Unit 1 : Introduction to Parallel Computing

Serial Computing:

1. Problem statement is broken into discrete instructions.
2. Instructions are executed one by one
3. Only one instruction is executed at any moment of time. (highly problematic)

Parallel Computing:

1. Multiple processing elements simultaneously for solving any problem.
2. Problems are broken into instructions and are solved concurrently as each resource that has been applied to work.

Advantages of Parallel over Serial:

1. Saves time and resources
2. Impractical to solve larger problems using serial computing
3. Can take advantage of non local resources when local resources are finite.

Motivating Parallelism

Moore's Law, coined by Gordon Moore in 1965, observes that the complexity of computer chips doubles approximately every 18 months. This prediction has largely held true over the years, leading to massive advancements in computing

power. However, there are limits to how far we can push this trend, which has sparked debate.

One major challenge in harnessing the increasing transistor count on chips is translating them into useful operations per second (OPS). To tackle this, we rely on parallelism, which means carrying out multiple tasks simultaneously. This can be done implicitly (where tasks are broken down and executed concurrently without explicit programming) or explicitly (where programmers intentionally design software to run tasks in parallel).

Another critical factor affecting overall computing speed is memory and disk access speed. While processors have become faster, the speed at which they can access data from memory hasn't kept pace. This creates a bottleneck in performance. To address this, we use memory hierarchies like caches, which store frequently accessed data closer to the processor, improving access times. Parallel platforms also help by providing larger caches and higher memory bandwidth, which can enhance overall memory system performance.

Furthermore, as networking technology advances, there's a growing trend towards using the internet as a giant parallel computing environment. Many applications naturally lend themselves to this distributed computing model. For instance, projects like SETI@home use the collective power of many computers connected over the internet to analyze data from outer space.

In summary, as transistor counts increase, we need to find ways to effectively utilize them for computation. Parallelism, both implicit and explicit, plays a crucial role in maximizing computing power. Additionally, addressing memory speed and leveraging distributed computing over networks are key strategies in overcoming performance bottlenecks and unlocking the full potential of modern computing systems.

Scope of Parallel Computing

1. In engineering and design, parallel computing helps optimize things like airfoil shapes, internal combustion engine efficiency, circuit layouts, and structural integrity. Especially in micro and nanoscale designs (like MEMS and NEMS),

parallel computing deals with complex physical phenomena at different scales, making it crucial for modeling and algorithm development.

2. Scientific applications, such as bioinformatics and astrophysics, benefit from parallel computing to analyze massive datasets. For example, in bioinformatics, researchers analyze genetic data to develop new drugs, while in astrophysics, they study galaxy evolution using telescope data.
3. Commercially, parallel computing powers web servers, database servers, and high-volume transaction systems. For instance, on Wall Street, large brokerage houses handle huge numbers of simultaneous user sessions and transactions using supercomputers.
4. In computer systems, parallel processing tackles challenges like intrusion detection in network security and factoring large integers in cryptography. Even embedded systems in cars rely on parallel algorithms for tasks like optimizing performance and handling.
5. Overall, parallel computing is essential for tackling complex problems across various fields, from scientific research to business operations and computer security.

1. **Stored-program computer architecture**:
    - This architecture is the foundation of most modern computers. In a stored-program computer, both program instructions and data are stored in the computer's memory.
    - This allows for more flexibility because programs can be easily modified and executed without physically changing the computer's hardware.
    - The CPU (Central Processing Unit) fetches instructions from memory, decodes them, and executes them. This sequence is repeated until the program finishes.
    - It contrasts with earlier computer architectures where instructions were hardcoded into the hardware, making it challenging to modify or update programs.

-

2. **General-purpose Cache-based Microprocessor architecture**:
    - This refers to the design of modern microprocessors, the "brains" of computers, which are optimized for speed and efficiency.
    - **General-purpose**: These microprocessors are designed to handle a wide range of tasks and applications. They are not specialized for a specific function but can execute various programs efficiently.
    - **Cache-based**: These microprocessors use cache memory, which is a small, fast memory unit located close to the CPU. Cache memory stores frequently accessed data and instructions, allowing the CPU to access them quickly, reducing the time spent waiting for data from the main memory.
    - **Microprocessor**: This is a single integrated circuit that contains the CPU, memory management unit, and other essential components of a computer. It performs all the arithmetic, logic, and control functions required for program execution.
    - **Architecture**: This refers to the overall design and organization of the microprocessor, including its instruction set, memory organization, and data pathways.

Parallel Programming Platforms: Implicit Parallelism

Pipelining:

1. Attempting to use resources such as ALU, Buses, registers to the fullest or continuously can be achieved by pipelining.
2. Instructions move from one stage to another to accomplish the assigned operation. Hence, at most of the times the part of the processor is handling instructions making the attribute of the processor being used continuosly.

3. Non pipelined : Processor fetches an instructioin and its decoded and they are read from the register and the computation is performed.
4. The problem with non pipelined is that hardware amount needed to perform these steps are different and the rest of the hardware is idle.
5. Pipeline is a technique of overlapping the execution of instructions.
6. 2 stage pipelining : Fetch instruction and execute instruction.
7. Pipeline is the process of fetching the next instruction when the current instruction is being executed.
8. All units of processor are operating.

Superscalar Execution:

1. Superscalar processor executes more than one instruction at a time during a single clock cycle.
2. This execution can fetch and decode several instructions at a time.
3. It exploits the potential of Instruction level parallelism (I L P)

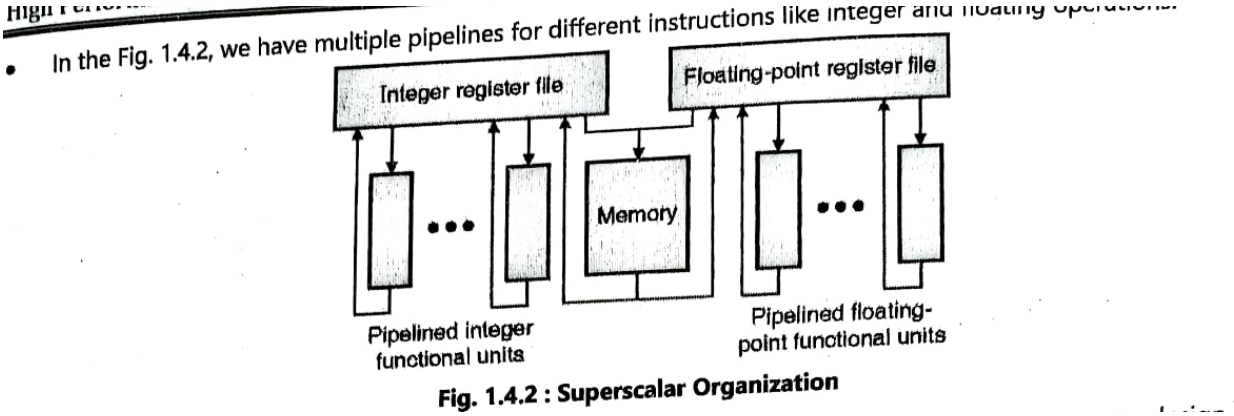


Fig. 1.4.2 : Superscalar Organization

However, the cost and complexity is a major issue in this design

Very Long Instruction Word Processors (VLIW)

1. Describes a architecture where a language compiler breaks the instructions in basic operations that can be performed by the processor in parallel.
2. Takes advantage of instruction level parallelism
3. Less complex approach - and multiple operations are performed simultaneously.
4. In VLIW processor, instructions consist of multiple independent operatons grouped together to store instructions in a single word.
5. This length of word is 52 bits to 2kbits.

   a) VLIW processor structure

   1. It contains multiple operations in a single instruction
   2. It relies on the compiler to find parallelism and schedule dependency free code.
   3. The multiple operations are independent instructions and have no flow dependencies.

   b) Adv disadv

   1. adv : no runtime dependencies, no runtime scheduling decisions
   2. disadv: no tolerance for difference in functional units

Trends in Microprocessor and architecture

1. Current processors use resources in functional units to execute multiple instructions in the same cycle.
2. Major trend : Use of complex architecture to exploit the Instruction level parallelism.
3. Superscalar : Hardware usage to dynamically find out data independent instruction.
4. VLIW : Relies on compiler to find ILP and schedule the execution

5. Performance has been driven by: Innovation in compilers, improvements in architecture and improvements in VLSI
6. Latest superscalar microprocessors, execute 4 to 6 instrcutions concurrently.

Limitations of Memory system performance

1. Effective performance also relies on the speed of the processor and the abilty of the memory sys to feed data to the processor.
2. Latency and bandwidth.
3. Latency : Time from the issue of the memory request to the time the data is available at the processor.
4. Bandwidth: Rate at which data can be pumped into the processor.
5. To improve the latency, cache can be used.
6. Impact of memory bandwidth (rate at which data can be read or stored in memory by the processor) : determined by the bandwidth and the memory units.
7. Bandwidth can be increased by increasing the size of memory blocks.
8. Hiding memory latency technique : Pre-fetching and multithreading
9. Pre-fetching : why not advance the loads so that by the time data is needed, its already there.
10. Multithreading: processor is capable of switching threads every cycle, and latency can hide latency effectively.

Dichotomy of Parallel Computing platforms

1. Control Structure of Parallel Platform
    a. Granularity is the ratio of computation to communication
    b. Its used to describe about the division of a task into a number of subtasks.

c. Granularity : a) Fine grained : system divided into large networks of a small part, b) Course grained : system is divided into smaller numbers of large parts.

d. Processing units in parallel computers either operate under the centralised control of a single control unit or work independently.

e. Models : SIMD and MIMD

f. SIMD :

   a. Single control unit that dispatches the same instruction to multiple processing units.

   b. This is used when alot of data has to perform the same operation.

   c. eg. Vector processors

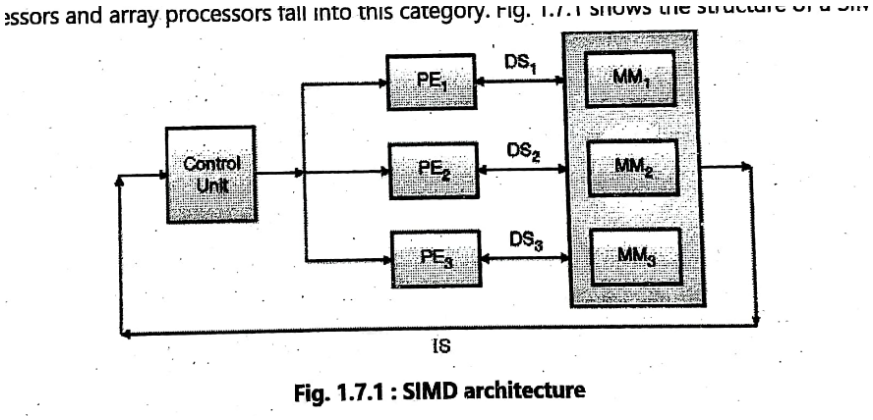


Fig. 1.7.1 : SIMD architecture

g. MIMD:

   a. Complete parallel processing example.

   b. Each processor has its control unit, each unit can execute different instructions on a different set of data.

   c. eg. SMPs (Symmetric multiprocessors)

SIMD	MIMD
It is also called as Array processor.	It is also called as multiprocessor.
Here, single stream of instruction is fetched.	Here, multiple streams of instructions are fetched.
The instruction stream is fetched by shared memory.	The instruction streams are fetched by control unit.
Here, instruction is broadcasted to multiple processing elements.	Here, instructions streams are decoded to get multiple decoded instruction streams.
SIMD computers require less hardware than MIMD.	Requires more hardware.

a) Data flow model:

1. Information exchange and flow within the system are represented by data flow.
    a. Moving data from input to file storage and report production, represent the flow of data in an information system.

b) Demand driven computation:

1. Top down method by requiring evaluation of the first demanding value.
2. Basically, its operations are only carried out when their outcomes are required by instructions.

c) Cache memory:

1. Hold instructions that the processor frequently needs to execute.

Communication Model of parallel platform:

1. Shared address space approach
2. Message passing approach

a. Shared address space approach

- Platforms that provide shared data space are called shared data machines.
- Processors interact by modifying data objects
- Classification: a) NUMA (Non uniform memory access) and b) UMA (Uniform memory access)
- NUMA - Time taken to access any memory is non-identical
- NUMA
    1. Each processor has its own memory module to access directly.
    2. It can access any other module belonging to another processor using shared bus.
    3. Shared logical space is present but physical memory is distributed among CPUs.
    4. All processors can see all memory

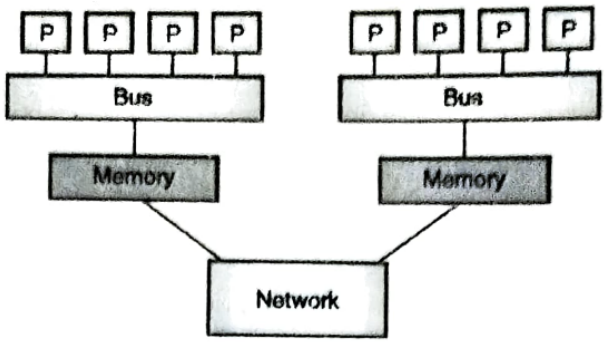


Fig. 1.7.4 : NUMA architecture

- UMA
    1. All processors shared a unique centralised memory so processors have same memory access time.
    2. Each processor gets equal priority to access the main memory of the machine.

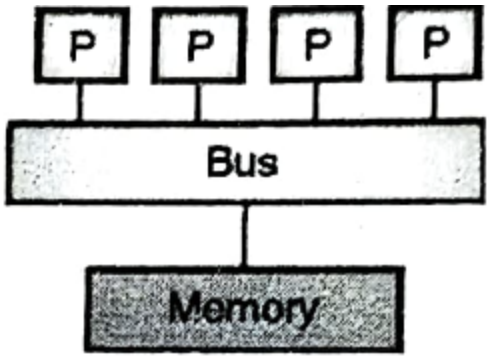


Fig. 1.7.5 : UMA Architecture

b. Message Passing approach

- Exchange messaging for sharing data
- Model allows multiple processors to read and write data to the message queue without being connected to each other.
- Messages are stored in the queue until the recipient receives them.

Physical Organisation of parallel platforms and stored program computer

Evolution of parallelism

1. Overlap of fetch and execute
2. Pipelining and multiple functional units.

Ideal model for computing

PRAM (Parallel Random Access Machine)

1. Shared memory multiprocessor
2. Unlimited number of processors which can access the shared memory in constant time and unlimited local memory.
3. Suitable for modern architectures.
4. Consists of control unit, global memory and private memory.

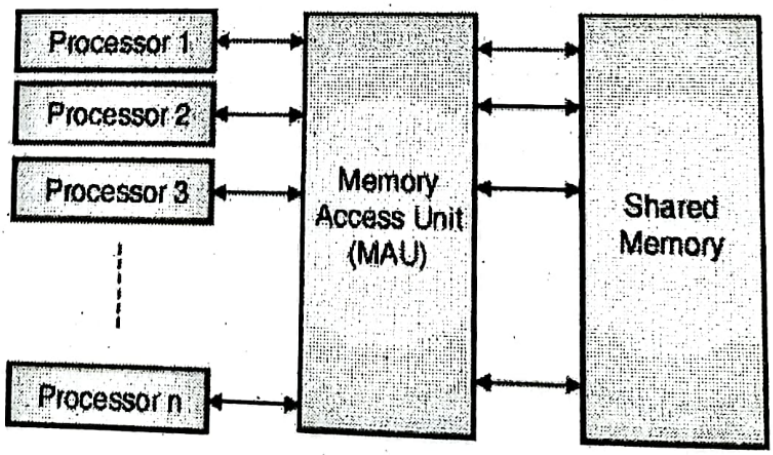
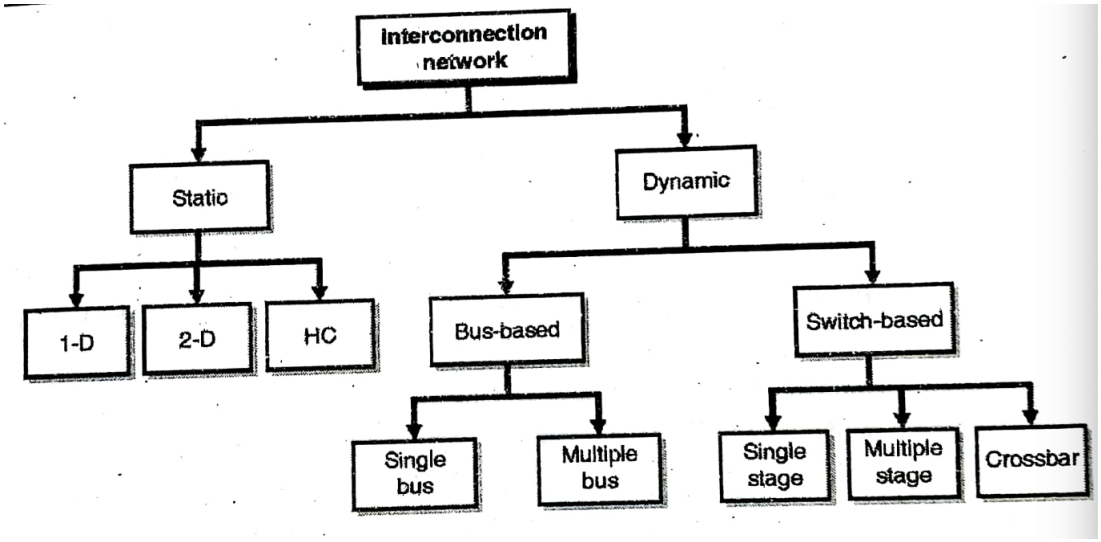


Fig. 1.8.3 : PRAM Architecture

Interconnection network for parallel computer



Static : Passive connections, communication nodes are fixed and cannot be reconfigured to have a diff connection path.

Dynamic: Connection can be reconfigured to establish new paths.

Table 1.9.1 : Difference between Static and Dynamic networks

Sr. No.	Static Networks	Dynamic Networks
1.	The connecting paths between the processing elements of the static networks are static or passive.	The connecting paths between the processing elements are dynamic or active.
2.	Establishing of links between two processor during the execution of program or dynamically is not possible.	Establishing of links between two processor during the execution of program or dynamically is possible.
3.	The static network is made up of fixed processor to processor or point to point connections.	Dynamic networks are made up of channels that can be switched to be present and being removed or disconnected.
4.	These networks are used in a distributed system.	These networks are used in a shared memory system with multiple processors.
5.	There are various types of static networks like linear array, ring, tree, star, mesh, cube, hypercube etc.	There are various types of dynamic networks like buses, crossbar switches, single stage dynamic network, multi stage dynamic network etc.

▼ Static networks Topologies:

1. Linear topology:
    - Processing elements are in a series.
    - Links are n-1, where n is the number of elements
    - Degree - 2 (middle nodes where each node is connected to 2, one on the left and one on the right)
    - Diameter - (n-1) - (first and last have to communicate through (n-1) elements)
    - Not efficient for large num of elements.

2. Ring Topology:
    - Number of links - n
    - Degree - 2
    - Diameter - n/2
    - There is a connection from the last to the first element for simpler communication.

3. Star topology
    - Diameter - 2
    - Degree - (n-1) (central element is connected to (n-1) elements)

4. Meshes and Torus
    - Mesh - Used for cases such as matrix multiplication
    - Degree - 4
    - Link - 2(n^2 -n)
    - Diameter 2(n-1)

- Torus is a variant of the mesh topology - has a better performance as you need to go through only 6 processing elements and the extra connection paths is better.

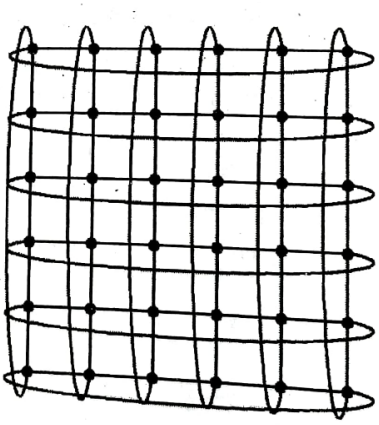


Fig. 1.9.6 : Torus

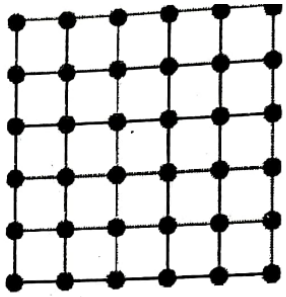


Fig. 1.9.5 : 2-D Mesh

Mesh

5. Hypercubes

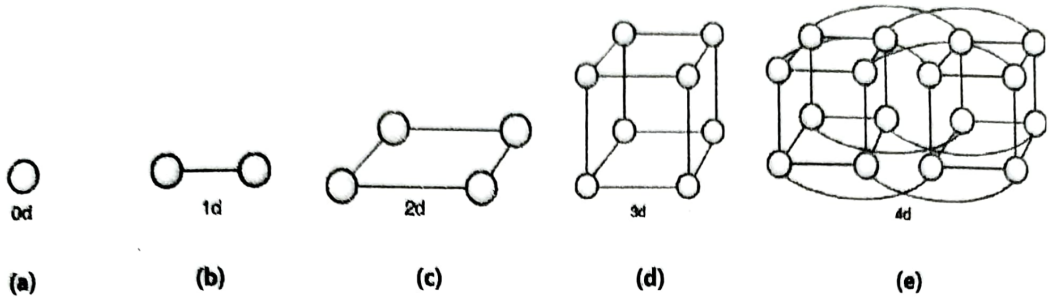


Fig. 1.9.7 : Hypercube of different dimensions i.e. 0, 1, 2, 3 and 4

- The diameter is “k” for k number of dimensions
- This topology is commonly used in intels IPSC.

6. Trees
    - There is a processing element at the top which is connected to 2 different processing elements.
    - Degree - 3
    - Links - (n-1)
    - Diameter increases as a logarithmic value.

7. Fully connected network
    - Network where each node interconnects all nodes of the network.
    - Every node has a direct link with all other nodes of the network.
    - Drawback : It requires too many connections.

8. Fat tree
    - In the normal tree network - every branch has the same thickness regardless of their place in the hierarchy.
    - In fat tree, branches near the top → thicker

- This has increased bandwidth of edges into root direction.

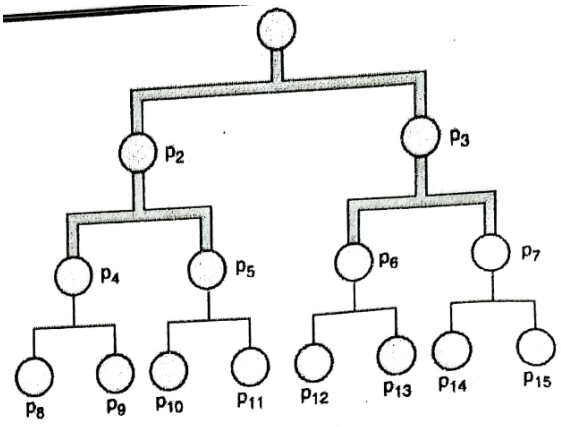


Fig. 1.9.12 : Fat Treez

▼ Dynamic Networks topologies:

1. Bus Based interconnect

   a) Single bus interconnect system

   - Simplest way to connect the multiprocessor systems. Makes use of local caches and hence reduces the access between the processor and memory.
   - Size - 2-50 elements
   - This can limit the bandwidth of the bus as only 1 processor can access the bus at a time.

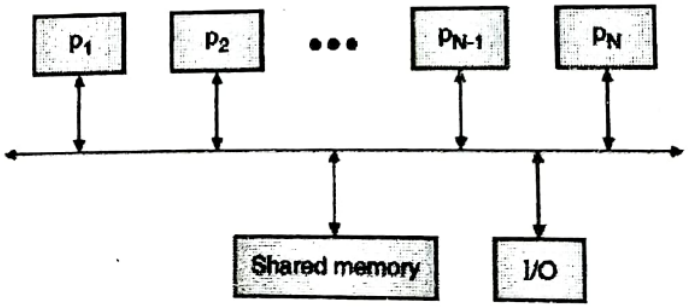


Fig. 1.9.13 : Single bus interconnect system

b) Multi bus interconnect system

- There are several buses to which multiple processors are connected.

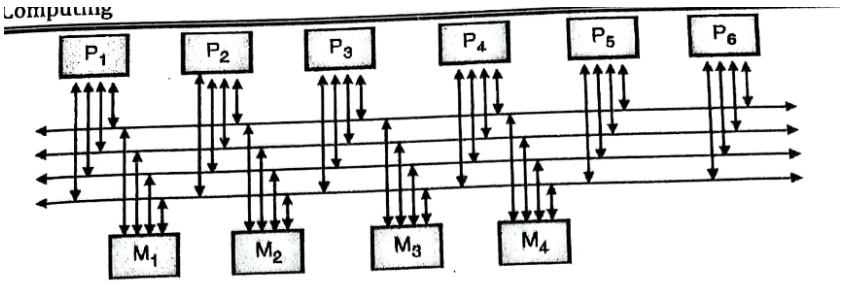


Fig. 1.9.14 : Multi Bus Interconnect System

2. Switch based interconnect network

a) Crossbar interconnect network

- Provides connections among all its inputs and outputs simultaneously.
- Its a non blocking network, allowing multiple input and output connections.
- Can have 1 to one and one to many message passing.
- Every incident packet → prefixed with a tag of the destination. The packet is recieved by the selector and it checks the status.

- If the output port is free the connection is established and data is transferred otherwise the connection is refused.

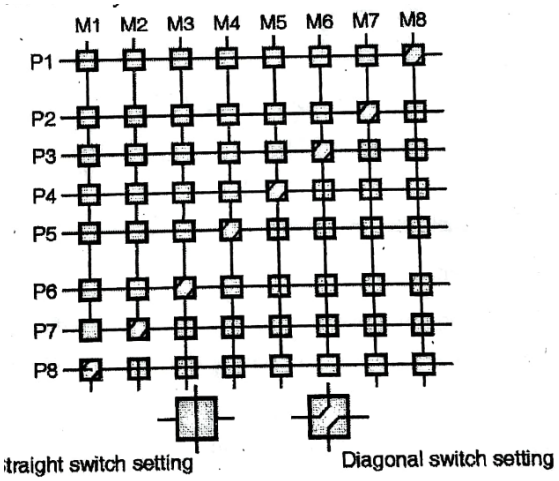


Fig. 1.9.15 : Crossbar network

b) Single stage interconnect network

- Single switching element between inputs and outputs of the network.

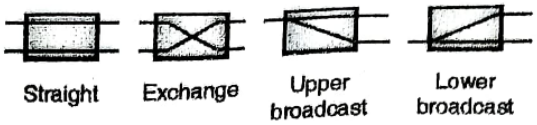


Fig. 1.9.16 : Different settings of 2 × 2 switching elements in single stage

- The upper broadcast makes a connection from the upper to all the output while the lower broadcast provides a connection from lower input to all the outputs.
- The single stage shuffle exchange network comprises of a perfect shuffle at the input, followed by the single stage of switching element and finally the output that is buffered.

c) Multistage interconnect network

- Data here need not be circulating in the network, it can directly reach the output port.
- If input stage and output stage are the same then its called as single sided if the input and outputs are differnt then its called as 2 sided.

Cache-Coherence in Multi-processor systems

1. 2 main challenges : How much program is going to be parallel and how much is going to be sequential
2. Solution: Caching the data and processes.
3. Cache increases the bandwidth and the bus speed, reduces latency.

Coherence:

1. Reads by any processor must return recent written value.
2. Writes by more than one processor must be serialised.
3. Basic problem of coherence: Support for migration and replication as caches allow movement of data frm one memory to cache and main memory to cache.
4. Replication → Multiple copies of same data in different caches.
5. Solution to cache coherence: 1. Snoopy bus and 2. Directory based protocol.

1. Snooping solution:
    a. Snoopy bus → Sends requests for data to processors.
    b. Processors snoop and see if they have a copy.

Basic approach

1. Write invalid protocol: Processor gains exclusive access of a block before invalidating all copies. When a processor writes, invalidates all copies of data that is in other caches.
2. Write update protocol: When a processor writes, it updates all shared copies of that block, and it broadcasts that value and updates the copies in the other caches.

2. Directory based protocol
    a. Keep track of what is shared in one centralised place.
    b. This is a non broadcasting protocol
    c. This protocol stores the location of all caches copies of every block of shared data.
    d. Cache location → Centralised
    e. Scales better than snoopy.
    f. Protocols : Handling read miss and handling a write.

Communication Costs in Parallel Machines

1. Parallel computing platform provides facility of execution of programs.
2. 2 modes of communication : Message passing and Shared memory.
3. Issues affecting overall communication
    a. Message passing cost in parallel computers
        - Startup time: Time spent to recieve and send nodes
        - Per hop time: Function of number of hops and includes factors such as network delays.

- Per word transfer time: Includes overheads that are determined by the length of the message.

b. Store and forward routing
   - Message traversing multiple hops is recieved at an intermediate hop.
   - Total cost : t (comm) = ts + (mtw +th) l
   - ts →startup time , m→ number of words th→per hop time

c. Packet routing
   - Breaks messages into packets and pipelines.
   - Each packet carries out sequencing, routing info, error checking.

d. Cost through routing
   - Concept of packet routing to an extreme by further divding messages into fixed size units called flits.
   - Identical to packet routing, however the per word transfer time is low.

Scalable design principles

1. Scalability: Provide enhanced resources to accomodate increase in performance
2. Asynchronous processing: Enables process execution without blocking resources.
3. Concurrency and parallelisation
4. Intelligent load distribution: Spreading the load across many instances to handle the requests.
5. Caching: Reduces the overhead of fetching it over and over again.
6. Queuing

7. Shared nothing architecture
8. Loose coupling high cohesion: reduce coupling and increasing cohesion.
9. Decentralisation, multiprocessor
10. Parallel system : Increase the number of processors and problem size.

N wide superscalar architectures

A wide issue architecture can issue more than one instruction per clock cycle.

Statically: Scheduled superscalar architecture. Executes instructions in the order presented.

VLIW : Determines which instructions to dispatch on a clock cycle.

Dynamically: Scheduled superscalar architecture, uses hardware logic.

Superscalar architecture processor:

- Consists of multiple execution units.
- Can execute independent instructions simultaneaously and can increase the speed.
- But the issue degree is restricted from 2-5.
- When an operation unit writes result back to the register file, result becomes visible from the next clock cycle.

ILP in superscalar processor:

1. Instruction level parallelism - Processor does the entire processing in one cycle.
2. ILP in superscalar processors have become complex that fowards the results of each instruction to several instruction units to reduce the delay.

Pipelining in superscalar processors

1. Pipelining is important for demonstrating the speed increase by the superscalar feature of processors.
2. Multiple instruction pipelines are used
3. To implement multiple operations simultaneously → Multiple execution units to execute each instruction independently.
4. Instruction fetch unit → fetches m instructions → decoded → instruction window takes decoded instructions → each instruction issued to a execution unit → executed → moved to the retire and write back unit.

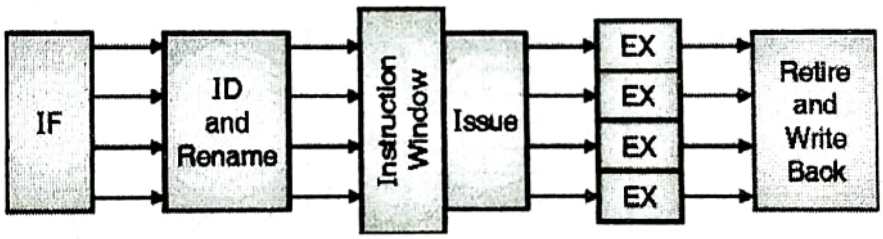


Fig. 1.13.3 : Block diagram of a typical superscalar processor

Multicore architecture

1. If there is demand for high computational power, to support these demands, efficient processors that have necessary hardware

components are required.

2. Single or centralised approaches relies on a single CPU.
3. One way of achieveing increase performance → increase the operation clock frequency, but with high cost of computation.
4. Also, we cant offer high performance with the cost of reduced battery life.
5. Multicore : processors incorporate multiple cores on the same chip and work on the same clock frequency supplied.

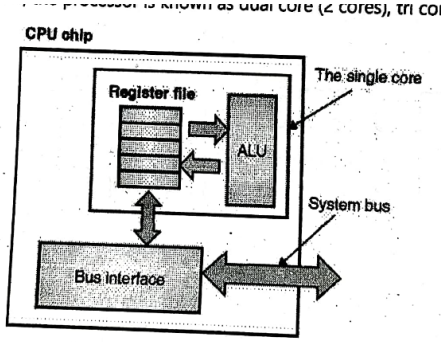


Fig. 1.14.1 : Single core processor

single processor, only one core is utilized by the CPU.

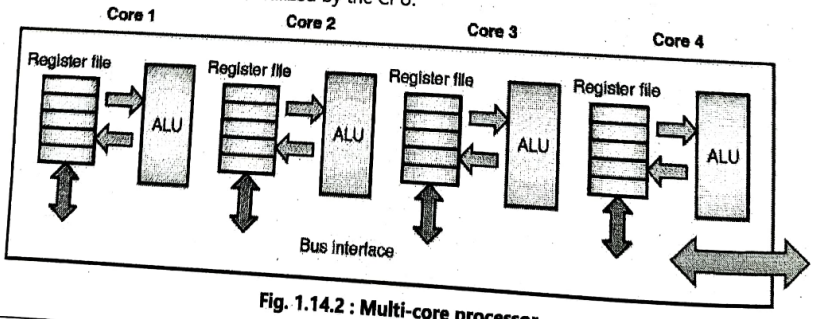


Fig. 1.14.2 : Multi-core processor

- Multicore processor implements multiprocessing.
- Implements separate execution pipeline for each and every core.
- Multicore processors come in the MIMD category.

- Advantage: Improvement in performance by using multiple cores to run tasks.
- OS perceives each core as a separate processor, and it maps threads to different cores.