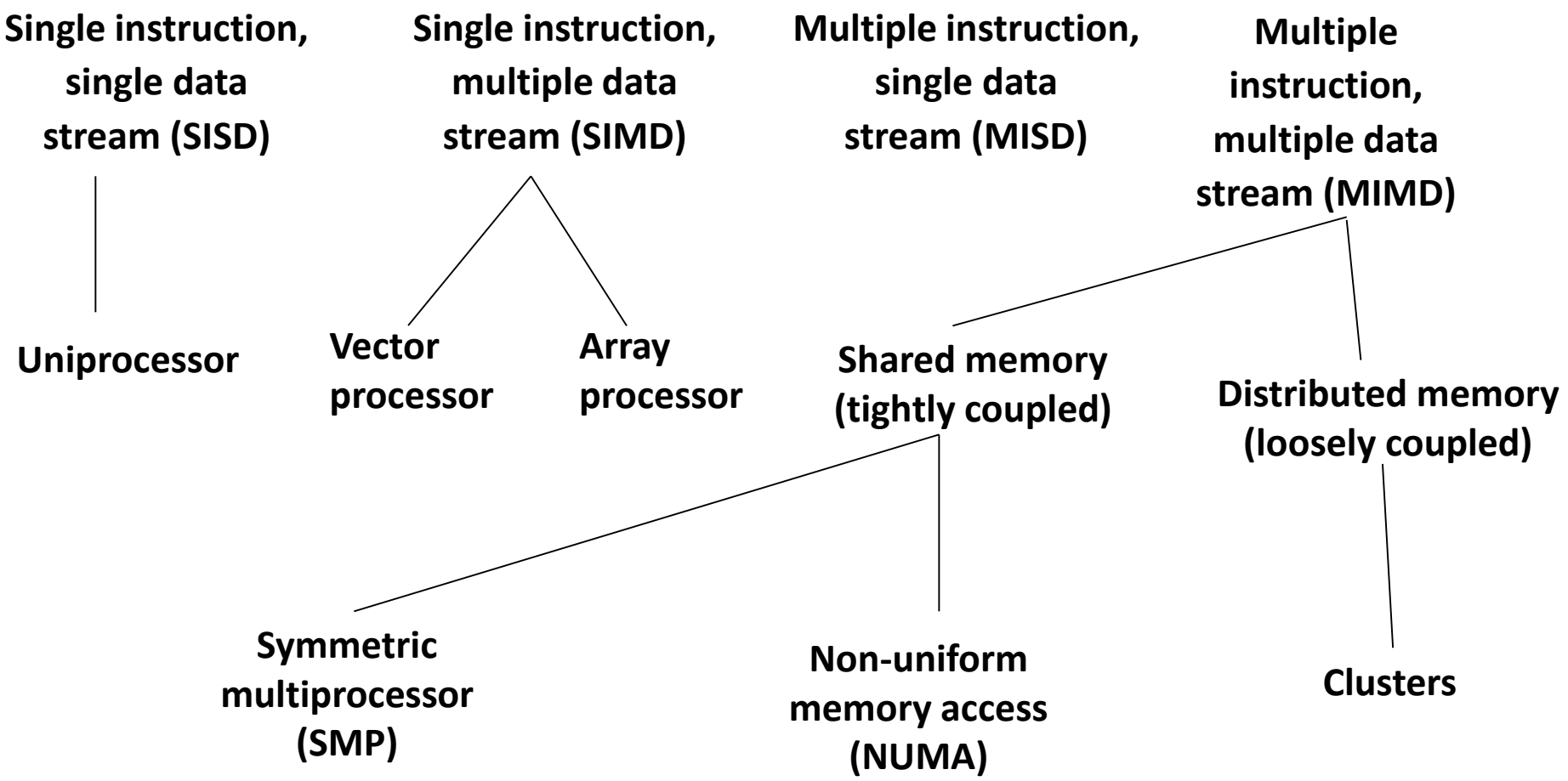
Amdahl's Law

- Scaling falls off as the number of processors increases due to lock (barriers for synchronizing masters and slaves) and memory collisions
- It is very difficult to compute (1-F)
  - serialized regions not only in our code, also in kernel and in HW
  - profiling is very important
- The private data cache of multiprocessor systems has to be kept consistent

Flynn’s Taxonomy

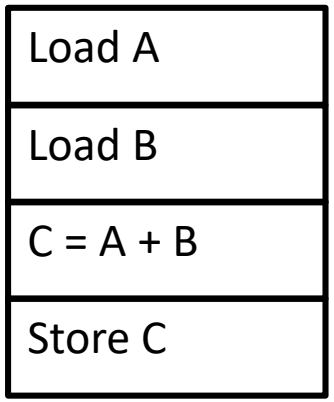
- SISD: Single instruction stream, single data stream
- SIMD: Single instruction stream, multiple data stream
- MISD: Multiple instruction stream , single data stream
- MIMD: Multiple instruction stream, multiple data stream

Processor Organizations



SISD

- A serial (non-parallel computer)
- Single instruction: one instruction per cycle
- Single data: only one data stream per cycle
- Easy and deterministic execution



Examples:

- Single CPU workstations
- Most workstations from HP, IBM and SGI are SISD machines

SISD (continued)

- Performance of a processor can be measured with:

  MIPS rate = f x IPC (instructions per cycle)
- How to increase performance:
  - increasing clock frequency
  - increasing number of instructions completed during a processor cycle (multiple pipelines in a superscalar architecture and/or out of order execution)
  - multithreading

SISD (continued)

Implicit multithreading

- concurrent execution of multiple threads extracted from a single sequential program
- Managed by processor hardware
- Improve individual application performance

Explicit multithreading

- concurrent execution of instructions from different explicit threads, either by interleaving instructions from different threads or by parallel execution on parallel pipelines

SISD – Explicit Multithreading

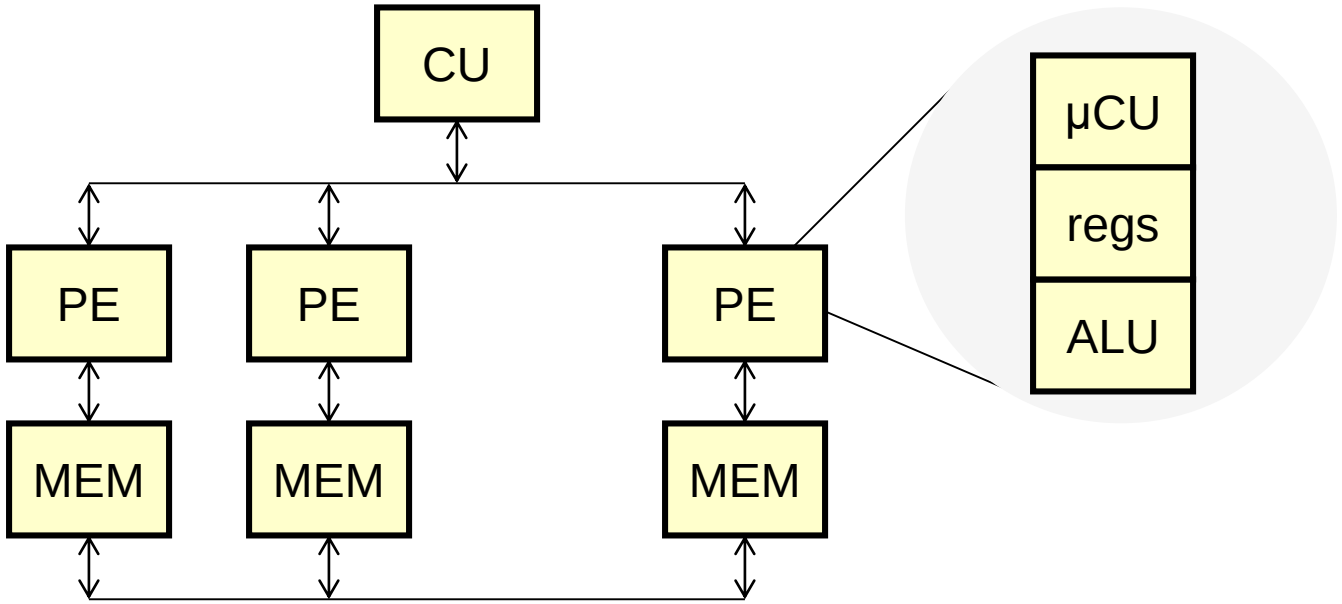
- Four approaches for explicit multithreading
  - Interleaved multithreading (fine-grained): switching can be at each clock cycle. In case of few active threads, performance degrades
  - Blocked multithreading (coarse-grained): events like cache miss produce switch
  - Simultaneous multithreading (SMT): execution units of a superscalar processor receive instructions from multiple threads
  - Chip multiprocessing: e.g. dual core (not SISD)
- Architectures like IA-64 Very Long Instruction Word (VLIW) allow multiple instructions (to be executed in parallel) in a single word

Intel's Hyper-threading Technology

- A single physical processor appears as two logical processors by applying two-threaded SMT approach
- Each logical processor maintains a complete set of architecture state (general-purpose registers, control registers,…)
- Logical processors share nearly all other resources such as caches, execution units, branch predictors, control logic and buses
- Partitioned resources are recombined when only one thread is active
- Add less than 5% to the relative chip size
- Improve performance by 16% to 28%

SMT: https://en.wikipedia.org/wiki/Simultaneous_multithreading

SIMD



- Homogeneous processing units
- Single instruction: All processor units execute the same instruction at any given time
- Multiple data: Each processing unit can operate on different data set

SIMD (continued)

- Each processing element has an associated data memory, so that each instruction is executed on a different set of data by the different processors
- Used by vector and array processors
- Vector processors act on array of similar data (only when executing in vector mode) and in this case they are several times faster than when executing in scalar mode
  - Example is NEC SX-8B

SIMD – Example

- A good example is the processing of pixels on screen
- A sequential processor would examine each pixel one at a time and apply the processing instruction
- An array or vector processor can process all the elements of an array simultaneously
- Game consoles and graphic cards make heavy use of such processors to shift those pixels
- Such designs are usually dedicated to a particular application and not commonly marketed for general purpose computing

SIMD – Example

MISD

- A single data stream is transmitted to a set of processors, each of which executes a different instruction sequence
- Each processing unit operates on the data independently via independent instruction stream
- This structure is not commercially implemented
- An example of use could be multiple cryptography algorithms attempting to crack a coded message

MIMD

- Multiple instruction: Every processor may execute a different instruction stream
- Multiple data: Every processor may work with a different data stream

- Examples
  - most of the current supercomputers
  - Grids
  - networked parallel computers
  - multiprocessor SMP computer

MIMD (continued)

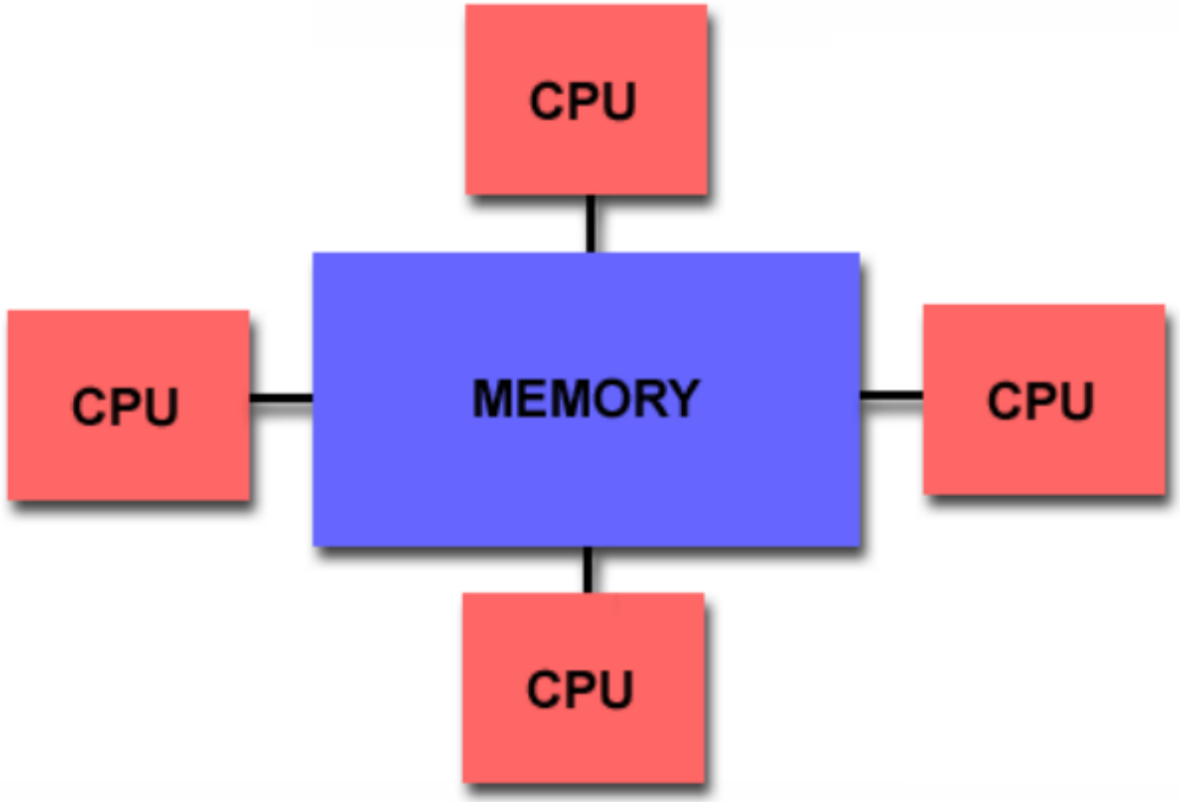
- MIMD systems are mainly:
  - Shared Memory (SM) systems:

    multiple CPUs all of which share the same address space (there is only one memory).
  - Distributed memory (DM) systems:

    each CPU has its own associated memory. CPUs are connected by some network (clusters).

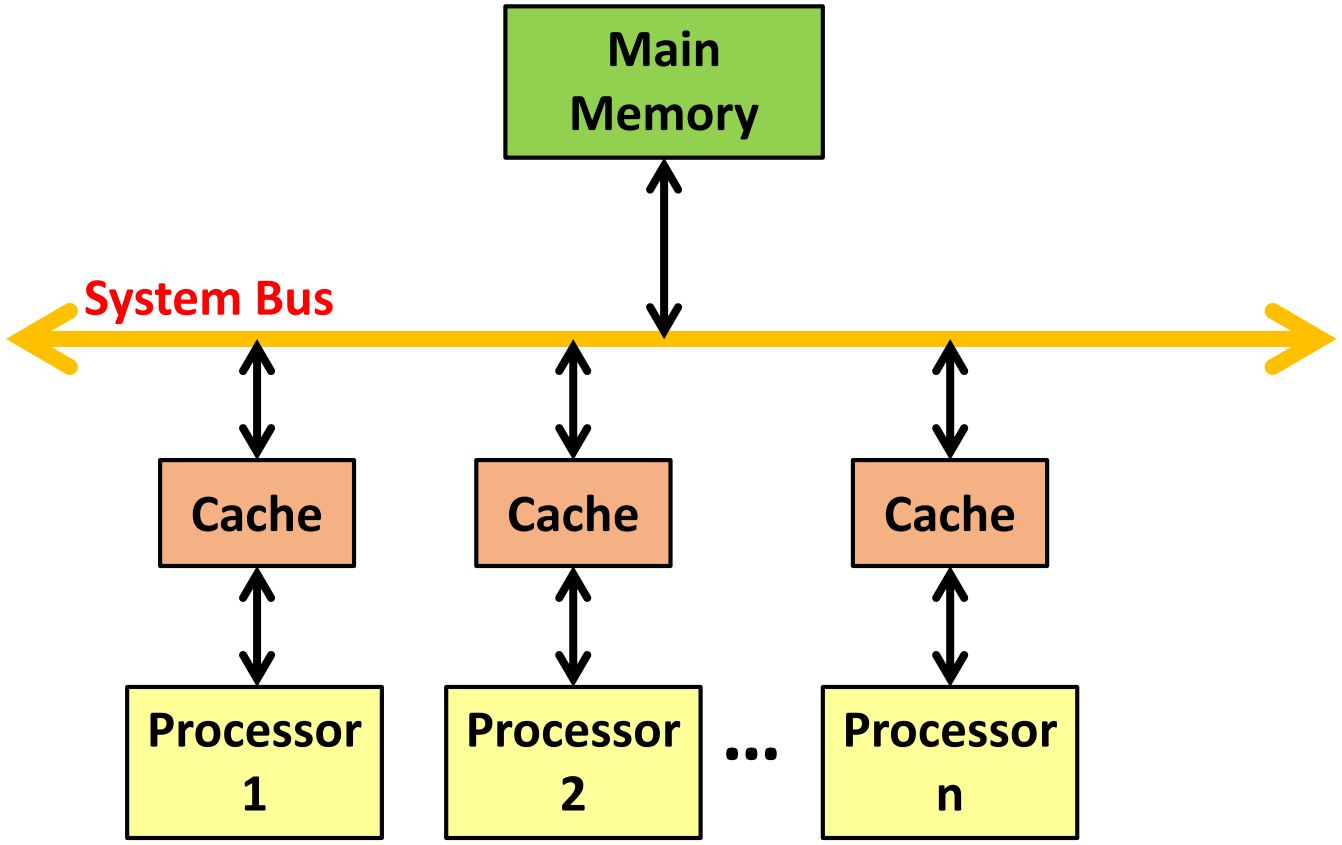
MIMD - Shared Memory

- All processors have access to all memory as a global address space

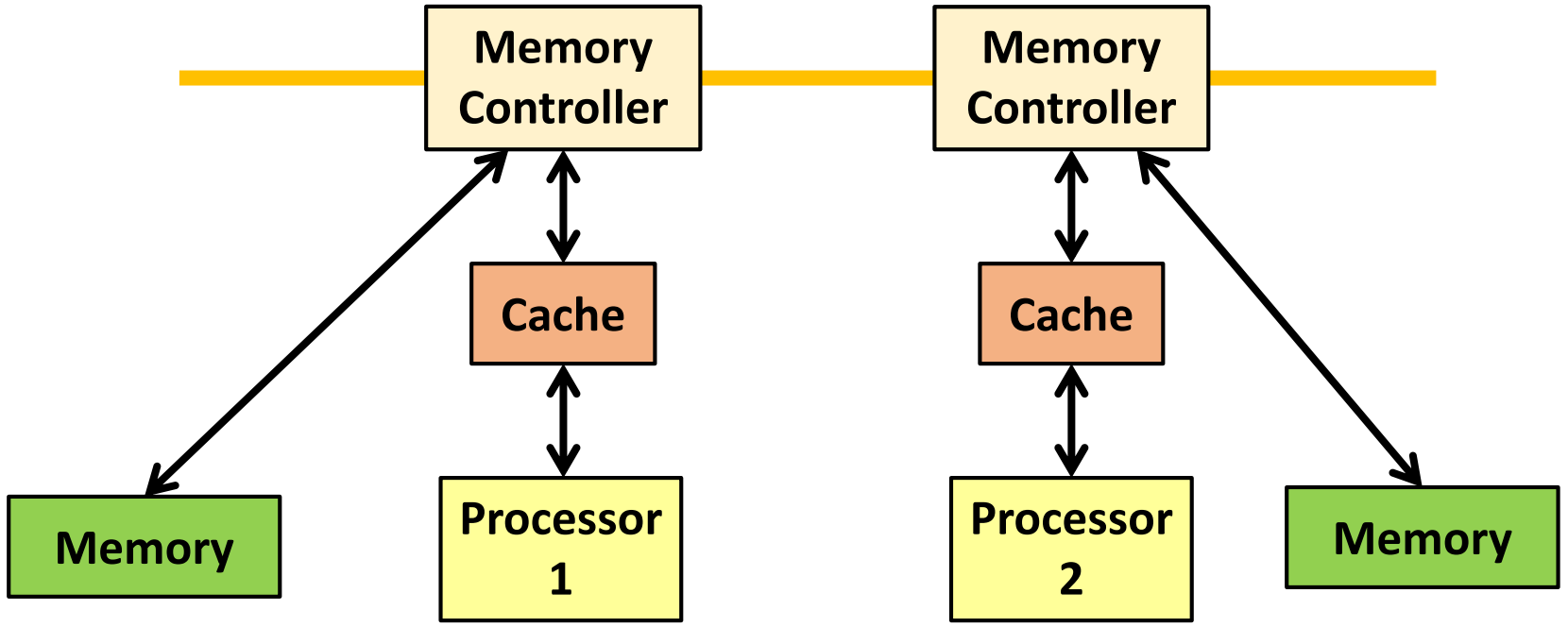


Uniform Memory Access (UMA)

- Mostly represented by Symmetric Multiprocessor (SMP) machines



Non-Uniform Memory Access (NUMA)

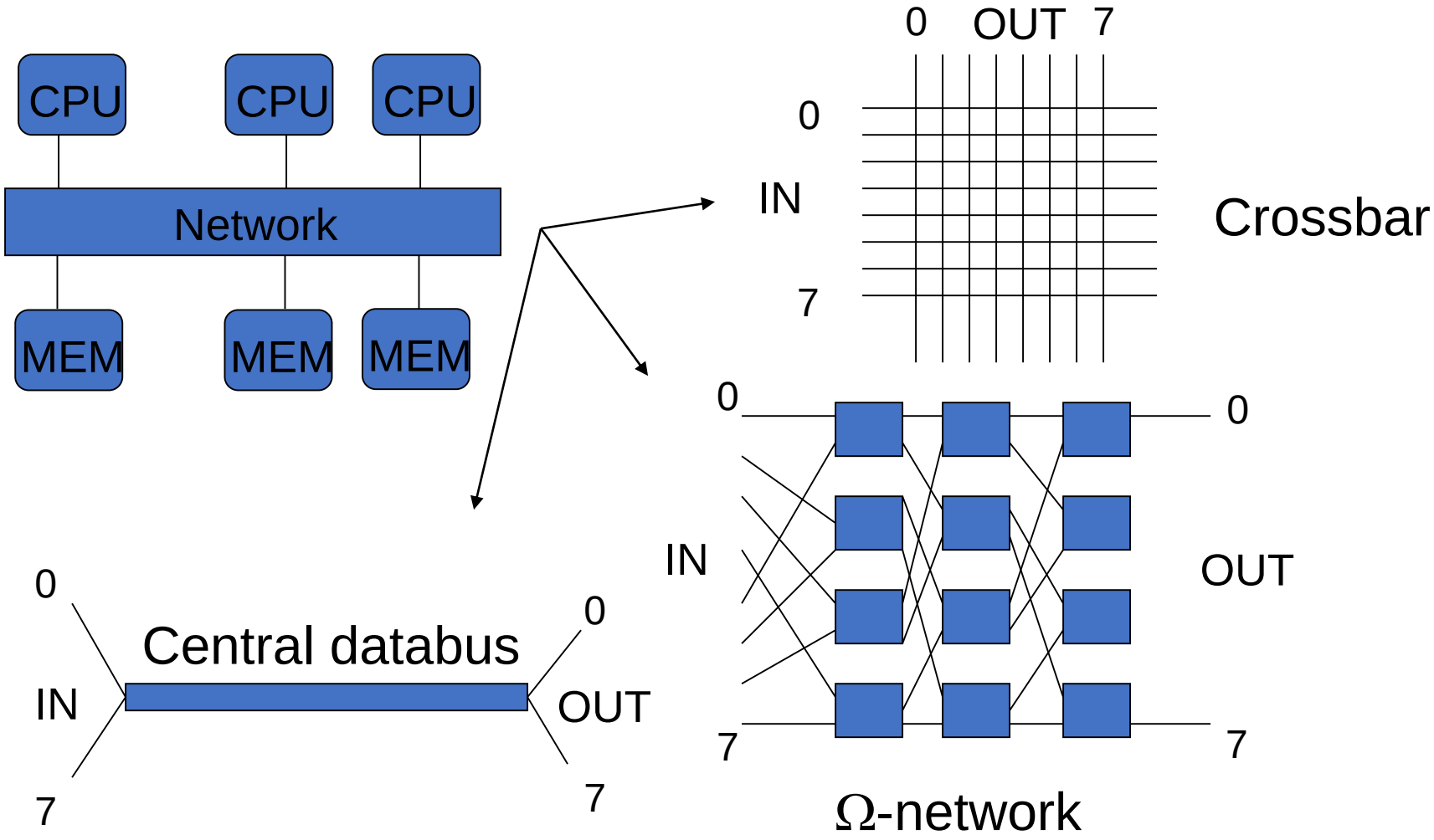


Shared Memory Interconnection Network

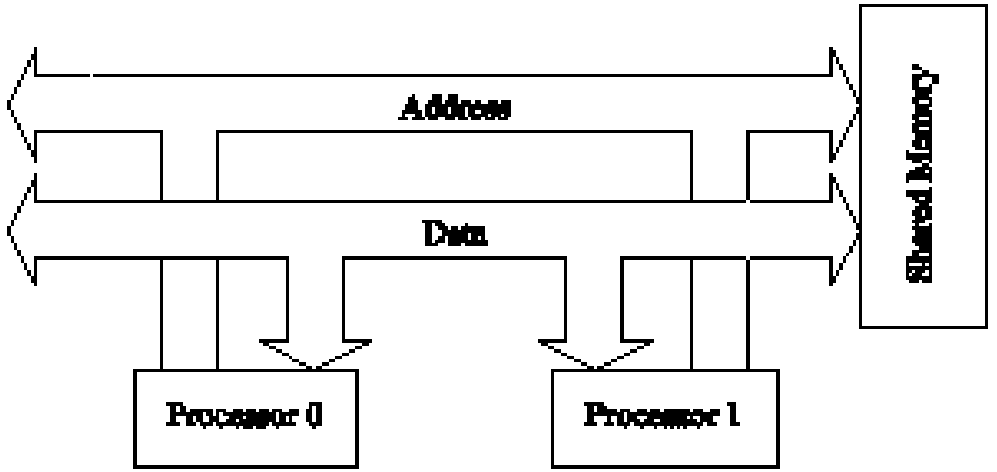
- The main problem → how to do interconnections of the CPUs, with each other and to the memory

- There are three main network topologies available:
  - Crossbar (n^2 connections - datapath without sharing)
  - Ω-network ($n \log_2 n$ connections - $\log_2 n$ switching stages and shared on a path)
  - Central databus (1 connections - n shared)

Shared Memory Interconnection (2)

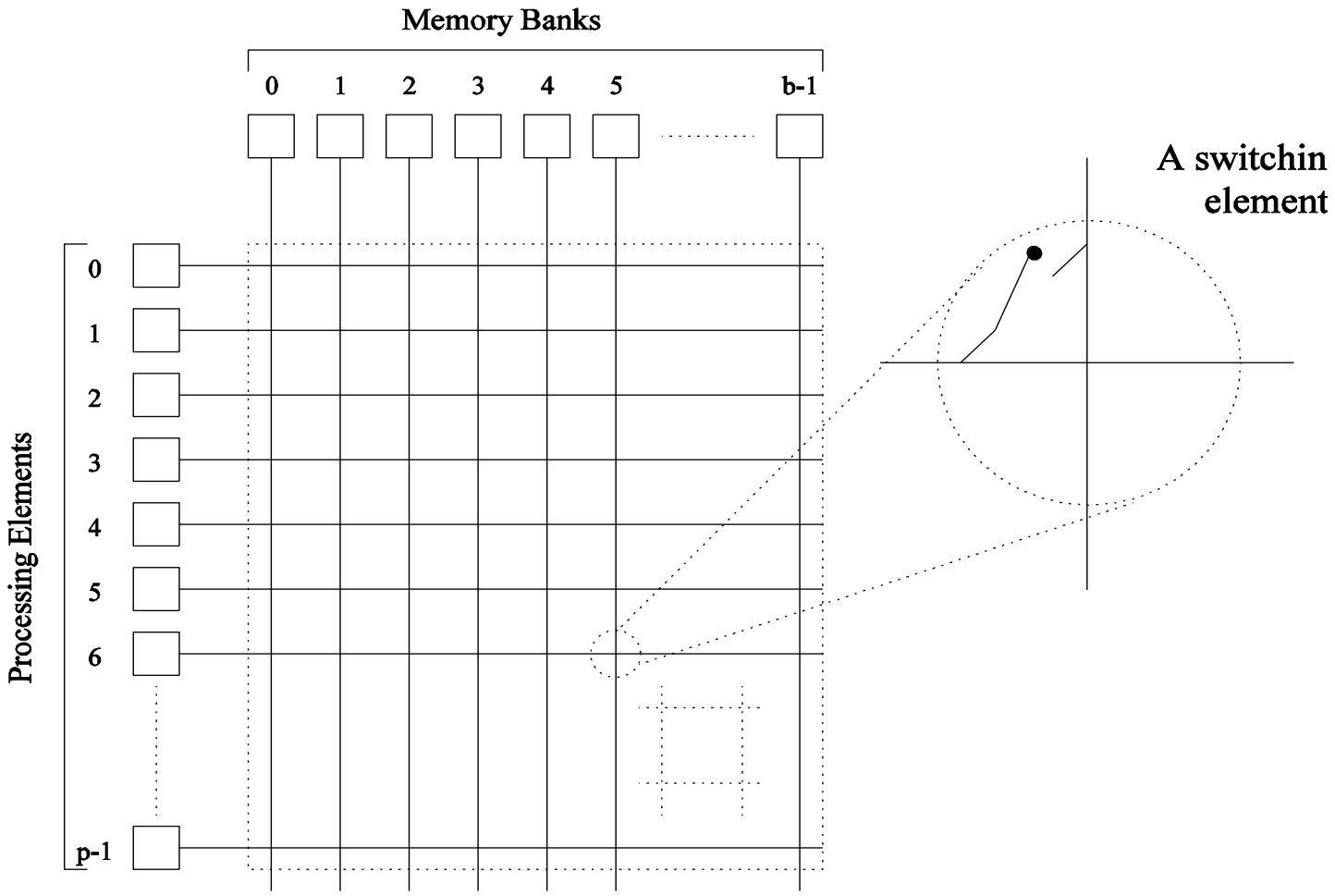


Network Topologies: Buses



- 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.
- 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: Crossbar



Network Topologies: Crossbar

- Uses an $p \times m$ grid of switches to connect p inputs to m outputs in a *non-blocking manner*
- The cost of a crossbar of p processors grows as $O(p^2)$
- This is generally difficult to scale for *large values of p*
- Examples of machines that employ crossbars include the Sun Ultra HPC 10000 and the Fujitsu VPP500

- Crossbars have excellent performance scalability but poor cost scalability
- Buses have excellent cost scalability, but poor performance scalability

Network Topologies: Multistage Omega Network

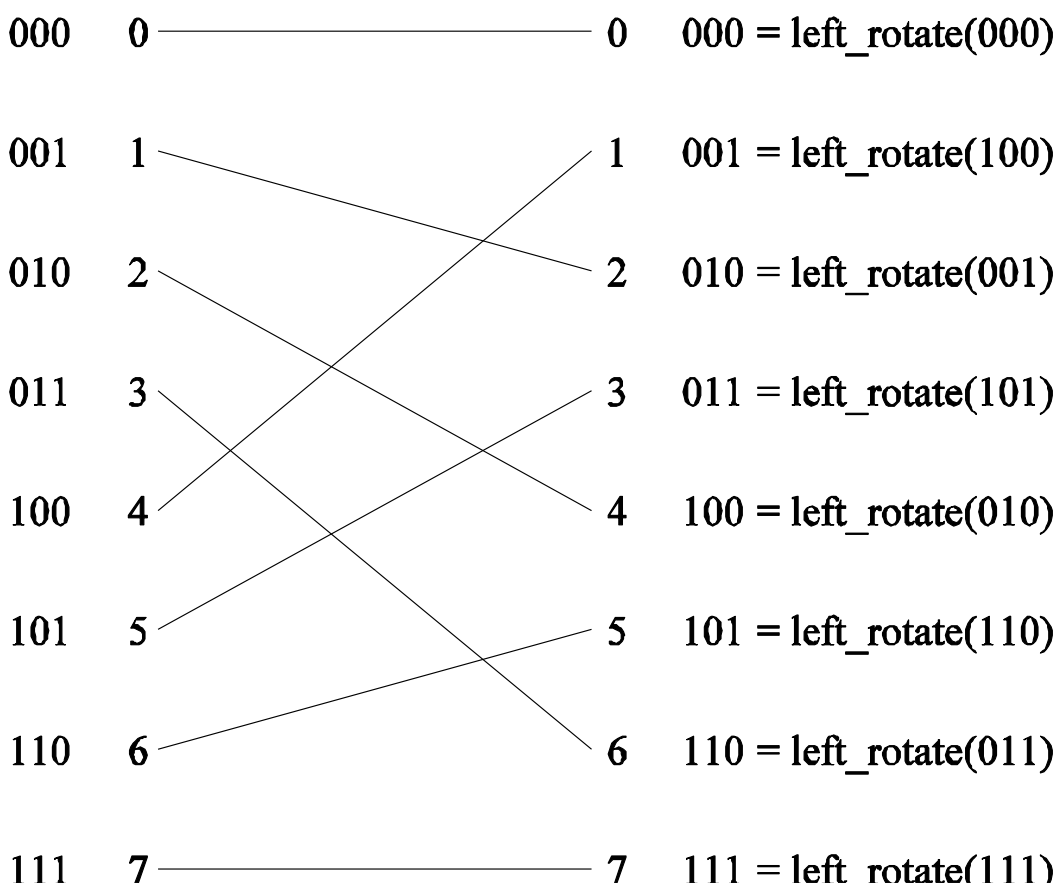
- One of the most commonly used multistage interconnects is the Omega network.
- This network consists of *log p* stages, where *p* is the number of inputs/outputs.

- At each stage, input *i* is connected j

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

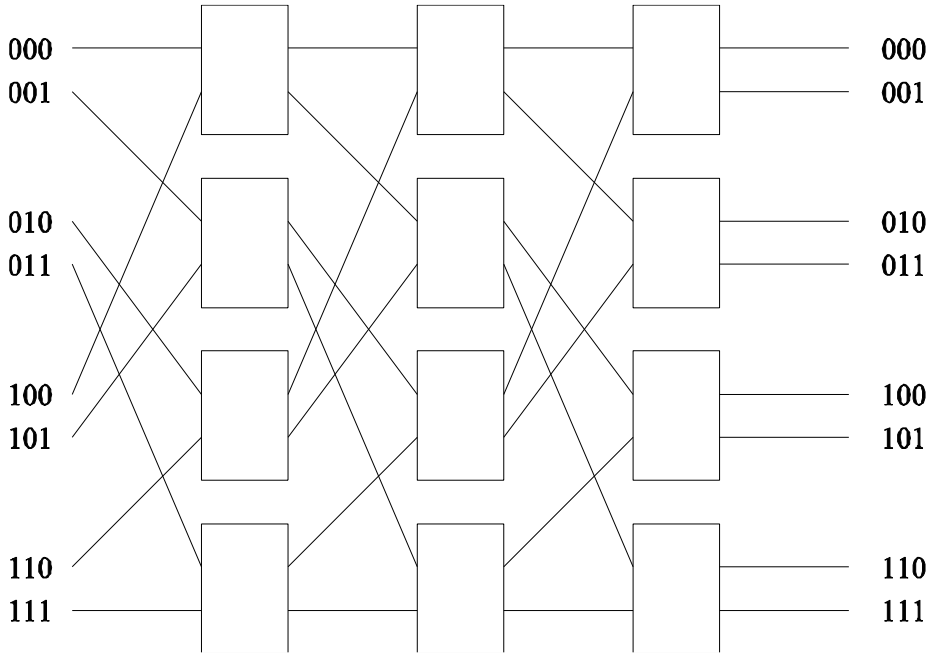
Network Topologies: Multistage Omega Network

- Each stage of the Omega network implements a perfect shuffle as follows:



8 x 8 Omega network

- A complete Omega network with the perfect shuffle interconnects and switches can now be illustrated:



- An omega network has $p/2 \times \log p$ switching nodes, and the cost of such a network grows as $(p \log p)$

Network Topologies: Multistage Omega Network

- Machines like IBM eServer p575 and SGI Altix 4000 use Ω-network
- Ω-network is much more interesting for a large number of processors
- Problem: the switches have to be fast enough, and also the width of the links is important. 16 bit parallel is better than serial links
- Multiprocessor vector-processors use crossbars instead (because at most only 32 processors)
- Synchronization is usually performed with special communication registers (CPU to CPU); if there is little synchronization shared memory is admitted

Sources

- Slides of Dr. Haroon Mahmood & Dr. Rana Asif Rahman, FAST