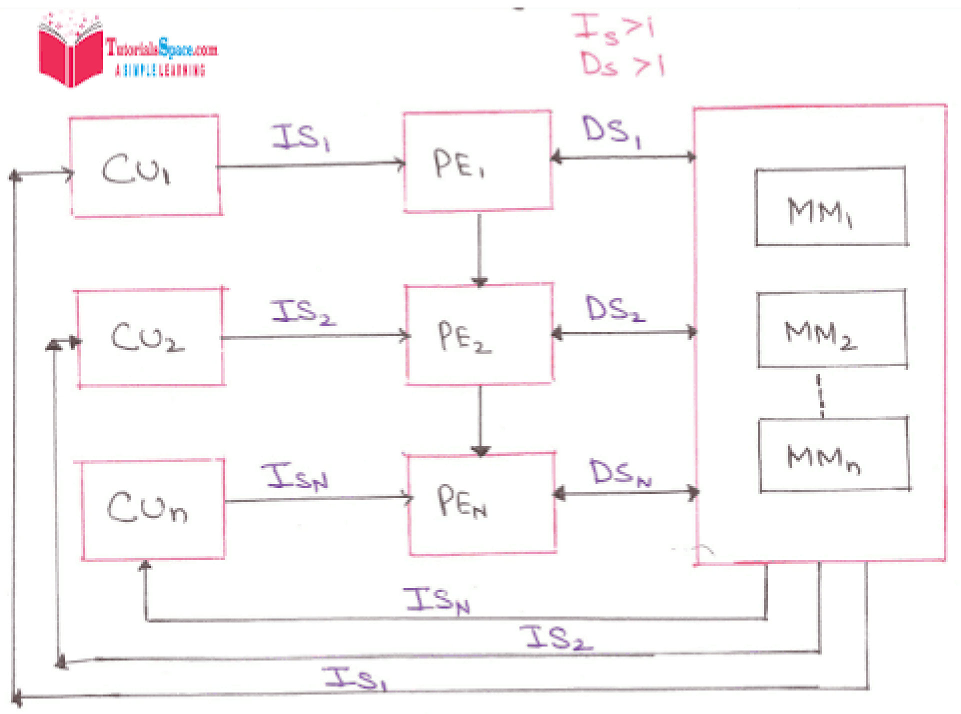
INTRODUCTION TO MIMD ARCHITECTURE

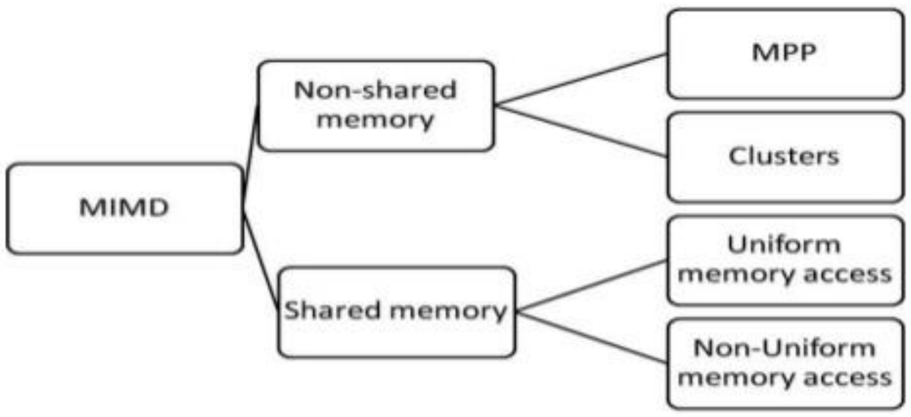
(MIMD) Architecture is one of the recent and popular computer architecture. Multiple instructions worked on multiple data to boost the performance of computer. MIMD architecture works with shared memory programming model and distributed memory programming model. Each model has its advantages and disadvantage. In this paper we present a detail review of recent work done in MIMD architecture, its comparison with other architectures especially with Single Instruction Multiple Data architecture (SIMD).

MIMD (Multiple Instruction Multiple Data) computers are basically computers with threads and process level architectures. MIMD is appropriate for programs restricted by condition statements. After advancement and development in integrated circuit technology the MIMD architectures became common. Microprocessors were produced after this advancement in technology. Microprocessors were very helpful and simple to design multiprocessor systems. Small systems, including only few processors were usual. These systems were called scalable parallel computers. here we present a complete review about MIMD architecture its comparison with other architectures and which architecture is better. II. MIMD ARCHITECTURE MIMD architecture comprises of many processing elements (Processors) which are connected with some memory through a common bus . Task and data is distributed among these processing elements. So at the same time all processing elements execute the instruction on their data corresponding to complete the given task. After advancement and development in integrated circuit technology the MIMD architectures became common. Microprocessors were produced after this advancement in technology. Microprocessors were very helpful and simple to design multiprocessor systems.



Multiple Instruction and Multiple Data Stream

MIMD Architecture: Classification



SHARED MEMORY

MIMD architecture works with two types of memory, shared memory is one of them. All Processing Elements (PE) share a common memory address. Every processing element can access any module via interconnected network directly. They communicate by writing into common address space. PE's are separate but they have a common memory. There are two advantages of shared memory, in shared memory the data and code is not divided into partitions, the main advantage of this is that those programming techniques which are used in uniprocessors can also be used for multiprocessor environment. In shared memory systems new programming language is not required. One another advantage is that physical movement of data is not required during process communication. There are two types of memory access in shared memory model, which are Uniform Memory Access and Non- Uniform Memory Access. In Uniform Memory Access all Processing Elements access the data with same access time. On the other hand in Non- Uniform Memory Access some processing elements are closer to memory and some access it later.

Types of MIMD Architecture:

Shared memory model

The processors are all connected to a "globally available" memory, via either software or hardware means. The operating system usually maintains its memory coherence.[3]

From a programmer's point of view, this memory model is better understood than the distributed memory model. Another advantage is that memory coherence is managed by the operating system and not the written program. Two known disadvantages are: scalability beyond thirty-two processors is difficult, and the shared memory model is less flexible than the distributed memory model.[3]

There are many examples of shared memory (multiprocessors): UMA (Uniform Memory Access), COMA (Cache Only Memory Access).[4]

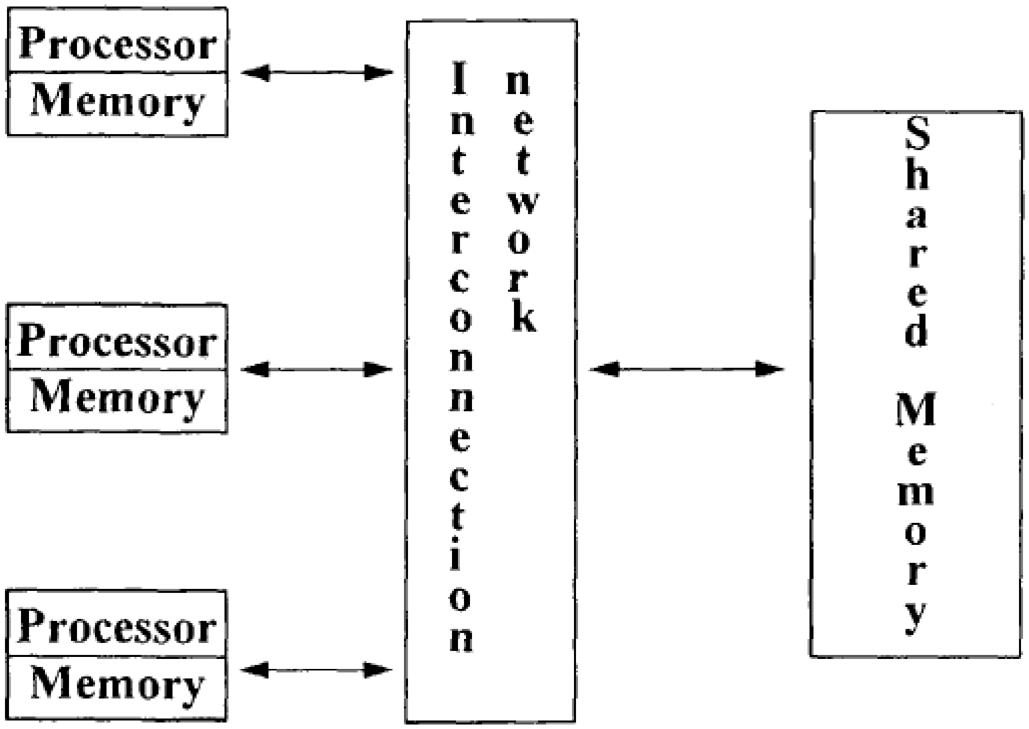
Bus-based (uniform)

MIMD machines with shared memory have processors which share a common, central memory. In the simplest form, all processors are attached to a bus which connects them to memory. This means that every machine with shared memory shares a specific CM, common bus system for all the clients.

For example, if we consider a bus with clients A, B, C connected on one side and P, Q, R connected on the opposite side, any one of the clients will communicate with the other by means of the bus interface between them.

Hierarchical (Non-uniform)

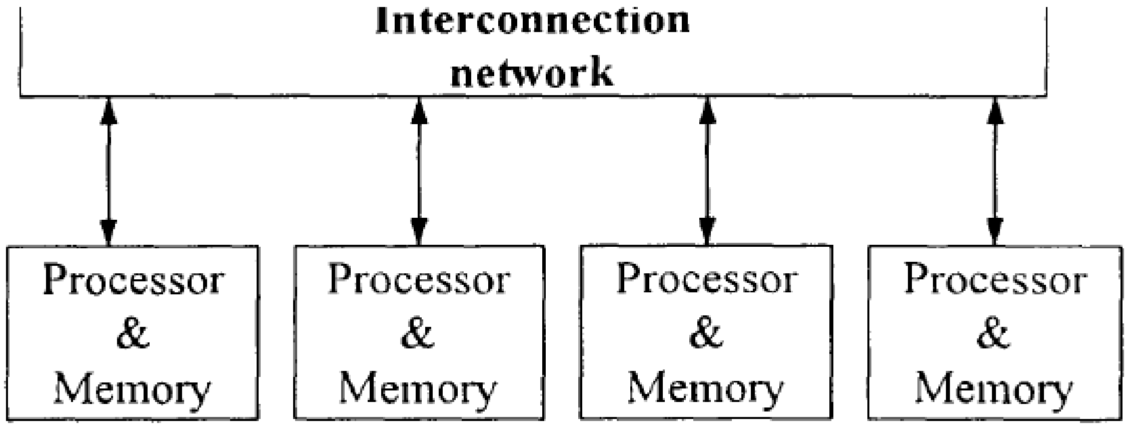
MIMD machines with hierarchical shared memory use a hierarchy of buses (as, for example, in a "Fat tree") to give processors access to each other's memory. Processors on different boards may communicate through inter-nodal buses. Buses support communication between boards. With this type of architecture, the machine may support over nine thousand processors.



Distributed memory

In distributed memory MIMD machines, each processor has its own individual memory location. Each processor has no direct knowledge about other processor's memory. For data to be shared, it must be passed from one processor to another as a message. Since there is no shared memory, contention is not as great a problem with these machines. It is not economically feasible to connect a large number of processors directly to each other. A way to avoid this multitude of direct connections is to connect each processor to just a few others. This type of design can be inefficient because of the added time required to pass a message from one processor to another along the message path. The amount of time required for processors to perform simple message routing can be substantial. Systems were designed to reduce this time loss and hypercube and mesh are among two of the popular interconnection schemes.

Examples of distributed memory (multiple computers) include MPP (massively parallel processors), COW (clusters of workstations) and NUMA (Non-Uniform Memory Access). The former is complex and expensive: lots of super-computers coupled by broad-band networks. Examples include hypercube and mesh interconnections. COW is the "home-made" version for a fraction of the price.



Hypercube interconnection network

In an MIMD distributed memory machine with a hypercube system interconnection network containing four processors, a processor and a memory module are placed at each vertex of a square. The diameter of the system is the minimum number of steps it takes for one processor to send a message to the processor that is the farthest away. So, for example, the diameter of a 2-cube is 2. In a hypercube system with eight processors and each processor and memory module being placed in the vertex of a cube, the diameter is 3. In general, a system that contains 2^N processors with each processor directly connected to N other processors, the diameter of the system is N. One disadvantage of a hypercube system is that it must be configured in powers of two, so a machine must be built that could potentially have many more processors than is really needed for the application.

A hypercube topology (cf. Figure I.11) uses $N=2^n$ processors arranged in an n–dimensional cube, where each node has $n = \log_2 N$ bi–directional links to adjacent nodes. Individual nodes are uniquely identified by n–bit numeric values that range from 0 to N–1. These are assigned in a manner that ensures adjacent node's value to differ by a single bit. Communication diameter of such a hypercube topology is $n = \log_2 N$. The degree of the nodes in such a topology grows in proportion to $\log_2 N$. Thus the number of nodes can be doubled at the cost of increasing the number of interconnections per node by a single communication link per node. Some of

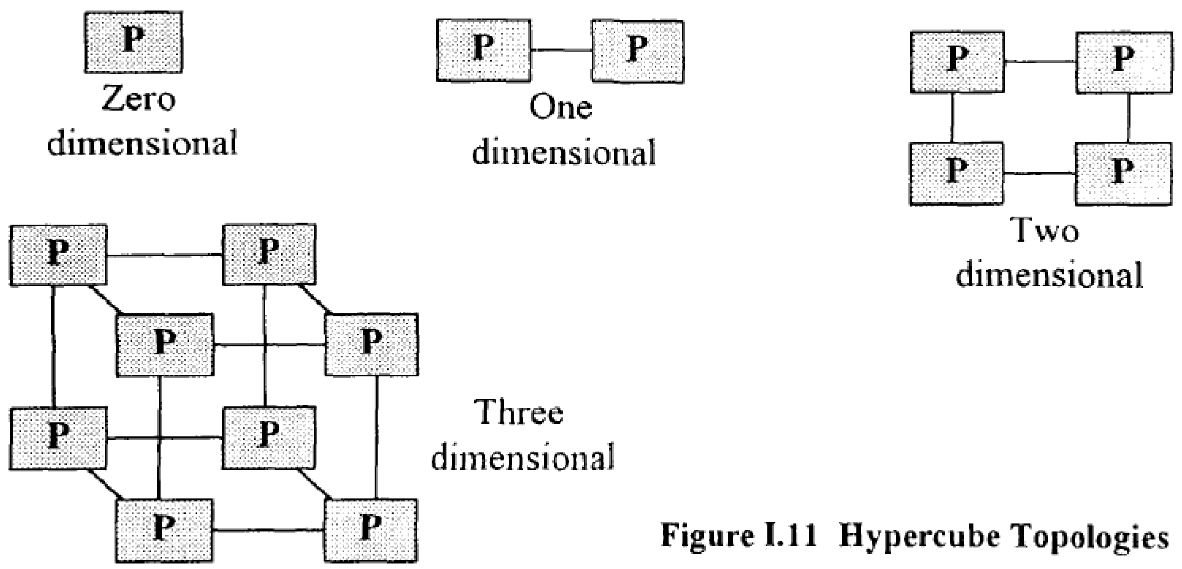


Figure I.11 Hypercube Topologies

Mesh interconnection network

In an MIMD distributed memory machine with a mesh interconnection network, processors are placed in a two-dimensional grid. Each processor is connected to its four immediate neighbors. Wrap around connections may be provided at the edges of the mesh. One advantage of the mesh interconnection network over the hypercube is that the mesh system need not be configured in powers of two. A disadvantage is that the diameter of the mesh network is greater than the hypercube for systems with more than four processors.

Mesh Topology Architectures

A symmetrical 2D mesh (cf. Figure I.9a), or lattice topology has n^2 nodes, each connected to their four immediate neighbours. Wrap–around connections (cf. Figure I.9b) at the edges are sometimes provided to reduce

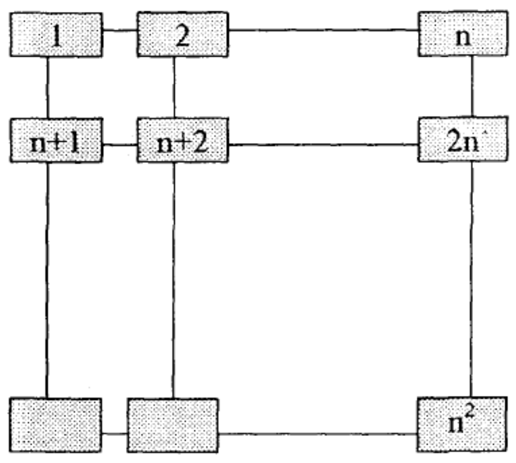


Figure I.9a nxn Mesh Topology

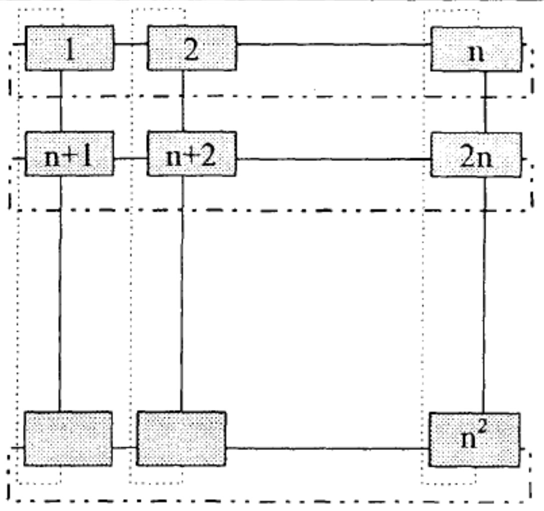


Figure I.9b nxn Warp Topology

the communication diameter from $2(n-1)$ to $2[n/2]$. Such a topology is also sometimes called as torus topology. Increasing the mesh size does not alter node degree. Meshes are relatively fault tolerant, since a single fault results in no more than two additional links being traversed to bypass the faulty node.