9.4 Parallel and Multiprocessor Architectures

- Single processor limitations
  - Heat and electromagnetic interference limit chip transistor density.
  - Speed of light – signals cannot be transmitted faster than the speed of light.
  - Economics – It may be possible to incrementally improve processor performance but who will pay for it.
- Single processor limitations are a motivation for parallelism involving multiple processors
- Parallel processor limitations
  - Given n processors running in parallel, perfect speedup would imply that computational job could complete in $\frac{1}{n}$ time, leading to an n-fold increase in power (or a run-time decrease by a factor of n.)
  - At best, parallel processing results in linear speedup in the number of processors.
  - Perfect speedup is not possible by Amdahl's law because the slower component will dominate.
  - Inherent serialization of work prohibits perfect parallelization.
  - The greater the sequential processing, the less cost-effective it is to employ a multiprocessing parallel architecture.
  - Most algorithms and existing code are inherently sequential, making the application of parallel processing limited to a narrow group of specially programmed applications.
  -

9.4.1 Superscalar and VLIW

- Superscalar and Very Long Instruction Word (VLIW) architectures exhibit **instruction-level parallelism**.
- Recall that pipelining divides the fetch-decode-execute cycle into stages that each carry out a small part of the process on a set of instructions.
- Ideally, an instruction exits the pipeline during each tick of the clock.
- *Superpipelining* occurs when a pipeline has stages that require less than half a clock cycle to complete.
- The pipeline is equipped with a separate clock running at a frequency that is at least double that of the main system clock.
- Superpipelining is only one aspect of superscalar design.
- Superscalar architectures include multiple execution units such as specialized integer and floating-point adders and multipliers.
- A critical component of this architecture is the *instruction fetch unit*, which can simultaneously retrieve several instructions from memory.
- A *decoding unit* determines which of these instructions can be executed in parallel and combines them accordingly.
- This architecture also requires compilers that make optimum use of the hardware.
- Very long instruction word (VLIW) architectures differ from superscalar architectures because the VLIW compiler, instead of a hardware decoding unit, packs independent instructions into one long instruction that is sent down the pipeline to the execution units.
- One could argue that this is the best approach because the compiler can better identify instruction dependencies.

- However, compilers tend to be conservative and cannot have a view of the run time code.

9.4.2 Vector Processors

- Vector computers are processors that operate on entire vectors or matrices at once.
  - These systems are often called supercomputers.
- Vector computers are highly pipelined so that arithmetic instructions can be overlapped.
- Vector processors can be categorized according to how operands are accessed.
  - **Register-register** vector processors require all operands to be in registers.
  - **Memory-memory** vector processors allow operands to be sent from memory directly to the arithmetic units.
- A disadvantage of register-register vector computers is that large vectors must be broken into fixed-length segments so they will fit into the register sets.
- Memory-memory vector computers have a longer startup time until the pipeline becomes full.
- In general, vector machines are efficient because there are fewer instructions to fetch, and corresponding pairs of values can be prefetched because the processor knows it will have a continuous stream of data.

9.4.3 Interconnection Networks

- MIMD systems can communicate through shared memory or through an interconnection network.
- Interconnection networks are often classified according to their topology, routing strategy, and switching technique.
- Of these, the topology is a major determining factor in the overhead cost of message passing.
- Message passing takes time owing to network latency and incurs overhead in the processors.
- Interconnection networks can be either static or dynamic.
- Processor-to-memory connections usually employ dynamic interconnections. These can be blocking or nonblocking.
  - Nonblocking interconnections allow connections to occur simultaneously.
- Processor-to-processor message-passing interconnections are usually static, and can employ any of several different topologies, as shown on the following slide.

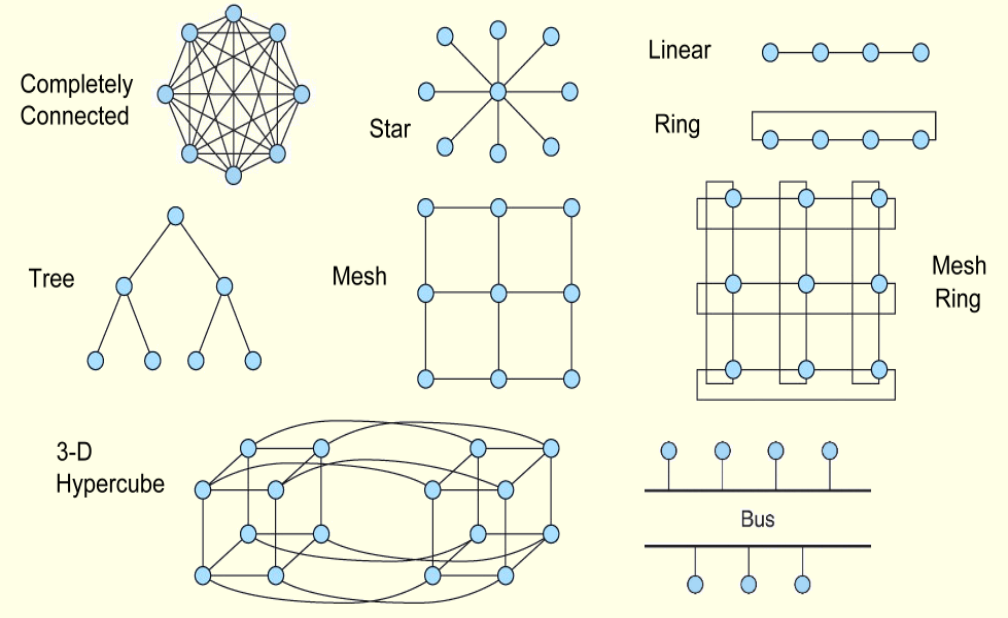

Figure 9.3 Static Network Topologies
a) Completely Connected
b) Star
c) Linear or Ring
d) Mesh and Mesh Ring
e) Tree
f) 3-D Hypercube

Figure 9.4 A Bus-Based Network

- Dynamic routing is achieved through switching networks that consist of crossbar switches or 2 × 2 switches.

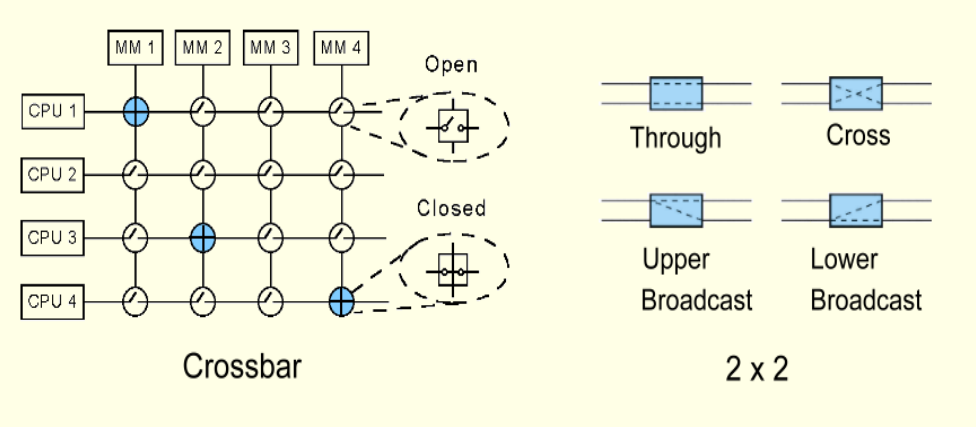

Figure 9.5 A Crossbar Network

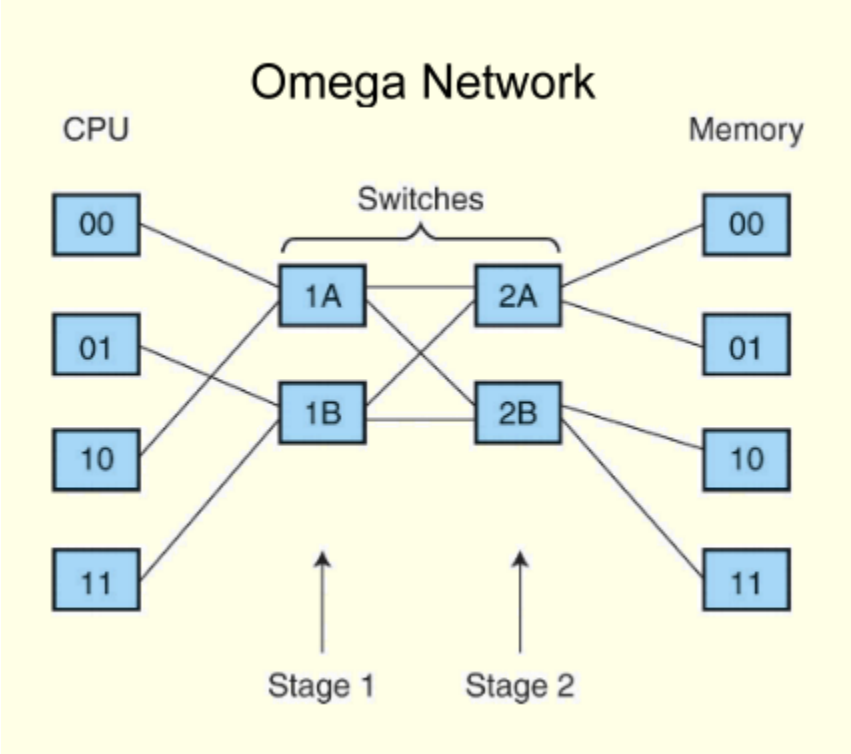

Figure 9.7 A Two-Stage Omega Network

- Multistage interconnection (or shuffle) networks are the most advanced class of switching networks.
- They can be used in loosely-coupled distributed systems, or in tightly-coupled processor-to-memory configurations.
- There are advantages and disadvantages to each switching approach.
- Bus-based networks, while economical, can be bottlenecks. Parallel buses can alleviate bottlenecks, but are costly.
- Crossbar networks are nonblocking, but require n^2 switches to connect n entities.
- Omega networks are blocking networks, but exhibit less contention than bus-based networks. They are somewhat more economical than crossbar networks, *n* nodes needing $\log_2 n$ stages with $n/2$ switches per stage.

Property	**Bus**	**Crossbar**	**Multistage**
Speed	Low – Bad	High – Good	Moderate
Cost	Low – Good	High – Bad	Moderate
Reliability	Low – Bad	High – Good	High
Configurability	High – Good	Low – Bad	Moderate
Complexity	Low - Good	High - Bad	Moderate

Table 9.2 Properties of the Various Interconnection Networks

9.4.4 Shared Memory Multiprocessors

- Tightly-coupled multiprocessor systems use the same memory. They are also referred to as shared memory multiprocessors.
- The processors do not necessarily have to share the same block of physical memory:
- Each processor can have its own memory, but it must share it with the other processors.
- Configurations such as these are called *distributed shared memory multiprocessors*.

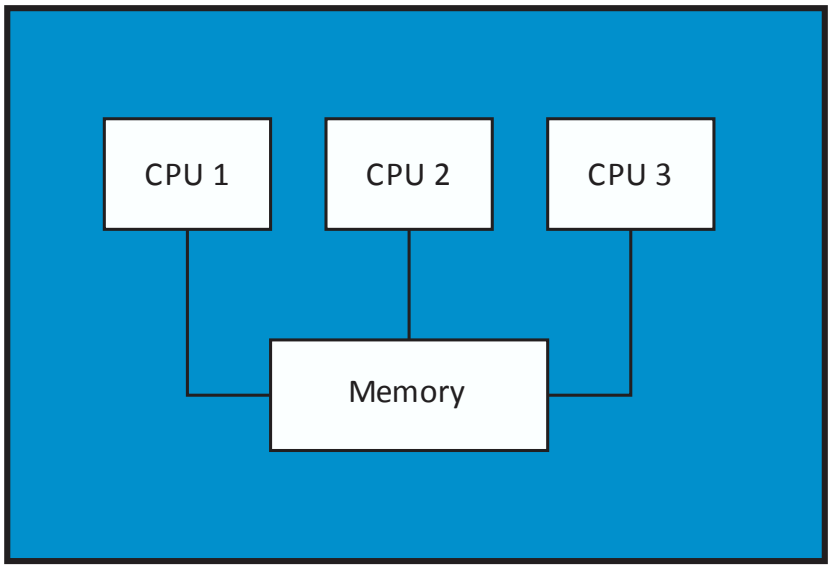

Figure 9.8 a) Global Shared Memory

- Tightly-coupled multiprocessor systems use the same memory. They are also referred to as shared memory multiprocessors.

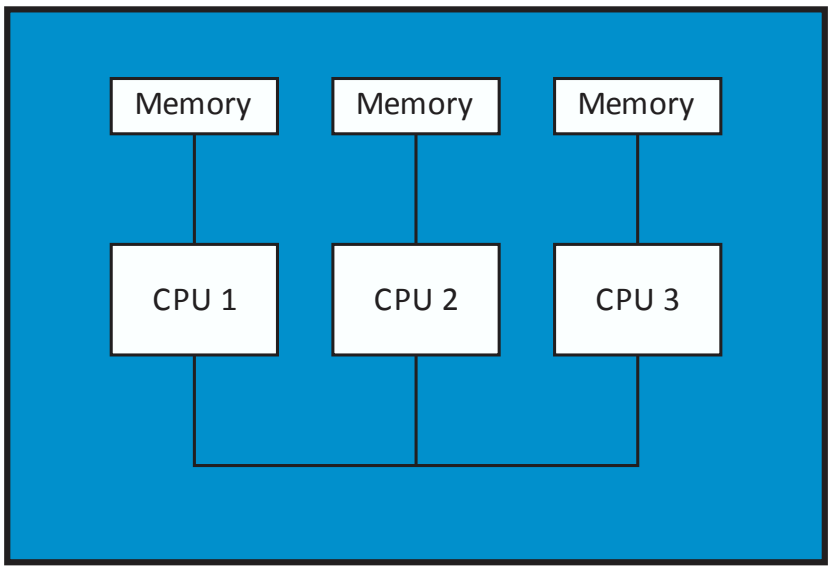

Figure 9.8 b) Distributed Shared Memory

- The processors do not necessarily have to share the same block of physical memory:
- Each processor can have its own memory, but it must share it with the other processors.
- Shared memory MIMD machines can be divided into two categories based upon how they access memory.
- In u*niform memory access* (UMA) systems, all memory accesses take the same amount of time.
- To realize the advantages of a multiprocessor system, the interconnection network must be fast enough to support multiple concurrent accesses to memory, or it will slow down the whole system.
- Thus, the interconnection network limits the number of processors in a UMA system.
- The other category of MIMD machines are the *nonuniform memory access* (NUMA) systems.
- While NUMA machines see memory as one contiguous addressable space, each processor gets its own piece of it.
- Thus, a processor can access its own memory much more quickly than it can access memory that is elsewhere.

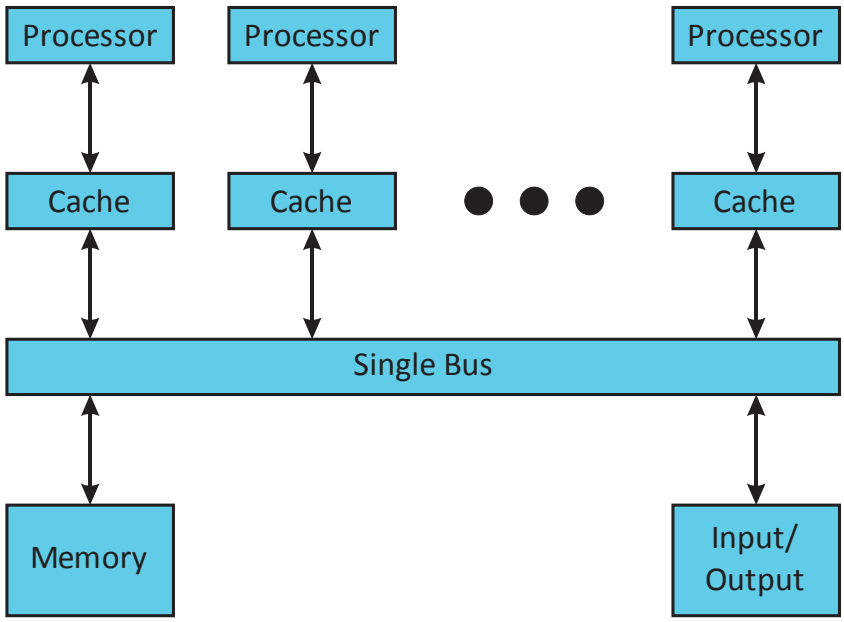

Figure 9.8 c) Global Shared Memory with Separate Cache at Processors

- Not only does each processor have its own memory, it also has its own cache, a configuration that can lead to *cache coherence* problems.
- Cache coherence problems arise when main memory data is changed and the cached image is not. (We say that the cached value is *stale*.)
- To combat this problem, some NUMA machines are equipped with *snoopy cache controllers* that monitor all caches on the systems. These systems are called *cache coherent NUMA* (CC-NUMA) architectures.
- A simpler approach is to ask the processor having the stale value to either void the stale cached value or to update it with the new value.
- When a processor's cached value is updated concurrently with the update to memory, we say that the system uses a *write-through* cache update protocol.
- If the *write-through with update* protocol is used, a message containing the update is broadcast to all processors so that they may update their caches.
- If the *write-through with invalidate* protocol is used, a broadcast asks all processors to invalidate the stale cached value.
- Write-invalidate uses less bandwidth because it uses the network only the first time the data is updated, but retrieval of the fresh data takes longer.
- Write-update creates more message traffic, but all caches are kept current.
- Another approach is the *write-back* protocol that delays an update to memory until the modified cache block must be replaced.
- At replacement time, the processor writing the cached value must obtain exclusive rights to the data. When rights are granted, all other cached copies are invalidated.

9.4.5 Distributed Computing

- Distributed computing is another form of multiprocessing. However, the term *distributed computing* means different things to different people.
- In a sense, all multiprocessor systems are distributed systems because the processing load is distributed among processors that work collaboratively.
- The common understanding is that a distributed system consists of very loosely-coupled processing units.
- Recently, NOW (Network Of Workstation)s have been used as distributed systems to solve large, intractable problems
- For general-use computing, the details of the network and the nature of the multiplatform computing should be transparent to the users of the system.
- Remote procedure calls (RPCs) enable this transparency. RPCs use resources on remote machines by invoking procedures that reside and are executed on the remote machines.
- RPCs are employed by numerous vendors of distributed computing architectures including the Common Object Request Broker Architecture (CORBA) and Java's Remote Method Invocation (RMI).
- Cloud computing is distributed computing to the extreme.
- It provides *services* over the Internet through a collection of loosely-coupled systems.
- In theory, the service consumer has no awareness of the hardware, or even its location.
- Your services and data may even be located on the same physical system as that of your business competitor.
- The hardware might even be located in another country.
- Security concerns are a major inhibiting factor for cloud computing.