Nadar Saraswathi College of Engineering & Technology

# NSCET E-LEARNING PRESENTATION

**LISTEN … LEARN… LEAD…**

# Computer Science Engineering

## II YEAR / IV SEMESTER

## CS8491-Computer Architecture

**B.Sri Chithra Devi,M.E**
**Assistant Professor**
**Nadar Saraswathi College of Engineering & Technology,**
**Vadapudupatti, Annanji (po), Theni – 625531.**

## UNIT-IV

# PARALLEL PROCESSING CHALLENGES

# PARALLEL PROCESSING CHALLENGES

- Parallel processing challenges.
- Flynn‘s classification.
- SISD, MIMD, SIMD, SPMD, and Vector Architectures .
- Hardware multithreading.
- Multi-core processors and other Shared Memory Multiprocessors.
- Introduction to Graphics Processing Units.
- Clusters, Warehouse Scale Computers.
- other Message-Passing Multiprocessors.

# Lecture1-Parallel Processing Challenges

- Instruction level dependencies and hazards during ILP
- If two instructions are not dependent, then they can execute simultaneously, assuming sufficient resources that is no structural hazards. if one instruction depends on another, they must execute in order though they may still partially overlap.

**Data dependencies**

- Data dependence means that one instruction is dependent on another if there exists a chain of dependencies between them. .

# Lecture1-Parallel Processing Challenges

- Compilers can be of great help in detecting and scheduling around these sorts of hazards; hardware can only resolve these dependencies with severe limitations.

**Name Dependencies**

- A name dependency occurs when two instructions use the same register or memory location, called a name, but there is no flow of data between them.
- Anti-dependence occurs when j writes a register/memory that i reads.

# Lecture1-Parallel Processing Challenges

- Output dependence occurs when i and j write to the same register/memory location
- The name used in the instruction is changed so that they do not conflict. This technique is known as register renaming (uses temp registers).

# Lecture1-Parallel Processing Challenges

**Data Hazards**

- Data dependency between instructions and they are close enough to cause the pipeline to stall three types of data hazards.
- read after write (RAW)—j tries to read a source before i writes it—this is the most common type and is a true data dependence.

example sequence logic 1. i=i+1; 2. j=i+1

- write after write (WAW)—j tries to write an operand before it is written by i—this corresponds to the output dependence.

# Lecture1-Parallel Processing Challenges

example sequence logic 1. i=i+1; 2. print i; 3. i=j+1

- write after read (WAR)—j tries to write a destination before i has read it—this corresponds to an anti-dependency

example sequence logic 1.read i ; 2. i=j+1

**Control Dependencies**

- A control dependency determines the order of an instruction i with respect to a branch,

```
if (p1)
    S1
if (p2)
```

# Lecture1-Parallel Processing Challenges

- S1 is control dependent on p1 and S2 is control dependent on p2 speculatively executed instructions do not affect the program state until branch result is determined.
- This implies that the instructions executed speculatively must not raise an exception or otherwise cause side effects.

**Implementation of ILP and overcoming hazards or dependencies**.

- To implement ILP, 3 methods are used to avoid any delay during ILP
- score boarding.
- Tomasulo's solution for dynamic scheduling.
- Branch prediction.

# Lecture1-Parallel Processing Challenges

- Parallel processing is the simultaneous use of more than one CPU to execute a program or multiple computational threads.
- Ideally, parallel processing makes programs run faster because there are more engines(CPUs or cores) running it.
- A parallel computer (or multiple processor system) is a collection of communicating processing elements (processors) that cooperate to solve large computational problems fast by dividing such problems into parallel tasks, exploiting Thread-Level Parallelism (TLP).

# Lecture1-Parallel Processing Challenges

**Advantages:**

- Faster execution time, so higher throughput.

**Disadvantages:**

- More hardware required, also more power requirements.
- Not good for low power and mobile devices.

**Challenges in parallel processing**

- Connecting your CPUs
- Dynamic vs Static—connections can change from one communication to next
- Blocking vs Nonblocking—can simultaneous connections be present.
- Connections can be complete, linear, star, grid, tree, hypercube, etc.

# Lecture1-Parallel Processing Challenges

**Bus-based routing**

- Crossbar switching—impractical for all but the most expensive super-computers
- 2X2 switch—can route inputs to different destinations
- Dealing with memory
- Various options:
    i)Global Shared Memory
    ii)Distributed Shared Memory
    iii)Global shared memory with separate cache for processors

# Lecture1-Parallel Processing Challenges

**Solutions:**

- UMA/NUMA machines
- Snoopy cache controllers
- Write-through protocols Global shared memory with separate cache for processors
- Potential Hazards:
- Individual CPU caches or memories can become out of synch with eachother. “Cache Coherence”

# Lecture 2-Flynn’s Classification

- Flynn's taxonomy is a classification of computer architectures, proposed by Michael J.Flynn in 1966. The four classifications defined by Flynn are based upon the number of concurrent instruction (or control) and data streams available in the architecture: Two types of information flow into a processor:
- The instruction stream is defined as the sequence of instructions performed by the processing unit.
- The data stream is defined as the data traffic exchanged between the memory and the processing unit.

# Lecture 2-Flynn's Classification

Computer architecture can be classified into the following four distinct categories:

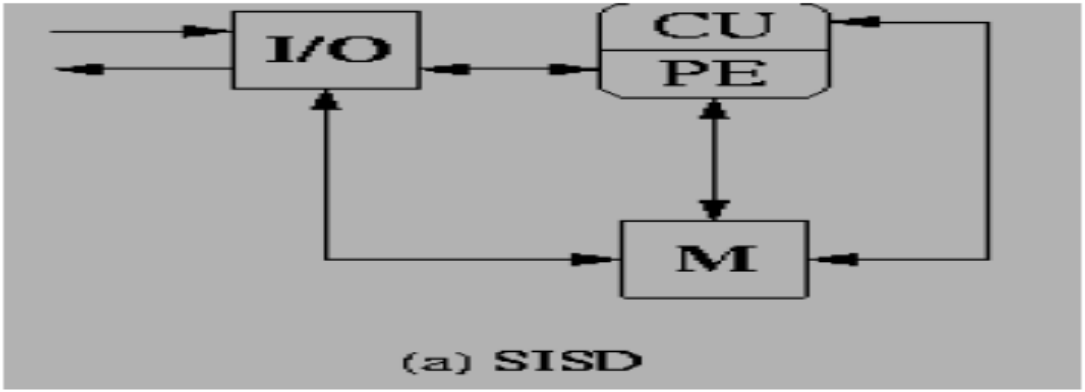
- Single-instruction single-data streams (SISD)
- Single-instruction multiple-data streams (SIMD)
- Multiple-instruction single-data streams (MISD)
- Multiple-instruction multiple-data streams (MIMD).

**Single-instruction single-data streams (SISD)**

- Conventional single-processor von Neumann computers are classified as SISD systems.

# Lecture 2-Flynn's Classification

where CU = control unit, PE= processing element M = memory

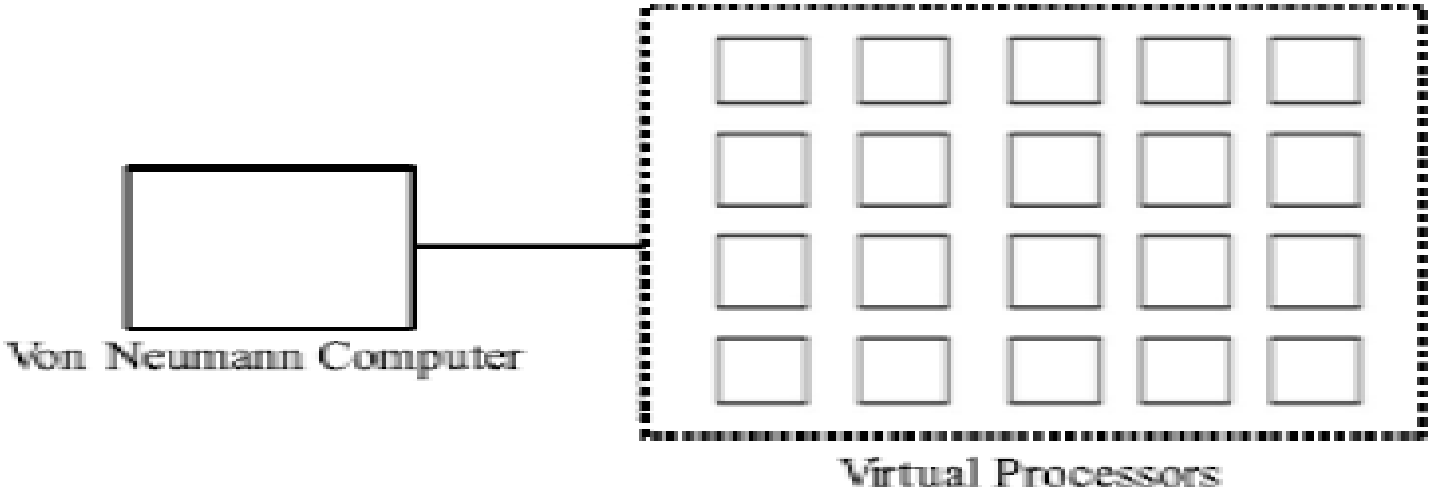


# Lecture 2-Flynn's Classification

**Single-instruction multiple-data streams (SIMD)**

- The SIMD model of parallel computing consists of two parts: a front-end computer of the usual von Neu The processor array is a set of identical synchronized processing elements capable of simultaneously performing the same operation on different data.
- Each processor in the array has a small amount of local memory where the distributed data resides while it is being processed in parallel.

# Lecture 2-Flynn's Classification



# Lecture 2-Flynn's Classification

- The processor array is connected to the memory bus of the front end so that the front end can randomly access the local processor memories as if it were another memory.
- The front end can issue special commands that cause parts of the memory to be operated on simultaneously or cause data to move around in the memory.
- The application program is executed by the front end in the usual serial way, but issues commands to the processor array to carry out SIMD operations in parallel. vonnewmann style, and a processor array.

# Lecture 2-Flynn's Classification

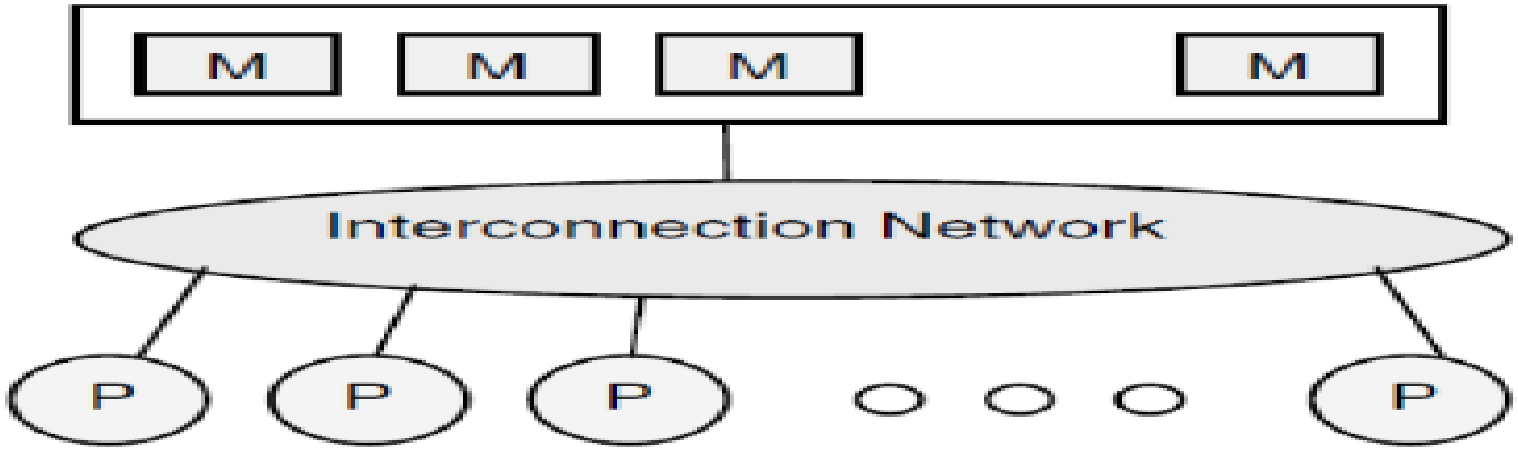
**Multiple-instruction multiple-data streams (MIMD)**

- Multiple-instruction multiple-data streams (MIMD) parallel architectures are made of multiple processors and multiple memory modules connected together via some interconnection network. They fall into two broad categories: shared memory or message passing.
- Processors exchange information through their central shared memory in shared memory systems, and exchange information through their interconnection network in message passing systems.
- A shared memory system typically accomplishes inter-processor coordination through a global memory shared by all processors.

# Lecture 2-Flynn's Classification

- Because access to shared memory is balanced, these systems are also called SMP(symmetric multiprocessor) systems.
- A message passing system (also referred to as distributed memory) typically combines the local memory and processor at each node of the interconnection network.
- There is no global memory, so it is necessary to move data from one local memory to another by means of message passing.
- This is typically done by a Send/Receive pair of commands, which must be written into the application software by a programmer.

# Lecture 2-Flynn's Classification




Shared Memory MIMD Architecture

# Lecture 2-Flynn's Classification

**Multiple-instruction single-data streams (MISD)**

- In the MISD category, the same stream of data flows through a linear array of processors executing different instruction streams.
- In practice, there is no viable MISD machine.

# Lecture 2-Flynn's Classification

**single program, multiple data (SPMD)**

- In computing, SPMD (single program, multiple data) is a technique employed to achieve parallelism; it is a subcategory of MIMD.
- Tasks are split up and run simultaneously on multiple processors with different input in order to obtain results faster.
- SPMD is the most common style of parallel programming.

# Lecture 3-Hardware Multithreading

- Hardware multithreading Increasing utilization of a processor by switching to another thread when one thread is stalled.
- Thread A thread includes the program counter, the register state, and the stack. It is a lightweight process; whereas threads commonly share a single address space, processes don't.
- Process A process includes one or more threads, the address space, and the operating system state.

There are three main approaches to hardware multithreading.

- Fine-grained multithreading
- Coarse-grained multithreading
- Simultaneous multithreading

# Lecture 3-Hardware Multithreading

**Fine-grained multithreading**

- Fine-grained multithreading switches between threads on each instruction, resulting in interleaved execution of multiple threads.
- This interleaving is often done in a round-robin fashion, skipping any threads that are stalled at that clock cycle.
- To make fine-grained multithreading practical, the processor must be able to switch threads on every clock cycle.

**Advantage**

- It can hide the throughput losses that arise from both short and long stalls, since instructions from other threads can be executed when one thread stalls.

# Lecture 3-Hardware Multithreading

**Disadvantage**

- It slows down the execution of the individual threads, since a thread that is ready to execute without stalls will be delayed by instructions from other threads.

**Coarse-grained multithreading**

- Coarse-grained multithreading switches threads only on costly stalls, such as last-level cache misses.

**Advantage:**

- Eliminates the need to have fast thread switching.
- Does not slow down execution of individual threads, since instructions from other threads will only be issued when a thread encounters a costly stall.

# Lecture 3-Hardware Multithreading

**Disadvantages:**

- Does not cover short stalls. This limitation arises from the pipeline start-up.
- Since instructions are issued from a single thread, when a stall occurs, the pipeline must be emptied or frozen.
- The new thread that begins executing after the stall must fill the pipeline before instructions will be able to complete.
- Due to this start-up overhead, coarse-grained multithreading is much more useful for reducing the penalty of high-cost stalls, where pipeline refill is negligible compared to the stall time.

# Lecture 3-Hardware Multithreading

**Simultaneous multithreading**

- Simultaneous multithreading (SMT) is a variation on hardware multithreading.
- Instructions from multiple threads are issued on same cycle. Uses register renaming and dynamic scheduling facility of multi-issue architecture.

**Advantage**

- Maximizes utilization of execution units

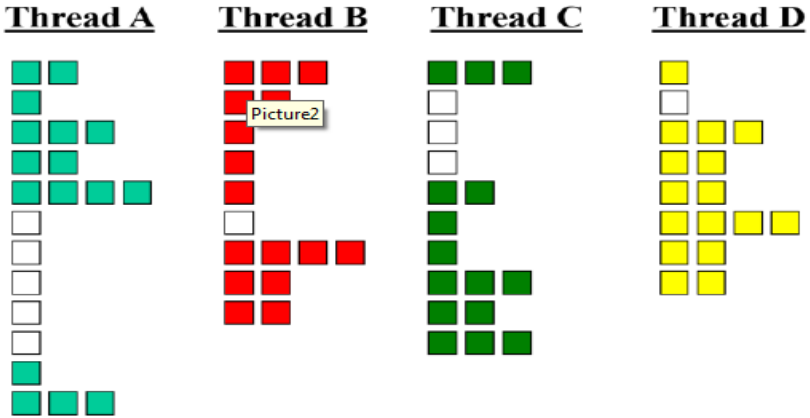
**Disadvantage**

- Needs more hardware support
- Register files, PC's for each thread
- Temporary result registers before commit
- Support to sort out which threads get results from which instructions

# Lecture 3-Hardware Multithreading

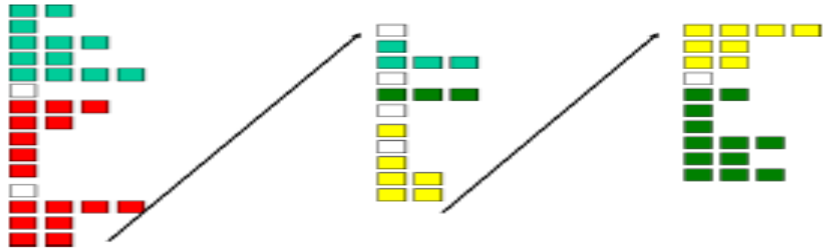
**Example**

**How four threads use the issue slots of a superscalar processor in different approaches.**

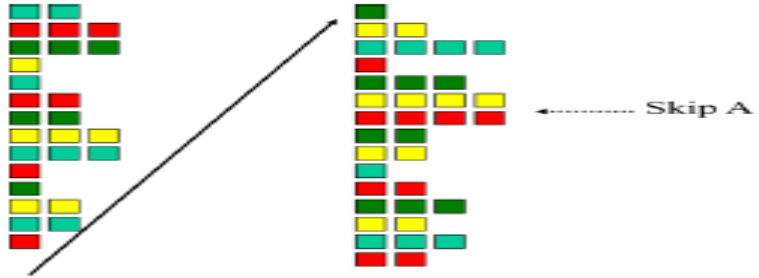


# Lecture 3-Hardware Multithreading

Coarse Multithreading

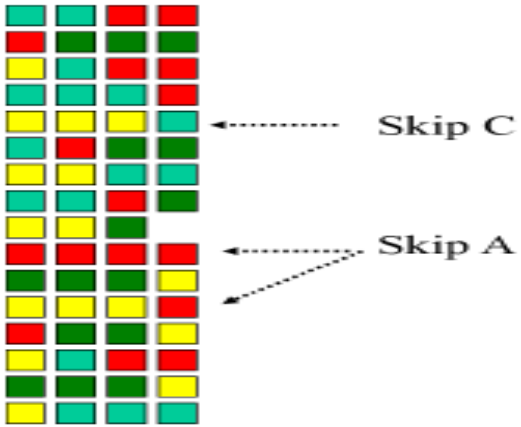


Fine Multithreading



# Lecture 3-Hardware Multithreading

**Simultaneous Multithreading**



# Lecture 4-Multicore processor

- A symmetric multiprocessor (SMP) can be defined as a standalone computer system with the following characteristic:
- There are two or more similar processor of comparable capability.
- processors share the same main memory and I/O facilities and are interconnected by a bus or other internal connection scheme.
- All processors share access to I/O devices, either through the same channels or through different channels.All processors can perform the same functions.
- 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.
- SMP has a potential advantage over uniprocessor architecture

# Lecture 4-Multicore processor

**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.

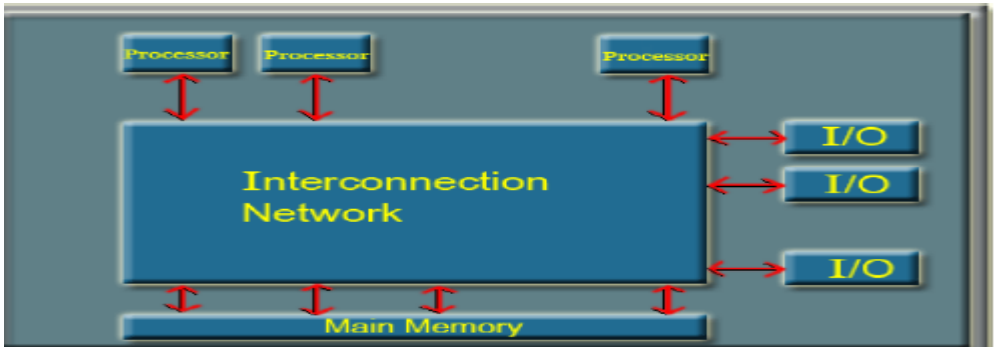
# Lecture 4-Multicore processor

**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.



# Lecture 4-Multicore processor

- 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.

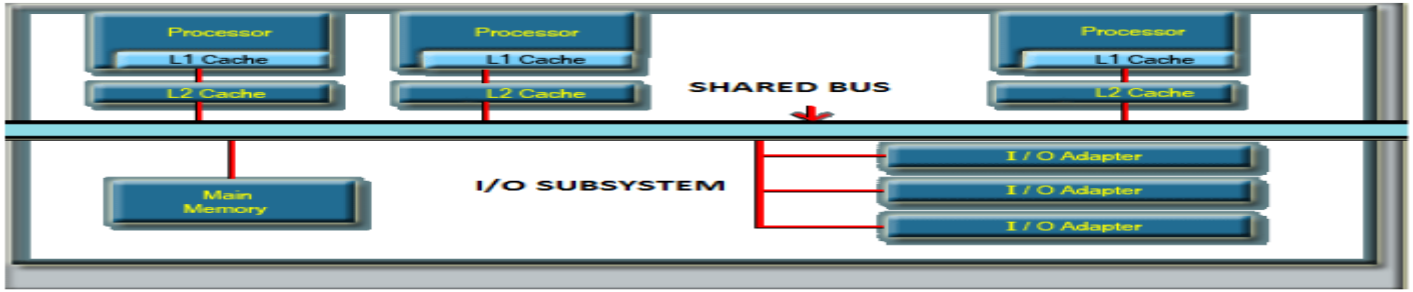
The organization of multiprocessor system can be classified as follows:

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

# Lecture 4-Multicore processor

**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



# Lecture 4-Multicore processor

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 in achieved.

# Lecture 4-Multicore processor

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.

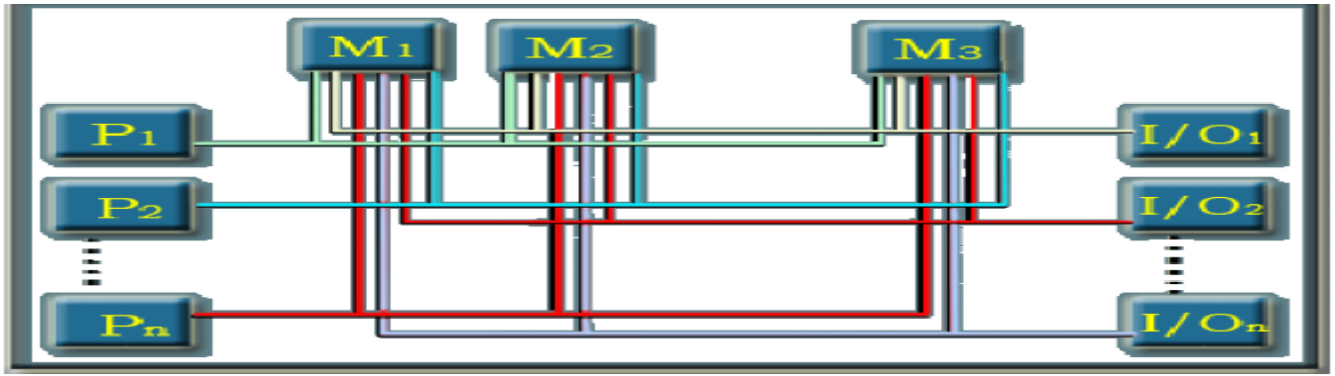
**Reliability:**

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.

# Lecture 4-Multicore processor

**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



# Lecture 4-Multicore processor

**Multiport Memory**

- 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.

# Lecture 4-Multicore processor

**Shared Memory Multiprocessor**

- Shared-memory processors are popular due to their simple and general programming model, which allows simple development of parallel software that supports sharing of code and data.
- Another name for shared memory processors is parallel random access machine (PRAM).
- The shared-memory or shared-address space is used as a means for communication between the processors.
- All the processors in the shared memory architecture can access the same address space of a common memory through an interconnection network.

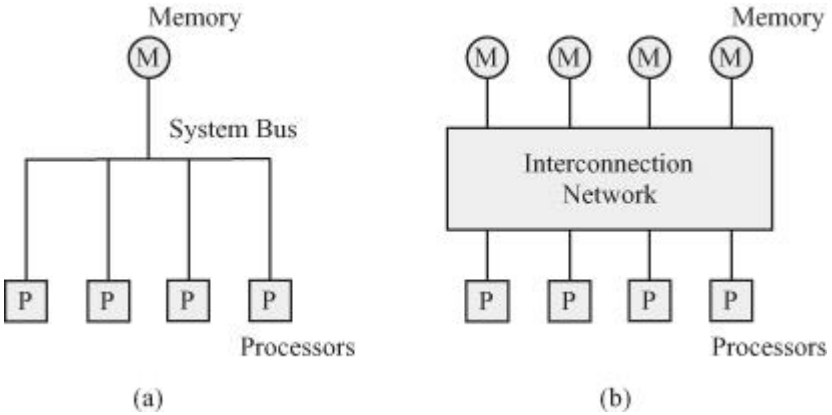
# Lecture 4-Multicore processor

**Shared Memory Multiprocessor**

- Typically, that interconnection network is a bus, but for larger systems, a network replaces the bus to improve performance.
- The performance we are referring to is the amount of processor/memory accesses that can be performed per unit time (throughput) and the time delay between a processor requesting memory access and the time when that request is granted (delay).
- Shared-memory multiprocessor architecture (PRAM). (a) The processors are connected to the shared memory using a single bus. (b) The processors and memory modules are connected using an interconnection network.

# Lecture 4-Multicore processor

## Shared Memory Multiprocessor



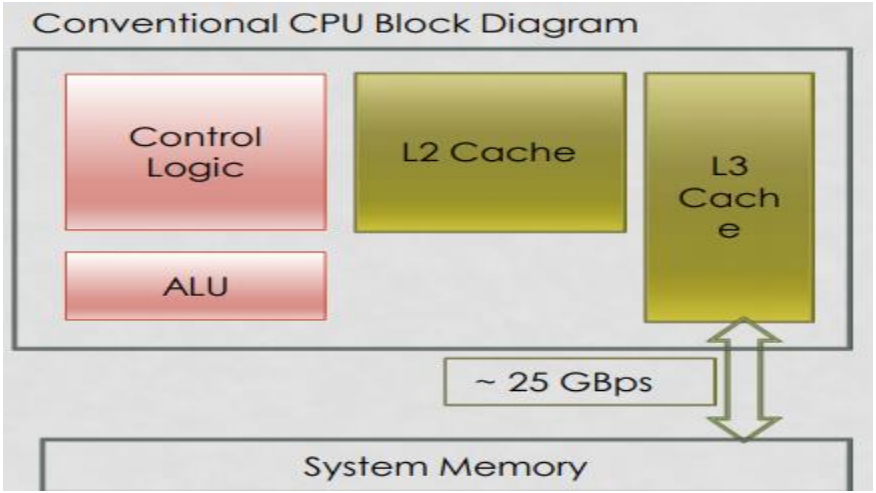
# Lecture 5-Graphics Processing Unit

**Graphics Processing Unit (GPU)**

- A specialized circuit designed to rapidly manipulate and alter memory
- Accelerate the building of images in a frame buffer intended for output to a display
- GPU -> General Purpose Graphics Processing Unit (GPGPU)
- A general purpose graphics processing unit as a modified form of stream processor
- Transforms the computational power of a modern graphics accelerator's shader pipeline into general-purpose computing power

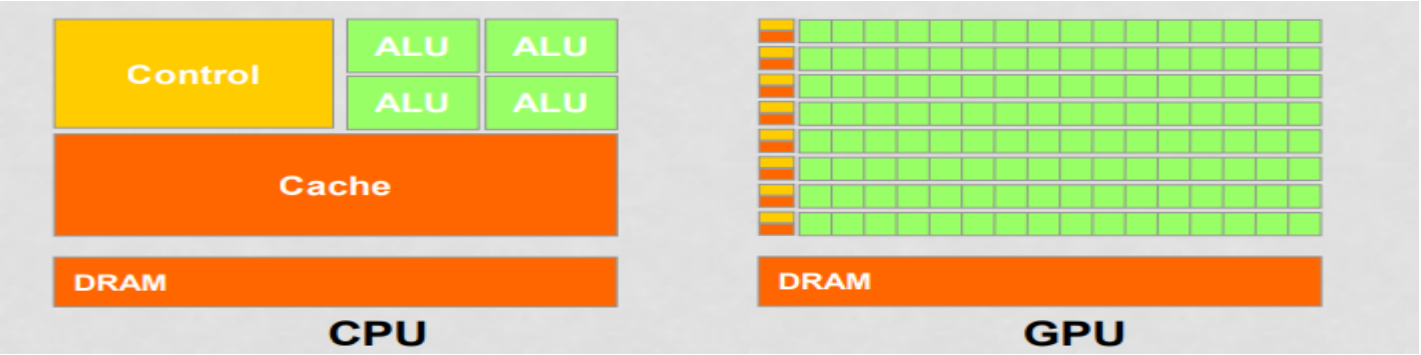
# Lecture 5-Graphics Processing Unit

## Conventional CPU architecture



# Lecture 5-Graphics Processing Unit

- The GPU is specialized for compute-intensive, highly data parallel computation (exactly what graphics rendering is about) .So, more transistors can be devoted to data processing rather than data caching and flow control.
- The fast-growing video game industry exerts strong economic pressure that forces constant innovation



# Lecture 5-Graphics Processing Unit

**GPU Vendors**

- AMD
- NVIDIA
- INTEL

**NVIDIA Architecture**

OpenCL

- Standards specification
- Compute Unified Device Architecture (CUDA) Framework
- A general purpose parallel computing architecture
- A new parallel programming model and instruction set architecture

# Lecture 5-Graphics Processing Unit

- Leverages the parallel compute engine in NVIDIA GPUs
- Software environment that allows developers to use C as a high-level programming language.
- A new parallel programming model and instruction set architecture
- Leverages the parallel compute engine in NVIDIA GPUs
- Software environment that allows developers to use C as a high-level programming language

# Lecture 5-Graphics Processing Unit

**AMD PLATFORM**

- OPENCL

Individual work-items execute on a single processing element

- Processing element refers to a single VLIW core
- Multiple work-groups execute on a compute unit
- A compute unit refers to a SIMD Engine

# Lecture 5-Graphics Processing Unit

**Intel GMA**

- Series of Intel integrated graphics processors built into various motherboard chip sets.
- These integrated graphics products allow a computer to be built without a separate graphics card, which can reduce cost, power consumption and noise. They rely on the computer's main memory for storage Performance penalty
- CPU and GPU access memory over the same bus

# Lecture 5-Graphics Processing Unit

- In early 2007, 90% of all PCs sold had integrated graphics.
- Sandy Bridge / Ivy Bridge
- The built-in GPU has up to 12 execution units in Sandy Bridge
- Ivy Bridge will have next Generation Intel HD Graphics with DirectX 11, OpenGL 3.1, and OpenCL 1.1

# Lecture 6-Cluster and warehouse scale computer

- A cluster is a collection of desktop computers or servers connected together by a local area network to act as a single larger computer.
- A warehouse-scale computer (WSC) is a cluster comprised of tens of thousands of servers.
- The cost may be on the order of $150M for the building, electrical and cooling infrastructure, the servers, and the networking equipment that houses 50,000 to 100,000 servers.

# Lecture 6-Cluster and warehouse scale computers

- A WSC can be used to provide internet services.
- search Google social networking.
- Facebook video sharing .
- YouTube online sales.
- Amazon cloud computing services.
- Rackspace and many more applications

# Lecture 6-Cluster and warehouse scale computers

**Important design factor for WSC:**

- WSC goals and requirements in common with servers.
- cost-performance
- work done per dollar energy effciency
- work done per joule dependability via redundancy network I/O
- interactive and batch processing workloads WSC aspects that are distinct from servers.

# Lecture 6-Cluster and warehouse scale computers

**Storage for WSC**

- A WSC uses local disks inside the servers as opposed to network attached storage (NAS).
- The Google  system (GFS) uses local disks and maintains at least three replicas to improve dependability by covering not only disk failures, but also power failures to a rack or a cluster of racks by placing the replicas on different clusters.
- A read is serviced by one of the three replicas, but a write has to go to all three replicas. Google uses a relaxed consistency model in that all three replicas have to eventually match, but not all at the same time.

# Lecture 6-Cluster and warehouse scale computers

**WSC Networking**

- A WSC uses a hierarchy of networks for interconnection.
- The standard rack holds 48 servers connected by a 48-port Ethernet switch. A rack switch has 2 to 8 uplinks to a higher switch.
- So the bandwidth leaving the rack is 6 (48/8) to 24 (48/2) times less than the bandwidth within a rack.
- There are array switches that are more expensive to allow higher connectivity. There may also be Layer 3 routers to connect the arrays together and to the Internet.
- The goal of the software is to maximize locality of communication relative to the rack.

# Lecture 6-Cluster and warehouse scale computers

**WSC Location**

- proximity to Internet backbone optical fibers proximity to users of service to reduce Internet access latency electricity availability and cost property tax rate low risk from environmental disasters stability of country low temperature to decrease cooling cost

# Lecture 6-Cluster and warehouse scale computers

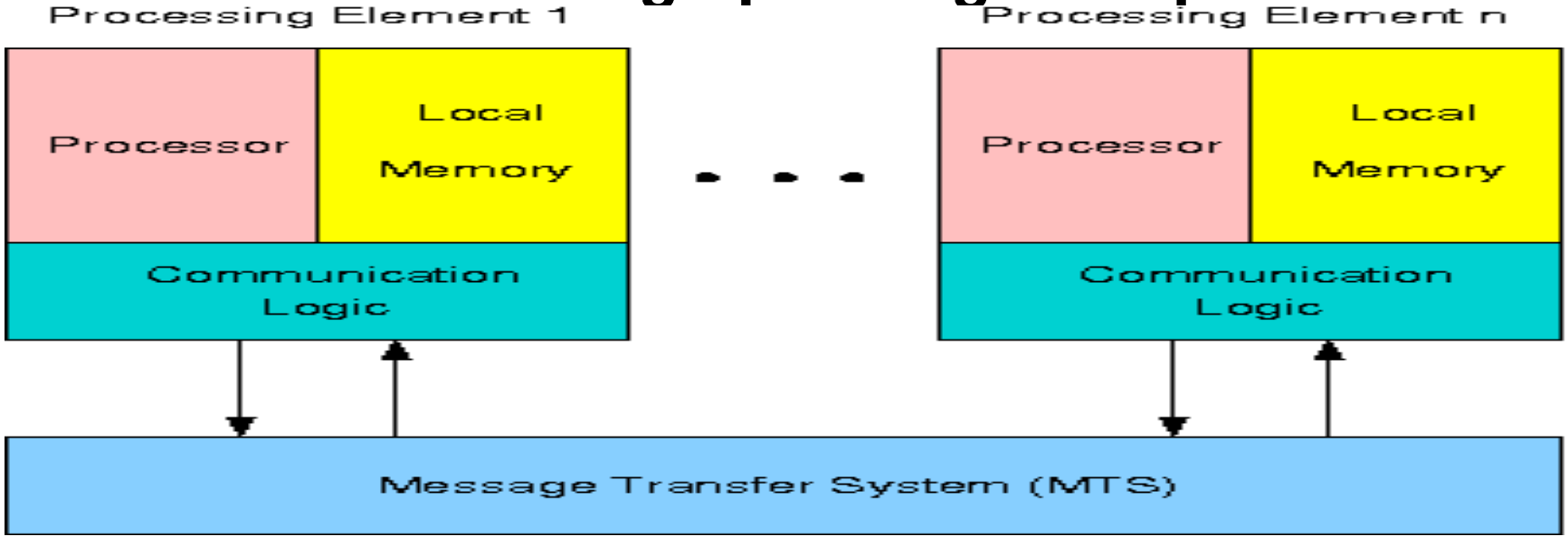
**Improvements in operational techniques**

- WSCs have led to innovations in system software to provide high reliability. failover - Automatically restarting an application that fails without requiring administrative intervention.
- firewall - Examines each network packet to determine whether or not it should be forwarded to its destination.
- virtual machine - A software layer that executes applications like a physical machine. Protection against denial-of-service attacks

# Lecture 7-Message passing multiprocessor

- Message-passing MIMD systems were developed as a means of overcoming two fundamental problems associated with systems in which a number of parallel processors are connected by means of a common shared store.
- The general structure of a message-passing multiprocessor system is depicted in the figure, from which it can be seen that there are two primary components; the processing elements (PEs) and the message transfer system (MTS).

# Lecture 7-Message passing multiprocessor



# Lecture 7-Message passing multiprocessor

- with both naming strategies mentioned above it is possible for communication events to occur between processes (effectively virtual processors) which reside on the *same* or different physical processors, and the MTS protocol must deal with both of these situations.