RELATION TO PARALLEL SYSTEMS

***Parallel Processing Systems divide the program into multiple segments and process them simultaneously*.**

The main objective of parallel systems is to improve the processing speed. They are sometimes known as **multiprocessor or multi computers or tightly coupled systems.** They refer to simultaneous use of multiple computer resources that can include a single computer with multiple processors, a number of computers connected by a network to form a parallel processing cluster or a combination of both.

Characteristics of parallel systems

A parallel system may be broadly classified as belonging to one of three types:

1. A *multiprocessor system*

2. A *multicomputer parallel system*

3. *Array processors*

1. A *multiprocessor system*

A *multiprocessor system* is a parallel system in which the multiple processors have *direct access to shared memory* which forms a common address space. The architecture is shown in Figure (a). Such processors usually do not have a common clock.

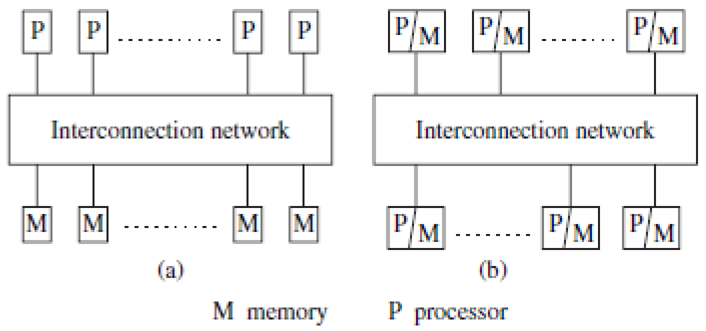

Figure: Two standard architectures for parallel systems. (a) Uniform memory access (UMA) multiprocessor system. (b) Non-uniform memory access (NUMA) multiprocessor.

i) Uniform Memory Access (UMA)

- Here, all the processors share the physical memory in a centralized manner with equal access time to all the memory words.
- Each processor may have a private cache memory. Same rule is followed for peripheral devices.

- When all the processors have equal access to all the peripheral devices, the system is called a **symmetric multiprocessor.**
- When only one or a few processors can access the peripheral devices, the system is called an **asymmetric multiprocessor.**
- When a CPU wants to access a memory location, it checks if the bus is free, then it sends the request to the memory interface module and waits for the requested data to be available on the bus.
- Multicore processors are small UMA multiprocessor systems, where the first shared cache is actually the communication channel.

ii) Non-uniform Memory Access (NUMA)

- In NUMA multiprocessor model, the access time varies with the location of the memory word.
- Here, the shared memory is physically distributed among all the processors, called local memories.
- The collection of all local memories forms a global address space which can be accessed by all the processors.
- NUMA systems also share CPUs and the address space, but each processor has a local memory, visible to all other processors.
- In NUMA systems access to local memory blocks is quicker than access to remote memory blocks.

Figure shows two popular interconnection networks – the Omega network and the Butterfly network, each of which is a multi-stage network formed of 2×2 switching elements. Each 2×2 switch allows data on either of the two input wires to be switched to the upper or the lower output wire. In a single step, however, only one data unit can be sent on an output wire. So if the data from both the input wires is to be routed to the same output wire in a single step, there is a collision.

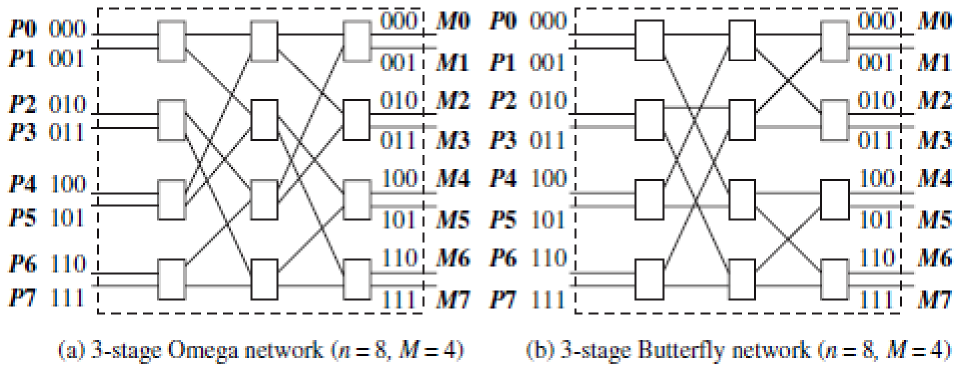

Figure: Interconnection networks for shared memory multiprocessor systems

Omega interconnection function

The Omega network which connects n processors to n memory units has n/2log2 n switching elements of size 2×2 arranged in log2 n stages. Between each pair of adjacent stages of the Omega network, a link exists between output i of a stage and the input j to the next stage according to the following perfect shuffle pattern which is a left-rotation operation on the binary representation of i to get j. The generation function is given as:

$$j = \begin{cases} 2i, & \text{for } 0 \le i \le n/2 - 1, \\ 2i + 1 - n, & \text{for } n/2 \le i \le n - 1. \end{cases}$$

The routing function from input line i to output line j considers only j and the stage number s, where s ∈ [0, log n – 1]. In a stage s switch, if the s + 1th most significant bit of j is 0, the data is routed to the upper output wire, otherwise it is routed to the lower output wire.

Butterfly network

A butterfly network links multiple computers into a high-speed network. For a butterfly network with n processor nodes, there need to be n (log n + 1) switching nodes. The generation of the interconnection pattern between a pair of adjacent stages depends not only on n but also on the stage numbers.In a stage (s) switch, if the s + 1th MSB of j is 0, the data is routed to the upper output wire, otherwise it is routed to the lower output wire.

2. A *multicomputer parallel system*

It is a parallel system in which the multiple processors *do not have direct access to shared memory*. The memory of the multiple processors may or may not form a common address space. Such computers usually do not have a common clock. The architecture is shown in Figure (b).

Torus or 2D Mesh Topology

A k X k mesh will contain k^2processor with maximum path length as 2*(k/2 -1). Every unit in the torus topology is identified using a unique label, with dimensions distinguished as bit positions.

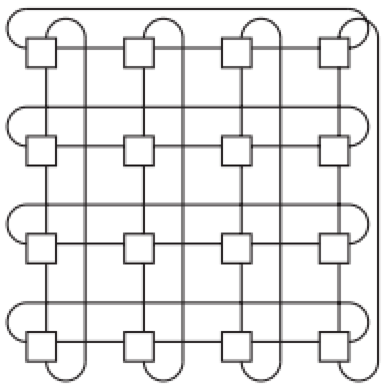

Fig : 2-D Mesh

Hypercube

The path between any two nodes in 4-D hypercube is found by Hamming distance. Routing is done in hop to hop fashion with each adjacent node differing by one bit label. This topology has good congestion control and fault tolerant mechanism.

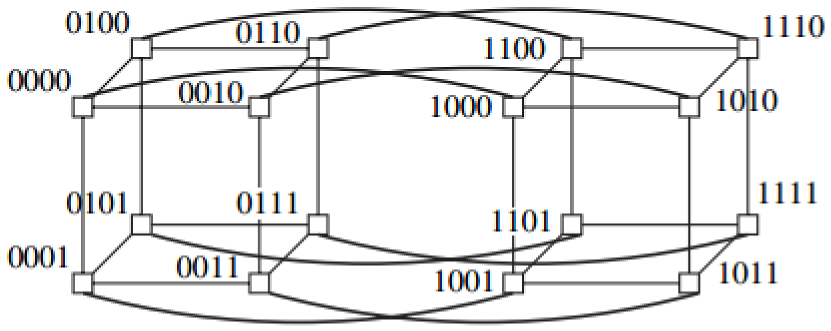

Fig : 4-D Hypercube

3. ArrayProcessors

They are a class of processors that executes one instruction at a time in an array or table of data at the same time rather than on single data elements on a common clock.

They are also known as vector processors. An array processor implement the instruction set where each instruction is executed on all data items associated and then move on the other instruction. Array elements are incapable of operating autonomously, and must be driven by the control unit.

Flynn’s Taxonomy

Flynn's taxonomy is a specific classification of parallel computer architectures that are based on the number of concurrent instruction (single or multiple) and data streams (single or multiple) available in the architecture.

Flynn's taxonomy based on the number of instruction streams and data streams are the following:

1. (SISD) single instruction, single data
2. (MISD) multiple instruction, single data
3. (SIMD) single instruction, multiple data
4. (MIMD) multiple instruction, multiple data

1. SISD (Single Instruction, Single Data stream)

- Single Instruction, Single Data (SISD) refers to an Instruction Set Architecture in which a single processor (one CPU) executes exactly one instruction stream at a time.
- It also fetches or stores one item of data at a time to operate on data stored in a single memory unit.
- Most of the CPU design is based on the von Neumann architecture and the follow SISD.
- The SISD model is a non-pipelined architecture with general-purpose registers, Program Counter (PC), the Instruction Register (IR), Memory Address Registers (MAR) and Memory Data Registers (MDR).

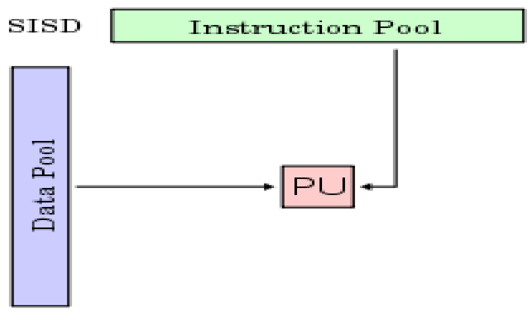

Fig : Single Instruction, Single Data Stream

SIMD (Single Instruction, Multiple Data streams)

- Single Instruction, Multiple Data (SIMD) is an Instruction Set Architecture that have a single control unit (CU) and more than one processing unit (PU) that operates like a von Neumann machine by executing a single instruction stream over PUs, handled through the CU.
- The CU generates the control signals for all of the PUs and by which executes the same operation on different data streams. The SIMD architecture is capable of achieving data level parallelism.

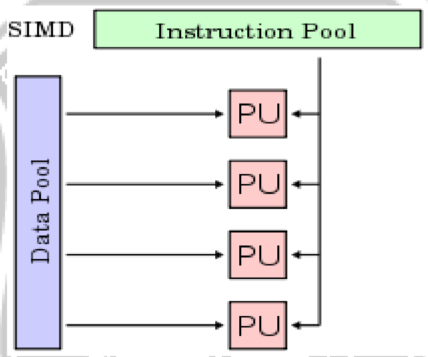

Fig :Single Instruction, Multiple Data streams

MISD (Multiple Instruction, Single Data stream)

- Multiple Instruction, Single Data (MISD) is an Instruction Set Architecture for parallel computing where many functional units perform different operations by executing different instructions on the same data set.
- This type of architecture is common mainly in the fault-tolerant computers executing the same instructions redundantly in order to detect and mask errors.

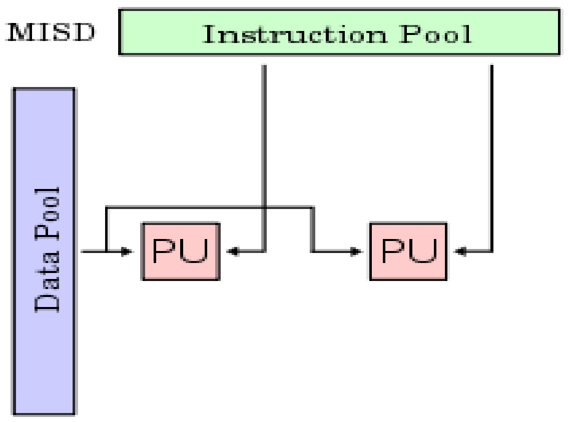

Fig : Multiple Instruction, Single Data stream

MIMD (Multiple Instruction, Multiple Data streams)

- Multiple Instruction stream, Multiple Data stream (MIMD) is an Instruction Set Architecture for parallel computing that is typical of the computers with multiprocessors.
- Using the MIMD, each processor in a multiprocessor system can execute asynchronously different set of the instructions independently on the different set of data units.
- The MIMD based computer systems can used the shared memory in a memory pool or work using distributed memory across heterogeneous network computers in a distributed environment.
- The MIMD architectures is primarily used in a number of application areas such as computer-aided design/computer-aided manufacturing, simulation, modelling, communication switches etc.

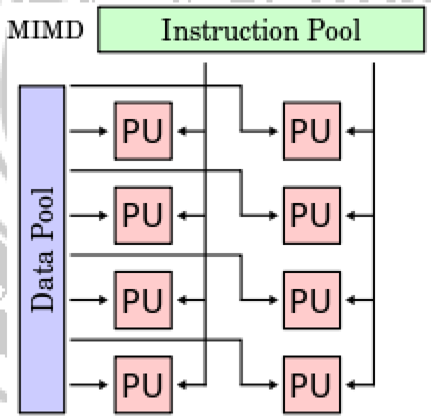

Fig :Multiple Instruction, Multiple Data streams

	Single	**Multiple**
Single	**SISD** Von Neumann Single computer	**MISD** May be pipelined computers
Multiple	**SIMD** Vector processors Fine grained data Parallel computers	**MIMD** Multi computers Multiprocessors

Coupling, parallelism, concurrency, and granularity

Coupling

The term **coupling** is associated with the configuration and design of processors in a multiprocessor system.

The degree of coupling among a set of modules, whether hardware or software, is measured in terms of the interdependency and binding and/or homogeneity among the modules.

The multiprocessor systems are classified into two types based on coupling:

1. Loosely coupled systems
2. Tightly coupled systems

Tightly Coupled systems:

- Tightly coupled multiprocessor systems contain multiple CPUs that are connected at the bus level with both local as well as central shared memory.
- Tightly coupled systems perform better, due to faster access to memory and intercommunication and are physically smaller and use less power. They are economically costlier.
- Tightly coupled multiprocessors with UMA shared memory may be either switch-based (e.g., NYU Ultracomputer, RP3) or bus-based (e.g., Sequent, Encore).
- Some examples of tightly coupled multiprocessors with NUMA shared memory or that communicate by message passing are the SGI Origin 2000

Loosely Coupled systems:

- Loosely coupled multiprocessors consist of distributed memory where each processor has its own memory and IO channels.
- The processors communicate with each other via message passing or interconnection switching.
- Each processor may also run a different operating system and have its own bus control logic.
- Loosely coupled systems are less costly than tightly coupled systems, but are physically bigger and have a low performance compared to tightly coupled systems.

- The individual nodes in a loosely coupled system can be easily replaced and are usually inexpensive.
- The extra hardware required to provide communication between the individual processors makes them complex and less portable.
- Loosely coupled multicomputers without shared memory are physically co-located. These may be bus-based (e.g., NOW connected by a LAN or Myrinet card) or using a more general communication network.
- These processors neither share memory nor have a common clock.
- Loosely coupled multicomputers without shared memory and without common clock and that are physically remote, are termed as distributed systems.

Parallelism or speedup of a program on specific system

- It is the use of multiple processing elements simultaneously for solving any problem.
- Problems are broken down into instructions and are solved concurrently as each resource which has been applied to work is working at the same time.
- This is a measure of the relative speedup of a specific program, on a given machine. The speedup depends on the number of processors and the mapping.
- It is expressed as the ratio of the time T(1) witha single processor, to the time T(n) with n processors.

Parallelism within a parallel/distributed program

- This is an aggregate measure of the percentage of time that all the processors are executing CPU instructions productively, as opposed to waiting for communication operations.

Concurrency

- Concurrent programming refer to techniques for decomposing a task into subtasks that can execute in parallel and managing the risks that arise when the program executes more than one task at the same time.
- The parallelism or concurrency in a parallel or distributed program can be measured by the ratio of the number of local non-communication and non-shared memory access operations to the total number of operations, including the communication or shared memory access operations.

Granularity

Granularity or grain size is a measure of the amount of work or computation that is performed by that task.

- Granularity is also the communication overhead between multiple processors or processing elements.
- In this case, granularity as the ratio of computation time to communication time, wherein, the computation time is the time required to perform the computation of a task and communication time is the time required to exchange data between processors.

Parallelism can be classified into three categories based on work distribution among the parallel tasks:

1. **Fine-grained:** Partitioning the application into small amounts of work done leading to a low computation to communication ratio.
2. **Coarse-grained parallelism:** This has high computation to communication ratio.
3. **Medium-grained:** Here the task size and communication time greater than fine-grained parallelism and lower than coarse-grained parallelism.

   Programs with fine-grained parallelism are best suited for tightly coupled systems.

Classes of OS of Multiprocessing systems:

- **Network Operating Systems:** The operating system running on loosely coupled processors which are themselves running loosely coupled software
- **Distributed Operating systems**: The OS of the system running on loosely coupled processors, which are running tightly coupled software.
- **Multiprocessor Operating Systems:** The OS will run on tightly coupled processors, which are themselves running tightly coupled software.