Multi-Processor / Parallel Processing

Parallel Processing:

Originally, the computer has been viewed as a sequential machine. Most computer programming languages require the programmer to specify algorithms as sequence of instructions.

Processor executes programs by executing machine instructions in a sequence and one at a time.

Each instruction is executed in a sequence of operations (fetch instruction, fetch operands, perform operation store result.)

It is observed that, at the micro operation level, multiple control signals are generated at the same time.

Instruction pipelining, at least to the extent of overlapping fetch and execute operations, has been around for long time.

By looking into these phenomenon, researcher has look into the matter whether some operations can be performed in parallel or not.

As computer technology has evolved, and as the cost of computer hardware has dropped, computer designers have sought more and more opportunities for parallelism, usual to enhance performance and, in some cases, to increase availability.

The taxonomy first introduced by Flynn is still the most common way of categorizing systems with parallel processing capability. Flynn proposed the following categories of computer system:

- **Single Instruction, Multiple Data (SIMD) system**: A single machine instruction controls the simultaneous execution of a number of processing elements on a lockstep basis. Each processing element has an associated data memory, so that each instruction is executed on a different set of data by the different processors. Vector and array processors fall into this category.

- **Multiple Instruction, Single Data (MISD) system**: A sequence of data is transmitted to a set of processors, each of which executes a different instruction sequence. This structure has never been implemented.
• **Multiple Instruction, Multiple Data (MIMD) system**: A set of processors simultaneously execute different instruction sequences on different data sets. SMPs, clusters, and NUMA systems fit into this category.

With the MIMD organization, the processors are general purpose; each is able to process all of the instructions necessary to perform the appropriate data transformation.

Further MIMD can be subdivided into two main categories:

- **Symmetric multiprocessor (SMP)**: In an SMP, multiple processors share a single memory or a pool of memory by means of a shared bus or other interconnection mechanism. A distinguish feature is that the memory access time to any region of memory is approximately the same for each processor.

- **Nonuniform memory access (NUMA)**: The memory access time to different regions of memory may differ for a NUMA processor.

The design issues relating to SMPs and NUMA are complex, involving issues relating to physical organization, interconnection structures, inter processor communication, operating system design, and application software techniques.

**Symmetric Multiprocessors:**

A symmetric multiprocessor (SMP) can be defined as a standalone computer system with the following characteristic:

1. There are two or more similar processor of comparable capability.
2. These processors share the same main memory and I/O facilities and are interconnected by a bus or other internal connection scheme.
3. All processors share access to I/O devices, either through the same channels or through different channels that provide paths to the same device.
4. All processors can perform the same functions.
5. The system is controlled by an integrated operating system that provides interaction between processors and their programs at the job, task, file and data element levels.

The operating system of a SMP schedules processors or thread across all of the processors. SMP has potential advantages over uniprocessor architecture:

- **Performance**: A system with multiple processors will perform in a better way than one with a single processor of the same type if the task can be organized in such a manner that some portion of the work done can be done in parallel.
• Availability: Since all the processors can perform the same function in a symmetric multiprocessor, the failure of a single processor does not stop the machine. Instead, the system can continue to function at reduce performance level.
• Incremental growth: A user can enhance the performance of a system by adding an additional processor.
• Sealing: Vendors can offer a range of product with different price and performance characteristics based on number of processors configured in the system.

Organization:

The organization of a multiprocessor system is shown in Figure 10.1

![Block diagram of tightly coupled multiprocessors](image)

**Figure 10.1:** Block diagram of tightly coupled multiprocessors

• There are two or more processors. Each processor is self sufficient, including a control unit, ALU, registers and cache.
• Each processor has access to a shared main memory and the I/O devices through an interconnection network.
• The processor can communicate with each other through memory (messages and status information left in common data areas).
• It may also be possible for processors to exchange signal directly.
• The memory is often organized so that multiple simultaneous accesses to separate blocks of memory are possible.
• In some configurations each processor may also have its own private main memory and I/O channels in addition to the shared resources.
The organization of multiprocessor system can be classified as follows:

- Time shared or common bus
- Multiport memory
- Central control unit.

**Time shared Bus:**

Time shared bus is the simplest mechanism for constructing a multiprocessor system. The bus consists of control, address and data lines. The block diagram is shown in Figure 10.2.

![Figure 10.2: Time shared bus](image)

The following features are provided in time-shared bus organization:

- **Addressing:** It must be possible to distinguish modules on the bus to determine the source and destination of data
- **Arbitration:** Any I/O module can temporarily function as “master”. A mechanism is provided to arbitrate competing request for bus control, using some sort of priority scheme.
- **Time shearing:** when one module is controlling the bus, other modules are locked out and if necessary suspend operation until bus access is achieved.

The bus organization has several advantages compared with other approaches:
• Simplicity: This is the simplest approach to multiprocessor organization. The physical interface and the addressing, arbitration and time sharing logic of each processor remain the same as in a single processor system.
• Flexibility: It is generally easy to expand the system by attaching more processor to the bus.
• Reliability: The bus is essentially a passive medium and the failure of any attached device should not cause failure of the whole system.

The main drawback to the bus organization is performance. Thus, the speed of the system is limited by the bus cycle time.

To improve performance, each processor can be equipped with local cache memory.

The use of cache leads to a new problem which is known as cache coherence problem. Each local cache contains an image of a portion of main memory. If a word is altered in one cache, it may invalidate a word in another cache. To prevent this, the other processors must perform an update in its local cache.

Multiport Memory:

The multiport memory approach allows the direct, independent access of main memory modules by each processor and io module. The multiport memory system is shown in Figure 10.3

![Multiport Memory Diagram](mywbut.com)
The multiport memory approach is more complex than the bus approach, requiring a fair amount of logic to be added to the memory system. Logic associated with memory is required for resolving conflict. The method often used to resolve conflicts is to assign permanently designated priorities to each memory port.

**Non-uniform Memory Access (NUMA)**

In NUMA architecture, all processors have access to all parts of main memory using loads and stores. The memory access time of a processor differs depending on which region of main memory is accessed. The last statement is true for all processors; however, for different processors, which memory regions are slower and which are faster differ.

A NUMA system in which cache coherence is maintained among the cache of the various processors is known as cache-cohence NUMA (CC-NUMA)

A typical CC-NUMA organization is shown in the Figure 10.4.

![Figure 10.4: CC- NUMA Organization](image)

There are multiple independent nodes, each of which is, in effect, an SMP organization.
Each node contains multiple processors, each with its own $L1$ and $L2$ caches, plus main memory.

The node is the basic building block of the overall CC NUMA organization.

The nodes are interconnected by means of some communication facility, which could be a switching mechanism, a ring, or some other networking facility.

Each node in the CC-NUMA system includes some main memory.

From the point of view of the processors, there is only a single addressable memory, with each location having a unique system-wide address.

When a processor initiates a memory access, if the requested memory location is not in the processors cache, then the $L2$ cache initiates a fetch operation.

If the desired line is in the local portion of the main memory, the line is fetch across the local bus.

If the desired line is in a remote portion of the main memory, then an automatic request is send out to fetch that line across the interconnection network, deliver it to the local bus, and then deliver it to the requesting cache on that bus.

All of this activity is atomic and transparent to the processors and its cache.

In this configuration, cache coherence is a central concern. For that each node must maintain some sort of directory that gives it an indication of the location of various portion of memory and also cache status information.

**Interconnection Networks:**

In a multiprocessor system, the interconnection network must allow information transfer between any pair of modules in the system. The traffic in the network consists of requests (such as read and write), data transfers, and various commands.
**Single Bus:**

The simplest and most economical means of interconnecting a number of modules is to use a single bus.

Since several modules are connected to the bus and any module can request a data transfer at any time, it is essential to have an efficient bus arbitration scheme.

In a simple mode of operation, the bus is dedicated to a particular source-destination pair for the full duration of the requested transfer. For example, when a processor uses a read request on the bus, it holds the bus until it receives the desired data from the memory module.

Since the memory module needs a certain amount of time to access the data bus, the bus will be idle until the memory is ready to respond with the data.

Then the data is transferred to the processors. When this transfer is completed, the bus can be assigned to handle another request.

A scheme known as the split-transaction protocol makes it possible to use the bus during the idle period to serve another request.

Consider the following method of handling a series of read requests possibly from different processors.

After transferring the address involved in the first request, the bus may be reassigned to transfer the address of the second request; assuming that this request is to a different memory module.

At this point, we have two modules proceeding with read access cycle in parallel.

If neither module has finished with its access, the bus may be reassigned to a third request and so on.

Eventually, the first memory module completes its access cycle and uses the bus to transfer the data to the corresponding source.

As other modules complete their cycles, the bus is needed to transfer their data to the corresponding sources.

The split transaction protocol allows the bus and the available bandwidth to be used more efficiently. The performance improvement achieved with this protocol depends on the relationship between the bus transfer time and the memory access time.

In split-transaction protocol, performance is improved at the cost of increased bus complexity.
There are two reasons why complexity increases:

- Since a memory module needs to know which source initiated a given read request, a source identification tag must be attached to the request.
- Complexity also increases because all modules, not just the processor, must be able to act as bus muster.

The main limitation of a single bus is that the number of modules that can be connected to the bus is not that large. Networks that allow multiple independent transfer operations to proceed in parallel can provide significantly increased data transfer rate.

**Crossbar Network:**

Crossbar switch is a versatile switching network. It is basically a network of switches. Any module $P_i$ can be connected to any other module $P_j$ by closing the appropriate switch. Such networks, where there is a direct link between all pairs of nodes are called fully connected networks.

In a fully connected network, many simultaneous transfers are possible. If $n$ sources need to send data to $n$ distinct destinations then all of these transfers can take place concurrently. Since no transfer is prevented by the lack of a communication path, the crossbar is called a nonblocking switch.

In the Figure 10.5 of crossbar interconnection network, a single switch is shown at each cross point. In actual multiprocessor system, the paths through the crossbar network are much wider.

**Figure 10.5: Crossbar Interconnection Network**
If there are $n$ modules in a network, then the number of cross point is $n^2$ in a network to interconnect $n$ modules. The total number of switches becomes large as $n$ increases.

In a crossbar switch, conflicts occur when two or more concurrent requests are made to the same destination device. These conflicting requests are usually handled on a predetermined priority basis.

The crossbar switch has the potential for the highest bandwidth and system efficiency. However, because of its complexity and cost, it may be cost effective for a large multiprocessor system.

**Multistage Network:**

The bus and crossbar systems use a single stage of switching to provide a path from a source to a destination.

In multistage network, multiple stages of switches are used to setup a path between source and destination.

Such networks are less costly than the crossbar structure, yet they provide a reasonably large number of parallel paths between source and destinations.

In the Figure 10.6, it shows a three-stage network that called a shuffle network that interconnects eight modules.

The term "shuffle" describes the pattern of connections from the outputs of one stage to the inputs of the next stage.

The switchbox in the Figure 10.6 is a $2 \times 2$ switch that can route either input to either output.

If the inputs request distinct outputs, they can both be routed simultaneously in the straight through or crossed pattern.
If both inputs request the same output, only one request can be satisfied. The other one is blocked until the first request finishes using the switch.

![Multistage Shuffle Network](image)

**Figure 10.6: Multistage Shuffle Network**

A network consisting of $s$ stages can be used to interconnect $2^s$ modules. In this case, there is exactly one path through the network from any module $P_i$ to any module $P_j$. Therefore, this network provides full connectivity between sources and destinations.

Many request patterns cannot be satisfied simultaneously. For example, the connection from $P_2$ to $P_7$ cannot be provided at the same time as the connection from $P_3$ to $P_6$.

A multistage network is less expansive to implement than a crossbar network. If $n$ nodes are to be interconnected using this scheme, then we must use $s = \log_2 n$ stages with $n/2$ switches per stage. Since each switch contains four switches, the total number of switches is

$$4 \times \frac{n}{2} \times \log_2 n = 2n \times \log_2 n$$

which, for a large network, is considerably less than the $n^2$ switches needed in a crossbar network.

Multistage networks are less capable of providing concurrent connection than crossbar switches. The connection path between $P_2$ and $P_4$ is indicated by RED lines in the Figure 10.6.