Nadar Saraswathi College of Engineering & Technology

Engineering Excellence for Empowerment

NSCET E-LEARNING PRESENTATION

LISTEN … LEARN… LEAD…

Department of Electronics and Communication Engineering

III YEAR / V SEMESTER

EC8552 – Computer Architecture and Organization

P.MAHALAKSHMI,M.E,MISTE

ASSISTANT PROFESSOR

Nadar Saraswathi College of Engineering & Technology,

Vadapudupatti, Annanji (po), Theni – 625531.

UNIT V

Advanced Computer Architecture

Parallel Processing Architectures

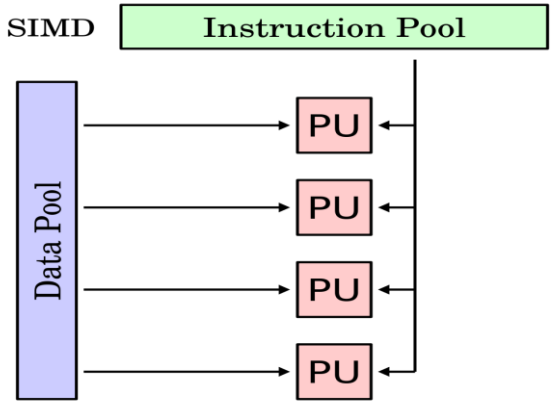
PARALLEL PROCESSING ARCHITECTURES

- ✓ Parallel computing architectures breaks the job into discrete parts that can be executed concurrently. Each part is further broken down to a series of instructions.
- ✓ Instructions from each part execute simultaneously on different CPUs. Parallel systems deal with the 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.
- ✓ Parallel systems are more difficult to program than computers with a single processor because the architecture of parallel computers varies accordingly and the processes of multiple CPUs must be coordinated and synchronized. The crux of parallel processing are the CPUs.
- ✓ 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.
- ✓ Parallelism in computer architecture is explained used Flynns taxonomy. This classification is based on the number of instruction and data streams used in the architecture. The machine structure is explained using streams which are sequence of items. The four categories in Flynns taxonomy based on the number of instruction streams and data streams are the following:

- ✓ (SISD) single instruction, single data
- ✓ (MISD) multiple instruction, single data
- ✓ (SIMD) single instruction, multiple data
- ✓ (MIMD) multiple instruction, multiple data

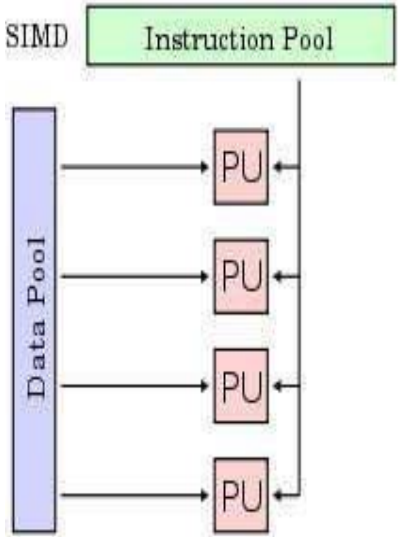
SISD (Single Instruction, Single Data stream)

- ✓ 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.



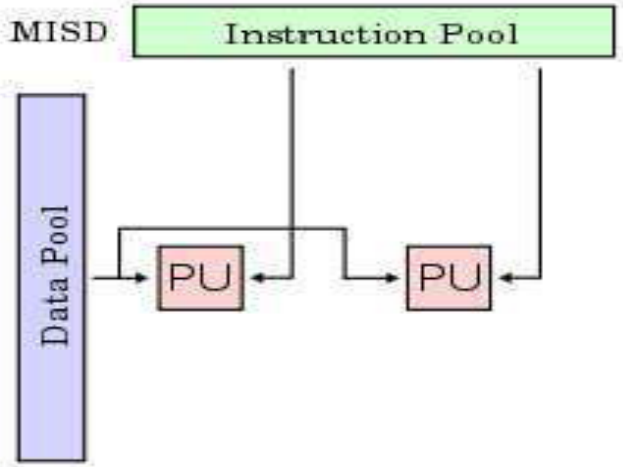
SIMD (Single Instruction, Multiple Data streams)

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.



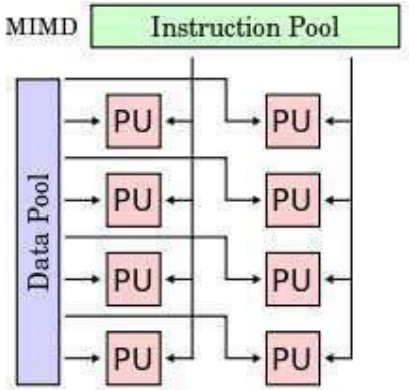
Multiple Instruction, Single Data stream

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



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, and communication switches etc.



Topic

Hardware Multithreading

Introduction

Multithreading enables the processing of multiple threads at one time, rather than multiple processes. Since threads are smaller, more basic instructions than processes, multithreading may occur within processes. Threads are instruction stream with state (registers and memory). The register state is also called thread context. Threads could be part of the same process or from different programs. Threads in the same program share the same address space and hence consume fewer resources.

The terms multithreading, multiprocessing and multitasking are used interchangeably. But each has its unique meaning:

Multitasking: It is the process of executing multiple tasks simultaneously. In multitasking, when a new thread needs to be executed, old thread s context in hardware written back to memory and new thread s context loaded.

Multiprocessing: It is using two or more CPUs within a single computer system.

Multithreading: It is executing several parts of a program in parallel by dividing the specific operations within a single application into individual threads.

Granularity: The threads are categorized based on the amount of work done by the thread. This is known as granularity. When the hardware executes from the hardware texts determines the granularity of multithreading.

The following are the objectives of hardware multithreading:

 To tolerate latency of memory operations, dependent instructions, branch resolution

by utilizing processing resources more efficiently. When one thread encounters a longlatency

operation, the processor can execute a useful operation from another thread.

 To improve system throughput By exploiting thread-level parallelism by improving

superscalar processor utilization

 To reduce context switch penalty

Advantages of hardware multithreading:

 Latency tolerance

 Better hardware utilization

 Reduced context switch penalty

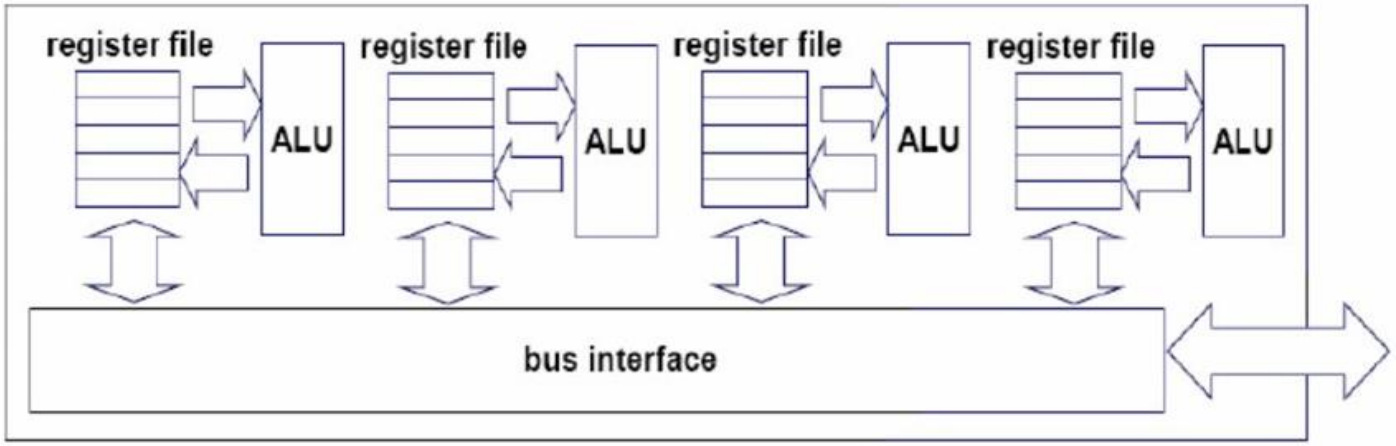
Topic

Multi-Core Architecture and Shared Memory Multiuprocessors

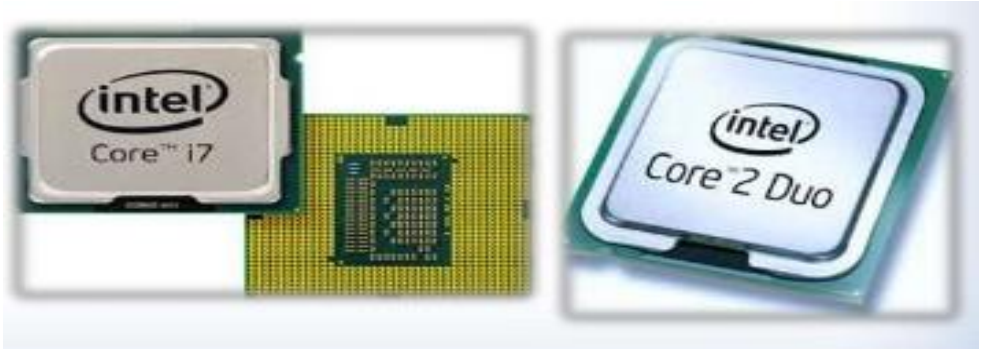
Introduction

- ✓ Latest processors which became available in the market after 2005.
- ✓ Contains two or more cores to process instructions at the same time in parallel.
- ✓ The multiple cores are embedded in the same die.
- ✓ The multicore processor may looks like a single processor but actually it contains two (dual-core), three (tricore),four (quad-core), six (hexa-core), eight (octa-core) or ten (deca-core) cores.
- ✓ Some processor even have 22 or 32 cores.
- ✓ Due to power and temperature constraint, the multicore processors are only practical solution for increasing the speed of future computers.

Architecture Diagram



Examples - Intel Core i7 and Intel Core 2 Duo



Advantage

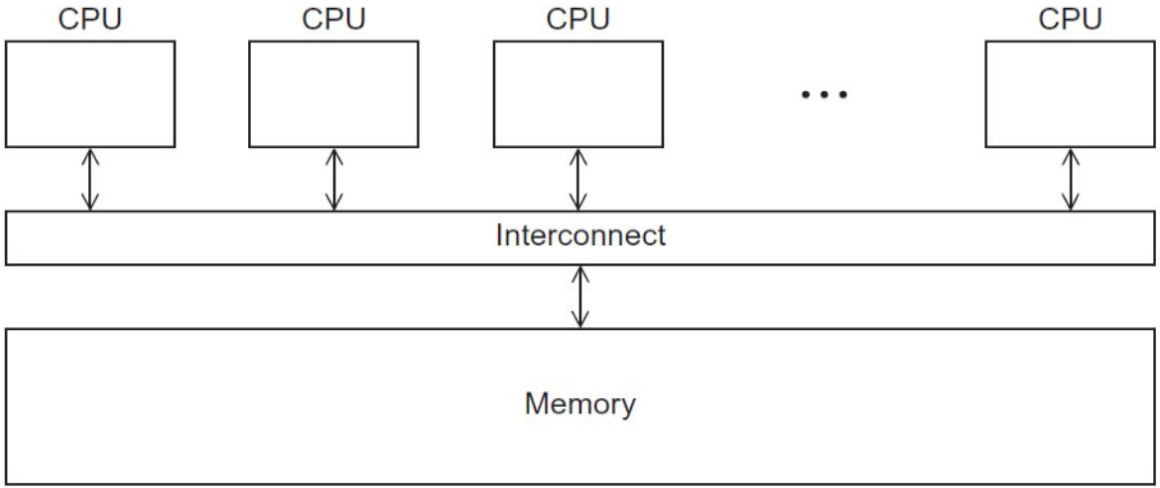
- ✓ High Performance ! Better use of clock rate
- ✓ Power Efficient
- ✓ Sufficient for smaller applications
- ✓ Simultaneous Execution
- ✓ Lesser Heat Generation

Disadvantage

- ✓ Chip Design
- ✓ Additional Resources / Efficiency Tradeoff
- ✓ Sharing the same system bus and memory bandwidth limits the real-world performance advantage.
- ✓ Mult-icore requires parallel programming to fulfill Moore's Law

i) Shared – Memory MIMD system

- ✓ Collection of autonomous processors is connected to a memory system via an interconnection network
- ✓ Each processor can access each memory location.
- ✓ Processors communicate implicitly by accessing shared data structures
- ✓ Most widely available shared memory systems use one or more multi-core processors. Multi-core processor has multiple CPUs or cores on a single chip



iii) Make the common case fast

- ✓ Making the common case fast will tend to enhance performance better than optimizing the rare case.
- ✓ The common case is often simpler than the rare case and it is often easier to enhance
- ✓ Common case fast is only possible with careful experimentation and measurement.

iv) Performance via parallelism

Computer architects have offered designs that get more performance by performing operations in parallel.

v) Performance via pipelining

- ✓ Pipelining is an implementation technique in which multiple instructions are overlapped in execution. Pipelining improves performance by increasing instruction throughput.
- ✓ For example, before fire engines, a human chain can carry a water source to fire much more quickly than individuals with buckets running back and forth.

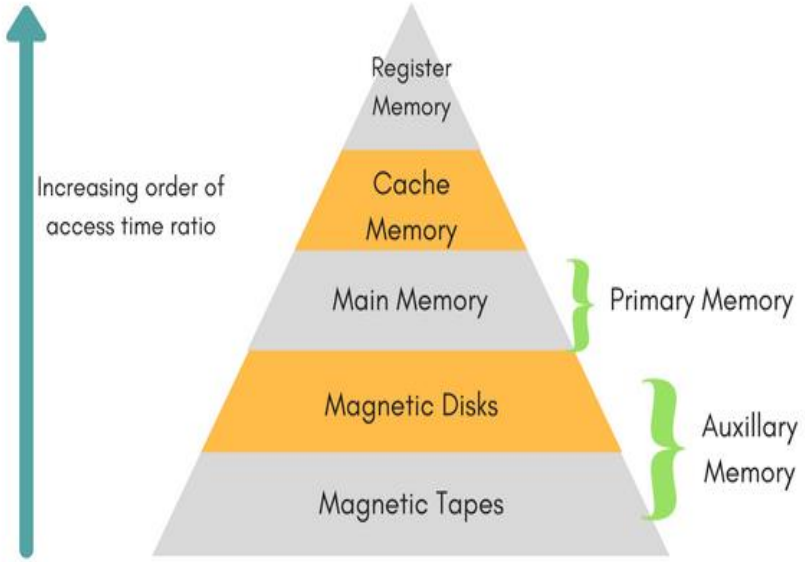
vi) Performance via prediction

- ✓ The next great idea is prediction.
- ✓ In some cases it can be faster on average to guess and start working rather than wait until you know for sure.
- ✓ This mechanism to recover from a misprediction is not too expensive and the prediction is relatively accurate.

vii) Hierarchy of memories

- ✓ Programmers want memory to be fast, large, and cheap.
- ✓ Memory speed often shapes performance
- ✓ Capacity limits the size of problems that can be solved.
- ✓ The cost of memory today is often the majority of computer cost.

- ✓ Architects have found that they can address these conflicting demands with a hierarchy of memories the fastest, smallest, and most expensive memory per bit is at the top of the hierarchy the slowest, largest, and cheapest per bit is at the bottom.
- ✓ Caches give the programmer the illusion that main memory is nearly as fast as the top of the hierarchy and nearly as big and cheap as the bottom of the hierarchy.



viii) Dependability via redundancy

- ✓ Computers not only need to be fast; they need to be dependable.
- ✓ Since any physical device can fail, systems can made dependable by including redundant components.
- ✓ These components can take over when a failure occurs and to help detect failures.

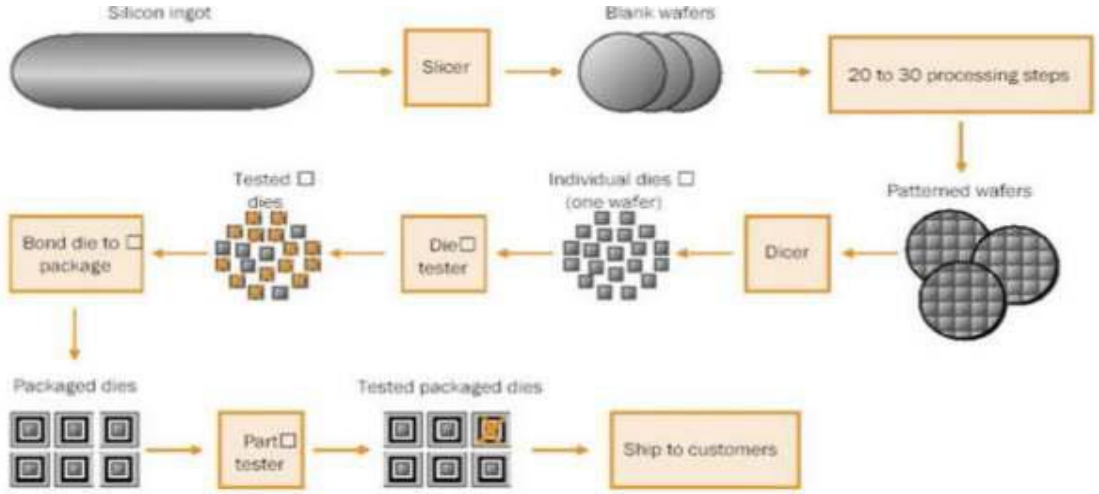
Technologies

- ✓ Up until the early 1970's computers used magnetic core memory, which was slow, cumbersome, and expensive and thus appeared in limited quantities. The situation improved with the introduction of transistor-based dynamic random-access memory (DRAM, invented at IBM in 1966) and static random-access memory (SRAM).
- ✓ A **transistor** is simply an on/off switch controlled by electricity. The **integrated circuit** (IC) combined dozens to hundreds of transistors into a simple chip. **Very large-scale integrated** (VLSI) circuit is a device containing hundreds of thousands to millions of transistors.

Manufacturing of IC:

- ✓ Integrated circuits are chips manufactured on silicon wafers.
- ✓ Transistors are placed on wafers through a chemical etching process. Each wafer is cut into chips which are packed individually.

Chip Manufacturing Process



- ✓ After being sliced from the silicon ingot, blank wafers are put through 20 to 40 steps to create patterned wafers.
- ✓ These patterned wafers are then tested with a wafer tester, and a map of the good parts is made. Then, the wafers are diced into dies. The good dies are then bonded into packages and tested one more time before shipping the packaged parts to customers.

Cost of an IC is found from

- ✓ Cost per die= (cost per wafer) / ((dies per wafer)*yield) Yield refers the fraction of dies that pass testing.
- ✓ Dies / wafer= wafer area / die area
- ✓ Yield=1 / (1 + (defects per area * die area)/2)2

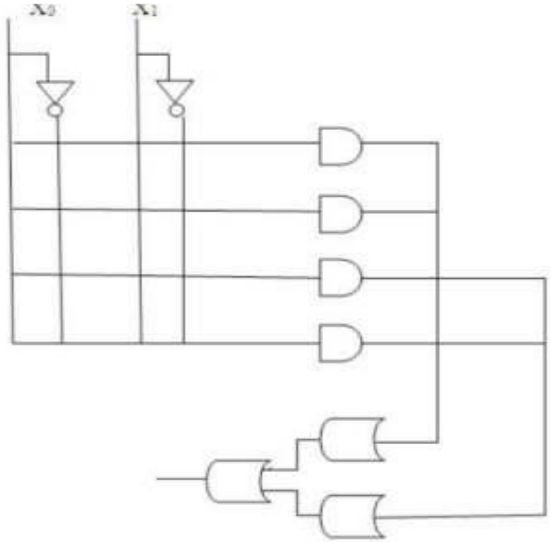
Programmable Logic Device (PLD)

- ✓ A programmable logic device (PLD) is an electronic component used to build reconfigurable digital circuits. Unlike a logic gate, which has a fixed function, a PLD has an undefined function at the time of manufacture. Before the PLD can be used in a circuit it must be programmed, that is, reconfigured.

The major limitations of PLD:

- ✓ Consume space due to large number of switches for programmability
- ✓ Low speed due to the presence of many switches.

Programmable Logic Device



Custom chips

An Application-Specific Integrated Circuit (ASIC) is an integrated circuit (IC) customized for a particular use, rather than intended for general-purpose use. Application-Specific Standard Products (ASSPs) are intermediate between ASICs and industry standard integrated circuits.

Topic

Introduction to Multiprocessor Network Topologies

Introduction

✓ In computing terms, **interconnection networks** provide connections between the different components of the **interconnection** mechanism.

✓ The **network** has processing elements or nodes that are at one end of the **network** and memory elements or nodes at the other end.

✓ The interconnect plays a decisive role in the performance of both distributed and shared memory systems .

✓ The network is composed of links and switches, which helps to send the information from the source node to the destination node. A network is specified by its topology, routing algorithm, switching strategy, and flow control mechanism.

✓ In many proposed or existing parallel processing architectures, an interconnection network is used to realize transportation of data between processors or between processors and memory modules.

Two categories:

i) Shared memory interconnects

ii) Distributed memory interconnects

Shared Memory Interconnects

Currently the two most widely used interconnects on shared-memory systems are busses and crossbars.

Bus interconnect

✓ A collection of parallel communication wires together with some hardware that controls access to the bus.

✓ Communication wires are shared by the devices that are connected to it.

✓ As the number of devices connected to the bus increases, contention for use of the bus increases, and performance decreases.

Switched interconnect(crossbar)

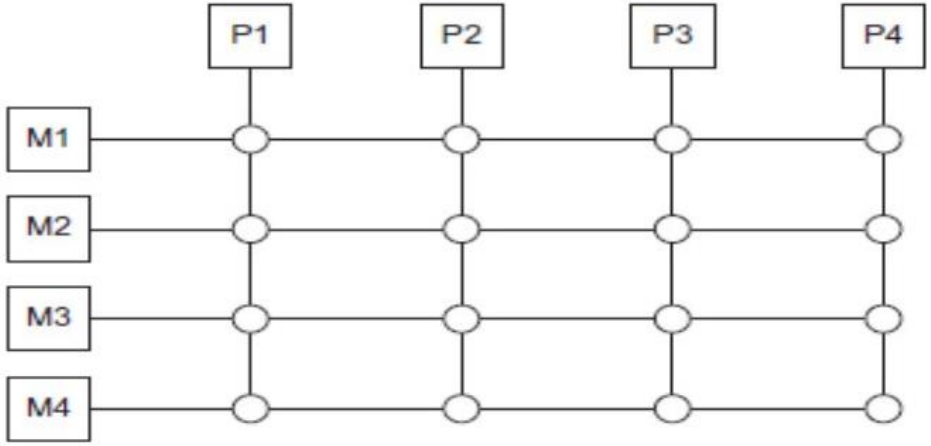
✓Uses switches to control the routing of data among the connected devices.

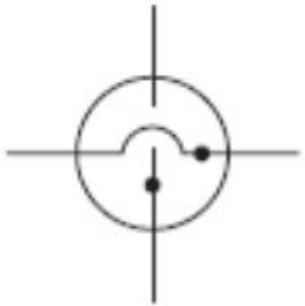
Crossbar

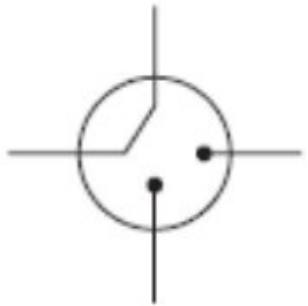
✓Allows simultaneous communication among different devices.

✓Faster than buses.

✓But the cost of the switches and links is relatively high.





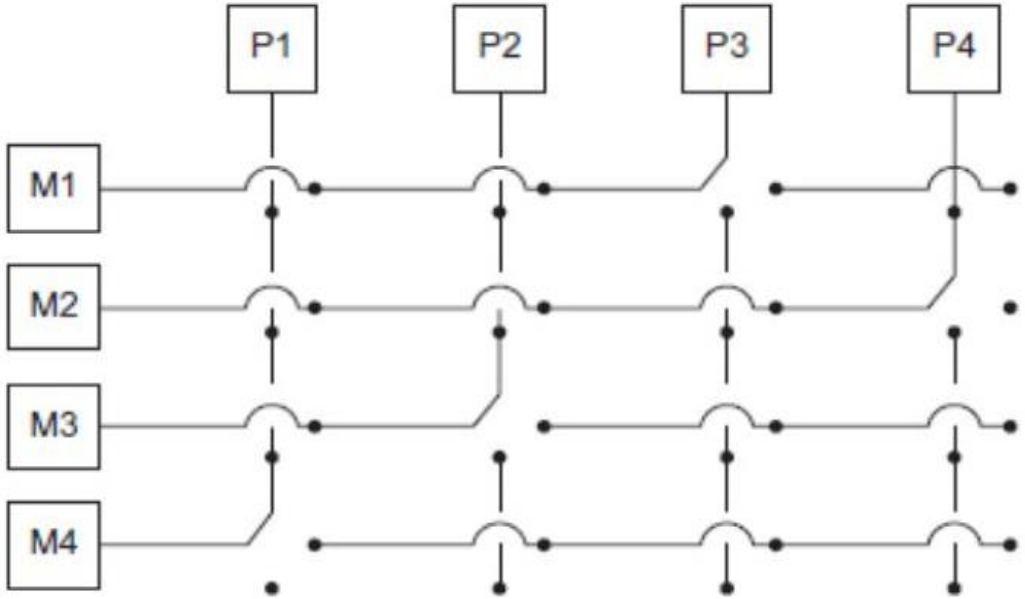


✓The lines are bidirectional communication links

✓ The squares are cores or memory modules

✓ The circles are switches.

✓ The individual switches can assume one of the two configurations shown in Figure (b).



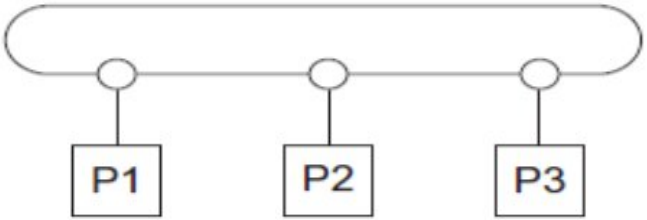
•Figure shows that the configuration of the switches if P_1 writes to M_4, P_2 reads from M_3, P_3 reads from M_1 and P_3 writes to M_2.

Distributed Memory Interconnects

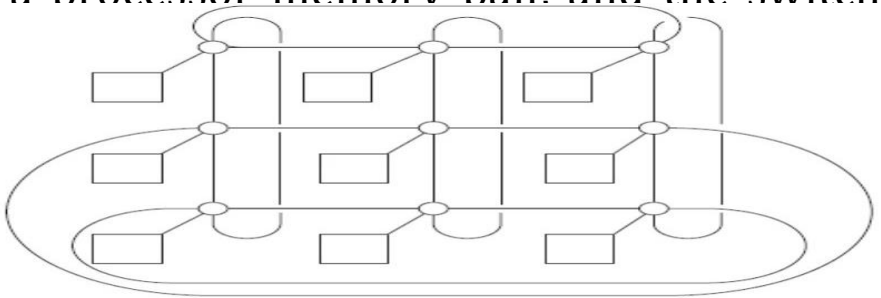
•Distributed Memory Interconnects are often divided into two groups

- i) Direct Interconnects ii) Indirect Interconnects
- **i) Direct Interconnects**

✓ Each switch is directly connected to a processor memory pair, and the switches are



(a) A Ring



(b) A Toroidal Mesh

✓ The circles are switches, the squares are processors, and the lines are bidirectional links.

✓ A ring is superior to a simple bus since it allows multiple simultaneous communications.

✓ The toroidal mesh will be more expensive than the ring, because the switches are more complex.